3.11 ■ LEVELS OF SIMULATION

So far we have considered models in which the microscopic complexity of the system of interest has been simplified considerably. Consider, for example, the motion of a pebble falling through the air. First we reduced the complexity by representing the pebble as a particle with no internal structure. Then we reduced the number of degrees of freedom even more by representing the collisions of the pebble with the many molecules in the air by a velocity-dependent friction term. The resultant phenomenological model is a fairly accurate representation of realistic physical systems. However, what we gain in simplicity, we lose in range of applicability.

In a more detailed model, the individual physical processes would be represented microscopically. For example, we could imagine doing a simulation in which the effects of the air are represented by a fluid of particles that collide with one another and with the falling body. How accurately do we need to represent the potential energy of interaction between the fluid particles? Clearly the level of detail that is needed depends on the accuracy of the corresponding experimental data and the type of information in which we are interested. For example, we do not need to take into account the influence of the moon on a pebble falling near the Earth's surface. And the level of detail that we can simulate depends in part on the available computer resources.

The terms *simulation* and *modeling* are frequently used interchangeably, and their precise meanings are not important. Many practitioners might say that so far we have solved several mathematical models numerically and have not yet done a simulation. Beginning with the next chapter, we will be able to say that we are doing simulations. The difference is that our models will represent physical systems in more detail, and we will give more attention to what physical quantities we should measure. In other words our simulations will become more analogous to laboratory experiments.

APPENDIX 3A: NUMERICAL INTEGRATION OF NEWTON'S EQUATION OF MOTION

We summarize several of the common finite difference methods for the solution of Newton's equations of motion with continuous force functions. The number and variety of algorithms currently in use is evidence that no single method is superior under all conditions.

To simplify the notation, we consider the motion of a particle in one dimension and write Newton's equations of motion in the form

$$\frac{dv}{dt} = a(t) \tag{3.31a}$$

$$\frac{dx}{dt} = v(t), \tag{3.31b}$$

where $a(t) \equiv a(x(t), v(t), t)$. The goal of finite difference methods is to determine the values of x_{n+1} and v_{n+1} at time $t_{n+1} = t_n + \Delta t$. We already have seen that Δt must be chosen so that the integration method generates a stable solution. If the system is conservative, Δt must be sufficiently small so that the total energy is conserved to the desired accuracy.

The nature of many of the integration algorithms can be understood by expanding $v_{n+1} = v(t_n + \Delta t)$ and $x_{n+1} = x(t_n + \Delta t)$ in a Taylor series. We write

$$v_{n+1} = v_n + a_n \Delta t + O\big((\Delta t)^2\big) \tag{3.32a}$$

$$x_{n+1} = x_n + v_n \Delta t + \frac{1}{2} a_n (\Delta t)^2 + O\big((\Delta t)^3\big). \tag{3.32b}$$

The familiar Euler algorithm is equivalent to retaining the $O(\Delta t)$ terms in (3.32):

$$v_{n+1} = v_n + a_n \Delta t \tag{3.33a}$$

$$x_{n+1} = x_n + v_n \Delta t \quad \text{(Euler algorithm).} \tag{3.33b}$$

Because order Δt terms are retained in (3.33), the local truncation error, the error in one time step, is order $(\Delta t)^2$. The global error, the total error over the time of interest, due to the accumulation of errors from step to step is order Δt. This estimate of the global error follows from the fact that the number of steps into which the total time is divided is proportional to $1/\Delta t$. Hence, the order of the global error is reduced by a factor of $1/\Delta t$ relative to the local error. We say that an algorithm is nth order if its global error is order $(\Delta t)^n$. The Euler algorithm is an example of a *first-order* algorithm.

The Euler algorithm is asymmetrical because it advances the solution by a time step Δt, but uses information about the derivative only at the beginning of the interval. We have already found that the accuracy of the Euler algorithm is limited and that frequently its solutions are not stable. We have also found that a simple modification of (3.33) yields solutions that are stable for oscillatory systems. For completeness, we repeat the Euler–Cromer algorithm here:

$$v_{n+1} = v_n + a_n \Delta t \tag{3.34a}$$

$$x_{n+1} = x_n + v_{n+1} \Delta t \quad \text{(Euler–Cromer algorithm).} \tag{3.34b}$$

An obvious way to improve the Euler algorithm is to use the mean velocity during the interval to obtain the new position. The corresponding *midpoint* algorithm can be written as

$$v_{n+1} = v_n + a_n \Delta t, \tag{3.35a}$$

and

$$x_{n+1} = x_n + \frac{1}{2}(v_{n+1} + v_n)\Delta t \quad \text{(midpoint algorithm).} \tag{3.35b}$$

Note that if we substitute (3.35a) for v_{n+1} into (3.35b), we obtain

$$x_{n+1} = x_n + v_n \Delta t + \frac{1}{2} a_n \Delta t^2. \tag{3.36}$$

Hence, the midpoint algorithm yields second-order accuracy for the position and first-order accuracy for the velocity. Although the midpoint approximation yields exact results for constant acceleration, it does not usually yield much better results than the Euler algorithm. In fact, both algorithms are equally poor because the errors increase with each time step.