LECTURE NOTES FOR THE COURSE "STOCHASTIC SIMULATION METHODS IN PHYSICS"

Andrea Pagnani

CONTENTS

I. INTEGRATING DIFFERENTIAL EQUATIONS

In this chapter we discuss how to numerically integrate differential equations: we consider the basic example of a harmonic oscillator. When introducing the different methods of numeric integration we focus on how to control the systematic error due to the discretization of the time interval. For didactic reasons, we will consider in particular the one dimensional harmonic oscillator case.

I.1. The harmonic oscillator

We just briefly recall the general solution of the harmonic oscillator in one dimension. We will assume that the body moves without any friction, while a spring tries to pull it back to the point where it is at rest, $x = 0$, i.e., the body equilibrium point. The force acting on the body is an elastic restoring force The Newton equation reads in this case:

$$m\ddot{x} = -Kx$$

Defining $\omega_0 := \sqrt{K/m}$, the general solution reads:

$$x(t) = A\cos(\omega_0 t + \delta)$$

where A and δ are two constants determined by the motion initial conditions (this is a second order differential equation, so we have two constants). A is the motion amplitude, δ is the phase at time zero. Note also that the cosine argument is expressed in radiants. The initial conditions of the problem can be given in different ways. For example, A and δ can be substituted by an initial position and velocity:

$$(A, \delta) \quad \Longleftrightarrow \quad \left(x_0, v_0 = \frac{dx}{dt}\Big|_{t=t_0}\right)$$

The relation is:

$$\begin{cases} A\cos(\delta) = x_0 \\ -A\omega_0\sin(\delta) = v_0 \end{cases} \quad \Longleftrightarrow \quad \begin{cases} A = \sqrt{\frac{\omega_0^2 x_0^2 + v_0}{\omega_0^2}} \\ \delta = -\text{atan}\left(\frac{v_0}{\omega_0 x_0}\right) \end{cases}$$

As it is well known the energy is a conserved quantity:

$$E = \frac{m}{2}v^2(t) + \frac{K}{2}x^2(t)$$

and thus it will be used as a *sensor* for detecting the consistency of the numerical approach. Finally we note that we could have used the complex number notation to achieve the same result. Indeed, the equation $\ddot{x} = -\omega_0^2 x$ admits as a solution $x(t) = x_0 e^{i\left(\omega_0^2 t + \delta\right)}$, which implies that:

$$\dot{x}(t) = i\omega_0 x(t) \tag{1}$$

I.2. Integration Algorithms

We now study how to numerically integrate the equations of motion characterizing a physical system. We consider the simple harmonic oscillator, coupled first order differential equations: easy to integrate in an exact way, as a first exercise. It allows us to define a method and to develop some techniques that will be useful to analyze more interesting physical systems.

In one spatial dimension, defining $\phi(x,t) := F(x,t)/m$ where we consider a most general case where the force field might depend from the position and time. Note that in the following we are not considering cases for which the force filed depends on the velocity (such as viscous friction forces). Introducing the "force" ϕ, Newton equations are of the form:

$$\ddot{x} = \phi(x,t)$$

A second order differential equation can be casted into the first order form as:

$$\begin{cases} \dot{v}(t) = \phi(x,t) \\ \dot{x}(t) = v(t) \end{cases} \tag{2}$$

For instance, in the harmonic oscillator case the "force" depends only on the particle position $x(t)$ and thus $\phi(x,t) = \phi(x)$ would be $\phi(x) = -\omega_0^2 x(t) = Kx(t)/m$. To study these equations with a computer we need to discretize the time variable. We consider very small, though finite, time increments Δt, and we are interested in the limit in which this increment tends to 0, i.e., $\Delta t \to 0$. Starting at a given time t_0 (which we usually set equal to zero) we compute the solution of Eq. 2 at the times $t_n = t_0 + n\Delta t$ for $n \in 1, 2, 3, \ldots$. To simplify the notations we will define

$$x_n := x(t_n) \quad , \quad v_n := v(t_n) \quad , \quad \phi_n := \phi(t_n)$$

As already mentioned, we are eventually interested in the $\Delta t \to 0$ but keeping it finite. In practice:

- When varying Δt, we should be able to bound to change our results as a function of Δt. This is analogous to what we did with the numerical integration for which we always have a way to cap the uncertainty of the result when the integration step $h \to 0$.

- The quantities that are conserved in the continuous limit and that are time-dependent in our numerical integration instead should not change more, at the time t, than by a maximum quantity. In case of the energy, for example, we want the quantity $(E(t) - E(0))/E(0)$ to be smaller than a predetermined quantity (for example, of no more than 0.01, meaning that at time t the energy has not changed of more than one percent).

Euler and Euler-Cromer

We start by expanding Eq. 2 as a Taylor series. Since it will be useful when writing more accurate and efficient algorithms, we keep terms $O(dt^2)$:

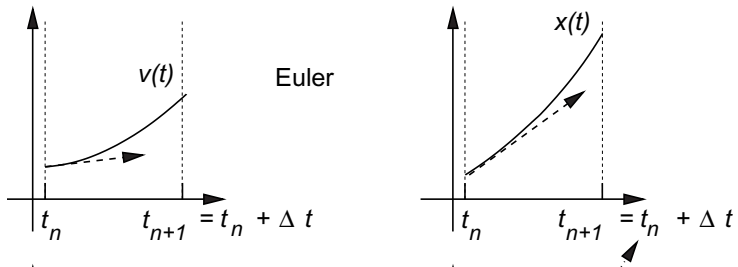
$$\begin{cases} v(t+dt) \simeq v(t) + \phi(t)dt + \mathcal{O}(dt^2) \\ x(t+dt) \simeq x(t) + v(t)dt + \frac{1}{2}\phi(t)dt^2 + \mathcal{O}(dt^3) \end{cases} \tag{3}$$

In the Euler method only the first order terms are taken into account and the infinitesimal dt is substituted with the small but finite term Δt. This is equivalent to applying the rectangle integration method and assuming the force and the velocity to be constant within the time interval Δt. We start from time t_0, add Δt at each time step, such that the time t_n is reached after n steps. After the next step, we have reached $t_{n+1} = t_n + \Delta t$ and, according to the definitions given in Eq. 3, we find the following iterative scheme:

$$\begin{cases} v_{n+1} = v_n + \phi_n \Delta t \\ x_{n+1} = x_n + v_n \Delta t \end{cases} \tag{4}$$

also commonly known as the Euler method.

$$\begin{cases} v_{n+1} = v_n + \phi_n \Delta t \\ x_{n+1} = x_n + v_n \Delta t \end{cases}$$



$$\begin{cases} v_{n+1} = v_n + \phi_n \Delta t \\ x_{n+1} = x_n + v_{n+1} \Delta t \end{cases}$$

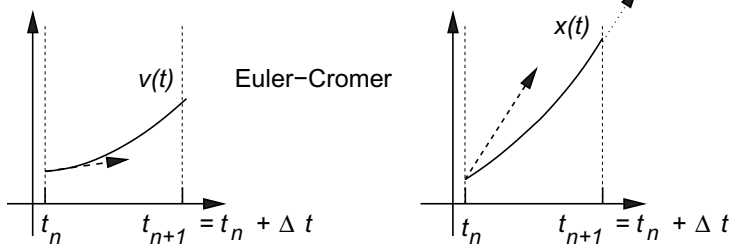


Figure 1. Iteration in a time interval Δt with the Euler method (on top) and with the Euler-Cromer method (bottom). The dashed lines indicate the directions in which the system moves, the dotted lines the way in which these displacements have been calculated (see discussion in text).

A pictorial representation of what is happening is given in Fig. 1. The left plot shows the system velocity, the right one its position. Since we discretized the time variable, our time steps start at time $t_n = t_0 + n\Delta t$ and end at the time $t_{n+1} = t_0 + (n+1)\Delta t$. We do not consider what happens in between these two times. Using this discretization procedure, we just want to estimate the true trajectory as well as possible. In the Euler method, both $v(t)$ and $x(t)$ are estimated by the tangent to the true trajectory in the point t_n. This tangent is then used to extrapolate the value of the trajectory to the next time instance. The error we make in this way decreases as Δt decreases. To evaluate the intrinsic error of the Euler method, we start by noticing that, as we only kept terms of the order dt in the Taylor series, at each integration step an error of the order $\mathcal{O}(\Delta t^2)$ is made. This is called the method local error. It accumulates throughout the numerical integration, as an integration over a time interval t takes $t/\Delta t$ integration steps. If, for example, t is equal to 60 seconds, with $\Delta t = 0.1$ seconds, we need 600 integration steps. If we improve the integration precision bringing Δt to 0.05 seconds, we need to perform 1200 elementary steps to cover the same 60 seconds. If we are to estimate a global error, whose order of magnitude is given by the product of the local error and the number of steps needed to reach the desired (fixed) time. Therefore the global error decreases with Δt as $\Delta t^2/\Delta t = \Delta t$. So, the Euler method is of the first order in Δt. This, together with some other reasons we discuss next, is why it is considered an awful integration method.

Hands on the code

Integrate the equations of the harmonic oscillator with the can also be studied as a function of ω_0. Verify how the system energy varies with time, by analyzing the ratio:

$$\frac{\Delta E_{\Delta t}}{E(0)} := \frac{E_{\Delta t}(t) - E(0)}{E(0)}$$

For $\Delta t \to 0$ this difference should be linear in Δt at a fixed time instance t. Check whether this is the case. Another interesting quantity is the parametric plot of $x(t)$ as a function of $v(t)$. What should be the shape of this plot? Instead, what shape does it have when integrating with the Euler method?

A small change leads to the Euler-Cromer method (often this change is even due to a programming error, because of updating a variable too soon: spend a few moments pondering this statement, and try to be sure you really got its meaning):

$$\begin{cases} v_{n+1} = v_n + \phi_n \Delta t \\ x_{n+1} = x_n + v_{n+1} \Delta t \end{cases} \tag{5}$$

As we shall see next, this method is not unstable when integrating oscillating systems, even if it is, as the Euler method is, of the order Δt (i.e., a "bad" method). This new method uses the starting and ending time in a more

balanced way. Indeed, while the acceleration, used to update the velocity, is calculated at the starting time t_n of the integration step, the velocity, used to update the position, is computed in the ending point t_{n+1}. In this way we hope to approximate the trajectory better. In order to estimate the system new position in the phase space at the time t_{n+1} we used more than just the information at the time t_n (leading to an awful approximation of the true motion trajectory). This situation is shown in the lower half of Fig. 1. The velocity changes exactly as in the case of the Euler method. However, the tangent of $x(t)$ is no longer estimated at time t_n, but rather at the time t_{n+1} (dotted line). The latter is used to increment $x(t_n)$. The dashed line is in the direction of the update. Its direction coincides with the one of the dotted line, which has been displaced in t_n.

Hands on the code

Repeat the same operations you performed for the Euler method in case of the Euler-Cromer method. Compare the answers. Some behaviors are very similar, while others are quite different. Which ones?

Analysis of the Euler and Euler-Cromer stability

As a first elementary stability analysis of the Euler method, we consider a first order, rather than a second order differential equation:

$$\dot{y}(t) = -f(y(t), t) \tag{6}$$

Applying the same discretization procedure we described before (to obtain the Euler method), this leads to the iterative rule

$$y_{n+1} = y_n - f_n \Delta t \tag{7}$$

where $f_n = f(x_n, t_n)$.

An integration method is stable if a small perturbation of the solution does not grow if we continue to iterate this solution. Suppose that at a given instant the numerically iterated solution (i.e., the system trajectory) differs by a given quantity ΔY from the correct solution of the discretized equation with finite differences (not the one of the original differential equation). We have that:

$$y_{n+1} + \Delta Y_{n+1} = y_n + \Delta Y_n - \left(f_n + \frac{\partial f}{\partial y}\Big|_{y,t=y_n,t_n} \Delta Y_n \right) \Delta t \tag{8}$$

where the term enclosed between square brackets includes the change of f_n due to the change of y_n by ΔY_n. If we subtract Eq. 7 by Eq. 8 we get:

$$\Delta Y_{n+1} = \left(1 - \frac{\partial f}{\partial y}\Big|_{y,t=y_n,t_n} \Delta t \right) \Delta Y_n := G_n \Delta Y_n \quad \text{where} \quad G_n := \frac{\partial f}{\partial y}\Big|_{y,t=y_n,t_n} \Delta t$$

It's now clear what it is happening. The error at time t_{n+1} is obtained by multiplying the error at time t_n by a factor G_n. If $|G_n| > 1$, the difference is amplified, if $|G_n| < 1$ the error decreases. Indeed, the stability of the method is dictated by the following condition:

$$-1 < 1 - \frac{\partial f}{\partial y} \Delta t < +1 \tag{9}$$

The previous relation implies two interesting consequences:

1. Since $\Delta t > 0$, the right inequality implies that the equation is stable only if
$$\frac{\partial f}{\partial y} > 0$$
2. The left inequality in turn tells us that in order for the solution to be stable, the time interval should be small enough:
$$\Delta t < 2 \left(\frac{\partial f}{\partial y} \right)^{-1}$$

Let us familiarize with the stability condition in Eq. 9, by analyzing the case where the function $f(y) \propto y$. Three cases are relevant:

$$\begin{cases} \dot{y}(t) + \alpha^2 y(t) = 0 & \rightarrow & y(t) = y_0 e^{-\alpha^2 t} & \text{exponential decay} \\ \dot{y}(t) - \alpha^2 y(t) = 0 & \rightarrow & y(t) = y_0 e^{+\alpha^2 t} & \text{exponential growth} \\ \dot{y}(t) \mp i\alpha^2 y(t) = 0 & \rightarrow & y(t) = y_0 e^{\mp i\alpha^2 t} & \text{oscillations} \end{cases} \tag{10}$$

When dealing with complex numbers, the stability condition in Eq. 9, takes the form (equivalent in the real case):

$$\left|1 - \frac{\partial f}{\partial y}\Delta t\right|^2 = \left|1 \pm i\alpha^2 \Delta t\right|^2 = 1 + \alpha^4 \Delta t^2 > 1$$

which implies that Eq. 9 is never verified, and the Euler method is unstable.

Exercises

1. Show that for the systems defined in Eq. 10, the Euler method is unstable in case of growing or oscillating solutions, while it is stable for decaying solutions if $\Delta t \leq 2/\alpha^2$.
2. Numerically verify what is stated in 1. The error can be introduced in various ways.

Other method: midpoint and leap-frog.

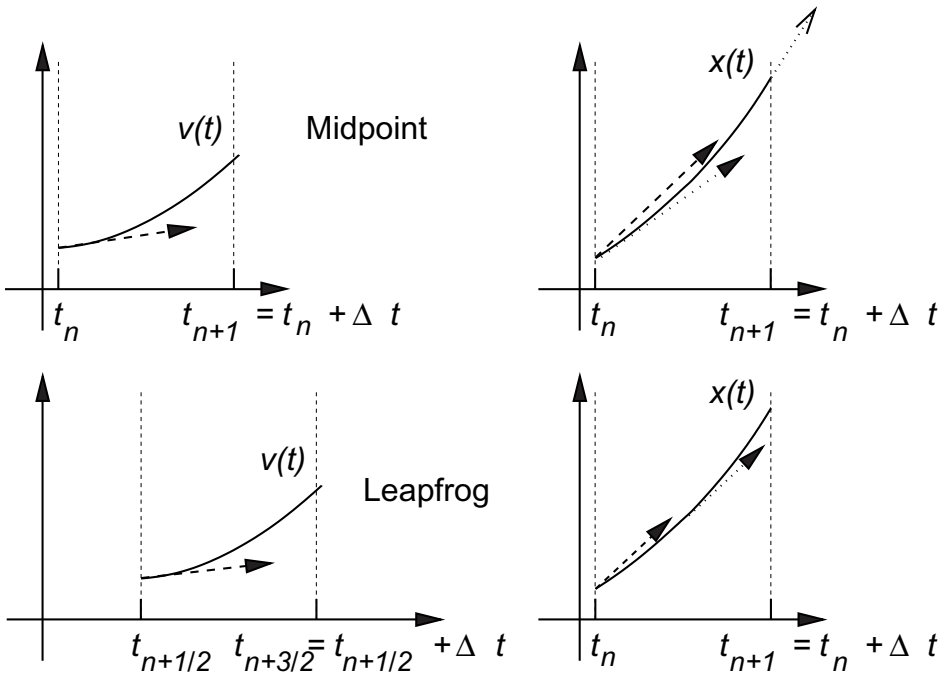


Figure 2. Iteration in a time interval Δt for the midpoint method (on top) and the leapfrog method (bottom). The dashed lines give the directions of the displacements. The dotted line show how the displacements have been calculated (see discussion in the text).

So far, we have tried, respectively with the Euler method and with the Euler-Cromer method, to compute the new value of x_{n+1} starting either from v_n or from v_{n+1}. A third possibility, as prescribed by the midpoint method, is based on the use of the quantity:

$$\bar{v}_{n,n+1} := \frac{v_n + v_{n+1}}{2}$$

This leads to the following iterative scheme:

$$\begin{cases} v_{n+1} = v_n + \phi_n \Delta t \\ x_{n+1} = x_n + \bar{v}_{n,n+1} \Delta t \end{cases} \tag{11}$$

An elementary time step in this new scheme is shown in Fig. 2 (analogous to Fig. 1 for the Euler and the Euler-Cromer method). The way in which the velocity is incremented remains the same. However, the tangent to $x(t)$ is computed both in the initial point and in the endpoint (dotted lines), and we move in the intermediate direction (dashed line). The true reason why we introduce this new scheme is immediately clear. If we substitute the first Eq. 11 in the second one, we find that:

$$x_{n+1} = x_n + \frac{v_n + v_{n+1}}{2} \Delta t = x_n + \frac{v_n + v_n + \phi_n \Delta t}{2} \Delta t = x_n + v_n \Delta t + \frac{\phi_n}{2} \Delta t^2$$

correctly reconstructing the first and second order terms when developing x in series. So, the midpoint method is of the order $\mathcal{O}(\Delta t^2)$ as far as the part in x is concerned and of the order $\mathcal{O}(\Delta t)$ for the velocity v.

Finally we analyze the leap-frog strategy. The velocities are defined at times which are halfway the steps bringing us from one value of x to the next. The algorithm is defined by:

$$\begin{cases} v_{n+\frac{1}{2}} = v_{n-\frac{1}{2}} + \phi_n \Delta t \\ x_{n+1} = x_n + v_{n+\frac{1}{2}} \Delta t \end{cases} \tag{12}$$

Note that the method is not able to start the iterative procedure in an autonomous way (it is not self-starting). Given the initial conditions x_0 and v_0, we cannot start the iteration cycle (as we do not know the value of v_1). A simple possibility (even though one should consider the consequences of this approach) is to make a first step with the Euler method, leading to:

$$v_{\frac{1}{2}} = v_0 + \frac{1}{2} \phi_0 \Delta t \tag{13}$$

This iterative scheme is shown in the lower half of Fig. 2. Note that the horizontal axis of $v(t)$ (in the left part of the figure) has changed. In case of the velocity we no longer start from the time t_n, but rather from time t_{n+1} , and, moving of the usual time step Δt, we reach time t_{n+3} . Therefore, the tangent of $x(t)$ is calculated in the midpoint of the interval (dotted line), and it is used to move from time t_n to time t_{n+1} (dashed line). Note that for sufficiently regular trajectories, the midpoint method is very similar to the leapfrog method.

Hands on the code

Repeat the studies you carried out for the Euler and Euler-Cromer method in the case of the midpoint method and of the leapfrog method. Compare the results.