CHAPTER 1 - NUMERICAL INTEGRATION

There are many techniques for performing numerical integrations, although no single technique is efficient for all problems. One family of techniques, which is used in the study of Newtonian mechanics and which forms the main content of this section, involves the analytical or numerical calculation of derivatives. Another approach, which we show for completeness in Sec. 1.1, is based upon Monte Carlo sampling. There is an appropriately large literature on numerical integration routines, including routines available in numerical libraries, and the interested reader is directed to Press *et al.* (1992) for further reading.

There are several approaches to the integration of differential equations. We present four of the standard algorithms: Euler (1.2), Verlet (1.3), predictor-corrector (1.4) and Runge-Kutta (1.5). In each case, we use the language of velocity and acceleration, rather than first and second derivatives, to make the application of the techniques to Newtonian mechanics explicit. Much of the material in Secs. 1.2 to 1.5 is adapted from Gould and Tobochnik (1987).

1.1 Monte Carlo integration

In its most primitive form, Monte Carlo integration determines the integral of a function by sampling its value at a number of randomly chosen points. In the Monte Carlo procedure, a one-dimensional integral is approximated by

$$\int_0^L f(x)\,dx = L\,\langle f\rangle \tag{1.1}$$

where

$$\langle f\rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{1.2}$$

In Eqs. (1.1) and (1.2), there are N sampling points, at values x_i, taken over the integration range $0 \le x_i \le L$.

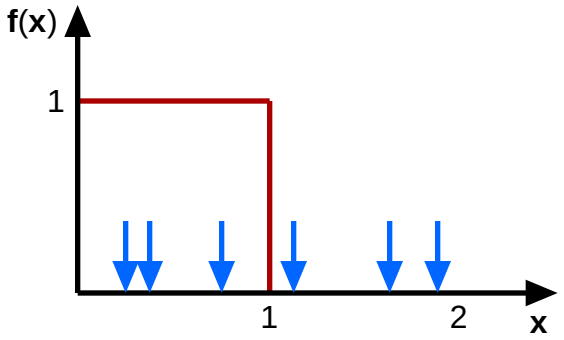


Fig. 1.1 Monte Carlo sampling of a step function. The sampling points are indicated by the blue arrows pointing towards the *x*-axis.

As an example, consider a step function $f(x)$ which has values:

$$f(x) = 1 \text{ for } 0 \le x \le 1 \qquad f(x) = 0 \text{ for } x > 1. \tag{1.3}$$

The function is shown in Fig. 1.1 by the red lines. Analytically, we know that the integral of this function over $0 \le x \le \infty$ is unity. A set of obviously artificial sampling points is indicated by the blue arrows in the figure. Suppose that we chose to sample the function using only the three points with $x < 1$, *i.e.*, $L = 1$. The function has the value of unity at each point, and we would evaluate the integral via

$$\int_0^1 F(x)\,dx = 1 \bullet (1+1+1) / 3 = 3 / 3 = 1 \tag{1.4}$$

Now consider the marginally more realistic situation where we take six sampling points, spread over $0 \le x \le 2$. In Fig. 1.1, only three of the points yield a value of 1 for the function, the others yield 0. Eq. (1.4) is changed to

$$\int_0^2 F(x)\,dx = 2 \bullet (1+1+1+0+0+0) / 6 = 6 / 6 = 1 \tag{1.5}$$

Obviously, we were lucky in our choice of sampling, since shifting just one point between $x < 1$ and $x > 1$ changes the value of the integral by 1/3. The evaluation of the integral using Eq. (1.1) becomes ever more accurate the larger the number of sampling points.

The example shows both the strength and potential weakness of the Monte Carlo integration technique.

•The strength is that although the function must be evaluated at a reasonable number of sampling points, the sample does not have to be huge. Of course, the more pathological the function, or the greater accuracy with which one wants to know the integral, the larger the sample must be.

•The weakness is that simple random sampling may be grossly inefficient. If the range of x in the example were large (say 10^6), then much of the sample would be $f(x) = 0$.

In our example, we avoided the weakness of random sampling by explicitly choosing the range of x to emphasize regions where $f(x) = 1$. There is a mathematical way of formalizing this bias called *importance sampling*, which is the heart of the Metropolis Monte Carlo procedure that we introduce in Chap. 4. In fact, Project 4 is equivalent to a numerical integration of a partition function using importance sampling. Formally, a bias is introduced such that the probability density P for sampling the function at x_i is not uniform in x. The probability density is normalized to unity via

$$\int_0^L P(x)\, dx = 1 \tag{1.6}$$

where $P(x)dx$ is the probability of selecting a sampling point between x and $x + dx$. The integral becomes

$$\int [f(x) / P(x)]\, P(x) dx = \langle f / P \rangle \tag{1.7}$$

The expectation $\langle f / P \rangle$ is just the mean value of f / P taken from the sampling points, as in Eq. (1.2). In the case of uniform sampling over the region $0 \le x \le L$, $P(x)$ is a constant equal to $1 / L$, and Eq. (1.7) reverts to Eq. (1.1).

We do not pursue the formalism of importance sampling further in this course, but refer the interested reader to Allen and Tildesley (1987) and Press *et al*., (1992).

1.2 Euler integration

In Newtonian mechanics, particularly as applied to many-body systems, one wishes to determine the trajectory of a particle given the forces to which it is subjected. That is, given a second derivative of the position with respect to time (acceleration), and some initial values for the positions and their first derivatives (velocities), the derivatives are integrated to find the positions as a function of time. Computationally,

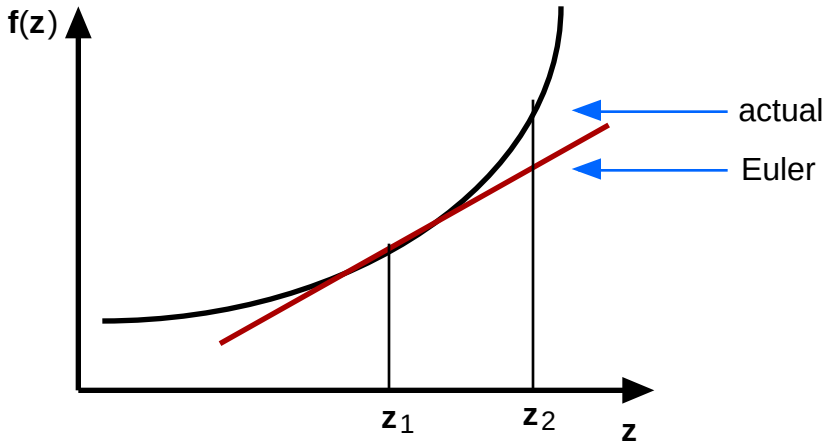


Fig. 1.2. Euler approach to obtaining the function $f(z)$ from its derivative. Knowing the derivative $(df / dz)_1$ at z_1 the Euler method predicts $z_2 = z_1 (df / dz)_1 (z_2 - z_1)$. As illustrated, the difference between the Euler prediction and the actual value of the function can be significant if the step size in z is large.

one breaks up the motion into discrete time steps and uses finite difference methods to integrate the equations of motion.

To illustrate the difficulties involved with the approach, consider the continuous function illustrated in Fig. 1.2. At a specific position x, the first (and higher) derivatives of the position with respect to time dx / dt, d^2x / dt^2 ... can be obtained either analytically [if an analytic expression is available for $x(t)$] or by taking numerical derivatives. If one wants to predict a value for x_{i+1} after some elapsed time Δt, then the most obvious solution is

$$x_{i+1} = x_i (dx / dt)_i \Delta t, \qquad (1.8)$$

where$(dx / dt)_i$ is the derivative of x with respect to time at position x_i. Although Eq. (1.8) may give reasonable results for small time steps, it can be in considerable error if:
•the time step is too large, as demonstrated in Fig. 1.2
•it is used repeatedly, so that small errors associated with each step accumulate to a large error.

The second problem is particularly important for integrating equations of motion, where thousands of integration steps may be needed to cover the appropriate time interval. Using mechanics notation, the Euler method is:

$$v_{i+1} = v_i + a_i \Delta t$$
$$x_{i+1} = x_i + v_i \Delta t, \qquad \text{(Euler)} \qquad (1.9)$$

where the velocity v_i and acceleration a_i are the first and second derivatives of x, evaluated at position x_i. A variation on the Euler method is the Euler-Cromer method:

$$v_{i+1} = v_i + a_i \Delta t$$
$$x_{i+1} = x_i + v_{i+1} \Delta t, \qquad \text{(Euler-Cromer)} \qquad (1.10)$$

1.3 Leapfrog and Verlet algorithm

As is clear from Fig. 1.2, one of the drawbacks to the Euler method is the use of derivatives that are evaluated at the beginning (or end) of the time-step. Derivatives evaluated part-way through the step would introduce a smaller error. In the leapfrog method, the position and velocity are propagated at alternate time intervals of $\Delta t / 2$. In what hopefully is obvious notation:

$$v_{i+1/2} = v_{i-1/2} + a_i \Delta t$$
$$x_{i+1} = x_i + v_{i+1/2} \Delta t. \qquad \text{(leapfrog)} \qquad (1.11a)$$

Unfortunately, Eq. (2B.10) does not say how to obtain $v_{1/2}$ from v_0. A procedure that is commonly adopted is

$$v_{1/2} = v_0 + a_0 \Delta t / 2. \qquad (1.11b)$$

Another approach that uses a more accurate evaluation of derivatives is the Verlet algorithm. The position update is second order in time, and the velocity update uses an average acceleration for the time step:

$$x_{i+1} = x_i + v_i \Delta t + (1/2)a_i \Delta t^2$$

(Verlet) (1.12)

$$v_{i+1} = v_i + (a_{i+1} + a_i)\Delta t / 2.$$

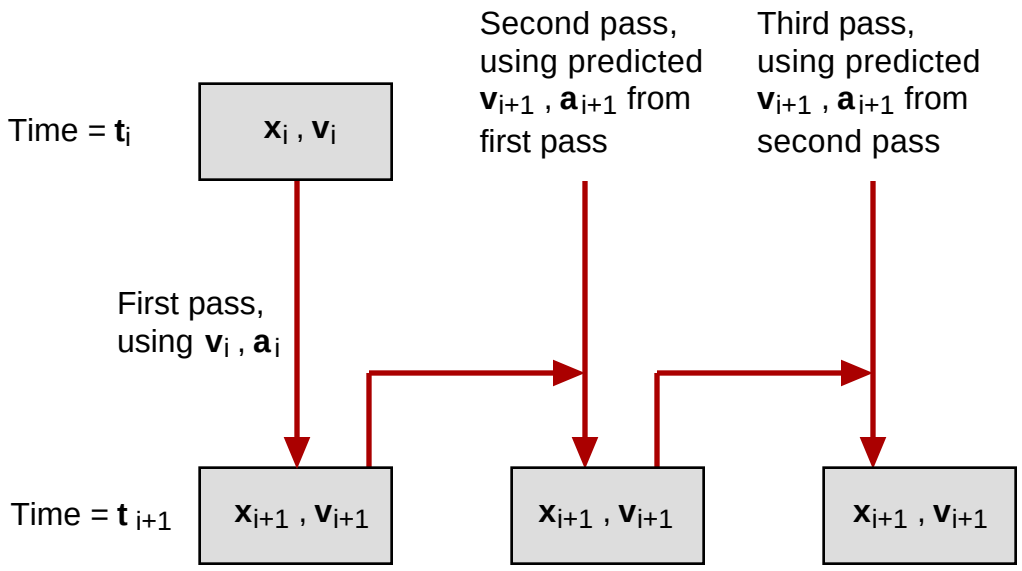
These two equations can be combined to give an alternative form to the Verlet algorithm in which the velocity is not explicitly calculated:

$$x_{i+1} = 2x_i - x_{i-1} + a_i \Delta t^2. \quad (1.13)$$

Like the leapfrog equations (1.11a), this form of the Verlet algorithm needs more information than x_0, a_0 to begin the integration.

1.4 Predictor-corrector method

The three methods presented in Secs. 1.2 and 1.3 involve simple algebraic expressions for determining the propagation of x, v in one time step. In the predictor-corrector approach, the predictions are iterated until the results are unchanged within some user-fixed bound. Schematically



Although the method takes more time to execute, because of the multiple iterations, it can be made to yield more accurate results. One sequence of updates is:

1. In a leapfrog sense, use x_{i-1} and v_i to predict x_{i+1}(pass 1):

$$x_{i+1}(\text{pass 1}) = x_{i-1} + 2v_i \Delta t \qquad (1.14a)$$

2. Evaluate the acceleration a_{i+1}(pass 1) at step i+1, using x_{i+1} as the coordinates for the forces that determine a_{i+1}.

3. Predict v_{i+1} and a new value for x_{i+1} at step i+1:

$$v_{i+1}(\text{pass 1}) = v_i + (a_{i+1}(\text{pass 1}) + a_i)\Delta t /2 \qquad (1.14b)$$

$$x_{i+1}(\text{pass 2}) = x_i + (v_{i+1}(\text{pass 1}) + v_i)\Delta t /2 \qquad (1.14c)$$

Steps 2 and 3 are repeated until the results converge to the desired accuracy.

1.5 Runge-Kutta

The final method that we outline for performing numerical integrations is the Runge-Kutta algorithm. As with many of the techniques already described, Runge-Kutta generates a more accurate value of the derivative than the Euler method. There are several *orders* of Runge-Kutta, and we mention only the second-order and fourth-order versions. See Press *et al.* (1992) for further details.

A first-order differential equation has the form

$$dy / dx = F(x,y) \qquad (1.15)$$

where $F(x,y)$ has a known functional form, and where y is to be determined as a function of x. The steps in the second-order Runge-Kutta method are:

1. Use $F(x_i, y_i)$ to find a trial value of $y_{i+1/2}$ corresponding to $x_{i+1/2} = x_i + \Delta x / 2$, where Δx is the actual step size.

2. Evaluate the derivative $F(x_{i+1/2}, y_{i+1/2})$.

3. Calculate y_{i+1} using $F(x_{i+1/2}, y_{i+1/2})$.

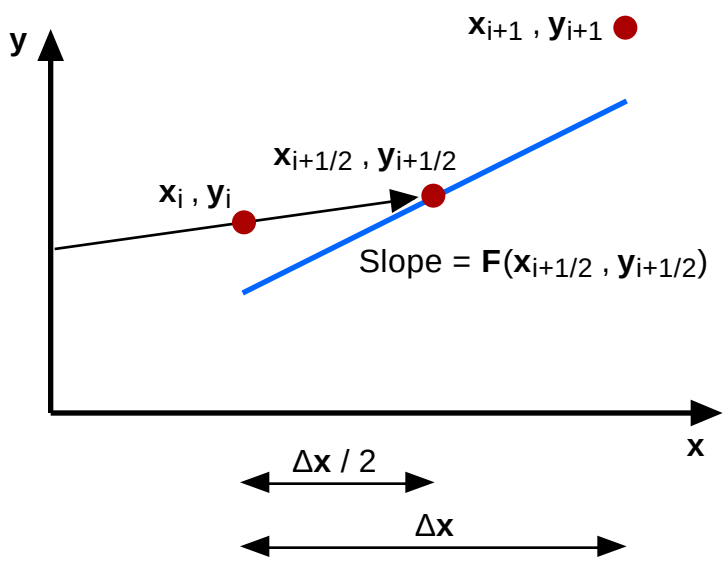


Fig. 1.3 Schematic representation of the second-order Runge-Kutta method.

The steps are displayed in Fig. 1.3. In symbols, steps 1 - 3 become:

$$k_1 = F(x_i, y_i)\,\Delta x$$

$$k_2 = F(x_i + \Delta x/2,\ y_i + k_1/2)\,\Delta x \qquad (1.16)$$

$$y_{i+1} = y_i + k_2.$$

As applied to Newtonian equations of motion, Eqs. (1.16) become

$$k_{1,x} = v_i\,\Delta t$$
$$k_{1,v} = a(x_i)\,\Delta t$$

$$k_{2,x} = (v_i + k_{1,v}/2)\,\Delta t \qquad (1.17)$$
$$k_{2,v} = a(x_i + k_{1,x}/2)\ \Delta t$$

$$x_{i+1} = x_i + k_{2,x}$$
$$v_{i+1} = v_i + k_{2,v},$$

where the acceleration $a(x_i)$ is determined at position x_i. You may find it useful to write out these equations in full (*i.e.*, eliminate the k's) to see how the propagation is handled.

Higher order versions of the Runge-Kutta method also are available. The fourth-order version is:

$$k_1 = F(x_i, y_i)\,\Delta x$$

$$k_2 = F(x_i + \Delta x/2,\ y_i + k_1/2)\,\Delta x$$

$$k_3 = F(x_i + \Delta x/2,\ y_i + k_2/2)\,\Delta x \qquad (1.18)$$

$$k_4 = F(x_i + \Delta x,\ y_i + k_3)\,\Delta x$$

$$y_{i+1} = y_i + (k_1/6) + (k_2/3) + (k_3/3) + (k_4/6).$$

References

M. P. Allen and D. J. Tildesley, *Computer Simulations of Liquids* (Oxford, 1987).

H. Gould and J. Tobochnik, *An Introduction to Computer Simulation Methods* (Addison-Wesley, Reading, Mass.,1988).

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery *Numerical Recipes in C*, 2nd edition (Cambridge, 1992) .

1.6 Project: simple harmonic motion

The techniques for numerically integrating differential equations vary in their accuracy. In this short project, techniques from Secs. 1.2 - 1.5 are used to propagate a spring executing simple harmonic motion. A review of the physics of simple harmonic motion can be found in the PHYS 120 on-line lectures notes at http://www.sfu.ca/~boal.

Physical system

An object executing simple harmonic motion has a displacement from equilibrium $x(t)$ described by

$$x(t) = A_0 \sin(\omega t + \delta), \tag{1.19}$$

where A_0 is the amplitude of the motion, δ is a phase shift determined by a boundary condition (*e.g.*, the value of x at t=0), and ω is the angular frequency. For a mass m attached to a spring with spring constant k, the angular frequency ω is given by

$$\omega = (k / m)^{1/2}. \tag{1.20}$$

The acceleration experienced by the object at displacement $x(t)$ is

$$a = - \omega^2 x, \tag{1.21}$$

where the minus sign emphasizes that the acceleration is in the opposite direction to the displacement from equilibrium.

Simulation parameters

Choose the amplitude A_0 to be unity, or alternatively work in reduced units of x / A_0. Similarly, take the phase shift $\delta = 0$, so that the conditions at $t = 0$ are

$$x(0) = 0 \qquad v(0) = \omega A_0 \qquad a(0) = 0. \tag{1.22}$$

Take the angular frequency ω to be unity, or alternatively work in reduced time units ωt.

Code

1. You will use this code as the basis of the flight to the Moon project in Chap. 3.

2. Use the integration techniques from Secs. 1.2 (Euler), 1.3 (leapfrog and Verlet) and 1.5 (second order Runge-Kutta) to propagate the displacement as a function of time. Set up the integration routines as simple functions so that they can be used in Project 3. *Note*: for the leapfrog algorithm, you may have to calculate $v(t = 1/2)$ analytically to get acceptable results, rather than use Eq. (1.11b).

3. Your integration procedure generates a sequence of values for x and v. The acceleration is not known *a priori*, but rather is obtained as a function of x from Newtonian mechanics $a = -(k/m)x$.

4. Integrate for more than 1000 steps with time step $\omega dt = 0.01$. Try several different values of ωdt ranging up to 0.1 or 0.2.

Analysis

1. Compare the displacements predicted by all four methods. Is one method noticeably better or worse than the others?

Report

Your report should contain the following elements:
•a paragraph or so for each of the four different methods used in the numerical integration
•data from a parameter set (number of steps and size of step ωdt) for which there is a clear difference between the integrated value of the differential equation and the exact solution of Eq. (1.19)
•short commentary on the results, particularly on the similarities or differences between techniques.
•a copy of your code.