A Report

On

**Lattice Gauge Theory**

Ravi KiranY

ID No. 2005S5P3485

C Hanuma Chaitanya

ID No. 2005S5P3755

A report submitted in the partial fulfillment of the requirements of

Study Oriented Project.

27 November 2007

Birla Institute of Technology and Science, Pilani – Goa Campus

Acknowledgement

I would like to thank Dr. Raghunath Ratabole for encouraging us to undertake this project, for the all the help and guidance given to write this report. I specially wish to thank the librarian without whose cooperation this report would not have been possible.

**Abstract**

The part of the project taken up under Study Oriented Project covered in this report consists of ‘The Lattice Gauge Theory’. This report documents the results and plots obtained after running the simulations of the ‘Gauged Version of Ising Model’. The algorithm used was Metropolis-Hastings Monte Carlo Algorithm and the coding was done in C.

**Table of Contents**

**Field theory**

**Applications of Field Theory**

**Gauge Theory**

**Formulation**

**Examples of Gauge Theory**

1. **Electromagnetism.**
2. **Weak Interactions (electro)**

1. **Strong Interactions**

**Solving a Field Theory – Lattice Approach**

**Z-2 Gauge Group**

**Introduction**

The part of the project taken up under Study Oriented Project covered in this report consists of ‘The Lattice Gauge Theory’. This report documents the results and plots obtained after running the simulations of the ‘Gauged Version of Ising Model’. The algorithm used was Metropolis-Hastings Monte Carlo Algorithm and the coding was done in C.

**Gauged Version of Ising Model**

The Ising model, named after the physicist Ernst Ising, is a mathematical model in statistical mechanics. In the gauged version the Ising model can be represented on a graph where its configuration space is the set of all possible assignments of +1 or −1 on each of the links of the lattice.

Energy function for a given configuration H ({p}) must be defined, where p is the value of the placket given by the product of all elements on the surrounding links and {p} denotes the set of configuration in a lattice gauge.

Example:

1

1

1

1

1

1

1

1

1

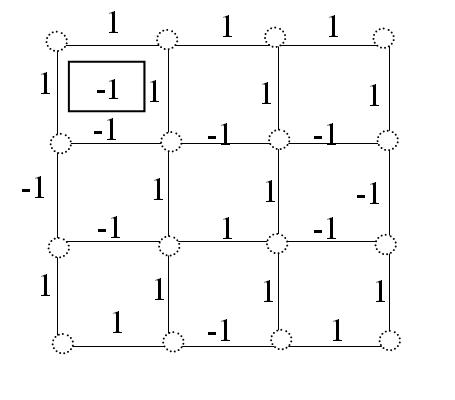
1

1

1

1

1



{ p } = { p11,p12,p13,p21,p22,p23,p31,p32,p33 }

p11=  = -1 [product of spins on the links of the placket p11]

Energy for the Gauged Ising model for a given configuration {p} is evaluated by the following formula



Where:  denotes the product of the spin variables on placket p

p denotes all possible plackets

 Is the Boltzmann temperature

The net spin is defined as

****

Where: N is the number of plackets for the square lattice

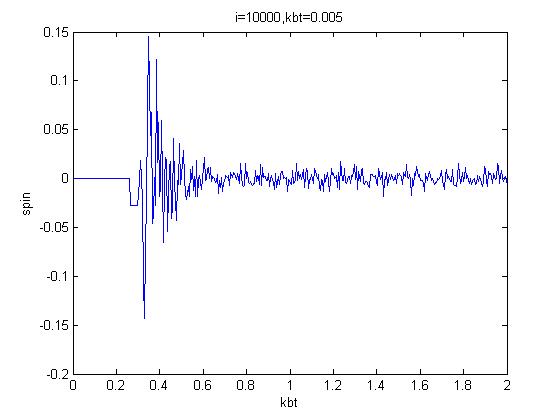
Sis the spin variable on the link (i,j)

**Boundary Conditions:**

In order to accomplish the finite size scaling in the lattice gauge and to ease the computational complications the lattice is bounded from both the sideways and also top - bottom and thus the lattice is taken as a torus instead of a square.

**Plots:**

Magnetization

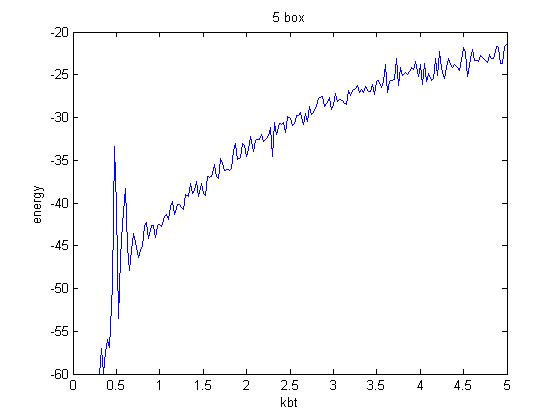
****

The average Spin for the lattice turns out to be a constant very close to zero which is co inciding the theoretical results for a 2-D Gauged Ising Model.

The above plot was obtained for 5 by 5 placket size gauge with 10,000 Monte-Carlo sweeps, the Kbt was gradually varied by 0.005.The variations are accounted for limitation in computational resources. The maximum variation turned out to be 0.15.

Therefore the “average spin” is not the parameter which determines the phase transition.

Energy:

****

The following plot is obtained for the average energy of the lattice .It shows a smooth variation as the Kbt increases and it finally tends to zero as the Kbt tends to zero.

The above plot was obtained for 5 by 5 placket size gauge with 10,000 Monte-Carlo sweeps, the Kbt was gradually varied by 0.005.The variations are accounted for limitation in computational resources

**Wilson’s Loop:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  | R |
|  |  |  |  |

.

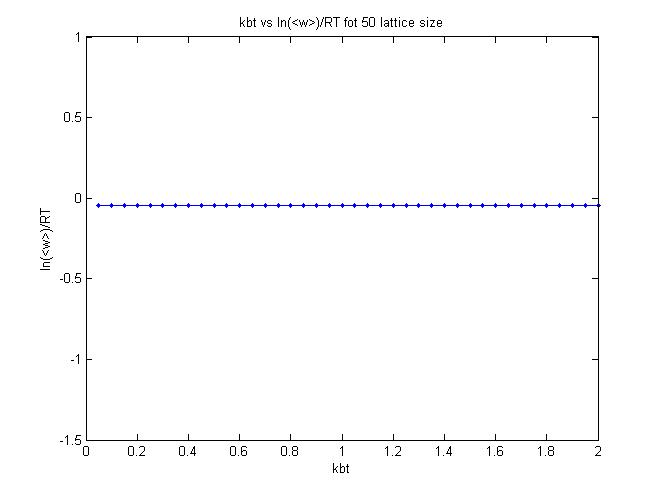
T

The results from magnetization as suggested by the plots cannot conclude on the phase transitions. The Wilson’s loop which is the probability weighted value of the placket expected to show the corresponding phase transitions.



The computational part the Wilson’s loop is to be taken as an arbitrary loop along the lattice sites. The product of the spins variables which lie on the loop is computed and is averaged over many Monte-Carlo loops. This is the Wilson’s loop average

**Plots:**



The above plot is obtained for a 50 by 50 placket size gauge with 10,000 Monte-Carlo sweeps, the Kbt was gradually varied by 0.005.

The ln (<W>)/RT turns out to be a constant having a value of about -0.14.The Theoretical results for a Gauged Ising Model explain that the plots follow “ **Area law**” at low temperatures i.e. The **ln (<W>)/RT** turns out to be a constant

for different Wilson loop sizes.

**Simulation**

**Monte Carlo methods**

Monte Carlo methods are a widely used class of computational algorithms for simulating the behavior of various physical and mathematical systems, and for other computations. Because of the repetition of algorithms and the large number of calculations involved, Monte Carlo is a method suited to calculation using a computer, utilizing many techniques of computer simulation. A Monte Carlo algorithm is often a numerical Monte Carlo method used to find solutions to mathematical problems (which may have many variables) that cannot easily be solved, for example, by integral calculus, or other numerical methods.

**Metropolis-Hastings algorithm**

In mathematics and physics, the **Metropolis-Hastings algorithm** is a rejection sampling algorithm used to generate a sequence of samples from a probability distribution that is difficult to directly sample from.

**The Algorithm**

The following is the algorithm used for the simulations.

**Start with a new config [n x n]**

**Choose flip point, flip this lattice point [p,q]**

**Compute the enrgy of this config [E1]**

**Compute the energy after the flip [E2]**

**Is E2 < E1**

**1)Accept the flip**

**2)Update E1=E2**

**3)Update the next flip point to the next lattice point**

**Is r1 > r2**

**1)Compute r1=**

**exp(-∆E/(Kb T)**

**2)Generate a random number between 0 and1.[=r2]**

**1)Do not accept the flip**

**2)Flip back to original config**

**3)Update the next flip point**

yes

yes

no

no

**Conclusion**

The plots of the “average spin” show the it is not the parameter which determines the phase transition for the 2-D Gauged Ising Model.

The importance of gauge theories for physics stems from the tremendous success of the mathematical formalism in providing a unified framework to describe the [quantum field theories](http://en.wikipedia.org/wiki/Quantum_field_theory) of [electromagnetism](http://en.wikipedia.org/wiki/Electromagnetism), the [weak force](http://en.wikipedia.org/wiki/Weak_force) and the [strong force](http://en.wikipedia.org/wiki/Strong_force).

The lattice gauge theories are studied using mathematical models which are simulated. The Random numbers are crucial in Monte Carlo Simulations and the choice of a good Random Number Generator dictates the correctness of the simulation.

**Appendix**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**#include <conio.h>**

**#include <math.h>**

**#if !defined( \_RNGS\_ )**

**#define \_RNGS\_**

**double Random(void);**

**void PlantSeeds(long x);**

**void GetSeed(long \*x);**

**void PutSeed(long x);**

**void SelectStream(int index);**

**void TestRandom(void);**

**#endif**

**main()**

**{ int conf[104][52],loop;unsigned int l;**

**int r,i,j,s,n,p,q,J=1;**

**int a,b,d,c,z,lp=10,l1,l2;**

**float spin=0,kbT,m[3000],R[500],loopavg;**

**double r1,r2,energy,power,chi,ham,fef=0,ftef=0,ef=0,tef=0,fef1,fef2;**

**FILE \*obj;**

**obj=fopen("lattice.dat","w");**

**clrscr ();**

**randomize();**

**printf ("enter no of boxes: ");**

**scanf ("%d",&n);**

**c=n+1;**

**d=2\*n+2;**

**for (j=1;j<=d;j++) /\*random matrix\*/**

**{**

**for (i=1;i<=c;i++)/\*printing the first conf and calculating the first spin\*/**

**{**

**r=rand();**

**if (r%2==1)**

**{**

**printf (" 1");**

**conf[j][i]=1;**

**spin=spin+1;**

**}**

**else**

**{**

**printf ("-1");**

**conf[j][i]=-1;**

**spin=spin-1;**

**}**

**}**

**printf ("\n");**

**}**

**/\*SPIN CALCULATION\*/**

**s=(c)\*(d);/\*2n+2\*/**

**printf("net spin of start config is %f\n",spin/s);**

**{**

**for(j=1;j<=2\*n+1;j=j+2)**

**{**

**for (i=1;i<=c;i++)**

**{ if(i==c)**

**{ /\*boundary conditions\*/**

**conf[j+1][n+2]=conf[j+1][1];**

**}**

**if(j==2\*n+1)**

**{ /\*boundary conditions\*/**

**conf[2\*n+3][i]=conf[1][i];**

**}**

**ef=(conf[j][i]\*conf[j+1][i]\*conf[j+2][i]\*conf[j+1][i+1]) ;**

**tef=tef+ef;**

**/\*subtract n from this tef later contrib for ef=ef-`1\*/**

**}**

**}**

**tef=tef\*-1; /\*total energy\*/**

**printf ("total enery at start(tef) %d(J)\n",tef);**

**}**

**printf("Working please wait..");**

**for(kbT=2;kbT>0.05;kbT=kbT-0.05)**

**{**

**z=1;**

**for(l=1;l<=10000;l++)**

**{**

**for(q=1;q<=d;q++)/\*is it not 2n+2,we are not flipping the inbetween rows \*/**

**{**

**for(p=1;p<=c;p++)**

**{**

**conf[q][p]=conf[q][p]\*-1 ;**

**if(p==c)**

**{ /\*boundary conditions\*/**

**conf[q+1][n+2]=conf[q+1][1];**

**}**

**if(q==2\*n+1)/\*each time it has to compute the value \*/**

**{ /\*boundary conditions\*/**

**conf[2\*n+3][p]=conf[1][p];**

**}**

**if (q%2==0)**

**{**

**if (p!=1)**

**{**

**fef1=(conf[q-1][p-1]\*conf[q][p-1]\*conf[q+1][p-1]\*conf[q][p]) ;**

**fef2=(conf[q-1][p]\*conf[q][p]\*conf[q+1][p]\*conf[q][p+1]) ;**

**}**

**else**

**{**

**fef1=(conf[q-1][n+1]\*conf[q][n+1]\*conf[q+1][n+1]\*conf[q][p]) ;**

**fef2=(conf[q-1][p]\*conf[q][p]\*conf[q+1][p]\*conf[q][p+1]) ;**

**}**

**}**

**else**

**{**

**if (q!=1)**

**{**

**fef1=(conf[q][p]\*conf[q+1][p]\*conf[q+2][p]\*conf[q+1][p+1]) ;**

**fef2=(conf[q-2][p]\*conf[q-1][p]\*conf[q][p]\*conf[q-1][p+1]) ;**

**}**

**else {**

**fef1=(conf[q][p]\*conf[q+1][p]\*conf[q+2][p]\*conf[q+1][p+1]) ;**

**fef2=(conf[2\*n+1][p]\*conf[d][p]\*conf[1][p]\*conf[d][p+1]) ;**

**}**

**}**

**/\*energy cntrib of flipped pt\*/**

**fef=2\*(fef1+fef2); /\*chng in energy after flip\*/**

**ftef=(tef+fef);/\*basic initial check\*/**

**if (fef<=0)**

**{**

**/\*it accepted changes\*/**

**tef=ftef;**

**}**

**else /\*second check compare exp wid random#\*/**

**{**

**power=(double)fef/kbT;**

**r1=exp(-1\*power);**

**/\*r=random(32000);\*/**

**r2=(double)Random();**

**if(r1>=r2)**

**{**

**tef=ftef;**

**}**

**else /\*rejected no changes to config\*/**

**{**

**conf[q][p]=conf[q][p]\*-1 ;**

**}**

**}**

**}**

**}**

**if (l>=2000&&l%100==0)**

**{ /\*lp=loop dimention also taking only five such loops\*/**

**loop=1;**

**for (l1=1;l1<=lp;l1++)**

**{**

**loop=conf[2\*lp+1][l1]\*conf[1][l1]\*loop;**

**}**

**for (l2=2;l2<=2\*lp;l2=l2+2)**

**{**

**loop=conf[l2][1]\*conf[l2][lp+1]\*loop;**

**}**

**R[z]=loop;**

**/\*printf(" chk pt 1 tef after 1 flip \n");\*/**

**spin=0;**

**for(b=1;b<=2\*n+2;b++)/\*2\*n+2\*/**

**{**

**for(a=1;a<=n+1;a++)/\*n+1\*/**

**{**

**spin=conf[b][a]+spin;**

**}**

**}Therefore the “average spin” is not the parameter which determines the phase transition.**

**m[z]=(float)spin/s;**

**z++;**

**}**

**}**

**spin=0;/\*net spin\*/**

**energy=0;/\*net energy\*/**

**z=z-1;/\*kk,so z=lmax /200+1\*/**

**for(a=1;a<=z;a++)**

**{**

**loopavg=0;**

**loopavg=R[a]+loopavg;**

**}**

**loopavg=(float)loopavg/z; /\*average spin on loop of size lp\*/**

**for(a=1;a<=z;a++)**

**{**

**spin=m[a]+spin;**

**}**

**spin=(float)spin/z;/\*spin is avg of m[]\*/**

**fprintf(obj,"%f, %.8lf, %.8lf,\n",kbT,spin,loopavg);/\*file \*/**

**printf(".");**

**}**

**printf(" done!!!\a");**

**fclose(obj);**

**getch();**

**}**

**#include <stdio.h>**

**#include <time.h>**

**#define MODULUS 2147483647 /\* DON'T CHANGE THIS VALUE \*/**

**#define MULTIPLIER 48271 /\* DON'T CHANGE THIS VALUE \*/**

**#define CHECK 399268537 /\* DON'T CHANGE THIS VALUE \*/**

**#define STREAMS 256 /\* # of streams, DON'T CHANGE THIS VALUE \*/**

**#define A256 22925 /\* jump multiplier, DON'T CHANGE THIS VALUE \*/**

**#define DEFAULT 123456789 /\* initial seed, use 0 < DEFAULT < MODULUS \*/**

**static long seed[STREAMS] = {DEFAULT}; /\* current state of each stream \*/**

**static int stream = 0; /\* stream index, 0 is the default \*/**

**static int initialized = 0; /\* test for stream initialization \*/**

**double Random(void)**

**/\* ----------------------------------------------------------------**

**\* Random returns a pseudo-random real number uniformly distributed**

**\* between 0.0 and 1.0.**

**\* ----------------------------------------------------------------**

**\*/**

**{**

**const long Q = MODULUS / MULTIPLIER;**

**const long R = MODULUS % MULTIPLIER;**

**long t;**

**t = MULTIPLIER \* (seed[stream] % Q) - R \* (seed[stream] / Q);**

**if (t > 0)**

**seed[stream] = t;**

**else**

**seed[stream] = t + MODULUS;**

**return ((double) seed[stream] / MODULUS);**

**}**