Pendulum Project

Lee Farrugia

1st December 2022

1 Abstract

2 Introduction & Theoretical Background

3 Materials & Methods

3.1 Language and Packages

Python 3.10.8, Numpy, Sympy, Scipy, Matplotlib.pyplot, Scipy.integrate, Scipy.fft, Matplotlib.patches

3.2 Methodology

1.

4 Results & Discussion

5 Conclusion

6 References

7 Appendix

```
import numpy as np
import scipy as sc
import matplotlib.pyplot as plt
from math import sin, cos
from sympy import *
from scipy.integrate import odeint
from scipy.fft import fft, fftfreq
from matplotlib.patches import Circle

x,y,r,t,m,v,i,g = symbols('x,y,r,t,m,v,i,g')
theta = Function('theta')(t)
```

```
x = r * sin(theta)
y = (-1*r) * cos(theta)
theta_dot = diff(theta, t)

vx = diff(x,t)
vy = diff(y,t)

vt = vx**2 + vy**2
T_rec = 0.5* m* v.subs(v, vt)
T_rot = 0.5* i* diff(theta,t)**2

T = T_rec + T_rot
U = m*g*r*(1 - cos(theta))
L = T - U

Q = diff(diff(L, theta_dot),t) - diff(L, theta)

def ode_func(theta,t,m,g,r,i):
    #theta
    theta1=theta[0]
    #theta_dot
    dtheta1_dt=theta[1]
    #second ode
    dtheta2_dt = (-m*g*r*(np.sin(theta1)))/(i+(m*(r**2)))
    dtheta_dt = [dtheta1_dt, dtheta2_dt]
    return dtheta_dt


g = 9.81 # acceleration due to the gravity
i = 0.025
r = 0.5 # radius of pendulum
m = 1 # mass
l = 1 # length of pendulum

# initial conditions
theta_0 = [(np.pi/2),0, (np.pi/2), (np.pi/2), (np.pi/4), 0, (np.pi/4)]
# time plot
t = np.linspace(0, 10, 1000)

# solving the ode
for a in range(len(theta_0)-1):
    theta = odeint(ode_func,theta_0[a:a+2],t,args=(m,g,r,i))

    plt.figure(figsize=(7.5, 10.5))
    plt.rcParams['font.family'] = 'STIXGeneral'
    plt.rcParams['mathtext.fontset'] = 'stix'
    plt.rcParams['font.size'] = 12
    plt.rcParams['font.weight'] = 'normal'
```

```
plt.minorticks_on()
plt.grid(visible=True, which='major', linestyle='-')
plt.grid(visible=True, which='minor', linestyle='--')
plt.xlim(0,10)

plt.plot(t,theta[:,0],'--', color='k',
↪  label=r'$\dot{\mathrm{\theta}}$')
plt.plot(t,theta[:,1], color='k', label=r'$\mathrm{\ddot{\theta}}$')
plt.xlabel('t/s')
plt.ylabel(r'$\Delta \theta$/rads')
plt.title(r'A graph of the change in $\mathrm{\theta}$ in time')

plt.legend()
plt.tight_layout()
plt.savefig(f'plots/Plot 1.{a+1}.png', dpi=800)
plt.close()

# Fourier transformation
# Sampling space = Frequency * total time
N = 100 * 10
# theta
x = theta[:,0]
# theta_dot
y = theta[:,1]
# Periodic Time
T = 1/N
yf = fft(y)
xf = fftfreq(N, T)[:N//2]

# Plotting conditions
plt.figure(figsize=(7.5, 10.5))
plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.size'] = 12
plt.rcParams['font.weight'] = 'normal'
plt.minorticks_on()
plt.grid(visible=True, which='major', linestyle='-')
plt.grid(visible=True, which='minor', linestyle='--')
# Plotting the data
plt.plot(xf, 2/N * np.abs(yf[0:(N//2)]), 'k')
plt.xlabel('Frequency')
plt.ylabel(r'$\Delta \theta$/rads')
plt.title('Fourier transformation of the Single Pendulum')
plt.tight_layout()
# Saving the figure
plt.savefig(f'plots/Plot 2.{a+1}.png', dpi=800)
plt.close()
```

```
plt.figure()

theta = odeint(ode_func,theta_0[0:2],t,args=(m,g,r,i))
theta_a = theta[:,0]
x = r * np.sin(theta_a)
y = (-1*r) * np.cos(theta_a)

tmax, dt = 10, 0.01
t1 = np.arange(0, tmax+dt, dt)

r = 0.05
trail_secs = 1
max_trail = int(trail_secs / dt)

def make_plot1(i):
    ax.plot([0, x[i]], [0, y[i]], lw=2, c='k')
    c0 = Circle((0, 0), r/2, fc='k', zorder=10)
    c1 = Circle((x[i], y[i]), r, fc='b', ec='b', zorder=10)
    ax.add_patch(c0)
    ax.add_patch(c1)

    ns = 20
    s = max_trail // ns

    for j in range(ns):
        imin = i - (ns-j)*s
        if imin < 0:
            continue
        imax = imin + s + 1
        alpha = (j/ns)**2
        ax.plot(x[imin:imax], y[imin:imax], c='r', solid_capstyle='butt',
        ↪  lw=2, alpha=alpha)

    ax.set_xlim(-l-r, l+r)
    ax.set_ylim(-l-r, l+r)
    ax.set_aspect('equal', adjustable='box')
    plt.axis('off')
    plt.savefig('frames1/_img{:04d}.png'.format(i//di), dpi=100)
    plt.cla()


fps = 10
di = int(1/fps/dt)
fig = plt.figure(figsize=(8.3333, 6.25), dpi=72)
ax = fig.add_subplot(111)

for i in range(0, t1.size-1, di):
    make_plot1(i)
```

```
x1,y1,x2,y2,l1,l2,v1,v2,t,m1,m2,i1,i2,g =
↪  symbols('x1,y1,x2,y2,l1,l2,v1,v2,t,m1,m2,i1,i2,g')
theta1 = Function('theta1')(t)
theta2 = Function('theta2')(t)

theta_dot1 = diff(theta1, t)
theta_dot2 = diff(theta2, t)

x1 = (l1/2)*sin(theta1)
x2 = l1*sin(theta1) + (l2/2)*sin(theta2)
y1 = (-1*(l1/2))*cos(theta1)
y2 = (-1*(l1*cos(theta1)))-(l2/2)*cos(theta2)

vx1 = diff(x1, t)
vx2 = diff(x2, t)
vy1 = diff(y1, t)
vy2 = diff(y2, t)

vt1 = vx1**2 + vy1**2
vt2 = vx2**2 + vy2**2

Trec1 = 0.5 * m1 * v1.subs(v1, vt1)
Trec2 = 0.5 * m2 * v1.subs(v2, vt2)

Trot1 = 0.5* i1* diff(theta1,t)**2
Trot2 = 0.5* i2* diff(theta2,t)**2

T = Trec1 + Trot1 +Trec2 + Trot2
U1 = ((m1*g*l1)/2)*1-cos(theta1)
U2 = ((m2*g)*((l1*(1-cos(theta1))))+((l2/2)*(1-cos(theta2))))
U = U1 + U2
L = T - U

Q1 = diff(diff(L,theta_dot1),t)-diff(L,theta1)
Q2 = diff(diff(L,theta_dot2),t)-diff(L,theta2)

l1, l2 = 1, 1
m1, m2 = 1, 1
g = 9.81

def ode_func_2(y, t, l1, l2, m1, m2):
    theta1, z1, theta2, z2 = y

    c, s = np.cos(theta1-theta2), np.sin(theta1-theta2)

    theta1dot = z1
```

```
    z1dot = (m2*g*np.sin(theta2)*c - m2*s*(l1*z1**2*c + l2*z2**2) -
    ↪  (m1+m2)*g*np.sin(theta1)) / l1 / (m1 + m2*s**2)
    theta2dot = z2
    z2dot = ((m1+m2)*(l1*z1**2*s - g*np.sin(theta2) + g*np.sin(theta1)*c)
    ↪  + m2*l2*z2**2*s*c) / l2 / (m1 + m2*s**2)
    return theta1dot, z1dot, theta2dot, z2dot

def total_E(y):
    th1, th1d, th2, th2d = y.T
    V = -(m1+m2)*l1*g*np.cos(th1) - m2*l2*g*np.cos(th2)
    T = 0.5*m1*(l1*th1d)**2 + 0.5*m2*((l1*th1d)**2 + (l2*th2d)**2 +
    ↪  2*l1*l2*th1d*th2d*np.cos(th1-th2))
    return T + V

tmax, dt = 10, 0.01
t = np.arange(0, tmax+dt, dt)
y0 = np.array([0, 0, np.pi/2, 0])

y = odeint(ode_func_2, y0, t, args=(l1, l2, m1, m2))

theta1, theta2 = y[:,0], y[:,2]

plt.close('all')

plt.figure(figsize=(7.5, 10.5))
plt.rcParams['font.family'] = 'STIXGeneral'
plt.rcParams['mathtext.fontset'] = 'stix'
plt.rcParams['font.size'] = 12
plt.rcParams['font.weight'] = 'normal'
plt.minorticks_on()
plt.grid(visible=True, which='major', linestyle='-')
plt.grid(visible=True, which='minor', linestyle='--')

plt.plot(t, theta1, color='k', label='Pendulum 1')
plt.plot(t, theta2,'--' , color='k', label='Pendulum 2')
plt.xlabel('t/s')
plt.ylabel(r'$\Delta \theta$/rads')
plt.title(r'A graph of the change in $\mathrm{\theta}$ in time')
plt.legend()
plt.savefig('plots/Plot 3.1.png', dpi=800)

plt.close('all')

x1 = l1 * np.sin(theta1)
y1 = -l1 * np.cos(theta1)
x2 = x1 + l2 * np.sin(theta2)
y2 = y1 - l2 * np.cos(theta2)
```

```
r = 0.05
trail_secs = 1
max_trail = int(trail_secs / dt)

def make_plot2(i):
    ax.plot([0, x1[i], x2[i]], [0, y1[i], y2[i]], lw=2, c='k')
    c0 = Circle((0, 0), r/2, fc='k', zorder=10)
    c1 = Circle((x1[i], y1[i]), r, fc='b', ec='b', zorder=10)
    c2 = Circle((x2[i], y2[i]), r, fc='r', ec='r', zorder=10)
    ax.add_patch(c0)
    ax.add_patch(c1)
    ax.add_patch(c2)

    ns = 20
    s = max_trail // ns

    for j in range(ns):
        imin = i - (ns-j)*s
        if imin < 0:
            continue
        imax = imin + s + 1
        alpha = (j/ns)**2
        ax.plot(x2[imin:imax], y2[imin:imax], c='r',
        ↪  solid_capstyle='butt', lw=2, alpha=alpha)

    ax.set_xlim(-l1-l2-r, l1+l2+r)
    ax.set_ylim(-l1-l2-r, l1+l2+r)
    ax.set_aspect('equal', adjustable='box')
    plt.axis('off')
    plt.savefig('frames2/_img{:04d}.png'.format(i//di), dpi=100)
    plt.cla()


fps = 10
di = int(1/fps/dt)
fig = plt.figure(figsize=(8.3333, 6.25), dpi=72)
ax = fig.add_subplot(111)

for i in range(0, t.size, di):
    make_plot2(i)

def fourier(theta1,theta2):
    # Fourier transformation
    # Sampling space = Frequency * total time
    N = 100 * 10
    # theta
    x1 = theta1
    # theta_dot
```

```
    y1 = theta1
    # Periodic Time
    T = 1/N
    yf1 = fft(y1)
    xf1 = fftfreq(N, T)[:N//2]

    # theta
    x2 = theta2
    # theta_dot
    y2 = theta2
    # Periodic Time
    T = 1/N
    yf2 = fft(y2)
    xf2 = fftfreq(N, T)[:N//2]

    plt.figure(figsize=(7.5, 10.5))
    plt.rcParams['font.family'] = 'STIXGeneral'
    plt.rcParams['mathtext.fontset'] = 'stix'
    plt.rcParams['font.size'] = 12
    plt.rcParams['font.weight'] = 'normal'
    plt.minorticks_on()
    plt.grid(visible=True, which='major', linestyle='-')
    plt.grid(visible=True, which='minor', linestyle='--')

    plt.plot(xf1, 2/N * np.abs(yf1[0:(N//2)]), 'k', label='Pendulum 1')
    plt.plot(xf2, 2/N * np.abs(yf2[0:(N//2)]), label='Pendulum 2')
    plt.xlabel('Frequency')
    plt.ylabel(r'$\Delta \theta$/rads')
    plt.title('Fourier transformation of the Double Pendulum')
    plt.legend()
    # Saving the figure
    plt.savefig('plots/Plot 3.2.png', dpi=800)
    plt.close()

fourier(theta1, theta2)
```