Tutorial 1-Answers

August 20, 2017

1 Question 1:

Use the mole balance to derive an equation analogous to the CSTR mole balance derived previously for a fluidized CSTR containing catalyst particles in terms of the catalyst weight, W, and other appropriate terms.

2 Question 2:

We are going to consider the cell as a reactor. The nutrient corn steep liquor enters the cell of the microorganism Penicillium chrysogenum and is decomposed to form such products as amino acids, RNA, and DNA. Write an unsteady mass balance on (a) the corn steep liquor, (b) RNA, and (c) penicillin. Assume the cell is well mixed and that RNA remains inside the cell.

3 Question 3:

What is wrong with this solution?

The irreversible liquid phase second order reaction:

$$2A \xrightarrow{k_1} B \;\; k_1 = 0.03\, dm^3/(mol.s)$$

is carried out in a CSTR. The entering concentration of A, C_{A0}, is 2 molar. and the exit concentration of A, C_A is 0.1 molar. The volumetric flow rate, $_o$, is constant at 3 dm3/s. What is the corresponding reactor volume?

Solution:

1. Mole balance:
$$V = \frac{F_{A0} - F_A}{-r_A}$$

2. Rate law (second order)
$$-r_A = kC_A^2$$

3. Combine
$$V = \frac{F_{A0} - F_A}{kC_A^2}$$

4.
$$F_{A0} = {}_oC_{A0} = 3\frac{dm^3}{s}.\frac{2\,mol A}{dm^3} = \frac{6mol\ A}{dm^3}$$

5.

$$F_A = {}_oC_A = 3\frac{dm^3}{s}.\frac{0.1\,mol\,A}{dm^3} = \frac{0.3mol\ A}{dm^3}$$

6.

$$V = \frac{(6-0.3)\frac{mol}{s}}{\left(0.03\frac{dm^3}{mol.s}\right)\left(2\frac{mol}{dm^3}\right)} = 47.5dm^3$$

4 Question 4:

This problem should be solved using Python. You can create a temporary notebook by accessing tmpnb.org from your cellphone, tablet or laptop. You can then programme the equations from scratch. What could work is to save a template code on your phone for differential equations and algebraic equation solutions.

There are initially 500 rabbits (x) and 200 foxes (y) on Farmer Oat's property. Use Polymath or MATLAB to plot the concentration of foxes and rabbits as a function of time for a period of up to 500 days. The predator–prey relationships are given by the following set of coupled ordinary differential equations:

$$\frac{dx}{dt} = k_1x - k_2xy$$

$$\frac{dy}{dt} = k_3xy - k_4y$$

Constant for growth of rabbits $k_1 = 0.02\,day^1$
Constant for death of rabbits $k_2 = 0.00004/(dayno.of\ foxes)$
Constant for growth of foxes after eating rabbits $k_3 = 0.0004/(dayno.of\ rabbits)$
Constant for death of foxes $k_4 = 0.04\,day^1$

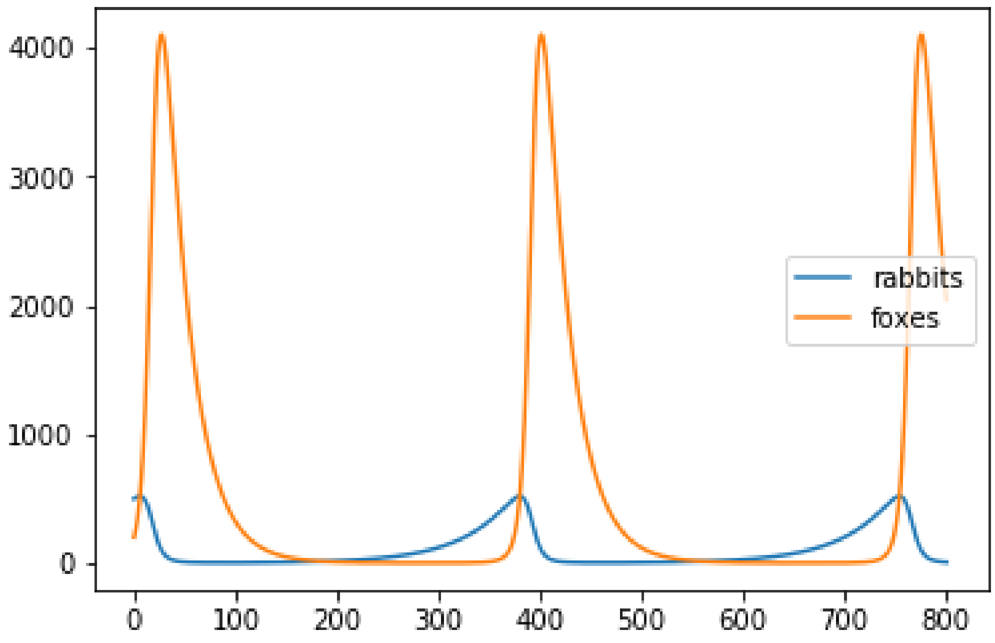
a) What do your results look like for the case of $k_3 = 0.00004/(dayno.of\ rabbits)$ and $t_{final} = 800\,days$? Also plot the number of foxes versus the number of rabbits. Explain why the curves look the way they do.

```
In [3]: from scipy.integrate import odeint
        from matplotlib import pyplot as plt
        from numpy import linspace
        % matplotlib inline

In [5]: xo = 500
        yo = 200
        varo = [xo, yo]
        tspan = linspace(0, 800, 10000)

In [6]: k1 = 0.02 # per day
        k2 = 0.00004 # per day per foxes
        k3 = 0.0004 # per day per rabbits
        k4 = 0.04 # per day
```

```
In [7]: def diffs(var, t):
            x = var[0]
            y = var[1]

            dxdt = k1*x-k2*x*y
            dydt = k3*x*y-k4*y

            return [dxdt, dydt]
```

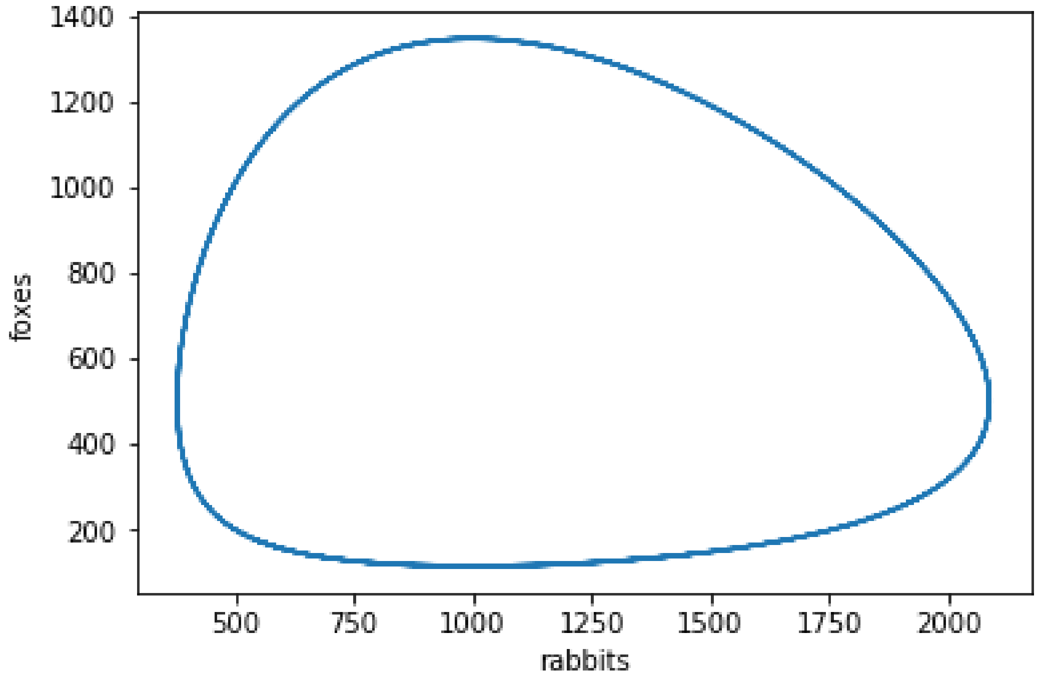
```
In [8]: var = odeint(diffs, varo, tspan)
        plt.plot(tspan, var)
        plt.legend(['rabbits', 'foxes'])
```

```
Out[8]: <matplotlib.legend.Legend at 0x1fe8b0521d0>
```

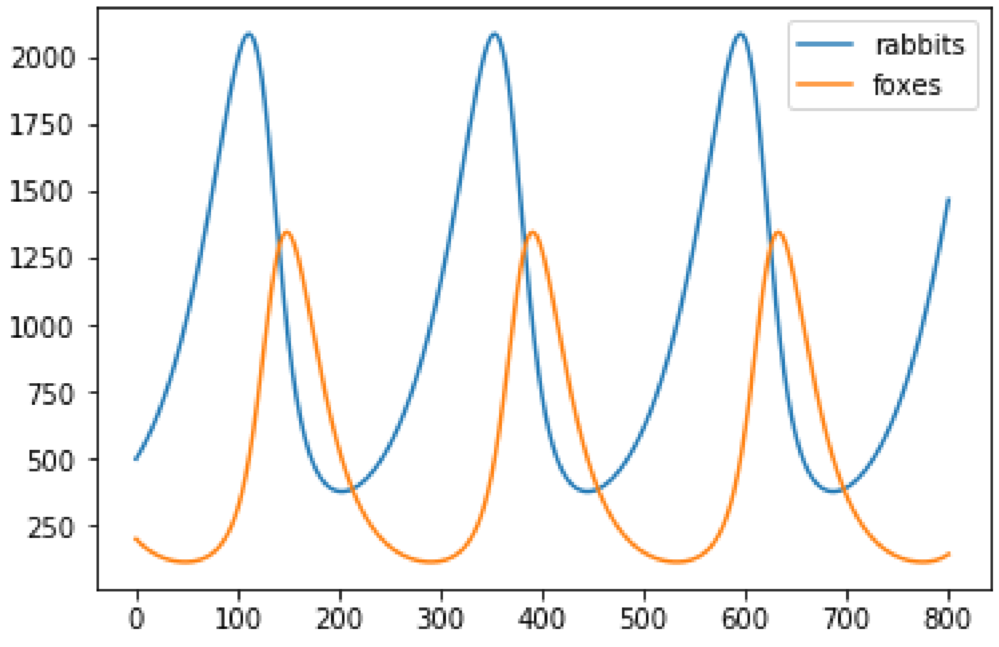


```
In [12]: plt.plot(var[:, 0], var[:, 1])
         plt.xlabel('rabbits')
         plt.ylabel('foxes')
```

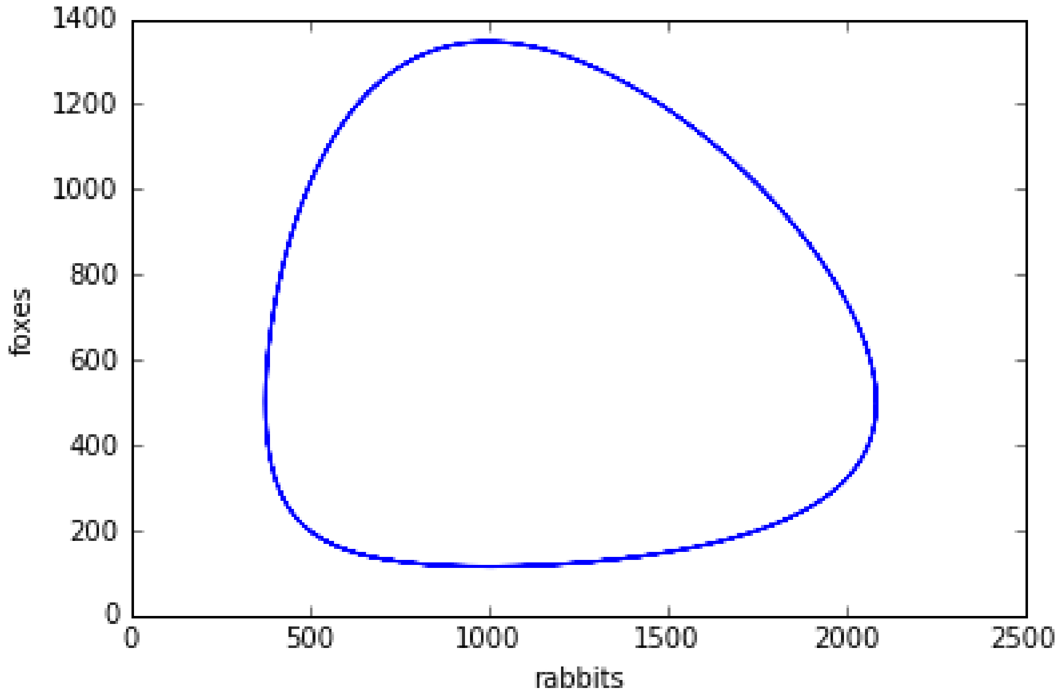
```
Out[12]: <matplotlib.text.Text at 0x1fe8e795080>
```



```
In [13]: k3 = 0.00004
         var = odeint(diffs, varo, tspan)
         plt.plot(tspan, var)
         plt.legend(['rabbits', 'foxes'])

Out[13]: <matplotlib.legend.Legend at 0x1fe8e8f6518>
```



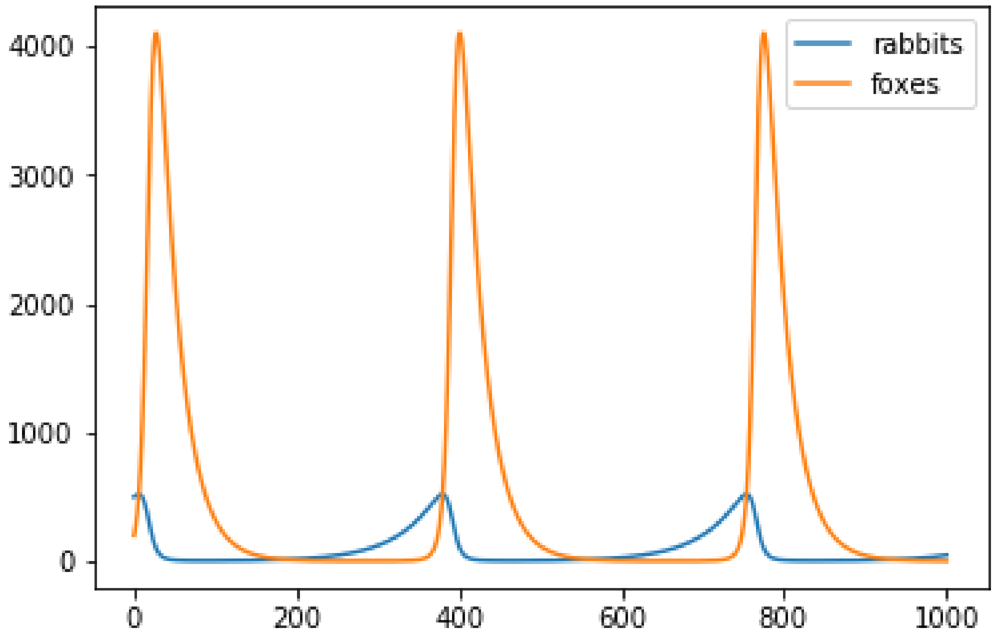
```
In [8]: plt.plot(var[:, 0], var[:, 1])
        plt.xlabel('rabbits')
        plt.ylabel('foxes')

Out[8]: <matplotlib.text.Text at 0xc813fd5e80>
```

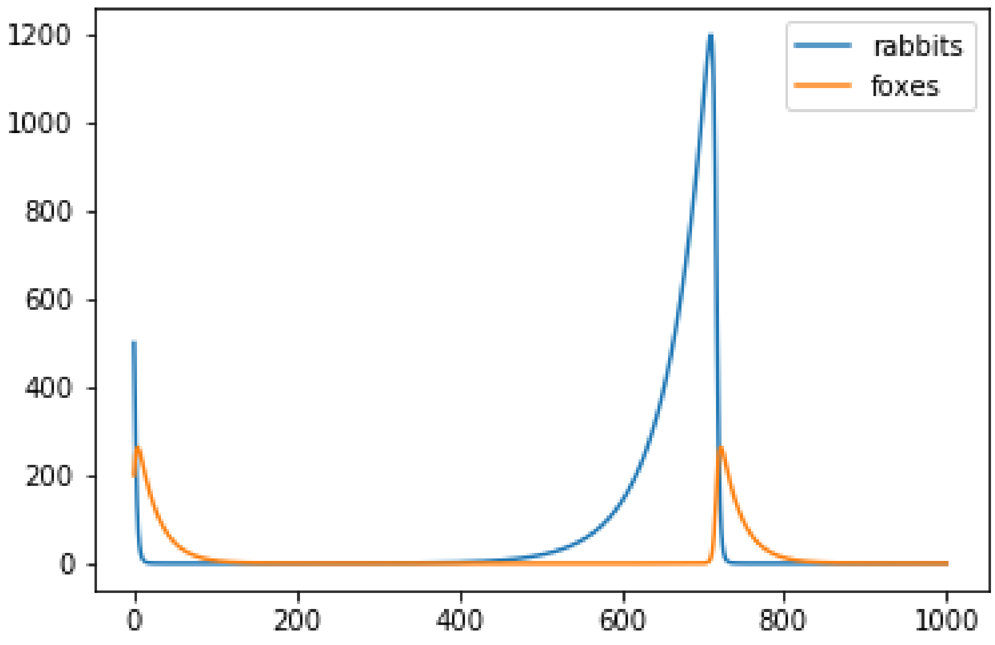


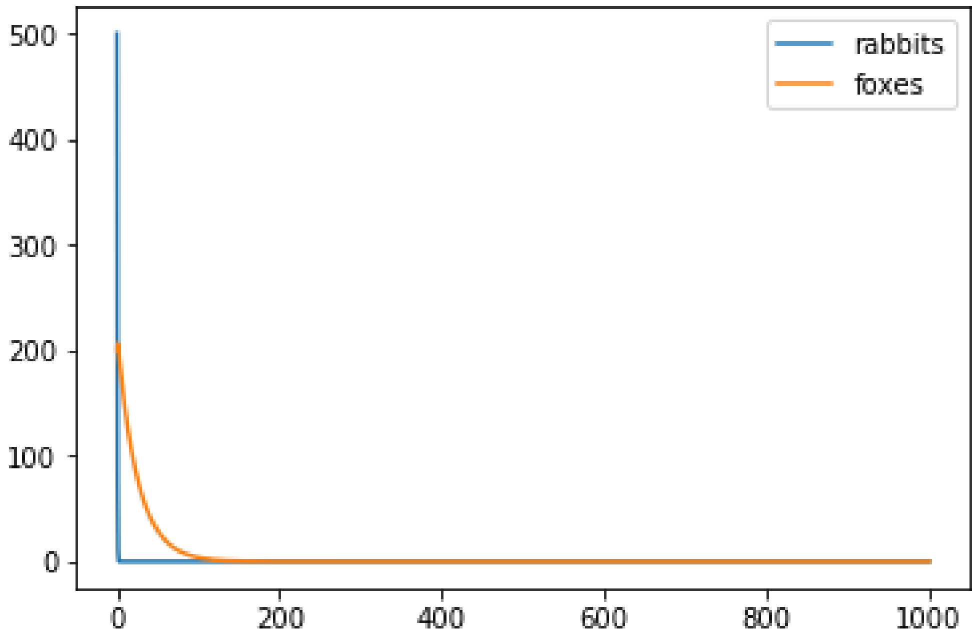
```
In [17]: k1 = 0.02
         k2 = 0.00004
         k3 = 0.0004
         k4 = 0.04
         tspan = linspace(0, 1000, 100000)

         def kchange(k1=k1, k2=k2, k3=k3, k4=k4):

             def diffs(var, t):
                 x = var[0]
                 y = var[1]

                 dxdt = k1*x-k2*x*y
                 dydt = k3*x*y-k4*y
```

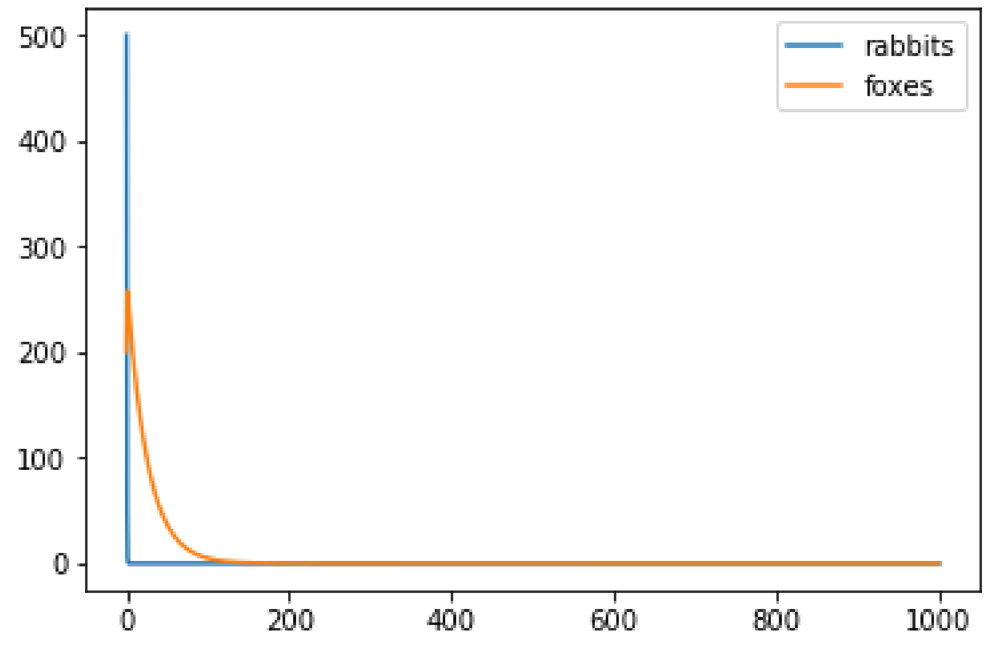
```
        return [dxdt, dydt]

    var = odeint(diffs, varo, tspan)
    plt.plot(tspan, var)
    plt.legend(['rabbits', 'foxes'])

from ipywidgets import interact

interact(kchange, k1 = (0, 0.1, 0.01), k2 = (0, 0.1, 0.00001), k3 = (0, 0.1, 0.00001)
```

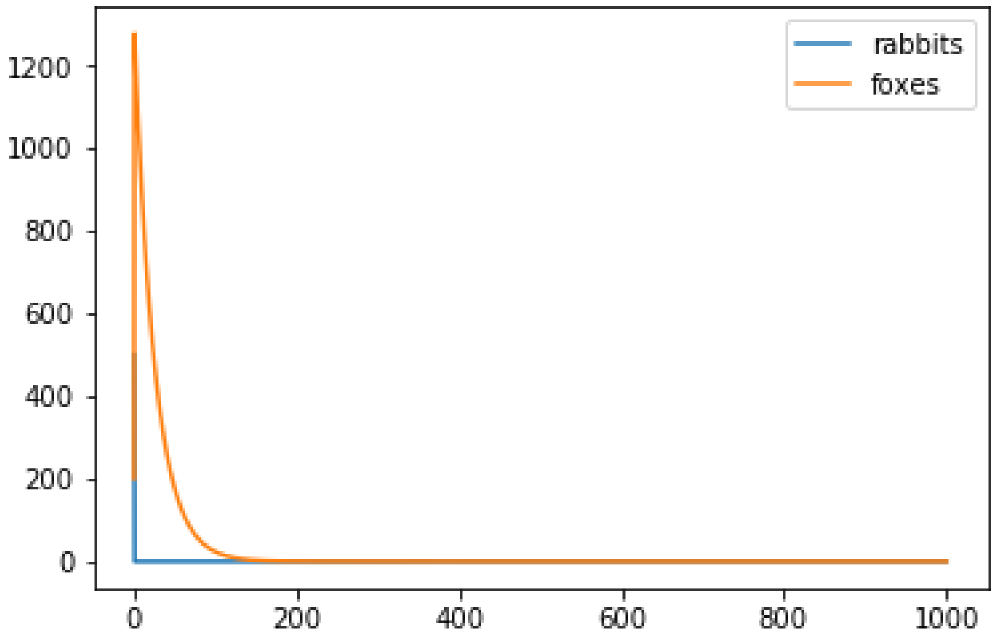
```
Out[17]: <function __main__.kchange>
```

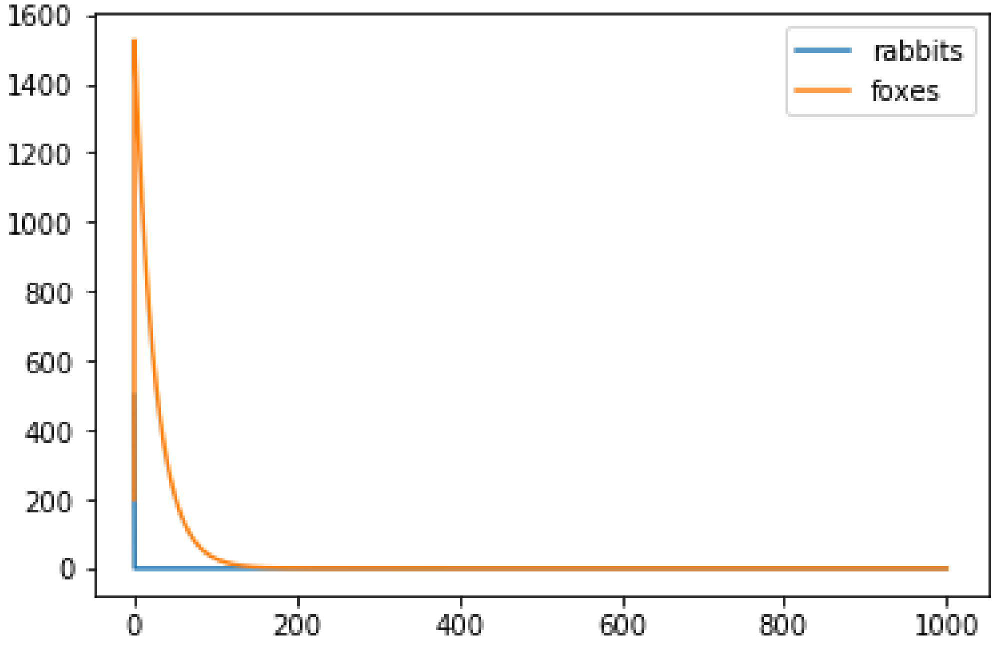



rabbits
foxes
1200
1000
800
600
400
200
0
0
200
400
600
800
1000

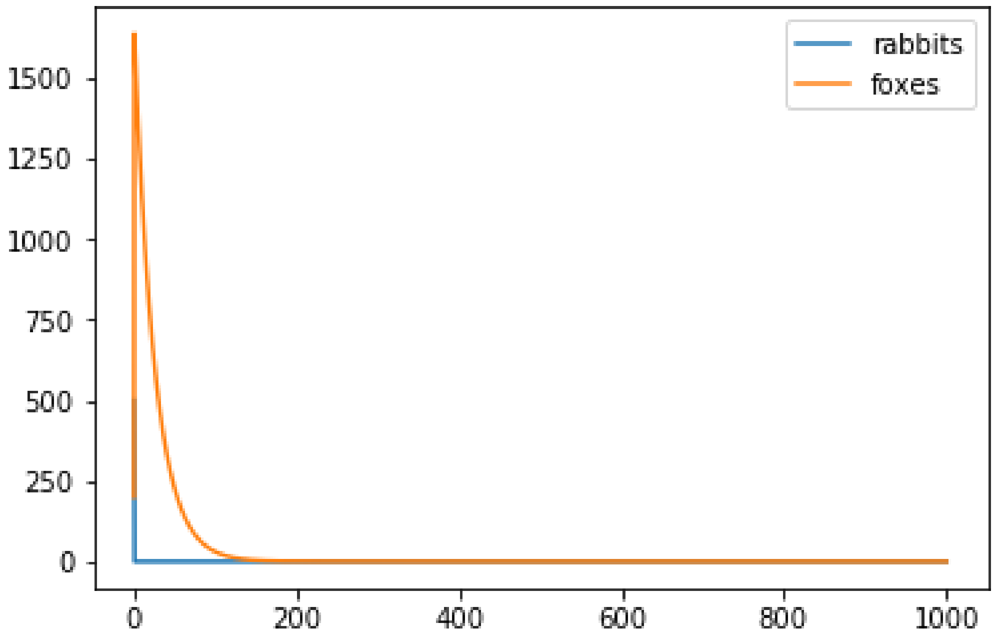

rabbits
foxes
500
400
300
200
100
0
0
200
400
600
800
1000


rabbits
foxes
500
400
300
200
100
0
0
200
400
600
800
1000


rabbits
foxes
1200
1000
800
600
400
200
0
0
200
400
600
800
1000



1600
1400
1200
1000
800
600
400
200
0
0
200
400
600
800
1000
rabbits
foxes



1500
1250
1000
750
500
250
0
0
200
400
600
800
1000
rabbits
foxes

5 Question 5

Let's solve a set of non-linear differential equations:

$$x^3y - 4y^2 + 3x = 1$$

$$6y^2 - 9xy = 5$$

initial guesses of x = 2, y = 2.

```
In [10]: from scipy.optimize import fsolve

In [11]: def func(var):
             x = var[0]
             y = var[1]
             func1 = x**3*y-4*y**2+3*x-1
             func2 = 6*y**2-9*x*y-5
             return [func1, func2]

         var = fsolve(func, [2, 2])

         print(var, func(var))

[ 2.38503866  3.79702789] [-4.2454928461665986e-13, 1.1368683772161603e-13]
```

6 Question 6

a) A CSTR is known to follow the following kinetics:

$$-r_A = \frac{k_1 C_A^{2.375}}{k_2 + C_A^{1.725}}$$

$$k_1 = 2.542\,(units - L, hours, moles)$$

$$k_2 = 1.375\,(units - L, hours, moles)$$

Determine the outlet concentration of a liquid phase CSTR with inlet concentration of A = 1.32 mol/L:, a total reactor volume of 500 L and a residence time in the reactor of 3 hours.

b) Determine how long the system in a) takes to reach steady state if it is assumed that the reactor fills instantly at time 0 and has the inlet concentration initially

```
In [14]: from scipy.optimize import fsolve

         def rA(FA):
             CA = FA/Q
             rA = -k1*CA**2.375/(k2+CA**1.725)
             return rA
```

```
        def func(FA):
            return FAo-FA+rA(FA)*V

        k1 = 2.542
        k2 = 1.375
        tau = 3 #hours
        V = 500 #L
        Q = V/tau #L/h
        CAo = 1.32 #mol/L
        FAo = CAo*Q

        FA = fsolve(func, FAo/10)
        [FA/Q, func(FA)]
```

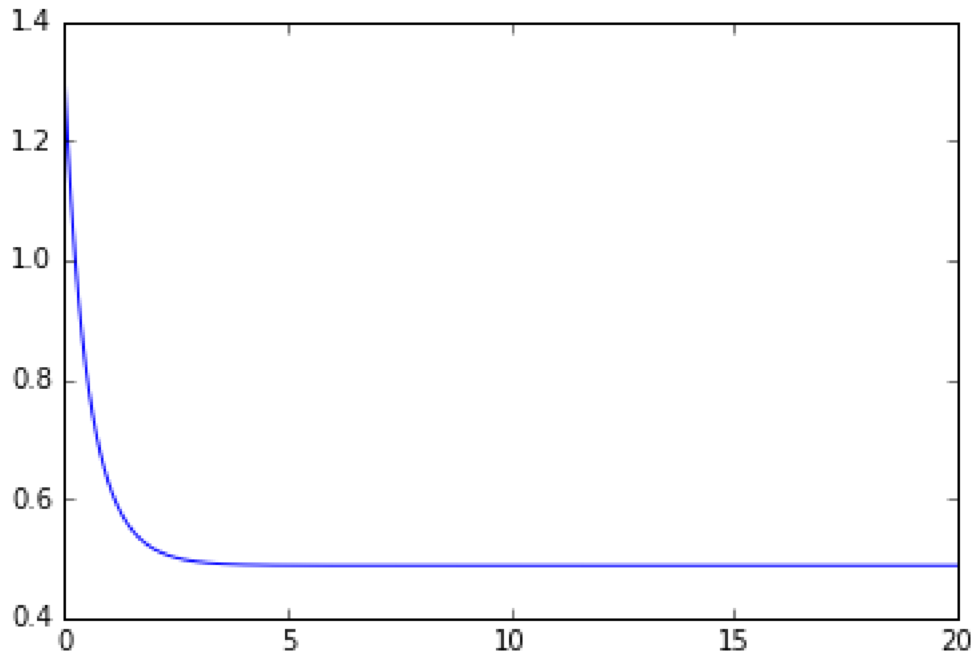
```
Out[14]: [array([ 0.48769]), array([ 0.])]
```

$$F_{Ao} - F_A + r_A V = \frac{dN_A}{dt}$$

```
In [15]: from scipy.integrate import odeint
         from numpy import linspace
         from matplotlib import pyplot as plt
         %matplotlib inline

         def CSTRtrans(NA, t):
             FA = NA/V*Q #remember C = C
             dNAdt = FAo-FA+rA(FA)*V
             return dNAdt
         tspan = linspace(0, 20, 1000)
         NAo = CAo*V
         print(NAo)
         NA = odeint(CSTRtrans, NAo, tspan)
         CA = NA/V
         FA = CA*Q
         plt.plot(tspan, FA/Q)
         FA[-1]/Q
```

```
660.0
```

```
Out[15]: array([ 0.48769])
```



```
In [ ]:
```