ISTANBUL TECHNICAL UNIVERSITY

FACULTY OF LETTERS AND SCIENCES

ADVANCED PHYSICS PROJECT

MONTE CARLO METHODS IN SPIN SYSTEMS

Mehmet Ali Anıl

Department: Department of Engineering Physics

Autumn 2011

ISTANBUL TECHNICAL UNIVERSITY

FACULTY OF LETTERS AND SCIENCES

ADVANCED PHYSICS PROJECT

MONTE CARLO METHODS IN SPIN SYSTEMS

Mehmet Ali Anıl

Department: Department of Engineering Physics

Autumn 2011

Summary

This work shows an implementation of Monte Carlo methods in statistical systems. A 2D system of Ising spins are simulated and the Ising Phase change is observed. Different systems are also simulated, such as a system of spins with 5 different energy levels, and a protein model are amongst those.

Contents

Summary ii

Nomenclature iv

List of Figures iv

1 Monte Carlo Methods 1
- 1.1 Monte Carlo Methods in Statistical Systems 3
  - 1.1.1 Metropolis Algorithm on a 2D Ising Chain 3
- 1.2 Metropolis Algorithm for an Arbitrary Statistical System 5
- 1.3 Ising Spins as an Example 6
- 1.4 Protein folding 8
- 1.5 The Code Used for this Project 9

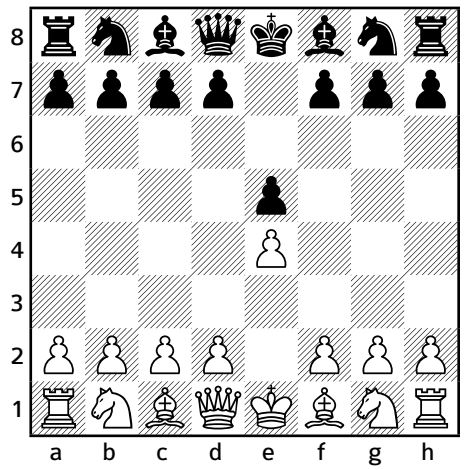
List of Figures

1.1 Randomly distributed 2D Ising spins 3
1.2 The dependence of magnetization M due to the temperature, β. . . 7
1.4 The dependence of magnetization M due to the temperature, β in 5-level spins. 8
1.6 Several chains that the algorithm converged. In these chains, $\beta \cong 3.9$, in hotter environments, the chains fail to satisfy the convergence criteria. 9
1.3 The final states of the 2D Ising spin systems for different β. 10
1.5 The final states of the modified 2D Ising spin systems for different β. 11

Chapter 1

Monte Carlo Methods

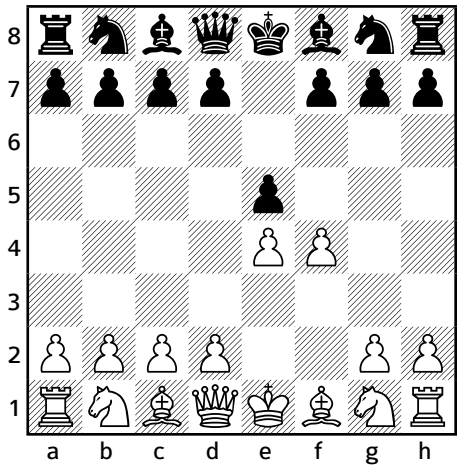
Monte Carlo methods account for a large set of methods that incorporate random number generation in solving a problem numerically. Monte Carlo algorithms are in wide use in many different search algorithms, especially in cases where minimum energy (or cost) is hard to define. In the case of a parameter optimization in a smooth energy landscape, finding a global minimum, or at least an exceptionally good local minimum is a well studied problem, and there are many ways to attack this problem. Problems arise when the parameter space is not continuous, or the energy landscape is not smooth, in which then small perturbations in initial conditions lead to different local minima. One example that might be given is a chess opening, **1 e4 e5**.



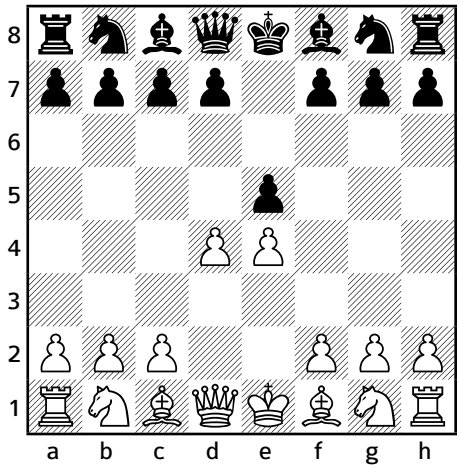
This is the **open game** opening, a mainstream opening in chess. We can think of the chess game as a statistical system where all configurations of the chess pieces are states that a game can be in. It is evident that a system can't jump from any state to another, for example, a bishop in white squares can't be in a black square [1] , but still, there are still a huge amount of positions (or states) available for the game to be

[1]Except for the exceptional case where a promoted pawn is chosen to be a second bishop of black tiles.

in and a game is actually how this state has evolved into a checkmate or a draw. In this particular game, the white has a many choices, for example, **2 f4 ZZ...** is called a **King's Gambit**, white offers a pawn in exchange of a positional advantage.



This one of the examples where another game or state, such as **1 e4 e5 2 d4...** seems to be a similar one, though for the player, this offers a very different set of opportunities than a King's Gambit would offer.



There isn't a straightforward way to reach the most advantageous state in a chess game, that's what makes it a game. But if this is the case, there is a hard time for a deterministic machine to make a move, since optimization algorithms such as gradient descent depend on the fact that small changes in parameters would lead to small changes in performance, it is evident this is not the case here. Or, one might argue rightly that our visual perception is a bad informant of a norm in all state space, since states here are not in a space, but are in a tree (or a graph) of recursively branching moves, all of the tips being checkmates of draws.
So an artificial intelligence [2] would go for choosing moves, walk the portion of the tree of states that is n moves away from the initial state, and calculate its advantage

[2]Let's assume that here that it is an AI that all previous knowledge of chess, openings, plays... are erased, an AI tabula rasa.

for every play. The deeper the AI goes in its search, better moves it will eventually find, but most of the time it takes a huge amount of time to traverse the tree of positions, so most game engines use a method called **Monte Carlo Tree Search**, in which the way that the tree is explored is randomized. The game tree grown in the memory with these random explorations are then used to make the best move available to the AI. This gives the AI a deeper search for a given amount of time, and a relatively good move. Randomized tree search is even more popular in games like Go, where the amount of moves available to the user is much greater.
So one doesn't lack motivation in order to incorporate random guessing in searching for an optimum state, especially in cases where the complexity is great enough to veil any deterministic ways to search for such minimum. Chess is not the perfect example here, but financial systems, biological systems, manufacturing variations are systems where Monte Carlo methods are proven effective.

1.1 Monte Carlo Methods in Statistical Systems

Another area where Monte Carlo methods are preferable is statistical systems, where thermal agitation can't be handled deterministically. The most popular algorithm in this genre is the **Metropolis–Hastings algorithm** [1], and it is the algorithm that this paper will focus on. First a particular example about Ising spins is presented, than the algorithm is generalized step by step.

1.1.1 Metropolis Algorithm on a 2D Ising Chain

Let us assume that there is a 2D spin glass of random spin distribution and size 50×50. Here, spins up and down are visualized as black and white squares.

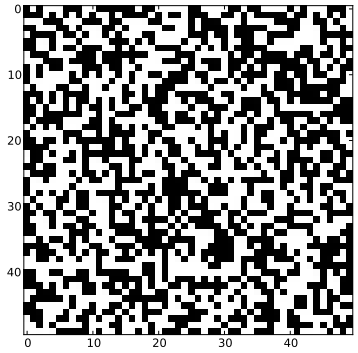


Figure 1.1: Randomly distributed 2D Ising spins. In this paper, spin systems will be visualized like this.

This configuration can be attributed to an energy value. If individual spins are

denoted as $\sigma_i \in -1, 1$, [3] then the Ising Hamiltonian can be loosely formulated in this way:

$$\mathcal{H} = -J \sum_{\langle ij \rangle} \sigma_i \sigma_j - h \sum_i \sigma_i \tag{1.1}$$

where σ_i is an individual spot on the 2D lattice, and the first sum is over spins where spin i and spin j are nearest neighbours.
So, if a spin is up, every spin that is down in its vicinity contributes $-J$ to the energy, and every spin that is up contributes J to its energy.
A non-degenerate system that is in thermal equilibrium with its surroundings has an probability of occupying a state with energy E_i of

$$p(E_i) = \frac{e^{-\beta E_i}}{Z(\beta)} \tag{1.2}$$

where $Z(\beta)$ is the partition function. In our case, Equation (**??**) is simpler:

$$p_1 = \frac{e^{-\beta E_1}}{e^{-\beta E_1} + e^{-\beta E_{-1}}} \tag{1.3}$$

$$p_{-1} = \frac{e^{-\beta E_{-1}}}{e^{-\beta E_1} + e^{-\beta E_{-1}}} \tag{1.4}$$

since there are only two states for a single spin to occupy. So, if one only observes this particular spin only from the fact that there is an ongoing thermal agitation, it will occupy the energetic state with probability p_1.
The Metropolis algorithm uses this fact. For a spin, if the rotated spin results in a lower energy configuration, this configuration is preserved. If the rotated spin results in a higher energy configuration, there is still a chance that this configuration is attained, since the thermal vibrations can contribute for that, and this probability is

$$p_{\text{flip}} = e^{-\beta \Delta E}$$

where ΔE is the energy change required to flip the spin. One can guess that such an algorithm will eventually reach down to states that flux of spins pumped up to an antisymmetric configuration will be equal to the flux of spins that lose energy by attaining a symmetric configuration. Here, β is a parameter that can be tuned, it is $1/k_B T$. When this equilibrium is reached, the macroscopic variables of the system is considered to be unchanging, and after the difference in consecutive calculations of the macroscopic variable is below some value, it can be used as is.

[3] I avoided double indices for the sake of simplicity

So if one writes the steps of the algorithm:

1. Create a random configuration of spins.
2. Do the following for Π times:
   (a) Choose a random spin.
   (b) Calculate the energy change, ΔE.
      i. If $\Delta E < 0$, then preserve the orientation of the spin.
      ii. If $\Delta E > 0$, then generate a random number $\alpha \in (0, 1)$
         A. If $\Delta E < \alpha$, then preserve the orientation of the spin.
         B. If $\Delta E > \alpha$, then flip the spin.
3. Calculate M , magnetization, if the value fo the magnetization is within error bounds, get out of the algorithm. If not, continue to do so until the error is within bounds.

This is how the algorithm is implemented for a 2D Ising spin configuration. The 3D case is rather easy to guess, it follows the same logic.

1.2 Metropolis Algorithm for an Arbitrary Statistical System

The aim in this project is not to make numerical calculations for various systems, such as 1D, 2D, 3D Ising chains. I'd rather like to generalize the Metropolis Algorithm in such a way that it can be implemented for various other statistical systems also. The aim is to code a modular Metropolis engine that a set of statistical systems can be implemented to the code, by using a backbone that is applicable to all of them.
I will explain the generalization procedure by applying it to the Ising case. What Metropolis algorithm does is the following. It takes a randomized **configuration**, and for a single **element** of the statistical ensemble. hen either from a function that calculates the energy difference, or by merely calculating the energies of the configurations and subtracting them, an energy difference ΔE is calculated. Then, the probabilities for every state this individual can be in is calculated, from a temperature and energy dependent **probability distribution**. Then, a new configuration is selected from the pool, depending on the probabilities of them. This loop is initiated for Π times, after macroscopic variables are calculated, and after these microscopic variables are settled, the algorithm quits.
With this reasoning, we can think of the following objects and functions for an arbitrary algorithm.

1. Parameters (β , J , k)
2. An ensemble definition ($N \times N$ spin system)
3. An element definition (A spin, up or down)
4. An energy definition with respect to some parameters (Hamiltonian (??))
5. A probability distribution with respect to some parameters (Boltzmann distribution)
6. Macro parameters to calculate from the ensemble (Magnetization)
7. If any, an external condition applied via changing parameters. (There were none)

By generalizing this, a Metropolis algorithm may be used for any statistical system with considerable modularity. For example, a gas can also be simulated with this approach.
So, this approach can be implemented as the following. There can be classes **Ensemble** and **Element** , in which the ensemble houses a list of elements.

1.3 Ising Spins as an Example

In order to exemplify this modular approach, a system of 20×20 spins were exposed to this algorithm. The class of *SpinGlass* houses all functions that incorporate for every action taken in the ensemble. Methods defined in the class will calculate the energy or the energy difference, the element list, the parameters, and the whole process of selecting an element, calculating the energy difference for it to be rotated, and proceed accordingly.
After each parameter change, which is triggered with a condition satisfied, the magnetization M, is calculated. The temperature β is swept from 0.1 to 1.05 and the following are the plot of magnetization versus the temperature. In this trial, the values for the coefficients are selected to be $J = 1$, $h = 0$, for simplicity.

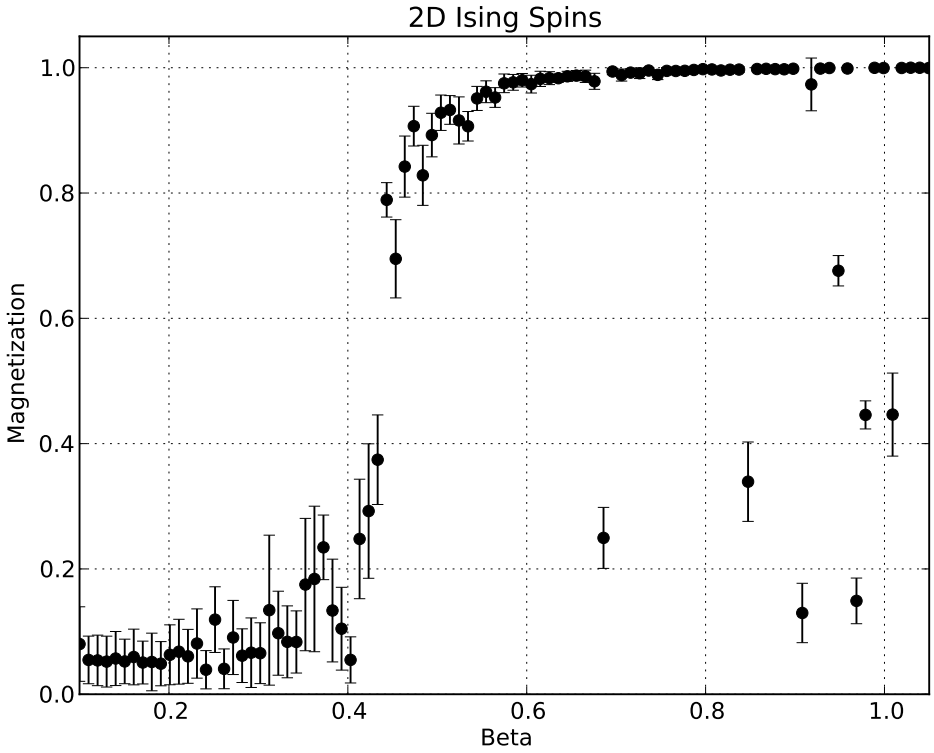


Figure 1.2: The dependence of magnetization M due to the temperature, β. The error bars correspond to the standard deviations and the dots correspond to the mean of the data set taken for that β value. The deviations from the orderly phase are from the fact that the standard deviation may turn out to be low though the algorithm hadn't converged.

The phase change at around $\beta = 0.44$ is evident. The exact calculation of the critical temperature was achieved by analytical means by Onsager. [2] The critical temperature is calculated via this condition

$$2\tan^2(2\beta_c J) = 1 \tag{1.5}$$

Which in our case predicts a critical temperature of $\beta_c = 0.4405$ in agreement with the results.

A very simple change that gives one the chance when this modular approach is taken is that when the elements, the definition of the spins and energy are changed, the whole code remains intact, since the core of the code applied to any system with an ensemble and energy definition.

One may change the definition of a spin, a spin that may take one the values from $\{-1, -0.5, 0, 0.5, 1\}$. This results in a different system, and one should update the Metropolis algorithm such that it encompasses systems of multiple energy levels. The algorithm may be updated such that if for one of the spins, there exists a lower

energy configuration, than the system takes the state with the lowest of those energies. If the system is in the lowest energy configuration, then the next state it is going to be in is determined by the probabilities determined by the Boltzmann distribution as it was in the case of Ising spins. This system is ran within a grid of 15×15, the macroscopic parameters are calculated accordingly.

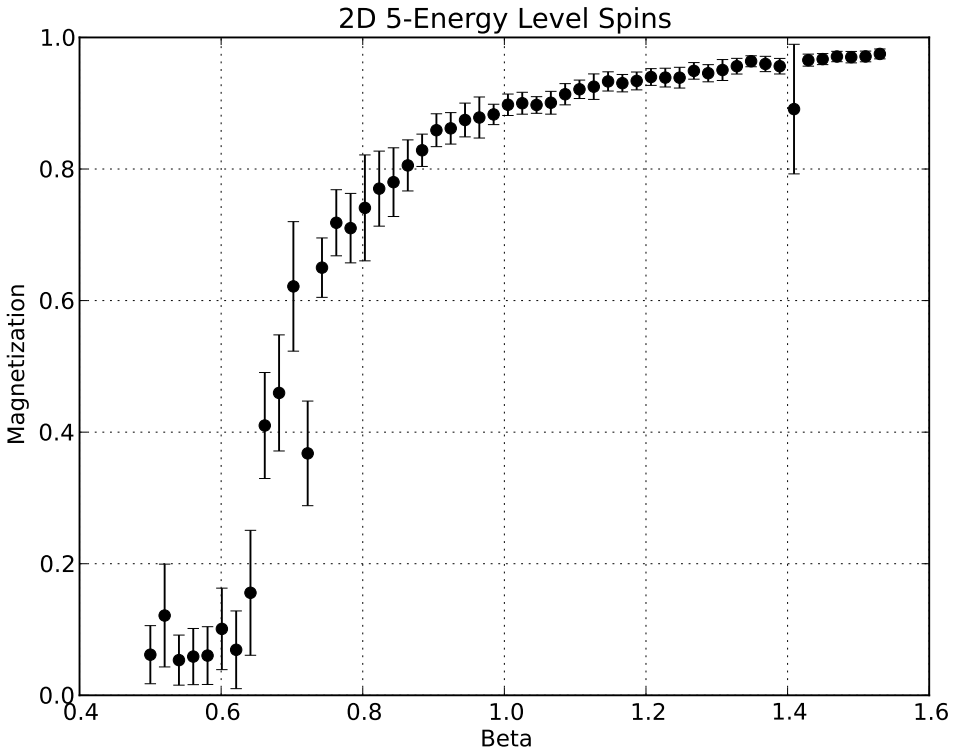


Figure 1.4: The dependence of magnetization M due to the temperature, β. The error bars correspond to the standard deviations and the dots correspond to the mean of the data set taken for that β value. The deviations from the orderly phase are from the fact that the standard deviation may turn out to be low though the algorithm hadn't converged.

The phase change is more gradual, and now, the critical temperature is higher, $\beta_c \approxeq 0.7$.

1.4 Protein folding

In order to get out of the world of spins, one might come up with different systems, such as a system where a protein is made out of kinks that interact with Coulomb potential. The simulation for such system is in progress, but didn't finish at the time when this report was being written. Though, the process is similar. The Monte

Carlo algorithm is changed in a way that the ensemble is a long 30-piece chain of pieces of same length, with corners randomly selected to be positively charged, negatively charged, or neutral. The perturbation is a small change in angle between two consecutive chains. The energy is the Coulomb potential of this arrangement of charges. Though the proteins could be visualized to be folding, some problems arise from the floating point precision.

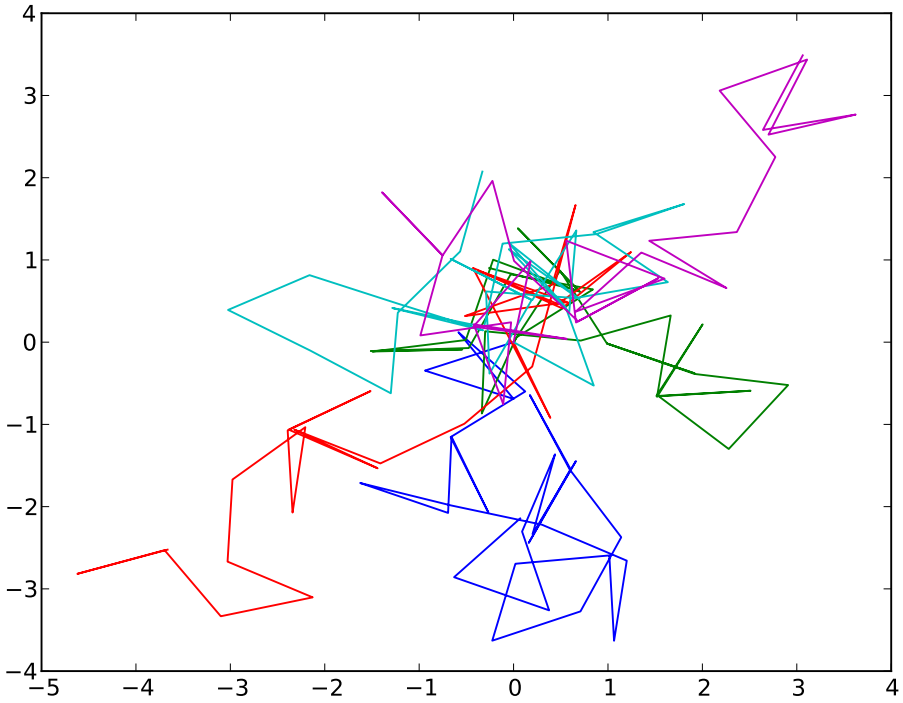


Figure 1.6: Several chains that the algorithm converged. In these chains, $\beta \cong 3.9$, in hotter environments, the chains fail to satisfy the convergence criteria.

1.5 The Code Used for this Project

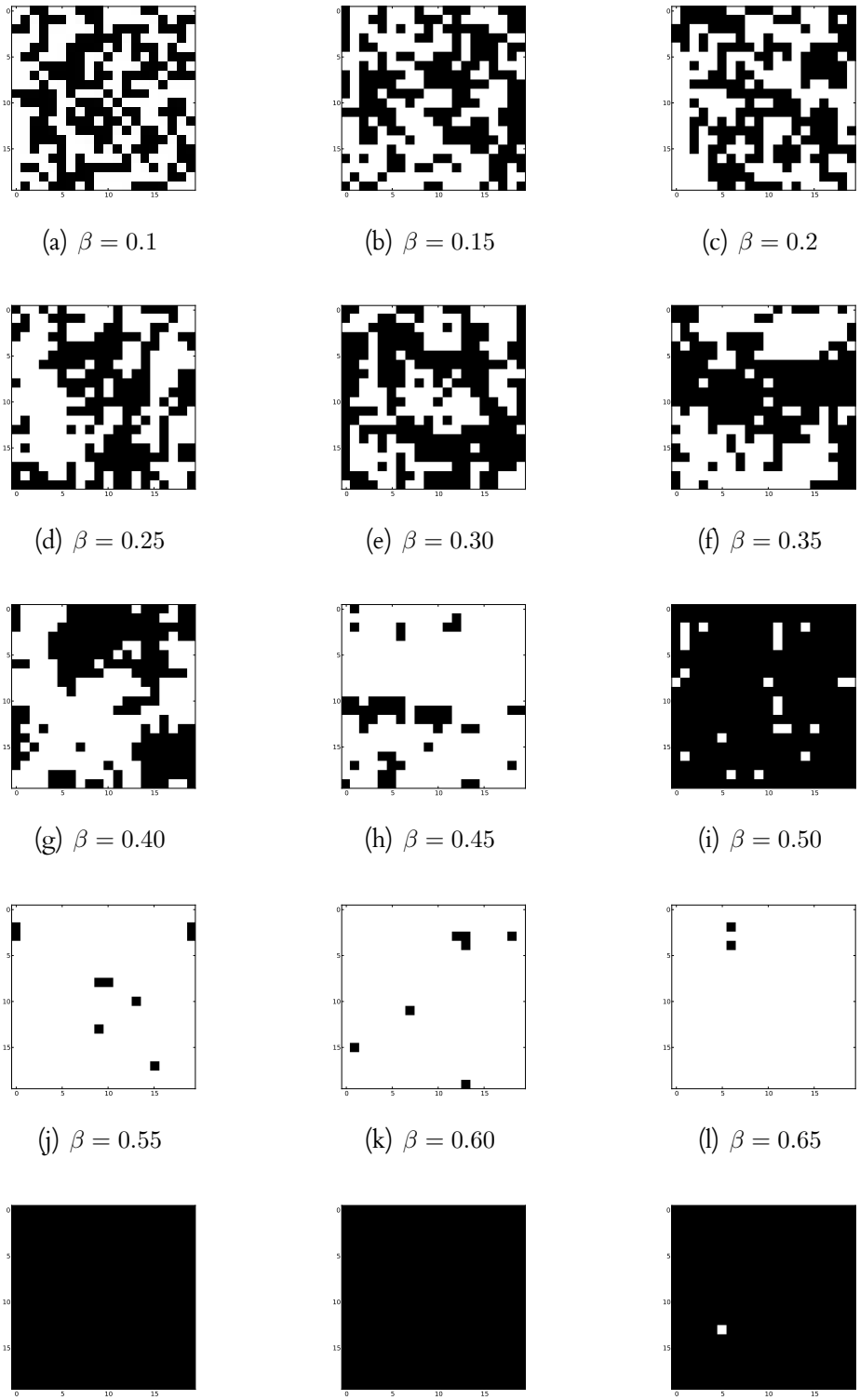
The code used in this project is named as *Monty Carlo* and is available as Git repository. In order to have a copy, contribute to the code, please copy the repository with the command:

```
1 git clone git://github.com/mehmetalianil/monty-carlo.git
```

This will create a folder and put the source code as it was on my pc when I commited my changes to the project. When there is an update, this update can be downloaded via the command:

```
1 git pull origin
```

If one has made a change to the source code on his/her computer, than that copy becomes a fork of my project, in order to maintain the simultaneity of these two copies, these changes must be merged together. Contact me via *mehmet.ali.anil@ieee.org* if you want to contribute to the code.



(a) $\beta = 0.1$ (b) $\beta = 0.15$ (c) $\beta = 0.2$

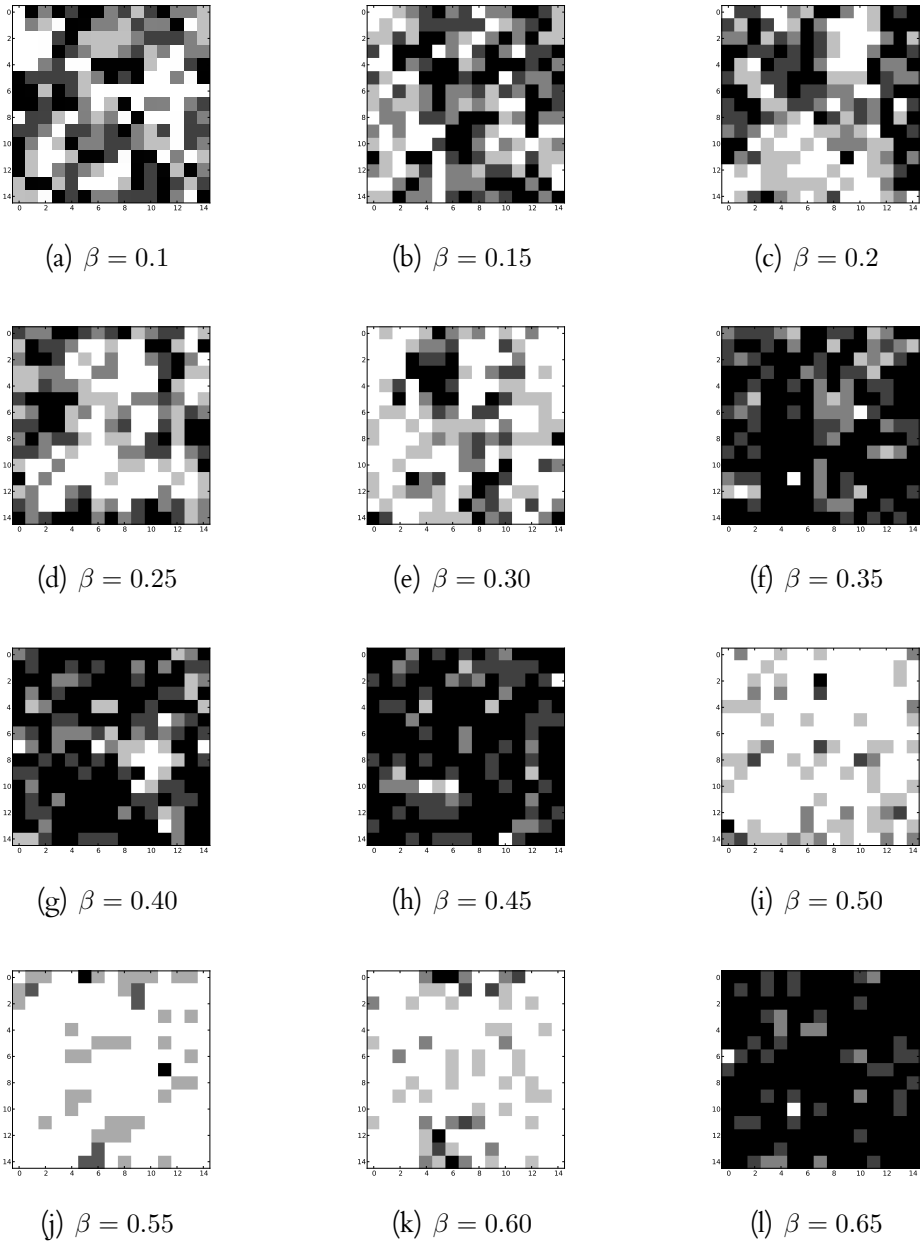
(d) $\beta = 0.25$ (e) $\beta = 0.30$ (f) $\beta = 0.35$

(g) $\beta = 0.40$ (h) $\beta = 0.45$ (i) $\beta = 0.50$

(j) $\beta = 0.55$ (k) $\beta = 0.60$ (l) $\beta = 0.65$

(m) $\beta = 0.70$ (n) $\beta = 0.75$ (o) $\beta = 0.80$

Figure 1.3: The final states of the 2D Ising spin systems for different β.



(a) $\beta = 0.1$ (b) $\beta = 0.15$ (c) $\beta = 0.2$

(d) $\beta = 0.25$ (e) $\beta = 0.30$ (f) $\beta = 0.35$

(g) $\beta = 0.40$ (h) $\beta = 0.45$ (i) $\beta = 0.50$

(j) $\beta = 0.55$ (k) $\beta = 0.60$ (l) $\beta = 0.65$

Figure 1.5: The final states of the modified 2D Ising spin systems for different β.

Bibliography

[1] A. W. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[2] Lars Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Phys. Rev.*, 65:117–149, Feb 1944.