Monte Carlo Simulation of the XY Spin-Glass Model on a 3D Cubic Lattice

Mattia Liguoro

Abstract

In this project, the main properties of the XY spin-glass model on a 3D cubic lattice were investigated using a Monte Carlo simulation. In particular, three Monte Carlo moves were implemented: the standard Metropolis move, the over-relaxation technique, and finally the parallel tempering method, with a discussion and classification of the speed-ups achieved through the various techniques.

1 Introduction to the Model

1.1 Phenomenology

Spin glasses represent a class of magnetic systems characterized by disordered interactions and frustration. They were originally discovered through experimental observations of alloys such as CuMn or AuFe, in which atoms with magnetic impurities (i.e., non-zero magnetic moments from valence electrons) interact via the RKKY mechanism (long-range oscillatory interactions mediated by conduction electrons). These systems are typically modeled using the Edwards-Anderson (EA) model or the Sherrington-Kirkpatrick (SK) model.
Due to the complex nature of the interactions among the spins in the sample, it is difficult to find a configuration that allows local minimization of the free energy. As a consequence of the system's disordered and frustrated nature, so-called "topological vortices" emerge. To understand their origin, let us briefly explore the key features of spin glasses:

1. **Disorder**: spin glasses exhibit random interactions between spins, which may be ferromagnetic (favoring parallel alignment) or antiferromagnetic (favoring antiparallel alignment). Such disorder inhibits the development of uniform long-range magnetic order in the system.

2. **Frustration**: due to the mixture of ferromagnetic and antiferromagnetic interactions, it is not always possible to simultaneously satisfy all local interactions. This leads to configurations in which the spins arrange themselves to minimize the overall energy, but without a simple global order.

3. **Topology**: in two- or three-dimensional systems, disorder and frustration can lead to configurations in which the spins follow circular or vortex-like patterns around a central point. This is analogous to vortices in fluids, where the flow follows a circular path.

This behavior can be visually understood through the two figures below:

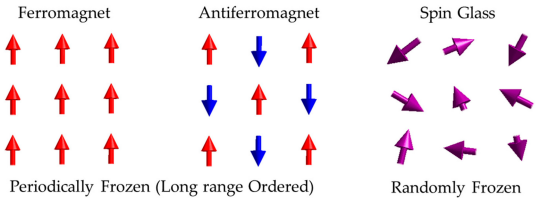

Figure 1: Random interactions in spin glasses

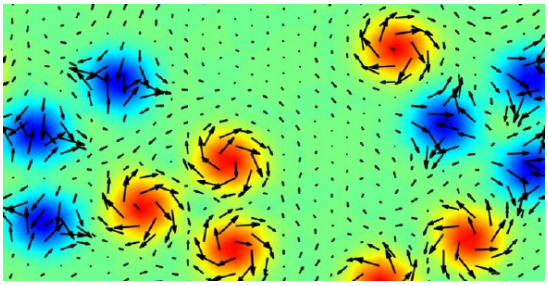

Figure 2: Topological vortices in spin glasses

1.2 Mathematical Treatment

From a theoretical point of view, it is often simpler to compute quantities of interest using the Edwards-Anderson (EA) approach rather than the SK one. With the EA approach, it suffices to consider a regular arrangement of spins and random nearest-neighbor interactions. In contrast, in the SK model, the spins are arranged randomly and the interaction is a fully connected graph (i.e., all spins interact with all others, so there is no notion of spatial proximity). Therefore, in this project we adopt the EA approach. First, we explicitly write down the Hamiltonian:

$$H = -\frac{1}{2}\sum_{\langle i,j\rangle} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j \tag{1}$$

where the sum runs over nearest neighbors, and J is the coupling matrix between nearest neighbors. The $\mathbf{S}_i$ are two-component spins (x and y) with unit norm ($||\mathbf{S}_i|| = 1$). Periodic boundary conditions (PBC) are imposed. The J_{ij} are independent stochastic variables drawn from a normal distribution with mean 0 and variance 1, $\mathcal{N}(J_{ij}; \mu = 0, \sigma^2 = 1)$.
This model is also known as the "random-bond model", in contrast to the SK model, which is referred to as the "random-site model". The reason is intuitive.
In spin-glass models, in addition to the usual thermal average denoted by $\langle\cdot\rangle$, it is necessary to introduce an additional average: the disorder average, denoted by $[\cdot]_J$.
Why is this new averaging needed? Because a single sample corresponds to a specific realization of the disorder, so, to conduct a proper statistical analysis, one must consider all possible configurations of the system, including all realizations of the interactions—and thus of the disorder J.

$$\mathcal{O} \equiv [\mathcal{O}(J)]_J = \int dP[J]\mathcal{O}(J) \tag{2}$$

Unlike classical ferromagnetic (or antiferromagnetic) models, in spin glasses it is not straightforward to identify the lowest energy state at a fixed temperature. The free energy landscape contains many local minima, often comparable in energy to the global minimum. A pictorial view of this situation is shown in the figure below:

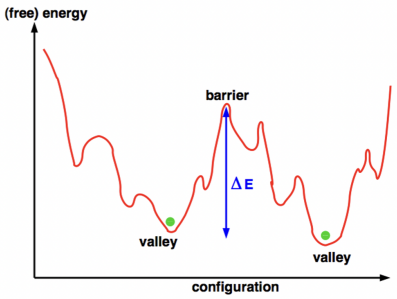

Figure 3: Pictorial representation of the free energy landscape in spin glasses

Of course, this applies only below the critical temperature. Above the critical temperature, the energy barriers vanish and only one valley remains, corresponding to the zero-magnetization state. At high temperatures, thermal fluctuations dominate and no order persists.
Since the system has access to many stable configurations, this leads to a breaking of ergodicity.
So how can we determine whether a disordered system exhibits long-range order?
It is clear that the standard order parameter used in ferromagnetic systems is not applicable here:

$$m = \frac{1}{N}\sum_i \langle s_i \rangle \tag{3}$$

So, what approach can we take? We can introduce a parameter that quantifies the similarity between two configurations (or, more generally, two systems). This is the Edwards-Anderson parameter, defined as:

$$q^{\mu\nu} = \frac{1}{N}\sum_i S_i^{\mu(1)} S_i^{\nu(2)} \tag{4}$$

Here, μ and ν denote the spin components (in the XY model, x and y), and (1) and (2) refer to two replicas with the same realization of the disorder. What information does $q^{\mu\nu}$ provide?

$$q^{\mu\nu} = \begin{cases} 1 & \text{if (1) and (2) are identical} \\ -1 & \text{if (1) and (2) are anticorrelated} \\ 0 & \text{if (1) and (2) are completely uncorrelated} \end{cases}$$

By studying $q^{\mu\nu}$ in momentum space, we can generalize it and obtain the following $\mathbf{k}$-dependent form:

$$q^{\mu\nu}(\mathbf{k}) = \frac{1}{N}\sum_i S_i^{\mu(1)} S_i^{\nu(2)} e^{i\mathbf{k}\cdot\mathbf{R_i}} \tag{5}$$

where $\mathbf{k}$ must satisfy periodic boundary conditions, and $\mathbf{R}_i$ is the position vector of the i-th spin with respect to the origin.
From this we can define the wave-vector-dependent spin-glass susceptibility:

$$\chi_{SG}(\mathbf{k}) = N\sum_{\mu\nu}\left[\left\langle |q^{\mu\nu}(\mathbf{k})|^2 \right\rangle\right]_J \tag{6}$$

Finally, the spin-glass correlation length is defined as:

$$\xi_L = \frac{1}{2\sin(k_{\min}/2)}\left[\frac{\chi_{\mathrm{SG}}(0)}{\chi_{\mathrm{SG}}(\mathbf{k}_{\min})} - 1\right]^{1/2} \tag{7}$$

where $\mathbf{k}_{\min} = \left(\frac{2\pi}{L}\right)(1,0,0)$.

2 Monte Carlo Simulation

2.1 Monte Carlo Moves: Metropolis, Over-relaxation, and Parallel Tempering

In the previous section, we laid out the theoretical and phenomenological foundations for understanding this model. We now aim to develop an algorithm for computing thermodynamic quantities of interest to study the system.
To achieve this, we employed a Monte Carlo simulation. Essentially, the simulation revolves around three types of Monte Carlo moves:

1. **Metropolis**: a spin is randomly selected from the lattice, and then rotated by an angle $S'_\theta = S_\theta + \xi\ \Delta\theta_{\max}$, where ξ is a random number drawn from a uniform distribution between -1 and 1, and $\Delta\theta_{\max}$ is a maximum angle chosen so that the acceptance rate lies between 30% and 50%. The move is accepted with probability:

$$p(S_\theta \to S'_\theta) = \min\left\{1, e^{-\beta\Delta E}\right\} \tag{8}$$

   where $\Delta E \equiv E_f - E_i$. This is by far the most computationally expensive move, as the energy must be calculated at every trial move, making it particularly costly for large systems.
   In practice, how is this algorithm implemented? If the system moves to a lower-energy state ($\Delta E < 0$), the move is accepted. Otherwise, if $\Delta E > 0$, the move is accepted with probability $\exp\{-\beta\ \Delta E\}$.

2. **Over-relaxation**: this move drastically reduces equilibration time. What does it involve? Essentially, it reflects a spin with respect to the local field that it feels.
   To define this move, we first calculate the local field:

$$\mathbf{H}_i = \sum_j J_{ij}\ \mathbf{S}_j \tag{9}$$

   In our case, the summation is limited to nearest neighbors.
   The actual move is then performed as:

$$\mathbf{S}'_i = -\mathbf{S}_i + 2\frac{\mathbf{S}_i \cdot \mathbf{H}_i}{\mathbf{H}_i^2}\mathbf{H}_i \tag{10}$$

   This move can be shown to conserve energy (hence also known as a "microcanonical move") and preserves the spin norm ($||\mathbf{S}'_i|| = ||\mathbf{S}_i|| = 1$).
   But if the energy doesn't change, how does this help with equilibration? The idea is to explore the configuration space as quickly as possible. This move allows faster transitions between configurations compared to Metropolis, as it requires no energy checks.

3. **Parallel tempering**: this move is essential to prevent the system from getting trapped in local minima of the free energy, which would otherwise compromise ensemble averages. It can be viewed as a generalization of the Metropolis move. The idea is to simulate N_T copies of the system in parallel, each at a different temperature. After a certain number of Metropolis and Over-relaxation steps, a Parallel Tempering move is attempted: a nearest-neighbor swap of configurations between two adjacent temperatures ($T_i \leftrightarrow T_{i+1}$). The acceptance probability is:

$$p(T_i \leftrightarrow T_{i+1}) = \min\left\{1, e^{\Delta\beta\Delta E}\right\} \tag{11}$$

   where $\Delta\beta \equiv 1/T_{i+1} - 1/T_i$ and $\Delta E \equiv E(T_{i+1}) - E(T_i)$.
   Typically, our system of interest will be the one at the lowest temperature $T_{\min}$ (i.e.,

system 0 in the simulations shown later), while the remaining replicas assist in promoting convergence at low temperatures.

Having defined the elementary moves, we now consider how to combine and order them. Whenever we refer to an "Over-relaxation move" or "Metropolis move," we implicitly mean a volume sweep — i.e., L^3 individual updates. Specifically, Over-relaxation moves involve updating all spins sequentially using equation (10), whereas Metropolis moves attempt to update L^3 spins (so most spins are updated). In Parallel Tempering, configuration exchanges are proposed between systems at adjacent temperatures T_i and T_{i+1}, for $i = 1, 2, \ldots, N_T - 1$.
A single "Monte Carlo move" is then defined as:

- $\times 10$ Over-relaxation sweeps
- $\times 1$ Metropolis sweep
- $\times 1$ Parallel Tempering move

This combination was found to provide the best performance for convergence and ensemble averaging. The results are consistent with those reported in the paper *Spin-Glass Transition of the Three-Dimensional Heisenberg Spin Glass. Campos, I., Cotallo-Aban, M., Martin-Mayor, V., Perez-Gaviro, S., & Tarancon, A.*, which states that an optimal number of OR steps is comparable to the system size L, in order to "allow the microcanonical wave to propagate through the entire system."
This strategy has been computationally verified to be the most efficient compromise for speeding up system equilibration.

2.2 Simulation Architecture

Let us now describe the structure of the program. A class named `sys` (system) was created. Within this class, several methods were implemented, including one for initializing the spins on a cubic lattice with random values (between 0 and 2π), and another for displaying the spin positions and values (mainly used for debugging purposes).
A method was also developed to initialize the J matrix according to the nearest-neighbor scheme. To that end, additional helper methods and structs were introduced to simplify the logic behind this initialization. Two auxiliary methods were created: one to find the coordinates of the six nearest neighbors of a given spin (accounting for PBC), and another to convert those coordinates into a single index using the relation:

$$n = kL^2 + jL + i \tag{12}$$

where i, j, k are the x, y, z components of the selected spin.
This indexing was necessary because, for interactions involving N neighbors, the full interaction matrix would be of size $L^3 \times L^3$. The idea is: select a spin (e.g., spin 0 at the origin), identify its nearest neighbors, and assign a new index to each using the above formula. As a result, for a fixed row (e.g., row 0), I have six non-zero entries drawn from a Gaussian distribution with mean 0 and variance 1 (as we assume nearest-neighbor interactions only).
Due to the inefficiency (albeit generality) of this algorithm for large systems, an auxiliary method was implemented for improved performance: `set_sparse_J()`. This method reduces the algorithmic complexity from $\mathcal{O}(L^6)$ to $\mathcal{O}(L^3)$. A matrix of structs named `J_sparse_element` was created, where each struct contains a variable `n` (index of the non-zero element) and `value` (its corresponding value). This sparse matrix has size $6L^3$. A possible representation of the

interaction matrix is:

$$\mathbf{J} = \begin{bmatrix} (n_{0,1}, \text{value}_{0,1}) & (n_{0,2}, \text{value}_{0,2}) & \cdots & (n_{0,6}, \text{value}_{0,6}) \\ (n_{1,1}, \text{value}_{1,1}) & (n_{1,2}, \text{value}_{1,2}) & \cdots & (n_{1,6}, \text{value}_{1,6}) \\ \vdots & \vdots & \ddots & \vdots \\ (n_{L^3-1,1}, \text{value}_{L^3-1,1}) & (n_{L^3-1,2}, \text{value}_{L^3-1,2}) & \cdots & (n_{L^3-1,6}, \text{value}_{L^3-1,6}) \end{bmatrix}$$

Once the J matrix was defined and the spins initialized, computing the local field became straightforward. Everything is based on equation (12) and on the inverse relation to retrieve i, j, k from a known index n:

$$k = \left\lfloor \frac{n}{L^2} \right\rfloor$$

$$j = \left\lfloor \frac{n \bmod L^2}{L} \right\rfloor$$

$$i = n \bmod L$$

Using these same relations and equation (1), the total energy of a given configuration was computed. Since each $\mathbf{S}_i$ has unit norm, the scalar product can be calculated simply using the cosine of the angular difference:

$$\mathbf{S}_i \cdot \mathbf{S}_j = \cos(\theta_i - \theta_j) \tag{13}$$

Next, methods were implemented to save and load spin configurations, allowing the simulation to be resumed at any point. As a consequence, read/write methods for the J matrix were also required, to ensure consistency between simulation runs.
Single-step Metropolis and Over-relaxation methods were then implemented. Their logic directly follows the descriptions provided earlier. The only important detail to highlight is that the Metropolis method, at each run, returns a struct containing the results: whether the move was accepted and the current system energy (used for plotting with a Python script).
The final method implemented for the `sys` class was the `run()` method, which simply iterates the Monte Carlo step `mc_moves` times (recall that one Monte Carlo step consists of 10 volume Over-relaxation moves and 1 volume Metropolis move). After exiting the loop, the full vector of the energy history as a function of Monte Carlo steps is saved to file.
Subsequently, the `replica` class was developed, used to create a vector of `sys` objects of size N_T. This was necessary to implement Parallel Tempering. Accordingly, a method to exchange configurations and a method to perform a Parallel Tempering step (which internally calls the exchange method) were created.
Another key method in this class is `parallel_evolve()`, which spawns a thread for each of the N_T systems. This allowed the systems to be executed truly in parallel. Thanks to this implementation, it became possible to attempt Parallel Tempering swaps on all N_T systems at the end of each Monte Carlo step.
Again, a `run()` method was implemented, integrating all previous components and saving to N_T files the energy histories of each parallel system. Finally, the last method added was `avg_run()` (averages run), which was developed to carry out measurements. This method evolves all N_T systems by one Monte Carlo step.
Note: the standard `run()` method was not used here because it includes a progress bar, which is unnecessary and annoying to see for a single step.

Afterward, the `replicas2` class was implemented to compute the overlap factor and thus evolve $N_T \times 2$ systems in parallel.
In addition to the usual `run()` method, a function was added to compute all overlap factors $q^{\mu\nu}$

for $\mu, \nu = x, y$. Then, a method was implemented to compute a portion of $\chi_{SG}(\mathbf{k})$. Given the linearity of expectation values, the following algebraic manipulation was used:

$$\chi_{SG}(\mathbf{k}) = N \sum_{\mu\nu} \left[\langle |q^{\mu\nu}(\mathbf{k})|^2 \rangle \right]_J = \left[\left\langle N \sum_{\mu\nu} |q^{\mu\nu}(\mathbf{k})|^2 \right\rangle \right]_J \tag{14}$$

The goal is to factorize the operations. The idea is to compute $\alpha(\mathbf{k}) \equiv N \sum_{\mu\nu} |q^{\mu\nu}(\mathbf{k})|^2$ for $\mathbf{k} = 0$ and $\mathbf{k}_{\min}$, and then perform averages over the same system and finally over disorder by varying the sample (and thus the realization of disorder). This step is the most computationally expensive, as obtaining a good statistical ensemble requires performing a number of measurements comparable to the number of Monte Carlo steps taken for each of the N_T systems, and running a huge number of systems in parallel. For instance, according to the reference paper *"Large-scale Monte Carlo simulations of the three-dimensional XY spin glass" by J. H. Pixley and A. P. Young*, in order to study the system at $L = 24$, 461 different samples were used, with each sample consisting of $N_T = 35$ systems running in parallel (all of this doubled since each system has its own replica). Each system required about 2.5 million Over-relaxation volume moves for equilibration (i.e., 250,000 Metropolis volume moves). After equilibration of these 32,270 systems, 2.5 million measurements were carried out on each one to compute averages over the specific realization of disorder, and then averaged again over disorder. Clearly, this is far beyond our computational resources without access to a computing cluster.
Once $\chi_{SG}(\mathbf{k})$ was computed, it was finally possible to calculate ξ_L, as defined in equation (7).

3 Results

3.1 Equilibration and Average Energy

As a first result, I present the energy vs. Monte Carlo steps plots for various values of L:

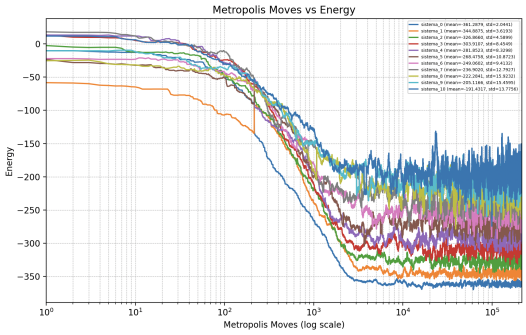

Figure 4: Equilibration of the system with $L = 6$

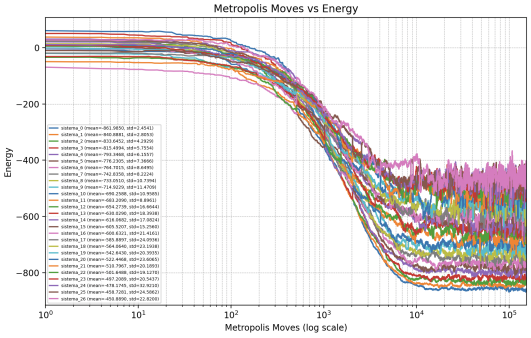

Figure 5: Equilibration of the system with $L = 8$

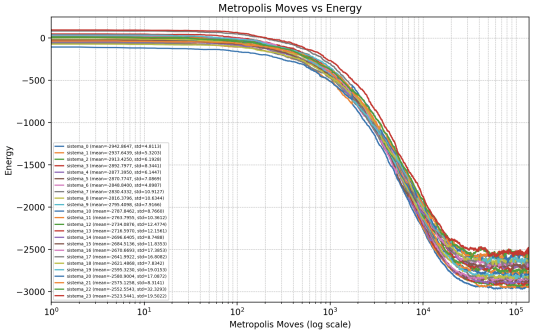

Figure 6: Equilibration of the system with $L = 12$

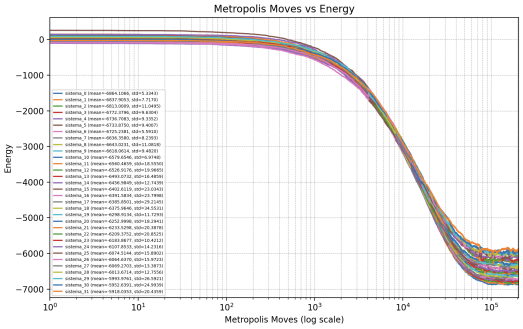

Figure 7: Equilibration of the system with $L = 16$

We observe that for smaller systems, energy fluctuations are far from negligible, whereas as system size increases, the relative error on energy becomes very small (down to $\sim 10^{-4}$).
To better illustrate the effect of Parallel Tempering, a zoom-in on the energy plot for $L = 12$ is shown:

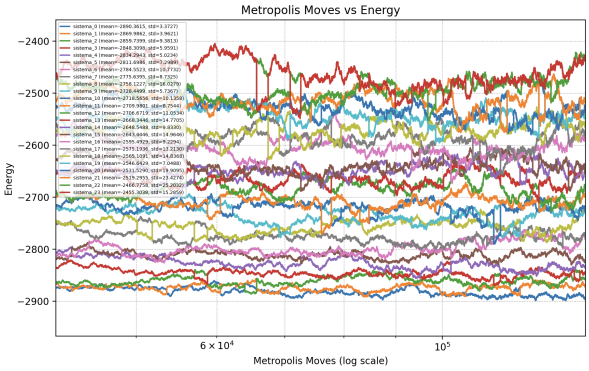

Figure 8: Parallel Tempering – $L = 12$

It is interesting to observe how the systems get "excited" and are able to escape local minima, allowing them to continue exploring the configuration space. This is due to two main reasons: from a physical perspective, energy barriers become lower at higher temperatures (while still below the critical temperature); mathematically, the Metropolis acceptance rule contains

temperature in the denominator of the exponential, so higher temperatures make it easier to accept moves with the same ΔE.

3.2 Speed-up

We now shift our attention to the speed-up obtained using all three Monte Carlo moves. It was observed that the machine time required to equilibrate using Parallel Tempering (PT) + Over-relaxation (OR) + Metropolis (MT) does not change significantly when PT is removed. In fact, from a computational point of view, PT is just a configuration (or temperature) swap. However, it was also observed that without PT, some systems with $L \geq 12$ remained trapped in local energy minima, resulting in higher final energies compared to simulations with PT. Thus, PT remains essential for producing correct ensemble averages.
Based on these considerations, the only time comparisons made are between simulations with all three moves and simulations using only Metropolis.

L	MT moves (OR+PT+MT)	$t_{\{OR+PT+MT\}}$ (s)	MT moves (MT)	$t_{\{MT\}}$ (s)	Speed-up (%)
8	153600	1.4147	384000	7.1615	506
12	138240	7.697	345600	37.026	481
16	204800	31.323	2252800	855.754	2732

Table 1: Performance analysis for different values of L and computational techniques

For very small systems, the difference is negligible, but for sufficiently large systems we observe a dramatic speed-up, reaching up to 2700%!
To illustrate how quickly convergence is achieved, the figure below compares the two techniques for $L = 16$, using the same parameters shown in Table 1.

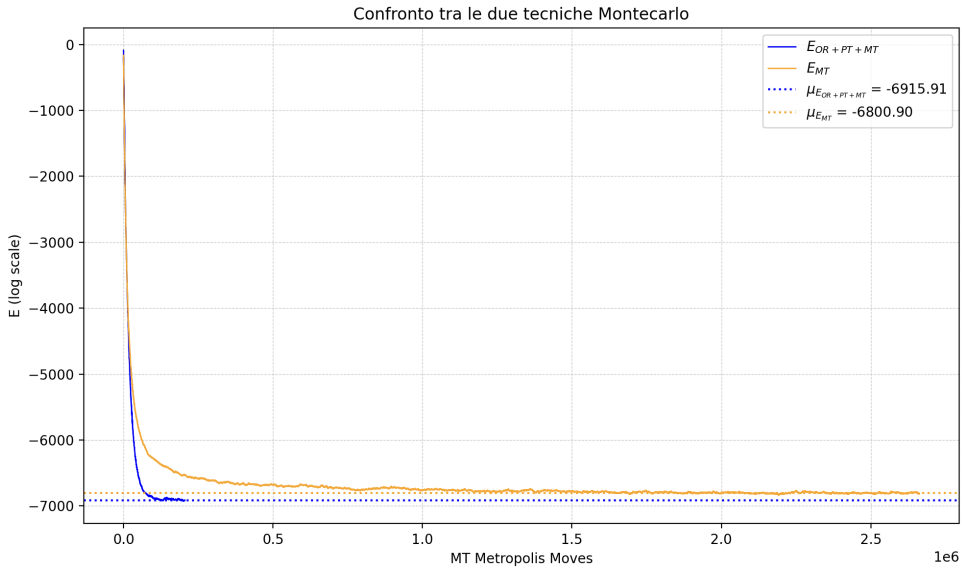

Figure 9: Comparison between the two techniques – system with $L = 16$

In addition to the significant reduction in the number of Metropolis steps, the graph shows that the system using all three Monte Carlo moves reaches a slightly lower equilibrium energy. This is clearly due to Parallel Tempering, which enabled the system to escape a local free energy minimum.

3.3 Correlation Length

Lastly, we present the results obtained for the correlation length as a function of system size. The only systems we were able to study were $L = 4$, $L = 6$, and $L = 8$. These were the only sizes for which we were able to simulate up to 3800 systems. The configurations used to obtain these values are shown below:

L	MC Moves	MT Moves	N_T	$\Delta\theta_{\max}$ (fractions of π)	$T_{\min}$	$T_{\max}$	MC Moves needed for measurements	N_{sample}	Total systems
4	200	12800	11	0.62	0.2	1.4	500	100	2200
6	250	54000	19	0.45	0.2	1.4	250	100	3800
8	300	153600	27	0.48	0.2	1.4	30	30	1620

Table 2: Monte Carlo simulation parameters and measurements for different values of L

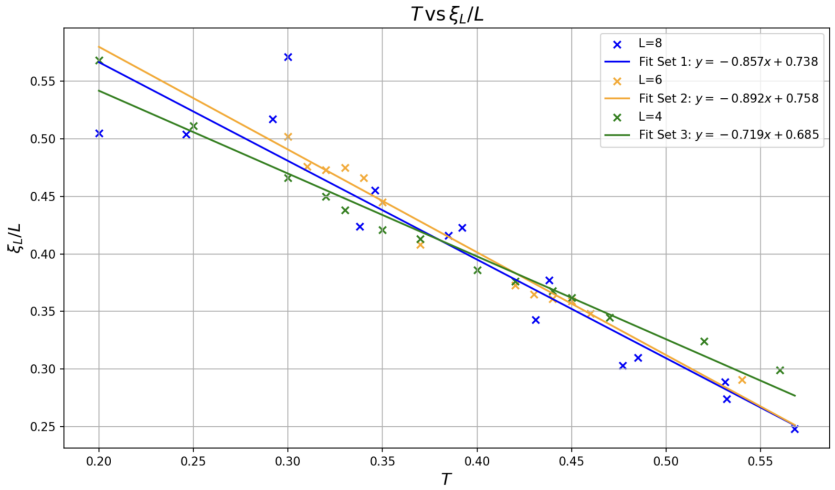

Figure 10: Correlation lengths (rescaled by system size) as a function of temperature: T vs ξ/L

We observe that the intersection of the three curves in the Pixley and Young paper occurs around $T \sim 0.36$, while in our case, it occurs at approximately $T \sim 0.37$.
Given the limitations in statistical ensemble size, we can consider these results satisfactory.

4 Conclusions

We can conclude that the most efficient way to achieve convergence for the 3D XY spin-glass model on a cubic lattice is the combination of the three Monte Carlo moves described above: $\times 10$ Over-relaxation, $\times 1$ Metropolis, and $\times 1$ Parallel Tempering. This approach enabled improved ensemble averages and significantly reduced convergence times. Speed-ups of up to 2700% were recorded compared to simple Metropolis Monte Carlo. Finally, the correlation lengths obtained are consistent with those reported in the work of Pixley and Young. We also observed a strong dependence of the results on the size of the statistical ensemble, suggesting that running the code on a computing cluster would be the most natural improvement — allowing access to system sizes and parallel configurations currently beyond our reach.