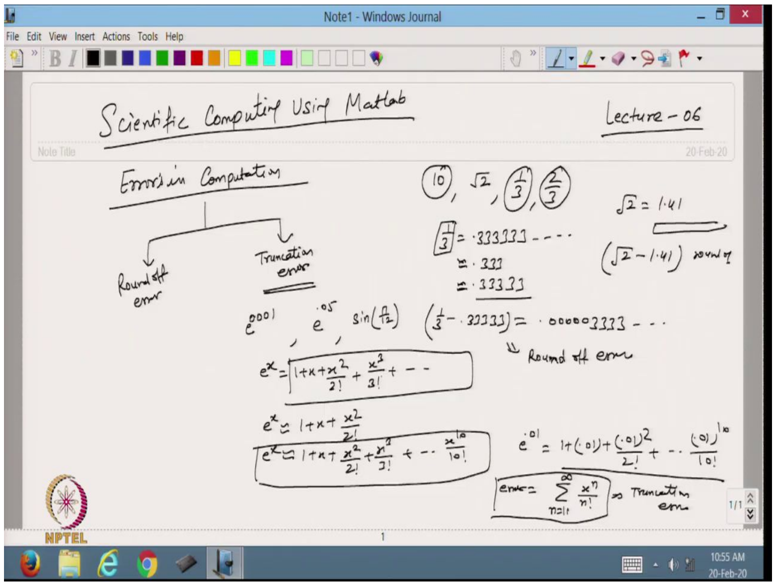
Scientific Computing Using Matlab

Professor Vivek Aggarwal
Professor Mani Mehra
Department of Mathematics
Indian Institute of Technology Delhi
Delhi Technological University

Lecture 6

In Continuation of Basics of Matlab

Hello viewers, welcome back to this course. So today we are going to discuss lecture number 6: Scientific Computing using Matlab. In the previous 5 lectures, we have discussed the basics of Matlab. Now, we will start with Scientific Computing.
(Refer Slide Time: 00:33)



Scientific Computing Using Matlab

Lecture - 06

Errors in Computation

Round off error

Truncation error

$\boxed{10}, \sqrt{2}, \boxed{\frac{1}{3}}, \boxed{\frac{2}{3}}$

$\boxed{\frac{1}{3}} = .333333 \dots$

$\simeq .333$

$\simeq .33333$

$\sqrt{2} = 1.41$

$(\sqrt{2} - 1.41)$ round off

$e^{.0001}, e^{.05}, \sin(\frac{\pi}{12})$

$(\frac{1}{3} - .33333) = .000003333 \dots$

$\Downarrow$ Round off error

$e^x = \boxed{1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots}$

$e^x \simeq 1 + x + \frac{x^2}{2!}$

$\boxed{e^x \simeq 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \frac{x^{10}}{10!}}$

$e^{.01} = 1 + (.01) + \frac{(.01)^2}{2!} + \dots \frac{(.01)^{10}}{10!}$

$\boxed{\text{error} = \sum_{n=11}^{\infty} \frac{x^n}{n!}} \Rightarrow$ Truncation error

So, in scientific computing using Matlab means we are taking the help of numerical analysis and the Matlab to do the numeric computation. So in lecture number 6 we are starting with the first one that how the error in computations. So this is the error in the computation, that if we do the computation using the computer, then we are generally dealing with the errors. Because we are doing the approximation. So, what are the errors that are coming up in the computation? So they are generally two types of errors.

So, there are two types of errors. Generally, we have that if we call it round off error and this is the error we call it truncation error. So, round off error means that suppose I have a number 10.

So, in the computer, if I am dealing with the 10, there is no problem. There is no error. But suppose in the computer I want to use what is $\sqrt{2}$, or I want to use what is $\frac{1}{3}$, or I want to use 2 /3. So in this case, what you will do is the computer is going to have 1/ 3. So 1/ 3 is basically 0.3333 and so on.

So, this is the number we have in the floating point form 1 /3 this one. So in the computer suppose, I take the value 1 / 3 = 0.33 or maybe I can take the approximate value of this 1 / 3 as 0.33333 and I stop here. So in this case I have introduced the error that is I have approximated this number 1 / 3 into the number with 0.333. So in this case, I am introducing my error and that error will be 1 / 3 - 0.33333. So the error will be here. I think it will be 0.0000033.

So this error is called; we call it the round off error. Round off error, because in this case we have rounded off the number 0.333 by this number and then the remaining number whatever is there that is called the error. So this is the round off error. So that the round off error is everywhere because whenever we are dealing with the fractional so this is a fraction. This is a fraction. This is an integer no problem. If there is no, I can use my 10 as such. But $\sqrt{2}$ so suppose I want to use $\sqrt{2}$, so I will use 1.41 or maybe I can use the Matlab. I can use its number with the six sixteen digit number.

So, in that case also in this I am approximating the $\sqrt{2}$ with 1.41. So whatever the error is there. So $\sqrt{2}$ - 1.41 so that is my round off error. So this error is introduced there. So that is called the round off error. And another error is a truncation error, truncation means suppose I have my series maybe I am I want to find what is the value of $e^{0.001}$ or I want to define, what is the value of e^{05} or maybe I want to define what is the value of sin (π/12).

So, in this case, I know that I have a exponential series .

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$$

So in this case it on the right hand side there is infinite series and of course, I cannot use the infinite number of terms to find out the value of this one. In the computation we have to truncate this series. So suppose I approximate this function as $e^x = 1 + x + \frac{x^2}{2!}$ or maybe I can. I suppose I want more accuracy. I will take more terms, so I will write down

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^{10}}{10!}$$

So in this case, I am approximating my exponential function with 10 degrees of polynomial. Now suppose I want to find what is the value of $e^{0.01}$. So in this case, I will put this value

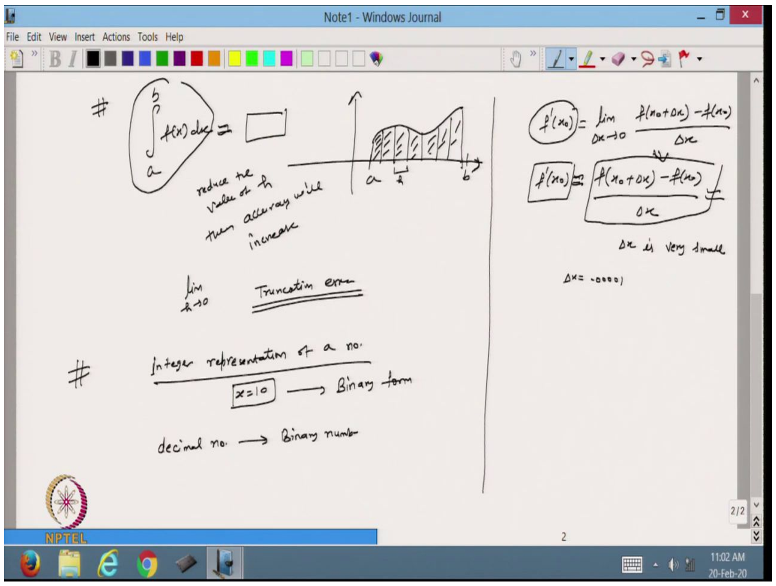
$$e^{0.01} = 1 + 0.01 + \frac{0.01^2}{2!} + \frac{0.01^3}{3!} + \ldots + \frac{0.01^{10}}{10!}$$

So I am using this value a for $e^{0.01}$ this polynomial. So it is a 10 degree polynomial. So in this case I have introduced the error. So error is

$$e^{0.01} - 1 + 0.01 + \frac{0.01^2}{2!} + \frac{0.01^3}{3!} + \ldots + \frac{0.01^{10}}{10!} = \frac{0.01^{11}}{11!} + \ldots +$$

So that will be your error. We have introduced in this number whenever I am defining $e^{0.01}$. So this error is called the truncation error. So this is the one way of one example of the truncation error.

(Refer Slide Time: 07:01)



$$\# \quad \int_a^b f(x)\,dx = \boxed{}$$

reduce the value of h then accuracy will increase

$\lim_{h \to 0}$ Truncation error

$\#$ Integer representation of a no.

$\boxed{x=10} \longrightarrow$ Binary form

decimal no. $\longrightarrow$ Binary number

$$\boxed{f'(x_0)} = \lim_{\Delta x \to 0} \frac{f(x_0+\Delta x) - f(x_0)}{\Delta x}$$

$$\boxed{f'(x_0)} \approx \boxed{\frac{f(x_0+\Delta x) - f(x_0)}{\Delta x}}$$

Δx is very small

$\Delta x = .0001$

Maybe I can define another error form of truncation error, like suppose I want to do the integration $\int_b^a f(x)\,dx$

and suppose I am unable to find the integration analytically. So I will use the computation for that one. So in the computer it is going to give you because this is the way we can define, suppose my function is something like this and this is my a and this is my b and I am defining this function. So in this case, what I will do is split this function into the number of rectangles. And then I will find out the area of these rectangles.

And add all these rectangles together and then it will give me the approximate value of this integral or maybe this is supposed to be the sub interval so I can take this as h. So I can say that for a small h if I take h, I will get some value some approximation of this integral and then I can even reduce the value of h, then the accuracy will increase.

So, the accuracy will increase but in all these forms, whatever the value I am taking h. So this value is whatever we are calculating using the computer. It gives you the approximate value only. Because we know that if I put a limit h tends to 0 only then this sub intervals are going to be infinite in numbers. And only then this value will be an exact value, we can, we are able to find but in the computer this is not possible.

So in this case, I will take a very, very small value of h but never I cannot choose the h=0. So whatever the form it is coming. So this is also an example of a truncation error. Because in this case also, we are truncating the infinite number of steps to the finite number of steps. So this is another form of truncation error.

Another error form of truncation error maybe I can define suppose I have a function, derivative the function at x_0 I want to define. So in this case, I know that the, it is that is a definition of the $\frac{d(f(x))}{dx} at\ x_0$. So this is the definition of the derivative. Now suppose I am finding the derivative with the help of a computer with the help of numerical analysis.

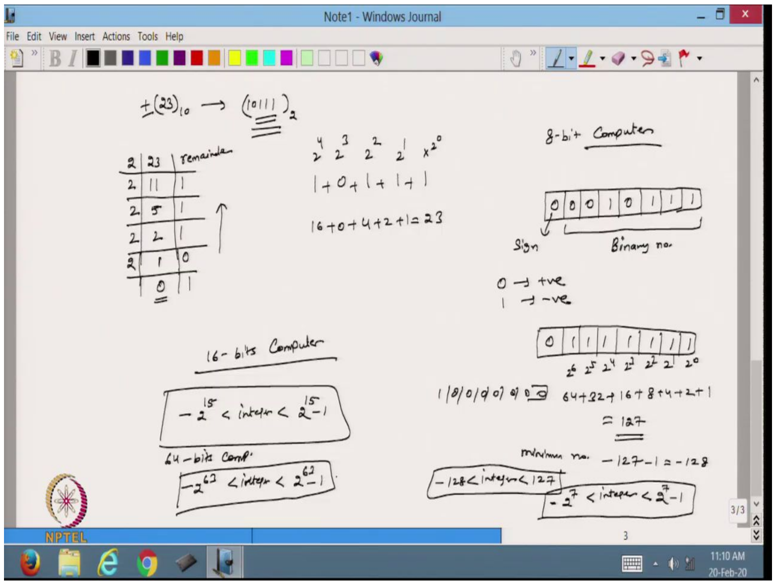
So, in this case what we do, we just approximate this function derivative with this x naught plus delta x minus fx naught divided by delta x. And we call it where delta x is very small, where delta x is very small. So in this case what we have done, we have value of the function derivative of the function approximating with this factor on the right hand side.

And where I am writing the delta x is very small because $\Delta x \to 0$, but in the computer, I need to find the delta x is very, very small. So I will choose the value of delta x. So maybe I can define the Δx = 0.0001 very small value. And based on this value I will find out the value of this factor f(x_0)+ Δx - this and that will be the approximate value of the derivative $f'(x_0)$. So in this case also I am truncating or in another form also we call it discretization error or the truncation error.

So we in this case also we are truncating the infinite number of values of △x to the finite number of △x. So that is another example of the truncation error or in the derivative form we also call it discretization error. Because in this case we are discretizing the derivative of the function.

Because here I am having the continuous form, but here we are defining changing into the discretion form. So that is another form of truncation error. So the question is how this error happens. So we start with how the number is saved in the computer. So let us start with this one. Suppose I have a number right, so first I will write down the integer (re) representation of a, of an integer, of a number.

It means suppose I have a number x = 10. So 10 is itself a representation. So there is no approximation and this will be saved in the computer. So I know that in the computer the number is first changed into the binary form. And then this will be saved in the computer. So suppose I have the number x = 10 and so before that I just tell you how we can change a decimal number into the binary number.
(Refer Slide Time: 12:55)



$\pm(23)_{10} \longrightarrow (10111)_2$

2	23	remainder
2	11	1
2	5	1
2	2	1
2	1	0
	0	1

$2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad \times 2^0$

$1 + 0 + 1 + 1 + 1$

$16 + 0 + 4 + 2 + 1 = 23$

8-bit Computer

0	0	0	1	0	1	1	1

Sign   Binary no.

$0 \to +ve$

$1 \to -ve$

0	1	1	1	1	1	1	1
	2^6	2^5	2^4	2^3	2^2	2^1	2^0

$64 + 32 + 16 + 8 + 4 + 2 + 1 = 127$

minimum no. $-127 - 1 = -128$

$\boxed{-128 < \text{integer} < 127}$

$\boxed{-2^7 < \text{integer} < 2^7 - 1}$

16-bits Computer

$\boxed{-2^{15} < \text{integer} < 2^{15} - 1}$

64-bits Comp.

$\boxed{-2^{63} < \text{integer} < 2^{63} - 1}$

So let us start with this one. Suppose I have a number 23 and this number I am having in the terms of decimal. Now I want to convert into the binary form. So what I will do I will write 23, and then this is the way I will divide it by 2. So it will be the 11 and then the remainder, remainder be 1. So in that case I will multiply by 2. So 5 into 2 and then remainder 1, then it is 2 by 2, 4 and then reminder 1 then I will divide by 2. So 2 means here I would get the 0. So 2 1 2 that will give you 0.

So 2 1 2 and the remainder is 0 and then I will again divide by 2. So it will be 0 and the remainder be 1. So I want to see go till the 0, I am getting here. So in this case, this is the way we will take this one. So the representation will be 10111. So 23 in the decimal form can be converted into the binary form with the help of this one.

Now suppose I want to find out that whether it is giving you the exact value or not. So I can show that this is 10111. So this number I will multiply by 2^0. This is 2^1, 2^2, 2^3, 2^4 and I am adding it all together. So that is 23. So in this way, we can change any number into the binary form.

Now suppose, so this is the integer we are printing. Now suppose I have a 23. So in this case, my number will be ended by this one. Now in the computer suppose I have an 8-bit computer. So an 8-bit computer means we have a memory that is divided into the 8-bits. So 1, 2, 3, 4, 5, 6, 7 and 8. So in this case what I will do, so the first one I will generally go for the sign. That this number is positive or negative. So that is the sign and the remaining part will be for the binary variables or binary numbers.

So in this case if I want to put 23 here, so this will be 1110100 and this is a 0. So this 0 is for positive and 1 is for negative. So that is the way this 23 will be saved in the computer. So suppose if I have an 8-bit computer and I want to check, what is the maximum number we can save, so in this case the maximum number can be saved is that all values should be 1, 1. So 1111111 so it is a seven 1 and here it is the sign.

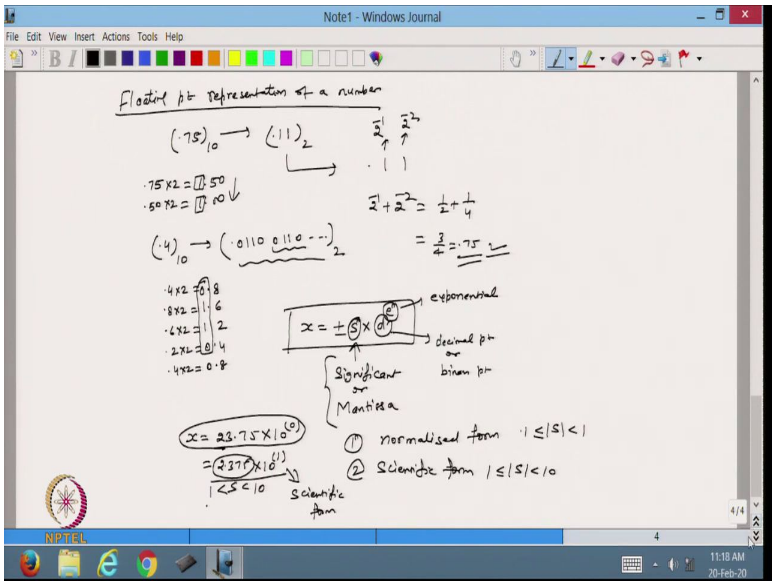
So suppose I take 0, so the positive sign. So in this case the maximum number is 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0. So add it gives you the value 127. So it gives you the value 127. It means the maximum number you can save in the 8-bit computer is whatever that is the integer number and that is 127.

So similarly I can go for minus 127 the minimum value, minimum number. So minimum I can go for minus 127, but the only thing is that in the computer, we know that if I have to save the value 0 with the 8-bit computer then this is 0 0 0 0 0 0 0. So I put the 0 here so that is a 0 and does not matter, if I put 1 here. Because there is no minus 0 and 0 there.

So generally what we do we add one number to the negative of the, this number. So I will add minus 1. So this will become minus 28. So in this case, I will say that if I work in the 8-bit computer and so that is the number integer I can save in the computer from minus 28 to 127. So this is the number we can define.

So this number I can also write as integer, we can save. So that is 2 raise to power, so 2 to raise to power 5 7. So this is the seven digit, 7 bits 1, 2, 3, 4, 5, 6, 7. So $2^7 - 1$. So that is 127 and this will be minus 2 raise to power so that is the range we can define when we are working with the 8-bit computer. Similarly, you can go for up to 16-bit. So in the 16-bit computer 16 bits computer. So this integer range it will be so 15 say up to it will go to up to 15 - 1 and - 2^{15}.

So that is the range we can define for the 16-bit computer. So these days we are working with 32-bit or 64-bit so accordingly you can check for the 64-bit computer, the present computer. So it will go to up to $2^{63} - 1$. So that is the range, -2^{63}. So you can see that it is a very big number we can deal with at the present computer. So that is how we can define the integer.
(Refer Slide Time: 20:32)


Note1 - Windows Journal
File Edit View Insert Actions Tools Help
Floating pt representation of a number
$(\cdot 75)_{10} \longrightarrow (\cdot 11)_2$
2^{-1} 2^{-2}
$\cdot 75 \times 2 = 1.50$
$\cdot 50 \times 2 = 1.00$
$2^{-1} + 2^{-2} = \frac{1}{2} + \frac{1}{4} = \frac{3}{4} = \cdot 75$
$(\cdot 4)_{10} \longrightarrow (\cdot 0110\ 0110 \cdots)_2$
$\cdot 4 \times 2 = 0.8$
$\cdot 8 \times 2 = 1.6$
$\cdot 6 \times 2 = 1.2$
$\cdot 2 \times 2 = 0.4$
$\cdot 4 \times 2 = 0.8$
$x = \pm S \times d^{e}$
exponential
decimal pt or binary pt
Significant or Mantissa
$x = 23.75 \times 10^{(0)}$
$= 2.375 \times 10^{(1)}$
$1 \le S < 10$
Scientific form
① normalised form $\cdot 1 \le |S| < 1$
② Scientific form $1 \le |S| < 10$
NPTEL
11:18 AM
20-Feb-20

Now the thing comes when we deal with the Floating point floating point representation of a number. Like I have 0.75, so that is given to me in the decimal form and I want to convert this one into the binary form. So integer we know but in the fractional part. Suppose, I have 0.75, then how I want to convert this one into the binary forms. So what I do I will multiply this by 2. So it will give you 1.50.

So I will keep this fractional part. And again, I will write 0.50 into 2 and now it gives you 1.00. So this fractional part is 0 now. So this is the number we can say and by this way we can calculate. So it gives me 0.11. So it means 0.11 gives me the value in the decimal form for this 0.75 and how I can verify because I can convert this one. So it is 0.11, so this will go to 2^{-1} and this will go to 2^{-1}. So it will give you $2^{-1} + 2^{-2}$. So it is 1 / 2 + 1 / 4. So that is the 3 / 4. So 3 / 4 is 0.75.

So that is verified. Maybe I can so this is the value of the 0.75 decimal form into the binary form. Maybe I have some other number 0.4 that is in the given to me in the decimal form. Now, I want to convert this one into the binary form. So what I will do I will write 0.4 into 2. So, that gives 0.8. Now, I will find 0.8 into 2. So that is 1.6, then 0.6 into 2. So that is 1.2, 0.2 into 2 that is 0. so I can put 0 here so 0.4.

So a 0.4 came again. Now, I will write 0.4 into 2. So it is 0.8 and then the same thing will repeat again and again. It means this is the binary form we can take so I can write from here that it will be 0.0110 then again 0110 then again. So this is the recurring relation and that is the representation of 0.4 in the decimal form to the binary form.

So from here we can see that 0.4 is a truncated value. But if I convert this one into the floating point into the binary form. Then this is a recurring relation we are getting. So that way we can define a, we can convert the function in the decimal form a floating point in the decimal form to the binary form.

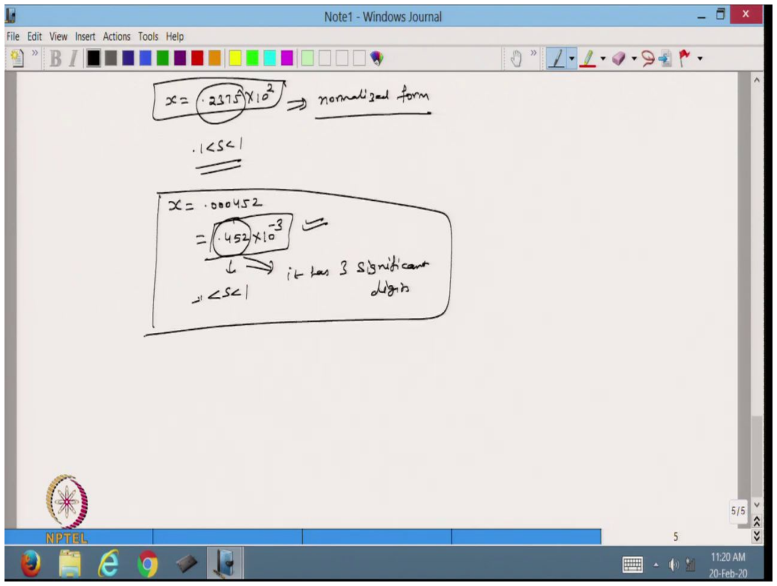
Now we are dealing with the floating point representation of the number. So how can we do that? So in the floating point suppose I have a number x. So this is our presentation. It can be a positive and negative and I will represent by S multiply by d and then e. So that is the floating point representation of a number in decimal form or in the binary form. So, what is this? So that is the sign this is what, so this is called Significant or Mantissa, so that it gives you the significant or mantissa.

This is, so that gives you the decimal, so this is, it can be a what we call it decimal part or binary form. And that is the exponential part, exponential. So in this case, I can represent any number so suppose I have a number x is equal to maybe 23.75. So this number we can write as I can write this number as 2.375 may be multiplied by 10.

So in this case, this is the power, power is 1. So that is the number we have in this case the same number if somebody asked me to write the same number. So this number can be written as 10^0. The only thing is that here we are putting this number. So we have two digits before the decimal. And in this case, we have one digit before the decimal. So in this case, we have to define what is the basically a well-accepted form of the number.

So that is called what we are doing here is that there are two forms that are well accepted by all over the world. So the first one is the normalized form and another is scientific form. So the normalized form, is that the value of S, so this the value of S is always lying between 0.1 to 1. And this in this case its value is from 1 to 10 less than 10. So this is a normalized form and this is the scientific form. So in this case 2.375×10^1 .

So if I have this one. So this form is not a scientific form or the normalized form because here the value of S is 23 so it is not following any of the categories. But in this case, I can say that this is the value of S and this value is S is all lying here between 0.1 to 1 , it is 2.3. So it is lying between 1 and 0. So this is a scientific form.
(Refer Slide Time: 28:18)


Note1 - Windows Journal
File Edit View Insert Actions Tools Help
$x = .2375 \times 10^2 \Rightarrow$ normalized form
$.1 < S < 1$
$x = .000452$
$= .452 \times 10^{-3}$
it has 3 significant digits
$.1 < S < 1$
NPTEL

Now, the same number I can write x is equal to 0.2375 into 10^2 . So in this case, my S number is now this is my S so it is 0.1 to 1. So that is the normalized form. So in the normalized form we put all the digits whatever the non-zero digits are there on the right of the decimal and raise the power whatever is needed. So, that is the normalized form of the value x. Maybe I take another value of x, suppose I have a number of 0.00452. So this is the number we have the real number or floating point number.

So this number I can write in the form of, so what I do. So, these are the three zeros this is unnecessary 0. So I can write this number as 0.452 into 10^{-3} . Because I have taken this dot to the three on the right hand side. So I have to multiply by this 10^{-3} . So that is the normalized form with this number. In this case my S is lying between 0.1 to 1. So my S is lying 0.1 less than 1 and this is the exponential we have. So in this case, I can say that this number is in the normalized form and this is significant because it has 3 significant digits.

So that is the normalized form of the decimal number. So today in this class. We have discussed about that the how the two type of error comes to the computer whenever we do the calculation they are (trunc) round off error and the truncation error, and then we also discuss that how we can convert a decimal number into the binary number which is used when we are dealing with the that how the number is allocated in the computer. So in the next class we will go further with this one. Thanks for watching. Thank you.