MATHEMATICAL PHYSICS 2 (LAB) ASSIGNMENT 5

NAME-ABDUL QADIR

- **ROLL NUMBER-245673982**

COURSE- B.Sc.(H) PHYSICS

Question 1 :

Given acceleration at equidistant time values, calculate position and velocity and plot them (Solve analytical and show final result). Use trapezoidal, Simpson's (1/3 & 3/8 rules) methods and compare the results.

PYTHON CODE –

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

t = np.array([0,1,2,3,4,5,6,7,8,9,10])
a = np.array([5,15,20,25,30,35,40,45,55,60,70])
h = t[1] - t[0]

def trapezoidal(y, h):
    return h * (y[0] + 2 * sum(y[1:-1]) + y[-1]) / 2

def simpsons_1_3(y, h):
    if len(y) % 2 == 0:
        return np.nan
    return h / 3 * (y[0] + 4 * sum(y[1:-1:2]) + 2 * sum(y[2:-2:2]) + y[-1])
```

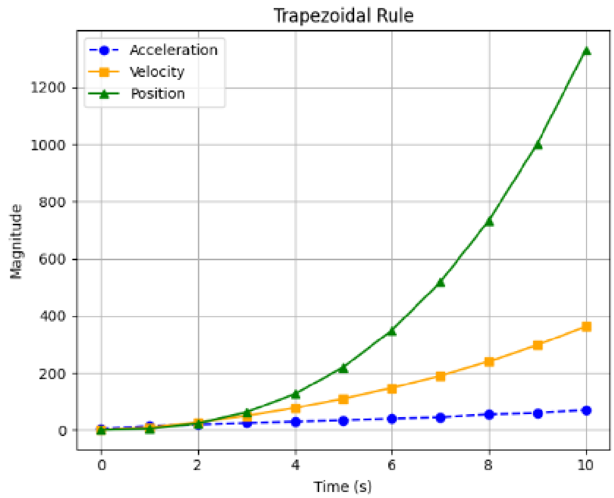
```
def simpsons_3_8(y, h):
    if (len(y) - 1) % 3 != 0:
        return np.nan
    n = len(y) - 1
    return 3 * h / 8 * (y[0] + 3 * sum(y[i] for i in range(1, n) if i % 3 != 0) + 2 *
sum(y[i] for i in range(3, n, 3)) +  y[-1] )

def integrate_method(a, h, method_func):
    velocity = [0]
    for i in range(2, len(a) + 1):
        val = method_func(a[:i], h)
        if np.isnan(val):
            velocity.append(velocity[-1])
        else:
            velocity.append(val)
    velocity = np.array(velocity)
    position = [0]
    for i in range(2, len(velocity) + 1):
        val = method_func(velocity[:i], h)
        if np.isnan(val):
            position.append(position[-1])
        else:
            position.append(val)
    return velocity, np.array(position)

v_trap, x_trap = integrate_method(a, h, trapezoidal)
v_13, x_13 = integrate_method(a, h, simpsons_1_3)
v_38, x_38 = integrate_method(a, h, simpsons_3_8)
df = pd.DataFrame({
    'Time (s)': t,
    'Acceleration': a,
    'Velocity (Trapezoid)': v_trap,
    'Position (Trapezoid)': x_trap,
    'Velocity (1/3)': v_13,
```

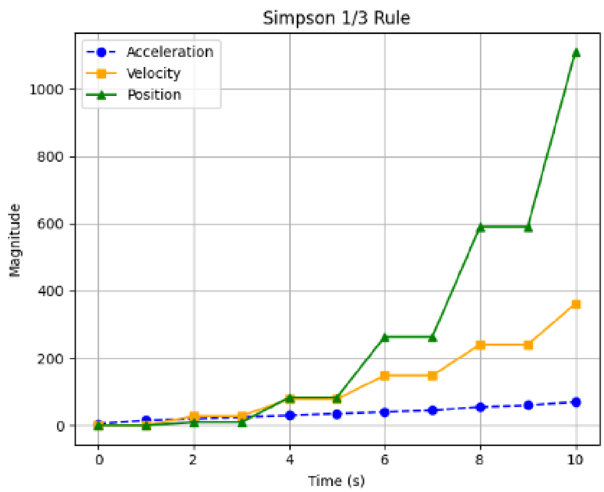
```
    'Position (1/3)': x_13,
    'Velocity (3/8)': v_38,
    'Position (3/8)': x_38
})
methods = ['Trapezoid', '1/3', '3/8']
titles = ['Trapezoidal Rule', 'Simpson 1/3 Rule', 'Simpson 3/8 Rule']
fig, axs = plt.subplots(1, 3, figsize=(18, 5), sharex=True)

for i, method in enumerate(methods):
    axs[i].plot(t, a, label='Acceleration', color='blue', linestyle='--', marker='o')
    axs[i].plot(t, df[f'Velocity ({method})'], label='Velocity', color='orange', marker='s')
    axs[i].plot(t, df[f'Position ({method})'], label='Position', color='green', marker='^')
    axs[i].set_title(titles[i])
    axs[i].set_xlabel('Time (s)')
    axs[i].set_ylabel('Magnitude')
    axs[i].legend()
    axs[i].grid(True)

plt.tight_layout()
plt.show()
print(df.round(2))
```

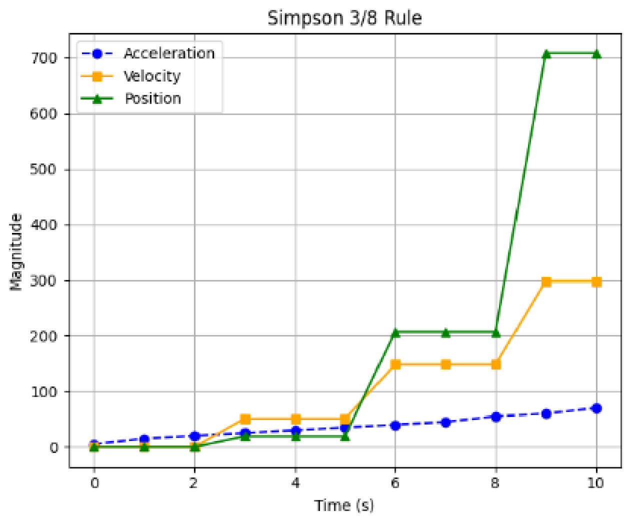
OUTPUT –

```
Time (s)  Acceleration (m/s²)  Velocity (Trapezoid)  Position (Trapezoid)  \
0         0                    5                   0.0                  0.00
1         1                   15                  10.0                  5.00
2         2                   20                  27.5                 23.75
3         3                   25                  50.0                 62.50
4         4                   30                  77.5                126.25
5         5                   35                 110.0                220.00
6         6                   40                 147.5                348.75
7         7                   45                 190.0                517.50
8         8                   55                 240.0                732.50
9         9                   60                 297.5               1001.25
10        10                  70                 362.5               1331.25
```

	Velocity (1/3)	Position (1/3)	Velocity (3/8)	Position (3/8)
0	0.00	0.00	0.00	0.00
1	0.00	0.00	0.00	0.00
2	28.33	9.44	0.00	0.00
3	28.33	9.44	50.62	18.98
4	78.33	82.78	50.62	18.98
5	78.33	82.78	50.62	18.98
6	148.33	262.78	148.12	207.42
7	148.33	262.78	148.12	207.42
8	240.00	590.00	148.12	207.42
9	240.00	590.00	298.12	708.05
10	361.67	1110.56	298.12	708.05







Question 2 :

Use integral definition of ln(x) to compute and plot ln(x) in a given range. Use trapezoidal, Simpson's (1/3 & 3/8 rules) methods and compare the results.

PYTHON CODE –

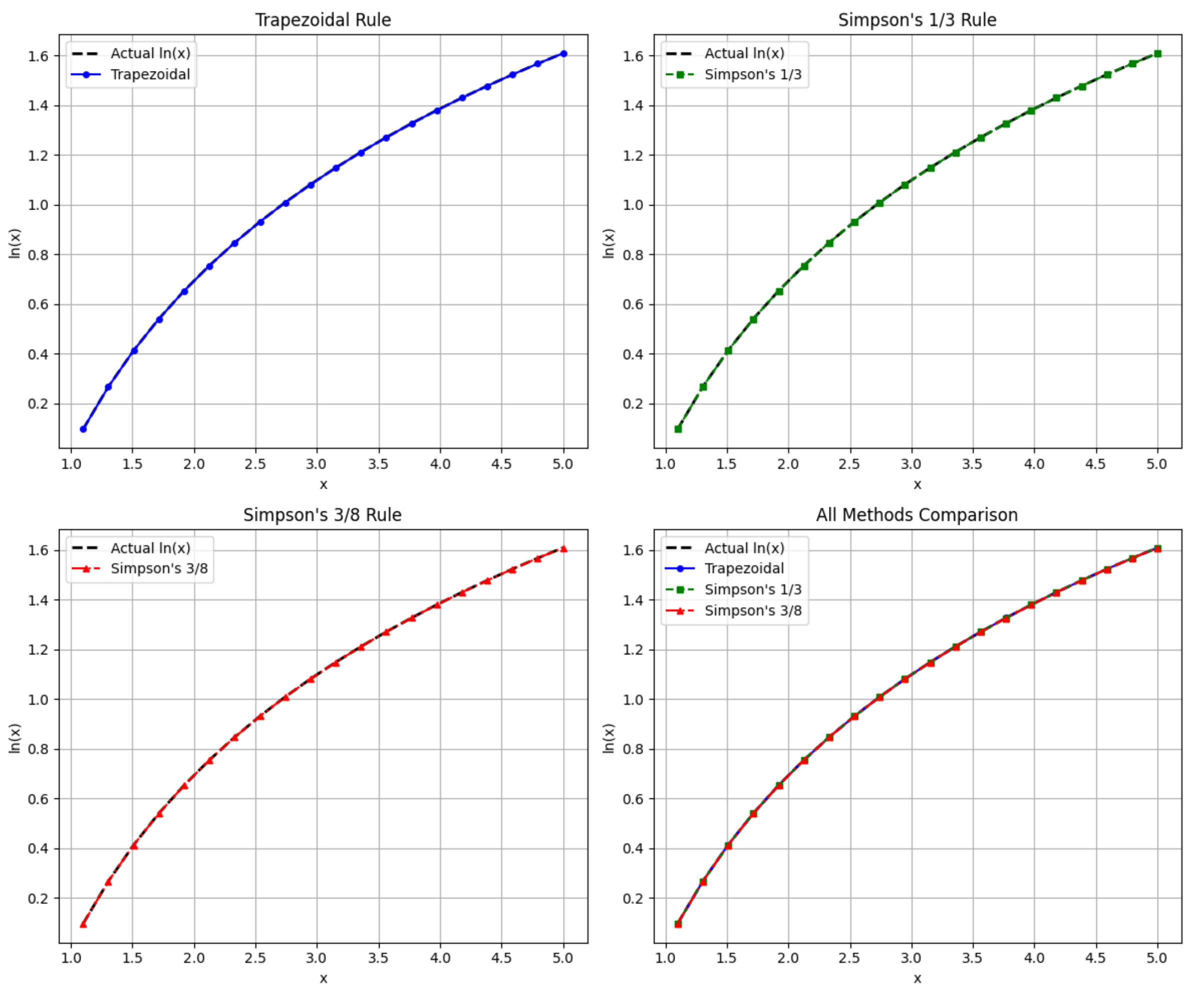
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

def f(t):
    return 1 / t
def trapezoidal(f, a, b, n):
    h = (b - a) / n
    result = f(a) + f(b)
    for i in range(1, n):
        result += 2 * f(a + i * h)
    return (h / 2) * result
def simpson1_3(f, a, b, n):
    if n % 2 != 0:
        n += 1
    h = (b - a) / n
    result = f(a) + f(b)
    for i in range(1, n):
        coeff = 4 if i % 2 != 0 else 2
        result += coeff * f(a + i * h)
    return (h / 3) * result
def simpson3_8(f, a, b, n):
    if n % 3 != 0:
        n += 3 - (n % 3)
    h = (b - a) / n
    result = f(a) + f(b)
    for i in range(1, n):
        coeff = 3 if i % 3 != 0 else 2
```

```
        result += coeff * f(a + i * h)
    return (3 * h / 8) * result

x_vals = np.linspace(1.1, 5, 20)
ln_trap = []
ln_simp_1_3 = []
ln_simp_3_8 = []

for x in x_vals:
    ln_trap.append(trapezoidal(f, 1, x, 100))
    ln_simp_1_3.append(simpson1_3(f, 1, x, 100))
    ln_simp_3_8.append(simpson3_8(f, 1, x, 99))

ln_actual = np.log(x_vals)
data = {'x': x_vals,'Actual ln(x)': ln_actual,'Trapezoidal': ln_trap,
    "Simpson's 1/3": ln_simp_1_3,"Simpson's 3/8": ln_simp_3_8
}
df = pd.DataFrame(data)
print(df.head())

fig, axs = plt.subplots(2, 2, figsize=(12, 10))

axs[0, 0].plot(x_vals, ln_actual, 'k--', label='Actual ln(x)', linewidth=2)
axs[0, 0].plot(x_vals, ln_trap, 'bo-', label='Trapezoidal', markersize=4)
axs[0, 0].set_title('Trapezoidal Rule')
axs[0, 0].legend()
axs[0, 0].grid(True)

axs[0, 1].plot(x_vals, ln_actual, 'k--', label='Actual ln(x)', linewidth=2)
axs[0, 1].plot(x_vals, ln_simp_1_3, 'gs--', label="Simpson's 1/3", markersize=4)
axs[0, 1].set_title("Simpson's 1/3 Rule")
axs[0, 1].legend()
axs[0, 1].grid(True)
```

```
axs[1, 0].plot(x_vals, ln_actual, 'k--', label='Actual ln(x)', linewidth=2)
axs[1, 0].plot(x_vals, ln_simp_3_8, 'r^-.', label="Simpson's 3/8", markersize=4)
axs[1, 0].set_title("Simpson's 3/8 Rule")
axs[1, 0].legend()
axs[1, 0].grid(True)

axs[1, 1].plot(x_vals, ln_actual, 'k--', label='Actual ln(x)', linewidth=2)
axs[1, 1].plot(x_vals, ln_trap, 'bo-', label='Trapezoidal', markersize=4)
axs[1, 1].plot(x_vals, ln_simp_1_3, 'gs--', label="Simpson's 1/3", markersize=4)
axs[1, 1].plot(x_vals, ln_simp_3_8, 'r^-.', label="Simpson's 3/8", markersize=4)
axs[1, 1].set_title('All Methods Comparison')
axs[1, 1].legend()
axs[1, 1].grid(True)

for ax in axs.flat:
    ax.set(xlabel='x', ylabel='ln(x)')
plt.tight_layout()
plt.show()
```

OUTPUT –

	x	Actual ln(x)	Trapezoidal	Simpson's 1/3	Simpson's 3/8
0	1.100000	0.095310	0.095310	0.095310	0.095310
1	1.305263	0.266405	0.266405	0.266405	0.266405
2	1.510526	0.412458	0.412459	0.412458	0.412458
3	1.715789	0.539873	0.539876	0.539873	0.539873
4	1.921053	0.652873	0.652878	0.652873	0.652873



Question 3 :

The work done by a constat temperature, pressure-volume thermodynamics process can be computed as

$$W = \int p \, dv$$

Where W is work, p is pressure and v is volume. Using a combination of the trapezoidal, Simpson's (1/3 & 3/8 rules) methods and compare the results.

Use the following data to compute the work done in kJ.

Pressure (kPa)	336	294.4	266.4	260.8	260.5	249.6	193.6	165.6
Volume (m^{-3})	0.5	2	3	4	6	8	10	11

PYTHON CODE –

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

pressure = np.array([336, 294.4, 266.4, 260.8, 260.5, 249.6, 193.6, 165.6])
volume = np.array([0.5, 2, 3, 4, 6, 8, 10, 11])

def trapezoidal_rule(x, y):
    integral = 0
    for i in range(len(x) - 1):
        integral += (x[i+1] - x[i]) * (y[i] + y[i+1]) / 2
    return integral

def simpsons1_3rule(x, y):
    if (len(x) - 1) % 2 != 0:
        print("Simpson's 1/3 rule requires an even number of intervals. Using only valid
points.")
        x, y = x[:-1], y[:-1]

    n = len(x) - 1
    h = (x[-1] - x[0]) / n
    integral = y[0] + y[-1]

    for i in range(1, n, 2):
        integral += 4 * y[i]
    for i in range(2, n-1, 2):
        integral += 2 * y[i]

    return (h / 3) * integral

def simpsons3_8rule(x, y):
    if (len(x) - 1) % 3 != 0:
```

```
        print("Simpson's 3/8 rule requires intervals in multiples of 3. Using only valid points.")
        valid_points = (len(x) - 1) - (len(x) - 1) % 3
        x, y = x[:valid_points + 1], y[:valid_points + 1]

    n = len(x) - 1
    h = (x[-1] - x[0]) / n
    integral = y[0] + y[-1]

    for i in range(1, n):
        if i % 3 == 0:
            integral += 2 * y[i]
        else:
            integral += 3 * y[i]

    return (3 * h / 8) * integral

work_trap = trapezoidal_rule(volume, pressure)

try:
    work_simp_1_3 = simpsons1_3rule(volume, pressure)
except Exception as e:
    work_simp_1_3 = str(e)

try:
    work_simp_3_8 = simpsons3_8rule(volume, pressure)
except Exception as e:
    work_simp_3_8 = str(e)

results = pd.DataFrame({
    "Method": ["Trapezoidal", "Simpson's 1/3", "Simpson's 3/8"],
    "Work Done (kJ)": [work_trap, work_simp_1_3, work_simp_3_8]
})
print(results)
```

```
plt.figure(figsize=(8, 6))
plt.plot(results["Method"], results["Work Done (kJ)"], marker='o', linestyle='-', color='purple')
plt.xlabel("Method")
plt.ylabel("Work Done (kJ)")
plt.title("Comparison of Numerical Integration Methods")
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT –

```
Simpson's 1/3 rule requires an even number of intervals. Using only valid points.
Simpson's 3/8 rule requires intervals in multiples of 3. Using only valid points.
          Method  Work Done (kJ)
0    Trapezoidal     2671.000000
1  Simpson's 1/3     2534.705556
2  Simpson's 3/8     2531.690625
```