Uma imagem com fogo de artifício, escuridão, espaço

Descrição gerada automaticamente

**Simulação e Modelação Computacional em Engenharia Física**

Relatório Laboratório #01

Grupo:

Daniela Marques, 59939, MIEF

Tiago Neves, 59249, MIEF

Rui Filipe, 60461, MIEF

**Introduction (**About one page)

Objective of the simulation - DANI

What physical phenomena are we trying to simulate, with a concise description - DANI

**Code**

Explain the code structure - DANI

How it works (what information it needs and it’s given to the code) – DANI

**Decisões v0**

1. Fronteiras abertas ou fechadas?

Decidimos utilizar fronteiras fechadas, ou seja, conectar as extremidades da rede de modo a ter um sistema fechado. Estas condições permitem evitar artefactos que possam surgir nos resultados devido ao tamanho finito da rede e à influência da fronteira, facilitando a obtenção de resultados mais fidedignos.

1. Tamanho da rede?

Decidimos seguir a recomendação de dimensões 10x10x10. Visto que utilizamos fonteiras fechadas, não é necessária uma rede muito extensa para obter bons resultados, que caso contrário seria necessária para compensar os efeitos da fronteira. Este tamanho reduzido permite correr bastantes ciclos de Monte Carlo com *runtime*s viáveis.

1. Ao variar a temperatura ou o campo deve-se começar com a rede inicializada sempre da mesma forma ou como terminou no valor anterior?

Começar com a rede como terminou no valor anterior de temperatura/campo captura melhor a dinâmica do sistema em resposta a mudanças nas condições. Porém, torna mais difícil a reprodução de resultados precisos e comparação de diferentes simulações, pois a inicialização não é padronizada. Pode ainda introduzir *bias* na análise, especialmente se a rede convergir para estados metaestáveis que não são representativos do equilíbrio termodinâmico. Utilizar inicialização padronizada e aumentar o número de ciclos compensa o facto de não capturar rapidamente a dinâmica do sistema em resposta à mudança de temperatura/campo, assim, foi esta a nossa escolha. De qualquer modo, testámos as duas hipóteses e iremos apresentar na próxima secção os resultados.

1. Que números de ciclos iniciais se devem não contabilizar para efeitos de obtenção de valores?

Empiricamente, escolhemos um valor para o número de ciclos a rejeitar igual a 10 vezes o tamanho da rede. Sistemas maiores podem levar mais tempo para equilibrar-se do que sistemas menores, sendo necessário descartar mais ciclos iniciais. Ainda assim, consideramos que o sistema converge relativamente rápido, não sendo necessário eliminar uma percentagem significativa dos ciclos.

**Preliminary results** – NEVES

Uma imagem com texto, diagrama, file, Gráfico

Descrição gerada automaticamenteUma imagem com texto, file, diagrama, captura de ecrã

Descrição gerada automaticamente

Figura 1 - Rede inicializada sempre da mesma maneira

Uma imagem com diagrama, texto, file, Gráfico

Descrição gerada automaticamenteUma imagem com texto, file, diagrama, Gráfico

Descrição gerada automaticamente

Figura 2 - Rede inicializada como terminou no valor anterior de temperatura e campo

Não se observam diferenças significativas entre as duas estratégias.

**Computing Complexity**

**transitionFunctionValues**: Itera sobre uma *range* fixa (-6 a 6) e realiza operações de tempo constante dentro do ciclo. Portanto, a sua complexidade temporal é O(1).

**init**: Inicializa uma matriz com um dado tamanho. Como apenas envolve a criação de uma matriz e o seu preenchimento com um valor constante, a sua complexidade temporal é O(*size*^3), onde size é o tamanho da rede. No entanto, visto que usamos um *numpy array*, a sua complexidade pode ser diferente, dependendo da maneira como a função *numpy.full()* é implementada na biblioteca *numpy*. De qualquer modo, esta função não tem um impacto significativo no *runtime*, como iremos observar na próxima secção.

**cycle**: Contém *nested loops* que iteram sobre o tamanho da rede três vezes. Dentro do ciclo mais interno, existem operações de tempo constante. Assim, a complexidade temporal global é O(*size*^3).

**ising**: Contém um ciclo que itera sobre o número de ciclos. Dentro do ciclo chama a função *cycle*, que tem uma complexidade temporal de O(*size*^3). Assim, a complexidade temporal global desta função é O(*num\_cycles* \* *size*^3).

**curie\_temp**: Itera sobre o vetor de temperaturas e chama a função *ising* para cada temperatura. Uma vez que a função *ising* tem uma complexidade temporal de O(*num\_cycles* \* *size*^3), a complexidade temporal global desta função é O(*num\_cycles* \* *size*^3 \* *pointsT*), onde *pointsT* é o tamanho do vetor de temperaturas.

**plotting\_curie\_temp:** Chama a função *curie\_temp* e faz um gráfico com os resultados da mesma. A complexidade temporal é dominada por esta função, logo, a complexidade temporal global é a mesma, O(*num\_cycles* \* *size*^3 \* *pointsT*).

**hysteresis**: Itera sobre os vetores de campos e temperaturas e chama a função *ising* para cada par de valores. Semelhante a *curie\_temp*, a complexidade temporal global desta função é O(*num\_cycles* \* *size*^3 \* *pointsT* \* *pointsH*), onde *pointsH* é o tamanho do vetor de campos.

**plotting\_hysteresis**: Chama a função *hysteresis* e faz um gráfico com os resultados da mesma. A complexidade temporal é dominada por esta função, logo, a complexidade temporal global é a mesma, O(*num\_cycles* \* *size*^3 \* *pointsT* \* *pointsH*).

Globalmente, os fatores dominantes que afetam a complexidade temporal neste código são o tamanho da rede (size^3) e o número de ciclos (num\_cycles).

**Profiler measurements** - NEVES

What the group takes from that information – NEVES

**Otimização** RUI

What were and why the optimization improvements were implemented

How much did those code changes improve the execution time

Seguindo a estrutura do código:

1. Na função **transitionFunctionValues** passaram a ser utilizadas *linked lists* para inicializar o *array* com os valores da função de transição. Não tem impacto no *runtime*, mas consideramos que é uma melhor implementação.
2. Na função **ising** foram feitas pequenas alterações para facilitar a leitura do código, sem impacto no *runtime*.
3. A função **curie\_temp**

Mais a frente falar do jit na função cycle

**Meta Parâmetros** RUI

Justify what parameters and meta parameters will be used in the final simulation

**Results**

Report of the results obtained with v1 code version for the more demanding input size - NEVES

Analyze and compare them with the expected results - NEVES

Short information on what were the main difficulties of the project, and what could eventually be improved - NEVES