Chemical reaction network

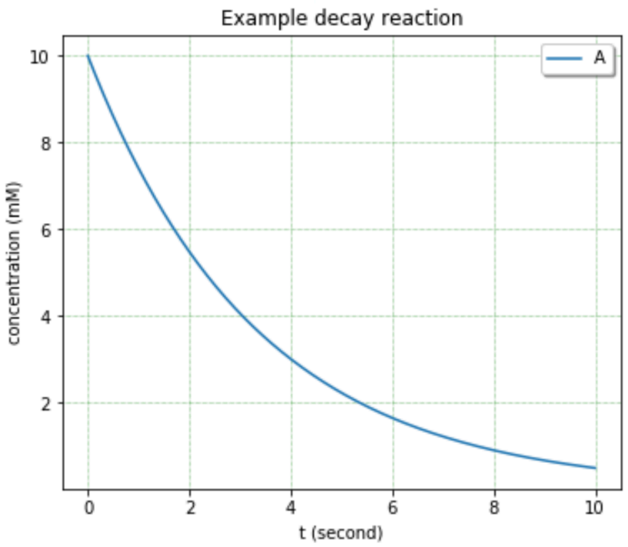
Decay reaction

$$A \xrightarrow{k}$$

$$\frac{d}{dt}a(t) = -ka(t)$$

$$a(t) = a_0 e^{-kt}$$

```
import numpy as np
from scipy.integrate import solve_ivp

def decay(t,a):
    k = 0.3
    dadt = -k*a
    return dadt

y_init = [10]
sim_time = np.linspace(0, 10, 1000)

result = solve_ivp(decay,
                   (sim_time[0], sim_time[-1]),
                   y_init,
                   t_eval=sim_time)
```



Production & Decay

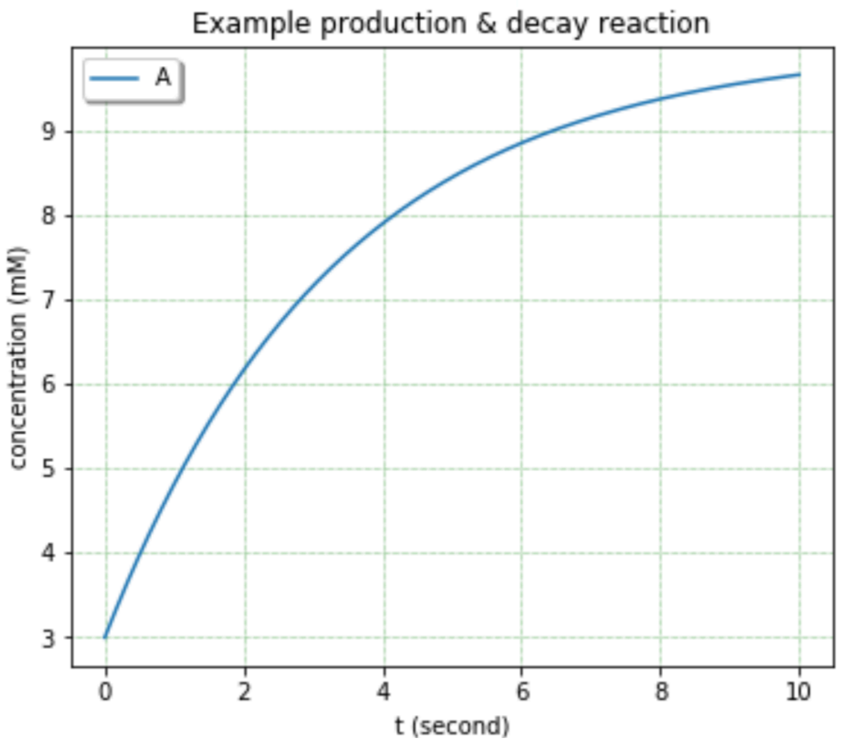
$$\xrightarrow{k_0} A \xrightarrow{k_1}$$

$$\frac{d}{dt}a(t) = k_0 - k_1 a(t)$$

$$\frac{d}{dt}a(t) = 0$$

$$a(\infty) = k_0/k_1$$

```python
def decay(t,a):
    k0 = 3
    k1 = 0.3
    dadt = k0-k1*a
    return dadt

y_init = [3]
sim_time = np.linspace(0, 10, 1000)

result = solve_ivp(decay,
                   (sim_time[0], sim_time[-1]),
                   y_init,
                   t_eval=sim_time)
```



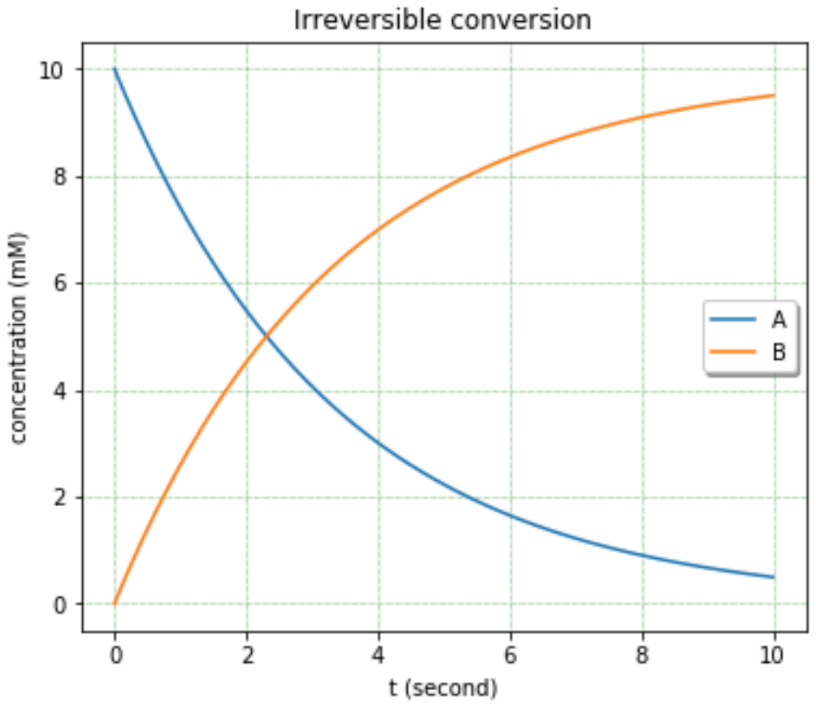
Irreversible Conversion

$$A \xrightarrow{k} B$$

$$\frac{d}{dt}a(t) = -ka(t)$$

$$\frac{d}{dt}b(t) = ka(t)$$

```
def decay(t, y):
    k = 0.3
    a, b = y
    dadt = -k*a
    dbdt = k*a
    return np.array([dadt, dbdt])

y_init = [10,0]
sim_time = np.linspace(0, 10, 1000)

result = solve_ivp(decay,
                   (sim_time[0], sim_time[-1]),
                   y_init,
                   t_eval=sim_time)
```



Reversible Conversion

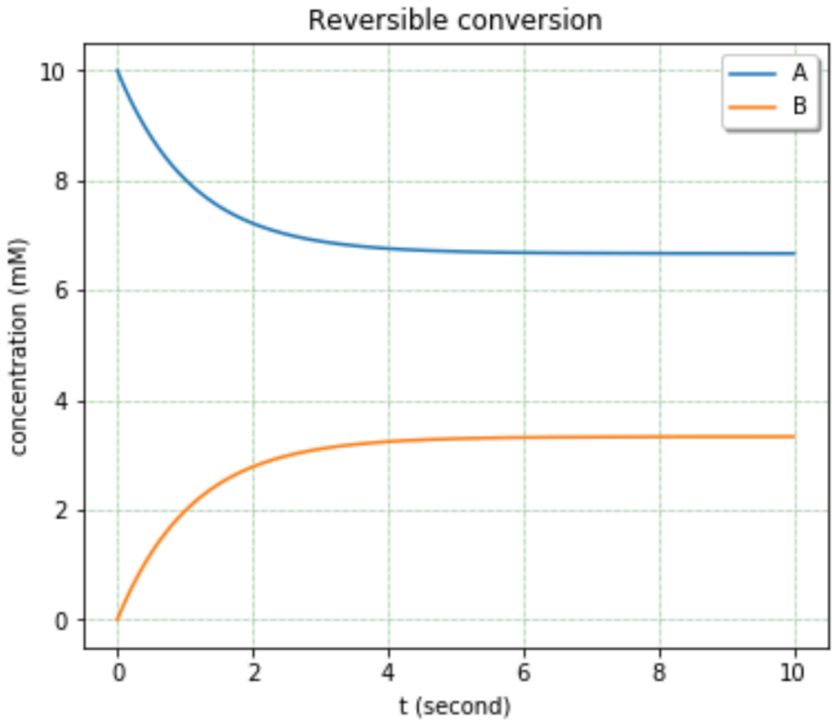
$$A \underset{k_-}{\overset{k_+}{\rightleftharpoons}} B$$

$$\frac{d}{dt}a(t) = k_-b(t) - k_+a(t)$$

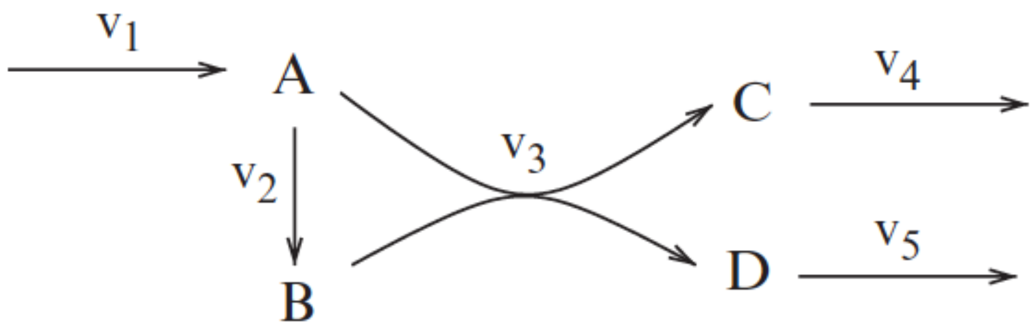
$$\frac{d}{dt}b(t) = k_+a(t) - k_-b(t)$$

```
def decay(t, y):
    kp = 0.3
    km = 0.6
    a, b = y
    dadt = km*b - kp*a
    dbdt = kp*a - km*b
    return np.array([dadt, dbdt])
```

```
y_init = [10,0]
sim_time = np.linspace(0, 10, 1000)

result = solve_ivp(decay,
                   (sim_time[0], sim_time[-1]),
                   y_init,
                   t_eval=sim_time)
```



Open chemical reaction network



$$\frac{d}{dt}a(t) = k_1 - k_2 a(t) - k_3 a(t) b(t)$$

$$\frac{d}{dt}b(t) = k_2 a(t) - k_3 a(t) b(t)$$

$$\frac{d}{dt}c(t) = k_3 a(t) b(t) - k_4 c(t)$$

$$\frac{d}{dt}d(t) = k_3 a(t) b(t) - k_5 d(t)$$

$k_1 = 3mM/s$

$k_2 = 2/s$

$k_3 = 2.5/mM/s$

$k_4 = 3/s$

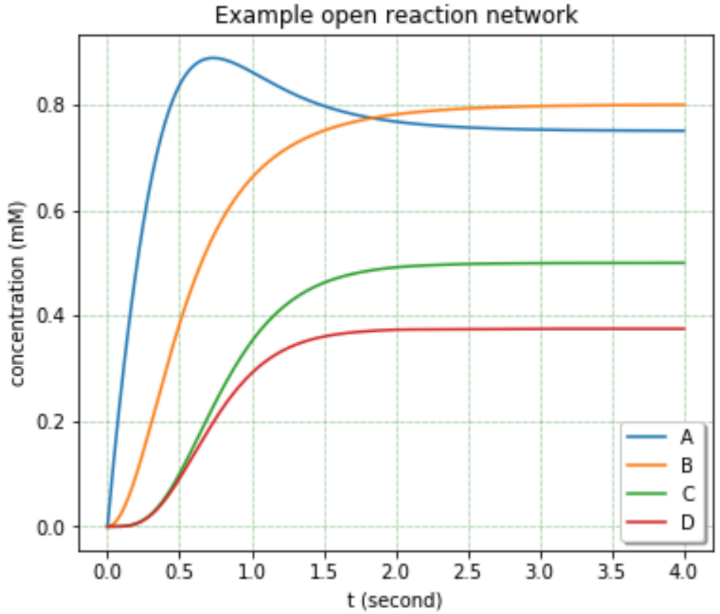
$k_5 = 4/s$

where, $v_1 = k_1$, $v_2 = k_2 a(t)$, $v_3 = k_3 a(t) b(t)$,

$v_4 = k_4 c(t)$, and $v_5 = k_5 d(t)$.

```
def open_reaction_network(t,y):
    k1 = 3
    k2 = 2
    k3 = 2.5
    k4 = 3
    k5 = 4

    a, b, c, d = y

    dadt = k1 - k2*a - k3*a*b
    dbdt = k2*a - k3*a*b
    dcdt = k3*a*b - k4*c
    dddt = k3*a*b - k5*d

    return np.array([dadt, dbdt, dcdt, dddt])
```

```
y_init = (0, 0, 0, 0)
sim_time = np.linspace(0, 4, 1000)

result = solve_ivp(open_reaction_network,
                   (sim_time[0], sim_time[-1]),
                   y_init,
                   t_eval=sim_time)
```



Programming the simulation using "`solve_ivp`" is the easy part. Thus, it should not hinder your research.

The real challenge is in the *formulation* of the ODEs, *analyzing* the system, and *understanding* the biology behind it. Simulation cannot answer your biological questions, but it can help you understand the behavior of the dynamical system that you are studying. So, practice simulation programming and make it your easiest part of scientific endeavor.