Introdução à Física Computacional II (4300318)

Prof. André Vieira
apvieira@if.usp.br
Sala 3120 – Edifício Principal

Aula 13

Aplicação: sistemas de dois níveis não interagentes

Uma coleção de sistemas de dois níveis

- Formulação alternativa do sistema (partícula) de dois níveis: estado 0, com energia 0, e estado 1, com energia ϵ.
- A energia total da coleção é

$$E\left(\{n_i\}\right)=\sum_{i=1}^{N} n_i \epsilon, \quad n_i \in \{0,1\}$$

- Como não há interações entre os sistemas que compõem a coleção, a função de partição pode ser calculada analiticamente.

Uma coleção de sistemas de dois níveis

- Um estado da coleção é definido pelo conjunto $\{n_i\}$ que especifica os "números de ocupação" n_i de cada sistema. Logo, a soma sobre estados requerida para o cálculo da função de partição é uma soma sobre todos os conjuntos $\{n_i\}$:

$$Z = \sum_{\{n_i\}} e^{-\beta E(\{n_i\})} = \sum_{n_1=0,1} \sum_{n_2=0,1} \cdots \sum_{n_N=0,1} e^{-\beta \sum_i n_i \epsilon}$$

$$= \left(\sum_{n_1=0,1} e^{-\beta n_1 \epsilon} \right) \left(\sum_{n_2=0,1} e^{-\beta n_2 \epsilon} \right) \cdots \left(\sum_{n_N=0,1} e^{-\beta n_N \epsilon} \right)$$

$$= \left(1 + e^{-\beta \epsilon} \right)^N$$

Uma coleção de sistemas de dois níveis

- Da função de partição calculamos a energia interna:

$$Z = \sum_{\{n_i\}} e^{-\beta E(\{n_i\})} = \left(1 + e^{-\beta \epsilon}\right)^N$$

$$U \equiv \left\langle E(\{n_i\}) \right\rangle = \frac{\sum_{\{n_i\}} E(\{n_i\}) e^{-\beta E(\{n_i\})}}{\sum_{\{n_i\}} e^{-\beta E(\{n_i\})}}$$

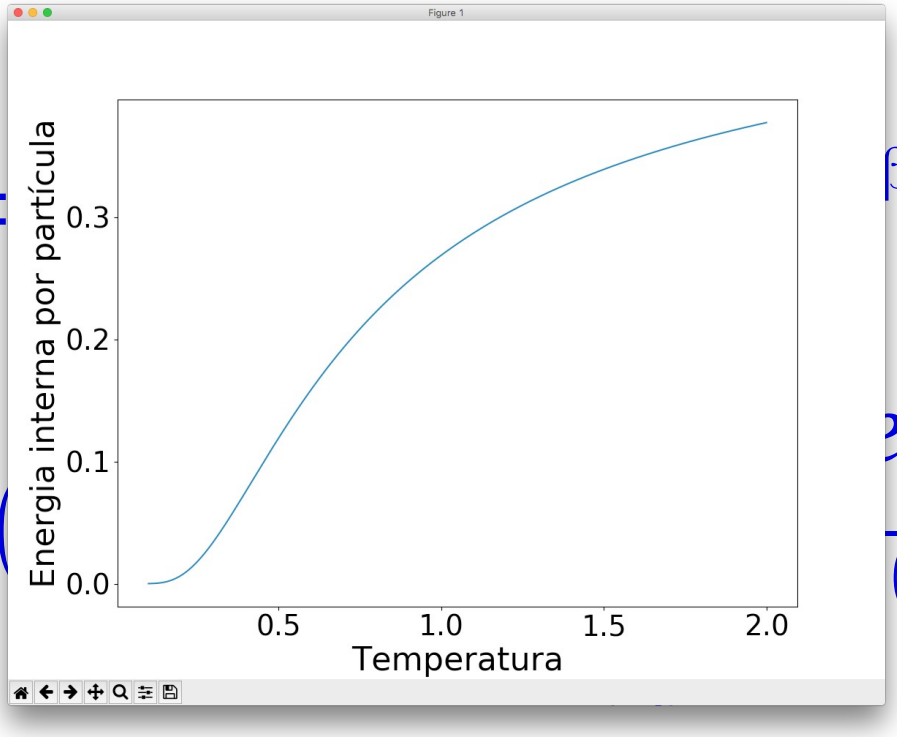
$$\boxed{U = -\frac{\partial \ln Z}{\partial \beta} = \frac{N \epsilon e^{-\beta \epsilon}}{1 + e^{-\beta \epsilon}}}$$

Uma coleção de sistemas de dois níveis

- Da função de partição calculamos a energia interna:

$$Z = \left(1 + e^{-\beta\epsilon}\right)^N$$

$$U \equiv \left\langle E\left(\right.\right. \qquad e^{-\beta E(\{n_i\})} \qquad \left.\left.\right)\right\rangle$$

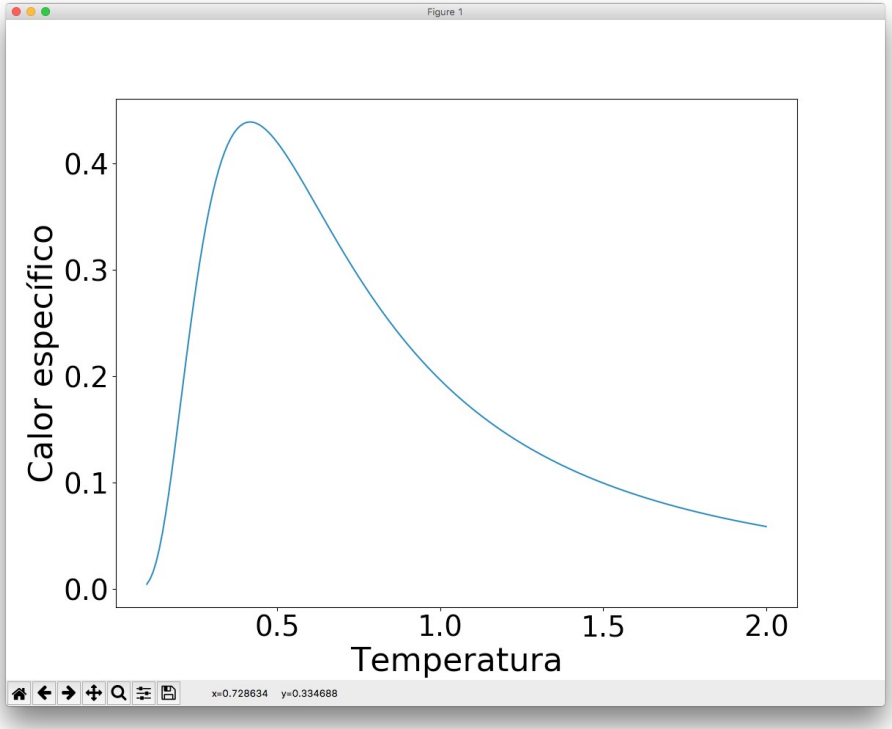


$$U = -\frac{\partial \ln Z}{\partial \beta} = \frac{N\epsilon e^{-\beta\epsilon}}{1+e^{-\beta\epsilon}}$$

Uma coleção de sistemas de dois níveis

- O calor específico é dado analiticamente por

$$c \equiv \frac{1}{N}\frac{\partial U}{\partial T} = \frac{e^{-\beta\epsilon}}{T^2\left(1+e^{-\beta\epsilon}\right)^2}.$$



Uma coleção de sistemas de dois níveis

- O calor específico é dado analiticamente por

$$c \equiv \frac{1}{N}\frac{\partial U}{\partial T} = \frac{e^{-\beta\epsilon}}{T^2\left(1+e^{-\beta\epsilon}\right)^2}.$$

- Seu cálculo numérico é mais preciso se usarmos as **flutuações** da energia:

$$c = -\frac{1}{N}\frac{\partial^2 \ln Z}{\partial T\,\partial \beta} = \frac{\left\langle E^2\right\rangle - \left\langle E\right\rangle^2}{N k_B T^2}.$$

O algoritmo de Metropolis

- O método da cadeia de Markov faz uso de taxas de transição escolhidas para satisfazer o balanceamento detalhado. A mais simples escolha é o **algoritmo de Metropolis**:

$$W_{ij} = \begin{cases} 1, & \text{se } E_j < E_i \\ e^{-\beta(E_j - E_i)}, & \text{se } E_j \geq E_i \end{cases}.$$

- A partir de um certo estado, definimos o novo estado a partir de um conjunto pré-determinado de movimentos possíveis (por exemplo, mudar a configuração de uma única partícula).

Simulações de Monte Carlo

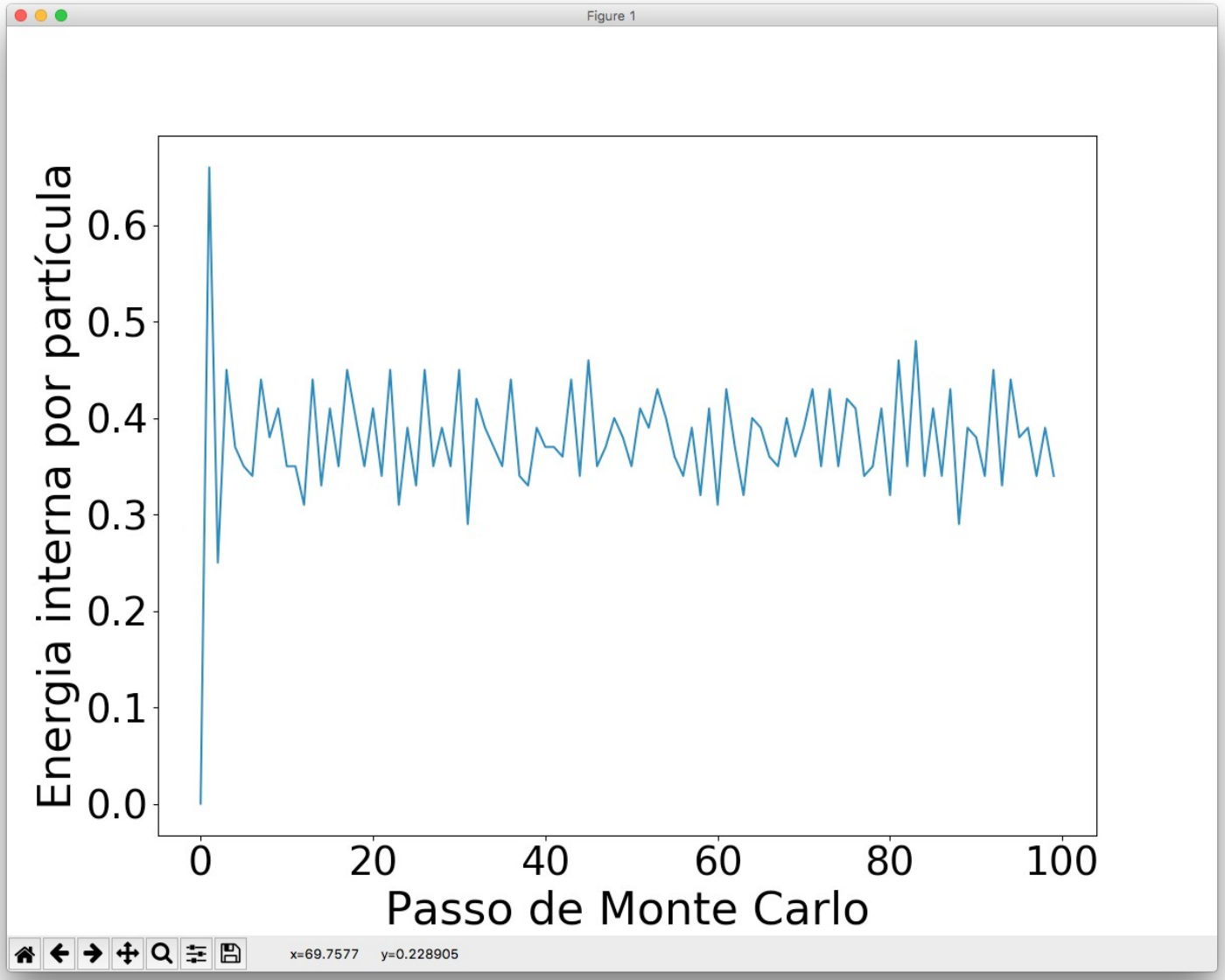
- Eis a receita para implementar uma simulação de Monte Carlo com uma cadeia de Markov.

  1. Escolha um estado inicial qualquer.
  2. Escolha aleatoriamente um movimento a partir do conjunto de possibilidades.
  3. Calcule a taxa de transição W_{ij} associada.
  4. Aceite o movimento com probabilidade W_{ij}; em particular, movimentos que diminuem a energia são sempre aceitos.
  5. Meça e acumule as quantidades de interesse.
  6. Repita a partir do passo 2.

Implementação do código: parte 0

```
# Implementa o cálculo da energia interna para uma coleção de
# sistemas de dois níveis (0 e epsilon) não interagentes.
# Vamos simular a coleção primeiro na temperatura mais alta,
# de modo a decidir quantos passos de Monte Carlo são necessários
# para a equilibração

# Parâmetros
N = 100          # Número de sistemas compondo a coleção
T = 2.0          # Temperatura
Passos = 100     # Número total de passos de Monte Carlo a realizar

# Simulação.
s = full(N,1)           # Partículas inicialmente no estado excitado
boltz = exp(-1.0/T)     # Probabilidade de transição exp(-dE/T)
E_sim = []              # Lista para armazenar os valores da energia
for passo in range(Passos):
    z = random.rand(N)  # Sorteamos N números aleatórios
    # Damos a cada partícula sequencialmente a chance de mudar de estado
    for i in range(N):
        dE = 1 - 2*s[i] # Variação da energia em caso de mudança
        if dE < 0 or z[i] < boltz:  # Mudança foi aceita?
            s[i] = 1 - s[i]         # Registramos a mudança
    energia = sum(s)            # Energia ao final do passo
    E_sim.append(energia/N)

# Traçamos o gráfico
plt.rcParams['xtick.labelsize'] = 28
plt.rcParams['ytick.labelsize'] = 28
plt.rcParams['axes.labelsize'] = 32

plt.figure(figsize=(12,9))
plt.plot(E_sim)
plt.ylabel("Energia interna por partícula")
plt.xlabel("Passo de Monte Carlo")
plt.show()
```

Parte 0: resultado



Como o sistema é não interagente, a equilibração ocorre em poucos passos de Monte Carlo.

Implementação do código: parte 1

```
from math import exp
from numpy import random,linspace,full
import matplotlib.pyplot as plt

# Implementa o cálculo da energia interna por partícula e do
# calor específico em função da temperatura para uma coleção de
# sistemas de dois níveis (0 e epsilon) não interagentes.
# O cálculo é feito tanto a partir das expressões analíticas
# exatas quanto por meio de simulações de Monte Carlo com o
# algoritmo de Metropolis. Trabalhamos com unidades em que epsilon
# e a constante de Boltzmann são ambas iguais a 1. Vamos simular
# a coleção primeiro na temperatura mais alta, e vamos resfriá-la
# progressivamente, em cada etapa aguardando um certo tempo de
# equilibração antes de acumular valores para o cálculo das médias.

# Parâmetros
N = 100          # Número de sistemas compondo a coleção
Tmin = 0.1       # Temperatura mínima
Tmax = 2         # Temperatura máxima
nT = 20          # Número de valores de temperatura entre Tmin e Tmax
Passos = 20      # Número total de passos de Monte Carlo a realizar
Equilibra = 10   # Número de passo de Monte Carlo para equilibração

# Criamos listas para armazenar os valores da temperatura (em ordem
# decrescente), da energia interna média e do calor específico.
dT = (Tmax-Tmin)/(nT-1)
T_lista = [(Tmax - i*dT) for i in range(nT)]
E_sim, C_sim = [], []   # Energia e calor específico segundo a simulação
# Listas para os resultados analíticos
T_exato = linspace(Tmin,Tmax,200)
E_exato, C_exato = [], []
for T in T_exato:
    boltz = exp(-1.0/T)                 # Fator de Boltzmann
    uboltz = 1.0 + boltz
    E_exato.append(boltz/uboltz)
    C_exato.append(boltz/(T*uboltz)**2)
```

Implementação do código: parte 1

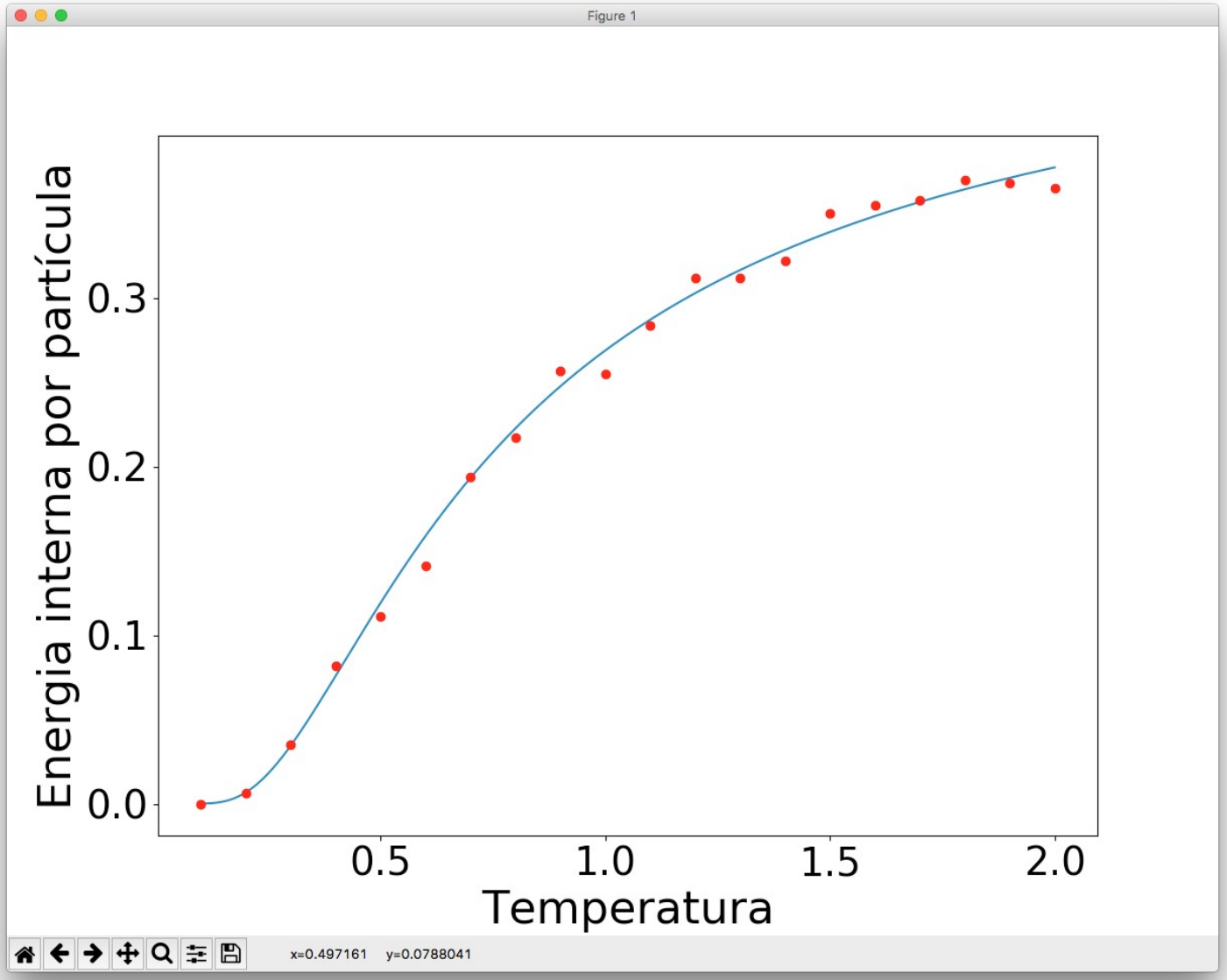
```
# Simulação. Há um laço externo que percorre as temperaturas e outro
# laço interno que percorre os passos de Monte Carlo, coletando médias
# apenas após os primeiros 10 passos.
s = full(N,1)   # Partículas inicialmente no estado excitado
for n in range(nT):
    T = T_lista[n]
    boltz = exp(-1.0/T)     # Probabilidade de transição exp(-dE/T)
    # Primeiro aguardamos os passos de equilibração
    for passos in range(Equilibra):
        z = random.rand(N)  # Sorteamos N números aleatórios
        # Damos a cada partícula sequencialmente a chance de mudar de estado
        for i in range(N):
            dE = 1 - 2*s[i] # Variação da energia em caso de mudança
            if dE < 0 or z[i] < boltz:  # Mudança foi aceita?
                s[i] = 1 - s[i]         # Registramos a mudança
    # Vamos começar a calcular médias
    acumula_E, acumula_E2 = 0.0, 0.0
    # Os passos restantes servem para cálculo das médias
    for passos in range(Equilibra,Passos):
        z = random.rand(N)  # Sorteamos N números aleatórios
        # Damos a cada partícula sequencialmente a chance de mudar de estado
        for i in range(N):
            dE = 1 - 2*s[i] # Variação da energia em caso de mudança
            if dE < 0 or z[i] < boltz:  # Mudança foi aceita?
                s[i] = 1 - s[i]         # Registramos a mudança
        energia = sum(s)            # Energia ao final do passo
        acumula_E += energia        # Acumulamos a energia
        acumula_E2 += energia**2    # Acumulamos a energia quadrática
    # Agora registramos as médias
    E_medio = acumula_E/(Passos-Equilibra)
    E2_medio = acumula_E2/(Passos-Equilibra)
    E_sim.append(E_medio/N)
    C_sim.append((E2_medio-E_medio**2)/N/T**2)
```

Parte 1: resultados

$N = 100$

$\text{Passos} = 20$

$\text{Equilibra} = 10$



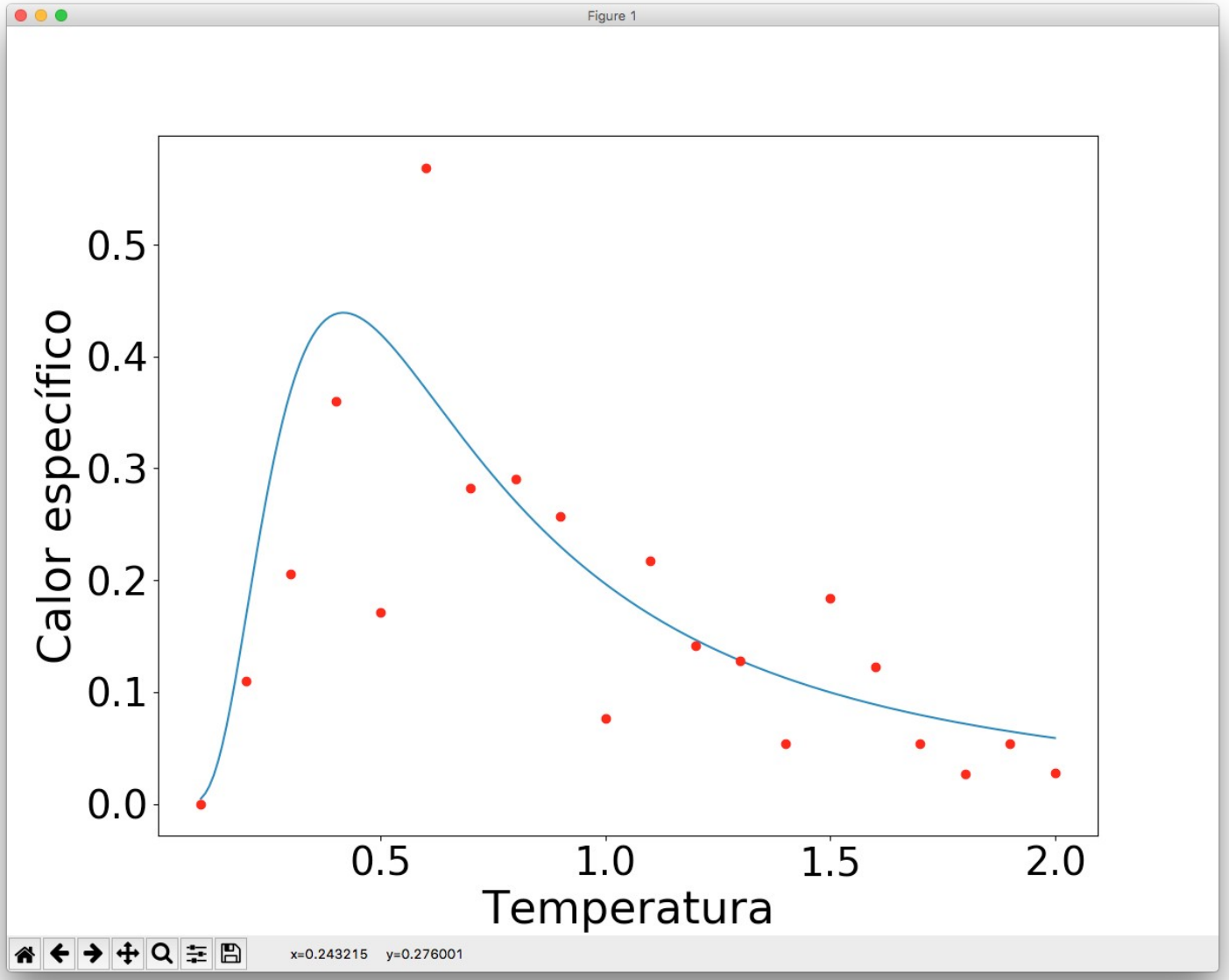
A curva exata está mostrada em azul e o resultado da simulação está em vermelho.

Parte 1: resultados

$N = 100$

$\text{Passos} = 20$

$\text{Equilibra} = 10$



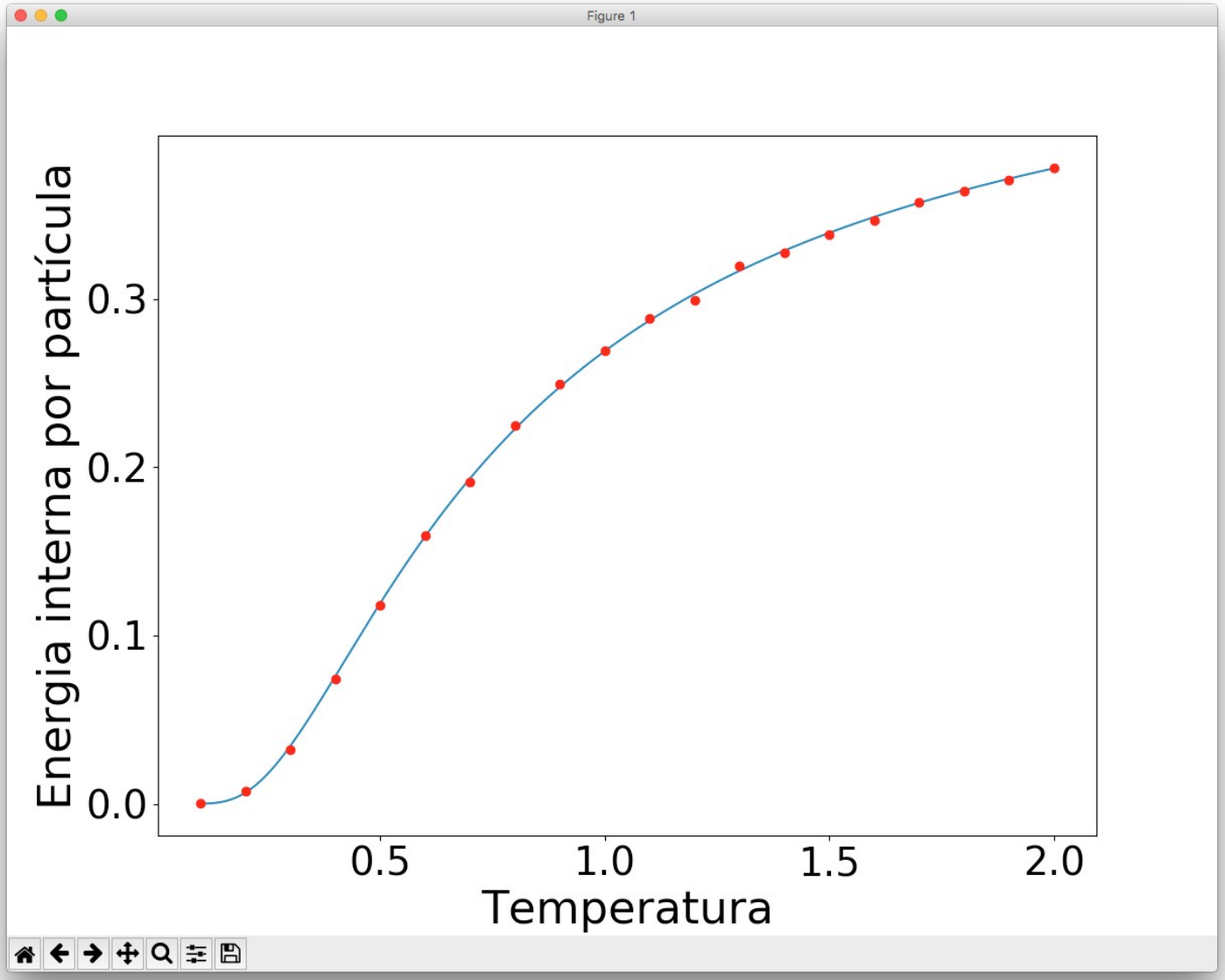
Note que os erros no calor específico são bem maiores do que aqueles na energia interna.

Parte 1: resultados

$N = 100$

$\text{Passos} = 200$

$\text{Equilibra} = 10$



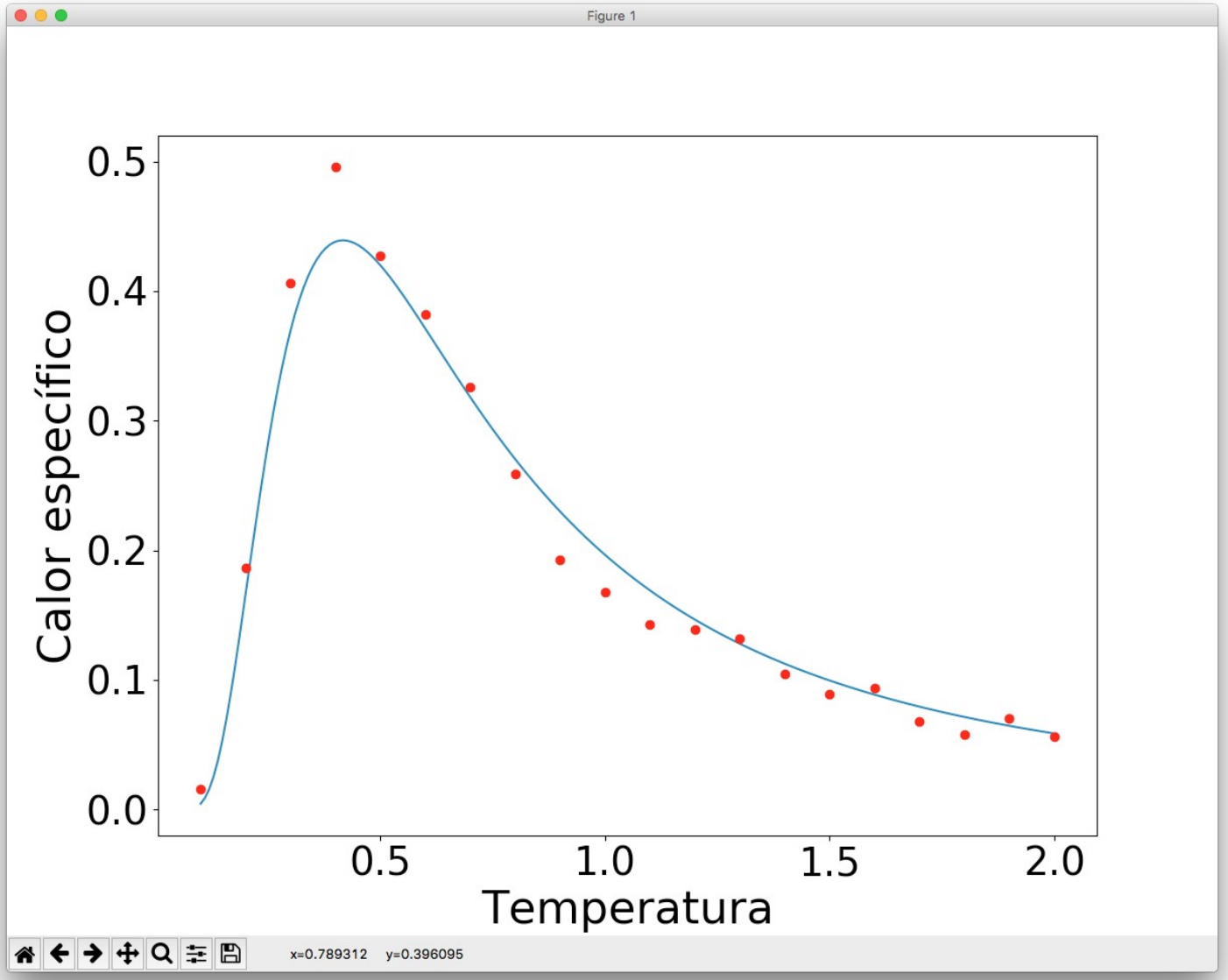
Aumentando número de passos de medidas, há uma grande diminuição dos erros.

Parte 1: resultados

$N = 100$

$\text{Passos} = 200$

$\text{Equilibra} = 10$



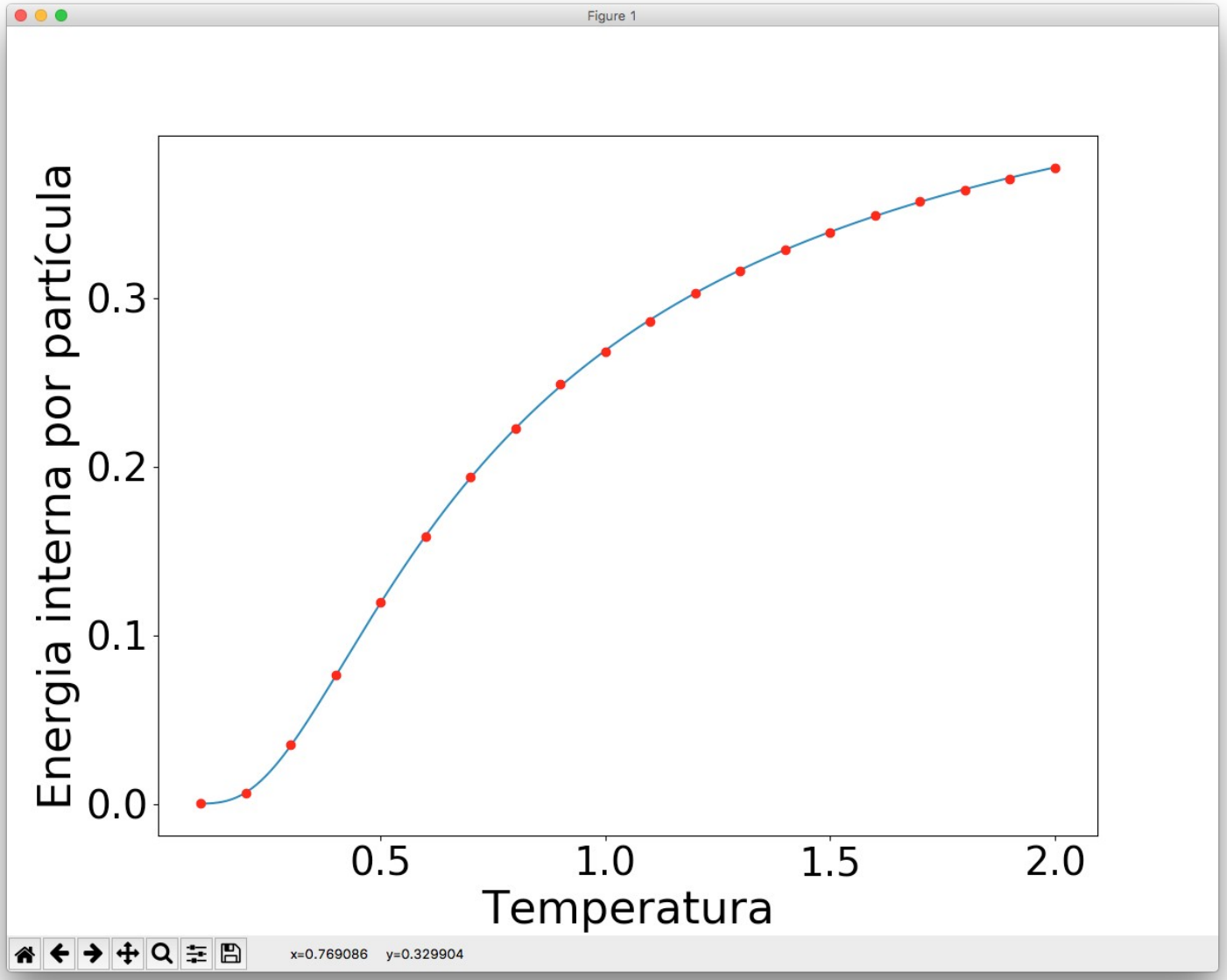
A diminuição dos erros ocorre também no calor específico, mas os erros são sempre maiores do que os da energia interna.

Parte 1: resultados

$N = 100$

$\text{Passos} = 2000$

$\text{Equilibra} = 10$



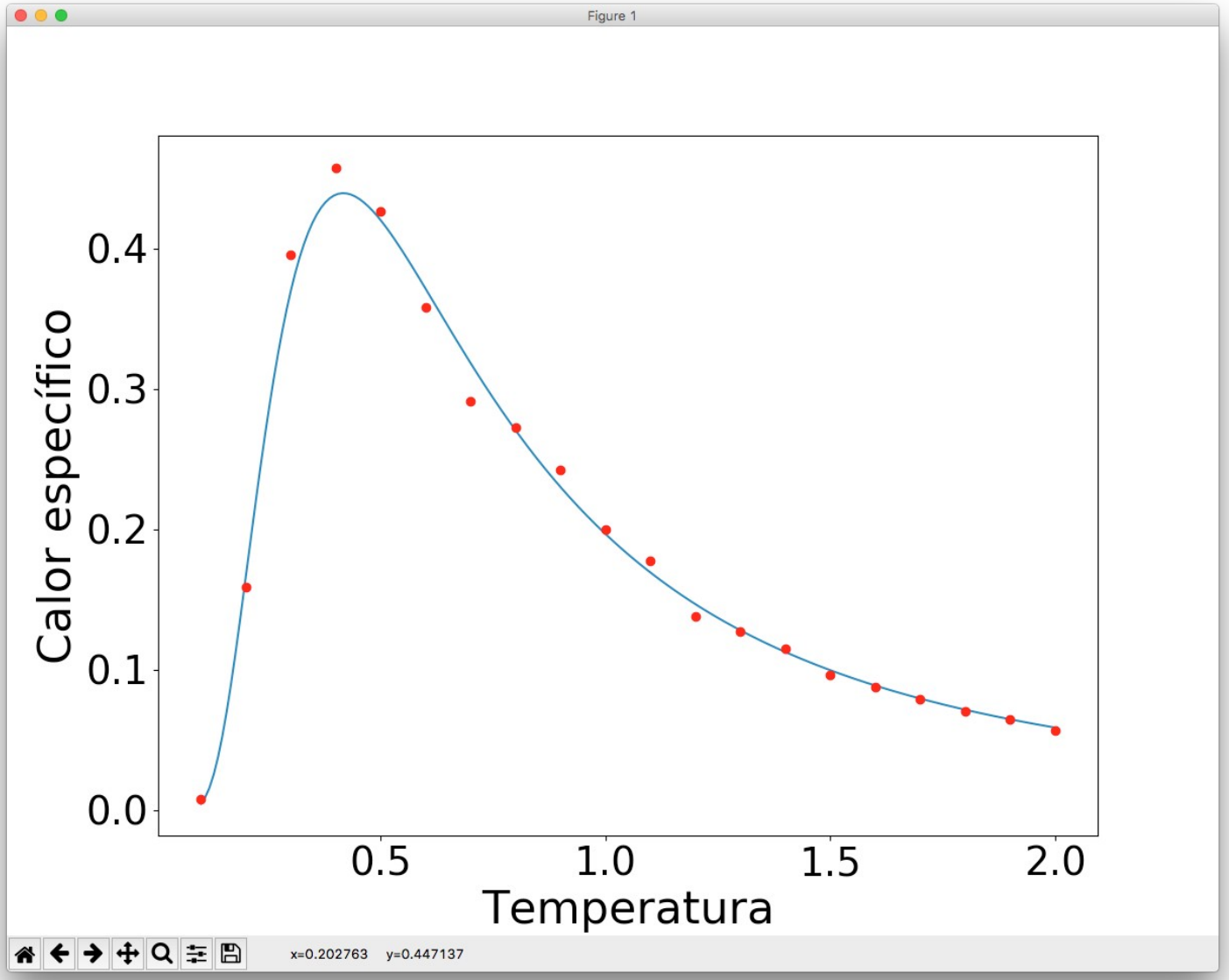
Aumentando ainda mais o número de passos de medidas, os erros na energia interna tornam-se praticamente imperceptíveis.

Parte 1: resultados

$N = 100$

$\text{Passos} = 2000$

$\text{Equilibra} = 10$



Novamente os erros no calor específico continuam relevantes, embora sigam diminuindo.

Como estimar erros?

- Nesse problema, poderíamos quantificar erros comparando com o resultado exato. Mas o que fazer nos casos em que não dispomos da solução exata? Afinal, são esses os casos em que é preciso utilizar simulações!

Como estimar erros?

- **Utilizamos métodos estatísticos**.
- Para uma grandeza qualquer g, tomamos medidas g_i e calculamos a média sobre um certo número M de amostras:

$$\langle g \rangle = \frac{1}{M} \sum_{i=1}^{M} g_i .$$

- Uma estimativa do erro no cálculo é o desvio padrão da média de g, dado por

$$\sigma_g = \sqrt{\frac{1}{M(M-1)} \sum_{i=1}^{M} \left(g_i - \langle g \rangle \right)^2} = \sqrt{\frac{\langle g^2 \rangle - \langle g \rangle^2}{M-1}} .$$

Como estimar erros?

- **Utilizamos métodos estatísticos**.

Para uma dedução desses resultados (e para tudo o que você sempre quis saber sobre análise e ajuste de dados mas tinha medo de perguntar), veja este artigo em inglês, disponibilizado também na página da disciplina no Moodle.

$$\sqrt{M(M-1)}\sum_{i=1}$$

Como estimar erros?

- Na nossa simulação, a cada passo de Monte Carlo realizamos uma medida E_i da energia, de modo que em cada temperatura estimamos

$$\langle E^{\alpha} \rangle = \frac{1}{M} \sum_{i=1}^{M} E_i^{\alpha}, \quad \sigma_E = \sqrt{\frac{\langle E^2 \rangle - \langle E \rangle^2}{M-1}}.$$

Como estimar erros?

- Por outro lado, o calor específico é estimado a partir das flutuações da energia,

$$c = \frac{\langle E^2 \rangle - \langle E \rangle^2}{N k_B T^2},$$

e portanto só pode ser calculado após todos os passos serem realizados. Como podemos estimar o erro em seu cálculo?

Como estimar erros?

- Por outro lado, o calor específico é estimado a partir das flutuações da energia,

$$c = \frac{\langle E^2 \rangle - \langle E \rangle^2}{N k_B T^2},$$

  e portanto só pode ser calculado após todos os passos serem realizados. Como podemos estimar o erro em seu cálculo?

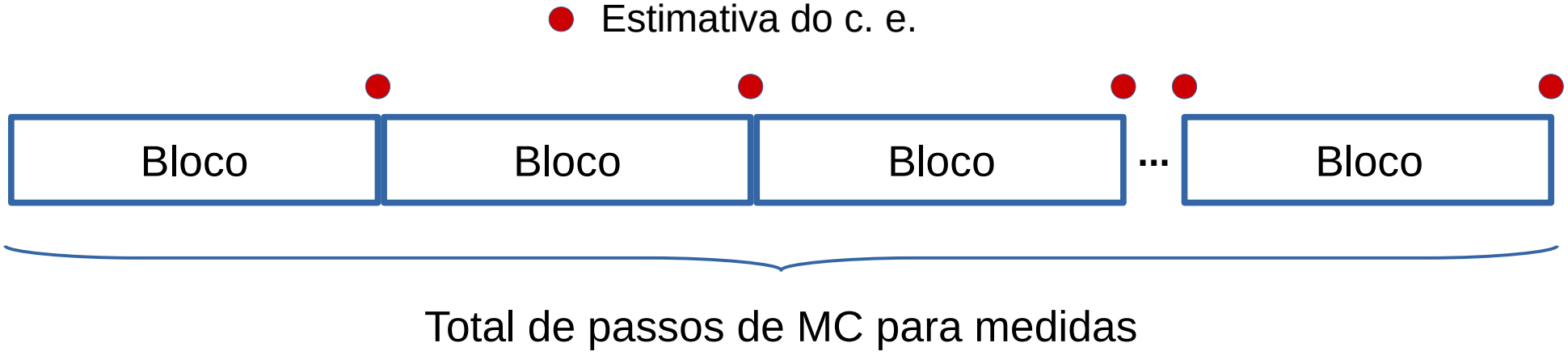
- A forma mais simples é produzir estimativas parciais, a cada bloco de passos de Monte Carlo, com tamanho fixo, e quantificar o erro pelo desvio padrão dessas estimativas.

Como estimar erros?

- Por outro lado, o calor específico é estimado a partir das flutuações da energia,

$$c = \frac{\langle E^2 \rangle - \langle E \rangle^2}{N k_B T^2},$$

e portanto só pode ser calculado após todos os passos serem realizados. Como podemos estimar o erro em seu cálculo?



Implementação do código: parte 2

```
# Parâmetros
N = 100                # Número de sistemas compondo a coleção
Tmin = 0.1             # Temperatura mínima
Tmax = 2               # Temperatura máxima
nT = 20                # Número de valores de temperatura entre Tmin e Tmax
Passos = 20            # Número total de passos de Monte Carlo a realizar
Equilibra = 10         # Número de passo de Monte Carlo para equilibração
M = Passos-Equilibra # Número de amostras para estimar médias
Bloco = M//5           # No. de passos de MC para estimar erro no c.e.
```

...

```
# Simulação. Definimos uma função para implementar um único passo
# de Monte Carlo. Buscamos coletar médias apenas após a equilibração.
s = full(N,1)   # Partículas inicialmente no estado excitado

def passo_MC(s,boltz):  # Função que implementa um passo de Monte Carlo
    z = random.rand(N)  # Sorteamos N números aleatórios
    # Damos a cada partícula sequencialmente a chance de mudar de estado
    for i in range(N):
        dE = 1 - 2*s[i] # Variação da energia em caso de mudança
        if dE < 0 or z[i] < boltz:  # Mudança foi aceita?
            s[i] = 1 - s[i]         # Registramos a mudança
    return(s)
```

Implementação do código: parte 2

```
for n in range(nT):
    T = T_lista[n]
    boltz = exp(-1.0/T)     # Probabilidade de transição exp(-dE/T)
    # Primeiro aguardamos os passos de equilibração
    for passo in range(Equilibra):
        s = passo_MC(s,boltz)
    # Vamos começar a calcular médias
    acumula_E, acumula_E2, acumula_C, acumula_C2 = 0.0, 0.0, 0.0, 0.0
    # Os passos restantes servem para cálculo das médias
    conta = 0
    for jext in range(M//Bloco):
        acumula_Ece, acumula_E2ce = 0.0, 0.0 # Para estimar c.e. e seu erro
        for jint in range(Bloco):
            s = passo_MC(s,boltz)
            energia = sum(s)
            acumula_Ece += energia        # Acumulamos a energia
            acumula_E2ce += energia**2    # Acumulamos a energia quadrática
            conta += 1
        Ece_medio = acumula_Ece/Bloco
        E2ce_medio = acumula_E2ce/Bloco
        dc = (E2ce_medio-Ece_medio**2)/N/T**2
        acumula_C += dc
        acumula_C2 += dc**2
        acumula_E += acumula_Ece      # Acumulamos a energia
        acumula_E2 += acumula_E2ce    # Acumulamos a energia quadrática
    # Agora registramos as médias
    E_medio = acumula_E/M
    E2_medio = acumula_E2/M
    E_sim.append(E_medio/N)
    E_erro.append(sqrt((E2_medio-E_medio**2)/N**2/(M-1)))
    C_medio = acumula_C/(M//Bloco)
    C2_medio = acumula_C2/(M//Bloco)
    C_sim.append((E2_medio-E_medio**2)/N/T**2)
    C_erro.append(sqrt((C2_medio-C_medio**2)/(M//Bloco-1)))
```

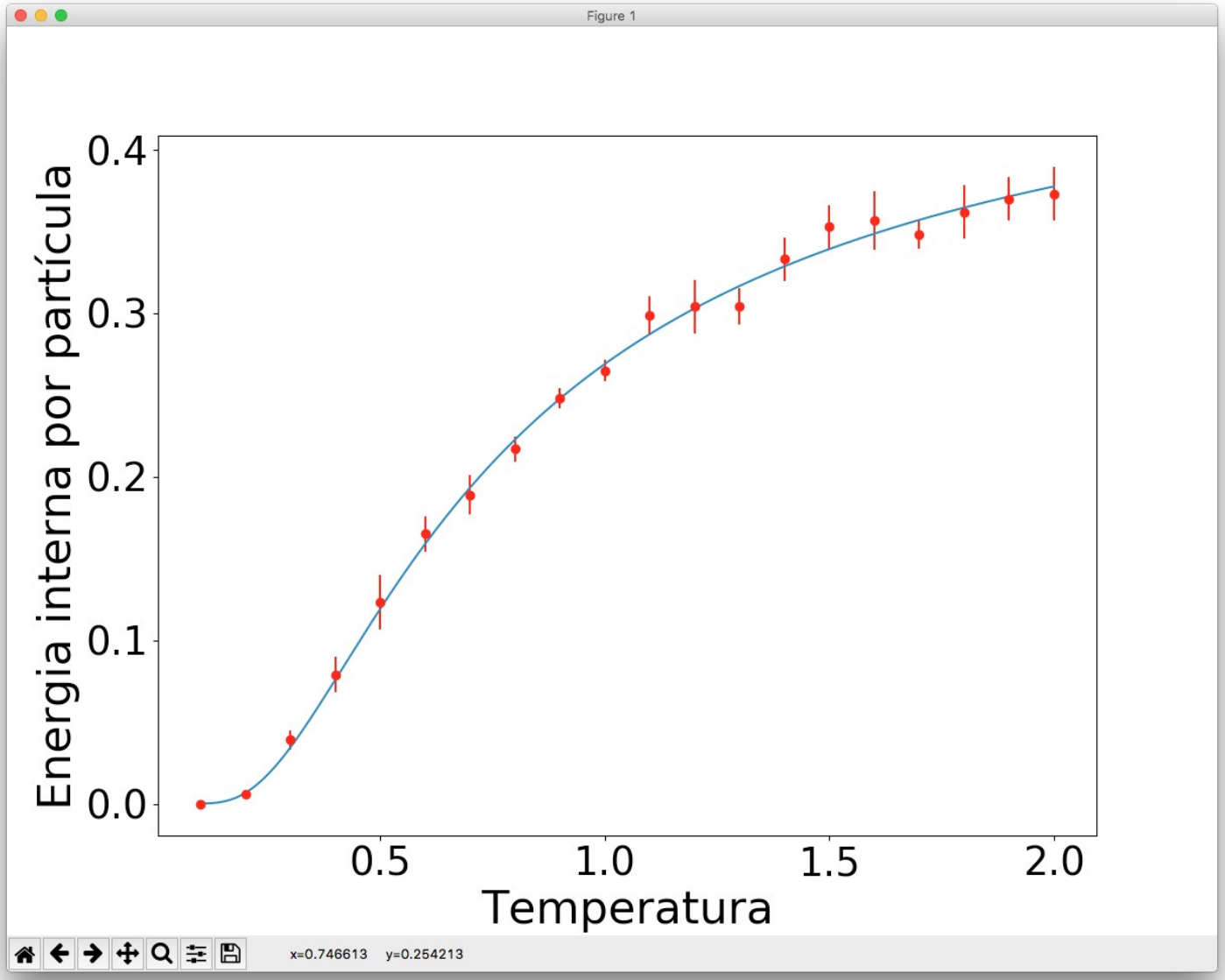
Note que para o calor específico médio utilizamos estimativas mais precisas.

Parte 2: resultados

$N = 100$

$\text{Passos} = 20$

$\text{Equilibra} = 10$



O resultado da simulação, em vermelho, agora contém barras que indicam a estimativa do erro.

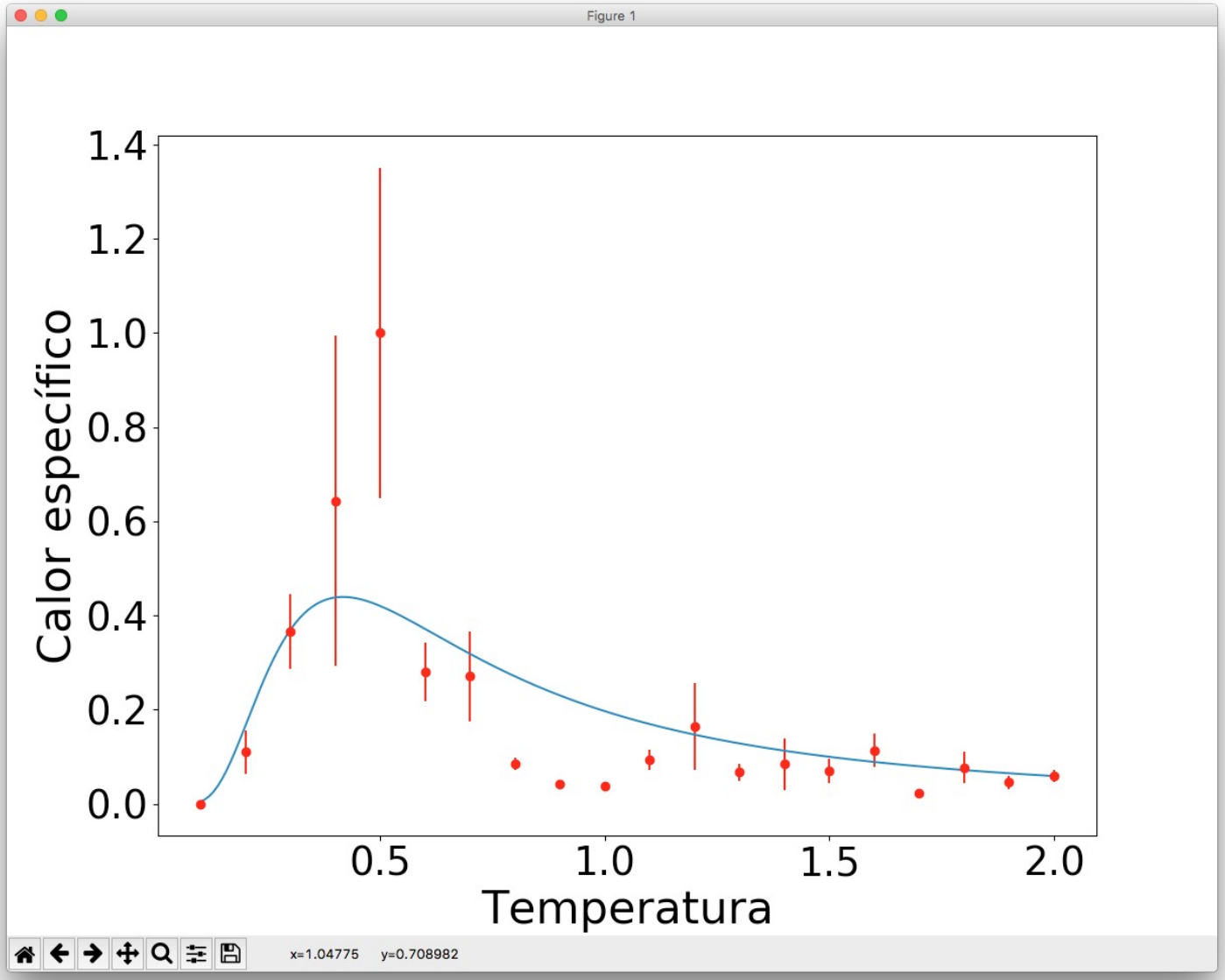
Parte 2: resultados

$N = 100$

$\text{Passos} = 20$

$\text{Equilibra} = 10$

$\text{Bloco} = 2$



Como utilizamos poucos passos, a própria estimativa do erro no calor específico é grosseira.

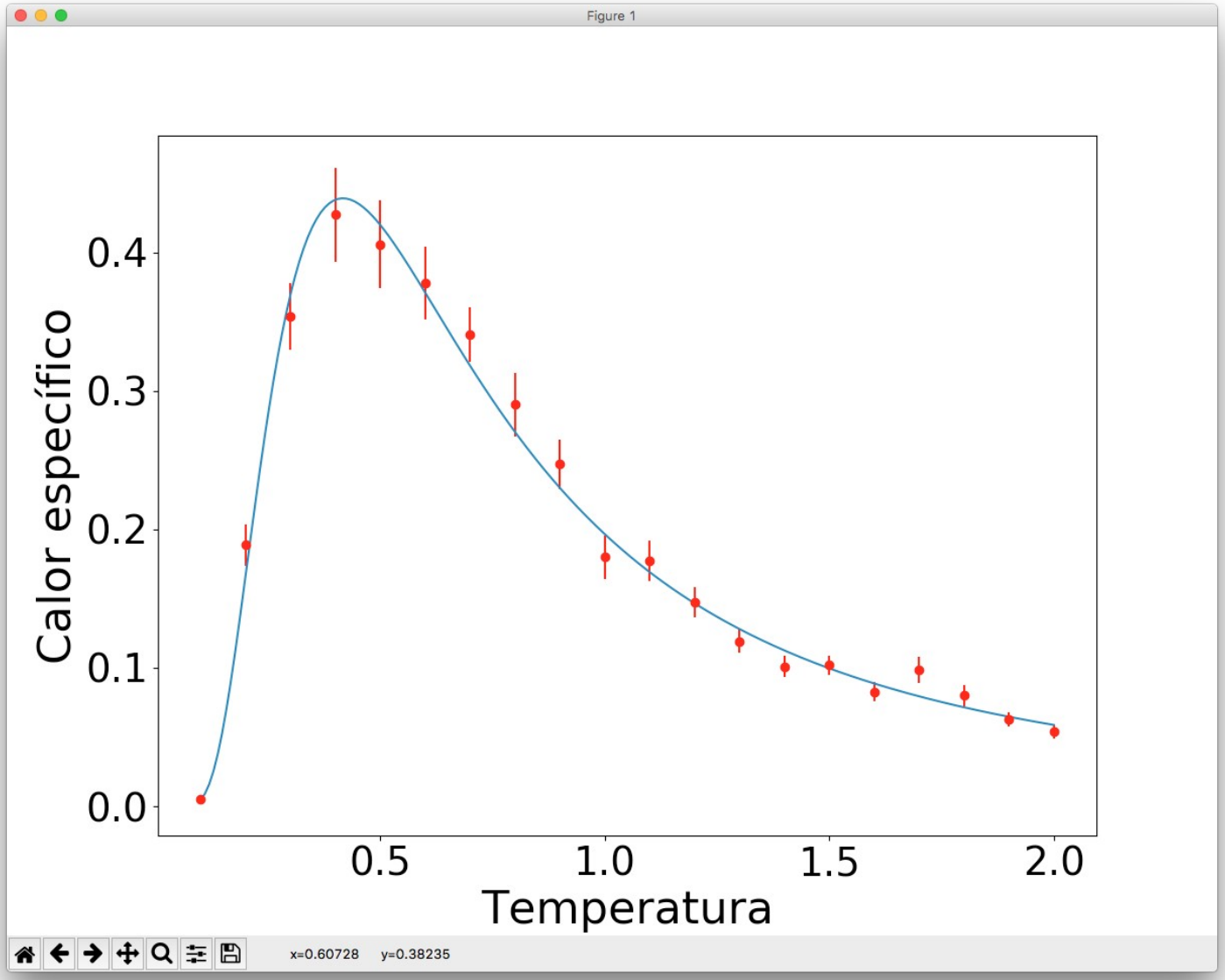
Parte 2: resultados

$N = 100$

$\text{Passos} = 500$

$\text{Equilibra} = 100$

$\text{Bloco} = 10$



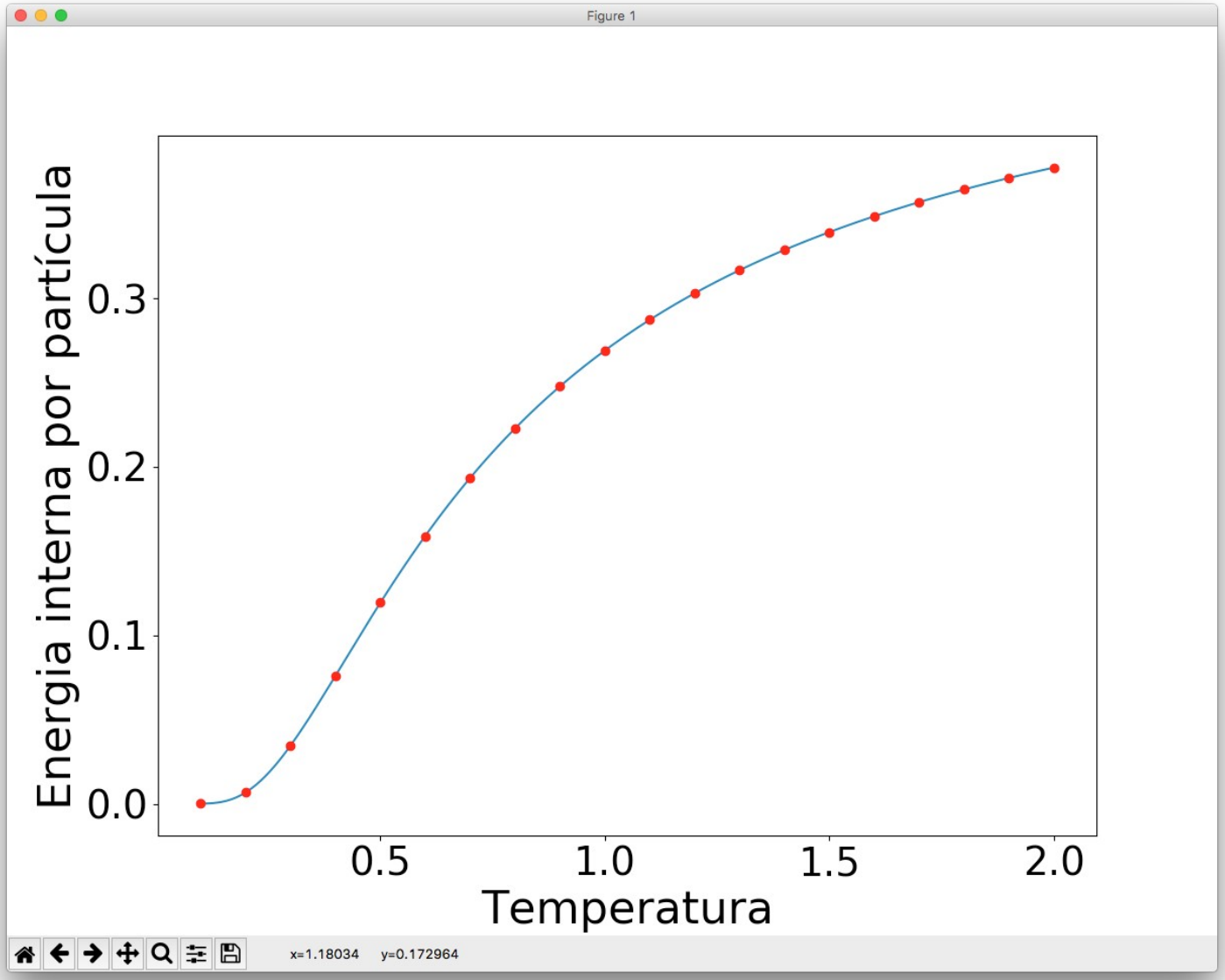
Utilizando mais passos, a estimativa do erro no calor específico torna-se mais confiável.

Parte 2: resultados

$N = 100$

$\text{Passos} = 2 \times 10^4$

$\text{Equilibra} = 100$



Aumentando mais o número de passos, aqui as barras de erro imperceptíveis indicam corretamente que o erro é desprezível.

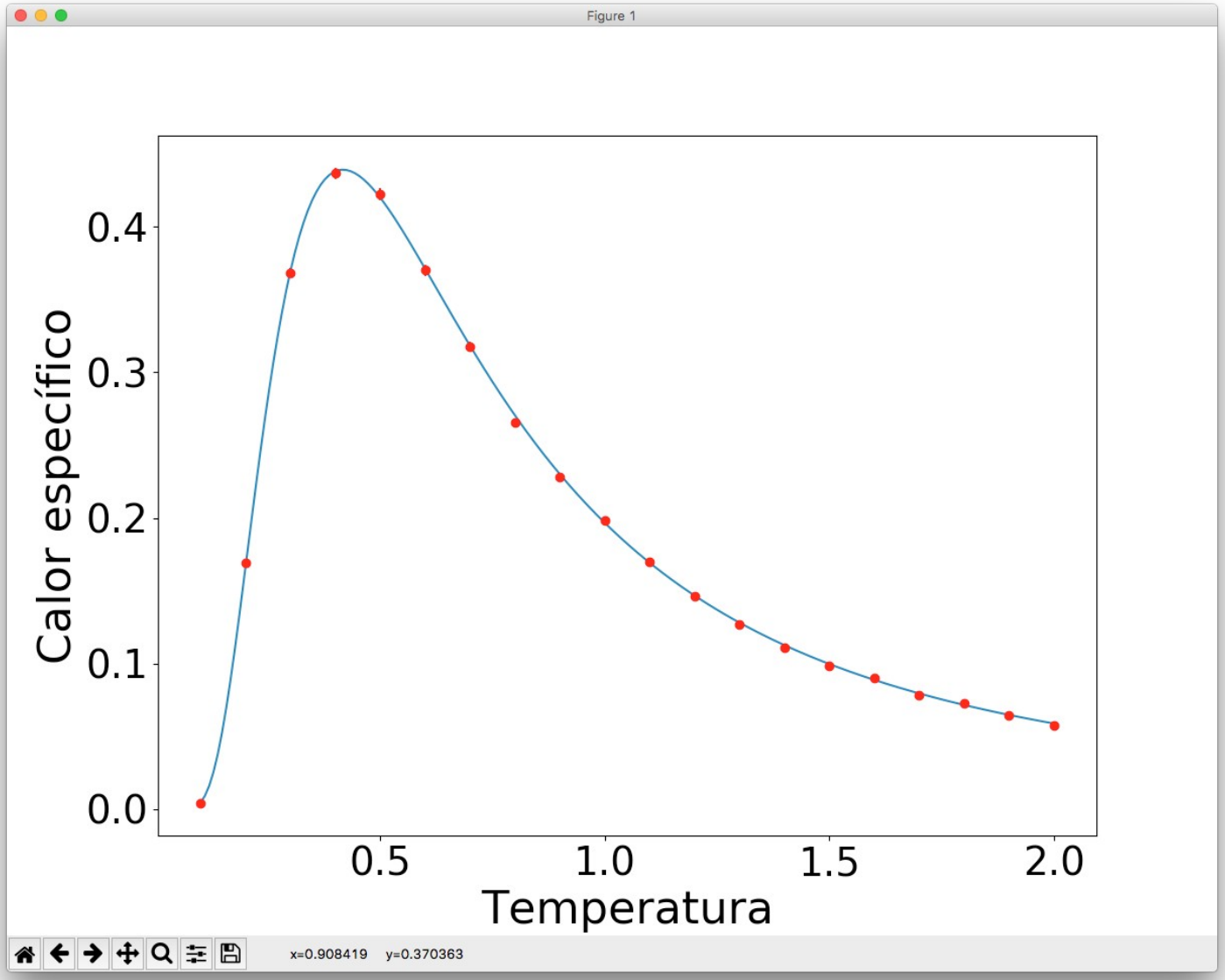
Parte 2: resultados

$N = 100$

$\text{Passos} = 2 \times 10^4$

$\text{Equilibra} = 100$

$\text{Bloco} = 199$



Olhando com atenção, ainda vemos pequenas barras de erro no calor específico, sugerindo corretamente pequenos erros.

Exercício no moodle

Há um único exercício, explorando o conteúdo da aula de hoje, e que pode ser feito com base nos programas dos exemplos desta aula e da aula anterior, disponíveis no moodle. O exercício deve ser entregue até o dia **17 de junho**.