# 12
# Thermodynamic Simulations: Ising Model

*This chapter applies the Ising model and the Metropolis algorithm to simulate magnetic materials. This is a more advanced application of the Monte Carlo techniques studied in Chap. 11, in which thermodynamic equilibrium is simulated. It is an important topic which may give readers the first true understanding of what is "dynamic" in thermodynamics.*

Your **problem** is to see if a simple model can explain the thermal behavior and phase transitions of ferromagnets.

## 12.1
## Statistical Mechanics (Theory)

Ferromagnetic materials contain finite-size *domains* in which, even in the absence of an external magnetic field, the spins of all the atoms are aligned in the same direction. When an external magnetic field is applied to these materials at low temperatures, the different domains align and the material becomes "magnetized." Yet, as the temperature is raised, the magnetism decreases and then goes through a *phase transition* at the Curie temperature, beyond which all magnetization vanishes.

When we say that an object is *at* a temperature $T$, we mean that the object's atoms are in thermodynamic equilibrium at temperature $T$. While this may be an equilibrium state, it is a dynamic one in which system's energy is fluctuating as it exchanges energy with the environment (it is thermo*dynamics* after all). However, each atom does have an average energy proportional to $T$.

In the present problem we deal with the thermal properties of magnetized materials. The magnetism arises from the alignment of the spins of the atoms within domains. When the number of atoms is large, the problem is too big to solve completely, and so statistical methods are used to obtain average quantities (in most cases that is all we can measure, anyway). If the system is described microscopically by classical or quantum mechanics, then this method is called *statistical mechanics*.

*Computationyal Physics. Problem Solving with Computers (2nd edn).*
Rubin H. Landau, Manuel José Páez, Cristian C. Bordeianu

ISBN: 978-3-527-40626-5

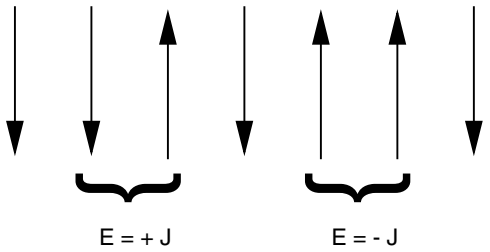


**Fig. 12.1** A 1D lattice of $N$ spins. The interaction energy $V = \pm J$ between nearest-neighbor pairs is shown for aligned and opposing spins.

Statistical mechanics starts with the elementary interactions among particles of a system and constructs the macroscopic thermodynamic properties such as temperature $T$ and internal energy $U$. The essential assumption is that all configurations of the system consistent with the constraints are possible. Because we have the temperature, volume, and number of particles fixed, we have a *canonical ensemble* or Boltzmann distribution. The energy $E(\alpha_j)$ of a state $\alpha_j$ in a canonical ensemble is not fixed, but rather is distributed with probabilities $P(\alpha_j)$ according to the Boltzmann distribution:

$$P(\alpha_j) = \frac{e^{-E(\alpha_j)/kT}}{Z(T)} \qquad Z(T) = \sum_{\alpha_j} e^{-E_j/kT} \tag{12.1}$$

where $k$ is Boltzmann's constant and $T$ is the temperature. The partition function $Z(T)$ in (12.1) is a weighted sum over states. Nonetheless, because we will be dealing with the *ratio* of probabilities, the $Z(T)$ factor cancels out, and we do not have to concern ourselves with it.

Notice that the Boltzmann distribution (12.1) does not require a thermal system to be in the state of lowest energy. Rather, it states that it is less likely for the system to have a high energy. Of course, as $T \to 0$, only the $E = 0$ state has a nonvanishing probability. For finite temperatures we expect the system's energy to have fluctuation on the order of $kT$.

## 12.2 An Ising Chain (Model)

As our model, we consider $N$ magnetic dipoles fixed on the links of a linear chain (Fig. 12.1). (It is a straightforward generalization to handle 2- and 3D lattices, and indeed we do it as an exploration.) Because the particles are fixed, their positions and momenta are not dynamical variables, and we need worry only about their spins.

We assume that the particle at site $i$ has spin $s_i$, which is either up or down:

$$s_i \equiv s_{z,i} = \pm\tfrac{1}{2} \tag{12.2}$$

Each possible configuration or state of the $N$ particles is described by the quantum state vector

$$|\alpha_j\rangle = |s_1, s_2, \ldots, s_N\rangle = \{\pm \tfrac{1}{2}, \pm \tfrac{1}{2}, \ldots\} \qquad j = 1, 2^N \tag{12.3}$$

Because the spin of each particle can assume any one of the *two* values, there are $2^N$ different possible states of the $N$ particles in the system. We do not need to concern ourselves with the symmetry of the wave function since fixed particles cannot be interchanged.

The energy of the system arises from the interaction of the spins with each other and with the external magnetic field $B$. We know from quantum mechanics that an electron's spin and magnetic moment are proportional to each other, so a "dipole–dipole" interaction is equivalent to a "spin–spin" interaction. We assume that each dipole interacts with the external magnetic field and with its nearest neighbor through the potential:

$$V_i = -J\,\mathbf{s}_i \cdot \mathbf{s}_{i+1} - g\mu_b\,\mathbf{s}_i \cdot \mathbf{B} \tag{12.4}$$

Here the constant $J$ is called the *exchange energy* and is a measure of the strength of the spin–spin interaction. The constant $g$ is the gyromagnetic ratio, that is, the proportionality constant between the angular momentum and magnetic moment. The constant $\mu_b$ is the Bohr magneton, the unit for magnetic moments.

Because the $2^N$ possible configurations of the spins become large even for small numbers of particles ($2^{20} > 10^6$), the computer can examine all possible spin configurations only for small $N$. Realistic samples with $\sim 10^{23}$ particles would be beyond imagination. Consequently, we apply a statistical approach based on the Boltzman distribution (12.1), and assume that the statistics are valid even for moderate values of $N$. Just how large $N$ must be for this to occur is one of the things we want you to discover with your simulations. In general, the answer depends on how good a description is required; for pedagogical purposes $N \succeq 200$ may appear statistical, with $N \succeq 2000$ more reliable.

The energy of the system to be used in the Boltzmann distribution (12.1) is the expectation value of the sum of $V$ over the spins of the particles:

$$E(\alpha) = \langle\alpha| \sum_i V_i |\alpha\rangle = -J \sum_{i=1}^{N-1} s_i s_{i+1} - B\mu_b \sum_{i=1}^{N} s_i \tag{12.5}$$

An apparent paradox in the Ising model occurs when we turn off the external magnetic field and thereby eliminate a preferred direction in space. This

means that the average magnetization should vanish, even though the lowest energy state would have all spins aligned. The answer to this paradox is that the system with $B = 0$ is unstable. Even if all the spins are aligned, there is nothing to stop a spontaneously reversal of all the spins. Indeed, natural magnetic materials have multiple finite domains with all the spins aligned, but with the different domains pointing in arbitrary directions. The instabilities in which a domains change directions are called *Bloch-wall transitions*. For simplicity, and to start, you may assume that $B = 0$, and just look at how the spins interact with each other. However, we recommend that you always include a very small external magnetic field in order to stabilize the simulation against spontaneous flipping of all the spins.

The equilibrium alignment of the spins depends critically on the sign of the exchange energy $J$. If $J > 0$, the lowest energy state will tend to have neighboring spins aligned. If the temperature is low enough, the ground state will be a *ferromagnet* with all spins aligned. If $J < 0$, the lowest energy state will tend to have neighbors with opposite spins. If the temperature is low enough, the ground state will be a *antiferromagnet* with alternating spins.

A simple model such as this has its limits. (Nonetheless, it is not hard to add improvements, such as longer range interactions, motion of the centers, higher multiplicity spin states, two and three dimensions, etc.) First, although the model is accurate in describing a system in thermal equilibrium, it is not quantitatively accurate in describing the *approach* to thermal equilibrium. On the one hand, nonequilibrium thermodynamics is a difficult subject in which the theory is not complete, while on the other hand, as part of our algorithm we postulate that only one spin gets flipped at a time, while a real magnetic material may flip many.

A fascinating aspect of magnetic materials is the existence of a critical temperature, the *Curie temperature*, above which the gross magnetization essentially vanishes. Below the Curie temperature the quantum state of the material has long-range order extending over macroscopic dimensions; above the Curie temperature there is only short-range order extending over atomic dimensions. Even though the 1D Ising model predicts realistic temperature dependences for the thermodynamic quantities, the model is too simple to support a phase transition. However, the 2D and 3D Ising models does support the Curie-temperature phase transition.

One of the most illuminating aspects of this simulation is the visualization showing that a system described by the Boltzmann distribution (12.1) does not have a single configuration. Rather, there is a continual and random interchange of thermal energy with the environment that leads to fluctuations in the total energy. Even at equilibrium, you should see the system fluctuating, with the fluctuations getting larger as the temperature rises.

### 12.2.1 Analytic Solutions

For very large numbers of particles, the thermodynamic properties of the 1D Ising model can be solved analytically [18]. The solution tells us that the average energy $U$ is

$$\frac{U}{J} = -N \tanh \frac{J}{kT} = -N \frac{e^{J/kT} - e^{-J/kT}}{e^{J/kT} + e^{-J/kT}} = \begin{cases} N & kT \to 0 \\ 0 & kT \to \infty \end{cases} \tag{12.6}$$

The analytic results for the specific heat per particle and the magnetization are

$$C(kT) = \frac{1}{N}\frac{dU}{dT} = \frac{(J/kT)^2}{\cosh^2(J/kT)} \tag{12.7}$$

$$M(kT) = \frac{Ne^{J/kT}\sinh(B/kT)}{\sqrt{e^{2J/kT}\sinh^2(B/kT) + e^{-2J/kT}}}. \tag{12.8}$$

The **2D Ising model** has an analytic solution, but it was not an easy one to find [19,20]. Whereas the internal energy and heat capacity are found in terms of elliptic integrals, the spontaneous magnetization per particle has the rather simple form

$$\mathcal{M}(T) = \begin{cases} 0 & T > T_c \\ \frac{(1+z^2)^{1/4}(1-6z^2+z^4)^{1/8}}{\sqrt{1-z^2}} & T < T_c \end{cases} \qquad \begin{matrix} kT_c \simeq 2.269185J \\ z = e^{-2J/kT} \end{matrix} \tag{12.9}$$

where the temperature is measured in units of the Curie temperature $T_c$, and $z$ is a dimensionless variable.

## 12.3 The Metropolis Algorithm

We need an algorithm to evaluate the $2^N$ sums that appear in the energy sum (12.5). This is analogous to a $2^N$-dimensional numerical integration, and we know from Section 11.6.2 that a Monte Carlo approach is best for high-dimensional integrations. Yet it would be a waste of time to generate all possible random configurations in a uniform manner because the Boltzmann factor essentially vanishes for those configurations whose energies are not close to the minimum energy. In other words, the majority of the terms we sum over hardly contribute at all, and it is quicker to restrict the sum somewhat to those terms which contribute the most.

In their simulation of neutron transmission through matter, Metropolis, Rosenbluth, Teller, and Teller [21] invented an algorithm to improve the Monte

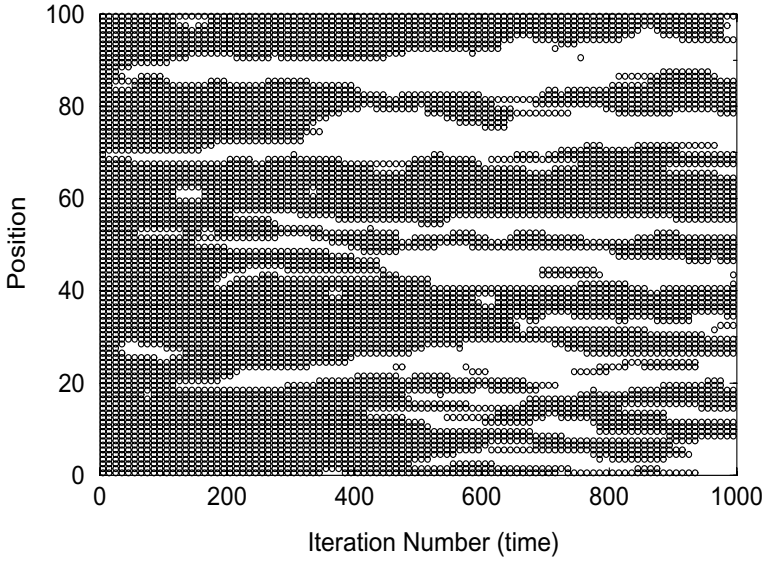


**Fig. 12.2** A 1D lattice of 100 spins aligned along the ordinate. Up spins are indicated by circles and down spins by blank spaces. The iteration number ("time") dependence of the spins is shown along the abscissa. Even though the system starts with all up spins (a "cold" start) the system is seen to form domains as it equilibrates.

Carlo calculation of averages. This *Metropolis algorithm* is now a cornerstone of computational physics. The sequence of configurations it produces (a *Markov chain*) accurately simulates the fluctuations that occur during thermal equilibrium. The algorithm randomly changes the individual spins such that, on the average, the probability of a configuration occurring follows a Boltzmann distribution. (We do not find the proof of this trivial or particularly illuminating.)

The Metropolis algorithm is a combination of the variance reduction technique, discussed in Section 11.7.1, and the von Neumann rejection technique, discussed in Section 11.7.4. There we showed how to make Monte Carlo integration more efficient by sampling random points predominately where the integrand is large, and how to generate random points with an arbitrary probability distribution. Now we would like have spins flip randomly, have a system that can reach any configuration in a finite number of steps (*ergodic* sampling), and have a distribution of energies described by a Boltzmann distribution. (A direct Monte Carlo simulation would take prohibitively long times.) In a sense, the Metropolis algorithm is a simulated version of the annealing process used to make sword blades hard and flexible. Let us say that we are making a blade for a sword and are hammering away at it while it is red hot to get its shape just right. At this high a temperature there is a lot of internal motion and not the long-range order needed for a hard blade. So, as

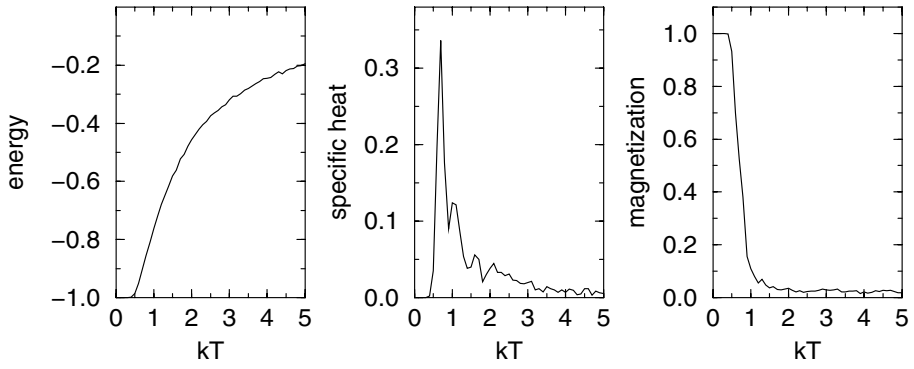


**Fig. 12.3** Simulation results for the energy, specific heat, and magnetization of a 1D lattice of 100 spins as a function of temperature.

part of the process, we *anneal* the blade; that is, we heat and slow cool it in order to reduce brittleness and increase strength. Too rapid a cooling would not permit long-range equilibration (and ordering) of the blade, which would lead to brittleness.

Use of the Metropolis algorithm is done via a number of steps. We start with a fixed temperature and an initial configuration for the spins and apply the algorithm until thermal equilibrium is reached (equilibration). Then the algorithm generates the statistical fluctuations about equilibrium from which we deduce the thermodynamic quantities such as the internal energy $U(T)$ and the magnetization $M(T)$. After that, the temperature is changed and the whole process is repeated in order to deduce the $T$ dependence for the thermodynamic quantities. It is the accuracy of the temperature dependences that provides the convincing evidence for the algorithm. Because the possible $2^N$ configurations can be a very large number, the amount of computer time needed can be very long, even though the program is simple. Typically, a small number $\simeq 10N$ of iterations is adequate for equilibration.
Here is an outline of the Metropolis algorithm:

1. Start with an arbitrary spin configuration $\alpha_k = \{s_1, s_2, \ldots, s_N\}$.
2. Generate a trial configuration $\alpha_{k+1}$:
   (a) Pick particle $i$ randomly.
   (b) Reverse $i$'s spin direction.
3. Calculate the energy $E(\alpha_{\text{tr}})$ of the trial configuration.
4. If $E(\alpha_{\text{tr}}) \leq E(\alpha_k)$, accept by setting $\alpha_{k+1} = \alpha_{\text{tr}}$.
5. If $E(\alpha_{\text{tr}}) > E(\alpha_k)$, accept with relative probability $\mathcal{P} = \exp(-\Delta E/kT)$:

(a) Choose a uniform random $r_j$ $0 \le r_i \le 1$.

(b) $\alpha_{k+1} = \begin{cases} \alpha_{\text{tr}} & \text{if } \mathcal{P} \ge r_j \text{ (accept)} \\ \alpha_k & \text{if } \mathcal{P} < r_j \text{ (reject)} \end{cases}$

The heart of this algorithm is its generation of a random spin configuration $\alpha_j$ (12.3) with probability

$$\mathcal{P}(\alpha_j) \propto e^{-E(\alpha_j)/kT} \tag{12.10}$$

The technique is a variation of von Neumann rejection (stone throwing of Section 11.4) in which a random *trial* configuration is either accepted or rejected depending upon the value of the Boltzmann factor. Explicitly, the ratio of probabilities for a trail configuration of energy $E_t$ to that of an initial configuration of energy $E_i$ is

$$\frac{\mathcal{P}_{tr}}{\mathcal{P}_i} = e^{-\Delta E/kT} \qquad \Delta E = E_{tr} - E_i \tag{12.11}$$

If the trial configuration has a lower energy ($\Delta E \le 0$), the relative probability will be greater than one and we accept the trial configuration as the new initial configuration with no further ado. However, if the trial configuration has a higher energy ($\Delta E > 0$), we do not reject out of hand because the system has moved away from its lowest energy state. Instead, we accept it with relative probability $\mathcal{P}_{tr}/\mathcal{P}_i = \exp(-\Delta E/kT) < 1$. To accept a configuration with a probability, we pick a uniform random number between 0 and 1, and if the probability is greater than this number, we accept the trial configuration; if the probability is smaller than the chosen random number, we reject it. (You can remember which way this goes by letting $E_t \to \infty$, in which case the probability $\to 0$ and nothing is accepted.) When the trial configuration is rejected, the next configuration is identical to the preceding one.

The key aspect of the Metropolis algorithm is that the weight given to a trial configuration depends on how far it is from the minimum-energy configuration. Those configurations that stray far from the minimum-energy configuration are deemphasized but not completely discarded. By permitting $\Delta E > 0$, we are permitting the system to go "uphill" for a while. This deviation away from a direct path to the minimum-energy configuration permits the algorithm to get away from a local minimum and instead find a global one. Its success relies on it not being too quick in "cooling" to the minimum-energy configuration; for this reason the algorithm is sometimes called *simulated annealing*.

How do you start? One possibility is to start with random values of the spins (a "hot" start). Another possibility (Fig. 12.2) is to start with all spins parallel or antiparallel (a "cold start" for positive and negative $J$, respectively).

In general, one tries to remove the importance of the starting configuration by letting the calculation "run a while" ($\simeq 10N$ rearrangements) before calculating the equilibrium thermodynamic quantities. You should get similar results for hot, cold or arbitrary starts, and by taking their average, you remove some of the statistical fluctuations.

### 12.3.1 Metropolis Algorithm Implementation

1. Write a program that implements the Metropolis algorithm, that is, that produces a new configuration $\alpha_{k+1}$ from the present configuration $\alpha_k$. (Alternatively, use the program `Ising.java`.)
2. Make the key data structure in your program an array `s[N]` containing the values of $s_i$. For debugging, print out + and $-$ to give the spins at each lattice point and the trial number.
3. The value for the exchange energy $J$ fixes the scale for energy. Keep it fixed at $J = 1$. (You may also wish to study antiferromagnets with $J = -1$, but first examine ferromagnets whose domains are easier to understand.)
4. The thermal energy $kT$ is in units of $J$ and is the independent variable. Use $kT = 1$ for debugging.
5. Use periodic boundary conditions on your chain to minimize end effects. This means that the chain is a circle with the first and last spins adjacent to each other.
6. Try $N \simeq 20$ for debugging and larger values for production runs.
7. Use the printout to check that the system equilibrates for
   (a) a totally ordered initial configuration (cold start) and
   (b) a random initial configuration (hot start).

Your cold start simulation should resemble Fig. 12.2.

### 12.3.2 Equilibration (Assessment)

1. Watch a chain of $N$ atoms attain thermal equilibrium when in contact with a heat bath. At high temperatures, or for small numbers of atoms, you should see large fluctuations, while at lower temperatures you should see smaller fluctuations.

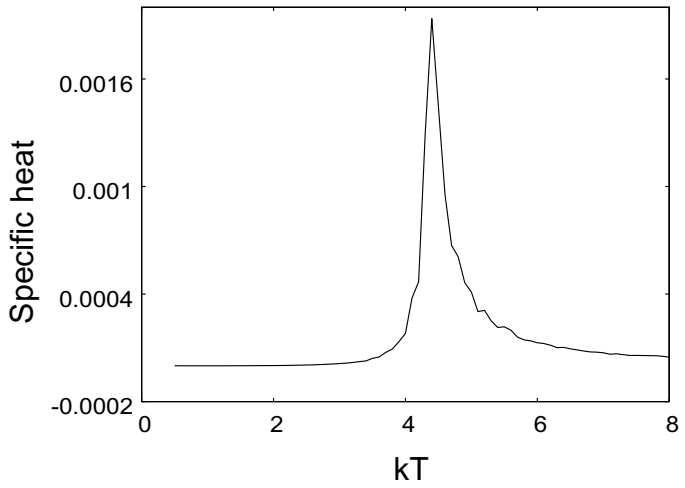


**Fig. 12.4** The specific heat of a 3D Ising lattice as a function of temperature.

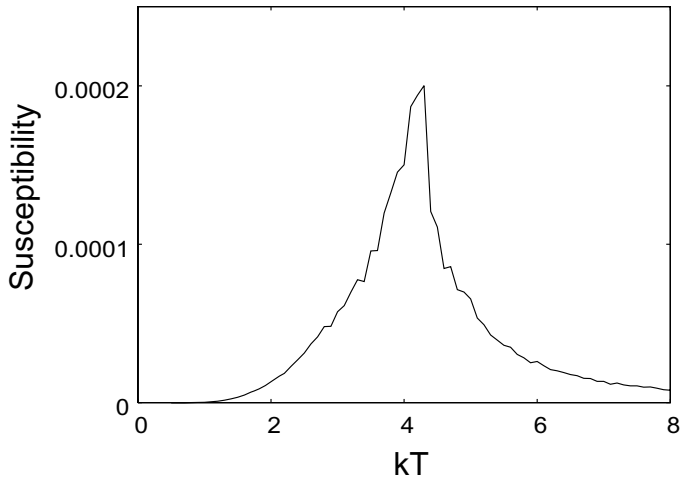


**Fig. 12.5** The susceptibility of a 3D Ising lattice as a function of temperature.

2. The largest $kT$ may be unstable for $b = 0$ because the system can absorb enough energy to flip all its spin.

3. Note how at thermal "equilibrium" the system is still quite dynamic with spins flipping all the time. It is the energy exchange that determines the thermodynamic properties.

4. You may well find that simulations at small $kT$ (say that $kT \simeq 0.1$ for $N = 200$) are slow to equilibrate. Higher $kT$ values equilibrate faster, yet have larger fluctuations.

5. Observe the formation of domains and the effect they have on the total energy. Regardless of the direction of a spin within a domain, the atom–atom interactions are attractive and so contribute negative amounts to the energy of the system when aligned. However, the $\uparrow\downarrow$ or $\downarrow\uparrow$ interactions between domains contribute positive energy. Therefore, you should expect a more negative energy at lower temperatures where there are larger and fewer domains.

6. Make a graph of average domain size versus temperature.

### 12.3.3 Thermodynamic Properties (Assessment)

For a given spin configuration $\alpha_j$, the energy and magnetization are given by

$$E_j = -J \sum_{i=1}^{N-1} s_i s_{i+1} \qquad \mathcal{M}_j = \sum_{i=1}^{N} s_i \tag{12.12}$$

At high temperatures we expect a random assortment of spins and so a vanishing magnetization. At low temperature we expect $\mathcal{M}$ to approach $N/2$ as all the spins get aligned. Although the specific heat (Fig. 12.4) can be computed from the elementary definition

$$C = \frac{1}{N}\frac{dU}{dT} \tag{12.13}$$

doing a numerical differentiation of a fluctuating variable is not expected to be accurate. A better way is to first calculate the fluctuations in energy occurring during $M$ trials

$$U_2 = \frac{1}{M}\sum_{t=1}^{M}(E_t)^2 \tag{12.14}$$

and then determine the specific heat from the energy fluctuations:

$$C = \frac{1}{N^2}\frac{U_2 - (U)^2}{kT^2} = \frac{1}{N^2}\frac{\langle E^2\rangle - \langle E\rangle^2}{kT^2} \tag{12.15}$$

1. Extend your program to calculate the internal energy $U$ and the magnetization $\mathcal{M}$ for the chain. Incorporate the fact that you do not have to recalculate entire sums for each new configuration because only one spin changes.

2. Make sure to wait for your system to equilibrate before you calculate thermodynamic quantities. (You can check that $U$ is fluctuating about its average.) Your results should resemble those shown in Fig. 12.3.

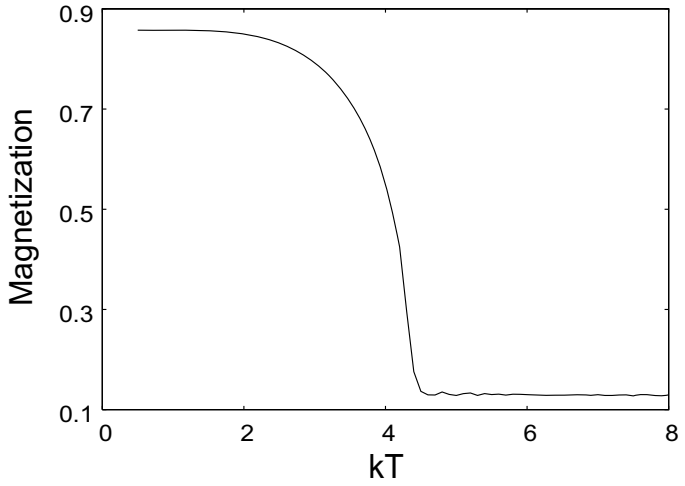


**Fig. 12.6** The magnetization of a 3D Ising lattice as a function of temperature.

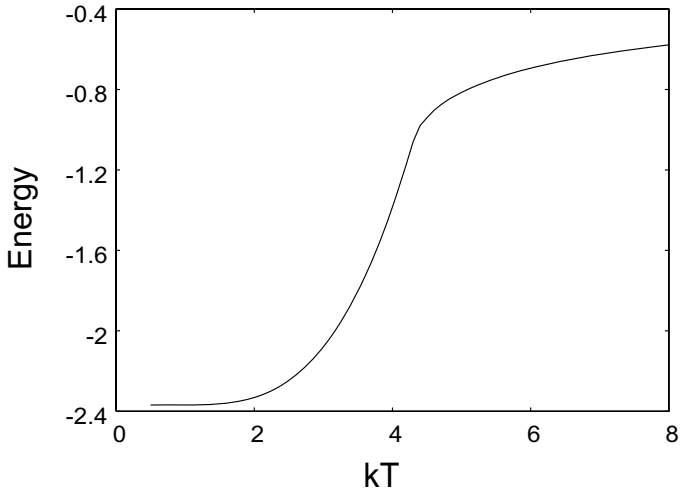


**Fig. 12.7** The energy of a 3D Ising lattice as a function of temperature.

3. Reduce the statistical fluctuations by running the simulation a number of times with different seeds, and taking the average of results.

4. The simulations you run for small $N$ may be realistic but may not agree with statistical mechanics, which assumes $N \simeq \infty$ (you may assume that $N \simeq 2000$ is close to infinity). Check that agreement with the analytic results for the thermodynamic limit is better for large $N$ than small $N$.

5. Check that the simulated thermodynamic quantities are independent of initial conditions (within statistical uncertainties). In practice, your cold and hot start results should agree.
6. Make a plot of the internal energy $U$ as a function of $kT$ and compare to the analytic result (12.6).
7. Make a plot of the magnetization $\mathcal{M}$ as a function of $kT$ and compare to the analytic result. Does this agree with how you expect a heated magnet to behave?
8. Compute the fluctuations of the energy $U_2$ (12.14), and the specific heat $C$ (12.15). Make a graph of your simulated specific heat compared to the analytic result (12.7).

### 12.3.4 Beyond Nearest Neighbors and 1D (Exploration)

- Extend the model so that the spin–spin interaction (12.4) extends to next-nearest neighbors as well as nearest neighbors. For the ferromagnetic case, this should lead to more binding and less fluctuation because we have increased the couplings among spins and thus increased the thermal inertia.
- Extend the model so that the ferromagnetic spin–spin interaction (12.4) extends to nearest neighbors in two, and for the truly ambitious, then three dimensions (the code `Ising3D.java` is available for instructors). Continue using periodic boundary conditions and keep the number of particles small, at least to start [17].

1. Form a square lattice and place $\sqrt{N}$ spins on each side.
2. Examine the mean energy and magnetization as the system equilibrates.
3. Is the temperature dependence of the average energy qualitatively different from the 1D model?
4. Identify domains in the printout of spin configurations for small $N$.
5. Once your system appears to be behaving properly, calculate the heat capacity and magnetization of the 2D Ising model with the same technique used for the 1D model. Use a total number of particles $100 \leq N \leq 2000$.
6. Look for a phase transition from an ordered to unordered configuration by examining the heat capacity and magnetization as a function of temperature. The heat should diverge at the phase transition (you may get only a peak), while the magnetization should vanish above the Curie temperature (Figs. 12.4–12.7).

**Listing 12.1:** `Ising.java` implements the Metropolis algorithm for a 1D Ising chain.

```
// Ising.java: 1D Ising model with Metropolis algorithm

import java.io.*;                              // Location of PrintWriter
import java.util.*;                                 // Location of Random

public class Ising {
  public static int N = 1000;                                // Number of atoms
  public static double B = 1.;                               // Magnetic field
  public static double mu = .33;                      // Magnetic moment of atom
  public static double J = .20;                              // Exchange energy
  public static double k = 1.;                              // Boltzman constant
  public static double T = 100000000.;

  public static void main(String[] argv)
                        throws IOException, FileNotFoundException {

    Random randnum = new Random(500767);          // Seed random generator
    PrintWriter q  = new
              PrintWriter(new FileOutputStream ("ising.dat"), false);

    int i, j, M = 5000;                                // Number of spin flips
    double[] state = new double[N];
    double[] test = state;
    double ES = energy(state), p, ET;                  // State, test's energy
                                                       // Set initial state
    for ( i=0 ;  i < N ;  i++ ) state[i] = -1.;
                                                     // Change state and test
    for ( j=1 ;  j <= M ;  j++ ) {
      test = state;
      i = (int)(randnum.nextDouble()*(double)N);
                                                              // Flip random atom
      test[i] *= -1.;
      ET = energy(test);
      p = Math.exp((ES-ET)/(k*T));
                                                              // Test trial state
      if (p >= randnum.nextDouble())  {
        state = test;
        ES = ET;
      }
      q.println(ES);                                  // Output energy to file
    }
    q.close();
  }

  public static double energy (double[] S)  {
                                                        // Method calc energy
    double FirstTerm = 0., SecondTerm = 0. ;
    int i;                                                     // Sum of energy

    for ( i=0 ;  i <= (N-2) ;  i++ )  FirstTerm += S[i]*S[i + 1];
    FirstTerm *= -J;
    for ( i=0 ;  i <= (N-1) ;  i++ )  SecondTerm += S[i];
    SecondTerm *= -B*mu;
    return (FirstTerm + SecondTerm);
}  }
```