Trabalho III - Método de Wang-Landau

Gabriel Victor Carvalho Rocha - 2018054907

In [2]:
```python
import matplotlib.pyplot as plt
from numba import jit
import numpy as np
import math
```

In [3]:
```python
N_ITER = 10000000
FLATNESS = 0.8
```

In [4]:
```python
@jit(nopython=True)
def estado_inicial(N):
    s = np.zeros(N, dtype=np.int8)
    for i in range(N):
        s[i] = np.sign(2 * np.random.random() - 1)
    return s
```

In [5]:
```python
@jit(nopython=True)
def vizinhos(L, N):
    viz = np.zeros((N, 4), dtype = np.int16)
    for k in range(N):
        viz[k, 0] = k + 1
        if (k + 1) % L == 0: viz[k, 0] = k + 1 - L
        viz[k, 1] = k + L
        if k > (N - L - 1): viz[k, 1] = k + L - N
        viz[k, 2] = k - 1
        if k % L == 0: viz[k, 2] = k + L - 1
        viz[k, 3] = k - L
        if k < L: viz[k, 3] = k + N - L
    return viz
```

In [6]:
```python
@jit(nopython=True)
def energia(s, viz, N):
    ener = 0
    for i in range(N):
        h = s[viz[i, 0]] + s[viz[i, 1]]
        ener -= s[i] * h
    ener = int((ener + 2 * N) / 4)
    return ener
```

In [7]:
```python
@jit(nopython=True)
def min_h(H, N):
    min_h = H[0]
    for i in range(2, N - 1):
        if H[i] < min_h: min_h = H[i]
    if H[-1] < min_h: min_h = H[-1]
    return min_h
```

In [8]:
```python
@jit(nopython=True)
def wang_landau(L, N_ITER, FLATNESS):
    N = L * L
    s = estado_inicial(N)
    viz = vizinhos(L, N)
    ener = energia(s, viz, N)
    ln_ge = np.zeros(N + 1, dtype=np.float64)
    H = np.zeros(N + 1, dtype=np.int64)
    Hc = np.zeros(N + 1, dtype=np.int64)
    m_micro = np.zeros(N + 1, dtype=np.float64)
    ln_f = 1.0
    m = s.sum()
    for it in range(N_ITER):
        for imc in range(N):
            k = np.random.randint(0, N - 1)
            h = s[viz[k, 0]] + s[viz[k, 1]] + s[viz[k, 2]] + s[viz[k, 3]]
            ener_2 = ener + int(s[k] * h * 0.5)
            if ln_ge[ener] > ln_ge[ener_2]:
                s[k] = -s[k]
                ener = ener_2
                m -= 2 * s[k]
            else:
                p = np.exp(ln_ge[ener] - ln_ge[ener_2])
                if np.random.random() < p:
                    s[k] = -s[k]
                    ener = ener_2
                    m -= 2 * s[k]
            H[ener] += 1
            ln_ge[ener] += ln_f
            m_micro[ener] += abs(m)
        if it % 1000 == 0:
            h_med = float(H.sum()) / float((N - 1))
            h_min = min_h(H, N)
            if h_min > (FLATNESS * h_med):
                Hc += H
                H = np.zeros(N + 1, dtype = np.int64)
                ln_f = 0.5 * ln_f
        if ln_f < 0.00000001: break
    m_micro = m_micro / Hc
    ln_ge = ln_ge - ln_ge[0] + np.log(2)
    return ln_ge, m_micro, ener, s
```

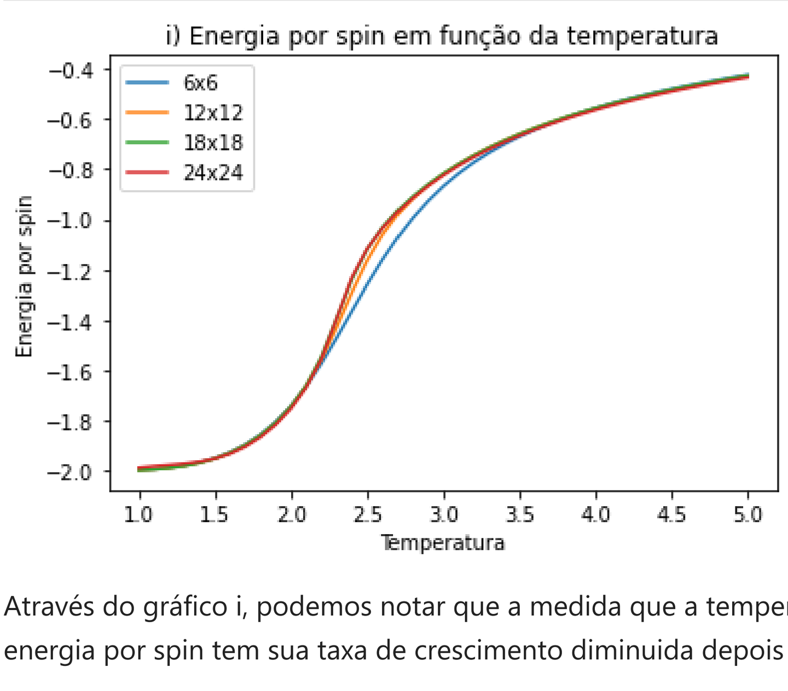
In [9]:
```python
@jit(nopython=True)
def energia_reescalada(e, N):
    return e * 4 - 2 * N
```

In [10]:
```python
@jit(nopython=True)
def maior_exp(b, N):
    maior = 0
    for e_esc in range(N):
        e = energia_reescalada(e_esc, N)
        if -b * e > maior:
            maior = -b * e
    return maior
```

In [15]:
```python
@jit(nopython=True)
def media_termodinamica(ln_ge, t, L):
    N = L * L
    ge = np.exp(ln_ge)
    z = e_t = e_t2 = 0
    b = 1 / t
    e_min = 0
    maior = maior_exp(b, N)
    for e_esc in range(N - 1):
        if e_esc == 1: continue
        e = energia_reescalada(e_esc, N)
        e_ = e - e_min
        expoente = np.exp(-b * e_ - maior)
        z += ge[e_esc] * expoente
        e_t += e_ * ge[e_esc] * expoente
        e_t2 += (e_ ** 2) * ge[e_esc] * expoente
    e_t /= z
    e_t2 /= z
    z *= np.exp(-b * e_min)
    cv = ((b ** 2) * (e_t2 - (e_t ** 2))) / N
    e = (e_t + e_min) / N
    return z, e, cv
```

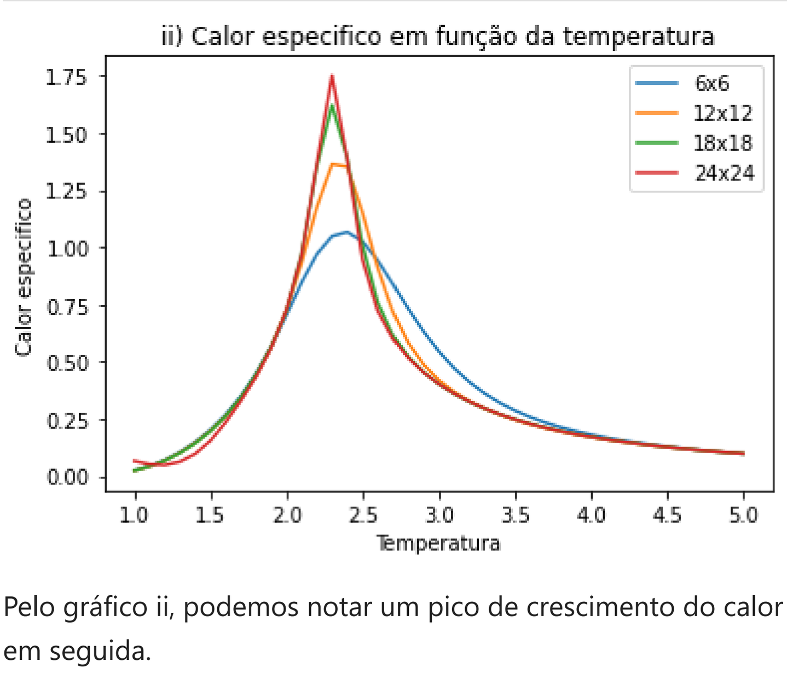
In [12]:
```python
ln_ge, m_micro = {}, {}
temperaturas = np.arange(start = 1, stop = 5.1, step = 0.1, dtype = np.float64)

for L in [6, 12, 18]:
    ln_ge_t, m_micro_t, ener_t, s_t = wang_landau(L, N_ITER, FLATNESS)
    ln_ge[L] = ln_ge_t
    m_micro[L] = m_micro_t

ln_ge[24] = np.array([0.69314718055994529, 7.0356163661506894, 7.7300972025421384, 12.708447990897012, 14.09136
m_micro[24] = np.array([568.16672687098605, 564.34123407338723, 562.51675534113497, 562.12983421828142, 559.830
```

In [16]:
```python
#i) Energia por spin, e, como função da temperatura

def grafico_temperatura_energiaporspin(L):
    e = np.array([])

    for t in temperaturas:
        z_t, e_t, cv_t = media_termodinamica(ln_ge[L], t, L)
        e = np.append(e, e_t)

    plt.plot(temperaturas, e, label=f"{L}x{L}")

for L in [6, 12, 18, 24]:
    grafico_temperatura_energiaporspin(L)

plt.xlabel("Temperatura")
plt.ylabel("Energia por spin")
plt.legend()
plt.title("i) Energia por spin em função da temperatura")
plt.show()
```



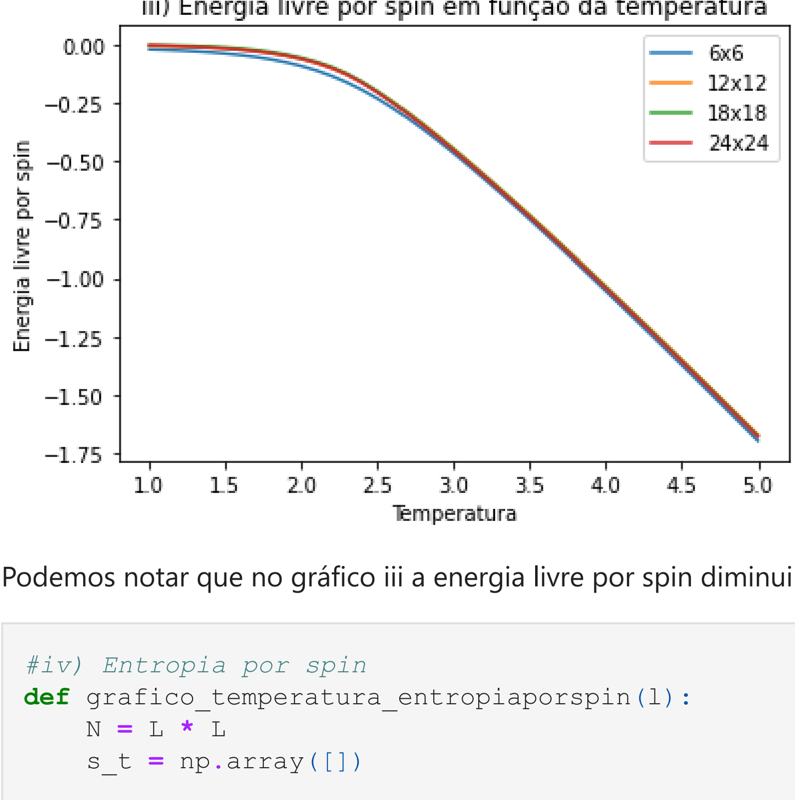
Através do gráfico i, podemos notar que a medida que a temperatura aumenta, o valor da energia por spin também aumenta, mas a energia por spin tem sua taxa de crescimento diminuida depois da temperatura 3.

In [17]:
```python
#ii) Calor específico, cv, como função da temperatura

def grafico_temperatura_caloresp(L):
    cv = np.array([])

    for t in temperaturas:
        z_t, e_t, cv_t = media_termodinamica(ln_ge[L], t, L)
        cv = np.append(cv, cv_t)

    plt.plot(temperaturas, cv, label=f"{L}x{L}")

for L in [6, 12, 18, 24]:
    grafico_temperatura_caloresp(L)


plt.xlabel("Temperatura")
plt.ylabel("Calor especifico")
plt.legend()
plt.title("ii) Calor especifico em função da temperatura")
plt.show()
```



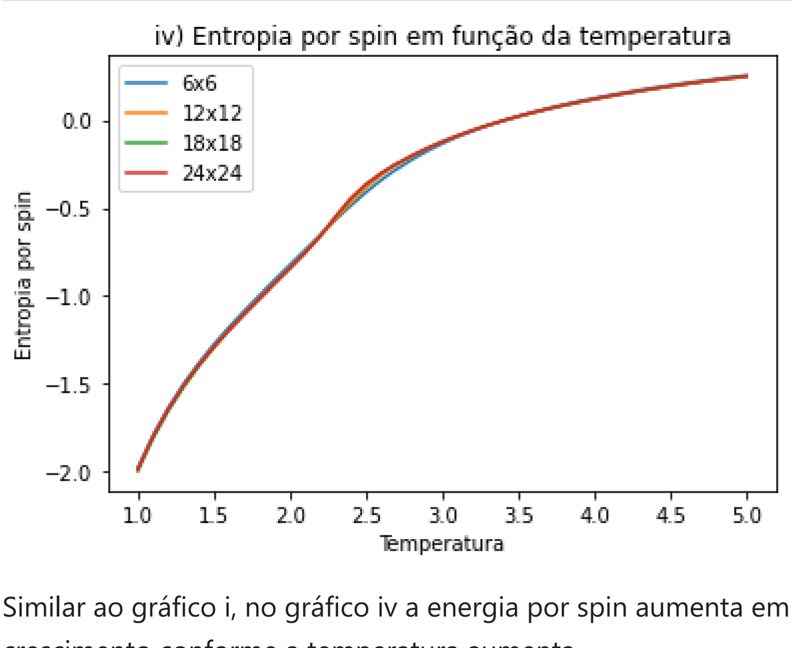
Pelo gráfico ii, podemos notar um pico de crescimento do calor especifico entre as temperaturas 2 e 2.5, tendo uma grande queda logo em seguida.

In [18]:
```python
#iii) Energia livre por spin
def grafico_temperatura_energialivreporspin(L):
    N = L * L
    f_t = np.array([])

    for t in temperaturas:
        z_t, e_t, cv_t = media_termodinamica(ln_ge[L], t, L)
        f_t = np.append(f_t, -(1 / (N * (1 / t))) * math.log(z_t))

    plt.plot(temperaturas, f_t, label=f"{L}x{L}")

for L in [6, 12, 18, 24]:
    grafico_temperatura_energialivreporspin(L)


plt.xlabel("Temperatura")
plt.ylabel("Energia livre por spin")
plt.legend()
plt.title("iii) Energia livre por spin em função da temperatura")
plt.show()
```



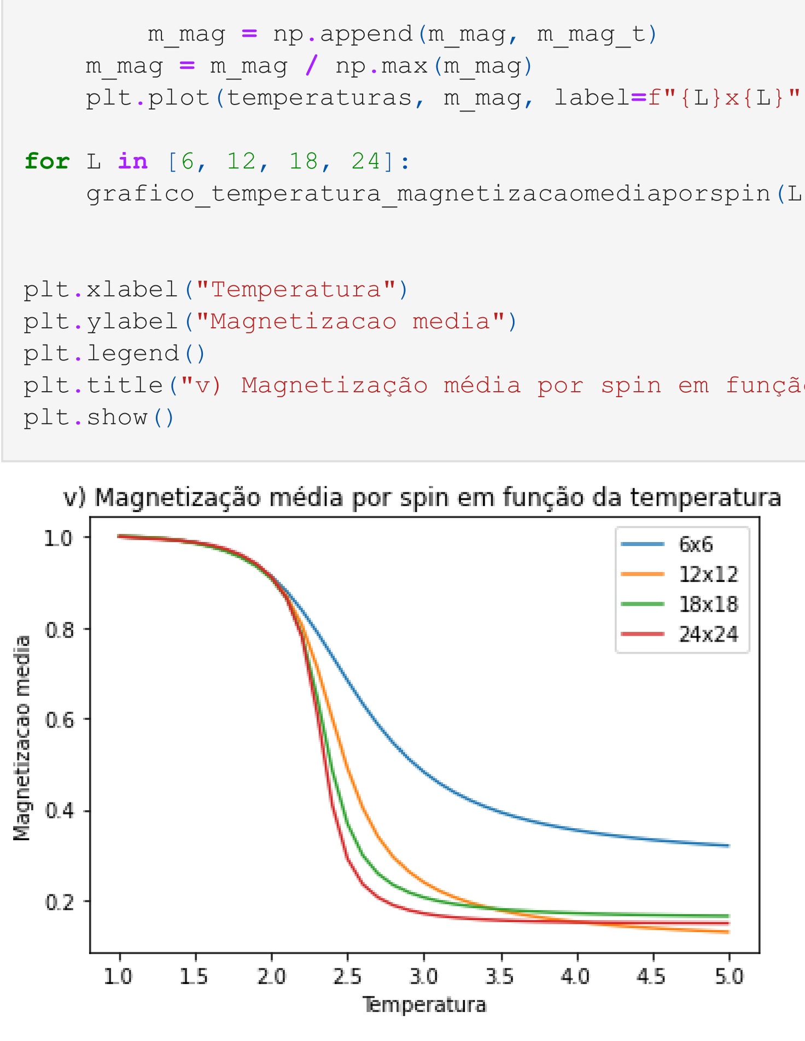
Podemos notar que no gráfico iii a energia livre por spin diminui conforme a temperatura aumenta.

In [19]:
```python
#iv) Entropia por spin
def grafico_temperatura_entropiaporspin(l):
    N = L * L
    s_t = np.array([])

    for t in temperaturas:
        z_t, e_t, cv_t = media_termodinamica(ln_ge[L], t, L)
        f_t = -(1 / (N * (1 / t))) * math.log(z_t)
        s_t = np.append(s_t, (e_t - f_t) / t)

    plt.plot(temperaturas, s_t, label=f"{L}x{L}")

for L in [6, 12, 18, 24]:
    grafico_temperatura_entropiaporspin(L)


plt.xlabel("Temperatura")
plt.ylabel("Entropia por spin")
plt.legend()
plt.title("iv) Entropia por spin em função da temperatura")
plt.show()
```



Similar ao gráfico i, no gráfico iv a energia por spin aumenta em relação a temperatura, possuindo também a diminuição da taxa de crescimento conforme a temperatura aumenta.

In [20]:
```python
#v) Magnetização média por spin, m/N.
def grafico_temperatura_magnetizacaomediaporspin(L):
    N = L * L
    ge = np.exp(ln_ge[L])
    m_mag = np.array([])

    for t in temperaturas:
        z_t, e_t, cv_t = media_termodinamica(ln_ge[L], t, L)
        b = 1 / t
        m_t = 0
        maior = maior_exp(b, N)
        for e_esc in range(len(m_micro[L])):
            if e_esc != 1 and e_esc != len(m_micro[L]) - 2:
                e = energia_reescalada(e_esc, N)
                m_t += m_micro[L][e_esc] * ge[e_esc] * (math.e ** (-b * e - maior))
        m_mag_t = m_t / (z_t * N)

        m_mag = np.append(m_mag, m_mag_t)
    m_mag = m_mag / np.max(m_mag)
    plt.plot(temperaturas, m_mag, label=f"{L}x{L}")

for L in [6, 12, 18, 24]:
    grafico_temperatura_magnetizacaomediaporspin(L)


plt.xlabel("Temperatura")
plt.ylabel("Magnetizacao media")
plt.legend()
plt.title("v) Magnetização média por spin em função da temperatura")
plt.show()
```



O grafico v podemos notar uma queda drastica da magnetização média após a temperatura 2, ficando bem próxima de 0.