Exercises

8.13 **Hysteresis and avalanches.**[1] [2] (Complexity, Computation) ④

A piece of magnetic material exposed to an increasing external field $H(t)$ (Fig. 8.14) will magnetize (Fig. 8.15) in a series of sharp jumps, or *avalanches* (Fig. 8.16). These avalanches arise as magnetic domain walls in the material are pushed by the external field through a rugged potential energy landscape due to irregularities and impurities in the magnet. The magnetic signal resulting from these random avalanches is called *Barkhausen noise*.

We model this system with a non-equilibrium lattice model, the *random field Ising model*. The Hamiltonian or energy function for our system is

$$\mathcal{H} = -\sum_{\langle i,j\rangle} J s_i s_j - \sum_i \big(H(t) + h_i\big)\, s_i, \qquad (8.27)$$

where the spins $s_i = \pm 1$ lie on a square or cubic lattice with periodic boundary conditions. The coupling and the external field H are as in the traditional Ising model (Section 8.1). The disorder in the magnet is incorporated using the *random field* h_i, which is independently chosen at each lattice site from a Gaussian probability distribution of standard deviation R:

$$P(h) = \frac{1}{\sqrt{2\pi}R} \mathrm{e}^{-h^2/2R^2}. \qquad (8.28)$$

We are not interested in thermal equilibrium; there would be no hysteresis! We take the opposite extreme; we set the temperature to zero. We start with all spins pointing down, and adiabatically (infinitely slowly) increase $H(t)$ from $-\infty$ to ∞.

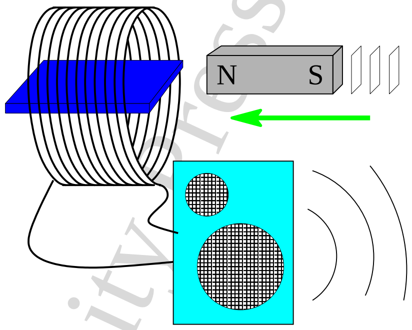

Fig. 8.14 Barkhausen noise experiment. By increasing an external magnetic field $H(t)$ (bar magnet approaching), the magnetic domains in a slab of iron flip over to align with the external field. The resulting magnetic field jumps can be turned into an electrical signal with an inductive coil, and then listened to with an ordinary loudspeaker. Barkhausen noise from our computer experiments can be heard on the Internet [68].

Our rules for evolving the spin configuration are simple: each spin flips over when doing so would decrease the energy. This occurs at site i when the local field at that site

$$J \sum_{j \text{ nbr to } i} s_j + h_i + H(t) \qquad (1)$$

changes from negative to positive. A spin can be pushed over in two ways. It can be triggered when one of its neighbors flips (by participating in a propagating avalanche) or it can be triggered by the slow increase of the external field (starting a new avalanche).

[1]From *Statistical Mechanics: Entropy, Order Parameters, and Complexity* by James P. Sethna, copyright Oxford University Press, 2007, page 182. A pdf of the text is available at pages.physics.cornell.edu/sethna/StatMech/ (select the picture of the text). Hyperlinks from this exercise into the text will work if the latter PDF is downloaded into the same directory/folder as this PDF.
[2]This exercise is largely drawn from [69]. It and the associated software were developed in collaboration with Christopher Myers.

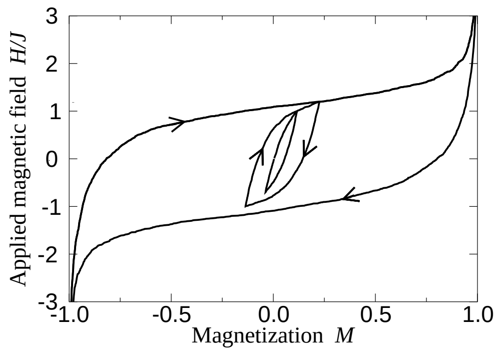

Fig. 8.15 Hysteresis loop with subloops for our model. As the external field is raised and lowered (vertical), the magnetization lags behind—this is called *hysteresis*. The magnetization curves here look macroscopically smooth.

We will provide hints files and graphics routines for different languages and systems on the computer exercises portion of the book web site [129].

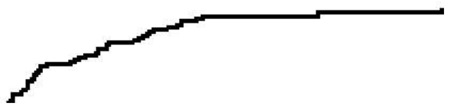

Fig. 8.16 Tiny jumps: Barkhausen noise. Blowing up a small portion of Fig. 8.15, we see that the magnetization is growing in a series of sharp jumps, or avalanches.

(a) *Set up lattices* `s[m][n]` *and* `h[m][n]` *on the computer. (If you do three dimensions, add an extra index to the arrays.) Fill the former with down-spins* (-1) *and the latter with random fields (real numbers chosen from the distribution 8.28). Write a routine* `FlipSpin` *for the lattice, which given i and j flips the spin from $s=-1$ to $s=+1$ (complaining if it is already flipped). Write a routine* `NeighborsUp` *which calculates the number of up-neighbors for the spin (implementing the periodic boundary conditions).*

On the computer, changing the external field infinitely slowly is easy. To start a new avalanche (or the first avalanche), one searches for the unflipped spin that is next to flip, jumps the field H to just enough to flip it, and propagates the avalanche, as follows.

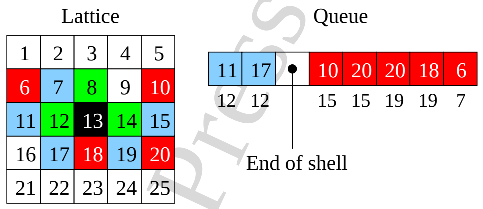

Fig. 8.17 Avalanche propagation in the hysteresis model. Left: a propagating avalanche. Spin 13 triggered the avalanche. It triggered the first shell of spins 14, 8, and 12, which then triggered the second shell 15, 19, 7, 11, and 17, and finally the third shell 10, 20, 18, and 6. Right: the first-in–first-out queue, part way through flipping the second shell. (The numbers underneath are the triggering spins for the spins on the queue, for your convenience.) The spin at the left of this queue is next to flip. Notice that spin 20 has been placed on the queue twice (two neighbors in the previous shell). By placing a marker at the end of each shell in the queue, we can measure the number of spins flipping per unit 'time' during an avalanche (Fig. 8.18).

Propagating an avalanche

(1) Find the triggering spin i for the next avalanche, which is the unflipped site with the largest internal field $J\sum_{j \text{ nbr to } i} s_j + h_i$ from its random field and neighbors.

(2) Increment the external field H to minus this internal field, and push the spin onto a first-in–first-out queue (Fig. 8.17, right).

(3) Pop the top spin off the queue.

(4) If the spin has not been flipped,[3] flip it and push all unflipped neighbors with positive local fields onto the queue.

(5) While there are spins on the queue, repeat from step (3).

(6) Repeat from step (1) until all the spins are flipped.

(b) *Write a routine* `BruteForceNextAvalanche` *for step (1), which checks the local fields of all of the unflipped spins, and returns the location of the next to flip.*

[3]You need to check if the spin is flipped again after popping it off the queue; spins can be put onto the queue more than once during an avalanche (Fig. 8.17).

(c) *Write a routine* `PropagateAvalanche` *that propagates an avalanche given the triggering spin, steps (3)–(5), coloring the spins in the display that are flipped. Run a* 300×300 *system at* $R = 1.4$, 0.9, *and* 0.7 *(or a* 50^3 *system at* $R = 4$, $R = 2.16$, *and* $R = 2$*) and display the avalanches.* If you have a fast machine, you can run a larger size system, but do not overdo it; the sorted list algorithm below will dramatically speed up the simulation.

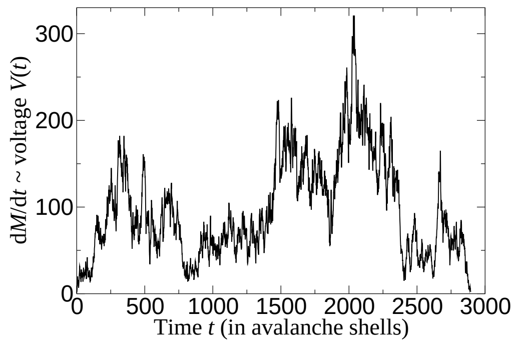

Fig. 8.18 Avalanche time series. Number of domains flipped per time step for the avalanche shown in Fig. 12.5. Notice how the avalanche almost stops several times; if the forcing were slightly smaller compared to the disorder, the avalanche would have separated into smaller ones. The fact that the disorder is just small enough to keep the avalanche growing is the criterion for the phase transition, and the cause of the self-similarity. At the critical point, a partial avalanche of size S will on average trigger another one of size S.

There are lots of properties that one might wish to measure about this system: avalanche sizes, avalanche correlation functions, hysteresis loop shapes, average pulse shapes during avalanches, . . . It can get ugly if you put all of these measurements inside the inner loop of your code. Instead, we suggest that you try the *subject–observer* design pattern: each time a spin is flipped, and each time an avalanche is finished, the subject (our simulation) notifies the list of observers.

(d) *Build a* `MagnetizationObserver`*, which stores an internal magnetization starting at* $-N$*, adding two to it whenever it is notified. Build an* `AvalancheSizeObserver`*, which keeps track of the growing size of the current avalanche after each spin flip, and adds the final size to a histogram of all previous avalanche sizes when the avalanche ends. Set up* `NotifySpinFlip` *and* `NotifyAvalancheEnd` *routines for your simulation, and add the two observers appropriately. Plot the magnetization curve* $M(H)$ *and the avalanche size distribution histogram* $D(S)$ *for the three systems you ran for part (c).*