Trabalho 1 - Dinâmica de Sistemas II

Grupo 8

Pedro Pastro Xavier de Oliveira - 10823390

Ludmylla Helena Camielli de Oliveira - 10771611

Maio, 2023

Sumário

Lista de Figuras

1 Introdução e Objetivos

O problema do pêndulo duplo é foco de pesquisa a muito tempo e é especialmente relevante para nós no contexto de modelagem de sistemas dinâmicos.

O objetivo deste relatório é apresentar os conceitos que englobam o problema e os resultados computacionais obtidos.

2 Revisão Teórica

Primeiramente, é necessário abordar as variáveis que serão fundamentais para o problema

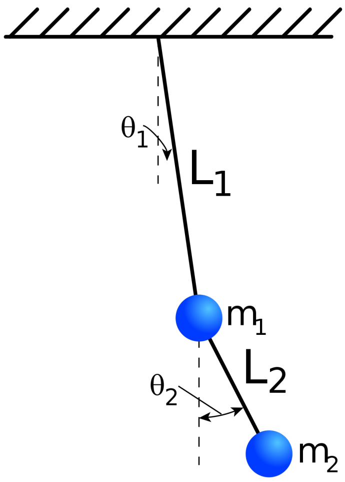

Figura 1: Modelo de pêndulo-duplo

Nesta situação, as massas, m_1 e m_2, bem como o comprimento das hastes rígidas L_1 e L_2, serão constantes em todo o período. Os ângulos de inclinação das hastes, θ_1 e θ_2, serão funções do tempo, e serão determinadas posteriormente para a resolução do exercício.

Podemos determinar que a Energia Cinética T do sistema será dada por:

$$T = \frac{m_1 + m_2}{2}(L_1^2 \dot{\theta_1}^2) + m_2 L_1 L_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + \frac{m_2}{2}(L_2^2 \dot{\theta_2}^2)$$

Já a Energia Potencial U pode ser escrita como:

$$U = -(m_1 + m_2) g L_1 \cos(\theta_1) - m_2 g L_2 \cos(\theta_2)$$

Por sua vez, a Lagrangiana L do sistema é uma *EDO* que une as duas equações na forma:

$$L = T - U$$

O método de resolução da Lagrangiana para cada massa é através do conjunto de equações:

$$\frac{\partial L}{\partial \theta_{1,2}} - \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta_{1,2}}} = 0,$$

que deve ser resolvido em termos de $\ddot{\theta_1}$ e $\ddot{\theta_2}$. A partir disso, teremos duas equações diferenciais de segunda ordem. Podemos reescrevê-las para utilizar o método de Runge-Kutta para obter 4 equações diferenciais de primeira ordem:

$$w_1 = \dot{\theta_1}; \dot{w_1} = \ddot{\theta_1}$$

$$w_2 = \dot{\theta_2}; \dot{w_2} = \ddot{\theta_2}$$

Que finalmente, podemos descrever como funções de θ_1, θ_2, w_1 e w_2, simbolicamente como:

$$\begin{cases} \dot{w_1} = f_1(\theta_1, \theta_2, w_1, w_2) \\ \dot{w_2} = f_2(\theta_1, \theta_2, w_1, w_2) \\ \dot{\theta_1} = w_1 = f_3(\theta_1, \theta_2, w_1, w_2) \\ \dot{\theta_2} = w_2 = f_4(\theta_1, \theta_2, w_1, w_2) \end{cases}$$

3 Resultados Computacionais

A partir de métodos computacionais utilizando os módulos *Scipy* e *Sympy* de programação em *Python*, conseguimos determinar os valores de $\theta_{1,2}$ com respeito ao tempo e as posições $(x_{1,2}, y_{1,2})$ de cada massa no período de tempo determinado dadas as condições iniciais do sistema. Como apenas a massa 2 está submetida a mais de 1 grau de liberdade, apenas os resultados desta foram considerados de interesse para o escopo deste relatório.

Num intervalo de 30 segundos, os parâmetros do sistema são:

- $g = 9,81\, m/s^2$
- $m_1 = 0,7\, kg$; $m_2 = 1\, kg$
- $L_1 = L_2 = 0,5\, kg$

Para os ângulos iniciais $\theta_1 = \pi/4$ e $\theta_2 = \pi/2$, foram encontrados os seguintes valores:

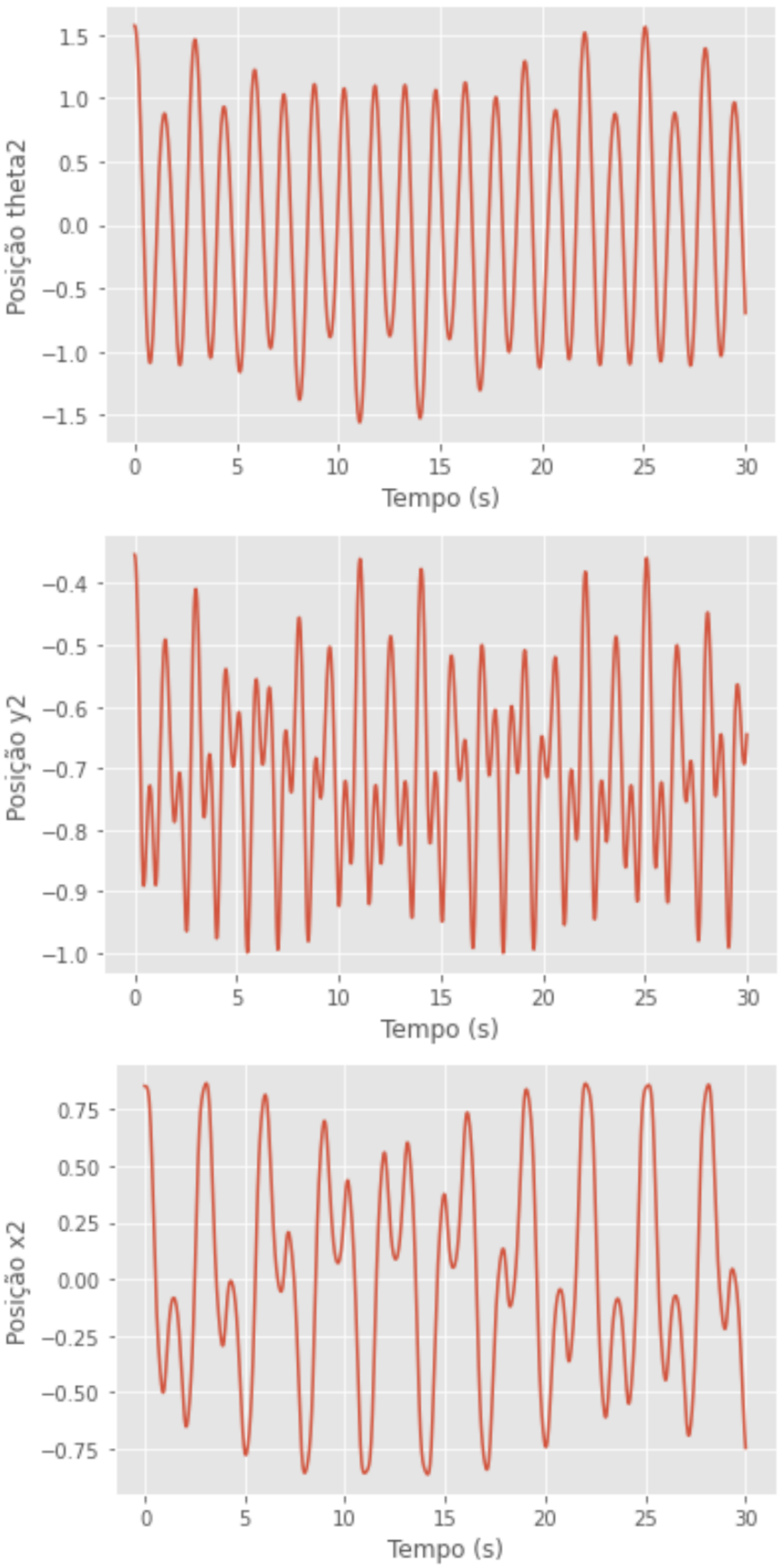

Figura 2: Resultados computacionais 1

Alterando os ângulos iniciais para $\theta_1 = \pi/2$ e $\theta_2 = 3\pi/4$:

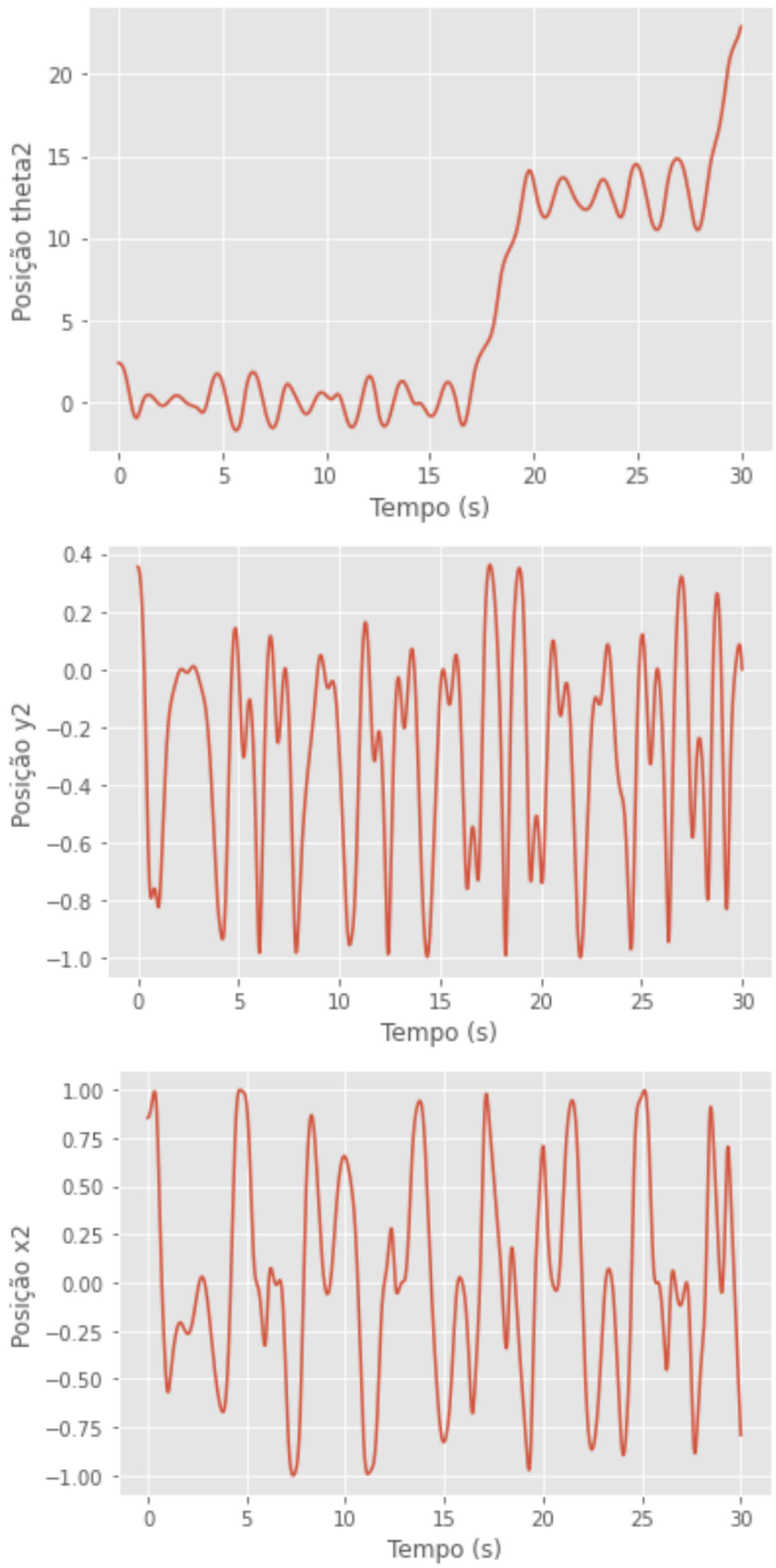

Figura 3: Resultados computacionais 2

4 Anexo - código-fonte

```
import numpy as np
import sympy as sp
import matplotlib.pyplot as plt
from scipy.integrate import odeint

%matplotlib inline
matplotlib.style.use('ggplot')


def vct(S, t, g, m1, m2, L1, L2):
  the1, z1, the2, z2 = S
  return [w1(z1),
          w1_d(t, g, m1, m2, L1, L2, the1, the2, z1, z2),
          w2(z2),
          w2_d(t, g, m1, m2, L1, L2, the1, the2, z1, z2)]

def coord(t, the1, the2, L1, L2):
  return (L1*np.sin(the1),
          -L1*np.cos(the1),
          L1*np.sin(the1) + L2*np.sin(the2),
          -L1*np.cos(the1) - L2*np.cos(the2))


# Definicao de variaveis do problema
t, g, m1, m2, L1, L2 = sp.symbols('t g m1 m2 L1 L2')
the1 = sp.Function(r'\theta_1')(t)
the2 = sp.Function(r'\theta_2')(t)

# Definicao de velocidade e aceleracao angulares
the1_d = sp.diff(the1, t)
the2_d = sp.diff(the2, t)
the1_dd = sp.diff(the1, t, t)
the2_dd = sp.diff(the2, t, t)

# Definicao das Energias Cinetica, Potencial e Lagrangiana
# Cinetica
T = (m1+m2)*(L1**2+the1_d**2)/2 +
```

```
        m2*L1*L2*the1_d*the2_d*sp.cos(the1-the2) +
        (m2*L2**2*the2_d**2)/2
    # Potential
    U = -(m1+m2)*g*L1*sp.cos(the1)
        - m2*g*L2*sp.cos(the2)
    # Lagrangeana
    L = T-U

    # Definicao das Equacoes de Lagrange para cada massa
    Leq1 = sp.diff(L, the1) - sp.diff(sp.diff(L, the1_d), t)
    Leq2 = sp.diff(L, the2) - sp.diff(sp.diff(L, the2_d), t)

    # Resolucao do sistema de Equacoes Lagrangianas
    sols = sp.solve([Leq1, Leq2], (the1_dd, the2_dd))

    # Adequando para integracao numerica de Runge-Kutta de quarta ordem
    # Definicao da funcao w
    w1 = sp.lambdify(the1_d, the1_d)
    w2 = sp.lambdify(the2_d, the2_d)
    w1_d = sp.lambdify((t,g,m1,m2,L1,L2,the1,the2,the1_d,the2_d), sols[
        the1_dd])
    w2_d = sp.lambdify((t,g,m1,m2,L1,L2,the1,the2,the1_d,the2_d), sols[
        the2_dd])

    # Definicao dos parametros
    t = np.linspace(0, 30, 1001)
    g = 9.81
    m1 = 0.7
    m2 = 1
    L1 = 1/2
    L2 = 1/2

    # Metodo de integracao numerica com scipy
    ans = odeint(vct, y0=[np.pi/2, 0, 3*np.pi/4, 0], t=t, args=(g,m1,m2,L1,
        L2))
    the1 = ans.T[0]
    the2 = ans.T[2]

    # Encontrar as coordenadas do sistema
```

```
x1, y1, x2, y2 = coord(t, ans.T[0], ans.T[2], L1, L2)

# Angulo theta_2 em funcao do tempo
plt.plot(t, the2)
plt.xlabel('Tempo (s)')
plt.ylabel('Posicao theta2')

# Posicao de y2 em funcao do tempo
plt.plot(t,y2)
plt.xlabel('Tempo (s)')
plt.ylabel('Posicao y2')

# Posicao de x2 em funcao do tempo
plt.plot(t,x2)
plt.xlabel('Tempo (s)')
plt.ylabel('Posicao x2')

# Fazer animacao gif
def animate(i):
  ln1.set_data([0, x1[i], x2[i]], [0, y1[i], y2[i]])

fig, ax = plt.subplots(1,1, figsize=(8,8))
ax.set_facecolor('k')
ax.get_xaxis().set_ticks([])    # enable this to hide x axis ticks
ax.get_yaxis().set_ticks([])    # enable this to hide y axis ticks
ln1, = plt.plot([], [], 'ro--', lw=3, markersize=8)
ax.set_ylim(-4,4)
ax.set_xlim(-4,4)
ani = animation.FuncAnimation(fig, animate, frames=1000, interval=50)
ani.save('pen.gif',writer='pillow',fps=25)
```

5 Bibliografia

- ARNOLD, Francisco José et al. Estudo do amortecimento do pêndulo simples: uma proposta para aplicação em laboratório de ensino. **Revista Brasileira de Ensino de Física**, v. 33, p. 4311-4311, 2011.