Module 2 : Equipment and Stability Constraints in System Operation

Lecture 7a : Numerical Solution of Differential Equations

Objectives

In this lecture you will learn the following

- How does one simulate the behaviour of a dynamical system numerically ?
- Runge-Kutta Fourth order method

Why numerical methods ?

The reason we have taken a minor diversion here is that we wish to know the techniques to "understand" how systems described by differential equations behave. For very simple systems like the one below:

$$\frac{dx}{dt} = ax$$

It is clear that x=0 is an "equilibrium" solution of the system, since, the LHS of the above equation equals zero at this value of x. In general, the behaviour of x, when its value at time t = 0 is x(0) is given by :

$$x(t) = e^{at}x(0)$$

Verify that the above "solution" satisfies the differential equation. Note that the solution is in terms of a well known exponential function. If a > 0 , then the magnitude of x(t) keeps increasing with time if x(0) is not zero. On the other hand, if a<0, then x(t) tends to go to zero as time progresses. Thus, we are able to gain an insight into the behaviour from the solution given above.

However, it turns out that for many systems, it is not possible to write down the solution in terms of well understood simple functions. This occurs when the RHS of the differential equations have terms which are nonlinear or time variant functions of the variables. For example, the behaviour of rotor angle and speed deviation for a synchronous machine is described by the non-linear differential equations:

$$\frac{d\delta}{dt} = (\omega - \omega_o)$$

$$\frac{d(\omega - \omega_o)}{dt} = \frac{\omega_B}{2H}(T_m - \frac{EE_b \sin\delta}{X})$$

To understand the behaviour of such a system one has to turn to the *numerical solution* of these equations.

Numerical Solution of Differential Equations

Since it is difficult to obtain the solution for a nonlinear set of differential equations, we try to utilize a computer program which numerically computes the solution at *discrete* points in time. An alternative would have been to implement a setup using scaled physical elements which mimic the differential equations, i.e., an analog computer. However, given their flexibility, numerical evaluation using computers is convenient and economical.

So how do we solve the differential equations numerically ? Let us consider a simple example:

$$\frac{dx}{dt} = ax$$

If we wish to obtain the solution denoted by x0, x1, x2, ...xk at the closely spaced time instants t = 0, h, 2h, ..., kh, then we can approximate the above equation as:

$$\frac{(x_{k+1} - x_k)}{h} = ax_k$$

Therefore if xk is known, xk+1 can be obtained by

$$x_{k+1} = (1 + ah)x_k$$

Therefore, if the initial condition x0 is known, we can recursively find the solution of x at discrete instants of time. The approximation is likely to work well only if h is "small enough".

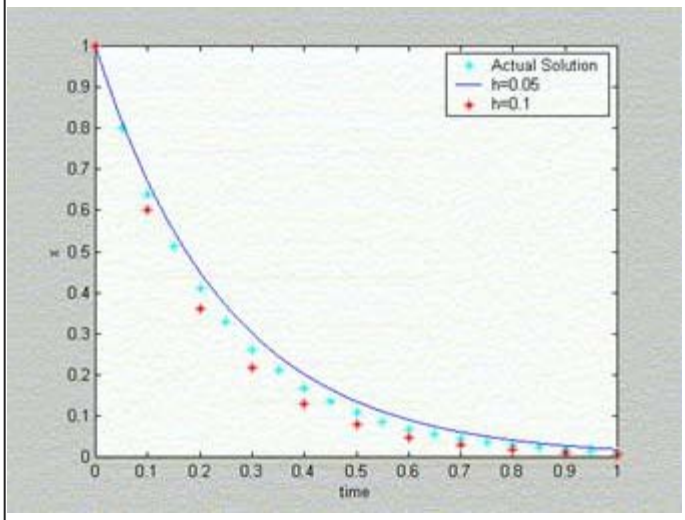
To illustrate this, consider the numerical solution at discrete points and compare it with the correct solution:

$$x(t) = e^{at}x(0)$$

Let us take the constant a = -4.

We see from the figure above that the approximation works better for smaller h.

The method that we have presentated is in fact the simplest possible one and is called "Euler's Method" of numerical integration.



(click to enlarge)

It is not difficult to see that another possible approximation of the differential equation could be:

$$\frac{(x_{k+1} - x_k)}{h} = ax_{k+1}$$

This is called the "Backward Euler" method. The numerical solution obtained using different approximations and different values of h could vary. Inappropriate selection of the method and the time step h can lead to errors which can cause erroneous conclusions.

Comparison of the two methods s

We have considered two methods: Euler Method and Backward Euler method. Euler's method is simple to implement on a computer program, but is known to have poorer numerical properties. In particular, it may wrongly show a system which is actually stable to be unstable. As an example consider the following equations:

$$\frac{dx_1}{dt} = -5x_1 + 0.01x_2$$
$$\frac{dx_2}{dt} = -100x_2$$

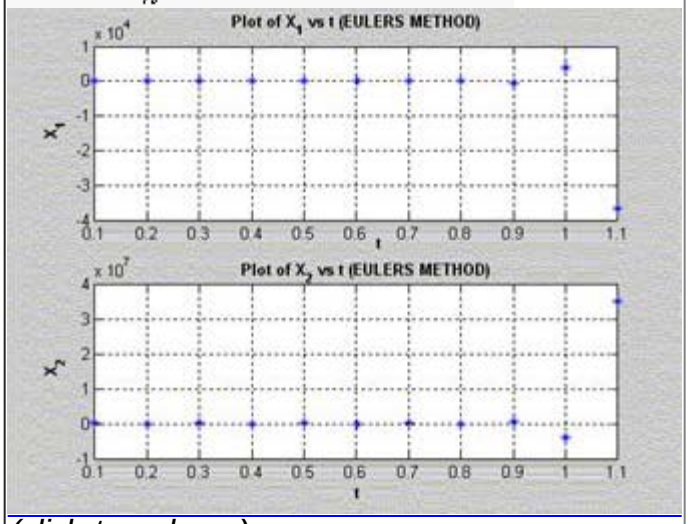
If we use Euler Method, we descretise the equations as follows.

$$\frac{(x_{1\_k+1} - x_{1\_k})}{h} = -5x_{1\_k} + 0.01x_{2\_k}$$
$$\frac{(x_{2\_k+1} - x_{2\_k})}{h} = -100x_{2\_k}$$

For a time step, h=0.1, with initial conditions x1(0)=1 and x2(0)=0.01, **the numerically evalauated response is unstable**, although this system is actually stable and has a time response given by:

$$x_1(t) = e^{-5t}x_1(0) + \frac{e^{-5t}}{9500}x_2(0) - \frac{e^{-100t}}{9500}x_2(0)$$
$$x_2(t) = e^{-100t}x_2(0)$$

One has to use a very small time step (say, h = 0.005s) if one has to avoid getting a grossly incorrect numerical solution. However, this would mean more computation time, even one may be interested only in capturing the slow response corresponding to the exp(-5*t) term in the response.

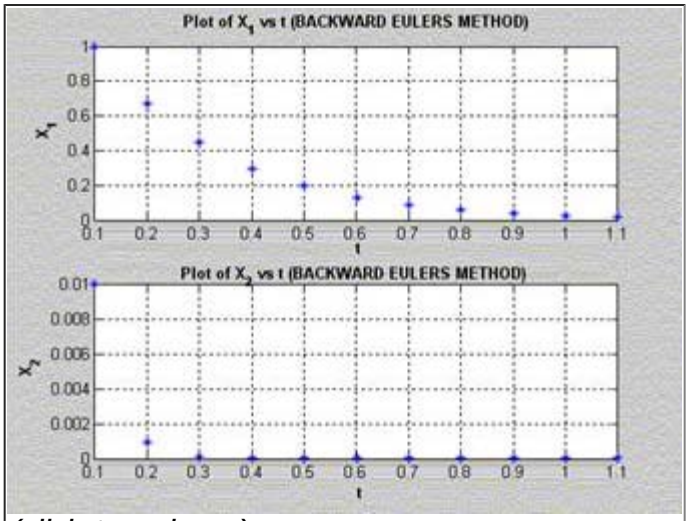


(click to enlarge)

Let us repeat this exercise with Backward Euler with the same h=0.1. The discretised equations are:

$$\frac{(x_{1\_k+1} - x_{1\_k})}{h} = -5x_{1\_k+1} + 0.01x_{2\_k+1}$$
$$\frac{(x_{2\_k+1} - x_{2\_k})}{h} = -100x_{2\_k+1}$$

Clearly, the solution obtained from Backward Euler Method does not "blow up". It also captures the slow transient corresponding to exp(-5*t) quite accurately with this value of time step.



(click to enlarge)

The problem with Backward Euler Method is evident when we apply it to the solution of swing equations:

$$\frac{\delta_{k+1} - \delta_k}{h} = (\omega_{k+1} - \omega_o)$$
$$\frac{\omega_{k+1} - \omega_k}{h} = \frac{\omega_B}{2H}\left(T_m - \frac{EE_b \sin\delta_{k+1}}{X}\right)$$

Clearly, getting the values of d and w at the k+1 th instant from the values at the k th instant is tough as we have to solve nonlinear *algebraic* equations (how does one solve them?).

Clearly there is a trade-off between numerical accuracy and complexity of implementation! We now consider a method which is not too difficult to implement and also gives reasonably good accuracy.

Runge-Kutta Fourth Order Method

Consider the differential equation:

$$\frac{dx}{dt} = f(x,t)$$

The Runge-Kutta 4th order algorithm uses intermediate points in an interval of duration h to calculate the state xk+1 from xk as follows:

$$k_1 = f(x_k, t_k)$$
$$k_2 = f(x_k + \frac{h}{2}k_1, t_k + \frac{h}{2})$$
$$k_3 = f(x_k + \frac{h}{2}k_2, t_k + \frac{h}{2})$$
$$k_4 = f(x_k + hk_3, t_{k+1})$$
$$x_{k+1} = x_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Note that the number of calculations per time step is larger than that for Euler Method. However, this method is more accurate than Euler method (for the same time step) for most systems.

Can you program an algorithm which implements this method for the swing equations ?

Recap

In this lecture you have learnt the following

- A system is said to be stable if it can withstand disturbances and come to an equilibrium
- Angular stability refers to the study of whether machines remain in synchronism after a disturbance
- The equations of a single machine infinite bus system reveal that torque is a nonlinear function of rotor angle

Congratulations, you have finished Lecture 7. To view the next lecture select it from the left hand side menu of the page.