Task4 - Cinética enzimática

Redes Biológicas y Biología de Sistemas

Sandra Mingo Ramírez

Resolver numéricamente $s + e \underset{k_2}{\overset{k_1}{\rightleftharpoons}} c \overset{k_3}{\longrightarrow} p + e$

Esta reacción se puede desglosar en tres reacciones:

$s + e \overset{k_1}{\longrightarrow} c$

$c \overset{k_2}{\longrightarrow} s + e$

$c \overset{k_3}{\longrightarrow} p + e$

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

$1s + 1e + 0c + 0p \overset{k_1}{\longrightarrow} 0s + 0e + 1c + 0p$

$0s + 0e + 1c + 0p \overset{k_2}{\longrightarrow} 1s + 1e + 0c + 0p$

$0s + 0e + 1c + 0p \overset{k_3}{\longrightarrow} 0s + 1e + 0c + 1p$

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

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

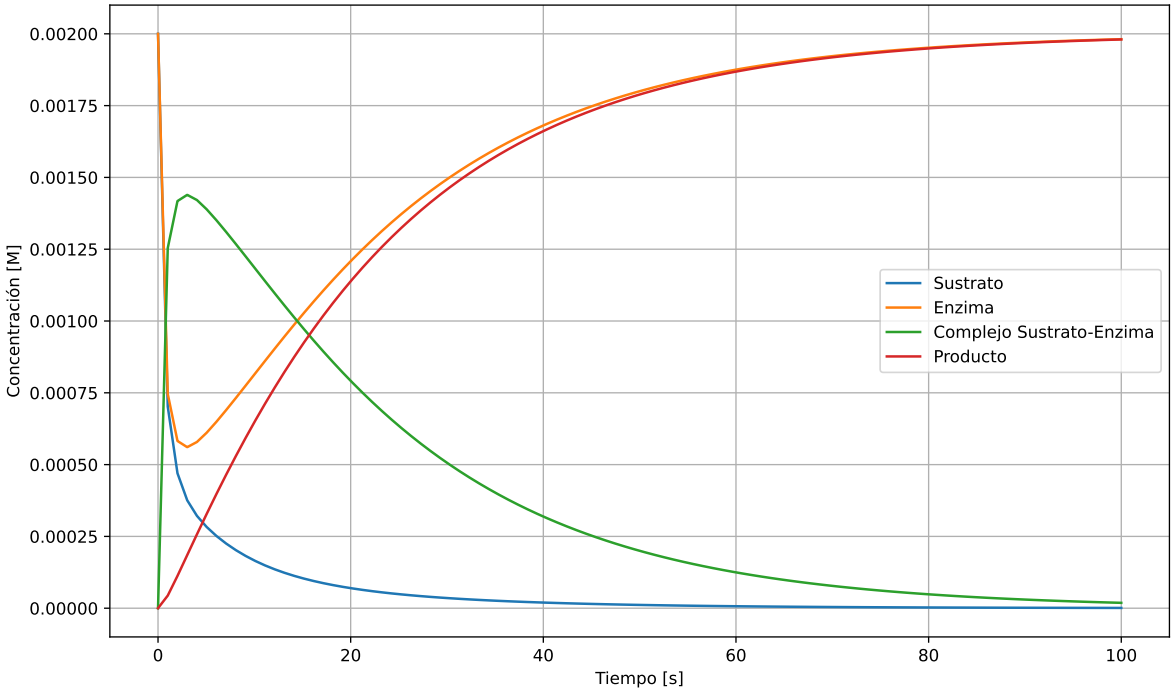
$dx_3/dt = k_1 x_1 x_2 - k_2 x_3 - k_3 x_3$

$dx_4/dt = k_3 x_3$

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:

$ds/dt = -k_1 se + k_2 c$

$de/dt = -k_1 se + k_2 c + k_3 c$

$dc/dt = k_1 se - k_2 c - k_3 c$

$dp/dt = k_3 c$

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:

$C_2 = -C_1 + C_3$

De esta forma:

$C_1X_1 + (-C_1 + C_3)X_2 + C_3X_3 + C_4X_4 = constante$

$C_1X_1 - C_1X_2 + C_3X_2 + C_3X_3 + C_4X_4 = constante$

Igualando $C_1 = C_4$ de las ecuaciones anteriores:

$C_1X_1 - C_1X_2 + C_3X_2 + C_3X_3 + C_1X_4 = constante$

Esto es válido para cualquier valor de C_1 y C_3, por lo que se establece $C_3 = 0$ para obtener las variables que dependen solo de C_1.

$C_1X_1 - C_1X_2 + C_1X_4 = constante$

Independientemente del valor de C_1, esas variables son constantes. Por ello, se puede especificar $C_1 = 1$:

$X_1 - X_2 + X_4 = constante$

Como es constante, es constante a todos tiempos:

$X_1 - X_2 + X_4 = X_1(0) - X_2(0) + X_4(0)$

Despejamos $X_1(t)$:

$X_1(t) = X_1(0) - X_2(0) + X_4(0) + X_2(t) - X_4(t)$

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

$dx_2/dt = -k_1(X_1(0) - X_2(0) + X_4(0) + X_2(t) - X_4(t))x_2 + k_2x_3 + k_3x_3$

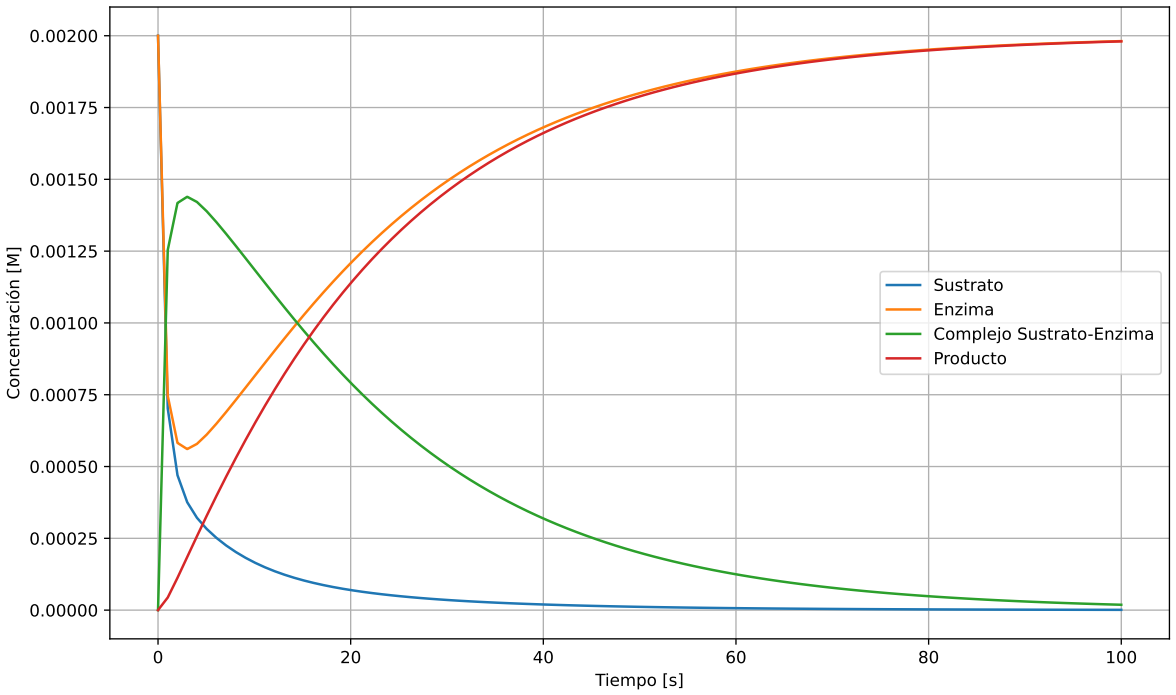
$dx_3/dt = k_1(X_1(0) - X_2(0) + X_4(0) + X_2(t) - X_4(t))x_2 - k_2x_3 - k_3x_3$

$dx_4/dt = k_3x_3$

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):
    B, C, D = y
    a0, b0, d0 = initial_conditions
    k1, k2, k3 = k
    dB_dt = -k1 * (a0 - b0 + d0 + B - D) * B + k2 * C + k3 * C
    dC_dt = k1 * (a0 - b0 + d0 + B - D) * B - k2 * C - k3 * C
    dD_dt = k3 * C
    return [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, c_0, d_0]
ci = [a_0, b_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, ci))

eq_eliminada = a_0 - b_0 + d_0 + solution[:, 0] - solution[:, 2]
```

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

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



Vemos que el gráfico resultante es igual al original, por lo que la simplificación se ha resuelto con éxito. Ahora pasamos de 3 ecuaciones diferenciales a 2. Actualmente, nuestras ecuaciones diferenciales son:

$$dx_2/dt = -k_1(X_1(0) - X_2(0) + X_4(0) + X_2(t) - X_4(t))x_2 + k_2x_3 + k_3x_3$$

$$dx_3/dt = k_1(X_1(0) - X_2(0) + X_4(0) + X_2(t) - X_4(t))x_2 - k_2x_3 - k_3x_3$$

$$dx_4/dt = k_3x_3$$

Previamente habíamos establecido que

$C_1X_1 - C_1X_2 + C_3X_2 + C_3X_3 + C_1X_4 = constante$

Ahora, en lugar de igualar $C_3 = 0$, igualaremos $C_1 = 0$:

$C_3X_2 + C_3X_3 = constante$

Como esto es igual para todos los valores, podemos establecer que $C_3 = 1$:

$X_2 + X_3 = constante$

Como es igual a todos los tiempos:

$X_2 + X_3 = X_2(0) + X_3(0)$

Ahora despejamos X_2:

$X_2 = X_2(0) + X_3(0) - X_3$

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

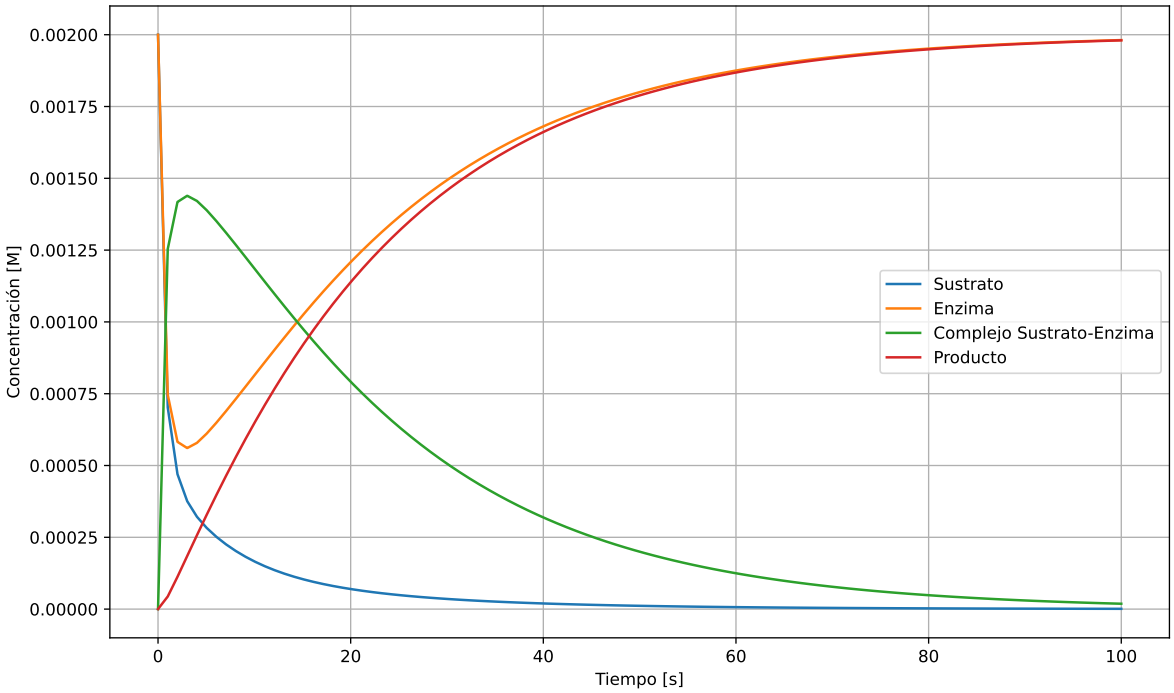
$dx_3/dt = k_1(X_1(0) - X_2(0) + X_4(0) + (X_2(0) + X_3(0) - X_3) - X_4(t))(X_2(0) + X_3(0) - X_3) - k_2x_3 - k_3x_3$

$dx_4/dt = k_3x_3$

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):
    C, D = y
    a0, b0, c0, d0 = initial_conditions
    k1, k2, k3 = k
    dC_dt = k1 * (a0 - b0 + d0 + (b0 + c0 - C) - D) * (b0 + c0 - C) - k2 * C - k3 * C
    dD_dt = k3 * C
    return [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 = [c_0, d_0]
ci = [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, ci))

dB_dt = b_0 + c_0 - solution[:, 0]
dA_dt = a_0 - b_0 + d_0 + dB_dt - solution[:, 1]

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

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


0.00200
0.00175
0.00150
0.00125
0.00100
0.00075
0.00050
0.00025
0.00000
Concentración [M]
0
20
40
60
80
100
Tiempo [s]
Sustrato
Enzima
Complejo Sustrato-Enzima
Producto