3 *N*-Body Particle Methods

3.1 INTRODUCTION TO THE *N*-BODY PROBLEM

The classical astrophysical "N-body" problem is deceptively simple in appearance, and its formulation is a model of brevity: Each member within an aggregate of N ($i = 1, \ldots, N$) point mass bodies, having masses m_i, experiences an acceleration that arises from the gravitational attractions of all the other bodies in the system

$$\frac{d^2\mathbf{x}_i}{dt^2} = -\sum_{j=1;\, j\neq i}^{N} \frac{Gm_j(\mathbf{x}_i - \mathbf{x}_j)}{|\mathbf{x}_i - \mathbf{x}_j|^3}. \tag{3.1}$$

The description of the problem is completed by specifying the initial velocities $\mathbf{v}_i(t = 0)$ and positions $\mathbf{x}_i(t = 0)$ for the N particles.

Equation (3.1) was originally stated and used by Newton and has been intensively studied for more than 300 years. Solutions to this equation describe intricate and diverse phenomena ranging from the orbit of the Moon around the Earth to the structure of the Kirkwood gaps in the asteroid belt to the evolution of globular clusters to the spiral structure of galaxies. Indeed, one could assemble a brilliant and fully modern career in astronomy by focusing exclusively on aspects of Equation (3.1). The richness of Equation (3.1) derives from its nonlinearity. The positions, $\mathbf{x}_i$, which are doubly differentiated to yield accelerations, appear in a sum of terms involving $|\mathbf{x}_i - \mathbf{x}_j|^{-2}$. Because the rate of change of each particle's velocity arises from a sum of inverse-square dependences, a slight change in initial conditions can lead to a complete change of long-term behavior. The goal of an N-body numerical method is to capture, as accurately as possible, this endless complexity.

The N-body problem divides naturally into two basic parts: (1) calculating the net force on a given particle at a given time, and (2) determining the new position of the particle at a somewhat advanced time. In this chapter we consider five different methods for advancing the positions of the particles. The first, Runge–Kutta integration, is a standard technique for solving ordinary differential equations. The second, Bulirsch–Stoer, provides a highly accurate method, which in practice is usually limited to systems of only a few bodies, such as nine planets orbiting a central star. The core of this method is *Richardson extrapolation* or *Richardson's deferred approach to the limit*, in which the position of a particle is integrated up to a given time with several different time step lengths, then the results are extrapolated to what they would be for a time step of zero length. The third method is known as the *symplectic map*, which has its major application in the integration of the orbits in a planetary system for very long times, but within the context where close encounters between bodies do not occur. Its speed is derived by capitalizing on the near Keplerian motion of a system of planets. The fourth method is essentially a predictor–corrector method, as applied by Aarseth to systems with moderate to large numbers of particles when reasonable accuracy is still required. The fifth, a second-order leapfrog method, is used for extremely large numbers of particles when integrations over only a few dynamical times are required

and high-order accuracy is not required. All of these methods may run into difficulties when two particles pass within short distances of each other. Such close encounters may be treated either by an inaccurate method, known as *softening*, or by an accurate method, known as *regularization*. Finally in this chapter, we give two examples of the approximate calculation of gravitational forces that are particularly applicable to large N systems: the hierarchical *tree* method and the method of Fourier analysis. A more detailed discussion of methods for calculating gravitational forces is given in Chapter 7.

3.2 EULER AND RUNGE–KUTTA METHODS

In attacking Equation (3.1) numerically (or, for that manner, any higher-order ordinary differential equation), one first dispenses with second derivatives by recasting the N second-order equations as a set of $2N$ coupled first-order equations. That is, any ordinary second-order differential equation of the generic form

$$A(x,t)\frac{d^2x}{dt^2} + B(x,t)\frac{dx}{dt} + C(x,t) = 0, \tag{3.2}$$

can be rewritten as a pair of coupled first-order differential equations

$$\frac{dx}{dt} = v(t)\,, \tag{3.3}$$

$$\frac{dv}{dt} = -\frac{B(x,t)}{A(x,t)}v(t) - \frac{C(x,t)}{A(x,t)}\,. \tag{3.4}$$

In this manner, the N second-order equations (3.1) can be expressed as a coupled set of $2N$ first-order equations

$$\frac{d\mathbf{x}_i}{dt} = \mathbf{v}_i, \tag{3.5}$$

$$\frac{d\mathbf{v}_i}{dt} = -\sum_{j=1;\,j\neq i}^{N} \frac{Gm_j(\mathbf{x}_i - \mathbf{x}_j)}{|\mathbf{x}_i - \mathbf{x}_j|^3}. \tag{3.6}$$

If we write $\mathbf{w}_i \equiv [\mathbf{x}_i, \mathbf{v}_i] \equiv (w_{i1}, w_{i2}, w_{i3}, w_{i4}, w_{i5}, w_{i6})$ to represent the six-dimensional phase space coordinates of particle i, then we can describe the state of the entire N-body system by a single $6N$-dimensional vector

$$\mathbf{W} \equiv [\mathbf{w}_1, \ldots, \mathbf{w}_N]. \tag{3.7}$$

The components, W_l, of $\mathbf{W}$ run from $l = 1 \rightarrow 6N$. For example, the x-component of the velocity of the $j = 3$ particle corresponds to the term W_{16}. The evolution of the system as described by Equation (3.5) and Equation (3.6), thus, takes the form

$$\frac{dW_l}{dt} = g_l(\mathbf{W}), \tag{3.8}$$

where the $6N$ functions g_l are given by the right-hand sides of Equation (3.5) and Equation (3.6).

When the equations are written in the form given by Equation (3.8), the notation reinforces the point that a single integration routine can be used for many problems by simply respecifying the driving functions g_l. Note that while the gravitational force law (Equation 3.1) has no explicit time dependence, a general-purpose integrator has no problem advancing functions of the form $f_i(t, \mathbf{W})$ that do contain explicit time dependences. Certainly, when it seems that a specialized N-body algorithm is a hopelessly dense thicket of arcane techniques, it helps to keep in mind that at the end of the day, one is just integrating a fixed set of first-order ordinary differential equations.

Revisiting the discussion from the previous chapter, recall that an ordinary differential equation describes how a dependent variable, such as a particle's position or velocity, changes smoothly in response to variation of the independent variable (which in the N-body problem is the time). The differential equations are a recipe that describes a continuous sequence of changes, and in order to obtain the particles' trajectories, the specification of $6N$ boundary values is required. Since, in general, the positions and velocities of the particles are all known at a specified starting moment, the N-body problem is an "initial value problem."

The simplest solution algorithm for the gravitational N-body problem proceeds by constructing a basic finite difference representation of the differential equations (3.8) over the interval $h \equiv \Delta t \equiv t^{n+1} - t^n$

$$W_l^{n+1} = W_l^n + h\, g_l\big(W_1^n, \ldots, W_l^n, \ldots, W_{6N}^n\big) = W_l^n + h\, g_l(\mathbf{W}^n). \tag{3.9}$$

With this formula, the updated set of dependent variables, W_l^{n+1}, is computed using only information available at the initial time, t^n. (Note that the superscripts n and $n+1$ are time step indices, and do not indicate powers of n.) By repeatedly applying the finite difference approximation (Equation 3.9), particle trajectories are marched forward in intervals of length h, and the solution is built up as a set of explicit updates to the initial positions and velocities. This straightforward approach is known as *Euler's method*, and although it is not appropriate for production work, it is well worth coding up to form the simplest possible incarnation of an N-body program. If you are new to numerical methods, there is no substitute for such a hands-on introduction.

Any finite-difference scheme is necessarily an approximation to the continuous variations dictated by the actual differential equations, and for the N-body problem, we get an enormous advantage from the global conservation of energy and angular momentum. That is, E_{tot} and $\mathbf{L}_{\text{tot}}$ both remain constant as the particles trace through their trajectories, where

$$E_{\text{tot}} = \frac{1}{2}m_{\text{tot}}v_{\text{com}}^2 - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1;j\neq i}^{N}\frac{Gm_im_j}{r_{ij}} + \sum_{i=1}^{N}\frac{1}{2}m_iv_i^2, \tag{3.10}$$

and

$$\mathbf{L}_{\text{tot}} = \sum_{i=1}^{N} m_i(\mathbf{x}_i \times \mathbf{v}_i), \tag{3.11}$$

where $v_{\rm com}$ is the velocity of the center of mass of the system, r_{ij} is the distance between particles i and j, and

$$m_{\rm tot} = \sum_{i=1}^{N} m_i \,. \tag{3.12}$$

The degree to which these quantities are conserved in a numerical solution scheme, for example,

$$\Delta E = \frac{E^{\rm final} - E^{\rm initial}}{E^{\rm initial}}, \tag{3.13}$$

is a measure of the fidelity to which the solution is being obtained. Good conservation of $E_{\rm tot}$ and $\mathbf{L}_{\rm tot}$ greatly bolsters confidence in the integrity of a numerical N-body integration. In general, however, for an arbitrary set of ODEs, convenient diagnostics like E_{tot} and $\mathbf{L}_{\rm tot}$ are not available. To stay on the right track, one needs to monitor accuracy by evaluating the fractional change in the solution obtained with varying step sizes, h.

Figure 3.1 shows the fractional change in energy of an Earth-mass planet started in a circular 1 AU orbit about a star of 1 $M_\odot$, as described by Equation (3.5) and Equation (3.6), numerically integrated with Euler's method for 100 years. The updates to $\mathbf{W}$ are performed with time steps of various lengths Δt. It is clear that at first, as

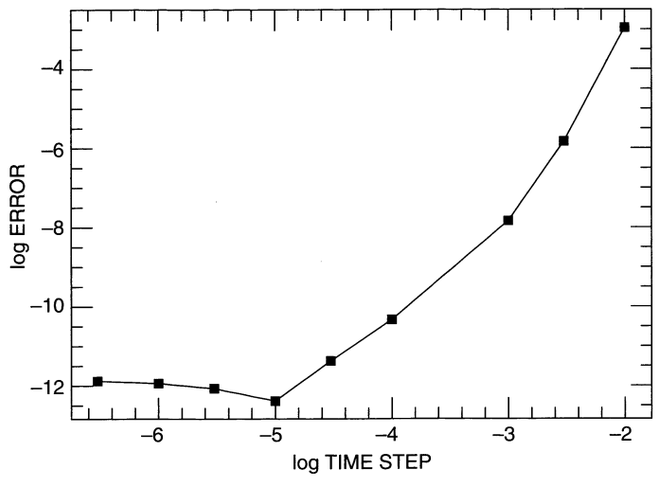

FIGURE 3.1 The log of the absolute value of the error ΔE is plotted as a function of the log of the time step (in years) for the integration by Euler's method of the orbit of a planet for 100 years at 1 AU from a star of 1 solar mass.

Δt decreases, the solution becomes increasingly accurate, albeit at an ever-increasing computational cost.

Curiously, however, Figure 3.1 also shows that the benefit obtained from using ever-smaller time step intervals saturates at a time step $\Delta t = 1 \times 10^{-5}$. This results from the computer's ability to represent numbers to only a finite number of decimal places. In the example shown, the variables are defaulted to "double precision," i.e., 14-digit precision. As the error ΔE approaches this limiting resolution, successive updates are given stochastic contributions resulting from roundoff error and the accuracy goes down. The combination of roundoff error and excessive computer time demands effectively shuts down the ability of Euler's method to follow the system for periods exceeding 10^5 years (which would, in this case, require 67 hours on a fast 2004-era desktop computer) and, more devastatingly, would entail an error buildup $\Delta E \sim 1 \times 10^{-1}$.

To trace the oppressive error buildup produced by Euler's method, we note that the exact value of the advanced dependent variable W_l^{n+1} (Equation 3.9) is given by a Taylor series composed of an infinite number of terms

$$W_l^{n+1} = W_l^n + \frac{h}{1!}\frac{dW_l}{dt} + \frac{h^2}{2!}\frac{d^2W_l}{dt^2} + \cdots + \frac{h^n}{n!}\frac{d^nW_l}{dt^n} + \cdots \tag{3.14}$$

where the derivatives are evaluated at time n. Note that the first two terms in this series comprise Euler's formula. When Euler's method is employed, the rest of the terms sum to give the total error in the finite difference estimate of W_l^{n+1}. The numerical time derivative, as determined by Euler's method, is said to be "first-order" accurate.

The fundamental unworkability of Euler's formula stems from its asymmetry; the increment $h dW_l/dt \equiv h g_l(t^n, \mathbf{W^n})$ to the dependent variable is based solely on the value of $\mathbf{W}$ evaluated at the beginning of the interval h. Any nonlinearity in g implies an inaccuracy in the updated value of the dependent variables. The key to improving Euler's method is the realization that the derivative function g_l, in general, can be computed for any trial values of $(t, \mathbf{W})$, where here we include the implicit time dependence in g_l. A higher order (and faster and more accurate) integration scheme, thus, can be designed in which the behavior of $g_l(t, \mathbf{W})$ in essence provides better guidance for the augmentation of W_l^n. In drawing together information regarding the topography of $g_l(t, \mathbf{W})$, a refined estimate for the slope $\overline{g_l(t, \mathbf{W})}$, needed to accurately update W_l over a particular interval h, can be obtained in terms of a weighted sum of values $g_l(t, \mathbf{W})$. Implementations of this strategy are called Runge–Kutta methods and the weightings applied to each of the k estimates of the slope can be tuned to provide cancellation of error terms in the Taylor series up to order $k + 1$.

The simplest Runge–Kutta integration scheme consists of first using the slopes $\mathbf{g}(t^n, \mathbf{W^n})$ and the Euler method to estimate the values $\mathbf{W}(t^n + h/2)$ at a point halfway through the desired interval h

$$\mathbf{W}\left(t^n + \frac{h}{2}\right) = \mathbf{W}_b = \mathbf{W}(t^n) + \frac{h}{2}\mathbf{g}(t^n, \mathbf{W}^n). \tag{3.15}$$

The values $\mathbf{W}_b$ better typify the true values of $\mathbf{W}$ within the interval $(t^n, t^n + h)$. A refined estimate $\mathbf{g}_b(t^n + h/2, \mathbf{W}_b)$, thus, provides a more accurate value for the slope required to augment $\mathbf{W}$ over the interval h.

A particular implementation of a Runge–Kutta integration formula is the heavily used classical fourth-order scheme, which consists of the following sequence of operations, involving four evaluations of the derivative functions $\mathbf{g}(t, \mathbf{W})$

$$\mathbf{f}_a = \mathbf{g}(t^n, \mathbf{W}^n) \tag{3.16}$$

$$\mathbf{W}_b = \mathbf{W}^n + \frac{h}{2}\mathbf{f}_a \tag{3.17}$$

$$\mathbf{f}_b = \mathbf{g}\left(t^n + \frac{h}{2}, \mathbf{W}_b\right) \tag{3.18}$$

$$\mathbf{W}_c = \mathbf{W}^n + \frac{h}{2}\mathbf{f}_b \tag{3.19}$$

$$\mathbf{f}_c = \mathbf{g}\left(t^n + \frac{h}{2}, \mathbf{W}_c\right) \tag{3.20}$$

$$\mathbf{W}_d = \mathbf{W}^n + h\mathbf{f}_c \tag{3.21}$$

$$\mathbf{f}_d = \mathbf{g}(t^n + h, \mathbf{W}_d) \tag{3.22}$$

$$\mathbf{W}^{n+1} = \mathbf{W}^n + \frac{1}{6}h\mathbf{f}_a + \frac{1}{3}h\mathbf{f}_b + \frac{1}{3}h\mathbf{f}_c + \frac{1}{6}h\mathbf{f}_d. \tag{3.23}$$

That is, a single integration step proceeds by evaluating (1) the initial slope, (2, 3) two successive slope estimates at the interval midpoint, and (4) an estimate of the slope at the end of the step interval. The four slopes are then combined, with weights $\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6}$ to produce an average slope capable of accurately bridging the interval h with a single Euler-method step.

To see in detail how this works, consider the concrete example of a nonlinear, first-order ordinary differential equation

$$\frac{dy}{dx} = f(x, y) = \left(x + \frac{y}{4}\right)^{1/2}, \tag{3.24}$$

to be integrated from the initial conditions $x = 1$, $y(1) = 1$ to $x = 2$. Using Euler's method to step across the interval yields a value

$$y(2) \sim y(1) + \Delta x \left[x(1) + \frac{y(1)}{4}\right]^{1/2} = 2.118. \tag{3.25}$$

The second-order Runge–Kutta formula gives a refined intermediate value $y(1.5) \sim 1.559$ and a revised slope estimate $f(1.5, 1.559) = 1.375$, yielding a revised end-point solution $y(2) \sim y(1) + \Delta x f(1.5, 1.559) = 2.37468$. The fourth-order Runge–Kutta Scheme (3.23) produces a value $y(2) = 2.37523$, which is quite close to the true value.

Fourth-order Runga–Kutta integration is at its best when ease of programming is of greater concern than actual computer speed. For example, one might want to check whether two planets orbiting an oblate star are in resonance or, alternately, one might

be interested in tracing the orbital behavior of stars in a particular realization of a galactic potential. In both of these cases, the functions $g_l(t, \mathbf{W})$ require modification from those in Equation (3.1), which might be time consuming to implement in a specialized preexisting code.

Often, in order to make a Runga–Kutta integration practical, one needs the ability to adaptively increase and decrease the time step as the integration proceeds. When bodies are close together or, more generally, when $g_l(t, \mathbf{W})$ is large, a small time step is required to maintain accuracy. If g_l becomes very small, a calculation can effectively grind to a halt if the time step is too small. *Step doubling* provides a simple way to implement adaptive time step control. One first advances the solution vector $\mathbf{W}^n$ by taking two Runge-Kutta time steps to cover an interval $2h$. Upon completion of the two steps, one has an estimate of the solution vector, which we call $\mathbf{W}_1$. One then makes a second estimate, $\mathbf{W}_1'$ by bridging the same interval with one large step of duration $(2h)$. Because the same starting derivative can be used in both cases, computing the time step of length $2h$ adds an overhead factor of $11/8 = 1.375$ to the computational cost.

Step doubling presents us with two estimates of the solution at $t = t^n + 2h$ and, thus, allows us to obtain a fractional estimate of the error

$$\delta_e = \left|\frac{(\mathbf{W}_1 - \mathbf{W}_1')}{\mathbf{W}_1}\right|. \tag{3.26}$$

If $\delta_e \geq \delta_{max}$, where δ_{max} is a predefined relative time step accuracy criterion, then adequate convergence has not been achieved. In this event, the time step is decreased (repeatedly if necessary) and the estimates $\mathbf{W}_1$ and $\mathbf{W}_1'$ are recomputed until $\delta_e \leq \delta_{max}$. On the other hand, if $\delta_e \leq \delta_{min}$, where δ_{min} is another predefined criterion defined such that $\delta_{min} < \delta_{max}$, then an attempt can be made to integrate the next $2h$ interval with a larger time step. The simple adaptivity imparted by step doubling can dramatically increase the efficiency of an integration.

Often, however, in a standard N-body integration, both small and large time steps are required simultaneously as one advances the set of particles. In a globular cluster, the tight binary stars that provide a source of gravitational energy for the cluster center have periods measured in hours, while loosely bound stars on the distant fringes take literally millions of years to fall from one side of the cluster to the other. In such cases, simple adaptive methods like time step doubling will not be adequate, and more specialized techniques, such as following each particle with its own time step, must be adopted.

3.3 THE DESCRIPTION OF ORBITAL MOTION IN TERMS OF ORBITAL ELEMENTS

A common problem in N-body dynamics involves the motion of one or more low-mass objects in orbit around a central body of considerably higher mass. The next four sections are primarily devoted to this problem. The motion of two point masses with any relative position and velocity can be described exactly, and it is no exaggeration to say that this result is one of the high-water marks of astronomy. For a pair of particles,

Equation (3.1) reduces to a single expression that encapsulates the time evolution of the separation $\mathbf{r}$ of the two bodies

$$\frac{d^2\mathbf{r}}{dt^2} = \frac{-G(m_1 + m_2)}{r^3}\mathbf{r}, \tag{3.27}$$

where $r = |\mathbf{r}|$. Conservation of angular momentum requires that the motion of the two bodies takes place in a plane, so we can switch to a polar coordinate system in which the origin is located at the position of mass m_1, and the angular position θ of mass m_2 is measured relative to a specified (but arbitrary) reference direction. It is then straightforward to show that the equation of relative motion for the bodies becomes

$$\frac{d^2r}{dt^2} - r\left(\frac{d\theta}{dt}\right)^2 = \frac{-G(m_1 + m_2)}{r^2}. \tag{3.28}$$

Because the specific angular momentum $l = r^2\dot{\theta}$ is conserved, the above equation can be written

$$\ddot{r} = \frac{-G(m_1 + m_2)}{r^2} + \frac{l^2}{r^3}, \tag{3.29}$$

indicating that the separation of the particles can be viewed as governed by the attractive force of gravity, as well as a repulsive restoring force provided by the angular momentum.

The time dependence in the equation of motion can be removed with the goal of obtaining a solution $r(\theta)$ that describes the entire locus of the relative positions of the particles in space. This is done by defining $u = 1/r$ and recasting Equation (3.28) as

$$\frac{d^2u}{d\theta^2} + u = \frac{G(m_1 + m_2)}{l^2}. \tag{3.30}$$

The solution to this ordinary, linear, second-order differential equation is:

$$u = \frac{G(m_1 + m_2)}{l^2}[1 + e\cos(\theta - \varpi)], \tag{3.31}$$

or, rewriting using r,

$$r = \frac{p}{1 + e\cos(\theta - \varpi)}. \tag{3.32}$$

The quantity $p = l^2/[G(m_1 + m_2)]$ is (rather antiquatedly) known as the *semilatus rectum*. The quantities e and ϖ are the two required constants of integration. The full generic range of two-body motion is provided by varying the constant e. The phase ϖ simply describes the orientation of the motion relative to the arbitrarily chosen reference direction. Because $\cos(\theta) = -\cos(\theta + \pi)$, we lose no generality in restricting the constant e (called the orbital eccentricity) to values $e > 0$.

When $e = 0$, the motion is simply a circle of radius $r = p$ with the familiar expression of the circular orbital velocity $v = \sqrt{G(m_1 + m_2)/r}$. Whenever $0 < e < 1$, the vector separating the bodies explores the full 2π range of angle, and is, therefore, bounded. The motion traces out an ellipse of semimajor axis $a = p/(1 - e^2)$, with

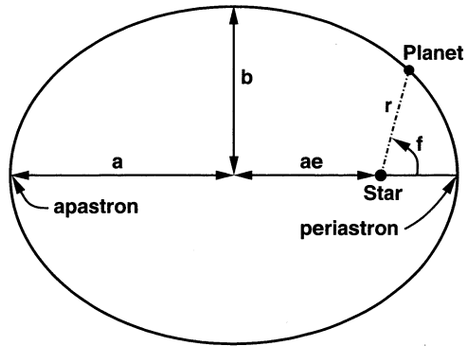

FIGURE 3.2 Diagram for an ellipse, illustrating the orbital elements a and e, and the semi-minor axis b.

the origin (the position of body m_1) located at one of the foci. For $e > 1$, the motion is hyperbolic, whereas $e = 1$ yields the separating case of a parabolic orbit.

The solution of the two-body problem indicates that, if the velocities and positions of the bodies are known at an initial time, then the subsequent motion is completely determined. The most analytically natural and intuitively pleasing description of the initial condition occurs when the orbit is described in terms of orbital elements (Figure 3.2 and Figure 3.3), which are a coordinate system basis that combines both positions and velocities. The full set of six orbital elements that describe a particle's position and velocity is given in the following list.

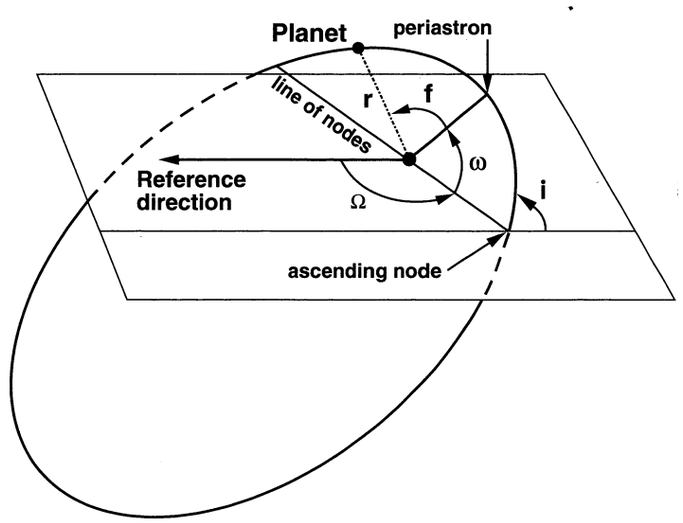

FIGURE 3.3 Diagram for an ellipse, illustrating the orbital elements Ω, ω, i, and f. The parallelogram is the reference plane inclined to the orbital plane by the angle i.

1. P, the period of the orbit, is related to the semimajor axis a via Kepler's Third Law

$$P^2 = \frac{4\pi^2}{G(m_1 + m_2)} a^3 \tag{3.33}$$

The quantity $n = 2\pi/P$, called the *mean motion*, is sometimes listed in lieu of P.

2. The eccentricity, e, determines the shape of the orbit. It is related to the fundamental constants of the system by

$$e = \left(1 + \frac{2El^2}{G^2(m_1 + m_2)^2}\right)^{1/2} \tag{3.34}$$

where E is the specific energy of the orbit

$$E = \frac{1}{2}v^2 - \frac{G(m_1 + m_2)}{r} = -\frac{G(m_1 + m_2)}{2a} \tag{3.35}$$

or, in terms of the semimajor axis a and the semiminor axis b,

$$e^2 = 1 - \frac{b^2}{a^2}. \tag{3.36}$$

3. M, the *mean anomaly*, is an angular measure that increases uniformly with time (a "clock hand") from 0 to 2π during the course of an orbit. M is related to the time through

$$M = n(t - T_{peri}), \tag{3.37}$$

where the *time of periastron passage*, T_{peri}, is the time at which m_1 and m_2 are at minimum separation, and n is the mean motion. From M and e, one can derive the angle f (*the true anomaly*), which describes the actual position of the planet in its orbit, relative to periastron, at time t.

4. The inclination, i, is the angle between an arbitrary reference plane and the plane of the orbit. In the convention shown in Figure 3.3, i is the angle between the orbital plane and the (x, y) plane of the (x, y, z) Cartesian coordinate system. In the solar system, the reference plane is usually taken to be that of the Earth's orbit.

5. The longitude of the ascending node, Ω, is defined using the *line of nodes*, which is the intersection line between the orbital plane and the (x, y) (reference) plane. Ω is then the angle between the line of nodes and an arbitrary reference direction in the reference plane, which in Figure 3.3 is the $-x$ direction.

6. The longitude of periastron, ϖ, is defined by

$$\varpi = \Omega + \omega \tag{3.38}$$

where ω, the *argument of periastron*, is the angle from the line of nodes to the periastron point. As such, ϖ is a dogleg angle, since it is composed as the sum of angles defined in two different planes.

The last three orbital elements fix the orbit's orientation in space. The transformations between the Cartesian system $(x, y, z, \dot{x}, \dot{y}, \dot{z})$ and the orbital element system $(P, M, e, i, \Omega, \varpi)$ are described in Murray and Dermott (1999). In general, if a system contains more than two bodies, then the concept of an elliptical orbit is only approximate, and all of the orbital elements associated with an orbiting body will vary with time. In this case, the state of the system defined at one particular epoch is referred to as a set of *osculating* orbital elements.

For simulations that are tied to actual observations of either solar system bodies or extrasolar planets, time is referenced to a continuous sequence of days called the Julian date (JD). A Julian day is defined as 86,400 seconds and the Julian date (or day number) is the number of such days that have elapsed since noon (universal time [UT]) on January 1, 4713 BC. Noon (UT) on New Year's Day 2000 corresponded to JD 2451545. The peculiar starting time of the Julian day sequence arises from the so-called Julian Cycle of 7980 years invented by Joseph Scaliger in 1583. The Julian Cycle and its starting date have no physical or astronomical significance, but the Julian date is, however, very widely used in astronomy. The conversion between JD and the calendar date is done using formulae that take leap years (and the 10 days omitted in October 1582 due to the conversion from the Julian to the Gregorian calendar) into account (see Montenbruck, 1989).*

The discovery of extrasolar planets (see, e.g., Marcy et al., 2000) is providing an exciting influx of opportunities for numerical few-body simulations that use the techniques described in this chapter. New planetary systems are currently being announced at a rate of several per month, and the number of known systems containing multiple planets is increasing quickly. With each discovery comes dynamical questions. Could terrestrial planets on habitable orbits exist within a particular system? Are there configurations that are consistent with the data and yet dynamically unstable? Are there alternate configurations of planets, which also provide good fits to the data?

A number of techniques have been used and proposed for detecting extrasolar planets — so far, the most successful have been the radial velocity and transit methods. Transits are detected by monitoring the stellar light output for the characteristic periodic dips in flux that occur when a planet passes in front of the star. This effect usually leads to a 1 to 2% decrease in brightness. The probability of observing a transit depends on the size of the star and the orientation and period of the orbit. For typical short-period "Hot Jupiter"-type planets with orbital periods of 3 to 5 days, transit probabilities are on the order of 10%. Transit measurements give the inclination i, which is the angle between the plane of the sky and the orbital plane, as well as P and the ratio of the planetary radius to the stellar radius.

* For phenomena, such as extrasolar planetary transits that require precise timing, the Julian date on Earth can be converted to the heliocentric Julian date at the solar system barycenter. For a given line of sight, this conversion accounts for the varying light travel time across the Earth's orbit during different parts of the year.

The stellar radial velocity half-amplitude, K, induced by one planet, m_2, orbiting the star m_1 is defined by

$$K = \left(\frac{2\pi G}{P}\right)^{1/3} \frac{m_2 \sin(i)}{(m_1 + m_2)^{2/3}} \frac{1}{\sqrt{1-e^2}} . \tag{3.39}$$

The inclination i is generally unknown unless the planet transits. Very accurate measurements of the Doppler shift of the absorption lines in the star's spectrum allow the velocity component of the star's motion along the line of sight to be determined at different times. Using the equations for Keplerian orbital motion, the star's line-of-sight velocity induced by the planet, as a function of time, is given by

$$v_r = K[\cos(f + \omega) + e\cos\omega], \tag{3.40}$$

where the *true anomaly*, f, is defined by

$$\tan(f/2) = \sqrt{\frac{1+e}{1-e}} \tan E/2 \tag{3.41}$$

and the *eccentric anomaly*, E, is computed by solving Kepler's equation at time t

$$\frac{2\pi}{P}(t - T_{peri}) = M = E - e\sin E . \tag{3.42}$$

Thus, from the observations of v_r vs. t, one obtains K, e, and P. The solution for E, along with the known e, allows one to determine the actual position of the planet in its orbit, relative to its periastron position, at a given time t. Equation (3.39) then gives the minimum mass of the planet, $m_2 \sin(i)$, as long as one has knowledge of the mass of the star and can assume that $m_2 << m_1$.

It is useful to remark on units. Imagine two 1-centimeter cubes of ice in space separated by 1 centimeter. A simple dimensional argument shows that if the cubes start from rest, they take approximately $\sqrt{\frac{1}{G}} \sim 1$ hour to come together. The cgs system of units (centimeters-grams-seconds) is not optimally tuned for a numerical solution of the problem, since 1 second is considerably smaller than the hour-long characteristic time. Indeed, for an N-body problem involving gram-sized masses separated by distances of a centimeter, an hour is a perfectly "natural" unit for measuring time, as it is the characteristic timescale over which significant evolution occurs.

In general, in an N-body problem, it is traditional to work in units where the gravitational constant

$$G = 1\frac{(LU)^3}{(TU)^2 MU} \tag{3.43}$$

where LU is the unit in which length is measured (i.e., cm, pc, AU, etc.), TU is the unit of time, and MU is the unit of mass. In general, one is free to choose two of the three measurement units. The demand that $G = 1$, then automatically selects the third unit. For example, to express the integration of a planetary system, we might choose $MU = 1M_\odot = 1.98911 \times 10^{33}$ gm and $TU = 1$ yr $= 31{,}557{,}600$ s. In this case, the unit of length is $LU = 5.0939 \times 10^{13}$cm.

3.4 THE FEW-BODY PROBLEM: BULIRSCH–STOER INTEGRATION

The need to integrate several bodies for hundreds to millions of orbits often arises in astronomy. One might wish to see, for example, whether a particular model planetary system is participating in a resonance. Or one might be interested in learning how the orbits of the planets would be affected if the solar system suffered an encounter with a passing star. Alternately, it might be of interest to study the dynamics of orbits within a specified galactic potential. For these types of problems, one requires numerical accuracy, flexibility, and speed. Bulirsch–Stoer integration (Stoer and Bulirsch, 1980, Section 7.2.14) performs very well on all three counts.

In the gravitational few-body problem, a single Bulirsch–Stoer integration step advances $6N$-coupled differential equations over a nonnegligible time interval while maintaining very high accuracy. In a simulation of a planetary system, a typical time step is often of the order of a tenth of an orbit of the shortest period planet in the system, and the fractional system energy accuracy across the time step is of order $\Delta E < 10^{-11}$.

The basic integration scheme for solving $d\mathbf{x}/dt = f(t, \mathbf{x})$ by Bulirsch–Stoer is the simple *modified midpoint* method for advancing a vector $\mathbf{x}(t)$ over a full time step $H = Nh$ by stringing together N equal substeps

$$\mathbf{x}_0 = \mathbf{x}(t) \tag{3.44}$$

$$\mathbf{x}_1 = \mathbf{x}_0 + hf(t, \mathbf{x}_0) \tag{3.45}$$

$$\mathbf{x}_n = \mathbf{x}_{n-2} + 2hf(t + [n-1]h, \mathbf{x}_{n-1}) \quad \text{for } n = 2, \ldots, N \tag{3.46}$$

$$\mathbf{x}(t+H) = \frac{1}{2}[\mathbf{x}_N + \mathbf{x}_{N-1} + hf(t+H, \mathbf{x}_N)] . \tag{3.47}$$

The subscripted $\mathbf{x}_n$s are approximate intermediate values at times $t + nh$. The error incurred by estimating $\mathbf{x}(t+H)$ with the above formulas can be expressed as a power series that contains terms to only even powers of h. This special property means that successively refined estimates of $\mathbf{x}(t+H)$ from Equation (3.47) (if they are done using values of N divisible by 2) can be combined in a weighted average to form a final estimate for $\mathbf{x}(t+H)$ that has a very high ratio of accuracy to computational effort.

The heart of the Bulirsch–Stoer method consists of generalizing this weighted average to identify it with a polynomial (a rational function could also be used) extrapolation from successive estimates of $\mathbf{x}(t+Nh)$, with the same value of H, but with different values of N. Thus, we evaluate $\mathbf{x}(t+H)k$ times, with $N = 2, 4, 6, 8, \ldots$, fit the results to a polynomial of order $k-1$, and then extrapolate to the value, $\mathbf{x}_{N\to\infty}$, that would occur if the step-size h were zero (Figure 3.4). This process is known as Richardson extrapolation.

The polynomials are given by

$$\mathbf{x}_{t+H}(h) = \mathbf{a}_0 + \mathbf{a}_1 h + \mathbf{a}_2 h^2 + \cdots + \mathbf{a}_k h^{k-1}. \tag{3.48}$$

The polynomial coefficients $\mathbf{a}_0 \ldots \mathbf{a}_{k-1}$ can be extracted from Lagrange's well-known formula that gives the unique polynomial of degree $k-1$ passing through points

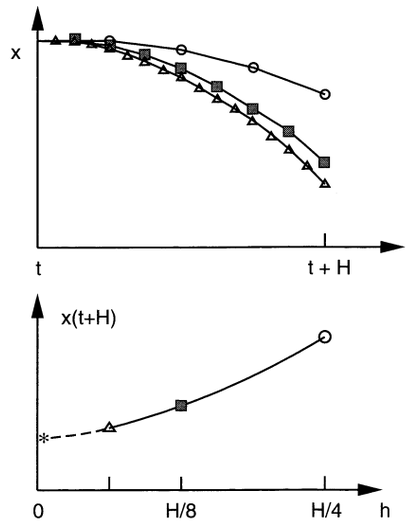

FIGURE 3.4 Schematic diagram for Richardson extrapolation. *Upper diagram*: the function x is integrated from time t to time $t + H$ using a subinterval $h = H/4$ (*open circles*), $h = H/8$ (*filled squares*), and $h = H/16$ (*open triangles*). *Lower diagram*: the values of $x(t + H)$ are plotted as a function of h (*solid line*) and then extrapolated to $h = 0$ (*dashed line*).

$h_1 \ldots h_k$ with values $\mathbf{x}_{t+H}(h_1), \mathbf{x}_{t+H}(h_2), \ldots, \mathbf{x}_{t+H}(h_k)$

$$\mathbf{x}_{t+H}(h) = \frac{(h-h_2)(h-h_3)\ldots(h-h_k)}{(h_1-h_2)(h_1-h_3)\ldots(h_1-h_k)}\mathbf{x}_{t+H}(h_1) + \frac{(h-h_1)(h-h_3)\ldots(h-h_k)}{(h_2-h_1)(h_2-h_3)\ldots(h_2-h_k)}\mathbf{x}_{t+H}(h_2) + \cdots + \frac{(h-h_1)(h-h_2)\ldots(h-h_{k-1})}{(h_k-h_1)(h_k-h_2)\ldots(h_k-h_{k-1})}\mathbf{x}_{t+H}(h_k). \quad (3.49)$$

Once the coefficients in Equation (3.48) have been determined, the expression is simply evaluated at $h = 0$ to obtain $\mathbf{x}_{t+H}(0)$.

The final step of the Bulirsch–Stoer process is the determination of numerical convergence. The whole procedure is redone with a different value of k, e.g., 10 evaluations of $\mathbf{x}(t + H)$ by the modified midpoint scheme instead of 8.† The two estimates of $\mathbf{x}_{N\to\infty}$ are compared. If the fractional difference is less than a given convergence criterion, for example 10^{-13}, then the solution for time step H is considered converged. If not, k is increased again until convergence is obtained. If the value of k becomes unacceptably large and convergence has still not been obtained, it may be necessary to reduce the basic time step H. Note that this convergence procedure is done at every time step H, so a single Bulirsch–Stoer time step requires a very large number of numerical operations. Why then use Bulirsch–Stoer? It turns out that for a given required accuracy, the Bulirsch–Stoer integration is several times faster than a fourth-order Runge–Kutta method.

† In the *Numerical Recipes in Fortran: The Art of Scientific Computing* routine *bsstep*, the sequence of k values is 2, 4, 6, . . . , 16.

For practical work, Equation (3.49) can be replaced by a more straightforwardly coded recursive formula, such as Neville's algorithm (Press et al., 1992, Sections 3.1, 16.4). Here we simply emphasize that once the polynomial has been constructed at $h_1 \ldots h_k$ using the known values $\mathbf{x}_{t+H}(h_1), \mathbf{x}_{t+H}(h_2), \ldots, \mathbf{x}_{t+H}(h_k)$ returned from successive applications of the modified midpoint method, we can simply evaluate the value of the polynomial at $h = 0$, which provides our estimate $\mathbf{x}_{N\to\infty}$.

In summary, Bulirsch–Stoer is the preferred method if high accuracy in orbital integrations is required. It is able to handle situations where two orbiting planets make a close encounter. If computational speed is the prime requirement, then the symplectic technique (Section 3.6) can be used, but in general it cannot handle close encounters. In most situations, in fact, the Bulirsch–Stoer should be used for few-body integrations.

3.5 LYAPUNOV TIME ESTIMATION

A persistently remarkable phenomenon in the gravitational N-body problem is the presence of chaos. Chaotic behavior (for a detailed introduction, see Gleick, 1987) stems from the fact that the trajectories of bodies in the N-body problem, which are started very close to one another in phase space (i.e., with very similar position and velocity initial conditions), can, in many cases, diverge exponentially in terms of their phase space distance as time goes on. Eventually, this divergence causes the trajectories to develop complete independence. In the gravitational N-body problem, the timescale that is required for the trajectories to diverge depends sensitively on the details of all the initial conditions (particle masses, velocities, and positions.) Chaos can be observed in any system having more than two bodies, but it is important to remember that not every N-body system will exhibit measurable chaos. Furthermore, among systems that are chaotic, there is a wide range of severity in the chaotic behavior. In some cases, the trajectories, while divergent, will be confined to a narrowly defined region of phase space. Such motion is termed "weakly chaotic." In other cases, initially neighboring orbits are completely dispersed after a period of time. This represents the regime of strong chaos.

As an example, consider the following situation. We adopt our standard initial model containing the Sun and the eight planets in the present-day solar system and, on a somewhat mischievous whim, we insert a clone of the Earth into an orbit whose osculating orbital elements are identical to the Earth's apart from a single exception. We advance the mean anomaly, M, of the Earth's twin by π, which places it initially on the other side of the Sun from the true Earth. We then integrate this augmented version of the solar system forward in time for a million years using the Bulirsch–Stoer program *integrator.f*.

Contrary to what one might expect, the respective motions of the Earth and its twin are not immediately mutually unstable. Rather, the two planets participate in a variation of the horseshoe orbit, in which, during their periodic close encounters, they

exchange energy and angular momentum. After any given time, the planet with the smaller osculating semimajor axis, a_1, will approach and will attempt to pass the planet with instantaneously larger semimajor axis, a_2. In the rotating frame, the inner planet is pulled forward, increases its energy and angular momentum, and, hence, increases its semimajor axis. The outer planet experiences the reverse behavior, winds up with a smaller semimajor axis, and the process repeats for the million-year duration of the simulation.

The chaotic trajectories arise largely from the hair-trigger sensitivity within the dynamics of the close encounter. Any slight difference in momentum prior to the encounter is amplified by the encounter itself. Successive differences between the two orbital paths are thereby magnified as the two planets pass energy and angular momentum back and forth, and the compounding of amplifications leads naturally to an exponential divergence in the two trajectories.

To quantify the orbital divergence, we can keep track of some measure of the phase space distance between two orbits. There are many potential definitions of the phase space distance d. The physical distance between the particles could be used, for example, or alternately, one could define $d = e_2 - e_1$. An exponential increase in phase space separation between the two trajectories can be written

$$d(t) = d(0)e^{\gamma(t-t_0)}, \tag{3.50}$$

where the constant γ is called the maximum Lyapunov characteristic exponent. (For a more detailed discussion of the characteristic exponents in gravitational few-body systems, see Lecar et al., 1992.) The exponent γ is inversely related to the characteristic (or e-folding) time for neighboring trajectories to drift apart, and if an orbit is chaotic, $\gamma > 0$. (A negative value for γ indicates converging trajectories.)

A practical scheme for measuring γ needs to maintain a local comparison of two trajectories. Once the phase space distance d is large, the accumulation of further phase space separation is not a measure of local conditions. A simple way to keep the estimation process local is to use the so-called shadowing method. A test particle is integrated from the initial starting condition that one wishes to test for chaos. A second shadow particle with slightly different initial conditions (in which the eccentricity, say, differs by one part in 10^6) is integrated alongside for a time Δt. At the close of the time interval, one estimates the exponent γ from

$$\gamma_{est}^1 = \frac{\ln[d(\Delta t)/d(0)]}{\Delta t}. \tag{3.51}$$

At the end of the time step, the shadow particle is returned to a phase space distance $d(0)$ away from the test particle, and the process is repeated. After n trials, the estimated value of the Lyapunov exponent is:

$$\gamma_{est}^n = \sum_{i=1}^{n} \frac{\ln[d(\Delta t)/d(0)]}{n\Delta t}. \tag{3.52}$$

As a specific example, Figure 3.5 and Figure 3.6 show the results of the integration of a chaotic system using a Bulirsch–Stoer procedure (*integrator.f*). Two planets are assumed to be in orbit around a star of 1 solar mass. Their initial orbital periods are

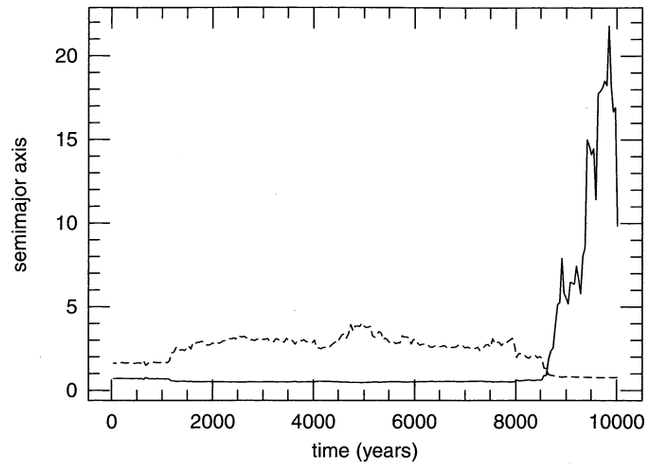

FIGURE 3.5 Semimajor axis (in astronomical units) as a function of time from an integration of the orbits of two planets around a star of 1 solar mass. *Solid curve*: Planet 1 whose initial orbital period was 250 days. *Dashed curve*: Planet 2 whose initial orbital period was 660 days. Note that the orbits cross at about 8600 years.

250 and 660 days, their orbital eccentricities are .28 and .27, their inclinations are zero (that is, they are in the same plane), and their masses are 1.89 and 3.75 Jupiter masses, for the inner and outer planet, respectively. Figure 3.5 shows the semimajor axes of the two planets as a function of time with the standard orbital parameters.

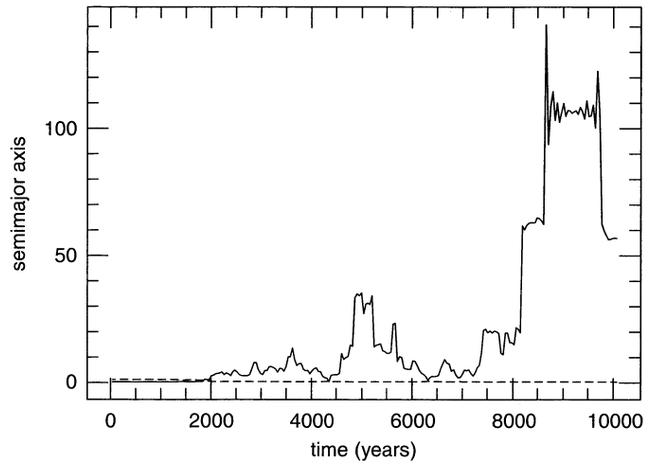

FIGURE 3.6 This figure is the same as Figure 3.5 except that the initial eccentricity of Planet 1 is changed from 0.28 to 0.28001. Note that the first orbit crossing occurs just before 2000 years.

In contrast, Figure 3.6 shows the results from a calculation in which all parameters are the same as the standard values, except that the eccentricity of the inner planet is increased from 0.28 to 0.28001. Clearly a change in the system that is smaller than the probable error of measurement of an eccentricity leads to a widely diverging result.

3.6 SYMPLECTIC INTEGRATION

The discussion in the previous sections has emphasized that many problems of astronomical interest involve the long-term behavior of small bodies orbiting a much larger parent body. The solar system provides a perfect example. While the orbit of a planet (the Earth, say) is well-described by a Keplerian ellipse over long timescales, the gravitational perturbations from other planets cannot be neglected. Indeed, the question of how the planets affect one another, and how the mutual interactions affect the long-term stability of the solar system has been a topic of intense interest from the time of Sir Isaac Newton onward.

The Newtonian gravitational interaction between two arbitrary point masses has an exact solution. This situation does not generally hold for systems containing three or more bodies. The underlying reason for the inherent difficulty in obtaining the motion for three gravitating objects was established by Jules Henri Poincaré in 1888 (see Murray and Dermott, 1999, Section 3.1 and 9.1), who showed that the three-body problem in nonintegrable. That is, it is formally impossible to write down a simple closed-form analytic solution for the future motion of a system of planets. Poincaré's work also emphasized the essential nonlinearity of the few-body problem in which the smallest change in initial conditions leads to completely different motions after long periods of time. Any solution for the motion of objects in the gravitational few-body problem, therefore, must be regarded as a sample from a statistical ensemble of equivalent solutions.

The advent of fast computers has revolutionized the problem of computing long-term planetary dynamics. At the time of this writing, a fast desktop computer, running for months on end, is capable of integrating the motion of the planets of the solar system for the entire lifetime of the Sun. Continued relentless increases in processor speed indicate that such calculations will eventually become interactive. The ability to do extremely long-term integrations has also benefited from so-called symplectic map algorithms that take specific advantage of the properties of near-Keplerian motion. The essence of the symplectic map is that the motion of a planet is divided into a Keplerian part and a non-Keplerian perturbative "kick" resulting from the interaction with bodies in the system other than the central body. Because the Keplerian part of the motion has an analytic solution, the computer can avoid wasting its time rediscovering the Keplerian ellipse on every orbit.

The symplectic algorithm relies on Hamilton's formulation of Newton's laws of dynamics (for a review, see Goldstein, 1980). The time development of the positions $\mathbf{q_i}$ and momenta $\mathbf{p_i}$ of the particles are written as a set of coupled first-order differential equations

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i} \tag{3.53}$$

$$\frac{-dp_i}{dt} = \frac{\partial H}{\partial q_i} \tag{3.54}$$

where H is the Hamiltonian for the N-body problem

$$H = \sum_{i=0}^{N-1} \frac{p_i^2}{2m_i} - \sum_{i<j} \frac{Gm_i m_j}{r_{ij}}. \tag{3.55}$$

Note that H is simply the total energy and, thus, is conserved. The idea behind the symplectic map is to separate the total Hamiltonian into a part corresponding to the Keplerian motion and a part that arises from the gravitational interactions with the other bodies

$$H = H_{Kepler} + H_{interaction}. \tag{3.56}$$

A Keplerian Hamiltonian is one that can be written in the form

$$H_{Kepler} = \frac{p^2}{2m} - \frac{GMm}{r}. \tag{3.57}$$

By contrast, in Equation (3.55), the definitions of p_i relative to a fixed origin, and the r_{ij}s in terms of particle–particle separations means that when there are more than two particles in the system, the terms in Equation (3.55) cannot immediately be read off in the form required to compose H_{Kepler}. That is, the momenta in the N-body Hamiltonian (Equation 3.55) refer to the center of mass, and the particles are not executing Keplerian motion about the center of mass.

Fortunately, however, a set of Keplerian Hamiltonians can be split off from H if the motion of the bodies is cast into Jacobi coordinates (see Wisdom and Holman, 1991). The system consists of a heavy mass m_0 and $N-1$ light masses m_i orbiting it. The center of mass of the entire system is taken to be the first Jacobi coordinate. The second Jacobi coordinate is the separation vector between the first planet and the central mass. Subsequent coordinates correspond to the separation vectors of the ith body relative to the center of mass of the $i-1$ preceding bodies. That is, if the first Jacobi coordinate (the COM) is labeled $\mathbf{x}'_0$, then the subsequent Jacobi coordinates are given by

$$\mathbf{x}'_i = \mathbf{x}_i - \frac{1}{\eta_{i-1}} \sum_{j=0}^{i-1} m_j \mathbf{x}_j, \tag{3.58}$$

where $\eta_i = \sum_{j=0}^{i} m_j$ is a running sum of the masses.

The Jacobi momenta are given by

$$\mathbf{p}'_i = \left(\frac{\eta_{i-1}}{\eta_i}\right) \mathbf{p}_i - \left(\frac{m_i}{\eta_i}\right) \sum_{j=0}^{i-1} \mathbf{p}_j \tag{3.59}$$

and

$$\mathbf{p}'_0 = \sum_{j=0}^{N-1} \mathbf{p}_j. \tag{3.60}$$

These are formed by multiplying the mass factors

$$m'_0 = M_{\rm tot} = \eta_{N-1} \quad \text{for } i = 0 \tag{3.61}$$

and

$$m'_i = \eta_{i-1} m_i / \eta_i \quad \text{for } 0 < i < N \tag{3.62}$$

by $d\mathbf{x}'_i/dt$ (see Saha and Tremaine, 1994).

The essential utility of the Jacobi coordinates in describing a star surrounded by more than one planet dates back to Newton (1687), who was the first to consider how the gravitational forces between the planets perturb their Keplerian orbits around the Sun. In Book I, Section II, proposition 69, of the Principia, he wrote:

> *And hence, if several lesser bodies revolve about a greatest one, it can be found that the orbits described will approach closer to elliptical orbits, and the description of the areas will become more uniform* [. . .] *if the focus of each orbit is located in the common center of gravity of all the inner bodies.*

In modern terms, one says that to first order, orbits in a multiple-planet system are Keplerian when written in terms of Jacobi coordinates. Using the Jacobi coordinates, one finds that the full N-body Hamiltonian H can be written (see Murray and Dermott, 1999) in the desired form $H = H_{Kepler} + H_{interaction}$, where

$$H_{Kepler} = \sum_{i=1}^{N-1} \left(\frac{(p'_i)^2}{2m'_i} - \frac{Gm_i m_0}{r'_i} \right) \tag{3.63}$$

and

$$H_{interaction} = \sum_{i=1}^{N-1} Gm_i m_0 \left(\frac{1}{r'_i} - \frac{1}{r_{i0}} \right) - \sum_{0<i<j} \frac{Gm_i m_j}{r_{ij}}, \tag{3.64}$$

where r_{i0} is, as usual, the distance between particle i and the massive object.

As outlined by Wisdom and Holman (1991), the basic idea behind the symplectic integrator is to imagine that the Hamiltonian Equation (3.56) represents a map that steps the system from an initial state at $t = t_0$ to an evolved state at a time $t = t_0 + \Delta t$. Operationally, the map can be carried out in two distinct steps. First, the Jacobi coordinates of the planets are advanced through the Keplerian portion of their trajectories. This can be done rapidly by taking advantage of the analytic nature of the solution. Numerically, this is accomplished by noting that the position and velocity vectors for a planet can be efficiently advanced through Keplerian motion using Gauss' f and g functions (see, e.g., Danby, 1988), discovered by Gauss in 1801 when working on the recovery of the asteroid Ceres. As we explain below, the computation of $\mathbf{r}(t)$, and $\mathbf{v}(t)$ can be done very quickly

$$\mathbf{r}(t) = f(t, t_0)\mathbf{r}(t_0) + g(t, t_0)\mathbf{v}(t_0), \tag{3.65}$$

$$\mathbf{v}(t) = \dot{f}(t, t_0)\mathbf{r}(t_0) + \dot{g}(t, t_0)\mathbf{v}(t_0), \tag{3.66}$$

$$f(t, t_0) = \frac{a}{r_0}[\cos(E - E_0) - 1] + 1, \tag{3.67}$$

$$g(t, t_0) = (t - t_0) + \frac{1}{n}[\sin(E - E_0) - (E - E_0)], \tag{3.68}$$

where r_0 is the magnitude of the position vector at time t_0 and E is the solution to Kepler's Equation (3.42) at time t.

The Keplerian steps for each particle are bracketed by first-order course corrections, which account for the planet–planet perturbations. Somewhat surprisingly, these course corrections, or "kicks," can be applied accurately in a simple impulsive fashion, in which the changes to the Jacobian vector velocities of the particles are given by

$$\Delta \mathbf{v}'_i = \Delta t \left(\frac{d\mathbf{v}'_i}{dt} \right)_{\rm interaction}, \tag{3.69}$$

where

$$\left(\frac{d\mathbf{v}'_i}{dt} \right)_{\rm interaction} = \frac{1}{m'_i} \left(-\frac{\partial H_{\rm interaction}}{\partial \mathbf{r}'_i} \right). \tag{3.70}$$

$H_{\rm interaction}$ is written most compactly in the form shown in Equation (3.64), which mixes ordinary interparticle distances, r_{ij} and Jacobi coordinates, r'_i.

The derivatives of the interaction Hamiltonian with respect to the Jacobi position coordinates are complicated, as they require the interaction Hamiltonian to first be written entirely in Jacobi coordinates. Murray and Dermott (1999) report the analytic formula

$$\left(\frac{d\mathbf{v}'_i}{dt} \right)_{\rm interaction} = GM'_i \left[\frac{\mathbf{r}_i'}{r'^3_i} - \frac{\mathbf{r}_{0i}}{r^3_{0i}} \right] - \left(\frac{\eta_i}{\eta_{i-1}} \right) \sum_{j=1}^{i-1} \frac{Gm_j}{r^3_{ji}} \mathbf{r}_{ji} + \sum_{j=i+1}^{N-1} \frac{Gm_j}{r^3_{ij}} \mathbf{r}_{ij} - \frac{1}{\eta_{i-1}} \sum_{j=0}^{i-1} \sum_{k=i+1}^{N-1} \frac{Gm_j m_k}{r^3_{jk}} \mathbf{r}_{jk}, \tag{3.71}$$

where $M'_i = (\eta_i/\eta_{i-1})m_0$.

The code *symplectic.f* utilizes the foregoing ingredients to implement a simplified routine for integrating few-body problems. The code is best suited to situations in which one desires to understand the long-term effect of planet–planet perturbations in the absence of close encounters. An excellent example is the evolution of the orbital elements of the planets in the solar system over timescales of millions to billions of years. This version of the code breaks down when two masses come within 3 R_H of each other, where R_H is the "Hill sphere" radius of the larger object

$$R_H = a_i \left(\frac{m_i}{3m_0} \right)^{1/3}, \tag{3.72}$$

where a_i is the usual semimajor axis. Under such circumstances, the code must be augmented by a "regularization" technique (Section 3.8) or a switch to a Bulirsch–Stoer routine.

The input to the code is made in terms of the Keplerian orbital elements (P,M, i, e, ϖ, and Ω), along with the planetary and stellar masses. The code first transforms the

orbital elements into astrocentric (star-centered) Cartesian coordinates (Murray and Dermott, 1999, Section 2.8). It then builds transformation matrices M and $\mathrm{M^t}$, which enable rapid conversion back and forth between Jacobi coordinates and ordinary coordinates

$$[\mathbf{r}_j'] = [\mathrm{M}][\mathbf{r}_j], \tag{3.73}$$

with

$$[\mathrm{M}][\mathrm{M^t}] = 1. \tag{3.74}$$

Once the transformation matrices have been precomputed, the routine is ready to enter its main loop. Within this main loop, the Jacobian coordinates of the bodies are evaluated, and the bodies are advanced through the Keplerian portion of their orbits for time step Δt. The Keplerian advances are performed in Cartesian coordinates by making use of Gauss' f and g functions (Equation 3.67 and Equation 3.68). The Gauss f and g functions rely on knowing the eccentric anomaly $E(t)$. The eccentric anomaly is related to the mean anomaly, M, through Kepler's Equation (3.42), which can be solved using a Newton–Raphson root-finding scheme. Following the Keplerian step, the interaction Hamiltonians are used to further update the Jacobian velocities in accordance with Equation (3.69). This completes the time step. The method is readily converted to second-order by using an interval $\Delta t/2$ for the first and last time steps and Δt for all the others.

Again it should be emphasized that the symplectic technique works well for planetary orbits that are near Keplerian, i.e., there is little gravitational interaction between the planets themselves. The big advantage to the technique is that it can run up to 10 times faster than the Bulirsch–Stoer method for a given number of planetary orbits. The disadvantage is that once the orbits of two planets begin to approach each other, the method becomes inaccurate and breaks down. In this situation use of a Bulirsch–Stoer code is highly preferable.

3.7 *N*-BODY CODES FOR LARGE *N*

Our discussion so far has focused on the evolution of configurations such as planetary systems that have only a few bodies. When studying the evolution of few-body systems, one is often interested in the subtle interplay of perturbations that unfold over thousands or even millions of orbits. In such a case, accuracy is the paramount concern. The goal is to keep the conservation of energy and angular momentum in as strict order as possible. For larger systems, however, such as galaxies, one is faced with trying to understand the collective behavior of enormous numbers of particles. In the case of a galaxy, the number of orbits that one needs to follow is relatively small. Take the Sun as an example. The Sun's orbital period around the center of the galaxy is roughly 250 million years, meaning that the Sun has executed roughly 18 orbits since its formation. Clearly, the relevant problem for the collective dynamics of a galaxy concerns integrating an enormous number of particles, albeit for a relatively small number of orbits.

Two factors conspire to make the large N-body problem a challenge. The first concerns the number of interactions. The gravitational force on an individual body

is the sum of the attractions of all the other bodies. Computing the force for all N bodies at a single time requires the evaluation of $\frac{1}{2}N(N-1)$ square roots. If forces are computed individually, this means that the computer time required to solve an N-body problem scales as N^2. No matter how fast the processor, N^2 algorithms are bound to become prohibitive for most astrophysical systems of interest.

A second problem is that of multiple time scales. In a self-gravitating system, such as a star cluster, stars that enter into tightly bound binary orbits or are in a region dense with other stars experience large gravitational forces. To accurately follow these stars, one is required to use a small time step. Such a small time step, however, when applied to the system as a whole, leads to disaster, as the vast majority of particles are updated far more often than necessary. Much of the art in N-body methods consists of implementing clever schemes to avoid these difficulties.

Perhaps the simplest strategy for integrating the motion of an assortment of particles with widely varying accelerations is to use the so-called "leapfrog" method in conjunction with time step doubling. The appropriate time step for each particle is first estimated using the criterion

$$\Delta t_i \simeq \eta\sqrt{\frac{1}{a_i}}, \tag{3.75}$$

where a_i is the magnitude of the acceleration of the ith particle and η is a small multiplicative factor. In order to give each particle an appropriate time step, the largest time step Δt_s, estimated from Equation (3.75), is subdivided into multiples of two

$$\Delta t_i = \frac{\Delta t_s}{2^{n_i}}, \tag{3.76}$$

with n_i chosen to be the smallest integer value for which $\Delta t_i < \eta\sqrt{1/a_i}$ is satisfied. For a particular particle, i, the leapfrog method consists of a simple second-order integration procedure to advance the positions and velocities

$$\mathbf{r}_i^{n+1/2} = \mathbf{r}_i^{n-1/2} + \Delta t_i \mathbf{v}_i^n + \mathcal{O}\left(\Delta {t_i}^3\right), \tag{3.77}$$

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \Delta t_i \mathbf{a}_i^{n+1/2} + \mathcal{O}\left(\Delta {t_i}^3\right). \tag{3.78}$$

Note that in this pair of equations, the velocities and positions are defined at intervals separated by $\Delta t_i/2$ in order to maintain second-order accuracy, while maintaining the overall simplicity of a first-order algorithm, whereas, in general, a set of initial conditions will specify all of the velocities and positions at a single point in time. A series of integration steps can be initialized by

$$\mathbf{r}_i^{n+1/2} = \mathbf{r}_i^n + \frac{1}{2}\Delta t_i \mathbf{v}_i^n + \frac{1}{8}\Delta {t_i}^2 \mathbf{a}_i^n, \tag{3.79}$$

which preserves second-order accuracy.

To see how multiple time steps are handled, consider two particles, "A" and "B," within the overall aggregate of bodies, whose individual time steps have been found to be $\frac{\Delta t}{2}$ and Δt. At the beginning of the first time step ($t = 0$), we know both $\mathbf{r}$ and $\mathbf{v}$ for both particles. The startup formula (Equation 3.79) is used to advance the position vectors to $\frac{1}{4}\Delta t$ for "A," and $\frac{1}{2}\Delta t$ for "B." Once the particles are in place, the accelerations are computed based on the position at $\Delta t/4$ for "A" and $\Delta t/2$ for "B" (note the time asymmetry inherent in this force calculation). The acceleration felt by particle "A" is used to advance "A's" velocity to $t = \Delta t/2$. This velocity is in turn used to advance the position of "A" to $t = (3/4)\Delta t$. With the time asymmetry between "A" and "B" thus reversed, the velocity is advanced again to $t = \Delta t$ and, in doing so, second-order accuracy is recovered.

The original and updated positions for particle "A" are averaged together to obtain a position estimate for "A" centered at $t = \Delta t/2$. This position, as well as the original position for particle "B," can be used to compute an acceleration on "B," which is used to advance its velocity across the entire interval Δt, upon which, the velocities of both particles are set at $t = \Delta t$. This centered velocity can then be used to make a final advance of the position of particle "A" to $t = (\frac{5}{4})\Delta t$, at which point the cycle is ready to be run again. A simple generalization of this procedure up through the hierarchy of time step widths allows all of the particles to be updated in a synchronous fashion that maintains second-order accuracy of the method.

The second-order leapfrog method is often used for systems that have large numbers of particles and are subject to a large degree of internal dynamical dissipation. An excellent example would be a simulation of a collision between galaxies or even the evolution of a rich cluster of galaxies. When two galaxies approach each other, the large-scale kinetic energy associated with the motion of their individual centers of mass is transferred into small scale random motion of the individual particles within each galaxy. This process of draining energy from the largest scales into the smallest causes the orbits of binary galaxies to coalesce over periods of at most a few tens of orbits. To survey the overall structure of the final (generally ellipsoidal) remnant galaxy, one requires neither a fully exact rendering of the potential nor a high-order integration of the positions and velocities. On the other hand, for systems that encompass fewer objects, but many dynamical timescales, accuracy, and a lack of numerical dissipation are paramount.

If the number of particles that one wishes to follow is not too large, say $20 < N < 10^4$, then it is most advisable to adopt a technique that computes gravitational forces via the direct summation of the terms in Equation (3.1), but which integrates the particles themselves with less precision than either the symplectic integrator or the Bulirsch–Stoer scheme. Problems in this regime might include, for example, the dynamical evolution of clusters of stars or the tidal disruption of dwarf galaxies. For cases of this sort, we suggest adopting a straightforward scheme for direct integration that combines the method of Aarseth (1985) for advancing particles in their trajectories, and the algorithm of Ahmad and Cohen (1973) for controlling the frequency with which the full sum of $\frac{1}{2}N(N-1)$ interactions in Equation (3.1) is updated. An optimized version of this method is available as the NBODY2 code written by Sverre Aarseth.

Aarseth's method is a version of a predictor–corrector scheme, which in its simplest form can be described as follows: Suppose the acceleration on a particle $\mathbf{a}_i(t_0)$ has been calculated from the right-hand side of Equation (3.1). Then at an advanced time $t = t_0 + \Delta t$, the velocity and position can be obtained from

$$\begin{aligned}\mathbf{v}_i(t) &= \mathbf{a}_i(t_0)\Delta t + \mathbf{v}_i(t_0)\\ \mathbf{r}_i(t) &= \frac{1}{2}\mathbf{a}_i(t_0)(\Delta t)^2 + \mathbf{v}_i(t_0)\Delta t + \mathbf{r_i}(t_0).\end{aligned} \tag{3.80}$$

One makes a provisional estimate of the radius at time t using the accelerations at t_0, recomputes the accelerations at t, and calculates an average acceleration $\bar{\mathbf{a}}_i = 0.5[\mathbf{a}_i(t_0) + \mathbf{a}_i(t)]$. This average is then used in Equations (3.80) to get the corrected positions and velocities at time t.

Aarseth's direct summation N-body algorithm hinges on representing the acceleration on a particular body of index i at time t in terms of a fourth-order polynomial based on knowledge of the acceleration at four previous times in the past, t_0, t_1, t_2, and t_3, with t_0 being the most recent

$$\mathbf{a}(t) = (((\mathbf{D}_4(t - t_3) + \mathbf{D}_3)(t - t_2) + \mathbf{D}_2)(t - t_1) + \mathbf{D}_1)(t - t_0) + \mathbf{a}_0, \tag{3.81}$$

where $\mathbf{a}_0$ is the acceleration at time t_0. This equation is a variant of the Lagrange interpolation formula. We identify $\mathbf{D}_0(t') = \mathbf{a}(t')$, where t' can be any past or future value of t. The "divided differences" $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{D}_3$, and $\mathbf{D}_4$ are defined as follows:

$$\begin{aligned}\mathbf{D}_1[t_j, t_k] &= \frac{\mathbf{D}_0(t_j) - \mathbf{D}_0(t_k)}{t_j - t_k} \quad (j = k - 1)\\ \mathbf{D}_2[t_j, t_k] &= \frac{\mathbf{D}_1[t_j, t_{k-1}] - \mathbf{D}_1[t_{k-1}, t_k]}{t_j - t_k} \quad (j = k - 2)\\ \mathbf{D}_3[t_j, t_k] &= \frac{\mathbf{D}_2[t_j, t_{k-1}] - \mathbf{D}_2[t_{k-2}, t_k]}{t_j - t_k} \quad (j = k - 3)\\ \mathbf{D}_4 &= \frac{\mathbf{D}_3[t, t_2] - \mathbf{D}_3[t_0, t_3]}{t - t_3}.\end{aligned} \tag{3.82}$$

Note that $\mathbf{D}_4$ involves the acceleration at the *advanced* time t, where information is not initially available. This point is needed because the expansion (Equation 3.81) is fourth order and, thus, five points are needed to define it. The predictor–corrector method is used to obtain quantities at this point.

The fourth-order polynomial for the acceleration, $\mathbf{a}$, can be converted into the first four terms of a Taylor series approximation of the acceleration in the neighborhood of t_0. When the acceleration of a particle is expressed as a Taylor series, one can immediately integrate the terms once with respect to time to obtain the particle velocity, and then again to obtain the position. The key idea underlying the method is that the instantaneous positions of particles at time t_i that are necessary to compute the acceleration of the particle i under consideration can be estimated at arbitrary t_i by integrating their acceleration polynomials. Each particle can then maintain a separate time step that is appropriate to its dynamical environment, allowing for a large increase in efficiency.

Equating the acceleration polynomial to a Taylor series, we find the following expressions for the successive derivatives of the acceleration at time $t = t_0$

$$\frac{d\mathbf{a}}{dt} = ((\mathbf{D}_4(t_0 - t_3) + \mathbf{D}_3)(t_0 - t_2) + \mathbf{D}_2)(t_0 - t_1) + \mathbf{D}_1 \tag{3.83}$$

$$\frac{d^2\mathbf{a}}{dt^2} = 2!(\mathbf{D}_4((t_0 - t_1)(t_0 - t_2) + (t_0 - t_2)(t_0 - t_3) + (t_0 - t_1)(t_0 - t_3)) + \mathbf{D}_3((t_0 - t_1) + (t_0 - t_2)) + \mathbf{D}_2) \tag{3.84}$$

$$\frac{d^3\mathbf{a}}{dt^3} = 3!(\mathbf{D}_4((t_0 - t_1) + (t_0 - t_2) + (t_0 - t_3)) + \mathbf{D}_3) \tag{3.85}$$

$$\frac{d^4\mathbf{a}}{dt^4} = 4!\mathbf{D}_4, \tag{3.86}$$

where the $\mathbf{D}_k$ are evaluated at $[t_0, t_k]$. The accelerations are now expressed in the form

$$\mathbf{a}(t) = a(\Delta t)^4 + b(\Delta t)^3 + c(\Delta t)^2 + d(\Delta t) + e. \tag{3.87}$$

The first time the acceleration is computed, the $\mathbf{D}_4$ terms are left out. Then, once a provisional acceleration at t has been obtained, it is used in a second pass to get a corrected acceleration. Once the positions and velocities have been updated by direct integration of Equation (3.87) for all particles, the actual acceleration on each particle is calculated by direct summation over all particles.

The time step required for any particular particle i depends on its acceleration. Particles accelerating rapidly require a small time step to adequately resolve the motion, whereas particles on nearly straight-line trajectories can be accurately integrated with large time steps. A simple method for estimating the time step for a particle involves the distance D_m and the relative velocity v_m with respect to its nearest neighbor. The travel time then is $\tau_1 \approx D_m/v_m$. The freefall time $\tau_2 \approx D_m^{3/2}$ because $\tau_{ff} \approx 1/\sqrt{G\rho}$ and G is usually set to 1. Thus, we can construct a timescale that is appropriate for each particle, that incorporates both τ_1 and τ_2

$$\Delta t_i = \frac{D_m^{3/2}}{\eta(1 + v_m D_m^{1/2})}, \tag{3.88}$$

where η is a dimensionless number less than 1. In practice, the time step for an individual particle is determined through the more sophisticated relation

$$\Delta t_i = \left[\frac{\eta(|\mathbf{a}||\mathbf{a}^{(2)}| + |\mathbf{a}^{(1)}|^2)}{(|\mathbf{a}^{(1)}||\mathbf{a}^{(3)}| + |\mathbf{a}^{(2)}|^2)}\right]^{1/2} \tag{3.89}$$

where the superscript in parentheses is the order of the derivative (i.e., $\mathbf{a}^{(1)} = d\mathbf{a}/dt$). The reader can verify that the right-hand side of Equation (3.89) indeed has the units of time. In this expression, the enthusiasm of the rate of change of the acceleration for decreasing the time step is judiciously tempered by contributions from lower-order derivatives of $\mathbf{a}$ appearing in the numerator. The time steps are adjusted during the calculation according to Scheme (3.76), so that only a discrete set of time steps is used.

3.8 CLOSE ENCOUNTERS AND REGULARIZATION

Close encounters pose a problem for N-body codes. During a close encounter between two point masses, the quantity $1/r_{ij}^2$ can become arbitrarily large, leading to arbitrarily large accelerations, even for particles with small masses. To follow large accelerations accurately, one requires small time steps, which slows down the algorithm and encourages the buildup of truncation error.

If a close encounter is a one-time event, as would be the case, for instance, with a moderately hyperbolic encounter between two stars, the overall performance of a code will not be severely compromised by an occasional downward spike in time step. Encounters become a continual problem, however, when a tight binary pair forms. Somewhat surprisingly, a chance encounter between three unbound stars can result in one member of the trio being ejected at high speed with the remaining pair stuck in a bound orbit. Once a binary forms, subsequent encounters with other stars in the cluster tend to sap energy from the binary, resulting in a decrease in the semimajor axis and an increase in orbital velocity. This process of "binary hardening" is essential to the overall dynamics of globular clusters, in which a small collection of tight binaries near the core of the cluster is able to release energy and support the core against catastrophic collapse, in much the same way that nuclear reactions between atoms provide the energy input (and attendant pressure) that supports a conventional star.

The simplest and least accurate, but still useful method for protecting an N-body algorithm against the catastrophic time sink provided by hard binaries is to include a softening term within the $\frac{1}{r^2}$ dependence of the gravitational force law. If we write, for example, the force exerted by particle j on particle i as

$$\mathbf{F}_{ij} = \frac{Gm_i m_j(\mathbf{r}_j - \mathbf{r}_i)}{\left(\epsilon^2 + |\mathbf{r}_i - \mathbf{r}_j|^2\right)^{3/2}}, \tag{3.90}$$

then the gravitational acceleration between the two particles saturates at a finite maximum magnitude of

$$|\mathbf{a}_j| = \frac{2Gm_i}{3^{3/2}\epsilon^2}, \tag{3.91}$$

which occurs when the particles reach a separation of $\frac{1}{\sqrt{2}}\epsilon$. By limiting the acceleration, the time step is not allowed to go to arbitrarily low values, but at the cost of not following the orbit of a tight binary star. Therefore, if softening is used, or more precisely, if the softening length ϵ is larger than the actual physical size of the particles that one is modeling, then tight binary orbits and detailed encounters cannot be followed.

When is it safe to use softening? It is safe whenever the dynamics of the system that one is trying to model are not driven by close encounters. That is, the use of softening is appropriate when a system is collisionless. A galaxy, for example, provides an excellent example of a collisionless system. The typical star has a radius 5×10^{10}cm, and the typical separation between stars is of order 5×10^{18}cm. The galaxy contains $\sim 10^{11}$ stars and is approximately 1.5×10^{23} cm in diameter. One can get a better intuitive sense for these numbers by imagining the construction of a scale model. Take a suitcase-sized box full of fine sand (a suitcase can easily hold 10^{11} sand grains) and spread the sand over a disk having a diameter roughly 100 times the width of North

America. At this level of rarefaction, the average separation between the sand grains is several miles. Because the average star in a galaxy must travel for vastly longer than the current age of the universe before running into another star, we say that the distribution is collisionless. Stars feel only the smooth global potential of the entire galaxy and remain unaffected by individual encounters. In the simulation of a galaxy, each particle represents not one star, but rather a bundle of stars on equivalent orbits. Stars on equivalent orbits can easily interpenetrate one another, hence, the use of a softened potential is well motivated.

However, in other problems, such as the evolution of a small cluster of stars, encounters and close binary formation cannot be neglected, and special treatment is needed. The idea underlying regularization is to introduce a coordinate transformation, which replaces the ordinary physical time with a regularized time. By effectively replacing the nonuniform oscillation of an unperturbed elliptical orbit by simple harmonic motion, the transformation allows the integration to be carried through the expensive moment of close approach between the two bodies with economy and accuracy. Regularization will even resolve the direct collision between two idealized point mass particles (i.e., two-body motion with zero angular momentum), although, in practice, the planets or stars in a real astrophysical simulation have finite radii, and the results of a true collision are very different from the mathematical trajectory of point masses.

The acceleration of the separation vector $\mathbf{R}$ for the components of a particular pair of particles within a larger N-body simulation is given by

$$\frac{d^2\mathbf{R}}{dt^2} = -G(m_1 + m_2)\frac{\mathbf{R}}{|\mathbf{R}|^3} + \mathbf{F}_{12} \tag{3.92}$$

where $\mathbf{F}_{12} = \mathbf{F}_1 - \mathbf{F}_2$ is the net external force per unit mass arising from the other bodies in the overall system. The regularization of Equation (3.92) involves transformation of both the time and the space coordinates. As the first step, we proceed by introducing a transformed, regularized time, τ, that is related to the ordinary time, t, by

$$dt = R^n d\tau \tag{3.93}$$

so that

$$\frac{d^2}{dt^2} = \frac{1}{R^{2n}}\frac{d^2}{d\tau^2} - \frac{n}{R^{2n+1}}\frac{dR}{d\tau}\frac{d}{d\tau}. \tag{3.94}$$

The equation of motion (Equation 3.92) becomes

$$\frac{d^2\mathbf{R}}{d\tau^2} = \frac{n}{R}\frac{dR}{d\tau}\frac{d\mathbf{R}}{d\tau} - G(m_1 + m_2)\frac{\mathbf{R}}{R^{3-2n}} + R^{2n}\mathbf{F}_{12} \tag{3.95}$$

For $n = 1$, which is the most practically useful choice, the separation between the particles is directly proportional to the rate at which real time passes in comparison to regularized time. In this case, by regularizing the time, we have removed the R^{-2} singularity in Equation (3.92) while introducing a term $\mathbf{R}/R$ that is indeterminate as the separation between the particles goes to zero. Thus, a second transformation is required to make a successful two-body regularization. This second ingredient involves

transforming the space coordinates to get rid of the indeterminacy. This transformation gets successively tougher as one goes from one to two to three dimensions. Therefore, we look at each case in turn. (For details concerning the steps that we skip, see Aarseth (2003).)

The one-dimensional spatial transformation was first introduced by Euler. In one dimension, there is no distinguishing between $\mathbf{R}$ and R. In the absence of an external force, Equation (3.95) becomes

$$\frac{d^2R}{d\tau^2} = \frac{1}{R}\left(\frac{dR}{d\tau}\right)^2 - G(m_1 + m_2). \tag{3.96}$$

This equation still runs into difficulties in numerical integrations. However, we can use energy conservation to improve things. The binding energy per reduced mass $\mu = m_1 m_2/(m_1 + m_2)$, is:

$$h = \frac{1}{2}\left(\frac{dR}{dt}\right)^2 - \frac{G}{R}(m_1 + m_2) \tag{3.97}$$

and is fixed for any given unperturbed binary orbit. Using the fact that

$$\frac{dR}{dt} = \frac{1}{R}\frac{dR}{d\tau}, \tag{3.98}$$

we have

$$\frac{d^2R}{d\tau^2} = 2hR + G(m_1 + m_2), \tag{3.99}$$

which is free of any problems as R goes through zero. If we further write $u^2 = R$, then the equation of motion reduces to a simple harmonic oscillator

$$\frac{d^2u}{d\tau^2} = \frac{1}{2}hu. \tag{3.100}$$

Equation (3.100) is readily integrated using the techniques discussed earlier in this chapter.

In numerical practice, then, one could handle the development of an effectively straight-line collision between two particles in a simulation by switching from (x, t) to (u, τ) when the bodies reach a prespecified minimum distance, using Equation (3.100) to resolve the collision and then switching back after the bodies are safely separated from one another.

A workable coordinate transformation for handling close binary encounters in two dimensions was originally described by Levi-Civita (1904). As in one dimension, the problem consists of eliminating the indeterminate term $\mathbf{R}/R$ from Equation (3.95). The regularization proceeds by first rewriting the components of $\mathbf{R} = (x, y)$ in terms of new variables u_1 and u_2

$$x = u_1^2 - u_2^2 \tag{3.101}$$

$$y = 2u_1u_2, \tag{3.102}$$

so that in matrix notation

$$\mathbf{R} = \mathcal{L}\mathbf{u}, \tag{3.103}$$

where

$$\mathcal{L}(\mathbf{u}) = \begin{bmatrix} u_1 & -u_2 \\ u_2 & u_1 \end{bmatrix}. \tag{3.104}$$

The transformation has the following mathematical properties (for arbitrary $\mathbf{u}$, $\mathbf{v}$):

1. $\mathcal{L}^T(\mathbf{u})\mathcal{L}(\mathbf{u}) = RI$, where I is the unit matrix

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.105}$$

2. $\frac{d}{d\tau}\mathcal{L}(\mathbf{u}) = \mathcal{L}\left(\frac{d\mathbf{u}}{d\tau}\right)$
3. $\mathcal{L}(\mathbf{u})\mathbf{v} = \mathcal{L}(\mathbf{v})\mathbf{u}$
4. $\mathbf{u}\cdot\mathbf{u}\mathcal{L}(\mathbf{v})\mathbf{v} - 2\mathbf{u}\cdot\mathbf{v}\mathcal{L}(\mathbf{u})\mathbf{v} + \mathbf{v}\cdot\mathbf{v}\mathcal{L}(\mathbf{u})\mathbf{u} = 0$

With these transformations, the equation of motion (Equation 3.95) can be rewritten in terms of the new variable $\mathbf{u}$, using properties two and three of Equation (3.105)

$$\begin{aligned} \frac{d\mathbf{R}}{d\tau} &= 2\mathcal{L}(\mathbf{u})\frac{d\mathbf{u}}{d\tau} \\ \frac{d^2\mathbf{R}}{d\tau^2} &= 2\mathcal{L}(\mathbf{u})\frac{d^2\mathbf{u}}{d\tau^2} + 2\mathcal{L}\left(\frac{d\mathbf{u}}{d\tau}\right)\frac{d\mathbf{u}}{d\tau}. \end{aligned} \tag{3.106}$$

When these expressions are substituted into the equation of motion, using property four and $n = 1$, plus some algebra, then

$$2\mathbf{u}\cdot\mathbf{u}\mathcal{L}(\mathbf{u})\frac{d^2\mathbf{u}}{d\tau^2} - 2\frac{d\mathbf{u}}{d\tau}\cdot\frac{d\mathbf{u}}{d\tau}\mathcal{L}(\mathbf{u})\mathbf{u} + G(m_1+m_2)\mathcal{L}(\mathbf{u})\mathbf{u} = (\mathbf{u}\cdot\mathbf{u})^3\mathbf{F}_{12}. \tag{3.107}$$

After still more manipulation, one obtains the equation in a form that contains neither singularities nor indeterminacies as R goes to 0

$$\frac{d^2\mathbf{u}}{d\tau^2} = \frac{1}{2}h\mathbf{u} + \frac{1}{2}R\mathcal{L}^{\mathrm{T}}(\mathbf{u})\mathbf{F}_{12}. \tag{3.108}$$

As was the case in 1-D, if one has only an isolated binary with no external perturbation, then h is exactly conserved, and the equation of motion reduces once again to the readily solvable simple harmonic oscillator. For the general case, however, the perturbation $\mathbf{F}_{12}$ leads to a nonconservation of binding energy and h must be expressed as

$$h = \left[2\frac{d\mathbf{u}}{d\tau}\cdot\frac{d\mathbf{u}}{d\tau} - G(m_1+m_2)\right]\Big/R. \tag{3.109}$$

Note that in ordinary coordinates

$$\frac{d}{dt}\left[\frac{1}{2}\left(\frac{dR}{dt}\right)^2 - \frac{G}{R}(m_1+m_2)\right] = \frac{d\mathbf{R}}{dt}\cdot\mathbf{F}_{12}, \tag{3.110}$$

which can be used, along with the coordinate transformations, to write

$$\frac{dh}{d\tau} = 2\frac{d\mathbf{u}}{d\tau}\cdot\mathcal{L}^T(\mathbf{u})\mathbf{F}_{12}. \tag{3.111}$$

Equation (3.108) and Equation (3.111) can be smoothly integrated as ordinary differential equations through $R = 0$.

An interesting application of the Levi-Civita transformation in a numerical N-body simulation was given by Szebehely and Peters (1967) in their solution of the long-standing Pythagorean three-body problem. In this problem, which idealizes the situation that can occur during an encounter between a binary star and a single star in an evolving star cluster, three point masses of masses $m_1 = 3$, $m_2 = 4$, and $m_3 = 5$ are placed at the vertices opposite the respective sides of a 3-4-5 right triangle. The three particles are initially at rest and the problem is to completely describe their subsequent motion.

As is the case for many bound encounters between a single star and a binary, the motion is extremely complex. The zero-velocity initial condition ensures that the total angular momentum $\mathbf{L} = 0$. This means that in theory, a triple collision is possible, in which case a solution for all time would not exist. Numerical integration using regularization (Figure 3.7), however, shows that no triple collision takes place. During the

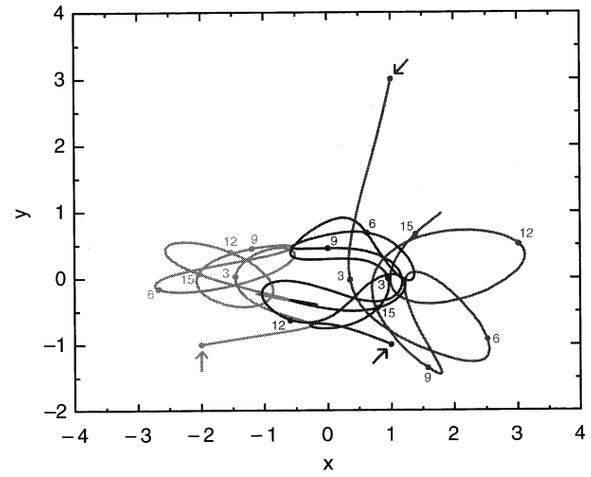

FIGURE 3.7 ***A color version of this figure follows page 212*** Integration of the three-body Pythagorean problem. The initial positions of the particles (indicated by arrows) are at the vertices of a right triangle. The particles have masses of 3, 4, and 5 grams. Positions in the (x, y) plane are expressed in cm. In these units, the unit of time is 3872 s. The equations are integrated for 16 time units, and the positions at selected times are marked on the curves. An initial close encounter between mass 4 and mass 5 occurs at about $x = -0.25$, $y = -0.75$. The end result of the simulation (not shown), which is determined at about $t = 60$, is the formation of a binary by masses 4 and 5 and the ejection of mass 3 from the system. (After Szebehely and Peters (1967). Figure courtesy of Evan Kirby.)

course of the calculation, the distances r_{ij} between the particles are monitored. When two bodies come closer than a threshold value r_{min}, the Levi-Civita transformation is applied to regularize the motion, with the third body playing the part of the perturber, which drives the harmonic oscillator equation. The two bodies are integrated through the close encounter using the regularized equation and are swapped back into Cartesian coordinates once $r_{ij} > r_{min}$. Despite the overhead associated with switching back and forth, the calculation is completed much more quickly and with higher accuracy than if regularization is not used.

The Levi-Civita procedure for regularizing two-dimensional motion transfers in a complicated way to three dimensions, the details of which are not treated here. The equation of motion (Equation 3.92) for a binary pair in three dimensions must first be transformed to a *four-dimensional* coordinate system before it can be regularized. In the context of the numerical N-body problem, this was first done by Kustaanheimo and Stiefel (1965), who gave the transformation

$$\begin{aligned} R_1 &= u_1{}^2 - u_2{}^2 - u_3{}^2 + u_4{}^2 \\ R_2 &= 2(u_1u_2 - u_3u_4) \\ R_3 &= 2(u_1u_3 + u_2u_4) \\ R_4 &= 0, \end{aligned} \tag{3.112}$$

so that if

$$\mathbf{R} = \mathcal{L}(\mathbf{u})\mathbf{u}, \tag{3.113}$$

one has

$$\mathcal{L}(\mathbf{u}) = \begin{bmatrix} u_1 & -u_2 & -u_3 & u_4 \\ u_2 & u_1 & -u_4 & -u_3 \\ u_3 & u_4 & u_1 & u_2 \\ u_4 & -u_3 & u_2 & -u_1 \end{bmatrix}. \tag{3.114}$$

The analysis outlined by Aarseth (2003; Section 4.4) shows, however, that the equations of motion reduce to exactly the same form as for the two-dimensional case, namely Equation (3.108) and Equation (3.111). Thus, the solution reduces to one that resembles a forced harmonic oscillator, and there is no problem as $R \to 0$.

3.9 FORCE CALCULATION: THE TREE METHOD

We first consider strategies for reducing the $\sim \mathcal{O}(N^2)$ dependence of computational time on the number of particles, which would result if the right-hand side of Equation (3.1) were simply summed over all particles. In any aggregate of gravitating bodies, the particles that are closest to the body being considered will exert the largest contributions to the instantaneous gravitational acceleration, while distant particles will have relatively little individual effect. Therefore, one can drastically speed up an N-body code by retaining careful consideration of nearby neighbors, while treating the more numerous distant bodies as simpler aggregates. This is the basic idea behind the hierarchical tree method.

The tree algorithm (as implemented, for example, by Barnes and Hut 1986, 1989) works by tessellating the volume of space that contains particles into a hierarchy of

nested cubic cells, or nodes, all of which contain at least one particle. The aggregate of cells is called the tree, and the finest subdivisions, those containing only a single particle, are the leaves.

Construction of the tree proceeds recursively from the top down. A large cube is first drawn that encompasses all of the particles in the simulation. This lowest-order node is then split into eight equal subvolumes. Those containing no particles are discarded, while those containing multiple particles are subject to further subdivision. As the tree is constructed, the total mass, the center-of-mass location, and the quadrupole moment of each node is computed and stored.

The quadrupole moment is derived from the $n = 2$ term in the multipole expansion of the gravitational potential arising from a distribution of mass

$$\Phi(\mathbf{R}) = -G\sum_{n=0}^{\infty} \frac{1}{|\mathbf{R}|^{n+1}} \int |\mathbf{r}|^n P_n(\cos\theta)\rho(\mathbf{r})d^3\mathbf{r}, \tag{3.115}$$

where the geometry used in the expansion is shown in Figure 3.8. In Equation (3.115), $P_n(x)$ is the nth Legendre Polynomial, and θ is the angle between the radius vectors of a mass element located at position $\mathbf{r} = (x_1, x_2, x_3)$ (inside the node) and the location $\mathbf{R} = (X_1, X_2, X_3)$ at which the potential is to be evaluated (outside the node). The $n = 0$ (the monopole) term corresponds to the potential $(-GM/R)$ that would occur

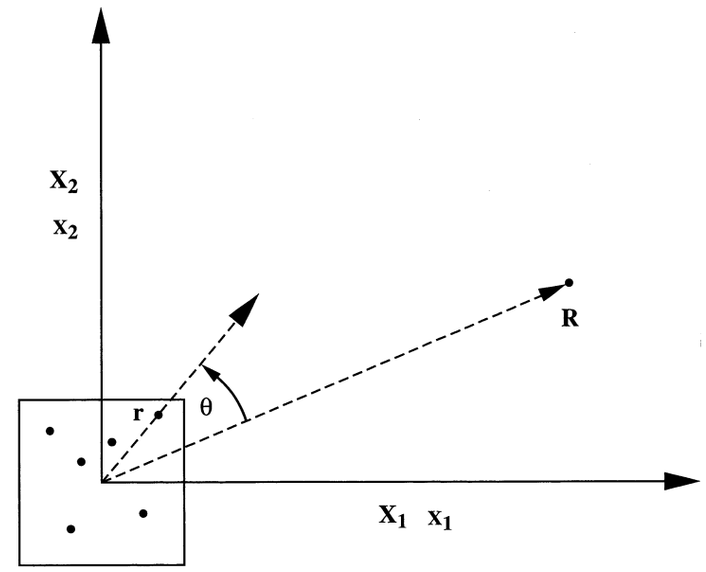

FIGURE 3.8 Geometry for the Legendre expansion in Equation (3.115), simplified to two dimensions. A node of the tree is indicated by the box. A particle i in the box has a radius vector $\mathbf{r}_i$. As long as the box is sufficiently far away from an external particle at $\mathbf{R}$, the contribution of the node to the potential at $\mathbf{R}$ is obtained from the sum of the monopole and quadrupole terms of the Legendre expansion of the potential from the particles in the box. The origin of the coordinate system is the center of mass of the box.

if all of the mass (M) in the node under consideration were concentrated at the center of mass. Mass cannot be negative, hence, the $n = 1$ (the dipole term) vanishes by symmetry. The $n = 2$ term gives the quadrupole contribution to the potential. It can be rewritten in the form

$$\Phi(\mathbf{R})_2 = -\frac{G}{2|\mathbf{R}|^3}\sum_{i,j=1}^{3}\frac{X_i X_j}{R^2}Q_{ij}\,, \tag{3.116}$$

where

$$Q_{ij} = \int \rho(\mathbf{r})\left(3x_i x_j - r^2\delta_{ij}\right)d^3\mathbf{r} \tag{3.117}$$

is the quadrupole moment tensor. For a collection of point particles within a node of the tree, the integral in Equation (3.117) reduces to a sum.

After the tree has been constructed, which is redone for every time step, the gravitational force on each particle is evaluated. These forces are computed by working systematically down through the tree. At each level, the width, w, of a node is compared to the distance, R, between the node's center of mass and the location of the current particle. If $w/R < \delta$, where δ is a specified fractional distance, then the attraction between all the particles in the node and the current particle is evaluated using the monopole and quadrupole contributions that have been stored for the node, and no further descent into the node's subdivisions is necessary. Alternately, if $w/R > \delta$, the node must be further subdivided. As this procedure is repeatedly applied, all of the nodes in the tree are either added to a monopole–quadrupole approximate contribution or are found to contain a single particle. For nodes containing single particles, the contribution to the gravitational force is obtained through direct summation.

With the switching criterion between direct summation and multipole expansion set to the form $w/R < \delta$, the sizes of cells contributing approximate monopole–quadrupole-based forces increases in direct proportion to the distance from the current particle, and the sum over N particles is replaced by a sum containing only $\sim \log N$ terms. For simulations in which δ is a small fraction of the total physical domain, the computational time required to obtain all the accelerations scales as $\mathcal{O}(N \log N)$. For large N, the difference between N^2 and $N \log N$ is enormous.

For a large aggregate of particles, essentially nothing is lost by approximating the direct interactions with the first terms of the gravitational multipole expansion. Any benefit from finely resolving distant aggregates of mass is often outweighed by the attendant accumulation of truncation errors and the errors arising from the use of finitely resolved time steps. Indeed, in the few-body techniques considered in previous sections, the planets, which are physically extended objects, were reduced to point masses, which amounts to truncating their multipole expansions at a single term. For a refined investigation of the overall dynamics of the solar system, one might, for example, approximate the Earth–Moon binary by both the total point mass contribution as well as the quadrupole contribution.

For one million particles, the factor of increase in speed provided by shifting from direct summation to the tree method can approach 70,000. One might ask, is there a way to further speed up the computation of the gravitational forces so that finding the accelerations becomes an $\mathcal{O} \sim (N)$ process?

Remarkably, an order N method was employed by Erik Holmberg of the Lund Observatory in Sweden in 1941. Instead of integrating the equations of motion with a computer, Holmberg modeled a two-dimensional system of gravitating particles as an actual physical distribution of movable light bulbs laid out on a gridded sheet of dark paper. Because the intensity of light from a point source diminishes as $1/r^2$, one can directly relate the intensity of the light at a particular spot to the gravitational acceleration. The N^2 process of computing the gravitational force on a given particle from all of the other particles reduces to a measurement of the total intensity of light in two perpendicular directions using a photocell and a galvanometer. Since one set of measurements is required for the location of each light bulb, the method scales as $\mathcal{O}(N)$.

With his intensity measurement analog method for computing the net gravitational acceleration on each of his light bulb "point masses," Holmberg could compute the change in trajectories that would occur over a time interval using a simple integration scheme such as Euler's method. A time step would then be completed by moving all of the light bulbs to their updated positions, at which point a new estimate of the gravitational acceleration could be made. Holmberg's scheme allowed him to gain a better understanding of important aspects of the dynamics of close encounters between disk galaxies, including the phenomena of orbital decay and the formation of tidal tails. Viewed from today, experiments such as this one seem quaint, but there is an important lesson to be drawn: Use of an analog method reduces an $\mathcal{O}(N^2)$ computation to $\mathcal{O}(N)$, foreshadowing a time when quantum computation will similarly reduce the computational time required from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$.

3.10 FORCE CALCULATION: FAST FOURIER TRANSFORMS

In its most familiar guise, the technique of Fourier analysis allows one to express a continuous time signal in terms of an infinite set of frequency components. That is, the distribution of frequencies, $H(\omega)$, corresponding to a time series $h(t)$ is given by

$$H(\omega) = \int_{-\infty}^{\infty} h(t)\exp(i\omega t)dt. \tag{3.118}$$

Even after learning to manipulate complex variables, some readers feel uncomfortable with the fact that the time series $h(t)$ is real, whereas the Fourier transform, $H(\omega)$, is a complex number: "Isn't a frequency a number of Hertz — a real quantity?"

$H(\omega)$ is complex because it has to express two pieces of information for every frequency; at each ω, we can write $H(\omega) = H_0(\omega)\exp[i\theta_0(\omega)]$. $H_0(\omega)$, which is purely real, gives the amplitude of the sinusoidal contribution for ω, whereas $\theta_0(\omega)$ gives its phase. The phase information allows us to correctly position the sinusoid relative to all the others so that when we make an inverse Fourier transform, the original (purely real) time-dependent signal

$$h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(\omega)\exp(-i\omega t)dt \tag{3.119}$$

is recovered.

The utility of Fourier transforms in the context of the N-body problem stems from a remarkable result known as the convolution theorem. If $g(t)$ and $h(t)$ are functions, then their convolution is:

$$g \star h = \int_{-\infty}^{\infty} g(\tau)h(t-\tau)d\tau, \tag{3.120}$$

and the Fourier transform of the convolution is $G(\omega)H(\omega)$, where G and H are the Fourier transforms of g and h, respectively.

The transform of the convolution of two functions, thus, is equal to the product of their transforms, and if these transforms are known, we can avoid explicitly computing the integral. To motivate this, note that the gravitational potential of a continuous field of sources

$$\Phi(\mathbf{x}) = -G \int \frac{\rho(\mathbf{x}')}{|\mathbf{x}-\mathbf{x}'|} d^3\mathbf{x}' \tag{3.121}$$

has the basic structure of the Expression (3.120), with $\mathbf{x}'$ playing the role of τ. This connection allows us to use Fourier transforms to do potential calculations.

The first step in making a simple Fourier transform potential solver is to define a three-dimensional Cartesian grid that encompasses all the particles. One then loops through the N particles and computes an index

$$ix = \mathrm{A}_{\mathrm{int}}\left[\frac{(x_i - x_{min})}{(x_{max} - x_{min})} Nc_x\right] \tag{3.122}$$

for each particle, along with analogous indices iy and iz. In the above formula, x_{max} and x_{min} are the predefined x-direction limits of the grid, and Nc_x is the number of cells assigned to the x-direction. The $\mathrm{A}_{\mathrm{int}}$ function indicates truncation to an integer value (e.g., $\mathrm{A}_{\mathrm{int}}[2.647] = 2$). Within the loop, as the cell indices of each particle are computed, the mass of each cell, $M_{ix,iy,iz}$, can be tallied, along with the three coordinate arrays, x_{ix}, y_{iy}, z_{iz}, for the cell centers. For example,

$$x_{ix} = x_{min} + \frac{(ix - 0.5)}{Nc_x}(x_{max} - x_{min}). \tag{3.123}$$

The softened gravitational potentials at the cell centers are given by

$$\Phi_{jx,jy,jz} = -\sum_{ix=0}^{Nc_x-1}\sum_{iy=0}^{Nc_y-1}\sum_{iz=0}^{Nc_z-1} M_{ix,iy,iz}\frac{G}{D},$$

where

$$D = \sqrt{\epsilon^2 + |x_{jx} - x_{ix}|^2 + |x_{jy} - x_{iy}|^2 + |x_{jz} - x_{iz}|^2}. \tag{3.124}$$

With this discretization, we have replaced the integral in Equation (3.121) with a sum over ix, iy, and iz. The Fourier transform and inverse transforms (Equation 3.118) and (Equation 3.119) can be similarly discretized through the use of multidimensional Fourier transforms, which map one set of $N = N_1N_2N_3$ numbers arranged in a three-dimensional grid $h(k_1, k_2, k_3)$ to a second set of N numbers in an identically arrayed

grid $H(n_1, n_2, n_3)$

$$H(n_1, n_2, n_3) = \sum_{k_3=0}^{N_3-1}\sum_{k_2=0}^{N_2-1}\sum_{k_1=0}^{N_1-1} h(k_1, k_2, k_3)\exp(E_1)\exp(E_2)\exp(E_3) \tag{3.125}$$

with $E_1 = 2\pi i k_1 n_1/N_1$, $E_2 = 2\pi i k_2 n_2/N_2$, and $E_3 = 2\pi i k_3 n_3/N_3$. Likewise, the discrete three-dimensional inverse Fourier transform is given by

$$h(k_1, k_2, k_3) = \frac{1}{N}\sum_{n_3=0}^{N_3-1}\sum_{n_2=0}^{N_2-1}\sum_{n_1=0}^{N_1-1} H(n_1, n_2, n_3)\exp(-E_1)\exp(-E_2)\exp(-E_3). \tag{3.126}$$

Note that multidimensional Fourier transforms are computed by taking successive one-dimensional transforms, so that the overall effort involves a comparable number of operations to taking a one-dimensional transform of an equivalently long ($N_{\mathrm{tot}} = N_1N_2N_3$) sequence of numbers. That is, we can make a minor alteration in the way that Equation (3.125) is expressed

$$H(n_1, n_2, n_3) = \sum_{k_1=0}^{N_1-1}\left[\sum_{k_2=0}^{N_2-1}\left[\sum_{k_3=0}^{N_3-1} h(k_1, k_2, k_3)\exp(E_3)\right]\exp(E_2)\right]\exp(E_1), \tag{3.127}$$

which makes it clear that we can compute the one-dimensional transforms from the inner nesting to the outer nesting, thus giving us the capability of computing the Fourier transform of a set of numbers defined on a grid.

In order to actually find the potentials on the grid that has been superimposed on the mass distribution, we leverage the fact that the Fourier convolution theorem has a discrete analogy. That is, if

$$Z_k = \frac{1}{\sqrt{2K}}\sum_{k'=-K}^{K-1} Y_{k-k'}X_{k'} \tag{3.128}$$

defines a convolution, then the Fourier transform $\hat{Z}_p$ is obtained from

$$\hat{Z}_p = \hat{X}_p\hat{Y}_p. \tag{3.129}$$

(For proof see Binney and Tremaine [1987, Section 2.8].) From the form of Equation (3.118), it is clear that a Fourier integral transform applies over an unbounded domain of time (or of space, when we identify t with a position coordinate). A localized function that goes to zero beyond a certain boundary can still be represented by periodic functions because an infinite number of frequency components are being called upon to provide perfectly destructive interference within the entire infinite region beyond the realm where $h(t)$ is nonzero. With a finite and discretized distribution, however, we have no such luxury. The "frequency components" $\hat{X}_p$ constitute a finite set and lead to a representation X_k from the inverse transformation that must be understood to be periodic.

The equation for the potential (Equation 3.124) has essentially the form required to implement the discrete Fourier transform. If we scale the size of the cells so that each cell is cubic with sides of unit length, then we have

$$\Phi_{ijk} = -\sum_{i'=0}^{K-1}\sum_{j'=0}^{K-1}\sum_{k'=0}^{K-1} X(i-i', j-j', k-k')M_{i',j',k'}, \tag{3.130}$$

where $X = G/D$. The range of the indices that are being summed over in Equation (3.130) is half of the range of the indices in the sums in the expression for the discrete Fourier convolution. In order to apply the Fourier convolution theorem to compute the potential, we need to augment the sums in Equation (3.130) so that they run over the full range of the indices. This is done by defining the mass cube $M_{i',j',k'}$ to be twice as large in every direction, but with all of the new cells occupied with zero particles. The geometric array $X(i-i', j-j', k-k')$ is also expanded so that the indices run from $-K$ to $K-1$. (All of these geometric terms can be precomputed and then reused every time the mass distribution changes.)

Given this simple modification, the Fourier convolution theorem can be applied to yield an estimate of the potential at each point in the grid. One obtains the three-dimensional Fourier transforms, $\hat{X}$, and $\hat{M}$ of the matrices X and M, and multiplies each individual term, $\hat{X}_{ijk}$ with its counterpart $\hat{M}_{ijk}$. The resulting matrix is the Fourier transform, $\hat{\Phi}$, of the potential. The potentials Φ themselves are then obtained by computing the discrete inverse Fourier transform.

With what has been discussed so far, the only advantage in computational speed from the Fourier method arises from the fact that X, with its profusion of square roots, can be precomputed. The Fourier transforms, if done in the naïve way by working term-by-term through the triple sums, are an order N^2 operation and, as such, add no benefit beyond a simple, direct summation over all grid cells to obtain the potential. The advantage of the Fourier transform comes from the fact that the discrete Fourier transform can be computed as an $N \log N$ operation, which, for large N, offers a gargantuan improvement in speed.

The order $N \log N$ *fast Fourier transform* (FFT) exploits the fact that the Fourier transform of N discrete points, whose indices run from 0 to $N-1$ can be rewritten as a sum of two transforms of half the length, whose indices run from 0 to $N/2-1$. That is, assume that the transform length N is an integer power of two, $N = 2^m$. The k^{th} component of the (1-D) transform

$$H_k = \sum_{j=0}^{N-1} h_j \exp(2\pi i jk/N) \tag{3.131}$$

can be expressed as the addition of a series of even terms with length $N/2$ to a series of odd terms of length $N/2$

$$H_k = \sum_{j=0}^{N/2-1} h_{2j} \exp[2\pi ik(2j)/N] + \sum_{j=0}^{N/2-1} h_{2j+1} \exp[2\pi ik(2j+1)/N], \tag{3.132}$$

which, with a slight rearrangement, are seen themselves to be two length $N/2$ individual Fourier series, with the second multiplied by the complex factor $\exp(2\pi ik/N)$

$$H_k = H_{ek} + \exp(2\pi ik/N)H_{ok} = \sum_{j=0}^{N/2-1} h_{2j} \exp[2\pi ikj/(N/2)] + \exp(2\pi ik/N) \sum_{j=0}^{N/2-1} h_{2j+1} \exp[2\pi ikj/(N/2)]. \tag{3.133}$$

In order to evaluate the components of H_k that have $k > N/2$, we operationally require the sums in Equation (3.133) to run from 0 to $N-1$. This demand is met by recognizing that the sums are periodic. Thus, the full 0 to $N-1$ range of k can be produced by augmenting each series on the right-hand side of Equation (3.132) by a repeated copy of itself for $k > N/2$.

The FFT draws its speed from the recursive applications of Equation (3.133). Using Equation (3.132), the even series, H_{ek} can be further subdivided to form

$$H_{ek} = H_{eek} + \exp[2\pi ik/(N/2)]H_{eok} \tag{3.134}$$

and the odd series becomes

$$\exp(2\pi ik/N)H_{ok} = \exp(2\pi ik/N)[H_{oek} + \exp[2\pi ik/(N/2)]H_{ook}]\ . \tag{3.135}$$

One proceeds through m subdivisions until one is left with N *one-point* Fourier transforms, each multiplied by the individual complex factor, $\exp(i\phi_k)$, built up through the m subdivisions.

Remarkably, the N one-point Fourier transforms (call them H_{1k}) that remain after m recursive applications of Equation (3.134) and Equation (3.135) have an exact (yet scrambled) one-to-one correspondence with the N terms of the original input function h. The mapping $k \to l$, which equates each term h_k with a one-point transform H_{1l} proceeds by bit-reversing the indices k (see, Press et al, 1992). That is, one first writes the integer k as a binary number with m digits, and then determines the integer l by reading off the binary digits of k from right to left, as opposed to the usual left to right. For example, for $N = 32$, $k = 19$ has binary representation 10011, indicating that the binary representation for l is 11001, i.e., $l = 25$.

To compute the transform, one first loops through the indices k to get the bit reversed values l and the corresponding array of one-point transforms $H_{1k} = h_l$. One then uses Equation (3.134) and Equation (3.135) to combine each pair of one-point transforms to form two-point transforms. Both terms in each transform are computed. The sets of two-point transforms are then combined to form four-point transforms, and the process continues a total of $m = \log_2 N$ times to form the final FFT. At each level, forming the combinations requires order N operations, making the entire procedure order $N \log N$.

As a concrete example, consider a four-point FFT of a discrete signal that has been sampled four times

$$h = [h_0, h_1, h_2, h_3]. \tag{3.136}$$

The bit reversal step yields $00 \rightarrow 00$, $01 \rightarrow 10$, $10 \rightarrow 01$, and $11 \rightarrow 11$, so that the one-point transforms are given by, $H_{10} = h_0$, $H_{11} = h_2$, $H_{12} = h_1$, and $H_{13} = h_3$. The rule (Equation 3.134) is used to combine H_{10} and H_{11} into a two-point transform

$$[H_{20} = h_0 + h_2,\ H_{21} = h_0 + h_2 \exp(i\pi)], \tag{3.137}$$

and to combine H_{12} and H_{13} into a second two-point transform with terms $[h_1 + h_3, h_1 + h_3 \exp(i\pi)]$.

Finally, with $N = 4$, the components of the four-point FFT can be assembled from the two-point FFT

$$\begin{aligned} H_{40} &= h_0 + h_2 + h_1 + h_3 \\ H_{41} &= h_0 + h_2 \exp(i\pi) + \exp(i\pi/2)[h_1 + h_3 \exp(i\pi)] \\ H_{42} &= h_0 + h_2 + \exp(i\pi)[h_1 + h_3] \\ H_{43} &= h_0 + h_2 \exp(i\pi) + \exp(3\pi i/2)[h_1 + h_3 \exp(i\pi)]. \end{aligned} \tag{3.138}$$

Notice that each transformed component, H_{4k}, is itself formed from the sum of four terms, suggesting at first glance that order $N^2 = 16$ operations are required to compute the transform. Upon inspection, however, one sees that the terms are not fully independent, having been assembled from precomputed sums of length $N/2$. For a four-point FFT, the overhead involved in computing the terms of form $\exp(i\theta)$ effectively cancels the gain in efficiency from the FFT. As N increases, however, the $N \log N$ scaling of the FFT algorithm imparts an increasingly massive advantage.

Exercise

Verify Equation (3.138) by calculating its inverse discrete Fourier transform.

Once the potential has been computed by employing the FFT and the discrete convolution theorem, the individual potentials can be differenced across the grid cells to obtain the acceleration components a_x, a_y, and a_z.

We may summarize the situation for a large N system, where it is not practical to calculate the gravitational forces by direct summation over all particles. The Fourier method, also known as the *particle-mesh* (PM) method, may be better in a situation where the mass is distributed, more or less, uniformly. A grid is overlaid on the system of particles and a number of grid-related quantities can be precomputed once and then used over and over again. An example would be the case of very large-scale cosmological simulations. However, if the mass distribution becomes clumpy, with a lot of mass concentrated into a few grid cells with large almost empty spaces in between, then the Fourier method will not give an accurate potential in the dense regions unless the grid is adaptively refined. In this case, the tree method would be better, since it automatically puts the leaves (smallest subdivisions) where the mass is and, thus, is adaptive in a fully Lagrangian sense. An example of such a problem would be the encounter and collision between two galaxies. Another alternative is the so-called "particle-particle-particle-mesh" (P^3M) scheme (Hockney and Eastwood, 1981, Efstathiou and Eastwood, 1981) in which the force acting on a given particle is split into two parts, one involving long-range smoothly varying forces, which are calculated with the Fourier method on a grid, and the other involving short-range forces arising only from a small set of particles within a specified distance from the given particle. The short-range forces are calculated by direct summation. This type of code is widely used in cosmological simulations. Nevertheless a (gridless) tree code may still have an advantage, since any grid imposes a particular geometry on the problem and can introduce errors as a result of force interpolation.

REFERENCES

Aarseth, S. (1985) *Dynamics of Star Clusters* (Dordrecht: Reidel), p. 251.

Aarseth, S. (2003) *Gravitational N-Body Simulations* (Cambridge: Cambridge University Press).

Ahmad, A. and Cohen, L. (1973) *Astrophys. J.* **179**: 885.

Barnes, J. and Hut, P. (1986) *Nature* **324**: 446.

Barnes, J. and Hut, P. (1989) *Astrophys. J. Suppl.* **70**: 389.

Binney, J. and Tremaine, S. (1987) *Galactic Dynamics* (Princeton: Princeton University Press).

Danby, J. (1988) *Fundamentals of Celestial Mechanics,* 2nd ed. (Richmond, VA: Willmann-Bell).

Efstathiou, G. and Eastwood, J. W. (1981) *Monthly Notices of the Royal Astronomical Society* **194**: 503.

Gleick, J. (1987) *Chaos: Making a New Science* (New York: Viking Penguin).

Goldstein, H. (1980) *Classical Mechanics* (Reading, MA: Addison-Wesley).

Hockney, R. W. and Eastwood, J. W. (1981) *Computer Simulation Using Particles* (New York: McGraw-Hill).

Holmberg, E. (1941) *Astrophys. J.* **94**: 385.

Kustaanheimo, P. and Stiefel, E. (1965) *J. Reine Angew. Math.* **218**: 204.

Lecar, M., Franklin, F., and Murison, M. (1992) *Astron. J.* **104**: 1230.

Levi-Civita, T. (1904) *Ann. Mat. Ser.* 3, **9**: 1.

Marcy, G. W., Cochran, W. D. and Mayor, M. (2000) *Protostars and Planets IV,* V. Mannings, A. P. Boss and S. S. Russell, Eds. (Tucson: University of Arizona Press), p. 1285.

Montenbruck, O. (1989) *Practical Ephemeris Calculations* (Heidelberg: Springer-Verlag).

Murray, C. and Dermott, S. (1999) *Solar System Dynamics* (Cambridge: Cambridge University Press).

Newton, I. (1687) *Philosophiae Naturalis Principia Mathematica* (London: Royal Society).

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1992) *Numerical Recipes in Fortran: The Art of Scientific Computing,* 2nd ed. (Cambridge: Cambridge University Press).

Saha, P. and Tremaine, S. (1994) *Astron. J.* **108**: 1962.

Stoer, J. and Bulirsch, R. (1980) *Introduction to Numerical Analysis* (New York: Springer–Verlag).

Szebehely, V. and Peters, C. F. (1967) *Astron. J.* **72**: 876.

Wisdom, J. and Holman, M. (1991) *Astron. J.* **102**: 1528.