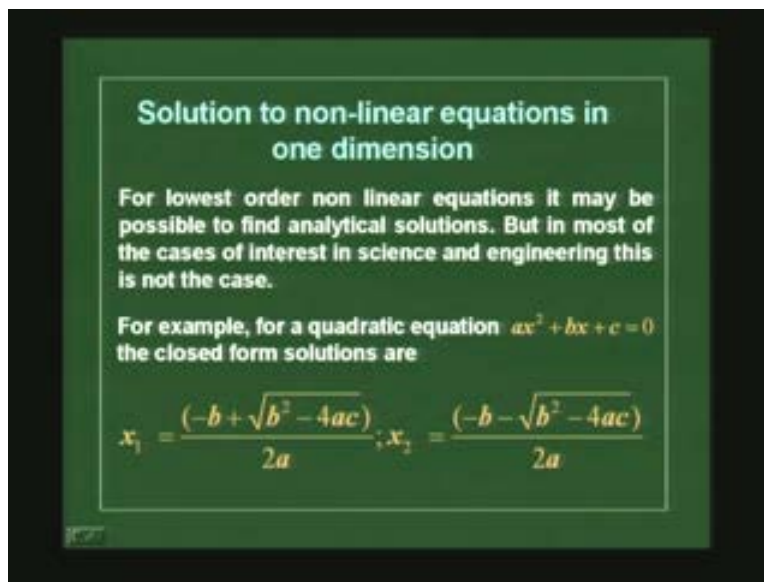


Numerical Methods and Programming
P. B. Sunil Kumar
Department of Physics
Indian Institute of Technology, Madras
Lecture - 22
Solving Non-Linear Equations
Newton Raphson Method

We were discussing methods to solve non-linear equations, so that is solution is to non-linear equations in one dimensions that is what we were discussing. So just briefly go over the methods which we discussed in the last class and also some of the implementations of these methods in the form of programs. So that is what we will be looking at today. So you remember that such numerical solutions are necessary, because you cannot always get close form solutions to non-linear equations just as in the case of a quadratic equation.

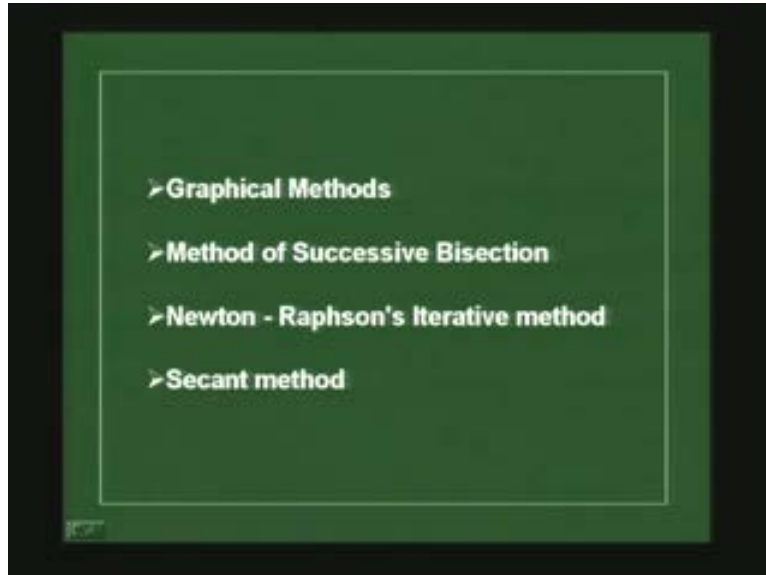
(Refer Slide Time: 01:22)



So numerical methods to solve them are absolutely required and we said that there are 4 different methods to solve or achieve this goal and they are basically graphical methods of successive bisection, Newton Raphson method and iterative method, iterative method Newton Raphson iterative method and secant method

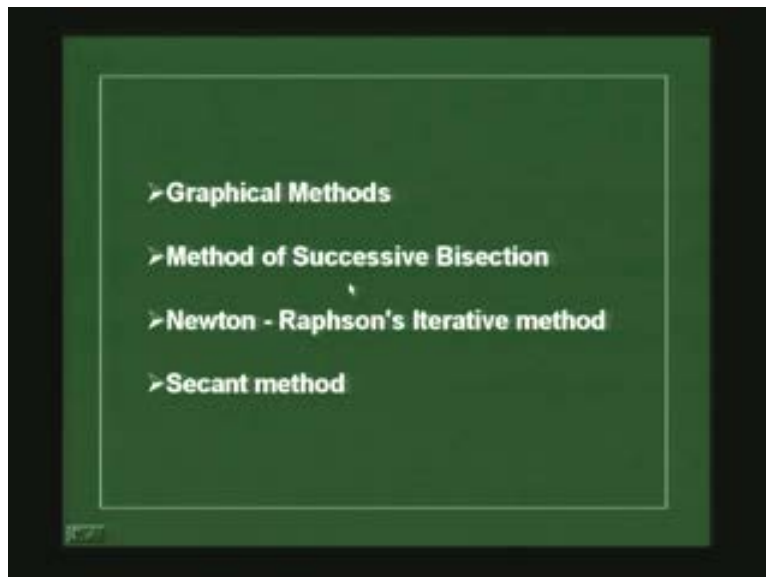
In fact all these methods are iterative except this graphical method and which you could simply plot a function and then look at the 0s, a word they cross the x axis in the case of one dimensional function which you are discussing here this is rather useful to get a rough estimate and that we looked at method of successive bisection and this bisection methods which we looked at basically the fixed points iteration and then we also looked at the mean value problem.

(Refer Slide Time: 02:00)



Okay, so we look at the 2 methods of doing this and that is what the implementation of that we will see in a few minutes.

(Refer Slide Time: 02:12)



So one in the in the bisection method actually the 2,1 is bisection method, so 2 iterative methods in this one is bisection methods, we looked at and which we said that we choose or we make 2 guesses that is x_0 and x_1 as the 0 of the function or the where the function crosses the x axis to be and we will choose x_0 and x_1 such that it is on the either side of the root right and we then take the mean of this as a new iteration point okay and then look at the function value at this point and in the function value at this point is negative

then we replace x_0 by this point and the function value at this point is that is f of x_2 is positive and we will replace x_1 value by this point and that is what we saw because we would chosen x_0 and x_1 such that f of x_0 is less than 0 and f of x_1 is greater than 0, that is if you actually go look at this function of this form.

(Refer Slide Time: 03:28)

Method of Successive Bisection

By now it must be clear to you that the function $f(x)$ has opposite signs on either side of the root. The bisection method exploits this property of the function.

Two points x_0 and x_1 enclose a root if $f(x_0)$ and $f(x_1)$ are of opposite signs. Let us say that $f(x_0) < 0$ and $f(x_1) > 0$. We now bisect the interval (x_0, x_1) and denote the midpoint of so that $x_0 < x_2$ and

$$x_2 = (x_0 + x_1) / 2$$

So let us say we look at the function of this form that is 1 minus x square minus plus log 10 of 1 plus x. Okay we had some function of this form plotted from 0 to 2 and we will plot 0 axis on that that is a now that is a x axis right so that is 0 line.

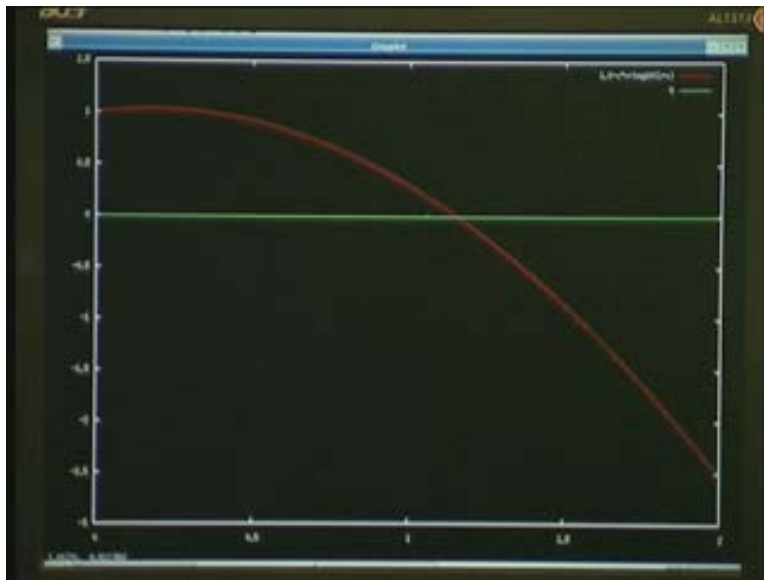
(Refer Slide Time: 04:43)

lot.info/faq/
 sts for help to
 ists.sourceforge
 and mods to
 ists.sourceforge

```
Terminal type set to 'x11'
gnuplot> plot 1.0-x*x+log10(1+x)
gnuplot> plot [0:2]1.0-x*x+log10(1+x)
gnuplot> plot [0:2]1.0-x*x+log10(1+x),0
gnuplot>
```

So we want to get this point where this function crosses this axis right, so that is what we want to get. So basically we want to get this point where this function crosses the x axis, okay and then it turns out that the one way to do that uses bisection method. So we will make a guess of where this is, this crossing point is and that is taken as x_0 and x_1 , so we are chosen such that x_0 is on the side where the function is negative and x_1 is on the side where the function is positive. So we have to choose make our initial guesses on either side of this crossing point that is a drawback of this particular method that we need to have an idea where the function the 0 of the function, the 0 of the function is and then we have um to choose x on either or initial guesses to be on either side on this point. So let us look at an implementation of this code first and then we will go and look at other methods.

(Refer Slide Time: 05:31)



So here is the code which would the midpoint or bisection method, okay so here is the code which does that so what I have done here is to right the code in such a way that the 2 function calls in this code. So that is the midpoint is our method, so midpoint method is been put into a function. So the main program here simply makes takes an initial guesses it asks you to feed in 2 initial guesses and you have to feed in this 2 initial guesses x_0 and x_1 such that f of x_0 is negative and f of x_1 is positive that it is on either side of the desired root.

So there are many roots in this program in this program assumed only one root, there are many roots we have to make many calls depending upon how many roots the function has. So we need to have an idea about how many roots the function has, before you make a call like this a program like this. Okay so, we this scanf function will as you know we have seen it before just reads it x_0 and x_1 , okay from the screen and then it feeds that into this function and call midpoint which is been defined, which returns the solution. So, since this midpoint function returns a solution we need to define the other way the what variable type it returns this and it returns the floating point and hence I have defined here

declared this midpoint to be a float okay and now this midpoint function which finds the 0 using this bisection method or the midpoint method and that function need to know what is the function or of which it is defined the root right.

So that is a user defined function, so that has been passed that is returned in another function called func here and this program need to know what that is so you note this here so I have called, so I am passing this variable here so this function is declared here, okay as external okay this is declared external some this again a float this is a variable type we will see why it is so. So this is a function, so this returns the function value at any point like this func basically has my function which or whose 0, I need to find whose 0, I need to find.

So that is applied in another function okay that is called func and that function is a, that pointer to that function is passed to that midpoint. So this main program itself does not call this func it is called by this function called midpoint. So, that the idea, so it needs to know what that function is and that is been passed here. Okay so now when this midpoint function takes it as function you could give any name here but it takes it as function and then it calls it as function.

So that is idea. So let us look at the structure again, so we have 2 functions here one is midpoint that is the method now this is going to be general structure this is going to be 1 or 2 programs we going to look at here we have a function called now one function which tells you the solution to the order 0 or of your equation and there could be midpoint method or fixed point iteration method or it could be Newton Raphson method or it could be secant method any method which you are going to look at of the different methods will be different functions.

So this main program simply calls one of those functions here, passing the initial guesses okay and a pointer to a function which contains my f, f of x for which I need to find the 0 so this is the passing pointer to that we have seen this in the initial part programming part that how to pass a pointer to a function to another function again that is what it says we are just passing a pointer to that function.

So now here is the midpoint function and again I told you that this is supposed to return a floating point variable. So I type casted it as float and similarly this midpoint is suppose to return that floating point variable which is a_0 . So again I declare it as floating point okay then this is the main thing this is very simple you know that all you need to do is to use this function, all you need to do is to take x_2 as x_1 plus x_0 by 2. So that is our method right, so we have initial guesses x_0 and x_1 which is passed on to this and this is, okay now this pointer to this function the one which gives you the function value. Now, this program here it could give any name it does not matter right and you have to just called this is a pointer which is passed on to this okay.

So here it would give any name that is an idea if you have a package if your are writing for a package find the roots of the equation then you can give any name here the user need not know that what you give your name here we just have to the user has to defined

a some sub routine called function whatever this case func and pass a pointer to that okay so then this function returns a float and it takes an argument which is float. So that is what it is and then what have I derived is some error to my the error estimate here start with initialize with 1 and then I say that you do this loop till this error is some user defined tolerance which I given as “.001” 10 power minus 4.

So this error is defined as if you remember is the initial guess minus the new guess divided by the new guess the absolute value that is the error that we have declared here for example it could be x_0 minus x_2 by x_2 or x_1 minus x_2 by x_2 depending upon after the first iteration we are going to replace x_0 or x_1 by x_2 . So x_0 and x_1 are our initial guesses and we come here and we compute x_2 as the mean the mean midpoint x_0 and x_1 x_0 and x_1 and then I compute the function value now I look in the function call in the function here f of x function x_2 the function x_2 when I called look at the function x_2 , the function is when I say function here it actually calls func because the function is a pointer to the function it called func with an argument float x that is a value x value which you want to evaluate the function and the function is evaluated at that point, 1 minus x square plus log to the base 10, 1 plus x that is our function. The function we just now saw, that is 1 minus square plus log to base 10 of 1 plus x .

So it just computes the function value at that point and returns it that is what it doing. So this function whenever you called function from this program, from this subroutine call midpoint and it actually calling func because we have passed a pointer to a func this function to this program. Okay so it computes that, we determine the new x_2 value midpoint between the x_0 and x_1 , it computes the function values at that point it evaluates it looks it less than 0 if it is less than 0 then the x_0 is replace by x_2 right because f of x_0 is negative if f of x_2 is negative and x_0 is replace by x_2 and so the error would be x_0 by x_2 on the other hand if function f of x_2 is greater than 0 then we will replace x_1 by x_2 because f of x_1 was positive. So we will replace by x_1 by x_2 the new x_1 value will given by x_2 and the error is x_1 minus x_2 by x_2 .

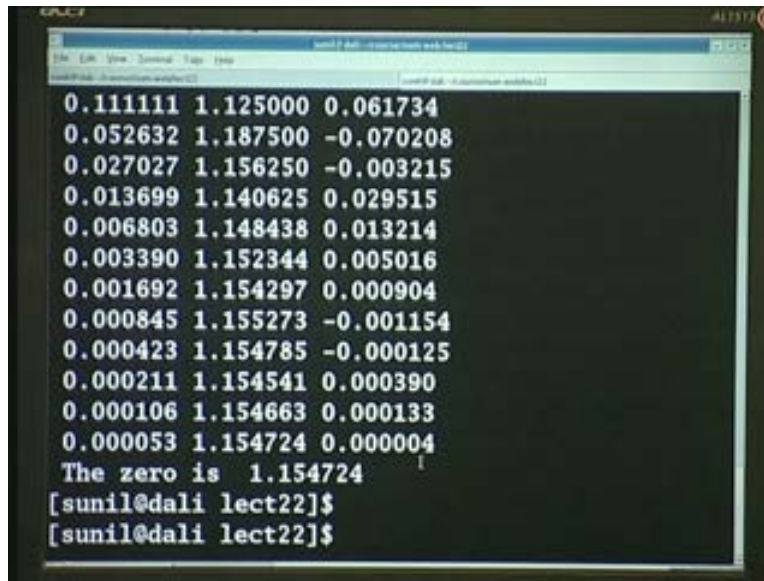
Okay so, at each iteration step okay I print out the error the new x_2 value and the function value at x_2 and then once it once it satisfies the error condition, that error is now less than or equal to “.00001”. Okay it returns that 2 value and comes out of the program and it goes here that is our answer and the 0 and it the main program lets print out the 0 of the function is a function as whatever it get as x here that is the idea. Okay let us see that how it works, so we will just compile this and again compute it i called it midpoint c minus lm because I am using log 10 it needs the math library.

So we will run this program now, so that is we will say that dot slash a dot out okay now program is waiting for 2 inputs let us go back and see, so it is waiting for this function here at this variable x_0 and x_1 and you remember f of x_0 is negative and f of x_1 is positive that is what we need to give. So f of x_0 is negative f of x_1 is positive because we are going to replace whenever the function value is negative, we are going to replace by x_0 by that and function value is positive, we are going to replace by x_1 by that x_0 f of x_0 is negative and if you plot this function we have seen that in earlier cases the 0 is around

between 1 and 2 and also we have to give it as 2 and 1 initial guesses the 2 f of 2 is negative 1 is positive, so that is what the output iteration step.

So we have the initial iteration step is given as initial value 1 and 2 and the initial iteration would be the first step should be “1.5” this was the initial midpoint rule is and the function value there is negative. So we will replace f of x_0 by x_0 by “1.5”. So we had x_0 as 2 that we are going to replace as “1.5” and then we will evaluate the function again in the second step and the function value is still negative now we are going to replace by x_0 by x_2 “1.25” and then iteration the function value is positive. So the function value is positive, so now we are going to replace x_1 that is 1 by “1.125” that is a way the iteration proceeds and you can see that the error which in this case it was x_2 minus x_0 by x_2 in this case x_2 minus x_1 by x_2 is decreasing monotonically.

(Refer Slide Time: 06:36)



```
0.111111 1.125000 0.061734
0.052632 1.187500 -0.070208
0.027027 1.156250 -0.003215
0.013699 1.140625 0.029515
0.006803 1.148438 0.013214
0.003390 1.152344 0.005016
0.001692 1.154297 0.000904
0.000845 1.155273 -0.001154
0.000423 1.154785 -0.000125
0.000211 1.154541 0.000390
0.000106 1.154663 0.000133
0.000053 1.154724 0.000004
The zero is 1.154724
[sunil@dali lect22]$
[sunil@dali lect22]$
```

So the function value itself is also going towards 0 and then, thus we will converge pretty fast monotonically in to the value “1.154724” as it is 0 of the function, okay that is the midpoint rule implementation another method which we looked at that time was the method of false position iteration after this we looked at the iteration the false point iteration we had slightly a different idea to find the 0 that is again we have guesses x_0 and x_1 such that f of x_0 is less than 0 f of x_1 is greater than 0.

So that was the same as the bisection or midpoint method, we just looked at, but we do not here at the mean x_0 and x_1 instead what we are trying to do here is was draw a line from x_0 , f of x_0 , to f_1 , f of x_1 and find a point which crosses the 0 axis and then we replace that as new x_1 value and that is what we are doing.

(Refer Slide Time: 19:04)

Method of False Position

Here again we will assume that the function has opposite signs on either sides of the root .

As in the bisection method we will make two guesses x_0 and x_1 with $f(x_0) < 0$ and $f(x_1) > 0$.

The new guess for the root x_2 is then the point of inter section of the line connecting $(x_0, f(x_0))$ to $(x_1, f(x_1))$ with the x-axis.

This is given by $x_2 = x_0 + \frac{f(x_0)(x_1 - x_0)}{f(x_0) - f(x_1)}$

Again we replace x_0 or x_1 by x_2 depending on the if $f(x_2) < 0$ or $f(x_2) > 0$ respectively.

So we had we will just look at the method the false position method, the summary of false position method let us say that the f of x like this okay then we will pick up new trail points which encloses the root right then we will evaluate the function f of x_0 and f of x_1 make sure that the error on either side of the 0.

(Refer Slide Time: 20:03)

False Position Method

$$f(x) = (667.38x^2)^6(1 - e^{-(1+667.38x^2)}) - 15$$

Pick two trial points which enclose root

Two points x_0 and x_1 enclose a root if $f(x_0)$ and $f(x_1)$ are of opposite signs

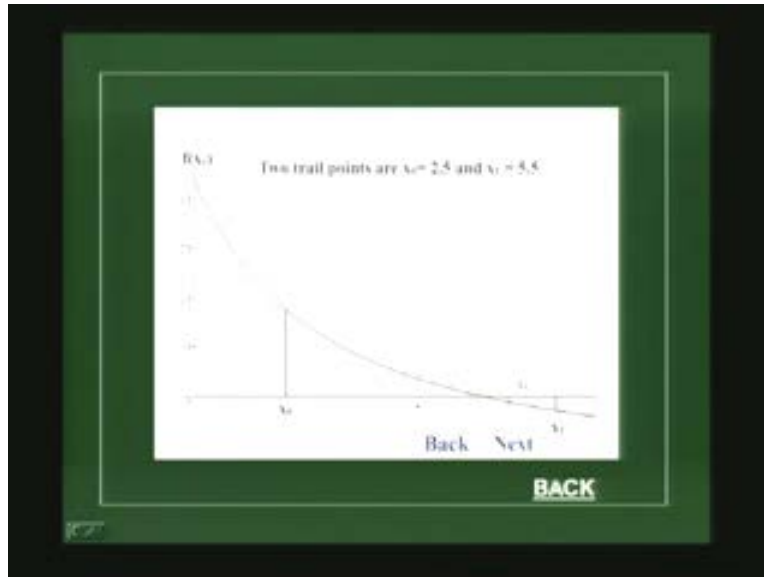
NEXT

BACK

Okay and then what we did was like this we have a function like this, some function like this and then we just picked up x_0 and x_1 values, that is just some guess we made sure that f of x_0 is positive in this is positive, is on either side of the 0 in the program we taking this x_0 of the x_0 and then we draw a line from this point to this point, okay we just

draw a line like this, we will just draw a line and that line, where that line crosses the if you have line here that line crosses the x axis that is somewhere here. Okay that would be my next guess now, if this value this is my x_2 ground value my x_2 value will be here were it crosses the 0 axis and that x_2 replace that x_1 because for that x_2 value the function is here in the case is negative that is a, that is a method which we are going to look at now we will just look at an implementation of the code here and that is called the false position method.

(Refer Slide Time: 20:18)



We will have program here the false position in the false position method again we have exactly the same structure that is what I said in the earlier program. So we are going to do, the only thing we are going to do to replace this function which we are called as midpoint method, okay we are going to just simply replace the method by false position by another function called false position and again past to this program the pointer to our user defined function which is again we take as $1 - x^2 + \log_{10}(1 + x)$. So we will minimize the same program, the same function, we will find the 0 of same function as what we did in the midpoint method but we will use a new function called false position.

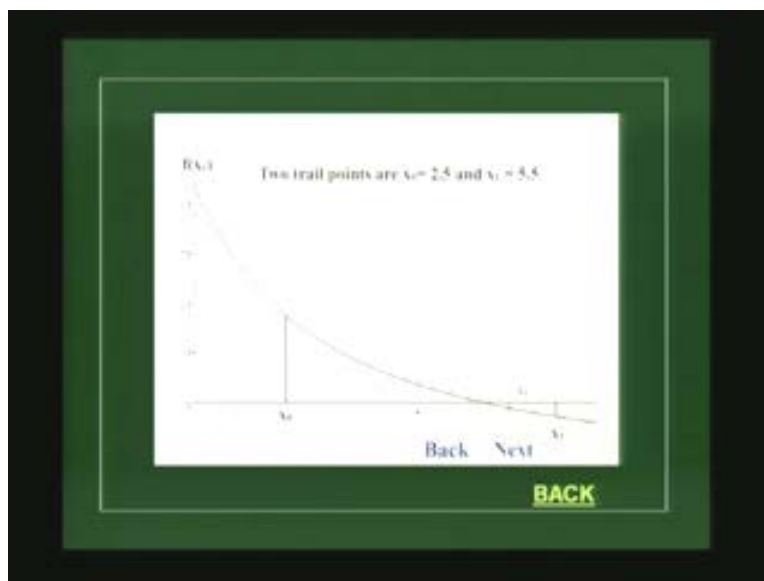
That is what we are going to do, so in this function what we do now again here is again, we written it as floating point this type cast as float here okay and so x_0 and x_1 it reads out also need 2 guesses both this midpoint and also the false position needs 2 guesses. So we will give x_0 and x_1 on either side we again we choose one f of x_0 to be negative and f of x_1 to be positive like in the other case. Now this function is here, so it gets the new x value as now this is by solving the equation of line connecting these 2 points which we just now saw.

(Refer Slide Time: 21:26)

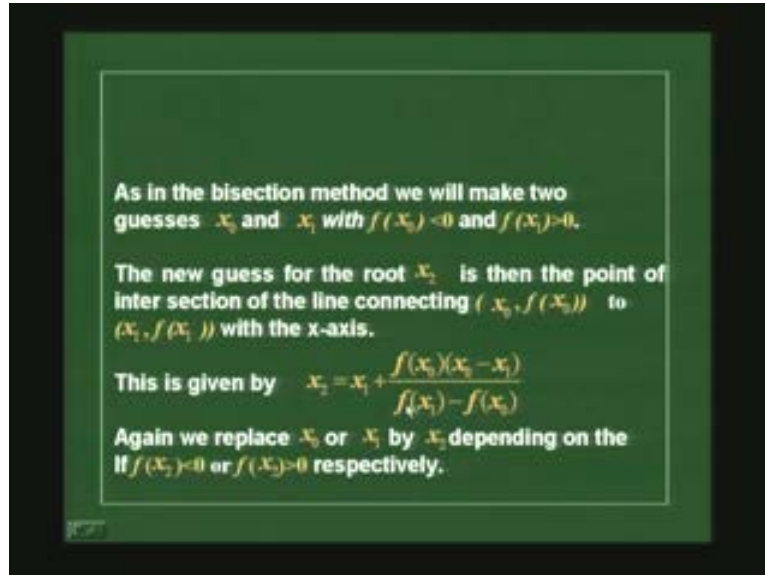
```
float false-position(x0,x1,function)
float x0,x1;
float function(float);
{
int i,j,n,m;
float e,f;
float x2;
e=1.0;
while(e>0.0001)
{
x2=x1+function(x1)*(x0-x1)/(function
(x1)-function(x0));
if(function(x2)<0)
{e=fabs((x0-x2)/x2);x0=x2;}
25,11-18 46%
```

We saw the equation of a line connecting this 2 points with the x axis, so we know how to right an equation of line connecting 2 points and find where the crossing point of that y equal to 0 line which is the x axis. So that gives this particular formula okay that gives this formula for the new value x_2 as x_1 plus f of x_0 into x_0 minus x_1 by f of x this is actually f of x_1 into x minus x_1 , x_1 minus f of x_0 this is our formula okay in this case I had taken f of x_1 as positive f of x_1 as negative.

(Refer Slide Time: 23:03)



(Refer Slide Time: 23:23)



That is the formula which you put in here. Okay and then that function value and again the function value is negative, if the function value is negative then we will replace x_0 by that and the error is x_2 minus x_2 of this midpoint none of the this is has been changed, if f of x if the function value is positive we will replace by x_1 by x_2 and then find the error as x_1 minus x_2 by x_2 print out again the error the new x value in the function value the new x_2 not changed the function is the same one minus x square plus log base to 10, 1 plus x okay

So the idea is that we could use the same we will replace this program here and pass the pointer to that function user defined function now if I want to minimize some other function. I do not want to change anything in this program the only thing I have to change is defined a new function instead of this func that you could see here, if I have a new function um and then pass the point to the next function to this program. So we will see how we could, what we get from this okay so we will now compile that false position. So we will compile this program here, now we will run this the program and again as in the earlier cases it just waiting for this two initial guesses x_0 and x_1 x_0 on the negative side and x_1 on the positive side of the function.

So again I will make an initial guesses 2 and 1 and you can see that of the method of the false position that converges pretty fast so when we ran the midpoint method it have to go the many iterations but here the convergence is much faster, okay even though both have linear convergences which we will see so, this particular method converges much faster and we of course we will get the same answer. So again this is monotonic the converges the error the decrease monotonically the function value goes to 0 monotonically and we have the 0 as "1.154694".

(Refer Slide Time: 23:46)

```
[sunil@dali lect22]$ gcc false-position.c -lm
[sunil@dali lect22]$ ./a.out
2 1
0.096332 1.106601 0.099017
0.029587 1.140340 0.030107
0.008812 1.150478 0.008935
0.002599 1.153476 0.002633
0.000764 1.154359 0.000774
0.000225 1.154618 0.000228
0.000066 1.154694 0.000067
The zero is 1.154694
[sunil@dali lect22]$
```

So that is the second method which we saw this thing and then we will look at some of the methods that is the fixed point iteration so we will look at that something we again saw in the earlier lecturer. So the fixed point iteration we will just go through that summarize what we saw in the last lecturer once again before we look at the program.

(Refer Slide Time: 26:16)

Fixed point iteration

This method can be applied whenever it is possible to write the equation $f(x)=0$ in the form $x=g(x)$. We start with a guess solution x_1 and obtain the new value from $x_{i+1}=g(x_i)$. The method is illustrated graphically below. It is obvious that the method will work only when the derivative $g'(x_i)$ is less than one.

[Block Diagram I](#)

[Block Diagram II](#)

So in this method we want to right the function f of x equal to 0 which you have to solve or you have to find the 0s of function of f of x , we want to replace that by function an equation of this form x equal to g_x basically what it says is, I should be able to right f of x

as x minus g_x and I will say x equal to g_x . So this I will use get to my iteration the iterative step that is we start with the 1 guess now I do not need two guess as we saw one problem with having a midpoint or false position method was that you need to have two initial guess on either side of the 0 that is more difficult to do.

So we will now looking at the methods which we need one guess and it can be on either side of 0 we saw that um this is fixed point iteration is 1 such method and were, we just make one initial guess and use this form and this kind of mapping okay and then we will use this form to generate a new value. If x the guess is actually the 0 of the function f of x then x should be equal to g of x right because, now we will replace the function f of x by g of x minus x .

So if x is actually 0 of the function f of x then x should be equal to g of x the iteration stop we will again look at the error in the same step as one before x old minus x new divided by or x_i minus x_{i+1} plus 1 divided x_i plus 1 by mod as our criterion for error. So error is defined as that that as to go the value which we defined that is the thing which we will again, so that is the method we will implement here then we will look at code here.

(Refer Slide Time: 26:29)



So similar structure, we will look at fixed point iteration that is this program okay again here not done not change anything expect changing this function name now to this fixed point so I just replace each time I want to it replace it by different method, I replaces this particular program, this particular function which is my fixed point which is now fixed point iteration and some slides changes we now need only one initial guess, so we do not need two initial guesses and we have to passed only that this function the fixed point turn to be a solution you be print out here. So gain we have to pass the pointer to the function which you want to look at or we want to find the 0 of that is func same as before that 1 minus x square plus log to the base 10 x but now we need to right that this a the x minus

g of x . So what this function should now return is g of x , okay let us some changes, let we go through this to once again.

We have the main program which called this method as fixed point and it passes one initial guess and a pointer to a function which now returns g of x is not f of x . Okay so now it will go to fixed point that function, now this function we initialize the error to be one and we demand the error to be as same less than 10 to the power of 4 for this criteria to be satisfied. Okay so now, we use x equal to g of x here x_2 equal to a function of x_1 and our initial guess.

So that is this function g of x that means so I need to write 1 minus square log 10 of x in the form of x minus g of x and then write g of x that is what I have done here. So now I have written it as here 1 plus log10 of 1 of x_1 plus x divide by x that is my g of x you can see that this minus x is my f of x right, so this minus x equal to 0 is my old f of x equal to 0 equation that is this function here 1 plus log to the base 10 or 1 plus x by x minus x equal to 0 is the same as the f of x equal to 0. So my g of x_1 plus log to the base 10 into 1 plus x whole divided by x divides the starting from here to here. Okay that is the function okay now we will use this mapping that is x_2 function is x_1 and now we do not have we just replace all the new functions now we do not check whether the function is negative or positive before we replace of whole value because of the only one guess here.

Okay so the error is very simple x_1 minus x_2 by x_2 that is what we have to do and we will printout the error the new x_2 value and the function value x_2 , and then remember that now we called x_2 as x_1 and I compute the error and I have to replace x_1 by x_2 now because the next round when I have to go I have to called new x_1 , so I had initial guess the initial guess now replace by x_2 here this is the next guess that is return by the function that is the fixed point iteration in short it is just writing x_i plus g of x_i checking whether actual 0 or not if not I will replace x_i this value by this let we continue with the iteration lets run this code again, so like before we run this code so now this is fixed okay we now have to make one guess let us gave it as 1 okay it goes through and gives as the this 0.

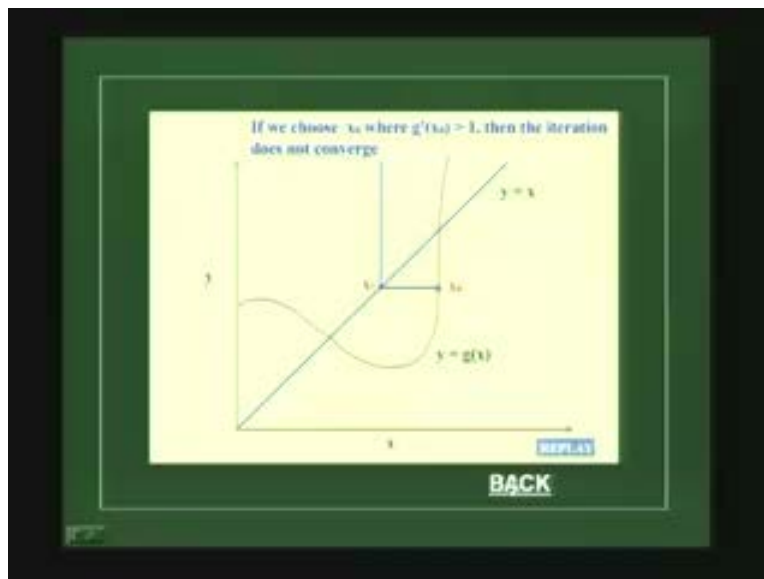
So we need to give only one guess I could also give the guess as 2 or anything. So it gives still the same as 0, so I just doing this iteration, so now remember this will not work all the time it worked in this particular case but we know that it may not work all the time and as you can see that much faster than the, it is not much faster than the this function it is not much faster than the false position method which we looked at, it have to go through many iteration before it reaches that. But the advantage of course is that we need only one guess that is the big advantage when it comes to the functions whose graphical form which will be not able to see of the function itself it will return some other program okay we do not know about 0.

(Refer Slide Time: 28:33)

```
0.000827 1.155158 1.154369
0.000683 1.154369 1.155020
0.000564 1.155020 1.154483
0.000465 1.154483 1.154927
0.000384 1.154927 1.154560
0.000317 1.154560 1.154863
0.000262 1.154863 1.154613
0.000216 1.154613 1.154819
0.000178 1.154819 1.154649
0.000147 1.154649 1.154789
0.000122 1.154789 1.154674
0.000100 1.154674 1.154769
0.000083 1.154769 1.154690
The zero is 1.154769
[sunil@dali lect22]$
```

So we cannot bracket it in that particular case this is definitely a great advantage this is to have an guess one point instead of 2 point guess but remember that it is an drawback the drawback being in that cases where it is a slope is greater than 1, okay if the function like this which has 20, let us say this particular case it will converges to that but when it is somewhere here we have to find this point it will not converge very easily.

(Refer Slide Time: 33:35)



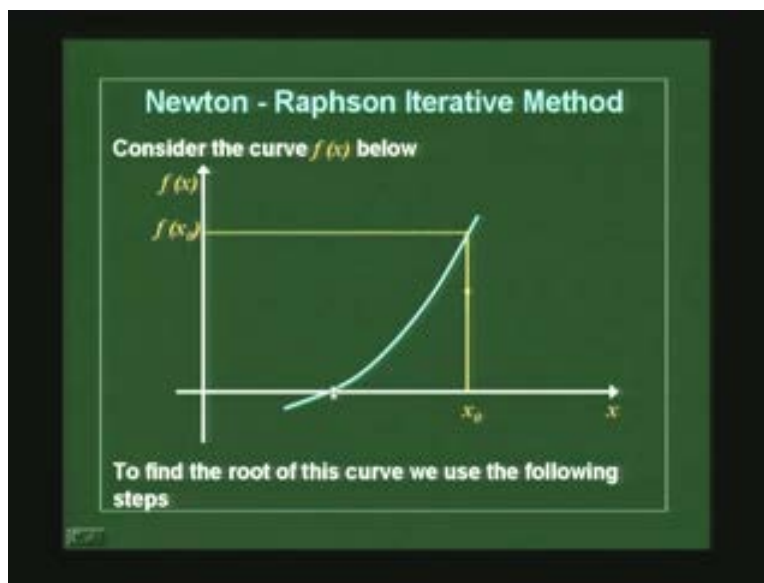
So remember that what we have doing is by fixed point iteration is actually solving finding the intersection two lines that is y equal to g of x and y equal to x that is equal to x equal to g of x in the intersection point of these 2 lines, this y equal to x, y equal to g of

x , we basically have an equation that x equal to g of x that is what we want to solve for it right. But we find that when you have intersection points like this were the slope of this g greater than the 1, we saw in the last lecturer that this will not converge okay it will not this is not guarantee to give a solution all the time that is the method which we looked at last time okay and then we briefly saw okay some other methods. So we will now go into another method which also again uses the same on the one initial guess that is Newton Raphson iterative scheme.

So in the Newton Raphson iterative scheme, we again make only one guess, okay let us say x_0 we took it also what we going to do is here is to actually draw a tangent at this point that is the basic idea of Newton Raphson method make you one guess and draw a tangent at that point to this line okay that is means we need to find tangent to this line that is the derivative of the function at that point extend the tangent all the way on to the x axis and find the point at which the tangent meets the x axis okay.

So that and then take that as your new guess that x value as next guess that is the function will be some were here then right and then you draw a another tangent okay find it where it meet the x axis and then you find the function value there and you draw the another tangent etcetera till you reach the 0. Okay that is the basic essence of Newton Raphson method.

(Refer Slide Time: 35:41)



Okay that is again summarized here. so we will determine some we will first make some initial guess here determine the slope the tangent of the function at that point x equal to x_0 let us called that f prime of x and then we will get the next root there okay by using this formula that is we will actually finding the point at which that tangent meets the x axis. So that is what the equation results from okay, so x_1 equal to x_0 minus f of x_0 by f prime of x basically what we trying to do is we are expanding the function f of x using a tailors series and keeping it up only up to order 1 and then saying that my f of x plus delta

x is f of x plus f prime of x is derivative of x at x into Δx and saying that f of x plus Δx is now 0 or we assuming that or we are solving that or finding the intersection of this line, this tangent line with the x axis that will result in the formula x_1 equal to x_0 minus f of x_0 by f prime of x_0 .

(Refer Slide Time: 36:51)

STEP 1:
Determine the slope of $f(x)$ at $x = x_0$. Call it $f'(x_0)$

STEP 2:
Pick the next approximation x_1 for the root by using the equation

$$\frac{f(x_0)}{(x_0 - x_1)} = \tan \theta = f'(x_0)$$

or,

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

(Refer Slide Time: 38:17)

In general we proceed from the i th to the $(i+1)$ th approximation using the relation

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

STEP 3:
The iterative cycle is stopped when two successive values of x_i are nearly equal within a prescribed tolerance

Okay that is the general form, so in general we will write it as in the x_i plus 1 in the iteration scheme we will write it as x_i plus 1 is given by the previous guess x_i minus the function value at x_i divided by the derivative of function at x_i . Okay and there of course as before the iterative cycle is stopped when x_i plus 1 and x_i are of the same or f of x_i

vanishes f of x_i is 0 x_i plus 1 equal to x_i the iteration stop okay now we could see the implementation of this okay.

(Refer Slide Time: 39:07)

$$f(x_{i+1}) = f(x_i + h) = f(x_i) + hf'(x_i) + \frac{h^2}{2!} f''(x_i) + \dots = 0$$

Neglecting all terms of second and higher degree in h we obtain

$$f(x_i) + hf'(x_i) = 0$$

$$h = -\frac{f(x_i)}{f'(x_i)}$$

(Refer Slide Time: 39:25)

If $f'(x_i) \neq 0$ for all x_i

$$h = (x_{i+1} - x_i) = -\frac{f(x_i)}{f'(x_i)}$$

$$x_{i+1} = x_i - \left[\frac{f(x_i)}{f'(x_i)} \right]$$

So in a in a quote, so as I said that the essence of this is that we will say that f of x_i plus 1 is actually f of x_i plus h , h is the Δx change in x which is the expanded Taylor series right and then I terminated here and I say that it goes to 0, so by my error in this particular case of this order and then I will solve for this that is what I am doing. So there are cases again in which this function, this will not work okay and we will see that particular case implemented and then the use of that okay. So before we look at the

disadvantage of Newton Raphson method we will just see one case where it is implementing.

Okay that is let you show you one program here its called Newton, okay so here is the function okay, so again I use the same structure I have the main program which will taking one guess again we need one guess here just like in the fixed point iteration. Okay and I called this function here pass this, call this function Newton here and pass that by initial guess and a pointer to my function that is to be whose 0 I need to be find. So now what is this program do this Newton this part which it uses an Newton Raphson method to find the 0 of the function okay, so now it call the function okay it is now this function which is to this pointer we have passed on to the function as again here that is been called here, so now we called that function.

Now this unlike the earlier case where the function was returning just the value of the function now we also need the derivative okay for the time being you forget this two lines I will come to this later we are now using the function of the form x minus exponential minus 2 star x . We are trying to find the solution of this equation that is x minus exponential minus 2 star x equal to 0 that is what we are trying to find the function is to 0s of find is x minus exponential minus 2 star x .

We will not use $1 - x^2 + \log(1 + x)$ at this point, we will find the solution and 0 of different function and now we need the derivative of the function so the derivative is given as $1 + 2 \cdot \text{derivative of this } 1 + 2 \cdot \text{exponential minus } 2 \cdot x$. Okay so now this function is called as slightly different, let me explain this. So this function this new func which we have the in this particular case okay is not returning anything okay its void it does not return any value. So it is unlike a previous case we have declared as float because it was returning a floating point number which is a function value.

So this particular function does not return any value it takes 3 arguments okay one floating point and two pointers and two floating point that what is it take argument, okay that is what it been declared here also this is also function, this is a function, this something called as function that is the three arguments that is float star and float star okay so the 3 arguments x_1 what I passed in to that function okay and f the function value at x_1 and the derivative of function x_1 it return through this thing.

Okay it replaces f by the function value at x_1 and replaces $derive$ by the derivative of the function f at x_1 that is what this thing can doing you can see that here is the pointer, so these are see this y derive are pointers such why star y aster xy equal to x minus exponential minus 2 x it guess the x value and it gives this computes this y and derive at this point okay and replaces this here and then it goes back. So when it goes back to this function now when it makes call here comes with new f value at x_1 and derivative value at x_1 .

So I uses that to find the new x_2 x values x_1 minus the function value by derivative note that asterisk f because this is pointers okay now I have the error as absolute value of the x

1 minus x_2 by x_2 and I am just printing it out and I replace by x_1 by x_2 just like the fixed point iteration scheme. So the difference between the fixed point iteration scheme this only the this step here that is here the new x_2 value is x_1 minus star f plus, star f divided by that is aster f because of the point derivative f by derivative of f that is what short it is and then I computed the derivative the new error and I print out the iteration scheme continue till the error less than “.001”, we will run this code here okay.

Again it needs a guess, so you will guess we will need now this is the different function we should plot the function and to see okay so our function was if you remember the one which your trying to solve was x minus exponential of minus 2 x because we will plot that function between 0 and 2 let say we plot it the function is like that, okay so we will draw the 0 axis that the function you have we expect the solution to be some were less than “.5” that is the x axis somewhere here is the solution is less than .5 that is what we trying to do find. So we will ran that solution to be point 042 “.42” and notice that I am printing out more than 1 variable here okay.

(Refer Slide Time: 39:45)

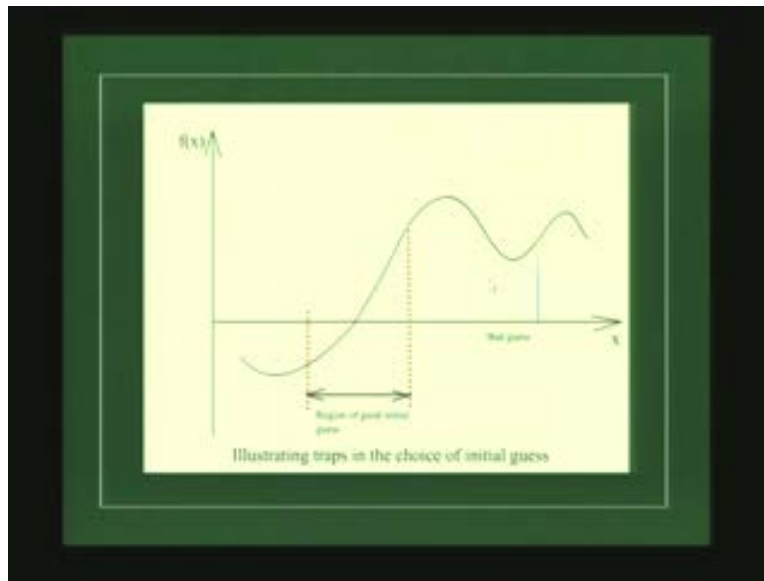
```

[sunil@dali lect22]$
[sunil@dali lect22]$
[sunil@dali lect22]$
[sunil@dali lect22]$ gcc newton-raphson.c -lm
[sunil@dali lect22]$ ./a.out
1
2.129686 0.319521 1.000000 0.864665 1.270671
0.240760 0.420843 0.319521 -0.208277 2.055596
0.012776 0.426289 0.420843 -0.010141 1.861967
0.000032 0.426303 0.426289 -0.000025 1.852629
The zero is 0.426303
[sunil@dali lect22]$
  
```

So I am printing out here the error the new x_1 , x_2 value the x_1 value, so this is the old guess this is the new guess the function value and the derivatives that is what we have printed out here. So the function value goes to 0 and the error goes to value which is less than the tolerance which we have defined in just 4 steps, so this is definitely much very faster, the really fast way of computing the zeros okay and then most often used method than anything else okay because it need only guess, okay the only one guess to be given of that 0 of that guess can be the anywhere and it converges very fast when it converges that is the advantage of this this is very fast only one guess now there are some disadvantages that is what we have to look at here that you can do at an endless cycle okay or even it diverges in cases that is we will look at some the pathological cases like this in this program. So here is the case where it can actually get in to an endless loop.

So let us say we want to find this we our initial guess happened to be here we want to go here, okay we want to find a 0 which its here and we do not know the functional form, so we just made an initial guess which is here. So what is that method, the method is to draw a tangent and that is what a blue line is showing you I have to draw a tangent and the tangent you look at it meet the x axis here right when I draw a tangent at this point here and it meet the x axis here and then I take the take the x value to be that okay then I draw a tangent here that that meet the x axis here. So but it then goes back here and draw it is x axis here so it keeps jump iterating between these two points.

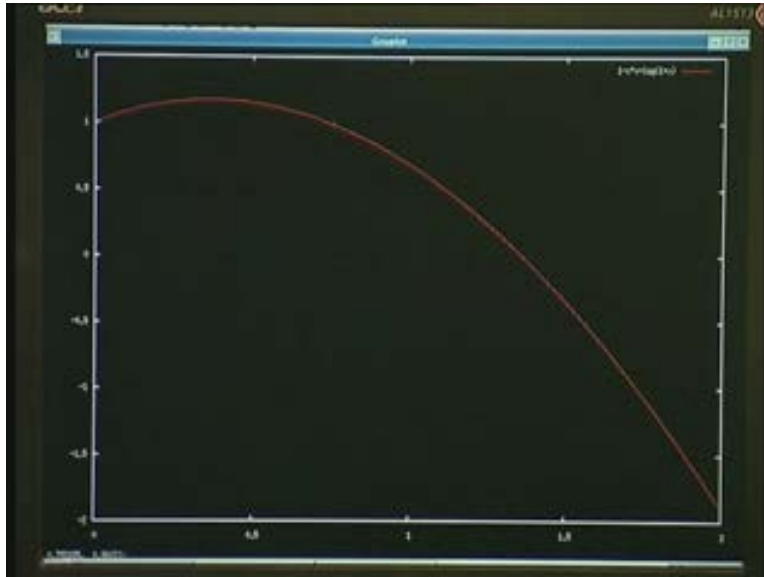
(Refer Slide Time: 46:44)



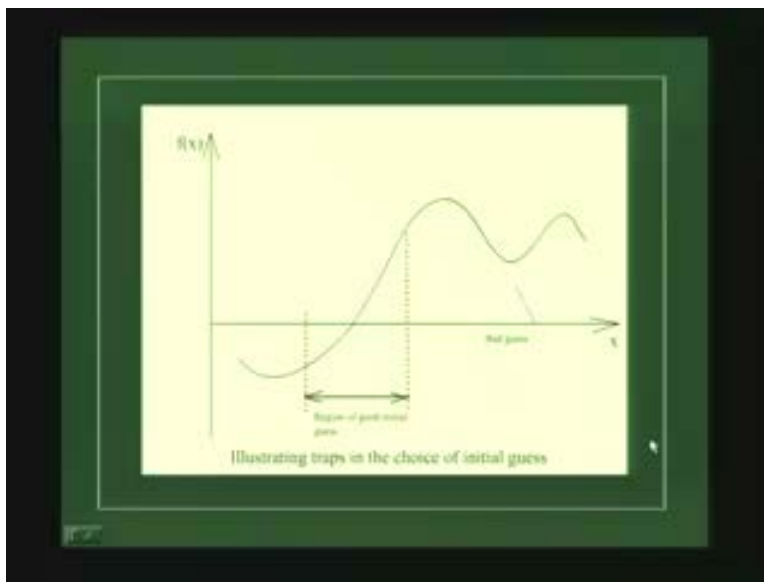
So we can get to an endless loop in this particular case another case happens can happen in which for example we will look at this another function so we will look at the whole function which we called as which we had $1 - x^2 \log(1 + x)$. Let us look at this function. Okay so we will look at this in this particular case, okay now if I make an initial guess let us say here okay and then also I can get in to trouble because if I, if I make an initial guess now that is the very large value somewhere very down there. Okay so the tangent here the tangent here meet the x axis somewhere very much down okay and then that is one case we can actually diverge if you if you for an example make a make a guess some were close to this okay then the value at which you are going to get the where the tangent going to meet the x axis that is going to be extremely large number.

Okay and then we can get in to that okay so there are two cases so you will see if $1 - x^2 \log(1 + x)$ if I try to run this and it does not converges so there are two pathological cases which I can tell you that initial guesses are somewhere close to were the second derivative vanishes and okay then you can have very small f' get in to an error because remember our function our method requires f by $f - f'$ to be form right. So that that can gives large error that is one reason were you can get this when you go here.

(Refer Slide Time: 48:01)

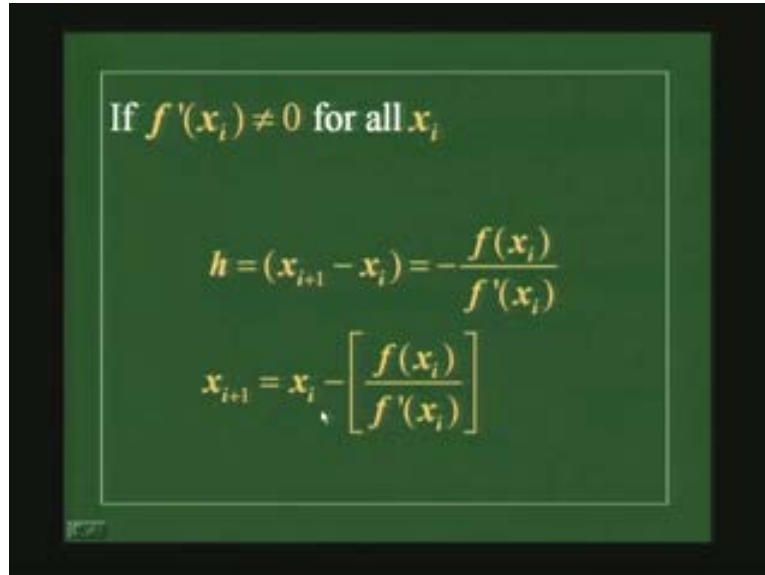


(Refer Slide Time: 49:27)



Okay so we will see here that is the formula you are using if f' is very small okay then also we can get large error in this we can diverge we can go to large value and then things diverge that is one case where it can go, it can go get into trouble another case which we saw get in to an infinite and do loop so it is an another case. So in the case for example if you take this particular function one minus x^2 we will actually have trouble using Newton Raphson method but other methods will work.

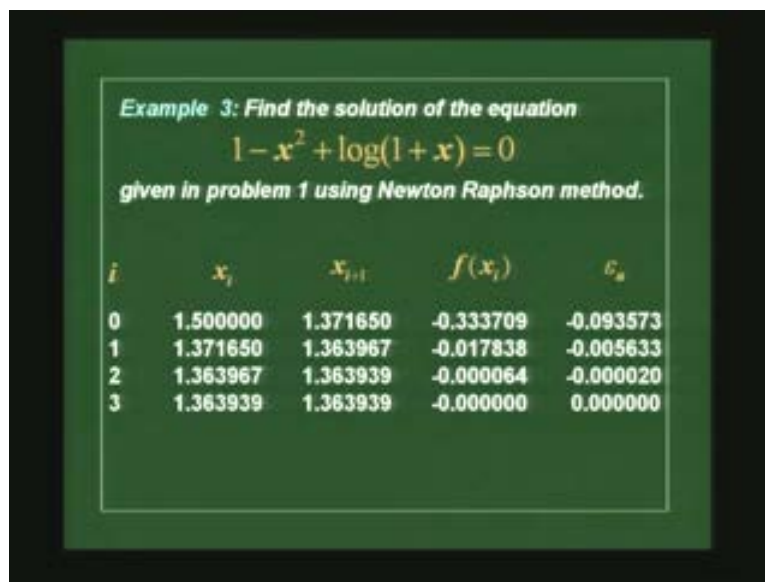
(Refer Slide Time: 49:50)



If $f'(x_i) \neq 0$ for all x_i

$$h = (x_{i+1} - x_i) = -\frac{f(x_i)}{f'(x_i)}$$
$$x_{i+1} = x_i - \left[\frac{f(x_i)}{f'(x_i)} \right]$$

(Refer Slide Time: 50:15)



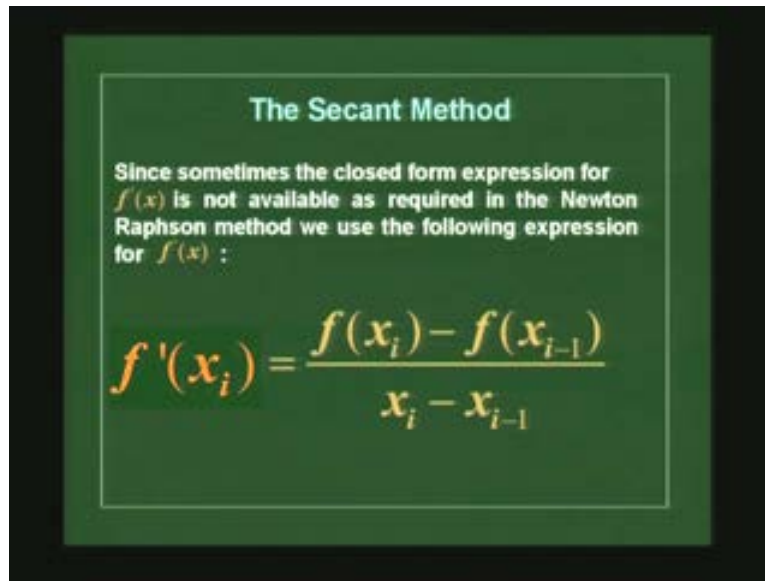
Example 3: Find the solution of the equation
 $1 - x^2 + \log(1 + x) = 0$
given in problem 1 using Newton Raphson method.

i	x_i	x_{i+1}	$f(x_i)$	ϵ_a
0	1.500000	1.371650	-0.333709	-0.093573
1	1.371650	1.363967	-0.017838	-0.005633
2	1.363967	1.363939	-0.000064	-0.000020
3	1.363939	1.363939	-0.000000	0.000000

So another way to is, another disadvantage of this program, this program of having is using Newton Raphson is we need to compute f' prime of x that is the derivative of x . So when the derivative of f of x is not available, okay we will again have problem with this method because this function value might be return by some other program and we do not know what the derivative is. So in that case, we may have to make an approximation derivative by what is called difference method, okay a differences approximation to a derivative.

We will see more of this later when we look at differentiation and integration, so here for the time being let us take this formula which is known to most of us, that derivative we will replace by what is called backward difference formula that is we take $f(x_i)$ minus $f(x_{i-1})$ divided by x_i minus x_{i-1} . So now this formula is backward difference method, if difference formula if x_{i-1} is some value which is less than x_i okay and then we will call the backward difference formula but in this particular case in this iterative scheme x_i minus x_{i-1} are simply just two guesses, two iterations, two guesses in the i th and $i-1$ iterative steps but this is the formula which we can use, so the $f(x_i)$ is the value x_i and x_{i-1} is another value. So we will make two guesses here so to start with x_i and x_{i-1} and then we will find this function this derivative $f'(x)$ as $f(x_i)$ minus $f(x_{i-1})$ divided by x_i minus x_{i-1} and we will use Newton Raphson formula again as x_{i+1} as x_i minus $f(x_i)$ by $f'(x)$.

(Refer Slide Time: 50:25)

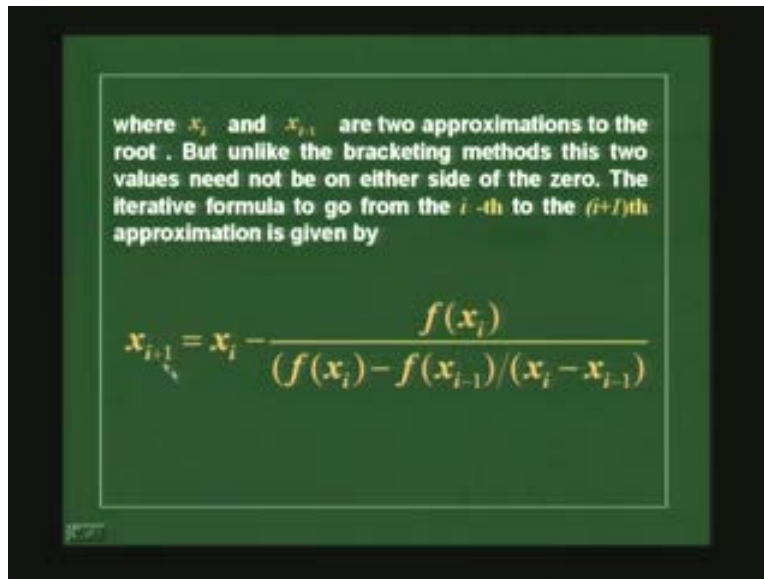


Okay so the idea is that we will not find $f'(x)$ exactly like in a function but we will simply replace that $f'(x)$ by this particular formula $f(x_i)$ minus $f(x_{i-1})$ divided by x_i minus x_{i-1} where we start with x_i and x_{i-1} are our guesses, now what is the difference this and the false position method, that false position method we made two guesses and we did something exactly like this, this is again going to be the line connecting x_i , $f(x_i)$ and x_{i-1} , $f(x_{i-1})$ line connecting these two points on the function. This point x_i , $f(x_i)$ and x_{i-1} , $f(x_{i-1})$ and this line where does it meet the x axis that is the point going to be new iteration point that is in the false position we made two guesses that is x_0 and x_1 and we said $f(x_0)$ the line connecting x_0 , $f(x_0)$ and x_1 , $f(x_1)$ where it meets the x axis is our new x value, and this is exactly same that sense that we are again going to use two guess values okay x_0 and x_1 and let us say and we are going to draw a line connecting these two okay and we are going to take the point at which that meets x axis as our new guess, what is the difference between these new method which we called the secant method and that the false position method

is that these two guesses x_0 and x_1 does not have to be either side of 0 it can be anywhere and hence it also has a disadvantage that it might not converges.

So it can just get into same kind of 0 and it will not may converge that is what the disadvantage of this again we will details here. So okay its similar to bracketing method but we do not bracket here so the advantage is that we do not have where the 0s, so we do not worry about actually bracketing it on other hand it has disadvantage that may not converge while the bracketing methods are granted to converge.

(Refer Slide Time: 54:13)



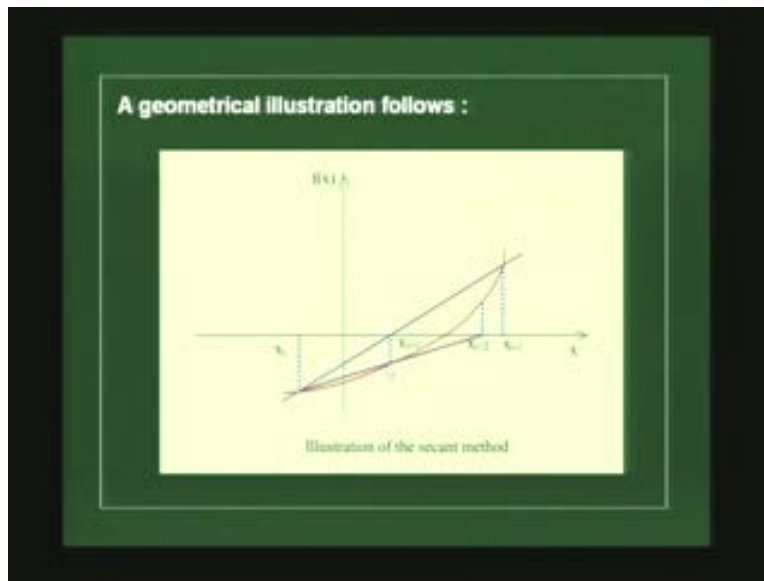
So here is the formula which going to use in this particular case that has been graphically shown here okay so you make some guesses here x_i minus x_i okay does it have to be on either side of 0 in this particular case it is, but it does not have to be on either side of 0. So we will draw this line here okay where it meet the x axis okay and now one more differences from the false position method is that, this x_i plus 1 now will replace x_i and x_i so now x_i minus 1 now go to x_i and x_i plus 1 go to x_1 if x_i minus goes x_1 x_i plus 1 will go to x_1 okay.

So we will always replace the last one that is the in the case of false position we the new point on the negative side that below the x axis, then we always we will replace x_0 with the x axis but here is nothing like this. So these first these two points were taken right and then then these two points this will be taken okay and then we will found the new intersection point right and we will take now these two points etc we will continue like that okay.

So now we will draw a line like this and will go here and then we will draw a line like this, in this case it will converge but in the false position method remember that we will always replace if the new iterative method that is the point in which it meet the x axis if this is less than 0 less than 0 that is function value is less than 0 it will replace x_0 if it is

more than 0 then replace x_1 , but here we do not do that we always replace the last point with the new one okay. So the irrespective of whether it is this less than 0 or greater than 0, we are going to say that the next line we are going to draw is between x_i and x_i plus 1 and the next line would be between x_i plus 1 and x_i plus 2 etcetera, okay that is the method.

(Refer Slide Time: 54:51)



Okay and we will see an implementation of this cases, the pathological cases both this and the Newton Raphson method where it does not work in the next lecturer and also we will do an more analysis of how what is the convergences rate of this three, of this methods which we looked at so far in the bracketing methods and Newton Raphson secant method we will see in the next lecturer.