Termodinâmica Estatística I

O Jogo do Quanta

Rafael Soares Andrade 96921
Humberto Sousa Martins 93724

Objetivo

O objetivo do projeto é, a partir de ferramentas computacionais, resolver o exemplo 4.2 do livro-texto (o jogo do quanta) e obter os gráficos necessários.

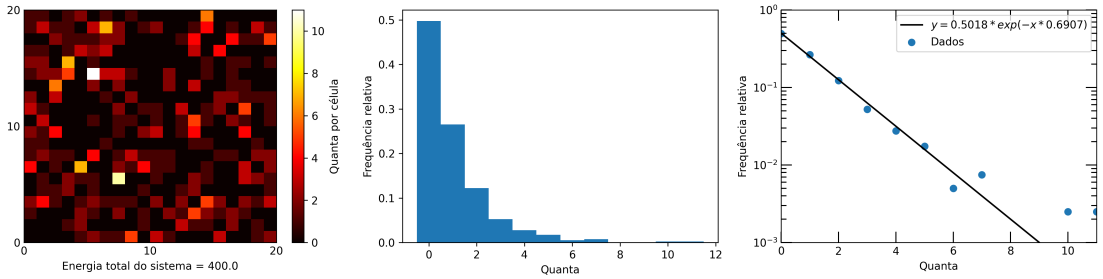
Procedimento e Dados

O projeto foi desenvolvido na linguagem de programação Python versão 3.6, a partir da ferramenta de edição e compilação "Google Colaboratory". Para a construção dos gráficos, foi utilizada a ferramenta de construção de gráficos Matplotlib. Foram usadas as bibliotecas numpy para manejo de matrizes e operações vetoriais e biblioteca scipy para ajuste de curvas.

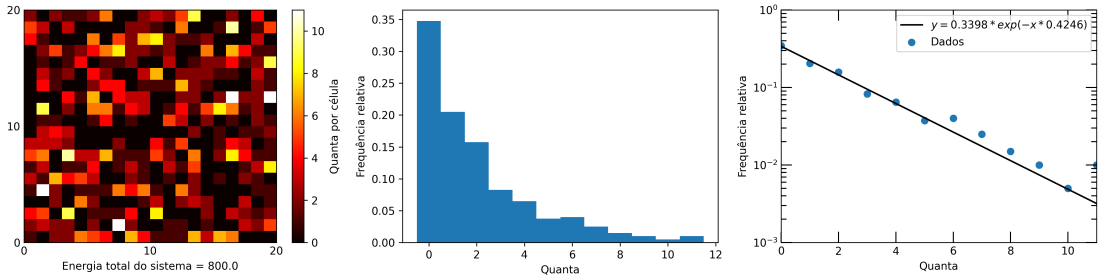
O primeiro objetivo é criar uma rede de células, denominada *"grid"* no programa, e preencher todas as células com um valor inicial de energia ε_0. Para isso foi definida uma função *"rede_inicial"* que recebe os parâmetros do tamanho do lado da rede (N) e valor de energia por célula inicial de ε_0 predefinidos, preenche todas as células com ε_0. Com isso a função retorna uma rede de tamanho $N \times N$ e a configuração energética desejada.

Com a rede inicial construída, desejamos efetuar a troca de um quanta de energia em um par de células aleatórias, para isso vamos diminuir em uma unidade a energia de uma célula aleatória e alocar em outra célula aleatória. Para implementar esse procedimento foi desenvolvida a função *"atualizar"* que efetua as trocas. Tal função recebe a rede inicial e o número de iterações a serem feitas (variável *"n_it"*) correspondente ao número de trocas de energia e os demais parâmetros das variantes opcionais, que serão explicados adiante. Foi percebido empiricamente que um valor de iterações mínimas necessárias para um completo descorrelacionamento do sistema pode ser dado pela equação $10 \times \varepsilon_0 \times N \times N$. A função testa se a célula sorteada possui um quanta de energia, e caso essa e caso não possua sorteia outra célula aleatória. Ao escolher com sucesso uma célula para perder energia, outra célula é sorteada para receber a mesma quantidade de energia. Esse procedimento se repete *"n_it"* vezes, e retorna um "grid" da rede atualizada pela função.

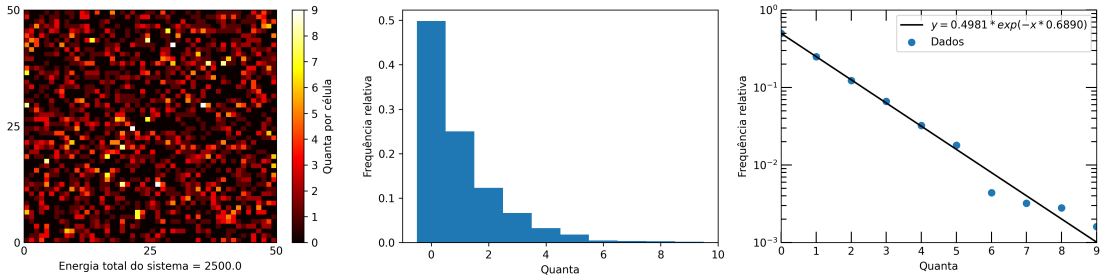
A função *"fit_boltzmann"* que define uma distribuição de Boltzmann a ser usada na construção do gráfico para ajuste de curvas. Para a construção dos gráficos, foi definida a função *"plotar"* que a partir da rede atualizada pela função *"atualizar"* imprime a imagem de distribuição de rede, o histograma em escala linear e em escala semi-logarítimica, além de, se requisitado por um parâmetro booleano, utiliza da função *"fit_boltzmann"* para realizar o ajuste de Boltzmann no histograma semi-log, como indicado na figura 1. A 1 mostra alguns resultados das funções que serão usadas posteriormente.



(a) Rede de tamanho 20x20 e $\varepsilon_0 = 1$



(b) Rede de tamanho 20x20 e $\varepsilon_0 = 2$



(c) Rede de tamanho 50x50 e $\varepsilon_0 = 1$

Figura 1: Gráfico da rede atualizada com as trocas de energia entre as células (esquerda). Histograma de energia por célula: (centro). Ajuste de Boltzmann para o histograma de frequência em escala semi-logarítimica..

A função *"atualizar"* permite modificação dos parâmetros *"gauss"* e *"negative"* para que a energia retirada de cada célula seja um valor real aleatório baseado em uma distribuição gaussiana e remover a restrição de energia negativa, respectivamente. Assim, obtemos resultados da evolução de sistemas com esses parâmetros, cujos gráficos estão a seguir:

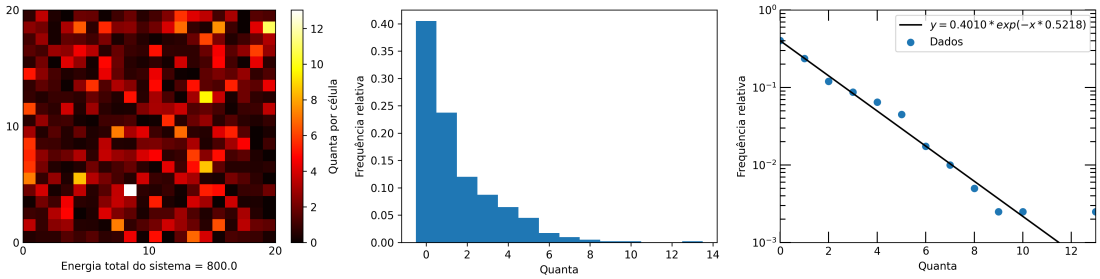


Figura 2: Rede com tamanho 20x20, $\varepsilon_0 = 2$ e a energia é trocada de acordo com o parâmetro *"gauss"*. Gráfico da rede atualizada com as trocas de energia entre as células (esquerda). Histograma de energia por célula: (centro). Ajuste de Boltzmann para o histograma de freqência em escala semi-logarítimica.

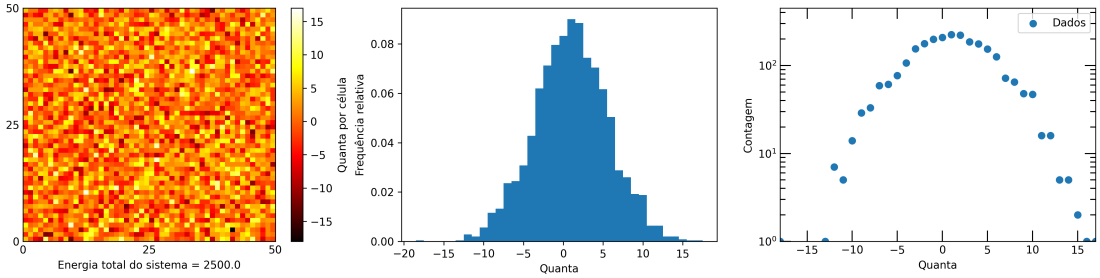


Figura 3: A rede acima possui tamanho 50x50, $\varepsilon_0 = 1$ e a energia é trocada de acordo com o parâmetro *"negative"*. Gráfico da rede atualizada com as trocas de energia entre as células (esquerda). Histograma de energia por célula: (centro)

O gráfico a seguir mostra 3 distribuições de uma rede inicial nas mesmas condições, atualizado pelos métodos *"Quanta"*, *"Gauss"* e *"Negative"*.

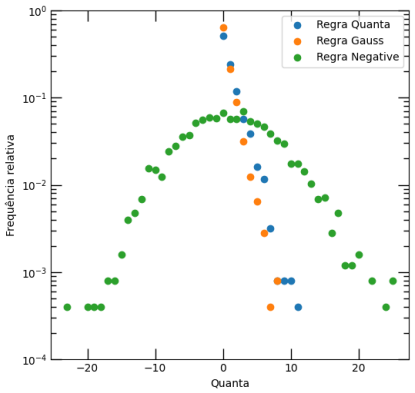
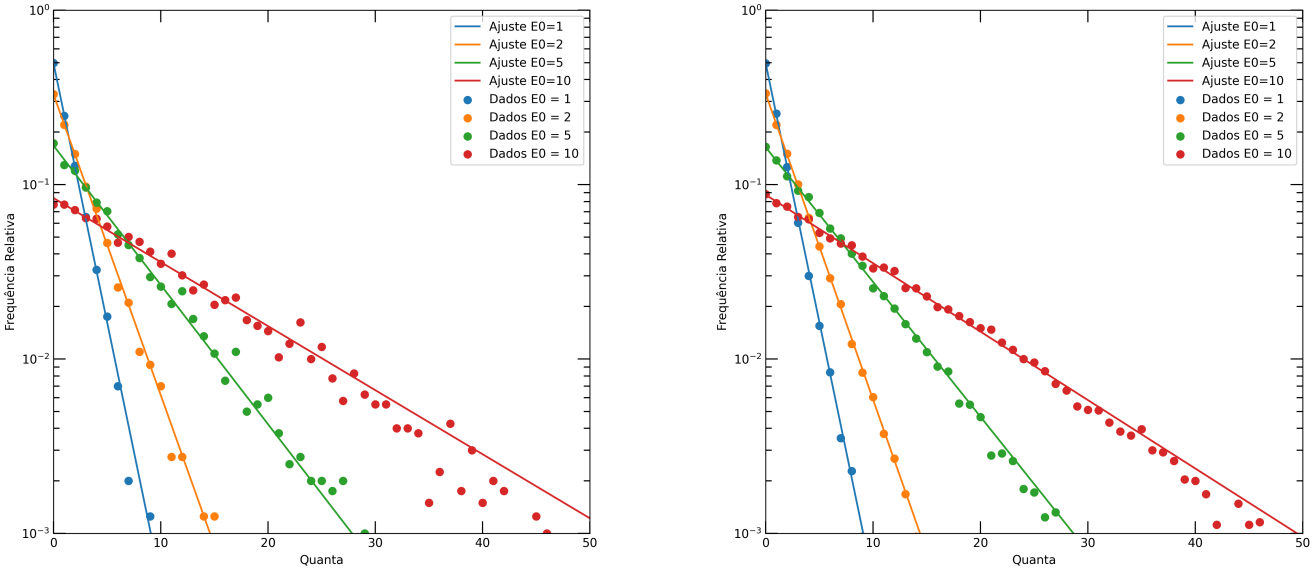


Figura 4: Gráfico de distribuições da rede sob diferentes condições. Rede com tamanho 50x50, $\varepsilon_0 = 1$

Foi definida a função *"experimento"* para efetuar repetições dos procedimentos anteriores. Para isso, a função recebe os valores do tamanho da rede,um conjunto de valores de ε_0, o número de interações e o número de repetições a serem feitas. A função então aciona as outras funções definidas anteriormente (*"rede_inicial"*, *"atualizar"* e *"plotar"*) pelo número de vezes determinado. Os dados são gerados e as

médias são obtidas. Os dados são plotados em um gráfico semi-log com o ajuste de curva de Boltzmann, como na figura 5



(a) Rede de tamanho 20x20

(b) Rede de tamanho 50x50

Figura 5: Gráfico da média dos valores de 10 repetições para um conjunto de ε_0 e diferentes tamanhos.

A tabela a seguir mostra os valores de ajuste das curvas mostradas, onde a função de ajuste toma a forma

$$y = a \times e^{-x \times b} \tag{1}$$

N	ε_0	a	b
20	1	0.4955823425311841	0.6780722932993565
	2	0.33244237577162006	0.40325198052340017
	5	0.16381696049896396	0.1773993575198439
	10	0.08595534082369743	0.08808754087327404
50	1	0.49737494752346273	0.6834202659125916
	2	0.3326119748751181	0.40393217676906434
	5	0.16394284553845295	0.17794585859318657
	10	0.08742895069881995	0.09035792496870305

Podemos relacionar o parâmetro b do ajuste com o parâmetro β da função de Boltzmann. Temos que $\beta = k_b T$, podemos considerar $k_b = 1\frac{J}{K}$ pois estamos trabalhando com unidades arbitrárias , logo $T = \frac{1}{b}K$.

O gráfico a seguir mostra que T aumenta linearmente com o aumento de ε_0, portanto, podemos entender que ε_0 faz o papel de temperatura no sistema. É notável também que T não está correlacionado fortemente com o tamanho do sistema (para ε_0 baixos os dados se sobrepõem).

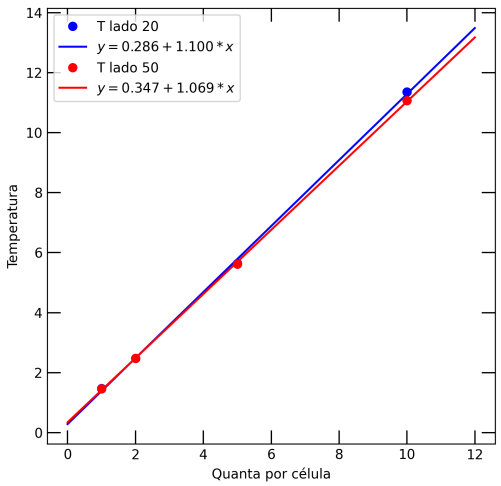


Figura 6: Gráfico dos valores de T encontrados por ε_0 inicial.

Conclusão

Para o experimento nas condições propostas pelo roteiro, podemos observar que certas variáveis tem um análogo físico, sendo o tamanho da rede análogo ao volume do sistema, a energia de cada célula análogo a energia cinética de uma partícula. O processo de atualizar a rede pode ser entendido como uma "termalização", pois leva o sistema de um macroestado altamente improvável para um macroestado esmagadoramente mais provável, e a distribuição do sistema após a atualização pode ser entendida como uma distribuição de partículas que tem uma energia cinética dada por uma função de Boltzmann, ou seja, a definição de temperatura.

Observamos que mesmo sistemas com tamanhos diferentes, por exemplo o de lado 20 e 50 das figuras 1a e 1c , por possuírem o mesmo valor de ε_0, possuem a mesma "densidade energética média", assim como distribuições de energia final muito semelhantes. Podemos também observar, através da 6, que o parâmetro b do ajuste está relacionado com a temperatura do sistema e que este parâmetro é linearmente dependente de ε_0. Assim podemos concluir que ε_0 faz o papel de temperatura do sistema.

Observamos também que nos sistemas onde foi permitido um valor real de transferência de energia, o sistema termalizou mais rapidamente, mas chegou em uma distribuição próxima do sistema quantizado.

Já nos sistemas onde foi permitida que uma célula assumisse energias negativas, foi observado que o sistema não forma uma distribuição de Boltzmann , e sim uma distribuição gaussiana centrada em ε_0. Tal comportamento faz sentido pois o jogo agora pode ser interpretado basicamente como uma "caminhada aleatória", onde cada célula tem o papel de um andador.

O código fonte e observações sobre sua execução seguem em anexo!

Seguem os documentos anexos.

OBS: O código fonte está formatado para ser compilado pelo programa "Jupyter Notebook" ou o Google Colaboratory, e deve ser executado em Python nas versões 3.6 ou superior. Os comentários foram escritos sem acentos e cedilha, por motivos de compatibilidade com o LaTeX

Recomendamos acesso ao código através do link:
https://colab.research.google.com/drive/1XmRtK3rzlX5JjVQpoMis_Rjp9PuxERQ8?usp=sharing

---

```
# -*- coding: utf-8 -*-
"""Lista Termo 2 ATUALIZADA.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1
       XmRtK3rzlX5JjVQpoMis_Rjp9PuxERQ8
"""

import matplotlib.pyplot as plt
import numpy as np
from matplotlib.ticker import MultipleLocator
from scipy.stats import norm
from scipy.optimize import curve_fit

rng = np.random.default_rng()

# Gera uma rede inicial e preenche conforme regras predefinidas
def rede_inicial(N,tipo,E0):
  grid = np.zeros((N,N),dtype=float)

  if tipo=='E0': # preenche todos os espacos da rede com um mesmo valor de
     energia de E0
    grid[:][:] = E0

  if tipo=='i+j': # preenche todos os espacos da rede de acordo com sua
     posicao i,j , sendo o valor i+j
    for i in range(N):
      for j in range(N):
        grid[i][j] = i+j

  if tipo=='random': # um espa o aleatorio recebe toda energia do sistema
    i,j=rng.integers(0,N,size=2)
    grid[i][j] = N*N*E0

  return grid # Retorna uma rede de lado NxN com a distribuicao de energia
     desejada

# TESTE DA FUN  O rede_inicial()

# Gera as condicoes iniciais do sistema
N = 20  # lado da rede quadrada
E0 = 1  # quanta inicial por celula

grid = rede_inicial(N,'E0',E0)
```

```
print(grid)

# Define a funcao de ajuste de boltzmann que sera usada no ajuste de curva
    no codigo a seguir
  def fit_boltzmann(x, a, b):
    return a * np.exp(-x*b)

# Plota o grafico da distribuicao de quantas de energia e histogramas
def plotar(grid,fit=False,savefig=False):
  fig, ax = plt.subplots(1,3,figsize=(18,4),dpi=100)
  N = len(grid)
# Plota a imagem da distribuicao na rede
  ax[0].imshow(grid,'hot',aspect = 'auto',extent = (0,N,0,N))
  ax[0].set_xlabel(f'Energia total do sistema = {grid.sum()}')
  ax[0].tick_params(length=0)
  ax[0].set_xticks([0,N/2,N])
  ax[0].set_yticks([0,N/2,N])
  ax[0].set_aspect('equal')

  gridcbar = ax[0].imshow(grid,'hot',aspect = 'auto',extent = (0,N,0,N))
  cbar = fig.colorbar(gridcbar,ax=ax[0], orientation="vertical")
  cbar.set_label('Quanta por c lula')

# Plota o histograma
  bins = range(int(grid.min()),int(grid.max())+2)
  hist = ax[1].hist(list(grid.reshape(1,N*N)),bins=bins,align='left',density
     =True,histtype='stepfilled')
  ax[1].set_xlabel('Quanta')
  ax[1].set_ylabel('Frequ ncia relativa')
  ax[1].set_aspect('auto')


# Plota o histograma no formato semi-log
  x=hist[1][0:-1]
  y=hist[0]
  ymin=1

  ax[2].scatter(x,y,label='Dados')
  ax[2].set_yscale('log')
  ax[2].set_ylim(1e-3,1)
  ax[2].set_xlim(x.min(),x.max())
  ax[2].tick_params(direction='in',which='minor',length=5, width = 1 , right
      = 'on' , top = 'on')
  ax[2].tick_params(direction='in',length=10, width = 1 , right = 'on' , top
      = 'on')
  ax[2].set_xlabel('Quanta')
  ax[2].set_ylabel('Frequ ncia relativa')
  ax[2].set_aspect('auto')

# Se ativado, faz o fit dos dados em uma funcao de boltzman a * exp(-x*b)
    plota o resutado
  if fit:
    popt, pcov = curve_fit(fit_boltzmann, x, y,p0=(0.5,0.5))
    a,b = popt
    ax[2].plot(x,fit_boltzmann(x,a,b),label=fr'$y = {a:.4f} * exp(-x*{b:.4f
       })$',c='k')
    ax[2].legend()
```

```
      if savefig:
        plt.savefig(f'NEG_lado_{N}_E0_{grid.sum()/(N*N)}.png',dpi=300)
      return x,y,(a,b)


    if savefig:
      plt.savefig(f'NEG_lado_{N}_E0_{grid.sum()/(N*N)}.png',dpi=300)


    pass
    return x,y

# juntando as 3 maneiras de atualizar

N = 50 # tamanho da rede quadrada
E0 = 2 # quanta inicial por celula
n_it = 20*E0*N*N # Numero de iteracoes que a rede ira sofrer


grid = rede_inicial(N,'E0',E0)
gridQ = atualizar(grid,n_it,negative=False,gauss=False)
grid = rede_inicial(N,'E0',E0)
gridG = atualizar(grid,n_it,negative=False,gauss=True)
grid = rede_inicial(N,'E0',E0)
gridN = atualizar(grid,n_it,negative=True,gauss=False)

x,y=[],[]
for i in [gridQ,gridG,gridN]:
  bins = range(int(i.min()),int(i.max())+2)
  hist = plt.hist(list(i.reshape(1,N*N)),bins=bins,align='left',density=True
     ,histtype='stepfilled')
  plt.close()
  x.append(hist[1][0:-1])
  y.append(hist[0])

x,y

fig,ax=plt.subplots(figsize=(6,6),dpi=100)
ax.plot(x[0],y[0],'o',label='Regra Quanta')
ax.plot(x[1],y[1],'o',label='Regra Gauss')
ax.plot(x[2],y[2],'o',label='Regra Negative')

ax.set_yscale('log')
ax.set_ylim(1e-4,1)
#ax.set_xlim(x.min(),x.max())
ax.tick_params(direction='in',which='minor',length=5, width = 1 , right = '
   on' , top = 'on')
ax.tick_params(direction='in',length=10, width = 1 , right = 'on' , top = '
   on')
ax.set_xlabel('Quanta')
ax.set_ylabel('Frequ ncia relativa')
ax.set_aspect('auto')
ax.legend()
plt.savefig(f'TRES_lado_{N}_E0_{grid.sum()/(N*N)}.png',dpi=100)

y
```

```
# TESTE DA FUNÇÃO plotar()
x,y,c= plotar(grid,fit=True,savefig=False)
y

# Atualiza os valores das celulas pelas regras definidas no roteiro
def atualizar(grid,n_it,gauss=False,negative=False):
  for count in range(n_it):
    if negative: # Caso o argumento negative=True a funcao atualizar()
       podera tirar quanta de celulas com energias menor que 1
      i,j=rng.integers(0,N,size=2)
    else: # Caso negative=False, a função ira testar se a celula tem
       energia >0 antes de retirar um quanta, caso seja <0 ira sortear outra
        celula
      sucesso=False
      while sucesso==False: #testa se uma celula aleatoria tem mais de 0
         quanta de energia
        i,j=rng.integers(0,N,size=2)
        if grid[i][j] > 0:
          sucesso=True

    if gauss: # Caso gauss=True a funcao atualizar() ira sortear um valor
       baseado em uma gaussiana para ser removido e acrescido das celulas
      q=abs(np.random.normal(grid[i][j]/2,0.1))
      if q>grid[i][j]: q= grid[i][j]
    else: # Caso gauss=False a funcao assume que um quanta de energia sera
       retirado e adicionado a outra celula
      q=1

    grid[i][j] -=q
    i,j=rng.integers(0,N,size=2)   # Sorteia novos valores de i,j
    grid[i][j] +=q

  return grid # Retorna uma grid modificada pela funcao

# TESTE DA FUNÇÃO atualizar()
N = 20 # tamanho da rede quadrada
E0 = 10 # quanta inicial por celula
n_it = 20*E0*N*N # Número de iteracoes que a rede ira' sofrer

grid = rede_inicial(N,'E0',E0)
grid = atualizar(grid,n_it,gauss=False,negative=False)
print('Formato de grid de saída',grid.shape,'Soma dos valores de energia',
   grid.sum(),'\nGrid:\n',grid)

# Este bloco de codigo junta as três funções definidas anteriormente,
   podemos escolher os parametros e recebemos os resultados

N = 50 # tamanho da rede quadrada
E0 = 10 # quanta inicial por celula
n_it = 10*E0*N*N # Numero de iteracoes que a rede ira sofrer

grid = rede_inicial(N,'E0',E0)
grid = atualizar(grid,n_it,negative=False,gauss=True)
x,y,c = plotar(grid,fit=True,savefig=True)
c

# Este bloco de codigo realiza um 'experimento', realizando a simulacao
```

```
    definida anteriormente deiversas vezes (variavel 'repeticoes'), e
    coletando o valor medios dos resultados
def experimento(N,E0,repeticoes,fit=True,savefig=False):
  fig, ax = plt.subplots(1,1,figsize=(8,8),dpi=100)
  ajuste = []
  for e0 in E0:
    n_it = 20*e0*N*N
    y_tot=np.zeros(100)
    x_max=np.zeros(1)
    for i in range(repeticoes): # Rodamos as tres fun  es pelo numero de
        repeticoes definido
      grid = rede_inicial(N,'E0',e0)
      grid = atualizar(grid,n_it)
      x,y = plotar(grid)
      plt.close()
      for j,k in enumerate(y): # Os histogramas sao gerados e somados a cada
          repeticao
        y_tot[j]+=k
      if len(x) > len(x_max): # Garante que o valor em x do histograma ser
          igual ao maior valor de x das repeticoes
        x_max=x
    y_final=y_tot[0:len(x_max)]/repeticoes # Divide o resultado obtido pelo
        numero de repeticoes para obeter a media

  # Plotamos o grafico semi-log do resultado obtido e fazemos o ajuste de
     curva pela funcao de Boltzmann e salvamos a imagem, o codigo e' igual
     ao definido na funcao plotar()
    ax.scatter(x_max,y_final,label=f'Dados E0 = {e0}')
    if fit:
      popt, pcov = curve_fit(fit_boltzmann, x_max, y_final,p0=(0.5,0.5))
      a,b = popt
      ax.plot(x_max,fit_boltzmann(x_max,a,b),label=f'Ajuste E0={e0}')
      T = (1/(b))
      ajuste.append((N,e0,a,b,T))

  # Propriedades da figura
  ax.legend()
  ax.set_yscale('log')
  ax.set_ylim(1e-3,1)
  ax.set_xlim(x_max.min(),50)
  ax.tick_params(direction='in',which='minor',length=5, width = 1 , right =
      'on' , top = 'on')
  ax.tick_params(direction='in',length=10, width = 1 , right = 'on' , top =
      'on')
  ax.set_xlabel('Quanta')
  ax.set_ylabel('Frequ ncia Relativa')
  if savefig:
    plt.savefig(f'N_{N}_E0_{e0}_n_it_{n_it}_repeticoes_{repeticoes}.png',dpi
        =300)


  return y_final,x_max,ajuste # Retorna o grafico e os valores finais de y e
      x do histograma

#TESTE DA FUNCAO experimento()

N = 20 # tamanho da rede quadrada
```

```
E0 = [1,2,5,10] # quanta inicial por espa o
#n_it = 10*e0*N*N # Numero de iteracoes que a rede ira' sofrer
repeticoes=10

y,x,ajuste = experimento(N,E0,repeticoes,savefig=True)

ajuste

dados20 = [(20, 1, 0.4955823425311841, 0.6780722932993565,
   1.4747690030722407),
 (20, 2, 0.33244237577162006, 0.40325198052340017, 2.4798390294377524),
 (20, 5, 0.16381696049896396, 0.1773993575198439, 5.636998994701207),
 (20, 10, 0.08595534082369743, 0.08808754087327404, 11.35234324952534)]
dados50 = [(50, 1, 0.49737494752346273, 0.6834202659125916,
   1.4632284845470158),
 (50, 2, 0.3326119748751181, 0.40393217676906434, 2.475663137308615),
 (50, 5, 0.16394284553845295, 0.17794585859318657, 5.619686841300219),
 (50, 10, 0.08742895069881995, 0.09035792496870305, 11.067097881523578)]

x,y=[],[]
fig, ax = plt.subplots(1,1,figsize=(6,6),dpi=100)
xx = np.linspace(0,12,1000)

def linha(x,a,b):
  return a+b*x


for N, E0, a, b, T in dados20:
  x.append(E0)
  y.append(T)
ax.plot(x,y,'ob',label='T lado 20')
popt, pcov = curve_fit(linha,x, y,p0=(0.5,0.5))
a,b=popt
ax.plot(xx,linha(xx,a,b),'b',label = f'$y={a:.3f}+{b:.3f}*x$')

x,y=[],[]
for N, E0, a, b, T in dados50:
  x.append(E0)
  y.append(T)
ax.plot(x,y,'or',label='T lado 50')
popt, pcov = curve_fit(linha,x, y,p0=(0.5,0.5))
a,b=popt
ax.plot(xx,linha(xx,a,b),'r',label = f'$y={a:.3f}+{b:.3f}*x$')

#ax.set_yscale('log')
#ax.set_ylim(1e-3,1)
#ax.set_xlim(x_max.min(),50)
ax.tick_params(direction='in',which='minor',length=5, width = 1 , right = '
   on' , top = 'on')
ax.tick_params(direction='in',length=10, width = 1 , right = 'on' , top = '
   on')
ax.set_xlabel('Quanta por c lula')
ax.set_ylabel('Temperatura')
ax.legend()

plt.savefig(f'parametros.png',dpi=300)
```