Chapter 2

Numerical Aspects

Nonlinear systems can exhibit a very large number of various phenomena, unlike linear systems. In many cases, these behaviors may be sorted out by looking into the physics of the problem, as we have seen in the introduction. However, in many other cases, physical insight does not suffice; the goal of this course is to give you tools to investigate the behavior of nonlinear systems. The first of these tools (and also the most obvious one) is to rely on *simulation*. While there are many strong reasons *not* to believe what simulation tells you, it remain the primary tool used by Aerospace Engineers for testing dynamic systems (satellite, plane) before actually flying.

A full course on numerical integration of differential equations would actually take one full semester. So consider this chapter as a (valuable) introduction to the topic.

2.1 Forward Euler method

Differential equations are encountered not only in the simple context of control, but also in virtually all branches of Mechanics, Physics, Chemistry and other disciplines. The few instances when one encounters "analytical", "closed-form" solutions to a given differential equation should not let one be fooled about the generality of such forms: most nonlinear differential systems just do *not* have a "nice" closed-form solution, and one needs to rely on simulation in order to find a solution

to the set of differential equations. In fact, one may even be fooled by the idea that a "closed-form" solution to a differential equation is of any help. Take for example the system

$$\frac{d}{dt}x = -x, \quad x(0) = 1. \tag{2.1}$$

Everyone knows the solution to this ODE is

$$x(t) = e^{-t}, \tag{2.2}$$

an "analytical" solution. However, you should note that this "closed-form" solution does not give you a clue about how it is to be computed, either by hand or through a computer. On the contrary, the differential equation (2.1) gives you quite a bit of information about how to compute e^{-t}: indeed, dx/dt may be approximated by $\Delta x/\Delta t$, where Δx is the increment of x during the amount of time Δt. Calling x_0, x_1, ..., x_k the successive values of x at instants 0, Δt, $2\Delta t$ and so on, the differential equation (2.1) may be approximated by

$$\frac{x_{k+1} - x_k}{\Delta t} = -x_k, \quad x_0 = 1. \tag{2.3}$$

This suggests the recursive solution scheme

$$x_{k+1} = x_k - x_k \Delta t \tag{2.4}$$

to obtain a numerical value for e^{-t}. This scheme is now simple enough to implement on a computer, thus showing the superiority of the differential form (2.1) over the "analytical" solution (2.2). The numerical scheme (2.4) is named *forward Euler method.* It is the simplest numerical solution scheme you may imagine and generalizes to **any** set of first-order differential equations

$$\frac{d}{dt}x = F(x), \quad x(0) = x_0$$

to take the form

$$x_{k+1} = x_k + F(x_k)\Delta t.$$

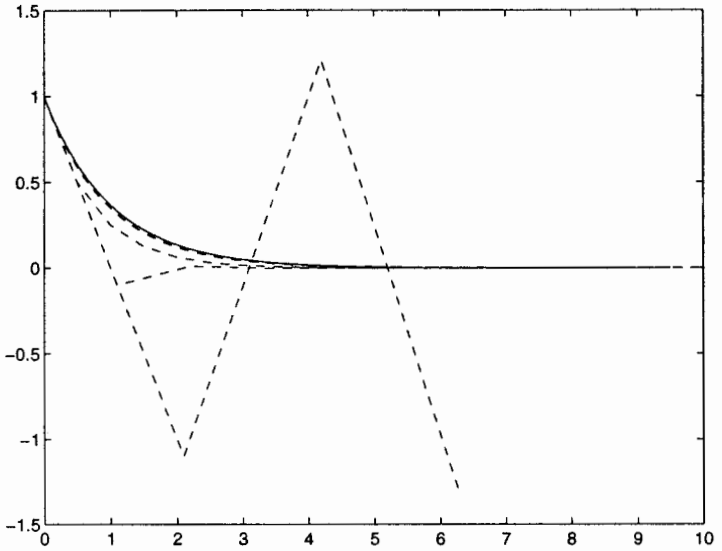

Figure 2.1: Time histories for computing e^{-t} via numerical integration. Plain is MATLAB e^{-t}. Dashed is numerical for $\Delta t = 0.01, 0.1, 0.5, 1.1, 2.1$.

Its great advantage is that you can code it with Matlab in two lines, even for very sophisticated systems. From now on, you have no excuse for not being able to simulate a (linear, nonlinear) system.

Let us see what the forward Euler method does to the system (2.1); we easily see that as Δt get larger and larger, the accuracy of the numerical solution degrades (it can be shown that this degradation is a *quadratic* function of Δt, and thus this method is said to be a *first-order* method). Moreover, when Δ gets past 2, the numerical solution is unstable. This can be formally proved by writing the forward Euler method as

$$x_{k+1} = (1 - \Delta t)x_k,$$

which is a divergent geometric series when $\Delta t > 2$. *Accuracy* and *stability* are the main two qualities of a numerical solution method. They must be considered separately, as we will see that accurate methods might not be stable and stable methods need not be accurate.

Note that all simulation plots you have seen so far were obtained

using the forward Euler method. In general, for the simple linear system

$$\frac{d}{dt}x = -\lambda x, \quad \lambda > 0$$

its numerical solution will be stable if $\lambda\Delta t < 2$ and accurate if $\lambda\Delta t << 1$. Thus, accuracy implies stability, but not the converse in this case.

2.2 Implicit Euler Method

In the previous section, we have seen that the accuracy and stability of the forward Euler method depend on how small Δt is chosen. The smaller Δt, the better the accuracy and stability of the numerical method. However, smaller time steps also imply longer computation times. So a trade-off needs to be done.

There exists a very common class of dynamical systems, named *stiff* systems for which this trade-off becomes very difficult. Stiff systems are systems in which very slow and very fast modes co-exist. For example, the transfer function

$$h(s) = \frac{100}{(s+1)(s+100)} \tag{2.5}$$

is a very stiff system, because it has relatively small and large poles (at -1 and -100); consider the problem of computing its step response; the first way of doing so it to use MATLAB with the command `step`. However, the main drawback associated to this procedure is that you have used a "black-box", that is, a procedure whose contents you don't know. As you will see later, it turns out that this porcedure is *unreliable* for stiff systems! Let us now construct *reliable* methods to solve this problem:

By inspection, you can say "well, the pole at -100 is so fast that I am going to neglect it. This step response must look like the step response of $1/(s+1)$".

Thus, any good numerical integration scheme should somehow take this in account, by not letting the fast pole interact with the numerical scheme. The step response of the transfer function (2.5) is obtained by

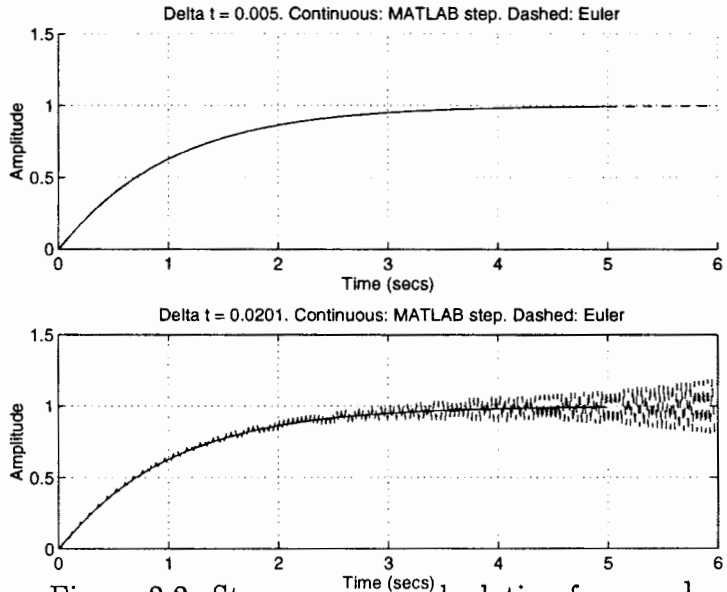

Figure 2.2: Step response calculation for $\frac{1}{(s+1)(s+100)}$

simulating the system

$$\begin{aligned} \frac{d}{dt}x_1 &= -x_1 + 1, \quad x_1(0) = 0 \\ \frac{d}{dt}x_2 &= -100x_2 + 1 \quad , x_2(0) = 0 \\ s(t) &= \frac{100}{99}(x_1 - x_2). \end{aligned}$$

In view of the statements of the previous section, the forward Euler method is not a good candidate for such a task: In order for it to work properly (that is, be stable and accurate), we need to have $100\Delta t << 1$, and thus the fastest mode forces a very small time step. Let us see this on Fig. 2.2. As soon as $100\Delta t > 2$, the numerical simulation becomes unstable. The fix for these problems has been found to use a more sophisticated numerical integration scheme, named *implicit* or *backward* Euler method. The idea is that while the forward Euler method approximates the differential equation

$$\frac{d}{dt}x = F(x)$$

by

$$\frac{x_{k+1} - x_k}{\Delta t} = F(x_k),$$

the implicit Euler method approximates the same differential equation by

$$\frac{x_{k+1} - x_k}{\Delta t} = F(x_{k+1}). \tag{2.6}$$

Thus, unlike the forward Euler method, the implicit Euler method requires the solution of a possibly nonlinear equation to compute x_{k+1} from x_k (thus an "implicit" method). Let us now see the effect of this new scheme on the stable dynamic system

$$\frac{d}{dt}x = -\lambda x, \quad \lambda > 0.$$

The corresponding implicit Euler method is then

$$x_{k+1} = x_k - \lambda x_{k+1}\Delta t,$$

or

$$x_{k+1} = \frac{1}{1 + \lambda\Delta t}x_k.$$

We now see that the resulting geometric series *is always convergent.* Thus the choice of the time step does not affect the stability of the method: *However* , this choice affects the accuracy of the method: the integration error varies also quadratically with Δt and this is therefore also a first-order method.

The global effect of this method is to correctly simulate the slow, significant modes of a system while filtering out the fast ones. A correct way to use the implicit Euler method is therefore to choose Δt such that $\lambda\Delta t >> 1$ for the modes that are to be neglected, and $\lambda\Delta t << 1$ for the others. Let's now use this method to compute the step response of (2.5). As we can see on Fig. 2.3, the implicit Euler method allows to use much larger integration steps, yet obtain a good level of accuracy. However, if the integration step is too large, (bottom plot of Fig. 2.3) accuracy is lost, though stability remains.

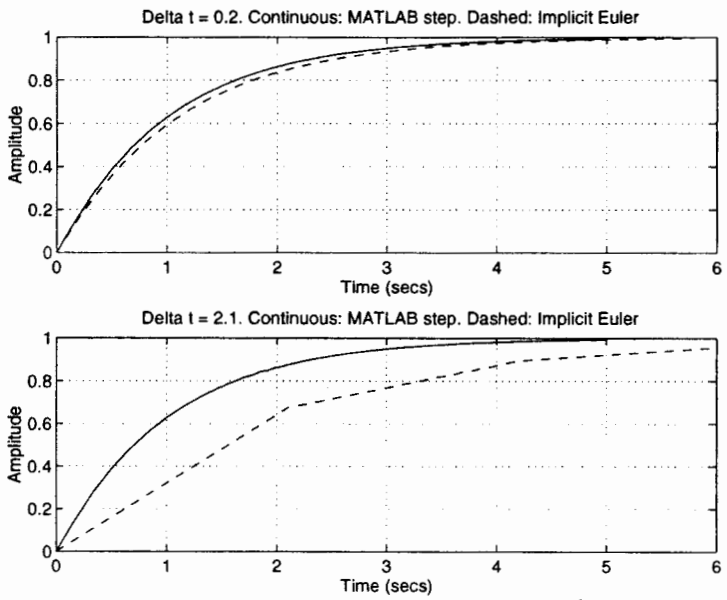

Figure 2.3: Step response calculation for $\frac{1}{(s+1)(s+100)}$: implicit Euler

2.3 Practical rules for nonlinear systems

While the forward and implicit Euler schemes are valid for almost any nonlinear system, the criteria we have derived on how to use them are based on linear systems examples. Unlike linear systems, a nonlinear system of the form

$$\frac{d}{dt}x = F(x)$$

may see its characteristic speeds change a lot, depending on the trajectory followed. Therefore, it is necessary to check at every time whether the chosen time step is too small or too big. This is done by inspecting the "local" dynamics of the system, that is, by computing the Jacobian matrix ΔF and inspecting its eigenvalues (modes). This usually leads to performant methods where the integration step varies at each iteration. Most examples you'll have to deal with aren't big enough for you to see the advantages of variable step length and you may want to stick to a "sufficiently small" one.

2.4 Higher-order methods

Forward and Implicit Euler methods are called first order numerical methods, because the amount of error developed during the integration varies quadratically with the size of the integration step. We now quickly present methods that allow to obtain higher-order errors and improved accuracy.

2.4.1 Forward Euler method is first order

Let us first understand why forward Euler is a first-order method. The forward Euler method for the dynamic system

$$\dot{x} = F(x)$$

is

$$x_{k+1} = x_k + F(x_k)\Delta t. \tag{2.7}$$

Let's compute the difference between x_{k+1} and $x(\Delta t)$, the solution of

$$\dot{x} = F(x), \;\; x(0) = x_k, \tag{2.8}$$

at time Δt. To do this, we first rewrite (2.8) in the integral form

$$x(\Delta t) - x_k = \int_0^{\Delta t} F(x(\tau))d\tau. \tag{2.9}$$

A Taylor expansion of the right-hand side gives

$$x(\Delta t) - x_k = \Delta t F(x(0)) + \frac{(\Delta t)^2}{2}\frac{dF}{dx}(x(0)).F(x(0)) + (\Delta t)^3\mathcal{O}(1). \tag{2.10}$$

Thus, by substracting (2.7) from (2.10), we obtain

$$x_{k+1} - x(\Delta t) = -\frac{(\Delta t)^2}{2}\frac{dF}{dx}(x(0)).F(x(0)) - \Delta t^3\mathcal{O}(1),$$

that is, a second-order error in Δt. This is why the forward Euler method is "first-order".

2.4.2 Higher-order methods

Assume now that we were to use the numerical scheme

$$x_{k+1} = x_k + \Delta t F(x_k) + \frac{\Delta^2}{2} \frac{dF}{dx}(x_k).F(x_k). \qquad (2.11)$$

Then, repeating the calculations as above would now yield a third-order error in Δ. Using higher Taylor expansions of the right-hand side of (2.9) would yield even more accurate numerical schemes (you do this in your homework). This is the essence of the Runge-Kutta method. the changes made for numerical implementation are the following: first, the computation of successive derivatives for F is a tedious task, and therefore it is replaced by approximate values obtained by computation of F only. Second, the scheme (2.11) is still a forward scheme and bears the same drawbacks as the forward Euler method: therefore it is usually modified to give implicit schemes similar to the implicit Euler method. Finally, this higher-order scheme also exists for time-dependent systems:

$$\frac{d}{dt}x = F(x,t),$$

at the expense of more complicated calculations.

The figure 2.4 shows how using second order information, for example, helps improve the accuracy of the numerical integration for a given integration step for the system

$$\frac{d}{dt}x = -x, \;\; x(0) = 0.$$

2.4.3 Computer routines

There exist MATLAB routines that implement the Runge-Kutta technique. These are named `ODE23` and `ODE45`, and they correspond to low and high order Runge-Kutta solvers. Please learn them and use them!

Problems

1. **State-space representation for transfer functions**

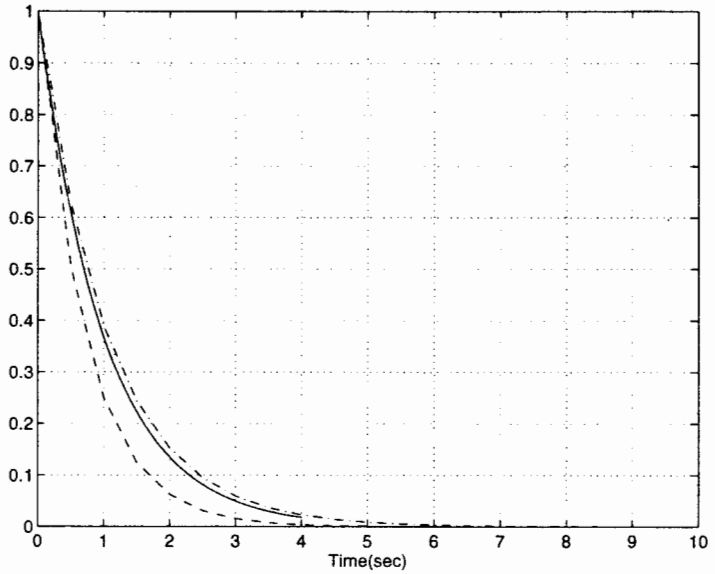

Figure 2.4: e^{-t}: exact (cont), 1st order (dashed), 2nd order (dot-dashed)

Transfer functions are merely an easy format for linear ODE. The methods proposed in this chapter require you to be able to get a state-space representation for your transfer function. Given the transfer function

$$h(s) = \frac{1}{s^3},$$

find an equivalent state-space representation for it in the form

$$\begin{aligned} \dot{x} &= Ax + bu \\ y &= cx + du. \end{aligned}$$

Please do the same for the transfer function

$$h(s) = \frac{(s-1)^2}{s^3}.$$

2. **Turning multiple-order order nonlinear systems into first-order ones**

   Rewrite the nonlinear differential equation (1.4) in Chapter 1 as a first-order system

$$\dot{x} = F(x).$$

3. **When Matlab fails and implicit Euler works**

   In this problem, we are going to experimentally demonstrate that knowing the methods is better than knowing "black box" routines. Consider the transfer function

   $$h(s) = \frac{10^{16}}{(s+1)(s+10^{16})}.$$

   We wish to compute the step response for such a transfer function.

   (a) First using the `step` command in MATLAB

   (b) Second using the implicit Euler method

   Plot your results and conclude.

4. **Runge-Kutta method**

   Can you outline what the numerical integration scheme (2.11) looks like when considering a scalar, *linear* system? As you try higher and higher order schemes, can you justify why the method is more and more exact?

   (**hint:** The taylor expansion of $e^{\lambda t}$ is

   $$e^{\lambda t} = 1 + (\lambda t) + (\lambda t)^2/2 + (\lambda t)^3/3! + \ldots \Big)$$

5. **Implementation of numerical methods on examples**

   Consider the nonlinear system

   $$\ddot{x} + 4\dot{x} - 2.5x^2 + 5x = 0,$$

   represented in state-space as

   $$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -4x_2 - 5x_1 + 2.5x_1^2. \end{aligned}$$

   Simulate this system using what you learned in this chapter (but please explain your choices), for the two initial conditions

   $$x_1(0) = 1.487, \quad x_2(0) = 2.5$$

and

$$x_1(0) = 1.488, \quad x_2(0) = 2.5$$

Is the on-line check of the validity for the numerical integration step important? Show your plots.

6. **Satellite control system with on-off controls**

A satellite along one of its main axis looks like

$$y = \frac{1}{s^2}u,$$

that is, a double integrator and u is the thruster. For money reasons, the thrust can only be switched to $+1$ or -1. An attitude control system was devised for this system:

$$u = +1 \text{ if } y + 0.2\dot{y} < 0$$
$$u = -1 \text{ otherwise}$$

Simulate this system over 20 seconds, starting from the initial conditions $y = 3$, $\dot{y} = 0$. What are the issues raised by this problem beyond the contents of this chapter? How may you solve them? Show your plots.