Simulação do Modelo de Ising

Nome: Arthur Pontes Nader

Matrícula: 2019022294

Bibliotecas

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

Funções

In [ ]:
```
def gerarRede(L):

    N = L*L
    rede = [np.random.choice([1,-1]) for i in range(N)]

    return np.array(rede)
```

In [ ]:
```
def gerarVizinhos(L):

  N = L*L
  vizinhos = np.zeros((N, 4), dtype = np.int32)

  for k in range(N):

      vizinhos[k][0] = k + 1
      if((k+1)%L) == 0:
          vizinhos[k][0] = k + 1 - L

      vizinhos[k][1] = k + L
      if(k > N - 1 - L):
          vizinhos[k][1] = k + L - N
      vizinhos[k][2] = k - 1

      if(k%L == 0):
          vizinhos[k][2] = k + L - 1

      vizinhos[k][3] = k - L
      if(k < L):
          vizinhos[k][3] = k + N - L

  return vizinhos
```

In [ ]:
```
def calcularEnergia(s, viz):

    energia = 0
    for i in range(len(s)):
        h = s[viz[i][0]] + s[viz[i][1]]
        energia = energia - s[i]*h

    return energia
```

In [ ]:
```
@jit(nopython=True)
```

```
def expos(beta):

    ex = np.zeros(5,dtype=np.float32)
    ex[0]=np.exp(8.0*beta)
    ex[1]=np.exp(4.0*beta)
    ex[2]=1.0
    ex[3]=np.exp(-4.0*beta)
    ex[4]=np.exp(-8.0*beta)

    return ex
```

In [ ]:
```
@jit(nopython=True)
def mcstep(beta,s,viz,ener,mag):

    N=len(s)
    ex=expos(beta)

    for i in range(N):
        h = s[viz[i,0]]+s[viz[i,1]]+s[viz[i,2]]+s[viz[i,3]] # soma dos vizinhos
        de = int(s[i]*h*0.5+2)
        if np.random.random() < ex[de]:
            ener=ener+2*s[i]*h
            mag -= 2*s[i]
            s[i]=-s[i]

    return ener,mag,s
```

In [ ]:
```
def Metropolis(L, configuracao_atual, temperatura,  iteracoes = 1100000):

    vizinhos = gerarVizinhos(L)

    beta = 1/temperatura
    energia = calcularEnergia(configuracao_atual, vizinhos)
    magnetizacao = np.sum(configuracao_atual)

    energias, magnetizacoes = [], []

    for i in range(iteracoes):

      energia, magnetizacao, configuracao_atual = mcstep(beta, configuracao_atual, vizin
      energias.append(energia)
      magnetizacoes.append(magnetizacao)

    return np.array(energias[100000:]), np.array(magnetizacoes[100000:]), configuracao_a
```

Funções auxiliares para simulação

In [ ]:
```
def calcularErro(dados, media, num_caixas = 10):

  return np.sqrt(np.sum(np.square(np.array(dados) - media)) / (num_caixas*(num_caixas-1)
```

In [ ]:
```
def calcularEstatisticas(energias, magnetizacoes, temperatura, L, num_caixas = 10):

  beta = 1/temperatura

  num_spins = L**2

  calores_especificos, energias_por_spin = [], []
  magnetizacoes_por_spin, suscet_magnetica = [], []

  for caixa in energias:
```

```
        energias_por_spin.append(np.mean(caixa)/num_spins)
        calor_espec = (beta**2/num_spins)*(np.mean(np.square(caixa)) - np.mean(caixa)**2)
        calores_especificos.append(calor_espec)

    for caixa in magnetizacoes:

        magnetizacoes_por_spin.append((np.mean(np.abs(caixa))/num_spins))

        susc = (beta/num_spins)*(np.mean(np.square(caixa)) - np.mean(caixa)**2)
        suscet_magnetica.append(susc)

    media_calor = np.mean(calores_especificos)
    erro_calor = calcularErro(calores_especificos, media_calor)

    media_energia = np.mean(energias_por_spin)
    erro_energia = calcularErro(energias_por_spin, media_energia)

    media_suscet = np.mean(suscet_magnetica)
    erro_suscet = calcularErro(suscet_magnetica, media_suscet)

    media_mag = np.mean(magnetizacoes_por_spin)
    erro_mag = calcularErro(magnetizacoes_por_spin, media_mag)


    return (media_calor, erro_calor), (media_energia, erro_energia), (media_suscet, erro_s
```

In [ ]:
```
temperaturas = np.linspace(3, 2, 11)
```

In [ ]:
```
def executarSimulacao(L):

    configuracao_atual = gerarRede(L)

    medias_energia, erros_ene = [], []
    medias_calor, erros_cal = [], []
    medias_magnet, erros_mag = [], []
    medias_suscept, erros_sus = [], []

    for temp in temperaturas:

        ener, mag, configuracao_atual = Metropolis(L, configuracao_atual, temp)
        c, e, s, m = calcularEstatisticas(np.split(ener, 10), np.split(mag , 10), temp, L)
        medias_energia.append(e[0])
        erros_ene.append(e[1])
        medias_calor.append(c[0])
        erros_cal.append(c[1])
        medias_magnet.append(m[0])
        erros_mag.append(m[1])
        medias_suscept.append(s[0])
        erros_sus.append(s[1])


    return (medias_energia, erros_ene), (medias_calor, erros_cal), (medias_magnet, erros_m
```

Função para plotar os resultados obtidos

In [ ]:
```
def gerarGrafico(valores_medios, erros, metrica, L):

    temperaturas = np.linspace(3, 2, 11)

    plt.figure(figsize=(16, 8))
    plt.errorbar(temperaturas, valores_medios[::-1], erros, fmt='ks', label='Media e err
    plt.legend()
    plt.grid()
```

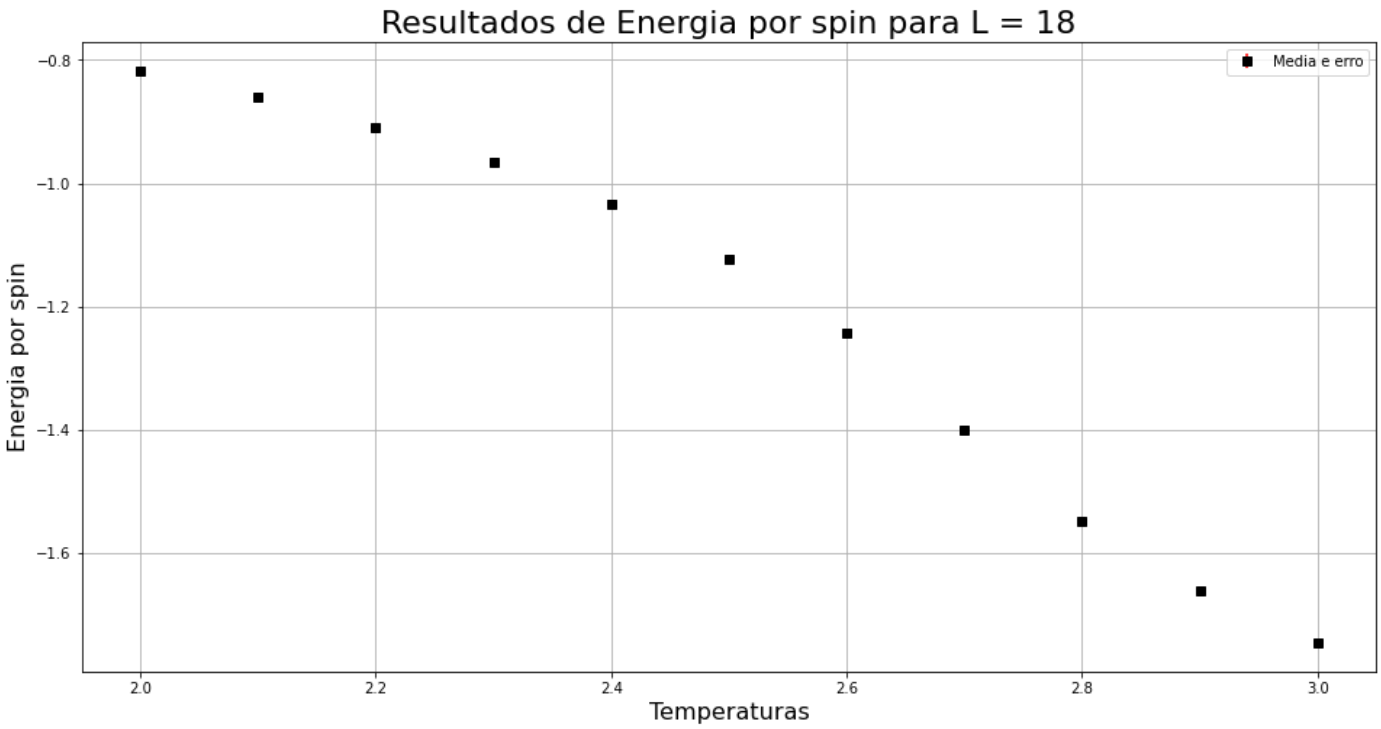
```
    plt.ylabel(metrica, fontsize = 16)
    plt.xlabel("Temperaturas", fontsize = 16)
    plt.title("Resultados de " + metrica + " para L = " + str(L), fontsize = 22)
    plt.show()
```

Execuções variando o valor de L

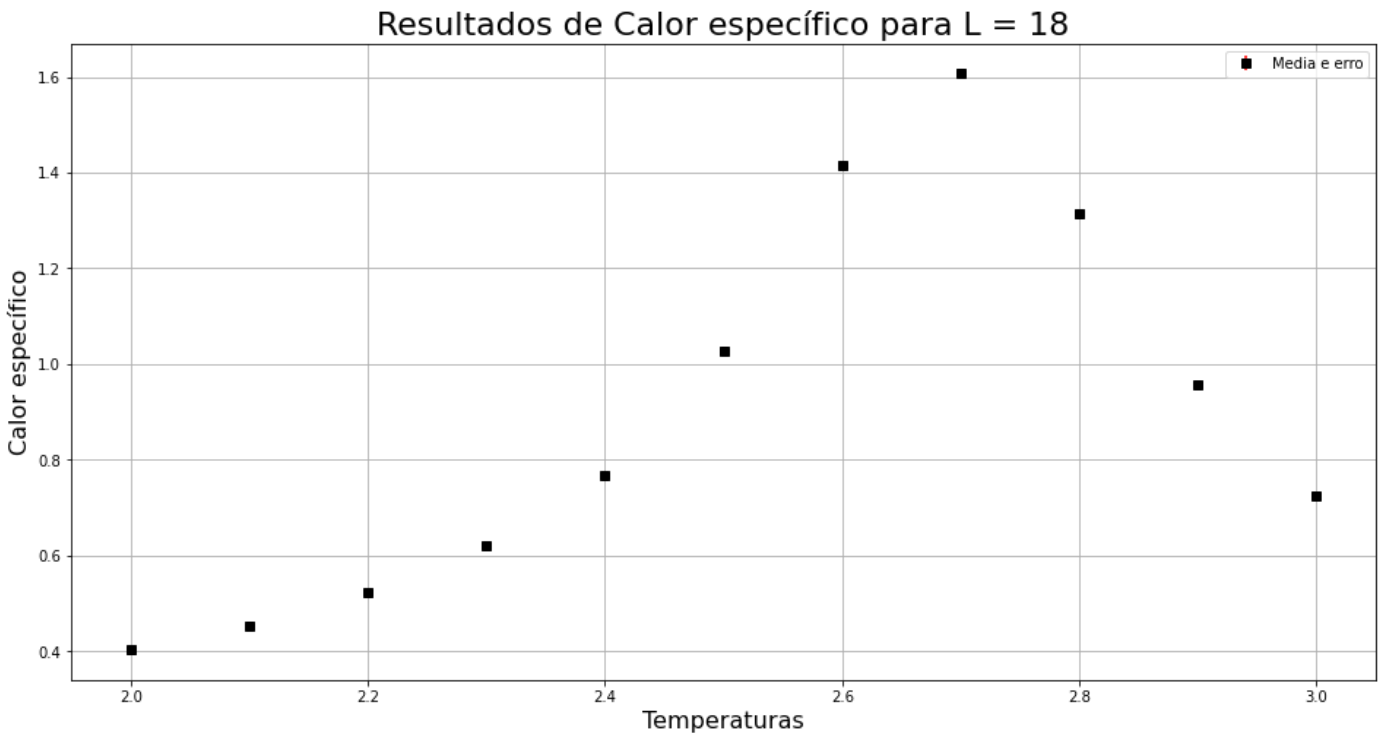
L = 18

In [ ]:
```
L = 18
energia, calor, magnetizacao, susceptibilidade =  executarSimulacao(L)
```

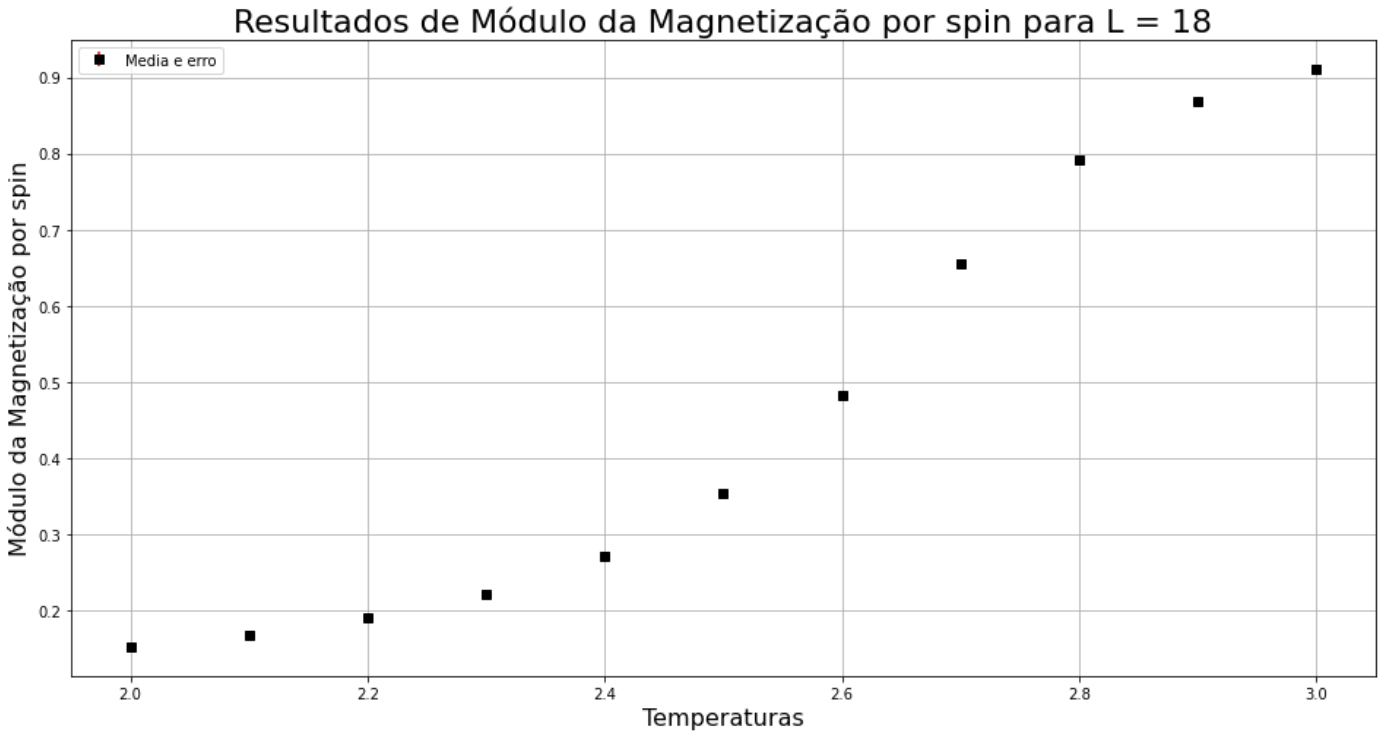
In [ ]:
```
gerarGrafico(energia[0], energia[1], "Energia por spin", L)
```



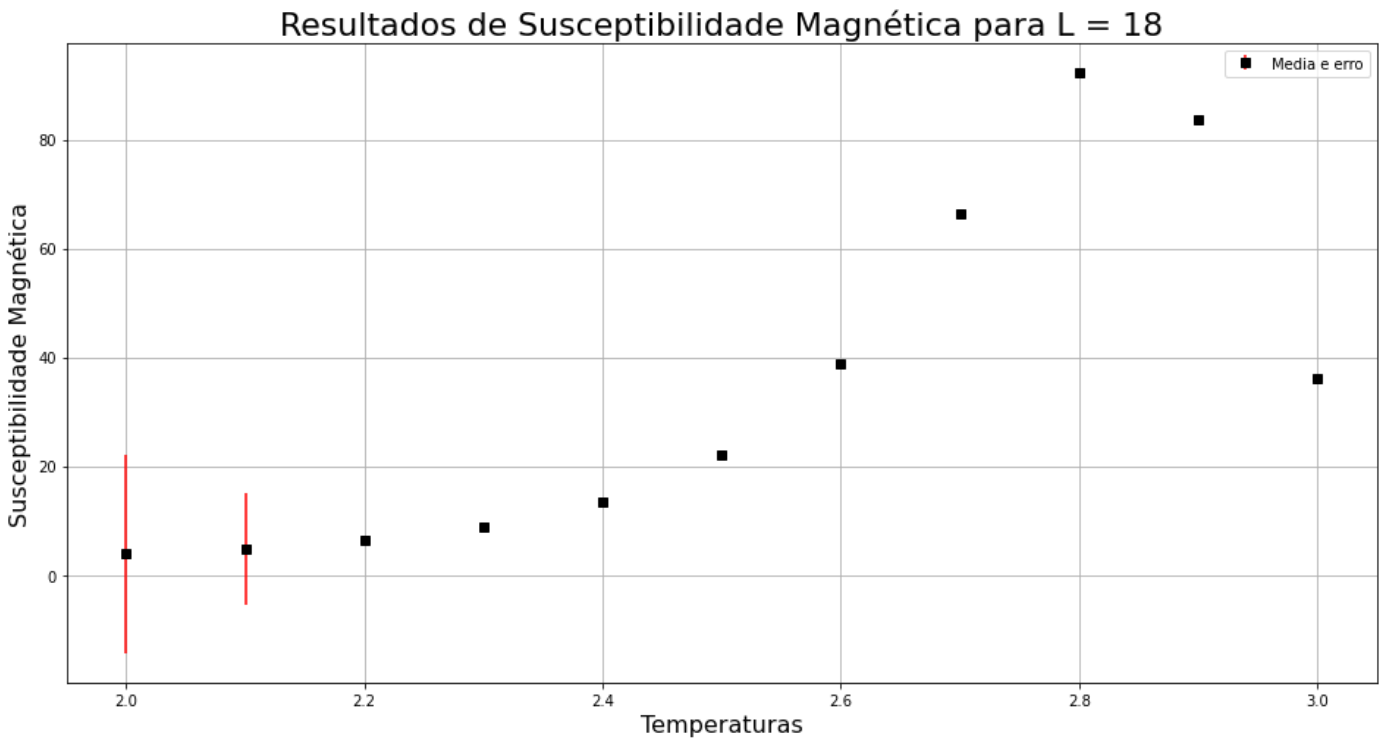
In [ ]:
```
gerarGrafico(calor[0], calor[1], "Calor específico", L)
```



In [ ]:
```
gerarGrafico(magnetizacao[0], magnetizacao[1], "Módulo da Magnetização por spin", L)
```



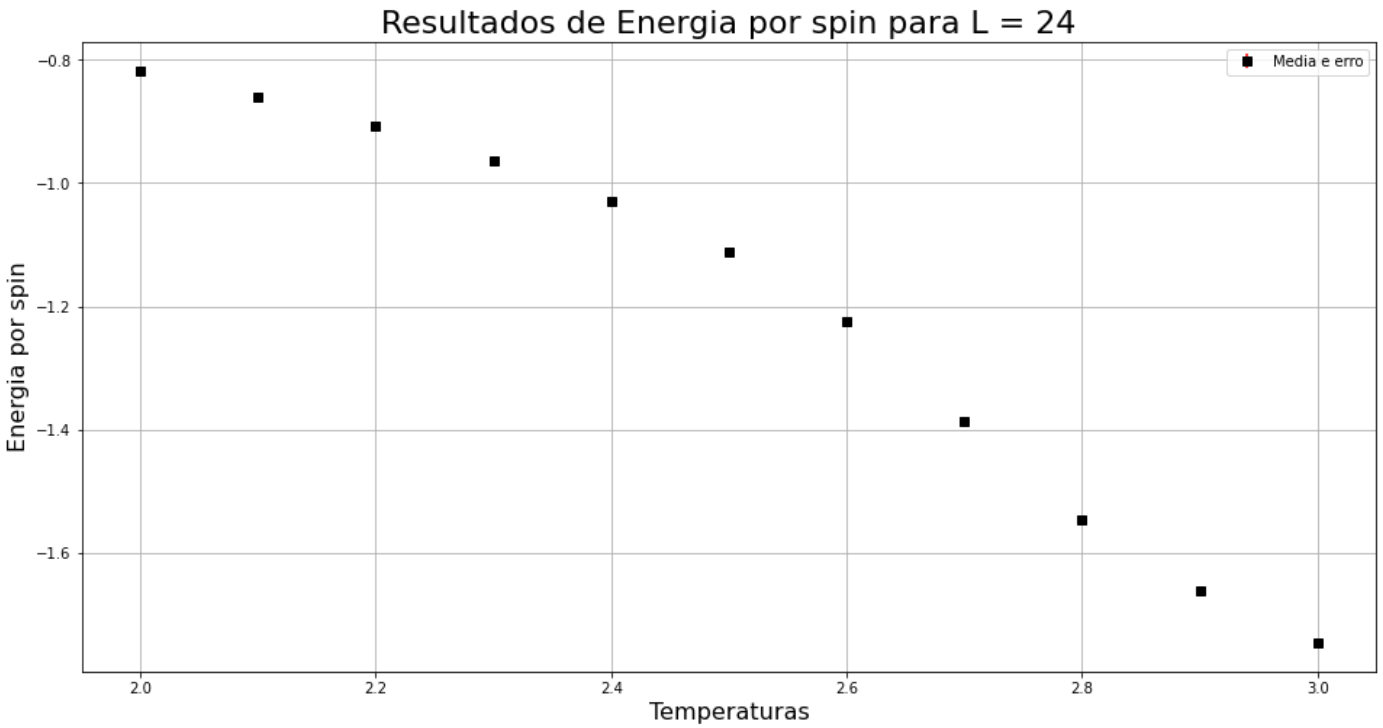
In [ ]:
```
gerarGrafico(susceptibilidade[0], susceptibilidade[1], "Susceptibilidade Magnética", L)
```



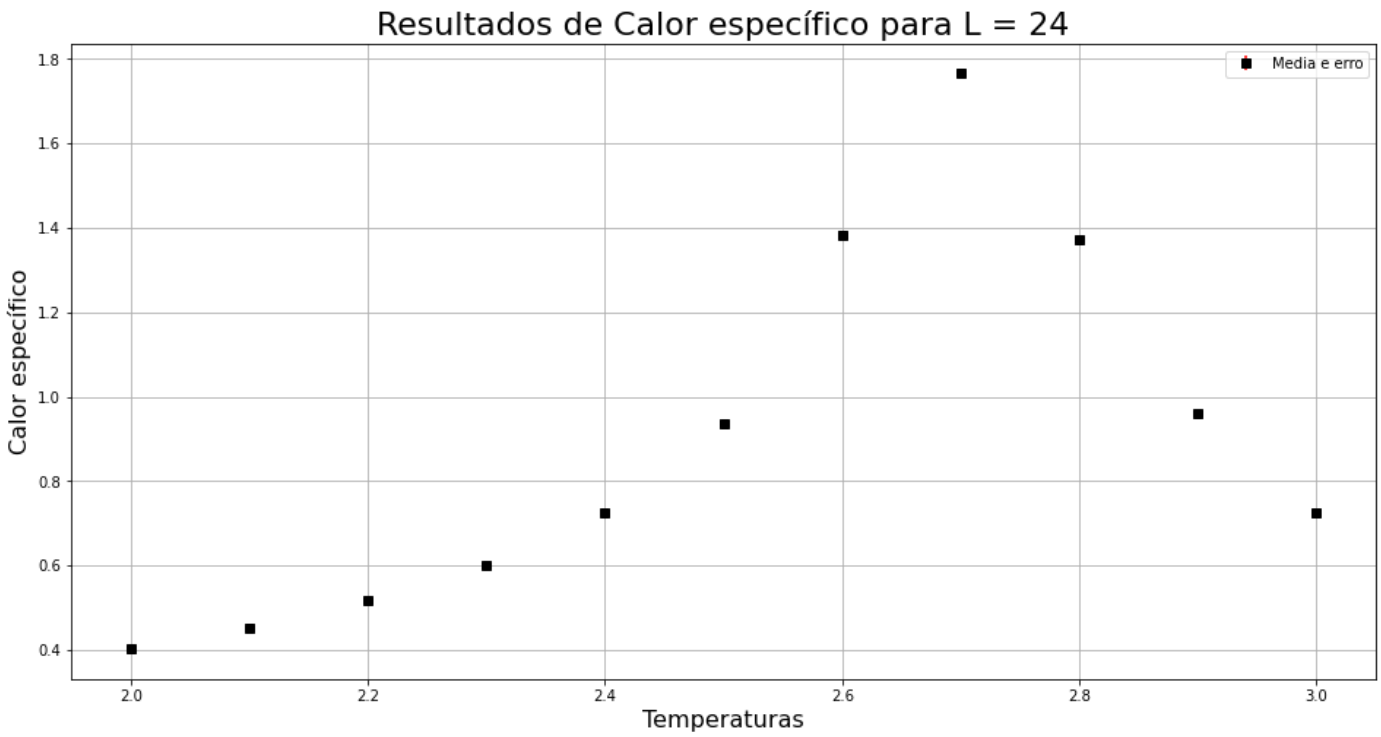
L = 24

In [ ]:
```
L = 24
energia, calor, magnetizacao, susceptibilidade =  executarSimulacao(L)
```

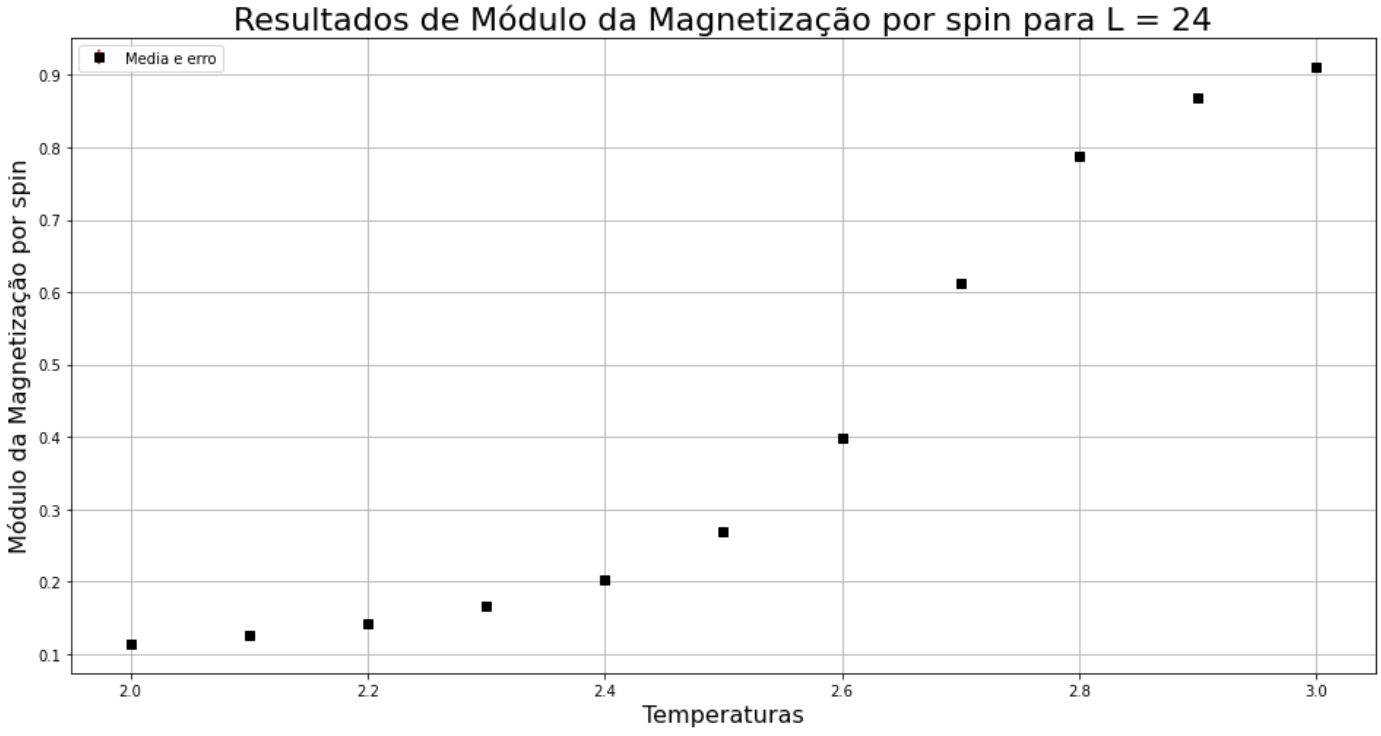
In [ ]:
```
gerarGrafico(energia[0], energia[1], "Energia por spin", L)
```



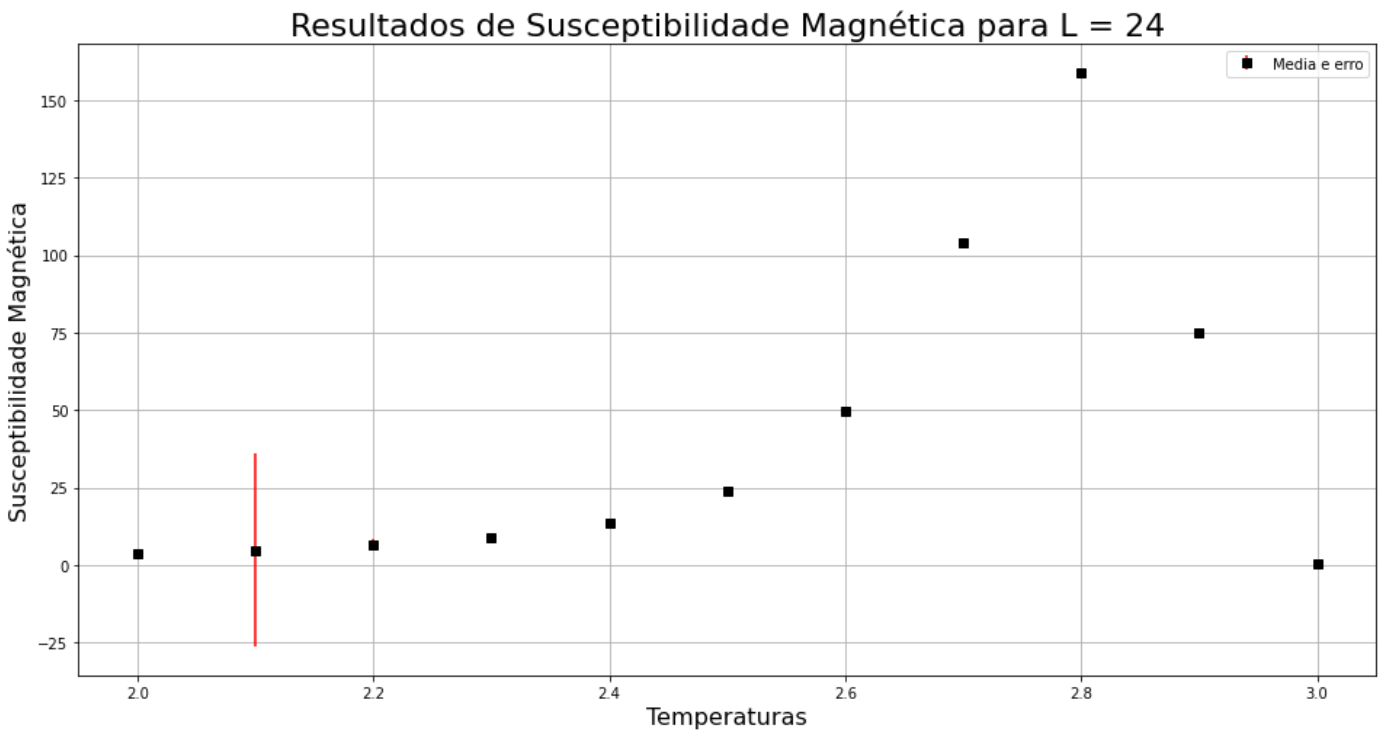
In [ ]:
```
gerarGrafico(calor[0], calor[1], "Calor específico", L)
```



In [ ]:
```
gerarGrafico(magnetizacao[0], magnetizacao[1], "Módulo da Magnetização por spin", L)
```



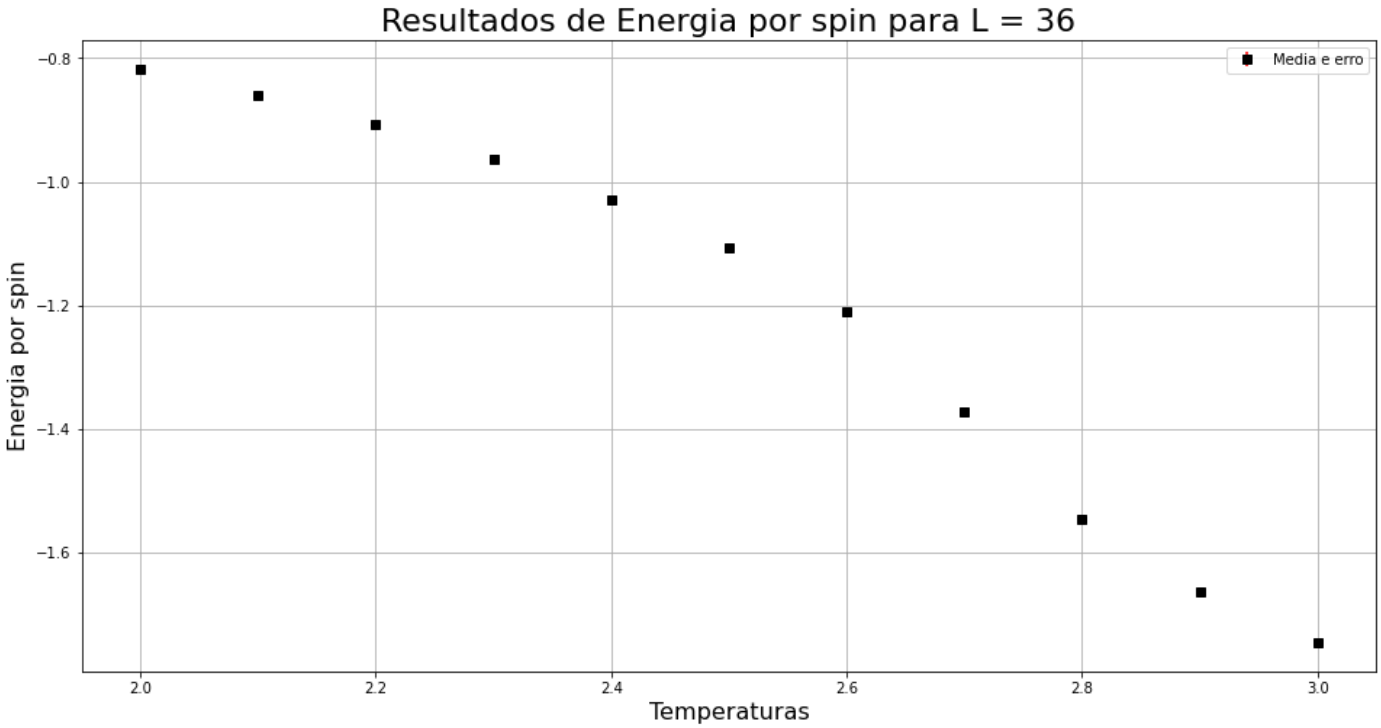
In [ ]:
```
gerarGrafico(susceptibilidade[0], susceptibilidade[1], "Susceptibilidade Magnética", L)
```



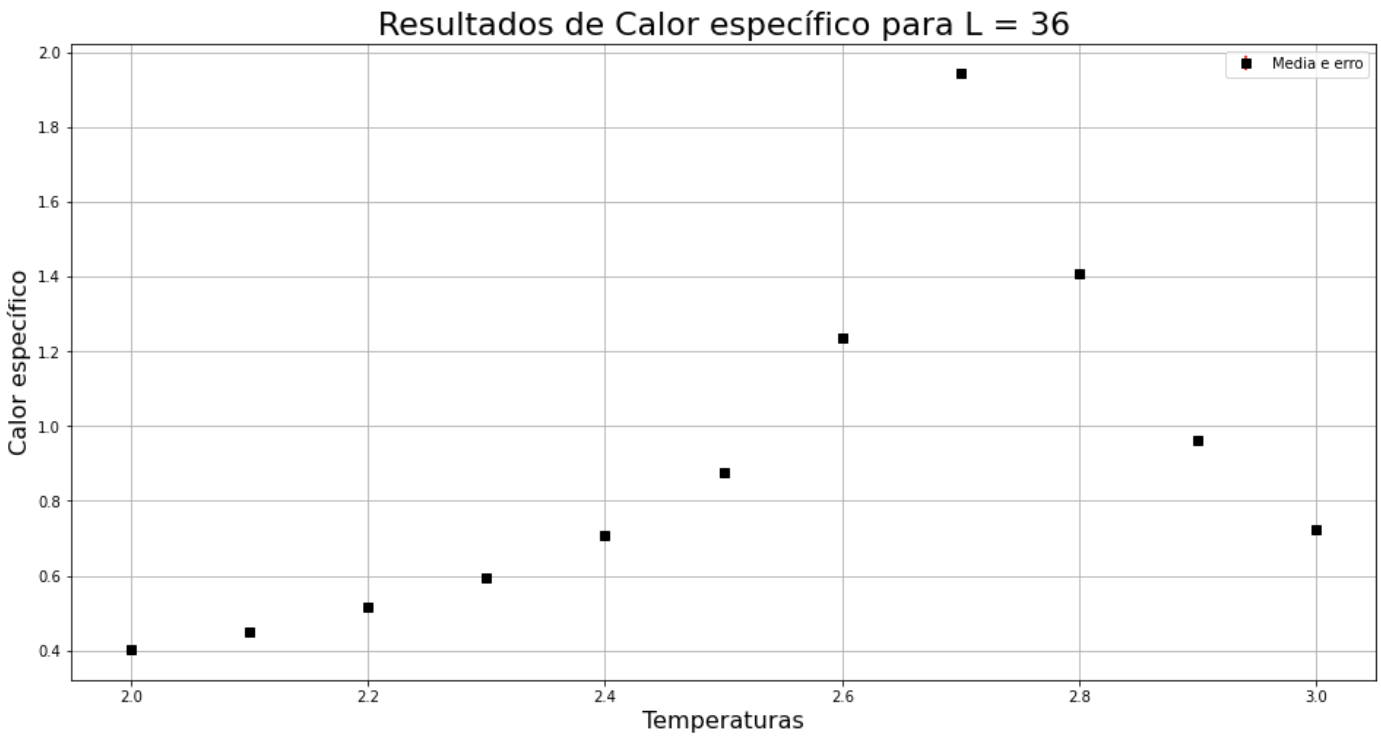
L = 36

In [ ]:
```
L = 36
energia, calor, magnetizacao, susceptibilidade =  executarSimulacao(L)
```

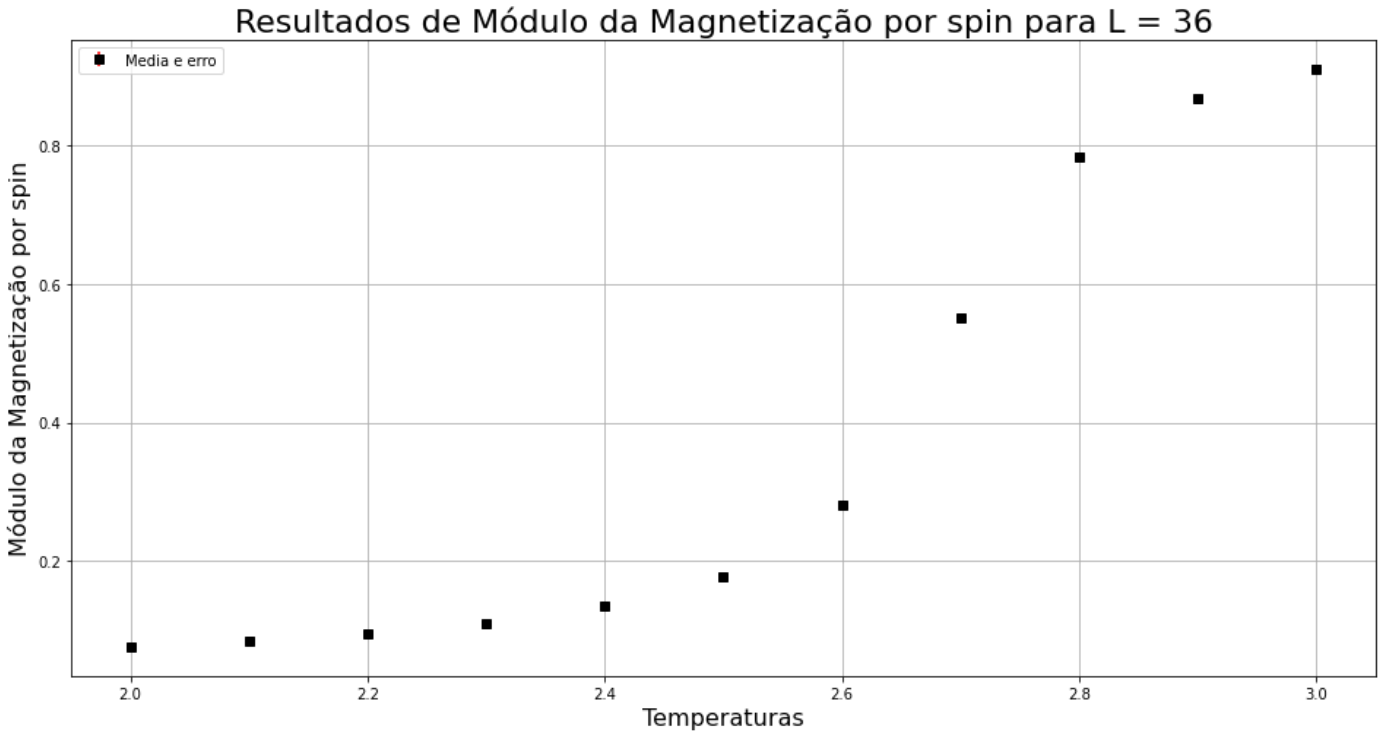
In [ ]:
```
gerarGrafico(energia[0], energia[1], "Energia por spin", L)
```



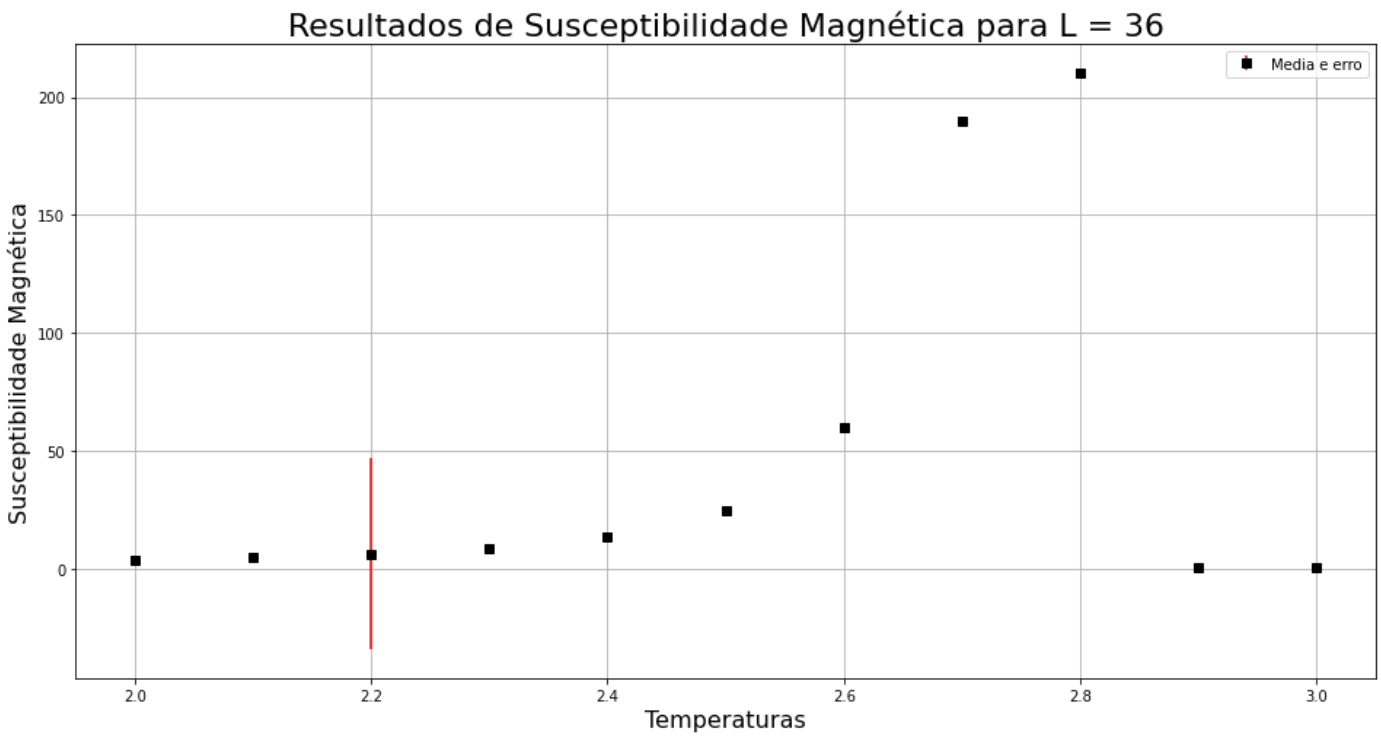
In [ ]:
```
gerarGrafico(calor[0], calor[1], "Calor específico", L)
```



In [ ]:
```
gerarGrafico(magnetizacao[0], magnetizacao[1], "Módulo da Magnetização por spin", L)
```



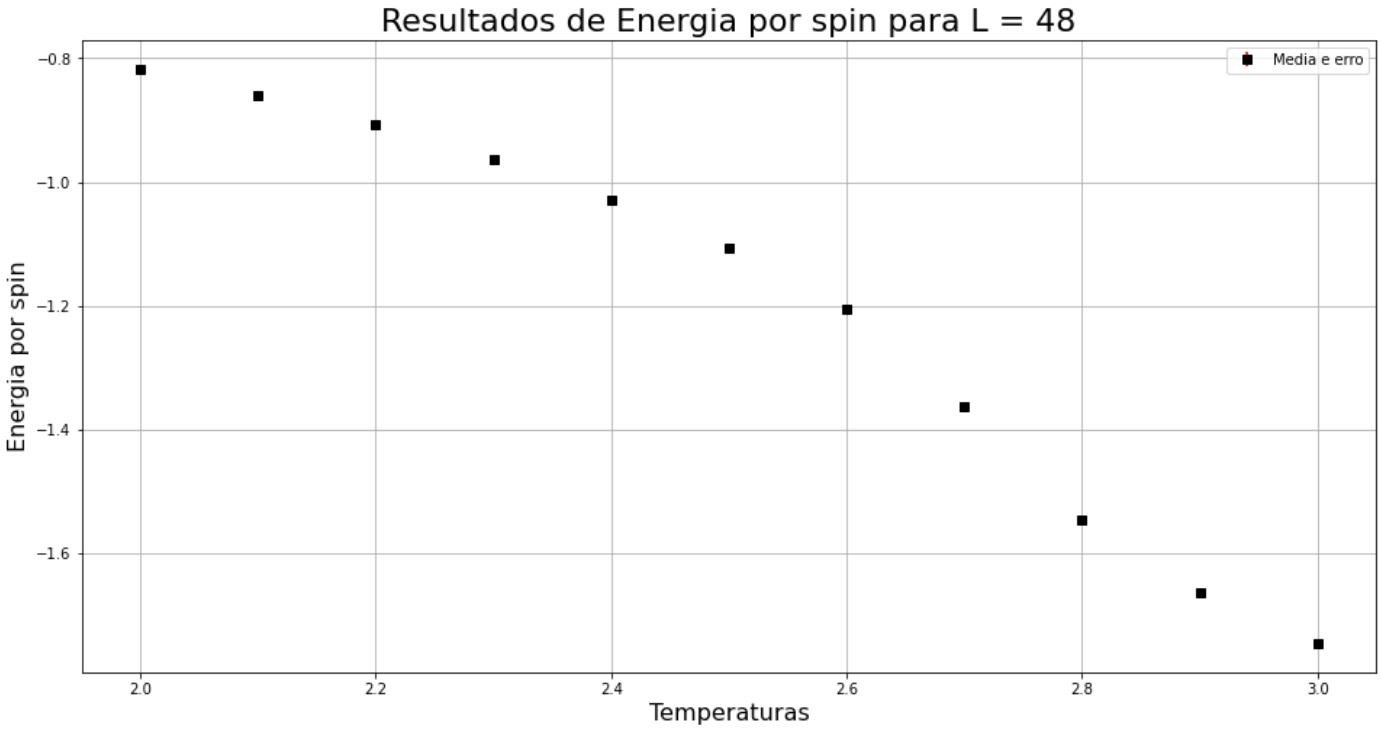
In [ ]:
```
gerarGrafico(susceptibilidade[0], susceptibilidade[1], "Susceptibilidade Magnética", L)
```



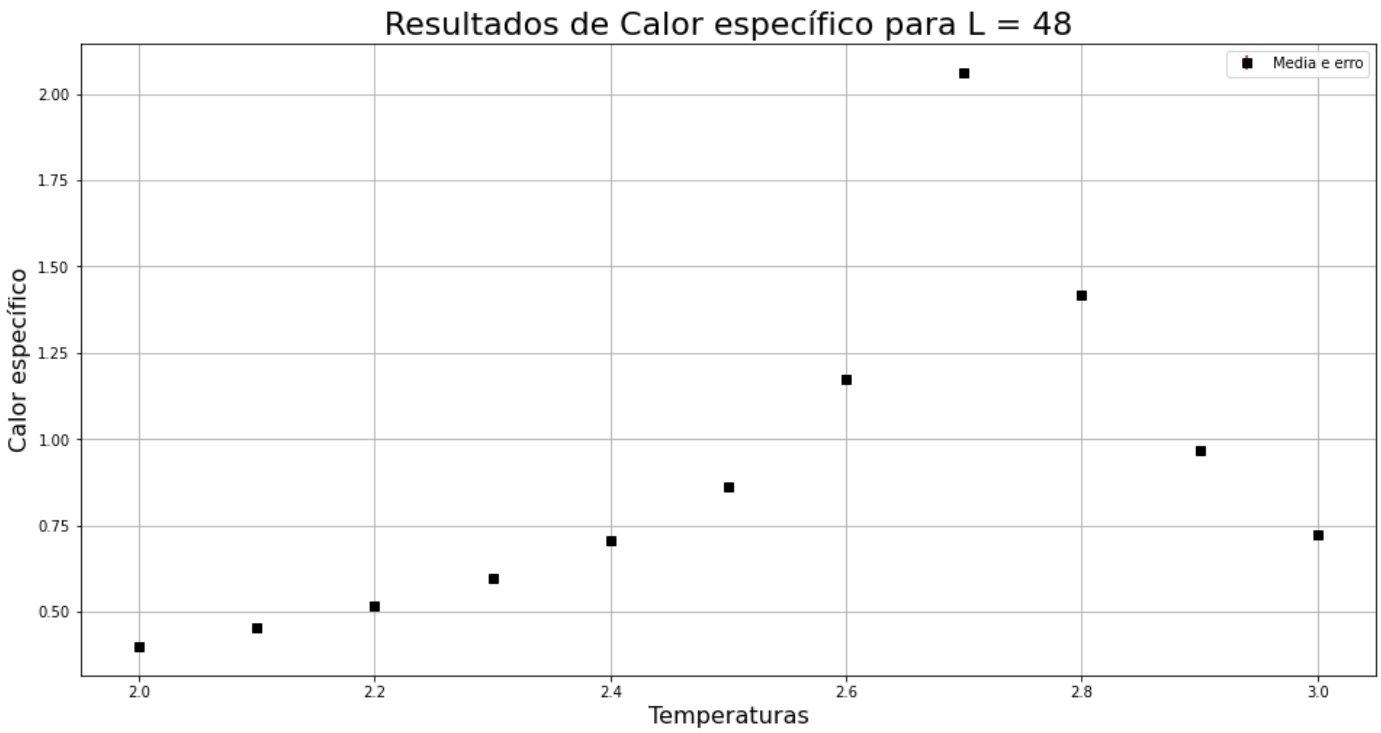
L = 48

In [ ]:
```
L = 48
energia, calor, magnetizacao, susceptibilidade =  executarSimulacao(L)
```

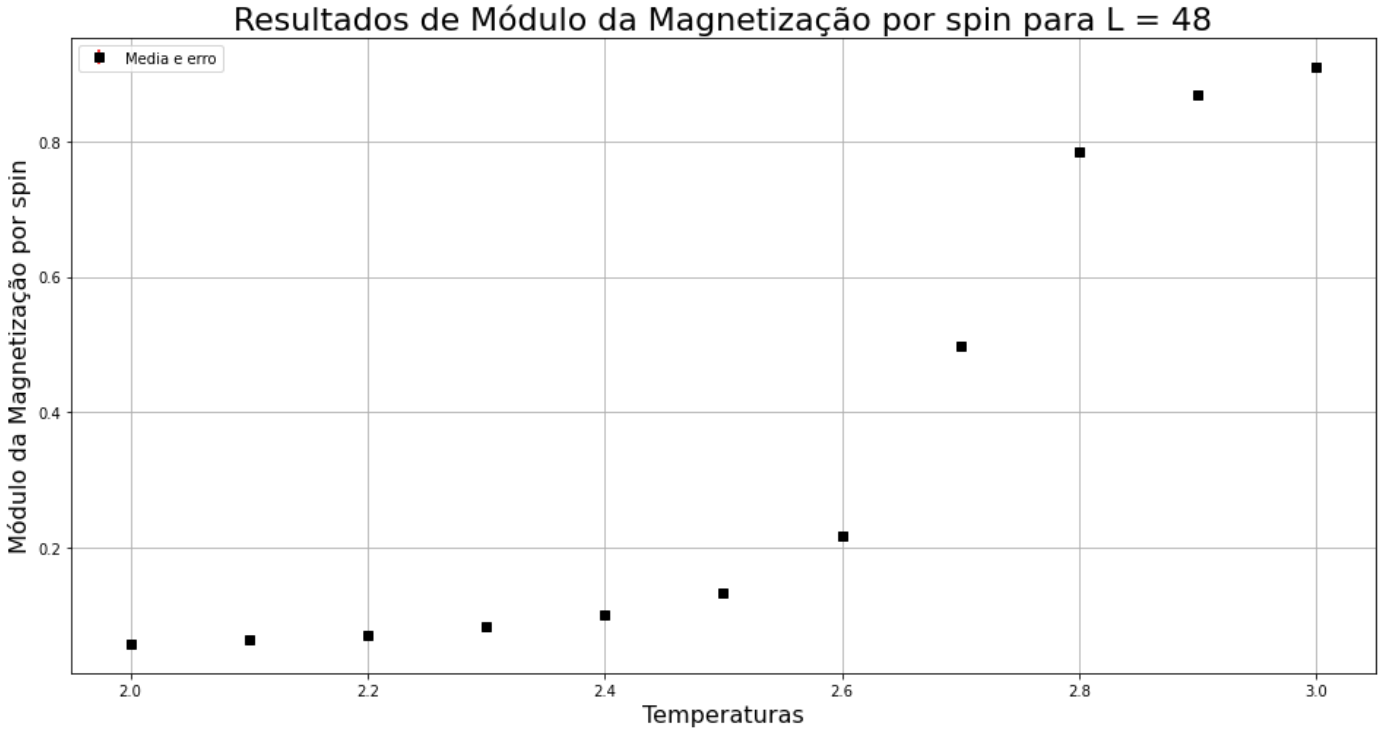
In [ ]:
```
gerarGrafico(energia[0], energia[1], "Energia por spin", L)
```



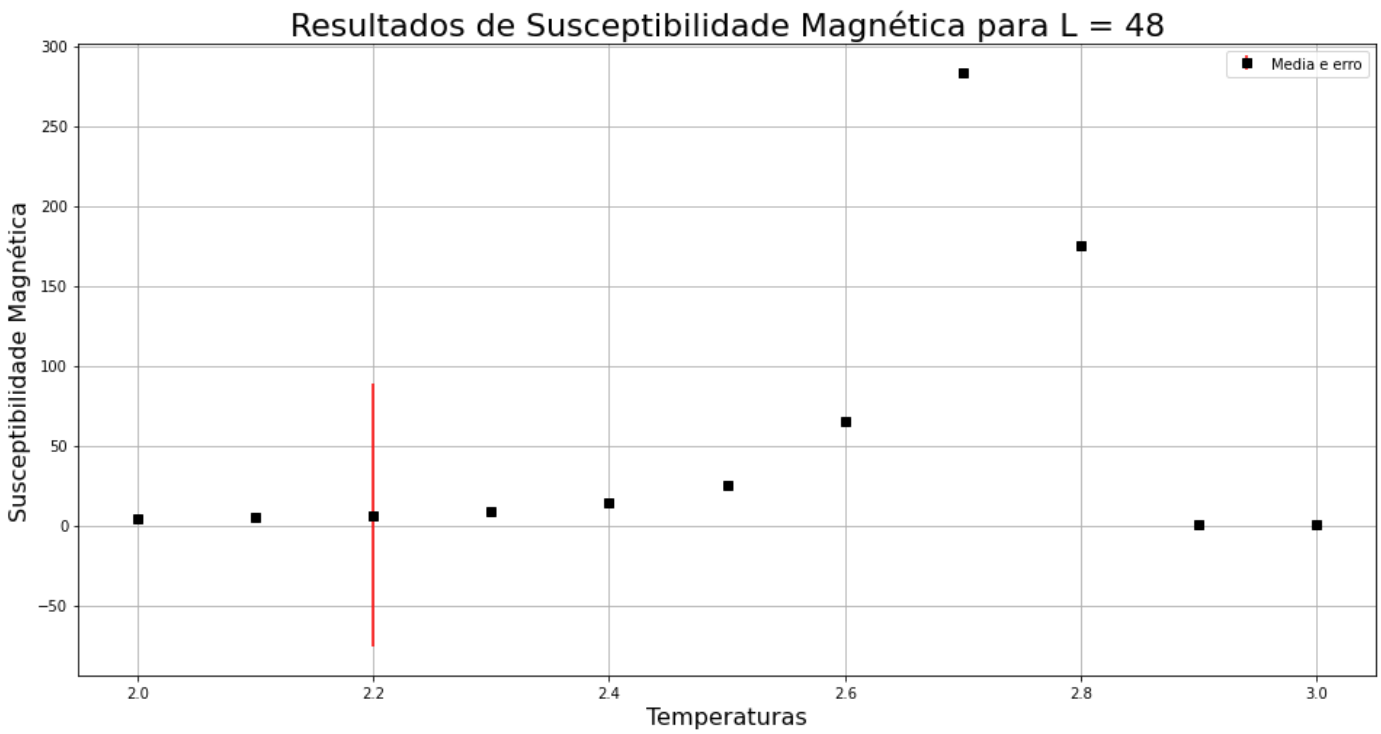
In [ ]:
```
gerarGrafico(calor[0], calor[1], "Calor específico", L)
```



In [ ]:
```
gerarGrafico(magnetizacao[0], magnetizacao[1], "Módulo da Magnetização por spin", L)
```



In [ ]:
```
gerarGrafico(susceptibilidade[0], susceptibilidade[1], "Susceptibilidade Magnética", L)
```



Análise dos resultados

Os resultados obtidos permitem observar que há uma tendência do calor específico e da susceptibilidade magnética aumentarem quando o tamanho da rede cresce. Isso ocorre para a maioria dos valores de temperatura. Assim, para redes muito grandes, é esperado que essas grandezas possuam valores muito altos.

Outro resultado interessante sobre o calor específico e a susceptibilidade magnética é que ambas tiveram seus valores mais altos nas temperaturas de 2.7 ou 2.8 para todos os tamanhos de rede considerados. O fato do ápice ter ocorrido nessas temperaturas para essas duas métricas é um indicativo de que a transição de fase do sistema ocorre em algum valor entre essas duas medidas.

Em geral, os erros estatísticos se mantiveram bem baixos, sendo que foram praticamente imperceptíveis nos gráficos de energia por spin, calor específico e magnetização por spin para qualquer um dos tamanhos de rede e para todas as temperaturas.

Ja para susceptibilidade magnética, as temperaturas de 2.0, 2.1 e 2.2 foram as que apresentaram maior erro considerando todos os tamanhos de rede simulados. Esse resultado se deve ao fato de que, como visto no laboratório prático anterior, o sistema pode apresentar comportamentos instáveis em baixas temperaturas, o que faz com que os valores não se estabilizem nem mesmo após uma quantidade muito grande de iterações.