MATLAB Programming for Numerical Computation

Dr.Niket Kaisare
Department Of Chemical Engineering
Indian Institute of Technology, Madras

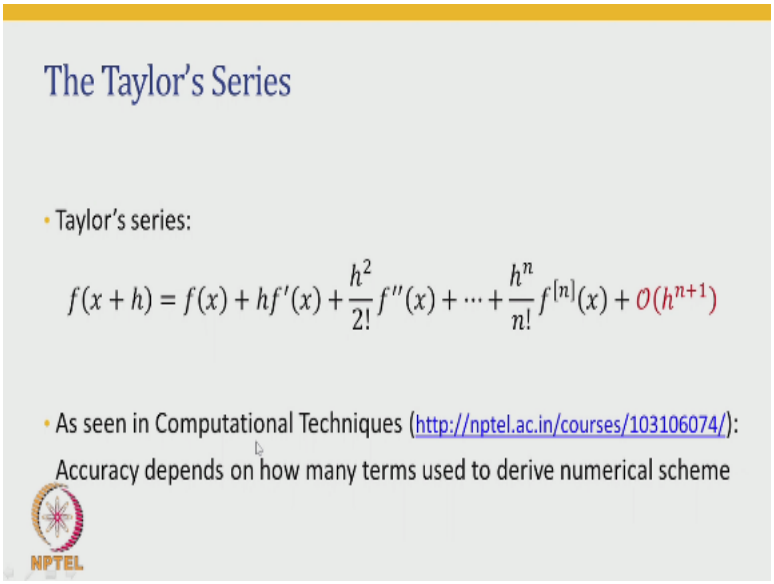
Module No. #02
Lecture No. #2.2
Errors and Approximations – Truncation and round of errors

Hello and welcome to MATLAB programming for numerical computations. We are in module 2. This is lecture 2.2. In this lecture we are going to cover truncation errors and round off errors. In the previous lecture we had seen Maclaurin series expansion of e ^ x. And saw that as we retain more number of terms in that series, the error reduces and we also saw what is known as machine precision.

So there is a least count that is associated with using real numbers in MATLAB or any other computer based program okay. We are going to use these concepts to understand what truncation error is and what round off error is and how that works okay. So, this is the Taylor's series. The Taylor's series is given by $f(x + h) = f(x) + hf'(x) + h^2/2! f''(x)$ so on and so forth.

If we retain only nth order terms in the in the Taylor's series or the Maclaurin series the error is of the order of h ^ n +1. This is something that we have also seen in the computational techniques course.

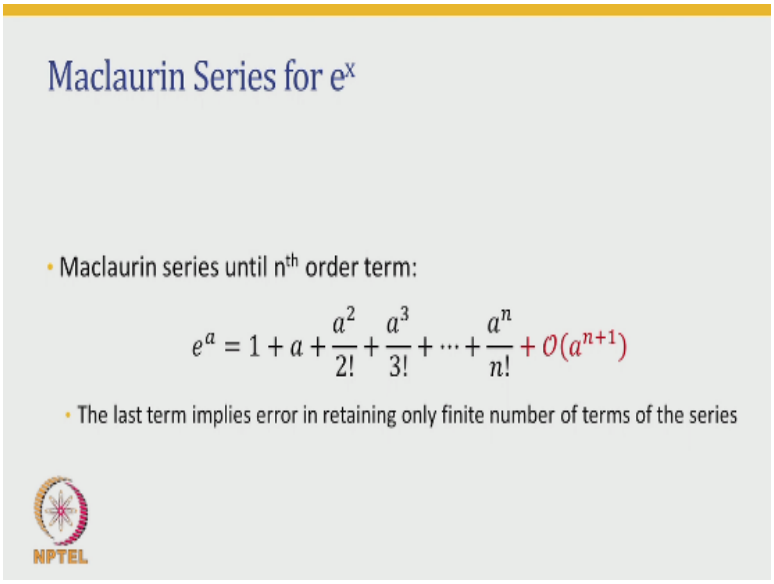
(Refer Slide Time: 01:29)



Taylor's series is some kind of a work horse, when it comes to computational techniques, it is used in order to derive a large number of computational techniques as you might have seen in a numerical methods or computational techniques course. The accuracy depends on how many terms are retained in order to derive that numerical technique.

So, greater number of terms we represent or larger is this value of n greater is going to be the accuracy of the numerical scheme that you are using for the computation purposes okay. So let us head back to the Maclaurin series.

(Refer Slide Time: 02:06)



So, what happens when we retain the Maclaurin series to the n order term, we are losing h ^ n. So, that is n+ 1 sorry, so that is the leading term when we comes to the accuracy so, when we are placing x with 0 and h with a, we are going to get this Maclaurin series for e ^ a .

And when we retain n terms, we are going to get the order of the accuracy to be a ^ n+1. So last term implies that the last error is of the order of n ^ n, a ^ n + 1, because we have retain only the finite number of series okay.

(Video Starts: 02:52) So let us head back to MATLAB and let us look at the Maclaurin series code that we had written earlier okay. So this is how we are written the code again. If you recall from module 1, this somewhat an inefficient way of writing a code in MATLAB. Because MATLAB, we can use the power of arrays, in order to calculate this in a more efficient way. So that is what first, I am going to do.

I am going to use the power of arrays in order to do that. So let us just delete this okay, a was 0. 1. Let us define a vector called vec = 1 to n okay. The various terms are going to be, terms are going to be equal to a ^ n are going to be the various terms. So a. ^vec are going to be the various terms divided by the factorial. The factorial we can calculate using the MATLAB command called cumprod c u m p r o d and cumprod vec okay. Now what is that going to do? So let us look at this over here. So a = 0 let us say a equal to 0. 1, vec = 1, 2, 3 okay.

So a^ vec is going to be a ^ 1, a ^ 2, a ^ 3 okay and we are using dot caret because it is an element by element power okay. Now let us see what cumprod of vec gives us. cumprod of vec is going to give us the first guy is 1, the second guy is 1 * 2 that is 2, third guy is 1 * 2 * 3 that is 6. Let us say if we were to do cumprod of say 1 to 5 this is what we are going to get.

The first guy is 1!, second is 2!, third is 3!, fourth is 4!, fifth is 5! Okay. So this is a vector which a ^ i. This is a vector which is i! and if we do an element by element division, we are going to get each element of the Maclaurin series okay. So we needed to do an element by element division. So we use the dot slash .And not just the slash. So these are going to be the various terms okay

and the expVal is going to equal to cumsum of the various terms. So what is e expVal going to be? (Video Ends 05:43)

ExpVal is the first guy is going to be a, the second guy is going to be a +a square, the third guy is going to be a + a square + a cube by 3! So on and so forth okay. So these are going to be individual term we have not yet added 1 so we will do that over here.

(Video Starts 06:02) So expVal is going to be 1 + cumsum of terms. So that those are going to be the first guy is going to be (Video Ends: 06:13)

The approximate value 1+ a, the second guy is going to be $1 + a + a^2 / 2!$ The third one is $1 + a + a^2 / 3!$, so on and so forth.

(Video Starts: 06:24) So this is what we are going to get okay. And this is the trueVal and error is the absolute value of difference between the trueVal and the expVal. So let us go and save this and run this in MATLAB and see whether we are going to get the same results as before or not. Clear all and clear the screen and let us run Maclaurin exponential and let us look at expVal. And we are getting 1.1, 1.105, and 1.1052 so on and so forth.

And error if we see, is similar to what we obtained last time. The only difference is in the last time when we did that, we had the term 0.1 as well which was a 0th order term. We do not need that term so we have gotten rid of that already okay. So we have now retained 1 term that is the first order term, the second term that is the second order term, third term that is the third order term and so on and so forth okay.

So this is what we did. When we wanted to get the Maclaurin series expansion of 0, e ^ 0.1? So what if we wanted this as for e ^ 0.01 as well, so in order to do that let us place a with 0.01. And let us see what we get back okay. And let us run this and we will see error again okay. So when we use it, I am sorry about that, this also needs to be changed and now let us run this.

And let us look at what error we get as we can see the error. When we have h smaller that means that value of a is smaller, the error has decreased quite significantly as well okay. So let us now do something is, let us say first, second and that order term for various different values of a. So let us look at the values of a equal to let us say 0. 1 okay.

And let us plot this particular 1 on plot. So let us say plot error, let us hold on so that particular plot is held. So let us look at now plotting how the error changes with respect to the various values of h. So what we are going to do is, for n = 3. Let us take various values of a as 0.1, 0.05, 0.02 and 0. 01 okay. These are the various values of a that we are going to use okay, aAll let us call back, we need vec, vec is going to be 1 to n.

So for i = 1 to length (aAll). ExpVal is this trueVal. So, these are the changes that we are going to do and a = aAll i okay. And we are going to plot aAll, error and we are going to name this axis also x label step size and y label error okay. So, this is what we are going to do over here okay. So, let us first run this and see that we are able to run this without error or not. If we do get an error, we will just go back and debug this code.

Yes, we are getting an error. We have an undefined variable err, so let us define err = a blank vector. And it now run this code and we are getting the error verses a step size okay. What we are seeing is, as a step size increases the value of error is also keeps increasing. So we do not want to look at the linear plot, we want to look at a logarithmic plot. So instead of saying plot, am going to say loglog.

So that will give me a logarithmic plot, a loglog plot over here okay. And this is what I am going to run okay. So let me run this so, I have gotten this logarithmic plot for the 3 particular values of n okay. And let me label that label n = 1, n = 2 and n = 3 sorry, it is not label, it is legend and show graph okay. So this is for n = 1, this line is for n = 2, and this line is for n = 3 okay.

So, the thing that you notice over here is that the as the step size increases, the error increases or as the step size decreases, the error is decreasing. Other thing that you will notice is, let us put a marker for this. Other thing that you will notice is that the slopes of the lines are different. They

are all approximately linear of straight lines. But the slopes of these lines are different and they are linear straight lines even when there are 4 points.

So they do look fairly straight to me. So, their slopes as you can see this one is a more gentle slope for n = 1, for n = 2 we have a steeper slope than n = 1 and for n = 3 we have an even steeper slope okay. So let us go and plot instead of plotting on a loglog plot, let us actually plot logarithm of aAll and error.

So let us say figure 2 plotting. Figure 2 will say plot log 10(aAll) and log 10(err) okay. And let us run this and yes we have gotten this plot also. So this is basically, we are instead of a loglog plot we have a linear plot. But we are plotting the logarithm of values. So let us click on this and query the various points that we have.

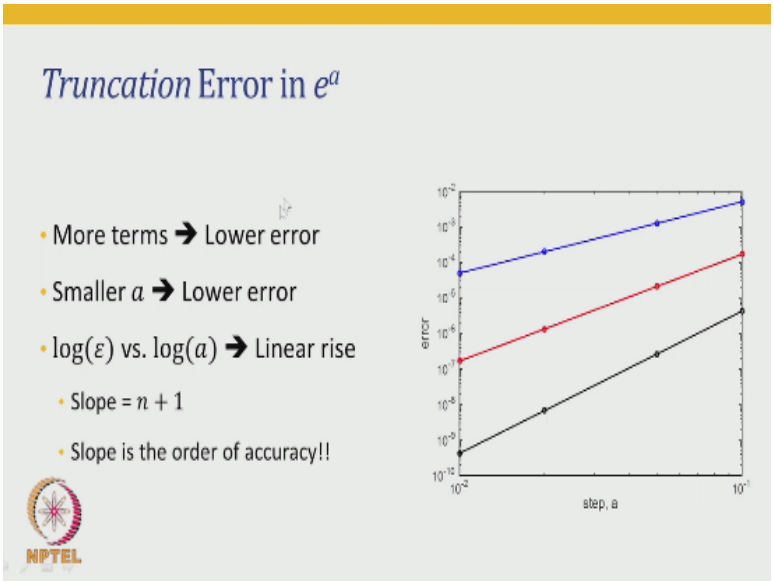
So the first is x = -1 and y = - 2. 28 okay and the last point over here is x = - 2 and y = - 4. 3. So let us get that slope so it is - 4.3 -2. .Let us go back and see minus 2.286, so minus and minus is going to become plus, divided by -2 +1. So, the slope for n equal to 1 is approximately 2. 01. Let us look at the case where n = 3. Here the y value is -5. 371. so that is minus 5. 371 + this value is -9. 379.

So, minus and minus will be become plus. Minus 9.379, whole thing is divided by -1 + 2 and the slope turns out to be 4. Likewise if we were to for this particular line, the slope will turn out to be 3. So this is a line with slope of 2, this is line with slope of 3 and this is a line with slope of 4. So if you see what we are getting is, slope of the nth order curve is n + 1. (Video Ends 16:36)

And that is what is the meaning of order of a ^ n +1 okay. So when we retained just these terms a ^ n+ 1 where n was equal to 1 becomes order of a ^ 2. So we are getting the accuracy is a square accuracy. When we retain a square by 2! term also the slope is of the slope is equal to 3 that is because the order of the accuracy is the step size ^ 4 and when we have a ^ 3 /3 term also included is a ^ 3 + 1 that is a ^ 4.

So the 3 lines have a slope of 2, 3 and 4 that is because the order of accuracy is n+ 1. So we retaining the first order term has a slope of 2, retaining second order term has a slope of 3, retaining third order term has a slope of 4. So, that is the meaning of this particular term order of accuracy. So what we are seen over here is that the order of accuracy increases or improves as we retain greater number of terms okay.

(Refer Slide Time 17:50)



So, to summarize what we see in truncation error is that we have greater the number of terms we have lower is the error. Smaller is the value of a, we have lower error and when we plot logarithm of error verses logarithm of a, we get a linear increase and the slope is n +1. The slope is nothing but the order of accuracy okay. With that I come to the end of this lecture. In the next lecture which also continuation of lecture 1. 2 where we are going to cover that round off error. And we are going to see tradeoff between round off error and truncation error. So thank you and see you in the next part of this lecture.