HW Coupled Equations

Riley Payung

Jason Kinser

CDS 411

November 19, 2020

In [1]:
```
import numpy as np
import scipy.integrate as si
import matplotlib.pyplot as plt
%matplotlib inline
```

Problem 1

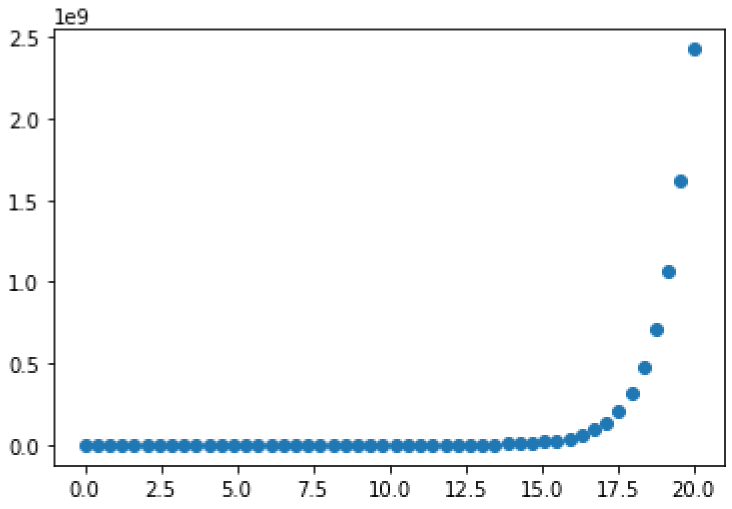
A model of exponential population growth described by the equation,

$$\frac{\mathrm{d}p}{\mathrm{d}t} = kp$$

Create a model function for this equation. Create time sampling points from 0 to 20. Run **odeint** using the model function. Plot the result.

In [2]:
```
# your function, code and plot

def model(p,t):
    k = 1;
    dpdt = k*p;
    return dpdt;
ts = np.linspace(0,20);
inits = [5];
ps = si.odeint(model,inits,ts);
plt.scatter(ts,ps)
plt.show()
```



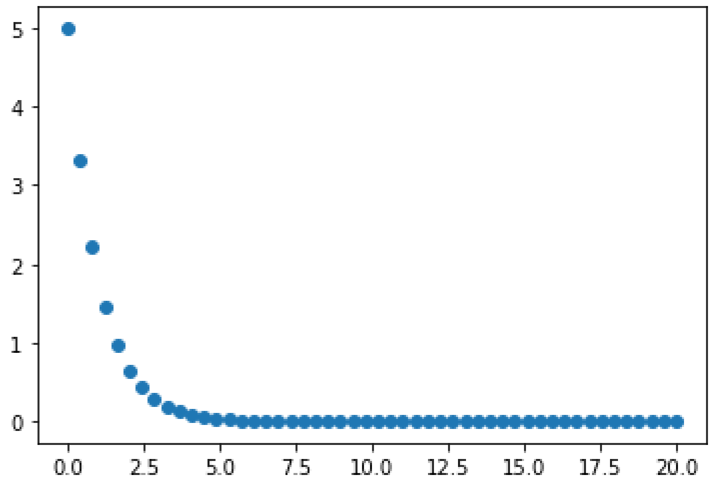
Problem 2

A model of exponential decay is described by the equation,

$$\frac{\mathrm{d}p}{\mathrm{d}t} = -kp$$

Create a model function for this equation. Create time sampling points from 0 to 20. Run **odeint** using the model function. Plot the result.

In [3]:
```
# your function, code and plot
def model(p,t):
    k = 1;
    dpdt = -k*p;
    return dpdt;
ts = np.linspace(0,20);
ps = si.odeint(model,[5],ts);
plt.scatter(ts,ps)
plt.show()
```



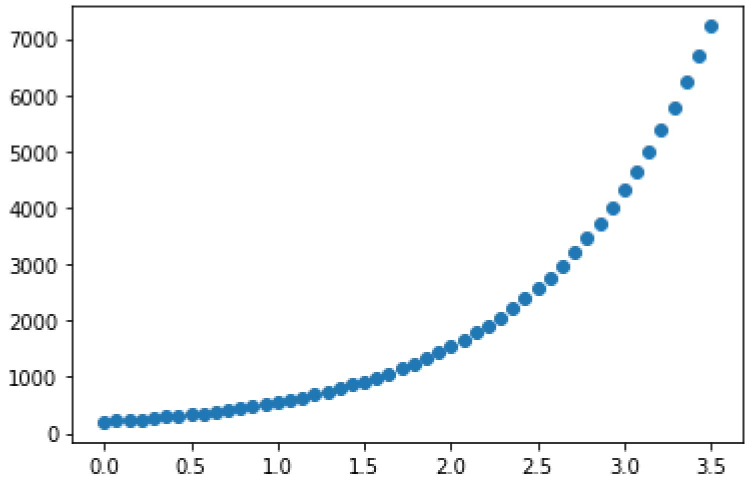
Problem 3

A change of velocity was described in the earlier chapters as $v_2 = v_1 + at$. A differential equation would describe it as,

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -at + v_0$$

Use the initial velocity of $v_0 = 0$, $a = -9.8$, initial height of 200, and time points go from 0 to 3.5.

In [4]:
```
ts = np.linspace(0,3.5);
def model3(init,t):
    v,y = init
    a = -9.8
    dydt = -a*t + v
    y += dydt + 0.5*a*t**2
    return (dydt,y)
ys = si.odeint(model3,[0,200],ts)
plt.scatter(ts,ys[:,1]);
plt.show()
```



Problem 4

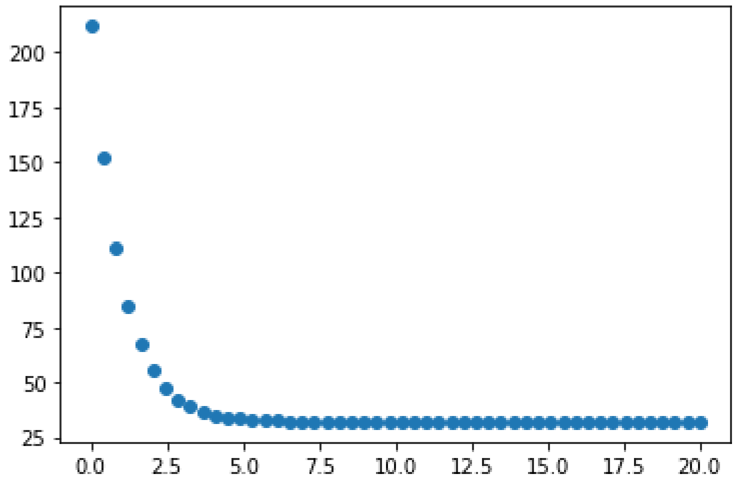
A heated object will cool by the equation

$$\frac{\mathrm{d}T}{\mathrm{d}t} = -k(T - T_c)$$

where T is object temperature, k is a constant, and T_c is the environment temperature (not the object temperature).

Use $k = 1$. The initial object temperature is 212 and the environment temperature is 32. Model this system and include the chart for time 0 to 20.

In [5]:
```
# your function, code, and plot
def model4(T,t):
    k = 1;
    Tc = 32;
    dTdt = -k*(T - Tc);
    return dTdt;
ts = np.linspace(0,20);
Ts = si.odeint(model4,[212],ts)
plt.scatter(ts,Ts);
plt.show()
```



Problem 5

Growth of bacteria initially increases, but then saturation occurs. It is modeled by,

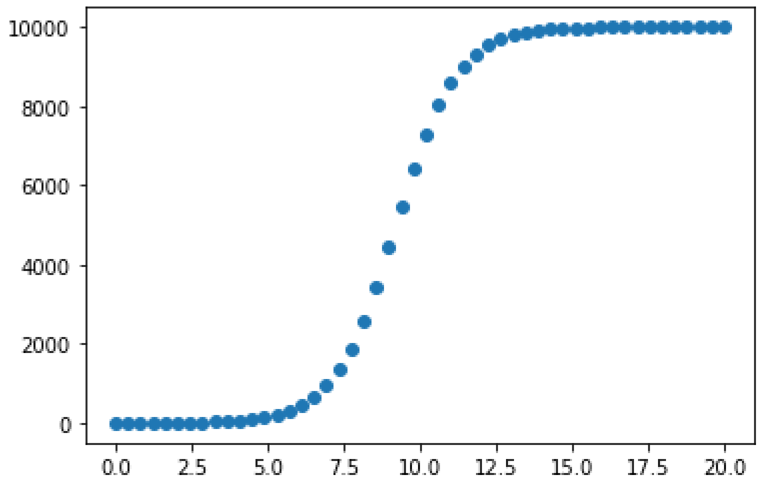
$$\frac{\mathtt{d}P}{\mathtt{d}t} = kP\left(1 - \frac{P}{P_{max}}\right)$$

where P is the population count, $k = 1$, and P_{max} is the maximum population allowed by the environment.

Model this function for time 0 to 20 with an initial population of 1 and a max population is 10,000.

Model this system and show a plot of your result.

In [6]:
```
# function, code and plot
def model5(p,t):
    k = 1;
    p_max = 10000;
    dpdt = k*p*(1-(p/p_max));
    return dpdt;
ts = np.linspace(0,20);
ys = si.odeint(model5,[1],ts)
plt.scatter(ts,ys);
plt.show()
```



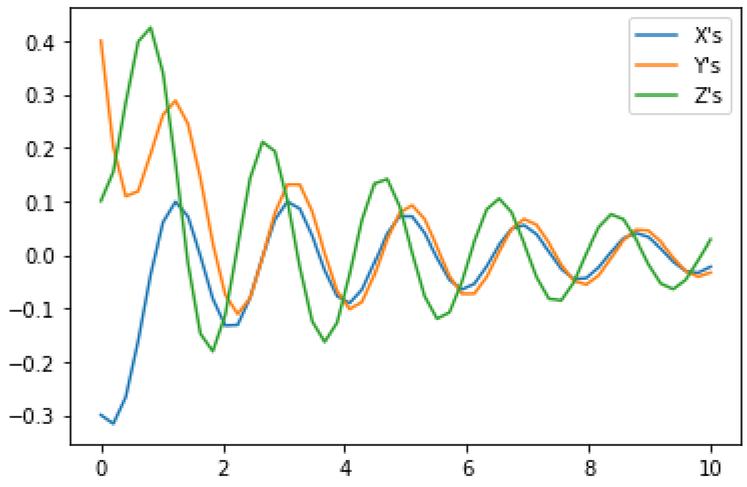
Problem 6

Model the following system,

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -y + 1.8z$$
$$\frac{\mathrm{d}y}{\mathrm{d}t} = 2x - 2y + 2z$$
$$\frac{\mathrm{d}z}{\mathrm{d}t} = -3x - 2.5y + 0.5z$$

Use the initial conditions $x_0 = -0.3$, $y_0 = 0.4$, and $z_0 = 0.1$. The time range is 0 to 10. Model this function and plot all three variables on a single chart.

In [7]:
```
# your code and plot
def model6(init,t):
    x0,y0,z0 = init;
    dxdt = -y0 + 1.8*z0
    dydt = 2*x0 - 2*y0 + 2*z0
    dzdt = -3*x0 - 2.5*y0 + 0.5*z0
    return (dxdt,dydt,dzdt);
init = [-0.3,0.4,0.1];
ts = np.linspace(0,10);
values = si.odeint(model6,init,ts)
xs = values[:,0]
ys = values[:,1]
zs = values[:,2]
plt.plot(ts,xs,label='X\'s');
plt.plot(ts,ys,label='Y\'s');
plt.plot(ts,zs,label='Z\'s');
plt.legend()
plt.show()
```



Problem 7

The famous skateboard problem (Taylor's classical mechanics) describes the angular motion of the skateboard as

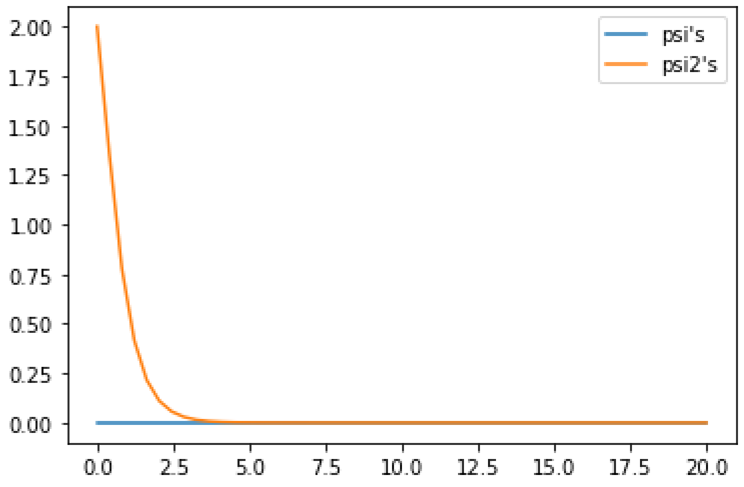
$$\frac{\mathrm{d}^2(\phi)}{\mathrm{d}t^2} = -\frac{g}{R}\sin(\phi)$$

$$\omega = \frac{\mathrm{d}(\phi)}{\mathrm{d}t}$$

Use $g = 9.8$ and $R = 6$. Initial conditions are $\phi_0 = 2$ and $\omega_0 = 0$, and the time span is 0 to 20.

Model this function and plot both $\frac{\mathrm{d}(\phi)}{\mathrm{d}t}$ and ϕ on the same graph.

In [9]:
```
# your code and plot
phi0 = 2;
omega0 = 0;
init = [omega0,phi0];
ts = np.linspace(0,20);
def model7(inits,t):
    omega,phi = inits;
    g = 9.8
    R = 6;
    dphdt = omega;
    d2phd2t = (-g/R)*np.sin(phi);
    return (dphdt,d2phd2t);
vals = si.odeint(model7,init,ts)
phis = vals[:,0];
phis2 = vals[:,1];
plt.plot(ts,phis,label='psi\'s')
plt.plot(ts,phis2,label='psi2\'s')
plt.legend()
plt.show()
```



Problem 8

Consider a case of two pendulums which both swing left and right. There is a spring that connects the two masses together. This is called a *connected pendulum*. (Picture of this on https://en.wikipedia.org/wiki/Pendulum_(mathematics (https://en.wikipedia.org/wiki/Pendulum_(mathematics)). Scroll down to Coupled Pendula.) This system is described by

$$m\frac{\mathrm{d}^2x_1}{\mathrm{d}t^2} = -\frac{mg}{L}x_1 + k(x_2 - x_1)$$
$$m\frac{\mathrm{d}^2x_2}{\mathrm{d}t^2} = -\frac{mg}{L}x_2 + k(x_1 - x_2)$$

The variables are m is mass, x_1 and x_2 are the positions of the masses on the end of the pendulum, k is a strength constant, L is the length of each pendulum. Use 100 time steps from 0 to 30. Uses $m = 1$ and $k = 1$.

The initial conditions are $x_1 = 0$, $x_2 = 1$, $\frac{\mathrm{d}x_1}{\mathrm{d}t} = 0$ and $\frac{\mathrm{d}x_2}{\mathrm{d}t} = 0$.

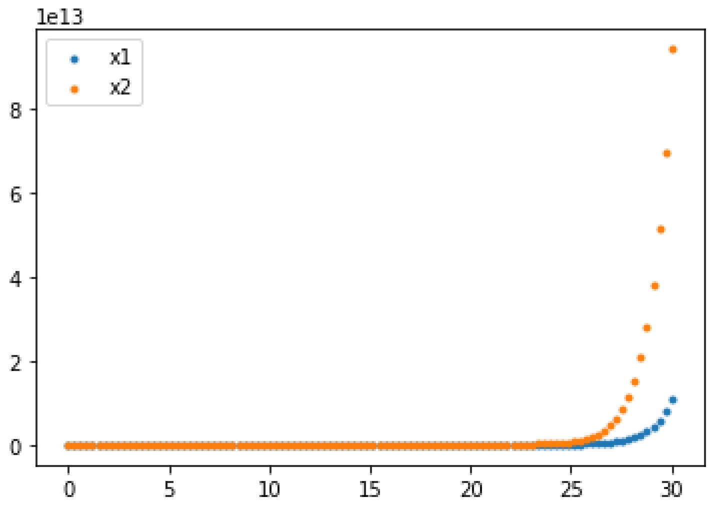
Model this system and plot x_1 and x_2 on the same graph.

Hint: If both pendulums start at the same position, then the spring between the masses will never stretch or expand. Thus, you can test your code by starting both masses at the same position and see if their motions are identical.

In [24]:
```
inits = [0,1,0,0];
g = -9.8
def model8(init,t):
    x1,x2,dx1dt,dx2dt = init;
    k = 1;
    m = 1;
    L=1;
    md2x1dt2 = -m*g*x1/L + k*(x2-x1);
    md2x2dt2 = -m*g*x2/L + k*(x1-x2);


    return (x1,x2,md2x1dt2,md2x2dt2)
ts = np.linspace(0,30,100);
xvals = si.odeint(model8,inits,ts);
```

In [25]:
```
plt.scatter(ts,xvals[:,2],marker='.',label='x1')
plt.scatter(ts,xvals[:,3],marker='.',label='x2')
plt.legend()
plt.show()
```



In [ ]:
```
```