Problem Set 6
EPS 528, Science of Complex Systems

Jonas Katona

November 16, 2022

Problem 1.

Solution. From lecture, we already know that the stationary distribution vector for M_1 is $\pi_1 \approx (0.3529, 0, 0.0504, 0.1345, 0.3361, 0.1261)$. In particular, note that any user will almost surely not end up on page 2. Why? Well, as we see in M_1, no pages link to page 2, and hence, if some user started on page 2, they would almost surely end up on another page after some time. Furthermore, the most pages link to pages 1 and 5, and this is indicated by how these have the top two highest probabilities in π_1. Page 3 also has a comparatively small probability associated with it because the only page which links to page 3 is page 4, which is not nearly as popular as pages 1 and 5. The only reason why page 4 has a higher probability is because page 5 links to it, and page 5 is almost the most popular page. Similarly, page 6 has a similar probability associated with it in π_1 because page 5 also links to it. Page 1 also only has the highest probability because it only links to one other page: Page 5, from which there are many paths a user could take to reach page 1 again.

M_1 is constructed as is in the case when page 1 links to page 5 only. This is reflected by letting $a_{51} = 1/2$, where the probability represented here is $1/2$ because the first column vector of M_1 has only two nonzero entries, each representing an equal probability; these correspond to the events of staying at page 1 or going to page 5. If page 1 doesn't link to anything, then in particular, page 1 doesn't link to page 5. Hence, $a_{51} = 0$, and to renormalize the first column vector of M_1, we let $a_{11} = 1$ and keep $a_{21} = \cdots = a_{61} = 0$. Accordingly,

$$\boxed{M_2 = \begin{pmatrix} 1 & 1/4 & 1/3 & 1/4 & 1/4 & 1/3 \\ 0 & 1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/4 & 0 & 0 \\ 0 & 1/4 & 1/3 & 1/4 & 1/4 & 0 \\ 0 & 1/4 & 0 & 1/4 & 1/4 & 1/3 \\ 0 & 0 & 0 & 0 & 1/4 & 1/3 \end{pmatrix}.} \tag{1}$$

The stationary distribution vector for M_2, π_2, will satisfy the matrix equation $M_2\pi_2 = \pi_2$. Hence, π_2 is an eigenvector for M_2 with eigenvalue 1. To find π_2, it is not worth it to find all of the eigenvectors and their associated eigenvalues for (1). Instead, we can just consider what $\pi_2 = (v_1, v_2, v_3, v_4, v_5, v_6)$ has to be, i.e., we have to solve a system of linear equations for $v_1, \ldots, v_6$ such that $M_2\pi_2 = \pi_2$. By inspection, we immediately see that the stationary distribution vector must be $\boxed{\pi_2 = (1, 0, 0, 0, 0, 0)}$, i.e., as $n \to \infty$, we can expect that everybody will end up on page 1 almost surely. This is because the net-surfers cannot go to any other pages once they make it to page 1, and since all of the other pages link to page 1 (or rather, there exists a path from any page to page 1), they are bound to eventually get there and be stuck with probability 100%. This indicates that in a graph, removing even one edge could drastically change the topological properties of the graph and the probabilities associated with going between nodes.

Computing M_3 is more complicated. As before, regardless of what M_3 is, we know that the stationary distribution vector for M_3, π_3, satisfies the matrix equation $M_3\pi_3 = \pi_3$. If we pick random pages to visit during 10% of the net-surfing time instead of navigating through links, our Markov matrix for this portion of the net-surfing *alone* is just $[U]_{ij} = 1/6$, i.e., a matrix where all entries are 1/6, because from any page, we can go to another page with equal probability, including back to the page we were previously on. Then, to compute M_3, we can use the law of total probability and condition on whether we are in that 10% of the net-surfing time where we pick random pages or acting according to M_2. Hence,

$$M_3 = \frac{9}{10}M_2 + \frac{1}{10}U$$
$$= \frac{9}{10}\begin{pmatrix} 1 & 1/4 & 1/3 & 1/4 & 1/4 & 1/3 \\ 0 & 1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/4 & 0 & 0 \\ 0 & 1/4 & 1/3 & 1/4 & 1/4 & 0 \\ 0 & 1/4 & 0 & 1/4 & 1/4 & 1/3 \\ 0 & 0 & 0 & 0 & 1/4 & 1/3 \end{pmatrix} + \frac{1}{10}\begin{pmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{pmatrix}$$
$$\Rightarrow \boxed{M_3 = \begin{pmatrix} 11/12 & 29/120 & 19/60 & 29/120 & 29/120 & 19/60 \\ 1/60 & 29/120 & 1/60 & 1/60 & 1/60 & 1/60 \\ 1/60 & 1/60 & 19/60 & 29/120 & 1/60 & 1/60 \\ 1/60 & 29/120 & 19/60 & 29/120 & 29/120 & 1/60 \\ 1/60 & 29/120 & 1/60 & 29/120 & 29/120 & 19/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 29/120 & 19/60 \end{pmatrix}.} \tag{2}$$

Since $M_3\pi_3 = \pi_3$, we need to find the eigenvector of eigenvalue 1 for M_3. We can compute the eigensystem of (2) using Mathematica,[1] from which we find that the eigenvector corresponding to eigenvalue 1 is $v_3 = (9023/521, 254/521, 1, 746/521, 746/521, 1)$. Normalizing v_3 such that its components add up to 1,

$$\boxed{\begin{array}{c} \pi_3 = \left(\frac{9023}{11811}, \frac{2}{93}, \frac{521}{11811}, \frac{746}{11811}, \frac{746}{11811}, \frac{521}{11811}\right) \\ \approx (0.7639, 0.0215, 0.0441, 0.0632, 0.0632, 0.0441)\,. \end{array}} \tag{3}$$

(3) shows how, even if a user scrolls through random pages only 10% of the time, we have a significantly different stationary distribution vector from the case for M_2. Of course, the probability for somebody to end up on another page other than page 1 is still pretty fractional, but this indicates something fundamental about Markov chains: A stationary distribution like π_2 is unstable under stochastic perturbations or random noise. Note also that the size of the probabilities associated with each page in π_3 follows a similar order as π_1, merely because the only thing we changed was the link from page 1 to page 5, which we removed. □

Problem 2.

Solution. At first, here is the MCMC simulation codes I used:

First part (`Ising_simulation.m`):

[1]The Mathematica script I used is attached to the end of this document.

```
function [ data ] = Ising_simulation(coeff, N, iter, vis)
% `coeff` is the reduced coupling constant (=J/(kT))
% `N` is the one-dimensional size of the grid
% (`N^2` is the size of the square)
% `iter` is the total number of iterations
% `vis` should be a logical value indicating if we should plot

spinsk =  2 * randi(2, N, N) - 3; % randomly assign spins to NxN grid

if vis
    figure(1)
    imagesc(spinsk)
    colormap(cool)
    colorbar
    caxis([-1 1])
    title('Initial spin configuration')
    xlabel('x-position')
    ylabel('y-position')
end

% calculate energy of initial system
% sum over nearest row neighbors and nearest column neighbors
Ek = sum(spinsk(2 : N, :) .* spinsk(1 : N - 1, :), 'all')...
    + sum(spinsk(:, 2 : N) .* spinsk(:, 1 : N - 1), 'all');
pik = exp(coeff * Ek);
% setup coordinate array
coords = zeros(2, N ^ 2);
for i = 1 : N
    coords(:, N * (i - 1) + 1 : N * i) = [i * ones(1, N); 1 : N];
end

M0 = N ^ 2 / 5;
Mfunc = @(x) M0 ^ (1 - (2 * x / iter));
Mq = M0;
sample = zeros(2, iter);
for k = 1 : iter
    % perturb current microstate
    % choose M random sites to perturb
    loc = randperm(N ^ 2, Mq);
    spinsy = spinsk;
    % perturb randomly chosen sites
    spinsy(sub2ind(size(spinsy), coords(1, loc), coords(2, loc)))...
        = 2 * randi(2, 1, Mq) - 3;
    % calculate energy of perturbed system
    Ey = sum(spinsy(2 : N, :) .* spinsy(1 : N - 1, :), 'all')...
        + sum(spinsy(:, 2 : N) .* spinsy(:, 1 : N - 1), 'all');
    % compute relative probabilities of both states
    piy = exp(coeff * Ey);
    % step 4
    r = rand;
    if (piy >= pik) || (r < (piy / pik))
        spinsk = spinsy;
        pik = piy;
    end
    % decrease M exponentially at each iteration
    Mq = max(1, ceil(Mfunc(k)));
    % save sample size at each iteration
    sample(1, k) = Mq;
    % save average spin at each iteration
    sample(2, k) = mean(spinsk, 'all');
end

if vis
    figure(2)
    plot(sample(1, :))
    title('Perturbation size vs. iteration')
    xlabel('Iteration #')
    ylabel('Perturbation size')

    figure(3)
    plot(sample(2, :))
    title('Average spin vs. iteration')
    xlabel('Iteration #')
    ylabel('Average spin')

    figure(4)
    imagesc(spinsk)
    colormap(cool)
    colorbar
    caxis([-1 1])
    title('Final spin configuration')
    xlabel('x-position')
    ylabel('y-position')
end

data = zeros(2, 1);
data(1) = mean(sample(2, end / 2 : end));
data(2) = std(sample(2, end / 2 : end));
```

Second part (`problem2tests.m`):

```
N = 20;
iter = 10 ^ 7;
coeffs = 0 : 0.1 : 1;
data = zeros(2, length(coeffs));

% iterate through each reduced coupling coefficient
for i = 1 : length(coeffs)
```

```
    datum = Ising_simulation(coeffs(i), N, iter, false);
    disp(datum)
    disp(coeffs(i))
    data(:, i) = datum;
end

% save the data (change name manually for each trial)
save problem2data3.mat data
```

Third part (`problem2plots.m`):

```
load problem2data1.mat data
data1 = data;
load problem2data2.mat data
data2 = data;
load problem2data3.mat data
data3 = data;

coeffs = 0 : 0.1 : 1;
figure(1)
plot(coeffs, abs([data1(1, :); data2(1, :); data3(1, :)]), 'o-')
title('Absolute mean spin vs. reduced coupling coefficient')
xlabel('$\frac{J}{kT}$', 'interpreter', 'latex')
ylabel('$\textrm{Absolute mean of average spin}$',...
    'interpreter', 'latex')
x1 = xline(0.4407, '--k', 'Critical value');
x1.LabelVerticalAlignment = 'bottom';

figure(2)
plot(coeffs, [data1(1, :); data2(1, :); data3(1, :)], 'o-')
title('Mean spin vs. reduced coupling coefficient')
xlabel('$\frac{J}{kT}$', 'interpreter', 'latex')
ylabel('$\textrm{Mean of average spin}$',...
    'interpreter', 'latex')
x1 = xline(0.4407, '--k', 'Critical value');
x1.LabelVerticalAlignment = 'bottom';

figure(3)
plot(coeffs, [data1(2, :); data2(2, :); data3(2, :)], 'o-')
title('Standard deviation vs. reduced coupling coefficient')
xlabel('$\frac{J}{kT}$', 'interpreter', 'latex')
ylabel('$\textrm{Standard deviation of average spin}$',...
    'interpreter', 'latex')
x1 = xline(0.4407, '--k', 'Critical value');
x1.LabelVerticalAlignment = 'bottom';
```

These codes are in three parts. The second one, `problem2tests.m`, references the first one, `Ising_simulation.m`, and saves the data in files which are called in the third one, `problem2plots.m`, for plotting. However, `Ising_simulation.m` consists of the main part of the code and contains the Metropolis algorithm I used. I followed the steps outlined in Jun's MCMC notes (Chapter 6.2, "Importance Sampling" in the draft for his textbook) as posted on bCourses pretty faithful, but I should also outline some specific details to my code:

- I *do not* calculate the total magnetization of each microstate at each timestep nor at the end of the simulation because we do not need it for this problem. However, since we save the spins at each iteration, we would have no problem computing it in theory. In fact, it would just necessitate adding a two lines of code: one before the for loop to initialize the vector of size 1x`iter`, and another within the loop to compute the total magnetization by summing over the elements in `spinsk` and saving this value.

- We can say the same as in the above bullet about the spin iteraction energy. *However*, there are two main caveats. Firstly, we would need to specify value of J; we did not need one in our current code because the relative probability of a state depends off of $\pi = \exp(-\beta E) = \exp\left(-\beta\left(-J\sum_{i,j} s_i s_j\right)\right) = \exp\left(\frac{J}{kT}\sum_{i,j} s_i s_j\right)$, which we observe only depends off of the so-called reduced coupling coefficient $J/(kT)$. Furthermore, `Ek` and `Ey` as computed in the code are technically scaled by J; I called these "energies" in the comments within my code, but these are not the precise energies E_k and E_y in the appropriate sense. However, `pik`$= \pi_k$ and `piy`$= \pi_y$ are exactly as labeled.

- We decrease our initial perturbation size exponentially at each iteration, which we will outline in greater detail later on in this problem. In particular, the rate at which this perturbation size decreases is such that *only* after exactly `iter`/2 iterations will the perturbation size be precisely one element in the grid. Hence, we not only greatly benefit the larger we make `iter`, but we have lots of room for the system to correct itself and reach a more energy-favorable microstate in the latter half of the iterations. This follows the same sort of intuition used in simulated annealing, where gradually decrease the perturbation size as well in finding a global optimum.

- For our tests, I let `iter`$= 10^7$ rather than 10^6 because I found that my results did not converge quickly enough to give reasonable results after only 10^6 iterations. It did sometimes, but not always; with these randomized algorithms, we only know that, the more iterations we run, the greater *probability* that the solution will converge within some desired error tolerance, but this is not known with absolute certainty. Anyways, with 10^7 iterations, while the code took noticeably longer to run (around an hour to sweep through one trial, i.e., one sampling of $J/(kT) = 0, 0.1, \ldots, 1$), it converged as expected for the three trials I tested on the first try, as we can see in Figures 1 and 2.

We now go through the derivation of the function $\varepsilon = \varepsilon(k)$, where $k = 1, \ldots,$`iter` and ε is the number of elements we perturb at iteration k. Note in our code that ε is `M`. At first, we impose that ε is of the form $\varepsilon(k) = \exp(A + Bk)$ for constants $A \in \mathbb{R}$ and $B < 0$ to be determined. Next, we impose our initial perturbation size $\varepsilon_0 \leq N^2$; in particular, for our code, we let $\varepsilon_0 = N^2/5$, such that $\varepsilon(0) = \varepsilon_0 = N^2/5 \Rightarrow A = \log\varepsilon_0 = \log(N^2/5)$. Finally, as explained in bullet 3 above, we need to impose $\varepsilon(\texttt{iter}/2) = 1 \Rightarrow B = -\log\varepsilon_0(2/\texttt{iter}) = -\log(N^2/5)(2/\texttt{iter})$. Hence, $\boxed{\varepsilon(k) = \varepsilon_0^{1-2k/\mathrm{iter}}}$. Of course, since our perturbation size can only be a whole number, we really only need $\lceil\varepsilon\rceil = \lceil\varepsilon(k)\rceil$.

In the analytic solution which Jun derived in lecture on Wednesday, we expect a continuous phase transition to occur at $\widetilde{J}^*\,(J/(kT))^* = \left[\log\left(1+\sqrt{2}\right)\right]/2 \approx 0.4407$, a result which can be derived via renormalization group techniques. In particular, for $\widetilde{J} < \widetilde{J}^*$, we expect

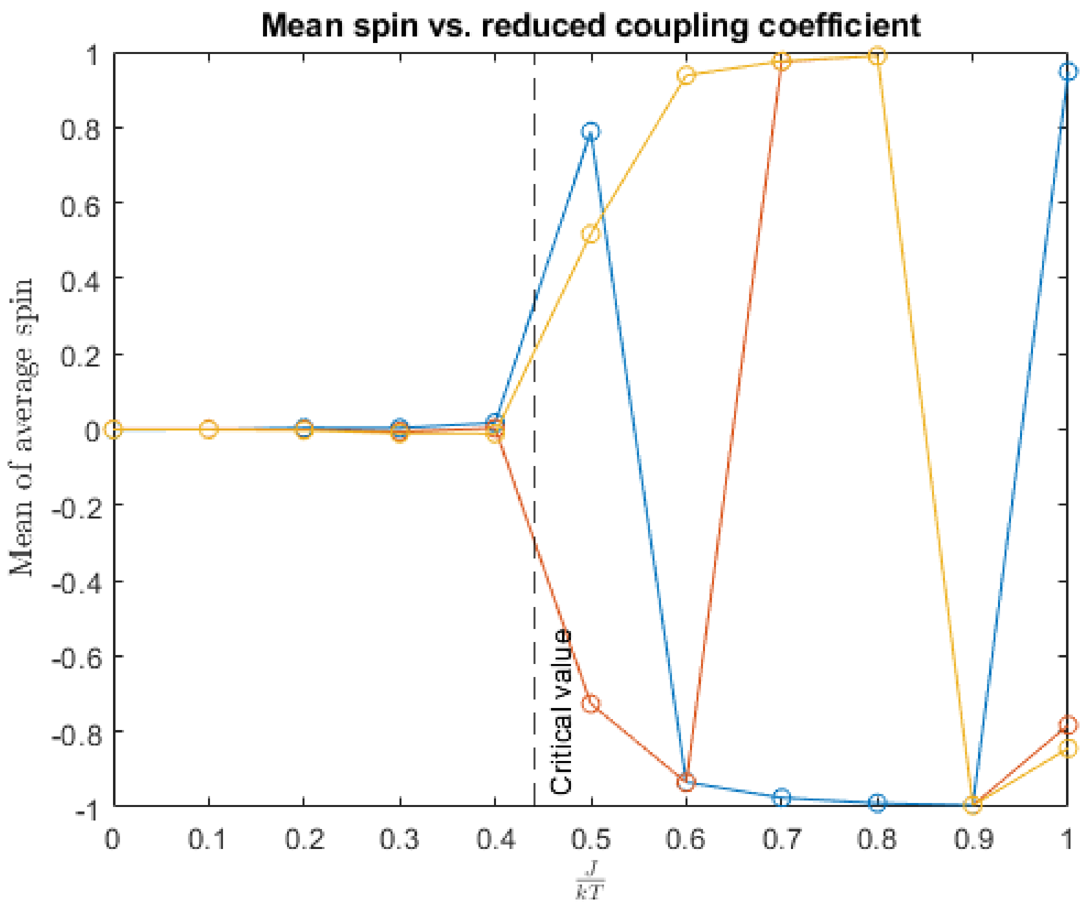

Figure 1: We observe that, with great consistency, $J/(kT) < 0.4407$ leads to a mean average spin of 0, indicating complete disorder and decorrelation across the spins. However, once $J/(kT) > 0.4407$, the system either approaches a mean average spin of $+1$ or -1.

that the system will have a mean-zero spin, with a roughly equal number of $+1$ and -1 spins; the spins are decorrelated. However, once $\widetilde{J}$ increases past $\widetilde{J}^*$, the energy favorable microstate will be dominated by all $+1$ or -1 as the system is completely synchronized and the spins are correlated. We explain how well our results compare to these in more detail in the captions of the figures. Note that in the first three figures, there are three curves indicating three separate trial runs across $J/(kT) = 0, 0.1, \ldots, 1$.

By the way, running `Ising_simulation.m` alone with `vis=true` will return plots for the initial configuration, number of perturbed cells vs. iteration, the average spin vs. iteration, and the final configuration. A sample of some results is found in Figure .

$\square$

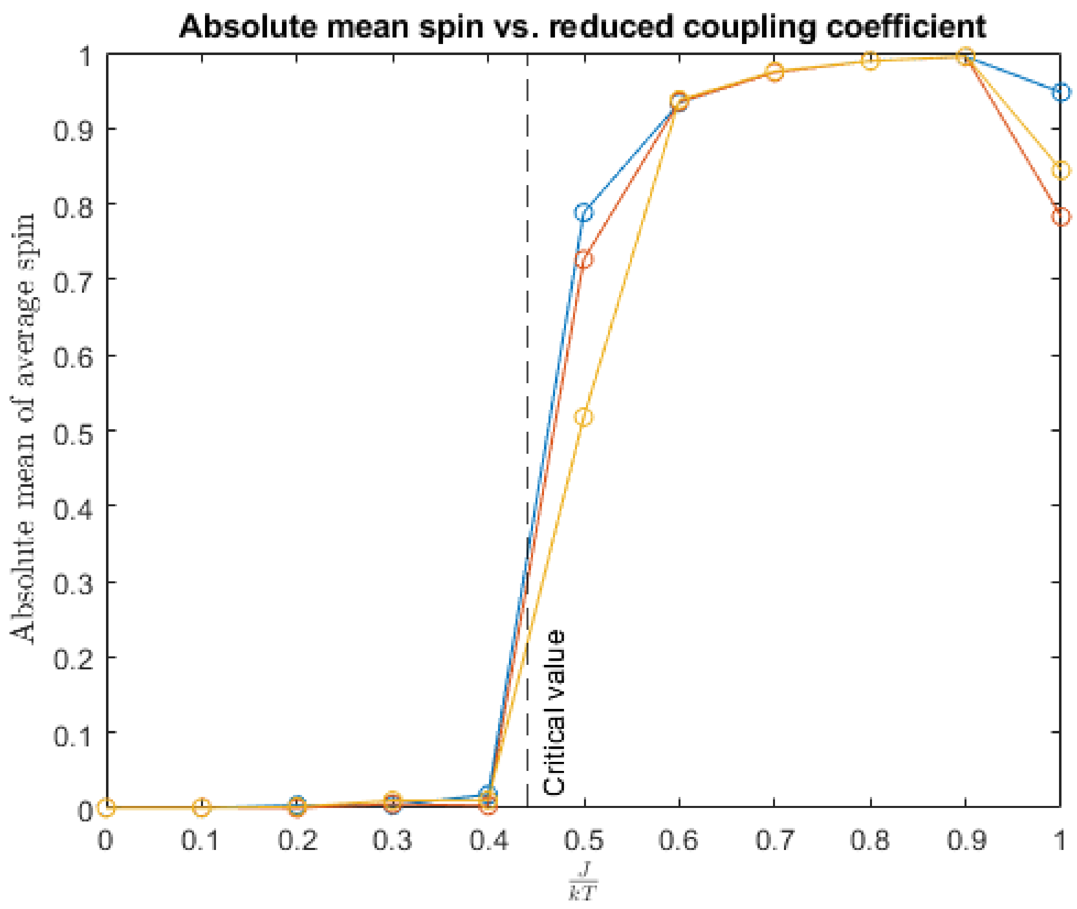

Figure 2: The results here are the same as in 2, except we plot the absolute value of the average spin to indicate the trend more clearly. In particular, I found that when running the simulation, the algorithm will rarely but spontaneously change the microstates from mostly $+1$ to -1 after a rather large number of perturbations.

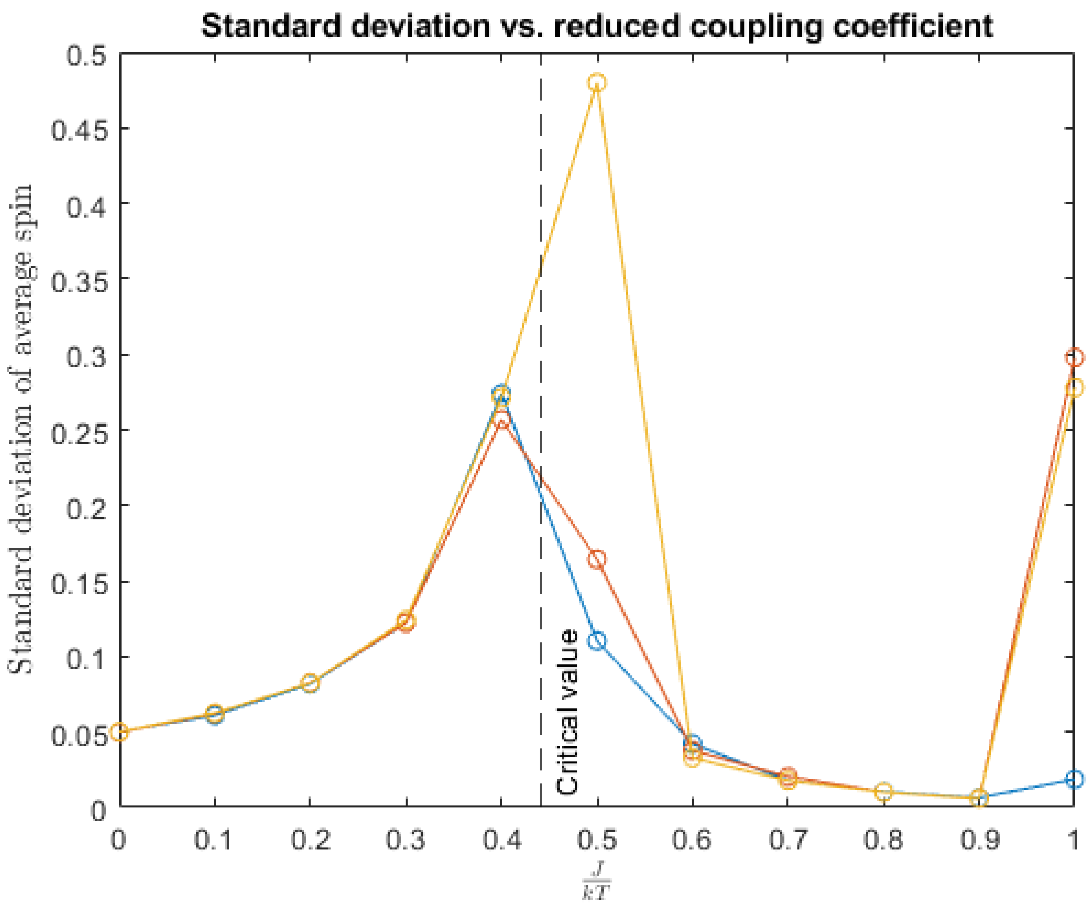

Figure 3: Note that the standard deviation peaks near the phase transition, which indicates that the algorithm converges slower the closer we are towards this phase transition.

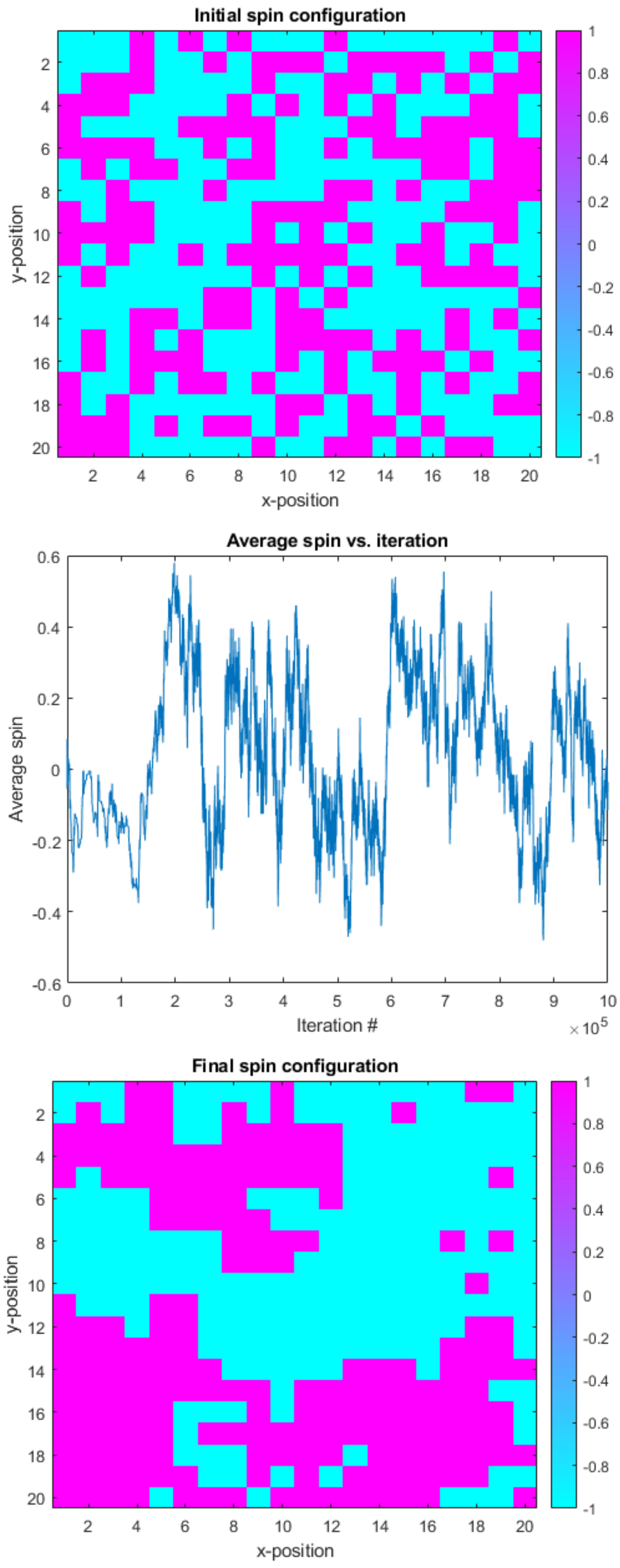

Figure 4: Sample results with $J/(kT) = 0.5$ and iter $= 10^6$.