Ising-Like Modelling of Ferromagnetic Materials

Kyle Windsor

June 2013

1 Abstract

This paper attempts to gauge and analyze the quantities associated with a ferromagnetic Ising process over time, such as temperature, equilibrium energy, and grid configuration. A correlation density matrix is computed and explored as well, and compared to other matrices computed for different temperatures. It is found that temperature and equilibrium energy are proportional to each other in the domain $2 \leq T \leq 3$. The correlation density matrices are all found to contain a peak near the line $y = x$ and peaks in the corners of the grid, while the configurations themselves become more homogenous with temperature. The actual state of the grid also becomes more 'noisy' with temperature, and at low temperatures, grid spins tend to cluster together.

2 Introduction

Ferromagnetic materials are innumerable in everyday life, from refridgerator magnets to bullet trains. It stands to reason that they would be useful to model to determine their relationship with temperature, time elapsed, and position in general. A statistical model of ferromagnetism is the Ising model, which can be applied to quantize a ferromagnetic material into regions containing a spin value and modify each based on the spins surrounding it to minimize energy. It is very difficult to compute the Ising model by hand desipite its simplicity, which makes it extremely suitable for numerical analysis.

The Metropolis Monte Carlo algorithm will be used to approach a steady state solution for a ferromagnetic material in the absence of an external magnetic field using Ising-like dynamics. The algorithm will be tested on three different temperatures - $T = 2$, $T = 2.3$, and $T = 3$ - and the generated energies will be compared. A correlation density matrix will be produced and analyzed for each aforementioned temperature, and an attempt will be made to explore and relate qualities of the material such as equilibrium energy, temperature, and grid position over time.

3 Approach

Implementation of the system was fairly straightforward. A grid sized 200×200 was used to simulate regions of the material. The equilibrium energy of the system was defined in the below equation, where $nn(i)$ are the nearest neighbors of the element at i.

$$E = -J \sum_{i=1}^{N} \sum_{j=nn(i)}^{N} s_i s_j \tag{1}$$

To sum the energy and the change in energy efficiently, elements in the matrix were looped through and had their associated energy change, ΔE, computed in a straightforward manner:

$$\Delta E = -2E \tag{2}$$

where E is the current contribution from the element. The value represents how much energy would be gained or lost due to a transition of the element alone, and is far more efficient than if the entire grid energy was calculated, an element flipped, and the new energy calculated and subtracted. The coefficient -2 is used as the element's state changes from -1 to $+1$, a difference of 2, with the negation implying the change in energy will oppose the current spin. The probability of flipping a region's state due to the Boltzmann factor was determined, where we assume $K = 1$ for simplicity:

$$\text{random}(0, 1) \leq \exp(\frac{-\Delta E}{KT}) \tag{3}$$

and modified to have a probability of 1 if $\Delta E \leq 0$. This "flipping" of the entire grid was calculated and performed for 20 steps per configuration, and 1000 configurations were generated per temperature and averaged for a total of 3000 configuration generations.

Correlation density also was implemented in a straightforward manner for each configuration. The elements are looped through to determine the correlation density value at each point:

$$r_{ij} = \frac{< s_i(t)s_j(t) > - < s_i(t) >< s_j(t) >}{\sigma_i \sigma_j} \tag{4}$$

where

$$\sigma_n =< s_n(t)^2 > - < s_n(t) >^2 \tag{5}$$

Plots were generated for the calculated energy and temperature, and for the correlation density matrix

for three different temperatures: $T = 2$, $T = 2.3$, and $T = 3$.

It is important to discuss the computational approach used. The Metropolis Monte Carlo scheme requires individual transitions in its methodology. This can be slow, however, and it would likely be more desireable to use simultaneous matrix-based algorithms (particularly in MATLAB). In practice, though, this is inaccurate in an Ising model. If all states are simultaneously computed, the system will tend to oscillate between different spins of the individual regions, and never fall into a metastable regime. Clearly, such a result is unrealistic and incorrect as a simulation. Though tedious and inefficient at higher values of grid size, we chose to run the simulation with one flip at a time to obtain valid results.

The energy computation was also a somewhat tedious process - in particular, determining ΔE for a certain cell's state flip. Indeed, using matrix operations to determine the energy for the entire grid was relatively efficient, but using matrix operations to determine one cell was not. Finding the sum of the nearest neighbors $nn(i)$ for a certain time was tested via multiple built-in MATLAB commands, such as $circshift$ or creating a new matrix. However, it was determined that the most efficient method to sum $nn(i)$ was to simply use conditional statements.

4 Results

The mesh energies of each temperature can be found in the table below.

Temperature	Avg. Energy/Site
2	-3.1131
2.3	-2.5731
3	-1.6353

Table 1: Table of equilibrium temperatures vs. equilibrium energies/site.

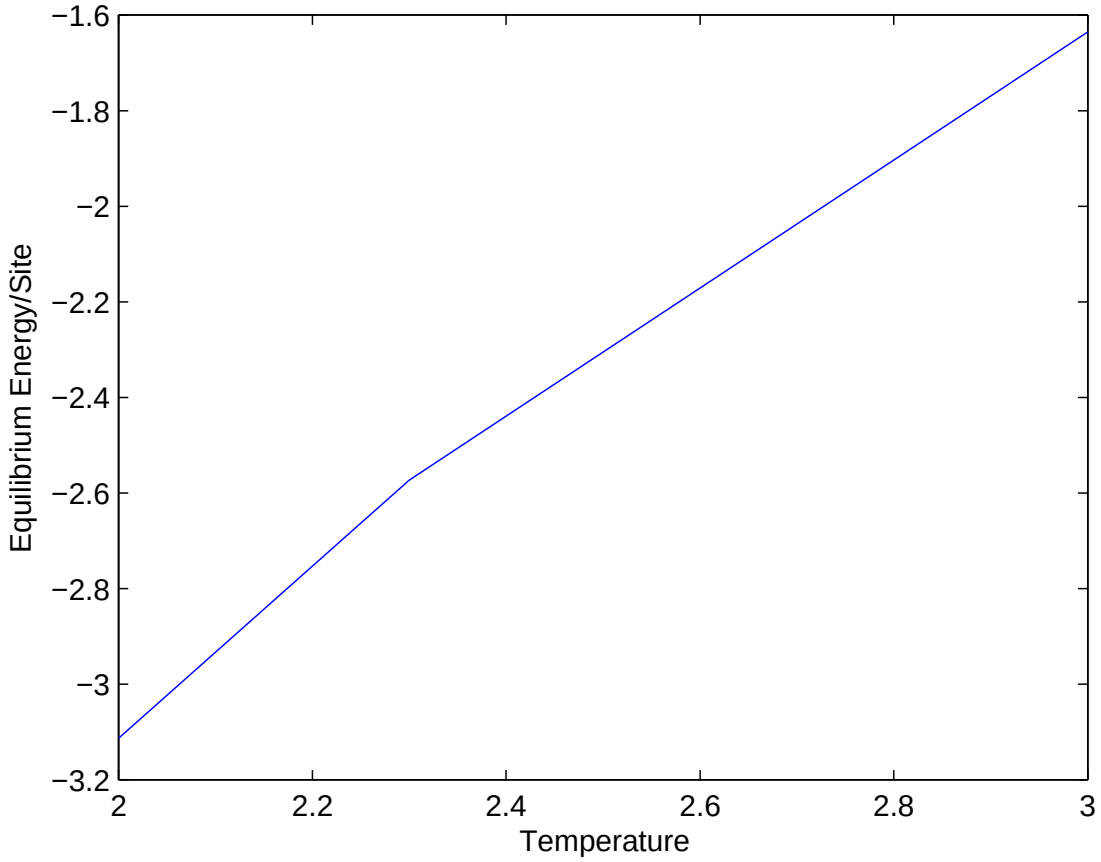


Figure 1: Graph of temperature vs. energy.

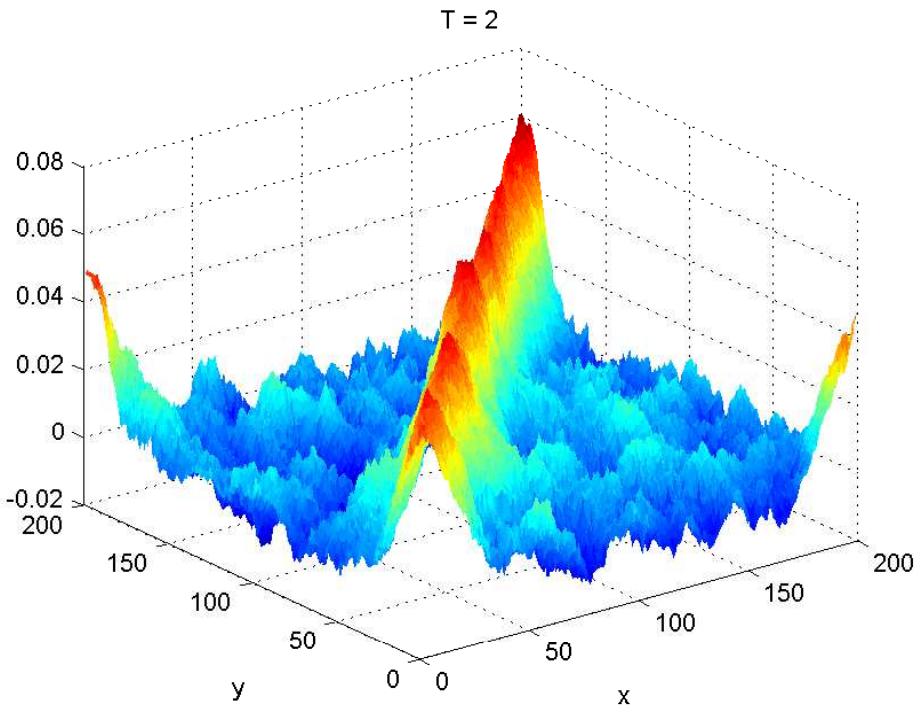


Figure 2: 3D plot of correlation density matrix, $T = 2$.

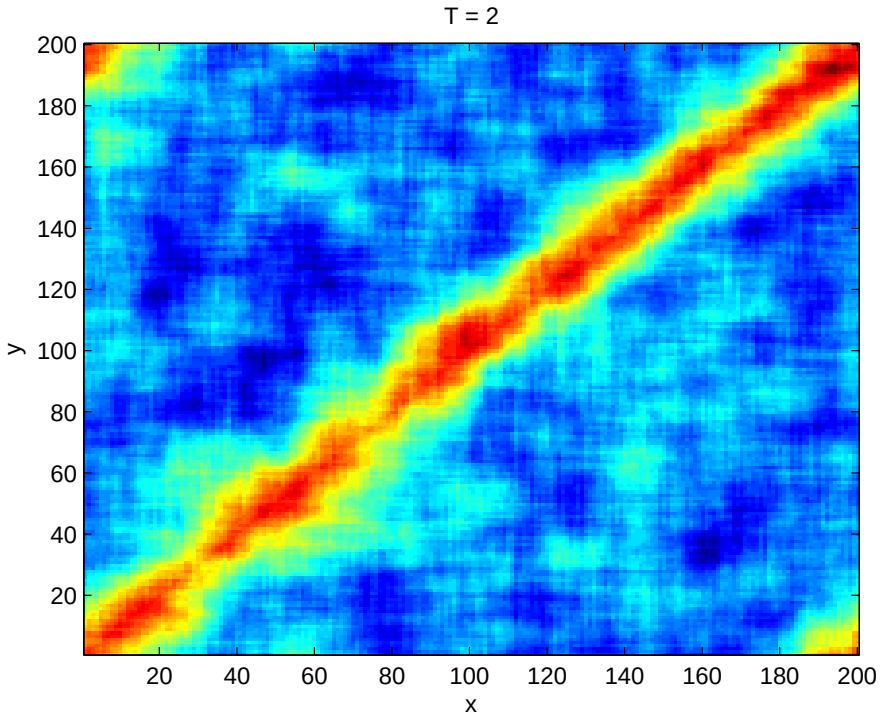


Figure 3: 2D plot of correlation density matrix, $T = 2$.

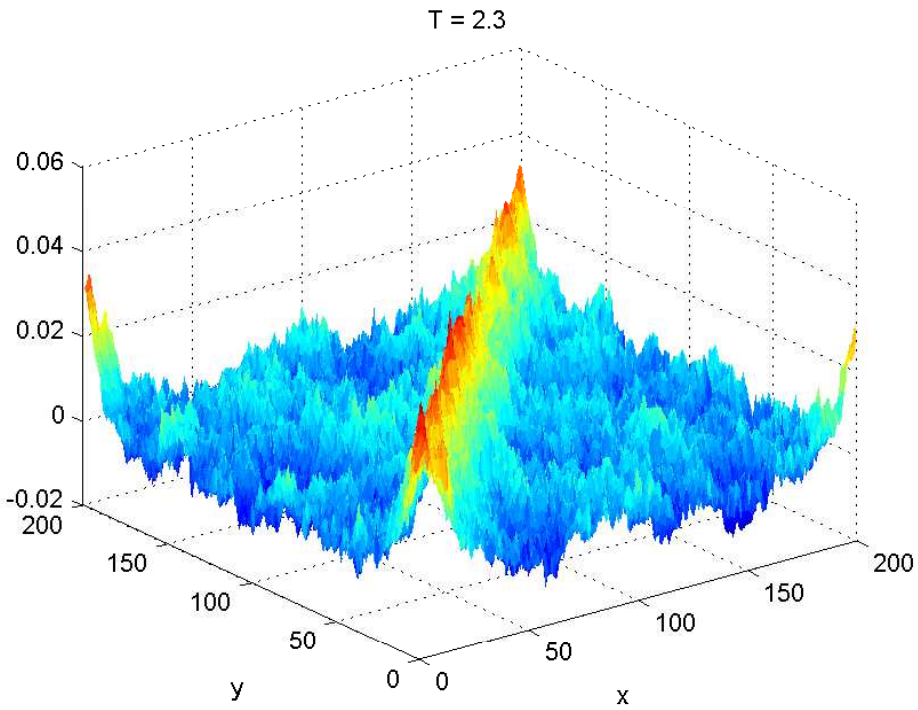


Figure 4: 3D plot of correlation density matrix, $T = 2.3$.

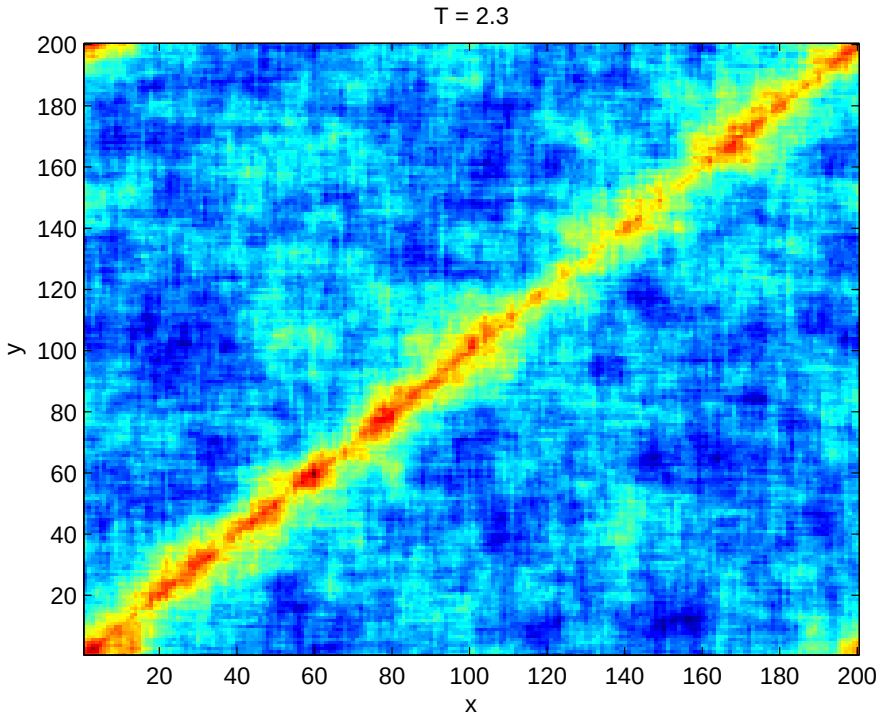


Figure 5: 2D plot of correlation density matrix, $T = 2.3$.

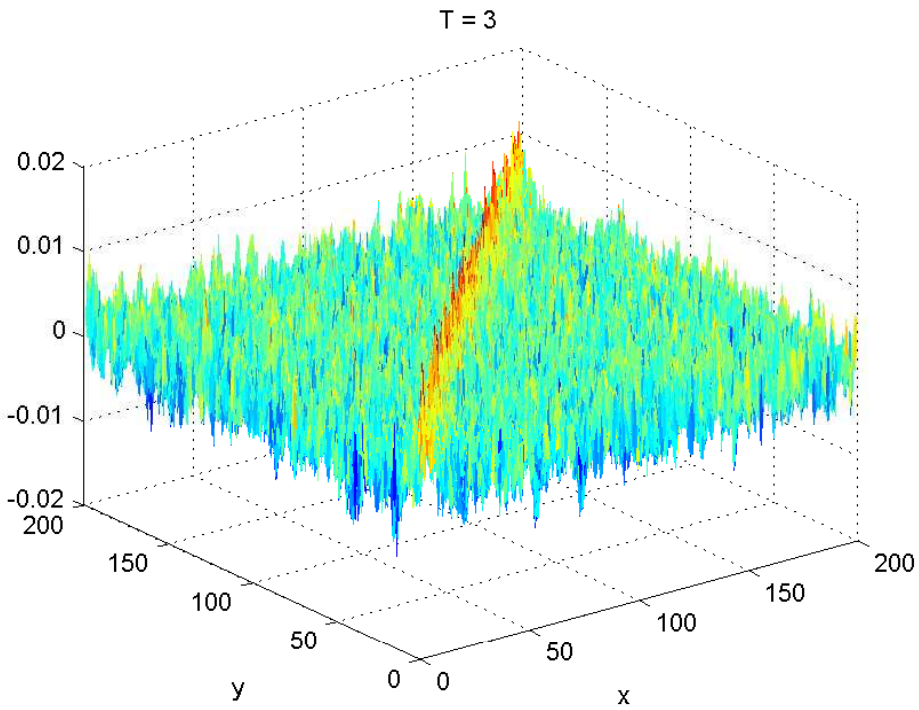


Figure 6: 3D plot of correlation density matrix, $T = 3$.

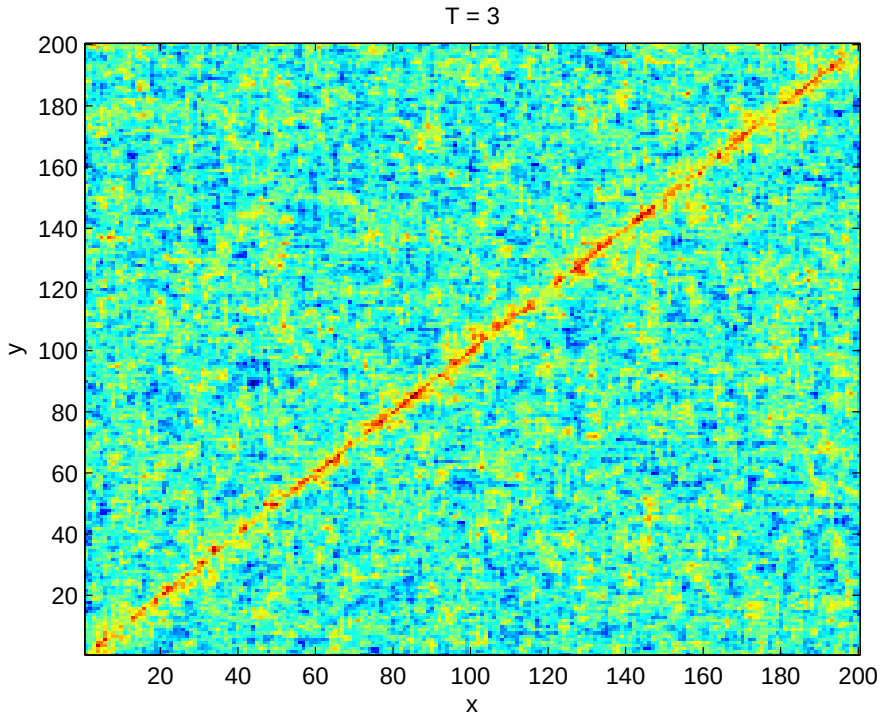
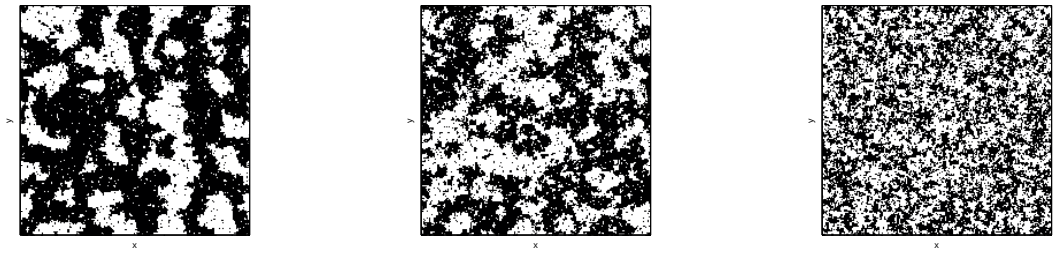


Figure 7: 2D plot of correlation density matrix, $T = 3$.

5 Discussion

As is plain to see in Figure 1, the equilibrium energy is directly proportional to temperature between $T = 2$ and $T = 3$.

It is relatively simple to observe that state of each cell has no effect on the net equilibrium energy, as different configurations tended to the same total equilibrium energy each position, regardless of the shape of the output grid. The final metastable configuration tended to become 'noisier' with temperature, while lower temperatures tended to cluster into groups of states. This can be illustrated below in Figure 8.



(a) $T = 2$ (b) $T = 2.3$ (c) $T = 3$

Figure 8: Sample state diagrams for different temperatures.

In Figures 2 through 7, there is a peak in the correlation density matrix about $y = x$, which has a discernably higher value than the surrounding noise, as well as fringe behavior at the grid corners $(1, 200)$ and $(200, 1)$. A feature of this phenomena is that the amplitude of the peak appears to decrease with temperature, but becomes sharper and more defined. Similarly, the amount of background noise in the region $y \neq x$ increases with temperature. In general, the correlation density matrix becomes more homogenous with temperature. One could extrapolate that at an arbitrarily large temperature, the correlation matrix becomes white noise, as with the state diagrams. In Figure 6, the corner peaks have completely disappeared - this is likely due to a finite grid size, and at a larger grid size, they would likely be observed to some degree.

6 Conclusion

In general, it is difficult to compute an Ising model by hand. With a computer, however, the task becomes much simpler and more in-depth than hand-computation could ever be. Using a Metropolis Monte Carlo algorithm implementation, we were able to perform simulations of ferromagnetic regions outside of a magnetic field as a function of temperature and time. We identified several phenomena, such as the proportionality

between temperature and equilibrium energy of the regions, and correlated them with a correlation density matrix, showing the peak around $y = x$ and demonstrating the increasing homogeneity with temperature. We showed that the homogeneity of the grid also increases with temperature.

Given more computation time, a further area of research could be the computation of other temperatures, in particular outside of the range $2 \leq T \leq 3$, as this will show if the equilibrium energy is truly directly proportional to the temperature. One could also use this data to determine large-scale behavior of the correlation density matrix. Another simulation improvement would be a larger grid size: we would be able to observe the behavior of the corner peaks of the correlation density matrix with more accuracy.

References

[1] Fraiman, D. et al. (2009). Ising-like dynamics in large-scale functional brain networks. *Phys Rev E, 79(6)*, doi: 10.1103/PhysRevE.79.061922

[2] *Ising model*. Retrieved from `http://en.wikipedia.org/wiki/Ising_model`

[3] *Monte carlo method*. Retrieved from `http://en.wikipedia.org/wiki/Monte_Carlo_method`

[4] Fricke, T. (2006). Monte carlo investigation of the ising model. Retrieved from `http://www.physics.ohio-state.edu/~braaten/statphys/Ising_MatLab.pdf`

Appendix: Code

Listing 1: main.m

```
x = [2 2.3 3]; % Data points
m = 1000; % Number of calculations to average
L = 200; % Grid size: LxL
j = 20; % Number of flips to perform per cycle

J = 1;
K = 1;

%%%%%%%%%%

[sx, sy] = size( x );

E = zeros( sy, 1 );

figure( 1 );
p = plot( x, E );
xlabel( 'Temperature' );
ylabel( 'Equilibrium Energy/Site' );
drawnow;

n = 1;

for i = x

        Et = 0;

        Et_temp = zeros( m, L, L );

        for k = 1:m

                [Er, A] = runenergy( L, i, j, 0 );

```

```
33              Et = Et + Er;
34              E( n ) = ( Et / k ) ./ L^2;
35              set( p, 'YData', E );
36              drawnow;
37
38              disp( ['Cycle #' num2str( k ) ' (T = ' num2str( i ) ') complete. E = ' num2str( E( n
                    ) )] );
39
40              Et_temp( k, :, : ) = A;
41
42          end
43
44          n = n + 1;
45
46          r = correlationdensity( Et_temp );
47
48          figure( 1 + n );
49          surf( r, 'EdgeColor', 'none' );
50          title( ['T = ' num2str( i )] );
51          xlabel( 'x' );
52          ylabel( 'y' );
53
54          figure( 1 + n + sy );
55          imagesc( r );
56          set( gca, 'YDir', 'normal' );
57          title( ['T = ' num2str( i )] );
58          xlabel( 'x' );
59          ylabel( 'y' );
60
61  end
```

Listing 2: runenergy.m

```
function [ret, P] = runenergy( L, T, n, paintGraph )

K = 1;

A = round( rand( L, L ) ) * 2 - 1;

for i = 1:n

        for j = 1:L

                for k = 1:L

                        delta = energy2( A, j, k, L ) .* -2;

                        if( delta <= 0 )

                                A( j, k ) = A( j, k ) * -1;

                        else

                                if( rand( 1 ) <= exp( -1 .* delta / ( K * T ) ) )

                                        A( j, k ) = A( j, k ) * -1;

                                end

                        end

                end

        end

        if( paintGraph )

```

```
35                figure( 2 );
36                img = image( ( A + 1 ) * 128 );
37                xlabel( 'x' );
38                ylabel( 'y' );
39                set( gca, 'YTickLabel', [], 'XTickLabel', [] );
40                axis square;
41                colormap bone;
42                drawnow;
43
44        end
45
46 end
47
48 ret = energy2( A, -1, -1, L );
49 P = A;
```

Listing 3: energy2.m

```
function ret = energy2( A, x, y, L )
% Compute energy at a position, or for entire grid.

J = 1;

if( x == -1 && y == -1 )

        % This is inefficient for calculating energy at a single grid point.
        s = circshift( A, [0 1] ) + circshift( A, [0 -1] ) + circshift( A, [1 0] ) + circshift( A,
            [-1 0] );
        ret = sum( sum( -1 * J * ( A .* s ) ) );

else

        s = 0;

        if( x == L )

                s = s + A( 1, y );

        else

                s = s + A( x + 1, y );

        end

        if( x == 1 )

                s = s + A( L, y );

        else

                s = s + A( x - 1, y );

```

```
34          end
35
36          if( y == L )
37
38                  s = s + A( x, 1 );
39
40          else
41
42                  s = s + A( x, y + 1 );
43
44          end
45
46          if( y == 1 )
47
48                  s = s + A( x, L );
49
50          else
51
52                  s = s + A( x, y - 1 );
53
54          end
55
56          ret = -1 * J * ( A( x, y ) .* s );
57
58  end
```

Listing 4: correlationdensity.m

```
function r = correlationdensity( tab )

[st, sx, sy] = size( tab );

A = zeros( sx, sy );

for i = 1:sx

        for j = 1:sy

                sig_i = mean( tab( :, i, : ) .^ 2 ) - mean( tab( :, i, : ) ) .^ 2;
                sig_j = mean( tab( :, :, j ) .^ 2 ) - mean( tab( :, :, j ) ) .^ 2;

                sig_i = permute( sig_i, [3 2 1] ).';

                dp = dot( sig_i, sig_j );

                si = permute( tab( :, i, : ), [1 3 2] );
                sj = tab( :, :, j );

                a = mean( dot( si, sj, 2 ) ) - dot( mean( si ), mean( sj ) );

                r( i, j ) = a ./ dp;

        end

end
```