MATH 151B - Applied Numerical Methods - Homework 1

Darren Tsang, 405433124, Discussion 1A

Produced on Wednesday, Aug. 12 2020 @ 05:55:01 PM

Question 1

Part A

We are given the following:

$$\frac{dy}{dt} = \frac{1+y}{t} = f(t,y), \quad 1 \leq t \leq 2, \quad y(1) = 2$$

This means that:

$$D = \{(t,y) |\ 1 \leq t \leq 2,\ -\infty < y < \infty\}$$

We notice that:

$$\max_{(t,y)\in D} \left(\left| \frac{\partial}{\partial y} \frac{(1+y)}{t} \right| \right) = \max_{(t,y)\in D} \left(\left| \frac{1}{t} \right| \right) \leq 1 = L$$

By Theorem 5.3, $f(t,y) = (1+y)/t$ satisfies a Lipschitz condition in y on D with Lipschitz constant $L = 1$. Furthermore, f is continuous on D, which means the IVP is well-posed by Theorem 5.6.

Part B

We are given the following:

$$\frac{dy}{dt} = y\cos(t) = f(t,y), \quad 0 \leq t \leq 1, \quad y(0) = 1$$

This means that:

$$D = \{(t,y) |\ 0 \leq t \leq 1,\ -\infty < y < \infty\}$$

We notice that:

$$\max_{(t,y)\in D} \left(\left| \frac{\partial}{\partial y} y\cos(t) \right| \right) = \max_{(t,y)\in D} \left(|\cos(t)| \right) \leq 1 = L$$

By Theorem 5.3, $f(t,y) = y\cos(t)$ satisfies a Lipschitz condition in y on D with Lipschitz constant $L = 1$. Furthermore, f is continuous on D, which means the IVP is well-posed by Theorem 5.6.

Question 2

Part A

Using Euler's method for $h = .5$:

$$y(1) = 2,$$
$$y(1.5) \approx y(1) + hf(1, y(1)) = 2 + .5\left(\frac{1+1}{1+2}\right) = 2.333333,$$
$$y(2) \approx y(1.5) + hf(1.5, y(1.5)) = 2.333333 + .5\left(\frac{1+1.5}{1+2.333333}\right) = 2.708333$$

Part B

```
import pandas as pd

def euler(func, start, end, initial, h):
  N = (end - start)/h
  t = [start]
  w = [initial]
  for i in range(1, round(N+1)):
    w.append(w[i-1] + h*func(t[i-1], w[i-1]))
    t.append(start + i*h)

  return(pd.DataFrame(data={'t':t, 'y_approx(t)':w}))
```

```
def func_q2(t,y):
  return (1+t)/(1+y)
```

```
print(euler(func_q2, 1, 2, 2, h = .5))
```

```
##      t  y_approx(t)
## 0  1.0     2.000000
## 1  1.5     2.333333
## 2  2.0     2.708333
```

```
print(euler(func_q2, 1, 2, 2, h = .2).tail(1))
```

```
##      t  y_approx(t)
## 5  2.0     2.729166
```

```
print(euler(func_q2, 1, 2, 2, h = .1).tail(1))
```

```
##       t  y_approx(t)
## 10  2.0     2.735541
```

```
print(euler(func_q2, 1, 2, 2, h = .01).tail(1))
```

```
##        t  y_approx(t)
## 100  2.0     2.741057
```

The results are above and in the table below.

Part C

The actual value of $y(2)$ is:

$$y(2) = \sqrt{2^2 + 2(2) + 6} - 1 \approx 2.741657$$

h	Approximation of y(2)	Actual y(2)	Absolute Error
.5	2.708333	2.741657	.033337
.2	2.729166	2.741657	.012504
.1	2.735541	2.741657	.006129
.01	2.741057	2.741657	.000613

As expected, the error gets smaller as h decreases.

Question 3

Part A

From the chain rule:

$$\frac{d}{dt}f(t,y) = \frac{\partial f(t,y)}{\partial t} + \frac{\partial f(t,y)}{\partial y}\frac{\partial y}{\partial t} = (-y^2e^{-t}) + (2ye^{-t})(y^2e^{-t}) = y^2e^{-t}(2ye^{-t}-1)$$

Part B

Using Euler's method

$$y(0) = 1,$$
$$y(.5) \approx y(0) + hf(0, y(0)) = 1 + .5(1^2e^{-0}) = 1.5$$
$$y(1) \approx y(.5) + hf(.5, y(.5)) = 1.5 + .5(1.5^2e^{-.5}) = 2.182347$$

Using Taylor method of order 2

$$y(0) = 1,$$
$$y(.5) \approx y(0) + hf(0, y(0)) + \frac{h^2}{2}\frac{df}{dt}\bigg|_{(0,y(0))} = 1 + .5(1^2e^{-0}) + \frac{.5^2}{2}(1^2e^{-0}(2*1*e^{-0}-1)) = 1.625000,$$
$$y(1) \approx y(.5) + hf(.5, y(.5)) + \frac{h^2}{2}\frac{df}{dt}\bigg|_{(.5,y(.5))} = 1.625 + .5(1.625^2e^{-.5}) + \frac{.5^2}{2}(1.625^2e^{-.5}(2*1.625*e^{-.5}-1)) = 2.620252$$

Part C

```
import math

def taylor_two(func, func_prime, start, end, initial, h):
  N = (end - start)/h
  t = [start]
  w = [initial]
  for i in range(1, round(N+1)):
    w.append(w[i-1] + h*func(t[i-1], w[i-1]) + h*h*.5*func_prime(t[i-1], w[i-1]))
    t.append(start + i*h)

  return(pd.DataFrame(data={'t':t, 'y_approx(t)':w}))
```

```
def func_q3(t, y):
  return(y*y*math.exp(-t))

def func_prime_q3(t, y):
  return(func_q3(t, y)*(2*y*math.exp(-t) - 1))
```

Using Euler's method

```
print(euler(func_q3, 0, 1, 1, h = .5))
```

```
##      t  y_approx(t)
## 0  0.0     1.000000
## 1  0.5     1.500000
## 2  1.0     2.182347
```

```
print(euler(func_q3, 0, 1, 1, h = .1).tail(1))
```

```
##       t  y_approx(t)
## 10  1.0     2.531887
```

```
print(euler(func_q3, 0, 1, 1, h = .01).tail(1))
```

```
##        t  y_approx(t)
## 100  1.0     2.695519
```

Using Taylor method of order 2

```
print(taylor_two(func_q3, func_prime_q3, 0, 1, 1, h = .5))
```

```
##      t  y_approx(t)
## 0  0.0     1.000000
## 1  0.5     1.625000
## 2  1.0     2.620252
```

```
print(taylor_two(func_q3, func_prime_q3, 0, 1, 1, h = .1).tail(1))
```

```
##       t  y_approx(t)
## 10  1.0      2.71146
```

```
print(taylor_two(func_q3, func_prime_q3, 0, 1, 1, h = .01).tail(1))
```

```
##        t  y_approx(t)
## 100  1.0     2.718205
```

Part D

Since we are given $y(t) = e^t$, we can easily find $y(1)$:

$$y(1) = e^1 = e$$

h	y(1) using Euler's	y(1) using Taylor of order 2	Error using Euler's	Error using Taylor of order 2
.5	2.182347	2.620252	.535935	.098030
.1	2.531887	2.71146	.186395	.006822
.01	2.695519	2.718205	.022763	.000077

The error from using Euler's is much higher than the error from using Taylor's of order 2 for the same h value. Furthermore, as h decreases the error decreases for both methods. Both of these observations are to be expected.