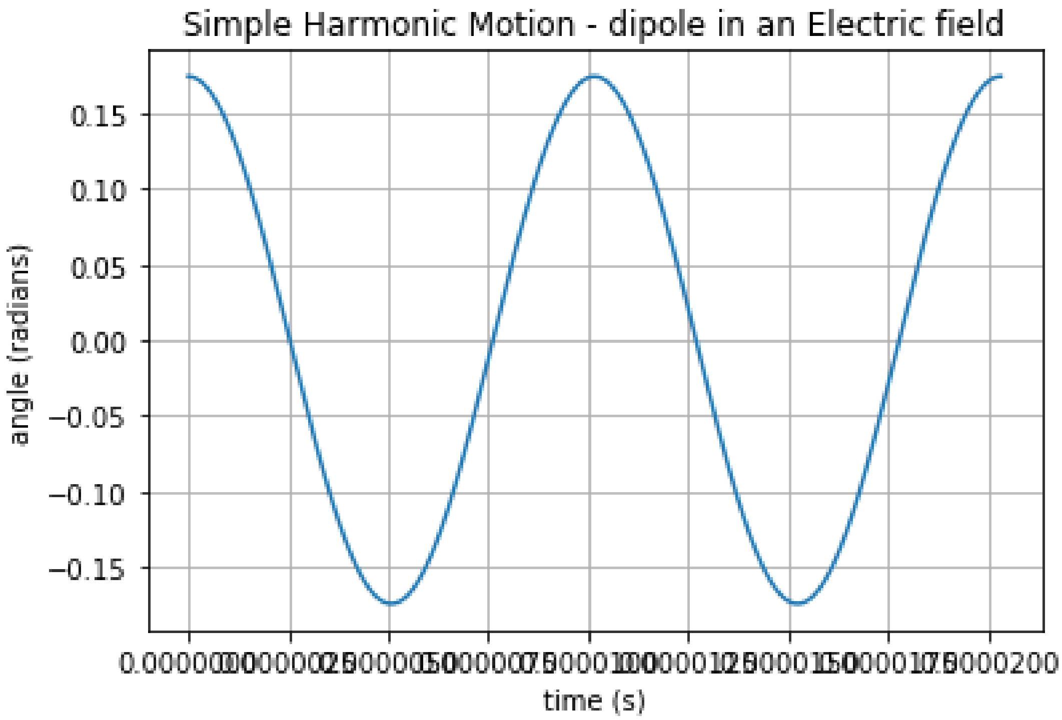
assignment2

January 23, 2018

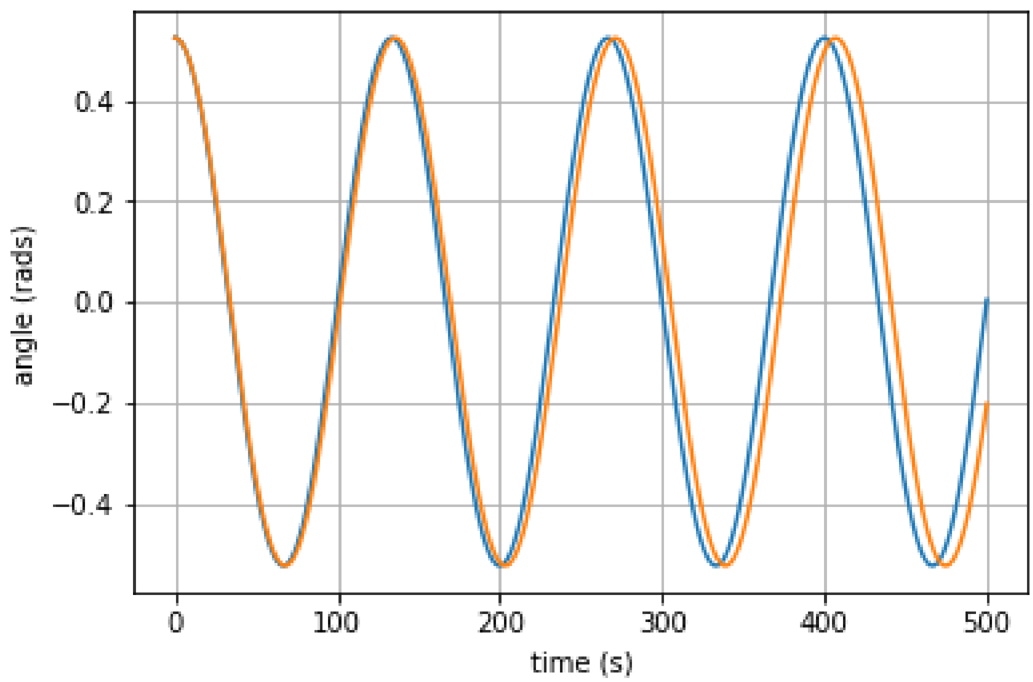
In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
import math as math

theta0=(10.0/180)*np.pi  #initial displacement in radians

q = 1.6*10**(-19)            #Charge on dipole in Coulombs
d = 5*10**(-2)            #distance separating the charges (m)
m = 1.67*10**(-27)            #mass of a single charge (kg)
E = 100              #magnitude of the electric field


############################################################
########        EDIT BELOW WITH YOUR VALUE       ###########
########       THIS IS THE ANGULAR FREQUENCY     ###########

omega = math.sqrt((2*q*E)/(m*d)) #enter your angular frequency


########     Determine the Period of the motion     ########
########      in terms of the angular frequency     ########

T = 2*math.pi/omega  #edit this for the period of motion #

t= np.linspace(0,2*T,2000) #time interval from t = 0 to t = 2T
theta = theta0*np.cos(omega*t) #solution for Simple harmonic motion

############################################################
########   The following code plots the data         #######

plt.plot(t,theta)
plt.title("Simple Harmonic Motion - dipole in an Electric field")
plt.xlabel("time (s)")
plt.ylabel("angle (radians)")
plt.grid()
plt.show()
############################################################
```

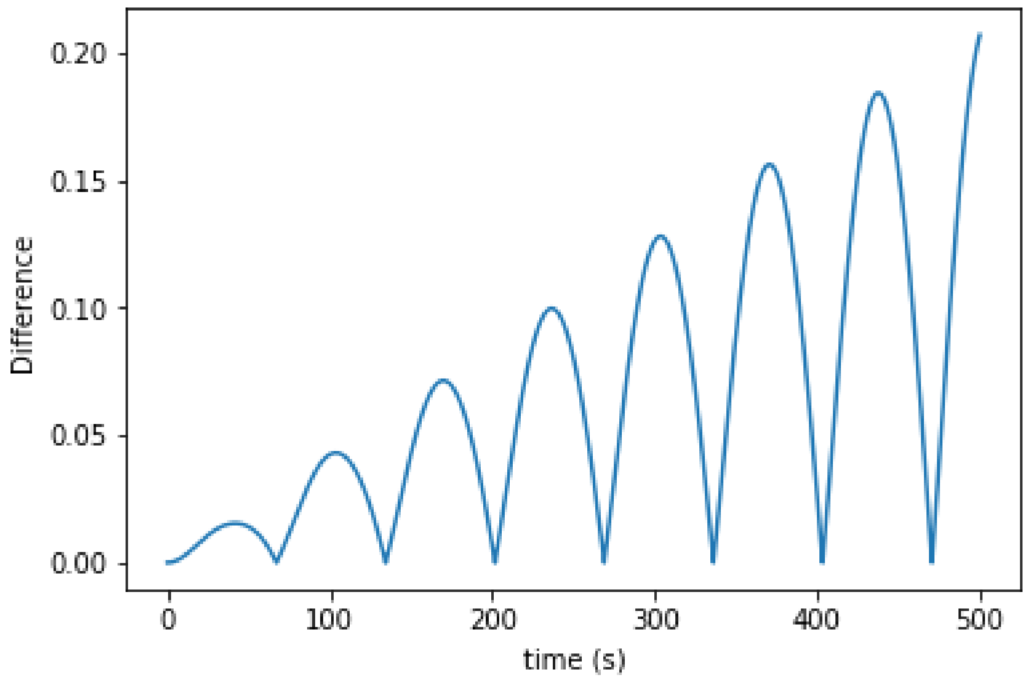



```
In [5]: import matplotlib.pyplot as plt
        import numpy as np
        import math as math
        from scipy.integrate import odeint

        ############################################################
        ################ Start of parameters  ###################
        theta_degrees = 30.0  #intial angle of displacement (degres)
        Tmax = 500.0           #upper bound for time (s)
        q = 1.0e-19              #Charge on dipole in Coulombs
        d = 3.0e-9             #distance separating the charges (m)
        m = 3.0e-5              #mass of a single charge (kg)
        E = 1000             #magnitude of the electric field

        N = 50000              #number of time intervals
        error = 0.01           #error should have been 0.01 not 0.05

        def torque(y, t, q, d, m, E):
            theta, omega = y
            return [omega, -2*q*E*np.sin(theta)/(m*d)]

        ############################################################
        ########        EDIT BELOW WITH YOUR VALUE       ###########
        ########       THIS IS THE ANGULAR FREQUENCY     ###########
```

```python
omega = np.sqrt((2.0*q*E/(d*m))) ## EDIT THIS ##

############################################################

################  End of parameters      ##################
############################################################

############################################################
############################################################

#Set the times to evaluate
t = np.linspace(0,Tmax,N)

#convert initial angle to radians
theta_rads = theta_degrees * np.pi / 180.0


##############################
# compute approximate theta values on the time intervals
Approx_Solution = theta_rads*np.cos(omega*np.array(t))

##############################
# Set the intial condition [initial angle, initial velocity]
x0 = [theta_rads, 0.0]

##############################
# numerically solve the real solution using odeint
sol2 = odeint(torque,x0,t,args=(q,d,m,E))

#############################
# Plot the two solutions
plt.plot(t,theta_rads*np.cos(omega*np.array(t)))
plt.plot(t,sol2[:,0])
plt.grid()
plt.xlabel("time (s)")
plt.ylabel("angle (rads)")
plt.show()


############################
# compute the difference between approx and actual on intervals
diff = [np.abs((a1-b1)) for a1,b1 in zip(Approx_Solution,sol2[:,0])]

#########################################
# Plot the difference vs time
plt.plot(t,diff)
plt.xlabel("time (s)")
plt.ylabel("Difference")
```

```
plt.show()

##########################################
#  Find when the two differ by .01 radians
flag = True
for i in range(len(t)-1):
    if abs(diff[i]> error):
        print("after t =", t[i],"the two solutions differ by more than .01 rads")
        flag = False
        break
if flag:
    print("solution always within",error," rads")
```






```
('after t =', 23.61047220944419, 'the two solutions differ by more than .01 rads')
```