PAPER

Brandon Finley HW #6 APPM 4650

☆ #4 (a)(b)(c)(d) on next page, sorry ☆

A) $y'(t) = y \quad 0 \le t \le 1 \quad y(0) = 1$; IVP solution: $y(t) = e^t$

i) $t_i = a + i \cdot h$

Use Taylor's Theorem...

$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i) \cdot y'(t_i) + \frac{(t_{i+1} - t_i)^2}{2} y''(\xi_i)$$

$h = t_{i+1} - t_i$

$$\Rightarrow y(t_{i+1}) = y(t_i) + h \cdot y'(t_i) + \frac{h^2}{2} y''(\xi_i)$$

for h small...

$$y(t_{i+1}) = y(t_i) + h \cdot y'(t_i) + 0$$

Apply IC... $(t_i = 0)$

$$y(t_{i+1}) = y(0) + h \cdot y'(0) \quad \Rightarrow w_0 = \alpha$$

$$= 1 + h \cdot 0 = 1 = \alpha$$

and we know $w_{j+1} = w_j + h \cdot f(t_i, w_i)$

so, $\boxed{w_{j+1} = w_j + h \cdot t_j \quad \text{with} \quad \underset{t_i = 0}{y(0)} = 1 = w_0}$

ii) $y'(t) = f(t, y) = y$

$$\frac{d}{dt}[f(t,y)] = f'(t,y) = 0$$

$$T^{(2)}(t_i, w_i) = f(t_i, w_i) + \frac{h}{2} \cdot f'(t_i, w_i)$$

$$= w_i + \frac{h}{2} \cdot 0 = w_i$$

$w_0 = 1$

$$w_{i+1} = w_i + h \cdot T^{(2)}(t_i, w_i) = w_i + h \cdot w_i = (h+1) w_i$$

$$\Rightarrow \boxed{w_{j+1} = w_j(1+h) \text{ with } w_0 = 1}$$

3a) $w_0 = \alpha$

$$w_{i+1} = \underbrace{w_i + a_1 \cdot f(t_i, w_i)}_{\text{Euler's}} + \underbrace{a_2 \cdot f(t_i + \alpha_2, w_i + \delta_2 \cdot f(t_i, w_i}_{\text{1 order higher}}$$

This means our function for approximation is Euler's plus 1 more order higher, implying that the $O(h'')$ is two orders higher than Euler's. This means the total error for w_{i+1} is $1+1 \Rightarrow$ Order 2. Then, we know local truncation error is one more than global $\Rightarrow \therefore$ LTE is $O(h^3)$. And so it cannot be $O(h^4)$ since it is most $O(h^3)$.

#4) $D = \{(t,y) \mid 0 \le t \le 1, -\infty < y < \infty\}$; $y(0) = 1$; $y' = f(t,y)$

a) $f(t,y) = e^{t-y} = e^t \cdot e^{-y} \Rightarrow \frac{\partial f}{\partial y} = e^t \cdot -1 \cdot e^{-y}$

$\Rightarrow |e^t \cdot -1 \cdot e^{-y}||y_1 - y_2| = |e^{t-y}||y_1 - y_2|$ $\boxed{\therefore \text{ Doesn't satisfy lipschitz}}$

$\boxed{\therefore \text{ automatically is not well posed}}$

b) $f(t,y) = \frac{1+y}{1+t} \Rightarrow$ not continuous at $t = -1$ & $[0,1]$ $\not\ni$ $\therefore$ continuous on $[0,1] \ni t$

$\frac{\partial f}{\partial y} = \frac{\partial}{\partial y}\left(\frac{1+y}{1+t}\right) = \frac{1}{1+t} \cdot \frac{\partial}{\partial y}(1+y) = \frac{1}{1+t}(1) = \frac{1}{1+t} \le L$

$\max\left\{\frac{1}{1+t}\right\} \in [0,1] \Rightarrow \frac{1}{1+0} = \boxed{1 = L = \text{lip. const.}}$

$\boxed{\therefore \text{ Satisfies lipschitz cond. + well-posed}}$

c) $f(t,y) = \cos(yt) \Rightarrow$ continuous on $t:[0,1]$

$\frac{\partial f}{\partial y} = \frac{\partial}{\partial y}(\cos yt) = |-\sin(yt) \cdot t| \le |\cancel{\sin yt}||t| \le L \Rightarrow \boxed{L = 1}$

$\boxed{\therefore \text{ Satisfies lipschitz cond. + well-posed}}$

d) $f(t,y) = \frac{y^2}{1+t} \Rightarrow$ not continuous at $t = -1$ & $t \in [0,1]$ $\not\ni$ $\therefore$ continuous on $[0,1] \ni t$

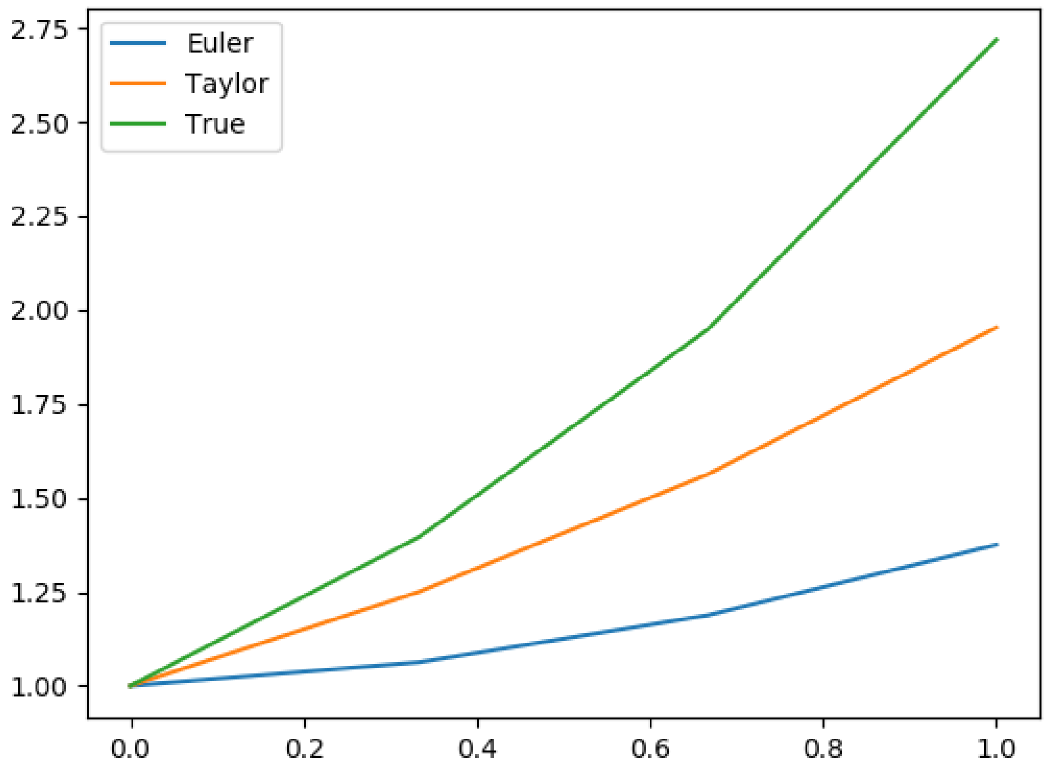
$\frac{\partial f}{\partial y} = \frac{\partial}{\partial y}\left(\frac{y^2}{1+t}\right) = \frac{1}{1+t} \cdot \frac{\partial}{\partial y}(y^2) = \frac{1}{1+t}(2y) = \frac{2y}{1+t}$

$\frac{2y}{1+t} \Rightarrow$ unbounded on $|y| < \infty$ $\therefore$ does not satisfy Lip

$\boxed{\therefore f \text{ does not satisfy lipschitz cond. is not well-posed}}$

RESULTS

```
Euler Error: 1.343281828459045
Taylor Error: 0.7651568284590451
```



```
-------------------------
h value:  0.5
Euler:  1.75  ||  0.04812994723872266
Midpoint:  1.7448534278929364  ||  0.04298337513165906
Modified Euler:  1.6957898094433836  ||  0.0060802433178936965
Runge-Kutta (Order 4):  1.7018264642602912  ||  4.358850098618028e-05
-------------------------
h value:  0.25
Euler:  1.3507223041859748  ||  0.3511477485753025
Midpoint:  1.7060506236181863  ||  0.004180570856908927
Modified Euler:  1.6922889668511303  ||  0.009581085910147058
Runge-Kutta (Order 4):  1.7017875639626732  ||  8.248879860417446e-05
-------------------------
h value:  0.125
Euler:  1.2305343696536553  ||  0.4713356831076221
Midpoint:  1.7025170254706268  ||  0.000646972709349436
Modified Euler:  1.699287637730054  ||  0.0025824150312232508
Runge-Kutta (Order 4):  1.701864352942993  ||  5.699818284243818e-06
-------------------------
h value:  0.0625
Euler:  1.1405003072165938  ||  0.5613697455446836
Midpoint:  1.7019992004527422  ||  0.00012914769146488148
Modified Euler:  1.7012199627456992  ||  0.0006500900155781419
Runge-Kutta (Order 4):  1.701869695391401  ||  3.57369876358149e-07
-------------------------
h value:  0.03125
Euler:  1.0797655309204708  ||  0.6221045218408066
Midpoint:  1.7018990954746076  ||  2.904271333026287e-05
Modified Euler:  1.701707562388208  ||  0.0001624903730692573
Runge-Kutta (Order 4):  1.7018700305682473  ||  2.21930300803308e-08
-------------------------
h value:  0.015625
Euler:  1.0429912456817594  ||  0.658878807079518
Midpoint:  1.7018769531574733  ||  6.900396195996805e-06
Modified Euler:  1.7018294586102383  ||  4.0594151039075044e-05
Runge-Kutta (Order 4):  1.7018700513810103  ||  1.3802670117968319e-09
-------------------------
h value:  0.0078125
Euler:  1.022407530804163  ||  0.6794625219571144
Midpoint:  1.7018717354654984  ||  1.6827042210731946e-06
Modified Euler:  1.7018599089806694  ||  1.0143780607929642e-05
Runge-Kutta (Order 4):  1.7018700526752566  ||  8.60207460817719e-11
-------------------------
h value:  0.00390625
Euler:  1.0114530161951432  ||  0.6904170365661342
Midpoint:  1.701870468297483  ||  4.1553620566681104e-07
Modified Euler:  1.7018675174766877  ||  2.53528458959984e-06
Runge-Kutta (Order 4):  1.7018700527559087  ||  5.368594457877407e-12
```

CODE

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

def f(t, w):
    return y
def true_f(t):
    return math.exp(t)
def Euler(a, b, alp, h):
    w_arr = np.zeros((int((b - a)/h)))
    w_arr[0] = alp
    for i in range(1, int((b - a)/h)):
```

```
        t = a + i*h
        w_arr[i] = w_arr[i - 1] + h*t
    return w_arr
def Taylor2(a, b, alp, h):
    w_arr = np.zeros((int((b - a)/h)))
    w_arr[0] = alp
    for i in range(1, int((b - a)/h)):
        w_arr[i] = (h + 1)*w_arr[i - 1]
    return w_arr
def main():
    a = 0
    b = 1
    alpha = 1
    h = 1/4
    x = np.linspace(0, 1, int((b-a)/h))
    euler = Euler(a, b, alpha, h)
    tay2 = Taylor2(a, b, alpha, h)
    y = np.zeros((len(x)))
    for i in range(0, len(x)):
        y[i] = true_f(x[i])
    plt.plot(x, euler, label='Euler')
    plt.plot(x, tay2, label='Taylor')
    plt.plot(x, y, label='True')
    plt.legend()

    error_euler = abs(euler[-1] - true_f(1))
    error_tay = abs(tay2[-1] - true_f(1))
    print('Euler Error:', error_euler)
    print('Taylor Error:', error_tay)

    plt.show()

if __name__ == "__main__":
    main()
```

```
import numpy as np
import math

def f(t, w):
    return -t*w + 4*t/w

def RungeKutta4(a, b, alp, h):
    w = alp
```

```
    for i in range(0, int((b - a)/h)):
        t = a + i*h
        tp1 = a + (i+1)*h
        k1 = h*f(t, w)
        k2 = h*f(t + h/2, w + k1/2)
        k3 = h*f(t + h/2, w + k2/2)
        k4 = h*f(tp1, w + k3)
        w = w + 1/6*(k1 + 2*k2 + 2*k3 + k4)
    return w

def Midpoint(a, b, alp, h):
    w = alp
    for i in range(0, int((b - a)/h)):
        t = a + h*i
        w = w + h*f(t + h/2, w + (h/2)*f(t, w))
    return w

def ModifiedEuler(a, b, alp, h):
    w = alp
    for i in range(0, int((b - a)/h)):
        t = a + h*i
        tp1 = a + h*(i + 1)
        w = w + (h/2)*(f(t, w) + f(tp1, w + h*f(t, w)))
    return w

def Euler(a, b, alp, h):
    w = alp
    for i in range(0, int((b - a)/h)):
        t = a + h*i
        w = alp + h*f(t, w)
    return w

def h_array(num):
    h = np.zeros((num - 2,))
    for i in range(1, num - 1):
        h[i - 1] = (1/2**i)
    return h


def ErrorStats(euler, mid, mod, rk4):
    true = 1.7018700527612773
    error_euler = abs(euler - true)
    error_mid = abs(mid - true)
    error_mod = abs(mod - true)
    error_rk4 = abs(rk4 - true)
```

```python
    print('Euler: ', euler, ' || ', error_euler)
    print('Midpoint: ', mid, ' || ', error_mid)
    print('Modified Euler: ', mod, ' || ', error_mod)
    print('Runge-Kutta (Order 4): ', rk4, ' || ', error_rk4)

def main():
    a = 0
    b = 1
    alpha = 1
    N = 10

    h_arr = h_array(N)
    for i in range(0, len(h_arr)):
        h = h_arr[i]
        print('--------------------------', '\nh value: ', h)
        euler = Euler(a, b, alpha, h)
        mid = Midpoint(a, b, alpha, h)
        mod = ModifiedEuler(a, b, alpha, h)
        RK4 = RungeKutta4(a, b, alpha, h)
        ErrorStats(euler, mid, mod, RK4)


if __name__ == "__main__":
    main()
```