

Movimento de dois carros - MRUV

Mário Leite

Quando um aluno entra numa faculdade para cursar Física (seja Bacharelado ou Licenciatura) no primeiro semestre ele estuda, obrigatoriamente, a Mecânica Newtoniana; e o primeiro assunto tratado são as equações de movimento. Neste contexto o destaque é o **MRUV (Movimento Retilíneo Uniformemente Variado)** em que a velocidade de um objeto aumenta (ou diminui) de um valor constante. E um dos problemas interessantes é o caso de dois carros, partindo num mesmo momento de pontos distantes um do outro, na mesma direção, porém em sentidos opostos. É o clássico problema em que um carro **C1**, com velocidade inicial **vo1** e aceleração constantes **a1**, parte de uma cidade **A** para outra cidade **B** em uma estrada reta. Ao mesmo tempo da cidade **B** um carro **C2**, com velocidade inicial **vo2** e aceleração constante **a2**, parte da cidade **B** para a cidade **A** na mesma estrada, em outra pista. Então, o problema consiste em monitorar suas posições, suas velocidades a cada instante **t**, e mostrar o ponto em que eles se encontram (quando um passa pelo outro) na mesma estrada.

A análise da Física empregada neste programa é baseada nas equações da Mecânica Clássica de movimento de corpos em MRUV), sendo **x** o caminho percorrido pelos carros no tempo **t**.

1) Para o carro **C1** no movimento de **A** para **B**: $x_1 = vo_1.t + (1/2).a_1.t^2$

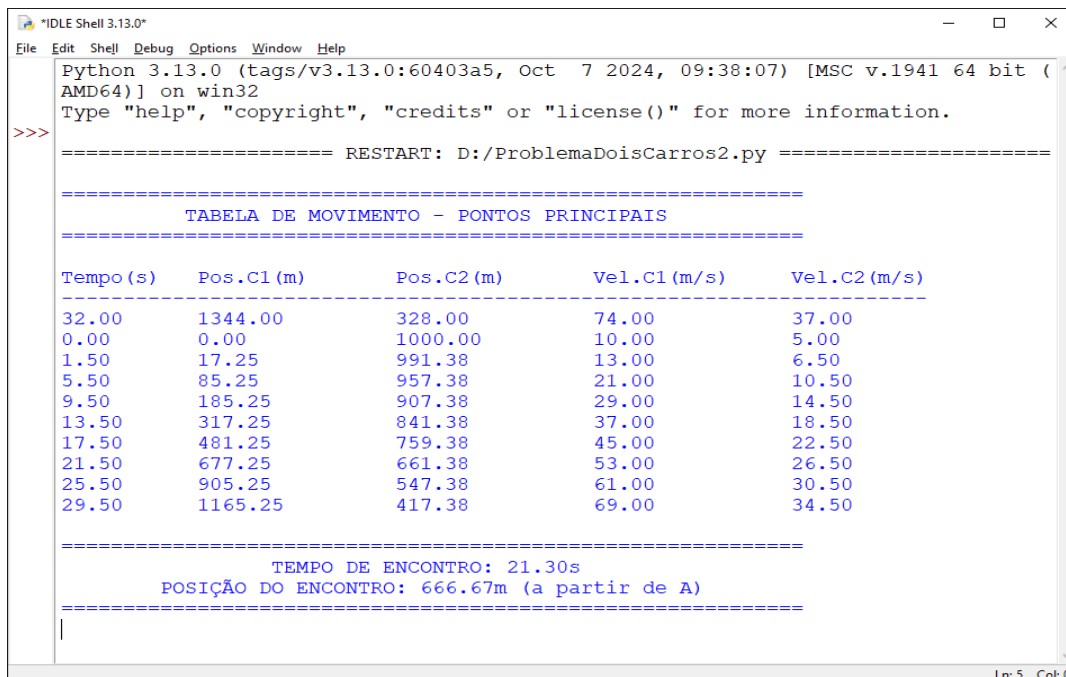
2) Para o carro **C2** no movimento de **B** para **A**: $x_2 = vo_2.t + (1/2).a_2.t^2$

Neste caso, para o carro **C2**, pode termos que: $x(t) = D - vo_2.t - (1/2).a_2.t^2$

Quando eles estão emparelhados ($x_1 = x_2$): $vo_1.t + (1/2).a_1.t^2 = D - vo_2.t - (1/2).a_2.t^2$

Resolvendo, encontramos uma equação do 2º Grau em **t**, cuja solução é a raiz quadrada positiva de **t** (tempo gasto no movimento para ocorrer o encontro dos carros). Assim, substituindo esse valor de **t** em qualquer uma das equações de movimento encontramos o ponto de encontro...

O programa "**ProblemaDoisCarros**" mostra na **figura 1** uma tabela com alguns valores desses movimentos, além da **figura 2** que apresenta os resultados em uma interface gráfica.



```
Python 3.13.0 (tags/v3.13.0:60403a5, Oct 7 2024, 09:38:07) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/ProblemaDoisCarros2.py =====

=====
TABELA DE MOVIMENTO - PONTOS PRINCIPAIS
=====
```

Tempo (s)	Pos.C1 (m)	Pos.C2 (m)	Vel.C1 (m/s)	Vel.C2 (m/s)
32.00	1344.00	328.00	74.00	37.00
0.00	0.00	1000.00	10.00	5.00
1.50	17.25	991.38	13.00	6.50
5.50	85.25	957.38	21.00	10.50
9.50	185.25	907.38	29.00	14.50
13.50	317.25	841.38	37.00	18.50
17.50	481.25	759.38	45.00	22.50
21.50	677.25	661.38	53.00	26.50
25.50	905.25	547.38	61.00	30.50
29.50	1165.25	417.38	69.00	34.50

```
=====
TEMPO DE ENCONTRO: 21.30s
POSIÇÃO DO ENCONTRO: 666.67m (a partir de A)
=====
```

Figura 1 - Tabela de saída dos indicadores dos movimentos dos carros

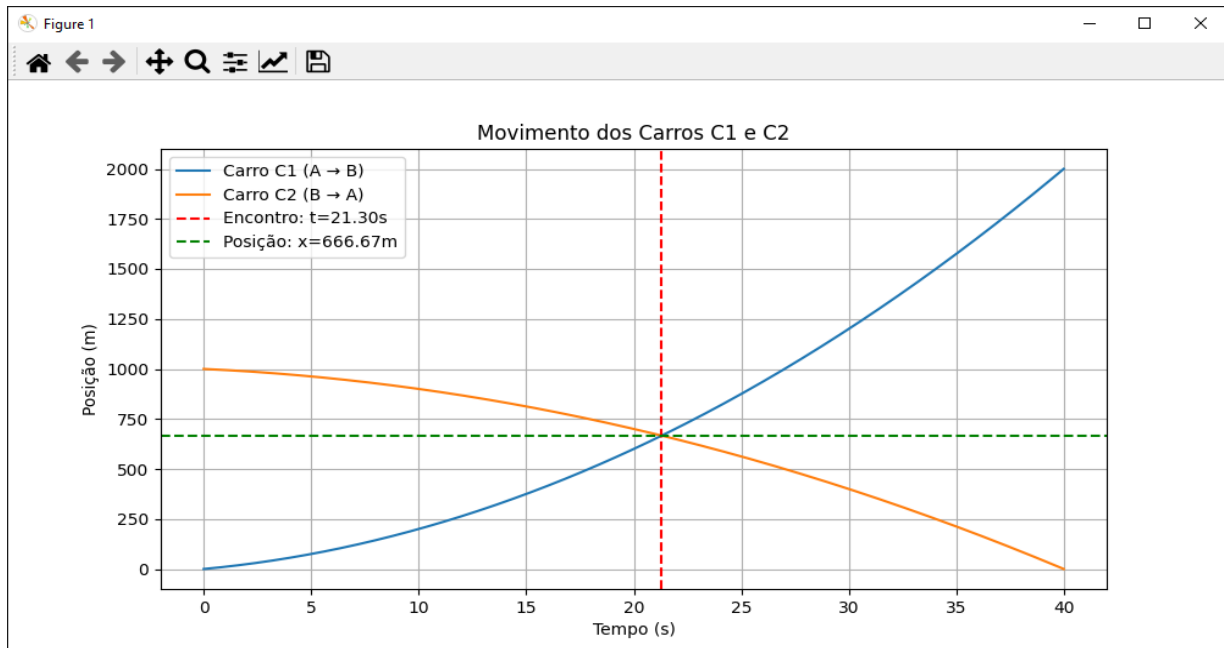


Figura 2 – Saída gráfica dos indicadores dos movimentos dos carros

```
'''
ProblemaDoisCarros.py
-----
Simula o clássico problema de dois carros em MRUV: O primeiro saindo de uma cidade A
em direção à uma cidade B; o outro partindo, ao mesmo tempo, de uma cidade B para a
cidade A (em sentidos opostos) numa mesma estrada, em pistas paralelas. O programa
mostras os valores de movimento: posição, velocidade e tempo até chegarem aos seus
respectivos destinos, inclusive o ponto em que um passa pelo outro (tempo e local do
encontro).
-----
'''

import numpy as np
import matplotlib.pyplot as plt
from tabulate import tabulate

def CalcularEncontro(vo1, a1, vo2, a2, D):
    a = (a1 + a2) / 2
    b = vo1 + vo2
    c = -D
    delta = b**2 - 4 * a * c
    if(delta < 0):
        return None, None
    tempoEncontro = (-b + np.sqrt(delta)) / (2 * a)
    posiEncontro = vo1 * tempoEncontro + 0.5 * a1 * tempoEncontro**2
    return tempoEncontro, posiEncontro

#-----
def SimularMovimento(vo1, a1, vo2, a2, D, dt=0.5): #aumentamos 'dt' para menos linhas
    resultado = CalcularEncontro(vo1, a1, vo2, a2, D)
    if(resultado is None):
        print("⚠ Não foi possível calcular o encontro")
        return None
    tempoEncontro, posiEncontro = resultado
    tempoMax = max(tempoEncontro * 1.5, 30) #limita o tempo de simulação
    tempos = np.arange(0, tempoMax + dt, dt)
    LstDados = []
```

```

for t in tempos:
    x1 = vo1 * t + 0.5 * a1 * t**2
    x2 = D - (vo2 * t + 0.5 * a2 * t**2)
    v1 = vo1 + a1 * t
    v2 = vo2 + a2 * t
    LstDados.append([t, x1, x2, v1, v2])

return tempos, LstDados, tempoEncontro, posiEncontro
=====
#Programa principal
vo1, a1 = 10, 2
vo2, a2 = 5, 1
D = 1000

#Simulação
resultado = SimularMovimento(vo1, a1, vo2, a2, D)
if(resultado is None):
    print("⊗ Erro na simulação!")
else:
    tempos, LstDados, tEncontro, xEncontro = resultado
    print("\n" + "="*60)
    print("TABELA DE MOVIMENTO - PONTOS PRINCIPAIS".center(60))
    print("="*60)
    #Para visualização na tabela considera apenas 10 pontos estratégicos + o encontro
    LstPontosChave = [0] + list(range(3, len(LstDados), len(LstDados)//8)) + [-1]
    LstPontosChave = sorted(set(LstPontosChave))

    #Adiciona o ponto de encontro se não estiver na lista
    idx_encontro = min(range(len(LstDados)), key=lambda i: abs(LstDados[i][0]-tEncontro))
    if(idx_encontro not in LstPontosChave):
        LstPontosChave.append(idx_encontro)
        LstPontosChave.sort()

    LstTabela = [LstDados[i] for i in LstPontosChave if i < len(LstDados)]

    #Imprime a tabela diretamente, sem usar tabulate
    print("\n{:<10} {:<15} {:<15} {:<15} {:<15}".format(
        "Tempo(s)", "Pos.C1 (m)", "Pos.C2 (m)", "Vel.C1 (m/s)", "Vel.C2 (m/s)"))
    print("-"*70)
    for linha in LstTabela:
        print("{:<10.2f} {:<15.2f} {:<15.2f} {:<15.2f} {:<15.2f}".format(*linha))

    print("\n" + "="*60)
    print(f"TEMPO DE ENCONTRO: {tEncontro:.2f}s".center(60))
    print(f"POSIÇÃO DO ENCONTRO: {xEncontro:.2f}m (a partir de A)".center(60))
    print("="*60)

    #Saída gráfica dos movimentos
    plt.figure(figsize=(10,5))
    plt.plot(tempos, [d[1] for d in LstDados], label="Carro C1 (A→B)")
    plt.plot(tempos, [d[2] for d in LstDados], label="Carro C2 (B→A)")
    plt.axvline(tEncontro,color='r',linestyle='--',label=f"Encontro: {tEncontro:.2f}s")
    plt.axhline(xEncontro,color='g',linestyle='--',label=f"Posição: {xEncontro:.2f}m")
    plt.xlabel("Tempo (s)"); plt.ylabel("Posição (m)")
    plt.title("Movimento dos Carros")
    plt.legend()
    plt.grid()
    plt.show()

#Fim do programa "ProblemDoisCarros" -----

```