UNIVERSIDADE FEDERAL DO AMAZONAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARAMENTO DE APOIO A PESQUISA
PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

Termalização do Modelo de Ising pelo Algoritmo de Parellel Tempering

Bolsista: Rosmael Colsoul de Miranda, FAPEAM

MANAUS
2015

UNIVERSIDADE FEDERAL DO AMAZONAS
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
DEPARAMENTO DE APOIO A PESQUISA
PROGRAMA INSTITUCIONAL DE INICIAÇÃO CIENTÍFICA

RELATÓRIO FINAL
PIB-E/0002/2014

Termalização do Modelo de Ising pelo Algoritmo de Parellel Tempering

Bolsista: Rosmael Colsoul de Miranda, FAPEAM
Orientador: Prof. Dr. Octávio Daniel Rodriguez Salmon

MANAUS
2015

SUMARIO

1. INTRODUÇÃO

O estudo de modelos magnéticos é importante não só para aprofundar os conhecimentos teóricos do magnetismo, mas também por causa de suas diversas aplicações tecnológicas, que abrangem desde a fabricação de dispositivos eletrônicos até na teoria da escolha social.

Consequentemente, surge a necessidade de estudar o comportamento destes modelos em função da temperatura e de outros parâmetros que possam afetar o ordenamento magnético, o qual é definido em termos de variáveis de spin. Uma variável de spin é proporcional ao momento magnético de um sítio da rede cristalina. Para isto, define-se uma função chamada Hamiltoniano do sistema, o qual representa a energia de interação entre os spins da rede e com campos externos ou campos de anisotropia. Por conseguinte, para cada valor da temperatura, e dos campos respectivos, os spins procuram o equilíbrio termodinâmico ao atingir uma configuração cuja da energia seja mínima. O algoritmo de Metrópolis é um Método de Monte Carlo para simular a evolução temporal dos spins a uma temperatura finita. Quando o sistema atinge o equilíbrio, podem ser obtidas as médias térmicas relevantes, tais como a magnetização, a energia média, assim como o calor específico e a susceptibilidade.

O projeto tem como objetivo proporcionar uma iniciação nas técnicas de programação paralela afim de termalizar um sistema ferromagnético pelo algoritmo de Metrópolis, em uma rede quadrada e cúbica. O modelo mais simples para descrever um sistema ferromagnético, é o Modelo de Ising. O algoritmo de Metrópolis em sua versão mais simples, não é eficiente para induzir o sistema a visitar o maior número de estados possíveis a baixas temperaturas, a fim de equilibrá-lo. Este problema aumenta quando o modelo apresenta interações com "desordem temperada", tal como é o caso dos vidros de spins. Assim, surge a necessidade de implementar técnicas de programação paralela para evitar que o sistema fique preso em uma barreira de energia que o restringe a um número limitado de estados accessíveis. O algoritmo que permite esta melhora é chamado "Parallel Tempering", o qual precisa ser implementado utilizando uma linguagem com recursos de programação paralela.

2. REVISÃO BIBLIOGRÁFICA

Em física e demais ciências naturais, magnetismo é a denominação associada ao fenômeno ou conjunto de fenômenos relacionados à atração ou repulsão observada entre determinados objetos materiais, particularmente intensas aos sentidos nos materiais ditos ímãs ou nos matérias ditos ferromagnéticos, e ainda, em perspectiva moderna, entre tais materiais e condutores de correntes elétricas, especificamente entre tais materiais e portadores de carga elétrica em movimento, ou ainda a uma das parcelas da interação total (Força de Lorentz) que estabelecem entre si os portadores de carga elétrica quando em movimento, explicitamente a parcela que se mostra nula na ausência de movimento de um dos dois, ou de ambos, no referencial adotado. Há de se ressaltar que a simples observação de atração ou repulsão entre dois objetos não é suficiente para caracterizar a interação entre os dois como de origem magnética, geralmente confundindo-se com certa facilidade, aos olhos leigos, os fenômenos magnéticos e elétricos. Tais fenômenos elétricos e magnéticos, apesar de hoje saber-se estarem profundamente correlacionados, têm em princípio de naturezas certamente diferentes.

2.1. Ferromagnetismo

Ferromagnetismo é o mecanismo básico pelo qual certos materiais (como ferro) formam ímãs permanentes, ou são atraídos por ímãs. Na física, vários tipos diferentes de magnetismo são distinguidos. Ferromagnetismo (incluindo ferrimagnetismo) é o tipo mais forte e é responsável por fenômenos comuns do magnetismo encontradas na vida cotidiana. Outras substâncias respondem fracamente a campos magnéticos com dois outros tipos de magnetismo, o paramagnetismo, e o diamagnetismo, mas as forças são tão fracas que elas só podem ser detectadas por instrumentos sensíveis em um laboratório. Um exemplo corriqueiro de ferromagnetismo é um ímã de geladeira usado para guardar notas em uma porta do refrigerador.

Um material ferromagnético tem um momento magnético espontâneo – um momento magnético mesmo em um campo magnético aplicado igual a zero. A existência de um momento espontâneo sugere que os spins dos elétrons e os seus momentos magnéticos estão arranjados de uma maneira regular. Apenas algumas substâncias são ferromagnéticas, as mais comuns são o ferro, níquel, cobalto e suas ligas, alguns compostos de metais de terras raras, e alguns minerais de ocorrência natural, tais como magnetita.

Existem dois modelos teóricos que descrevem o ferromagnetismo que são o modelo de Ising e o modelo de Weiss onde ambos são baseados na hamiltoniana que utilizam grandes aproximações. Além dos modelos teóricos existem os de: Medição dos Campos, Temperatura de Curie.

2.2. Modelo de Ising

É um modelo simples que é usado para descrever um sistema magnético. Ele postula uma estrutura periódica d-dimensional, que é composta de dipolos magnéticos, cada um deles associado com um spin (S_i), em que "i" representa a estrutura da rede. Supõe-se que os spins podem ser para cima ou para baixo, com o valor associado de +1 para cima e -1 associado para baixo. A hamiltoniana de um sistema é:

$$E = H = -J \sum_{<kl>}^{N} S_k S_l - B \sum_{k}^{N} S_k$$

Onde J é uma constante de acoplamento, <kl> é uma soma sobre vizinhos mais próximos, B é um campo magnético externo, e N é o número de dipolos magnéticos no sistema. A constante de acoplamento J indica a resistência da ligação entre os spins adjacentes e dependendo do seu sinal, o sistema pode preferir ferromagnetismo (J > 0) ou anti-ferromagnetismo (J < 0). A primeira solução do modelo de Ising foi dada por si só Ernest Ising, mas apenas para problemas uni dimensionais (d=1).

Na primeira dimensão o modelo de Ising não apresenta uma transição de fase, enquanto que nas dimensões mais elevadas existe a TC temperatura de transição, acima do qual o parâmetro de ordem é zero. Parâmetro de ordem para o ferromagnetismo acoplamento ferromagnético é a magnetização M.

$$M = \sum_{i} S_i$$

3. MÉTODOS UTILIZADOS

Para descrever o sistema ferromagnético esta sendo utilizado o modelo de Ising que é definido numa coleção discreta de variáveis denominada *spins*, que podem assumir os valores +1 ou −1. Os spins S_i interagem em pares, com a energia possuindo um valor quando os dois spins são iguais e um segundo valor quando diferentes.

Considerando um aglomerado de spins que estão interagindo sob efeitos coletivos, ao elevar a temperatura gradativamente do sítio onde os spins estão localizados, estes começam a perder seus efeitos coletivos (começar a perder o fenômeno de histerese), e os mesmos com o aumento da temperatura começam a se desemparelhar, ou seja, começar a ter efeitos aleatórios. Próximo da temperatura crítica T_c os spins já estão totalmente desemparelhados, ou seja, distribuídos aleatoriamente.

O modelo de Ising estuda exatamente os efeitos que ocorrem próximos da temperatura T_c, o que apesar de ser um modelo simples (não trivial) nos mostrar bastante informação sobre o sistema desejado (onde ocorrem efeitos aos pares).

Usando o modelo de Ising e modelando de forma a utilizar o algoritmo de Metrópolis que é um Método de Monte Carlo para simular a evolução temporal dos spins a uma temperatura finita, foi possível implementar um algoritmo na linguagem de programação C, afim de gerar arquivos no formato txt ou dat para a construção das curvas descritas pela magnetização por spin versus o tempo n, e da magnetização média em relação a temperatura. Este algoritmo é compilado na CPU (Unidade Central de Processamento) que por muitas vezes dependendo da arquitetura da rede, leva muito tempo para compilar. Além disso o algoritmo de Metrópolis em sua versão mais simples, não é eficiente para induzir o sistema a visitar o maior número de estados possíveis a baixas temperaturas, a fim de equilibrá-lo. Este problema aumenta quando o modelo apresenta interações com desordem temperada. Assim, surge a necessidade de implementar uma técnica de programação paralela para evitar que o sistema fique preso em uma barreira de energia.

Para melhorar o modelo de Ising será usado o algoritmo de Parallel Tempering, que é um algoritmo que é implementado utilizando uma linguagem com recursos de programação paralela.

A plataforma de computação paralela usada é o CUDA (Compute Unified Device Architecture) que é um modelo de programação inventado pela NVIDIA. Ela permite aumentos significativos de performance computacional ao aproveitar a potência da unidade de processamento gráfico (GPU), muito usada para solucionar problemas computacionais complexos em uma fração do tempo necessário em uma CPU. O CUDA permite a programação em linguagem C e C++.

Modelando o modelo de Ising para o Algoritmo de Parallel Tempering, para simular a evolução temporal dos spins a uma temperatura finita, foi possível implementar um algoritmo na linguagem de programação C++, afim de gerar arquivos no formato txt ou dat para a construção das curvas descritas pela magnetização por spin versus o tempo n, e da magnetização média em relação a temperatura. E assim comparar o tempo de execução do algoritmo compilado usando a GPU com o algoritmo usando a CPU.

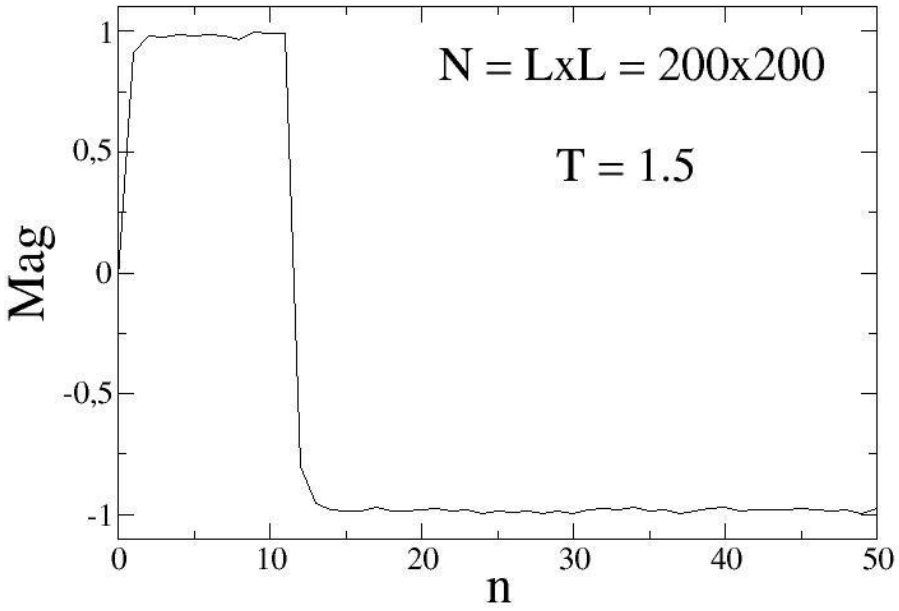
4. RESULTADOS E DISCUSSÕES

Os algoritmos implementados criam arquivos no formato txt e dat na forma de uma tabela, onde podemos verificar a variação da magnetização, do tempo, e de n (número de passos de Monte Carlo). O código em C, e o arquivo dat está disponibilizado no Apêndice.

Usamos o arquivo gerado em outra ferramenta computacional de nome Grace, que é um aplicativo que gera gráfico (existem vários outros aplicativos para essa finalidade), onde através dela poderá ser observado o comportamento da magnetização.

Agora podemos ver dois gráficos, e analisar o comportamento da magnetização em relação ao tempo n, e da magnetização media após atingir o equilíbrio em relação a temperatura.

A primeira analise será feita no gráfico da magnetização versus o tempo n, que pode ser observada na figura (1.0).

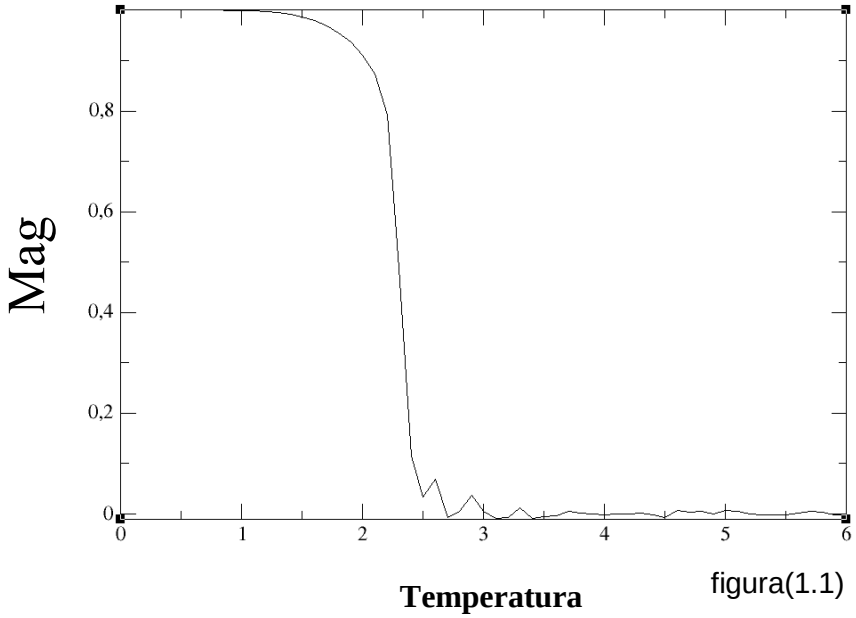


figura(1.0)

A figura (1.0) descreve a magnetização por spin versus n para L=200 (40000 spins) e T=1.5.

Pode ser observado que o sistema atinge o equilíbrio para $10 < n < 15$, porque a magnetização começa a ser estável, e só tem pequenas flutuações ao redor de um valor médio. Esse é o valor de equilíbrio da magnetização para a temperatura T=1.5.

O gráfico da figura (1.1) representa a magnetização média versus a temperatura.



figura(1.1)

Podemos observar que a magnetização média tem pequenas oscilações em torno de zero com o aumento da temperatura.

Foi constatado durante o projeto que o tempo de execução na CPU é muito maior do que a de GPU e a diferença aumentam com o tamanho da estrutura. A GPU supera a CPU na simulação do modelo de Ising for fator de 50 a 200, dependendo do tamanho da estrutura. Para tamanhos de redes maiores (mais de cerca de 50x50) a diferença de desempenho cresce a favor da GPU.

5. CONCLUSÕES

O presente projeto contribuiu satisfatoriamente em proporcionar uma iniciação nas técnicas de programação paralela, bem como no aprofundamento de conhecimentos e técnicas em um campo muito importante da física que o Magnetismo. Podemos concluir ainda que o GPU supera a CPU na eficiência, tempo de execução, em muitos problemas computacionais complexos, isto, pois utiliza a técnica de programação paralela. Sendo assim o algoritmo de Parallel Tempering é mais eficiente que o algoritmo de Metrópolis na termalização do modelo de Ising, pois, usa a técnica de programação paralela o que diminiu o tempo de execução, bem como resolve o problemas do algoritmo de Metrópolis que não é eficiente para induzir o sistema a visitar o maior número de estados possíveis a baixas temperaturas, a fim de equilibrá-lo.

6. FONTES E REFERENCIAS BIBLIOGRÁFICAS

[1] The CUDA Handbook: A Comprehensive Guide to GPU Programming, Nicholas Wilt, Addison-Wesley,First Edition, 2013.

[2] Guide to Monte Carlo Simulations in Statistical Physics, David, P. Landau, K. Binder 2a Edição , 2005.

[3] Parallel tempering: Theory, applications, and new perspectives , David J. Earlab and Michael W. Deema, Phys. Chem. Chem. Phys., 2005,7, 3910-3916.

[4] NVIDIA, NVIDIA CUDA C Programming Guide, (versão 3.2)

7. CRONOGRAMA EXECUTADO

Nº	Descrição	Ago 2014	Set	Out	Nov	Dez	Jan 2015	Fev	Mar	Abr	Mai	Jun	Jul
1	Estudo da linguagem CUDA e do Modelo de Ising	**x**	**x**	**x**	**x**								
2	Estudo do Modelo de Ising pelo algoritmo de Metrópolis tradicional					**x**							
3	Implementação do Algoritmo de Parallel Tempering no modelo de Ising nas redes quadrada e cúbica.						**x**	**x**	**x**				
4	Análise dos resultados.									**x**	**x**	**x**	
5	- Elaboração do Resumo e Relatório Final (atividade obrigatória) - Preparação da Apresentação Final para o Congresso (atividade obrigatória)												X

8. APÊNDICE

Programa em C e arquivo dat para a magnetização versus n.

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<time.h>
#include "aleatorio.h"
#define L 200

int** aloca_matriz() {
      int** mat = (int**) malloc (sizeof(int*)*L);
      int i;
      for (i = 0; i < L; i++)
                  mat[i] = (int*) malloc (sizeof(int));

      return mat;
         }

int indice(int n) // funÃ§Ã£o que devolve o indice  considerando
a periodicidade da rede
{
  if(n<0) return(L-1);
  if(n=L) return(0);

   return(n);
}

float  magnetizacao(int** mat){ // funÃ§Ã£o que calcula a
magnitude e retorna a magnitude dividida por N = L*L, onde L Ã©
o tamanho da matriz
      int m = 0;
      int i, j;
      for (i = 0; i < L; i++) {
            for(j = 0; j < L; j++) {
                  m += mat[i][j];
            }
      }

      float divisao = (float) m / (L * L);

      return divisao;

}

int main()
{
 long int sem;
  float dE,T=1.5;
int i, j,n,total=50;// o "n" equivale a uma unidade de tempo e
total Ã© o tempo total
int** mat;
FILE *arquivo;
```

```
arquivo=fopen("mag_pasoL200T1y5.dat","w");
sem=-time(NULL); //Gera uma 'random seed' baseada no retorno da
funcao time()

mat = aloca_matriz();

for (i = 0; i < L; i++)
          for(j = 0; j < L; j++)
                    if (ran1(&sem) < 0.5)
               mat[i][j]=1;
                       else mat[i][j]=-1; // inicializa a
matriz de spins

 fprintf(arquivo, "%d\t%f\n",n, magnetizacao(mat));

for(n = 1; n <=total; n ++)
  {
    for (i = 0; i < L; i++)
     for(j = 0; j < L; j++)
              {
                dE=
2*mat[i][j]*(mat[indice(i+1)][j]+mat[indice(i-
1)][j]+mat[i][indice(j+1)]+mat[i][indice(j-1)]); // variaÃ§Ã£o
da energia de um spin apÃ³s ser invertido
                 if(dE <  0) mat[i][j]=-mat[i][j];
                   else
                      if(ran1(&sem) < exp(-dE/T)) mat[i][j]=-
mat[i][j];
               }
               fprintf(arquivo, "%d\t%f\n",n,
magnetizacao(mat));
         }

free(mat);
fclose(arquivo);
return(0);
}
```

Imagem do arquivo dat para a magnetização versus n, onde a primeira coluna (coluna da esquerda) apresenta os valores de n, e a segunda apresenta os valores da magnetização.

```
0     -0.022700
1     0.913050
2     0.980850
3     0.975650
4     0.984900
5     0.977250
6     0.984800
7     0.980650
8     0.966200
9     0.997500
10    0.988750
11    0.991150
12    -0.802550
13    -0.951750
14    -0.979500
15    -0.987450
16    -0.985000
17    -0.969600
18    -0.983150
19    -0.987500
20    -0.980000
21    -0.973900
22    -0.985000
23    -0.982450
24    -0.997500
25    -0.984050
26    -0.989900
27    -0.982500
28    -0.993750
29    -0.987600
```

Programa em C e arquivo dat para a magnetização versus o tempo.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
#include "aleatorio.h"
#define max 100
#define L 40
#define N 1600

long int sem;
int S[max][max];

void inicializar();
int indice(int n);
```

```
main()
{
double T,dE,prob,m,magmed;
int i,j,paso,pasoseq=1000,pasosmed=1000;
FILE *arquivo;

arquivo=fopen("dados.dat","w");

for(T=0.1;T<=6.0;T=T+0.1)

{
inicializar();

sem=-time(NULL);
for(paso=1;paso<=pasoseq;paso=paso+1)
{
  for(i=1;i<=L;i=i+1)
    for(j=1;j<=L;j=j+1)
     {
       dE=2*S[i][j]*(S[indice(i-
1)][j]+S[indice(i+1)][j]+S[i][indice(j+1)]+S[i][indice(j-1)]);
       if(dE < 0 ) S[i][j]=-S[i][j];
       else if(dE>0)
            {
              prob=exp(-dE/T);
                if(ran1(&sem) < prob) S[i][j]=-S[i][j];
             }
      }

}

magmed=0;

for(paso=1;paso<=pasosmed;paso=paso+1)
{
  for(i=1;i<=L;i=i+1)
    for(j=1;j<=L;j=j+1)
     {
       dE=2*S[i][j]*(S[indice(i-
1)][j]+S[indice(i+1)][j]+S[i][indice(j+1)]+S[i][indice(j-1)]);
       if(dE < 0 ) S[i][j]=-S[i][j];
       else if(dE>0)
            {
              prob=exp(-dE/T);
                if(ran1(&sem) < prob) S[i][j]=-S[i][j];

             }

      }

 m=0;
   for(i=1;i<=L;i=i+1)
    for(j=1;j<=L;j=j+1)
      m=m+S[i][j];
```

```
   m=m/N;

 magmed=magmed+m;
}

magmed=magmed/pasosmed;

fprintf(arquivo,"%lf\t%lf\n",T,magmed);

}
fclose(arquivo);
return(0);
}

void inicializar()
{
int i,j;

for(i=1;i<=L;i=i+1)
for(j=1;j<=L;j=j+1)
S[i][j]=1;

}

int indice(int n)
{
  if(n == 0) return (L);
   if(n==L+1) return(1);
   return(n);
}
```

Imagem do arquivo dat da magnetização versus o tempo, onde a coluna da esquerda corresponde ao tempo e o da direita a magnetização.

```
0.100000   1.000000
0.200000   1.000000
0.300000   1.000000
0.400000   1.000000
0.500000   1.000000
0.600000   0.999999
0.700000   0.999977
0.800000   0.999904
0.900000   0.999701
1.000000   0.999264
1.100000   0.998466
1.200000   0.997070
1.300000   0.994762
1.400000   0.991332
1.500000   0.986731
1.600000   0.978640
1.700000   0.969370
1.800000   0.954334
1.900000   0.937421
2.000000   0.910289
2.100000   0.873909
2.200000   0.791911
2.300000   0.487628
2.400000   0.115172
2.500000   0.032993
2.600000   0.068214
2.700000   -0.007285
2.800000   0.005103
2.900000   0.036566
3.000000   0.005037
```