Chapter 24

Simulation Methods

Mihir Arjunwadkar, *Simulation Methods*, in: P. V. Panat, *Thermodynamics and Statistical Mechanics*, Narosa Publishing House (2008).

The focus of this chapter is computer simulation methods in statistical physics. We first present a somewhat wider perspective on modeling, simulation, computation, and the two principle simulation methodologies of statistical physics. In the rest of the chapter, we discuss these two methodologies in their most basic form, and illustrate them through exploratory case studies.

The bibliography section at the end of this book lists select resources that may be useful as more comprehensive introductions to the topics discussed in this chapter. This list is in no way representative of the enormous literature on simulation in physics. We would also like to point the interested reader to the enormous number of resources on the topic, such as tutorials, lecture notes, codes, etc., that is available in the public domain over the internet.

24.1 Simulation in Statistical Physics

24.1.1 Reality, Models, Mathematics

Simply stated, a *model* is a description of a relevant aspect of the system or phenomenon of interest. The key step of the modeling process is that of making bold, meaningful, and useful abstractions of reality — This is the step that separates what is of interest from what is not, and consequently simplifies the description and the analysis that follows. Familiar abstractions in Physics include, e.g., the notion of a point particle. The scientific methodology is driven by an urge to describe reality rationally through the use of such models, the need to make predictions about it in a *quantitative* fashion, and the process of hard experimental validation of all models against reality. Indeed, a model or a theory is considered scientific only if it is manifestly *falsifiable* — Falsifiability of a model means that it is possible to make experimentally testable predictions from the model and, if these predictions turn out to be inconsistent with reality, then that could, in principle, invalidate the model.

If we are willing to ignore deep philosophical questions about the nature of reality or the nature of understanding, we might even go as far as saying that models represent

the state of our understanding of reality. For example, celebrated theories of Physics that represent our current state of understanding about the physical reality could be seen as mathematical models of physical reality. These theories have historically evolved through the same process of falsification, and through a tight handshake between the *theoretical* and *experimental* approaches.

Mathematics turns out to be a powerful, expressive, and economical language for describing reality in most branches of the scientific enterprise. In any case, the need for a *quantitative* understanding makes the use of Mathematics inevitable. As a result, we end up with mathematical models of reality.

24.1.2 Computation and Simulation

Once a mathematical model of a system under study is formulated, the next and inevitable task is to extract meaningful information from the model and to make predictions about the behaviour of the underlying system. The purely analytical approach to elucidating mathematical properties of a model is conventionally called *theory* (as opposed to *experiment*). In this approach, one invariably needs to resort to *computation* of some sort or the other, either to gain insights for analytical work, or because purely analytical treatment of a model becomes increasing unwieldy as a function of the complexity and sophistication of the model.

A *computational simulation* is a form of computation which one could meaningfully think of as *imitating* the behaviour of the underlying system, and not just as the solution of, say, a bunch of ordinary differential equations.

The use of computation-based methodologies has been on the rise over the past few decades because of the ever-increasing availability of inexpensive computing power. Because of its somewhat specialized nature, computation and simulation are sometimes considered a third scientific methodology, besides theory and experiment. Indeed, computation can be seen to extend the power of theory beyond what is possible by purely analytical means, and to help bridge the gap between theory and experiment. Moreover, in problems where an accurate, well-validated model of reality is available, simulation can, at times, almost entirely replace an actual experimental investigation. Theory, computation, and experiment thus form a tightly-linked trio of scientific methodologies.

24.1.3 Principal Simulation Methods of Statistical Physics

To a good approximation, it could be said that simulation methodologies in Statistical Physics come in two flavours, *deterministic* and *stochastic*. Molecular dynamics (MD) simulations belong, predominantly, to the deterministic category. At the other end of the spectrum of simulation methods are what are called the Monte Carlo (MC) methods, which are *stochastic* by nature. These two methodologies in their most basic form will be discussed in the rest of this chapter.

24.2 Molecular Dynamics (MD)

The basic idea of MD is to explore the behaviour of a physical system by computing, with sufficient accuracy, the trajectories of individual particles constituting the system. Along the way, relevant information about the system is accumulated, which is used for computing physical quantities as statistical averages along the trajectory. The formal basis of why MD works for statistical mechanics problems is the (often assumed) *ergodic*

hypothesis, i.e., equivalence of ensemble and time averages, which allows computing ensemble average of physical quantities of interest as time averages along a sufficiently long phase space trajectory of the system. As a side remark, note that the meaning of *particle*, as always, depends on the context and on how the system is being modeled: It could mean an individual atom, a molecule, a group of atoms that form a chemical group that is part of an even larger molecule (a protein, e.g.), or an entire star.

In this section, we will explore the MD method in its most basic form. With a view of understanding the behaviour of the method itself as thoroughly as possible, we have restricted the discussion in this section to elementary constant energy MD simulations. We note here in passing that constant energy MD (together with a constant number of particles) corresponds to a microcanonical ensemble, whereas constant temperature MD (again with a constant number of particles) corresponds to a canonical ensemble. Many important directions (e.g., constant temperature MD simulations) are not even touched upon in this introductory chapter – we refer the reader to more comprehensive texts such as those listed in the bibliography section at the end of this book.

24.2.1 Equations of Motion

The molecular dynamics method attempts to numerically integrate the equations of motion of a system of N interacting particles. In usual notation, let us write the equations of motion as

$$m_i \frac{d^2 \vec{r}_i}{dt^2} = \vec{F}_i, \tag{24.1}$$

where $i = 1, \ldots, N$ is the particle index. This Newtonian form of the equations of motion is not binding; any other convenient form (say, Lagrangian or Hamiltonian) of the equations of motion is equally alright to work with. We restrict ourselves to conservative systems and velocity-independent forces for the sake of this chapter, which makes the force $\vec{F}_i$ a function of only the particle positions $(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)$. To make things simpler, we will also assume that no external forces act on this system.

The Lennard-Jones Potential

For concreteness, let us consider a system of particles interacting with one another via the well-known Lennard-Jones potential. A conventional forms for this potential is

$$\begin{aligned} V(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) &= \sum_{i=1}^{N} \sum_{j=1}^{i-1} v(r_{ij}), \\ \text{where } \; v(r_{ij}) &= 4\epsilon \left\{ \left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right\}, \end{aligned} \tag{24.2}$$

and $r_{ij} = |\vec{r}_i - \vec{r}_j|$ is the distance between particles i and j.

A number of characteristics of this potential are worthy of note. First of all, this potential has a pairwise additive form. Further, it depends on particle positions $(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)$ only through the $N(N-1)/2$ pairwise distances $|\vec{r}_i - \vec{r}_j|$. As a consequence, it respects the homogeneity (translational invariance) and isotropy of the space in which the particles lives. For any pair of particles, the potential has a shallow minimum at distance

$$\sigma_* = 2^{\frac{1}{6}} \sigma. \tag{24.3}$$

	ϵ K	σ_* Å	m amu	ρ g/cm^3	$\sqrt{m\sigma^2/4\epsilon}$ s
He	10.80	2.57	4.002602	0.1785	8.58×10^{-13}
Ne	36.68	2.79	20.1797	0.901	1.13×10^{-12}
Ar	120.0	3.38	39.948	1.784	1.07×10^{-12}
Kr	171.0	3.6	83.8	3.74	1.38×10^{-12}
Xe	221.0	4.1	131.29	5.8971	1.73×10^{-12}

Table 24.1: Lennard-Jones parameters for noble gas systems. The last two columns list, respectively, the density ρ at 293 K, and the characteristic time scale associated with each system. The units used in this table do not conform to a single system of units (such as SI), but are commonly used because of their convenience in the atomic domain.

For distances less than this, the interactions become sharply repulsive, whereas for distances larger than this, the interactions are weakly attractive. The characteristic energy and length scales in this potential are, respectively, 4ϵ and σ; values of these parameters for a number of noble gases are given in Table 24.1.

The Lennard-Jones force $-\nabla_i V(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)$ on the i particle can be written as

$$\begin{aligned} \vec{F}_i(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) &= \sum_{j(\neq i)=1}^{N} f(r_{ij})(\vec{r}_i - \vec{r}_j), \\ \text{where } f(r_{ij}) &= 6\frac{4\epsilon}{\sigma^2}\left\{2\left(\frac{\sigma}{r_{ij}}\right)^{14} - \left(\frac{\sigma}{r_{ij}}\right)^{8}\right\}. \end{aligned} \tag{24.4}$$

System of Units for Simulation

When setting up a MD simulation, it is not very convenient to work in standard measurement units (such as Kelvin, Joule, meter, or kilogram) for at least two reasons:

1. From a computational perspective: Real numbers (*real* as in mathematics) with a fractional part are represented in typical modern digital computers as *floating-point* numbers that allow a finite and fixed amount of storage space per number, both in the *significand* (also called *mantissa*) and the *exponent*. The set floating-point numbers (and the resulting finite-precision arithmetic) has many peculiar characteristics (more about this later in the chapter).

   If we choose to stick to, say, SI units, then numerical magnitudes of most quantities are likely to be either too small or two large for the finite-precision arithmetic to handle in a meaningful fashion. This usually results in a catastrophic loss of precision, making the numerics unreliable.

2. From a purely physical perspective, it is always a good idea to work with a set of units that is based on the natural length/time/energy/mass scales in the problem. This way, the results of an analysis or computation become independent of the specific details of a system. The mathematical reason for this is that such choice of units transforms the equations of motion into a generic, dimensionless form.

   For the Lennard-Jones potential (as applied to noble gases, for example), this means that if the set of simulation units is based on the natural scales of 4ϵ

(energy), σ (length), and m (mass), then one single analysis or computation would become applicable equally to all noble gases (it would just need to be rescaled for the specific values of these quantities for a specific system).

Example. For a Lennard-Jones system, the above choice of amounts to setting $4\epsilon = \sigma = m = 1$ in Eq. 24.1–24.4. This choice also implies $\sqrt{m\sigma^2/4\epsilon}$ as the unit of time (See Table 24.1 for typical values of this time scale). Another conventional choice of units is $m = 1, \epsilon = 1$, and $\sigma_* = 1$.

Example. Consider the equation of motion of a one dimensional simple harmonic oscillator

$$\frac{d^2x}{dt^2}(t) = -\frac{k}{m}x(t). \tag{24.5}$$

There is no natural length scale in this problem. However, the natural time scale in this system is ω^{-1}, where $\omega = \sqrt{k/m}$ is the frequency of oscillation of the simple harmonic oscillator.

24.2.2 Integrating Equations of Motion: the Verlet Algorithm

Having obtained the equations of motion of a system of particles, the next task is to solve them numerically to get particle trajectories. One of the simplest, robust, and hence popular, method for numerically integrating equations of motion (Eq. 24.1) is the Verlet algorithm. In the present context, the term "numerical integration of a differential equation" simply means obtaining the values of positions $\vec{r}_i$ at time $t+\delta$ from their values at time t. The Verlet scheme is obtained by Taylor-expanding a particle's trajectory $\vec{r}(t+\delta)$ both forward and backward in time upto order 3 in δ; i.e.,

$$\begin{aligned}
\vec{r}(t+\delta) &= \vec{r}(t) + \frac{\delta}{1!}\vec{v}(t) + \frac{\delta^2}{2!}\vec{a}(t) + \frac{\delta^3}{3!}\vec{b}(t) + O(\delta^4) \\
\vec{r}(t-\delta) &= \vec{r}(t) - \frac{\delta}{1!}\vec{v}(t) + \frac{\delta^2}{2!}\vec{a}(t) - \frac{\delta^3}{3!}\vec{b}(t) + O(\delta^4),
\end{aligned}$$

where $\vec{v}$ and $\vec{a}$ stand, respectively, for the velocity and the acceleration the particle, and $\vec{b}$ for the third derivative of $\vec{r}$ with time. With some rearrangement after adding the above two expansions, we obtain the expression for the Verlet iterative solver:

$$\vec{r}(t+\delta) = 2\vec{r}(t) - \vec{r}(t-\delta) + \delta^2\vec{a}(t). \tag{24.6}$$

It is easy to see that the error (per time step of δ) in the resulting trajectory is $O(\delta^4)$, that is, of the order of the first *neglected* term in the expansions above.

To use this recurrence operationally, one needs to specify two initial position values $\vec{r}(0)$ and $\vec{r}(\delta)$ along a trajectory, from which the trajectory is computed forward in time using the above recurrence relation given a fixed and sufficiently small value of the *time step* parameter δ. The meaning of "sufficiently small" needs to be decided on a case by case basis by assessing the numerical stability of the Verlet method (see Figure 24.1 and the accompanying discussion), and from physical considerations (such as the allowable extent of numerical non-conservation of total energy).

Example. Consider the one dimensional simple harmonic oscillator (Eq. 24.5). Given two initial values at $t = 0$ and δ, the corresponding Verlet recurrence takes the form

$$x(t+\delta) = \left(2 - \delta^2\frac{k}{m}\right)x(t) - x(t-\delta),$$

and generates the trajectory values $x(0), x(\delta), x(2\delta), \ldots$. Velocities would need to be computed, for $t = \delta, 2\delta, \ldots$ as

$$v(t) = \frac{x(t+\delta) - x(t-\delta)}{2\delta}.$$

The Velocity Verlet Algorithm

Often times one is interested in computing the values of particle velocity along the trajectory. The Verlet algorithm in the form developed above does not directly incorporate this computation in its structure. Instead, one has to resort to some sort of finite-difference approximation for the velocity, such as

$$\vec{v}(t) = \frac{\vec{v}(t+\delta) - \vec{v}(t-\delta)}{2\delta}, \tag{24.7}$$

in order to compute velocities.

A variant of the Verlet algorithm, called the velocity Verlet method, resolves this situation by setting up a double recurrence in position and velocity. Given initial conditions $\vec{r}(0)$ and $\vec{v}(0)$, this modified recurrence takes the form

$$\begin{aligned} \vec{r}(t+\delta) &= \vec{r}(t) + \delta\vec{v}(t) + \frac{\delta^2}{2}\vec{a}(t) \\ \vec{v}(t+\delta) &= \vec{v}(t) + \frac{\delta}{2}\left(\vec{a}(t) + \vec{a}(t+\delta)\right). \end{aligned} \tag{24.8}$$

Here, $\vec{a}(t+\delta)$ is computed from the updated positions $\vec{r}(t+\delta)$.

Example. Consider the one dimensional simple harmonic oscillator (Eq. 24.5) again. Given initial values of x and v at $t = 0$, the corresponding Velocity Verlet recurrences take the form

$$\begin{aligned} x(t+\delta) &= \left(1 - \frac{\delta^2}{2}\frac{k}{m}\right) x(t) + \delta v(t) \\ v(t+\delta) &= v(t) - \frac{\delta}{2}\frac{k}{m}\left(x(t) + x(t+\delta)\right). \end{aligned} \tag{24.9}$$

and generates the phase-space trajectory $(x(0), v(0)), (x(\delta), v(\delta)), (x(2\delta), v(2\delta)), \ldots$.

The Verlet Method: Energy Conservation and Numerical Stability

The energy conservation behaviour of the Verlet method (in its numerical stable regime) is a consequence of the time-reversal invariance of the Verlet recurrence; i.e., the form of the Verlet recurrence (Eq. 24.6) remains unchanged under the transformation $\delta \to -\delta$.

Example. Consider the velocity Verlet method (Eq. 24.9) as applied to the one dimensional simple harmonic oscillator (Eq. 24.5) again. The behaviour of the Velocity Verlet solver as a function of the time step parameter δ is illustrated in Figure 24.1 for the simple harmonic oscillator with $k = m = 1$. For $\delta \leq 0.1$, we see that the computed solution is quite close to the true solution $\cos(t)$, the phase-plane curve traced by the oscillator is indeed a circle with unit radius, and the total energy $(x^2 + v^2)/2$ is nicely conserved (within numerical fluctuations that are not visible on the scale of the y-axis), We see that the numerical solution, although it remains oscillatory, starts differing from the exact solution increasingly as δ increases. This is also reflected as a progressively shrinking and distorting phase-plane curve, and progressively larger fluctuations in the total energy (which should ideally remain constant at the value of 1/2).

Beyond $\delta = 2$, the Verlet recurrence (Eq. 24.9) produced unphysical, non-oscillatory solutions (not shown in the figure). Whether this behaviour with respect to the time step is a numerical artifact can only be answered through a formal stability analysis; see below.

It is important to assess the numerical stability of a differential equation solver with reference to the differential equation of interest. Such analysis provides, in the least, an upper bound on the allowable time step δ. A full treatment of this topic is not possible within the scope of this chapter; however, we discuss below how this could be done for the Verlet solver (Eq. 24.6) as applied to the simple harmonic oscillator.

Example. Let us write the simple harmonic oscillator equation in the form

$$\frac{d^2x}{dt^2}(t) = -\omega^2 x(t).$$

Since we are interested in oscillatory solutions to this equation ($\omega^2 > 0$), let us write an *ansatz* of the form

$$y(t) = y_0 e^{i\alpha t}$$

for the discrete solution $y(t)$ of the Verlet recurrence (Eq. 24.6)

$$y(t+\delta) = (2 - \delta^2\omega^2)y(t) - y(t-\delta).$$

This will give us a handle over how far the solution y of the Verlet recurrence differs from the true oscillatory solution (of the equation of motion) of the form $x(t) = e^{i\omega t}$, as a function of the time step δ. Here, ω is the frequency of oscillation of the true solution $x(t)$ to the original equation of motion, and α is to be interpreted as the frequency of the solution $y(t)$ to the Verlet recurrence. Substituting this *ansatz* in the Verlet recurrence, we get

$$y_0 e^{i\alpha t}\left(e^{i\alpha\delta} - (2 - \omega^2\delta^2) + e^{-i\alpha\delta}\right) = 0,$$

which implies

$$e^{i\alpha\delta} - (2 - \omega^2\delta^2) + e^{-i\alpha\delta} = 0.$$

With some algebraic manipulations, this reduces to the identity

$$\omega^2\delta^2 = 4\sin^2\frac{\alpha\delta}{2}.$$

Clearly, there is no real value of α that would satisfy the above identity when $\delta^2 > 4/\omega^2$. This analysis shows that

1. as observed in the numerical results, $\omega\delta = 2$ represents the boundary between physical (oscillatory) and unphysical (non-oscillatory) solutions of the Verlet recurrence, and
2. this observed behaviour is not an artifact of the finite-precision numerics that was used to compute the solutions.

From purely physical considerations it could be argued that the choice of the time step parameter δ should be such that the fastest motions in the system are adequately represented in the numerical solution. Thus, e.g., when choosing a value for δ for the velocity Verlet algorithm, the typical velocity magnitudes expected to occur in the system being simulated should also be considered. These magnitudes would depend on, amongst other things, the initial velocities, and the typical magnitudes of forces/accelerations expected to occur in the system.

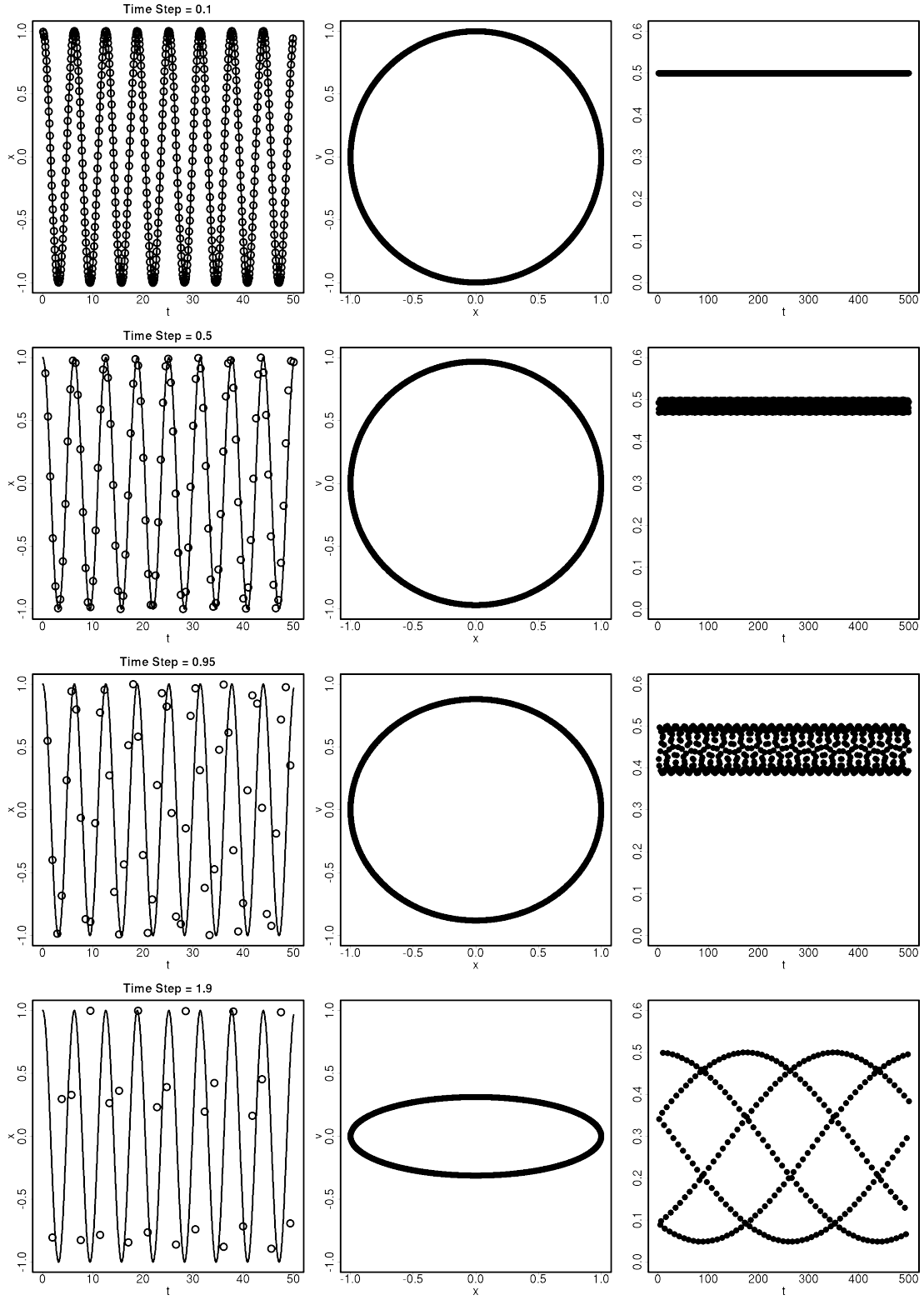

Figure 24.1: Behaviour of the velocity Verlet algorithm for the simple harmonic oscillator. The four rows correspond to $\delta = 0.1, 0.5, 0.95, 1.9$ respectively. For each row, the first column compares the computed solution (circles) for $x(t)$ with the exact solution $\cos(t)$ (continuous curve), the second column plots the energy ellipse in the x–v phase plane, and the third column plots total energy $(x^2 + v^2)/2$ as a function of t.

24.2.3 Schematics of a Bare-Basics MD Simulation

Having discussed the key elements of a MD simulation, let us return once again to a system of interacting particles (e.g., via the Lennard-Jones potential). Assuming velocity Verlet, the schematic structure of a bare-basics MD simulation is as follows:

1. Choose a convenient system of units; see discussion in Sec. 24.2.1. Choose parameters that define the system, such as the dimensionality D of the space, number N of particles in the system. Also choose values of simulation parameters such as the time step size δ, the total number N_δ of such steps, etc.
2. Specify the initial conditions for this system of N particles: For velocity Verlet, this means the initial positions $\vec{r}_i(0)$ and the initial velocities $\vec{v}_i(0)$, $i = 1, \ldots, N$.
3. For $t = \delta, 2\delta, \ldots, N_\delta\delta$:
   - compute new positions $\vec{r}_i(t)$ and velocities $\vec{v}_i(t)$ of all the particles $i = 1, \ldots, N$ from the previous positions $\vec{r}_i(t-\delta)$ and velocities $\vec{v}_i(t-\delta)$ using the velocity Verlet recurrence (Eq. 24.8).
   - store this updated information for later analysis, or compute any relevant quantities such as the total kinetic and potential energies, etc.

By construction, this simulation conserves the number of particles N and the total energy E. The volume of the system is not conserved in the present bare-basic scheme. Constant volume simulations need the system to be enclosed in a fixed-volume enclosure; this feature will be added in Sec. 24.2.5 using a device called periodic boundary conditions.

This all needs to be implemented using an appropriate programming language (such as `Fortran, Scheme, Haskell, C, C++, Java, ...`) or a scripting/computing environment (such as `python, matlab, R, ...`). The key to good computational work is the spirit of self-learning and exploration. We thus leave the programming-related details for the reader to fill-in. A variety of books on this subject already include enough details on programming and, oftentimes, actual codes[1]. A selection of such resources is included in the bibliography section at the end of this book.

24.2.4 Breathing-Mode Oscillations of Highly Symmetric Clusters

Having gained some insight into the behaviour of the Verlet solvers, let us explore the dynamics of a bit more complex system, i.e., highly symmetric Lennard-Jones clusters. Such systems whose physics is quite well-understood and where one knows what to expect at least qualitatively, are highly valuable as test cases for establishing the sanity of the simulation system and for validating the computer programs for the simulation. They often provide unexpected insights into the behaviour of the simulation system (the computer, programming language used, simulation codes, etc., taken as a whole).

Specifically, let us consider a cluster of 4 particles that are confined to the x–y plane, and interact with one another via the Lennard-Jones potential (Eq. 24.2). Our choice of simulation units is defined by $4\epsilon = \sigma = m = 1$ (see Sec. 24.2.1 and Table 24.1). For initial positions, let us place them at the corners of a square with side length $a = 0.89 \times \sigma_*$, centred at the origin, and sides parallel to the coordinate axes. Let us

[1] A suite of `Fortran` codes that accompanied the famous book *Computer Simulations of Liquids* by Allen and Tildesley (Oxford University Press, 1987) is available at the website `http://www.ccp5.ac.uk/librar.shtml#ALLENTID`

set the initial velocities to zero. For this problem, let us choose a time step $\delta = 0.05$ (in units of $\sqrt{m\sigma^2/4\epsilon}$).

As noted before, the Lennard-Jones potential is invariant under translations and rotations of the system, i.e., it respects these two symmetries of the space in which the system lives. It is clear from the symmetry of our initial state that all forces in this system act along the diagonals of the square. As such, our four particles will always be placed at the corners of a square at all times, and the dynamics of this system is completely determined by the initial size a of this square. For example, if $a \ll \sigma$, then all pairwise interactions will be highly repulsive, leading to a sharp explosion, with the four particles flying away to infinity along the four diagonal directions. If $a \gg \sigma$, then the weakly attractive interaction would lead to an initial collapse of the system followed possibly by an explosion. For $a \approx \sigma_*$, we expect to see stable breathing-mode oscillations of this square configuration. Indeed, these expectations are fulfilled as seen in a series of simulation snapshots in Figure 24.2 for the oscillatory regime.

Our Visualization Scheme. Our convention for visualizing instantaneous configurations of Lennard-Jones particles confined to a two-dimensional plane is as follows: we represent each Lennard-Jones particle by a circle of radius σ_*. With this convention, overlapping and non-overlapping pairs of circles respectively represent pairwise repulsive and attractive interactions. The centre of each circle is emphasized with a black dot, and the gray trail represents a short-time trace of this black dot over the immediate past. Depending on the particle trajectories, this trace gets smeared or erased.

Conservation of Energy, Momentum, and Angular Momentum

In Figure 24.3, we see that the total energy of this system is fairly well-conserved – i.e., within numerical fluctuations. Although these fluctuations in total energy appear quite insignificant on the scale of variation of the kinetic and potential energies taken together, they are very much present, and this is the effect of the finiteness of the time step δ. Accordingly, we expect these fluctuations to diminish at smaller values of the time step δ. In the example of Figure 24.3, the range of fluctuations reduces from about 25% of the mean value at $\delta = 0.05$ to about 1% at $\delta = 0.01$. We also expect conservation of linear and angular momentum for this system. This expectation is also fulfilled extremely well (these results are not shown).

A Finite-Precision Surprise

Ground realities of present-day computing invariably hit hard if we continue the similar simulations for longer time. Starting from highly symmetric initial configurations with appropriately spaced particles with zero velocities, we expect stable breathing-mode oscillations for our system to continue indefinitely. However, at some point in time, we often see our four particles eventually freeing themselves from the symmetric configuration, their motions becoming randomized and unpredictable.

Let us explore the dynamics of a 6-fold symmetric initial configuration of 7 particles in two dimensions, with one at the centre and the rest six sitting at the vertices of a regular hexagon. Let us set the nearest-neighbour pairwise distance in the initial configuration to $1.2\sigma_*$ and all the initial velocities to zero as before. We thus expect stable breathing-mode oscillations of this cluster, with the total force on the particle at the centre to remain exactly zero at all times, and the remaining six particles oscillating

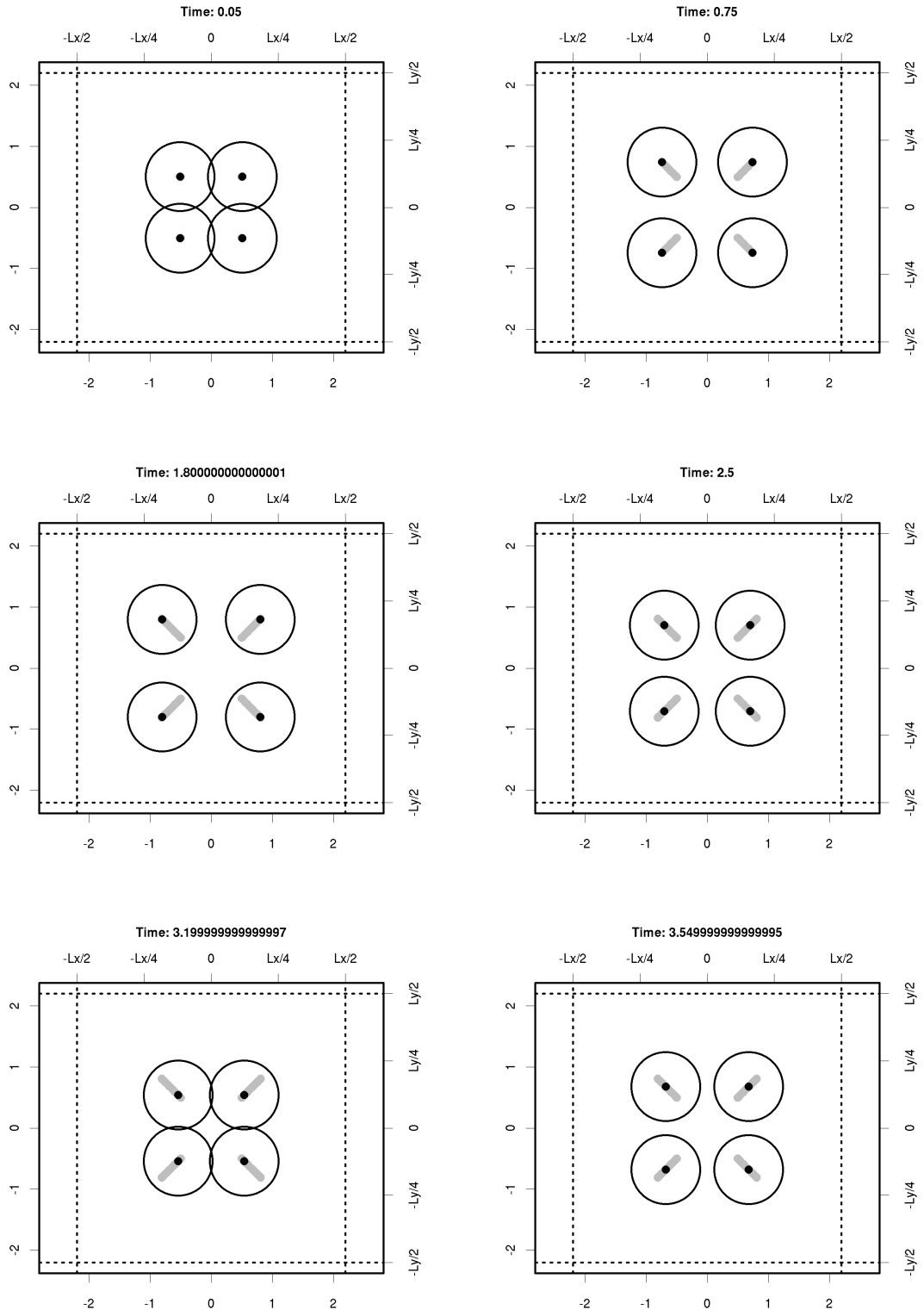

Figure 24.2: Snapshots of an oscillating Lennard-Jones square at $t = 0.05, 0.75, 1.8, 2.5, 3.2, 3.55$ (rowwise from top left to bottom right). See discussion in the text and the note on the visualization scheme used.

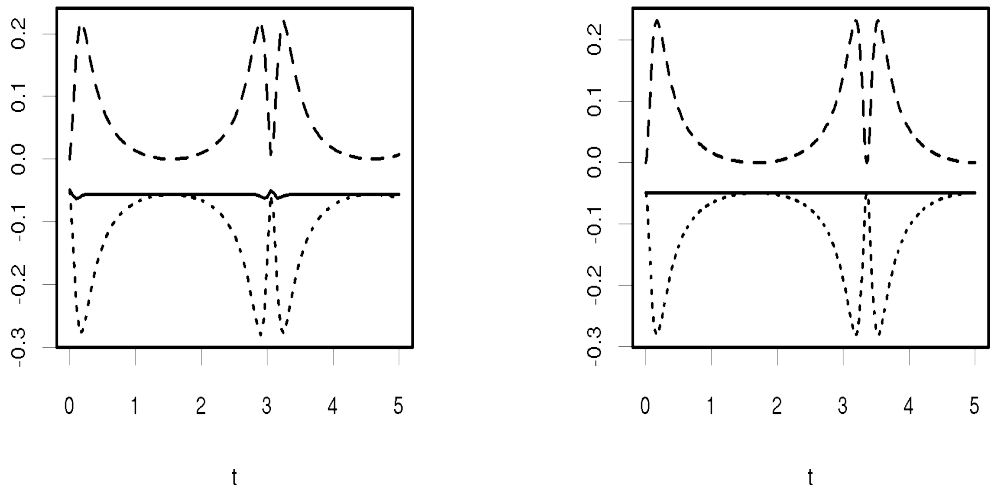

Figure 24.3: Energy conservation and the time step δ. Dashed curve: kinetic energy, dotted curve: potential energy, solid curve: total energy. All quantities are per particle, plotted as functions of time t. Left-hand plot: $\delta = 0.05$, right-hand plot: $\delta = 0.01$. Notice reduction of fluctuations in total energy at the smaller value $\delta = 0.01$.

in radial directions in a synchronized fashion. The time step parameter $\delta = 0.05$ for this illustration.

Figure 24.4 shows snapshots of the initial dynamics of this system, where we indeed see stable breathing-mode oscillations of the kind depicted in this figure. These oscillations continue upto a total time t of about 20 units, or 14 cycles of oscillation. These oscillations are also reflected in the initial part of the energy versus time plot (Figure 24.6). We also notice fairly good energy conservation (i.e, to within small numerical fluctuations) all throughout the simulation.

However, after a time of about 24 units, we start noticing something strange in the energy versus time plot: the nice oscillatory behaviour of the kinetic and potential energies has disappeared, and is replaced by some sort of patternless, seemingly random variations that still conserve the total energy. In fact, simulation snapshots (Figure 24.5) clearly show that the motions of individual particles have become visibly desynchronized and randomized.

What is causing this strange behaviour? Clearly, the qualitative dynamical behaviour we expected based on the physics for this system is not at all wrong or unjustified. The culprit for the observed randomization, as it turns out, is the peculiar way in which numbers with a fractional part are represented in present-day computing systems (i.e., the hardware, the operating system, programming language used, actual codes, etc., considered as a whole), and the corresponding finite-precision arithmetic of these numbers.

A Quick Detour into Finite-Precision Arithmetic The finite-precision representation of numbers

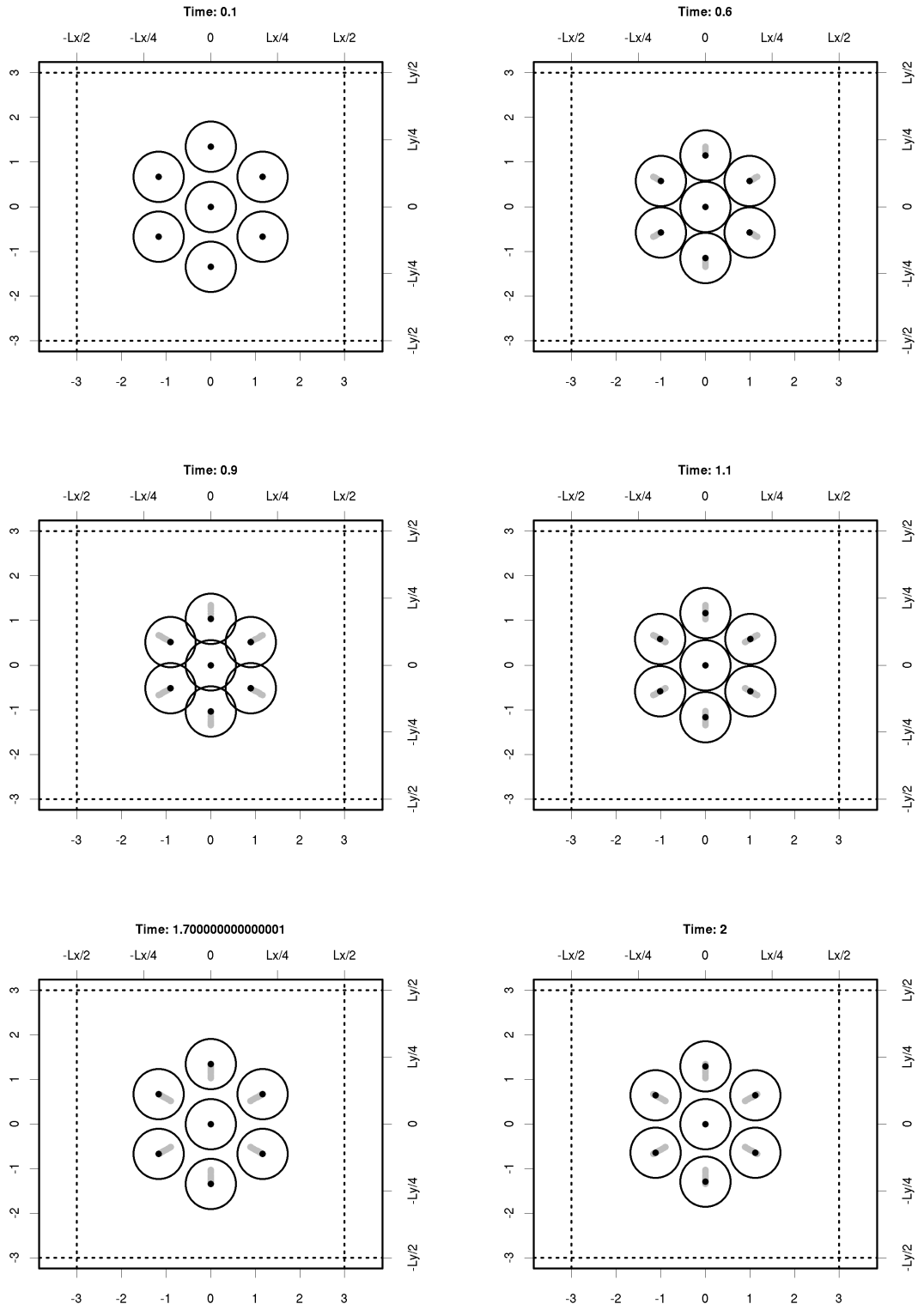

Figure 24.4: Snapshots of a hexagonal oscillating Lennard-Jones cluster at $t = 0.1, 0.6, 0.9, 1.1, 1.7, 2.0$ (rowwise from top left to bottom right).

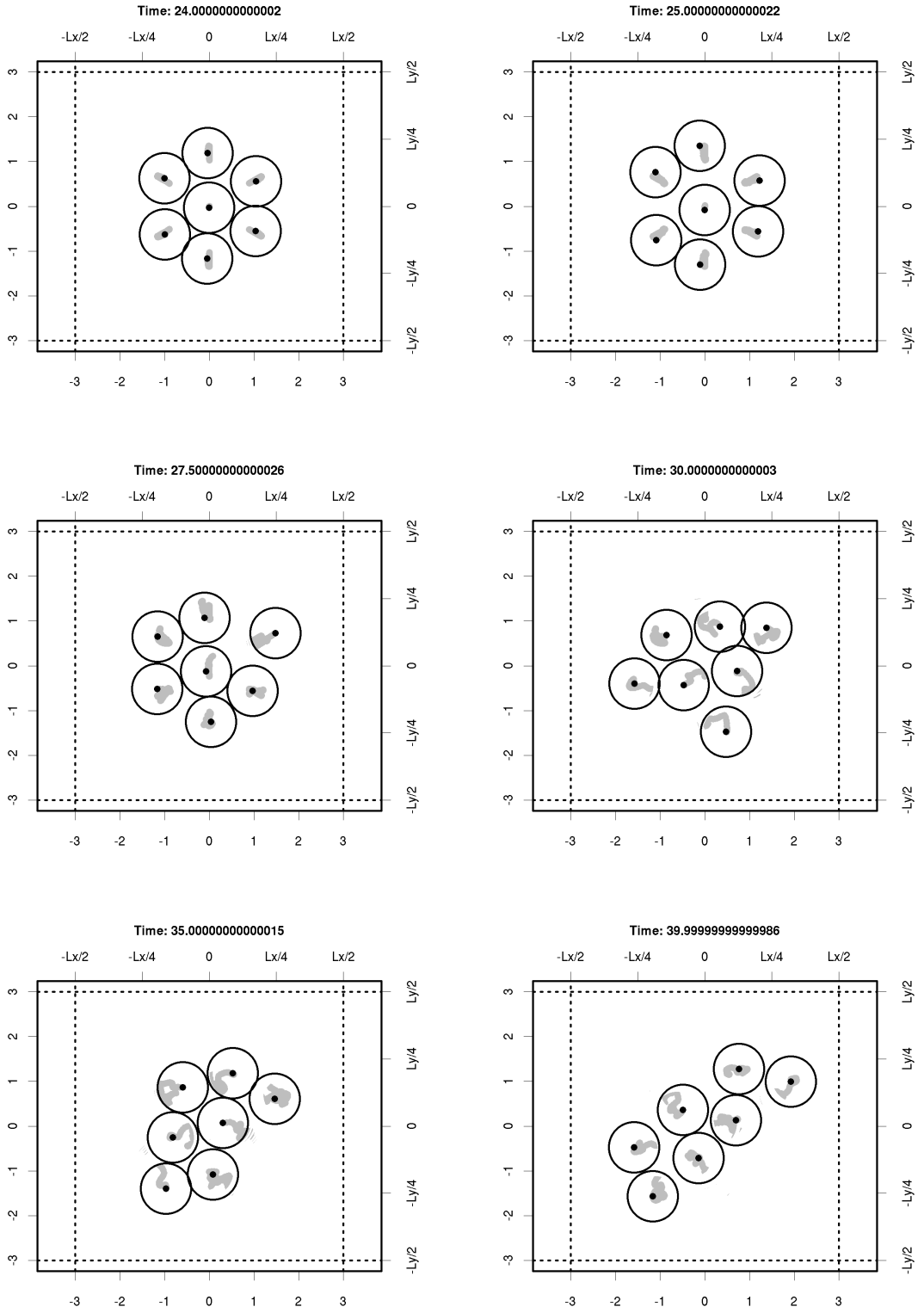

Figure 24.5: Snapshots of the (randomized) Lennard-Jones hexagon at $t = 24, 25, 27.5, 30, 35, 40$ (rowwise from top left to bottom right).

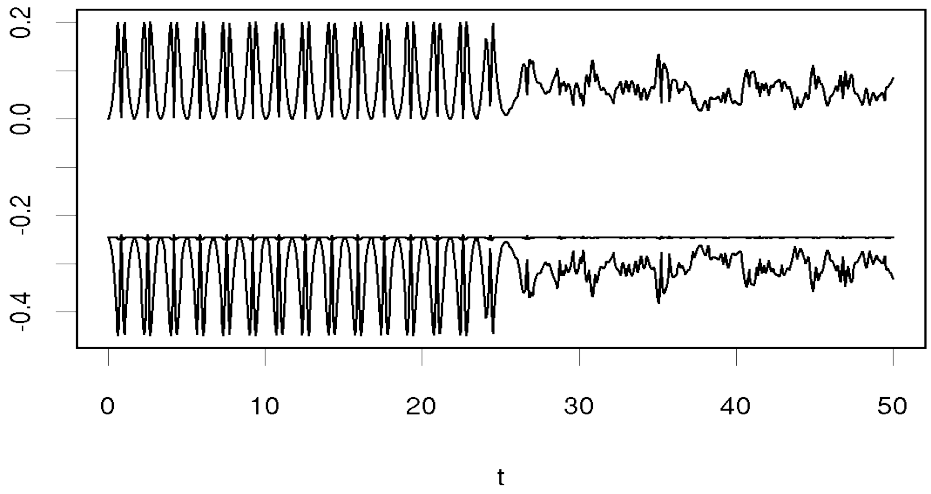

Figure 24.6: Energy conservation for the hexagonal cluster of Figures 24.4 and 24.5. Top curve: kinetic energy, bottom curve: potential energy, middle near-flat curve: total energy. All quantities are per particle, plotted as functions of time t. Notice the clear distinction between the oscillatory regime ($t < 24$) and the randomized regime. Also note that the total energy is conserved equally well in both regimes.

with a fractional part used in present-day computing systems is of the form

$$d_0.d_1 \ldots d_{p-1} \times \beta^e,$$

which stands for the real number

$$\left(d_0 + d_1\beta^{-1} + \ldots + d_{p-1}\beta^{-(p-1)}\right)\beta^e.$$

Here, β is the (integer) base of the representation (typically $\beta = 2$), $0 \leq d_0, d_1, \ldots, d_{p-1} < \beta$ are the integer digits of the real number being represented in this manner, p is the precision parameter, and e is the *exponent* that takes integer values between some e_{min} and e_{max}. The size of the storage space required per number depends on the parameters of the representation, i.e., $\beta, p, e_{min}, e_{max}$. It is clear that not all real numbers can be represented in this fashion. The resulting set of numbers that *can* be represented in this fashion, called *floating-point numbers*, is necessarily a *discrete* set, unlike the (mathematical) set of real numbers, and has non-uniform distribution (specifically, there are equal numbers of floating-point numbers in intervals defined by successive *powers* of β). Floating-point representations employed by most present-day computing systems conform to the IEEE standard 754 for floating-point arithmetic. A consequence of the finite precision is the *round-off error* that propagates through finite-precision arithmetic operations. Because of this, the result of a computation – e.g., potential and force computation, Verlet update, etc. – depends on the order in which arithmetic operations are performed, unlike in exact arithmetic. All books on numerical analysis/algorithms discuss this topic in varying detail; a few related resources are listed in the bibliography section at the end of this book.

There are interesting consequences of this. For example, the dynamics of our 4-atom system in the previous example resulting from the same initial square, but now rotated

by (say) 45° about the origin, would conceivably be different. Similarly, the same initial square configuration, now translated away from the origin, could be expected to lead to a very different dynamics: This is because of the fact that the set of floating-point numbers is a discrete set that has a non-uniform distribution and is symmetric only around the value 0. Thus, the rotational and translational invariance of our physical system need not always be respected by the finite-precision arithmetic employed for simulation.

If the idea is to simulate the dynamics of a specific system in a detailed manner and as accurately as possible, then of course such embarrassingly unphysical randomization is an unwanted artifact of the simulation system. A saving grace, from a statistical mechanical perspective (at least for a class of equilibrium problems), is that such randomization emerging from a completely unphysical source actually seems to help attain equilibrium *despite* the initial conditions, so long as it does not violate any constraints or conservation principles applicable to the simulation.

It must be understood that such artifacts of the finite-precision arithmetic are highly dependent on how the simulation is implemented in a programming language or a computing environment, initial conditions, the value of simulation parameters such as the time step δ, etc. The possibility of encountering such seemingly strange artifacts must always be kept in mind.

24.2.5 Simulating Extended Systems

From a statistical mechanical perspective, the interest is often focussed on understanding (or predicting) properties of *extended* systems consisting of large (technically, infinite) number of particles, unlike the small finite systems we explored in the previous section. On the other hand, simulations can only be done for a *finite* number of particles. The grand challenge in the simulation domain, then, is to infer the behaviour of a system in the thermodynamic limit from finite-size simulations. This is usually a hard and a tedious task.

Eliminating Surfaces Using Periodic Boundary Conditions

The least that could be done to finite system to make it resemble an infinite system is to eliminate all surfaces – Extended systems have no surfaces. The simplest way of incorporating this characteristic of extended systems into a simulation is a device called *periodic boundary conditions* (PBC).

For example, a line segment of (finite) length L has two boundaries – its left and right end points. PBC turn this line segment into a circle of circumference L by joining together the two end points. Mathematically, this is equivalent to a x *modulo* L operation, i.e., remainder of the division of x by L, which effectively maps all real numbers x onto points on this circle. The same transformation, when applied to a square, turns it into a two-dimensional torus. It could be extended to rectangular regions in any number of dimensions by applying the *modulo* L transformation to each orthogonal coordinate direction.

As the space in which a physical system lives gets wrapped around in this fashion, the interactions between particles also need to be wrapped around. This is especially true when the range of interactions (e.g., a few σ for Lennard-Jones particles) is comparable to the length L of the "simulation box" – This possibility is typically realized when the density ρ of the physical system being simulated is high enough, and the simulation box

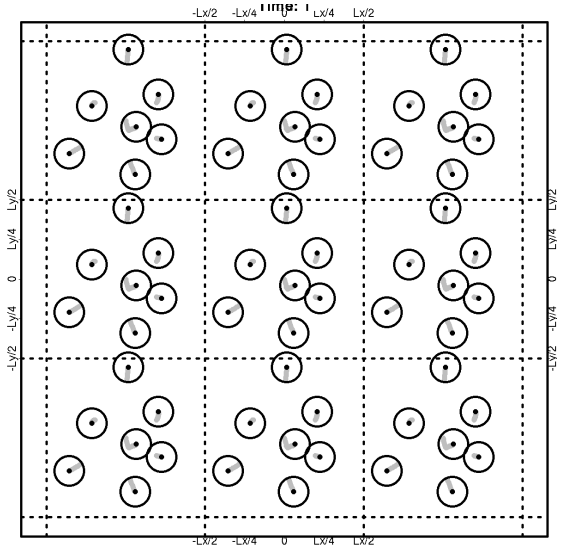

Figure 24.7: Periodic boundary conditions (PBC) visualized as the simulation box (central square) and the first shell of its images. Under PBC, particles leaving the simulation box from the right (top) wall re-enter from the left (bottom) wall and vice versa. A particle is considered to interact not only with all particles in the simulation box, but with all image particles as well. By construction, all image particles mimic their counterpart in the simulation box at the centre.

size L is chosen to match this density value for a fixed number N of particles in the simulation.

A conventional way of visualizing this wrapping-around is to consider the simulation box and its "images" which are translated from itself by integer multiples of L along all coordinate directions (see Figure 24.7). In two dimensions, e.g., the first shell of images consists of 8 images which are translations of the simulation box by translation vectors $(-L,-L)$, $(-L,0)$, $(-L,+L)$, $(0,+L)$, $(+L,+L)$, $(+L,0)$, $(+L,-L)$, and $(0,-L)$. The wrapping-around of interparticle interactions is then taken into account by considering the interaction of each particle in the simulation box not only with all other particles in the simulation box but also with the images of all particles in a shell of images of the simulation box.

PBC are typically implemented in MD simulations by identifying, at each time step, the particles that crossed the boundaries of the simulation box during that time step, and bringing them back from the opposite face. Specifically, if coordinate values within the simulation box lie between $-L/2$ and $L/2$ along each coordinate direction, then this prescription amounts to replacing, e.g., x by either $x-L$ or $x+L$ depending on whether $x > L/2$ or $x < -L/2$. This way of implementing PBC (instead of the formal *modulo* L operation) assumes that the time step δ is reasonably small enough; i.e., the fastest-moving particle in the system travels at most a small fraction of the size L of the simulation box.

To accommodate PBC in the schematics of Sec. 24.2.3, one needs to specify additionally the simulation box size L and the number of image shells around the simulation box. Typically, the density ρ of the physical system being simulated (see, e.g., Table 24.1) is used to determine L by fixing the number of particles N in the simulation.

With the inclusion of PBC, our MD simulations will now conserve the volume V

of the system, in addition to the particle number N and the total energy E. Such microcanonical simulations are sometimes referred to as constant NVT simulations.

Periodic Boundary Conditions Violate Isotropy

A side-effect of incorporating PBC in a simulation must be noted: While the translational invariance in a system remains unbroken in presence of PBC, the PBC do break the isotropy (or rotational invariance) of a system. As a consequence, unlike in the simulations of the previous section which did not incorporate PBC, the total angular momentum is not a conserved quantity any more (even if the original physical system is isotropic). We thus expect to see large fluctuations in it. However, over length scales far smaller than the simulation box length L, we could expect that the system would still appear *locally* isotropic.

Approach to Equilibrium

We end this section with the example of a simulation of an extended system as implemented through PBC. Broadly speaking, thermodynamic equilibrium is characterized by *stationarity* (i.e, time-independence) of the probability distributions describing the system. This implies that static properties of the system also become time-independent. In simulations, whether a system has attained equilibrium or not can thus be judged by monitoring the time evolution of a static property of the system. For example, for a two-dimensional system, we may define a quantity of the form

$$\begin{aligned} h &= \frac{1}{2}\left(h_x + h_y\right), \text{ where} \\ h_x &= \int P(v_x)\log(P(v_x))dv_x, \text{ and} \\ h_y &= \int P(v_y)\log(P(v_y))dv_y, \end{aligned} \tag{24.10}$$

and $P(v_x)$ is, e.g., the probability distribution function (PDF) of the x-components of particlewise velocities. Figure 24.8 shows a possible behaviour of this quantity as the system attains equilibrium. From the figure, it is clear that this quantity becomes stationary fairly quickly for the initial conditions used (see below), after just about 100 time steps. It must be understood that numerical simulations with finite systems will always involve fluctuations.

This particular simulation consisted of $N = 100$ Lennard-Jones particles confined to a two-dimensional box with PBC. The simulation box length L was determined from this N and an arbitrary density value of $\rho = 0.75$ in simulation units (i.e, on an average 0.75 particles per unit σ^2). As initial positions, these 100 particles were placed on a regular square grid over the simulation box. The time step δ was 0.005. Initial velocities were randomly chosen as +4 or -4 (this number is, again, to be interpreted in our simulation units) for each component of velocity of each particle, ensuring that the centre-of-mass velocity remains zero. In effect, the initial velocity distributions $P(v_x)$ and $P(v_y)$ have the shape of discrete and equally tall spikes at ± 4, as seen in the left-hand plot in Figure 24.9. The right-hand plot in the same figure plots position component against the corresponding velocity component across particles. For our specific initial state, this reflects the velocity spikes at ± 4, and the regular cubic lattice structure for position components. More generally, such a plot is indicative of the joint probability distribution

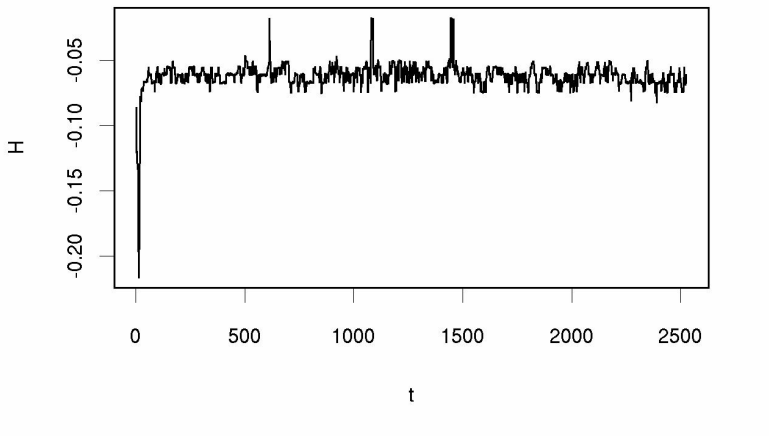

Figure 24.8: Approach to equilibrium as seen through the H function (Eq. 24.10).

of a position component with the corresponding velocity component across all particles. Since there is no conceptual distinction between the x and y directions for our system, we have not treated or plotted them separately.

Boltzmann's H Theorem Boltzmann's H theorem states that the approach of the system to equilibrium is accompanied by minimization of what is called the H function, which is defined as

$$H(t) = \int f(\vec{r}, \vec{v}, t) \log(f(\vec{r}, \vec{v}, t)) d\vec{r} d\vec{v}.$$

Once equilibrium is attained, this quantity remains stationary at its minimum value. Here, f is the single-particle number distribution function over the phase space and a solution of the Boltzmann transport equation. A 1971 paper by E.T. Jaynes[2] argues that the H theorem can be violated by systems with appreciable potential energy starting from specific classes of initial conditions and, in fact, the approach to equilibrium here is accompanied by a *maximization* of the H function. These classes of initial conditions are common enough to be encountered in experiments and simulations. Lennard-Jones systems, especially at the high density value (0.75) and initial conditions used for the simulation behind Figure 24.8 are also known to violate the H theorem[3]. The quantity h that we defined (Eq. 24.10) is a kind of H function that is restricted to velocities.

It is interesting to look at how the velocity distributions themselves evolve to stationarity as the system approaches equilibrium. Starting from a velocity distribution with sharp peaks at ± 4 (Figure 24.9), Figure 24.10 shows the distribution of velocity components (bottom left) and componentwise position-velocity correlations (bottom right) as before. This figure is a snapshot of these quantities over the first 50 times steps, that is for $0 \leq t < 50$. In principle, the same figure could have been done for each time step separately; however, such pooling together of data across time steps help reduce the fluctuations to some extent and make the generic features of the distributions stand out

[2]E.T. Jaynes, *Violation of Boltzmann's H Theorem in Real Gases*, Phys. Rev. A4, 747–750 (1971).

[3]V. Romero-Rochin and E. Gonzalez-Tovar, *Comments on Some Aspects of Boltzmann H Theorem Using Reversible Molecular Dynamics*, J. Stat. Phys. 89, 735–749 (1997).

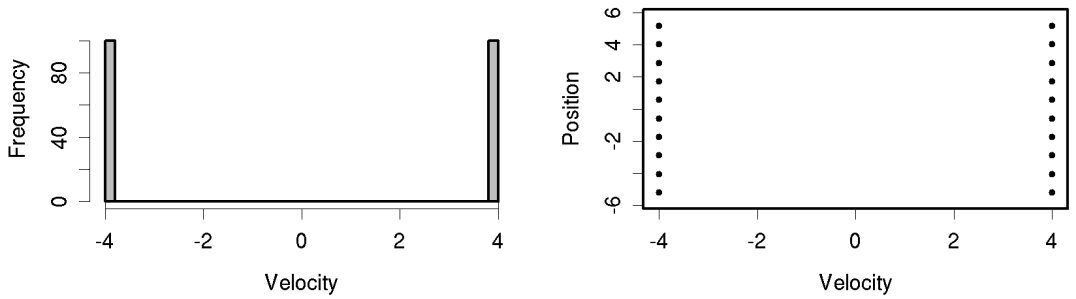

Figure 24.9: Left: Initial distribution of velocity components. Right: Componentwise position-velocity correlations.

clearly. In this figure, we also see the distribution of velocity *magnitudes* (top left), and and position-velocity *magnitude* correlations (top right).

Notice how the original sharp peaks at ± 4 at $t = 0$ got smeared out by the process of equilibriation. This smearing effect is also clearly seen in both the right-hand plots when compared to the original discrete, ordered picture at $t = 0$ in Figure 24.9. We just state here that the shape of these distributions becomes more or less time-independent (barring fluctuations, of course) after about 100 time steps or $t = 0.5$ (we have not included any figure to demonstrate this).

24.3 Monte Carlo (MC) Methods

Monte Carlo methods are a class of versatile methods for

- generating random samples from arbitrary high-dimensional multivariate probability distributions, and
- for estimating expectation values of a quantity of interest with respect to a high-dimensional multivariate distribution. As a consequence of this, Monte Carlo methods can be used for estimating the value of a (high-dimensional) integral I that can be expressed as expectation value of a function h with respect to a probability density function f (i.e., I could be expressed in the form $\int h(x)f(x)dx$).

In the context of statistical physics, this translates to

- randomly sampling configurations (or phase space points) of a system in such a way that their distribution follows the probability density function of an appropriate ensemble (e.g., the canonical density function $f(\cdot) = \exp\left(-\beta E(\cdot)\right)/Z$), and
- estimating expectation values of physical properties of interest with respect to this ensemble-specific probability density function f.

It is perhaps no wonder that Monte Carlo methods originated (in the form of the celebrated 1953 paper by Metropolis et al.[4]) in the realm of statistical physics which is replete with high-dimensional distributions and integrals. What makes these methods

[4]Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller, *Equation of State Calculations by Fast Computing Machines*, J. Chem. Phys. 21, 1087–1092 (1953).

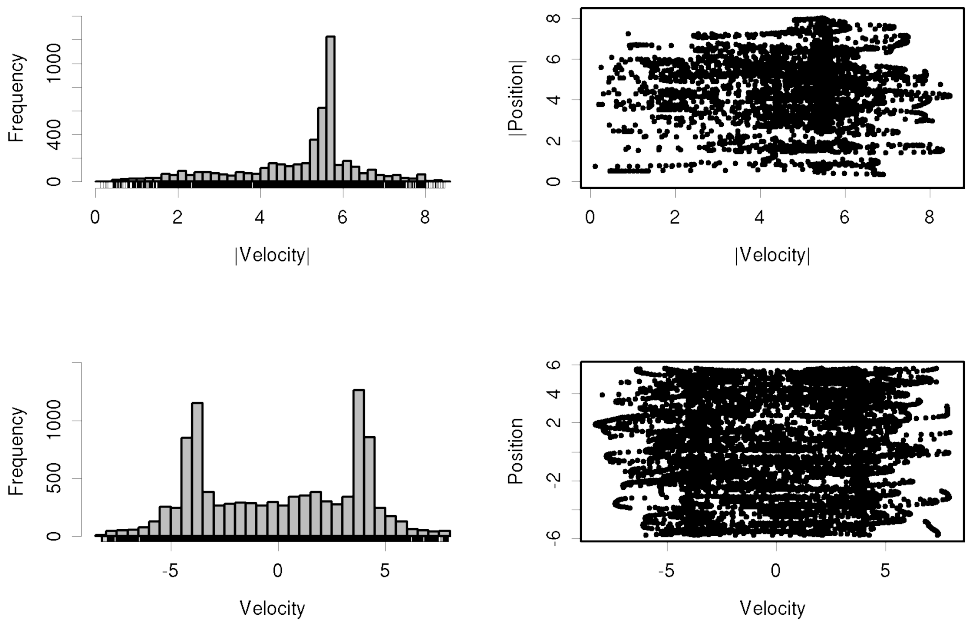

Figure 24.10: $t = 0.25$ (time step 50). Top left: Velocity *magnitude* distribution. Top right: position-velocity *magnitude* correlations. Bottom left: Velocity component distribution. Bottom right: Componentwise position-velocity correlations.

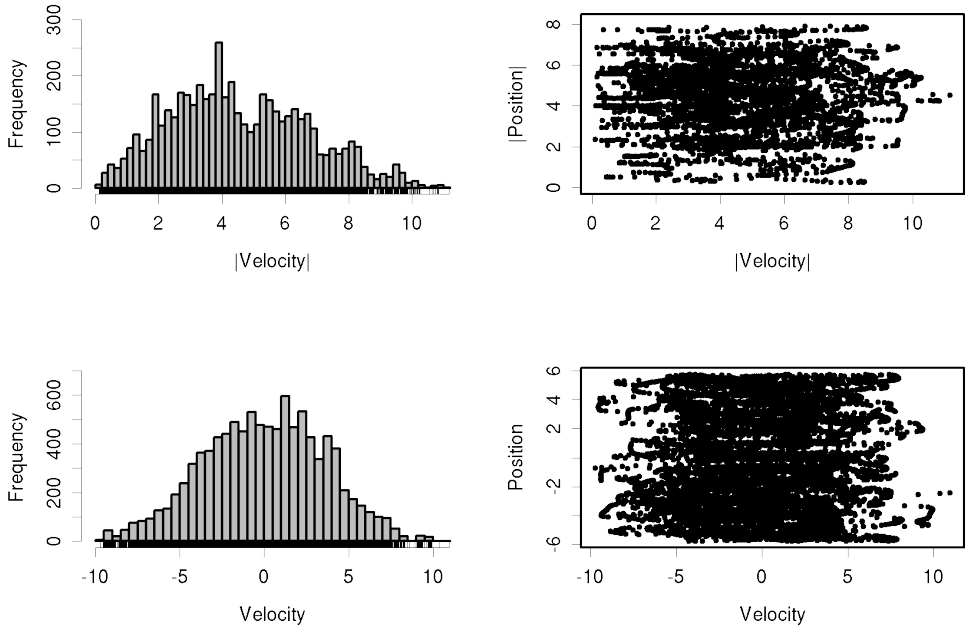

Figure 24.11: $t = 0.5$ (time step 100). Conventions same as in Figure 24.10.

immensely useful tools for exploring probability distributions is the fact that these methods do not require that the normalization constant (e.g., $1/Z$) be known. The formal basis of why (and under what conditions) Monte Carlo methods work is founded in the theory of Markov chains.

24.3.1 The Metropolis Algorithm

Consider a system, at a constant temperature T, consisting of N particles interacting via a potential $V(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)$. The probability distribution function (PDF) that describes the *spatial* structure of the system has the form

$$P(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N) = \frac{1}{Z} \exp\left\{-\frac{V(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)}{kT}\right\} \equiv P(\{\vec{r}\}). \tag{24.11}$$

A *configuration*[5] $\vec{R}$ of this system is specified by the positions $(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)$ of all particles. We wish to generate a sample of such configurations $\vec{R}_1, \vec{R}_2, \ldots, \vec{R}_M$, of the system in such a fashion that their probabilities of occurrence are in accordance with the PDF (Eq. 24.11). In addition, we may wish to compute expectation values, with respect to this PDF, of quantities of interest (such as, average kinetic and potential energies, a quantity called the *radial distribution function* that describes spatial correlations between particles, etc.)

The Metropolis algorithm for this task, in its conventional form, is as follows. Specify an arbitrary configuration $\vec{R}_0$. If possible, choose this from a high probability density region of the PDF (Eq. 24.11). Suppose we have generated a sequence of such configurations $\vec{R}_0, \vec{R}_1, \ldots, \vec{R}_{n-1}$ using the Metropolis algorithm (to be described shortly). The next configuration $\vec{R}_n$ is generated by repeating the steps 1–4 below N times (giving each particle a chance to get displaced on an average once):

1. Choose particle i at random (with uniform probability) from the N particles of the system.
2. Propose a change in its position by giving it a random displacement, i.e.,

   $$\vec{\rho}_i = \vec{r}_i + \delta\vec{u}.$$

   Here, δ is a parameter that determines the maximum displacement along each coordinate. $\vec{u}$ is a random vector (with the same dimensionality D as that of $\vec{r}_i$), with each of its coordinates being picked with uniform density over the interval $(-1, 1)$. This is typically accomplished by using uniform (pseudo-)random number generators available in most programming languages, computing environments, and numerical/statistical libraries. In effect, we have picked $\vec{\rho}_i$ a random point in a D-dimensional hypercube of length δ centered on $\vec{r}_i$.
3. Compute the *acceptance probability* $a(\vec{\rho}_i, \vec{r}_i)$ for the displaced position $\vec{\rho}_i$ given the current position $\vec{r}_i$, where

   $$a(\vec{\rho}_i, \vec{r}_i) = \min\left\{1, \frac{P(\ldots, \vec{\rho}_i, \ldots)}{P(\ldots, \vec{r}_i, \ldots)}\right\}. \tag{24.12}$$

[5] The notation $\vec{R}$ in this chapter has the same meaning as $\{\vec{r}\}$ in the rest of this book. In this chapter, we have use both the notations as per convenience.

4. Accept the proposed position $\vec{\rho}_i$ with probability $a(\vec{\rho}_i, \vec{r}_i)$. This is accomplished as follows: generate a uniform random number u from the interval $(0, 1)$. If $u < a(\vec{\rho}_i, \vec{r}_i)$, then accept the proposed change and displace the ith particle to its new position $\vec{\rho}_i$. If $u > a(\vec{\rho}_i, \vec{r}_i)$, reject the proposed change, and the ith particle remains in its present position $\vec{r}_i$.

Steps 1–4 above are often collectively referred to as a *Monte Carlo step*. Because of the special forms of the Boltzmann PDF $P(\vec{R})$ (Eq. 24.11) and the acceptance probability (Eq. 24.12), the Metropolis algorithm accepts all "downhill" moves that result in the lowering of the potential energy $V(\vec{R})$ of the system. These are precisely the moves that take the system "uphill" in the PDF $P(\vec{R})$. Proposed moves that tend to increase the potential energy of the system are not rejected outright – they are accepted in a probabilistic fashion in step 4 above. This feature allows, in principle, the Metropolis method to move from local peaks in the PDF (equivalently, local minima in the potential energy) to other or higher peaks in the PDF (i.e., other or lower potential energy configurations) to eventually sample the configuration space as described probabilistically by the PDF.

No Need to Know the Normalization Factor

A highly desirable feature of the Metropolis algorithm is that the PDF $P(\vec{R})$ need not be known precisely all the way upto the normalization factor $1/Z$. This is because the acceptance probability involves only a *ratio* of PDF values (proposed change against the current position), which makes the normalization factor $1/Z$ redundant for the purpose of this algorithm. This is a very useful feature because such normalization factors are usually extremely hard or impossible to compute.

Step Size δ and the Acceptance Ratio

This algorithm, in the form above, needs the step size parameter δ to be specified. A standard prescription for choosing the right value of δ is to monitor the average *acceptance ratio* α, defined as the ratio of the number of accepted changes to the number of proposed changes, and adjust the step size δ such that α remains within a pre-specified range around 0.5, say between 0.4 to 0.6.

Why Metropolis Works

This algorithm, the way it is described here, may sound rather *ad hoc*. However, it has a well-established theoretical justification in the theory of Markov chains. Specifically, provided that the step size δ is not unreasonably large or small, the chain of configurations thus generated turns out to be an *ergodic* Markov chain. A Markov chain is called ergodic if, in our context, it is possible to go from every configuration to every other configuration (not necessarily in a single step). Such Markov chains have a unique stationary distribution (i.e., one that does not change once it is attained). Furthermore, it is guaranteed to be attained eventually. The form of the acceptance probability above guarantees, through what is called the *condition of detailed balance*, that this stationary distribution will indeed be our desired PDF $P(\vec{R})$.

Monitoring Equilibriation and Computing Averages

All this boils down to mean that this algorithm will *eventually* start producing configurations of the system that are indeed distributed according to $P(\vec{R})$. Practically, the

problem of how long one needs to wait for this to happen needs to be addressed in an empirical fashion, i.e., by trial and error. The evolution of the Markov chain to the desired PDF is often called *thermalization* or *equilibriation* by analogy to the process of equilibriation of a physical system.

Expectation values of physical quantities computed are thus to be computed after rejecting the initial transient portion of the chain of configurations, i.e., after the Markov chain has attained the desired PDF. Expectation values of a quantity $A(\vec{R})$ of interest, which is typically a function of the configuration $\vec{R}$, is computed as a simple average over a sufficiently long equilibrium portion of the Markov chain, i.e.,

$$\langle A \rangle = \frac{1}{M} \sum_{i=1}^{M} A(\vec{R}_i).$$

This is indeed an estimate of the true expectation value of A because the PDF $P(\vec{R})$ is now represented in the sample of configurations itself.

24.3.2 The Lennard-Jones Fluid Again

Let us illustrate these ideas again with the example of a two-dimensional Lennard-Jones system. As an example of a physically-relevant quantity to be estimated through the simulation, let us consider the radial distribution function $g(r)$.

The Radial Distribution Function

The spatial structure of an isotropic fluid (such as Lennard-Jones) is often characterized by what is variously known as the *radial distribution function* or the *pair correlation function*. This is formally defined as

$$g(\vec{r}_1, \vec{r}_2) = \frac{N(N-1)}{\rho^2} \frac{\int d\vec{r}_3 \ldots \vec{r}_N \exp\left(-\beta V(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)\right)}{\int d\vec{r}_1 \ldots \vec{r}_N \exp\left(-\beta V(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N)\right)}.$$

For an isotropic and translationally invariant system, $g(\vec{r}_1, \vec{r}_2)$ depends only on $|\vec{r}_1 - \vec{r}_2|$ and not on $\vec{r}_1$ and $\vec{r}_2$ separately. We thus denote it by $g(r)$. This function serves as a link between simulations and experiments, because its Fourier transform

$$S(k) = 1 + 4\pi\rho \int \frac{\sin(kr)}{kr} g(r) r^2 dr$$

is an experimentally measurable quantity (e.g., through X-ray scattering). Furthermore, for a system of particles interacting with a pairwise additive potential (such as Lennard-Jones) $g(r)$ has the form $\exp(-\beta v(r))$ in the dilute gas limit ($\rho \to 0$). Here, $v(r)$ is the potential energy of a single pair of particles (see Eq. 24.2).

For a solid in a perfectly crystalline state, the $g(r)$ comprises of sharp peaks because crystalline order allows only a discrete set of pair distances to occur. In the liquid state, depending on the temperature, these sharp peaks get smeared out because of the mobility of particles in the liquid state. However, smeared-out relics of the first few peaks still survive, because particle-particle correlations at short distances still exist in the liquid state. In the gas phase, only the first peak survives in a much broadened form. For a Lennard-Jones fluid at constant temperature, so long as the potential energy dominates (i.e., at low enough temperatures) it would be difficult to find pairs of particles that are closer than σ. We therefore expect that $g(r) = 0$ for $r \lesssim \sigma$ for a Lennard-Jones fluid.

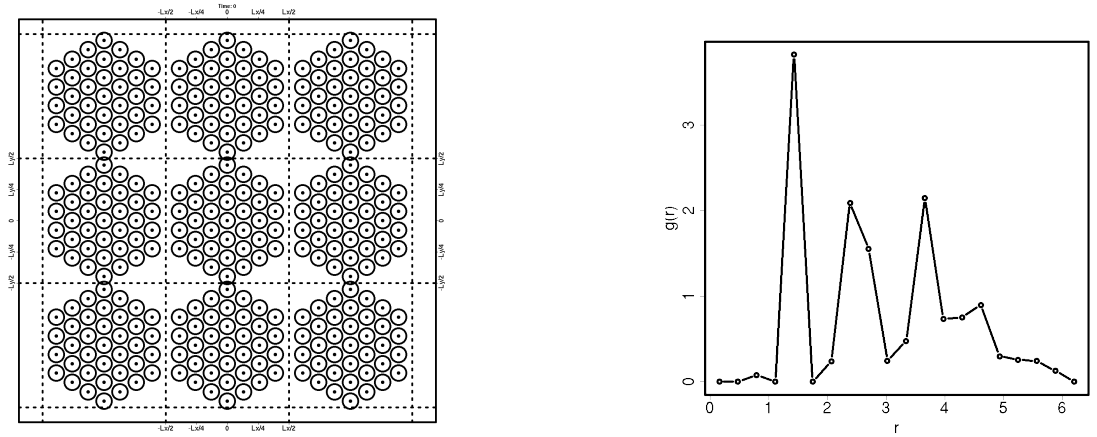

Figure 24.12: Initial configuration for MC simulation at $\beta = 10$ and the corresponding $g(r)$. Notice the sharp peaks characteristic of solid-like order.

Operationally, $g(r)$ is estimated by counting $n(r)$, the average number of particle pairs with distances in the range $r - \frac{1}{2}\Delta r$ and $r + \frac{1}{2}\Delta r$. The estimate of $g(r)$ is simply this number suitably normalized and further averaged over all angular variables to account for isotropy:

$$\begin{aligned} g(r) &= \frac{V}{\frac{1}{2}N(N-1)} \frac{n(r)}{2\pi r \Delta r} \text{ (two dimensions)} \\ &= \frac{V}{\frac{1}{2}N(N-1)} \frac{n(r)}{4\pi r^2 \Delta r} \text{ (three dimensions).} \end{aligned} \tag{24.13}$$

Here, V stands for the volume of the system in three dimensions and area of the system in two dimensions.

A consequence of PBC is that the maximum possible separation between any pair of particles along any one coordinate direction is $L/2$. Geometrically, this could be understood as follows: PBCs turn a line segment into a circle. On a circle, the distance between a pair of points is not unique (i.e., the clockwise and anticlockwise distances match only for pairs of diametrically opposite points). This ambiguity is resolved by taking the shorter of these two distances. In simulations with PBC, this is done in a coordinatewise fashion. Thus the maximum possible separation between any pair of particles in D dimensions is $L\sqrt{D}/2$. This must be taken into consideration when computing $g(r)$ in a configuration R of the system.

The quantity $n(r)$ in the definition of $g(r)$ (Eq. 24.13) is estimated by dividing the interval $0 \leq r \leq L\sqrt{D}/2$ into N_g intervals of equal length $\Delta r = L\sqrt{D}/(2N_g)$, and counting the number of pairs with distances in each of these intervals.

Illustrative Results for $g(r)$

The MC simulation of a Lennard-Jones system presented here was performed at a fairly low temperature of $kT = 0.1$ ($\beta = 10$). The value of the step size parameter for the Metropolis algorithm was $\delta = 0.15$, chosen by trial-and-error so as to keep the average acceptance ratio α in the range $0.4 \lesssim \alpha \lesssim 0.6$.

Assuming that the system is solid-like at this temperature, we constructed a peculiar initial state consisting of $N = 37$ particles arranged in a two-dimensional hexagonal

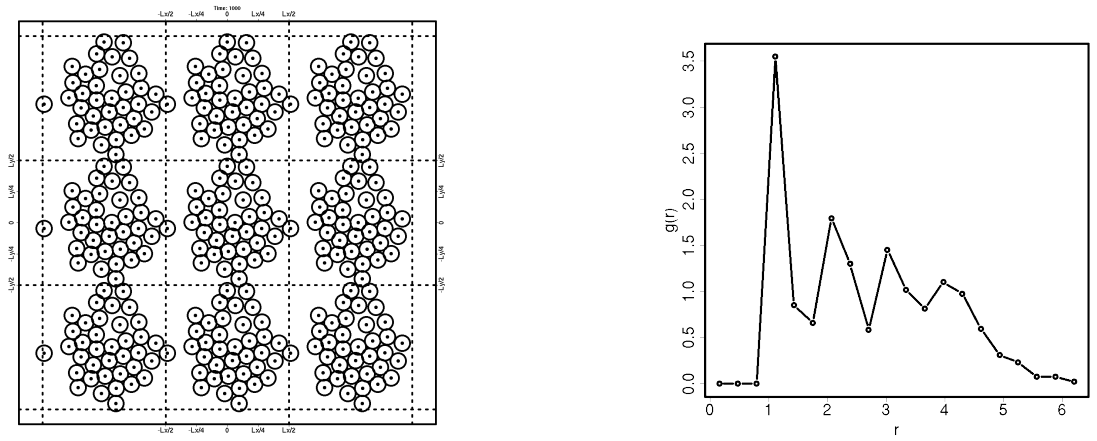

Figure 24.13: A typical Configuration generated by the Metropolis algorithm ($t = 1000, \beta = 10$), and the corresponding $g(r)$.

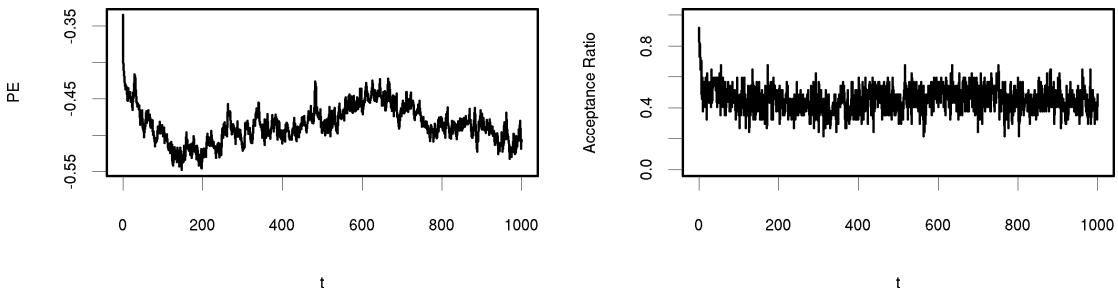

Figure 24.14: Potential energy V and the average acceptance ratio α as functions of the Monte Carlo step index t ($\beta = 10$).

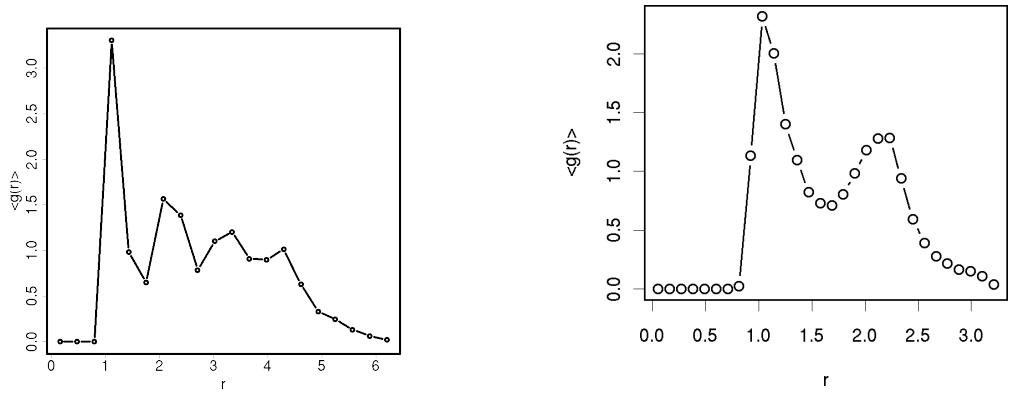

Figure 24.15: Left-hand plot: Monte Carlo averaged $g(r)$ at $\beta = 10$. Right-hand plot: the same quantity for a 16-particle system at density $\rho = 0.75$ and $\beta = 1$, for comparison.

packed structure (Figure 24.12). we have thus employed the periodic boundary conditions (PBC) with $L \approx 9$, treating this as an extended fluid system. This corresponds to a density $\rho \approx 37/81 \approx 0.46$ particles per unit area (density measured in σ^{-2} units).

The behaviour of the potential energy $V(\vec{R}_t)$ as a function of the Monte Carlo "time" t (i.e., configuration index along the Markov chain) is shown in Figure 24.14. The corresponding behaviour of the Boltzmann PDF $P(\vec{R})$ (Eq. 24.11) is not shown separately; it follows from the behaviour of the potential energy. Notice the distinction between the non-equilibrium and the equilibrium portions of the Markov chain of configurations. Figure 24.14 also shows the variation of the average acceptance ratio α. A typical configuration in the equilibrium portion of the chain is shown in Figure 24.13.

For estimation of quantities of interest, one needs to discard the initial non-equilibrium portion of the chain (approximately the initial 200 configurations in this simulation). As mentioned before, the word "equilibriation" in the Markov Chain Monte Carlo context means attaining the desired stationary probability distribution function $P(\vec{R})$ by the Markov chain. Figure 24.15 shows the radial radial distribution function $g(r)$ averaged over the equilibrium portion of the Markov chain (i.e., after discarding the initial 200 configurations). The presence of multiple sharpish peaks are indicative of a solid-like phase. For comparison, the same figure also includes a similarly averaged-out $g(r)$ at $\beta = 1$ for a Lennard-Jones system consisting of 16 particles at density $\rho = 0.75$. Here, the third and higher peaks are absent, which indicates a liquid-like phase.

Further Resources

Computational Physics

1. Harvey Gould, Jan Tobochnik, and Wolfgang Christian, *Introduction to Computer Simulation Methods* (Addison-Wesley, 2006).
2. Franz Vesely, *Computational Physics: an Introduction* (Kluwer Academic/Plenum Publishers, Second Edition, 2001).
3. J.M. Thijssen, *Computational Physics* (Cambridge University Press, 1999).
4. M.P. Allen and D.J. Tildesley, *Computer Simulations of Liquids* (Oxford University Press, 1989).

Molecular Dynamics

1. D.C. Rapaport, *The Art of Molecular Dynamics Simulation* (Cambridge University Press, 1995).
2. J.M. Haile, *Molecular Dynamics Simulation: Elementary Methods* (John Wiley & Sons, 1997).
3. Daan Frenkel and B. Smit, *Understanding Molecular Simulation* (Academic Press, Second edition, 2001).

Monte Carlo Methods

1. M.H. Kalos and P.A. Whitlock, *Monte Carlo Methods. Vol. 1: Basics* (Wiley Interscience, 1986).
2. M.E.J. Newman and G.T. Barkema, *Monte Carlo Methods in Statistical Physics* (Oxford University Press, 1999).
3. D.P. Landau, *A Guide to Monte Carlo Simulations in Statistical Physics* (Cambridge University Press, Second edition, 2005).
4. K.P.N. Murthy, *Monte Carlo Methods in Statistical Physics* (University Press, 2004).
5. Gilks, Richardson, and Spiegelhalter, *Markov Chain Monte Carlo Methods in Practice* (Chapman and Hall, 1996).

Numerics

1. H.M. Antia, *Numerical Methods for Scientists and Engineers* (Hindusthan Book Agency, Second edition, 2002).
2. W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing* (Cambridge University Press, Second edition, 1992).