

Moch. Aildho Candra Ramadhan

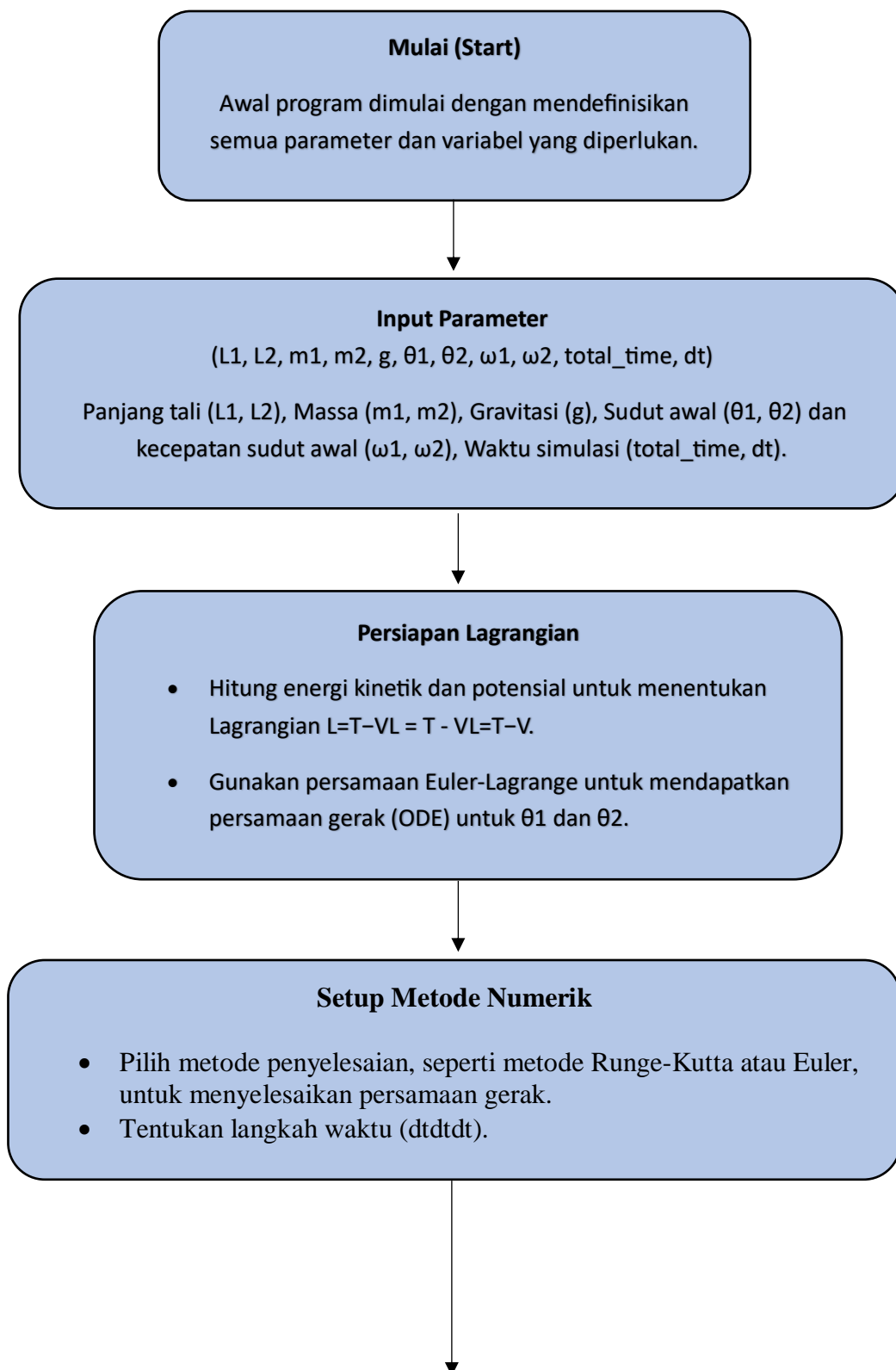
1227030020

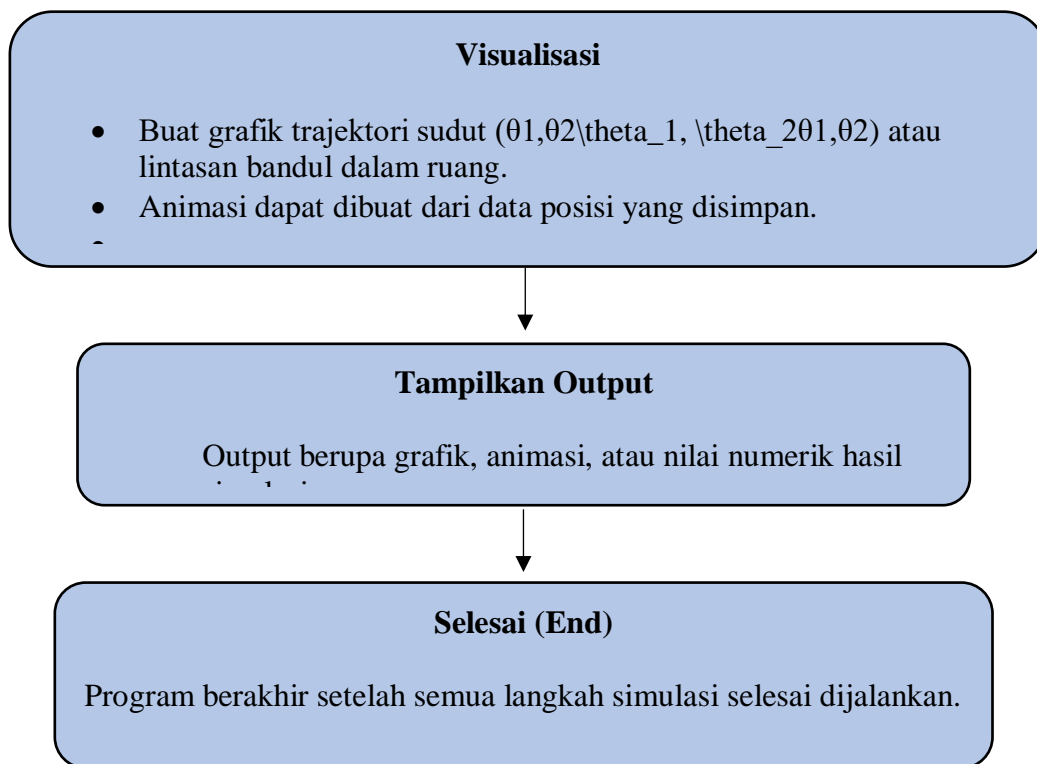
Materi-8 Analisis Double Pendulum

Soal 1 Buatlah diagram alir (flowchart) dari kode program double pendulum di atas!

Jawab :

Flowchart





Soal no 2 Jelaskan algoritma pada kode program di atas!

Jawab:

Struktur Algoritma

1. **Inisialisasi Parameter**

Input nilai-nilai awal untuk panjang tali, massa, gravitasi, kondisi sudut, dan langkah waktu.

2. **Hitung Persamaan Gerak**

Menentukan percepatan sudut berdasarkan persamaan Lagrangian.

3. **Iterasi Waktu**

Untuk setiap waktu t :

- Hitung nilai baru untuk θ_1, θ_2 dan ω_1, ω_2
- Simpan data.

4. **Konversi Posisi ke Koordinat Cartesian**

Transformasi nilai sudut menjadi posisi Cartesian (x, y).

5. **Visualisasi dan Output**

- Plot grafik posisi dan lintasan.
- Animasi gerakan pendulum.

Soal no 3 Analisis grafik dan animasi double pendulum!

Jawab:

- **Grafik Waktu terhadap Sudut (θ_1, θ_2)**

Grafik menunjukkan bagaimana sudut bandul pertama (θ_1) dan kedua (θ_2) berubah seiring waktu. Pada awal simulasi, pergerakan sudut biasanya teratur, namun dengan cepat menjadi kacau akibat sifat chaos dan sensitivitas terhadap kondisi awal.

- **Grafik Energi Sistem**

Energi total sistem ideal konstan, menunjukkan energi yang Fluktuasi energi kinetik dan potensial menggambarkan transfer energi antara bandulnya.

- **Grafik Fase**

Hubungan antara sudut (θ) dan kecepatan sudut (ω) menghasilkan pola fase yang kompleks dan tak beraturan, yang menunjukkan ciri khas dari sistem chaos.

- **Animasi Double Pendulum**

Animasi memperlihatkan transisi gerak dari teratur ke kacau, menggambarkan dinamika sistem dan distribusi energi secara visual pada simulasinya.

- **Pola Chaos**

Double pendulum menunjukkan sensitivitas tinggi terhadap perubahan kecil dalam kondisi awal, menghasilkan pola gerak yang sangat tidak stabil dan sulit diprediksi.

Lampiran Kode Pemrograman

```
# -*- coding: utf-8 -*-

"""ANALISIS DOUBLE PENDULUM Alldho

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1fiJiTBWF3_FkRPtD3PqnoseIbVOzDff
"""

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

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))
```

```
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()

sols = smp.solve([LE1, LE2], (the1_dd, the2_dd),

                  simplify=False, rational=False)

sols[the1_dd]

a = LE1.subs([(smp.sin(the1 - the2), the1 - the2),

              (smp.cos(the1 - the2), 1),

              (smp.cos(the1), 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.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)
```

```
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])

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='--')

```

```

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

    trail2 = 50

    ln2.set_data(x1[::10][i:max(1,i-trail1):-1], y1[::10][i:max(1,i-trail1):-1]) # Use the built-
in function max()

    ln3.set_data(x2[::10][i:max(1,i-trail2):-1], y2[::10][i:max(1,i-trail2):-1]) # Use the built-
in function max()

fig, ax = plt.subplots(1, 1, figsize=(8, 8))

ax.set_facecolor('k')

ax.get_xaxis().set_ticks([])

ax.get_yaxis().set_ticks([])

ln1, = ax.plot([], [], 'ro--', lw=3, markersize=8)

ln2, = ax.plot([], [], 'ro-', markersize=8, alpha=0.05, color='cyan')

ln3, = ax.plot([], [], 'ro-', markersize=8, alpha=0.05, color='cyan')

```

```
ax.set_ylim(-44, 44)
```

```
ax.set_xlim(-44, 44)
```

```
ani = animation.FuncAnimation(fig, animate, frames=2000, interval=50)
```

```
HTML(ani.to_html5_video())
```

Link google collab

https://colab.research.google.com/drive/1fiJiTBWF3_FkRPtD3PqnoselbVOzDff#scrollTo=StElVj-_oz_