TUGAS 01

Numerical Methods & Programming

Nama : Rifqi Pasha Alviansyah
NIM : 12224010
Kelas : K-02

1. Model Persamaan

Model gerak penerjun payung:

$$m\frac{dv}{dt} = mg - cv \tag{1}$$

dengan:

- $m = 68.1$ kg
- $g = 9.81$ m/s^2
- $c = 10,\ 12.5,\ 15$ kg/s

Simulasi dilakukan untuk $0 \leq t \leq 100$ detik dengan:

$$\Delta t = 0.1,\ 0.5,\ 1$$

2. Solusi Analitik

Persamaan diferensial gerak penerjun payung dengan hambatan udara:

$$m\frac{dv}{dt} = mg - cv \tag{2}$$

Dengan kondisi awal $v(0) = 0$, solusi eksaknya adalah:

$$v(t) = \frac{mg}{c}\left(1 - e^{-\frac{c}{m}t}\right) \tag{3}$$

Dimana:

- m = massa (kg)
- g = percepatan gravitasi (9.81 m/s^2)
- c = koefisien hambatan
- t = waktu (detik)

a. Import Library

```python
import math
import numpy as np
import matplotlib.pyplot as plt
```

b. Perhitungan Solusi Analitik dengan Python

```python
# Analytical Solution

def hitung_v(massa, koef_hambatan, waktu):
g = 9.81  # percepatan gravitasi
kec_terminal = (massa * g) / koef_hambatan
kecepatan = kec_terminal * (1 - math.exp(- (koef_hambatan / massa) * waktu))
return kecepatan

massa = 68.1
waktu = 100

# C = 10
kecepatan1 = hitung_v(massa, 10, waktu)
print(f"C = 10 -> {kecepatan1:.2f} m/s")

# C = 12.5
kecepatan2 = hitung_v(massa, 12.5, waktu)
print(f"C = 12.5 -> {kecepatan2:.2f} m/s")

# C = 15
kecepatan3 = hitung_v(massa, 15, waktu)
print(f"C = 15 -> {kecepatan3:.2f} m/s")
```

Output

```
C = 10   -> 66.81 m/s
C = 12.5 -> 53.44 m/s
C = 15   -> 44.54 m/s
```

c. Plot Grafik Solusi Analitik

```
def hitung_v_array(massa, koef_hambatan, waktu):
g = 9.81
kec_terminal = (massa * g) / koef_hambatan
return kec_terminal * (1 - np.exp(-(koef_hambatan / massa) * waktu))

massa = 68.1
koef_list = [10, 12.5, 15]
waktu = np.linspace(0, 100, 1000)

plt.figure(figsize=(8,6))

for c in koef_list:
v = hitung_v_array(massa, c, waktu)
plt.plot(waktu, v, label=f'C = {c}')

plt.xlabel('Waktu (detik)')
plt.ylabel('Kecepatan (m/s)')
plt.title('Grafik Solusi Analitik Kecepatan vs Waktu')
plt.legend()
plt.grid()
plt.show()
```

Hasil Grafik

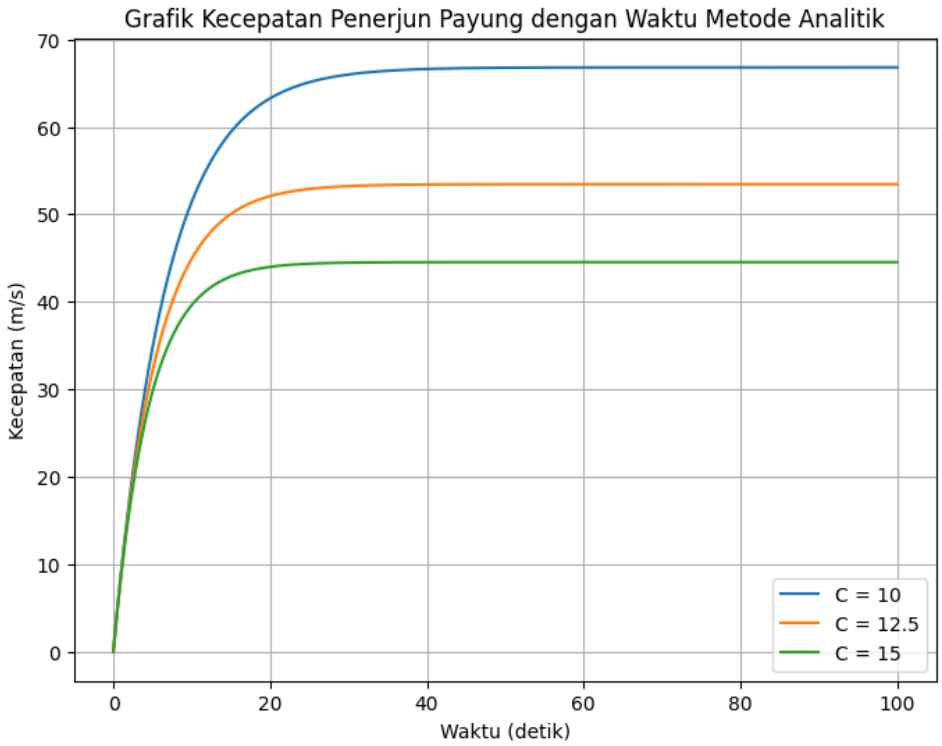

Figure 1: Grafik solusi analitik untuk c = 10, 12.5, dan 15

Analisis

Dari grafik terlihat bahwa semakin besar koefisien hambatan (c), maka kecepatan terminal semakin kecil dan kurva lebih cepat mencapai kondisi stabil.

3. Solusi Numerik (Metode Euler)

Persamaan diskret:

$$v_{i+1} = v_i + \left(g - \frac{c}{m} v_i\right) \Delta t \tag{4}$$

Kondisi awal:

$$v(0) = 0$$

a. Perhitungan Solusi Numerik dengan Python

```python
# Numerical Solution (Metode Euler)

def hitung_v(massa, koef_hambatan, dt, t_maks):
g = 9.81
t = 0
v = 0

while t <= t_maks:
print(f"t = {t:.2f} s, v = {v:.4f} m/s")
v += (g - (koef_hambatan / massa) * v) * dt
t += dt
massa = 68.1
t_maks = 100

# C = 10
hitung_v(massa, 10, 0.1, t_maks)
hitung_v(massa, 10, 0.5, t_maks)
hitung_v(massa, 10, 1, t_maks)

# C = 12.5
hitung_v(massa, 12.5, 0.1, t_maks)
hitung_v(massa, 12.5, 0.5, t_maks)
hitung_v(massa, 12.5, 1, t_maks)

# C = 15
hitung_v(massa, 15, 0.1, t_maks)
hitung_v(massa, 15, 0.5, t_maks)
hitung_v(massa, 15, 1, t_maks)
```

b. Plot Grafik Numerik untuk $\Delta t = 0.1$

```python
def hitung_v_array(massa, koef_hambatan, dt, t_maks):
g = 9.81
t = 0
v = 0
```

```
    t_values = []
    v_values = []

    while t <= t_maks:
    t_values.append(t)
    v_values.append(v)
    v += (g - (koef_hambatan / massa) *
        v) * dt
    t += dt

    return np.array(t_values), np.array
       (v_values)

    massa = 68.1
    t_maks = 100
    dt = 0.1
    C_values = [10, 12.5, 15]

    plt.figure(figsize=(8,5))

    for C in C_values:
    t_vals, v_vals = hitung_v_array(
       massa, C, dt, t_maks)
    plt.scatter(t_vals, v_vals, label=f
       'C = {C}', s=15)

    plt.title(f'Grafik Metode Numerik (
       Euler) untuk dt = {dt}')
    plt.xlabel('Waktu (s)')
    plt.ylabel('Kecepatan (m/s)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

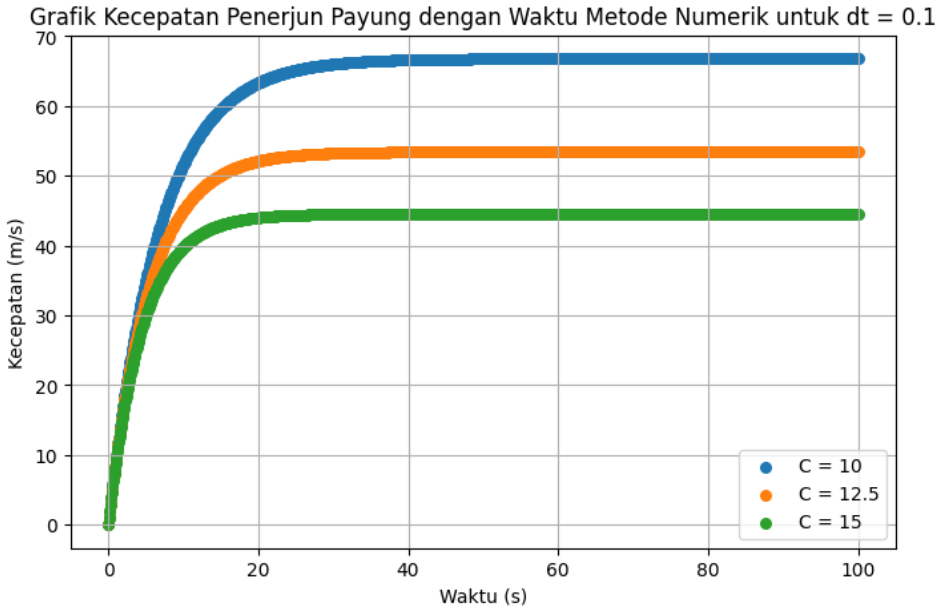

Figure 2: Grafik metode numerik untuk dt = 0.1

c. Plot Grafik Numerik untuk $\Delta t = 0.5$

```
1                   def hitung_v_array(massa,
                        koef_hambatan, dt, t_maks):
2                   g = 9.81
3                   t = 0
4                   v = 0
5                   t_values = []
6                   v_values = []
7
8                   while t <= t_maks:
9                   t_values.append(t)
10                  v_values.append(v)
11                  v += (g - (koef_hambatan / massa) *
                        v) * dt
12                  t += dt
13
14                  return np.array(t_values), np.array
                        (v_values)
15
16                  massa = 68.1
17                  t_maks = 100
18                  dt = 0.1
```

```
C_values = [10, 12.5, 15]

plt.figure(figsize=(8,5))

for C in C_values:
t_vals, v_vals = hitung_v_array(
    massa, C, dt, t_maks)
plt.scatter(t_vals, v_vals, label=f
    'C = {C}', s=15)

plt.title(f'Grafik Metode Numerik (
    Euler) untuk dt = {dt}')
plt.xlabel('Waktu (s)')
plt.ylabel('Kecepatan (m/s)')
plt.legend()
plt.grid(True)
plt.show()
```

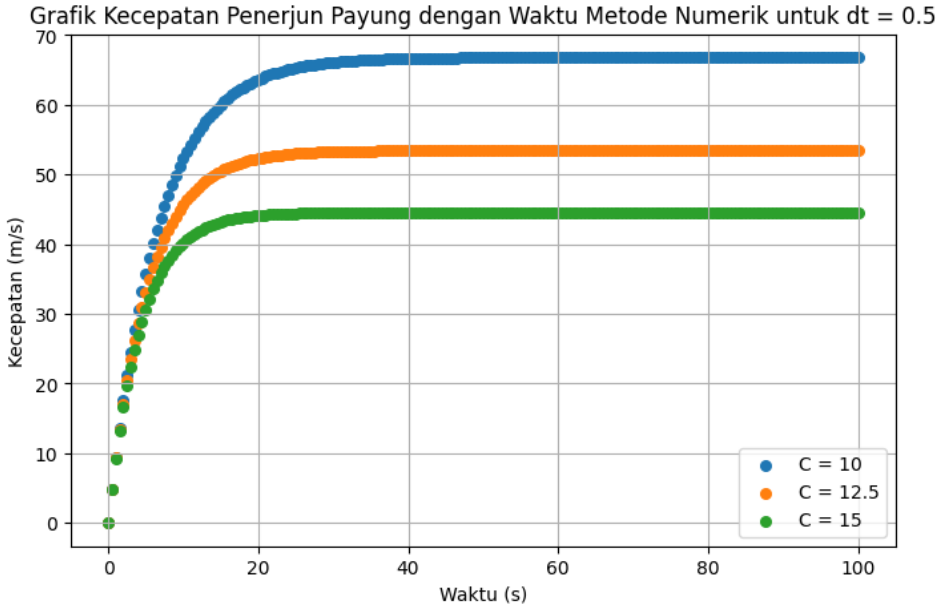

Figure 3: Grafik metode numerik untuk dt = 0.5

d. Plot Grafik Numerik untuk $\Delta t = 1.0$

```
def hitung_v_array(massa,
    koef_hambatan, dt, t_maks):
g = 9.81
```

```
                    t = 0
                    v = 0
                    t_values = []
                    v_values = []

                    while t <= t_maks:
                    t_values.append(t)
                    v_values.append(v)
                    v += (g - (koef_hambatan / massa) *
                        v) * dt
                    t += dt

                    return np.array(t_values), np.array
                        (v_values)

                    massa = 68.1
                    t_maks = 100
                    dt = 0.1
                    C_values = [10, 12.5, 15]

                    plt.figure(figsize=(8,5))

                    for C in C_values:
                    t_vals, v_vals = hitung_v_array(
                        massa, C, dt, t_maks)
                    plt.scatter(t_vals, v_vals, label=f
                        'C = {C}', s=15)

                    plt.title(f'Grafik Metode Numerik (
                        Euler) untuk dt = {dt}')
                    plt.xlabel('Waktu (s)')
                    plt.ylabel('Kecepatan (m/s)')
                    plt.legend()
                    plt.grid(True)
                    plt.show()
```

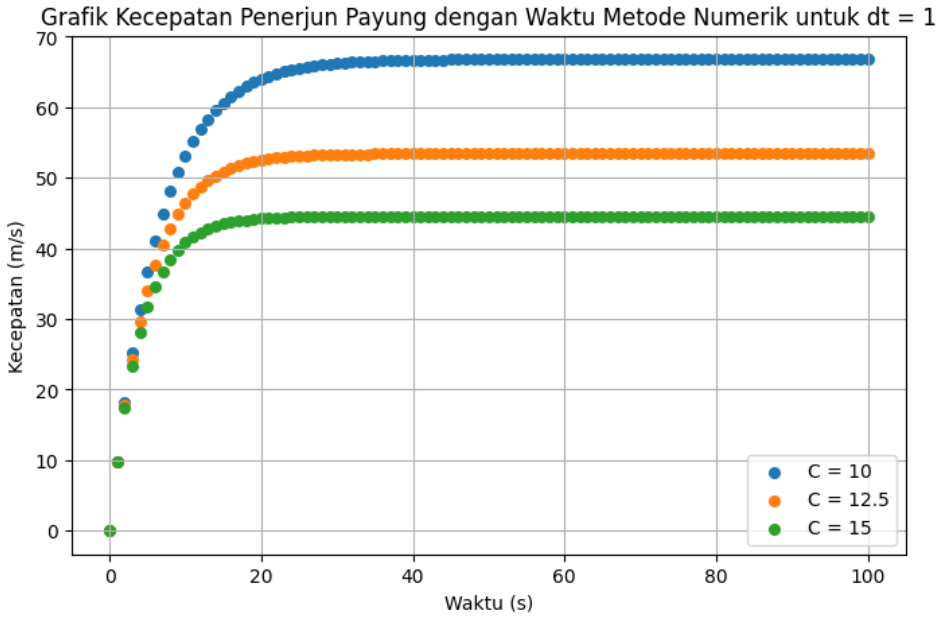

Figure 4: Grafik metode numerik untuk dt = 1.0

Semakin kecil Δt, kurva semakin halus dan mendekati solusi analitik.

4. Plot Perbandingan Analitik dan Numerik

Grafik berikut menunjukkan perbandingan langsung solusi analitik dan numerik untuk setiap Δt.

$\Delta t = 0.1$

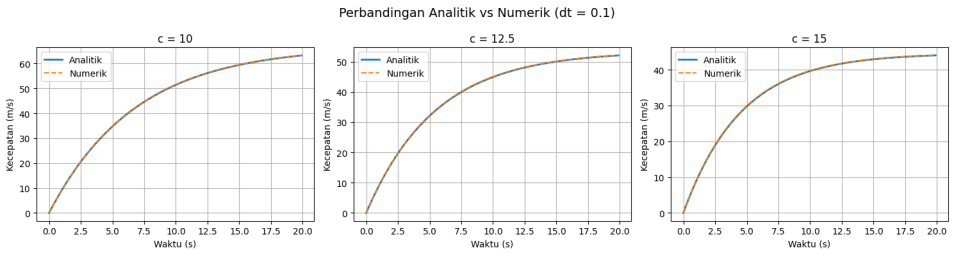

Figure 5: Perbandingan analitik dan numerik ($\Delta t = 0.1$)

$\Delta t = 0.5$

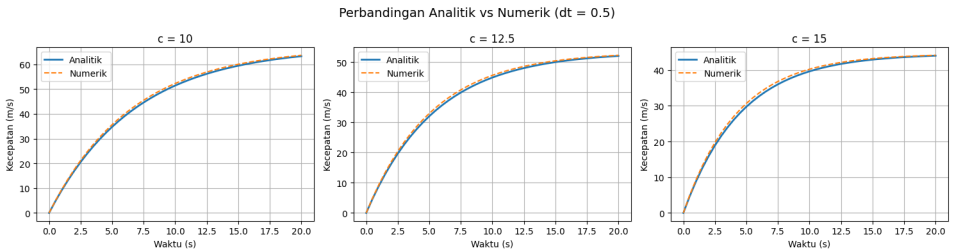

Figure 6: Perbandingan analitik dan numerik ($\Delta t = 0.5$)

$\Delta t = 1$

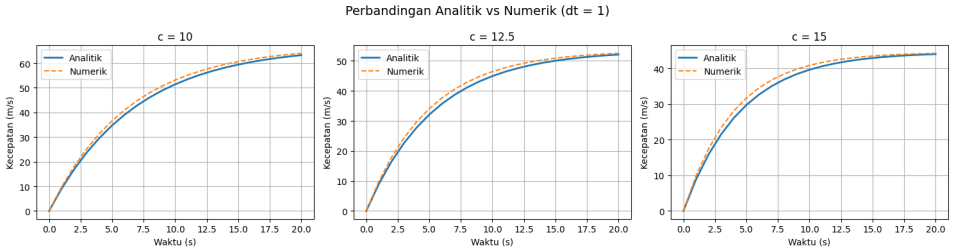

Figure 7: Perbandingan analitik dan numerik ($\Delta t = 1$)

5. Hasil Analisis dan Kesimpulan

Berdasarkan Gambar 5–7, terlihat bahwa hasil metode numerik semakin mendekati solusi analitik ketika nilai Δt dibuat lebih kecil. Untuk $\Delta t = 0.1$, grafik numerik hampir berhimpit dengan grafik analitik. Sedangkan saat Δt diperbesar menjadi 0.5 dan 1, mulai terlihat selisih antara keduanya.

Hal ini menunjukkan bahwa metode Euler cukup baik digunakan, tetapi sangat bergantung pada besar langkah waktu yang dipilih. Semakin besar Δt, hasil yang diperoleh semakin menyimpang dari solusi eksak.

Selain itu, dari semua grafik juga terlihat bahwa semakin besar nilai koefisien hambatan (c), maka kecepatan terminal yang dicapai semakin kecil. Artinya, hambatan udara yang lebih besar membuat penerjun payung lebih cepat mencapai kondisi stabil dengan kecepatan yang lebih rendah.

Secara umum, metode numerik dapat mendekati solusi analitik dengan baik selama nilai Δt dipilih cukup kecil.