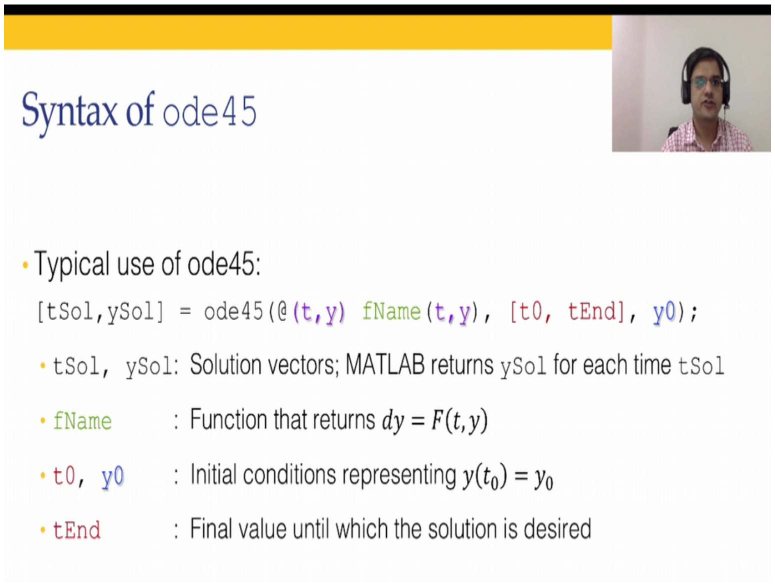
MATLAB Programming for Numerical Computation

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

Module No. #07
Lecture No. 7.3
Ordinary differential equations – Initial value problems MATLAB ode 45 Algorithm

Hello and welcome to MATLAB programming for numerical computations. We are in module 7. In this module, we are covering ode initial value problems. We are in lecture 7.3, and in this lecture, we will consider a MATLAB algorithm, known as ode45 in order to solve ode initial value problems. Ode45 is a kind of a workhorse in MATLAB. It is the first algorithm that you go to in order to solve ode problems okay.

(Refer Slide Time: 01:04)



Syntax of ode45

- Typical use of ode45:

```
[tSol,ySol] = ode45(@(t,y) fName(t,y), [t0, tEnd], y0);
```

  - `tSol, ySol`: Solution vectors; MATLAB returns `ySol` for each time `tSol`
  - `fName` : Function that returns $dy = F(t,y)$
  - `t0, y0` : Initial conditions representing $y(t_0) = y_0$
  - `tEnd` : Final value until which the solution is desired

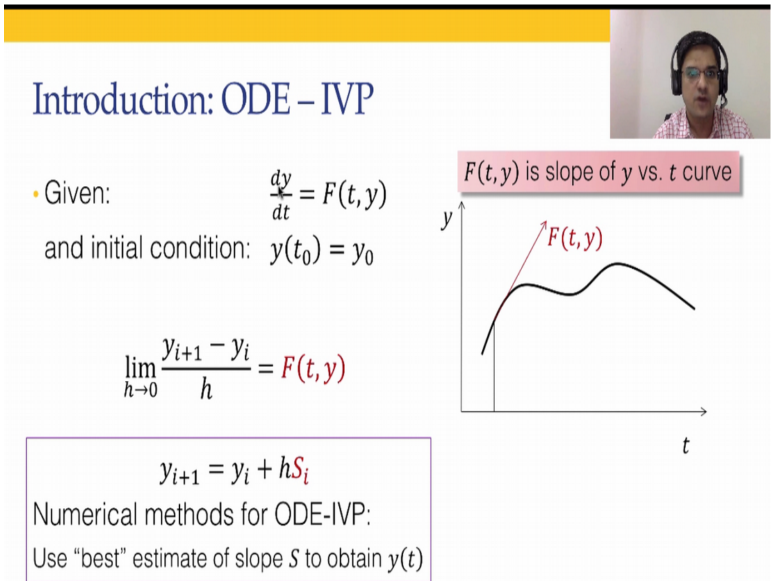
Let us go and look at the syntax of ode45. So, typical use of ode45 is as follows okay. Now, as we have seen in the previous lecture in rk2, we obtained the solution yi, at various values of ti okay. Likewise, ode45 also, returns the values of yi at various values of ti, between t0 and tEnd. The intermediate values at which computations are done, are calculated internally by ode45. Ode45 uses a Range-Kutta fourth order method, an rk4 method, recall that we have covered rk2 method in the previous lecture.

Not only that, not only does it cover and uses an rk4 method, it uses what is known as adaptive step size rk4 method. What that means is, that the step size h is not a constant value, recall that in the previous lecture, we took the step size h= 0.1 and we later on change that to 0.01 and so on okay. We had predetermined what the step size is going to be.

ode45, does not predetermine what the step size is going to be. But at each step, it calculates what the optimum step size is, in order to minimize the error, to get the error among, within the tolerance that, we have set for ode45. These are rather advanced concepts, which we are not going to cover in this particular course. However, I wanted to point this out to you, because this is something that you will come across again and again, if you are going to use ode45 or any other MATLAB odes solvers.

All the solvers in MATLAB are variable step size, or adaptive step size solvers where, MATLAB internally decides the step size to get the errors within the required tolerance intervals. Okay, so the typical way how we will use ode45 is ode45 at (t, y) recall, we go to our previous lecture, lecture 1 okay.

(Refer Slide Time: 03:20)



Introduction: ODE – IVP

- Given: $\frac{dy}{dt} = F(t,y)$

  and initial condition: $y(t_0) = y_0$

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

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

Numerical methods for ODE-IVP:

Use "best" estimate of slope S to obtain $y(t)$

$F(t,y)$ is slope of y vs. t curve

We were given ode in the form of dy /dt = f (t, y) okay. So, the function, f is a function of both t and y and we have to provide that function. So, that function takes in 2 arguments, time t the dependent variable y. And returns, the vector dy okay. In this case, because we are interested in looking at single variable problem, it will return just a scalar okay.

So ode45 gives (t, y) as the arguments, which are passed on to the function fName okay. The initial time t0, and the final time tEnd, and the initial solution y0, are also passed on to the solve okay. So the fName, is the name of the function that returns dy /dt as a function f (t, y). (Video Starts: 04:24) Let us look at the examples that we have been covering so far, in this module using ode45. Let us look at rk2heuns method. This was our code for rk2heuns method. You will now solve it, using ode45. Edit runODE, solve ode ivp using okay.

So t0 = 0, y0 = 1, tEnd = 5. We do not need h over here okay, solving using ode45. Let us look at the syntax, the syntax is, (tSol, ySol) = ode45 (tSol, ySol) = ode45 @ (t, y). So t and y are the variables that are passed on to the function that ode45 code is going to call. Let us call this function, my firstODEfun (t, y), again the order of t and y is important, it is it has to be (t, y) and not (y, t). This is one area where some students make a mistake, that mistake is not very common but some students do make that mistake.

Next one, is in square brackets, we need to give that tspan, the time span (t0, tEnd) okay, next we need to give y0, y0, that should be a and plot okay. So, let us run this and see what happens. Okay, it did not run, probably there were some error, undefined function firstODEfun. Why did we get that, because we have not yet defined our function firstODEfun? So let us go ahead and do that. Again, see when you run MATLAB functions, in order to solve ode or to solve nonlinear equations, solve whatever, you get a bunch of errors. And the errors with trace back to the calling functions.

Let us make that function, firstODEfun function dy = firstODEfun (t, y) okay. And dy is nothing but -2*t*y okay, that is all there is to it, okay. And if we save this, we run this, we will be able to solve this particular problem. So, let us not run firstODEfun, but run odescript okay. So let us clear all and run odescript okay. And we get the solution as seen over here and the solution smoothly goes on to y = 0 okay. That is the solution that we get.

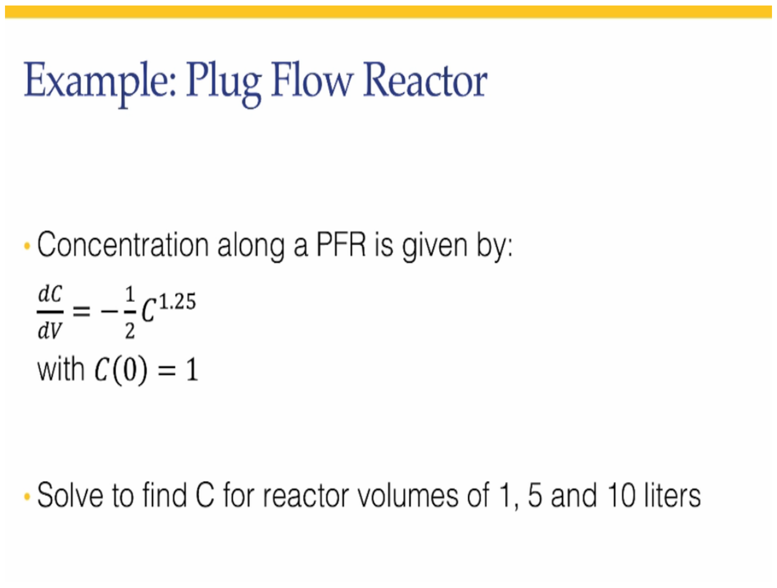
Let us go on to firstODEfun and may be make a small mistake somewhere. So, let us make this as t1. Here you want to pass how the errors look like, let us run ode and see how they are going to look like okay. So undefined function or variable t1, the error is noticed in firstODEfun on line number 2. So, let us go to firstODEfun at line number 2 and see what that error is t1 is undefined, because our input variable v sec was t okay.

Keep in mind, our t and y are basically dummy variables. So, could have very well named this as t1 and that would not be a problem at all okay. For that matter we could have named this as MATLAB and that would also not be a problem course okay. Let us call this that is also not going to be a problem at all. So, let us run this, run ode and see whether we get the solution or not clear all okay. The first ode we have just called the first argument as MATLAB, the second argument as course, of course it makes absolutely no sense.

But, I am showing you, again this is something that some people do get stomp on and that is the reason why, I am showing you this particular crazy example. Let us run this and we get the exact same solution okay. This is demonstrate to you is that, the arguments of this what you call them is not important. The order of these arguments is very important. This has to be y and this has to be t.

You cannot replace t and y, you cannot switch t and y, the t argument has to come first and y argument has to come second, you can call it by whatever name you wish to call it and it will not be a problem, as long as the order of the arguments is maintained the same okay. Let us save this and just for our sake, we will run this once again and we see that we get same solution okay. So, that was the example, for solving ode of the type -2(t, y) okay. (Video Ends: 11:05)

(Refer Slide Time: 11:07)



Example: Plug Flow Reactor

- Concentration along a PFR is given by:

$$\frac{dC}{dV} = -\frac{1}{2}C^{1.25}$$

with $C(0) = 1$

- Solve to find C for reactor volumes of 1, 5 and 10 liters

Let us take one more example, and this example was covered in computational techniques course in module 7 of the computation techniques lecture course. And this one is to find the concentration along a plug flow reactor. dc /dv is going by -1/2*c ^ 1.25. With the initial condition c at v =0 was equal to 1. So we want to solve to find c at values of volume 1, 5 and 10.

(Video Starts: 11:36) So let us go over here and editPFRode, solve plug flow reactor simulation okay. So, our c0 was equal to 1.0, our v0 was equal to 0, our Vend was we had 1, 5 and 10 okay. And what else do we need, that is pretty much it that is all that we need. We will need, vSol, cSol okay. C is the dependent variable, V is the independent variable equal to ode45 @ the independent variable t, y. t is the independent variable that is v, c pfrFun okay. We can call this as (t, y) also. We just want to call this as V, V, VSpan, c0. Okay we will come back. Using ode45 okay so over here clear all and help ode45 okay.

So, ode45 TOUT, YOUT = ode45 (ODEFUN that is function name, TSPAN, y0). TSPAN = (t0 TFINAL) integrates from time t0 to tfinal. To obtain solutions, at specific times t0, t1, t2 up to tfinal, all increasing, all decreasing okay, for our purpose we will just use increasing. Use TSPAN = (T0, T1 upto TFINAL) okay. So, what we want is, we want a solution at 1, 5 and 10. So our VSpan is going to be v0, 1, 5 and 10 which is nothing but Vend. So, that is going to be our VSpan okay. So we have this we save this okay.

And next what we need to do is we want to create the pfrFun. So let us new function dc = pfrFun (V, C). dc = -1/2 *c ^ of 1.2, 0.5 * C ^ 1.25. That is it and we need to call it as pfrFun.

So just check, pfrFun is wanted to be V and C are the order of the arguments. And so, and vspan is v0 followed by the values of v at which we want the solution okay. So let us run this and we have got the solution. So vSol you will see is 0, 1, 5 and 10. And cSol is the concentration at these values. We start with the concentration of 1.0 the concentration falls to 0.6 where, the volume is 1 liter and the volume is 5 liters, the concentration has fallen to 1.4 and when the volume is 10 liters, the concentration has fallen to 0.039 okay. So, this is how we will use ode45. (Video Ends: 16:02)

With that, we come to the end of lecture 7.3. What we covered in this lecture is, method to use MATLAB ode 45 code, to solve ode initial value problem, we used two different examples, to use for ode45. And using the two examples, we showed how simple it is in order to run ode45 for single variable problem. With this, we come to the end of this lecture 7.3 and I will see you in the next lecture. Thank you bye.