Projeto 05: Simulações de Monte Carlo

7600073 - Física Estatística Computacional - 2024/01
20/05/2024

Prof. Dr. Francisco Castilho Alcaraz
Guilherme Santana de Almeida (12694668)

Resumo

Neste projeto estudaremos simulações de Monte Carlo no contexto de modelos de Spin, mas precisamente do Modelo de Ising Bidimensional. Calculamos magnetização, energia e capacidade térmica, varrendo uma gama de processos, desde recozimento, têmpera e loops térmicos. Também determinamos computacionalmente a temperatura crítica (T_c) do sistema e vimos a relação entre transições de fase e o tamanho do sistema.

Introdução

Simulações de Monte Carlo utilizam de números aleatórios para o cálculo de médias, de certa forma temporais. Essas simulações apresentam uma nova dimensão de tempo, a do tempo de Monte Carlo, e é nela que nossas médias são feitas. Ou seja, se simularmos sistemas físicos estaremos realizando uma dinâmica num tempo não-real. Ainda assim, estas simulações se fazem muito úteis, principalmente na termodinâmica como veremos nesse projeto. Como objeto de estudo dessas simulações vamos usar o...

...Modelo de Ising. Ele foi idealizado como um modelo matemático para o estudo do ferromagnetismo. Em duas dimensões, temos uma grade quadrada de comprimento L com $N = L^2$ sítios que podem assumir dois valores, -1 e 1, representando os spins de um sistema, e podem interagir com seus vizinhos imediatos. A Hamiltoniana H (com campo magnético zero) desse sistema é dada por:

$$H = -J \sum_{\langle ij \rangle} \sigma_i \sigma_j \tag{1}$$

onde a soma é sobre os vizinhos (esquerda, direita, cima e baixo). O estado em cada sítio é dado por $s(x, y) = 1$ ou -1, onde $x, y = 1, ..., L$. Com isso, podemos escrever explicitamente a energia total do sistema numa dada configuração:

$$E = -\frac{J}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} s(i,j)[s(i-1,j) + s(i,j+1) + s(i,j-1) + s(i+1,j)] \tag{2}$$

Disso, tiramos as seguintes conclusões: A energia mínima é $-2JN$, e acontece quando todos os spins estão orientados, todos 1 ou -1. Num magneto, essa orientação (ou não) dos spins está relacionada com a temperatura na qual os sistema se encontra. Logo, podemos recuperar o Ensemble Canônico da termodinâmica, e verificar que a probabilidade não

normalizada (também chamada de fator de Boltzman) do sistema se encontrar numa configuração com energia E é

$$P(E_i) \approx e^{-E_i/T} \approx e^{-\beta E_i} \quad (k_B = 1) \tag{3}$$

com $\beta = \frac{1}{T}$ e a normalização dada pela função partição

$$Z = \sum_i e^{-\beta E_i} \tag{4}$$

A dinâmica de Monte Carlo que executaremos é chamada de Heat Bath (Banho Térmico) e ela começa da seguinte forma: Escolhemos aleatoriamente um sítio (i, j), e nessa configuração a energia total do sistema é E_i com probabilidade $P(E_i)$. Se "fliparmos"o spin, fazendo $s(i, j) = -s(i, j)$ a energia total agora será E_f com probabilidade $P(E_f)$. Acontece que, se quisermos comparar essas duas configurações, só precisamos olhar para o sítio escolhido e seus vizinhos, afinal o único termo diferente é $s(i, j)$. Então, definindo $H = J[s(i-1, j) + s(i, j+1) + s(i, j-1) + s(i+1, j)]$, as probabilidades são dadas por

$$P(E_i) = \frac{e^{s\beta H}}{Z} \quad , \quad P(E_f) = \frac{e^{-s\beta H}}{Z} \tag{5}$$

e a normalização depende só das duas configurações da seguinte forma

$$Z = e^{s\beta H} + e^{-s\beta H} \tag{6}$$

Esse "flip"do spin será concretizado depois de compararmos essas probabilidades com um número aleatório v entre 0 e 1, caso v seja maior que $P(E_i)$, por exemplo, flipamos o spin. Todo esse processo é realizado N vezes (número total de sítios), escolhendo aleatoriamente um sítio a cada iteração, e após essas N iterações podemos dizer que uma unidade de tempo (ou uma iteração) de Monte Carlo se passou. Nossa simulação será concluída após um número necessário de iterações de Monte Carlo.

Tarefa A

Nessa tarefa, simularemos a dinâmica de Monte Carlo para $L = 60, 100$ e escalas de temperatura bastante distintas, fazendo $\beta = 3$ e $\beta = 0.1$. **E a partir daqui, fixamos J = 1 durante todo o projeto!** Aproveito para definir a magnetização por sítio $m = M/N$ do sistema, onde M é igual

$$M = \sum_i \sigma_i = \sum_{i=1}^{L} \sum_{j=1}^{L} s(i, j) \tag{7}$$

Essa será a quantidade termodinâmica que veremos aqui. Essa e outras quantidades termodinâmicas precisam ser calculadas como médias, como já dito antes. Essas médias precisam começar a serem feitas após o sistema termalizar, ou seja, atingir o equilíbrio. Aqui isso foi feito arbitrariamente, ou seja, realizamos um número razoável de iterações de termalização para depois fazer iterações em que de fato calculamos essas quantidades.

Nesta tarefa, esse número de iterações de termalização é mais arbitrário ainda, e isso vem das escalas de temperatura que $\beta = 3$ e $\beta = 0.1$ apresentam.

Voltamos a equação 3. No regime de baixas temperaturas $E_i >> k_B T$, a probabilidade de achar o sistema numa configuração é máxima se a energia dela for mínima (todos os spins orientados), então é mais provável achar poucos sítios com spin oposto a seus vizinhos. Já no regime de altas temperaturas $E_i << k_B T$ temos o oposto, os sítios preferem estar com o spin contrário a seus vizinhos, e a rede fica desordenada.

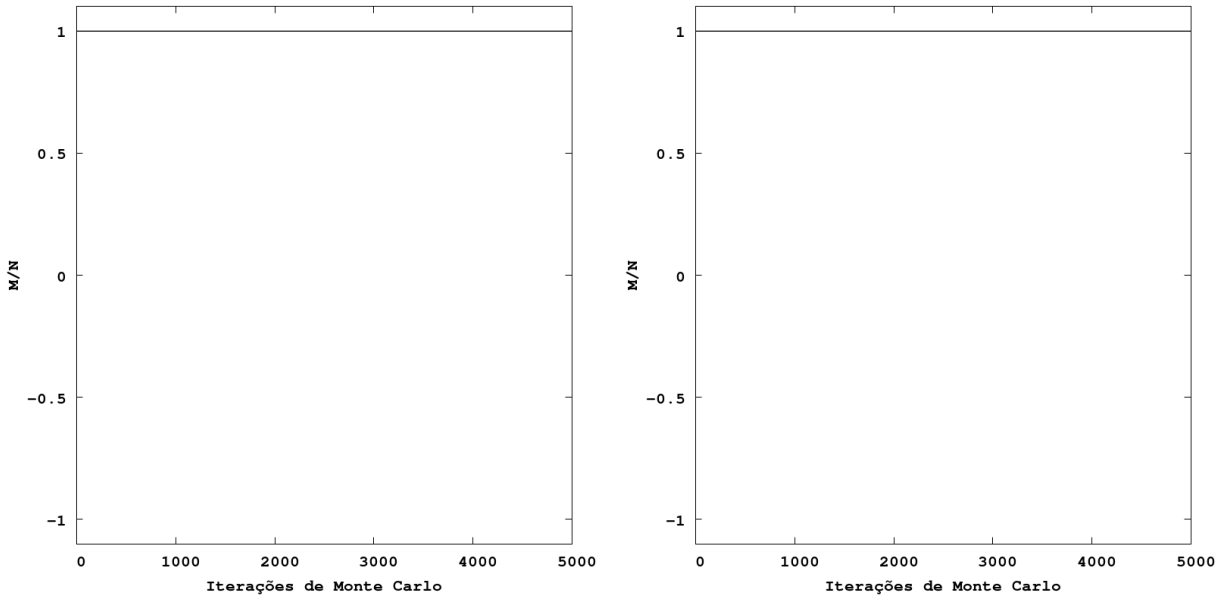
Logo, para $J = 1$:

$$\beta = 3 \Rightarrow \text{Baixa Temperatura} \Rightarrow \text{Ordem}$$
$$\beta = 0.1 \Rightarrow \text{Alta Temperatura} \Rightarrow \text{Desordem}$$

Então esperamos que $M = 1$ para $\beta = 3$ e $M = 0$ para $\beta = 0.1$.

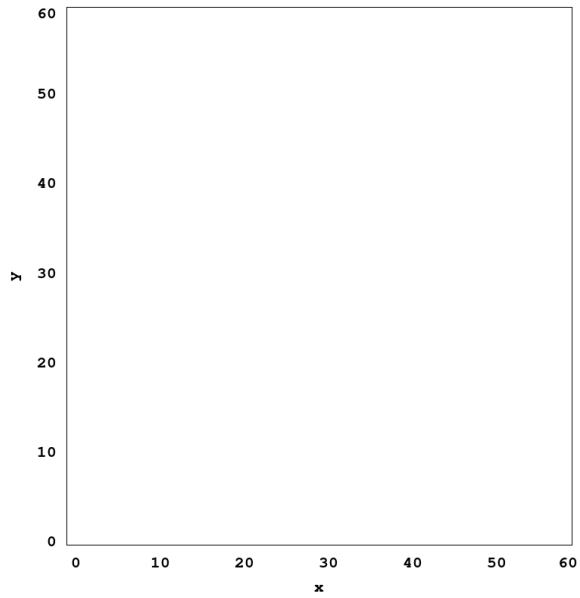
(A1) $\beta = 3$

Como estamos começando a grade com todos os spins orientados na mesma direção $M = 1$, para $\beta = 3$ esperamos que **o sistema continue ordenado** e varie praticamente nada, já que ele está na configuração mais provável. Nos gráficos de configuração abaixo, branco significa spin $= 1$, e preto spin $= -1$.

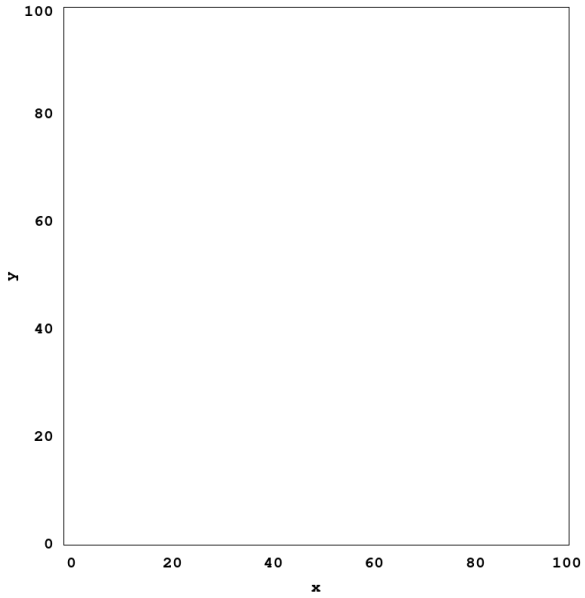


(a) L = 60 **(b)** L = 100

Figura 1: Magnetização de um sistema de spins bidimensional partindo de uma configuração totalmente ordenada com temperatura $\beta = 3$.



(a) L = 60

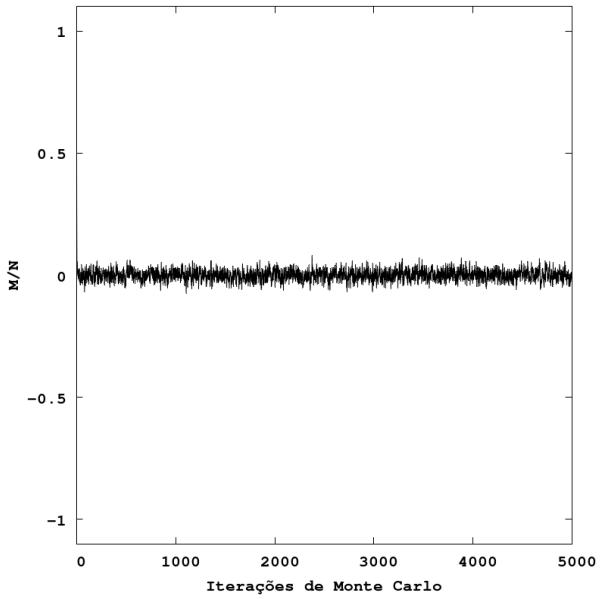


(b) L = 100

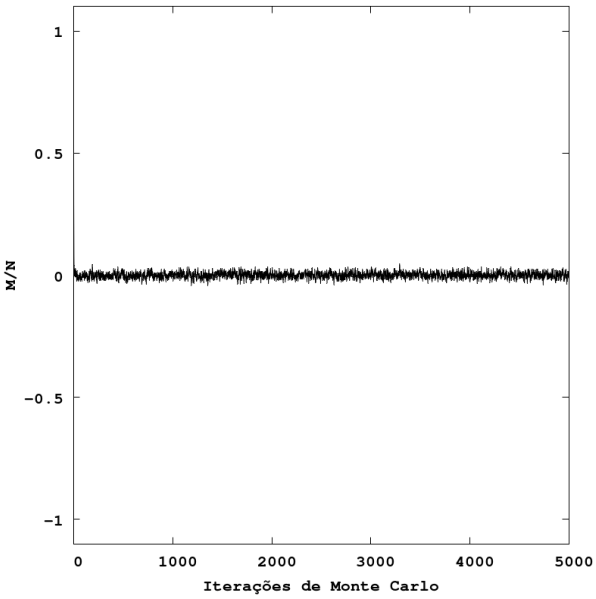
Figura 2: Configurações finais de um sistema de spins bidimensional partindo de uma configuração totalmente ordenada com temperatura $\beta = 3$.

(A2) $\beta = 0.1$

Para $\beta = 0.1$ **o sistema transicionará para uma configuração desordenada**. E isso ocorrerá muito rapidamente, não sendo necessário muitas iterações de termalização, dado em vista que $\beta = 0.1$ é uma temperatura muito alta.

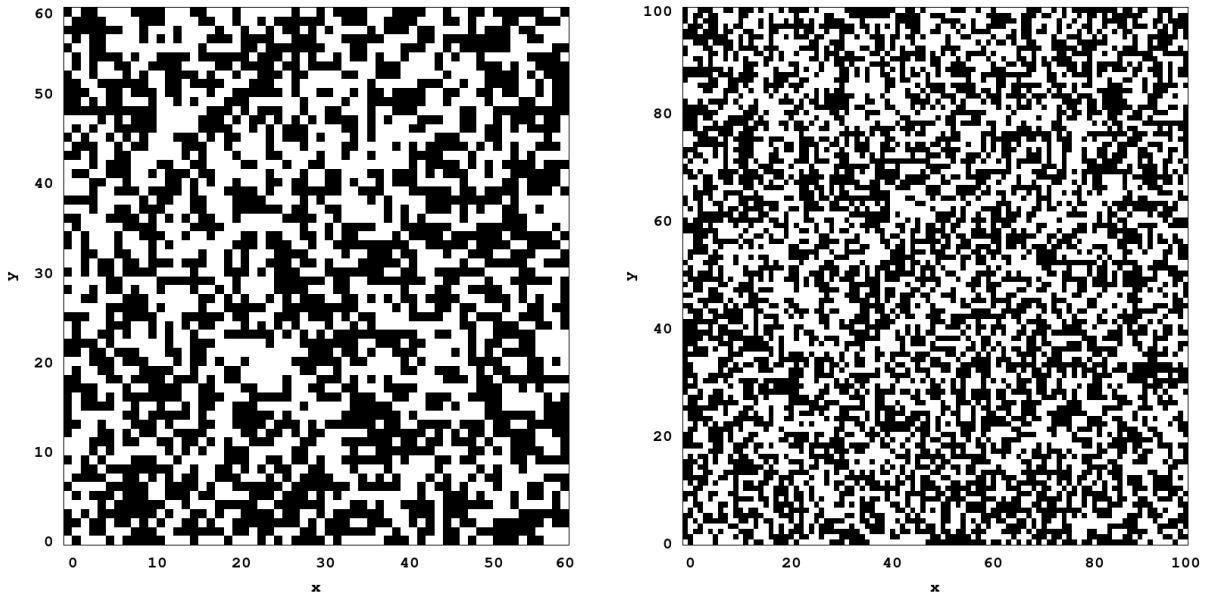


(a) L = 60



(b) L = 100

Figura 3: Magnetização de um sistema de spins bidimensional partindo de uma configuração totalmente ordenada com temperatura $\beta = 0.1$.



(a) L = 60 **(b)** L = 100

Figura 4: Configurações finais de um sistema de spins bidimensional partindo de uma configuração totalmente ordenada com temperatura $\beta = 0.1$.

Tarefa B

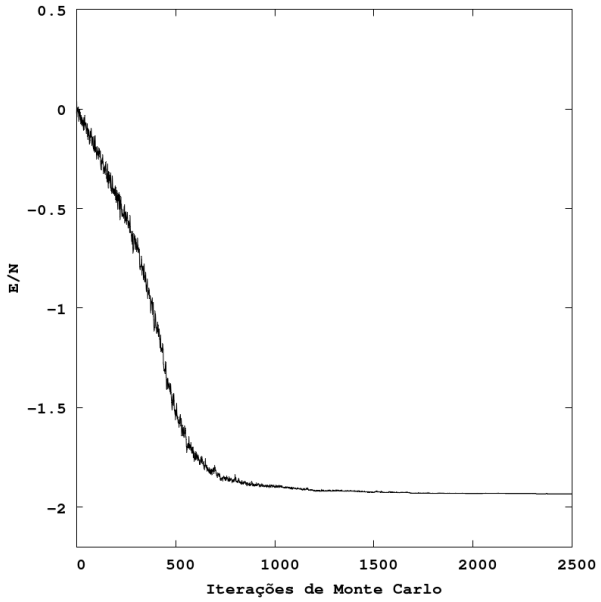
Desta vez, realizamos dois processos térmicos chamados *Annealing* (Recozimento) e *Quenching* (Têmpera). Em ambas, a configuração inicial corresponde a uma temperatura infinita, ou seja, com o sistema totalmente desordenado e aleatório. E com grade de tamanho $L = 60$.

(B1) Recozimento

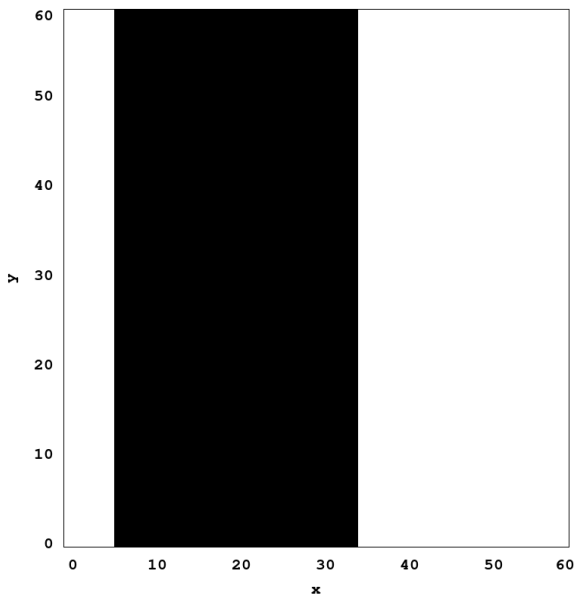
Para realizar o recozimento, começamos com um temperatura correspondente à configuração inicial de mesma temperatura, ou seja, $\beta = 0$. A cada iteração de Monte Carlo aumentamos a temperatura de $\Delta\beta = 0.001$, até atingirmos $\beta = 3$. Então, no total faremos 3000 iterações de Monte Carlo.

Esse processo simula o tratamento feito com metais, no qual se aquece o metal ou liga desejado até uma certa temperatura, e logo em seguida o resfria lentamente até a temperatura ambiente. O metal ou liga resultantes dependem da temperatura inicial e taxa de resfriamento utilizadas, podendo formar diferentes tipos de estruturas cristalinas. Aqui, começamos à temperatura infinita e resfriamos de $\Delta\beta$ no tempo irreal de Monte Carlo, qual será o metal que surgiria disso?

Abaixo seguem os gráficos da densidade de energia E/N e a configuração final atingida. O gráfico de energia foi cortado um pouco para melhor visualizar a região de termalização. É importante notar a rapidez com que os spins se alinham, para compararmos com o próximo processo. Vemos que após boa parte dos spins se alinharem, forma-se um platô, tão longo quanto, seguido logo após pela termalização.



(a) Energia por sítio.



(b) Configuração final.

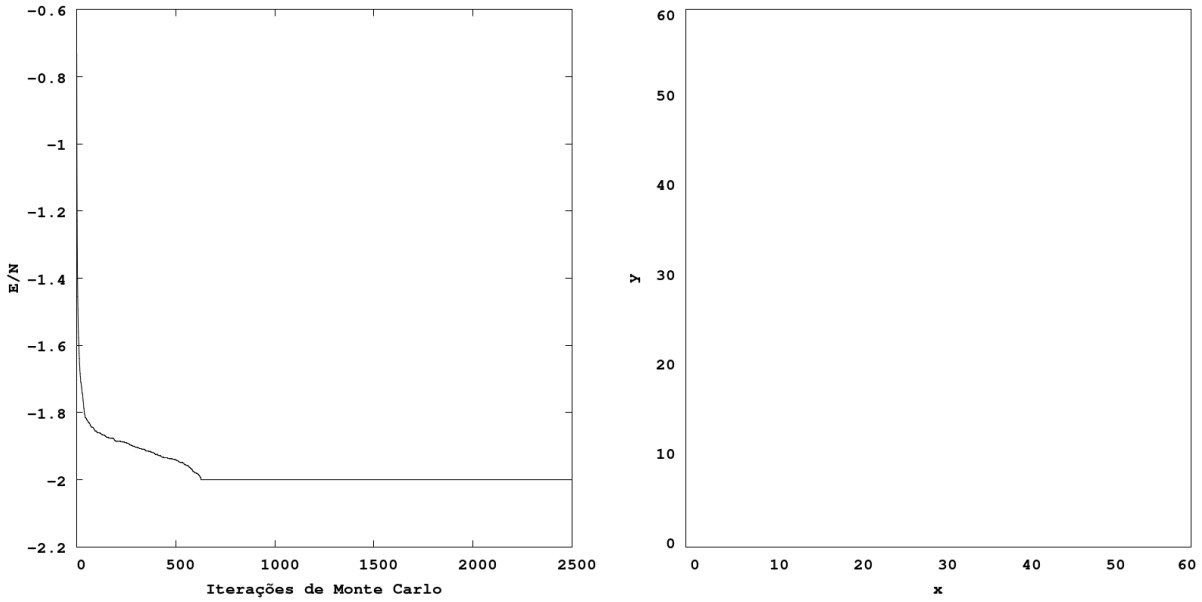
Figura 5: Processo de recozimento para uma grade de tamanho $L = 60$, com configuração inicial à temperatura infinita $\beta = 0$ e temperatura final $\beta = 3$, com passos de $\Delta\beta = 0.001$.

A configuração em faixas vista acima é fruto da simulação computacional. Na vida real não existe magneto com esse tipo de estrutura. Essa configuração também não dura para sempre, só é preciso que um dos spins das bordas de cada região flipem, para todo o sistema ficar alinhado, e para isso é preciso esperar um tempo de Monte Carlo maior. Esse fenômeno computacional é provavelmente o que ocasiona flutuações na capacidade térmica à baixas temperaturas, que veremos na tarefa C.

(B2) Têmpera

Na têmpera, resfriamos o metal rapidamente, o submergindo em água ou óleo logo após sair do forno. Isso é útil quando esquentamos nosso material de tal forma que atingimos a estrutura cristalina desejada. Para manter essa estrutura, criamos um choque térmico no material. Aqui isso equivale a começar com uma configuração de temperatura alta, mas logo na primeira iteração de Monte Carlo, reduzimos essa temperatura e a mantemos fixa enquanto nosso sistema evolui.

Ou seja, fixamos $\beta = 3$ durante toda a dinâmica, apesar de começarmos com a configuração inicial correspondente a $\beta = 0$. Por consistência, também faremos 3000 iterações de Monte Carlo nesse processo.



(a) Energia por sítio. **(b)** Configuração final.

Figura 6: Processo de têmpera à temperatura fixa $\beta = 3$ para uma grade de tamanho $L = 60$, com configuração inicial à temperatura infinita $\beta = 0$.

Perceba que a energia cai muito mais rapidamente, ou seja, um quasi-equilíbrio é atingido imediatamente após o início do processo, seguido por um platô longo, e finalmente a termalização.

Tarefa C

O processo que será visto agora evidencia a física ao redor do ponto crítico do sistema, que denotamos β_c (ou T_c). Nele ocorre o que se chama transição de fase, que é caracterizada pela descontinuidade na energia em função da temperatura (também chamada parâmetro de ordem), ou de suas derivadas. Caso a energia for descontínua, temos uma transição de fase de primeira ordem. Caso a energia seja continua, mas a derivada a respeito da temperatura, também conhecida como capacidade térmica, for descontínua, temos uma transição de fase de segunda ordem, e assim por diante.

Agora vamos pensar no processo dessa tarefa, o loop térmico. O que queremos fazer é iniciarmos uma configuração a temperatura infinita ($\beta = 0$) e diminuirmos essa temperatura até próximo do zero absoluto, para então aumentá-la de novo à temperatura infinita.

Quando diminuímos a temperatura, os spins começam a se organizar, porém, a medida que chegamos à temperatura final e imediatamente começamos a aquecer novamente, o sistema não teve tempo de termalizar, e alguns sítios vão estar bem alinhados com seus vizinhos, enquanto outros não tiveram esse tempo para socializar. E na volta, aumentando a temperatura, esses sítios alinhados não vão ter tempo de brigar com seus vizinhos completamente. Perceba que, nesses regimes de altas e baixas temperaturas, o que importa são apenas os vizinhos próximos, sítios distantes não influenciam a si mesmos. Na verdade, a distância de influência que cada sítio possui chama-se comprimento de cor-

relação, e ela é máxima na temperatura crítica. Logo, como temos duas configurações diferentes vindas de cada lado, a energia toma caminhos diferentes também. Para cada ciclo de esquenta ou esfria, o gráfico da energia é parecido com 5, mas a figura que surge devido esse comportamento é chamada **histerese**. Histerese também é definida como a dependência do sistema com seu histórico, e é um comportamento visto em diversas outras áreas como economia e biologia.

(C1)

Para $L = 60, 80$ e 100 e $\Delta\beta = 0.001$ e 0.0001, vamos realizar o loop térmico na faixa $\beta = (0,\ 1.75)$. O objetivo é evidenciar a histerese, pois com ela podemos calcular a temperatura crítica, que se encontra em seu centro. E devido a física já discutida, fica claro que esperamos que a região de histerese se alargue conforme se aumenta $\Delta\beta$.

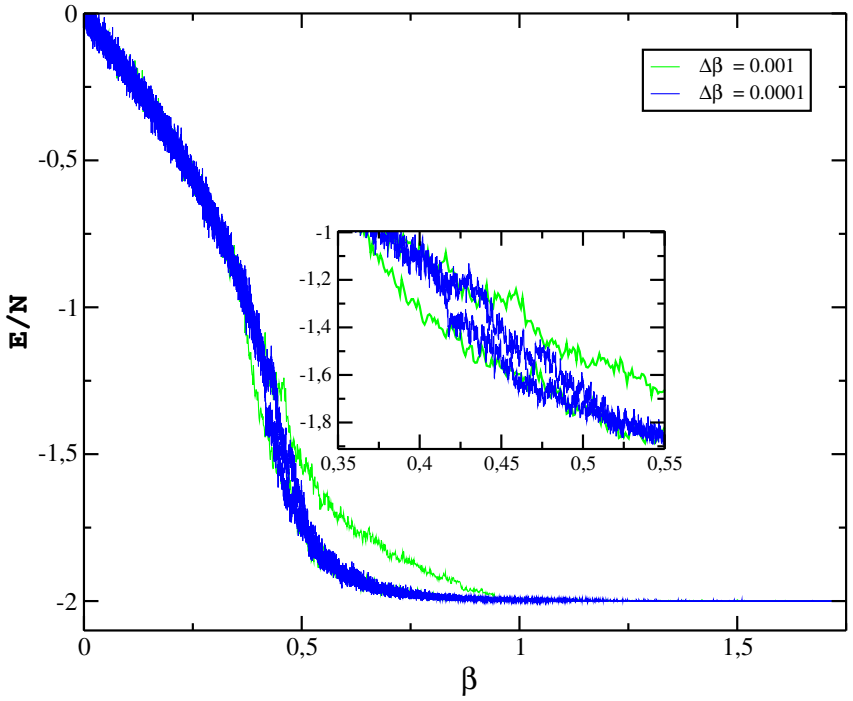


Figura 7: Histereses para L = 60.

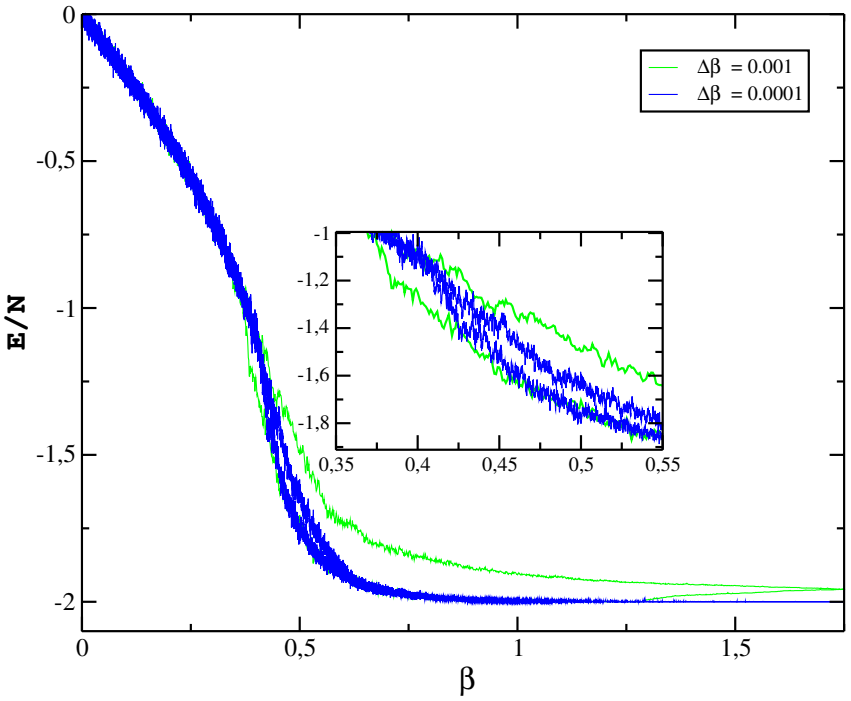


Figura 8: Histereses para L = 80.

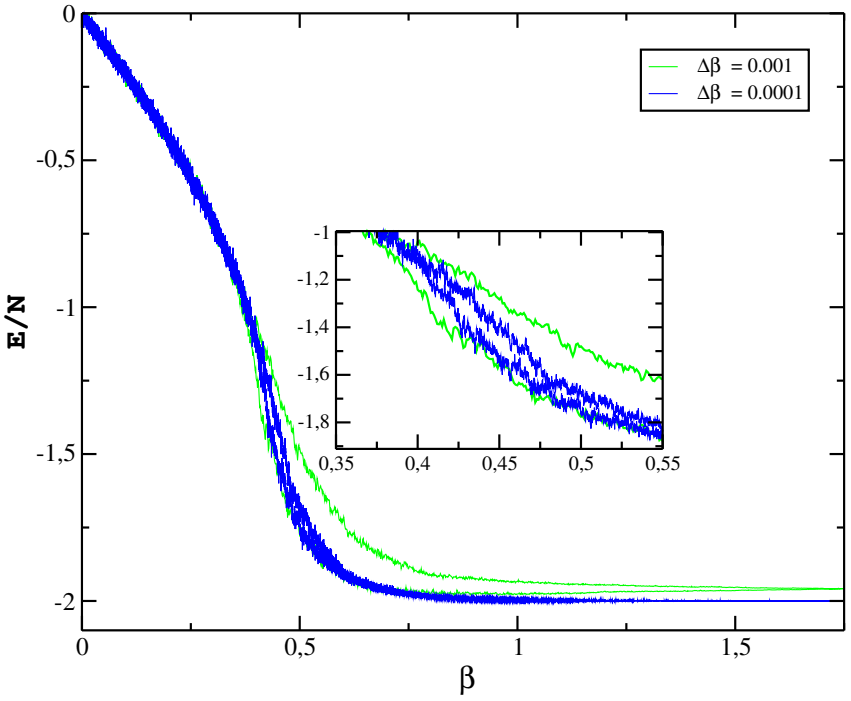


Figura 9: Histereses para L = 100.

Logo, podemos estimar que o centro da histerese fica entre 0.35 e 0.55. A temperatura crítica se encontra por aqui, e vamos usar esse intervalo na próxima tarefa.

(C2)

Como vimos, a correlação entre spins depende da temperatura e a distância de correlação é máxima na temperatura crítica. Se olharmos para o centro da histerese, é lá que encontraremos esse valor. Uma forma de fazer isso é criando essa curva de correlação. É sabido que essa curva cai exponencialmente para cada lado da temperatura crítica, e seu máximo aumenta conforme o tamanho do sistema aumenta, L no nosso caso. E para sistemas reais com L muito grande, a curva diverge. Isso vem de resultados conhecidos da termodinâmica e física estatística. Outro resultado conhecido é a solução do modelo de Ising em duas dimensões, que foi resolvido analiticamente em 1944 por Lars Onsager [2].

Ele provou que a temperatura crítica T_c é dada por:

$$T_c = \frac{2}{\ln 1 + \sqrt{2}} \approx 2.27 \quad (\text{para } J = 1 \text{ e } k_B = 1) \tag{8}$$

o que equivale a $\beta_c = 0.44$. Então o que estamos tentando fazer é chegar na mesma resposta usando física e computação!

Finalmente, como fazer isso? Bom, podemos calcular a capacidade térmica, já que a transição de fase do sistema é de segunda ordem. Temos duas formas de fazer isso: com o método direto ou com um resultado da termodinâmica de equilíbrio, ambos se parecem, respectivamente com:

$$C_T = \lim_{\Delta T \to 0} \frac{\Delta E}{\Delta T} \quad \text{ou} \quad C_T = \frac{\sigma_E^2}{T^2} \tag{9}$$

onde σ_E^2 é a variância da energia. Por praticidade, vamos calcular utilizando a variância.

Escolhemos uma faixa da histerese, algo em torno de $\beta = 0.35$ e 0.55, com $L = 60, 80$ e 100, com passo de $\Delta\beta = 0.01$. E também, com uma configuração inicial com metade dos spins ordenados e a outra metade desordenada, para simular os dois extremos de temperatura. Com isso, rodamos um programa suficientemente demorado para o sistema ter tempo de termalizar e evoluir, e os resultados ficarem mais precisos. Isso faz dessa parte a mais computacionalmente custosa e demorada (com 250000 iterações de Monte Carlo no total para cada L!). Devido à quantidade de números aleatórios usados, o Prof. Hoyos sugeriu o uso do algorítimo rkiss05, o melhor e mais rápido gerador de números aleatórios segundo ele. E também, o método da variância pode apresentar picos indesejados na curva, frutos de uma relíquia computacional mas que podem ser diminuídos com uma termalização mais longa. E para aumentar a resolução, escolheu-se uma faixa mais próxima da temperatura crítica onde podemos aumentar a precisão para $\Delta\beta = 0.001$.

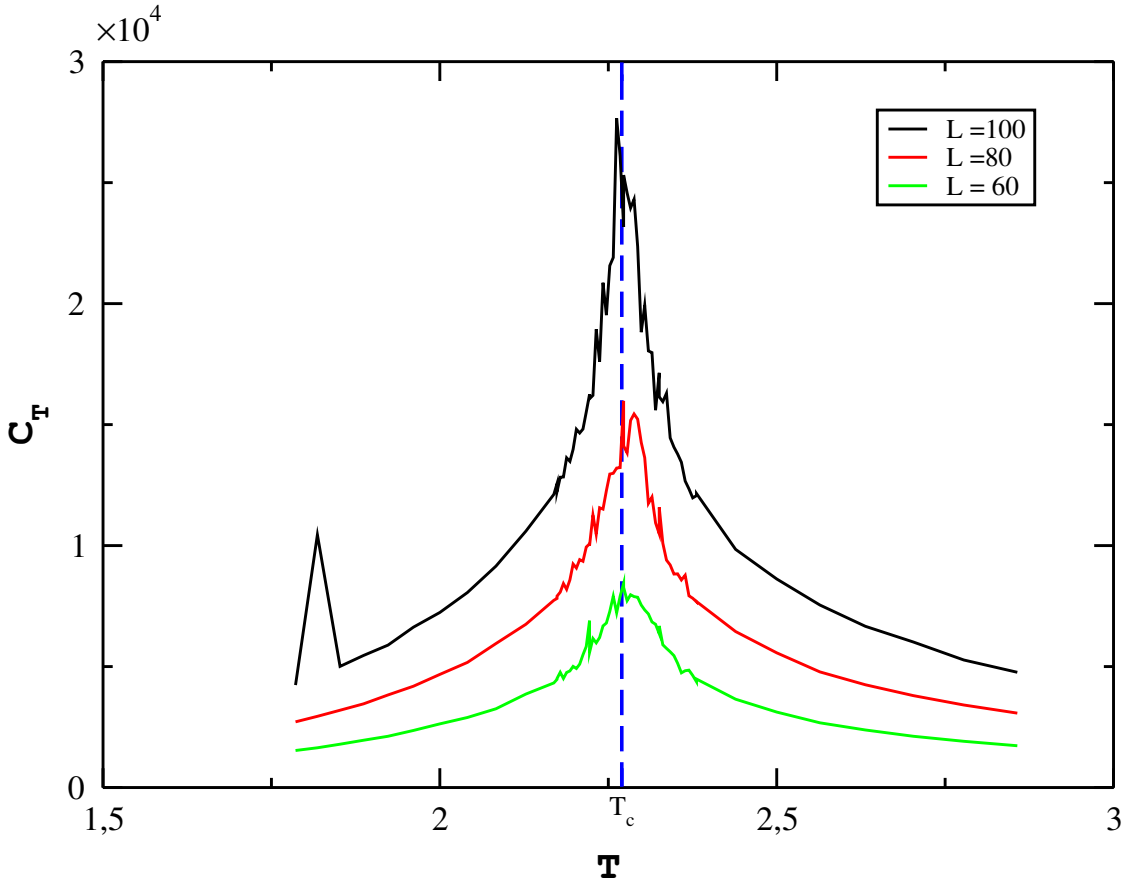
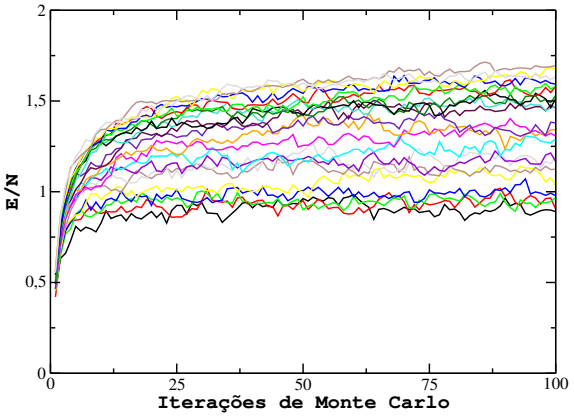


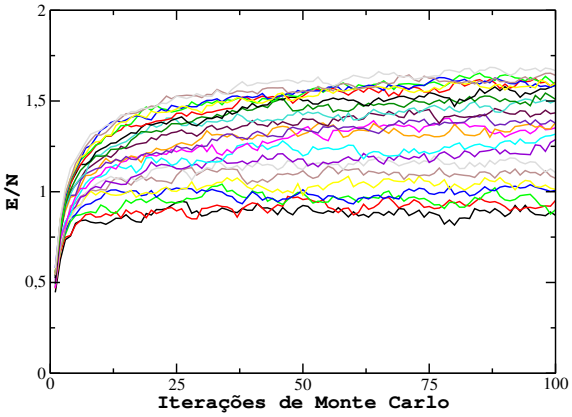
Figura 10: Capacidade térmica em função da temperatura (T) para grades de tamanhos $L = 60$, 80 e 100.

Se retirarmos o valor de T para o máximo de cada curva, recuperamos $T_c = 2.27$, aproximadamente. E como esperado, o pico aumenta com o tamanho do sistema. Na verdade, se fizermos um gráfico da capacidade térmica máxima, ou seja, na temperatura crítica, em função de $\ln L$, iríamos notar uma relação linear com coeficiente positivo. Mas não podemos fitar uma curva com só 3 pontos, então não vai ter gráfico.

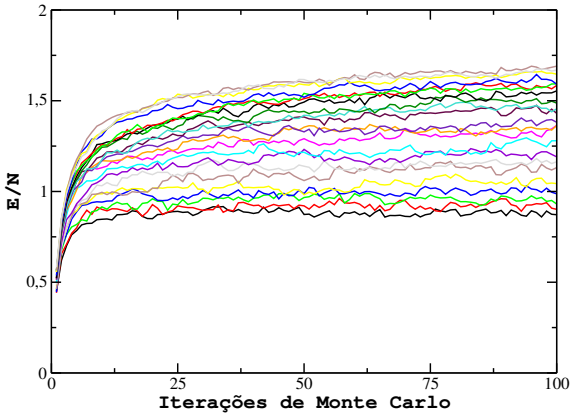
As curvas de energia para cada β estão abaixo. Percebe-se a diferença de variação entre elas, todas partindo da mesma configuração inicial. A variância de cada uma dessas curvas é que foi calculado.



(a) $L = 60$.



(b) $L = 80$.



(c) $L = 100$.

Figura 11: curvas de energia E/N para diferentes temperaturas partindo da mesma configuração inicial.

Tarefa D

Para finalizar o projeto, vamos ver um pouco da quebra espontânea de simetria, responsável pela transição de fase existente em sistemas como esse. O modelo de Ising possui uma simetria global (chamada $Z(2)$), que basicamente significa que se fliparmos todos os spins de uma configuração de energia E, a energia da nova configuração tem mesma energia E, e logo, mesma probabilidade. Então em média, a magnetização seria zero, afinal essas duas configurações equiprováveis tem magnetizações opostas. Mas já vimos como para temperaturas baixas a magnetização média é diferente de zero, então essa simetria precisa ser quebrada, e isso ocorre espontaneamente com a evolução do sistema.

Essa transição de fase faz com que nosso sistema vá de uma configuração com magnetização M para outra com $-M$. E o tempo médio entre essas transições deve crescer com o tamanho do sistema, afinal, gelo para água é uma transição de fase, assim como água para gelo, mas com temperatura fixa ninguém vê essa transição de fase acontecer

espontaneamente, muito menos a segunda. Então para sistemas tão grandes quanto um cubo de gelo ($3.3 \times 10^2 2$ moléculas!) esse tempo tangência infinito. Mas para o modelo de Ising com $L = 4, 5, ..., 10$ esse tempo é computável, e é o que vamos fazer!

Com uma configuração inicial totalmente desordenada e temperatura $\beta = 1/2$, vamos simular grades com os comprimentos mencionados, calculando quantas vezes o sinal de M é trocado, começando após 10000 iterações de termalização, e fazendo a conta durante 500000 iterações de Monte Carlo!

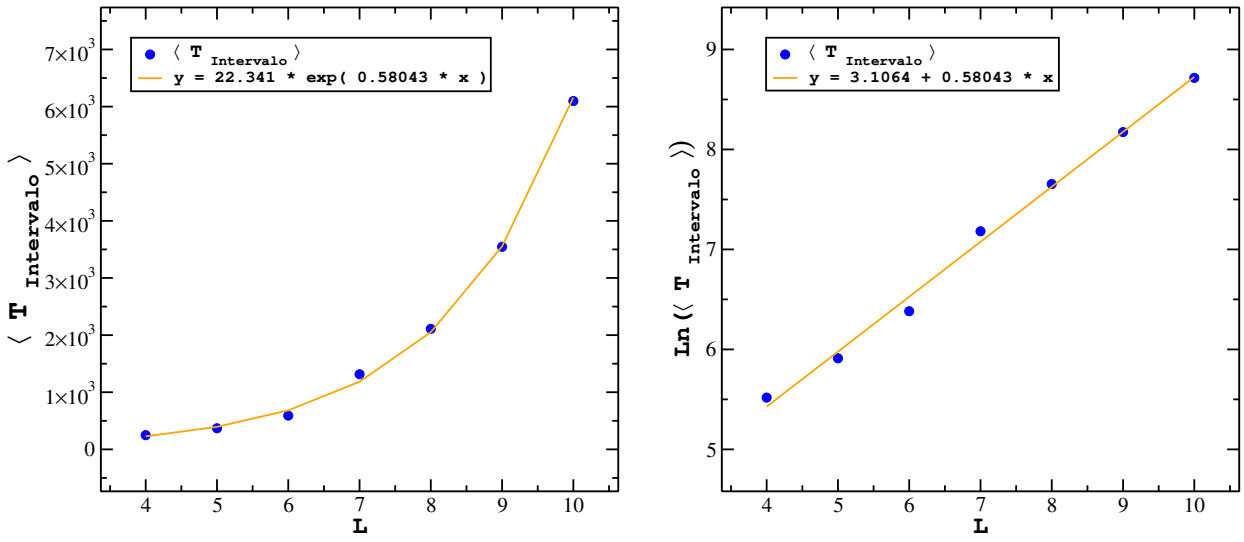


Figura 12: Intervalos de tempo médio entre transições de fase em função do tamanho L do sistema.

Podemos ver claramente o comportamento esperado em ambos os gráficos! Um crescimento exponencial em L. E para completar, um gráfico das transições de fase representadas pela magnetização abaixo, para $L = 10$:

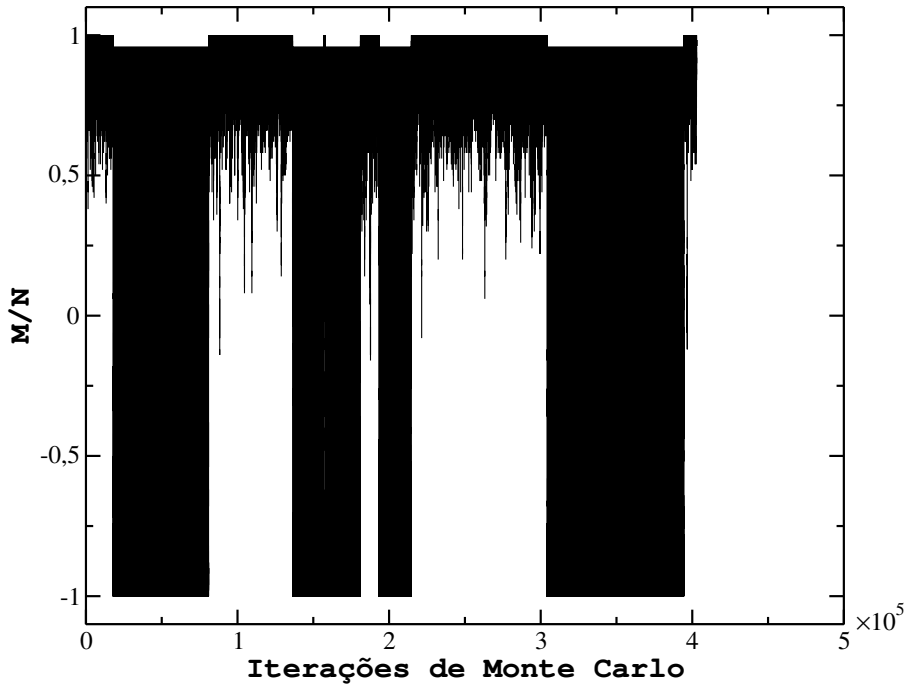


Figura 13: Magnetização em função das iterações de Monte Carlo para um sistema com $L = 10$ e $\beta = \frac{1}{2}$.

Referências

[1] Nicholas J. Giordano e Hisao Nakanishi. *Computational Physics*. 2ª ed. Upper Saddle River (NJ): Prentice Hall, 2006.

[2] Lars Onsager. "Crystal statistics. I. a two-dimensional model with an order-disorder transition". Em: *Phys. Rev.* 65.3-4 (fev. de 1944), pp. 117–149.

Apêndice

GIFs

Dentro das pastas das tarefas A até C coloquei alguns GIFs da dinâmica do sistema, com grades de $L = 500$. Cada um tem até um minuto e meio de duração.

Também existe um GIF da termalização na temperatura crítica, para tentar ilustrar o comportamento singular nesse ponto. Nele podemos ver que se forma um padrão fractal na figura: clusters dentro de cluster dentro de clusters... E assim por diante. $L = 500$ ainda é pequeno para vislumbrar esse fenômeno, mas é o suficiente para começar.

Códigos

Tarefa A

```
!! tarefa-A-12694668.f90
program tarefaA
  character(26) :: M_data(4)
  character(37) :: config_file(4)
  M_data = [character(len=26) :: "./tarefa-A1/A1-L-60-M.dat", "./tarefa-A1/A1-L-100-M.dat", &
                                 &"./tarefa-A2/A2-L-60-M.dat", "./tarefa-A2/A2-L-100-M.dat"]
  config_file = [character(len=37) :: "./tarefa-A1/A1-L-60-final-config.out", "./tarefa-A1/A1-L-100-final-config.out",&
                                      &"./tarefa-A2/A2-L-60-final-config.out", "./tarefa-A2/A2-L-100-final-config.out" ]

  call ising(60, 3d0, M_data(1), config_file(1))
  call ising(100, 3d0, M_data(2), config_file(2))
  call ising(60, 0.1d0, M_data(3), config_file(3))
  call ising(100, 0.1d0, M_data(4), config_file(4))

  contains
  subroutine plot(g, L, config_file)
    integer, intent(in) :: L
    character(*), intent(in) :: config_file
    byte, intent(in) :: g(L,L)
    character(1) :: isimb(-1:1)
    isimb(1) = '1'
    isimb(-1) = '0'
    open(unit=2,file=config_file,status='unknown')
    do i=1, L
      write(2,'(200a2)') (isimb(g(i,j)), j=1, L)
    end do
    close(2)
  end subroutine

  subroutine heatbath(g, Beta, L, M)
    implicit none
    integer, intent(in) :: L
    real(8), intent(in) :: Beta
    real(8), intent(inout) :: M
    byte, intent(inout) :: g(L,L)

    integer :: x, y, i
    real(8) :: Pi, r(2), u
```

```
39      byte :: H
40
41      do i=1, L*L
42        call RANDOM_NUMBER(r)
43        x = 1 + floor(L*r(1))
44        y = 1 + floor(L*r(2))
45
46        H = ( g(mod(x-2+L,L)+1,y) + g(mod(x,L)+1,y) + g(x,mod(y-2+L,L)+1) +
          ↪  g(x,mod(y,L)+1) )
47        Pi = exp( Beta*g(x,y)*H) / ( exp(-Beta*g(x,y)*H) + exp(Beta*g(x,y)*H) )
48
49        call RANDOM_NUMBER(u)
50        if (u .gt. Pi) then
51          g(x,y) = -g(x,y)
52          M = M + 2*g(x,y)
53        end if
54      end do
55    end subroutine
56
57    subroutine ising(L,Beta,M_data,config_file)
58      implicit none
59      integer, intent(in) :: L
60      real(8), intent(in) :: Beta
61      character(*), intent(in) :: M_data, config_file
62
63      integer :: i, nterm = 50000, niter = 10000
64      real(8) :: M
65      byte :: g(L,L)
66
67      open(unit=1,file=M_data,status='unknown')
68
69      g = 1
70      M = 1d0*L*L
71
72      do i=1, nterm + niter
73        call heatbath(g, Beta, L, M)
74        write(1,*) i, M / (L*L)
75      end do
76      call plot(g,L,config_file)
77      close(1)
78    end subroutine
79  end program
```

Tarefa B1

```
!! tarefa-B1-12694668.f90
program tarefaB1
  call annealing(60, 0d0, 3d0, 0.001d0, "B1-Annealing-L-60-Energy.dat",
  ↪  "B1-Annealing-L-60-Final-Config.out")

  contains
  subroutine plot(g, L, config_file)
    integer, intent(in) :: L
    character(*), intent(in) :: config_file
    byte, intent(in) :: g(L,L)
    character(1) :: isimb(-1:1)
    isimb(1) = '1'
    isimb(-1) = '0'
    open(unit=2,file=config_file,status='unknown')
    do i=1, L
      write(2,'(200a2)') (isimb(g(i,j)), j=1, L)
    end do
    close(2)
  end subroutine

  subroutine heatbath(g, Beta, L, E)
    implicit none
    integer, intent(in) :: L
    real(8), intent(in) :: Beta
    real(8), intent(inout) :: E
    byte, intent(inout) :: g(L,L)

    integer :: x, y, i
    real(8) :: Pi, r(2), u
    byte :: H

    do i=1, L*L
      call RANDOM_NUMBER(r)
      x = 1 + floor(L*r(1))
      y = 1 + floor(L*r(2))

      H = ( g(mod(x-2+L,L)+1,y) + g(mod(x,L)+1,y) + g(x,mod(y-2+L,L)+1) +
      ↪  g(x,mod(y,L)+1) )
      Pi = exp( Beta*g(x,y)*H) / ( exp(-Beta*g(x,y)*H) + exp(Beta*g(x,y)*H) )

      call RANDOM_NUMBER(u)
      if (u .gt. Pi) then
        g(x,y) = -g(x,y)
        E = E - 2d0*g(x,y)*H
```

```
        end if
      end do
    end subroutine

    subroutine annealing(L,Beta_i,Beta_f,dB,E_data,config_file)
      implicit none
      integer, intent(in) :: L
      real(8), intent(in) :: Beta_i, Beta_f, dB
      character(*), intent(in) :: E_data, config_file

      integer :: i, j
      real(8) :: E , Beta, init(L,L)
      byte :: g(L,L)

      open(unit=1,file=E_data,status='unknown')

      call RANDOM_NUMBER(init)
      g = 2*nint(init)-1

      E = 0
      do i=1, L
        do j=1, L
          E = E - g(i,j)*( g(mod(i,L)+1,j) + g(i,mod(j,L)+1) )
        end do
      end do

      do i=int(Beta_i), int((Beta_f-Beta_i)/dB)
        Beta = i*dB
        call heatbath(g, Beta, L, E)
        write(1,*) i, E / (L*L)
      end do
      call plot(g,L,config_file)
      close(1)
    end subroutine
end program
```

Tarefa B2

```
!! tarefa-B2-12694668.f90
program tarefaB2
  call quenching(60, 3d0, "B2-Quenching-L-60-Energy.dat",
  ↪  "B2-Quenching-L-60-Final-Config.out")

  contains
```

```
6     subroutine plot(g, L, config_file)
7       integer, intent(in) :: L
8       character(*), intent(in) :: config_file
9       byte, intent(in) :: g(L,L)
10      character(1) :: isimb(-1:1)
11      isimb(1) = '1'
12      isimb(-1) = '0'
13      open(unit=2,file=config_file,status='unknown')
14      do i=1, L
15        write(2,'(200a2)') (isimb(g(i,j)), j=1, L)
16      end do
17      close(2)
18    end subroutine
19
20    subroutine heatbath(g, Beta, L, E)
21      implicit none
22      integer, intent(in) :: L
23      real(8), intent(in) :: Beta
24      real(8), intent(inout) :: E
25      byte, intent(inout) :: g(L,L)
26
27      integer :: x, y, i
28      real(8) :: Pi, r(2), u
29      byte :: H
30
31      do i=1, L*L
32        call RANDOM_NUMBER(r)
33        x = 1 + floor(L*r(1))
34        y = 1 + floor(L*r(2))
35
36        H = ( g(mod(x-2+L,L)+1,y) + g(mod(x,L)+1,y) + g(x,mod(y-2+L,L)+1) +
          ↪  g(x,mod(y,L)+1) )
37        Pi = exp( Beta*g(x,y)*H) / ( exp(-Beta*g(x,y)*H) + exp(Beta*g(x,y)*H) )
38
39        call RANDOM_NUMBER(u)
40        if (u .gt. Pi) then
41          g(x,y) = -g(x,y)
42          E = E - 2d0*g(x,y)*H
43        end if
44      end do
45    end subroutine
46
47    subroutine quenching(L,Beta,E_data,config_file)
48      implicit none
49      integer, intent(in) :: L
50      real(8), intent(in) :: Beta
```

```
51      character(*), intent(in) :: E_data, config_file
52
53      integer :: i, j
54      real(8) :: E, init(L,L)
55      byte :: g(L,L)
56
57      open(unit=1,file=E_data,status='unknown')
58
59      call RANDOM_NUMBER(init)
60      g = 2*nint(init)-1
61
62      E = 0
63      do i=1, L
64        do j=1, L
65          E = E - g(i,j)*( g(mod(i,L)+1,j) + g(i,mod(j,L)+1) )
66        end do
67      end do
68
69      do i=1, 3000
70        call heatbath(g, Beta, L, E)
71        write(1,*) i, E / (L*L)
72      end do
73      call plot(g,L,config_file)
74      close(1)
75    end subroutine
76  end program
```

Tarefa C1

```
1   !! tarefa-C1-12694668.f90
2    program tarefaC1
3     integer :: L_list(3)
4     character(35) :: E_data(6)
5     E_data = [character(len=35) ::
      ↪  "./C1-L-60-E-Beta-1E3.dat","./C1-L-60-E-Beta-1E4.dat",&
6
                                    ↪  &"./C1-L-80-E-Beta-1E3.dat","./C1-L-80-E-Beta-1E4.dat",&
7
                                    ↪  &"./C1-L-100-E-Beta-1E3.dat","./C1-L-100-E-Beta-1E4.dat"]
8
9     L_list = [60, 80, 100]
10    do i=0, 2
11      call termloop(L_list(i+1), 0d0, 1.75d0, 0.001d0, E_data(2*i+1))
12      call termloop(L_list(i+1), 0d0, 1.75d0, 0.0001d0, E_data(2*i+2))
```

```
13      end do
14
15      contains
16      subroutine heatbath(g, Beta, L, E)
17        implicit none
18        integer, intent(in) :: L
19        real(8), intent(in) :: Beta
20        real(8), intent(inout) :: E
21        byte, intent(inout) :: g(L,L)
22
23        integer :: x, y, i
24        real(8) :: Pi, r(2), u
25        byte :: H
26
27        do i=1, L*L
28          call RANDOM_NUMBER(r)
29          x = 1 + floor(L*r(1))
30          y = 1 + floor(L*r(2))
31
32          H = ( g(mod(x-2+L,L)+1,y) + g(mod(x,L)+1,y) + g(x,mod(y-2+L,L)+1) +
            ↪  g(x,mod(y,L)+1) )
33          Pi = exp( Beta*g(x,y)*H) / ( exp(-Beta*g(x,y)*H) + exp(Beta*g(x,y)*H) )
34
35          call RANDOM_NUMBER(u)
36          if (u .gt. Pi) then
37            g(x,y) = -g(x,y)
38            E = E - 2d0*g(x,y)*H
39          end if
40        end do
41      end subroutine
42
43      subroutine termloop(L,Beta_min,Beta_max,dB,E_data)
44        implicit none
45        integer, intent(in) :: L
46        real(8), intent(in) :: Beta_min, Beta_max, dB
47        character(*), intent(in) :: E_data
48
49        integer :: i, j
50        real(8) :: E, Beta, init(L,L)
51        byte :: g(L,L)
52
53        open(unit=1,file=E_data,status='unknown')
54
55        call RANDOM_NUMBER(init)
56        g = 2*nint(init)-1
57
```

```
58        E = 0
59        do i=1, L
60          do j=1, L
61            E = E - g(i,j)*( g(mod(i,L)+1,j) + g(i,mod(j,L)+1) )
62          end do
63        end do
64
65        !! Ida
66        do i=int(Beta_min), int((Beta_max-Beta_min)/dB)
67          Beta = i*dB
68          call heatbath(g, Beta, L, E)
69          write(1,*) Beta, E / (L*L)
70        end do
71
72        !! Volta
73        do i=int((Beta_max-Beta_min)/dB)-1, int(Beta_min), -1
74          Beta = i*dB
75          call heatbath(g, Beta, L, E)
76          write(1,*) Beta, E / (L*L)
77        end do
78        close(1)
79      end subroutine
80    end program
```

Tarefa C2

```
1   !! tarefa-C2-12694668.f90
2   program tarefaC2
3     call ising(60, 123, 0.35d0, 0.55d0, 0.01d0, "./C2-L-60-E.dat", "./C2-L-60-Ct.dat")
4     call ising(80, 456, 0.35d0, 0.55d0, 0.01d0, "./C2-L-80-E.dat", "./C2-L-80-Ct.dat")
5     call ising(100, 789, 0.35d0, 0.55d0, 0.01d0, "./C2-L-100-E.dat",
      ↪  "./C2-L-100-Ct.dat")
6
7
8     !! O passo de dB = 0.01 é muito grande para o gráfico de Variância versus Beta
      ↪  exibir uma curva
9     !! mais precisa, então usando o valor esperado de T_c  2.27 (Beta_c  0.44),
      ↪  diminuímos o passo para 0.001 e rodamos
10    !! novamente no intervalo (Beta_c-0.02,Beta_c+0.02) para somarmos aos dados
      ↪  anteriores.
11    call ising(60, 123, 0.42d0, 0.46d0, 0.001d0, "./dummy.dat",
      ↪  "./C2-L-60-Ct-Extra.dat")
12    call ising(80, 456, 0.42d0, 0.46d0, 0.001d0, "./dummy.dat",
      ↪  "./C2-L-80-Ct-Extra.dat")
```

```
13    call ising(100, 789, 0.42d0, 0.46d0, 0.001d0, "./dummy.dat",
      ↪  "./C2-L-100-Ct-Extra.dat")
14
15    contains
16    subroutine ising(L,seed,Beta_i,Beta_f,dB,E_data,Ct_data)
17      implicit none
18      integer, intent(in) :: L, seed
19      real(8), intent(in) :: Beta_i, Beta_f, dB
20      character(*), intent(in) :: E_data, Ct_data
21
22      integer, parameter :: nterm = 50000, niter = 200000
23      integer :: i, j, k, m
24      real(8) :: E, Beta, init(L,L), x, xx,
        ↪  Energies(nterm+niter,0:ceiling((Beta_f-Beta_i)/dB))
25      real(8) :: fb(-4:4)
26      byte :: g(L,L)
27
28      open(unit=1,file=E_data,status='unknown')
29      open(unit=2,file=Ct_data,status='unknown')
30
31      call kissinit(seed)
32
33      do i=1, L/2
34        do j=1, L
35          init(i,j) = 2*nint(rkiss05())-1
36        end do
37      end do
38      init(L/2+1:L,1:L) = 1
39
40      g = init
41
42      do i=0, ceiling((Beta_f-Beta_i)/dB)
43        g = init
44        Beta = Beta_i + i*dB
45
46        do m=-4,4
47          fb(m) = exp(-Beta*m)
48        end do
49
50        x = 0
51        xx = 0
52        E = 0
53        do j=1, L
54          do k=1, L
55            E = E - g(j,k)*( g(mod(j,L)+1,k) + g(j,mod(k,L)+1) )
56          end do
```

```
57        end do
58
59        !! Termalização
60        do m=1, nterm
61          call heatbath(g, fb, L, E)
62          Energies(m,i) = E / (L*L)
63        end do
64
65        do m=1, niter
66          call heatbath(g, fb, L, E)
67          Energies(nterm+m,i) = E / (L*L)
68          x = x + E
69          xx = xx + E*E
70        end do
71        x = x/niter
72        xx = xx/niter
73        write(2,*) 1/Beta, (xx - x*x) * (Beta**2) ! Beta, Calor Específico / N
74      end do
75
76      do i=1, nterm
77        write(1,*) i, Energies(i,:)
78      end do
79      close(1)
80      close(2)
81    end subroutine
82
83    subroutine heatbath(g, fb, L, E)
84      implicit none
85      integer, intent(in) :: L
86      real(8), intent(in) :: fb(-4:4)
87      real(8), intent(inout) :: E
88      byte, intent(inout) :: g(L,L)
89
90      integer :: x, y, i
91      real(8) :: Pi, u
92      byte :: H
93
94      do i=1, L*L
95        x = 1 + floor(L*rkiss05())
96        y = 1 + floor(L*rkiss05())
97
98        H = ( g(mod(x-2+L,L)+1,y) + g(mod(x,L)+1,y) + g(x,mod(y-2+L,L)+1) +
          ↪  g(x,mod(y,L)+1) )
99        Pi = fb(g(x,y)*H) / ( fb(-g(x,y)*H) + fb(g(x,y)*H) )
100
101       u = rkiss05()
```

```
102         if (u .gt. Pi) then
103           g(x,y) = -g(x,y)
104           E = E - 2d0*g(x,y)*H
105         end if
106       end do
107     end subroutine
108
109     FUNCTION rkiss05()
110     implicit none
111
112     integer,parameter      :: r8b= SELECTED_REAL_KIND(P=14,R=99)   ! 8-byte reals
113     integer,parameter      :: i4b= SELECTED_INT_KIND(8)            ! 4-byte integers
114     real(r8b),parameter    :: am=4.656612873077392578d-10       ! multiplier 1/2^31
115
116     real(r8b)             :: rkiss05
117     integer(i4b)          :: kiss
118     integer(i4b)          :: x,y,z,w              ! working variables for the four
        ↪  generators
119     common /kisscom/x,y,z,w
120
121     x = 69069 * x + 1327217885
122     y= ieor (y, ishft (y, 13)); y= ieor (y, ishft (y, -17)); y= ieor (y, ishft (y, 5))
123     z = 18000 * iand (z, 65535) + ishft (z, - 16)
124     w = 30903 * iand (w, 65535) + ishft (w, - 16)
125     kiss = ishft(x + y + ishft (z, 16) + w , -1)
126     rkiss05=kiss*am
127     END FUNCTION rkiss05
128
129
130     SUBROUTINE kissinit(iinit)
131     implicit none
132     integer,parameter      :: r8b= SELECTED_REAL_KIND(P=14,R=99)   ! 8-byte reals
133     integer,parameter     :: i4b= SELECTED_INT_KIND(8)            ! 4-byte integers
134
135     integer(i4b) idum,ia,im,iq,ir,iinit
136     integer(i4b) k,x,y,z,w,c1,c2,c3,c4
137     real(r8b)    rdum
138     parameter (ia=16807,im=2147483647,iq=127773,ir=2836)
139     common /kisscom/x,y,z,w
140
141     !!! Test integer representation !!!
142     c1=-8
143     c1=ishftc(c1,-3)
144     !     print *,c1
145     if (c1.ne.536870911) then
146        print *,'Nonstandard integer representation. Stoped.'
```

```
147         stop
148       endif
149
150       idum=iinit
151       idum= abs(1099087573 * idum)                ! 32-bit LCG to shuffle seeds
152       if (idum.eq.0) idum=1
153       if (idum.ge.IM) idum=IM-1
154
155       k=(idum)/IQ
156       idum=IA*(idum-k*IQ)-IR*k
157       if (idum.lt.0) idum = idum + IM
158       if (idum.lt.1) then
159          x=idum+1
160       else
161          x=idum
162       endif
163       k=(idum)/IQ
164       idum=IA*(idum-k*IQ)-IR*k
165       if (idum.lt.0) idum = idum + IM
166       if (idum.lt.1) then
167          y=idum+1
168       else
169          y=idum
170       endif
171       k=(idum)/IQ
172       idum=IA*(idum-k*IQ)-IR*k
173       if (idum.lt.0) idum = idum + IM
174       if (idum.lt.1) then
175          z=idum+1
176       else
177          z=idum
178       endif
179       k=(idum)/IQ
180       idum=IA*(idum-k*IQ)-IR*k
181       if (idum.lt.0) idum = idum + IM
182       if (idum.lt.1) then
183          w=idum+1
184       else
185          w=idum
186       endif
187
188       rdum=rkiss05()
189
190       return
191     end subroutine kissinit
192     end program
```

Tarefa D

```
!! tarefa-D-12694668.f90
program tarefaD
  open(unit=1,file="D-Tempo.dat",status='unknown')
  open(unit=2,file="D-Magnetization.dat",status='unknown')
  Tempo = 0
  do i=4,10
    call ising(i,1d0/2d0,Tempo)
    write(1,*) i, Tempo, log(Tempo)
  end do
  close(1)
  close(2)

  contains
  subroutine heatbath(g, Beta, L, M)
    implicit none
    integer, intent(in) :: L
    real(8), intent(in) :: Beta
    real(8), intent(inout) :: M
    byte, intent(inout) :: g(L,L)

    integer :: x, y, i
    real(8) :: Pi, r(2), u
    byte :: H

    do i=1, L*L
      call RANDOM_NUMBER(r)
      x = 1 + floor(L*r(1))
      y = 1 + floor(L*r(2))

      H = ( g(mod(x-2+L,L)+1,y) + g(mod(x,L)+1,y) + g(x,mod(y-2+L,L)+1) +
      ↪  g(x,mod(y,L)+1) )
      Pi = exp( Beta*g(x,y)*H) / ( exp(-Beta*g(x,y)*H) + exp(Beta*g(x,y)*H) )

      call RANDOM_NUMBER(u)
      if (u .gt. Pi) then
        g(x,y) = -g(x,y)
        M = M + 2*g(x,y)
      end if
    end do
  end subroutine
```

```
40
41    subroutine ising(L,Beta,Tempo)
42      implicit none
43      integer, intent(in) :: L
44      real(8), intent(in) :: Beta
45      real, intent(out) :: Tempo
46
47      integer, parameter :: nterm = 10000, niter = 500000
48      integer :: i, j, Ntroca
49      real(8) :: Ma, Mb, init(L,L)
50      byte :: g(L,L)
51
52      Ntroca = 1
53      Ma = 0
54      Mb = 0
55
56      call RANDOM_NUMBER(init)
57      g = 2*nint(init)-1
58
59      do i=1, L
60        do j=1, L
61          Ma = Ma + g(i,j)
62        end do
63      end do
64
65      do i=1, nterm
66        call heatbath(g,Beta,L,Ma)
67      end do
68      Mb = Ma
69
70      do i=1, niter
71        call heatbath(g,Beta,L,Mb)
72
73        if (L .eq. 10) write(2,*) i, Mb/(L*L)
74
75        if (Ma*Mb .lt. 0d0) then
76          Ntroca = Ntroca + 1
77          Ma = Mb
78        end if
79      end do
80
81      Tempo = niter/Ntroca
82    end subroutine
83  end program
```