The Law of Mass action

The law of mass action is an approach to modelling the dynamic nature of enzymes as they undergo morphological transformations. However, whilst this strategy is commonly used in statistical chemistry, it is not limited to that context, allowing for assessment of transient dynamics and stability in any linear dynamic system. Example systems are the conversion of adenosine diphosphate to adenosine triphosphate and population dynamics, from growth to extinction (see appendix 1 for population models and Appendix 2. To show how they related to the law of mass action). Using the law of mass action, we can predict long term transient dynamics from limited datasets and generate a mathematical framework descriptive of real-world phenomena. An example system, in which A is convert to B at rate k_1 is provided below;

$$[A] \underset{k_2}{\overset{k_1}{\rightleftarrows}} [B]$$

Where A, B are two arbitrary enzymes, $[\cdot]$ denotes their concentation and k_1 and k_2 are the rates of transformation. For this model, the transient dynamics are determined by;

$$\frac{d[A]}{dt} = -k_1[A] + k_2[B]$$
$$\frac{d[B]}{dt} = -k_2[B] + k_1[A]$$

The model is positive definite stable when $\frac{d[A]}{dt} = \frac{d[B]}{dt} = 0$, where $\frac{[A]}{[B]} = \frac{k_2}{k_1} \mid k_1, k_2 > 0$. If $k_1 = 0$ or $k_2 = 0$ then the model is stable with strong damping and will undergo an extinction of sorts in one of its reagents. Hence;

$$\text{If } k_1 = 0 \text{ then } \frac{dB}{dx} = -k_2[B] \;\therefore [B] \underset{t\to\infty}{\longrightarrow} 0$$

If there exists some recursive element, then the model can exhibit unstable behaviours such as if;

$$[A] \underset{k_2}{\overset{k_1}{\rightleftarrows}} [B] + 0.01[A]$$

Then investigating the rate of change in concentration of A as time tends to infinite we see;

$$[A] \underset{t\to\infty}{\longrightarrow} \infty \text{ as } \frac{dA}{dt} = -0.99k_1[A] + k_2[B] \text{ since } [A]_{t+1} = [A]_t + \frac{dA}{dt}$$

Below are the plots of the transient dynamics of the two models previously explored during this document.

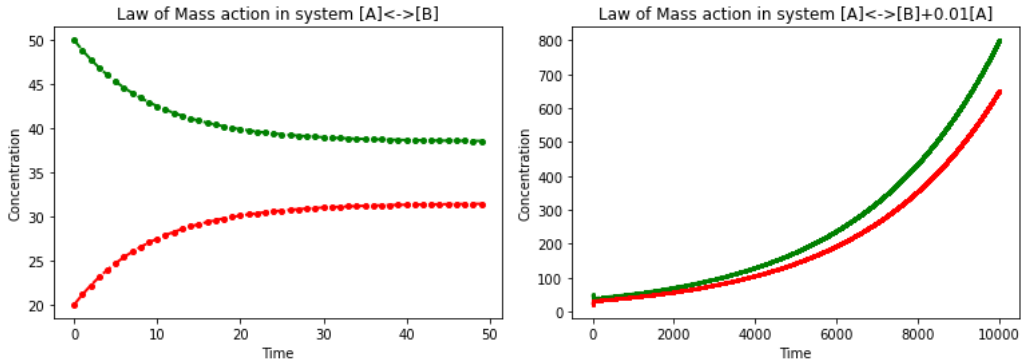

Fig 1.a) Stabilization of [A] over time 1.b) Explosion in the concentration of [A] and [B] respectively

The two models above have identical initial conditions: $[A]_0 = 50, [B]_0 = 20, k_1 = 0.045, k_2 = 0.055$. The only difference in their transient dynamics arises from the positive feedback of the second $[A]$ substrate component, resulting in a runaway in total product.

Whilst the two models used in this document thus far have been in the context of enzyme interactions, it's important to note that this approach is effective for any ordinary differential equation wherein $\frac{d}{dt}$ can be defined. Below we define a more complicated model with multiple stages and feedbacks loops to investigate how model complexity can impact stability and dynamics. Here we define a 3-stage reaction with 2 feedback loops as the following.

$$[A] \overset{k_1}{\rightarrow} [B] \overset{k_2}{\rightarrow} 0.5[C] + 0.5[D]$$
$$[C] \overset{k_3}{\rightarrow} [A]$$
$$[D] \overset{k_4}{\rightarrow} [B]$$

This model is of particular importance due to its feedback loops and damped dynamics allowing for investigations of extinction style events, in which multiple products are dependent on a single substrate. To implement the extinction event, we implement the logic:

$$\text{If } [A]_i < 3 \text{ then } \forall t > i, [A]_t = 0$$

The initial conditions for the model are as follows; $[A]_0 = 0, [B]_0 = 0, [C]_0 = 0, [D]_0 = 0, k_1 = 0.7, k_2 = 0.2, k_3 = 0.3, k_4 = 0.2$. Their transient dynamics are as follows.

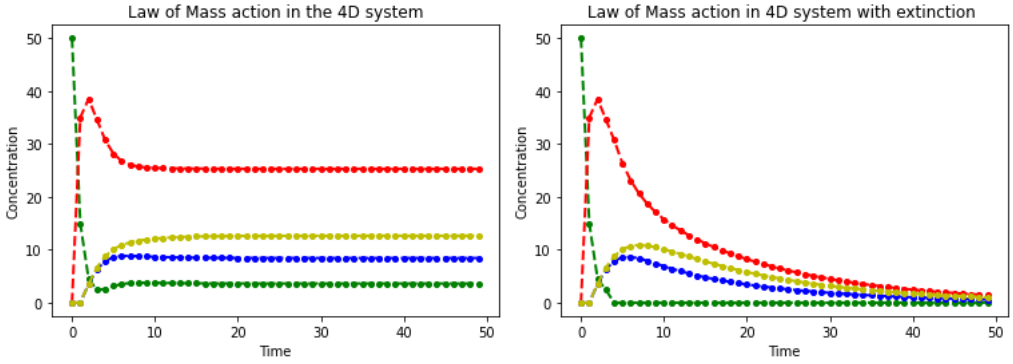

Fig 2.a) Stabilization of the 4D Fig 2.b) Chain extinction as $\frac{dA}{dt} = 0$.

Conclusion

The law of mass action is a powerful method for assessing population dynamics abstract from any physical system though it is most applied in chemical and pharmacokinetic contexts. Hopefully from Appendix 1. and 2. It is clear how it is related to population models and ordinary differential equations more generally. In the following document, 3.2. Tissue compartment models, we will explore how the law of mass action and ordinary differential equations can be applied in series into tissue compartment models and investigate its application in the context of pharmacokinetic modelling.

Appendix 1: Population Models

Population growth in the world shows an exponential increase up to a ceiling determined by environmental and behavioural factors.

$$\frac{dP}{dt} = (b-d)P \rightarrow P(t) = P_0 e^{(b-d)t}$$

To test the effectiveness of this linear model in predicting population changes, we will use the population table described below;

Date	1980	1985	1990	1995	2000
Population	1500	1576	1657	1742	1832

Table 1. Artificial population data

Given we know $P(0) = 1500 = P_0 e^{5k*0} = 1500$, we can extract k to create a simple general model of the population growth dynamics hence;

$$P(1) = 1584 = P_0 e^{5k} \rightarrow \frac{1576}{1500} = e^{5k} \rightarrow k = \frac{\ln\left(\frac{1576}{1500}\right)}{5}$$

This gives an approximation of $k = 0.0098$, which is only a 2% on the actual value ($k = 0.01$) used to generate the artificial data.

Appendix 2: Relationship to population models

In their general form, exponential population models take the form:

$$\frac{dP}{dt} = (b-d)P \rightarrow P(t) = P_0 e^{(b-d)t}$$

Where P is the population count, b is the birth rate, d is the death rate and P_0 is the constant of integration.

$$[A] \underset{k_2}{\overset{k_1}{\rightleftarrows}} [B]$$

Which in its derivative form is given by:

$$\frac{dA}{dt} = -k_1 A + k_2 B \rightarrow A(t) = A_0 e^{-k_1 t} + \frac{k_2 B(t)}{k_1}$$
$$\frac{dB}{dt} = -k_2 B + k_1 A \rightarrow B(t) = B_0 e^{-k_2 t} + \frac{k_1 A(t)}{k_2}$$

Substituting $B(t)$ into $A(t)$, $A(t)$ rearranges to:

$$-k_2 B_0 e^{-k_2 t} = k_1 A_0 e^{-k_1 t}$$

Taking the same form as the population model shown prior.

Python Code:

```
import numpy as np
import matplotlib.pyplot as plt

"""
Case 1: [A]<->[B]
stable
"""

A0 = 50;
```

```
B0 = 20;
k1 = 0.045;
k2 = 0.055;
t_max =50;

A_memory = np.zeros([1,t_max]);
B_memory = np.zeros([1,t_max]);

for i in range(0,t_max):
   if i==0:
      A_memory[0,i]=A0;
      B_memory[0,i]=B0;
   else:
       dA= -k1*A_memory[0,i-1] + k2*B_memory[0,i-1]
       dB = k1*A_memory[0,i-1] -k2*B_memory[0,i-1]

       A_memory[0,i] = A_memory[0,i-1]+dA;
       B_memory[0,i] = B_memory[0,i-1]+dB;

plt.plot(np.transpose(A_memory),'go--', linewidth=2, markersize=4)
plt.plot(np.transpose(B_memory),'ro--', linewidth=2, markersize=4)
plt.xlabel('Time')
plt.ylabel('Concentration')
plt.title('Law of Mass action in system [A]<->[B]')
plt.show();

"""
Case 2: [A]<->[B]+[A]
unstable
"""
A0 = 50;
B0 = 20;
k1 = 0.045;
k2 = 0.055;
t_max =10000;

A_memory = np.zeros([1,t_max]);
B_memory = np.zeros([1,t_max]);

for i in range(0,t_max):
   if i==0:
      A_memory[0,i]=A0;
      B_memory[0,i]=B0;
   else:
       dA= -k1*A_memory[0,i-1] + k2*B_memory[0,i-1] +k2*0.01*A_memory[0,i-1]
       dB = k1*A_memory[0,i-1] -k2*B_memory[0,i-1]
```

```
        A_memory[0,i] = A_memory[0,i-1]+dA;
        B_memory[0,i] = B_memory[0,i-1]+dB;

plt.plot(np.transpose(A_memory),'go--', linewidth=2, markersize=2)
plt.plot(np.transpose(B_memory),'ro--', linewidth=2, markersize=2)
plt.xlabel('Time')
plt.ylabel('Concentration')
plt.title('Law of Mass action in system [A]<->[B]+0.01[A]')
plt.show();

"""
Case 3:
   [A]->[B]->[C]+[D]
   [C]->[A]
   [D]->[B]
Intermediary Points
"""

A0 = 50;
B0 = 0;
C0 = 0;
D0 = 0;

k1 = 0.7
k2 = 0.2
k3 = 0.3
k4 = 0.2
t_max = 50;

A_memory = np.zeros([t_max])
B_memory = np.zeros([t_max])
C_memory = np.zeros([t_max])
D_memory = np.zeros([t_max])

for i in range(0,t_max):
   if i ==0:
      A_memory[i]=A0;
      B_memory[i]=B0;
      C_memory[i]=C0;
      D_memory[i]=D0;
   else:
      dA = -k1*A_memory[i-1] + k3*C_memory[i-1]
      dB = k1*A_memory[i-1] + k4*D_memory[i-1]-k2*B_memory[i-1]
      dC = 0.5*k2*B_memory[i-1]-k3*C_memory[i-1]
      dD = 0.5*k2*B_memory[i-1]-k4*D_memory[i-1]

      A_memory[i] = A_memory[i-1]+dA;
```

```
        B_memory[i] = B_memory[i-1] + dB;
        C_memory[i] = C_memory[i-1] +dC;
        D_memory[i] = D_memory[i-1] +dD;

plt.plot(np.transpose(A_memory),'go--', linewidth=2, markersize=4)
plt.plot(np.transpose(B_memory),'ro--', linewidth=2, markersize=4)
plt.plot(np.transpose(C_memory),'bo--', linewidth=2, markersize=4)
plt.plot(np.transpose(D_memory),'yo--', linewidth=2, markersize=4)
plt.xlabel('Time')
plt.ylabel('Concentration')
plt.title('Law of Mass action in the 4D system')
plt.show();

"""
Case 4: Encoding extinction
        [A]->[B]->[C]+[D]
        [C]->[A]
        [D]->[B]
        if [A] < 3 then dA=0
"""

A0 = 50;
B0 = 0;
C0 = 0;
D0 = 0;

k1 = 0.7
k2 = 0.2
k3 = 0.3
k4 = 0.2
t_max = 50;

A_memory = np.zeros([t_max])
B_memory = np.zeros([t_max])
C_memory = np.zeros([t_max])
D_memory = np.zeros([t_max])

for i in range(0,t_max):
    if i ==0:
        A_memory[i]=A0;
        B_memory[i]=B0;
        C_memory[i]=C0;
        D_memory[i]=D0;
    else:
        if A_memory[i-1]<3:
            dA=0;
            A_memory[i]=0
```

```
        else:
            dA = -k1*A_memory[i-1] + k3*C_memory[i-1]
            A_memory[i] = A_memory[i-1]+dA;
        dB = k1*A_memory[i-1] + k4*D_memory[i-1]-k2*B_memory[i-1]
        dC = 0.5*k2*B_memory[i-1]-k3*C_memory[i-1]
        dD = 0.5*k2*B_memory[i-1]-k4*D_memory[i-1]

        B_memory[i] = B_memory[i-1] + dB;
        C_memory[i] = C_memory[i-1] +dC;
        D_memory[i] = D_memory[i-1] +dD;

plt.plot(np.transpose(A_memory),'go--', linewidth=2, markersize=4)
plt.plot(np.transpose(B_memory),'ro--', linewidth=2, markersize=4)
plt.plot(np.transpose(C_memory),'bo--', linewidth=2, markersize=4)
plt.plot(np.transpose(D_memory),'yo--', linewidth=2, markersize=4)
plt.xlabel('Time')
plt.ylabel('Concentration')
plt.title('Law of Mass action in 4D system with extinction ')
plt.show();
```