simplicity, choose the impurity to be five spins in the shape of a + sign. What is the effect of the impurity on the lifetime of the metastable state? What is the probability of droplet growth on and off the impurity as a function of quench depth B?

(e) The questions raised in parts (b)–(d) become even more interesting when the interaction between the spins extends beyond nearest neighbors. Assume that a given spin interacts with all spins that are within a distance R with an interaction strength of $4J/q$, where q is the number of spins within the interaction range R. (Note that $q = 4$ for nearest neighbor interactions on the square lattice.) A good choice is $R = 10$, although your preliminary simulations should be for smaller R. How does the value of T_c change as R is increased? ■

Project 15.36 The n-fold way: Simulations at low temperature

Monte Carlo simulations become very inefficient at low temperatures because almost all trial configurations will be rejected. For example, consider an Ising model for which all spins are up, but a small magnetic field is applied in the negative direction. The equilibrium state will have most spins pointing down. Nevertheless, if the magnetic field is small and the temperature is low enough, equilibrium will take a very long time to occur.

What we need is a more efficient way of sampling configurations if the acceptance probability is low. The *n-fold way* algorithm is one such method. The idea is to accept more low probability configurations but to weight them appropriately. If we use the usual Metropolis rule, then the probability of flipping the ith spin is

$$p_i = \min[1, e^{-\Delta E/kT}]. \tag{15.83}$$

One limitation of the Metropolis algorithm is that it becomes very inefficeint if the probabilities p_i are very small. If we sum over all the spins, then we can define the total weight

$$Q = \sum_i p_i. \tag{15.84}$$

The idea is to choose a spin to flip (with probability one) by computing a random number r_Q between 0 and Q and finding spin i that satisfies the condition

$$\sum_{k=0}^{i-1} p_k \le r_Q < \sum_{k=0}^{i} p_k. \tag{15.85}$$

There are two more ingredients we need to make this algorithm practical. We need to determine how long a configuration would remain unchanged if we had used the Metropolis algorithm. Also, the algorithm would be very inefficient because on average the computation of which spin to flip from (15.85) would take $O(N)$ computations. This second problem can be easily overcome by realizing that there are only a few possible values of p_i. For example, for the Ising model on a square lattice in a magnetic field, there are only $n = 10$ possible values of p_i. Thus, instead of (15.85), we have

$$\sum_{\alpha=0}^{i-1} n_\alpha p_\alpha \le r_Q < \sum_{\alpha=0}^{i} n_\alpha p_\alpha, \tag{15.86}$$

where α labels one of the n possible values of p_i or classes, and n_α is the number of spins in class α. Hence, instead of $O(N)$ calculations, we need to perform only $O(n)$ calculations. Once we know which class we have chosen, we can randomly flip one of the spins in that class.

Next we need to determine the time spent in a configuration. The probability in one Metropolis Monte Carlo step of choosing a spin at random is $1/N$, and the probability of actually flipping that spin is p_i, which is given by (15.83). Thus, the probability of flipping any spin is

$$\frac{1}{N}\sum_{i=0}^{N-1} p_i = \frac{1}{N}\sum_{\alpha=0}^{n-1} n_\alpha p_\alpha = \frac{Q}{N}. \tag{15.87}$$

The probability of not flipping any spin is $q \equiv 1 - Q/N$, and the probability of not flipping after s steps is q^s. Thus, if we generate a random number r between 0 and 1, the time s in Monte Carlo steps per spin to remain in the current configuration will be determined by solving

$$q^{s-1} \le r < q^s. \tag{15.88}$$

If $Q/N \ll 1$, then both sides of (15.88) are approximately equal, and we can approximate s by

$$s \approx \frac{\ln r}{\ln q} = \frac{\ln r}{\ln(1 - Q/N)} \approx -\frac{N}{Q}\ln r. \tag{15.89}$$

That is, we would have to wait s Monte Carlo steps per spin on the average before we would flip a spin using the Metropolis algorithm. Note that the random number r in (15.88) and (15.89) should not be confused with the random number r_Q in (15.86).

The n-fold algorithm can be summarized by the following steps:

(i) Start with an initial configuration and determine the class to which each spin belongs. Store all the possible values of p_i in an array. Compute Q. Store in an array the number of spins in class α, n_α.
(ii) Determine s from (15.89). Accumulate any averages, such as the energy and magnetization weighted by s. Also, accumulate the total time `tTotal += s`.
(iii) Choose a class of spin using (15.86) and randomly choose which spin in the chosen class to flip.
(iv) Update the classes of the chosen spin and its four neighbors.
(v) Repeat steps (ii)–(iv).

To conveniently carry out step (iv), set up the following arrays: `spinClass[i]` returns the class of the ith spin, `spinInClass[k][alpha]` returns the kth spin in class α, and `spinIndex[i]` returns the value of k for the ith spin to use in the array `spinInClass[k][alpha]`. If we define the local field of a spin by the sum of the fields of its four neighbors, then this local field can take on the values $\{-4, -2, 0, 2, 4\}$. The ten classes correspond to these five local field values and the center spin equal to -1 plus these five local field values and the center spin equal to $+1$. If we order these ten classes from