MonteCarloParte1

August 14, 2024

1 Modelo de Ising via algoritmo de Metroplis - Atividade de Monte Carlo - Parte 1

Aluno: João Victor Campos Matrícula: 2020035272

[13]:
```
import numpy as np
import matplotlib.pyplot as plt
from numba import jit, typed
from typing import Callable
```

[14]:
```
class SpinLattice:
    def __init__(self, L, T, spins):
        self.L = L # lado da rede
        self.N = self.L * self.L # num de spins
        self.T = T # temperatura
        self.spins = spins # matriz de spins
        self.beta = 1 / self.T

    def get_viz(self) -> np.ndarray:
        #self.N = self.L * self.L
        dimension = (self.N, 4)
        self.viz_list = np.zeros(dimension, dtype=np.int16)

        for i in range(self.N):

            self.viz_list[i,0] = i+1
            if (i+1) % self.L == 0 :
                self.viz_list[i,0] = i+1-self.L

            self.viz_list[i,1] = i+self.L
            if i > (self.N-self.L-1):
                self.viz_list[i,1] = i + self.L -self.N

            self.viz_list[i,2] = i-1
            if i % self.L == 0:
                self.viz_list[i,2] = i-1 + self.L
```

```
            self.viz_list[i,3] = i-self.L
            if i<self.L:
                self.viz_list[i,3] = i-self.L+self.N
        return self.viz_list
```

[15]:
```
def random_config(N):
    # atribuir spins a spin_matrix da Classe
    spins = np.random.choice([-1,1] , N)
    return spins
```

[16]:
```
def get_ener_mag(spins, viz):
    N = spins.size
    ener = 0
    for i in range(N):
        h = spins[viz[i,0]] + spins[viz[i, 1]]
        ener -= spins[i]*h
        mag = np.sum(spins)
    return ener, mag
```

[26]:
```
@jit(nopython=True)
def monte_carlo_step(S: np.ndarray, viz: np.ndarray, ex: typed.Dict, ener:␣
 ↪float, mag: int | float):
    for i in range(S.size):
        # Random site index
        k = np.random.randint(S.size)

        h = S[viz[k,0]] + S[viz[k,1]] + S[viz[k,2]] + S[viz[k,3]]
        delta_E = 2*S[k]*h

        if np.random.rand() < ex[delta_E]:
            S[k] = -S[k]
            ener = ener + delta_E
            mag = np.sum(S)
    return S, ener, mag
```

[27]:
```
def create_exp_dict(beta):
    ex = typed.Dict()
    ex[-8] = np.exp(8.0 * beta)
    ex[-4] = np.exp(4.0 * beta)
    ex[0] = 1.0
    ex[4] = np.exp(-4.0 * beta)
    ex[8] = np.exp(-8.0 * beta)
    return ex
```

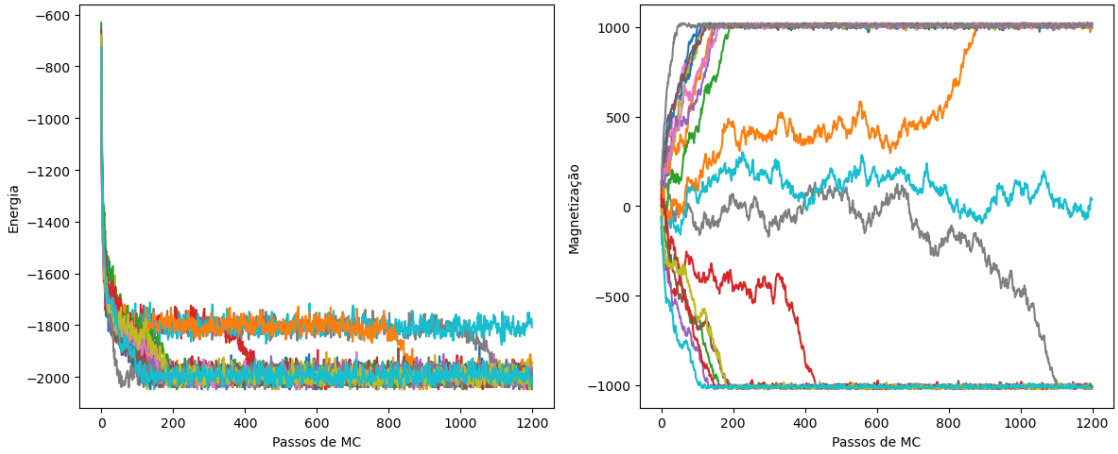
[76]:
```
def make_subplots(energy_vecs: list[list], mag_vecs:list[list]):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,5))
```

```
    for i in range(len(energy_vecs)):
        ax1.plot(energy_vecs[i])
    ax1.set_ylabel("Energia")
    ax1.set_xlabel("Passos de MC")


    for j in range(len(mag_vecs)):
        ax2.plot(mag_vecs[j])
    ax2.set_xlabel("Passos de MC")
    ax2.set_ylabel("Magnetização")
    plt.tight_layout()
```

1.1 Tamanho 32 e Temperatura 1.5

[54]:
```
NUM_CONFIGS = 20
ENERGY_CONTAINER = []
MAG_CONTAINER = []
for j in range(NUM_CONFIGS):
    Rede1 = SpinLattice(L=32, T=1.5, spins=random_config(32*32))
    ex = create_exp_dict(Rede1.beta)

    # Pegar a primeira energia e mag
    ener, mag = get_ener_mag(Rede1.spins, Rede1.get_viz())

    monte_carlo_steps = 1_200

    Energias_Rede1 = []
    Mags_Rede1 = []

    for _ in range(monte_carlo_steps):
        spins, ener, mag = monte_carlo_step(S=Rede1.spins,
                                            viz=Rede1.get_viz(),
                                            ex=ex,
                                            ener=ener,
                                            mag=mag)
        Energias_Rede1.append(ener)
        Mags_Rede1.append(mag)

    ENERGY_CONTAINER.append(Energias_Rede1)
    MAG_CONTAINER.append(Mags_Rede1)
```

[71]:
```
make_subplots(energy_vecs=ENERGY_CONTAINER, mag_vecs=MAG_CONTAINER)
```



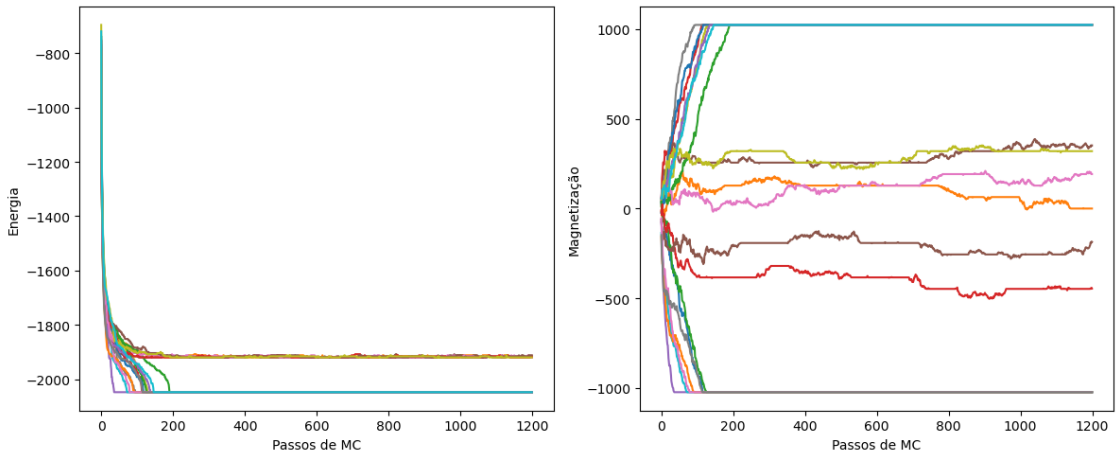
No exemplo acima que utilizou os parâmetros das notas de aula como base, verifica-se comportamento semelhante ao exposto. A energia para a maioria termaliza perto dos 500 passos, mas algumas demoram mais. Ainda, há uma das configurações iniciais que não termalizou com os 1200 passos de monte carlo.

1.2 Tamanho 32 e Temperatura 0.5

[74]:
```
NUM_CONFIGS = 20
ENERGY_CONTAINER = []
MAG_CONTAINER = []
for j in range(NUM_CONFIGS):
    Rede1 = SpinLattice(L=32, T=0.5, spins=random_config(32*32))
    ex = create_exp_dict(Rede1.beta)

    # Pegar a primeira energia e mag
    ener, mag = get_ener_mag(Rede1.spins, Rede1.get_viz())

    monte_carlo_steps = 1_200

    Energias_Rede1 = []
    Mags_Rede1 = []

    for _ in range(monte_carlo_steps):
        spins, ener, mag = monte_carlo_step(S=Rede1.spins,
                                            viz=Rede1.get_viz(),
                                            ex=ex,
                                            ener=ener,
                                            mag=mag)
        Energias_Rede1.append(ener)
        Mags_Rede1.append(mag)
```

```
    ENERGY_CONTAINER.append(Energias_Rede1)
    MAG_CONTAINER.append(Mags_Rede1)
```

[75]:
```
make_subplots(energy_vecs=ENERGY_CONTAINER, mag_vecs=MAG_CONTAINER)
```

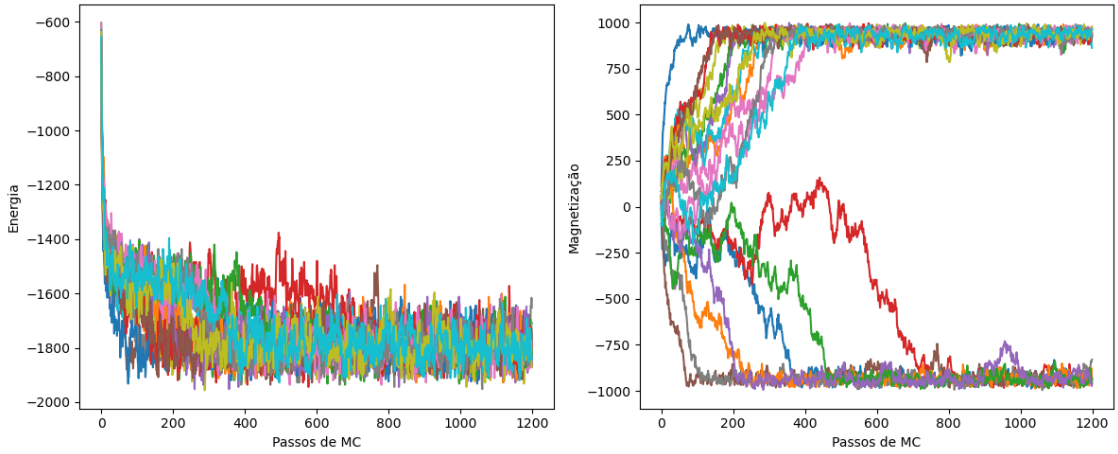


1.3 Tamanho 32 e Temperatura 2.0

[78]:
```
NUM_CONFIGS = 20
ENERGY_CONTAINER = []
MAG_CONTAINER = []
for j in range(NUM_CONFIGS):
    Rede1 = SpinLattice(L=32, T=2.0, spins=random_config(32*32))
    ex = create_exp_dict(Rede1.beta)

    # Pegar a primeira energia e mag
    ener, mag = get_ener_mag(Rede1.spins, Rede1.get_viz())

    monte_carlo_steps = 1_200

    Energias_Rede1 = []
    Mags_Rede1 = []

    for _ in range(monte_carlo_steps):
        spins, ener, mag = monte_carlo_step(S=Rede1.spins,
                                            viz=Rede1.get_viz(),
                                            ex=ex,
                                            ener=ener,
                                            mag=mag)
        Energias_Rede1.append(ener)
        Mags_Rede1.append(mag)
```

```
    ENERGY_CONTAINER.append(Energias_Rede1)
    MAG_CONTAINER.append(Mags_Rede1)
```

[79]:
```
make_subplots(energy_vecs=ENERGY_CONTAINER, mag_vecs=MAG_CONTAINER)
```



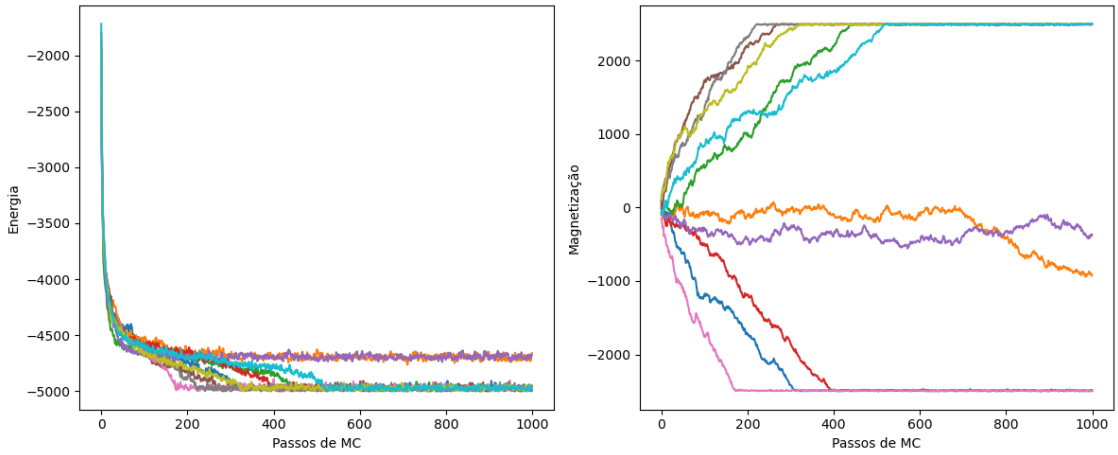
Mantendo o mesmo tamanho mas aumentando a temperatura, a flutuação de energia é **bem maior**.

1.4 Tamanho 50 e Temperatura 1.2

[81]:
```
NUM_CONFIGS = 10
ENERGY_CONTAINER = []
MAG_CONTAINER = []
for j in range(NUM_CONFIGS):
    Rede1 = SpinLattice(L=50, T=1.2, spins=random_config(50*50))
    ex = create_exp_dict(Rede1.beta)

    # Pegar a primeira energia e mag
    ener, mag = get_ener_mag(Rede1.spins, Rede1.get_viz())

    monte_carlo_steps = 1_000

    Energias_Rede1 = []
    Mags_Rede1 = []

    for _ in range(monte_carlo_steps):
        spins, ener, mag = monte_carlo_step(S=Rede1.spins,
                                            viz=Rede1.get_viz(),
                                            ex=ex,
                                            ener=ener,
                                            mag=mag)
        Energias_Rede1.append(ener)
```

```
        Mags_Rede1.append(mag)

    ENERGY_CONTAINER.append(Energias_Rede1)
    MAG_CONTAINER.append(Mags_Rede1)
```

[82]:
```
make_subplots(energy_vecs=ENERGY_CONTAINER, mag_vecs=MAG_CONTAINER)
```



Nessa simulação com o tamanho de 50, vê-se que das 10 configurações, 2 delas (roxo e laranja) não foram capazes de termalizar a magnetização e o mesmo vale para a energia. No geral por volta de 500 passos todas haviam termalizado exceto por essas citadas.

MonteCarloParte2

August 17, 2024

1 Modelo de Ising - Estimativa de Grandezas Termodinâmicas - Parte 2

Aluno: João Victor Campos

Os valores de energia por spin e magnetização por spin são obtidos ao dividir o valor médio calculado para energia e magnetização pelo número de spins da rede.

[2]:
```
from  Ising_Utils import *
from typing import Tuple
```

$$c_v = \frac{\beta^2}{N}\left(\langle E^2\rangle - \langle E\rangle^2\right) \tag{1}$$

$$\chi = \frac{\beta}{N}\left(\langle M^2\rangle - \langle M\rangle^2\right) \tag{2}$$

[3]:
```
@jit(nopython=True)
def rms(Vec):
    N = len(Vec)

    Vec_mean = np.mean(Vec)
    Vec_square_mean = np.mean(Vec*Vec)

    # M_mean = np.mean(Energy_Vec)
    # M_square_mean = np.mean(Mag_vec* Mag_vec)

    root_ms =  (Vec_square_mean - Vec_mean**2)

    return root_ms
```

[4]:
```
# Parâmetero Globais
TEMPERATURAS = [2.0, 2.1 , 2.2, 2.3 , 2.4, 2.5 , 2.6 , 2.7 , 2.8, 2.9, 3.0]
# TEMPERATURAS = [2.0, 2.2, 2.4, 2.6 ,2.8, 3.0]

NUM_CAIXAS = 10
L_REDE = [20, 30, 40, 50]
```

```
# L_REDE = [18]


CV_CONTAINTER = []
CV_ERROS = []

CHI_CONTAINER = []
CHI_ERROS = []

ENERGY_CONTAINER = []
ENERGY_ERROS = []

MAG_CONTAINER = []
MAG_ERROS = []

MONTE_CARLO_STEPS = 110_000
```

[5]:
```
for L in L_REDE:
    print("-" * 100)
    print(f"Tamahno da rede = {L}")
    for T in TEMPERATURAS:
        print(f"\tTemperatura = {T} ")
        Rede = SpinLattice( L=L, T=T, spins=random_config(L*L) )
        ex = create_exp_dict(Rede.beta)

        ener, mag = get_ener_mag(spins= Rede.spins, viz=Rede.get_viz())

        energias = np.empty(MONTE_CARLO_STEPS)
        mags = np.empty(MONTE_CARLO_STEPS)

        for i in range(MONTE_CARLO_STEPS):
            s, ener, mag = monte_carlo_step(S=Rede.spins, viz=Rede.get_viz(),
                                            ex=ex,ener=ener, mag=mag)
            energias[i] = ener
            mags[i] = mag       # Slice dentro do mesmo tamanho L
        energias = energias[10_000:]
        mags = mags[10_000:]

        # Armazenar os valores de cada caixa
        cv_is = np.empty(NUM_CAIXAS)
        chi_is = np.empty(NUM_CAIXAS)

        mag_is = np.empty(NUM_CAIXAS)
        E_is = np.empty(NUM_CAIXAS)

        m = len(energias) / NUM_CAIXAS
```

```
        # Metodo das Caixas
        for k in range(NUM_CAIXAS):

            E_i = energias[ int(m * (k)) : int( m * (k+1) )]
            mag_i = mags[ int(m * (k)) : int( m * (k+1) ) ]


            # adiciona aos array
            cv_is[k] = (Rede.beta**2 / m) * rms(E_i)
            chi_is[k] = (Rede.beta / m) * rms(mag_i)
            E_is[k] = np.sum(E_i) / m
            mag_is[k] = abs( sum( mags[int(m * (k)): int(m * (k+1))] ) ) / m

        # A adicionar ao CONTAINER
        CV_medio, CV_erro = np.mean(cv_is), np.std(cv_is)
        CHI_medio, CHI_erro = np.mean(chi_is), np.std(chi_is)
        E_medio, E_erro = np.mean(E_is) , np.std(E_is)
        MAG_medio, MAG_erro = np.mean(mag_is) , np.std(mag_is)

        CV_CONTAINTER.append(CV_medio)
        CV_ERROS.append(CV_erro)

        CHI_CONTAINER.append(CHI_medio)
        CHI_ERROS.append(CHI_erro)

        ENERGY_CONTAINER.append(E_medio)
        ENERGY_ERROS.append(E_erro)

        MAG_CONTAINER.append(MAG_medio)
        MAG_ERROS.append(MAG_erro)

print("RODOU")
```

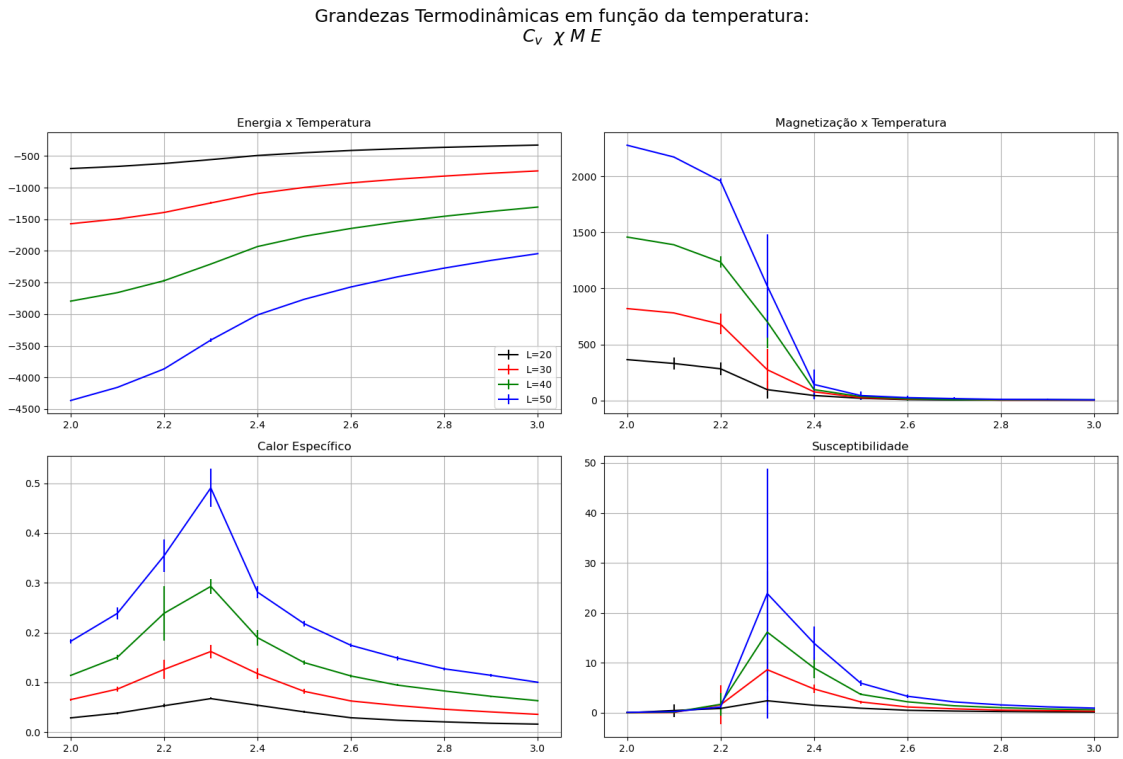
```
--------------------------------------------------------------------------------
--------------------
Tamahno da rede = 20
        Temperatura = 2.0
        Temperatura = 2.1
        Temperatura = 2.2
        Temperatura = 2.3
        Temperatura = 2.4
        Temperatura = 2.5
        Temperatura = 2.6
        Temperatura = 2.7
        Temperatura = 2.8
```

```
        Temperatura = 2.9
        Temperatura = 3.0
--------------------------------------------------------------------------------
--------------------
Tamahno da rede = 30
        Temperatura = 2.0
        Temperatura = 2.1
        Temperatura = 2.2
        Temperatura = 2.3
        Temperatura = 2.4
        Temperatura = 2.5
        Temperatura = 2.6
        Temperatura = 2.7
        Temperatura = 2.8
        Temperatura = 2.9
        Temperatura = 3.0
--------------------------------------------------------------------------------
--------------------
Tamahno da rede = 40
        Temperatura = 2.0
        Temperatura = 2.1
        Temperatura = 2.2
        Temperatura = 2.3
        Temperatura = 2.4
        Temperatura = 2.5
        Temperatura = 2.6
        Temperatura = 2.7
        Temperatura = 2.8
        Temperatura = 2.9
        Temperatura = 3.0
--------------------------------------------------------------------------------
--------------------
Tamahno da rede = 50
        Temperatura = 2.0
        Temperatura = 2.1
        Temperatura = 2.2
        Temperatura = 2.3
        Temperatura = 2.4
        Temperatura = 2.5
        Temperatura = 2.6
        Temperatura = 2.7
        Temperatura = 2.8
        Temperatura = 2.9
        Temperatura = 3.0
RODOU
```

[6]:
```
num_redes = len(L_REDE)

# Calor Especifico
Cv_rede = np.array_split(CV_CONTAINTER, num_redes)
Cv_erro = np.array_split(CV_ERROS, num_redes)
# Susceptibilidade
Chi_rede = np.array_split(CHI_CONTAINER, num_redes)
Chi_erro = np.array_split(CHI_ERROS, num_redes)
# Magnetizacao
Mag_rede = np.array_split(MAG_CONTAINER, num_redes)
Mag_erro = np.array_split(MAG_ERROS, num_redes)
# Energia por spin
Energia_rede = np.array_split(ENERGY_CONTAINER, num_redes)
Energia_erro = np.array_split(ENERGY_ERROS, num_redes)
```

[7]:
```
fig, (ax_up, ax_down) = plt.subplots(2, 2, figsize=(16,10))
fig.suptitle('Grandezas Termodinâmicas em função da temperatura:\n' + r"$C_v$ ␣
 ↪$\chi$ $M$ $E$", y=1.07, size=18);

c = ["black", "red", "green", "blue"]

ax_up[0].set_title("Energia x Temperatura") ; ax_up[0].grid()
ax_up[1].set_title("Magnetização x Temperatura") ;  ax_up[1].grid()
ax_down[0].set_title("Calor Específico") ; ax_down[0].grid()
ax_down[1].set_title("Susceptibilidade") ; ax_down[1].grid()

# Plot energias
for i in range(num_redes):
    ax_up[0].errorbar(TEMPERATURAS, Energia_rede[i], Energia_erro[i],␣
 ↪color=c[i], label=f"L={L_REDE[i]}")
    ax_up[0].legend()

# Plot Mag
for i in range(num_redes):
    ax_up[1].errorbar(TEMPERATURAS, Mag_rede[i], Mag_erro[i], color=c[i],␣
 ↪label=f"L={L_REDE[i]}")

# Plot Calor Especifico
for i in range(num_redes):
    ax_down[0].errorbar(TEMPERATURAS, Cv_rede[i], Cv_erro[i], color=c[i],␣
 ↪label=f"L={L_REDE[i]}")

# Plot Susceptibilidade
for i in range(num_redes):
    ax_down[1].errorbar(TEMPERATURAS, Chi_rede[i], Chi_erro[i], color=c[i],␣
 ↪label=f"L={L_REDE[i]}")
```

```
fig.tight_layout()
```



1.1 Respostas às perguntas da atividade

1.1.1 1 - Quais critérios você utilizou para escolher os valores dos parâmetros descritos acima?

Resposta:
Já havia feito uma atividade semelhante, então sabia quais tamanhos de rede seriam interessantes trabalhar, bem como a faixa de temperatura.

1.1.2 2 - Descreva o comportamento observado para as principais grandezas termodinâmicas – Energia por spin, Magnetização por spin, calor específico e susceptibilidade magnética – em função da temperatura. Ou seja, ao variar a temperatura, o que acontece com o valor destas grandezas? Quais são os limites para baixas e altas temperaturas? Há algum pico ou vale? O comportamento está em acordo com o que você esperava?

Resposta:
A energia por spin média é maior (menos negativa) para tamanhos de redes maiores. A magnetização por spin vai a 0 para temperaturas muito altas, isso deve significar a perda do caráter

ferromagnético após o ponto de Curie. O calor específico para rede menores é também menor, aumenta até determinada temperatura, satura, e depois diminui. O comportamento da susceptibilidade é semelhante.

1.1.3 3 - Ao variar o tamanho do sistema, como as curvas destas grandezas em função da temperatura se modifica? Há algum intervalo de temperaturas no qual as grandezas são independentes do tamanho do sistema? Em regiões onde há variação com o tamanho do sistema, como a grandeza é modificada quando L aumenta?

Resposta: A partir do $T > 2.6$, $C_v, \chi, \mathcal{M}$ são independentes do tamanho do sistema e vão um valor constante (ou nulo).

1.1.4 4 - Como é o comportamento dos erros estatísticos à medida que a temperatura varia? Tem algum valor de temperatura em torno do qual os erros são maiores? Você enxerga algum motivo para isso? Os erros estatísticos dependem do tamanho do sistema? Como?

Resposta:
O erro estatístico é tanto maior quanto maior o sistema, bem como quanto mais próximo da temperatura de transição de fase que ocorre por volta de $T = 2.3$.

1.1.5 5 - Com base no comportamento encontrado, identifique possíveis fases do sistema, descrevendo as principais características das fases encontradas.

Resposta: Muito provavelmente uma fase ferromagnética com um valor de magnetização não nulo e um fase paramagnética após a transição de fase.

1.1.6 6 - Estime, utilizando os dados das suas simulações, a temperatura de transição de fase do sistema no limite termodinâmico, i.e., para o limite em que o tamanho do sistema é infinito.

Resposta:

Com o tamanho do sistema indo para infinito, a curva estreitaria em valores próximos a $T = 2.3$ onde haveria uma descontinuidade das grandezas termodinâmicas.