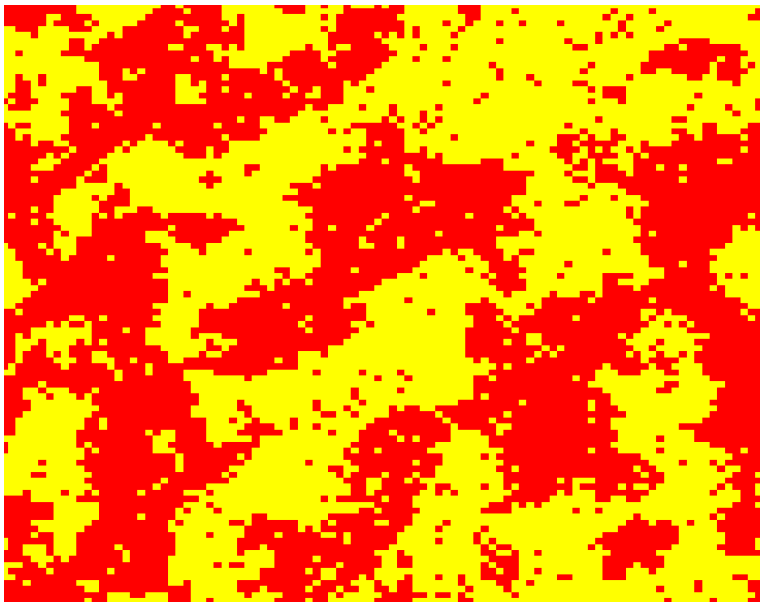
Simple numerical analysis of 2D Ising model

Stefano Boccelli



http://boccelliengineering.altervista.org

Fall, 2015

1 Intro

This document is about a little script I've made one afternoon just to plot some colors. No guarantees at all, not even that what I wrote is correct.

In this document I'd like to talk about a naive implementation of a Metropolis algorithm for simulating the Ising model in 2 dimensions. I'll show a few results for a 100×100 spins lattice, whose values can be only ± 1.

In the end I'll compare the computed magnetization to the analytical results due to Onsager.

Contents

2 Algorithm

The numerical solution is based on a Metropolis algorithm. Here are the main steps and a few Matlab/Octave commands:

Step 1 - Initial conditions

Creating a $N \times N$ lattice, with randomly chosen spins, or a lattice of parallel spins. For a better thermalization, one should better choose random spins when above the critical point and parallel spins if underneath.

```
% Spin da -1 oppure 1
spinMat = round(rand(N,N))*2 - 1

% Spin tutti su
spinMat = ones(N,N)
```

Step 2 - Cycle

A cycle is started: at every step a lattice position is randomly chosen, it's spin is flipped and the energy variation is computed. The cycle keeps going until some chosen variables reach a steady state.

Step 2.1 - Energy computation

Well, a random spin on the lattice is choosen and flipped. If no external field is applied, energy can be written: $E = -J \sum_{p.v.} s_j s_i$, so $\Delta E_i = 2J s_i \sum_{p.v.} s_j$.

Step 2.2 - Probability computation

This step attributes to out "flipping action" the real probability. One must accept the new configuration with a probability linked to the caused energy variation.

The probability associated to the flipping is given by the Boltzmann factor: $P_{flip} = e^{-\beta \Delta E}$. The spin is actually flipped if:

```
if rand() < P_flip  % ESEGUI INVERSIONE
  spinMat(ii,jj) = -spinMat(ii,jj);
  E = E + dE;
end
```

Note that for a quicker convergence, Metropolis methods usually simply accept the flipping if $\Delta E < 0$ while compute the probability if it's bigger than 0.

3 Details on the numerical method

3.1 Boundary conditions

I've imposed periodic BCs, implemented through the `module(x)` function: Here's an Octave/Matlab implementation of the "module" function:

```
function j = modulofunction(i,N)
  if (i > N)
    j = i - N;
  elseif (i < 1)
    j = N + i;
  else
    j = i;
  end
return
```

3.2 Initial conditions

One can use as initial conditions a lattice of randomly choosen spins (some are up and some others down), or a uniform lattice. As stated before, choosing smartly the ICs makes convergence a lot easier and makes thermalization faster: one should use a uniform grid if trying to compute a state under the critical temperature, while random spins are better for simulations above the critical point.

Another possibility, that shows useful when problems are big, is starting a simulation with the equilibrium condition that was obtained for a similar temperature (maybe in the previous step).

3.3 Convergence

I've assumed the method was converged when the total energy of the spin lattice and the total spin did reach a reasonably stationary value. Total spin (aka magnetization) is computed with the command:

```
SpinTot = sum(sum(spinMat));
```

Here's a messy convergence plot for Total Energy and Total Spin at a certain temperature. Note that since the system is 100×100, the maximum value for the Total Spin is $1E+03$, thus fluctuations are smaller than 10%. I've then choosed to average the last half of the values.

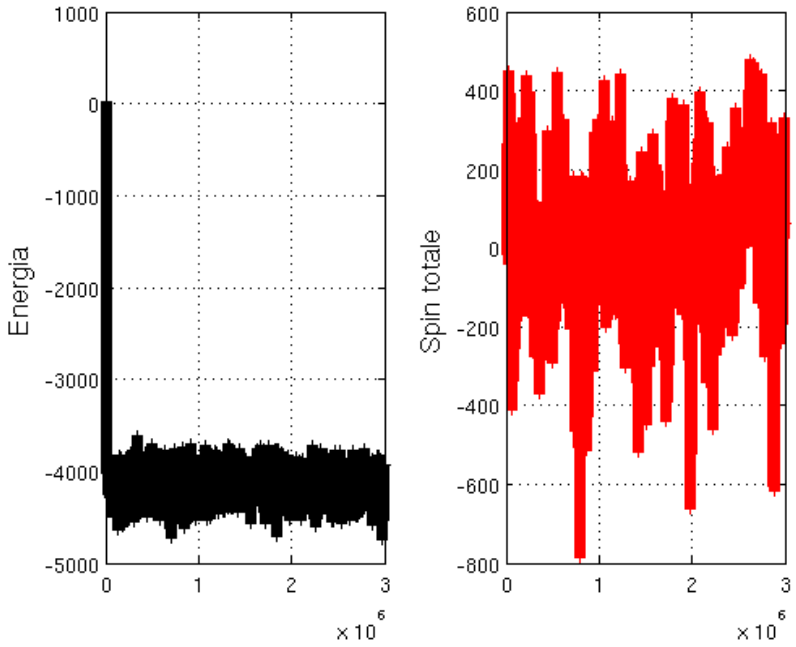


Figure 1: Convergence plot for $T/Tc = 2.21$
Note that starting from a random (disordered initial condition), convergence is quite quick and the Total Spin soon approaches the value 0.

4 Results

4.1 Near the critical point

As we know from both theory and experiments, while approaching the critical point, fluctuations become bigger and bigger and correlation lengths diverge. Here is shown the 100×100 spin lattice in some conditions approaching the critical Temperature.

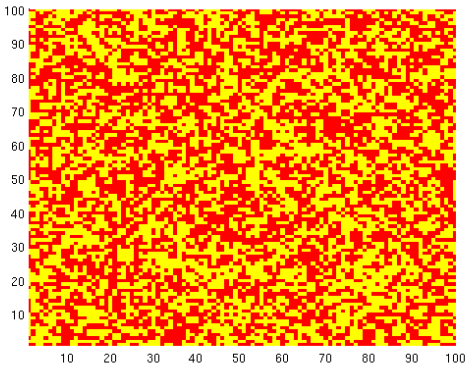


Figure 2: $T/Tc = 3.54$

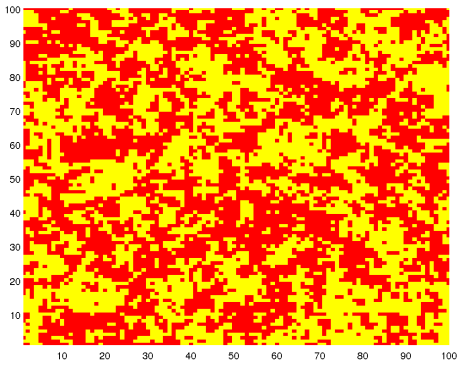


Figure 3: $T/Tc = 1.26$

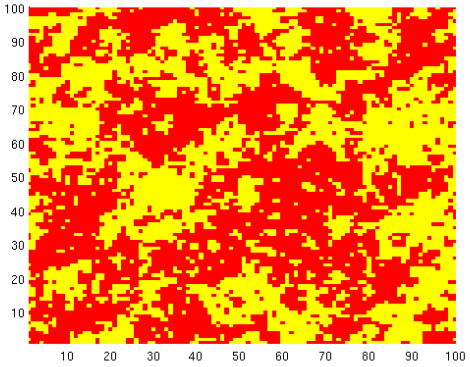


Figure 4: $T/Tc = 1.11$

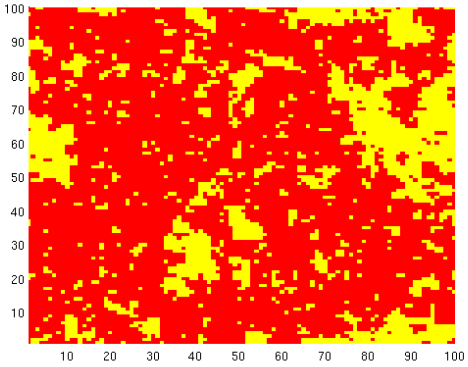


Figure 5: $T/Tc = 1.02$

4.2 Metropolis vs Onsager

Magnetization was computed for some temperatures across the critical point and is here plotted superimposed to a line showing the analytical divergence power law ($\beta = 0.125$, by Onsager). The critical temperature for 2D Ising can be demonstrated to be around 2.26 if $k_B = 1$ and the Heisenberg constand is $J = 1$.

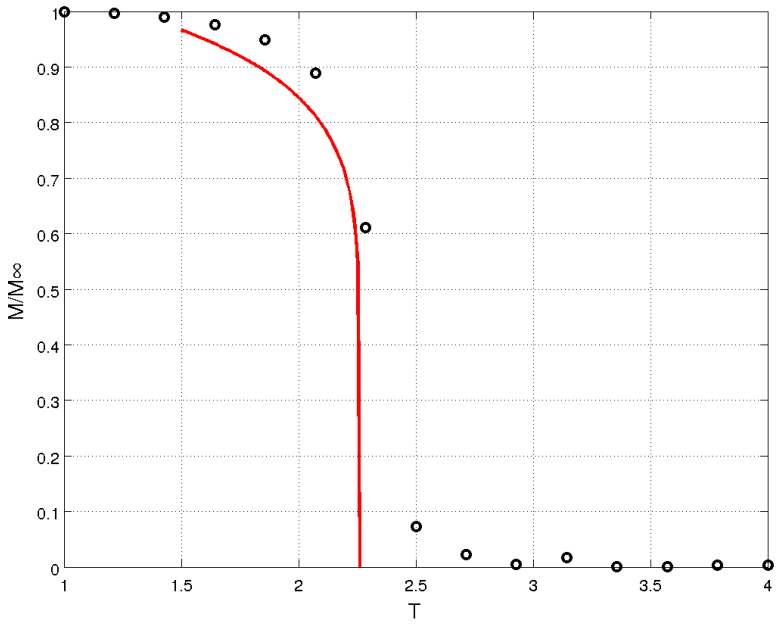


Figure 6: Spontaneous Magnetization vs Temperature for 2D Ising model. Solid line: power law, exponent $\beta = 0.125$. Markers: numerical results

5 Matlab script

Here's a script, that runs on Matlab 2014. Sorry, comments are in Italian.

```
% ##############################################################

close all
clear
clc

global spinMat xMat yMat N

% Various parameters
N = 100;   % spins per side
```

```
NmaxIter =  5000000;

kB   = 1;
T    = 8; % Temperature
beta = 1/( kB*T );

J = 1;


% Creating spin and coordinates matrix
% coordinates are not used anymore, anyway..
spinMat = round(rand(N,N))*2-1;
%spinMat = ones(N,N);
[xMat,yMat] = meshgrid([1:N],[1:N]);
E = 0; % ARBITRARY!
Evect = zeros(NmaxIter,1);
Svect = zeros(NmaxIter,1);

% Plotting Initial condition
plotfunction(0)

plotNum   = 0;
plotCount = 0;

for it = 1:NmaxIter

  % Flipping a random spin
  rspin = round( (N-1)*rand(2,1) ) + 1; % this gives a random index
                                        % inside the matrix

  spinVal = spinMat(rspin(1),rspin(2));
  deltaE  = deltaEcalc(rspin, J);


%  % Probability of change
%  % this commented out accepts the flipping a priori if DeltaE < 0
%  if deltaE < 0  % THEN FLIP!
%
%    E = E + deltaE;
%    spinMat(rspin(1), rspin(2)) = -spinVal;
%
```

```
%  else           % LET'S SEE IF HE'S LUCKY!

    probFlip = exp(-beta*deltaE);

    if rand() < probFlip
      E = E + deltaE;
      spinMat(rspin(1),rspin(2)) = -spinVal;
    end

%  end

  % Saving Energy Value
  Evect(it) = E;

  % magnetization:
  Svect(it) = sum(sum(spinMat));

  % PLOT
  plotCount = plotCount + 1;

  if plotCount > round(NmaxIter/5)

    fprintf('Step %d of %d\n', it, NmaxIter)

    plotNum = plotNum + 1;
    plotCount = 0;
    plotfunction(plotNum)

  end

end

% Energy Convergence plot
figure
subplot(1,2,1)
plot([1:NmaxIter],Evect(1:end),'+k')
grid on
ylabel('Energia','FontSize',14)

subplot(1,2,2)
plot([1:NmaxIter],Svect,'+r')
```

```
grid on
ylabel('Spin totale','FontSize',14)
% ############################################################
```

Function for plots

```
% ############################################################
function plotfunction(numero)

global spinMat xMat yMat

close all
h = figure;
hold on

imagesc(spinMat)
colormap(autumn)
xlim([1,size(spinMat,2)])
ylim([1,size(spinMat,1)])
print(h, '-dpng', ['./imgs/ising_',num2str(numero),'.png'])

pause(0.0000001) % without this it will plot only at the end

return
```