

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

Module No. #07

Lecture No. #7.1

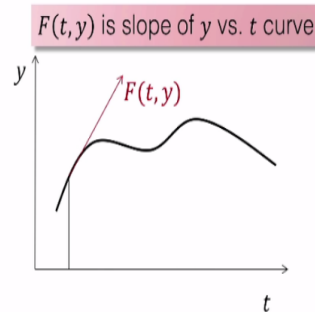
Ordinary differential equations – Initial value problems – Introduction and Euler's Methods

Hello and welcome to MATLAB programming for numerical computations. We are in week number 7 in module 7 we are going to cover ordinary differential equations initial value problem. In this module we are going to tackle methods for solving ordinary differential equations. Lecture 7.1 we are going to go over introducing ourselves to ODE and take up 1 example and solve it using Euler's method. Euler's method is one of the simplest methods for solving ODES.

(Refer Slide Time: 00:48)

Introduction: ODE – IVP

- Given: $\frac{dy}{dt} = F(t, y)$
and initial condition: $y(t_0) = y_0$
- Aim is to find: $y(t_i)$



Discussion and theory for numerical ODE-IVP:

Computational Techniques, Module-7 (<http://nptel.ac.in/courses/103106074/25>)

So let us let us look into the ODE problem. So introduction, what is an ODE. ODE is when we want to solve an equation of the form $\frac{dy}{dt} = f(t, y)$ subject to initial conditions, y at some time $t=0$ is given us y_0 . We start with that initial condition and march forward in time at each time finding the value of y at that corresponding time instant. In this way we will get the overall curve y as a function of t which i have plotted over here.

Now the function $f(t, y)$ is nothing but the slope of this y versus t curve at any point in the domain okay. So if we are going to start with a y_0, t_0 we will then march on and find $y_1, t_1, y_2, t_2, y_3, t_3$ so on and so forth is what we are going to actually find. Eventually we will develop the entire curve y as a function of t . And that is the actual solution that we are intending to find using a numerical scheme.

The discussion and theory for numerical methods for solving ODE-IVP initial value problems is discussed in computational techniques course module 7 the link for which is given over here. In this lecture or for that matter in this course we are not going to cover any theory or discussions with regarding ODE-IVP. Rather we will use MATLAB in order to solve the initial value problems. Let us get started with 1 of the simplest methods which is known as the Euler's method. Before we go on to Euler's methods ah method let us look at how we would want to solve this ODE.

(Refer Slide Time: 02:44)

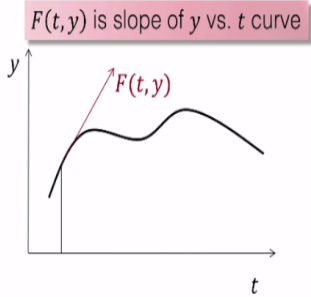
Introduction: ODE – IVP


- Given: $\frac{dy}{dt} = F(t, y)$
and initial condition: $y(t_0) = y_0$

$$\lim_{h \rightarrow 0} \frac{y_{i+1} - y_i}{h} = F(t, y)$$

$$y_{i+1} = y_i + hS_i$$

Numerical methods for ODE-IVP:
Use "best" estimate of slope S to obtain $y(t)$





We can write dy/dt in a finite difference form $(y_{i+1} - y_i) / h$ where h is known as the step size okay. If we remove the limit h tends to 0 and take it as a very small value, we can rewrite this equation in the form $y_{i+1} = y_i + h * F(t, y)$. Now if the function f is calculated at y_i and t_i we get what is known as Euler's forward difference or sorry. Then we get what is known as Euler's explicit method.

In general numerical methods for ODE-IVP we try to find certain slope S_i which is the best estimate that can be used in order to march forward along this curve. So, S_i is going to be dependent on y , y 's and t 's and we are going to find that particular S_i such that we get y_{i+1} as close as possible to the true solution.

(Refer Slide Time: 03:59)

Example



- ODE-IVP:
 - ODE: $y' = -2ty$
 - Initial: $y(0) = 1$
 - Analytical solution: $y(t) = \exp(-t^2)$
- Euler's Explicit Method:
$$y_{i+1} = y_i + hF(t_i, y_i)$$

In the simplest case we can consider S_i as nothing but f calculated at $T(i), y_i$ and that is what Euler's explicit method is. $y_{i+1} = y_i + h * f(t_i, y_i)$ is the formula that we will use for Euler's explicit method. At the initial time we have y_0 t_0 we can use this in order to compute now y_0 is known, t_0 is known and y_0 is known therefore f at t_0 y_0 is known. We can use this to compute y_1 . Once y_1 is known and t_1 is known we can use them to compute y_2 .

That we can use to compute y_3 so on and so forth okay. So this becomes very simple straight forward method in order to solve ODE initial value problems. The problems with Euler's explicit method are 2. The first problem is that the method is not very accurate to tackle this problem. We are going to talk about Runge- Kutta method in lecture 7.2 and thereafter, the second problem is regarding the step size that we need to choose in order to get the system to be to be stable.

Stability of numerical methods something that I will introduce in this lecture but that this beyond the scope of this course. I will introduce the concepts of stability and so on, mainly for the sake of completeness although it is not strictly a part of this MATLAB course okay.

(Refer Slide Time: 05:33)

Euler's Implicit Method



$$y_{i+1} = y_i + hF(t_{i+1}, y_{i+1})$$

Thus, use nonlinear solver (such as fsolve) to solve:

$$\underbrace{y_{i+1} - hF(t_{i+1}, y_{i+1}) - y_i}_{f(y_{i+1})} = 0$$

Unlike explicit method, Euler's implicit method is "globally stable"

Computational Techniques, Module-7 Part-5 (<http://nptel.ac.in/courses/103106074/29>)

So the example that we will solve is $dy/dt = -2ty$ with the initial condition $y_0 = 1$. If we solve this analytically the solution we are going to get y at anytime t is going to be equal to e exponential of $-t$ square. What we want to do is use Euler's explicit method and compare the solution using Euler's explicit method to the analytical solution that we have written down over here. Let us go on to MATLAB to solve this problem using Euler's explicit method okay.

(Video Starts: 06:09) Let us create a file called Euler's explicit, solve ODE-IVP. So we have put down some parameters t_0 , y_0 was what was given to us let us say that we want to solve it until time $t_{\text{End}} = 5$. And let us say the step size that we are going to be use is 0.1. So put this down ah all of this over here, initializing, okay. I have put a transpose over here because we want our t and y to be column vectors.

y is the solution vector and that we will initialize as zeros of size $(N+1)$. And y_1 which is the first value is equal to y_0 . Keep in mind you would have realized by now that MATLAB does not have array location 0. The array start with index 1 and therefore we will use the index 1 for time $T = 0$ okay. In addition to this we also need to specify the value of n . So n is going to be nothing but $t_{\text{End}} - t_0, t_0 / h$ okay.

Let us save and run this to see that we do not get any errors okay. So we are not getting any errors over here let us now continue solving using Euler's explicit method for okay. f_i is going to be nothing but $-2 \cdot T(i) \cdot y(i)$. That is our f_i and $Y(i+1)$ is $y(i) + h \cdot f_i$. So that is our Euler's explicit method. So let us go and look at us $Y(i+1)$ is $y(i) + h \cdot f(t_i, y_i)$ okay. Our f was $-2ty$. So f at time i , is going to be nothing but -2 multiplied by the vector t , i th location multiplied by vector y , the i th location okay.

That is going to be our f_i . $Y(i+1)$ is nothing but $y(i) + h \cdot f_i$ okay and end, this should do. So that completes our Euler's explicit method. Let us plot this, `plot(T, Y)`. So let us hope that this runs without an error run. And this is the numerical solution using Euler's explicit method. Let us also compute the errors. So plot results and obtain errors $y_{\text{True}} = \exp(-T^2)$ correct yeah. $\exp(-T.^2)$. We need dot because this element by element squaring and $\text{err} = \text{abs}$, absolute value ($y_{\text{True}} - y$) okay.

So that is what we have, let us clear our screen and that is called Euler's explicit here, solves this and let us look at error err let us call `max`, max value of err and the `max(ERR)` is 0.035. If h was reduced to 0.01 let us run this okay. And see `max(ERR)` okay and the maximum error went from 0.03 to 0.003. So by reducing h by 1 order of magnitude the maximum error also reduced by 1 out of magnitude.

Let us save this by reducing h further and we find `max(ERR)` and the `max(ERR)` fell by another order of magnitude. We now instead of $3 \cdot 10^{-3}$, we now have $3 \cdot 10^{-4}$. So from here you would have probably guessed start the order of accuracy for the global truncation error rather for Euler's explicit method is the order of h^1 . This is something that will cover in a little bit more details in 1 of the later lectures in this module okay. (Video Ends: 12:42)

Okay so let us go back, change our h to our old value and let us go back to our problem okay. So this was the example that we solved using Euler's explicit method. Now in Euler's explicit method our f we calculated at (t_i, y_i) . We could have very well calculated f at $T(i+1)$, $Y(i+1)$ instead of (t_i, y_i) . If we were to do that we will get what is known as Euler's implicit method.

(Refer Slide Time: 13:03)

Euler's Implicit Method



$$y_{i+1} = y_i + hF(t_{i+1}, y_{i+1})$$

Thus, use nonlinear solver (such as fsolve) to solve:

$$\underbrace{y_{i+1} - hF(t_{i+1}, y_{i+1}) - y_i}_{f(y_{i+1})} = 0$$

Unlike explicit method, Euler's implicit method is "globally stable"

Computational Techniques, Module-7 Part-5 (<http://nptel.ac.in/courses/103106074/29>)

In the explicit method t_i and $Y(i)$ were already known and you could directly calculate this and assign it to $Y(i+1)$. However that is not possible in Euler's implicit method because our $y(i)$ itself depends on y sorry, our $Y(i+1)$ depends on itself through this function f . So this is a nonlinear equation which we need to solve. We can do it by using a nonlinear solver such as fsolve. fsolve is something that we covered in module 5 of this course.

So if we were to rewrite this we will rewrite this as $Y(i+1) - h * F \text{ at } (T(i+1), Y(i+1)) - y(i) = 0$. And this we can assign it to fsolve and we can solve okay. Now Euler's implicit method is not a part of this particular course. (Video Starts: 14:05) However for the sake of completeness I have already created Euler's implicit code and I will display that clear all, close all, clc edit Euler's implicit okay.

This is the code that I have created earlier for same values of y_0 , t_0 , t and h everything else is remaining same, now our f is calculated at $T(i+1)$, $Y(i+1)$. So first I will calculate $T(i+1)$ okay. $Y(i+1)$ is yet unknown and therefore we need to use fsolve in order to solve this okay. So $Y(i+1)$ is the unknown quantity so this is unknown this over here is unknown. So we have used what is known as anonymous functions in MATLAB in order to give this particular f of x . (Video Ends: 14:58)

So that f is nothing but $y - h * (-2 * t * y) - Y(i)$. So this is what was written over here okay. (Video Starts: 15:14) And the initial guess that we are going to give is capital $Y(i)$ okay an anonymous function is another thing that we have not covered in this in this course. We are not going to cover anonymous function in MATLAB in this particular course so these are 2 concepts which are actually beyond the syllabus if you will okay.

I have just shown it for the sake of completeness, so that you can understand how we can do this. Let us also plot and obtain errors as before. Plot (T, Y) , $ERR = \text{abs}(y_{\text{True}} - y)$. We also need y_{True} . y_{True} is $\exp(-T)$ and there is a cap dot cap sorry, instead of cap okay. This is basically what we have done for Euler's implicit. Let us run this and hope that we will not get any error okay.

One thing you will notice is that Euler's implicit method takes a little bit longer than Euler's explicit method in order to run that is because we need to solve f solve at each line okay. And the figure looks approximately similar what we had in Euler's explicit method. Let us go and find out the max (ERR). Max (ERR) is again of the order of 0.01. If you recall in Euler's explicit method also with $h=0.1$. The error we got was 0.03 okay.

So the order of accuracy of Euler's implicit method is similar to that of Euler's explicit. (Video Ends: 17:06) So now the question is, if the order of accuracy is similar between Euler's implicit and explicit and implicit method is much more difficult to solve than Euler's explicit method, why are we interested in implicit methods in the first case. The reason why implicit methods are very useful is because implicit methods are globally stable okay.

This was covered in the computational techniques course module 7.5 link for which is given over here. Basically what that means is, if we start increasing the step size, we will need to sometimes increase step size in order to speed up the overall computation. If we start increasing the step size there will come threshold value of h beyond which Euler's method explicit method will become unstable. This means that you will not get a stable solution.

Euler's implicit method on the other hand is globally stable which means you can choose any value of h of your liking. And Euler's implicit method will remain stable. The stability is the

reason why implicit methods are popular especially for solving tough ODE problems okay. (Video Starts: 18:17) Let us kind of let us now show what loss of stability means.

Okay, we will just clear this and we will go back to Euler's explicit method. So now if h instead of 0.1 let us say if we were to take 0.25. And we run this okay. We get the solution. Now the solution qualitatively looks similar to that we saw earlier. Only thing is it looks more bumpy and that is because the errors are higher with step size of 0.25. While the errors are higher the solution is still stable.

Now let us increase this to 0.5. Let us run Euler's explicit. And we see that is something funny that is happening. There is nothing that happened in the first 2 steps rather. In the third step the solution directly went from 1.0 to 0.0 okay. And that is the reason why that happens is $h=0.5$ is at a border of the stability. If we increase our h beyond 0.5, we will make the overall solution unstable.

So let us go and increase this to $h=1$ and we will see what instability. Let us run this okay as you can see the value of $T(i)$ sorry, value of $Y(i)$ at $t=5$ is greater than 100. And this particular series solution is diverging okay. The reason is because Euler's explicit method in this particular example has a limit of stability at $h=0.5$. Any value of $h > 0.5$ will result in unstable solution.

Let us actually go and change h to 2.5 and see what happens okay. And our solution has gone to 12 and if again same thing if we were to change 5 we will see a similar behavior. Let us change to 10 see what the behavior is okay. And you see we have gone that the solution has reached something like $-1 * 10^4$. So we have very clearly gone this taken the solution to an unstable range. Let us compare that with Euler's implicit.

We will use $h=2.5$ and $t_{\text{End}} = 10$. Keep things the same as before $t_{\text{End}}=10$, $h=2.5$. Let us save this okay. And let us run this and see okay. Although this solution is very approximate, we do not have a very good approximation using ODE using the ODE solver Euler's method. We still have the solution to be stable. The reason is that Euler's implicit method is globally stable. (Video Ends: 21:36)

So we can choose the size, the step size h as large as we want. And the solution will still remain stay okay. So with that I will come to the end of lecture 7.1. What we covered in lecture 7.1 is introduce ourselves to ODE initial value problems, took up 1 example and solve it using Euler's method. Euler's method was the simplest method that we could choose.

And we compare Euler's implicit and Euler's explicit. In the rest of this module we are going to work with more accurate explicit methods known as the Runge- Kutta family of this in the next lecture. I am going to introduce the second order Runge-Kutta method. So that is our plan for remainder of this module. See you in the next lecture. Thank you.