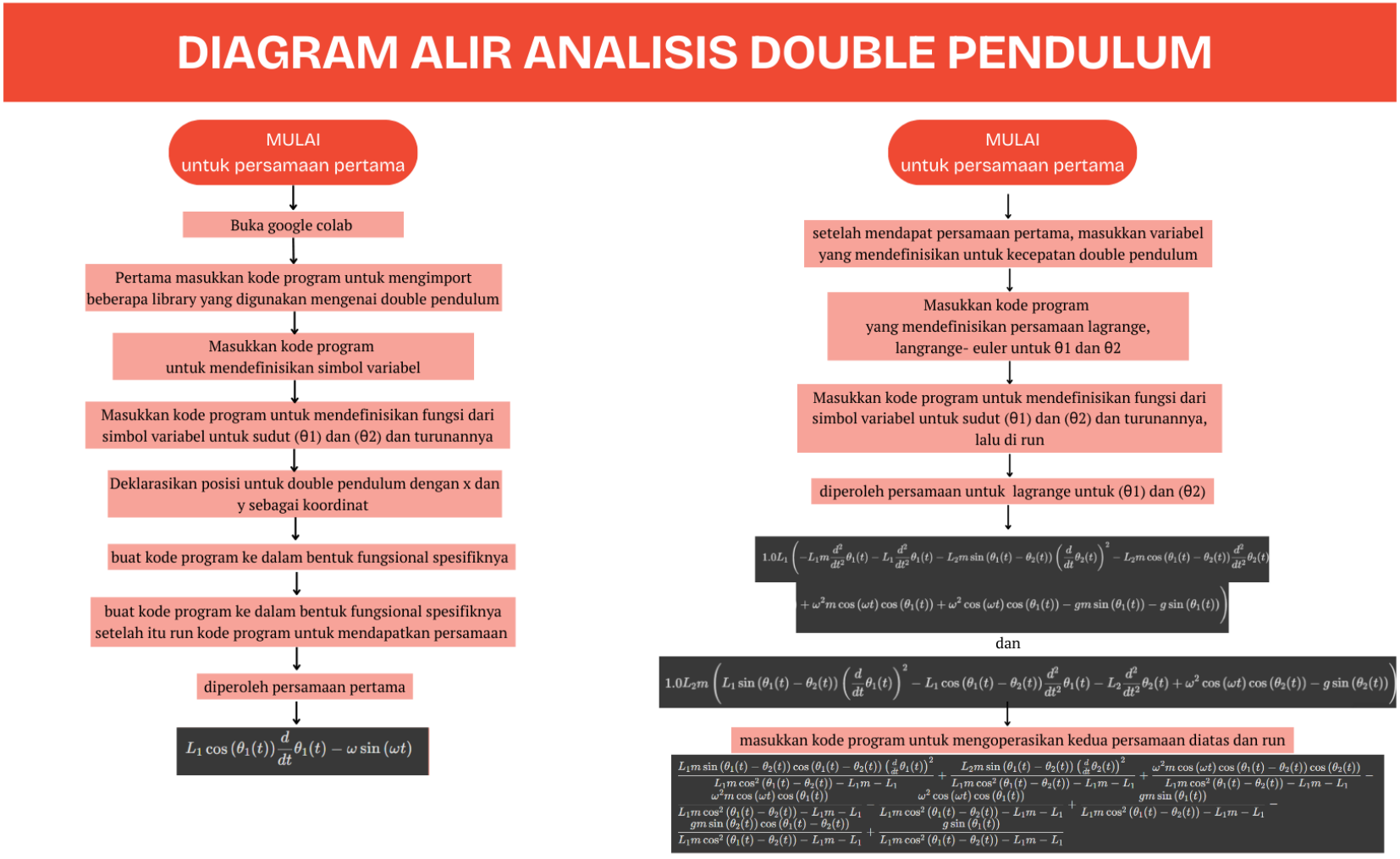
TUGAS 8 :ANALISI DOUBLE PENDULUM

Nama : Novianti Zakiah

NIM : 1227030026

1. Diagram alir



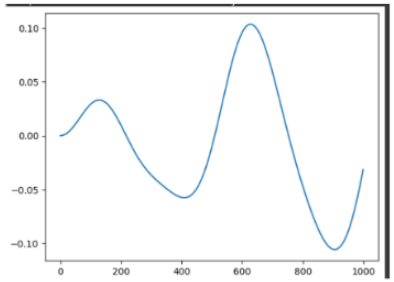
masukkan kode program untuk menstubstitusikan persamaan diatas kemudian run

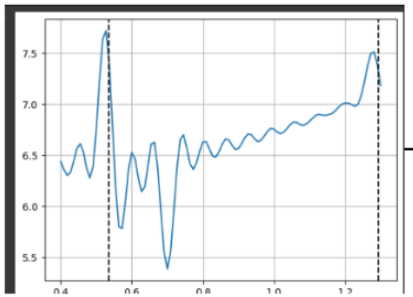
$$-\sqrt{-\frac{Cg\left(-2.0+\frac{1.4142135623731\left(C^2L_1^2+CL_1+0.5\right)^{0.5}}{CL_1}-\frac{1}{CL_1}\right)}{CL_1+1.0}}$$

$$\frac{3\sqrt{37935706248590}\sqrt{\frac{g}{L_1}}}{10000000}$$

mengubah persamaan eksak dan memasukkan ke dalam persamaan numerik dan memplot ke bentuk grafik

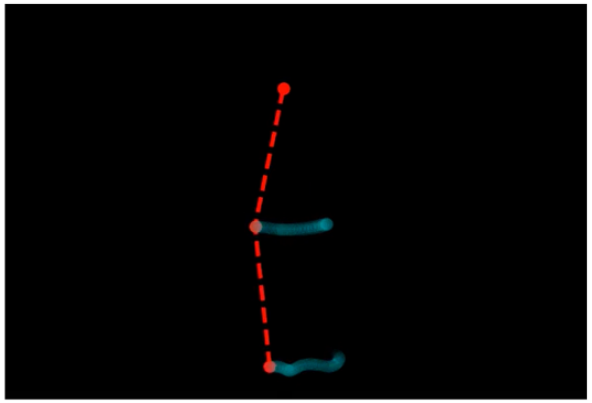
grafik yang diperoleh





masukkan kode program untuk memunculkan animasi double pendulum, kemudian run

animasi yang diperoleh



2. Algoritma kode program

Masukkan library yang dibutuhkan

```
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
```

numpy untuk operasi numerik.
Sympy untuk komputasi simbolik.
scipy.integrate.odeint untuk integrasi numerik persamaan diferensial.
matplotlib.pyplot untuk membuat plot 2D.
matplotlib.animation untuk membuat animasi.
PillowWriter dan HTML untuk menyimpan dan menampilkan animasi

Kemudian masukan kode program untuk mendefinisikan variable yang diperlukan sympy :

```
t, m, g, L1, L2, w, C, alph, beta = smp.symbols(r't m g L_1, L_2 \omega C \alpha \beta')
```

masukkan kode program untuk mendefinisikan theta 1 dan 2 :

```
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)
#Mendeklarasikan nilai x1(teta1),y1(teta1), dan x2(teta1,teta2), y2(teta1,teta2)
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)
#Masukkan ke dalam bentuk fungsional spesifik dari x1,y1,x2,y2
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)
# definisi fungsi numerik dari vx1, vy1, vx1, vy2
smp.diff(x1,t)
```

the1 dan the2 Sudut untuk pendulum pertama dan kedua, didefinisikan sebagai fungsi dari waktu t.
the1_d, the1_dd Turunan pertama dan kedua dari the1.
the2_d, the2_dd: Turunan pertama dan kedua dari the2
x1, y1 Koordinat ujung pendulum pertama.
x2, y2 Koordinat ujung pendulum kedua.

maka diperoleh persamaan :

$$L_1 \cos\left(\theta_1(t)\right) \frac{d}{dt}\theta_1(t) - \omega \sin\left(\omega t\right)$$

Masukkan kode program persamaan lagrange dan lagrange-euler untuk theta 1 dan 2 :

```
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
```

vx1_f, vx2_f, vy1_f, dan vy2_f digunakan untuk menghitung komponen kecepatan dari titik-titik pendulum secara numerik.
T dihitung dari kecepatan ujung-ujung pendulum.
V dihitung berdasarkan posisi vertikal pendulum.
L adalah selisih antara T dan V
LE1 dan LE2 adalah persamaan Euler-Lagrange untuk the1 dan the2.

Diperoleh persamaan untuk theta 1 dan 2 :

$$1.0L_1\left(-L_1 m\frac{d^2}{dt^2}\theta_1(t) - L_1\frac{d^2}{dt^2}\theta_1(t) - L_2 m\sin\left(\theta_1(t)-\theta_2(t)\right)\left(\frac{d}{dt}\theta_2(t)\right)^2 - L_2 m\cos\left(\theta_1(t)-\theta_2(t)\right)\frac{d^2}{dt^2}\theta_2(t) + \omega^2 m\cos(\omega t)\cos\left(\theta_1(t)\right) + \omega^2\cos(\omega t)\cos\left(\theta_1(t)\right) - gm\sin\left(\theta_1(t)\right) - g\sin\left(\theta_1(t)\right)\right)$$

Dan

$$1.0L_2 m\left(L_1\sin\left(\theta_1(t)-\theta_2(t)\right)\left(\frac{d}{dt}\theta_1(t)\right)^2 - L_1\cos\left(\theta_1(t)-\theta_2(t)\right)\frac{d^2}{dt^2}\theta_1(t) - L_2\frac{d^2}{dt^2}\theta_2(t) + \omega^2\cos(\omega t)\cos\left(\theta_2(t)\right) - g\sin\left(\theta_2(t)\right)\right)$$

Kemudian masukkan kode pogram untuk memsubstitusikan lagrange theta 1 dan 2 :

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

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

maka diperoleh :

$$\frac{L_1 m \sin\left(\theta_1(t) - \theta_2(t)\right) \cos\left(\theta_1(t) - \theta_2(t)\right) \left(\frac{d}{dt}\theta_1(t)\right)^2}{L_1 m \cos^2\left(\theta_1(t) - \theta_2(t)\right) - L_1 m - L_1} + \frac{L_2 m \sin\left(\theta_1(t) - \theta_2(t)\right) \left(\frac{d}{dt}\theta_2(t)\right)^2}{L_1 m \cos^2\left(\theta_1(t) - \theta_2(t)\right) - L_1 m - L_1} + \frac{\omega^2 m \cos(\omega t) \cos\left(\theta_1(t) - \theta_2(t)\right) \cos\left(\theta_2(t)\right)}{L_1 m \cos^2\left(\theta_1(t) - \theta_2(t)\right) - L_1 m - L_1} - \frac{\omega^2 m \cos(\omega t) \cos\left(\theta_1(t)\right)}{L_1 m \cos^2\left(\theta_1(t) - \theta_2(t)\right) - L_1 m - L_1} - \frac{\omega^2 \cos(\omega t) \cos\left(\theta_1(t)\right)}{L_1 m \cos^2\left(\theta_1(t) - \theta_2(t)\right) - L_1 m - L_1} + \frac{g m \sin\left(\theta_1(t)\right)}{L_1 m \cos^2\left(\theta_1(t) - \theta_2(t)\right) - L_1 m - L_1} - \frac{g m \sin\left(\theta_2(t)\right) \cos\left(\theta_1(t) - \theta_2(t)\right)}{L_1 m \cos^2\left(\theta_1(t) - \theta_2(t)\right) - L_1 m - L_1} + \frac{g \sin\left(\theta_1(t)\right)}{L_1 m \cos^2\left(\theta_1(t) - \theta_2(t)\right) - L_1 m - L_1}$$

Kemudian masukkan deklarasi untuk menyederhanakan persamaan gerak simbolik LE1 dan LE2 :

```
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.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]
```

maka diperoleh :

$$-\sqrt{-\frac{Cg\left(-2.0+\frac{1.4142135623731\left(C^2L_1^2+CL_1+0.5\right)^{0.5}}{CL_1}-\frac{1}{CL_1}\right)}{CL_1+1.0}}$$

$$\frac{3\sqrt{37935706248590}\sqrt{\frac{g}{L_1}}}{10000000}$$

```
#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)
```

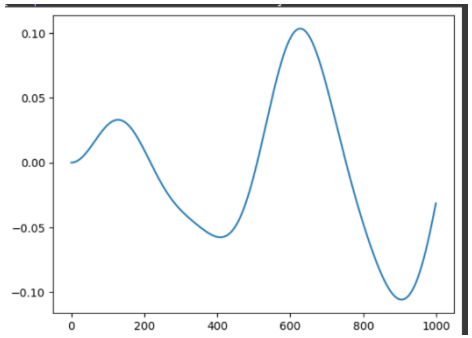
dz1dt_f dan dz2dt_f adalah fungsi untuk menghitung percepatan sudut secara numerik.

```
#mendefinisikan persamaan differensial fungsi s
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),
    ]
```

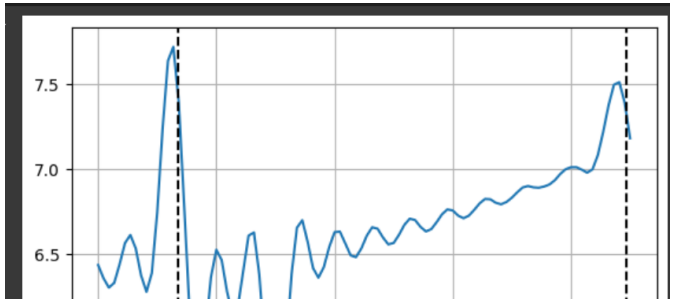
dSdt mengembalikan turunan pertama dan kedua dari the1 dan the2.

```
#Menambahkan salah satu contoh fungsi numerik untuk mendapatkan nilai
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])
```

untuk memplot data yang ada di grafik , maka diperoleh :




```
#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()
```



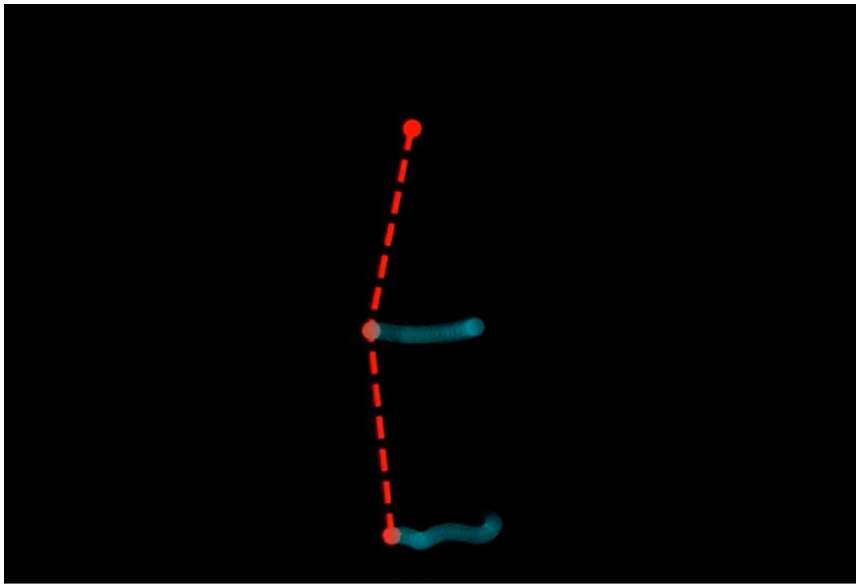
Masukkan kode pogram untuk membuat animasi double pendulum :

```
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])   # jejak dan garis pada benda 1
    ln3.set_data(x2[::10][i:max(1,i-trail2):-1], y2[::10][i:max(1,i-trail2):-1])   # jejak dan garis pada benda 2
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([], [], 'ro-',markersize = 8, alpha=0.05, color='cyan')   # line for Earth
ln3, = ax.plot([], [], 'ro-',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())
```

Grafik double pendulum :



3. Penjelasan grafik

Grafik pertama menunjukkan hubungan antara waktu pada sumbu-x dengan suatu variabel, yang merepresentasikan sudut osilasi, pada sumbu-y. Grafik ini menunjukkan pola osilasi dengan amplitudo yang naik turun secara berkala. Nilai variabel tampak mencapai puncak di sekitar waktu 600 sebelum menurun kembali, kemudian naik lagi di waktu selanjutnya. Pola ini menunjukkan bahwa sistem berosilasi secara periodik dengan perubahan energi yang naik turun seiring waktu, kemungkinan mencerminkan sifat non-linier atau gesekan dalam sistem.

Grafik kedua menampilkan osilasi lain dengan sumbu-x yang mewakili waktu dan sumbu-y menunjukkan kecepatan. Dua garis putus-putus vertikal yang tampak pada grafik dapat menandakan batas waktu tertentu , mungkin saat fenomena signifikan terjadi. Grafik ini menunjukkan pola fluktuasi dengan beberapa puncak tajam dan lembah, yang mungkin menandakan resonansi atau instabilitas dalam sistem. Secara keseluruhan, grafik ini menunjukkan tren kenaikan dari kiri ke kanan, yang mungkin menunjukkan peningkatan energi atau frekuensi sistem secara bertahap.

Gambar ketiga merupakan visualisasi dari gerak sistem pendulum dua lengan (double pendulum). Garis putus-putus merah menggambarkan jalur pergerakan titik berat atau poros dari lengan pertama, sementara garis biru mewakili jalur ujung dari lengan kedua pendulum. Visual ini menunjukkan pola gerak yang sangat kompleks dan acak, yang merupakan karakteristik dari sistem non-linier yang bersifat chaos. Pergerakan kedua lengan pendulum ini menunjukkan pola yang sulit diprediksi dan sangat sensitif terhadap perubahan kecil dalam kondisi awal, ciri khas dari sistem chaos seperti double pendulum.

Secara keseluruhan, ketiga grafik ini menggambarkan karakteristik dinamis dari sebuah sistem osilasi yang kompleks, kemungkinan besar dari model double pendulum yang menunjukkan sifat chaos, ketidakstabilan, dan sensitivitas tinggi terhadap perubahan kondisi awal.