EM 215 – NUMERICAL METHODS

LAB01 ASSIGNMENT – 1

E/20/280

Pathirage R.S.

Q1)

- Forward difference approximation

Step size(h)	f'(0.5)	Absolute error(e)
1.0000000000	-2.0000000000	0.9500000000
0.5000000000	-1.4375000000	0.3875000000
0.2500000000	-1.2265625000	0.1765625000
0.1250000000	-1.1345703125	0.0845703125
0.0625000000	-1.0914306641	0.0414306641
0.0312500000	-1.0705108643	0.0205108643
0.0156250000	-1.0602054596	0.0102054596
0.0078125000	-1.0550903797	0.0050903797
0.0039062500	-1.0525421202	0.0025421202
0.0019531250	-1.0512702949	0.0012702949
0.0009765625	-1.0506349565	0.0006349565

FIGURE1: Forward difference approximation

- Backward difference approximation

Step size(h)	f'(0.5)	Absolute error(e)
1.0000000000	-0.5000000000	0.5500000000
0.5000000000	-0.7625000000	0.2875000000
0.2500000000	-0.8984375000	0.1515625000
0.1250000000	-0.9716796875	0.0783203125
0.0625000000	-1.0101318359	0.0398681641
0.0312500000	-1.0298797607	0.0201202393
0.0156250000	-1.0398921967	0.0101078033
0.0078125000	-1.0449340343	0.0050659657
0.0039062500	-1.0474639833	0.0025360167
0.0019531250	-1.0487312309	0.0012687691
0.0009765625	-1.0493654250	0.0006345750

FIGURE2: Backward difference approximation

- Centered difference approximation

Step size(h)	f'(0.5)	Absolute error(e)
1.0000000000	-1.2500000000	0.2000000000
0.5000000000	-1.1000000000	0.0500000000
0.2500000000	-1.0625000000	0.0125000000
0.1250000000	-1.0531250000	0.0031250000
0.0625000000	-1.0507812500	0.0007812500
0.0312500000	-1.0501953125	0.0001953125
0.0156250000	-1.0500488281	0.0000488281
0.0078125000	-1.0500122070	0.0000122070
0.0039062500	-1.0500030518	0.0000030518
0.0019531250	-1.0500007629	0.0000007629
0.0009765625	-1.0500001907	0.0000001907

FIGURE3: Centered difference approximation

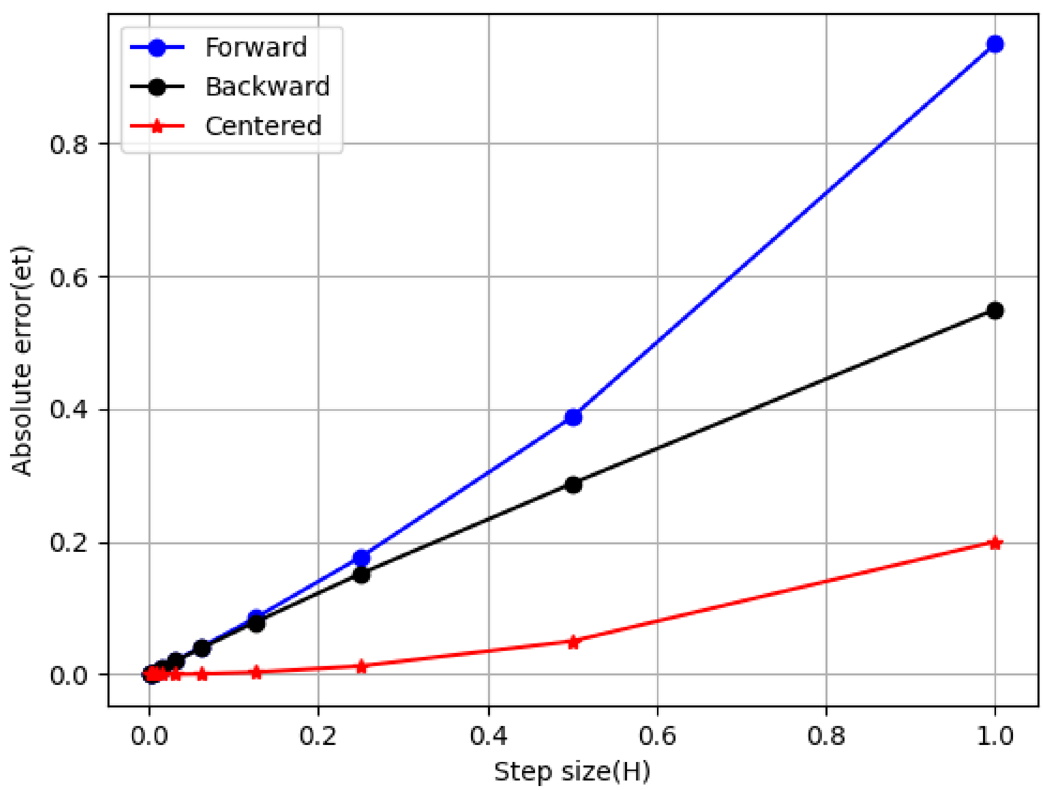

FIGURE4: Absolute errors of forward, backward, centered , values plotted against step size

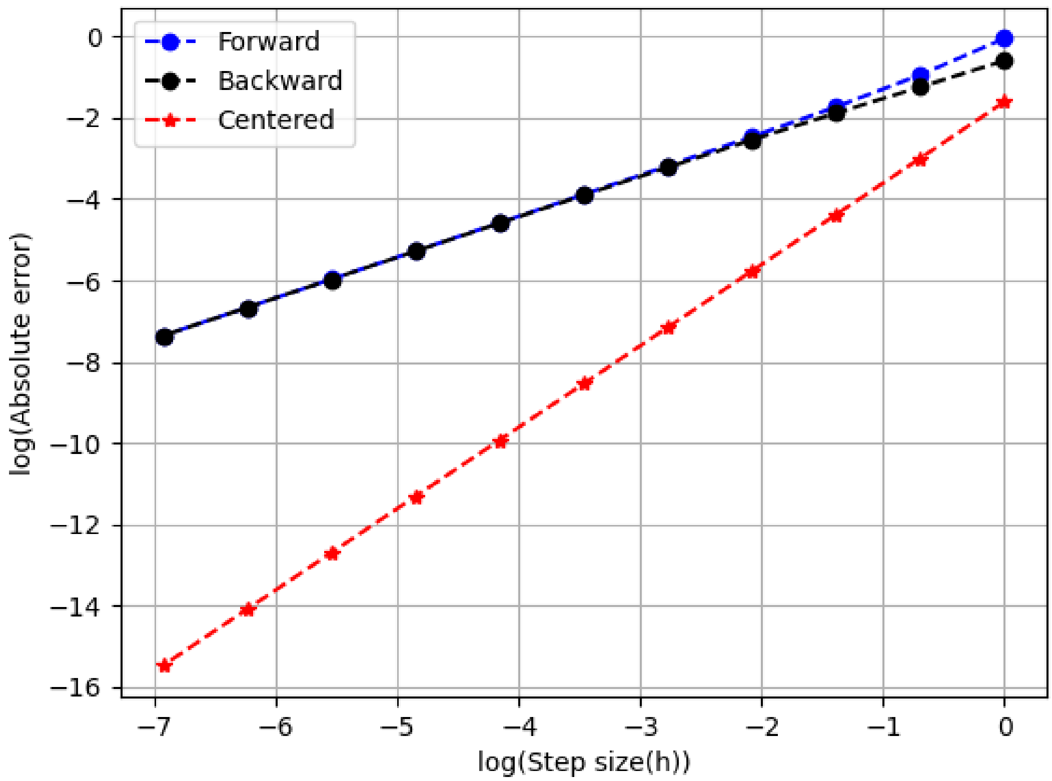

FIGURE4: Absolute errors of forward, backward, centered , values plotted against step size

```python
import numpy as np
import matplotlib.pyplot as plt
from tabulate import tabulate
# Define the function and its derivative
def f(x):
    return -0.1*x**4 - 0.5*x**2 - 0.5*x + 1.2
def f_prime(x):
    return -0.4*x**3 - x - 0.5

# Initial values
xi = 0.5
f1xi = f_prime(xi)

# Different step sizes
h_values = np.array([1, 1/2, 1/2**2, 1/2**3, 1/2**4, 1/2**5, 1/2**6, 1/2**7, 1/2**8, 1/2**9, 1/2**10])

# Initialize result arrays
res = []
etF = []
etB = []
etC = []
F = []
B = []
C = []
```

```python
# Calculate derivatives and errors for each step size
for h in h_values:
    xip1 = xi + h
    xim1 = xi - h

    fxip1 = f(xip1)
    fxim1 = f(xim1)

    dxF = (fxip1 - f(xi)) / h  # forward
    dxB = (f(xi) - fxim1) / h  # backward
    dxC = (fxip1 - fxim1) / (2 * h)  # centered

    et = [f1xi - dxF, f1xi - dxB, f1xi - dxC]  # true error
    res.append([h, f1xi, dxF, dxB, dxC, np.abs(et)])

    F.append(dxF)
    B.append(dxB)
    C.append(dxC)
    etF.append(et[0])
    etB.append(et[1])
    etC.append(et[2])

# Convert results to numpy arrays
res = np.array(res, dtype=object)
etF = np.array(etF)
etB = np.array(etB)
etC = np.array(etC)
F = np.array(F)
B = np.array(B)
C = np.array(C)

#draw the table
head = ["Step size(h)", "f'(0.5)", "Absolute error(e)"]
#Forward
abs_etF = [abs(x) for x in etF]
data = [[f"{h:.10f}", f"{df:.10f}", f"{er:.10e}"] for h, df, er in zip(h_values, F, abs_etF)]
# display table
print(tabulate(data, headers=head, tablefmt="grid"))

# Backward
abs_etB = [abs(x) for x in etB]
data = [[f"{h:.10f}", f"{df:.10f}", f"{er:.10e}"] for h, df, er in zip(h_values, B, abs_etB)]
# display table
print(tabulate(data, headers=head, tablefmt="grid"))
```

```python
# Centered
abs_etC = [abs(x) for x in etC]
data = [[f"{h:.10f}", f"{df:.10f}", f"{er:.10e}"] for h, df, er in zip(h_values, C, abs_etC)]
# display table
print(tabulate(data, headers=head, tablefmt="grid"))

# Plot truncation error vs step size
plt.figure()
P1, = plt.plot(h_values, abs_etF, '-ob', label='Forward')  # Forward
P2, = plt.plot(h_values, abs_etB, '-ok', label='Backward')  # Backward
P3, = plt.plot(h_values, abs_etC, '-*r', label='Centered')  # Centered
plt.xlabel('Step size(H)')
plt.ylabel('Absolute error(et)')
plt.legend()
plt.grid(True)

# Plot log-log graph
plt.figure()
P1L, = plt.plot(np.log(h_values), np.log(abs_etF), '--ob', label='Forward')  # Forward
P2L, = plt.plot(np.log(h_values), np.log(abs_etB), '--ok', label='Backward')  # Backward
P3L, = plt.plot(np.log(h_values), np.log(abs_etC), '--*r', label='Centered')  # Centered
plt.xlabel('log(Step size(h))')
plt.ylabel('log(Absolute error)')
plt.legend()
plt.grid(True)
plt.show()
```

FIGURE5: Python code used

Q2)

a)

Analitical solution for v,

$$\frac{dv}{dt} = g - \frac{c}{m} v$$

$$\frac{dv}{dt} + \frac{c}{m} v = g$$

$\times e^{\int \frac{c}{m} dt}$

$$v e^{\int \frac{c}{m} dt} = \int g e^{\int \frac{c}{m} dt}$$

$$v e^{\frac{c}{m} t} = \frac{m}{c} g e^{\frac{c}{m} t} + k$$

$$\boxed{v = \frac{m}{c} g + k e^{-\frac{c}{m} t}} \text{ --- (A)}$$

$t = t_x, \quad v = v_x$

$$v_x = \frac{m}{c} g + k e^{-\frac{c}{m} t_x}$$

$$k = \left(v_x - \frac{m}{c} g\right) e^{\frac{c}{m} t_x}$$

$k \to (1),$

$$v = \frac{m}{c} g + k e^{-\frac{c}{m} t}$$

$$\boxed{v = \frac{m}{c} g - \frac{m}{c}\left(g - \frac{m}{c} v_x\right) e^{\frac{c}{m}(t_x - t)}}$$

b)

Numerical

$t = 0 - 10, \quad t_x = 10s, \quad V_x = 44.87 m/s$

$$\frac{v_i - v_{i-1}}{dt} = g - \frac{m}{c} v_i$$

$$v_{i-1} = \left(\frac{c}{m} v_i - g\right) dt - v_i$$

c)

d)

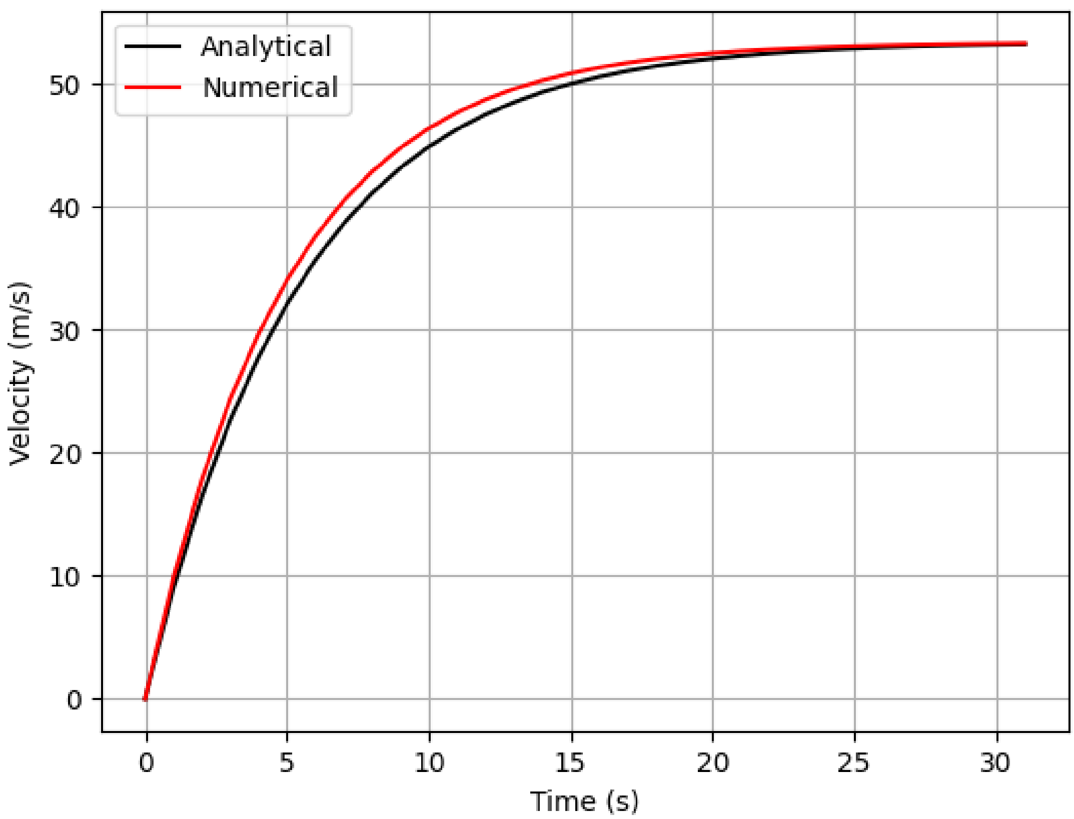

FIGURE6: Velocity from Analytical and Numerical methods vs Time with dt = 1s

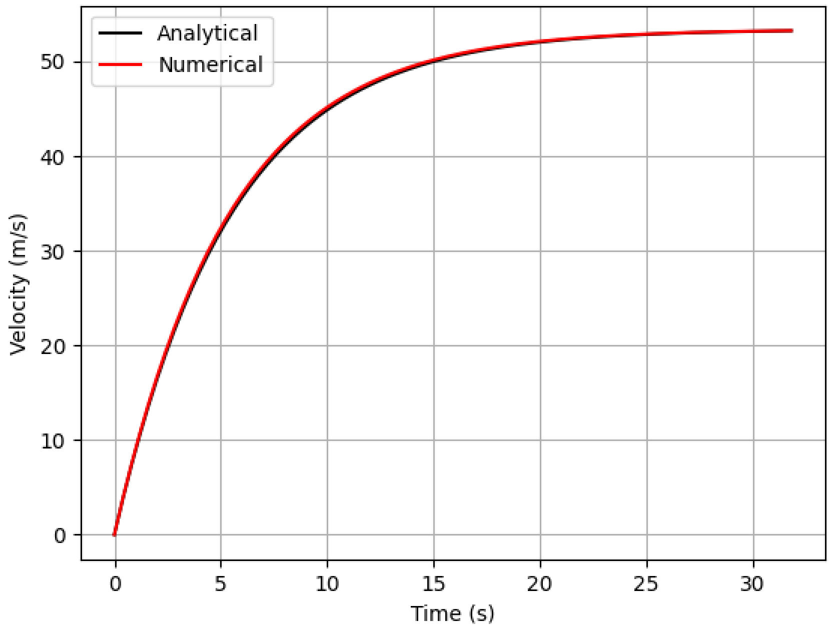

FIGURE7: Velocity from Analytical and Numerical methods vs Time with dt = 0.2s

e) As the graphs 6 and 7 suggest lower the dt value, more accurate numerical answer gets due to the reduction of truncation error. How ever if we try to reduce dt significantly, it may lead to more error due the incretion of truncation error.