JAWABAN

PRAKTIKUM FISIKA KOMPUTASI

ANALISIS SISTEM DOUBLE PENDULUM

Nama : Gilang Pratama Putra Siswanto

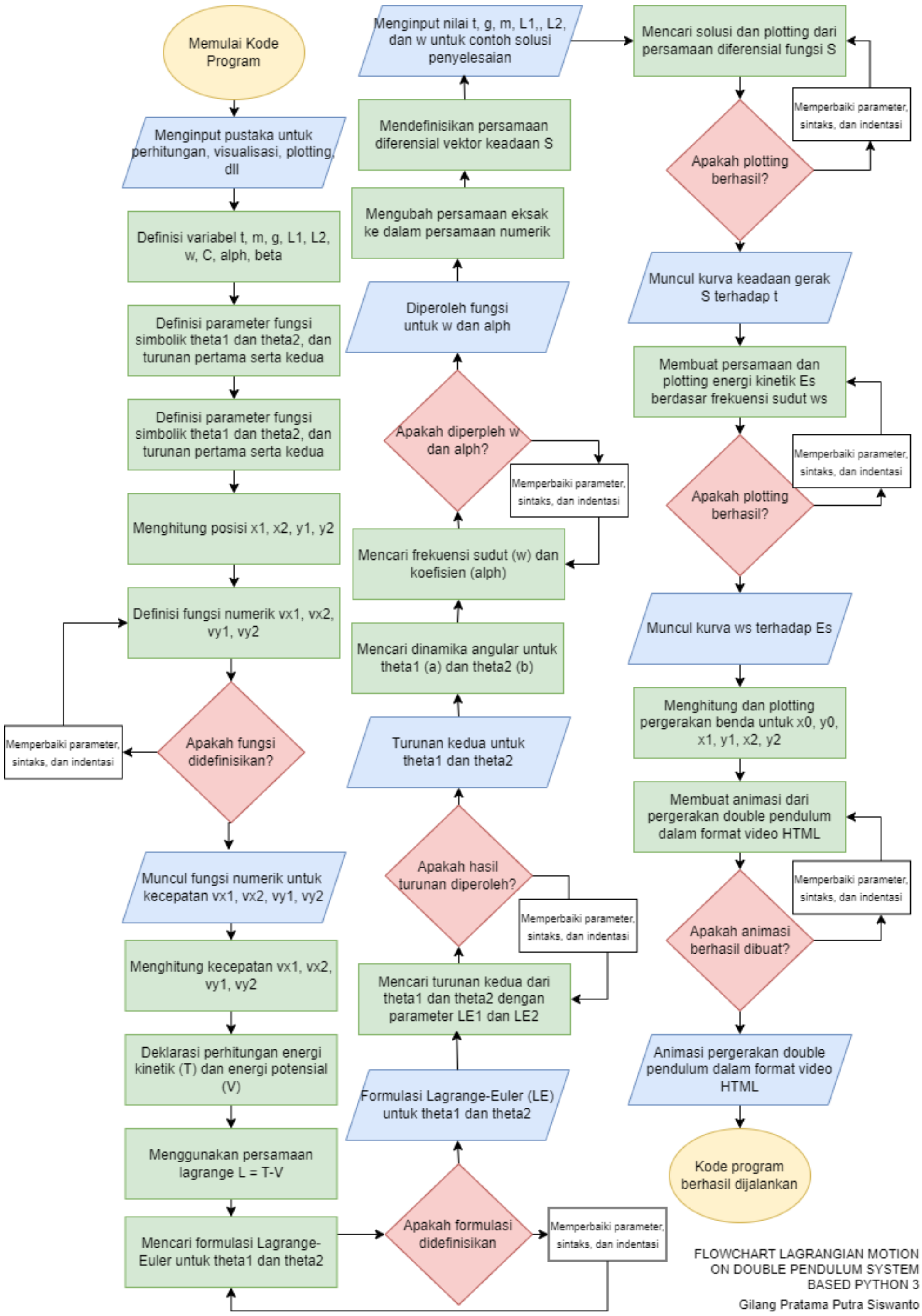
NIM : 1227030017

Prodi : Fisika (Angkatan 2022)

Kode Tugas : TUGAS 8 – ANALISIS SISTEM DOUBLE PENDULUM

BABAK I – DIAGRAM ALIR *(FLOWCHART)* KODE PROGRAM

Gambar 1 – Diagram Alir Kode Program Analisis Gerak Sistem Pendulum Ganda berbasis Metode Lagrange



BABAK II – PENJELASAN ALGORITMA KODE PROGRAM

Untuk menganalisis dinamika gerak dari sistem pendulum ganda menggunakan metode Lagrange, diaplikasikan piranti lunak Python 3 untuk menyelesaikan persoalan gerak pendulum ganda *(double pendulum)* secara numerik.

```
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
```

Pada tahap awal, digunakan beberapa pustaka utama di antaranya yaitu numpy untuk penyelesaian perhitungan matematis, sympy untuk simplifikasi suatu fungsi atau ekspresi matematis, kemudian pustaka integrate untuk mengaplikasikan teknik integrasi maupun diferensiasi dari suatu fungsi, dan pustaka matplotlib, mpl_toolkits.mplot3d, matplotlib.animation, dan IPython.display untuk visualisasi dari simulasi gerak *double pendulum* baik melalui grafik maupun penampil gerak dalam format video berbasis HTML.

```
# Menentukan 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')

# Mendefinisikan theta 1 dan theta 2 dan menyatakan fungsi
waktu.
# Juga definisi turunan pertama dan kedua.
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)
```

Tahapan selanjutnya yang dilakukan adalah mendeklarasikan parameter utama dari suatu gerak pendulum, yaitu waktu (t), massa (m), percepatan gravitasi (g), panjang tali untuk pendulum (L), frekuensi sudut (w), serta beberapa koefisien untuk gerak pendulum seperti C, alph, dan beta. Digunakan fungsi smp.symbols untuk simplifikasi parameter ke dalam penyelesaian persamaan. Kemudian, didefinisikan fasa dari masing-masing pendulum sebagai the1 dan the2, serta mendefinisikan turunan pertama dan kedua dari fasa masing-masing pendulum sebagai the_d dan the_dd. Digunakan smp.symbols untuk simplifikasi theta sebagai suatu fungsi gerak, dan digunakan fungsi smp.diff untuk turunan dari theta terhadap waktu (t).

```
# Mendeklarasikan nilai x1(teta1), y1(teta1),
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)
```

Setelah didefinisikan fasa 1 dan 2, tahapan selanjutnya adalah mendeklarasikan nilai dari x1, y1, x2, dan y2 dengan parameter pergerakan bergantung terhadap waktu (t).

```
# 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, vx2, vy2
smp.diff(x1, t)
```

Untuk memperoleh pendekatan yang sesuai dengan penyelesaian secara eksak, maka nilai x1, y1, x2, dan y2 ke dalam bentuk fungsional spesifik dengan mengacu pergerakan objek terhadap sudut yang berubah. Bentuk ini merepresentasikan posisi pada pendulum 1 dan 2 dengan parameter theta1 dan theta2. Setelah didefinisikan, tahapan selanjutnya adalah mendefinisikan fungsi numerik kecepatan benda pada x1 (vx1), x2 (vx2), y1 (vy1), dan y2 (vy2) berupa turunan dari posisi x terhadap waktu.

```
# Define velocity functions using lambdify
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))
```

Tahapan selanjutnya yang dilakukan adalah mendefinisikan fungsi kecepatan dari x1, x2, y1, dan y2 sebagai vx1_f, vx2_f, vy1_f, dan vy2_f mengunakan fungsi lambdify untuk penyusunan fungsi dengan parameter turunan adalah x dan y pada kedua pendulum terhadap waktu.

```
# Lagrangian formula
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

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

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


LE1
LE2
```

Setelah diperoleh kecepatan dari kedua pendulum, tahapan selanjutnya yang dilakukan adalah mengaplikasikan persamaan Lagrange dengan acuan gerak pada metode Lagrange adalah adanya energi kinetic (T) dan energi potensial (V). Untuk energi kinetik total pada kedua pendulum, diakumulasikan energi kinetik pada pendulum satu dengan parameter vx1 dan vy1 dan pendulum dua dengan parameter kecepatan vx2 dan vy2. Kemudian, energi potensial dipengaruhi oleh energi potensial dari kedua pendulum dengan acuan ketinggian bergantung pada nilai y1 dan y2. Persamaan Lagrange ini representasi dari pengurangan dari energi kinetik total (T) dikurangi energi potensial total (V). Setelah diperoleh persamaan Lagrange, tahapan selanjutnya adalah mencari persamaan Lagrange-Euler untuk theta1 dan theta2. Diaplikasikan turunan pertama untuk persamaan Lagrange yang diperoleh (L) terhadap theta1 untuk pendulum 1 dan theta2 untuk pendulum 2 dikurangi turunan L terhadap waktu dan turunan dari hasil diferensiasi theta1 dan theta2 terhadap waktu. Hasil keduanya disimplifikasi menggunakan fungsi simplify kemudian dijalankan untuk memperoleh persamaan dari Lagrange-Euler pendulum 1 (LE1) dan Lagrange-Euler pendulum 2 (LE2).

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

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

Bagian algoritma ini kemudian merepresentasikan dari solusi dengan menggunakan fungsi solve untuk menyelesaikan persamaan Lagrange-Euler denga parameter LE1 dan LE2 terhadap turunan kedua dari theta1 dan theta2 serta hasil akhir ditampilkan menggunakan fungsi sols[the1_dd].

```
# Substitute and simplify expressions for Lagrange-Euler
equations
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()
```

Pada tahapan ini, disubtitusi hasil dari LE1 dan LE2 dengan mensimplifikasi keduanya melalui parameter trigonometri seperti fungsi sin dan cos, parameter simbolik seperti C, frekuensi sudut w, dan koefisien alph, kemudian parameter massa m dan panjang tali L1 dan L2, serta melalui pendekatan linear berdasar deret Taylor. Hasil substitusi ini kemudian memperoleh simplifikasi menjadi parameter a untuk LE1 dan b untuk LE2.

```
yeet = smp.solve([a.args[1], b.args[2]], (w, alph))
yeet[2][0]
yeet[0][0]
```

Tahapan selanjutnya, digunakan fungsi solve untuk definisi dari parameter yeet dengan mencari solusi berdasar parameter fungsi a dan b untuk menghasilkan fungsi akhir yaitu frekuensi (w) dan koefisien proporsional (alph). Penggunaan keduanya akan diterapkan untuk mensimulasikan gerak *double pendulum* baik melalui grafik maupun animasi.

```
# Mengubah persamaan eksak dan memasukan 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)
```

Pada tahapan ini, didefinisikan dz1dt_f dan dz2dt_f sebagai transformasi dari persamaan eksak yang diperoleh sebelumnya ke dalam persamaan numerik. Digunakan fungsi lambdify untuk menyusun fungsi dengan parameter yang digunakan adalah waktu (t), massa (m), percepatan gravitasi (g), frekuensi sudut (w), panjang tali pendulum 1 dan 2 (L1, L2), fasa pada pendulum 1 dan 2 (the1, the2), serta turunan pertamanya (the1_d, the2_d).

```
# 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),
    ]
```

Tahapan selanjutnya yang ditempuh adalah mendefinisikan persamaan diferensial dari fungsi S sebagai dSdt. Didefinisikan dSdt sebagai fungsi dari S yang bergantung terhadap waktu dan mengacu pada beberapa parameter utama seperti sudut theta1 dan theta2, frekuensi sudut z1 dan z2. Hasil ini kemudian dikembalikan berupa turunan theta1 terhadap waktu atau dthe1dt_f sebagai frekuensi sudut z1 dan turunan theta2 terhadap waktu atau dthe2dt_f sebagai frekuensi sudut z2.

```
#Menambahkan salah satu contoh fungsi numerik untuk
mendapakan 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 mengaplikasikan secara langsung solusi persamaan gerak dari *double pendulum,* maka didefinisikan nilai dari beberapa parameter seperti waktu (t) dari 0 hingga 1000 detik dengan interval 20 detik, kemudian nilai percepatan gravitasi (g) sebesar 9.81 m/s^2, masssa kedua pendulum diasumsikan sama dengan nilai 1 kg, dan panjang tali L1 sebesar 20 m serta panjang tali L2 memiliki panjang yang sama. Kemudian, nilai frekuensi sudut diperoleh melalui akar dari pembagian percepatan gravitasi terhadap L1. Tahapan diferensiasi didefinisikan sebagai ans menggunakan fungsi odeint (persamaan diferensial biasa) dengan parameter fungsi dSdt terhadap waktu t. Hasil diferensiasi kemudian diplotting untuk membentuk kurva fungsi S terhadap t.

```
# 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
```

Setelah diperoleh grafik perubahan S terhadap t, tahapan selanjutnya yang dilakukan adalah memperoleh nilai energi dengan parameter utama yaitu Untuk memperoleh nilai E, ditetapkan waktu dari t 0 detik hingga detik 100. Kemudian, dihitung nilai turunan dari fungsi S terhadap waktu. Kemudian, dihitung nilai kecepatan pada x1, x2, y1, dan y2 sebagai vx1, vx2, vy1, dan vy2 untuk kalkulasi nilai energi kinetiknya.

```
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()
```

Tahapan selanjutnya yang dilakukan adalah membuat *plotting* hubungan antara frekuensi sudut terhadap energi kinetik yang dihasilkan sistem. Dilakukan plotting dari sumbu x berupa frekuensi sudut 0.4 hingga 1.3 untuk asumsi sudut yang kecil, kemudian diperoleh nilai energi untuk sumbu y. Diberikan batas dengan label konstanta k. Kemudian, ditampilkan kurva dengan 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])  # 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())
```

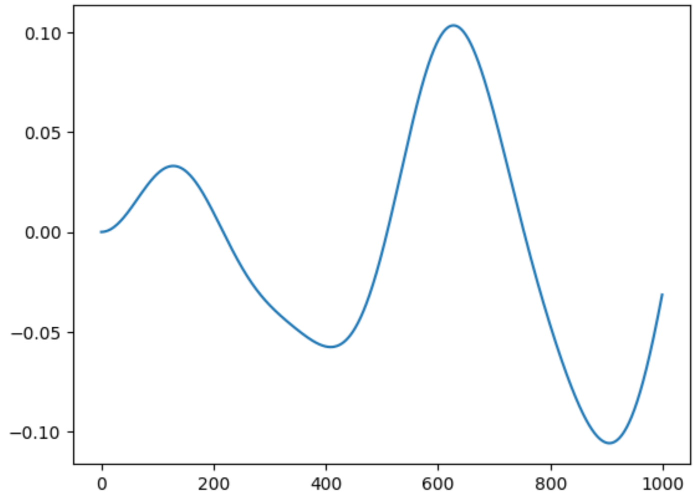
Bagian kode ini bertujuan untuk membuat animasi gerak double pendulum berdasarkan hasil solusi numerik dari persamaan diferensialnya. Proses dimulai dengan mendefinisikan parameter sistem seperti gravitasi g, panjang batang L1 dan L2, serta nilai awal sudut (y0). Solusi numerik dihitung menggunakan fungsi odeint, yang memecahkan sistem persamaan diferensial untuk posisi sudut (theta1,theta2) dan kecepatan sudut terhadap waktu t.Fungsi get_x0y0x1y1x2y2 digunakan untuk menghitung koordinat kartesian titik-titik pada sistem pendulum: titik tetap (x0,y0), posisi pendulum pertama (x1,y1), dan posisi pendulum kedua (x2,y2), berdasarkan waktu dan solusi sudut. Hasil koordinat ini digunakan untuk menggambarkan lintasan pendulum. Fungsi animate kemudian mendefinisikan logika untuk memperbarui posisi pendulum dalam animasi. Posisi setiap pendulum direpresentasikan sebagai garis yang dihubungkan oleh titik-titik koordinat. Selain itu, fungsi ini juga menambahkan jejak pergerakan sebagai efek visual, dengan panjang jejak diatur melalui variabel trail1 dan trail2.

Plot dasar untuk animasi dibuat menggunakan matplotlib. Grafik memiliki latar belakang hitam dengan sumbu yang disembunyikan, dan masing-masing pendulum digambarkan sebagai titik merah yang terhubung dengan garis. Dua objek garis tambahan (ln2 dan ln3) digunakan untuk menunjukkan jejak pergerakan pendulum. Akhirnya, animasi dibuat menggunakan fungsi FuncAnimation dari modul matplotlib.animation, yang memperbarui posisi pendulum pada setiap frame sesuai dengan data yang dihitung. Animasi dapat ditampilkan langsung di notebook melalui HTML(ani.to_html5_video()). Kode ini juga menyediakan opsi untuk menyimpan animasi sebagai file GIF dengan mengaktifkan baris komentar ani.save.

BABAK III – PENJELASAN GRAFIK DAN ANIMASI

Ketika menjalankan program, diperoleh dua kurva dan satu animasi dari sistem persamaan gerak pendulum ganda menggunakan metode Lagrange. Grafik pertama yaitu perubahan keadaaan gerak S terhadap waktu yang disajikan dalam kurva sebagai berikut.

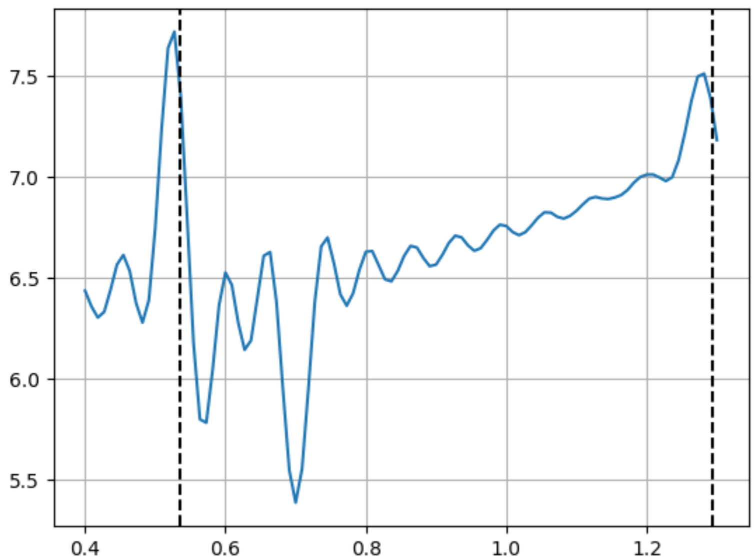
Gambar 2 – Grafik Keadaan Gerak Sistem S Terhadap Waktu



Pada grafik, ditentukan waktu dari t 0 detik hingga t 1000 detik untuk mengamati perubahan keadaaan gerak sistem S. Perubahan keadaan gerak sistem S ini mencakup di dalamnya perubahan posisi kedua pendulum 1 dan 2, baik dalam sumbu x maupun y dengan pengamatan mengacu pada perubahan fasa kedua pendulum, serta turunan pertama dan keduanya. Sistem S merepresentasikan gerak dari double pendulum, dengan fokus utama pada perubahan posisi, kecepatan sudut, atau percepatan sudut kedua pendulum dalam satuan waktu. Pola yang diamati pada grafik menunjukkan osilasi yang tidak teratur, mencerminkan sifat *sistem double pendulum* yang bersifat non-linear dan sangat sensitif terhadap kondisi awal *(chaotic dynamics)*. Pada waktu t 0 hingga t 200 detik, grafik menampilkan osilasi dengan amplitudo kecil yang mencerminkan gerakan stabil. Namun, seiring waktu, amplitudo mulai meningkat hingga mencapai puncaknya di sekitar t 600 detik. Setelah itu, grafik menunjukkan pola penurunan amplitudo hingga mendekati minimum pada sekitar t 800 detik. Siklus ini menggambarkan transfer energi bolak-balik antara pendulum pertama dan kedua, serta pengaruh gravitasi dan panjang batang pada perubahan posisi. Grafik ini juga mengindikasikan adanya perbedaan fase antara dua pendulum, yang dapat terlihat dari bentuk gelombang yang asimetris. Turunan pertama dari grafik (gradien) akan menunjukkan kecepatan perubahan posisi pendulum, sementara turunan kedua (kelengkungan) akan mengindikasikan percepatan yang terkait dengan gaya pemulih dalam sistem. Amplitudo yang bervariasi menunjukkan karakteristik sistem yang dinamis dan pengaruh signifikan dari gaya eksternal serta momen sudut pada gerak total.

Selanjutnya, parameter yang divisualisasikan adalah kurva energi kinetik sistem terhadap frekuensi sudut sistem. Kurva ditampilkan sebagai berikut.

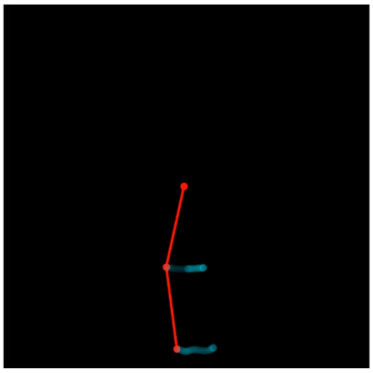
Gambar 3 – Grafik Energi Kinetik Sistem Terhadap Frekuensi Sudut



Grafik ini merepresentasikan hubungan antara energi kinetik sistem dan frekuensi sudut pada sistem gerak *double pendulum.* Pada grafik, ditentukan frekuensi sudut dari 0.4 hingga 1.3 untuk mengamati perubahan energi kinetik dari sistem S. Kurva ini menggambarkan bagaimana energi kinetik bervariasi secara non-linear dengan perubahan frekuensi sudut, yang merupakan karakteristik umum dari sistem dinamis kompleks seperti *double pendulum.* Pada frekuensi sudut 0.4, energi kinetik sistem berada pada nilai sedang dengan fluktuasi kecil. Ketika frekuensi sudut meningkat menuju 0.6, terjadi lonjakan tajam energi kinetik, menunjukkan bahwa pada rentang ini ada kondisi resonansi atau transfer energi yang efisien di dalam sistem. Setelah mencapai puncak, energi kinetik menurun drastis, menggambarkan adanya stabilitas yang hilang pada frekuensi tersebut. Pada rentang frekuensi sudut 0.7 hingga 1.3, energi kinetik kembali menunjukkan pola fluktuasi dengan amplitudo yang lebih kecil, tetapi secara bertahap meningkat. Fenomena ini dapat diinterpretasikan sebagai efek kombinasi antara gaya pemulih dan momen inersia dari pendulum, di mana sistem mencoba mencapai keseimbangan baru. Peningkatan energi kinetik yang terlihat di frekuensi sudut 1.2.

Output lainnya yang dihasilkan melalui kode program disusunnya animasi pergerakan sistem *double pendulum* dalam format video HTML. Adapun potongan *frame* dari cuplikan pergerakan kedua pendulum disajikan sebagai berikut.

Gambar 4 – Animasi Gerak *Double Pendulum*



Gerak sistem double pendulum dalam animasi menunjukkan sifat khas dari dinamika non-linear dan kekacauan. Sistem ini terdiri dari dua pendulum yang terhubung secara serial, di mana pendulum pertama terpasang ke titik tetap, dan pendulum kedua terpasang ke ujung pendulum pertama. Ketika pendulum dilepaskan, kedua massa mengikuti gerakan yang terlihat teratur. Pendulum pertama berayun dengan amplitudo besar, sementara pendulum kedua mengikuti gerakan yang dipengaruhi oleh dinamika pendulum pertama. Setelah beberapa waktu, gerakannya menjadi tidak teratur, di mana sudut dan kecepatan kedua pendulum berubah secara drastis dan tidak berulang. Ini adalah tanda khas dari sistem chaos, di mana perubahan kecil pada kondisi awal dapat menyebabkan perbedaan besar pada jalur gerakan.