Códigos

1.2 Simule el comportamiento del pH (no menos de 1800 s), iniciando con un valor de pH de 6.0. Analice los resultados. Use una válvula inicialmente cerrada.

```
import numpy as np

import matplotlib.pyplot as plt

#Parámetros

Tao = 45  # constante de tiempo [s]

K = 0.07  # ganancia del sistema [pH/% apertura]

d = 2.0   # perturbación

set_point = 6.5

Ap_val = 0

Kp = 2.5  # Ganancia proporcional (ajustable)

t_final = 3600     # Duración en segundos

ph_0 = 6.0         # Valor inicial de #ph

dt = 1           # Paso de tiempo

t = np.arange(0, t_final + dt, dt)



#tiempo de simulacion

t_total = 3600 #segundos

Delta_t = 1

n = int(t_total / Delta_t)
```

```
#Inicio

ph = np.zeros(len(t))

ph[0] = ph_0


#Ecuación del modelo empírico

def dph_dt(ph, Ap_val, d):

   return (-ph + K * Ap_val + d) / Tao     #bval cerrada


#Usando el Método de Euler

for i in range(1, len(t)):

  ph[i] = ph[i-1] + dt * dph_dt(ph[i-1], Ap_val, d)


# Grafica

plt.figure(figsize=(8,5))

plt.plot(t, ph, color='green', label="Sol. Euler")

plt.xlabel('Tiempo [s]', fontsize=12)

plt.ylabel('pH', fontsize=12)

plt.title('pH en el biorreactor', fontsize=14)

plt.grid(True)

plt.legend()

plt.tight_layout()

plt.show()
```

1.3 Simule el comportamiento del sistema ante una variación de d de 2.0 a 1.5 iniciando en un punto de estado estacionario. Evalúe y grafique el error respecto al set-point.

```
import numpy as np

import matplotlib.pyplot as plt


#Parámetros

Tao = 45       #[s]

K = 0.07       #[pH / % apertura]

set_point = 6.5


#Tiempo de simulación

t0, tf, dt = 0, 3600, 1

n = int((tf - t0) / dt)

t = np.linspace(t0, tf, n+1)



# Inicio

pH = np.zeros(n+1)

Bval = np.zeros(n+1)  # sin control

error = np.zeros(n+1)

perturbacion = np.zeros(n+1)


#Condición inicial

d_inicial = 2.0

pH[0] = K * Bval[0] + d_inicial


#Simulación sin control
```

```
for i in range(n):
    d = 2.0 if t[i] < tf / 2 else 1.5
    perturbacion[i] = d
    dpHdt = (-pH[i] + K * Bval[i] + d) / Tao
    pH[i+1] = pH[i] + dpHdt * dt

perturbacion[-1] = perturbacion[-2]
error = set_point - pH

#Gráficas
fig, axs = plt.subplots(3, 1, figsize=(12, 9), sharex=False)

#Para configurar el límite del eje x
xlim = [0, 3600]
xticks = np.arange(0, 3700, 600)  # ticks cada 600 s

#pH
axs[0].plot(t, pH, 'b', label='pH')
axs[0].axhline(y=set_point, color='k', linestyle='-.', label='Setpoint')
axs[0].set_ylabel('pH')
axs[0].set_title('Evolución del pH sin control')
axs[0].set_xlim(xlim)
axs[0].set_xticks(xticks)
axs[0].legend()
axs[0].grid(True)

#Error
axs[1].plot(t, error, 'orange', label='Error (setpoint - pH)')
axs[1].set_ylabel('Error')
```

```
axs[1].set_title('Error respecto al setpoint')
axs[1].set_xlim(xlim)
axs[1].set_xticks(xticks)
axs[1].legend()
axs[1].grid(True)

# Perturbación
axs[2].plot(t, perturbacion, 'r', label='Perturbación (d)')
axs[2].set_ylabel('d')
axs[2].set_xlabel('Tiempo [s]')
axs[2].set_title('Perturbación (acumulación de ácido)')
axs[2].set_xlim(xlim)
axs[2].set_xticks(xticks)
axs[2].legend()
axs[2].grid(True)

plt.tight_layout()
plt.show()
```

1.4 Usando el código del numeral 1.3, simule una variación en Bval para subir el pH a 6.5.

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

#Parámetros
Tao = 45              #[s]
set_point = 6.5
K = 0.07              #(pH / % apertura válvula)
```

```
#Parámetros de controlador PI
Kp = 10              #ganancia proporcional
Ki = 0.05            #ganancia integral

#Tiempo de simulación
t0, tf, dt = 0, 3600, 1   # 1 hora de simulación
n = int((tf - t0) / dt)
t = np.linspace(t0, tf, n + 1)

#Inicio
pH = np.zeros(n + 1)
d = np.zeros(n + 1)
error = np.zeros(n + 1)
Bval_array = np.zeros(n + 1)

#Condición inicial en estado estacionario sin el control
d[0] = 2.0
Bval_init = 0
pH[0] = K * Bval_init + d[0]  #pH en estado estacionario inicial

#Variable para la integral del error
integral_error = 0

#Simulacion
for i in range(n):
   #La perturbación la cambio a mitad de la simulación
   d[i] = 2.0 if t[i] < tf / 2 else 1.5
```

```
  #Error respecto al set_point
  error[i] = set_point - pH[i]

  #Para acumular integral del error
  integral_error += error[i] * dt

  #Control PI:
  u = Kp * error[i] + Ki * integral_error

  #Señal de control entre 0 y 100%
  Bval = max(0, min(100, u))
  Bval_array[i] = Bval  #para llevar un registro de la apertura de válvula en cada instante

  #Modelo con control:
  dpH_dt = (-pH[i] + K * Bval + d[i]) / Tao
  pH[i + 1] = pH[i] + dpH_dt * dt

#Últimos valores para d, error y Bval
d[-1] = d[-2]
error[-1] = set_point - pH[-1]
Bval_array[-1] = Bval_array[-2]

#Gráficas
fig, axs = plt.subplots(3, 1, figsize=(12, 10), sharex= False, gridspec_kw={'height_ratios': [2, 2, 1]})

xlim = [0, 3600]
xticks = np.arange(0, 3700, 600)
```

```
#1) pH con set point
axs[0].plot(t, pH, label='pH', color='blue', linewidth=2)
axs[0].axhline(y=set_point, color='green', linestyle='--', linewidth=2, label='Set point')
axs[0].set_ylabel('pH')
axs[0].set_ylim(1, 7)
axs[0].set_title('Evolución del pH con Set point')
axs[0].set_xlim(xlim)
axs[0].set_xticks(xticks)
axs[0].legend()
axs[0].grid(True)

#2) pH y apertura válvula
axs[1].plot(t, pH, label='pH', color='blue', linewidth=2)
axs[1].set_ylabel('pH', color='blue')
axs[1].set_ylim(1, 7)
axs[1].tick_params(axis='y', labelcolor='blue')
axs[1].set_title('pH y Apertura de válvula')
axs[1].set_xlim(xlim)
axs[1].set_xticks(xticks)

ax2 = axs[1].twinx()
ax2.plot(t, Bval_array, label='Bval (% apertura válvula)', color='red', linewidth=2)
ax2.set_ylabel('Apertura válvula (%)', color='red')
ax2.tick_params(axis='y', labelcolor='red')
ax2.set_ylim(0, 110)

lines1, labels1 = axs[1].get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
axs[1].legend(lines1 + lines2, labels1 + labels2, loc='upper right')
```

```
axs[1].grid(True)

#3) pH y perturbación d
axs[2].plot(t, pH, label='pH', color='blue', linewidth=2)
axs[2].set_ylabel('pH', color='blue')
axs[2].set_ylim(1, 7)
axs[2].tick_params(axis='y', labelcolor='blue')
axs[2].set_title('pH y Perturbación (d)')
axs[2].set_xlim(xlim)
axs[2].set_xticks(xticks)
axs[2].set_xlabel('Tiempo [s]')

ax3 = axs[2].twinx()
ax3.plot(t, d, label='Perturbación d', color='black', linewidth=2)
ax3.set_ylabel('d', color='black')
ax3.tick_params(axis='y', labelcolor='black')
ax3.set_ylim(1, 2.2)

lines1, labels1 = axs[2].get_legend_handles_labels()
lines2, labels2 = ax3.get_legend_handles_labels()
axs[2].legend(lines1 + lines2, labels1 + labels2, loc='upper right')
axs[2].grid(True)

plt.tight_layout(pad=3)
plt.show()
```

2.1. (30%) Simule la dinámica de X y S en el biorreactor anterior.

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

#Parámetros
V = 50              #Volumen biorreactor [m^3]
Y = 0.6             #Coeficiente rendimiento Biomasa-Sustrato
mu_max = 0.5        #Vel_max de crecimiento [1/h]
K1 = 0.35           #Parámetro de saturación [kg/m^3]
K2 = 5              #Parámetro de inhibición [m^3/kg]

#Condiciones iniciales
S0 = 10             #Concentración Sustrato en el biorreactor [kg/m^3]
X0 = 0.1            #Concentración de Biomasa en el biorreactor [kg/m^3]

#Variables de operación
Sin=50              #Concentración Sustrato a la entrada del biorreactor [kg/m^3]
F = 3               ##Caudal o Flujo de operación (F_in=F_out) [m^3/h]

#Tiempo
t0 = 0              #Tiempo incial [h]
tf = 500            #Tiempo final [h]
dt = 0.0001         #Paso [h] - cada cuanto tiempo se calcula un punto
n = int((tf - t0) / dt)
t = np.linspace(t0, tf, n + 1)

#Inicio
S = np.zeros(n + 1)
X = np.zeros(n + 1)
Sin = np.full(n + 1, Sin)  #Vector constante
```

```
S[0] = S0
X[0] = X0

#Simulación
for i in range(n):
    mu = mu_max * S[i] / (K2 * S[i] + K1)
    dXdt = mu * X[i] - (F/V) * X[i]
    dSdt = (F/V) * Sin[i] - (F/V) * S[i] - (mu * X[i])/Y

    X[i + 1] = X[i] + dXdt * dt
    S[i + 1] = S[i] + dSdt * dt

#Graficas
fig, axs = plt.subplots(3, 1, figsize=(12, 10), sharex= False)

#Panel 1: X y S
axs[0].plot(t, X, label='Biomasa X (kg/m³)', color='blue')
axs[0].plot(t, S, label='Sustrato S (kg/m³)', color='green')
axs[0].set_ylabel('Concentración')
axs[0].set_title('Concentración de Biomasa y Sustrato')
axs[0].legend()
axs[0].grid(True)

#Panel 2: Caudal F (constante)
axs[1].plot(t, np.full_like(t, F), label='Caudal F (m³/h)', color='red')
axs[1].set_ylabel('Caudal (m³/h)')
axs[1].set_title('Caudal de operación')
axs[1].legend()
axs[1].grid(True)
```

```
#Panel 3: Concentración de sustrato de entrada Sin (constante)
axs[2].plot(t, Sin, label='Sustrato entrada Sin (kg/m³)', color='purple')
axs[2].set_xlabel('Tiempo (h)')
axs[2].set_ylabel('Concentración S_in')
axs[2].set_title('Concentración de sustrato en la alimentación')
axs[2].legend()
axs[2].grid(True)

plt.tight_layout()
plt.show()
```

2.2 Por problemas de suministro de la melaza con la que se prepara el sustrato, sólo se dispondrá de una solución de azúcares para alimentar el biorreactor con 15 kg/m^3, entre las 150 y 174 h. Simule en el mismo código, este problema operativo

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

#Parámetros
V = 50
Y = 0.6
mu_max = 0.5
K1 = 0.35
K2 = 5

#Condiciones iniciales
S0= 10
X0= 0.1

#Variables de operación
```

```
F= 3

#Tiempo
t0 = 0
tf = 500 #[h]
dt = 0.0001
n = int((tf - t0) / dt)
t = np.linspace(t0, tf, n + 1)

#Inicio
S= np.zeros(n + 1)
X= np.zeros(n + 1)
Sin= np.zeros(n + 1)

S[0] = S0
X[0] = X0

#Simulación
for i in range(n):
    #Cambio en concentración sustrato de entrada
    if 150 <= t[i] <= 174:
        Sin[i] = 15
    else:
        Sin[i] = 50

    mu = mu_max * S[i] / (K2 * S[i] + K1)
    dXdt = mu * X[i] - (F / V) * X[i]
    dSdt = (F / V) * Sin[i] - (F / V) * S[i] - (mu * X[i]) / Y

    X[i + 1] = X[i] + dXdt * dt
```

```
    S[i + 1] = S[i] + dSdt * dt

Sin[-1] = Sin[-2]  #Para evitar un valor indefinido al final

#Graficas
fig, axs = plt.subplots(3, 1, figsize=(12, 10), sharex= False)

#Panel 1: X y S
axs[0].plot(t, X, label='Biomasa X (kg/m³)', color='blue')
axs[0].plot(t, S, label='Sustrato S (kg/m³)', color='green')
axs[0].set_ylabel('Concentración')
axs[0].set_title('Concentración de Biomasa y Sustrato (Con problema operativo)')
axs[0].legend()
axs[0].grid(True)

#Panel 2: Caudal F (constante)
axs[1].plot(t, np.full_like(t, F), label='Caudal F (m³/h)', color='red')
axs[1].set_ylabel('Caudal (m³/h)')
axs[1].set_title('Caudal de operación')
axs[1].legend()
axs[1].grid(True)

#Panel 3: Concentración de sustrato de entrada Sin (variable)
axs[2].plot(t, Sin, label='Sustrato entrada Sin (kg/m³)', color='purple')
axs[2].set_xlabel('Tiempo (h)')
axs[2].set_ylabel('Concentración S_in')
axs[2].set_title('Concentración de sustrato en la alimentación (variable)')
axs[2].legend()
axs[2].grid(True)
```

```
plt.tight_layout()
plt.show()
```