Assignment 1a

ME EN 2450
Ryan Dalby
u0848407

$$\frac{dy}{dt} = 2\frac{Q}{A}\sin^2(t) - \frac{\alpha(1+y)^{3/2}}{A}$$

$A = 850 m^2$ $Q = 325 m^3/s$
$\alpha = 200 m^{3/2}/s$

$y(t=0) = 2m$ $\frac{dy}{dt} = f(t,y)$

$$y_{i+1} = y_i + \left(2\frac{Q}{A}\sin^2(t_i) - \frac{\alpha(1+y_i)^{3/2}}{A}\right)h$$

a) estimate $y(t=1.5)$ w/ $h = 0.5s$

$$y_{i+1} = y_i + \left(2\left(\frac{325}{850}\right)\sin^2(t_i) - \frac{200(1+y_i)^{3/2}}{850}\right)(0.5)$$

y = 2m	t = 0
y = 1.3887	t_i = 0.5
y = 1.0422	t = 1
y = .9696	t = 1.5

$$y(0.5) = 2 + \left(\frac{650}{850}\sin^2(0) - \frac{200(3)^{3/2}}{850}\right)(0.5)$$

$y(0.5) = 1.3887$

$$y(1) = 1.3887 + \left(\frac{650}{850}\sin^2(0.5) - \frac{200(1+1.3887)^{3/2}}{850}\right)(0.5)$$

$y(1) = 1.0422$

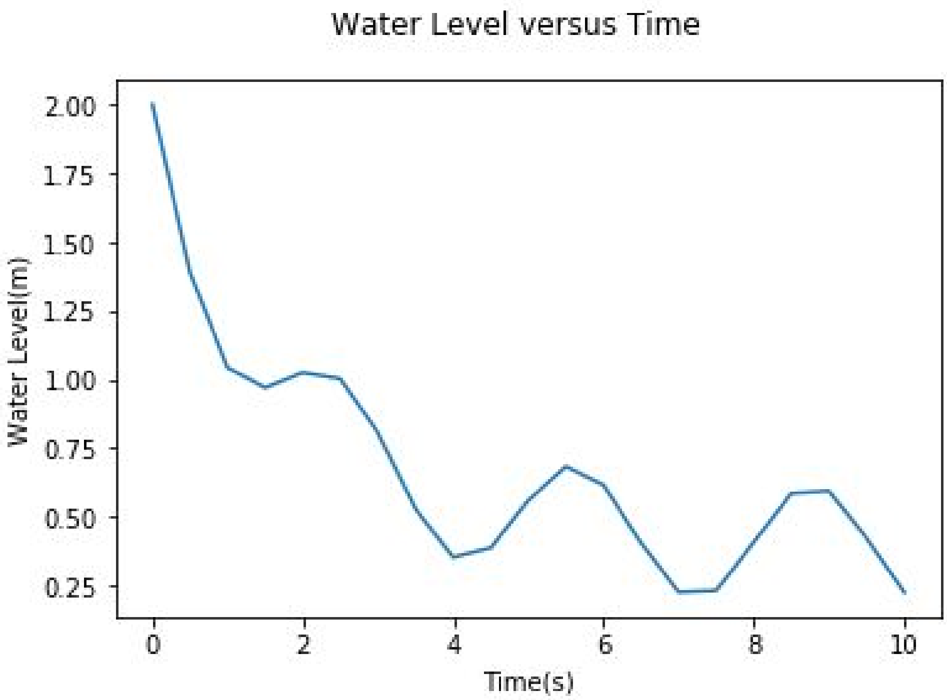
$$y(1.5) = 1.0422 + \left(\frac{650}{850}\sin^2(1) - \frac{200(1+1.0422)^{3/2}}{850}\right)(0.5)$$

$$\boxed{y(1.5) = 0.9696}$$

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 15 16:26:30 2019

@author: Ryan Dalby
u0848407
ME EN 2450
HW1a

"""

#Imports
import numpy as np
import math
import matplotlib.pyplot as plt


#Define ODE for problem given
f = lambda t,y: 2 * (325/850) * (math.sin(t)**2) - (200*(1+y)**(3/2))/850


def plot_tank(h):
    '''Driver function to solve then plot solution given ODE from t=0 to t=10 and a given step size
    (note if certain non typical step sizes are given it is possible that the y value at a step may
    and as a result may get complex values for subsequent steps)'''
    w = eulers(f, 2, 0, 10, h)
    t = w[0]
    y = w[1]
    plt.plot(t, y)
    plt.xlabel("Time(s)")
    plt.ylabel("Water Level(m)")
    plt.suptitle("Water Level versus Time")
    plt.show()

def eulers(func, yInitial, tInitial, tFinal, h):
    '''General Euler's method implementation that takes a given first order ODE(dy/dt) func depende
    solves given an initial value (yInitial, tInitial) from tInitial to tFinal with step size h.
    Returns 2d NumPy array with [0] index being t values and [1] index being corresponding y soluti
    y = yInitial
    t = tInitial
    yAns = [yInitial]
    tAns = [tInitial]
    while(t < tFinal): #Will end once we are on tFinal or past it
        y = y + f(t,y) * h
        t += h
        yAns.append(y)
        tAns.append(t)
    return np.array([tAns, yAns])

plot_tank(.5)
```

ME EN 2450
Ryan Dalby u0848407
HW1a

c)



d)

The numerical approximation I got seems accurate based on the situation. Looking at my calculations by hand I clearly see that between y(0) and y(1.5) y is decreasing throughout the whole interval. Looking at the differential equation as well we notice that as y becomes small we get a small second term and the first term that has sin^2(t) in it dominates. This is reflected in the solution as we get oscillating increasing/decreasing behavior just as a sinusoidal derivative would behave.