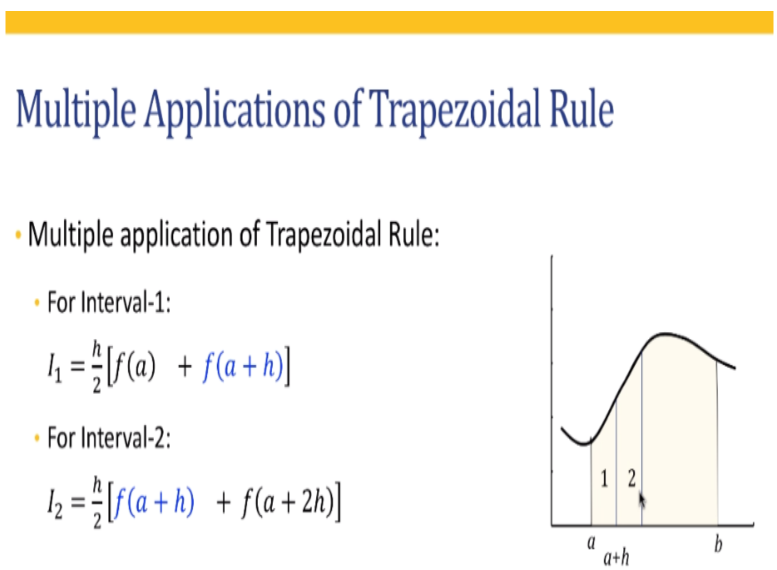
MATLAB Programming for Numerical Computation
Dr Niket Kaisare
Department of Chemical Engineering
Indian Institute of Technology, Madras

Module No.#02
Lecture No. #3.5
Numerical Integration – Multistep Trapezoid/ Simpson's Rule

Hello and welcome to MATLAB programming for numerical computations. We are in module 3. In this module we are covering numerical differentiation and numerical integration. In the second part of this module we have been covering numerical integration. In the previous lecture 3.4 we saw application of trapezoidal rule and Simpson's 1/3 rule. In today's lecture we are going to cover multiple applications of trapezoidal rule to find area under a curve.

(Refer Slide Time: 00.44)



If you recall from the previous lecture, integration is nothing but finding area under the curve f(x) between points a and b. We applied trapezoidal rule in order to calculate this area. And we found that the errors can be quite high in using a single application of trapezoidal rule. Instead what we can do is, we can divide this entire region into multiple intervals.

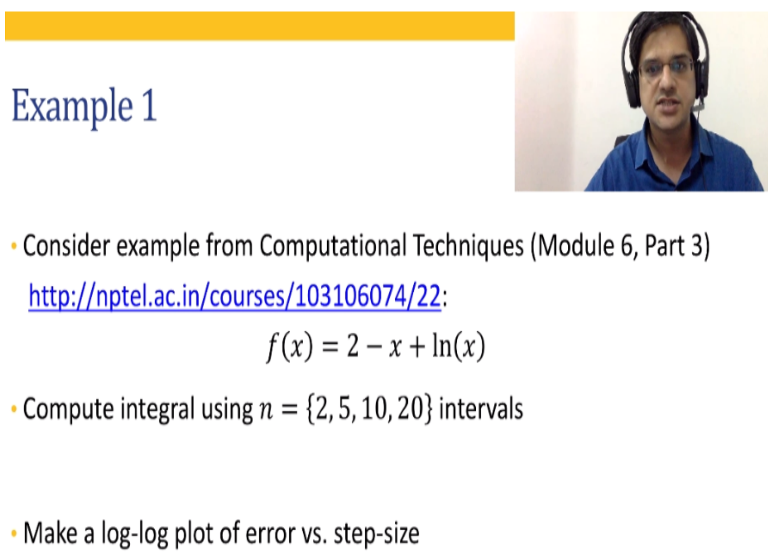
And then calculate area using trapezoidal rule for each of those intervals. Then we sum up all those areas in order to get net area using trapezoidal rule. How do we do that is as follows. For the first interval that will be that lies between a and (a+ h) the area under this region using the trapezoidal rule is given by h / 2 * f (a) + f (a+ h).

The second interval was from (a+ h) to (a+ 2h). For this interval our area using the trapezoidal rule is going to be h / 2 * f. f at (a+ h) + f at (a+ 2h). We keep doing that until the last interval. Once we get all these areas, our integral i is nothing but summation of all these individual areas or all these individual integrals that we have obtained using the trapezoidal rule.

The step size h is nothing but (b- a) / n. When we have n intervals we are going to have (n+1) points in the x axis. How that is, if the entire zone was just 1 single interval that means if we were doing a single application of trapezoidal rule, we will have 2 points a and b. If we are using 2 applications of the trapezoidal rule, we will have a, the midpoint and b that will result in 2 intervals, a to the midpoint and midpoint to b.

So 2 intervals will have 3 points on the x axis, 3 intervals will have 4 points on the x axis, 4 intervals will have 5 points and so on. And n intervals will have (n+ 1) points on the x axis. What we will do now is, go on to MATLAB and solve this using multiple applications of the trapezoidal rule. This is one of the 2 methods that I will show you in today's lecture for using multiple applications of the trapezoidal rule.

(Refer Slide Time: 03.27)



Let us consider the example that we had solved in the previous lecture. That was to find the integral of 2- (x+ ln x) okay. So let us go onto MATLAB and do this problem. (Video Starts: 03:39) What we will do is, we will open the file that we had use in the previous lecture. That file we called as numIntegral. This was to calculate the integral of 2- (x+ ln x) using a single application.

What we will do is, we will change the 2 multiple applications of trapezoidal rule. I will save this as a different file. Let us call this as say multiStepIntegral okay. So the problem setup a =1, b =2. You want to integrate this function from 1 to 2. trueVal is this and let us say the number of sets n that we want to take initially. Let us say it is 2, right. That is what we said.

We will do okay. So trapezoidal rule multiple applications and let us delete the section about Simpson's 1/3 rule and displaying the results. And this is what we are left with okay. So n is 2 for n = 2 or h was (b-a) / n. So what h is (b-a) / n and we had used myFunInt to calculate all the functions.

What we will instead do is, we will again keep using myFunInt but we want to do this for all the data points that means for all the x values x1, x2, x3 up to x n+ 1. So let us define this xVecor. xVec = a. To a in steps of h 2 b. So this we will result in (n+ 1) dimensional xVector okay. What we want to do is, calculate the functions f at x1, f at x2 and so on.

We will do this using our myFunInt. Let us open myFunInt. If we give our xVec as our x as the vector, what we see is that this will result in f evaluated at all these points. This is because of this function is already vectorized (2- x) is going to give us a vector which is of (n+ 1) size + log of x log of x is also going to give us a vector of (n+ 1) size for (n+ 1) size vector was added to (n+ 1) size vector will give us again (n+ 1) size vector.

So we do not need to make any changes myFunInt okay. So we can just say fVec, the vector of function value is nothing but myFunInt (xVec). And that should be sufficient and let me delete this one. So what let us do now is, just check and verify that this is indeed works. So let us run this okay. And let us also myFunInt at 1, myFunInt calculated at 2, and myFunInt calculated at 1.5. So these are the3 points when we have 2 intervals. Let us check what fVec looks like.

Let us type fVec, the first guy is 1 which is here and the last guy is 0.6931 which is over here and the middle guy is 0.9055. So this is indeed what we expect. So let us clear all clc. Now what we have fVec is f1, f2, f3 and so on up to f (n+ 1). What we want to do now is, to calculate i1 i2 and so on up to i (n+ 1).

The way we are going to do this is, we will do this in a for loop. for i = 1 to n. So these are the n intervals okay. Iinterval = zeroes because there are n intervals zeroes of size of n, 1 okay. The integral for the first interval. So Iinterval for the Iinterval. This is nothing but (f1+ f2) * h/ 2, for the second interval is (f2+ f3) * h /2, third interval (f3+ f4) * h / 2.

So on ith interval if it is going to be (fi+ fi+ 1) / h / 2. So that is what we will put over here is, h/ 2 sorry, multiplied by h/2, h/2 * (fVec i+ fVec i+ 1). End and I_trap, I_trap using the first method is going to be just the sum of all these values .So I will just say sum Iinterval err1= abs (trueVal- I_trap 1).

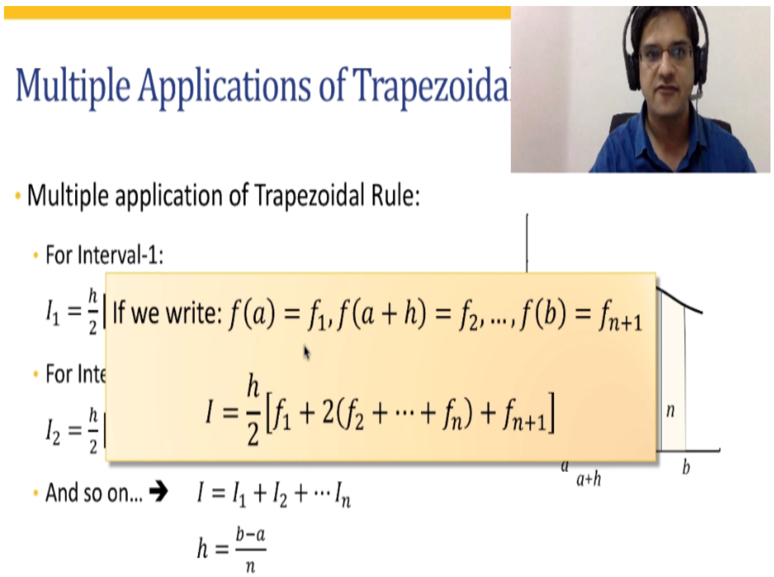
For h=num2str h. error= err1 sorry, num2str err1 okay. So that is what we have. Let us save this and that is run. Hopefully it will run without giving us any errors okay. Let us go to MATLAB command prompt and see what the results are. So the integral value is 0.8760 and for step size of 0.5, the error is 10 ^ - 2 okay.

Let us now change from n = 2 to n = 20. So by doing this the step size is going to decrease by 1 order of magnitude. So let us run and see what happens to the error. So when we will run this by decreasing the step size by 1 order of magnitude, the error has decreased by 2 orders of magnitude. The error has gone from 10 ^ - 2 to the 10 ^ - 4.

Let us decrease the step size by 1 more order of magnitude. Let us save this and run this. What we will see is that the error will decrease further 2 orders of magnitude. So when we divided the step size by 10, the error decreases by factor of 100. When we divided the step size further by a factor of 10, the error further decreased by a factor of 100. If you recall, what this means, this means nothing but the method is h to the power 2 accurate.

It is the second order accurate method from the point of few of global truncation error. (Video Ends: 11:57) What I will do now is try to solve this problem using a second way and which is going to be an easier way from writing a MATLAB program. What we are going to do is, we will just add up i1, i2, i3 and so on up to i n. So you will have h/ 2 multiplied by f (a) + 2* f (a+ h) so on and so forth.

(Refer Slide Time: 12:15)



So finally the result that we will get is, if we were to write f(a) as f1, this guy is f2 ,this guy is f3 up to f(n+ 1), the result that we are going to get is, i is h / 2 * f1+ 2 times summation of all the middle guys + 1 times f (n+ 1). It is very easy to get this formula. You can very quickly try it out and get this formula. So what will go, now go and do now in a MATLAB is.

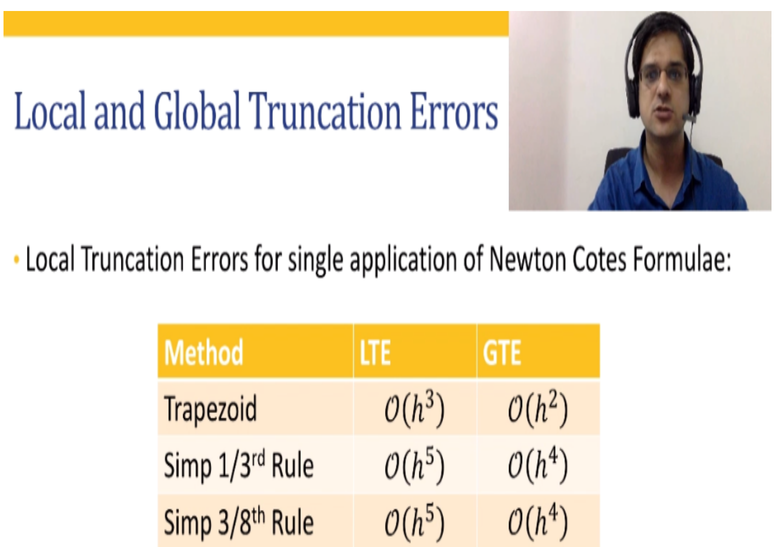
(Video Starts: 12:50) To solve this using this direct formula from multiple intervals using the direct formula okay. Using the direct formula I_trap2 is nothing but h/2 multiplied by f1 which is nothing but fVec1, fVec1 + 2 * summation of all the middle guys+ 2* summation fVec2 to fVec from 2 to n okay, + fVec n + 1 okay. That is going to be our I_trap2 using the direct method.

So now what this guy does, this guy will extract the subpart of the vector from the second to nth value. And we just want to sum it up as we have written over here. And multiplied that with 2 and add f1 and fn+ 1. And that will give us the desired result will be multiple it with h /2. So display I_trap1. I will also display I_trap2. err2 = abs (trueVal- I_trap2).

Yeah for h = num2str err2. Let us save this and let us run it for again for n = 200 okay. So as you can see from both these methods, the result is the same as we can see displayed over here and the errors that we can see between the trueVal and the numerical values are also the same okay. (Video Ends: 15:32) So what we have done really is to calculate a function, sorry, calculate a integral using multiple applications of the trapezoidal rule.

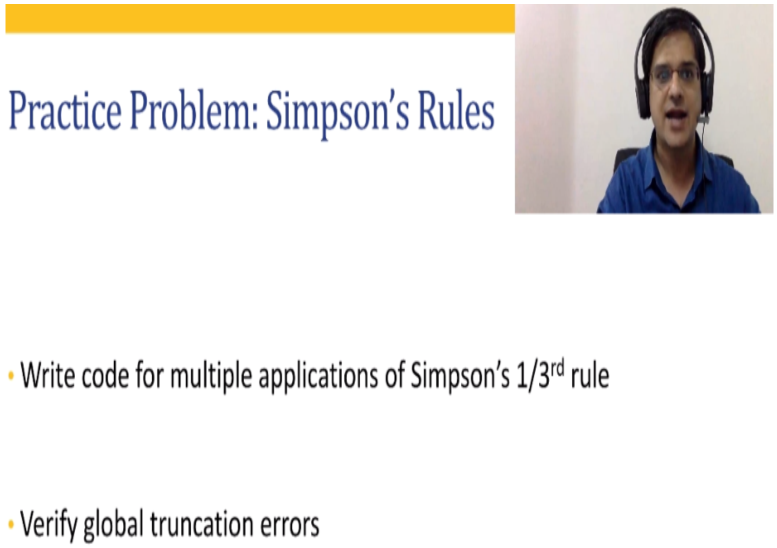
The first 1 was by using by calculating the integral for each of those intervals and summing that up, the second and the more efficient way and the method that I will recommend finally is to calculate the overall result. Pre calculate the overall result and then code that in MATLAB as shown over here okay. So that is what we have primarily covered in today's lecture and we have used that for the example of f(x) = 2- (x+ lnx).

(Refer Slide Time: 16:21)



What we also did was, observe that for the trapezoidal rule. The global truncation error is h to the power 2. So it is the second order of the accurate method. When we use multiple implementation of the trapezoidal rule in contrast local truncation error was, h to the power 3 for trapezoidal rule.

(Refer Slide Time: 16:44)



What you can do as the practice problem is, you can solve the same example by writing a code for multiple applications of the Simpson's 1/3 rule. And you can then go ahead and verify the global truncation error and see for yourself that reducing the step size by 1 order of magnitude indeed reduces the errors by 3 orders sorry, by 4 orders of magnitude in the Simpson's 1 third okay.

So with that I come to the end of this lecture. In the next lecture we are going to cover integration using 2 MATLAB functions trapez and quard. So we will cover them in the next lecture. Thank you and see you in the next lecture goodbye.