Replica Exchange Monte Carlo sampling on an Ising Spin Glass

Justin Smith

May 1, 2014

Dr. Adrian Roitberg
University of Florida
Project for the computational chemistry course.
Program with source code is available on Mordor in the `/scratch/justin/Proj_CUDA_v2` directory.

1 Purpose

I chose this project to explore the use of parallel computational methods on a system of Ising spin glass using the Replica Exchange Monte-Carlo (REMC) method. To accomplish this goal, it was necessary to learn C/C++ and how to implement the CUDA and openMP libraries in a C/C++ program. The rest of this paper will explain the implementation of the Ising model for energy calculations and the use of REMC along with results from the program showing that it works as it was designed.

2 The Ising Model

The Ising model is a system of particles in the form of a lattice. Each particle in the lattice is described by a spin with the properties of spin up (+1) or spindown (-1). In Figure 1, you can see an example of such a system. For an N particle system, the energy of the spin glass outside of any external magnetic field is described by the following hamiltonian,

$$\hat{H} = \sum_{i}^{N} \sum_{i,j} J S_i S_j \tag{1}$$

Here, S_i is the spin value of the i^{th} particle, S_j is all of the i^{th} particles nearest neighbours and J is the interaction constant between the spins. A J value which is negative gives rise to an anti-ferromagnetic system and a positive J gives a ferromagnetic system.

Though many other variations on the Ising spin model exist, each of varying complexity, I only worked with the simplified anti-ferromagnetic system with no external magnetic field. In other words, I used $J = -1$. But, the program was designed to be easily modifiable so that more complex systems of spin glass could be used, including an external magnetic field. I plan on playing around with this in the future.

-1	1	1	-1	1
1	-1	-1	1	-1
1	1	1	1	-1
-1	-1	-1	-1	-1
1	1	-1	1	1

Figure 1: An example of a 5X5 Ising spin glass.

3 Monte-Carlo Sampling

In this project, a single Monte-Carlo "move" is simply flipping the spin of a randomly chosen particle in the system. The energy of the system can then be calculated using the Hamiltonian in Eqn. 1. I use the same Metropolis criterion as other molecular Monte-Carlo algorithms to calculate the acceptance probability. I explain this method in greater detail in the Monte-Carlo Implementation section.

4 Replica Exchange Monte-Carlo

Replica Exchange Monte-Carlo sampling is a form of Monte-Carlo sampling where a number of parallel systems are computed at one time, then exchanged among some property

that energy is dependent upon, such as temperature. In the method in this project, the replicas are exchanged between different temperatures. At points throughout the computation, the normal Monte-Carlo cycles stop running, and an exchange of systems between temperatures is proposed. The determination on the acceptance of this proposition is given by the Metropolis-Hastings algorithm,

$$P(RE_{i\to j}) = \begin{cases} 1 & : \Delta \leq 0 \\ exp(-\Delta) & : \Delta > 0 \end{cases}$$

Here, $\Delta = [\beta_n - \beta_m](E_i - E_j)$ where $\beta_l = 1/k_B T_l$. Temperatures m and n correspond to replicas j and i, respectively. This algorithm assures that detailed balance is satisfied, in much the same way as the Monte-Carlo metropolis acceptance criterion.

5 The Program

The program was written in C/C++ on a Linux distribution. It was designed so that the energies are calculated on Graphics Processing Units (GPUs), using the CUDA architecture, and the Monte-Carlo cycles and random number generation is computed in parallel on CPUs using openMP. Initially, the program was written entirely on GPUs for all main calculations, but given some issues in my original design method, and the limited time to finish the project, it was faster to run the Monte-Carlo calculations and pseudo random number generation on CPUs. Though, I have ideas on redesigns that could significantly increase program execution speed by putting all main calculations back on GPUs.

I must also mention that to simplify calculations further, a $k_B = 1$ was used. This was to simplify the system so that low values of temperature can be chosen in the computations. Though this can easily be changed to any value of $k_B = 1$ so that realistic systems can be considered.

5.1 Calculation Implementation

The energy calculations are done in two different parts of the program to lessen to computational load. Given the program is working on a system of spin glass, I chose to use a method known as single spin flip dynamics. This was my first mistake in properly paralleling the Monte-Carlo cycles on GPUs, I will talk more on this later in the Monte-Carlo Implementation subsection.

The first energy calculation is the individual energy calculations of the Monte-Carlo cycles. In each cycle, only the energy difference of a single spin flip is calculated, rather than flipping the spin then recalculating the entire system energy. In other words, suppose a random particle, P_i, is chosen in one of the N replicas. The energy of the particle in its current state is calculated, then the spin is flipped and its energy is calculated. The difference is then used in the Metropolis acceptance criterion. This prevents the entire system energy from being calculated in each cycle, saving potentially billions of computations. This part is all currently done in parallel on CPUs.

The second energy calculation is used in the saving of histogram data and in the replica exchange part of the program. So, it only needs to be calculated rarely in comparison to the number of Monte-Carlo cycles. This part is done entirely on GPUs using the CUDA architecture. This computation is best suited for GPUs since it involves a very large number of computations all of which can be done in parallel. For a 128×128 system with 6 replicas this means a total number of 100k energies must be calculated then summed. This is perfect for GPUs and their large number of cores.

5.2 Monte-Carlo Implementation

The Monte-Carlo method was implemented in a multiple CPU parallel environment. Basically, between each energy sampling for saving histogram data, thousands of MC cycles are ran over all of the replicas. My original implementation of this on GPUs had each individual replica calculating one at a time in a serial fashion. Since there are so few replicas in the computations being ran, this was wasteful since GPU clock speeds are far lower than CPU clock speeds. So, I moved the method over to CPUs to speed it up. Right after doing this I came to the realization that the order in which each replica calculates its Monte-Carlo cycles does not need to be done serially, especially in very large systems where its highly unlikely the same random particle in a system will be chosen and computed at the same time. Though safeties could be implemented to prevent this if necessary. I will look more into this in the future as I believe it could greatly increase the execution speed of the program.

Periodic boundary conditions are used in the Monte-Carlo cycles to simulate each replica as a small piece of a larger system.

5.3 Replica Exchange Implementation

The Replica Exchange program is also ran on CPUs. However, the calculations are done so rarely that running the computations in parallel would be more harmful than useful since the energies are already calculated in advance as part of the histogram data storing system.

Instead of exchanging replicas or temperatures, I simply use a permutation function in the form of an array within the program. This permutation function stores the information that tells the system which replica is in which temperature at any point in the program execution. Data from this permutation function is stored as well. So, the Replica Exchange part of the program doesn't actually swap systems or temperatures but updates the permutation function to give this effect.

6 Program Results

One system was ran in this program to give the following results. The system was ran 5 times with replica exchange disabled and 5 times with it enabled. The system in question is a 128×128 particle Ising spin glass. Instead of using a convergence criterion (I didn't have time to tweak this correctly for implementation), I ran the program with 2000 replica exchange attempts.

Each replica exchange attempt had 25 energy samples (50k total) and there are 49152 Monte-Carlo cycles per sample per replica (2.5 billion cycles per replica). This arbitrary number of cycles was chosen, and kept constant between runs, so that each system could be compared. The comparison is in the form of how low of an energy state it achieved in that given number of cycles and how well the final ensemble average energies agree between the runs. There were 6 total replicas ran in parallel, with the 6 temperatures of 2.05K, 2.083K, 2.117K, 2.15K, 2.183K, and 2.217K per run. Given that a ($k_B = 1$) was used, the energies in the following data have arbitrary units of energy. They are simply used to show that the program works as designed.

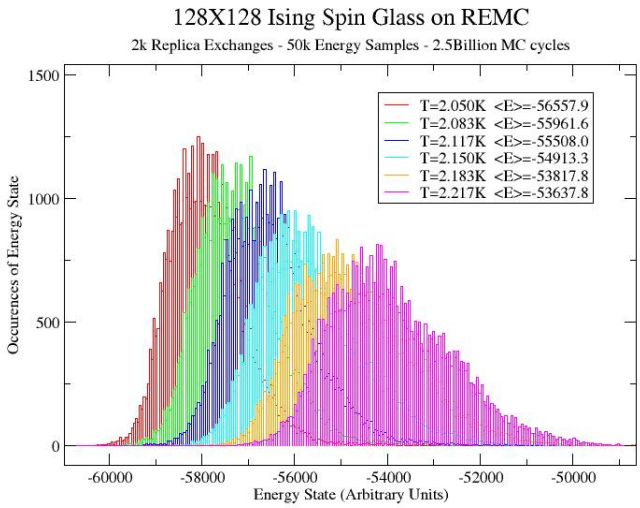


Figure 2: Histogram data gathered from the first REMC run.

Figure 2, shows the histogram data for the first run with replica exchange enabled. The shape of the curves are as expected for a Boltzmann distribution as well as the fact that the lower temperatures occur lower in energy.

The temperatures for these runs were chosen to give an acceptance ratio of the Replica Exchange around the same value for all replicas and 0.2 overall probability so that each replica would remain in a temperature for a significant amount time.

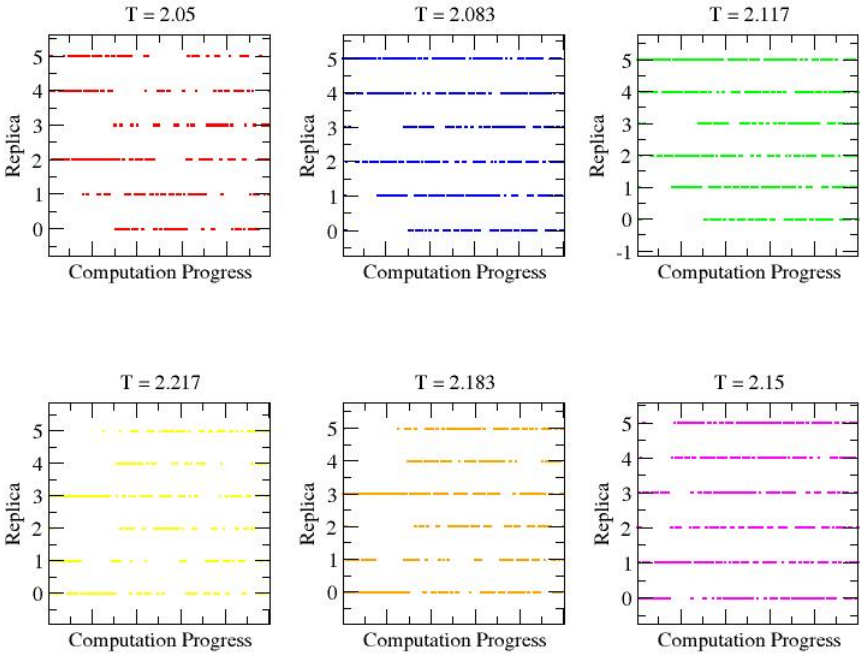
Table 1 shows the values for the acceptance ratio from the first replica exchange run. All other runs on this system had similar values to these.

Table 1: Actual values of the exchange probability for the first run with replica exchange enabled.

Replicas	Exchange Probability
$P(T_0 - T_1)$	0.192
$P(T_1 - T_2)$	0.226
$P(T_2 - T_3)$	0.183
$P(T_3 - T_4)$	0.1885
$P(T_4 - T_5)$	0.1895
$P(T_5 - T_0)$	0

In Figure 3, graphs are shown for all six temperatures of the first run. These graphs show that each replica achieved a random walk through temperature space, which shows that the replica exchange portion of the program worked as expected.

Figure 3: Graphs that show each replicas random walk through temperature space for the first REMC run. Temperatures are in units of K.



6.1 REMC vs. MC only

The purpose of replica exchange is to get a better sample of the energy space in a shorter amount of time. To show that this was achieved with the program, I compare the average ensemble energies of the lowest temperature ensemble of the MC only runs with the cycle at which the same energy was achieved with replica exchange enabled MC runs. In table 2 you can see the data for this comparison. Aside from one value, the one corresponding to the 4th REMC run and 2nd MC only run, every REMC run achieved a lower average ensemble energy in far less cycles than the MC only runs.

Table 2: $< E >_{2000}$ is the average energy of the lowest temperature ensemble for the MC only runs at the 2000th cycle. The REMC columns give the cycle at which the corresponding MC average ensemble energy was surpassed by the average ensemble energy of the REMC runs.

Lowest T (MC)		**Run Number (REMC)**				
Run	$< E >_{2000}$	1	2	3	4	5
1	-57046.21	165	255	220	411	264
2	-57868.319	1812	1788	1309	N/A	1314
3	-56188.954	82	122	114	221	146
4	-57645.462	517	761	522	995	617
5	-57601.092	446	667	467	902	572

In Table 3, the total ensemble average energy ($< E >$) along with the standard deviation (σ) of the 5 runs with REMC and MC only method are tabulated. Even though there are only 5 values of each, it is enough to see that the standard deviation is huge for the MC set. In Figure 4, the final ensemble energies for each run where 0 is the first and 4 is the last run is graphed. The red line with squares shows the MC only runs and the blue lines with circles shows the REMC runs. This clearly shows an outlier which is greatly effecting the statistical values for the MC only runs. However, more runs will need to be ran to determine if this was just a bad starting system or if there is a minima in which the MC only calculation was stuck. In any case, even with only 5 data points the REMC energies agree with a very small standard deviation. Again though, more points will need to be ran to show this data is statistically sound.

6.2 Program Run Times

Even though the program was designed to run in parallel, the computations on the 128 x 128 system took a significant amount of time. The computations took around 1 hour and 13 minutes on my 4 core AMD Phenom 3.4GHz CPU and one GTX 295 GPU.

Mordor was also used to compute some of the data, and it took around 34 minutes on 12 cores and 1 Titan GTX GPU. Though the interesting part is that, on the 4 core machine, the CPU time was around 4 and a half hours. But on Mordor with 12 cores it was using about 11 hours of CPU time. This increase in CPU time to shorten real time is expected, but the extent of the difference shows that the program likely needs to be optimized better for multiple cores.

Table 3: The average ensemble energies and standard deviations of all runs REMC runs and all MC only runs at the end of the program run.

Run Type	$< E >$	σ
REMC	-57894.30	29.4964
MC only	-57270.01	675.8923

7 Conclusion

The program seems to have achieved what it was designed to do. The replica exchange method accelerated the sampling of the energy space by finding lower energy configurations faster than the regular MC method could. As for the design, while the computation run times on the 128x128 system seemed high, I currently have no other system to compare

c

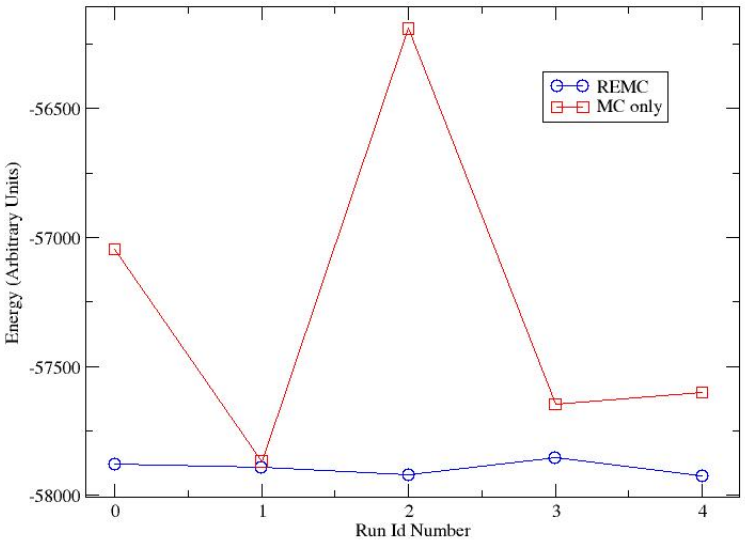


Figure 4: A comparison of the average ensemble energies of the REMC runs to the MC only runs.

this with. Even though the program is crude, it is my first attempt at a program of this scale. Were I to remake the program with the knowledge I have gained in this project, the program would likely be significantly faster.

References

[1] Yuji Sugita and Yuko Okamoto, "Replica exchange molecular dynamics method for protein folding." Chemical Physics Letters 314. 1999, 141-151.

[2] Swendsen, Robert H. and Wang, Jian-Sheng, "Replica Monte Carlo Simulation of Spin-Glass" Physical Review Letters 57. 1986, 21.

[3] Ferguson, David M., Siepmann, and Truhlar , "Monte Carlo Methods in Chemical Physics" John Wiley & Sons, inc. 1986, volume 105.