

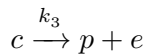
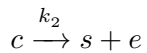
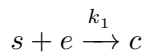
Task4 - Cinética enzimática

Redes Biológicas y Biología de Sistemas

Sandra Mingo Ramírez

Resolver numéricamente $s + e \xrightleftharpoons[k_2]{k_1} c \xrightarrow{k_3} p + e$

Esta reacción se puede desglosar en tres reacciones:

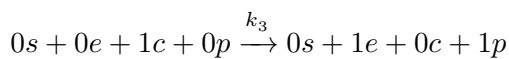
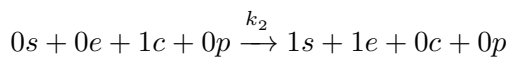
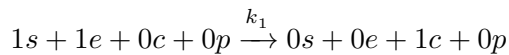


Los pasos para resolver esto son:

1. Sacar las 4 ecuaciones diferenciales: s, e, c, p
2. Resolver numéricamente
3. Mediante la ley de conservación de la masa, pasar de 4 ecuaciones diferenciales a 3, y si es posible a 2.

Ecuaciones diferenciales

Las formulaciones completas son



Obtenemos las matrices A y B:

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Con esto podemos calcular $(B - A)^T$:

$$(B - A) = \begin{pmatrix} -1 & -1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 0 & 1 & -1 & 1 \end{pmatrix}$$

$$(B - A)^T = \begin{pmatrix} -1 & 1 & 0 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

$$k = \begin{pmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{pmatrix}$$

$$X^A = \begin{pmatrix} X_1 X_2 \\ X_3 \\ X_3 \end{pmatrix}$$

Aplicando la fórmula $(B - A)^T \cdot k \cdot X^A$, esto se resuelve de la siguiente forma:

$$\begin{pmatrix} -1 & 1 & 0 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} k_1 x_1 x_2 \\ k_2 x_3 \\ k_3 x_3 \end{pmatrix} = \begin{pmatrix} -k_1 x_1 x_2 + k_2 x_3 \\ -k_1 x_1 x_2 + k_2 x_3 + k_3 x_3 \\ k_1 x_1 x_2 - k_2 x_3 - k_3 x_3 \\ k_3 x_3 \end{pmatrix}$$

De forma que: $dx_1/dt = -k_1 x_1 x_2 + k_2 x_3$

$$dx_2/dt = -k_1 x_1 x_2 + k_2 x_3 + k_3 x_3$$

$$dx_3/dt = k_1 x_1 x_2 - k_2 x_3 - k_3 x_3$$

$$dx_4/dt = k_3 x_3$$

Resolver numéricamente las ecuaciones diferenciales

```

from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt

# Definir el sistema de ecuaciones diferenciales
def differential_equations(y, t, k):
    A, B, C, D = y
    k1, k2, k3 = k
    dA_dt = -k1 * A * B + k2 * C
    dB_dt = -k1 * A * B + k2 * C + k3 * C
    dC_dt = k1 * A * B - k2 * C - k3 * C
    dD_dt = k3 * C
    return [dA_dt, dB_dt, dC_dt, dD_dt]

# Parámetros de reacción
k_1 = 1e3
k_2 = 0.1
k_3 = 0.05
k = [k_1, k_2, k_3]

# Concentraciones iniciales
a_0 = 0.002 # s
b_0 = 0.002 # e
c_0 = 0.0 # c
d_0 = 0.0 # p
y0 = [a_0, b_0, c_0, d_0]

# Tiempo de integración
simulation_time = 100 # Tiempo de reacción
time_points = 100 # Número de mediciones, puntos en el tiempo

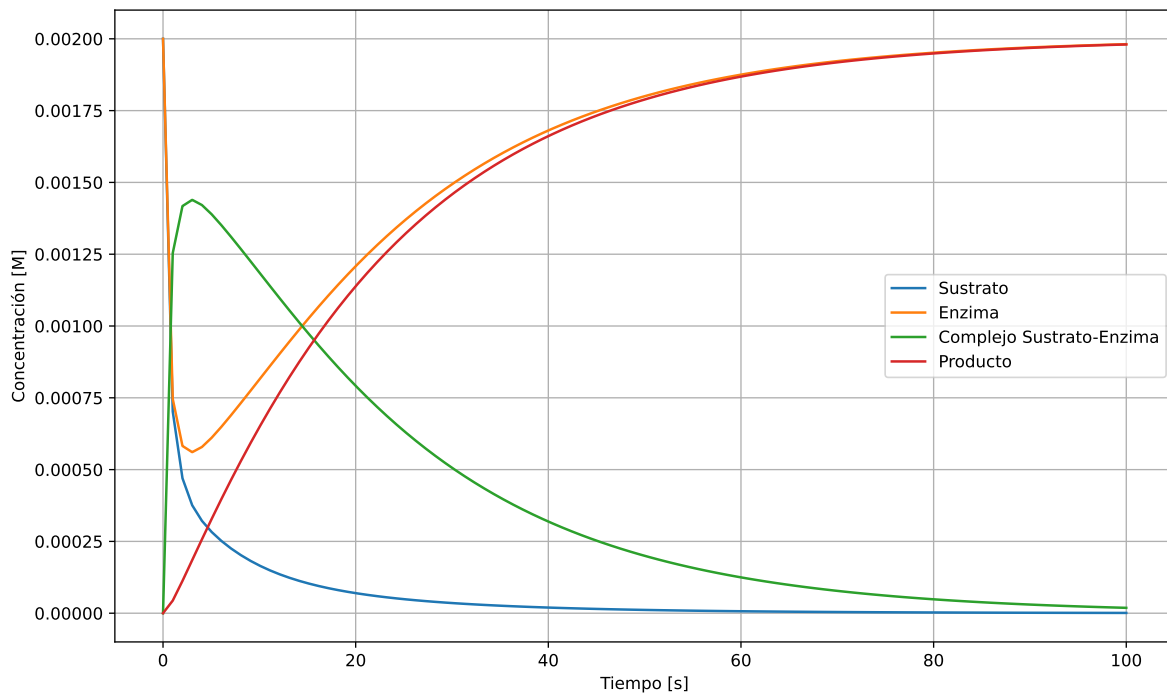
t = np.linspace(0, simulation_time, time_points)

# Resolver ecuaciones diferenciales
solution = odeint(differential_equations, y0, t, args=(k,))

# Graficar resultados
plt.figure(figsize=(10, 6))
plt.plot(t, solution[:, 0], label='Sustrato')
plt.plot(t, solution[:, 1], label='Enzima')
plt.plot(t, solution[:, 2], label='Complejo Sustrato-Enzima')
plt.plot(t, solution[:, 3], label='Producto')

```

```
plt.xlabel('Tiempo [s]')
plt.ylabel('Concentración [M]')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Ley de conservación de masas

Actualmente tenemos 4 ecuaciones diferenciales:

$$ds/dt = -k_1se + k_2c$$

$$de/dt = -k_1se + k_2c + k_3c$$

$$dc/dt = k_1se - k_2c - k_3c$$

$$dp/dt = k_3c$$

Primero tenemos que encontrar

$$C \cdot (B - A)^T = 0$$

$$[C_1 \quad C_2 \quad C_3 \quad C_4] \begin{pmatrix} -1 & 1 & 0 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix} = 0$$

De esta forma obtenemos tres ecuaciones:

$$-C_1 - C_2 + C_3 = 0$$

$$C_1 + C_2 - C_3 = 0$$

$$C_2 - C_3 + C_4 = 0$$

Que realmente son dos porque dos de ellas son la misma pero multiplicadas por -1 . Así, realmente tenemos:

$$C_1 + C_2 - C_3 = 0$$

$$C_2 - C_3 + C_4 = 0$$

Cogiendo la primera, resolvemos por ejemplo:

$$C_1 = -C_2 + C_3$$

De esta forma:

$$(-C_2 + C_3)X_1 + C_2X_2 + C_3X_3 + C_4X_4 = \text{constante}$$

Igualando $C_3 = C_4 = 0$ y $C_2 = 1$:

$$-X_1 + X_2 = \text{constante}$$

Como es constante, es constante a todos tiempos:

$$-X_1(t) + X_2(t) = -X_1(0) + X_2(0)$$

Despejamos $X_2(t)$:

$$X_2(t) = -X_1(0) + X_2(0) + X_1(t)$$

Sustituyendo esto en las ecuaciones diferenciales, nos quedamos con lo siguiente:

$$dx_1/dt = -k_1x_1(-X_1(0) + X_2(0) + X_1) + k_2x_3$$

$$dx_3/dt = k_1x_1(-X_1(0) + X_2(0) + X_1) - k_2x_3 - k_3x_3$$

$$dx_4/dt = k_3x_3$$

Ahora podemos insertar esto en el código:

```

from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt

# Definir el sistema de ecuaciones diferenciales
def differential_equations(y, t, k, initial_conditions):
    A, C, D = y
    a0, b0 = initial_conditions
    k1, k2, k3 = k
    dA_dt = -k1 * A * (-a0 + b0 + A) + k2 * C
    dC_dt = k1 * A * (-a0 + b0 + A) - k2 * C - k3 * C
    dD_dt = k3 * C
    return [dA_dt, dC_dt, dD_dt]

# Parámetros de reacción
k_1 = 1e3
k_2 = 0.1
k_3 = 0.05
k = [k_1, k_2, k_3]

# Concentraciones iniciales
a_0 = 0.002 # s
b_0 = 0.002 # e
c_0 = 0.0 # c
d_0 = 0.0 # p
y0 = [a_0, c_0, d_0]
ci = [a_0, b_0]

# Tiempo de integración
simulation_time = 100 # Tiempo de reacción
time_points = 100 # Número de mediciones, puntos en el tiempo

t = np.linspace(0, simulation_time, time_points)

# Resolver ecuaciones diferenciales
solution = odeint(differential_equations, y0, t, args=(k, ci))

eq_eliminada = solution[:, 0] - a_0 + b_0

# Graficar resultados
plt.figure(figsize=(10, 6))
plt.plot(t, solution[:, 0], label='Sustrato')

```

```

plt.plot(t, solution[:, 1], label='Complejo Sustrato-Enzima')
plt.plot(t, solution[:, 2], label='Producto')
plt.plot(t, eq_eliminada, label='Enzima')

plt.xlabel('Tiempo [s]')
plt.ylabel('Concentración [M]')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

```

