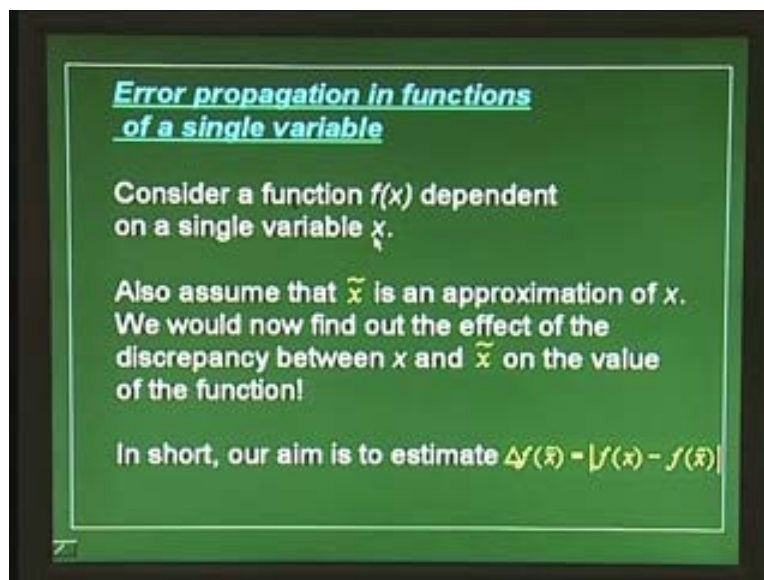


Numerical Methods and Programming
P. B. Sunil Kumar
Department of Physics
Indian Institute of Technology, Madras
Lecture - 8
Polynomial Interpolation

Today again, we will look at error propagation and the effects of finite representation of floating point numbers on programs, and the results we get from programs. Last class we had looked at error propagation in functions of single variables, and we said what we mean by this is that, if there is an, if there is an error in variable x , and then what is the effect of that error on the function f of x . That is x is a floating point variable that is represented on a machine using a finite number of digits.

(Refer Slide Time: 02:04)

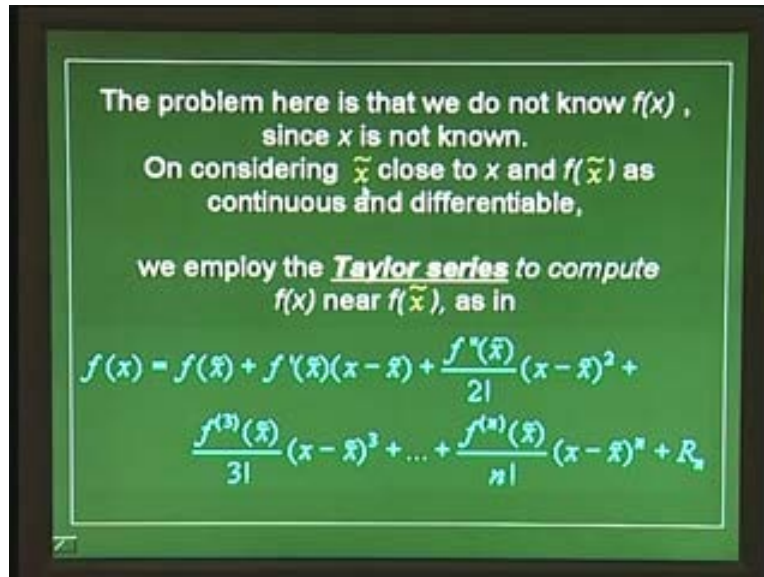


And because of that a function which depends on this variable, a function of this variable would, the numerical value we get from this function would depend, would have some error. So now the question is: how does this error in this propagate into that? That is what we looked at and we said that if we assume x tilde as an approximation to x , that is x tilde is the finite machine floating point representation of the actual value real variable x , and then what would be the effect of the discrepancy between x and x tilde on the value of the function? That is what we are going to look at, or we have to estimate Δf_x tilde which is actually the difference between f of x , the real variable x , the actual variable x minus the floating point variable x , tilde f of x tilde.

We are not interested in the sign, so we take the modulus of that. So that is the estimate which we are to make. So what we did was, we could assume that x tilde is close to x , the real value x , and we could expand f of x tilde around f of x , or f of x around f of x tilde.

So $f(x)$ is the real value which you want, and we are saying \tilde{x} and x are very close to each other, and we know that when a function, if you want to evaluate a function which is close to some other to a variable which we know to a value which we know, we could expand the function in the Taylor series around that.

(Refer Slide Time: 03:24)



This is basically a series of derivatives of the function. Here, we assume the function to be a continuous differentiable function. That is important. So we have a continuous differentiable function, and then we could write $f(x)$ as $f(\tilde{x})$ plus f' , which is the derivative of “ f ” evaluated at \tilde{x} into $x - \tilde{x}$, that is Δx , and the second derivative which I have written as f'' divided by 2 factorial, into Δx square plus f''' . So this is the third derivative. Again, evaluate it at \tilde{x} . So this notation is $f(x)$, means $f(\tilde{x})$ means, “ f ” evaluated at \tilde{x} , f' of \tilde{x} means f' evaluated at \tilde{x} , and similarly all derivatives.

So you could expand it in the series. We are only interested in this particular calculation only up to the first order. So we could say $f(x)$ is $f(\tilde{x})$ plus f' of \tilde{x} and $x - \tilde{x}$ which is our Δx . So from here, you can see that I can make an estimate of what will be $f(x) - f(\tilde{x})$ as, how does $f(x) - f(\tilde{x})$ depend on $x - \tilde{x}$? And you can make an estimate of that, and you can see that it will be proportional to the derivative, the first derivative of “ f ” at \tilde{x} . That is what we wrote it here as. So that is the estimate you will have. $f(x) - f(\tilde{x})$ approximately equal to f' of \tilde{x} into Δx . Or the error which we originally want to estimate is Δf , is f' of \tilde{x} into $x - \tilde{x}$. That is what basically we get.

(Refer Slide Time: 06:07)

Dropping the second and higher terms and rearranging we obtain

$$f(x) - f(\bar{x}) \cong f'(\bar{x})(x - \bar{x})$$

or

$$\Delta f(\bar{x}) = |f'(\bar{x})||x - \bar{x}|$$

So, if the derivative is large then the error propagation would be very large. So, if function has a small derivative, then the error we make in the floating point representation does not have a serious implication, serious repercussion. That is what we would see.

(Refer Slide Time: 06:32)

Error propagation in functions of more than one variable

When there are more than one independent variable.

For example, if we have a function of two variables u and v , the Taylor series is written as

$$f(u_{n+1}, v_{n+1}) = f(u_1, v_1) + \frac{\partial f}{\partial u}(u_{n+1} - u_1) + \frac{\partial f}{\partial v}(v_{n+1} - v_1) + \frac{1}{2!} \left[2 \frac{\partial^2 f}{\partial u \partial v}(u_{n+1} - u_1)(v_{n+1} - v_1) + \frac{\partial^2 f}{\partial u^2}(u_{n+1} - u_1)^2 + \frac{\partial^2 f}{\partial v^2}(v_{n+1} - v_1)^2 \right]$$

Then the question we looked at was, how do we extend this calculation into functions of more than one variable? In that case, we could again have a Taylor series in two variables. We have an example here of two variables. Here is a Taylor series in two variables. Look at that, this is very similar to what we did for one, say, variable. So now

we want to evaluate it at some point, i plus 1 of two variables u and nu . There are two variables, and these two variables are to be evaluated at some point, i plus 1. Let us say there is a discrete set of points, and then we know the function value at some other point.

That is close by. So we assume that u_i , and u_i plus 1, the two different approximations of the variable u and nu_i and nu_i plus 1, are two different approximations of the variable i plus 1, and we assume these two approximations are very close to each other, and then we can take the function value at nu_i and u_i , and then expand it around that. So that is the derivative of the function with respect to u multiplied by the difference in the approximation, and the derivative of the function with respect to nu , and the difference in their approximation. So if there are, that is the first order term. And the second order term, there are 3, because we have second derivatives. There is a second derivative of “ f ” with respect to u , and second derivative of “ f ” with respect to nu and the cross derivative.

So there are three terms here. So you can see that now, it is very similar to the function of one variable. The only thing now there are since two variables, so even if the function is well behaved it is not well behaved even if it is a very small derivative with respect to one of the variables, u . If it has a large derivative with respect to the other variable, it would have still large propagation of error because we also have cross derivatives. But it is the first order, it is what we keep, is again is only this term. So we will have f derivative with respect to u and f derivative with respect to nu . That is what we will see.

(Refer Slide Time: 09:03)

On dropping all the second-order and higher terms

$$\Delta f(u, v) = \left| \frac{\partial f}{\partial u} \right| \Delta u + \left| \frac{\partial f}{\partial v} \right| \Delta v$$

where Δu and Δv = estimates of the errors in u and v , respectively.

Generalizing it further for n independent variables x_1, x_2, \dots, x_n having errors $\Delta x_1, \Delta x_2, \dots, \Delta x_n$ the following general relationship holds.

$$\Delta f(x_1, x_2, \dots, x_n) \cong \left| \frac{\partial f}{\partial x_1} \right| \Delta x_1 + \left| \frac{\partial f}{\partial x_2} \right| \Delta x_2 + \dots + \left| \frac{\partial f}{\partial x_n} \right| \Delta x_n$$

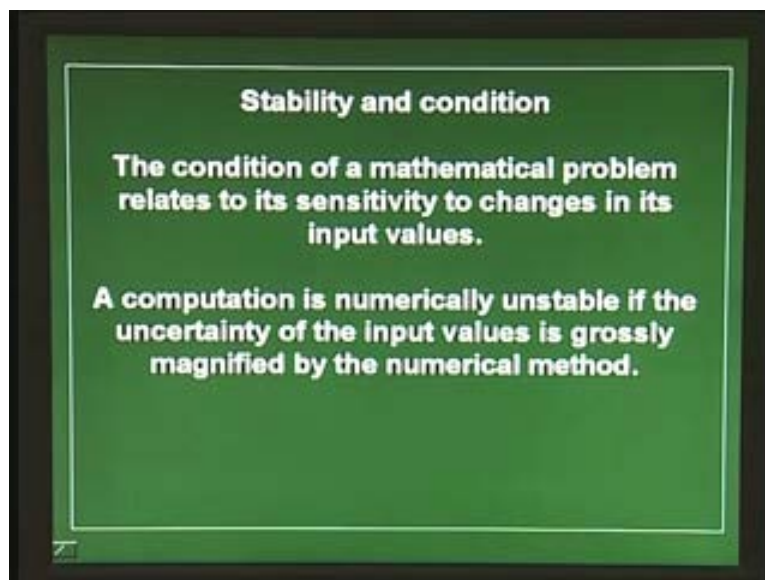
Now, we could write Δf as derivative with respect to u , and derivative with respect to nu . Again, we take the mod, because we are only interested in the, only interested in the total maximum derivative. So the relative sign is also not important. We are interested in the maximum error, possible maximum error, that is propagated. Hence, we take here the mod of both and then sum it up. So the error in one plus the error in other, does not cancel each other, it is an error. So it has to be taken the mod and the sum. The normal

error analysis routine. So this is the derivative, and these are the errors in the two variables which we assume. So now, we can generalize it further and write it for n variables. Then we would write it like this. So, we have a Δf into approximations, into variables up to n variables, and each of them approximated by the corresponding tilde variables: x_1 tilde, x_2 tilde, up to x_n tilde.

And then we have the derivative as the derivative of the function with respect to each one of these variables. That is what we saw. So we could actually look at the further implications of this. We saw that because of floating point representation, there are some errors which are introduced into the program, and we saw that these errors can propagate depending upon what the nature or the function of the variable is. So now, the question is, if you are doing a series of calculations with this, so how do we kind of quantify this? How do we say that this particular calculation is not viable because the error as it will not be possible to do this particular calculation because there will be too much errors in this thing, and the answers will be unreliable?

So how to make a quantitative estimate of this, and how do we classify programs or calculations which are doable, and which are not doable? So that is what we try to do in the next section. So we do this through 2 definitions. One is stability, and other is condition. So let us look at what a condition means. So a condition here, this is the definition here, is I am doing a definition here. The condition of the mathematical problem here, in this case, a numerical problem, relates to its sensitivity to changes in its input values. So, we will have to make a quantitative, we will have to make, we have to define a quantity which would actually tell me what the condition of this is. We will do that in a few examples.

(Refer Slide Time: 11:55)



If there is a particular step, a particular mathematical operation which is pretty crucial in the calculation, and it is extremely sensitive to the input values to it, if you want to calculate a function f , and evaluate it some value x , and if it is extremely sensitive to the value of x which is given to it, then that, we would say that is an ill-conditioned function, or ill conditioned problem. That is what you have to look at. So, and then stability is, it contains, a computation is numerically unstable if the uncertainty in the input values is grossly magnified by the numerical method. In the dynamical systems kind of language, one would quantify this with something called Lyapunov exponent, etcetera, which is basically saying that if you introduce a small error at one point, so you take a particular function and make two different approximations of that function, or you take a particular function and put an input value to it, x , let us say, f of x , and x as we know, is approximated.

And we two close approximations to the real value of x . x is a real number, and let us say we have floating point representation, and we have two different approximations of that, and we put that into this function, and then look at, as we continue the calculation, as we continue the calculation, how does these two function values vary? Let us say we have a series of calculations to do, and we start with f of x , and we are making these two different approximations to x . So we are starting from slightly different points and the question is as we continue the calculation, the answers we obtained, How close they remain or do they actually go away from each other variable very fast?

If they do go very fast away from each other, then we say that it is not a very stable routine because small changes in the input would get amplified as we go further down. That is the stability of the routine. So it will be unstable routine. It will be almost impossible to do this, do that particular calculation on a machine which has a finite number of precision, a finite precision. We would see these things in using some examples here.

(Refer Slide Time: 14:37)

The first order Taylor series relationship can be employed to estimate the relative error of $f(x)$ as in

$$\frac{f(x) - f(\tilde{x})}{f(x)} \approx \frac{f'(\tilde{x})(x - \tilde{x})}{f(\tilde{x})}$$

The relative error of x is given by $\frac{x - \tilde{x}}{\tilde{x}}$

A condition number is the ratio of these relative errors

$$\text{Condition number} = \frac{\tilde{x} f'(\tilde{x})}{f(\tilde{x})}$$

So let us quantify this thing. So, we just saw that we could expand, if you have an approximate variable. We could Taylor expand the function around the real value, around the approximate value, and quantify, and get the difference $f(x) - f(\tilde{x})$.

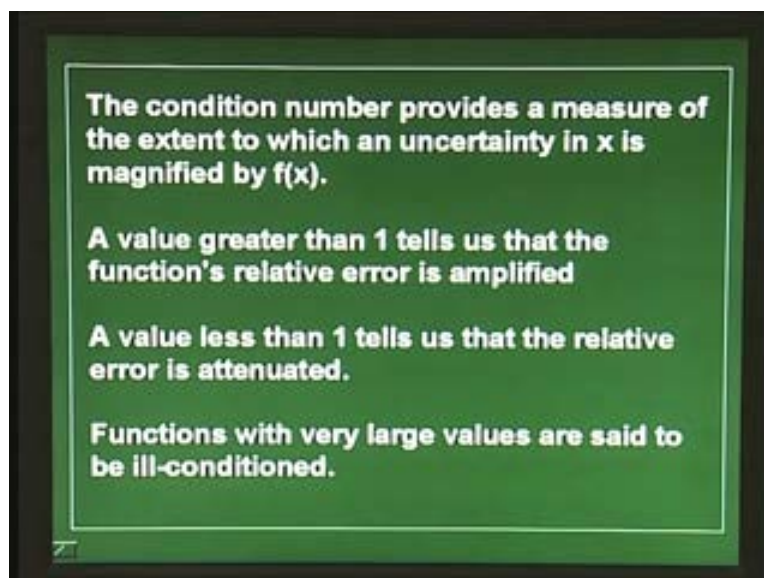
So now, the question is we take the relative error.

So you take $f(x) - f(\tilde{x})$ divided by $f(x)$. So that will be the relative error, and by the Taylor expansion method we would say that it is $f'(\tilde{x}) \Delta x$ divided by $f(\tilde{x})$. So what have we done here? We saw in just a few minutes back, that $f(x)$ can be expanded around \tilde{x} , and it will be $f(x) = f(\tilde{x}) + f'(\tilde{x}) \Delta x + \dots$. So $f(x) - f(\tilde{x})$ is simply up to first order, approximated as $f'(\tilde{x}) \Delta x$. That is the derivative of "f" at \tilde{x} into Δx , and we approximate this $f(x)$ by $f(\tilde{x})$ here in the denominator. So that is the relative error we approximate. Similarly, the relative error in x is given by $x - \tilde{x}$ divided by \tilde{x} .

So, then we define a condition number as basically the ratio of this, that is, the relative error in the function divided by the relative error in the variable. So if I change, if I have an error in the relative error in the variable, how much that gets amplified when I evaluate the function. That is the condition number. So you can see that if I divide this by this, I would get that. I get $\tilde{x} f'(\tilde{x})$ divided by $f(\tilde{x})$. That is the condition number. That is an important quantity.

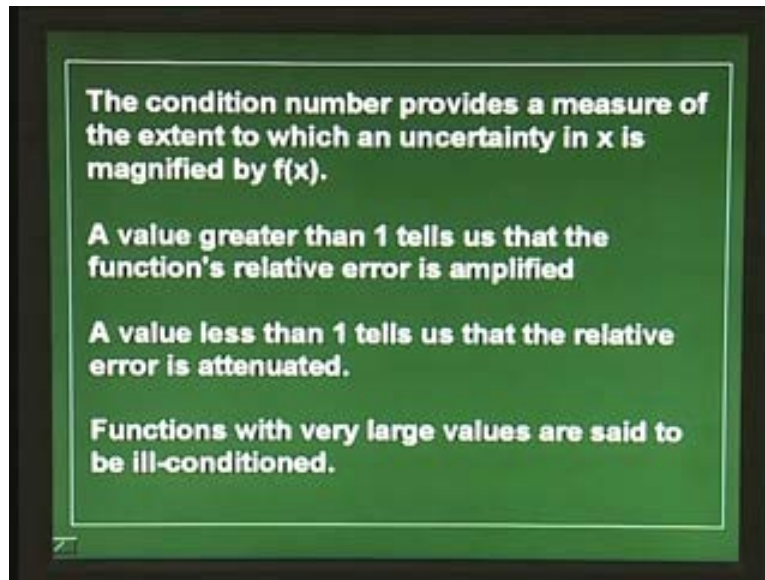
Any step, any program which you run, and you find that you are getting stability problems, the first thing to look at is the condition number for each of the steps involved, and see how sensitive, how bad it is. What do we mean by bad? I said this is the ratio of this relative error in f to a relative error in x . So if the ratio is greater than 1 or close to 1, then you know that this is bad. If it is less than 1 then it is good. That is what we would see. Let us look at a few examples and see how this is calculated.

(Refer Slide Time: 17:30)



It is summarized here. The condition number provides a measure of the extent to which an uncertainty in x is magnified in the function. So a value greater than 1 would tell us that the function's relative error is amplified, and the value less than 1 tells us that relative error is attenuated. Functions with very large values are said to be ill conditioned. So this is the definition. What do we mean by ill conditioned? That if you are asked to calculate the condition and classify operations involved in a program as well conditioned and ill conditioned then, this is what you would calculate. You would simply calculate $x \tilde{f}'(x)$ divided by $f(x)$. So here is an example.

(Refer Slide Time: 18:22)



We want to compute square root of x , so we have the function x as square root of x , and now we have an approximation for x , and the question is, how does it get amplified, or how does it reflect on the function root of x square root of x ? So what is the derivative of f of x here, it is $1/2\sqrt{x}$. So what was the condition? The condition is f' of x into x by f of x . Again, note we are only interested in the absolute value of that number. So that will be $1/2\sqrt{x}$ divided \sqrt{x} into x .

So we would get that as $1/2$. So we have f' of x into x divided by f of x . f' of x is $1/2\sqrt{x}$, and f of x is \sqrt{x} . That is $x/2$. So for values close to 1 we would get that as, this is $1/2$. So if you have values for any value of x , we have this error condition as $1/2$, which is less than 1, and hence, we would say that it is pretty well conditioned. So it is not bad. So any approximation, you make in x . So you do not have to worry, when you compute square root of x because it is always less than 1, the condition is less than 1. So it is well conditioned. So let us look at another problem.

(Refer Slide Time: 20:02)

Example-2

$$f(x) = \frac{10}{1-x^2}$$

The condition gives that

$$\left| \frac{f'(x)x}{f(x)} \right| = \left| \frac{[20x/(1-x^2)^2]x}{10/(1-x^2)} \right| = \frac{2x^2}{|1-x^2|}$$

And this number can be very large for x near 1 or -1.
This function is quite ill-conditioned.

So, here I have a function f of x which is 10 by 1 minus x square. Again, we have to compute the derivative multiplied by x and divided by the function itself, and then that would give us $20x$ by 1 minus x square, the whole square divided by 10 by 1 minus x square. That is a function, the derivative, and the variable x multiplied. So that gives us $2x$ square divided by 1 minus x square. So now, you see this again, mod of this function. So it is $2x$ square divided by mod of 1 minus x square. This particular function, it looks like the condition is function of x , the condition itself is a function of x , so it depends on what the x value is. For example, x value close to 0, this seems to be pretty good, because the denominator will be close to 1, and this will be the numerator, will be close to 0. So the condition number will be very small.

That is good, well-conditioned, but when you have x going close to 1 so, you have the denominator going almost close to 0, and then you have this condition number, very large, and that is a very bad thing. So that means this is quite ill conditioned for variable values close to 1. Again, look at x larger than 1 with x very large, much larger than 1. Then this denominator again goes to, because it is the mod we are interested in. So, it again goes to very large value. So the condition would not be too bad. We can actually say that it is still not very good, but it is not too bad. We could say x very large. We could say that this can be approximated as x square itself, and then it will just 2, it is still larger than 1.

But for x going close to 1, this will be extremely large. So this is very ill conditioned, and we should avoid steps like this in a problem which is sensitive, that is, which tends to be unstable. It is sensitive, if a problem down, in a program, the mathematical operations down this particular operation is sensitive, to what goes into that, then this is a bad thing to do. That is what this condition tells us.

So the condition is a quantity which we should compute whenever we want to know, what would be the crucial step which would determine the error of the program, total error in the program. We want to really analyze a program and see what is the crucial error-making steps? Then, we should compute the condition of each of these steps. So that is what we have done here. We have an operation like this then, it is sensitive to the value of x.

(Refer Slide Time: 23:04)

Example:

The condition of the function $f(x) = 2(\sqrt{x+1} - \sqrt{x})$ for large x such that $x = 10^4$ is

$$\left| \frac{f'(x)x}{f(x)} \right| = \frac{x \left(\frac{1}{\sqrt{x+1}} - \frac{1}{\sqrt{x}} \right)}{\sqrt{x+1} - \sqrt{x}} = \frac{x}{\sqrt{x}\sqrt{x+1}} \approx 1$$

the condition number is not large!!

$f(12682) = 2(112.618 - 112.614) = 0.008$
 The correct answer is $f(12682) = 0.00887968$
 The error we have here is 10%!.

Here is another example. The condition of the function f of x is equal to, we just actually look at this, the condition of the function f of x is equal to 2 root of x, plus minus root of x. Let us look at this function. So for large value of x this function, let us look at the condition. This f prime of x into x divided by f prime of x, that is, x into 1 minus, 1 by root of x, plus 1 minus 1 by root of x, divided by root of x, plus 1 minus root of x. So, that is x by root x, into root x, plus 1. So we can say that this is again of the order 1 if x is large, and so the condition number is not really large. So this is a kind of counter example which I wanted to give you. If x is very large, I could say that x plus 1 is close to x. So, and then I have root x into root x, and that is close to x and x by x is 1.

So condition number is not very large. It is close to 1, but it is not large. So we should think it should be okay. So if you try to compute this thing using a number like 12682, I just took some number, 12682, the function, and I evaluate this function, and I keep some 3 decimal places after that thing. So I have total number, I have 1, 2, 3, 4, 5, 6 digits. So if I keep it 6 digits, and if I compute this, I will get “.008” as the answer. This is kind of rounded to three digits after the decimal point. 6 digits I get here, and then I have rounded to three digits after the decimal point. I will get “.008”.

That is the number. The correct answer turns out to be “.00887968”. That is the error we have here, is about 10%, correct, even though the condition is not too bad. So that is what we would see. So one way, the best way to compute this is not evaluating it as root of x

plus 1 minus root x. That is what it means because, even though the condition is good, you have a large error. This is a counter example we will say x is, the condition is close to 1. So it is not too larger than 1. So it is okay. That is what we would think normally, but here is a case where that is not actually very much true.

(Refer Slide Time: 26:01)

We analyze the computational process to understand this error. It consists of the following computational steps:

$$x_0 = 12682 \quad x_1 = x_0 + 1$$
$$x_2 = \sqrt{x_1} \quad x_3 = \sqrt{x_0}$$
$$x_4 = x_2 - x_3$$

Now we calculate the error propagated to the fourth step from the previous one. We define a function g , which defines the dependence of the final answer x_4 on x_3 .

So then, what is the reason for this? So let us try to analyze this thing. So we have different steps. I have also a program here. We can just look at that program. That is the program. So we have this program.

(Refer Slide Time: 26:18)

```
main()
{
  int i;
  float y,x,y1,x1,f;
  x=12682.0;
  x1=sqrt(x);
  y=x+1.0;
  y1=sqrt(y);
  f=2^(y1-x1);
  printf("%f %f %f %f \n", x,x1,y,y1);
  printf("%f %f %f \n", x,f,2.0/(pow(1
2683.0,.5)+pow(12682.0,.5)));
}
```

10,12 Bot

So what we have, we have a function variable x , and then these are the steps involved in the program. So we have x_1 , I call variable x_1 , I have defined all this. This is only a main program, no subroutines, no functions. So I have defined these variables using floating point. That is floating point variables y , x , y_1 , x_1 and f and I have this number as "12682.0". That is the floating number. What do I want to compute? I want to compute root of x plus 1 minus root x into 2. That is what we want to compute.

So what are the operations involved? First, I had to compute the root of x , and then I have to compute x plus 1, and then take the root of that. I call this y , and I call square root of that as y_1 , and then I have to take the difference. So, there are the three four steps involved, and then I am just printing out that. So what is the value of that? And you see that what you get if I run this. You would get, so if I run this I would get it as the following. I get, so I am just printing out the number, its root, the number plus 1 and its root, and then we take the difference between these two, and that's the answer.

That's the answer which you get. The problem is that when you approximate it here we can see that. If I do this on this machine, and I already get it as ".0088881" is the difference. I am computing it in two different ways. Both give me roughly the same answer. I just compute this, as you can see, using two different ways. So, one is using a function called power, and the other one is just using the function called square root, and you can see that there is already a difference in these two. This is ".8881" and this is ".8880".

So there is already a difference in these two methods, which if you are interested in very small numbers is not very good, because we have an the error of in the sixth decimal place already by just using two different functions to compute exactly the same thing. So you can see if I use a 3 digit or 6 digit number here, I am going to get a large error.

(Refer Slide Time: 29:36)

We analyze the computational process to understand this error. It consists of the following computational steps:

$$x_0 = 12682 \quad x_1 = x_0 + 1$$
$$x_2 = \sqrt{x_1} \quad x_3 = \sqrt{x_0}$$
$$x_4 = x_2 - x_3$$

Now we calculate the error propagated to the fourth step from the previous one. We define a function g , which defines the dependence of the final answer x_4 on x_3 .

So that is because I am assuming this “1.1126188” by “112628” and “112614388” as “1121614”. So I am just chopping off these numbers here. So that is error introduced in that the error is as bad as 10%. That is the point which we are trying to see. Even though the condition is pretty good, that is what the programmers do. Now the question is what? So we again summarize that the number of steps involved I have the function, the variable, the variable plus 1, the square root of the variable, the square root of x_0 plus 1, and both these functions, and then the difference between these two. So now, what we are going to do is we are trying to look at something called an error propagation. That is what we wanted to look at now.

So we calculate the error that is propagated in this step. Let us, for example, let us take the fourth step, that is this one. Name them as 0,1,2,3,4 here just to tell you what are the steps involved. I call this the 0th step, the first step, the second step, the third step, and the fourth step. So the question I want to know, I want to answer, is what is the error propagated to the fourth step, because of an error I made in the third step? Let us look at that. We will try to see how to compute this.

We know how to define condition. We know if it is, we just saw sometimes back how do we define the condition. The condition is defined as the variable x into f prime of variable divided by f for a particular function f of x . So we will just use the same thing here. So what we are going to do is compute error propagation. What we do is we take this fourth step as our function. So we will take this as my function, x_4 , the fourth step is my function, and what is the variable here?

(Refer Slide Time: 31:07)

We analyze the computational process to understand this error. It consists of the following computational steps:

$$x_0 = 12682 \quad x_1 = x_0 + 1$$

$$x_2 = \sqrt{x_1} \quad x_3 = \sqrt{x_0}$$

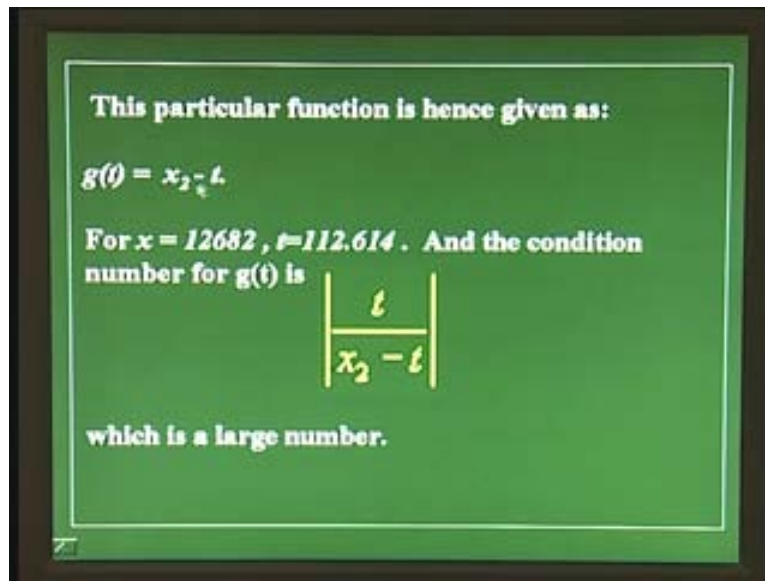
$$x_4 = x_2 - x_3$$

Now we calculate the error propagated to the fourth step from the previous one.
We define a function g , which defines the dependence of the final answer x_4 on x_3 .

The input to the variable is x_3 . The third step answer is the input to my fourth step. It is clear. There are four steps involved in this calculation, and I am trying to calculate the error propagated from one step to another, and what I want to compute the error propagated to the fourth step, or the sensitivity of the fourth step because of the error in

the third step, that is what error propagation which I would say. So I will take this as my input variable. I will call as t something like that, and this is my function. So I will have function, some number minus a variable, and I locate the sensitivity of this or the condition of this, with respect to that variable. That is what we would look at. So we are going to call that as some x_2 minus t .

(Refer Slide Time: 32:01)



I would define this function as g of t x_2 minus t . So remember this, I am going to call t . That is my variable coming in from the previous step, and this is my function. So that is what I am going to call, and I am going to compute the sensitivity of this function. So what is the f prime of this function with respect to t , because I want to look at the sensitivity or the stability, the condition with respect to t , so the variable is “ t ” here, and we want to look at the condition of that with respect to t , and that derivative of g of t g prime would be just one mod of that we are interested, and then we want to multiply it by the function, the variable, that is t . So remember, that is g prime of t into t divided by g of t , g prime of t is 1, and minus 1 g of into t mod of that is t and divided by x_2 minus t . That is what we want to look at.

So now, the point is, if you look at the previous step, these, the differences between these two are very small. So that is the point here. So with x_2 and x_3 , the difference is very small. So what we see here is the difference is small. So this particular step is bad when x_2 and t are close to each other, this particular step is bad, hence, we are getting large error possible. So let me summarize this again. What are we trying to say here? I am trying to say that, if you look at the function f of x itself as one function, and look at its condition with respect to x , we find that this is pretty okay. We find that for x large, this should definitely be not too bad. But we write that down into a series of mathematical operations and look at the condition of a particular step which is the last step, and then we see there it is bad.

(Refer Slide Time: 33:52)

Example:

The condition of the function $f(x) = 2(\sqrt{x+1} - \sqrt{x})$
for large x such that $x = 10^4$ is

$$\left| \frac{f'(x)x}{f(x)} \right| = \frac{x \left(\frac{1}{\sqrt{x+1}} - \frac{1}{\sqrt{x}} \right)}{\sqrt{x+1} - \sqrt{x}} = \frac{x}{\sqrt{x}\sqrt{x+1}} \approx 1$$

the condition number is not large!!

$f(12682) = 2(112.618 - 112.614) = 0.008$
The correct answer is $f(12682) = 0.00887968$
The error we have here is 10%!.

So this is what this actually tells you, that what is important is not just the function, we have to actually see how that is implemented in a program. What are the series of operations which are carried out, and then look at each of them, each of those series of operations. So I would, if you give me this kind of a function, then I would, what I would do is actually multiply and divide this function by root x plus 1 by root minus root x , then I would get, and the denominator, it has root x plus 1, plus root x , and the numerator it is just 1. I will multiply and divide this function by root x plus 1 minus root x . It should not change the value of the function, and at the same time, you would see that the condition is completely different. So that is something we should try to see.

So this is, so here is an example of a function which has good condition with respect to x , but on the way, somewhere, it has a problem in this step. So we should try to avoid that step, that is, there is subtraction between those two numbers which are close to each other, and that is the one which is causing trouble. So we have to avoid that. One way to do that is to multiply and divide that function by root x plus 1 minus root x .

To summarize, we saw that there is a problem with this computation, that is, it is sensitive to the number of digits you have k and then the reason for that also we saw is because of this particular step in which we actually subtract root of x plus 1 root of x from root of x plus 1 when x is a large number, and this step introduces large error because of the loss of the significant digits. We also saw that by looking at the condition number, that is, the condition number was for that particular step was t by x_2 minus t .

So, when t and x_2 are close enough, that is when they are large, but they are similar numbers. In this case, x_2 was root of x plus 1, and t was root of x , so they were very close, similar numbers, because x itself is a large number. In that case, this becomes a very large number, and that is the reason we saw is that this is a large number of large

amount of error. So one, this kind of analysis also helps us to correct this kind of steps in the program.

(Refer Slide Time: 36:38)

If we multiply and divide $f(x)$ by $\sqrt{x+1} + \sqrt{x}$ we get,

$$f(x) = \frac{1}{\sqrt{x+1} + \sqrt{x}}$$

Following the earlier analysis

$$x_0 = 12682 \quad x_1 = x_0 + 1$$

$$x_2 = \sqrt{x_1} \quad x_3 = \sqrt{x_0}$$

$$x_4 = 1/(x_2 + x_3)$$

We find the error propagation (condition number) from step 3 to step 4 as $x_3 \left| \frac{1}{(x_2 + x_3)} \right|$

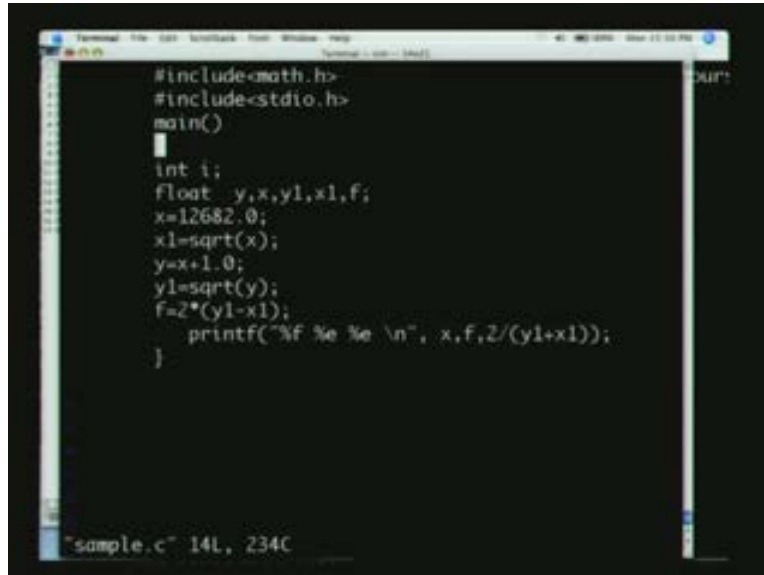
Which is much smaller than $x_3 \left| \frac{1}{(x_2 - x_3)} \right|$

For example, we could do multiply and divide this f of x which was root of x plus 1 minus root of x by root of x plus 1 plus root x . That would not change the function value, but the way the function is written will now be instead of root of x plus 1 minus root x , it will be 1 by root of x plus 1 plus root x .

So, for the earlier analysis, what we did for this now, we will follow the same thing, just the same analysis to see how the condition number of this particular function x . So we had x equal to x_0 as 12682 as a large number, and then we had to compute root of x plus 1 root of x_1 as x_0 plus 1, and then we had root of x_1 , and then we had another step which is root of x_0 , and then we subtracted these 2. That is what we had done earlier. Root of x 1 minus root of x_0 as what we did earlier. So instead of that now we will do 1 by x_2 plus x_3 . So we had x_2 minus x_3 , and we did the analysis with respect to x_3 . Now, we do this here, 1 by x_2 plus x_3 , and then we look at the error propagation or the condition number for this step from 3 to 4.

So then, we write the condition number, in this case, as x_3 multiplied by f prime, that is the function, the derivative of this function, with respect to x_3 divided by the function, and that will give us, now if you use this function instead of root of x plus 1 minus root x , we use this function, then that will, the condition number will turn out to be x_3 mod of 1 by x_2 plus x_3 . So which is much smaller than x_3 into 1 by x_2 minus x_3 , which is what we got earlier. So we wrote, we use this x_3 as t in that case. So this is much smaller than this when x_2 and x_3 are large, and they are close to each other, the values, and then this number, would be very small compared to what this number is, which is very large, which is x_3 divided by 1 by x_2 minus x_3 , which is a large number. You can also see this in a program. Now I will show you this in a program which will actually reflect this thing.

(Refer Slide Time: 39:06)



```
#include <math.h>
#include <stdio.h>
main()
{
    int i;
    float y,x,y1,x1,f;
    x=12682.0;
    x1=sqrt(x);
    y=x+1.0;
    y1=sqrt(y);
    f=2*(y1-x1);
    printf("%f %e %e %e \n", x,f,2/(y1+x1));
}
```

Here is the program which actually computes this. “x” is now defined as 12682, and we are taking the square root of x. So that is, and then we have y here as x plus 1, and we have taken a square root of y, so you have square root of y, y_1 minus x_1 is also square root of x_1 too. The function was 2 star square root of x plus 1 minus square root of x, and now that is given here by this particular function, f. So when I print that function, f, and then the other way of doing that is, as we just saw, to multiply and divide it by y_1 plus x_1 .

So that would give us y_1 , 2 by y_1 plus x_1 , that is, 2 by root of x plus 1 plus root of x. That is just this function. So that is just this function here. So we do this two different ways of thing, and we will just this program and print out. Now we are using all variables are declared as float, that is single precision variable, and we look at the value of f, and this 2 by y_1 plus x_1 and see what we get.

So when we do that so, so we see that the value which we get, so I am using this e format, so when we write, so remember what we are doing is we are writing out f which is y root of x plus 1 minus x_1 root of x, and this 2, 1 by root of x plus 1 plus root of x. So, that is, the two values which you are printing out here. So one value gives us “.8806” 10 power of minus 3 and the other gives you “.879685” 10 power minus 3, and this is closer to the, this one is closer to the correct answer than this one, which is now off by, off in the third decimal digit here.

So now, we can also, I should also like to point that, this comes because of loss of significant digits. So we can also improve this by keeping more digits in our computation. That is, instead of using float let us use double, and then let us use double and then do the same computation. Now we see that both these methods give the same answer. This is also an evidence that what the error what we saw earlier, that is, in this particular case, this difference between these two numbers here which is the same function computed in two different ways, it actually comes from the loss of significant digits.

(Refer Slide Time: 42:05)

Operation		Estimated Error
Addition	$\Delta(\tilde{u} + \tilde{v})$	$\Delta\tilde{u} + \Delta\tilde{v}$
Subtraction	$\Delta(\tilde{u} - \tilde{v})$	$\Delta\tilde{u} + \Delta\tilde{v}$
Multiplication	$\Delta(\tilde{u} \times \tilde{v})$	$ \tilde{u} \Delta\tilde{v} + \tilde{v} \Delta\tilde{u}$
Division	$\Delta\left(\frac{\tilde{u}}{\tilde{v}}\right)$	$\frac{ \tilde{u} \Delta\tilde{v} + \tilde{v} \Delta\tilde{u}}{ \tilde{v} ^2}$

So that is what we see. So now, so let us just summarize how do we get the total error in different operation which is important. We just saw that each operation is important. So if you have error in a summation, then we have error adding up in these two here. Subtraction again, the error adding up in these two again.

(Refer Slide Time: 43:06)

Problem:

Calculate $f(x) = e^x$

at $x=12$ using Taylor series.

Compare your results using the "correct" value.

Explain the result using the condition of some of the steps involved in the computation.

Remember this, we are not interested in the in between sign, we are looking at the total error. The total error is the sum of these two and the multiplication, it depends on what the function itself is, so value of that variable itself is, that is what is important. And in the division, you would have again similar one. So we have the value of the variable itself

is important to find out the total error in the function, so now, we could actually look at as a problem, look at the function f of x as e to the power of x .

There are two things we have to look at. One is the previous step which we saw. That is, how do we compute this differently such that the error can be slightly minimized or changed, or how do we avoid this particular step of subtracting each other. And then another thing which we would look at, how do we compute a function f of x as e to the power of x using a Taylor series, and how do we estimate the errors in each of these steps. So these two things which you now look at, we saw two different functions which are ill conditioned. So we saw one function which is f of x . We said 2 times root of x plus 1 minus root of x .

So that is one function, which we saw and which we looked at, and then we found that it has, so we said the steps which involve in one of the steps, that is, the steps which involve taking the difference between this is ill conditioned, even though the function itself is conditioned by, with respect to x . So, one way, how do we compute this in a better way, so avoid that step. So remember the problem was in the last step which we wrote it as we said, x_4 , that is we said x_2 minus x_3 , and we had said x_2 is root of x plus 1 and x_3 as x to the power of 1 by 2. So we had these 3, 4 steps. So this is the one which I had problem with, the step, to avoid this particular step.

So how do we do that? We can do it in the following way, that we would say that this function f of x . Now I would write it as 2 times x plus 1 to the power 1 by 2 minus x to the power 1 by 2 divided by x plus 1 to the power 1 by 2 plus x to the power 1 by 2 multiplied by x plus 1 to the power 1 by 2 plus x to the power 1 by 2. So I could do it this way. So then, I have it as 2 times x plus 1 minus x divided by x plus 1 to the power 1 by 2 plus x to the power 1 by 2. So that I have it as 2 divided by x plus 1 to the power 1 by 2 plus x to the power 1 by 2. So now you can see that I am doing the same computation, but I have avoided taking the difference between these two numbers which was the problem when x is large.

So when taking the difference between the two numbers which are large, and I want to take now, the difference is small. So taking difference between two numbers which are large by itself is not a problem. The problem is when the numbers are large and they are close to each other. Then you lose precision. That was the problem. One way to do that is to just avoid it taking the difference, and we can do it by this way. So small changes like this in the step would change the reliability of the program considerably. So one could now compute what is the, what would be the stability, or what is the condition of this function itself. So that is what we can look at. So now, the function now changed to f of x is equal to 2 divided by x plus 1 to the power 1 by 2 plus x to the power 1 by 2.

How do we compute the condition? The condition is, we just saw, the condition would be the condition number is f prime of x into x divided by f of x mod. So f prime of x we can compute here. So f prime of x here is 2 divided by x to the power 1 by 2 square, and then I take the derivative of that function, and then we would get is as, we are not interested in

the signs, we are only interested in the total value of the function. So we have that. So we have the value evaluated.

(Refer Slide Time: 48:01)

$$f(x) = \frac{2}{(x+1)^2 + x^2}$$

$$\text{Condition} = \left| \frac{f'(x) \cdot x}{f(x)} \right|$$

$$f'(x) = \frac{2 \cdot \frac{1}{2} (x+1)^{-3/2} + \frac{1}{2} x^{-3/2}}{((x+1)^2 + x^2)^2}$$

And then you have to find this condition, and you could evaluate that at x equal to, we said x equal to 12682, and then we could, what the condition is, that is one problem which we discussed. So you would see the condition of this, and compare that with the condition of this function, the condition of this function compared with the condition of this function, which go exactly the same. That would be interesting stuff to look at. Another thing which we, another problem which we discussed, which also is to be carried out, is to look at the function f of x which is e to the power of x . f of x is equal to e to the power of x , and then use the Taylor series for this, for large x . So f of x is equal to e to the power of x , and you want to compute this at x equal to 12 which is large profile.

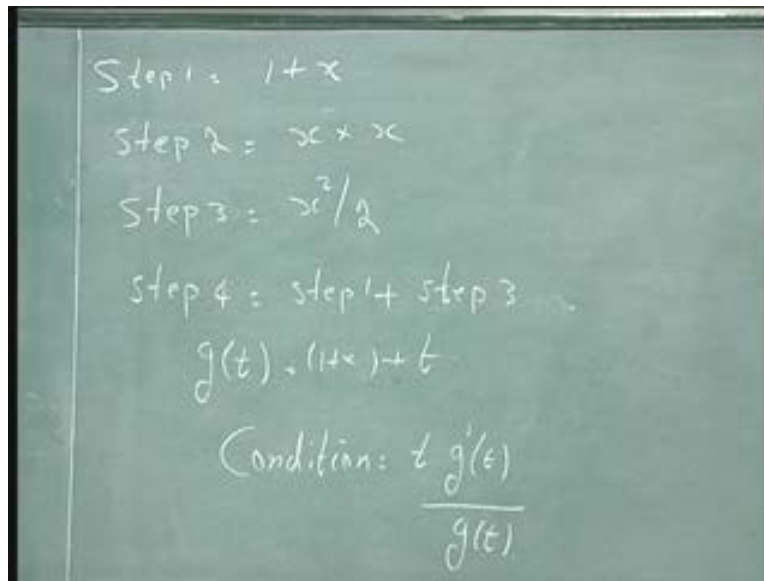
So to compute e to the power of x for x large value, that is something which we want to look at. So we could write f of x as $1 + x + x^2$ by 2 factorial, etcetera, and then compute this for the large x value, and then you could see what the error which you would develop is. Remember, you could also compute this function f of x by 1 by e to the power of minus x . We have seen it in two classes, that these two functions would have a large difference. So you could compute this as e to the power of x , and 1 by e to the power of minus x , and then look at the condition of the function and the stability of each of these, or the error propagation in each of these steps. Remember, just to remind you what I mean by error propagation, is this particular thing I can write it as many steps.

I would say step 1, 1 would be to actually write $1 + x$. so that is step 1. And then step 2 would be to write it as $1 + x + x^2$ by 2 , or you could introduce another step here actually. So you could say that step 2 would be to actually compute x^2 , that is, x into x . And then, step 3 would be to actually to look at x^2 by 2 . And step 4 would

be to look at step 1 plus step 3. So you could write it down in a very detailed series of operations like this during each of these operations.

And then look at the stability of each of these. So now, if I want to look at the stability of the error propagation from step 3 to step 4, then I would write this as a function of t , and then I would, substitute this value here, that is step one that is, $1 + x$ some value there, and then I write the step 3 function as t there. That is what we saw, and now this would mean that in the step 4. I am taking this as my variable, and I am looking at the condition of this step with respect to this variable. Step 3, so step 3 I have taken as " t " here, and then I would write this function as g of t $1 + x$ plus t , and then I will look at the condition number, condition as t into g prime of t , that is, derivative, that is derivative of g with respect to t divided by g of t .

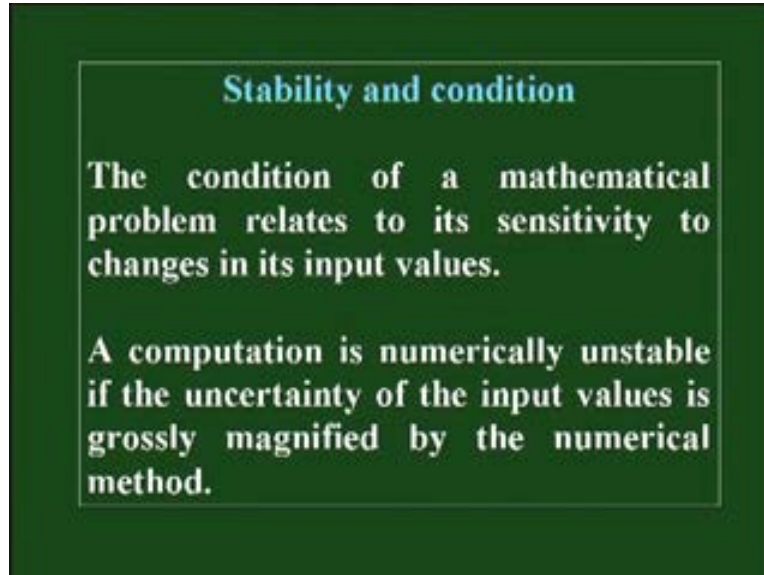
(Refer Slide Time: 51:59)



Step 1 = $1 + x$
Step 2 = $x \times x$
Step 3 = $x^2 / 2$
Step 4 = step 1 + step 3
 $g(t) = (1+x) + t$
Condition: $t \frac{g'(t)}{g(t)}$

So you could do this for different steps for this function as well as this function and see what the difference is which you get is. So far, we have seen a programming methods and techniques, and also some of the points at which errors can creep into a program, particularly into a C program. So now, we look at some of the methods that are commonly used in scientific and engineering computation.

(Refer Slide Time: 52:10)

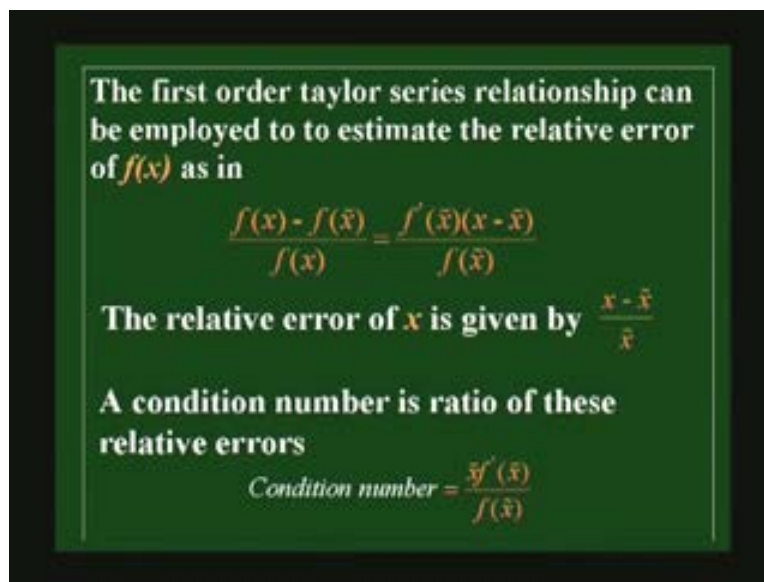


Stability and condition

The condition of a mathematical problem relates to its sensitivity to changes in its input values.

A computation is numerically unstable if the uncertainty of the input values is grossly magnified by the numerical method.

(Refer Slide Time: 52:29)



The first order Taylor series relationship can be employed to estimate the relative error of $f(x)$ as in

$$\frac{f(x) - f(\bar{x})}{f(x)} = \frac{f'(\bar{x})(x - \bar{x})}{f(\bar{x})}$$

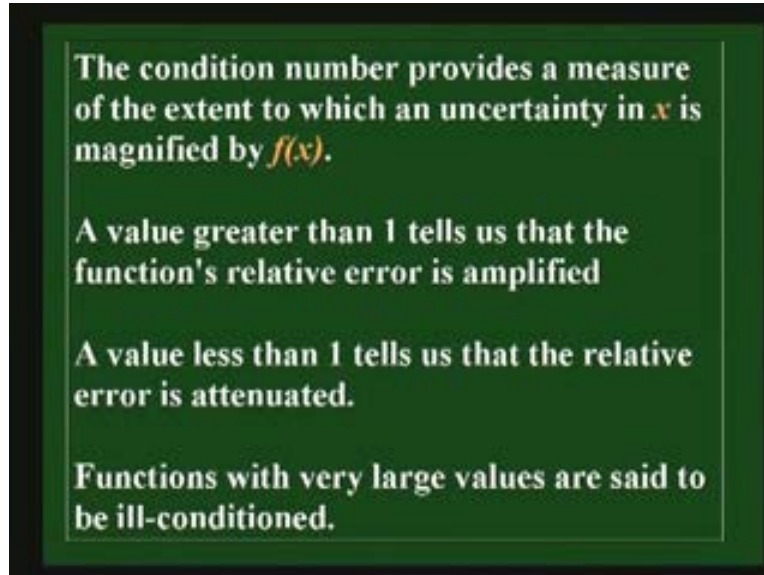
The relative error of x is given by $\frac{x - \bar{x}}{\bar{x}}$

A condition number is ratio of these relative errors

$$\text{Condition number} = \frac{\bar{x} f'(\bar{x})}{f(\bar{x})}$$

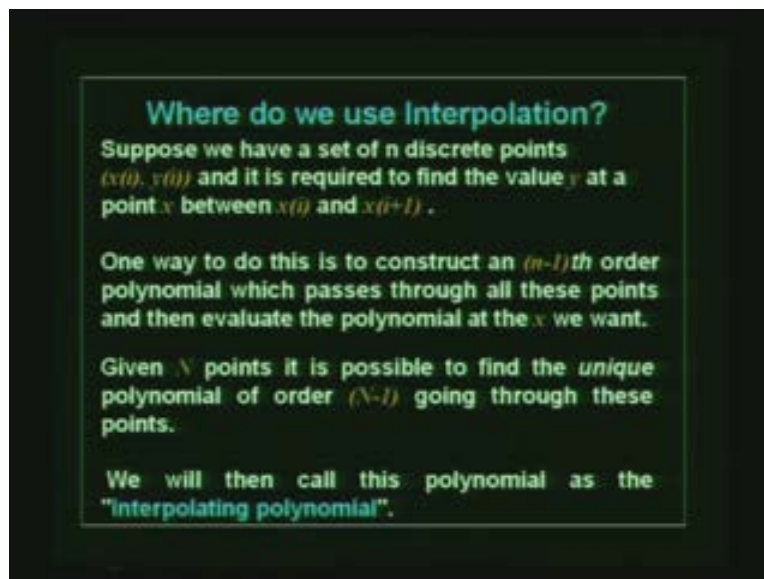
One of them is the interpolation of a given set of data. So the question comes, where do we use interpolation? That is, suppose we have a set of discrete points, x_i and y_i , which are acquired, let us say, from an experiment or from a discrete numerical simulation.

(Refer Slide Time: 52:51)



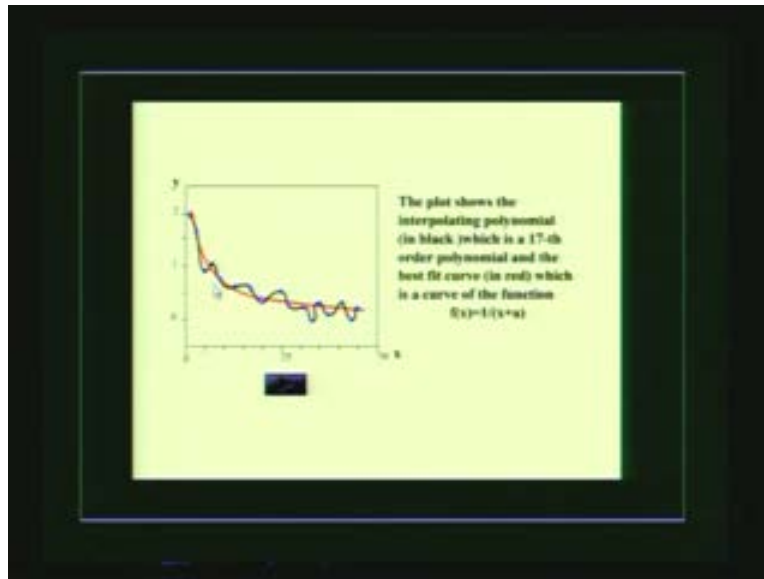
Then in some cases it may be required to find the value of y , that is, this is the dependent function y at any, at a point, which is in between two tabulated points, x of i and x of i plus 1. In this case, we need a function which would faithfully go through all the tabulated points and then, we can use that function to evaluate the value at a desired point x in between any two tabulated points. So that is the basic use of interpolation. You would need that for finding many things, maybe for finding derivatives or many other archives. So one way is to construct, so, such a function is to use a polynomial. So if we have n points, we can construct n minus 1th order polynomial which passes through all these points. And then we can evaluate the polynomial at the point where x we want.

(Refer Slide Time: 55:10)



So now this polynomial is that is the n minus 1th order polynomial which goes through these n points is a unique polynomial even though we can write this polynomial in various different ways. Now this unique polynomial, we will be call an interpolating polynomial. So that is what the next section of the course would look at various ways of constructing the polynomial. Now given this n points x_i and y_i , how do we actually construct a polynomial which goes through all of them? That is what we would, we are looking at in the coming lectures. So before we do that, I just want to demonstrate to you the difference between an interpolating polynomial and a fit.

(Refer Slide Time: 56:01)



Fit is another one, another way of modeling a data which is commonly used. So the difference between here, this plot, is we try to demonstrate the difference between an interpolating polynomial and a fit. So these blue points which appear is the discrete set of values which we have obtained, and, let us say, from an experiment or from some numerical simulation or something of that order, and then you have this red curve which goes through all these points which goes through this point which does not touch all these points which basically goes through this data.

Now that is an example of a fit. So we normally use fit, which we see later in the program, later in this course, that how to construct this fit, but for the time being here, we understand that we need to understand that what a fit means also. So a fit is just, is used when we know how the function is supposed to behave and we want to know how good the behavior, we obtain in an experiment or a simulation compared to the expected value. That is where we use a fit, and a fit is a function which is predetermined, and we just look at how does, how good that function is to model a given data.

Compared to that, a polynomial is something which goes through all the points. So now, we can see that this curve which goes through all these points, now this one is a polynomial that goes through all the points, so an interpolating polynomial and the

function are exactly the same value at the tabulated points. In fact, we use that condition to determine the interpolating polynomial. So that is the big difference between an interpolating polynomial and a fit.

In the case of fit, we actually have a predetermined form, and we are looking at how good that form is to a given set of data, and fit need not go through any of the tabulated points. What it should do, is to reduce the error of, between the tabulated points and the function values at every tabulated x value, while an interpolating polynomial need to go through all those points and it should be exactly the same at the tabulated points. So now we will look at, in the next lecture, how this, such a polynomial can be constructed, and what are the different ways of constructing such a polynomial, etcetera.