Relatório da primeira "tarefa": Modelo de Ising

Aluno: Thales Aparecido
NUSP: 11739745

Introdução

Em nossa primeira tarefa devemos implementar um modelo de Ising para sólidos, nesse caso representamos um material magnético por um grid em que cada elétron tem seu spin para cima ou para baixo (+1 ou -1):

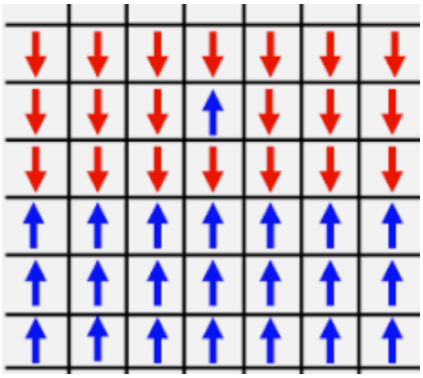


Figure 1: Eletróns representados em um grid

Nesse caso, podemos sempre representar a energia em tal sistema por:

$$H = -\frac{1}{2}J\sum_{\langle u,v\rangle} s_n s_m - B\sum_n s_n$$

Onde o primeiro termo J representa uma interação local entre os spins e o segundo termo B representa um campo magnético. Definimos também a magnetização do sistema como:

$$M = \frac{1}{N}\sum_n s_n$$

Nosso objetivo no presente trabalho é estudar a evolução do sistema usando o algoritmo proposto por Metropolis, qual seja:

1. Inicialize o sistema com uma configuração aleatória de spins. Uns pra cima, uns pra baixo.
2. Execute a seguinte interação t vezes, onde t é o número de time steps da sua simulação:
   (a) Escolha um spin aleatório pra virar. Esse é o seu s_n.
   (b) Calcule a mudança na energia total caso o s_n mude de sinal. Pra isso você não precisa recalcular toda a energia, é só ver a parte que mudou (portanto a vizinhança de s_n).
   (c) Chame essa mudança de energia de ΔH.
      i. Se $\Delta H < 0$, a mudança é aceita.
      ii. Se $\Delta H > 0$, a mudança é aceita com uma probabilidade $P = exp(-/T)$. Pra calcular isso, você calcula P, e depois gera um número aleatório x entre 0 e 1. Se $x < P$, você aceita a mudança de spin, e se $x > P$, você não aceita a mudança de spin.

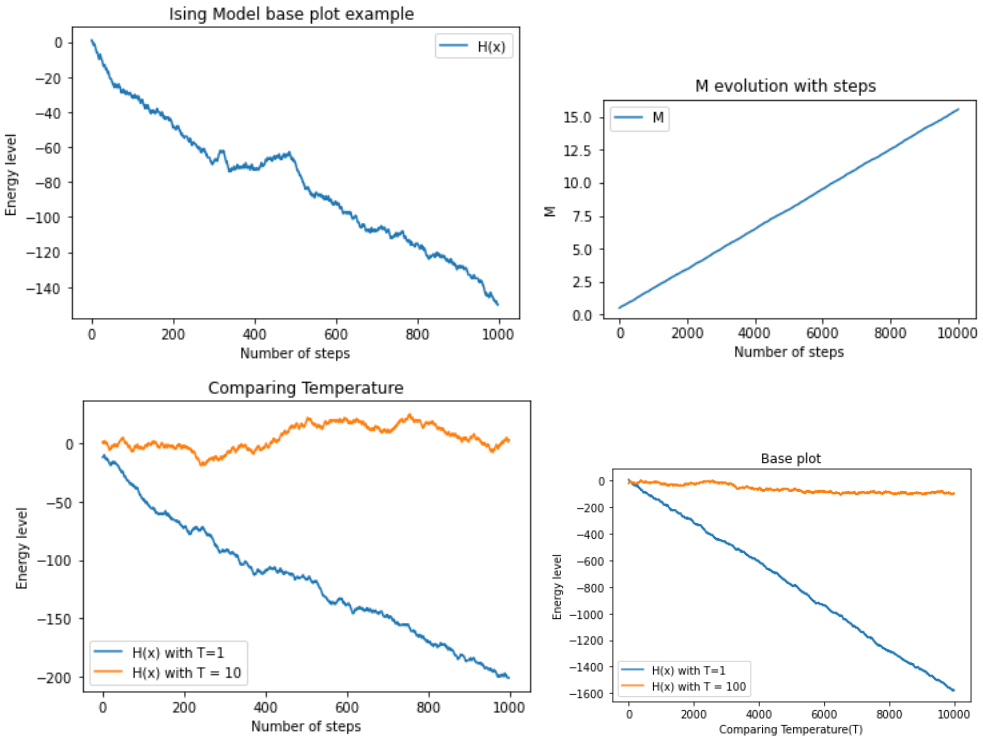
Métodos

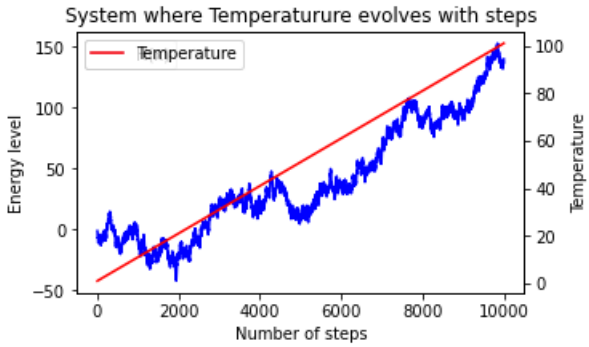
Criamos um programa em python com uma classe Lattice, munida de métodos .delta_h e .step, que retornam, respectivamente, o valor da variação de energia criada no objeto por uma mudança de coordenda e produz um passo do algoritmo de Metropolis. Além disso, tal classe também possui o atributo .M que retorna o

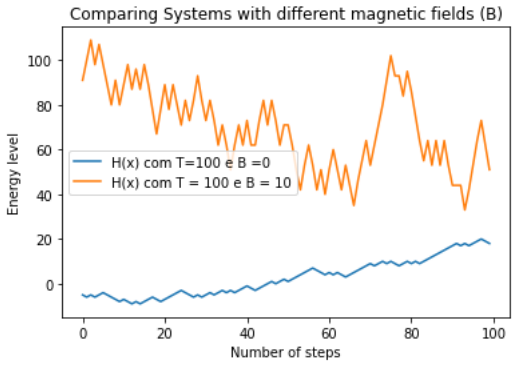
nível de magnetização em dado momento do objeto. Tal classe é criada definindo um número de dimensões e tamanho, sendo possível criar simulações de qualquer dimensão ou tamanho. Por último criamos uma função simulation(), também em Metropolis.py, que é capaz de rodar uma simulação de maneira automatizada, dados tamanho, dimensão, temperatura, campo magnético e J, gerando ao fim um gráfico com o uso de matplotlib.

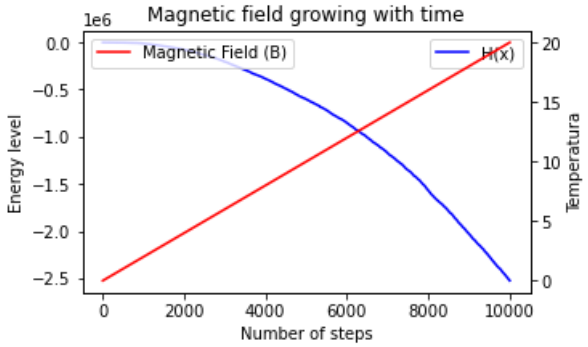
Resultados

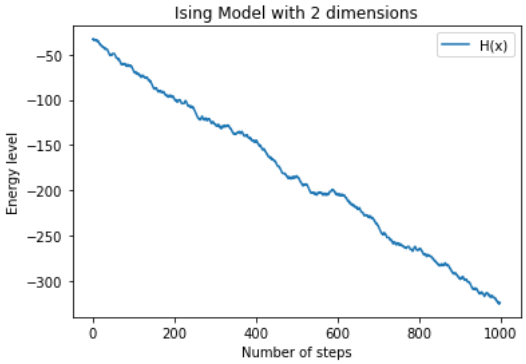
Segue o resultado de alguns testes feitos pelo programa SimulacaoMetropolis.py que gera alguns gráficos usando matplotlib:

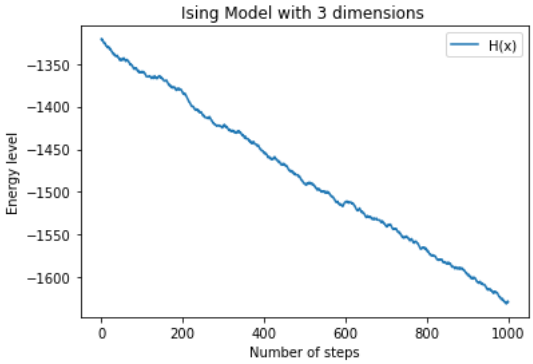












Conclusão

Após ver diversas simulações e gráficos da progressão do algoritmo de Metropolis concluímos que:

1. O Sistema tende irremediavelmente para uma situação de menor energia, que corrobora com uma maior magnetização, isto é, mais spins de mesmo sentido próximos. Vemos isso no gráfico da evolução de M e também no gráfico de progressão "Ising Model base plot".

2. O aumento da temperatura retarda a estabiliazção do sistema, como vemos em "Comparing Temperatures".

Para mais informações sobre o modelo criado, acessar: https://github.com/thalesdaviddom/IC/tree/main/IsingModel