Modelo_de_Ising_2D_-_Enumeracao_Exata_-_Guilherme_de_Abreu_Lima_Buitrago_Miranda_-_2018054788

June 27, 2021

1 Introdução à Física Estatística Computacional

1.1 Modelo de Ising 2D - Enumeração Exata

Aluno: Guilherme de Abreu Lima Buitrago Miranda

Matrícula: 2018054788

1.1.1 Imports

[1]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('seaborn-colorblind')
plt.ion()
```

1.1.2 Funções

As funções abaixo foram implementadas conforme pseudocódigo fornecido no enunciado do exercício.

[2]:
```
def create_neighborhood_array(l, n):
    neighborhood = []
    for i in range(n):
        # creating list of lists
        neighborhood.append([])

        if (i + 1) % l == 0:
            neighborhood[i].append(i + 1 - l)
        else:
            neighborhood[i].append(i + 1)

        if i + 1 > n - l:
            neighborhood[i].append(i + l - n)
        else:
            neighborhood[i].append(i + l)
```

```
        if i % l == 0:
            neighborhood[i].append(i + l - 1)
        else:
            neighborhood[i].append(i - 1)

        if i < l:
            neighborhood[i].append(i + n - l)
        else:
            neighborhood[i].append(i - l)

    return np.array(neighborhood)
```

[3]:
```
def calculate_E(n, s, neighborhood):
    E = 0
    for i in range(n):
        h = s[neighborhood[i, 0]] + s[neighborhood[i, 1]]
        E -= s[i] * h

    return E
```

[4]:
```
def gray_flip(t, n):
    k = t[0]
    if k + 1 > n:
        return None

    t[k] = t[k + 1]
    t[k + 1] = k + 1

    if k != 0:
        t[0] = 0
    return k, t
```

[5]:
```
def enumera_ising(n, neighborhood):
    g = np.zeros((4 * n) + 1)
    s = np.repeat((np.array(-1)), n)

    t = np.arange(n + 1)
    e_ = np.arange(4 * n + 1) - 2 * n

    E = -2 * n
    g[E + 2*n] = 2

    for i in range(2 ** (n - 1) - 1):
        k, t = gray_flip(t, n)
        h = np.sum( s[neighborhood[k,:]] )
        E = E + 2 * s[k] * h
        g[E + 2 * n] = g[E + 2 * n] + 2
```

```
        s[k] = -s[k]

    return g[g > 0], e_[g > 0]
```

[6]:
```
def media_termodinamica(g, e_range, beta, n):
    Z = 0
    var_e_ = 0
    var_e_2 = 0

    for i, e_ in enumerate(e_range):
        e_ -= e_range[0]
        Z += g[i] * np.exp( (-beta * e_) )
        var_e_ += e_ * g[i] * np.exp( (-beta * e_) )
        var_e_2 += (e_ *  e_) * g[i] * np.exp( (-beta * e_) )

    var_e_ /= Z
    var_e_2 /= Z
    Z *= np.exp(- beta * e_range[0])
    cv = (beta * beta) * (var_e_2 - (var_e_ * var_e_)) / n
    var_e = (var_e_ + e_range[0]) / n

    return Z, var_e, cv
```

[7]:
```
def execute_all(l, n):
    neighborhood = create_neighborhood_array(l, (l * l))
    if (l == 6):
        #obs: o código não rodou a tempo da data de entrega.
        #Como sugerido por outros colegas, obtive os dados prontos pela internet

        g = np.array([2, 72, 144, 1620, 6048, 35148, 159840,
              804078, 3846576, 17569080, 71789328, 260434986, 808871328,␣
↪2122173684,
              4616013408, 8196905106, 11674988208, 13172279424, 11674988208,␣
↪8196905106,
              4616013408, 2122173684, 808871328, 260434986, 71789328,␣
↪17569080, 3846576, 804078,
              159840, 35148, 6048, 1620, 144, 72, 2])

        e = np.array([-72, -64, -60, -56, -52, -48, -44, -40, -36, -32, -28,␣
↪-24, -20, -16,
              -12, -8, -4, 0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48,␣
↪52, 56, 60, 64, 72])
    else:
        g, e = enumera_ising(n, neighborhood)

    x = np.linspace(1, 5, 50)
    cvs = []
```

```
    Zs = []
    es = []
    for x_ in x:
        z, e_, cv = media_termodinamica(g, e, 1/x_, n)
        cvs.append(cv)
        Zs.append(z)
        es.append(e_)

    return cvs, Zs, es, x, g, e
```

[14]:
```
def plot(y_list, x):
    for y in y_list:
        plt.plot(x, y)
```

1.1.3 Execução

As células abaixo invocam os métodos que realizam os cálculos e, em seguida, geram os seguintes gráficos:

i) ln(( ))como função de / ;

ii) Energia por spin, , como função da temperatura;

iii) Calor específico, , como função da temperatura;

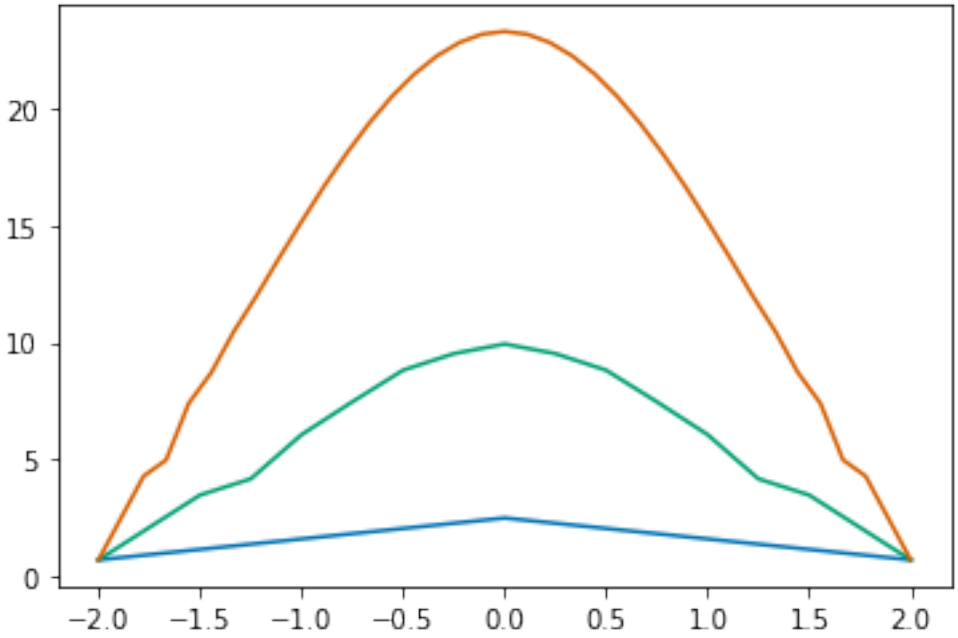
iv) Energia livre por spin, ( )=− 1/ ln ;

v) Entropia por spin, ( ) = − ( ) / .

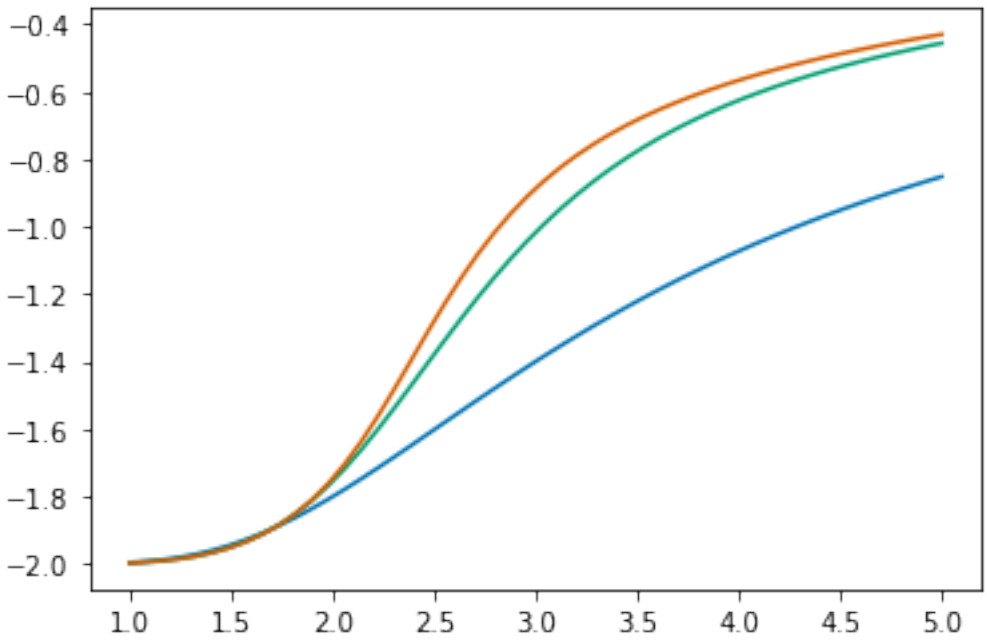
[8]:
```
ls = [2, 4, 6]
ns = [l * l for l in ls]

ns.append(6 * 6)

cvs = []
Zs = []
es = []
xs = []
gs = []
es_ = []

for l in ls:
    cv, Z, e, x, g, E = execute_all(l, (l * l))
    cvs.append(cv)
    Zs.append(Z)
    es.append(e)
    xs.append(x)
    gs.append(g)
    es_.append(E)
```

[9]:
```
plt.plot((es_[0]/ns[0]), np.log(gs[0]))
plt.plot((es_[1]/ns[1]), np.log(gs[1]))
plt.plot((es_[2]/ns[2]), np.log(gs[2]))
```

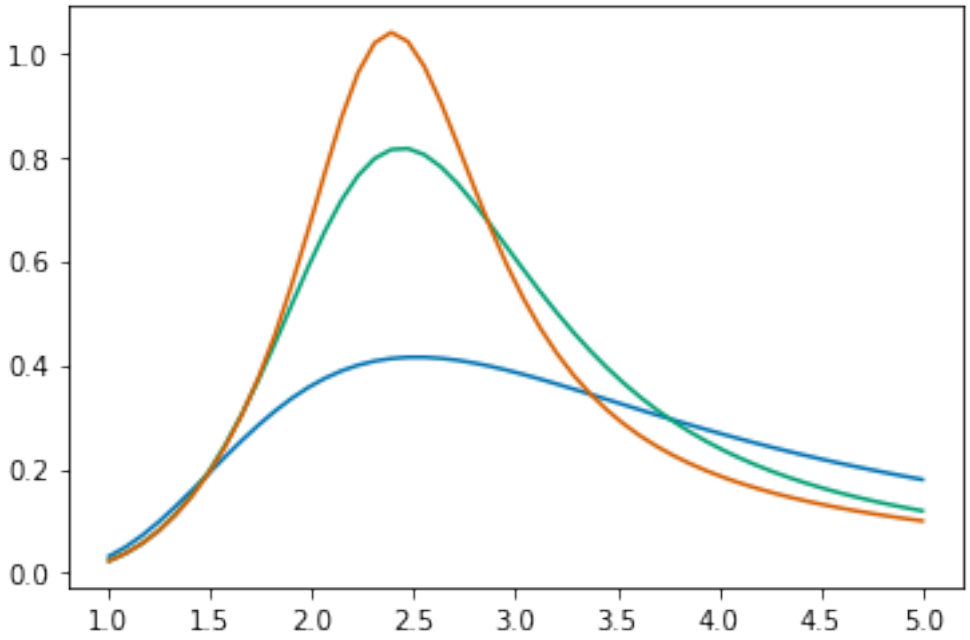
[9]:
```
[<matplotlib.lines.Line2D at 0x7f5a2bce2b50>]
```



[10]:
```
plot(es, x)
```

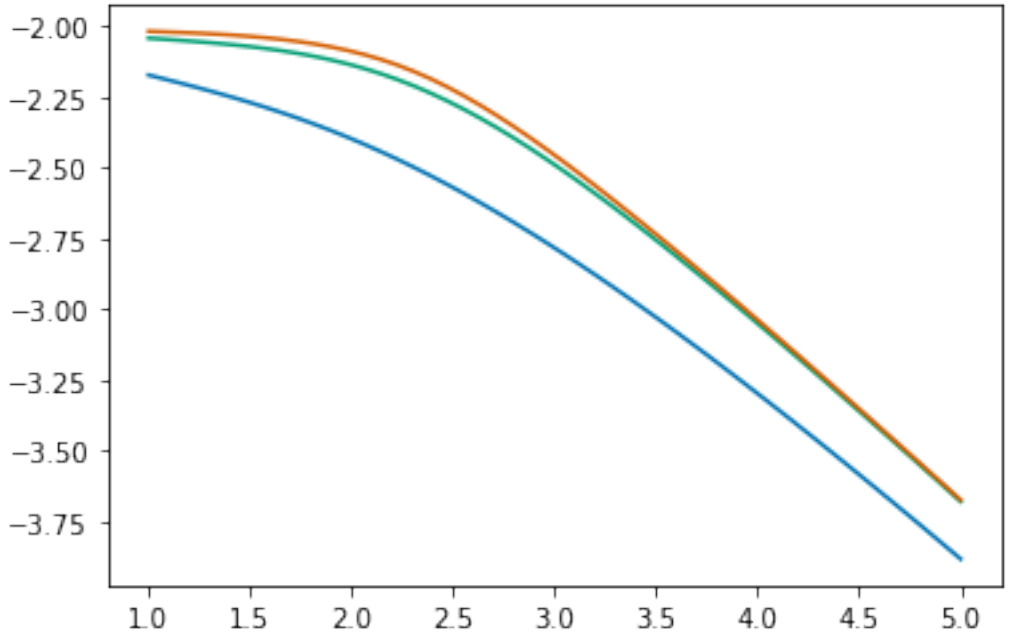


[11]:
```
plot(cvs, x)
```

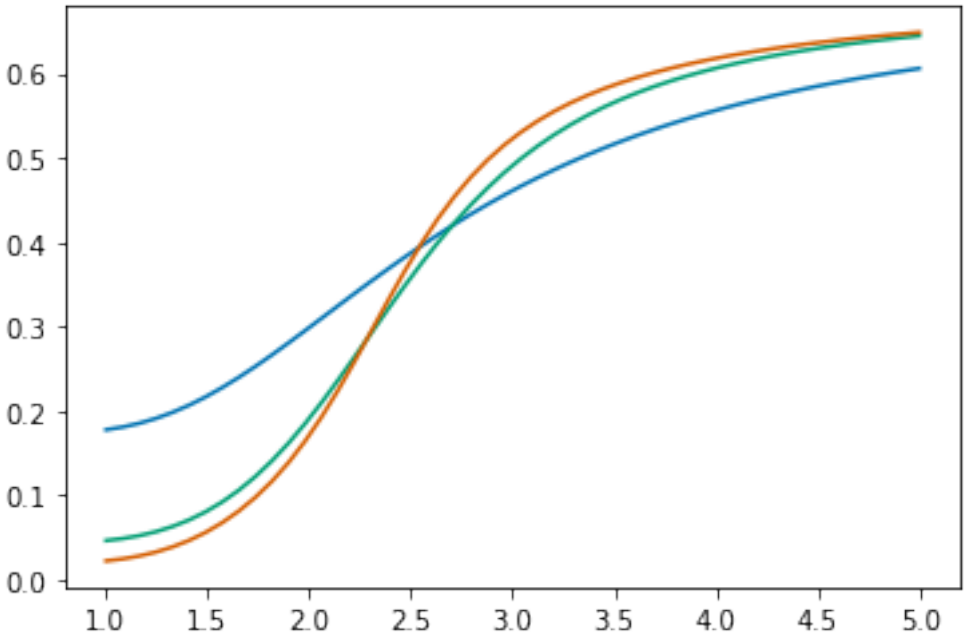


[12]:
```
def calc_f(T, Z, n):
    beta = 1/T
```

```
    y = (-1/(n * beta)) * np.log(Z)
    return y

fs = []
fs.append(calc_f(x, Zs[0], ns[0]))
fs.append(calc_f(x, Zs[1], ns[1]))
fs.append(calc_f(x, Zs[2], ns[2]))

plot(fs, x)
```



[13]:
```
def calc_s(T, e_, f):
    return ((e_ - f)/T)

ss = []
ss.append(calc_s(x, es[0], fs[0]))
ss.append(calc_s(x, es[1], fs[1]))
ss.append(calc_s(x, es[2], fs[2]))

plot(ss, x)
```



1.1.4 Análises dos resultados obtidos

Antes de partir para a análise dos resultados propriamente ditos, quero reiterar que os dados da execução do modelo para a instância 6x6 do problema foram obtidos pela internet, visto que o longo tempo para a obtenção do resultado inviabilizaria a entrega do trabalho dentro do prazo.

Assim, com relação aos resultados e gráficos obtidos, pode-se perceber, com relação ao primeiro, que independente do tamanho do sistema, há uma concentração da energia no ponto central da visualização, ou seja, sistemas com alta energia são raros.

Já no gráfico de energia por spin, nota-se que, a medida que a temperatura do sistema aumenta, a energia por spin também aumenta. Além disso, as curvas tem um formato de S, o que significa que crescem mais rápido no começo do aumento da energia e, no fim, desaceleram o crescimento.

O gráfico de calor específico em função da temperatura gerado encontra-se em conformidade com visto do livro de referência, assim como a explicação apresentada neste.

Com relação ao quarto gráfico apresentado, vê-se que a energia livre por spin decresce a medida que a temperatura aumenta, enquanto a entropia por spin cresce (também em formato de S), conforme pode ser visto no gráfico 5.

[ ]: