2. Precipitation is placed on a random site of the (square) lattice.
3. Water flows from this site to a nearest neighbor site with a probability proportional to $e^{E\Delta h}$, where Δh is the difference in height between the site and a neighbor, and E is a parameter. If $\Delta h < 0$, then the probability is equal to zero, and the flow will not return to the site previously visited if there is a nonzero probability of flowing to another site.
4. Step 3 is repeated until the water flows to the bottom of the lattice, $y = 0$.
5. Each lattice point that has been visited by the flowing water has its height reduced by a constant amount D. This process represents erosion.
6. Any site at which the height difference Δh with a neighbor exceeds a critical amount M is reduced in height by an amount $\Delta h/S$, where S is another parameter in the model. This process represents mud slides.
7. Steps 2–6 are repeated until you wish to analyze the resulting network. The river network is defined as follows. Every site in the lattice receives one unit of precipitation. Then water flows from a site to the nearest neighbor with the smallest height. Then the water flows to the neighbor of this new site with the smallest height. This flow continues until the flow reaches the bottom of the lattice. This process is repeated for each site, and the number of times that a site receives water is recorded. The river network is defined as the set of all sites that received at least R units of water, where R is another parameter of the model.

The parameters of the model can be related to measurable quantities, and the different steps of the algorithm correspond to real dynamical processes. If you are interested, you will find rich literature on the structure of river networks. An efficient way of understanding this literature is to think in terms of a model such as the one we have presented. Of course, the best way to understand the model is by converting it into a working program.

19.4 ■ CONSTRAINED DYNAMICS

After finishing many of the simulations in the text, you might wonder whether similar techniques can be used for systems in everyday life such as a roller coaster at an amusement park. How would you simulate the motion of such a system? In this case the motion of the roller coaster is constrained to a curved surface. Such a simulation is of general interest and *physically-based modeling* has become an important technique in computer animation and computer graphics (see Pixar and Eberly). The approach we will discuss is also of interest in simulating polymers where it is frequently necessary to introduce geometrical restrictions such as constant bond lengths (see Rapaport). The following considerations remind us that much of computer science is of interest in physics.

We will consider *holonomic constraints*—constraints that can be eliminated by expressing the coordinates in terms of a new set of variables. The new variables, which are fewer than the original ones, are called *generalized coordinates* and implicitly satisfy the constraints. An example of a system with a holonomic constraint is a simple pendulum. The state of the system can be described by the (x, y) coordinates of the swinging mass along with a constraint that the connecting rod has a fixed length. This constraint is holonomic because we can also describe the system by the variable θ, the angle of the mass from the pivot. The constraint is automatically satisfied in the θ-representation. Other examples of holonomic constraints include particles restricted to motion on a curve and two particles restricted to be a given distance apart.

More generally, suppose the original system coordinates are given by $r = (r_1, r_2, \ldots, r_n)$, where each r_i corresponds, for example, to a Cartesian component of a particle in the system. For example, r_1 might correspond to the horizontal position x and r_2 might correspond to the vertical position y of the first particle. If there are holonomic constraints, not all values of r correspond to valid system configurations. The idea is to introduce a set of generalized coordinates $q = (q_1, q_2, \ldots, q_f)$ and express r as a function of q such that $r(q)$ is always a valid system configuration. The reduction in the configurational dimension $n - f$ equals the number of holonomic constraints.

Our goal is to derive an ordinary differential equation of motion in terms of the generalized coordinates, that is, $\ddot{q} = \ddot{q}(q, \dot{q}, t)$. The standard way to do so is to first determine the Lagrangian, $L = T - U$, where T is the kinetic energy and U the potential energy. The equations of motion, derived in intermediate classical mechanics textbooks, are then given by

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} = 0. \tag{19.3}$$

We can convert (19.3) to a set of first-order differential equations in the usual way and solve them numerically. Unfortunately, in many cases, because the numerical solvers are not exact, the constraints may not be satisfied. Various methods have been developed to maintain the constraints. One of the more common methods called *Shake* iteratively enforces each constraint one at a time until all the constraints are satisfied within the desired level of accuracy. In the following, we discuss a more recent technique developed by several computer scientists.

We begin with Newton's equations of motion in Cartesian coordinates, which we express as

$$\mu_i \ddot{r}_i = F_i(r, \dot{r}) = F_i^{(a)} + F_i^{(c)}, \tag{19.4}$$

where F_i is the total force on a component of a particle, and μ_i is the associated mass. For example, for the motion of two particles in two dimensions of mass m_1 and m_2, respectively, we have $F_1 = F_{1x}$, $F_2 = F_{1y}$, $F_3 = F_{2x}$, $F_4 = F_{2y}$, $\mu_1 = \mu_2 = m_1$, and $\mu_3 = \mu_4 = m_2$. The total force is the sum of the applied force $F^{(a)}$ and constraint force $F^{(c)}$. For example, the applied force could be due to gravity, interparticle potentials, and friction. The constraint forces are not initially known but are such that the evolution of r_i satisfies the constraints.

Because forces that do work are best interpreted as applied forces, we assume without loss of generality that the constraint forces do no work. This assumption uniquely specifies $F^{(c)}$ and implies that

$$\sum_i F_i^{(c)} \dot{r}_i = 0, \tag{19.5}$$

if $\dot{r}$ is consistent with the constraints. To express $\dot{r}$ in terms of q and $\dot{q}$, we introduce the Jacobian of $r(q)$, $J_{i,j} = \partial r_i / \partial q_j$. We will see how to compute the matrix J in the example