Editors: Harvey Gould, hgould@physics.clarku.edu
Jan Tobochnik, jant@kzoo.edu

HYSTERESIS, AVALANCHES, AND NOISE

Matthew C. Kuntz, Olga Perković, Karin A. Dahmen, Bruce W. Roberts, and James P. Sethna

AS COMPUTERS INCREASE IN SPEED AND MEMORY, SCIENTISTS ARE INEVITABLY LED TO SIMULATE MORE COMPLEX SYSTEMS OVER LARGER TIME AND LENGTH SCALES. ALTHOUGH A SIMPLE, STRAIGHTFORWARD ALGORITHM IS OFTEN THE MOST EFFICIENT FOR SMALL systems, especially when the time needed to implement the algorithm is included, the scaling of time and memory with system size becomes crucial for larger simulations.

In our studies of hysteresis and avalanches in the *zero-temperature random-field Ising model*, a simple model of magnetism, we often have had to do very large simulations. Previous simulations were usually limited to relatively small systems (up to 900^2 and 128^3),[1,2] although there have been exceptions.[3] In our simulations, we have found that larger systems (up to a billion spins) are crucial to extracting accurate values of the critical exponents and understanding important qualitative features of the physics.

In this column, we will show three algorithms for simulating these large systems. The first uses the *brute-force method*, which is the standard method for avalanche-propagation problems. This algorithm is simple but inefficient. We have developed two efficient and relatively straightforward algorithms that provide better results. The *sorted-list algorithm* decreases execution time, but requires considerable storage. The *bits algorithm* has an execution time that is similar to that of the sorted-list algorithm, but it requires far less storage.

The zero-temperature random-field Ising model

This model is defined by the energy function

$$\mathcal{H} = -\sum_{<i,j>} J s_i s_j - \sum_i \left[H(t) + h_i\right] s_i \,, \tag{1}$$

where the spins $s_i = \pm 1$ sit on a D-dimensional hypercubic lattice with periodic boundary conditions. The spins interact ferromagnetically with their z nearest neighbors with strength J, and experience a uniform external field $H(t)$ and a random local field h_i. We choose units such that $J = 1$. The random local field is distributed according to the Gaussian distribution $\rho(h)$ of width R:

$$\rho(h) = \frac{1}{\sqrt{2\pi}R} e^{-h^2/2R^2} \,. \tag{2}$$

The external field is increased arbitrarily slowly from $-\infty$ to $+\infty$.

The model's dynamics includes no thermal fluctuations: each spin flips deterministically when it can gain energy by doing so. That is, it flips when its local field

$$h_i^{\text{eff}} = J\sum_j s_j + h_i + H \tag{3}$$

changes sign. This change can occur in two ways: a spin can be triggered when one of its neighbors flips (by participating in an avalanche) or when the external field increases (starting a new avalanche).

H. Ji and Mark Robbins introduced the zero-temperature random-field Ising model to study fluid invasion in porous media and front propagation in disordered systems.[3] We have used this model[4] in a different way to model noise in hysteresis loops in disordered materials. In particular, we wish to understand Barkhausen noise in magnetic materials with quenched disorder.[5] Researchers have found that when an external field is gradually applied, many materials magnetize not continuously, but in a noisy way, with jumps (avalanches) of all sizes. (You can hear the noise by wrapping the magnetizing material in a coil of wire and amplifying the signal into a speaker. The signal makes a crackling noise when a permanent magnet is brought close, quite similar to the crackling noises heard in fires, crisped rice cereals, and crumpled paper. See, for example, *http://SimScience.org/crackling/*.) In the steepest part of the hysteresis loop, these avalanches have a power-law distribution of sizes with an exponent $\tau \approx 1.5$ and a power-law distribution of avalanche times with an exponent $\alpha \approx 2$. Power laws are also found in the power spectrum.

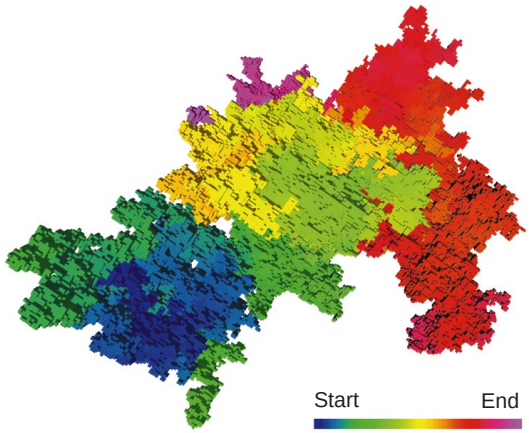

Figure 1. A 3D view from one side of an avalanche of 282,785 spins in a 200 × 200 × 200 system at $R = 2.3$ (within 6% of the critical disorder R_c). The avalanche generally grew from left to right. It has many branches and holes; the large avalanches in 3D probably have a fractal dimension a little less than three. On the right, several dark red spots poke through the middle of the light green area. The green area stopped growing, but other parts of the avalanche later filled in the holes.

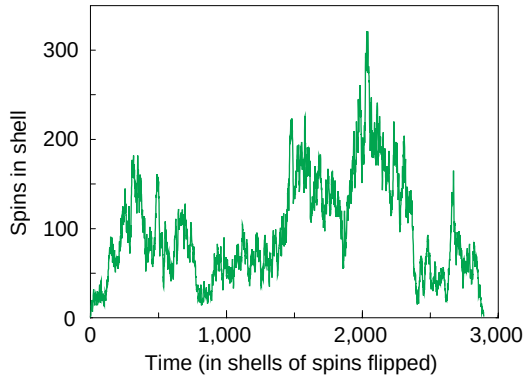

Figure 2. A time series showing the number of spins that flipped in each shell of the avalanche in Figure 1.[6] The avalanche is a series of bursts: near the critical point, the avalanche is always on the verge of halting, so it proceeds in fits and starts.

This model is interesting because, as in the disordered magnetic materials it attempts to model, the avalanches can have a broad range of sizes. If all the avalanches were small, understanding them would be straightforward and not very interesting. Indeed, at large disorder R, the chance that a spin that has just flipped will trigger one of its z neighbors scales roughly as zJ/R. If this quantity is smaller than unity (large disorder), all avalanches will be small: the noise will be a series of small pops of about the same size. This behavior is uninteresting not because it is simple, but because the behavior depends strongly on the details of the model at short distances, where the model is at best a caricature of a real material.

It is also easy to understand the system in the small disorder regime $zJ/R >> 1$, where almost all the spins flip over in one *infinite avalanche*. In many systems (for example, fracture and first-order phase transitions), a single nucleation event leads to the release of the stored energy in a single catastrophic event.

We focus on the crossover between these two limiting cases, where the system exhibits crackling noise with avalanches of all sizes. For a particular value of the disorder $R = R_c$, a spin that has just flipped will on average flip exactly one neighbor as the external field $H(t)$ increases to a particular value H_c. The avalanches at R_c, H_c (the critical point) are finely balanced between stopping and growing forever. They advance in fits and starts (see Figures 1 and 2) and come in all sizes (see Figures 3 and 4), with a probability that decreases as a power law of the number of spins in the avalanche.

At H_c, the distribution of avalanche sizes decays with an exponent of $\tau \approx 1.6$ (quite close to the experimental results), and integrated over all H, the distribution decays with an exponent $\tilde{\tau} \approx 2$. Below the critical disorder R_c, an infinite avalanche flips a nonzero fraction of the spins in the system even as the system size goes to infinity. Very large avalanches occur even for disorders far above the critical disorder. In 3D, two decades of power law still scale 50% above the critical point. However, the convergence to the expected asymptotic power law is very slow (see Figure 3).

This behavior means that we see critical scaling even if we do not fine-tune R to R_c, but we must use very large systems to get close enough to R_c to obtain a convincing power law. In practice, we need simulations of approximately a billion spins to understand the physics in 3D.[7–9] Two dimensions remains a challenge because the proper scaling is not clear even for $30{,}000^2$ spins.[7–10]

The basics: brute force

This method is the easiest to implement and is competitive for system sizes up to approximately 10,000 spins. We store a spin direction and a random field for each site of the lattice. We proceed by measuring the magnetization at specific predetermined values of H. We start with magnetization $M = -N$ and a large negative field H_0 and then increment to H_1, check all spins in the lattice, and flip those spins in a positive local field. Then we check the neighbors of the flipped spins again to see if their local fields are now positive. This procedure continues until we have checked all the neighbors of flipped spins. We then repeat the procedure for a new field H_2, and so on. This approach gives the correct magnetization at the fields H_n: the order in which spins flip does not in-

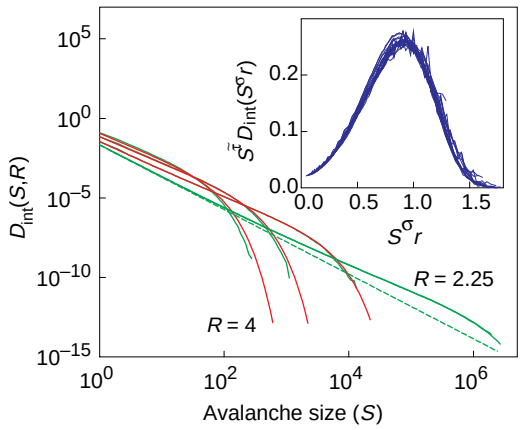

Figure 3. The distribution of avalanche sizes for different values of the disorder R in 3D. Some avalanches remain large (hundreds of spins) for R a factor of two above the critical value $R_c \sim 2.16$, where we expect a pure power law. The avalanches are enormous (millions of spins) when the system is still 4% away from the critical point, so we need large systems. The inset is a scaling collapse of the data: the thin lines in the main figure show the scaling prediction for the avalanche sizes stemming from the scaling collapse.[7–9] The scaling predictions already work well at R = 4. The pure asymptotic power-law behavior does not yet appear at R = 2.25, when six decades of scaling have occurred.

Figure 4. A 30,000 × 30,000 simulation with disorder R = 0.65, where each pixel represents a 30 × 30 square, and each avalanche is a different color. There are avalanches of all sizes, with many smaller avalanches and fewer large ones.

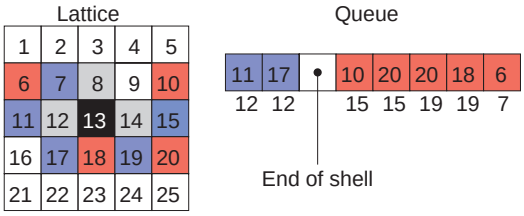

Figure 5. Using a queue to propagate an avalanche. The colored spins are spins that either have flipped or will flip in the current avalanche. Spin 13 triggered the avalanche; then, the light gray spins (14, 8, 12) were put on the queue as the first shell. As they flipped, the second shell—the blue spins (15, 19, 7, 11, 17)—were put on the queue. As the first blue spins (15, 19, 7) flipped, the dark red spins (10, 20, 20, 18, 6) were added to the queue as the start of the third shell. The next spin to flip is on the queue's left. When this spin flips, its neighbors will be checked, and the neighbors that are ready to flip will be added to the queue's right. The small numbers below the spins in the queue indicate which neighbor caused the spin to be put on the queue. Different neighbors can cause a spin (such as spin 20) to be put on the queue more than once. We must be careful to flip the spin only once.

fluence the final state.[4,11,12] However, unless the increments in H are very small, several avalanches might occur in a given increment, and all information about single avalanches (such as histograms of avalanche sizes) will be distorted.

The time for the brute-force method scales as $O(NXT)$, where X is the number of fields H_n at which the magnetization is measured, and T is the average time needed to check the neighbors of the flipped spins, measured in units of shells of neighbors.

A variation on this approach propagates one avalanche at a time (see Figure 5):

1. Find the triggering spin for the next avalanche by checking through the lattice for the unflipped site with the largest internal field
$$h_i^{\text{int}} = J\sum_j s_j + h_i^{\text{eff}} - H \ .$$
2. Increment the external field so that it is just large enough to flip the site, and push the spin onto a first-in, first-out (FIFO) queue (see Figure 5, right).
3. Pop the top spin off the queue.
4. If the spin has not been flipped, flip it and push all unflipped neighbors with positive local fields h_{eff} onto the queue. You will need to check if the spin is flipped after popping it off the queue as well as before installing it! Spins can be put onto the queue several times.
5. While there are spins on the queue, repeat from Step 3.
6. Repeat from Step 1 until all the spins are flipped.

This method is related to the propagation of cluster flips in the Wolff algorithm.[13] Using a queue instead of recursion has two advantages. First, recursion is slower and more

memory-intensive, because each recursive call must push all local variables and all registers onto the system stack (which usually has a preallocated limit). If we use our own queue, we need only to push the next spin's coordinate on the queue each time, and we can make the queue as large as necessary.

Second, to produce a natural spin-flip order, we want to flip all spins that are ready to flip at a given time before we flip the spins that they cause to flip. If we put spins that are ready to flip on a FIFO queue, we correctly achieve this order. This procedure corresponds to doing a breadth-first search. Recursion, which is the same as putting the spins on a last-in, first-out (LIFO) stack, would explore all possible consequences of flipping the first neighbor it looks at before it considers the second neighbor. This depth-first search produces an unnatural spin-flip order (although the final state after the avalanche is unchanged[4,11,12]). The dynamics during the avalanche of Figure 2 assumes one shell of spins flipped during each time slice, which is easy to determine by placing markers on a FIFO spin queue, as Figure 5 shows. Each time the marker is popped off the queue, a new shell is started and the marker is put back on the end of the queue.

Doing the brute-force algorithm one avalanche at a time is very inefficient except at very low disorders. Sweeping through the entire lattice for each avalanche takes $O(N)$ time per avalanche. Because there are $O(N)$ avalanches, the total running time scales as $O(N^2)$. A hybrid approach, finite steps in field followed by internal propagation of avalanches, could be quite efficient if the magnetization at those fields is the sole quantity of interest. A brute-force method is probably necessary when simulating systems with long-range interactions.[14]

Time efficiency: sorted lists

The brute-force method is very inefficient at locating the next avalanche's origin, which immediately leads us to think of storing the several largest local fields in each sweep. The logical conclusion of this thinking is to store a list of all the spins in the system, sorted according to their random fields.

Unfortunately, life is complicated by the fact that spins experience not only their local random fields but also fields from their neighbors. To find the next avalanche's origin, we use the sorted-list algorithm:

1. Define an array nextPossible[$n\uparrow$], $n\uparrow = 0, 1, \ldots z$, which points to the location in the sorted list of the next spin that would flip if it had $n\uparrow$ neighbors. Initially, all the elements of nextPossible[$n\uparrow$] point to the spin with the largest local random field, h_i.
2. From the $z + 1$ spins pointed to by nextPossible, choose the one with the largest internal field,
$$h_i^{\text{int}}\left[n_\uparrow\right] = n_\uparrow - n_\downarrow + h_i = 2n_\uparrow - z + h_i \ .$$
(Do not check values of $n\uparrow$ for which the pointer has fallen off the end of the list. You will want a variable stopNUp.)
3. Move the pointer nextPossible[$n\uparrow$] to the next spin on the sorted list. If you have fallen off the end of the list, decrement stopNUp. (It is obvious that you can move to the next spin if the current spin is flipped or if it starts a new avalanche. If it has too few neighbors up, you can also pass it by. Why?)
4. If the spin with the largest $h_i^{\text{int}}\left[n_\uparrow\right]$ has $n\uparrow$ up neighbors, flip it, increment the field, and start an avalanche. Otherwise, go back to step 2.

Figure 6 shows an example of the sorted list and the pointers from nextPossible.

The spins can be sorted in $O(N\log N)$ time. Storage with this algorithm is $N \times$ sizeof(int) for the sorted array (if we reduce the D-dimensional coordinates to one number), and $N \times$ (sizeof(spin) + sizeof(double)) for the lattice. Various other compromises between execution speed and storage are possible, but all leave the running time $O(N\log N)$. The sorted-list algorithm is fast: the largest system sizes we can store on a reasonable workstation execute $1{,}000^2$ and 100^3 spins in a few seconds. It is the method of choice for these small systems or when you are interested in the behavior for nonmonotonically increasing fields.

Space efficiency: one bit per spin

The combination of the rapid execution of the sorted-list algorithm and large finite-size effects led us to develop the bits algorithm, which is optimized for memory efficiency. The key is to recognize that we need never generate the random fields! In invasion percolation[15] (and in the interface problem[3] analogous to ours), the random fields are generated only on sites along the growing cluster's boundary. For our problem, we extend this idea: for each change in a spin's local field given by Equation 3, we generate only the probability that it will flip. Storing the random fields is unnecessary because the external field, the configuration of the spin's neighbors, and the knowledge that the spin has not yet flipped give us all the information we need to determine the probability that the spin will flip. The only quantity that we must store for each site of the lattice is whether the spin is up or down. Thus, we can store each site of the lattice as a computer bit, saving large amounts of memory.

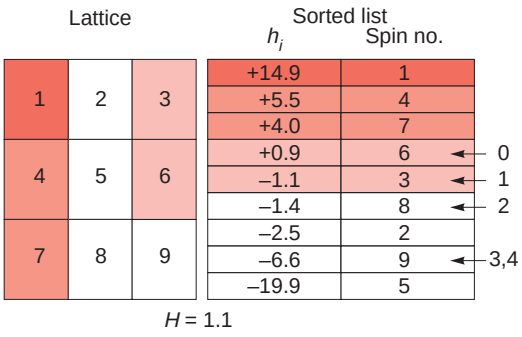

Figure 6. Using a sorted list to find the next spin in the avalanche. The colors indicate spins that have already flipped. In the sorted list, the arrows to the right indicate the nextPossible[$n\uparrow$] pointers—the first spin that would not flip with $n\uparrow$ neighbors up. The spins pointed to are the next avalanche's possible starting locations. Some pointers point to spins that have already flipped, meaning that these spins have more than $n\uparrow$ neighbors up. In a larger system, the unflipped spins will not all be contiguous in the list.

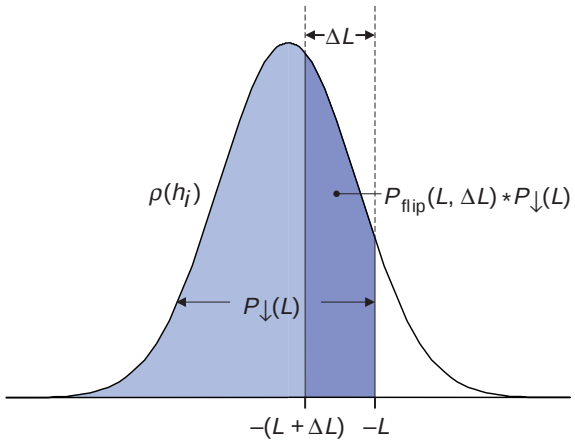

Figure 7. The probability that a spin will not have flipped by the time its local field reaches L is the probability that the random field is less than $-L$. The shaded area of the Gaussian represents this probability. The area of the darker region divided by the area of the shaded region represents the probability that the spin will flip before the field reaches $-(L + \Delta L)$.

For a monotonically increasing external field, the conditional probability that a spin flips before its nonrandom local field, $H_{nr} \equiv H + (2n\uparrow - z)$, reaches $H_{nr} + \Delta H_{nr}$, given that it has not flipped by H_{nr}, is

$$P_{\text{flip}}\left(H_{nr}, \Delta H_{nr}\right) = \frac{\left[P_{\downarrow}\left(H_{nr}\right) - P_{\downarrow}\left(H_{nr} + \Delta H_{nr}\right)\right]}{P_{\downarrow}\left(H_{nr}\right)}, \qquad (4)$$

where $P_{\downarrow}(H_{nr})$ is the probability that a spin points down when the local field is H_{nr}. A spin with local field H_{nr} will point down if its random field h_i satisfies $h_i + H_{nr} \le 0$. This condition implies that the probability that a spin with $n\uparrow$ up neighbors points down is

$$P_{\downarrow}\left(n\uparrow, H\right) = \int_{-\infty}^{-H_{nr}\left(n\uparrow, H\right)} \rho(h)\,dh$$

$$= \frac{1}{2} + \frac{1}{2}\,\text{erf}\left(-H_{nr}\left(n\uparrow, H\right)\ \sqrt{2}R\right) \qquad (5)$$

$$= \frac{1}{2}\,\text{erfc}\left(H_{nr}\left(n\uparrow, H\right)\ \sqrt{2}R\right). \qquad (6)$$

(Writing $P_{\downarrow}$ in terms of the erfc function removed some problems with rounding at large negative fields H.) Figure 7 illustrates these probabilities.

Finding the next avalanche is subtle when the random fields are not stored: changing the external field H introduces a probability that any unflipped spin in the lattice might flip. Inspired by the continuous-time Monte-Carlo algorithm,[16,17] we keep track of $N_{n\uparrow}$, the number of down spins that have $n\uparrow$ up neighbors. Given the probabilities that spins with $n\uparrow$ up neighbors will flip, we calculate both the change in the external field ΔH needed to flip the next spin and the probability that the next spin to flip has $n\uparrow$ up neighbors. We then randomly choose $n\uparrow$ and randomly search through the lattice for a spin with $n\uparrow$ up neighbors. The time the search takes is the part of the algorithm that scales worst for large N. If $N_{n\uparrow}$ spins are left, this search will take $O(N/N_{n\uparrow})$ average time. Summing over $N_{n\uparrow}$ and $n\uparrow$ yields a bound of order $zN\log N$. In one of our programs, we use a tree structure to do this search more efficiently; this complication decreases the running time by 40% for a 500^2 system at $R = 1$.

How do we calculate ΔH? From Equation 4, the probability that a single spin with $n\uparrow$ neighbors up has not flipped in the range ΔH is $1 - P_{\text{flip}}(n\uparrow, H, \Delta H) = P_{\downarrow}(n\uparrow, H + \Delta H)/P_{\downarrow}(n\uparrow, H)$. The probability that no spin with $n\uparrow$ up neighbors has flipped in this range is

$$P_{n\uparrow}^{\text{none}} = \left[\frac{P_{\downarrow}\left(n\uparrow, H + \Delta H\right)}{P_{\downarrow}\left(n\uparrow, H\right)}\right]^{N_{n\uparrow}}. \qquad (7)$$

The probability that no spin has flipped between H and $H + \Delta H$ is

$$P^{\text{none}}\left(\Delta H\right) = \prod_{n\uparrow=0}^{z} P_{n\uparrow}^{\text{none}}. \qquad (8)$$

To find ΔH, we choose a random number r uniformly distributed between zero and one, and set ΔH so that $P^{\text{none}}(\Delta H) = r$.

Unfortunately, we cannot solve for ΔH analytically; we must find a numerical solution. To do this efficiently, we need a good initial guess. Analogous with nuclear decay, if spins flip with a constant rate Γ, we expect the probability that no spins have yet flipped to be $e^{-\Gamma \Delta H}$. For a first approximation of ΔH, we assume that the spin-flip rate Γ is a constant leading to

$$\Delta H_1 = -\frac{\log(r)}{\Gamma(H)}, \tag{9}$$

where $\Gamma(H)$ is given by

$$\begin{aligned}\Gamma(H) &= -\frac{d\log\left(P^{\text{none}}(\Delta H = 0)\right)}{d\Delta H} \\ &= -\sum_{n_\uparrow=0}^{z} N_{n_\uparrow} \frac{d\log\left(P_\downarrow(n_\uparrow, H+\Delta H)\right)}{d\Delta H} \\ &= \sum_{n_\uparrow=0}^{z} N_{n_\uparrow} \frac{\rho\left(H_{\text{nr}}(n_\uparrow, H)\right)}{P_\downarrow(n_\uparrow, H)} \equiv \sum_{n_\uparrow=0}^{z} \Gamma(n_\uparrow, H).\end{aligned} \tag{10}$$

We can make a better second guess by looking at the error in our first guess. If the error is $\Delta r = P^{\text{none}}(\Delta H) - r$, we can make an improved second guess for ΔH by aiming for $r - \Delta r$:

$$\Delta H_2 = -\frac{\log(r - \Delta r)}{\Gamma(H)}. \tag{11}$$

We can then use these two guesses as input into a root-finding routine. Although these guesses are usually very good for small $|H|$ and lead to quick solutions, they can be very bad for large $|H|$. If the guesses for ΔH are very large, choosing two arbitrary guesses might be better. In our code, if $\Delta H_1 > 20$, we use $\Delta H_1 = 0$ and $\Delta H_2 = 20$ for the two guesses.

Our algorithm for finding the next avalanche can be summarized in the following steps:

1. Choose a random number r uniformly distributed between zero and one.
2. Precalculate the values of $P_\downarrow(n_\uparrow, H)$ using Equation 6. These values will be used repeatedly in solving for ΔH.
3. Calculate guesses for ΔH using Equations 9 and 11, and use them as input to a root-finding routine to find the exact solution for ΔH.
4. Increment H by ΔH.
5. Calculate the array probFlip[$n_\uparrow$] for use in the remainder of the avalanche, where probFlip[$n_\uparrow$] is the probability at the current field H that a spin will flip when its number of up neighbors changes from $n_\uparrow$ to $n_\uparrow + 1$ (see Equation 4).
6. Calculate the rates for flipping spins for each $n_\uparrow$ at the current field $H = H_{\text{old}} + \Delta H$:

$$\Gamma(n_\uparrow, H) = N_{n_\uparrow} \rho\left(H_{nr}(n_\uparrow, H)\right) / P_\downarrow(n_\uparrow, H), \tag{12}$$

   and calculate the total rate

$$\Gamma(H) = \sum_{n_\uparrow=0}^{z} \Gamma(n_\uparrow, H).$$

7. Choose a random number uniformly distributed between zero and Γ, and use it to select $n_\uparrow$.
8. Search randomly in the lattice for an unflipped spin with $n_\uparrow$ up neighbors.
9. Start the avalanche at that spin. During an avalanche, the algorithm is essentially the same as the brute-force algorithm.
10. Push the first spin onto the queue.
11. Pop the top spin off the queue.
12. If the spin is unflipped, flip it, find $n_\uparrow$, and decrease $N_{n_\uparrow}$ by one. Otherwise, skip to step 14.
13. Look at all unflipped neighbors. For each unflipped neighbor, find the current number of up neighbors, $n_\uparrow$; decrease $N_{n_\uparrow - 1}$ by one, and increase $N_{n_\uparrow}$ by one. Push the spin on the queue with probability probFlip[$n_\uparrow - 1$], as calculated in Step 5.
14. If any spins are left in the queue, repeat from Step 11.
15. If any spins are unflipped, repeat from Step 1.

This algorithm is about half as fast in practice as the sorted-list algorithm, but faster than we expected. The overhead for solving for ΔH is presumably compensated by the time saved not shifting data in and out of cache. Systems of 10^9 spins take a few days of CPU time on a reasonable workstation; $30{,}000^2$ systems take less than 15 hours on a 266-MHz Pentium II.

Calculating histograms and correlations

Characterizing the critical properties of our model requires several functions. The simplest function is the magnetization, M, as a function of the external field, H. We also calculate distributions of avalanche sizes (see the inset in Figure 3) and correlation functions. Some care must be taken to make sure

Suggestions for further study

C++ source code implementing all three algorithms is available at *www.lassp.cornell.edu/sethna/hysteresis/code/*. Each algorithm is in a separate, well-documented class. There are classes for detecting spanning avalanches, measuring avalanche size distributions, and measuring correlation functions. There are also both a Microsoft Windows and a command-line interface to the code. The command-line interface should be portable to any computer with a C++ compiler and an implementation of the Standard Template Libraries. We have also compiled executables for Windows 95/NT, Linux, and several other flavors of Unix. The running times in Figure 8 in the main article all come from the code compiled under Linux.

Phase transitions in the loop's shape

Download our program from our Web site. Also download DynamicLattice and xmgr if you are using Unix. Under Windows, press OK in the opening dialog box to run a simulation with the default parameters. Under Unix, type `run` at the >>> prompt. (More detailed instructions are on our Web site.) Try the other two algorithms. For smaller systems, brute force works acceptably, but for $L = 500$ it is rather slow.

You should see an animation of the avalanches as the external field is ramped upward and the spins flip. After the simulation ends, you should obtain a graph of $M(H)$, the avalanche size distribution $D(S)$, and the correlation function $G(x, R)$. The $M(H)$ curve shows the bottom half of the hysteresis loop: it should consist of many jumps of various sizes. The top half is pretty much the same shape, $-M(-H)$, but the details of the jumps are different.

The avalanche size distribution, $D(S)$, measures the number of jumps as a function of size, S. It should look like a fairly good power law and be a straight line on a log-log plot.

A log-log plot of the correlation function looks much less linear. At small distances, the correlation function decreases as a power law, but the power-law behavior bends over after only a decade or so. This behavior is a symptom of R not being quite at the critical disorder, R_c.

1. According to our scaling theory,[1–4] near the critical point, $D(S,R_c) \sim S^{-\tilde{\tau}}$. What is your best estimate for the exponent $\tilde{\tau}$ at the default value, $R = 1.0$?
2. Do a simulation at a smaller value of disorder, say $R = 0.8$. Does the behavior of $G(x, R)$ remain a power law over larger values of x?
3. Do a simulation with $D = 3$, $L = 50$, and $R = 2.5$. What is $\tilde{\tau}$ in 3D? (To obtain better data, do several runs and use the averaging option.) Consider $R = 2.1$. Does the $M(H)$ curve's shape look qualitatively different? We believe that the power-law distributions occur for $R_c \sim 2.16$. At this value, the hysteresis loop first develops a macroscopic jump (the infinite avalanche). You obtain power laws and scaling at the phase transition in the shape of the hysteresis loop, between smooth loops and ones with an infinite avalanche.

Space, time, and bits

We have a local, somewhat older supercomputer with 4 Gbytes of RAM. How large a system could we run in this memory using the three algorithms we discussed? Ignore all the memory requirements except those that scale linearly with the number of spins, N; the rest are negligible. Assuming the time spent in bits continues to grow as $N \log N$ after the last data point in Figure 8, how long will the largest possible bits simulation take to complete?

Programming

To do the following problems, you'll need our program's source code and a C++ compiler that supports the Standard Template Libraries. The source code, details on how to work with it, and links to compilers supporting these libraries are on our Web page.

Run the simulation described in the first problem to test

that the calculation of these functions does not dramatically increase the simulation's runtime or memory requirements.

When doing calculations with a billion spins, we cannot output any quantity that scales linearly with the system size. Instead of computing $H(M)$ at each avalanche (approximately 1 Gbyte of data, which would rapidly fill our disk), we are forced to compute $H(M_n)$ at prechosen points.

The characteristic feature of the critical point is the appearance of an infinite avalanche. An infinite avalanche's equivalent in a finite system is an avalanche that spans the entire system in at least one dimension. To tell whether we are above or below the critical point, we need to detect these *spanning avalanches*. In three and higher dimensions, the number of spanning avalanches as a function of R is also interesting to study. The obvious way of detecting spanning avalanches is to mark each row as a spin flips in it, and check at the end of the avalanche to see if all the rows contain flipped spins. However, this method requires $O(N^{1/D})$ operations per avalanche. Because there are many small avalanches, this method is unacceptable. A preferable method is to keep track of the $2 \times D$ boundaries of the avalanche as it grows. If a pair of boundaries meet, the avalanche is a spanning avalanche. We must take care to treat the periodic boundary conditions properly.

that the program works. Then try these problems:

- We have designed our code so that adding new types of measurements is easy. One quantity that experimentalists measure is the change in magnetization with time. In our code, we record the time series of the whole run, with each avalanche represented as a single point. Each avalanche also has an interesting structure (see Figure 2 in the main article). Add a new class to the program that records the time series within the largest avalanche.
- Implement the brute-force algorithm. You can either replace the `BruteForceHysteresisSimulation` class or implement it from scratch.
- Implement the sorted-list algorithm by replacing the `SortedListHysteresisSimulation` class.
- Implement the bits algorithm.

More information on how to write your own `BruteForceHysteresisSimulation`, `SortedListHysteresisSimulation`, and `BitsHysteresisSimulation` classes is on our Web site.

References

1. J.P. Sethna et al., "Hysteresis and Hierarchies: Dynamics of Disorder-Driven First-Order Phase Transformations," *Physical Rev. Letters*, Vol. 70, No. 21, 1993, pp. 3347–3350.
2. O. Perković, K.A. Dahmen, and J.P. Sethna, "Avalanches, Barkhausen Noise, and Plain Old Criticality," *Physical Rev. Letters*, Vol. 75, No. 24, 11 Dec. 1995, pp. 4528–4531.
3. O. Perković, K.A. Dahmen, and J.P. Sethna, "Disorder-Induced Critical Phenomena in Hysteresis: A Numerical Scaling Analysis," cond-mat #9609072, Los Alamos Nat'l Laboratory, Los Alamos, N.M.; http://xxx.lanl.gov/abs/cond-mat/9609072.
4. O. Perković, K.A. Dahmen, and J.P. Sethna, "Disorder-Induced Critical Phenomena in Hysteresis: Numerical Scaling in Three and Higher Dimensions," cond-mat #9807336, Los Alamos Nat'l Lab, Los Alamos, N.M.; http://xxx.lanl.gov/abs/cond-mat/9807336.

Another useful function is the avalanche size distribution, $D(S)$, defined as the number of avalanches that flip S spins during the simulation, divided by the total number of spins. Like the $M(H)$ curve, the avalanche size distribution scales linearly with the system size. Thus, we need bins up to size N, the size of the largest possible avalanche. Logarithmic binning is the obvious solution, with bin n including all sizes $b_a^{n-1} < S < b_a^n$. We have chosen b_a from 1.01 to 1.1. Large bins are preferable for lower statistical noise. This choice is particularly important in the tail of very large avalanches, where small bins would contain few avalanches. However, very large bins will systematically alter the shape of the scaling functions (although they will not change the critical exponents). It is important to divide the final population in each bin by the number of integers in the bin (and not just the bin width). We should also ignore the early bins that do not contain any integers.

We calculate the correlation function $G(x, R)$ in an avalanche, where $G(x, R)$ gives the probability that the first spin in an avalanche will cause a spin a distance x away to flip in that avalanche. At the beginning of each avalanche, we record the coordinates of its first spin. Then, for each subsequent spin in the avalanche, we calculate the distance x to the first spin and add one to the appropriate bin. Logarithmic binning is not necessary for the correlation function, because the correlation function's size is proportional to the system's length, not the total number of spins. Thus, we use a fixed bin size $b_c = 1$. At the end of the simulation, each bin should be normalized by the number of spins that are between $x - b_c/2$ and $x + b_c/2$ from the origin.

The only tricky part of calculating $G(x, R)$ comes from the periodic boundary conditions. If the avalanche crosses a boundary, two points at opposite ends of the avalanche can come close together. Because we do not calculate $G(x, R)$ for spanning avalanches, we know that at least one row in every dimension will be untouched by the avalanche. To calculate separations, we use the periodicity of the lattice and the continuity of the avalanche to shift the coordinates so that they are all on one side of these empty rows. Because we are already keeping track of the avalanche's boundaries for the detection of spanning avalanches, finding an empty row is easy.

Figure 8 shows the running times of the three algorithms as a function of system size. The brute-force algorithm can be useful when you care only about $M(H)$ at a few points but is otherwise too slow for large systems. The sorted-list algorithm is the fastest algorithm, but a 128-Mbyte machine can handle system sizes of only up to six million spins. The bits algorithm is almost as fast as the sorted-list algorithm, and asymptotically uses only one bit of memory per spin.

Working on hysteresis, avalanches, and noise with our model has been very rewarding. The simulations are beautiful and entertaining in themselves. Developing faster, more space-efficient algorithms was amazingly satisfying: each new method not only eased our lives and our computer budgets but also opened a whole new window on the model's behavior. Developing new ways of measuring what was happening in the simulations was also fun: watching the spins flip and measuring the avalanche size distribution was only the beginning. We have written a few exercises that we hope will entertain and inform you as they did us (see the "Suggestions for further study" sidebar). CiSE

Acknowledgments

NSF DMR-9805422 and DOE DEFG02-88-ER45364 supported portions of this work. We also gratefully acknowledge an equipment grant from Intel and the support of the Cornell Theory Center. We thank the department editors, Harvey Gould and Jan Tobochnik, for careful and helpful editing suggestions.

References

1. E. Vives et al., "Universality in Models for Disorder-Induced Phase Transitions," *Physical Rev. E*, Vol. 52, No. 1, 1995, pp. R5–8.
2. R. Blossey, T. Kinoshita, and J. Dupont-Roc, "Random-Field Ising Model for the Hysteresis of the Prewetting Transition on a Disordered Substrate," *Physica A*, Vol. 248, 1998, p. 247.
3. H. Ji and M.O. Robbins, "Percolative, Self-Affine, and Faceted Domain Growth in Random Three-Dimensional Magnets," *Physical Rev. B*, Vol. 46, No. 22, 1992, pp. 14519–14527.
4. J.P. Sethna et al., "Hysteresis and Hierarchies: Dynamics of Disorder-Driven First-Order Phase Transformations," *Physical Rev. Letters*, Vol. 70, No. 21, 1993, pp. 3347–3350.
5. J.-G. Zhu and H.N. Bertram, "Self-Organized Behavior in Thin-Film Recording Media," *J. Applied Physics*, Vol. 69, 1991, p. 4709.
6. J.P. Sethna, O. Perković, and K.A. Dahmen, in *Scale Invariance and Beyond*, B. Dubrulle, F. Graner, and D. Sornette, eds., Springer-Verlag, Berlin, 1997, p. 87. In this reference, Figure 1 (right) is incorrect: it shows dM/dt for the whole hysteresis loop, not just one avalanche.
7. O. Perković, K.A. Dahmen, and J.P. Sethna, "Avalanches, Barkhausen Noise, and Plain Old Criticality," *Physical Rev. Letters*, Vol. 75, No. 24, 11 Dec. 1995, pp. 4528–4531.
8. O. Perković, K.A. Dahmen, and J.P. Sethna, "Disorder-Induced Critical Phenomena in Hysteresis: A Numerical Scaling Analysis," cond-mat #9609072, Los Alamos Nat'l Laboratory, Los Alamos, N.M.; http://xxx.lanl.gov/abs/cond-mat/9609072.
9. O. Perković, K.A. Dahmen, and J.P. Sethna, "Disorder-Induced Critical Phenomena in Hysteresis: Numerical Scaling in Three and Higher Dimensions," cond-mat #9807336, Los Alamos Nat'l Laboratory, Los Alamos, N.M.; http://xxx.lanl.gov/abs/cond-mat/9807336.
10. B. Drossel and K. Dahmen, "Depinning of a Domain Wall in the 2D Random-Field Ising Model," *European Physical J. B*, Vol. 3, No. 4, II June 1998, pp. 485–496.
11. A.A. Middleton, "Asymptotic Uniqueness of the Sliding State for Charge-Density Waves," *Physical Rev. Letters*, Vol. 68, No. 5, 1992, pp. 670–673.
12. A.A. Middleton and D.S. Fisher, "Critical Behavior of Charge-Density Waves below Threshold: Numerical and Scaling Analysis," *Physical Rev. B*, Vol. 47, No. 5, 1993, pp. 3530–3552.
13. U. Wolff, "Collective Monte Carlo Updating for Spin Systems," *Physical Rev. Letters*, Vol. 62, No. 7, 1989, pp. 361–364.
14. S. Zapperi et al., "Dynamics of a Ferromagnetic Domain Wall: Avalanches, Depinning Transition, and the Barkhausen Effect," *Physical Rev. B*, Vol. 58, No. 10, 1 Sept. 1998, pp. 6353–6366.
15. L. Furuberg et al., "Dynamics of Invasion Percolation," *Physical Rev. Letters*, Vol. 61, No. 18, 1988, pp. 2117–2120.
16. A.B. Bortz, M.H. Kalos, and J.L. Lebowitz, "A New Algorithm for Monte Carlo Simulation of Ising Spin Systems," *J. Computational Physics*, Vol. 17, 1975, p. 10.
17. M.A. Novotny, "A New Approach to an Old Algorithm for the Simulation of Ising-Like Systems," *Computers in Physics*, Vol. 9, 1995, p. 46.

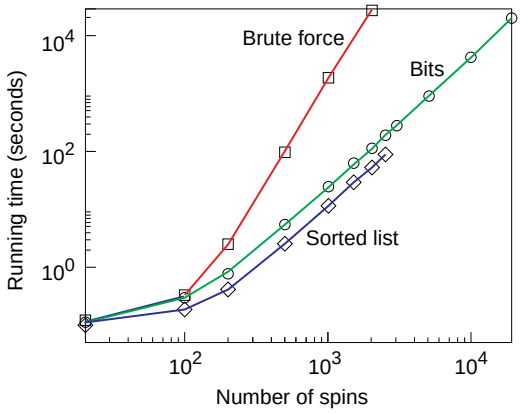

Figure 8. The running times for the three algorithms for 2D systems with $R = 1.0$ on a 266-MHz Pentium II with 128 Mbytes of memory. The brute-force running time grows quadratically; the sorted-list algorithm and the bits algorithm have run times that grow approximately linearly (the log$N$ is not visible). The largest bits simulation was 64 times larger than the largest sorted-list simulation.

Matthew C. Kuntz is a graduate student in Cornell University's Department of Physics. He is writing his PhD dissertation on hysteresis and Barkhausen noise in magnetic materials. He received his BA in physics from Amherst College. He is a member of the Sigma Xi science research society and the Materials Research Society. Contact him at the Physics Dept., Clark Hall, Cornell Univ., Ithaca, NY 14853; mck10@cornell.edu.

Olga Perković works for McKinsey & Company. She received her PhD in physics from Cornell University. Contact her at olga_perkovic@mckinsey.com.

Karin A. Dahmen is an assistant professor of physics at the University of Illinois at Urbana-Champaign. She is studying nonequilibrium dynamical systems, including pattern formation in homogeneous systems and inhomogeneous systems with quenched disorder. She received her PhD in physics from Cornell University. Contact her at the Dept. of Physics, Univ. of Illinois, Urbana, IL 61801; dahmen@uiuc.edu.

Bruce W. Roberts works for Starwave Corporation. He received his PhD in physics from Cornell University. Contact him at bwr@halcyon.com.

James P. Sethna is a professor in Cornell University's Department of Physics. He is studying hysteresis and Barkhausen noise in magnetic materials and multiscale modeling of defects in solids. He received his PhD in physics from Princeton University. He is a member of the American Physical Society and the Materials Research Society. Contact him at LASSP, Physics Dept., Clark Hall, Cornell Univ., Ithaca, NY 14853-2501; sethna@lassp.cornell.edu; www.lassp.cornell.edu/sethna.