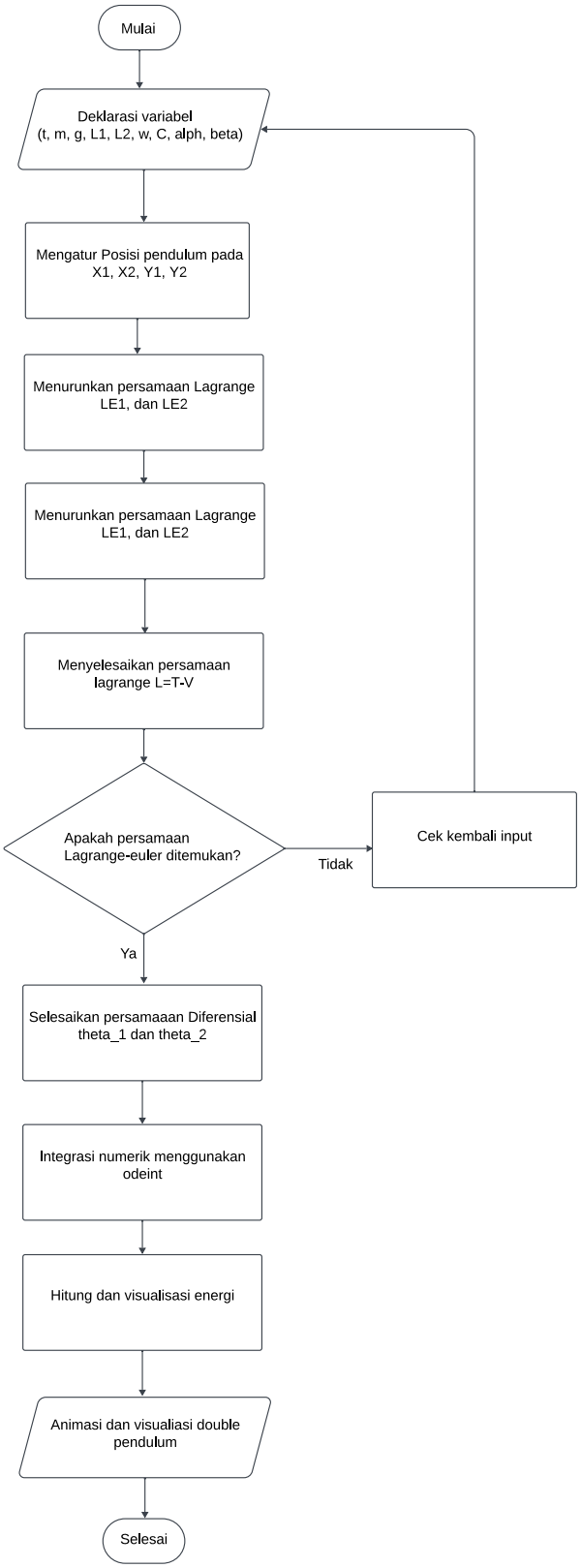
Praktikum Fisika Komputasi

Analisis Double Pendulum (Tugas 8)

Ramli Zhafran Amarillo (1227030027)

1.) Diagram Alir (*flowchart)*



2.) Penjelasan code

```
import numpy as np
import sympy as smp
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import PillowWriter
from IPython.display import HTML

# Menetukan variabel yang diperlukan untuk sympy
t, m, g, L1, L2, w, C, alph, beta = smp.symbols(r't m g L_1,
L_2 \omega C \alpha \beta')

the1, the2, = smp.symbols(r'\theta_1 \theta_2',
cls=smp.Function)

the1 = the1(t)
the1_d = smp.diff(the1, t)
the1_dd = smp.diff(the1_d, t)

the2 = the2(t)
the2_d = smp.diff(the2, t)
the2_dd = smp.diff(smp.diff(the2, t), t)

x1, y1, x2, y2 = smp.symbols('x_1, y_1, x_2, y_2',
cls=smp.Function)
x1 = x1(t, the1)
y1 = y1(t, the1)
x2 = x2(t, the1, the2)
y2 = y2(t, the1, the2)

x1 = smp.cos(w*t) + L1 * smp.sin(the1)
y1 = -L1 * smp.cos(the1)
x2 = smp.cos(w*t) + L1 * smp.sin(the1) + L2 * smp.sin(the2)
y2 = -L1 * smp.cos(the1) - L2 * smp.cos(the2)

smp.diff(x1, t)

vx1_f = smp.lambdify((t, w, L1, L2, the1, the2, the1_d,
the2_d), smp.diff(x1, t))
```

```
vx2_f = smp.lambdify((t, w, L1, L2, the1, the2, the1_d,
the2_d), smp.diff(x2, t))
vy1_f = smp.lambdify((t, w, L1, L2, the1, the2, the1_d,
the2_d), smp.diff(y1, t))
vy2_f = smp.lambdify((t, w, L1, L2, the1, the2, the1_d,
the2_d), smp.diff(y2, t))

# Rumus lagrange
T = 1/2 * (smp.diff(x1, t)**2 + smp.diff(y1, t)**2) + 1/2 * m
* (smp.diff(x2, t)**2 + smp.diff(y2, t)**2)
V = g * y1 + m * g * y2
L = T - V

# Persamaan Lagrange-Euler untuk theta1
LE1 = smp.diff(L, the1) - smp.diff(smp.diff(L, the1_d), t)
LE1 = LE1.simplify()

# Persamaan Lagrange-Euler untuk theta2
LE2 = smp.diff(L, the2) - smp.diff(smp.diff(L, the2_d), t)
LE2 = LE2.simplify()

LE1

LE2

# Menyelesaikan persamaan
sols = smp.solve([LE1, LE2], (the1_dd, the2_dd),
                 simplify=False, rational=False)

sols[the1_dd] #d^2 / dt^2 theta_1

a = LE1.subs([(smp.sin(the1-the2), the1-the2),
              (smp.cos(the1-the2), 1),
              (smp.sin(the1), the1),
              (the1, C*smp.cos(w*t)),
              (the2, C*alph*smp.cos(w*t)),
              (m, 1),
              (L2, L1),
              ]).doit().series(C, 0, 2).removeO().simplify()
b = LE2.subs([(smp.sin(the1-the2), the1-the2),
              (smp.cos(the1-the2), 1),
              (smp.cos(the1), 1),
```

```
                (smp.cos(the2), 1),
                (smp.sin(the1), the1),
                (smp.sin(the2), the2),
                (the1, C*smp.cos(w*t)),
                (the2, C*alph*smp.cos(w*t)),
                (m, 1),
                (L2, L1),
                ]).doit().series(C, 0, 2).removeO().simplify()

yeet = smp.solve([a.args[1], b.args[2]], (w, alph))

yeet[2][0]

yeet [0][0]

smp.limit(yeet[1][0].subs(C, beta/L1).simplify(), beta,
smp.oo)

# Mengubah persamaan eksak dan memasukkan ke dalam persamaan
Numerik
dz1dt_f = smp.lambdify((t, m, g, w, L1, L2, the1, the2,
the1_d, the2_d), sols[the1_dd])
dthe1dt_f = smp.lambdify(the1_d, the1_d)

dz2dt_f = smp.lambdify((t, m, g, w, L1, L2, the1, the2,
the1_d, the2_d), sols[the2_dd])
dthe2dt_f = smp.lambdify(the2_d, the2_d)

def dSdt(S, t):
    the1, z1, the2, z2 = S
    return [
           dthe1dt_f(z1),
           dz1dt_f(t, m, g, w, L1, L2, the1, the2, z1, z2),
           dthe2dt_f(z2),
           dz2dt_f(t, m, g, w, L1, L2, the1, the2, z1, z2),
       ]

t = np.linspace(0, 20, 1000)
g = 9.81
m=1
L1 = 20
L2 = 20
```

```
w = np.sqrt(g/L1)
ans = odeint(dSdt, y0 = [0, 0, 0, 0], t=t)

plt.plot(ans.T[0])

# Membuat Persamaan energi kinetik
def get_energy(w):
    t = np.linspace(0, 100, 2000)
    ans = odeint(dSdt, y0=[0.1, 0.1, 0, 0], t=t)
    vx1 =
vx1_f(t,w,L1,L2,ans.T[0],ans.T[2],ans.T[1],ans.T[3])
    vx2 =
vx2_f(t,w,L1,L2,ans.T[0],ans.T[2],ans.T[1],ans.T[3])
    vy1 =
vy1_f(t,w,L1,L2,ans.T[0],ans.T[2],ans.T[1],ans.T[3])
    vy2 =
vy2_f(t,w,L1,L2,ans.T[0],ans.T[2],ans.T[1],ans.T[3])
    E = 1/2 * np.mean(vx1**2 + vx2**2 + vy1**2 + vy2**2)
    return E

ws = np.linspace(0.4, 1.3, 100)
Es = np.vectorize(get_energy)(ws)

plt.plot(ws, Es)
plt.axvline(1.84775*np.sqrt(g/L1), c='k', ls='--')
plt.axvline(0.76536*np.sqrt(g/L1), c='k', ls='--')

#tautochrone
#plt.axvline(np.sqrt(np.pi*g**(-1/2)),  c='k', ls='--')
plt.grid()

t = np.linspace(0, 200, 20000)
g = 9.81
m = 1
L1 = 20
L2 = 20
w = ws[ws > 1][np.argmax(Es[ws > 1])]
ans = odeint(dSdt, y0=[0.1, 0.1, 0, 0], t=t)

def get_x0y0x1y1x2y2(t, the1, the2, L1, L2):
    return (np.cos(w*t),
            0*t,
```

```
            np.cos(w*t) + L1*np.sin(the1),
            -L1*np.cos(the1),
            np.cos(w*t) + L1*np.sin(the1) + L2*np.sin(the2),
            -L1*np.cos(the1) - L2*np.cos(the2),
           )

x0, y0, x1, y1, x2, y2 = get_x0y0x1y1x2y2(t, ans.T[0],
ans.T[2], L1, L2)

def animate(i):
    ln1.set_data([x0[::10][i], x1[::10][i], x2[::10][i]],
[y0[::10][i], y1[::10][i], y2[::10][i]])
    trail1 = 50         # Panjang jejak benda 1
    trail2 = 50         # Panjang jejak benda 2
    ln2.set_data(x1[::10][i:max(1, i-trail1):-1],
y1[::10][i:max(1, i-trail1):-1])
    ln3.set_data(x2[::10][i:max(1, i-trail2):-1],
y2[::10][i:max(1, i-trail2):-1])

fig, ax = plt.subplots(1, 1, figsize=(8, 8))
ax.set_facecolor('k')
ax.get_xaxis().set_ticks([])   # Menyembunyikan garis sumbu x
ax.get_yaxis().set_ticks([])   # Menyembunyikan garis sumbu y
ln1, = plt.plot([], [], 'ro--', lw=3, markersize=8)
ln2, = ax.plot([], [], 'o-', markersize=8, alpha=0.05,
color='cyan')   # Line for Earth
ln3, = ax.plot([], [], 'o-', markersize=8, alpha=0.05,
color='cyan')
ax.set_ylim(-44, 44)
ax.set_xlim(-44, 44)

# Animasi gerak double pendulum
ani = animation.FuncAnimation(fig, animate, frames=2000,
interval=50)
# ani.save('pen.gif', writer='pillow', fps=50)
HTML(ani.to_html5_video())
plt.show()
```

Penjelasan :

Pertama, bagian awal kode diimpor pustaka penting, seperti **numpy** untuk komputasi numerik, **sympy** untuk manipulasi aljabar simbolik, dan **matplotlib** untuk visualisasi. **scipy** menyediakan metode odeint untuk menyelesaikan persamaan diferensial. Setelah impor, beberapa variabel simbolik didefinisikan menggunakan sympy.symbols, seperti t (waktu), m (massa), g (gravitasi), dan L1, L2 (panjang lengan pendulum). Fungsi the1 dan the2 melambangkan sudut pendulum sebagai fungsi waktu t, yang nantinya digunakan untuk menghitung posisi pendulum dalam koordinat Kartesian (x1, y1, x2, y2). Dengan menggunakan **smp.diff**, derivatif dari sudut ini dihitung untuk mendapatkan kecepatan sudutnya.

Kemudian masukan **sols[the1_dd] #d^2 / dt^2 theta_1** kode solusi konsep mekanika Lagrange untuk menghitung gerak pendulum. Kode menghitung energi kinetik T dan energi potensial V dari sistem, dan Lagrangian L ditentukan sebagai selisih antara energi kinetik dan potensial. Kemudian, dengan menggunakan persamaan Lagrange-Euler, kode menghasilkan persamaan gerak untuk kedua sudut theta1 dan theta2. Setelah itu, kode menggunakan **smp.solve** untuk mencari solusi simbolik dari persamaan diferensial ini, yang merepresentasikan percepatan sudut theta1 dan theta2. Solusi ini diubah menjadi fungsi numerik menggunakan **lambdify**, sehingga bisa digunakan dalam simulasi.

Pada bagian terakhir mensimulasi gerak pendulum menggunakan metode odeint untuk menyelesaikan persamaan diferensial dengan kondisi awal tertentu. Setelah menghitung perubahan sudut terhadap waktu, kode membuat plot sederhana dari gerakan salah satu pendulum. Untuk animasi, kode mendefinisikan fungsi untuk memperbarui posisi pendulum pada setiap frame. **matplotlib.animation** digunakan untuk membuat animasi gerakan pendulum ganda, output akhirnya berupa pendulum dengan gerak chaos yang tidak teratur.

3.) Analisis Grafik dan Animasi

Grafik dari kode double pendulum menunjukkan bagaimana sudut dan kecepatan kedua pendulum berubah seiring waktu. Awalnya, gerakan tampak teratur, tetapi dengan cepat berubah menjadi pola yang kacau dan sulit diprediksi. Grafik ini menunjukkan bahwa gerakan double pendulum sangat sensitif terhadap perubahan kecil pada kondisi awal. Jika energi kinetik dan potensial diplot, kita akan melihat fluktuasi besar di antara keduanya, meskipun total energi tetap konstan, sesuai dengan hukum kekekalan energi.

Animasi dari kode ini memperlihatkan gerakan dinamis kedua pendulum yang sangat menarik. Pendulum sering kali bergerak bersama sebentar, lalu berubah menjadi pola gerakan yang kacau dan tidak terduga. Animasi ini menunjukkan bagaimana sedikit perubahan awal dapat menghasilkan pola gerakan yang sangat berbeda, mencerminkan sifat chaos dalam sistem ini. Dengan visualisasi ini, kita dapat memahami kompleksitas dan ketidakpastian gerakan double pendulum dengan lebih jelas.