Scientific Computing Using Matlab

Professor Vivek Aggarwal

Professor Mani Mehra

Department of Mathematics

Indian Institute of Technology Delhi

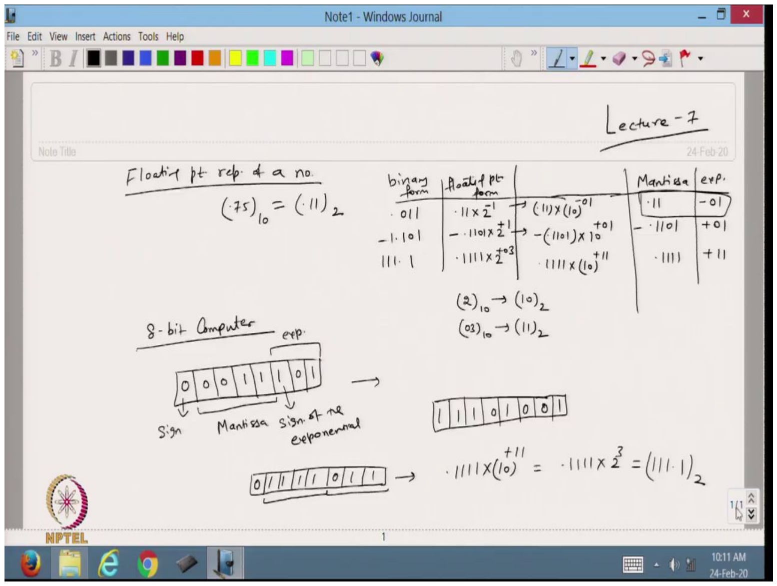
Delhi Technological University

Lecture 7

Floating Point Representation of a Number

Hello viewers, welcome back to this course. So today we will start with the lecture 7. So in the previous lecture we have discussed the integer representation of a number and we have also discussed little bit about the floating point representation of a number. So we will continue with from, from there.

(Refer Slide Time: 00:48)



So, today we will discuss how the floating point representation of a number is saved in the computer. So for example suppose in the last class we have discussed that any floating point number in the decimal form, like in the previous class, we have converted this 0.75. So that was equal to 1 1 in the binary form.

So, now suppose we have numbers, so that number is given to me in the form of binary. So suppose I have a number like 0.011, or maybe I have number 1.101, or maybe I have number

111.1. So this one can be presented with the floating point form. So, floating point form is that we can write this number as 0.11 into 2^0 , so it can be written as 2^{-1} or this number I can be written as minus 0.1101 into 2^1 or I can write this number 0.1111 into 2^3 .

But in the, because we are writing this number in the form of a binary. So in the binary, we know that 2 in the decimal form that can be written as 1 0 in the binary form or maybe 3 if I write 3 here so that can be written as 1 1 in the binary form. So we can write this number again in the form of 0.11 into 1 0.

So, this 10_2 is not in the decimal form. It is the binary representation of number 2. So this can be written as minus 0 1 and this number I can write the minus 0.1101 multiply by 10 plus 0 1 and this one is 1111 into 2^2 , so $2_{10} = 10_2$ raise to power 11. So that is the representation of a floating point in the form of a binary. So, this one I can write as.

So, I can, so how can I save that floating point number. So first I will convert this one into the binary form and then, so it is an 8-bit computer. So the first bit is for signs, then the next bit next 4 bit I will take for the mantissa part. And this is for the exponential part. And that is the sign of the exponential.

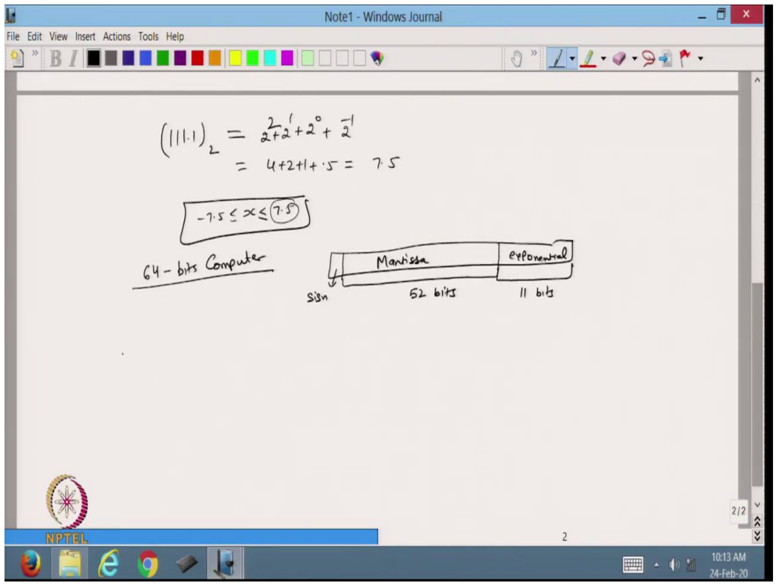
So in this case the mantissa is this one 0.11 and this is 0.1101 and this is 0.1111. So that is the mantissa. And the exponential part is minus 0 1 plus 0 1 plus 1 1. So that is the exponential part. So now suppose I want to save this number in the 8-bit computer. So 0 is for plus sign and 1 is for negative. So I will put it here, so mantissa is the next 4-bits. But I need this 1 1 only. So I will put in this case I will put just 1 1 and that is 0 0 and the exponential part is negative so I put 1 and 01 so that gives me the number which we have saved in the computer. So the same way I can put the next one is the same way.

So, if I have an 8-bit computer, so I will put here the negative sign so 1 and then I will put 1 1 0 1 and then 0 so that is the sign and 0. So in this way if I work with the 8-bit this number can be saved in the memory. So the next question is that suppose I go with the 7 and I have an 8-bit computer. So what is the maximum number we can save?

So this one is 1 2, 3, 5, 6 so in this case, what I will do is possible for me if I so that is a sign so it does not matter the next four digit I can put 1 1 1 1. So this one I can put and in the exponential also, I can put the 0 that is a positive and 1 1. So this number can be written as $0.1111\text{x } (10)_2^{+11}$.

So that is the number we can say so it means that 11_2 is means this number is equivalent to $0.1111 \text{ x } 2^3$. So this one it means I have taken the three digits after the decimal here so I can write 111.1. So that is the number we have in the binary form. So this number I can convert into

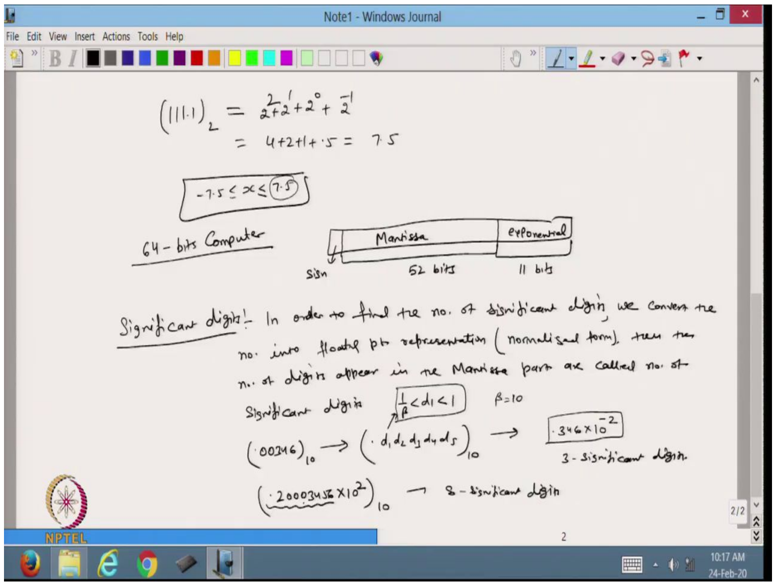
the decimal form by, it is converting into the decimal form.
(Refer Slide Time: 07:35)



So this number is so I can convert this number one 111.1 2. So this one can be written as $2^0 + 2^1 + 2^2 + 2^{-1}$. So this can be written as 4 + 2 + 1 + 0. 5. So I can write this as a number 7.5.

So, this is the largest number we can save in the 8-bit computer. So I can say that if I have a 8-bit computer then this is the number 7.5, I can maximum, biggest number I can compute and the lowest one can be minus 7.5. So that is the number we can deal with in the floating point for the 8-bit computer. The similar way we can define these days we have 64-bits computers.

So in this 64-bits computer we have these are the so in this case, the first is for the sign, next so these are the 52 bits. So 52 bits is for the mantissa part and the next 11 bit, so that is for the exponential. So, 52+ 11 = 63 +1= 64. So this is the way a number can be used in the 64 bits. So definitely in this case my number I like in the 8-bit computer I was able to calculate only 7.5 the highest number but in this case the highest number will go will become a very big number. Because in this case we have 52 bits in the mantissa and 11 bits in the, in the exponential part. So this way a binary number a floating-point number can be saved in the computer. So, now after doing this one.
(Refer Slide Time: 09:53)



$$(111.1)_2 = 2^2+2^1+2^0+2^{-1}$$
$$= 4+2+1+.5 = 7.5$$

$$\boxed{-7.5 \le x \le 7.5}$$

64 - bits Computer

Significant digits:- In order to find the no. of significant digits, we convert the no. into floating pt. representation (normalised form), then the no. of digits appear in the Mantissa part are called no. of significant digits $\boxed{\frac{1}{\beta} < d_1 < 1}$ $\beta = 10$

$(.00346)_{10} \longrightarrow (.d_1 d_2 d_3 d_4 d_5)_{10} \longrightarrow \boxed{.346 \times 10^{-2}}$ 3 - significant digits

$(.20003456 \times 10^2)_{10} \longrightarrow$ 8 - significant digits

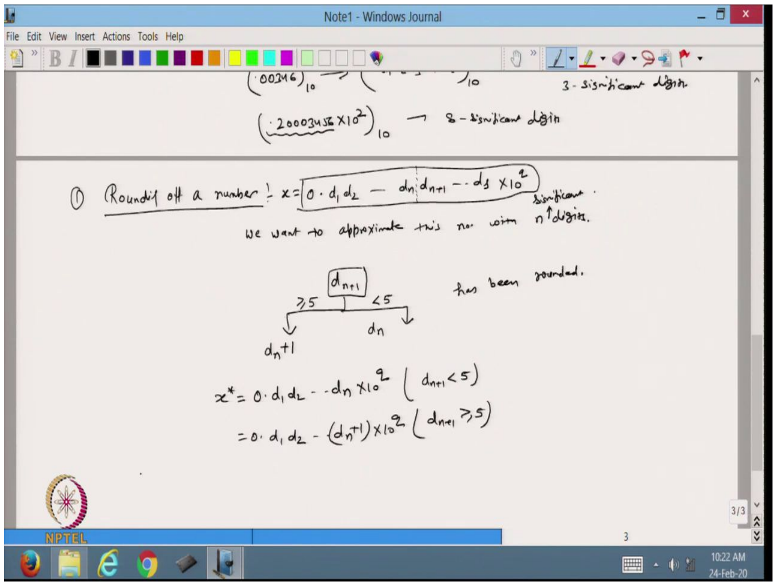
We will define what about the significant digit. So significant digit, so in order to find the number of significant digits. We convert the number into floating point representation. So, floating point representation is in this case I am taking the normalized form or I can normalized form. Then the number of digits appearing in the mantissa part are called the number of significant digits.

So, for example suppose, I have a number like this. I have numbered 0.00346. So that is my number in the decimal part. But this number is because in the normalized form I know that the value is here, so this number can be written as 0.d1, d2, d3, d4, d5. So in the normalized form the d1 should be less than 1 by beta 1 where beta is the basis I am taking so that base is 10. Because we are dealing with the decimal form, so 1 by 10 is 0.1.

So it can be greater than this one. So it should be greater than 0.1 less than 1 so that is the normalized form. But here it is 0, so I will do what I will do. I will write this number as 0.346 and that will write 10^{-2} . So, in this case, I have taken the two numbers on the left hand side. So I will write this one this so in this case now this is my normalized form in the floating point. So I can write that this number has three significant digits.

Because this is a 1, 2 and 3. So these are the three significant digits we can have. Similarly, suppose I have a number like 0.2003456 x 10^2 . So that is my number in the form. So in this case, this is the normalized form. So I can say that 4 and so all 8 significant digits. So that is so in this form that this number is having the normalized form. And this has 8 significant digits. So this is about the significant digit.

(Refer Slide Time: 13:56)



$(\cdot 00346)_{10} \rightarrow (\quad)_{10}$ 3 - significant digit

$(\underset{\sim\sim\sim}{\cdot 20003456} \times 10^{2})_{10} \rightarrow$ 8 - significant digit

① Rounding off a number :- $x = \boxed{0 \cdot d_1 d_2 - d_n \vdots d_{n+1} - \cdot d_s \times 10^{q}}$ significant.

we want to approximate this no. with n digits.

$$\begin{tikzcd} & \boxed{d_{n+1}} \arrow[dl, "\geq 5"'] \arrow[dr, "< 5"] & \\ d_n+1 & & d_n \end{tikzcd}$$

has been rounded.

$x^* = 0 \cdot d_1 d_2 - \cdot d_n \times 10^{q} \quad (d_{n+1} < 5)$

$= 0 \cdot d_1 d_2 - (d_n+1) \times 10^{q} \quad (d_{n+1} \geq 5)$

Now we find out how the errors occur in the computer. So for this one, I will find out that there are two types of approximation numbers. So first is called rounding, rounding off or rounding a number, a number. Because we know that in the computer whenever we are dealing with the fractional part, we have to approximate the number.

So in that case how can we approximate the number? So this is done by the rounding of a number. So this is how we will do that. So let I suppose I have a number like d1, d2, dn, dn plus 1 up to ds. So that is a decimal part before that it is 0, into 10 raise to power, so 10 raise power is my base I am taking 10 and then the suppose I have here the quotient q, exponential part the q.

So this is a number given to me and that number I have written s means it has a number of significant digits. So this is the normalized form. Now suppose we want to approximate this number. We want to, we want to approximate this number with n digits in n significant digits, n significant digits.

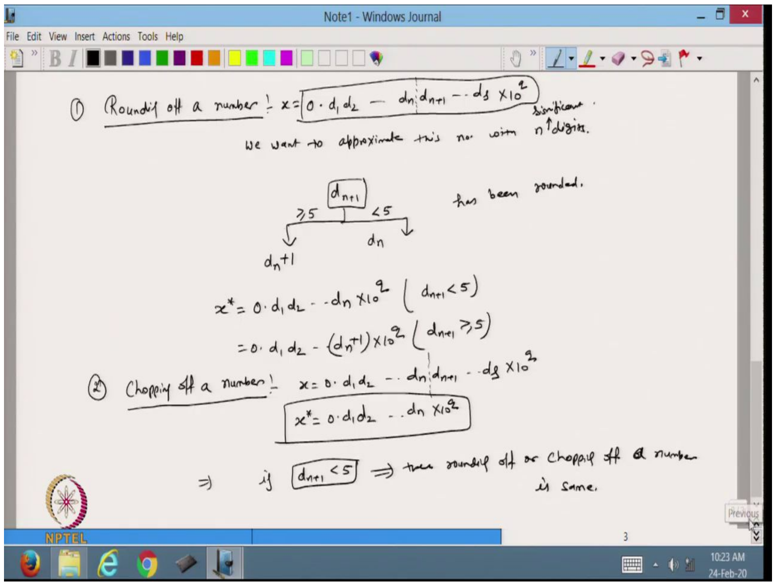
So, what I will do in this case, so what I will do is that now I want to approximate this number with n significant digits. So how we can approximate, so this is the two way. So now I will take, I have to look because here I want to approximate or truncate this number. So in this case, I have to take care of what is this one. So I have to take care, what is my digit value, value of the digit at n plus 1th places.

So, this number can be written as, so if this number $d_n + 1$ is greater than equal to 5 and less than

5. So what will happen in this case? If this number is greater than equal to 5 then the number at nth digit will be added to 1 and in this case, I will have my number d_n that will be the same.

So that is the called the rounding off a number or we call it that number this number has been rounded off, has been rounded. So in this case my new number, so suppose my, this is my x the exact number. So I can say that x star is my new number and this is written as d_1, d_2 up to d_n into 10^q .

So this is happening when my $d_{n+1} < 5$ and this will be equal to $0.d_1$, d_2, $d_n + 1$. So one value is increased and this will be equal to this one, if $d_{n+1} \geq 5$. So that is called the rounded off. So this is the way we can do the rounding of a number.
(Refer Slide Time: 18:00)



① Rounding off a number :- $x = \boxed{0 \cdot d_1 d_2 - d_n d_{n+1} - \cdot d_s \times 10^q}$

we want to approximate this no. with n significant digits.

$$\begin{tikzcd} & \boxed{d_{n+1}} \arrow[dl, "\geq 5"'] \arrow[dr, "<5"] & \\ d_n+1 & & d_n \end{tikzcd}$$

has been rounded.

$$x^* = 0 \cdot d_1 d_2 \cdot \cdot d_n \times 10^q \quad (d_{n+1} < 5)$$
$$= 0 \cdot d_1 d_2 - (d_n+1) \times 10^q \quad (d_{n+1} \geq 5)$$

② Chopping off a number :- $x = 0 \cdot d_1 d_2 - \cdot d_n d_{n+1} \cdot \cdot d_s \times 10^q$

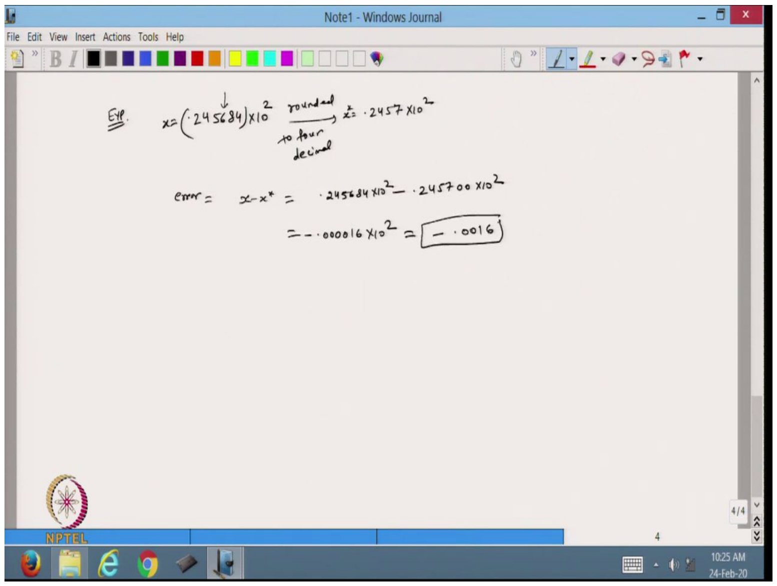
$$\boxed{x^* = 0 \cdot d_1 d_2 \cdot \cdot d_n \times 10^q}$$

$\Rightarrow$ if $\boxed{d_{n+1} < 5} \Rightarrow$ then rounding off or chopping off a number is same.

The second one is the chopping, chopping off a number. So in this case, we do not care what is the value of dn plus 1. So suppose we have this number the same number of I have dn, dn plus 1 up to ds. So s number of significant digits 10^q . So in this case, I want to chop this number using only n digits.

So now I do not care about what is the value of the d_{n+1} . So in this case, I just chop this number and I will approximate this number by $0.d_1$ d_2 up to d_n x 10 raise 2 power q. So that is my chopping off number. So in this case, if you see that if my $d_{n+1} < 5$ then the rounding off number and the chopping off number is the same. So that should be taken care of. So from here, I can say that if the $d_{n+1} < 5$ then rounding off or chopping off a number is the same, that is we have to take care.

(Refer Slide Time: 19:40)



Exp.

$$x = (.245684) \times 10^2 \xrightarrow[\text{to four decimal}]{\text{rounded}} x^* = .2457 \times 10^2$$

$$\text{error} = x - x^* = .245684 \times 10^2 - .245700 \times 10^2$$

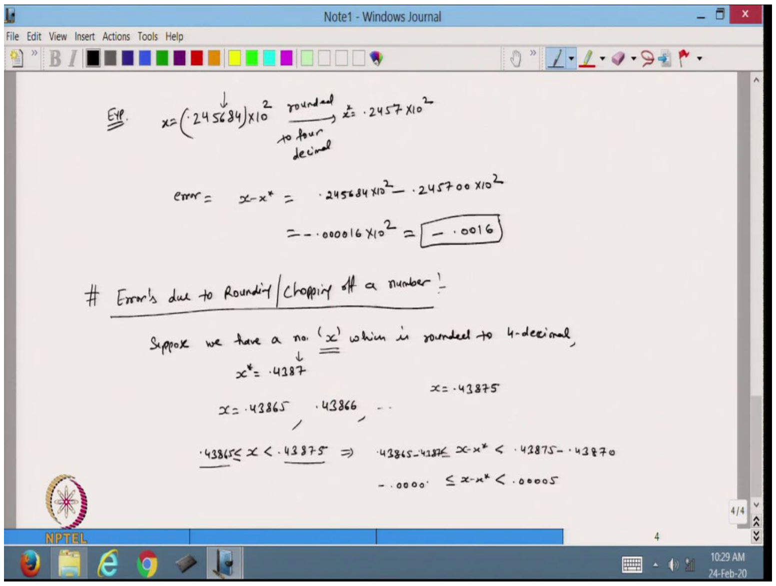
$$= -.000016 \times 10^2 = \boxed{-.0016}$$

So for example, so I take the example one, so suppose I take a number 0.245684 x 10^2 and I will round off this number up to four decimal. So in this case four decimals means here I want to know. So in this case, my next digit is greater than 1, so I can write 2457 x10^2 , so that is the rounded number in this case.

And now suppose I have rounded this number. So now I want to see the, what is the error because this is my x and this is my x^*. So I want to find the error. So error means I will want to find what is the difference between x and x^*. So from here I can see that my x = 0.245684 x 10^2 minus so this number is 24570 and 0.

So that number I have taken so this is the rounded number and so this number becomes bigger as compared to the previous number. So from and I can see that first three digits, so it is 0.000 and this is 684 and it is coming from the 700. So I can put a negative sign here. And then it is 0.16 x 10. So this is minus 0.0016. So that is the error we have in this number when we have rounded this number up to four decimal. So this error is introduced there.

(Refer Slide Time: 21:44)



Eg. $x = (\overset{\downarrow}{.245684}) \times 10^2 \xrightarrow[\text{to four decimal}]{\text{rounded}} x^* = .2457 \times 10^2$

$\text{error} = x - x^* = .245684 \times 10^2 - .245700 \times 10^2$

$= -.000016 \times 10^2 = \boxed{-.0016}$

# Error's due to Rounding / Chopping off a number :-

Suppose we have a no. 'x' which is rounded to 4-decimal,

$x^* = .438\overset{\downarrow}{7}$

$x = .43875$

$x = .43865, .43866, \ldots$

$\underline{.43865} \le x < \underline{.43875} \Rightarrow .43865 - .43870 \le x - x^* < .43875 - .43870$

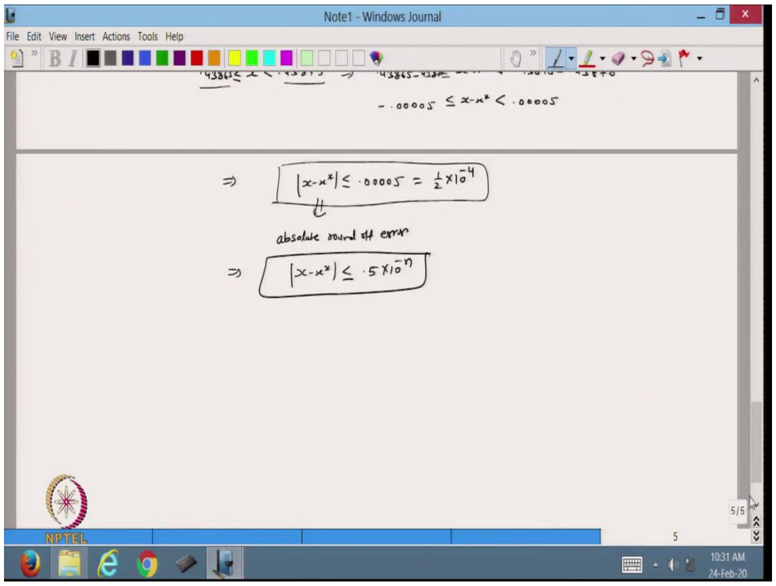
$-.0000 \le x - x^* < .00005$

Next thing we want to discuss the errors due to rounding or chopping off a number. So how much the errors we can find when we rounded off a number or a chopping off a number. Suppose we have a number that is represented by x, which is rounded to four decimal and let the number come if x star is equal to 0.4387.

So, I want to find what is the value of x. So my x star is given to me and I have rounded the number x up to the four decimal. So in this case, so here it is 7. So I can say that x can be 0.43865 it can be 0.43866 so like this one or so in this case I can say that my x ≥ 0.43865.

So this is the number, maximum x can be always greater than this one because now what is the smallest number, the smallest numbers can be, suppose my x = 0.43875. So if I have this number, then this becomes the 8. So I can say that this number can be only less than 0.43875. So I am not putting equal because the equal sign means if it is 4 here, then this number will be 0.4387. So that is the lower bound of this number and this is the upper bound of this number. So from here I can write, now what I do is I want to see what is the $x - x^*$. So I want to find out the error that is how much is the error between $x - x^*$.

So, I will subtract my x^* form each of the numbers. So it will be 43865 - 0.4387. This number is 0.43875 - 0.4387 87 means this is 0. So from here, I can say that my $x - x^*$ can be less than equal to, from here so this is 875 (come) going this one, so 0.00005 and this number can be written as, so this is a not an equal sign is less than and this one is 870 so I can minus 0.00005.
(Refer Slide Time: 25:28)



$$-.00005 \le x - x^* < .00005$$

$$\Rightarrow \boxed{|x - x^*| \le .00005 = \tfrac{1}{2} \times 10^{-4}}$$

absolute round off error

$$\Rightarrow \boxed{|x - x^*| \le .5 \times 10^{-n}}$$

So using this number. So from here, I can write that my $|x - x^*|$ value can be written now as 0.00005. So this is the number we can have and this one I can write as ½ x 10^{-4} . So in this case because we have rounded the number up to four decimal. So this is a 10^{-4} is coming and half. So from here, I can say that if I take the absolute value of the rounded error, I can take that absolute round off error.

So that is a round off error. So this error is always less than equal to ½ x 10^{-4} . So wherever the digit I am rounding so this can be written as, so from here, I can say that if I am rounding my number at the nth digit. So this is always less than 0.5 x10^{-n} . So this is a, we can say that the rounded off error is always less than equal to 0.5 x10^{-n} , if I have we are rounding the number after n digits. So that is the way we can define.

Now, we can say that what about the error due to chopping of a number. So in the chopping of a number what will get. So in the chopping of the number we are not rounding this. So in this case my $x - x^* < 10^{-n}$. Because 0.5 will not come because we are not increasing the number by one digit. So in this case my chopping error will be this one. So that is the error due to the chopping of a number. So from here, I can say that the rounding, rounding of error is always half than the chopping of error. So that is why we always prefer to calculate the rounding off error.

Because rounding off error is if from here you can see the rounding off error is half of the chopping error. So that is always useful to go for the rounding error. So this is a, this is about, all about the errors due to the rounding off and the chopping off and in the next lecture we will discuss further about the errors in the computer. Thanks for watching. Thanks.