|  |  |
| --- | --- |
|  |  |
| Projecto de Simulação |  |
|  |  |
|  | 07-04-2025Computação e Programação (LEMat,LEQ) Departamento de Matemática, IST Constança Dionísio, nº113878, LEQ  Beatriz Sousa, nº113898, LEQ  Constança Quaresma, nº 114580, LEMat  Miguel Murtinheira, nº113893, LEMat |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | |  | |
|  | Índice [Capa 1](#_Toc194680525)  [Índice 2](#_Toc194680528)  [Objetivo 3](#_Toc194680529)  [*Módulos Python* 4](#_Toc194680530)  [Módulo main.py 5](#_Toc194680531)  [Módulo particula.py 6](#_Toc194680532)  [Módulo espaco\_virtual.py 7](#_Toc194680533)  [Módulo resultado\_simulacao.py 9](#_Toc194680534) | | |  | |
|  |  | |  | |  | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | |  | |
|  | ObjetivoEste documento tem como propósito explicar como foi feito a simulação de uma produção de um certo produto químico em função do tempo, utilizando a simulação discreta estocástica, implementada em Python. | | |  | |
|  |  | |  | |  | |

Módulos Python  
  
Estes são os módulos utilizados para o simulador da reação:

main.py   
– módulo principal de arranque

particula.py  
 – módulo de gestão de partícula

espaco\_virtual.py   
– módulo de gestão do recipiente das reações

reacoes.py  
 – módulo que contém uma classe usada nas reações  
  
resultado\_simulacao.py  
 - módulo que mostra o resultado da simulação

Módulo main.py  
  
Este é módulo principal da aplicação e é o ponto de partida da mesma.  
  
A primeira parte declara as variáveis principais e inicializa o recipiente.

Depois define duas funções auxiliares que serão utilizadas mais à frente no código e que servem para adicionar e remover partículas.   
  
De seguida adiciona as partículas P e Q e calcula o número de iterações que vai ser utilizado durante o processo de reações.  
  
Por cada iteração usa a classe recipiente para a recolha de estatísticas, move as partículas e verifica as reações que, entretanto, ocorreram.

Ainda na iteração verifica se as partículas P e Q atingiram o limiar inferior definido e nesse caso adiciona mais partículas. Verifica também se as partículas R atingira o limiar máximo e no caso de isso acontecer retira seletivamente partículas R.

Por último, mostra os resultados da simulação utilizando a classe **ResultadoSimulacao**.

Módulo particula.py  
  
Este é módulo contém uma classe que se chama **Atomo** e que gere a criação e movimento da partícula.  
  
Esta classe **Atomo** tem quatro variáveis e define uma função, **mover**, e um método construtor.  
  
As variáveis servem para controlar a posição e velocidade da partícula.

O construtor ( **\_\_init\_\_**) é chamado durante a inicialização da classe e tem por missão definir, aleatoriamente, os vetores da posição e velocidade das partículas.

A função **mover** é utilizada para deslocar a partícula em função da velocidade.  
Verifica primeiro se a partícula não atingiu um dos limites do recipiente, se tal aconteceu inverte o vetor de velocidade da partícula fazendo-a mover-se na direção contrária.

Módulo espaco\_virtual.py  
  
Este é módulo é constituído por uma classe **Recipiente**, que representa uma grelha   
onde ocorrem as reações que são simuladas.

Esta classe contém variáveis internas, o método construtor e mais quatro funções que serão descritas adiante.

As variáveis são usadas para armazenar dados estatísticos sobre as partículas e é definido uma variável como array que serve para armazenar as partículas.

Duas funções, **adicionar\_particular** e **remover\_particulas**, servem para adicionar partículas P ou Q (adiciona ao array) e remover as partículas R, esta última apenas remove se a partícula se encontra no quadrante esquerdo anterior, verifica isso analisando o vetor posição da partícula.

A função **mover\_particulas** percorre o array de partículas e invoca, para cada uma delas, o método **mover**().

A função **recolher\_estatisticas**, serve para armazenar a contagem das partículas consoante o tipo, se é P, Q ou R.

A função **verifica\_reacoes**, é a função que verifica se as partículas estão numa mesma posição da grelha de forma a reagirem umas com as outras.  
Começa por percorrer a grelha guardando num array as partículas que encontram em cada posição.

De seguida vai separar as partículas da mesma posição consoante o tipo em array diferentes.  
Percorre então o array das partículas P e Q em paralelo e vai criar uma molécula R e eliminar as partículas P e Q originais.

Por último, percorre o array das moléculas R e por cada duas moléculas R vai criar duas partículas P e duas partículas Q, eliminando as moléculas R durante o processo.   
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Módulo reacoes.py  
  
Este é módulo contém apenas uma classe auxiliar, **ReacaoHolder**, usada para armazenar temporariamente um objeto que guarda as partículas em determinada posição da grelha.  
  
Tem apenas duas variáveis que representam a posição da grelha e um array que serve para guardar as partículas que se encontrar nessa posição.

Módulo resultado\_simulacao.py  
  
Este é módulo contém uma classe, **ResultadoSimulacao**, que serve para desenhar um gráfico e mostrar as estatísticas finais da simulação.

A classe usa a biblioteca **matplotlib**, e contém apenas um método, **desenhar\_resultado**, que desenha um gráfico que mostra a evolução das partículas P e Q e moléculas R ao longo do tempo da simulação.

Mostra também o número de partículas e moléculas criadas e consumidas durante a simulação.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Lorem Ipsum é simplesmente texto de preenchimento da indústria de impressão e tipografia. |
|  |