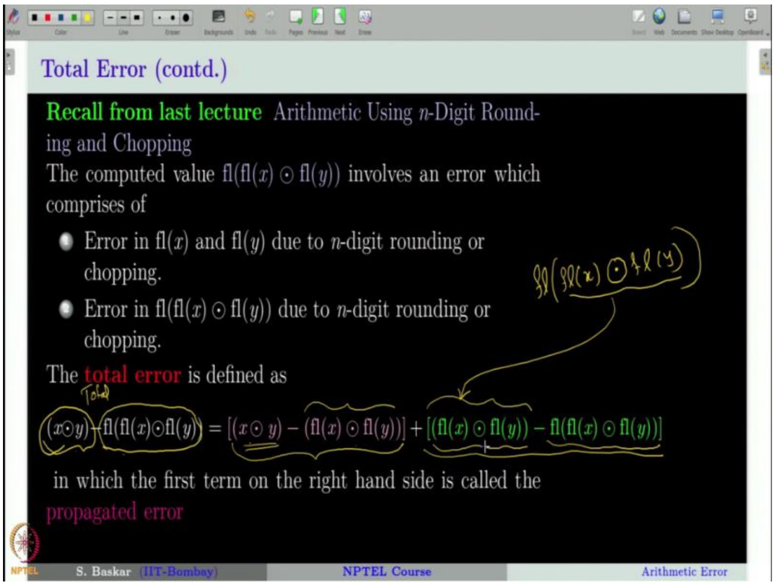
Numerical Analysis
Prof. S. Baskar
Department of Mathematics
Indian Institute of Technology – Bombay

Lecture – 06
Arithmetic Error: Condition Number and Stable Computation

Hi, let us continue our discussion on arithmetic error. So far we have learnt what is meant by floating-point representation and floating-point approximations and we also learnt what is meant by loss of significance and then we have also seen through an example the danger in losing significant digits in our calculation. Further we have also seen that only subtracting two very close positive numbers will lead to loss of significance.

Whereas adding two positive numbers, multiplication and division or relatively safe as far as the amplification of relative error is concerned. In this class we will learn how to judge whether a function is good or bad when we go to evaluate it on a computer using a concept called condition number. Before going into this, let us continue our discussion and understand what is meant by total error involved in a calculation.

(Refer Slide Time: 01:43)



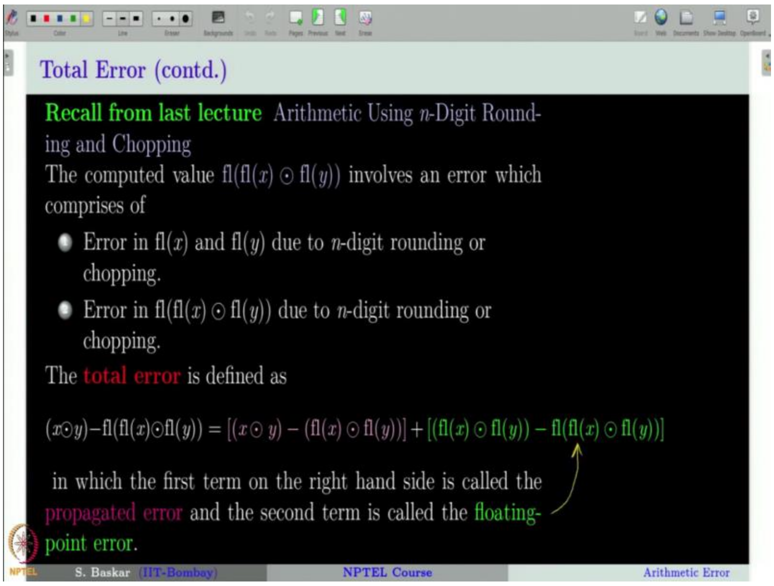
Recall from our previous discussion that arithmetic using n-digit rounding and chopping is done using the steps which leads to the computed value given by this. When we want to compute $x \odot y$ we land up getting this value on a computer. This is what we have understood in the previous discussions. Now the question is what is the total error involved in this calculation.

First thing is the error in fl(x) and fl(y) due to n-digit rounding or chopping and next is after calculating we further do a floating-point approximation one more time. Therefore, there are two levels of approximations involved in this calculation and therefore two levels of errors are involved in this calculation. Therefore, the total error is defined as the value that we want to find this is ultimately what we are interested in and this is what our computer has calculated and given to us.

The difference between these two is called the total error and that can be written as $x \odot y$ which is the true value. Now what we are doing is we are adding and subtracting this intermediate one. So, remember you took fl(x), fl(y) and then first you calculated that and then you went to make the approximation. So, this value is what is added and subtracted here. Now you can see that the true value minus this value is the first level of error.

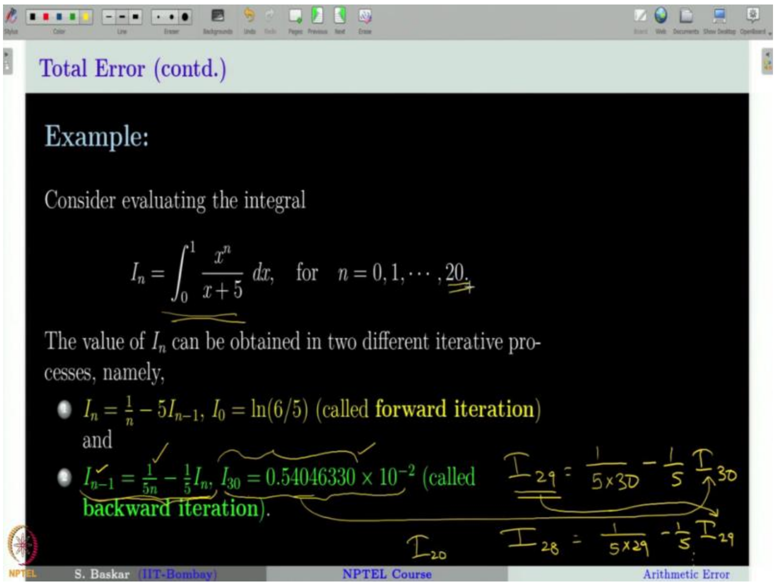
And further in the second level this value is further made an approximation here. Therefore, this is the second level of the error that is involved in it. Adding these two leads to the total error. In this the first part is called the propagation error.

(Refer Slide Time: 04:16)



And the second part of the error is called the floating-point error that is this part.

(Refer Slide Time: 04:31)



Let us have one more example to have a feeling of how these errors propagate from one step to another step. Let us take $\int_0^1 \frac{x^n}{x+5}\,dx$. I want to perform this integral for some n say for instance n = 0, 1, 2 up to 20. There are two ways that you can do it. One is you can use this formula. One can easily derive this formula from this integral I will leave it to you to derive that.

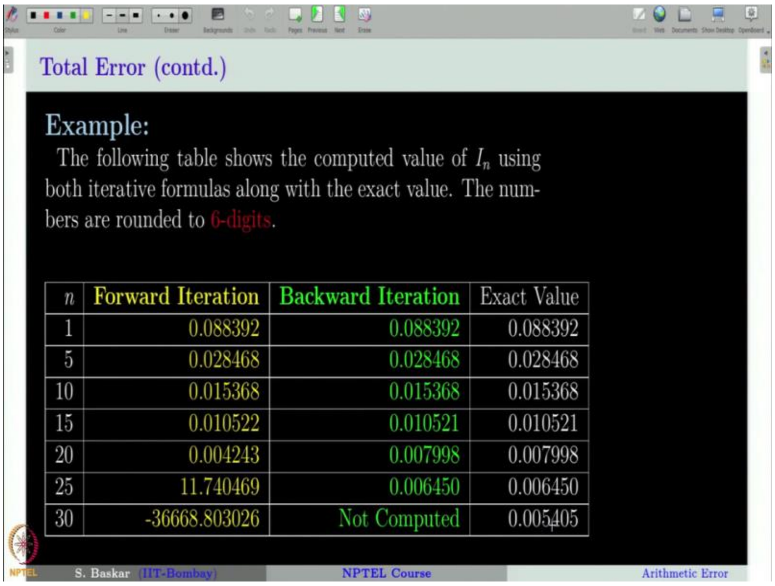
So, what you can do is you start with n = 0, clearly n = 0 will give you a very simple calculation and that can lead to the exact value of I_0. Once you have this then you can substitute I_0 here say you take n = 1 you have I_1 which is equal to $(1-5)I_0$ this one. I_0 is already known to us here. So, you can substitute that to get I_1. Once you have I_1 then to get I_2 you can use this formula again $\frac{1}{2} - 5I_1$ and so on you can go on like this.

So, you start with the exact value of the integral and go on with the iteration and you can reach whatever n that you want to get. So, that is the idea. There is also one more way of getting it. What you do is you start with some $n > 20$ and from there you decrease your n and reach n = 20. In that case you can also use this formula, this can also be derived from this integral it is very simple.

You can see how to derive that and now what we do is we start with some n greater than 20. For instance, I have started with n = 30. In that case we have the integral value which is, I will not say it is exact, but it is pretty good approximation. How I found, well I used Simpson's rule to get it and I will start with this value and you can see how the iteration is defined when you have n = 30.

Then this will be I_{29} which is equal to $\frac{1}{5\times 30}$ that is this first term minus $\frac{1}{5}I_{30}$. We know already what is I_{30} so you plug in this here and you get the value of I_{29}. Once you have this you can go for I_{28} that is equal to $\frac{1}{5\times 29} - \frac{1}{5}I_{29}$ and so on. I_{29} again you know you can plug in that here and you can go on like this. So, in that way you can also reach I_{20} and get the value of I_{20} here.

(Refer Slide Time: 08:11)



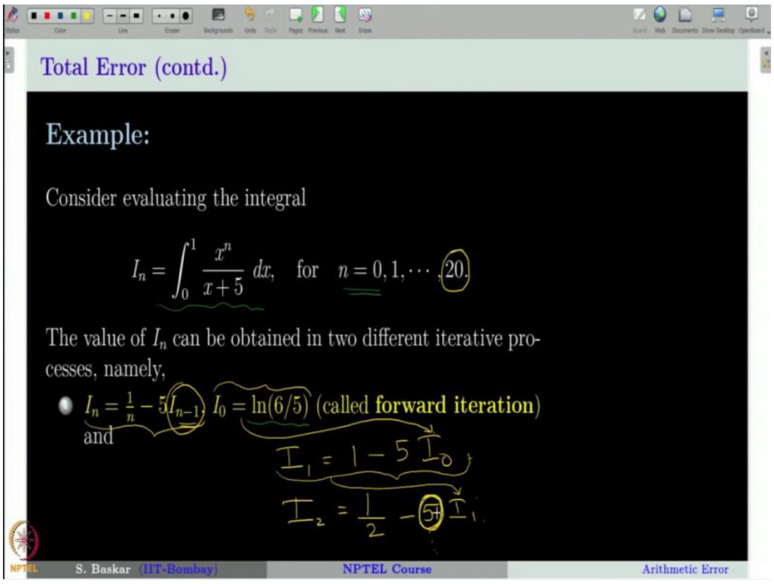
So, I have performed this calculation and I have shown the results in this table. I have used the 6-digit rounding in this calculation. So, from the forward what I am doing is from I_0, I have calculated I_1 and so on and similarly for the backward I started from somewhere I_{30} and came up to I_1 so from the backward direction and I am also showing the exact value. You can see I am just not showing it for 1, 2, 3 and so on I am just jumping from 1 to 5, but to get I_5 I have to calculate I_1, I_2, I_3, I_4 and then only I will get I_5.

I am only not displaying it here and then going forward I have I_{10}. Remember for this I have to come forward and for this I have to go backward from the larger value of *n* to smaller value of *n* I am coming backward. So, you can see that the forward value and the exact value are pretty coinciding for some *n* and after 15 or 20 onwards there is a drastic difference between exact value and the computed value here.

However, you can see that the backward iteration is maintaining the accuracy very well when compared to the exact solution that is what we observe here. You can see that at the 30th iteration, remember I have started with n = 30 for backward and I have gone upward and reached 1 and even when I reached 1 it was pretty good approximation whereas for forward I started with I_0 and went forward in n and reached I_{30}.

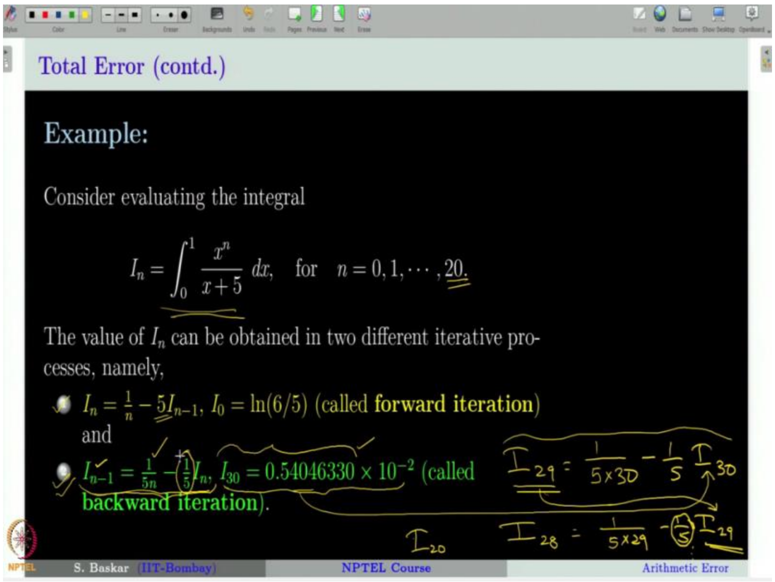
You can see that the value that is computed using our forward formula is no way near to the exact value. Now what went wrong in this forward iteration is the question.

(Refer Slide Time: 10:33)



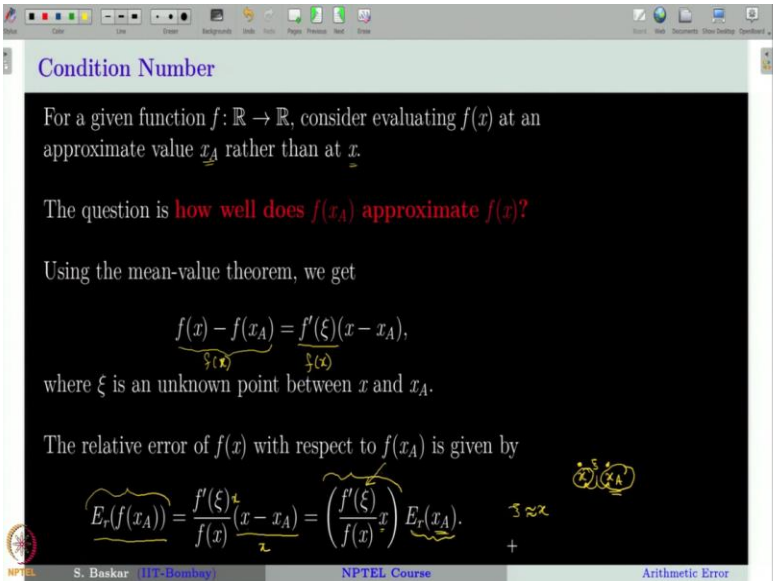
Let us go back to the formula and see what went wrong. Let us take this at every iteration you make a approximation here and that approximated value is plugged in here and therefore this I_1 is the value plus some error and that error is getting amplified 5 times at every step.

(Refer Slide Time: 11:05)



Whereas what is happening in the backward formula is, the error that you committed in I_{29} is when it is substituted here the error is getting divided by 5. So, that is why the propagation of error in the backward formula is rather very slow whereas the propagation of error from one step to the other step in the forward formula is very fast because every time it is multiplied by 5 whereas here it is divided by 5, that is the idea here. So, this shows how the propagation error can go from one step to the other step.

(Refer Slide Time: 11:53)



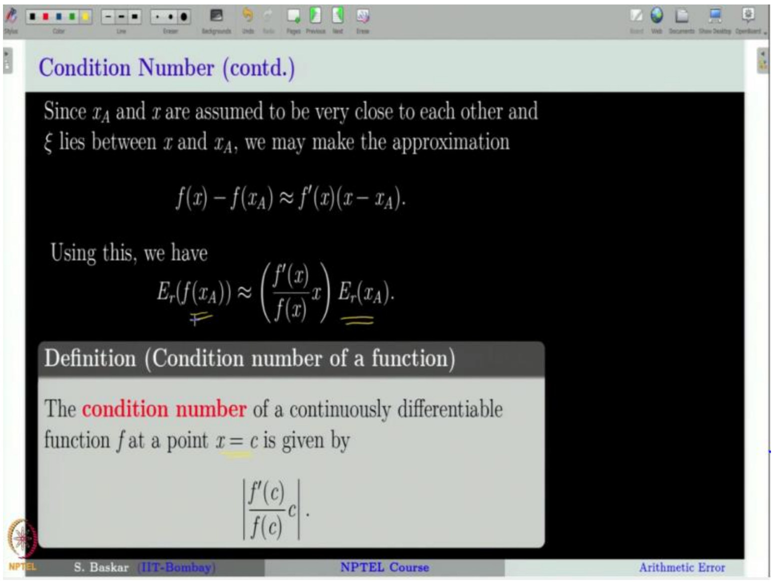
Let us now take the next topic of evaluating a function on a computer. So, how will we judge whether evaluating a function on a computer is good or not, it is based on a concept called condition number of a function. Let us try to understand what is meant by the condition number of a function. Let us take a function $f:\mathbb{R}\to\mathbb{R}$ and we want to find $f(x)$, but instead of x we are only provided with an approximate value say x_A.

Therefore, we are only getting $f(x_A)$, but not $f(x)$. So, let us consider this situation and see how the relative error will get into the function value when compared to the initial error in x_A when compared to x. For that we will use the mean value theorem to write $f(x)-f(x_A)$ this is the error in the function value and that can be written as $f'(\xi)(x-x_A)$ for some unknown ξ lying between x and x_A.

Now from here we want to get the relative error. Therefore, you divide this by $f(x)$ and therefore this side also we have to divide by $f(x)$ and that will give you relative error in $f(x_A)$ when compared to $f(x)$ and that is equal to $\frac{f'(\xi)}{f(x)}(x-x_A)$. Now what you do you also want to make this a relative error. Therefore, divide this also by x then you will have x multiplied here and divide here that gives you $\left(\frac{f'(\xi)}{f(x)}x\right)E_r(x_A)$ when compared to x.

So, this is what we are getting. From here you can clearly see that the relative error in the function value is amplified by this factor of the relative error in x_A when compared to x that is what we are understanding.

(Refer Slide Time: 14:41)



Now instead of having this expression which is not very good for us because this ξ is not known to us. So, what we will do is, we will take ξ approximately equal to x because in numerical analysis we need to have a quantified number rather than something which is not known to us. Here you can see that x and x_A are pretty close to each other that is what is the understanding of having x_A is approximately equal to x.

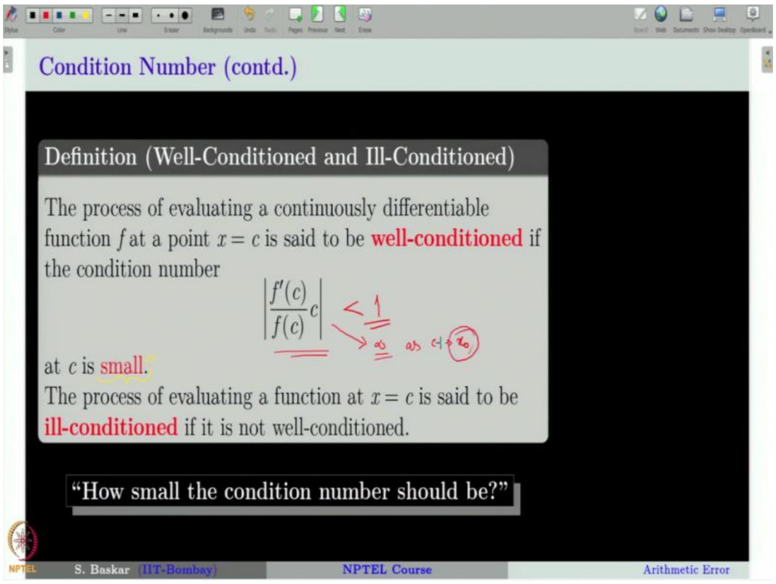
And now ξ is lying between these two numbers. Therefore, it may be a good approximation to take either ξ equal to this or this. Look we do not know what is x_A because that is decided by the computer. When you plug in x the computer takes that x as an approximate value by chopping or rounding or whatever it does. Therefore, this number is not known to us.

Whereas this number is known to us because this is the point where we want to find the function value. Therefore, it is fair to take i approximately equal to ξ and plug in that into your mean value expression this is only approximately equal to, remember it is not exactly equal to, and then based on that you get the expression like this. Now you see this factor is something which is kind of known to us because all this arguments are known to us.

And that is what is called the condition number. Let us define the condition number of a function. You are given a C^1 function, condition number of a C^1 function that is continuously differentiable function at a point c is given by $\left|\frac{f'(c)}{f(c)}c\right|$ because this number will give you an idea of how much the initial error is amplified and got into the function value.

So therefore this number plays an important role in getting a feeling of how good or bad a function is when you go to evaluate it on a computer.

(Refer Slide Time: 17:23)



So, this is what we call as well-conditioned or ill-conditioned. The process of evaluating a continuously differentiable function f at a point c. Remember this is point-wise analysis. You give me a point c now I want to know whether the process of evaluating that function at the point c is good or bad. If the condition number is small then we say that the process of evaluating the function at that point is well-conditioned otherwise it is ill-conditioned.

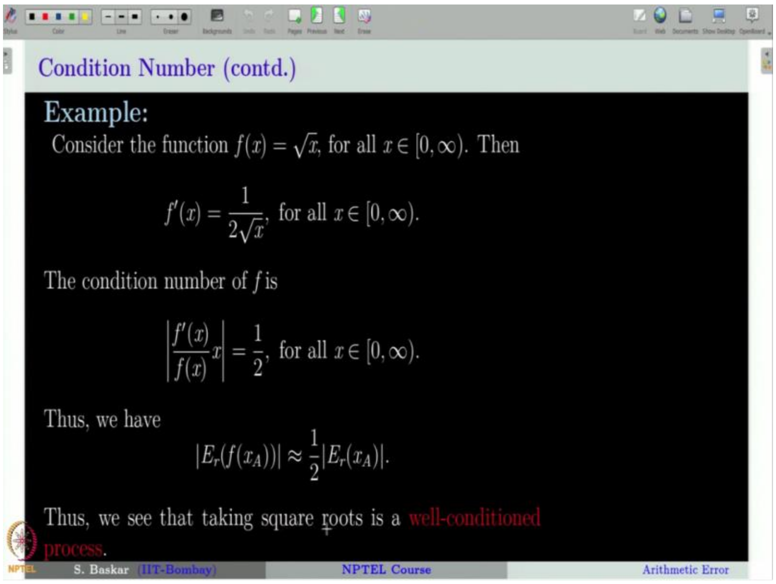
Now this is rather a very vague definition because it does not quantify which is mean by small or big. Well, this is something which you really cannot quantify in practical applications because it depends on two factors. One is, in what application you are working with and second thing is what level of error that you can really afford to tolerate. Suppose, you are working with a very powerful computer it can handle more error.

Whereas if you have a very small computer maybe even a small error can propagate very fast. Similarly, it also depends on what kind of applications you are working with. For instance, we have seen that in missile path finding we cannot afford to have even a very small error like 0.1% or something like that whereas if you are working with some applications like in finance and all even 30%, 40% error maybe good.

Therefore, you really cannot quantify this smallness in practical situations. However, from the academic point of view whenever this number is less than 1 we always say that it is well-conditioned. On the other hand, if this number is tending to infinity as c tends to some number say x_0 then we say that evaluating that function in a neighborhood of x_0 is going to be very bad.

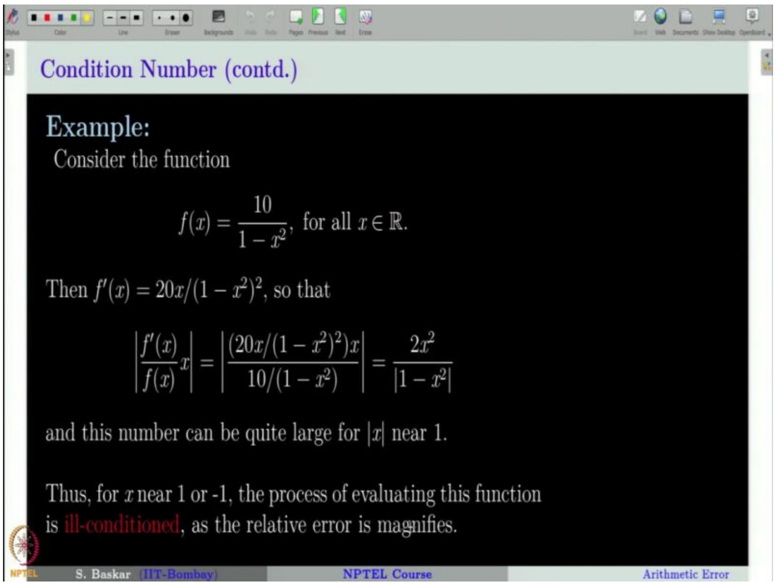
These are something which you can surely tell because if it is less than 1 it means something like what we did in the backward iteration in the previous integration, there the error is getting reduced because of the $\frac{1}{5}$ factor. The similar thing will happen here. Whereas if the condition number is going to infinity it means it is going to amplify the error in the function value drastically when you go to compute the function value very near to the bad point x_0. So, that is the idea.

(Refer Slide Time: 20:22)



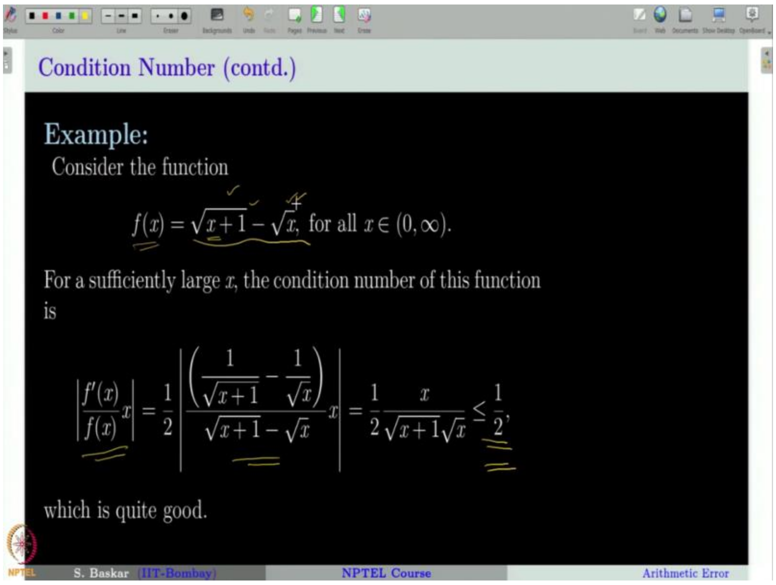
Let us see some examples consider the function $f(x) = \sqrt{x}$, f' is given by this and from there you can immediately find the condition number of the function $f(x) = \sqrt{x}$ and that gives as a feeling that evaluating the square root of a number is going to be very nice on a computer, it is not going to be dangerous at all because the error in the function value is going to be approximately half of the error that you have committed in the argument so that sounds good. Therefore, we will declare that finding square root on a computer is well-conditioned.

(Refer Slide Time: 21:13)



Let us take another example $f(x) = \frac{10}{1-x^2}$ and here you can see that the condition number is given by this and from here you can see that as $x \to 1$ the condition number is blowing up to infinity and that shows that evaluating this function near either 1 or – 1 is going to be ill-conditioned definitely.

(Refer Slide Time: 21:48)

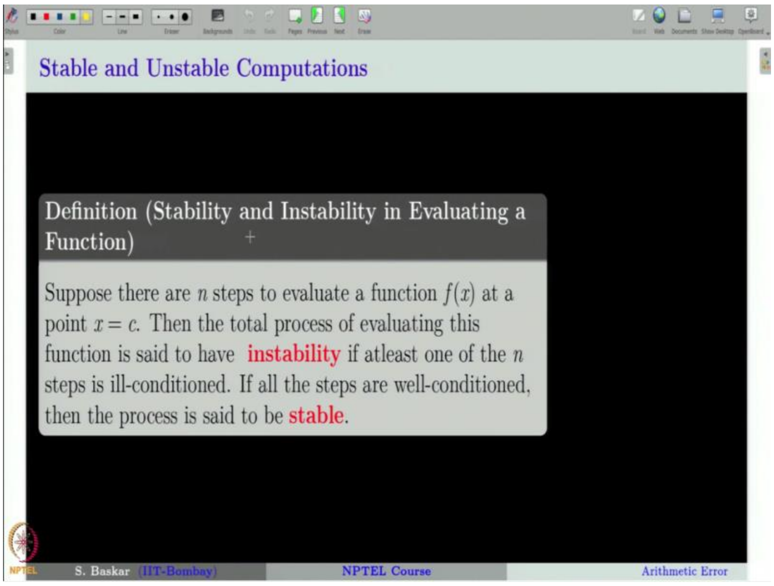


Let us take another example and this is an interesting example. Look this function is not involving one single operation something like plus or minus or square root, these are one single operation. Here you can see that the function evaluation includes more than one operation. It has an addition, it has a square root and it has subtraction. Let us try to see what is the condition number of this function. With little effort you can see that the condition number of this function is less than or equal to $\frac{1}{2}$.

That gives us a feeling that evaluating this function on a computer is going to be very nice and therefore it sounds as if it is well-conditioned, but actually this is not true because we have seen an example in our previous class that a similar function has amplified the error drastically because of the loss of significance. You can see that if x is very large then this number and this number are very close to each other.

And therefore, you tend to get loss of significance in the process of evaluating this function. Therefore, that understanding shows that this function is not going to be well behaved for large values of x. However, the condition number says that it is very good. So, there is a conflict between these two ideas. The actual thing is that when you go to study the behavior of a function which involves many arithmetic operations then just seeing the condition number of the function is not enough. You need to look for the stability criteria.

(Refer Slide Time: 23:49)

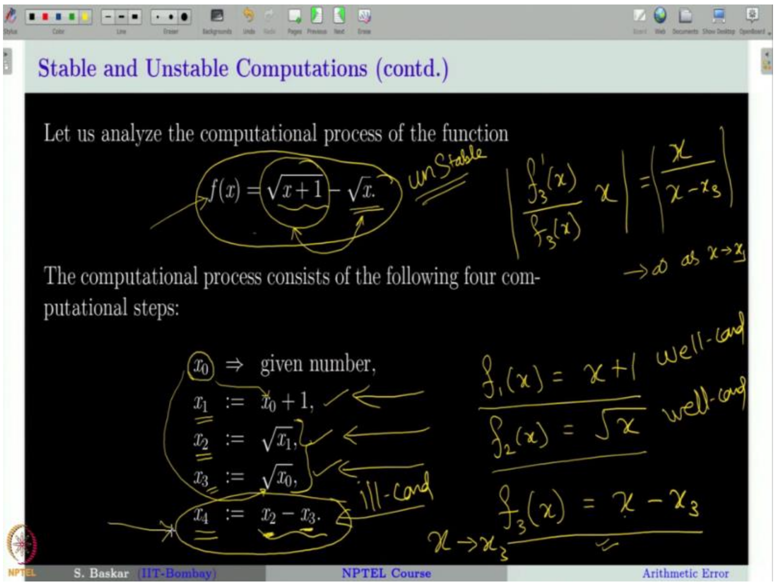


Now, what is meant by stability let us define that. Suppose that, there are n steps to evaluate a function $f(x)$ at a point $x = c$. Just like in the previous example we have there are how many steps you take x and then you add with 1 that is one step and then you find the square root of that, that is another step and then you find square root of x that is another step and then you subtract this.

So, 1, 2, 3, 4 steps are involved in evaluating this function. In this case just finding well-condition of this function is not a good idea to judge whether the function is good or not, what you have to do is you have to look for the well-conditioned or ill-conditioned of each of the steps that is what the stability idea says. Suppose, there are *n* steps to evaluate a function $f(x)$at a point *c*.

Then the total process of evaluating this function that is finding the value of this function involves four steps that is what we meant by saying total process of evaluating this function is said to be unstable if at least one of these steps is ill-conditioned. If all the steps are well-conditioned then only we will say that the process of evaluating this function is stable.

(Refer Slide Time: 25:29)



Let us take this example and see whether this is stable or unstable. In the previous example we have seen that it is a well-conditioned function as far as the conditioned number of the function is concerned. Now, we will split this function into each step and see what is the condition number of each of this steps. How will you split? You want to find the value of that function at some point x_0. The first step is you will take that x_0 and add it with 1, that is this that is the first step.

Once you do that you call it as x_1 and then take the square root of x_1 that will give you the second step. Let us call this as x_2 and then you go to the second term take x_0 and plug in here and get the square root let us call it as x_3 that is step 3 and then once you have x_2 and x_3 then you subtract both of them you get x_4. Now, you have to find the condition number for each of this by considering each step as a individual function that is let us take $f_1(x) = x + 1$.

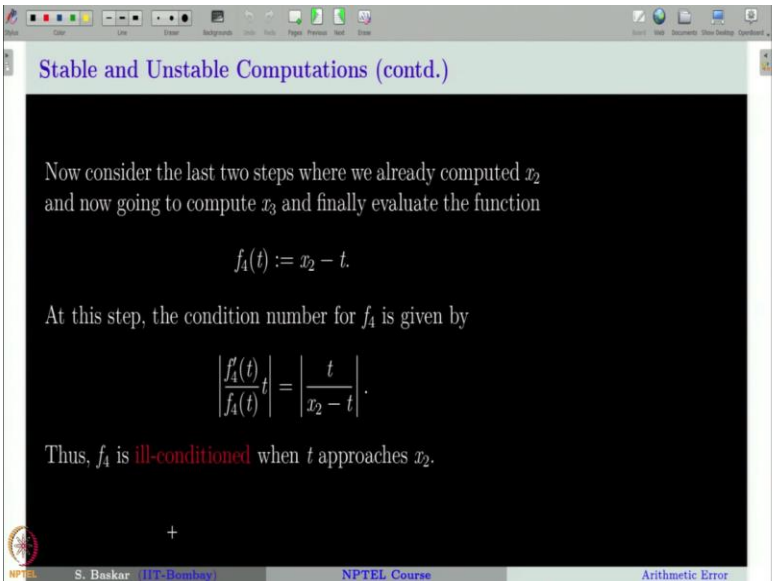
This is one function find the condition number of this function then take $f_2(x) = \sqrt{x}$ that involves these two steps find the condition number of this function and then take $f_3(x)$ equal to either you can write $x - x_3$ or $x_2 - x_3$. One of them you can fix and vary the other one that is enough. Let me take $x - x_3$ you can also take $x_2 - x$ it does not matter for you as long as seeing the condition number of the function, this step alone.

Now, from your previous experience you can see that this is a well-conditioned function, this is also well-conditioned. You can easily judge that this is not going to be well-conditioned especially when $x \rightarrow x_3$ that is whenever *x* goes to very close to x_3 which is precisely going to happen when *x* is very large. When *x* is very large, this term and this term are going to be pretty close to each other.

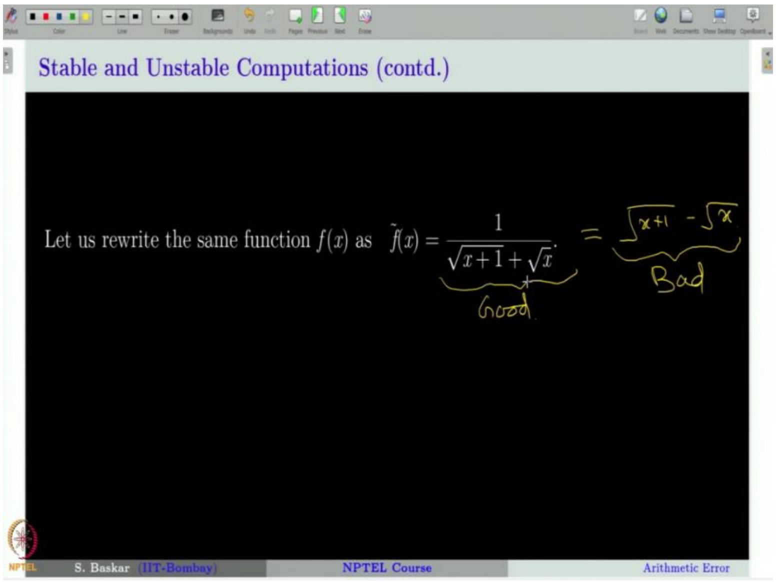
Therefore, when *x* tends to x_3 this will be very bad, why it is so? Let us try to see that $\frac{f_3'(x)}{f_3(x)}x$ that is the condition number that is going to be $\frac{1}{x-x_3}x$ and obviously this tends to infinity as $x \rightarrow x_3$. Therefore, this step is a very bad step, it is ill-conditioned and that will make the process of evaluating this function as unstable.

Individually, if you see the condition number of this function that is misleading, it is well-conditioned. However, the process of evaluating this is unstable. Therefore, this is very dangerous to evaluate on a computer because of this step, this is the idea. As we have seen in the previous example there is a scope to rewrite this function in a different form, mathematically equivalent and that form can give us a better result. What is that form?

(Refer Slide Time: 29:38)



(Refer Slide Time: 29:44)



That is nothing, but $\frac{1}{\sqrt{x+1}+\sqrt{x}}$. So, this is actually equal to $\sqrt{x+1} - \sqrt{x}$. But if you find the value using this expression this is bad, but the same if you use this expression this is very good. Why? You can again split this function into each individual steps like what we did in the previous example you can see that each of this steps are good even you have at one step dividing 1 by some *x* that is also good we have seen, there is no subtraction involved in this.

The basic idea in deciding whether evaluating some expression is good or not is to see whether it involves subtracting two positive numbers which are close to each other that is the overall message that we are taking from this analysis. With this, our chapter on arithmetic error is finished. Thank you for your attention.