Aluno: João Victor Alcantara Pimenta 11820812

Professor: Francisco Castilho Alcaraz

1 Prolegômenos

Trataremos do modelo de Ising para tratar de uma grade de muitos corpos com dois estados possíveis. Especificamente tratemos de uma grade de spins nos problemas a seguir. Como mesmo para pequenas grades tratar de todo estados possíveis do corpo para definir seu equilíbrio vamos abordar de forma estocástica a evolução do sistema. Possibilitaremos trocas de estado aleatórias com base na probabilidade da energia resultante ser a energia do sistema tratado. Especificamente, calcularemos:

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

Onde $s(i,j)$ representa o valor do spin no quadrante com coordenadas (i,j). J é uma normalização de energia.

Sabemos para sistemas físicos nesse ensemble que a energia será normalmente distribuída e terá uma probabilidade associada a cada configuração possível dada por

$$P(E) \propto e^{-\beta E}$$

Outra grandeza é interessante de ser estudada neste sistema é a magnetização por spin, definida por

$$m = \frac{M}{N} = \frac{1}{N}\sum_{i=1}^{L}\sum_{j=1}^{L} s(i,j)$$

Em princípio, analisar a probabilidade de cada estado bastaria para definir o estado de equilíbrio e a distribuição do sistema. Contudo, com tantos estados, outra abordagem deve ser tomada.

2 O algoritmo

Ao invés de percorrer todos os estados, vamos, de um estado inicial, percorrer estados de forma aleatória de acordo com a probabilidade associada a essa mudança de estado. Vamos explorar o método.

Iniciaremos em um estado de spins na nossa malha (de tamanho L) que pode ser aleatória ou ordenada, dependendo do efeito que queremos observar. À cada passo temporal do nosso ciclo faremos L^2 escolhas aleatórias de células e trocaremos o spin correspondente com probabilidade associada à nova energia do sistema ao se realizar a mudança.

Ou seja, sendo E_0 o estado anterior e E_1 o possível novo estado, teremos as probabilidados associadas,

$$P(E_0) \propto e^{-\beta E_0}$$

$$P(E_1) \propto e^{-\beta E_1}$$

E assim por diante fazendo mudanças até que o sistema entre em equilíbrio. Neste ponto, tomamos as médias para definir as grandezas de equilíbrio.

Para que tudo seja possível, definiremos algumas funções de importância. Nominalmente definiremos uma função para inicializar nossa grade da forma apropriada. No exemplo, temos 'SetupOrdered', 'SetupRandom' e 'SetupMixed' que inicializam de forma ordenada (igual), aleatória os spins e mista.

Uma função de 'Flip' que muda o estado de um spin de acordo com as condições que descrevemos. Esa função depende de outra duas, uma 'Exponential' que pré-define a distribuição para o β considerado e uma função 'Pos' que garante que nossas condições de contorno sejam cíclicas.

'Rfmag' e 'REnergy' e seus respectivos delta são responsáveis por manter os valores de energia e magnetização atualizados ao longo da evolução do sistema.

```
      SUBROUTINE EXPONENTIAL(BETA)
            IMPLICIT REAL*8 (A-H,O-Z)
            REAL*8 EXP_RESULTS(-4:4)
            COMMON  /EXP_RESULTS/ EXP_RESULTS

            DO I = -4, 4
            EXP_RESULTS(I)=DEXP(-BETA*I)/(DEXP(-BETA*I)+DEXP(BETA*I))
            END DO

      END SUBROUTINE EXPONENTIAL

      SUBROUTINE POS()
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100)
            INTEGER IPOS(0:L+1)
            COMMON /POSITION/ IPOS

            DO 10 I=1,L
                  IPOS(I) = I
 10         END DO

            IPOS(0) = L
            IPOS(L+1) = 1

      END SUBROUTINE POS

      SUBROUTINE SETUPRANDOM()
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            COMMON /LATTICE/ LATTICE

            DO 10 I=1,L
                  DO 20 K=1,L
                        if (rand() < 0.5) then
                              LATTICE(I,K) = 1
                        else
                              LATTICE(I,K) = -1
```

```
                        end if
20              END DO
10        END DO
      END SUBROUTINE SETUPRANDOM

      SUBROUTINE SETUPORDERED()
            PARAMETER (L=100)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            COMMON /LATTICE/ LATTICE

            DO 10 I=1,L
                  DO 20 K=1,L
                        LATTICE(I,K) = 1
20                END DO
10          END DO
      END SUBROUTINE SETUPORDERED

      SUBROUTINE SETUPMIXED()
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            COMMON /LATTICE/ LATTICE

            DO 10 I=1,L
                  DO 20 K=1,L
                        if (MOD(I,2) == 0) then
                              if (RAND() < 0.5) then
                                    LATTICE(I,K) = -1
                              else
                                    LATTICE(I,K) = 1
                              end if
                        else
                              LATTICE(I,K) = 1
                        end if
20                END DO
10          END DO
      END SUBROUTINE SETUPMIXED

      SUBROUTINE FLIP(ITOTAL)
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100, J=1.0)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            DIMENSION EXP_RESULTS(-4:4), IPOS(0:L+1)
            COMMON /LATTICE/ LATTICE
            COMMON /EXP_RESULTS/ EXP_RESULTS
            COMMON /POSITION/ IPOS

            I = 1 + FLOOR(RAND()*L)
            K = 1 + FLOOR(RAND()*L)

            I_S_DELTA_M =  J*(LATTICE(IPOS(I-1),K) +
     &                        LATTICE(IPOS(I+1),K) +
     &                        LATTICE(I,IPOS(K-1)) +
     &                        LATTICE(I,IPOS(K+1)))
     &                        *LATTICE(I,K)

            IF (RAND() < EXP_RESULTS(I_S_DELTA_M)) THEN
                  LATTICE(I,K) = -LATTICE(I,K)
                  CALL DeltaMag(LATTICE(I,K), ITOTAL)
                  CALL DeltaEnergy(I,K)
            END IF

      END SUBROUTINE FLIP
```

```
      FUNCTION RFMAG(ITOTAL)
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            COMMON /LATTICE/ LATTICE

            RFMAG = 0
            DO 10 I=1,L
                  DO 20 K=1,L
                        RFMAG = RFMAG + LATTICE(I,K)
   20             END DO
   10       END DO
            RFMAG = RFMAG/ITOTAL
      END FUNCTION RFMAG

      SUBROUTINE DeltaMag(pm, ITOTAL)
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100)
            BYTE PM
            COMMON /PARAM/ ENERGY, RMAG

            RMAG = RMAG + (PM/ITOTAL)*2

      END SUBROUTINE DeltaMag

      FUNCTION REnergy()
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100, J=1.0)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            DIMENSION IPOS(0:L+1)
            COMMON /LATTICE/ LATTICE
            COMMON /POSITION/ IPOS

            REnergy = 0
            DO 10 I=1,L
                  DO 20 K=1,L
                        REnergy = REnergy
     &                        + LATTICE(I,K)*(LATTICE(IPOS(I-1),K) +
     &                        LATTICE(IPOS(I+1),K) +
     &                        LATTICE(I,IPOS(K-1)) +
     &                        LATTICE(I,IPOS(K+1)))
   20             END DO
   10       END DO
            REnergy = -J*REnergy/2
      END FUNCTION REnergy

      SUBROUTINE DeltaEnergy(i, k)
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100, J=1.0)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            DIMENSION IPOS(0:L+1)
            COMMON /LATTICE/ LATTICE
            COMMON /PARAM/ ENERGY, RMAG
            COMMON /POSITION/ IPOS

            Delta = 2*J*LATTICE(i,k)*(LATTICE(IPOS(i-1),k) +
     &                        LATTICE(IPOS(i+1),k) +
     &                        LATTICE(i,IPOS(k-1)) +
     &                        LATTICE(i,IPOS(k+1)))

            ENERGY = ENERGY - Delta

```

```
165       END SUBROUTINE DeltaEnergy
```

3 Tarefa A

Tomaremos $L = 60,100$ para a simulação nesta tarefa. Simulemos. Além das funções já definidas, teremos nosso programa principal

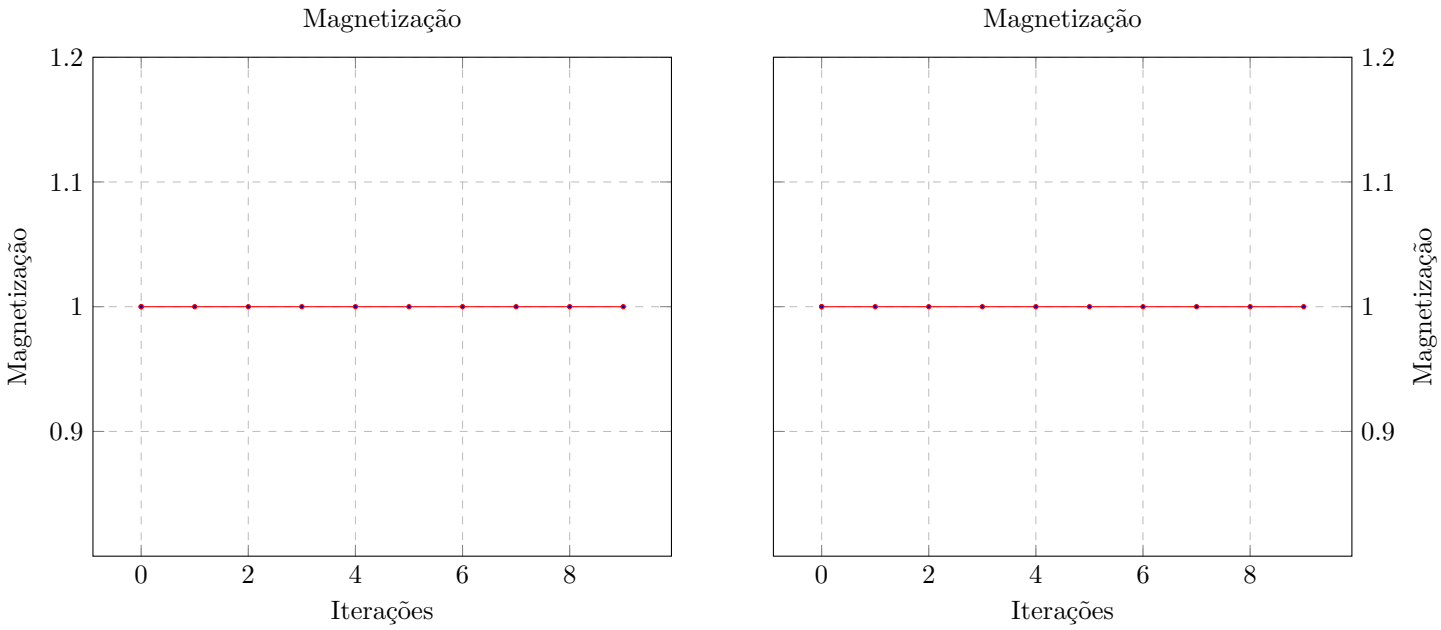
```
      PROGRAM ISING
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100, N=10000, MED = 10000)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            DIMENSION EXP_RESULTS(-4:4) ,IPOS(0:L+1)
            CHARACTER*1 SYMBOLS(-1:1)
            COMMON /POSITION/ IPOS
            COMMON /LATTICE/ LATTICE
            COMMON /EXP_RESULTS/ EXP_RESULTS
            COMMON /PARAM/ RMAG

            SYMBOLS(-1) = '-'
            SYMBOLS(1) = '+'

            ITOTAL = L*L

            OPEN(1,FILE='ISING.DAT',STATUS='UNKNOWN')
            OPEN(2,FILE='ISING_data.DAT',STATUS='UNKNOWN')

            CALL EXPONENTIAL()
            CALL SETUPORDERED()
            CALL POS()

            rmag = RFMAG(ITOTAL)
            WRITE(2,*) rmag

            DO 20 WHILE (flag < 10)

                  rmag_old = rmag

                  DO 10 I=1,N
                        CALL FLIP(ITOTAL)
   10             END DO

                  WRITE(2,*) rmag

                  IF(abs(rmag-rmag_old)<0.01)then
                        flag = flag + 1
                  ELSE
                        flag = 0
                  END IF

   20       END DO

            Acum_mag = 0
            DO 40 I=1,MED
                  CALL FLIP(ITOTAL)
                  Acum_mag = Acum_mag + rmag
   40       END DO
            Acum_mag = Acum_mag/MED
            WRITE(*,*) Acum_mag

            DO 50 K=1,L
                  WRITE(1,'(100A2)') (SYMBOLS(LATTICE(I,K)),I=1,L)
   50       END DO
```

```
56
57              CLOSE(1)
58
59          END PROGRAM ISING
```

3.1 A1

Para a subseção faremos $\beta = 3$ em uma unidade já facilitada onde $1/J$. Checaremos o estado final e a progressão da magnetização.

Podemos ver pelas figuras e plotagens que os dos sistemas se mantém ordenados para $\beta = 3$.

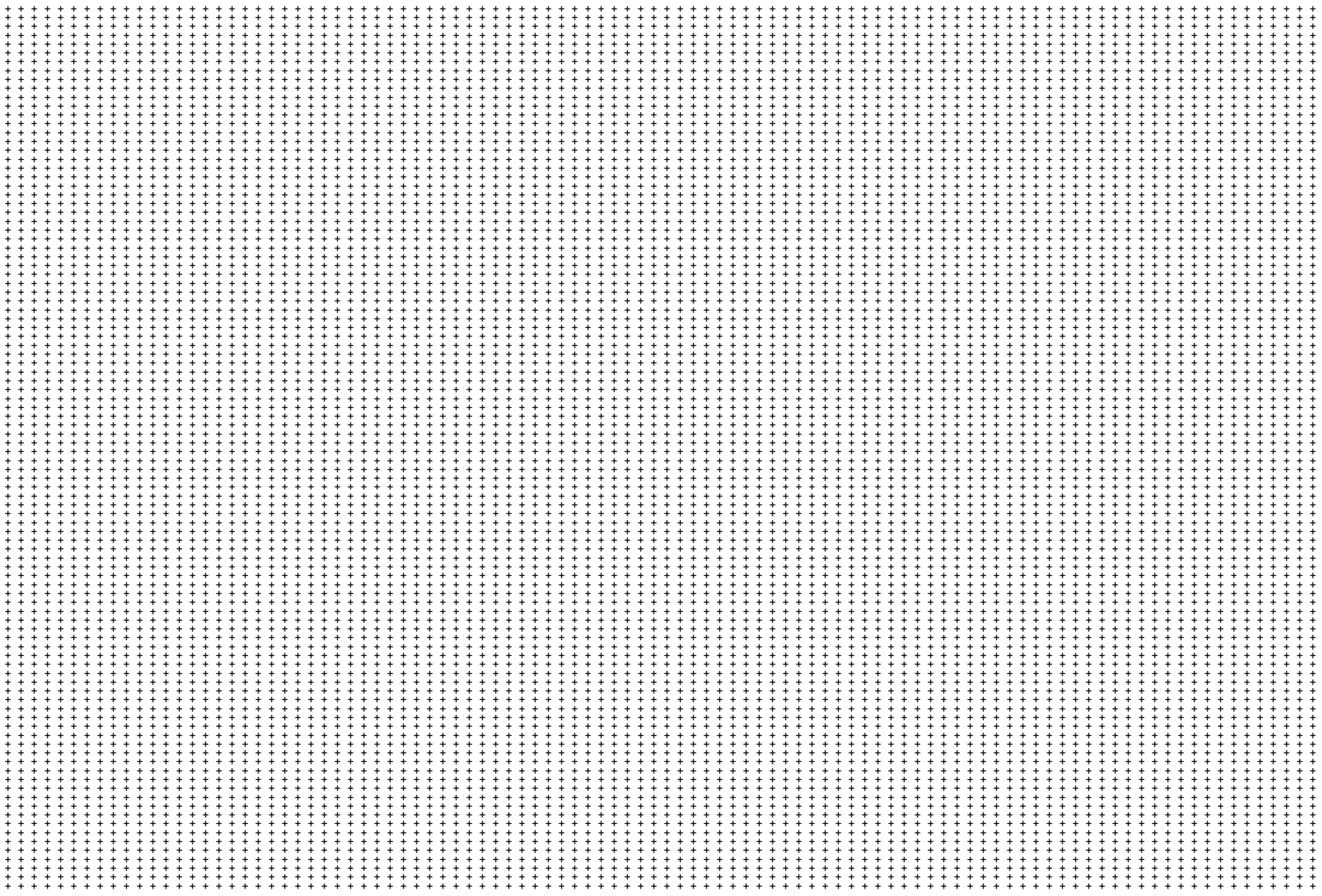


(a) L = 100

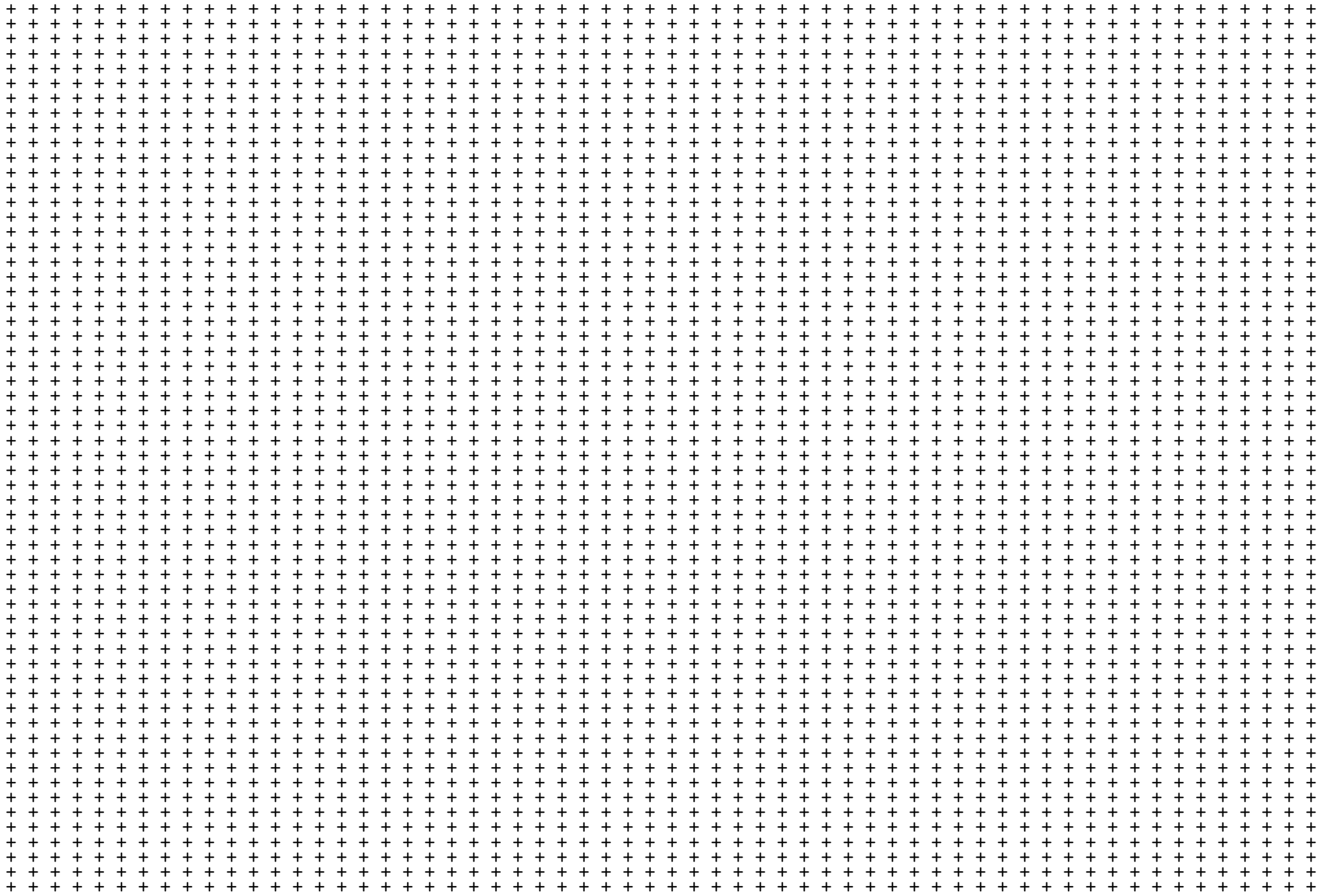
(b) L = 60

Figure 1: Magentizações $\beta = 3$

L=60 e $\beta = 3$



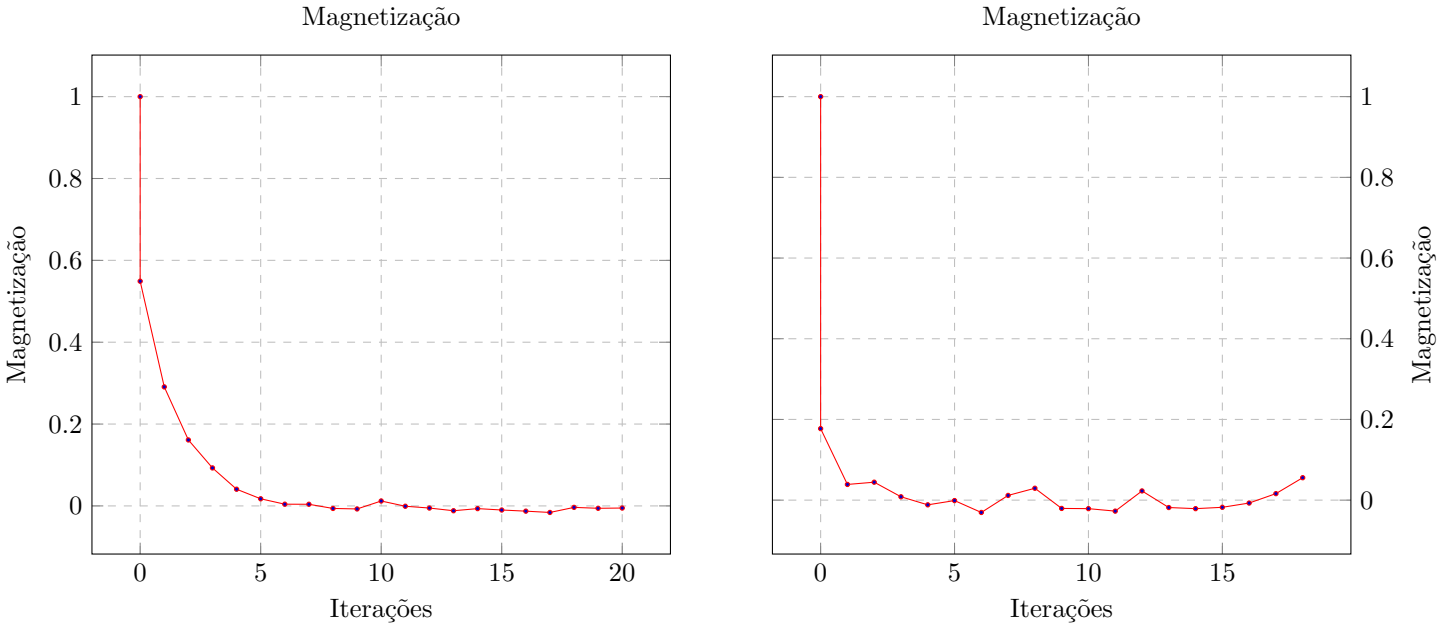
L=60 e $\beta = 3$



3.2 A2

Para a subseção faremos $\beta = 0.1$ em uma unidade já facilitada onde $1/J$. Checaremos o estado final e a progressão da magnetização.

Podemos ver pelas figuras e plotagens que os dos sistemas não se mantém ordenados para $\beta = 0.1$.

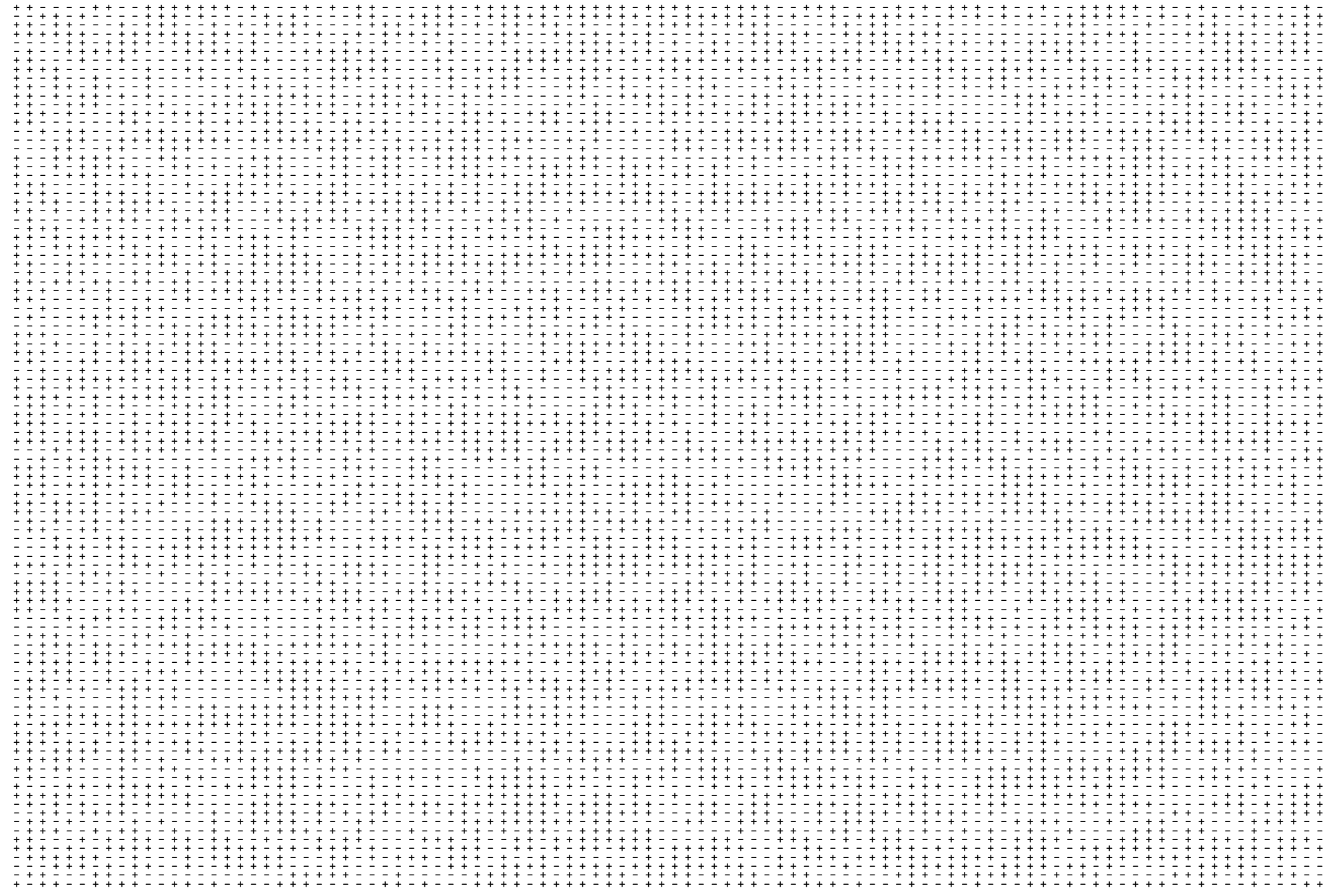


(a) L = 100

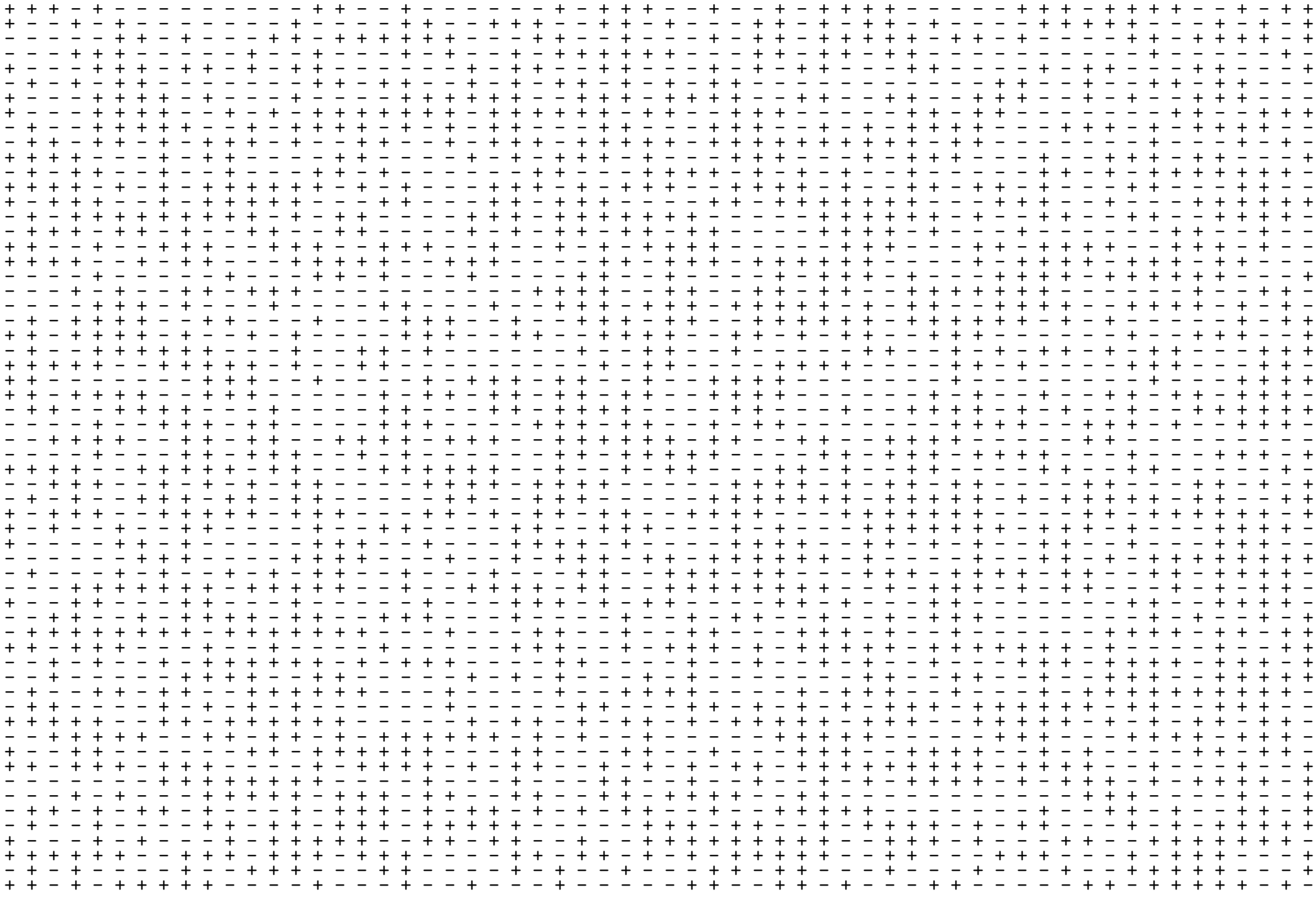
(b) L = 60

Figure 2: Magentizações $\beta = 0.1$

L=100 e $\beta = 0.1$



L=60 e $\beta = 0.1$



4 Tarefa B

Recozimentos e Têmperas são dois processos térmicos de importância para alguns sistemas. No Recozimento, aqueceremos nossa amostra de forma que ela esteja sempre em equilíbrio, vagarosamente deixando que ela se estabilize. Já na Têmpera, o aquecimento é rápido e fora da temperatura de equilíbrio. Tomaremos $L = 60$.

4.1 B1

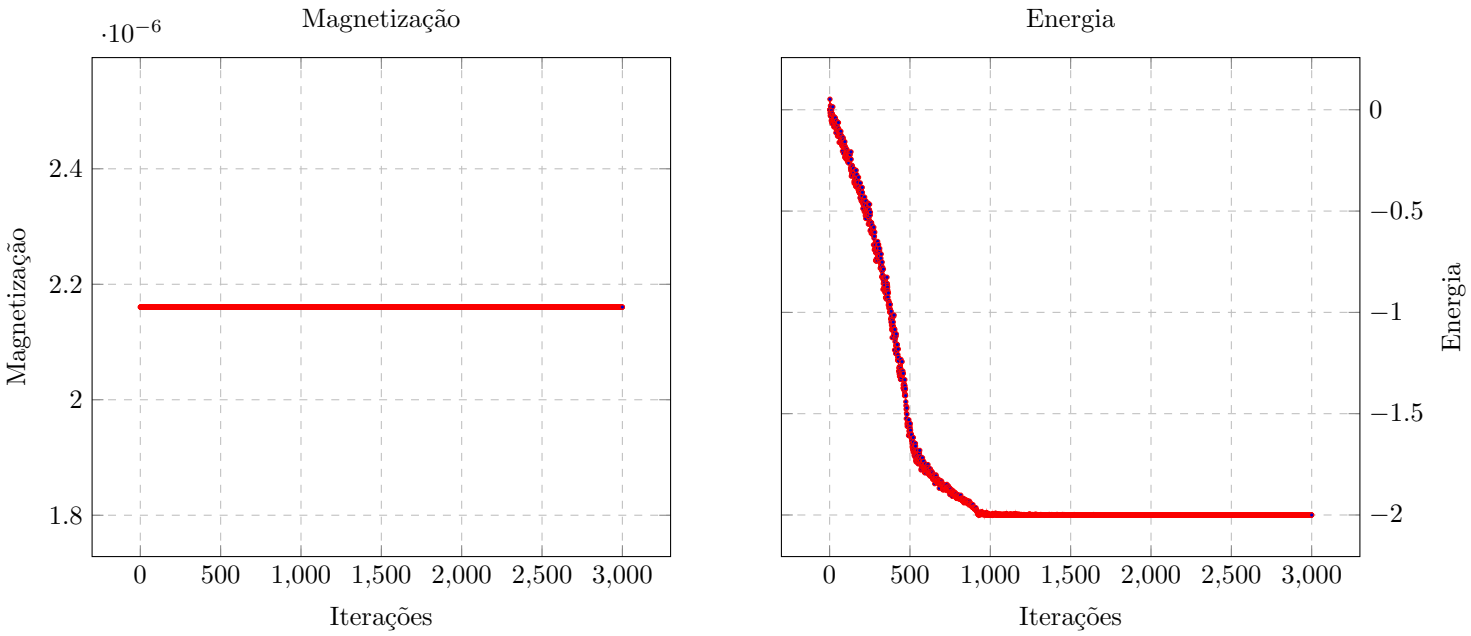
Iniciando em $\beta = 0$ com uma configuração compatível (aleatória), somaremos em β um pequeno passo por vez garantindo que o sistema se mantenha em equilíbrio.

Usaremos,

```
      PROGRAM ISING
           IMPLICIT REAL*8 (A-H,O-Z)
           PARAMETER (L=60, N=10000, MED = 10000)
           BYTE, DIMENSION (1:L,1:L) :: LATTICE
           DIMENSION EXP_RESULTS(-4:4), IPOS(0:L+1)
           CHARACTER*1 SYMBOLS(-1:1)
           COMMON /POSITION/ IPOS
           COMMON /LATTICE/ LATTICE
           COMMON /EXP_RESULTS/ EXP_RESULTS
           COMMON /PARAM/ ENERGY, RMAG

           SYMBOLS(-1) = '-'
           SYMBOLS(1) = '+'

           ITOTAL = L*L
           STEP = 0.001

           OPEN(1,FILE='./B/b1mag.DAT',STATUS='UNKNOWN')
           OPEN(2,FILE='./B/b1ene.DAT',STATUS='UNKNOWN')
           OPEN(3,FILE='./B/b1lattice.DAT',STATUS='UNKNOWN')

           CALL POS()
           CALL SETUPRANDOM()

           rmag = RFMAG(ITOTAL)
           energy = REnergy()

           WRITE(1,*) 0,rmag/Itotal
           WRITE(2,*) 0,energy/Itotal

           DO 10 ICICLE = 1, 3000
                WRITE(*,*) ICICLE
                BETA = STEP*ICICLE
                CALL EXPONENTIAL(BETA)

                DO 30 I=1,N
                     CALL FLIP(ITOTAL)
 30             END DO

                WRITE(1,*) ICICLE, RMAG/ITOTAL
                WRITE(2,*) ICICLE, ENERGY/ITOTAL

 10         END DO

           DO 50 K=1,L
                WRITE(3,'(60A2)') (SYMBOLS(LATTICE(I,K)),I=1,L)
50         END DO

           CLOSE(1)
```

```
50            CLOSE(2)
51
52        END PROGRAM ISING
```

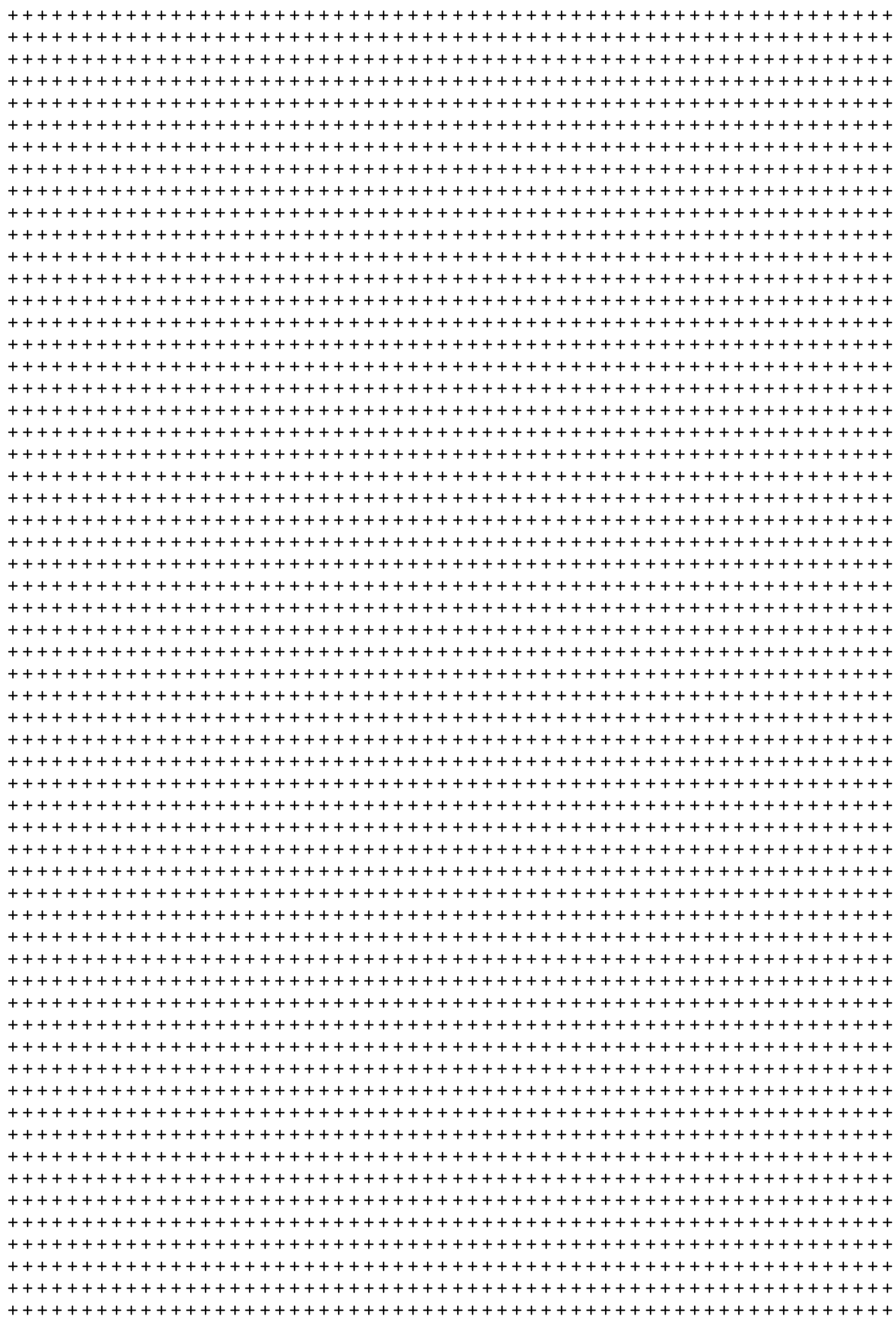
Que nos dá resultados (note escala)



(a) Magnetização

(b) Energia

Figure 3: Magentizações β



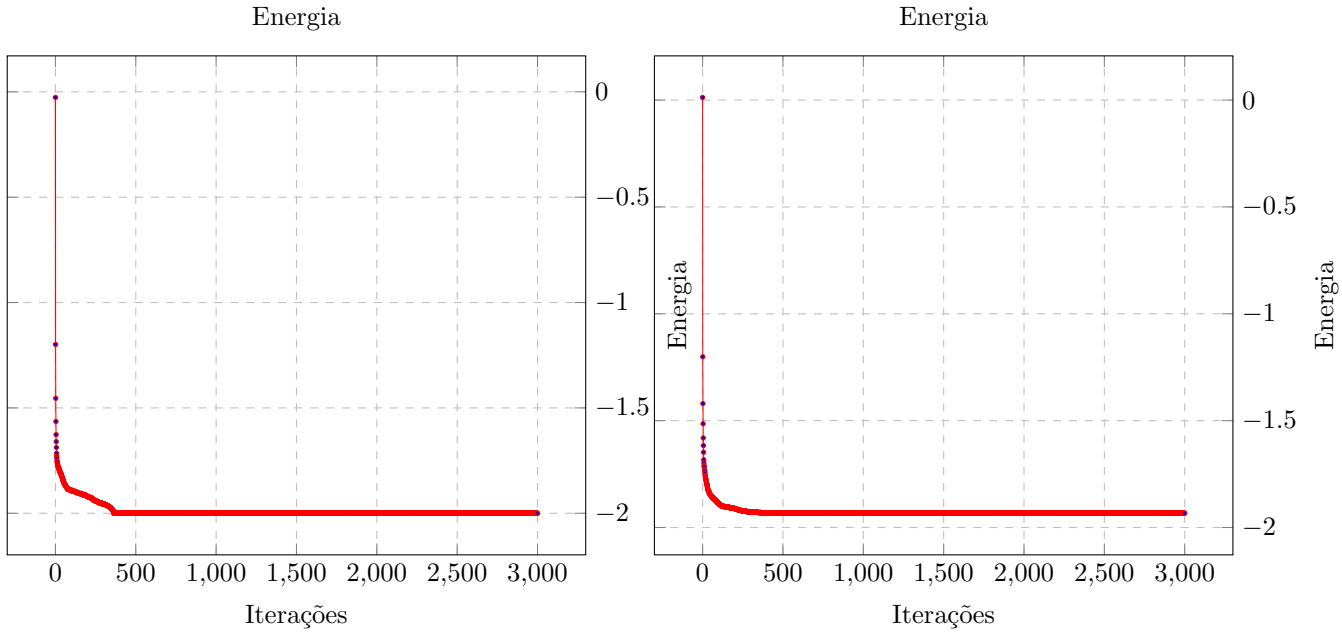
4.2 B2

Iniciando em $\beta = 3$ com uma configuração aleatória, rodaremos as mesmas iterações, como uma têmpera para ver como o sistema se comporta.Aqui, um efeito curioso se manifesta e observaremos o sistema convergir completamente para algumas condições iniciais e não convergir para outras. Mostraremos um caso de cada.

Usaremos,

```
      PROGRAM ISING
           IMPLICIT REAL*8 (A-H,O-Z)
           PARAMETER (L=60, N=10000, MED = 10000, beta = 3)
           BYTE, DIMENSION (1:L,1:L) :: LATTICE
           DIMENSION EXP_RESULTS(-4:4), IPOS(0:L+1)
           CHARACTER*1 SYMBOLS(-1:1)
           COMMON /POSITION/ IPOS
           COMMON /LATTICE/ LATTICE
           COMMON /EXP_RESULTS/ EXP_RESULTS
           COMMON /PARAM/ ENERGY, RMAG

           SYMBOLS(-1) = '-'
           SYMBOLS(1) = '+'

           ITOTAL = L*L
           STEP = 0.01

           iseed = 1234
           CALL SRAND(iseed)

           OPEN(1,FILE='./B/b2ccmag.DAT',STATUS='UNKNOWN')
           OPEN(2,FILE='./B/b2ccene.DAT',STATUS='UNKNOWN')
           OPEN(3,FILE='./B/b2cclattice.DAT',STATUS='UNKNOWN')

           CALL POS()
           CALL EXPONENTIAL(BETA)
           CALL SETUPRANDOM()

           rmag = RFMAG(ITOTAL)
           energy = REnergy()
           WRITE(1,*) 0, rmag/ITOTAL
           WRITE(2,*) 0, energy/ITOTAL

           DO 10 ICICLE = 1, 3000

                DO 30 I=1,N
                     CALL FLIP(ITOTAL)
 30             END DO

                WRITE(1,*) ICICLE, rmag/ITOTAL
                WRITE(2,*) ICICLE, energy/ITOTAL

 10         END DO

           DO 50 K=1,L
                WRITE(3,'(60A2)') (SYMBOLS(LATTICE(I,K)),I=1,L)
50         END DO

           CLOSE(1)
           CLOSE(2)

     END PROGRAM ISING
```

Que nos dá resultados (note escala)

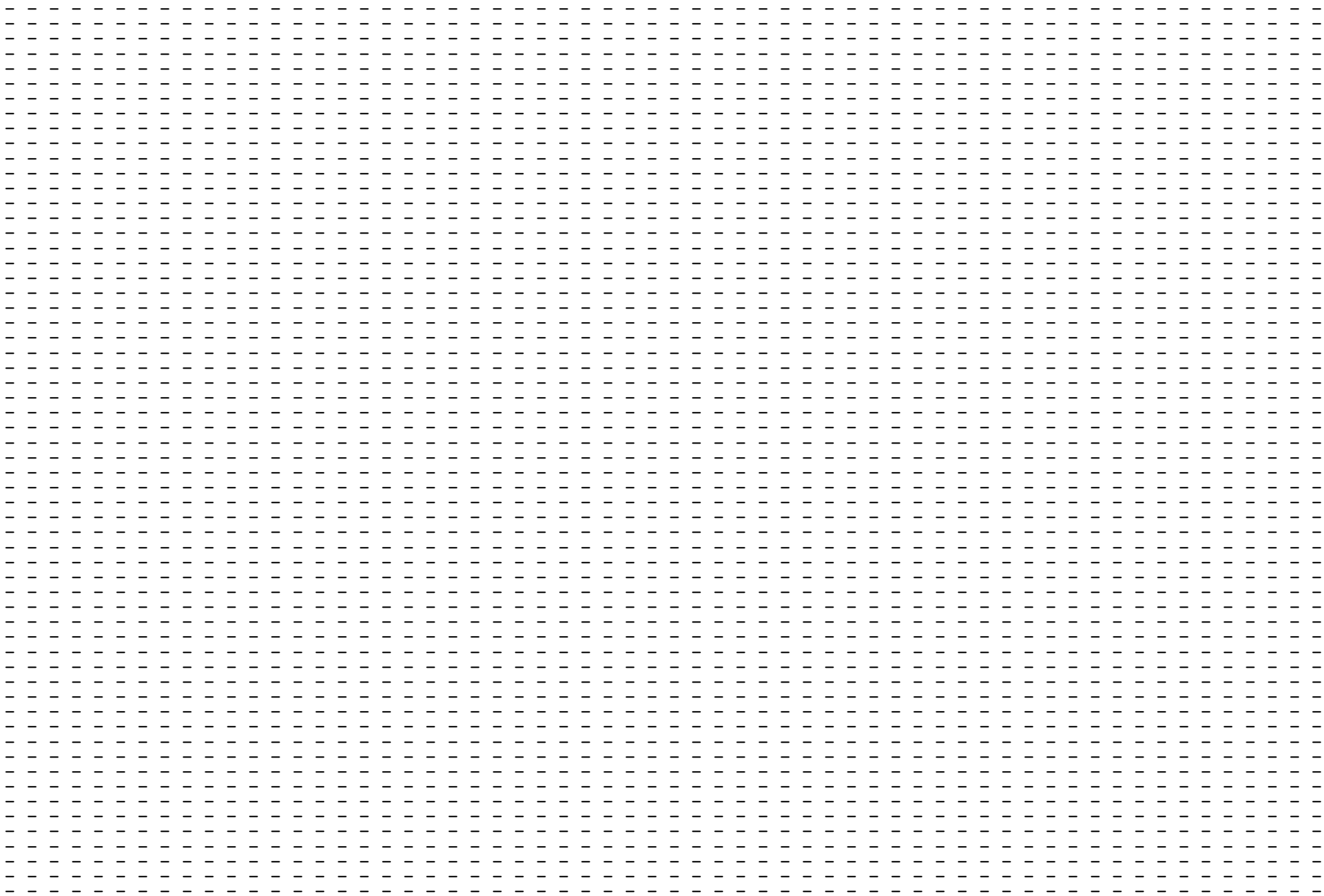


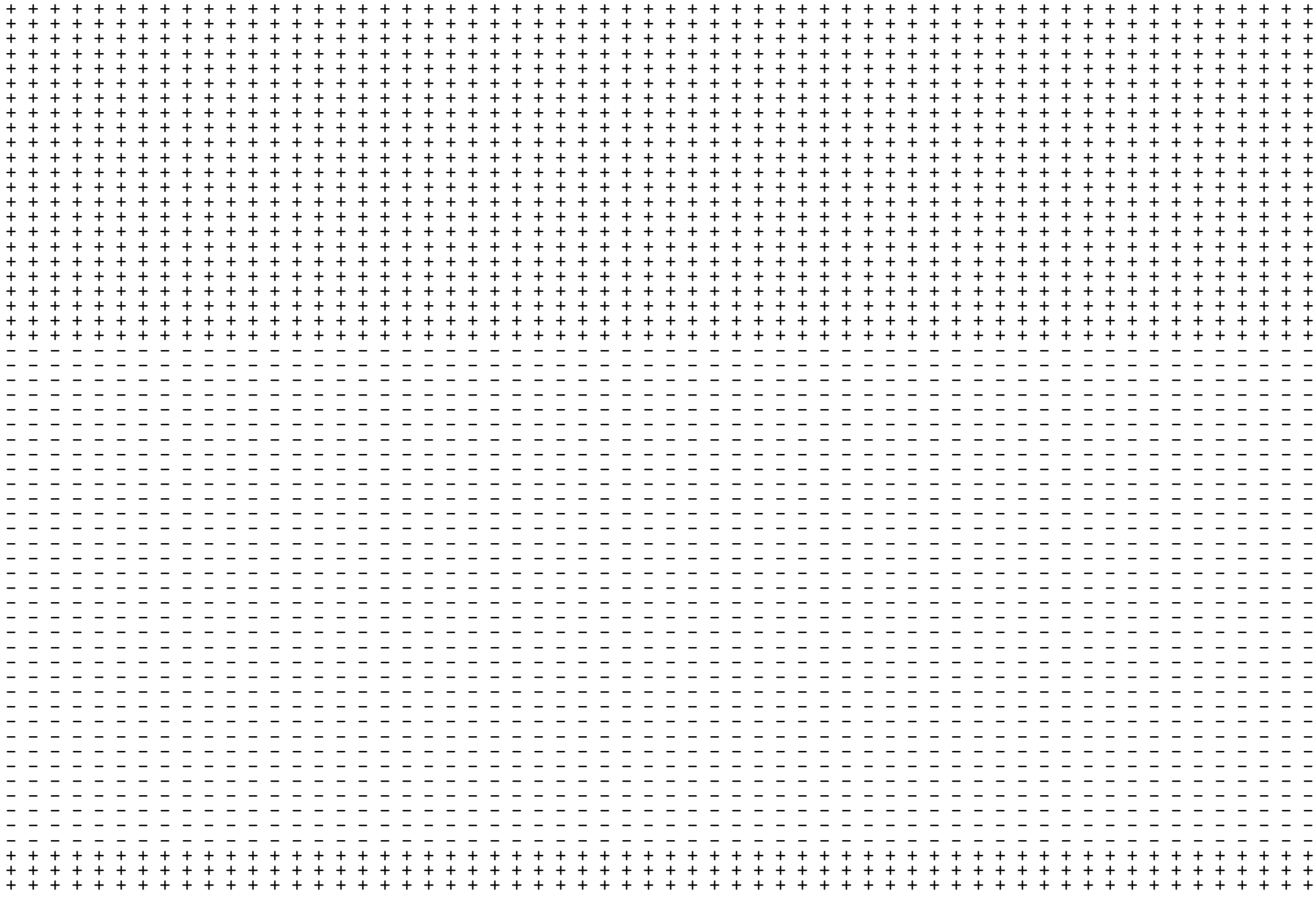
(a) Energia com convergência total

(b) Energia sem convergência total

Figure 4: Magnetizações $\beta = 3$

Alinhamento total





5 Tarefa C

Faremos agora o denominado ciclo térmico onde iniciaremos nosso sistema em uma temperatura no infinito e traremos ela para próxima do zero absoluto e voltaremos para o infinito sem permitir que o sistema entre em equilíbrio. Fazendo uma iteração por temperatura considerada.

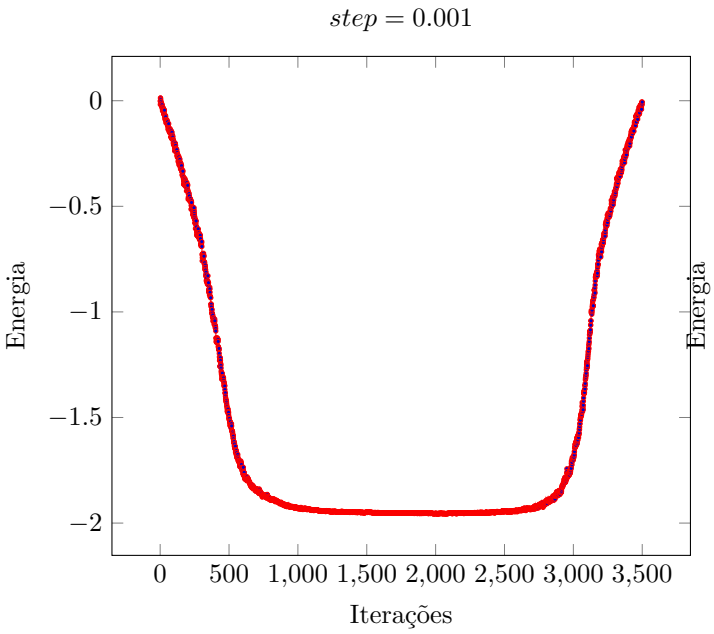
5.1 C1

Além das funções, escrevemos

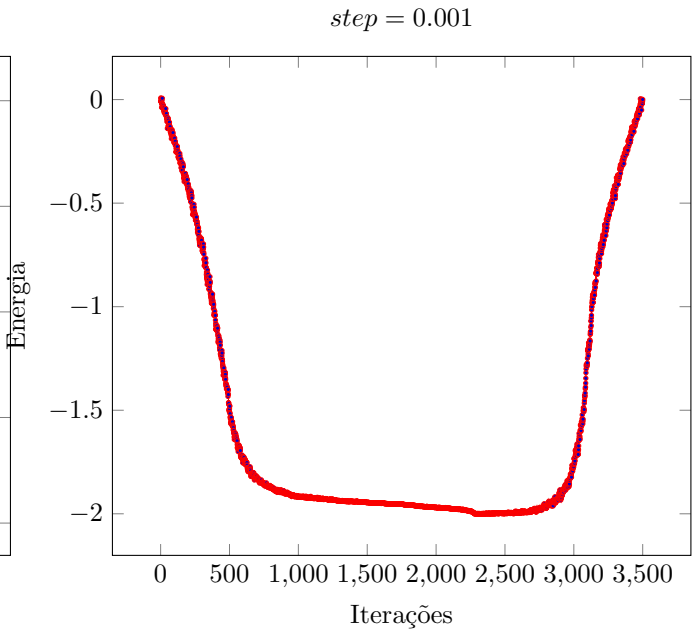
```
      PROGRAM ISING
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=100, N=10000, MED = 10000)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            DIMENSION EXP_RESULTS(-4:4), IPOS(0:L+1)
            CHARACTER*1 SYMBOLS(-1:1)
            COMMON /LATTICE/ LATTICE
            COMMON /EXP_RESULTS/ EXP_RESULTS
            COMMON /PARAM/ ENERGY, RMAG
            COMMON /IPOS/ IPOS

            SYMBOLS(-1) = '-'
            SYMBOLS(1) = '+'

            ITOTAL = L*L

            OPEN(1,FILE='c1mag.DAT',STATUS='UNKNOWN')
            OPEN(2,FILE='c1ene.DAT',STATUS='UNKNOWN')

            CALL SETUPRANDOM()
            CALL POS()
            CALL EXPONENTIAL(0.d0)

            rmag = RFMAG(ITOTAL)
            energy = REnergy()


            WRITE(1,*) 0, rmag/ITOTAL
            WRITE(2,*) 0, Printenergy/Itotal

            DO 10 ICICLE = 1, 1750
                  WRITE(*,*) ICICLE
                  BETA = STEP*ICICLE
                  CALL EXPONENTIAL(BETA)

                  AcumEnergy = 0
                  DO 20 I=1,N
                        CALL FLIP(ITOTAL)
                        AcumEnergy = AcumEnergy + ENERGY
 20               END DO
                  Printenergy = AcumEnergy/N

                  WRITE(1,*) ICICLE, rmag/ITOTAL
                  WRITE(2,*) ICICLE, Printenergy/Itotal

 10         END DO

            DO 30 ICICLE = 1749, 0, -1
                  WRITE(*,*) ICICLE
                  BETA = STEP*ICICLE
                  CALL EXPONENTIAL(BETA)
```

```
52
53                 AcumEnergy = 0
54                 DO 40 I=1,N
55                       CALL FLIP(ITOTAL)
56                       AcumEnergy = AcumEnergy + ENERGY
57  40             END DO
58                 Printenergy = AcumEnergy/N
59
60                 WRITE(1,*) 1750+(1750-ICICLE), rmag/ITOTAL
61                 WRITE(2,*) 1750+(1750-ICICLE), Printenergy/Itotal
62  30         END DO
63
64           CLOSE(1)
65           CLOSE(2)
66
67      END PROGRAM ISING
```

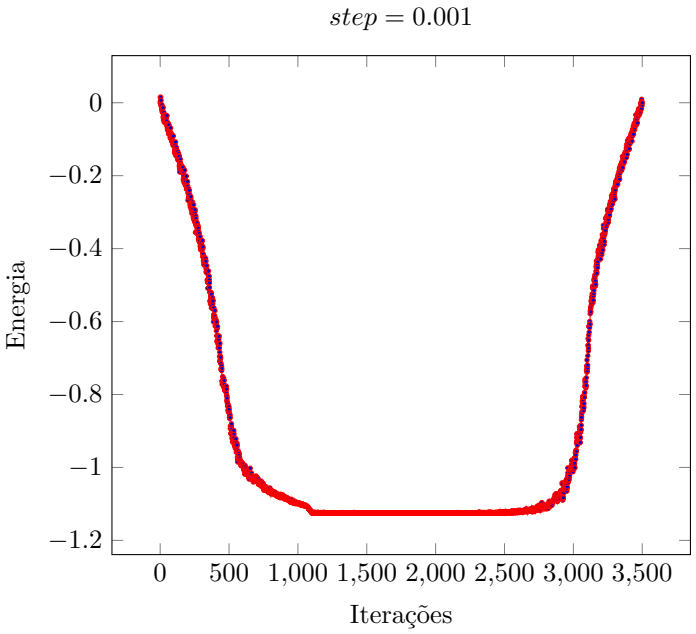
Que resulta para os parâmetros explicitados nos gráficos a seguir. Os dados nos permitem colocar a temperatura de transição em algum ponto entre $0.2 < \beta < 0.4$ mas com pouca precisão.



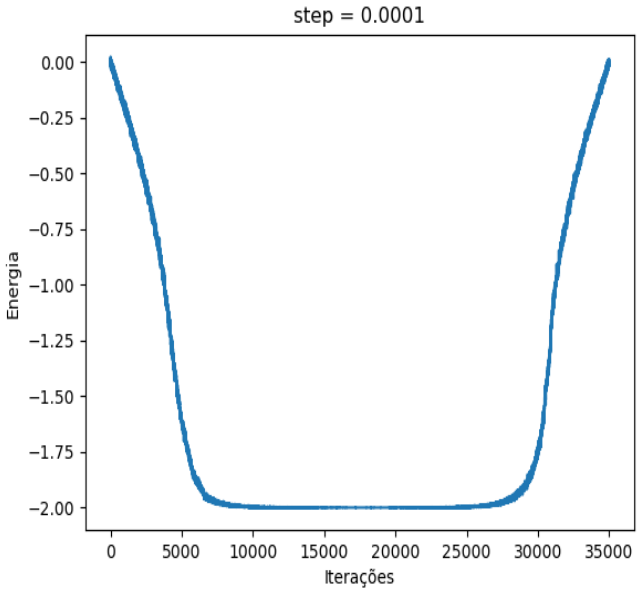
(a) L=100



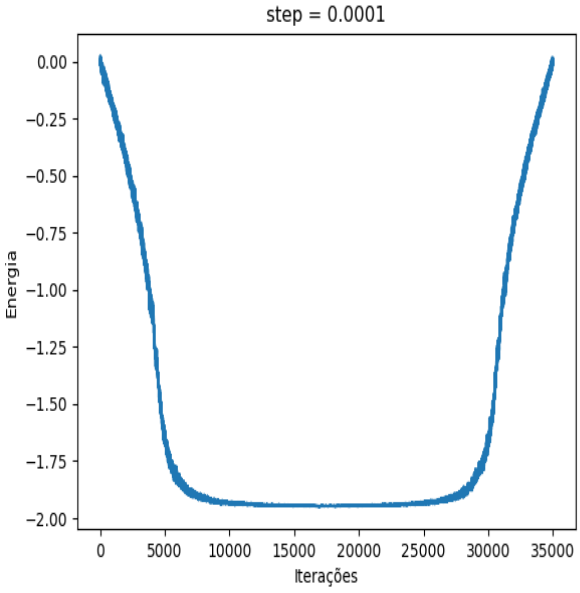
(b) L=80



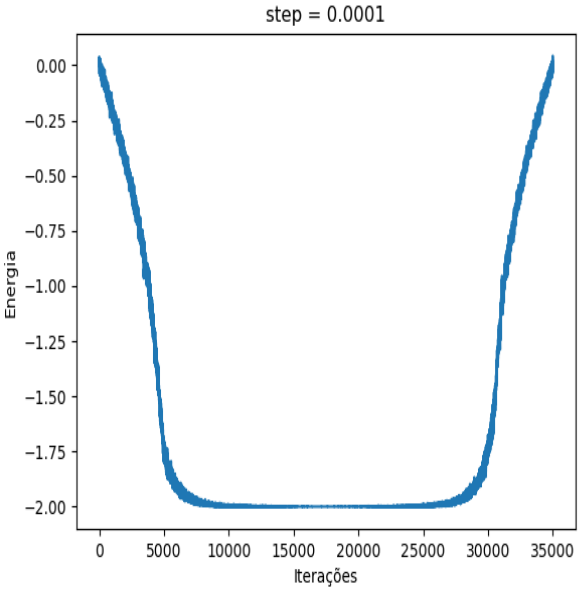
(c) L=60



(d) L=100



(e) L=80



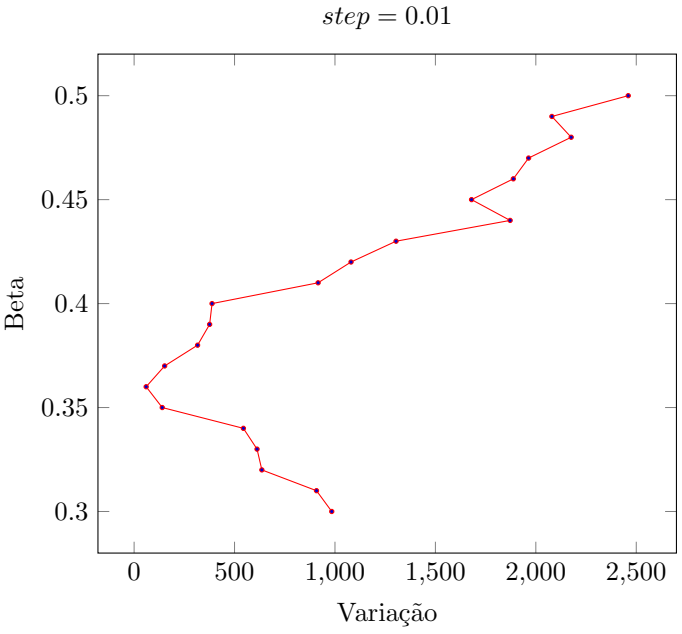
(f) L=60

5.2 C2

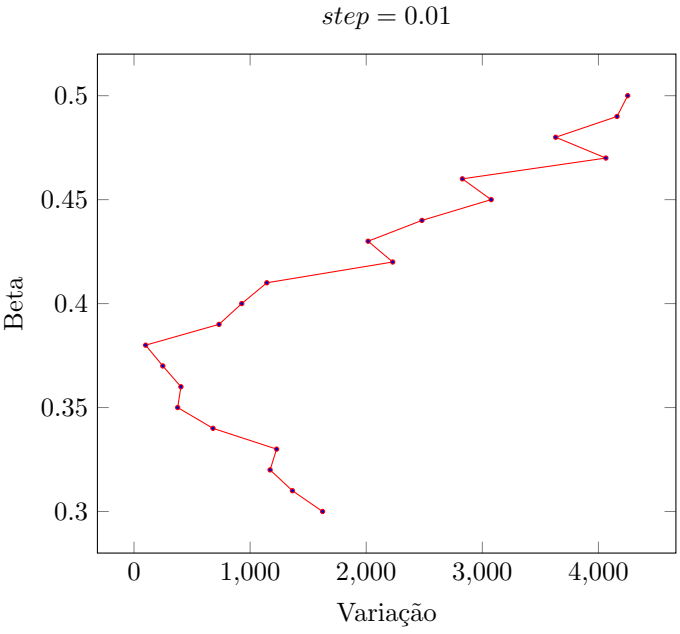
Para esta sub-tarefa exploraremos mais de perto a região de transição para podemos observar, a partir de uma grade intermediaria como a energia altera para diferentes betas. estamos em busca do pico nos gráficos, onde a variação é maior. Podemos mais facilmente agora determinar o β de transição em β próximo de 0.4.

Além das funções, escrevemos

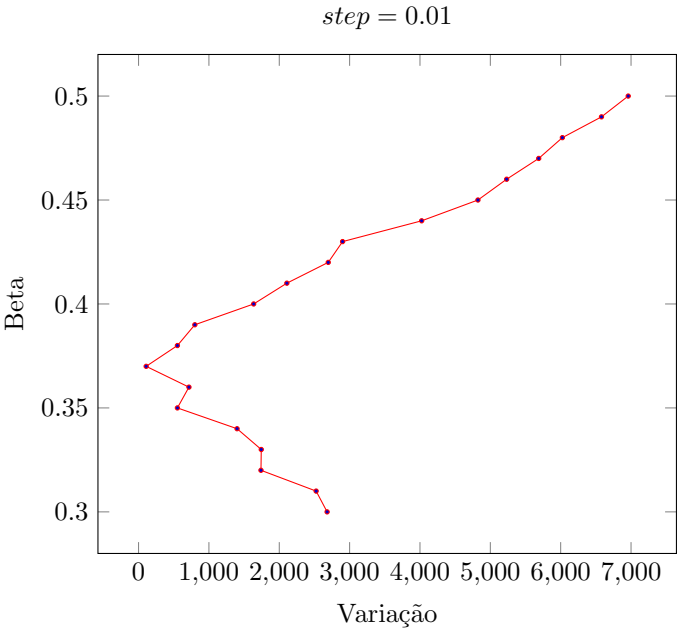
```
      PROGRAM ISING
            IMPLICIT REAL*8 (A-H,O-Z)
            PARAMETER (L=60, N=10000, MED = 10000)
            BYTE, DIMENSION (1:L,1:L) :: LATTICE
            DIMENSION EXP_RESULTS(-4:4), IPOS(0:L+1)
            CHARACTER*1 SYMBOLS(-1:1)
            COMMON /LATTICE/ LATTICE
            COMMON /EXP_RESULTS/ EXP_RESULTS
            COMMON /PARAM/ ENERGY, RMAG
            COMMON /IPOS/ IPOS

            SYMBOLS(-1) = '-'
            SYMBOLS(1) = '+'

            ITOTAL = L*L

            OPEN(2,FILE='./C/c2ene60.DAT',STATUS='UNKNOWN')

            CALL POS()

            STEP = 0.01

            DO 10 ICICLE = 0, 50

                  WRITE(*,*) ICICLE

                  CALL SETUPMIXED()
                  BETA = STEP*ICICLE
                  CALL EXPONENTIAL(BETA)
                  energy = REnergy()
                  rmag = RFMAG(ITOTAL)

                  EI = ENERGY
                  DO 30 K=1,200
                        DO 20 I=1,N
                              CALL FLIP(ITOTAL)
20                      END DO
30                NEND DO
                  EF = ENERGY

                  WRITE(2,*) abs(EF-EI), BETA

10          END DO

            CLOSE(2)

      END PROGRAM ISING
```



(g) L=60



(h) L=80



(i) L=100

6 Tarefa D

Exploraremos nesta seção a quebra espontânea de simetria. Vamos perceber que o tempo de inversão dos dois estados simétricos de energia cresce exponencialmente com L. Para isso, mediremos este tempo para alguns cenários.

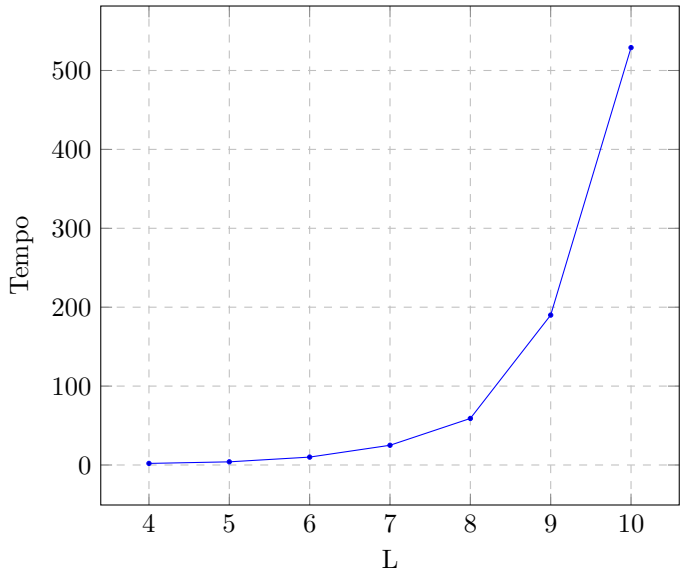
6.1 D1

Escrevemos

```
      PROGRAM ISING
           IMPLICIT REAL*8 (A-H,O-Z)
           PARAMETER (Lm = 10, N=10000, MED = 10000)
           BYTE, DIMENSION (1:Lm,1:Lm) :: LATTICE
           DIMENSION EXP_RESULTS(-4:4), IPOS(0:Lm+1)
           CHARACTER*1 SYMBOLS(-1:1)
           COMMON /POSITION/ IPOS
           COMMON /LATTICE/ LATTICE
           COMMON /EXP_RESULTS/ EXP_RESULTS
           COMMON /PARAM/ ENERGY, RMAG

           SYMBOLS(-1) = '-'
           SYMBOLS(1) = '+'

           STEP = 0.01

           OPEN(1,FILE='dout.DAT',STATUS='UNKNOWN')

           DO 100 L = 4, 10
                 CALL POS(L)

                 BETA = 0.5
                 ITOTAL = L*L
                 CALL EXPONENTIAL(BETA)
                 CALL SETUPRANDOM(L)

                 rmag = RFMAG(ITOTAL, L)
                 energy = REnergy(L)

                 flag = 0
                 icontador = 0
                 iquantas = 0
                 Medtime = 0

                 DO 10 ICICLE = 0, 3000

                       rmag_old = rmag

                       DO 30 I=1,N
                             CALL FLIP(ITOTAL, L)
30                     END DO

                       IF(rmag*rmag_old < 0) THEN
                             WRITE(*,*) Medtime
                             Medtime = Medtime + icontador
                             flag = 0
                             icontador = 0
                             iquantas = iquantas + 1
                       END IF

                       icontador = icontador + 1
```

```
52
53  10              END DO
54
55                  rMedTime = Medtime/iquantas
56                  WRITE(1,*) L, rMedTime
57
58  100         END DO
59
60              CLOSE(1)
61
62      END PROGRAM ISING
```

Tempo médio



Que tem claro comportamento exponencial onde vamos modelar $ae^{-bx} + c$ e teremos $a = 0.0148$, $b = -1.0486$, $c = 0.7623$.