N87-26537

A STUDY OF THE XY MODEL BY THE MONTE CARLO METHOD

by
Peter Suranyi and Paul Harten

ABSTRACT

The MPP is used to perform Monte Carlo simulations for the two dimensional XY model on lattices of sizes up to 128x128. A parallel random number generator was constructed, finite size effects were studied, and run times were compared with those on a CRAY X-MP supercomputer.

Keywords: XY model, Spin System, Monte Carlo, Random Number Generator

BACKGROUND

XY model

The XY planar model is a two dimensional latticework of interacting spins. Each spin is a vector of unit magnitude in the XY plane. The Hamiltonian of the system is

$$H = -J \Sigma S_i \cdot S_j = -J \Sigma \cos(\theta_i - \theta_j) \qquad (1)$$

where J is the spin-spin coupling, S_i is the spin at site i, θ_i is the angle that S_i makes with an arbitrary axis, and the summation is over nearest neighbor pairs.

Unlike the Ising model which has a finite mean magnetization or long range order for temperatures below a certain critical temperature, the XY model has a different type of behavior in that there is no long range order for low temperatures. However, it does seem to have a critical temperature T_c where it goes through a phase transition.

Kosterlitz and Thouless show (ref.1) that there exists certain topological defects in the XY model which govern its unusual behavior at low temperatures. These defects are vortices of spin. Going around any closed path in the system the change in spin from site to site will add up to an integral multiple of 2π. This number is a constant of the system so that whenever a vortex is created.

Below the critical temperature, vortex-antivortex pairs are bound and stay close together. Above the critical temperature there is enough heat-energy in the system to break the bond and the vortices go off independently. This is the phase transition that the system goes through. It is under a class of what are known as defect mediated phase transitions.

Statistical mechanics

Using a heat bath algorithm, the probability of any state i of the system is given by the Boltzman factor

$$P(i) = \exp(-\beta H_i) = \exp(-H_i/kT) \qquad (2)$$

where H_i is the Hamiltonian of the system, k is the Boltzman constant, and T is the temperature. States of the system are values of θ at each of the sites. It follows then that the partition function is given by

$$Z = \Sigma \exp(-\beta H_i) \qquad (3)$$

where the summation is over all states of the system. The expectation value of any variable x is given by

$$\langle x \rangle = Z^{-1} \Sigma\, (x)\exp(-\beta H_i) \qquad (4)$$

where again, the summation is over all states of the system.

Monte Carlo

The Monte Carlo method approximates these expectation values by averaging the value of x over a random sampling of states near equilibrium, these being the states of highest probability. Since the values that the θ 's take on are continuous, the summation over all states is an integration from 0 to 2π over all θ. The random sampling near equilibrium then represents evaluations of this integral at a very sharp exponential peak.

The states near equilibrium are arrived at through a Markovian process where the transitional probability of going from state 1 to state 2, Q(1,2), follows the detailed balance condition

$$Q(1,2)P(1) = Q(2,1)P(2) \qquad (5)$$

where P(1) and P(2) are the probabilities of state 1 and 2 respectively.

The Metropolis method used sets Q(1,2) = 1 if the energy of state 2 is lower than the energy of state 1, and sets

$$Q(1,2) = P(2)/P(1)= \exp(-\beta\Delta E) \qquad (6)$$

otherwise. Here ΔE is the change in energy from state 1 to state 2. A little reflection will show that this indeed does fulfill the detailed balance condition. Not only does this method allow the system to approach the state of lowest energy (equilibrium) but it also allows the system to get out of any local minimum that it might find itself.

XY MODEL PROGRAM

System update

A 128x128 parallel array of real numbers is defined where the elements take on values from 0 to 2π. This is a lattice of spins and the numbers represent the angle θ of the spin at that site with respect to some axis. When any site is updated it depends only upon its four nearest neighbors so that for the purpose of updating there are actually two separate sub-lattices in a checkerboard pattern. One sub-lattice is updated at a time so as not to affect the updating of other sites. This conditional update is done, quite naturally, with a checkerboard mask.

For a site to be updated the θ values of its four nearest neighbors are brought into memory locations at that site using the ROTATE function. Along the sides of the lattice the ROTATE function brings in the θ values of the opposite sides, thus giving the system periodic boundry conditions. A trial spin is arrived at by randomly adding a value $\Delta\theta$, where $-\phi < \Delta\theta < \phi$, to the spin at that site. The value of ϕ is optimized for the fastest convergence. The difference in energy between the trial state and the original state is given by

$$\Delta E = H_t - H_o = -J\, \Sigma \cos(\theta_t - \theta_k)$$
$$+J\, \Sigma \cos(\theta_o - \theta_k) \qquad (7)$$

where the t and the o subcripts are the trial spin and the original spin, and the summation index k is over the four nearest neighbors.

A random number R is then generated, if the energy of the new state is lower or if the random number $R < \exp(-\beta\Delta E)$ then the trial spin is kept, otherwise it is thrown away. This whole process is done for all sites on the sub-lattice in parallel. Following this, the complement sub-lattice is updated in the same manner and we have a new state of the system near equilibrium.

Expectation values

The system is initially in a given state usually one of zero temperature where all of the spins are aligned. Then it is in effect brought into contact with a heat bath of temperature T. A warm-up period follows and the system is allowed to come to equilibrium at the temperature T. For the XY model, this warm-up takes on the order of 10,000 to 20,000 lattice updates. Sample values of x are taken every 100 updates to promote independence between values. Usually x will be a scalar property of the system so that the SUM function is used at this step.

The total number of samples taken are broken up into several equal divisions and x is averaged in each of these divisions. From these divisional averages both the total average of x and its standard deviation can be calculated.

Random number generator

The basic idea of the algorithm used to generate a plane of random numbers in parallel is as follows. Begin with two prime numbers a,p and set up an integer parallel array with a different power of a at every site where the powers range from 1 to 16384 (128*128). Multiplying every element by the constant a**16385 will always give a different power of a.

The next step is to take all numbers modulo p so that the randomness comes in as a**m mod p where m is some integer. Since

$$(ab)\bmod p = [(a \bmod p)(b \bmod p)]\bmod p, \quad (8)$$

If we take modulo p at every step we will always have a**m mod p no matter how large m is.

Each time after multiplying by a**16385 mod p we have a new plane of random integers from 0 to p. To have a plane of random real numbers from 0 to 1, it is only necessary to divide by p.

This is an example of the parallel random number generator setup, which is done only once, and the generator itself. Here a=SEED, p=MAXRND, and IRND is the parallel array of integers:

```
PROCEDURE SETRND(VAR SEED,SEEDM:INTEGER
                        ;VAR IRND:ILAT);
  VAR I,J:INTEGER;
BEGIN
  SEEDM:=SEED;
  FOR I:=0 TO 127 DO
  BEGIN
    FOR J:=0 TO 127 DO
    BEGIN
      WHERE (ROW_INDEX=I) AND
          (COL_INDEX=J) DO IRND:=SEEDM;
      SEEDM:=SEEDM*SEED MOD MAXRND;
    END;
  END;
END;
```

ROW_INDEX and COL_INDEX must be declared as parameters in the PROGRAM line. SEEDM=a**16385 mod p

RAND is a parallel array of real numbers.

```
PROCEDURE GENRND(VAR SEEDM:INTEGER;
          VAR IRND:ILAT;VAR RAND:RLAT);
BEGIN
  IRND::13:=ROTATE(IRND::13,1,0);
  IRND:=IRND*SEEDM MOD MAXRND;
  RAND:=IRND/MAXRND;
END;
```

The bit plane ROTATE will be explained shortly.

RESULTS

Fourier components

For the purpose of finding some good seeds to use in this psuedo-random number generator, there are a few constraints to consider. It is necessary that p < sqrt(2**31-1) in order to insure that there is no erflow. Also, the fourier components of any random number generator must give results that are close to 0.

A program was written to look at the first 10 sine and cosine fourier components. From the results of this program it became clear that the multiplication by a**16385 mod p was not enough to insure randomness. But, the additional operation of rotating the thirteenth bit of every number in the random integer array one element north, did much to improve the randomness.

Two seeds found to give very good psuedo-random distributions are a=16307 and p=16309. Figure 1 shows the fourier components of the distribution generated from these seeds. The dashed lines on either side of zero represent one standard deviation assuming perfect randomness.

Given a perfectly random distribution and an infinite amount of numbers generated, the fourier components would be zero. Since finite amounts are being dealt with, there will be fluctuations away from zero even with perfectly random distributions. The points will do a random dance around zero. It is clear that these two seeds come very close to giving a random distribution.

More to the point is the result of using this psuedo-random number generator in simulating the XY model.

XY model simulation

The XY model simulation program was run at various values of β near the phase transition. Figure 2 shows the relation of the expectation value of nearest neighbor correlations, $<\cos(\Theta_i - \Theta_{i+1}>$, to β. The error bars are smaller than the points themselves. The lower curve is the result of a 128x128 lattice of spins and the upper curve is the result of a 16x16 lattice of spins.

The 16x16 lattice of spins was run for the purpose of comparison with some previous results done on a Cray X-MP. The results from the Cray are also shown in figure 2 and are sitting under the same points as generated by the MPP. This is exact agreement.

The distance between the curves of different sized lattices come from what are known as finite size effects. These are mostly due to the fact that in addition to direct spin-spin interaction there is also interaction coming around periodic boundries. In an infinite sized lattice, this would not happen.

The two curves come together at approximately $\beta = 1.15$. This is where the phase transition is taking place. Theoretical predictions (ref.2) give $\beta_C = 1.14$.

Cray-1 CPU time

From similar runs of an XY model simulator written for a Cray X-MP, very rough calculations show that it takes 1.36E-3 secs of CRAY-1 CPU time per lattice update. On the MPP, calculations show that it takes 1.86E-4 secs per lattice update. This is approximately seven times faster than the CRAY-1 time.

CONCLUSION

The MPP is a very useful tool in simulating the two dimensional XY model. Properties of the XY model can now be investigated which are realistically impossible to do anywhere else.

ACKNOWLEDGEMENTS

This work was supported in part by the US Department of Energy under Grant Number FG02-84ER40153

REFERENCES

1. J.M. Kosterlitz and D.J.Thouless,
   J.Phys. C6(1973)1181;
   J.M. Kosterlitz,J.Phys.C7(1974)1046.
2. D.C.Mattis,Phys.Lett. 104A(1984)357.

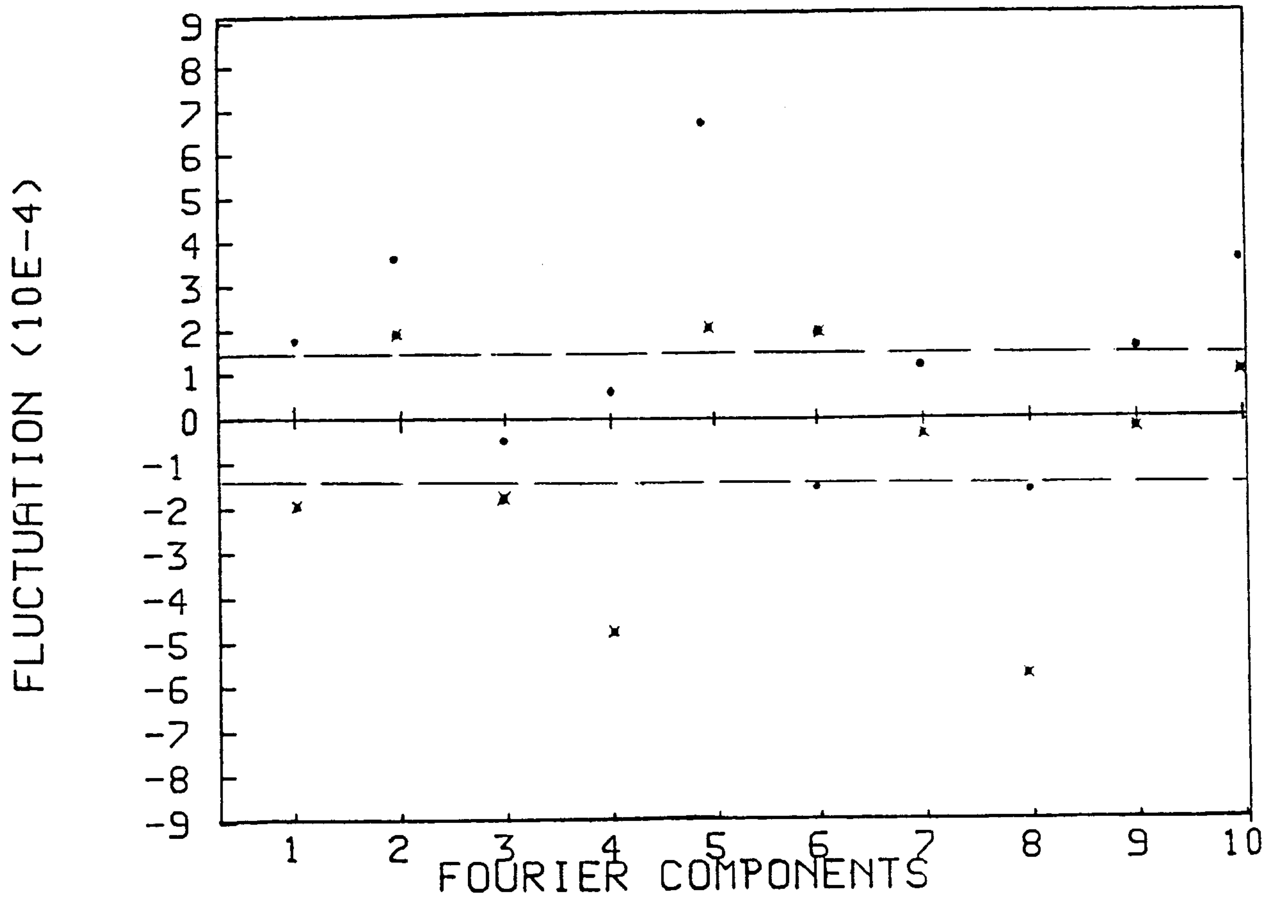

Figure 1. Random Fluctuations of the Fourier Components Away from Zero

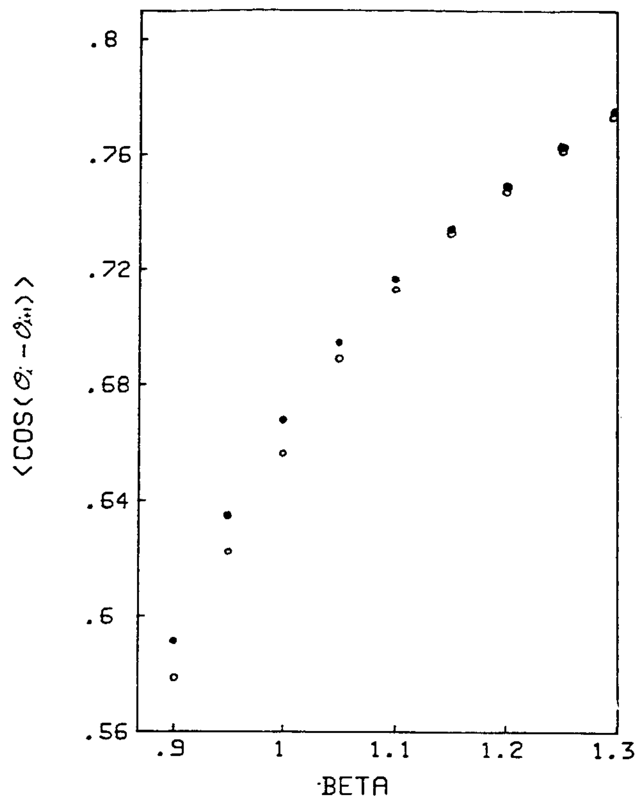

Figure 2. Nearest Neighbor Correlations vs β.