Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Matemática

Laboratorio 2

Controlabilidad, observabilidad, estabilidad y detectabilidad

Estudiantes: = Maximiliano S. Lioi
Benjamin Pastene
Profesor: = Héctor Ramírez C
Auxiliar: = Matías V. Vera
Ayudante de laboratorio: S. Adrián Arellano
Santiago de Chile

Índice de Contenidos

Índice de Figuras

Índice de Códigos

El objetivo del problema es diseñar un control que permita llevar los tres tanques a las concentraciones $\left(C^1_{t_f}, C^2_{t_f}, C^3_{t_f}\right) = (0,0,0)\,[kg/\mathrm{m}^3]$.

1. Parte A: Modelamiento y simulación

1.1. Ejercicio 1:

Mostrar que la dinámica del sistema resulta ser:

$$(S)\begin{cases} \dot{C}_1(t) = C_1(t)\frac{-f_1}{V_1+t\cdot\Delta_1} + u(t)\frac{f_1}{V_1+t\cdot\Delta_1} \\ \dot{C}_2(t) = C_1(t)\frac{f_4}{V_2+t\cdot\Delta_2} - C_2(t)\frac{-(f_4+f_5)}{V_2+t\cdot\Delta_2} + u(t)\frac{2f_5}{V_2+t\cdot\Delta_2} \\ \dot{C}_3(t) = C_1(t)\frac{f_3}{V_3+t\cdot\Delta_3} + C_2(t)\frac{f_6}{V_3+t\cdot\Delta_3} + C_3(t)\frac{-(f_3+f_6)}{V_3+t\cdot\Delta_3} \end{cases}$$

Donde $u(t)$ es el control, y $\Delta_1 := (f_1 - f_2 - f_3 - f_4)\,, \Delta_2 := (f_4 + f_5 - f_6 - f_7)$ y $\Delta_3 := (f_3 + f_6 - f_8)$.

Solución:

Dado los datos del problema, se puede probar que en el caso de un tanque con n tubos de entrada y k de salida cumple que

$$V(t) = V_0 + \sum_k ((f_e^k) - f_s)t \;\; ; \; \dot{C}(t) = \frac{\sum_k (C_e^k(t) f_e^k) - C(t)\sum_k (f_e^k)}{V(t)}$$

Podemos controlar la concentración que entra por el flujo f_1, es decir,

$$C_e^1(t) = u(t)$$

Además, por suposiciones del problema, la concentración que entra por f_5 es el doble de aquella que entra por f_1, es decir,

$$C_e^5(t) = 2u(t)$$

Estudiando las dinámicas, tenemos que en el tanque 1, entra flujo por f_1 y sale por f_2, f_3, f_4, se tiene la dinámica

$$\dot{C}_1(t) = \frac{C_e^1 f_1 - C_1(t) f_1}{V_1 + t\Delta_1}$$

$$= C_1(t)\frac{-f_1}{V_1 + t\Delta_1} + u(t)\frac{f_1}{V_1 + t\Delta_1}$$

Con respecto al tanque 2, entra flujo por f_4 y f_5, como el flujo f_4 llega desde el tanque 1, como el tanque homogeneiza la mezcla, se deduce que

$$C_e^4(t) = C_1(t)$$

Pues llega con la concentración del tanque 1 que es homogenea en todo tiempo

Sale flujo por f_6 y f_7, por las condiciones del problema, $C_e^5(t) = 2u(t)$, escribiendo la dinámica se tiene

$$\begin{aligned}\dot{C}_2(t) &= \frac{C_e^4(t)f_4 + C_e^5(t)f_5 - C_2(t)(f_4+f_5)}{V_2 + t\Delta_2} \\ &= C_1(t)\frac{f_4}{V_2+t\Delta_2} + u(t)\frac{2f_5}{V_2+t\Delta_2} - C_2(t)\frac{f_4+f_5}{V_2+t\Delta_3}\end{aligned}$$

Con respecto al tanque 3, entra flujo por f_3 desde T_1 y por f_6 desde T_2. Luego como las mezclas se vuelven homogeneas, se tiene que

$$C_e^3(t) = C_1(t) \quad ; \quad C_e^6 = C_2(t)$$

Por lo que la dinámica queda

$$\begin{aligned}\dot{C}_3(t) &= \frac{C_e^3 f_3}{V_3+t\Delta_3} + \frac{C_e^6 f_6}{V_3+t\Delta_3} - \frac{C_3(t)(f_3+f_6)}{V_3+t\Delta_3} \\ &= C_1(t)\frac{f_3}{V_3+t\Delta_3} + C_2(t)\frac{f_6}{V_3+t\Delta_3} + C_3(t)\frac{-(f_3+f_6)}{V_3+t\Delta_3}\end{aligned}$$

1.2. Ejercicio 2:

Se escribe el sistema matricial de la forma $\dot{X} = AX + BU$, considerando los datos los flujos, volumenes iniciales y concentraciones iniciales siguientes siguientes (en [m^3/s]):

=

f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
4	1	2	1	5	3	3	5

=

V_1	V_2	V_3
200	100	300

=

C_0^1	C_0^2	C_0^3
5000	0	1000

Solución:
Se tiene que $\Delta_1 = \Delta_2 = \Delta_3 = 0$, por lo que el sistema $\dot{X} = AX + BU$ es independiente del tiempo y con la forma

$$A = \begin{bmatrix} -\frac{4}{200} & 0 & 0 \\ \frac{1}{100} & -\frac{6}{100} & 0 \\ \frac{2}{300} & \frac{3}{300} & -\frac{5}{300} \end{bmatrix} \quad B = \begin{bmatrix} \frac{4}{200} \\ \frac{10}{100} \\ 0 \end{bmatrix}$$

1.3. Ejercicio 3

Reescriba el nuevo sistema autónomo, y utilizando el comando *solve ivp* simule trayectorias del sistema lineal para distintos controles (nulo, constantes, sinusoidales, feedbacks, bang bang, etc).

Solución:
Caso Nulo: $u(t) = 0$

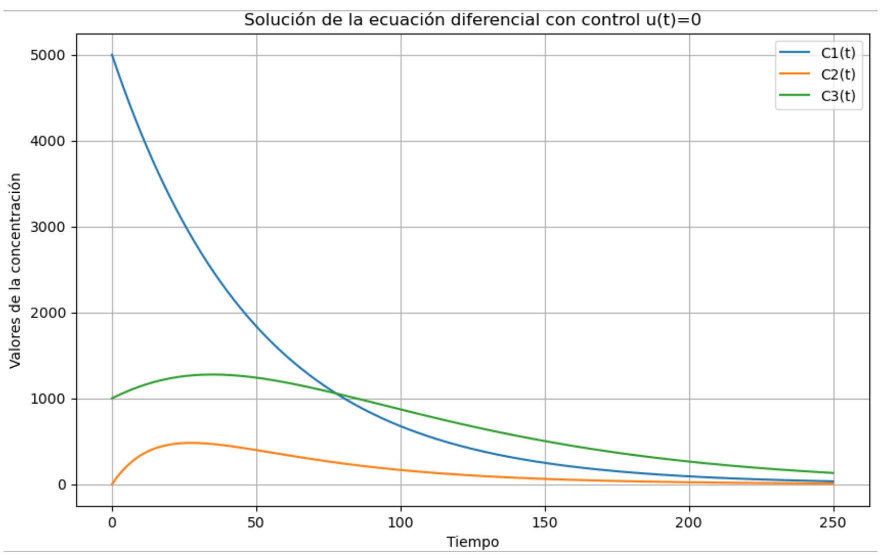

Figura 1: Sol. de la ec. dif. con control nulo.

Caso Constante: $u(t) = 2000$

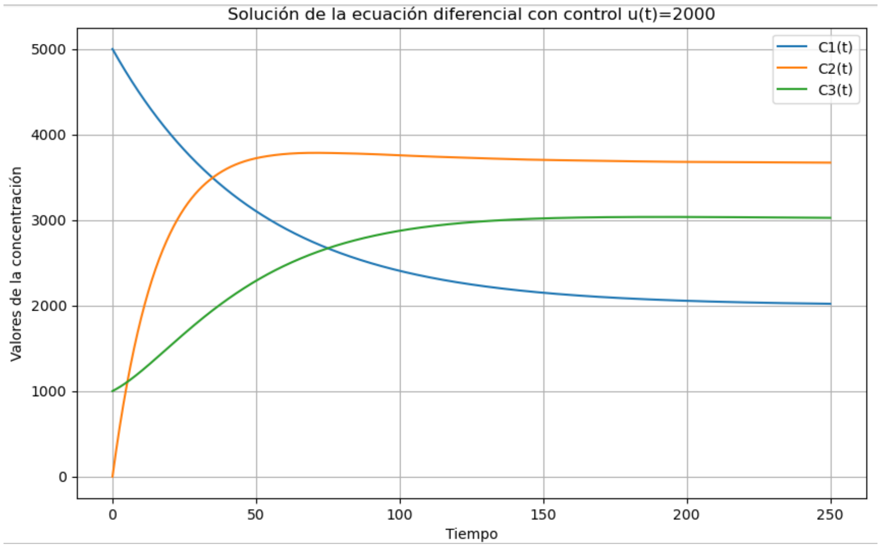

Figura 2: Sol. de la ec. dif. con control cte.

Caso Sinusoidal: $u(t) = 2000 \cdot cos(t)$

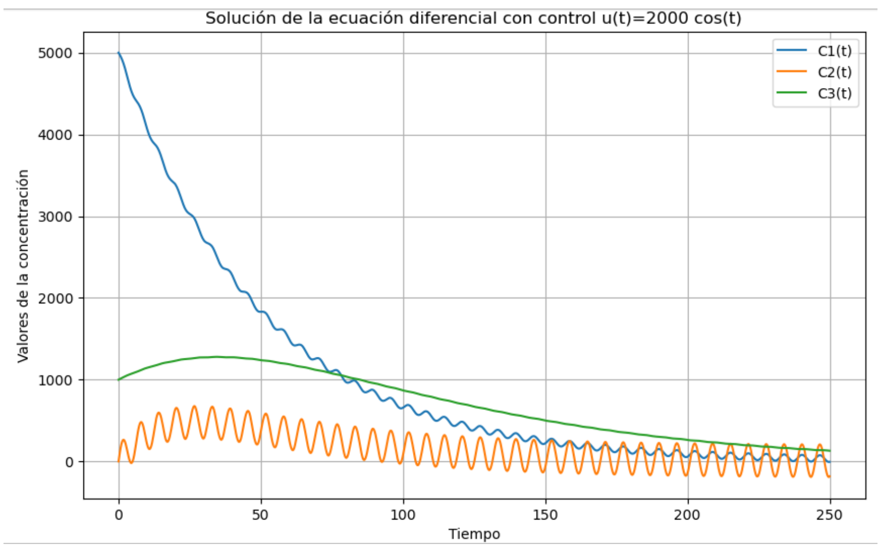

Figura 3: Sol. de la ec. dif. con control sinusoidal.

Caso Bang-bang: $u(t) = \begin{cases} 3000 & si \quad 0 \leq t \leq 5 \\ -3000 & si \quad 5 < t \leq 10 \\ 0 & si \quad t > 10 \end{cases}$

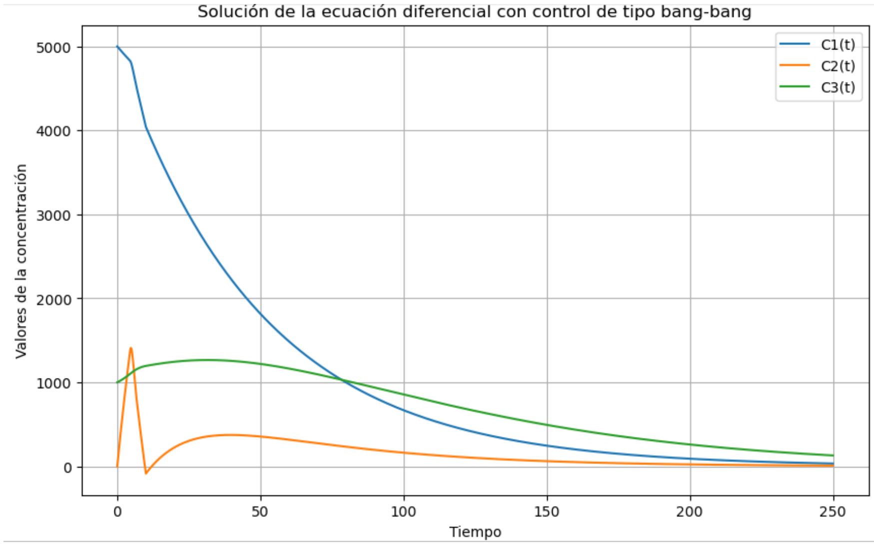

Figura 4: Sol. de la ec. dif. con control de tipo bang-bang.

Caso Feedback: $u(t) = K \cdot x(t) = \begin{pmatrix} 10 & 2 & 4 \end{pmatrix} \cdot x(t)$

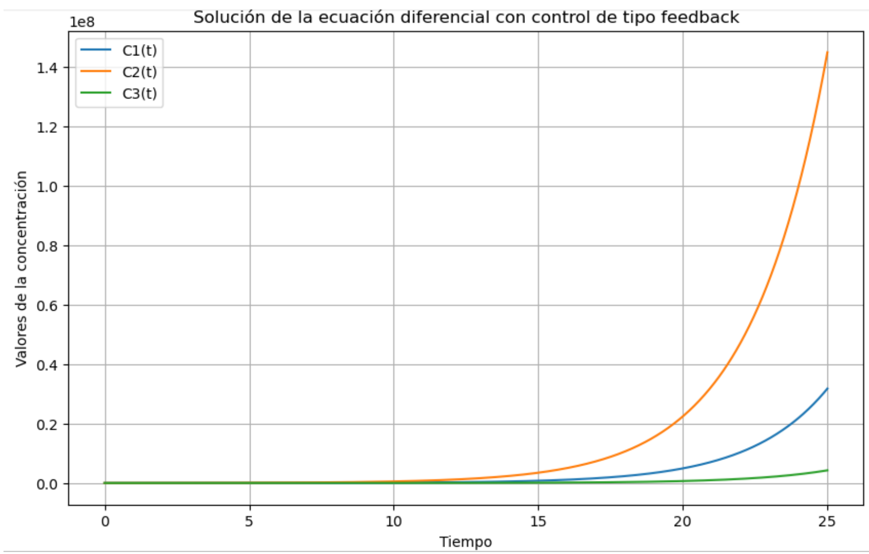

Figura 5: Sol. de la ec. dif. con control de tipo feedback.

2. Parte B: Controlabilidad, observabilidad y estabilidad

2.1. Ejercicio 4

Utilizando python (sin el paquete de control), calcule la matriz de controlabilidad para el sistema. Compare el resultado con el obtenido usando el comando *ctrb* del paquete *control* de python. ¿Es el sistema controlable?

Solución:

Código 1: Cálculo de la matriz de Kalman sin paquete de control

```
# Calculo de la matriz de Kalman
# Definir las matrices A y B
A = np.array([[-4/200, 0, 0], [1/100, -6/100, 0], [2/300, 0.01, -5/300]])
B = np.array([4/200, 10/100, 0]).reshape(-1, 1)  # Convierte B en una matriz de
    ↪ una sola columna
# Definir matrices para la matriz de Kalman
AB = np.dot(A, B)
A2B = np.dot(np.dot(A, A), B)
Kalman = np.concatenate((B, AB, A2B), axis=1)
print(Kalman)
```

Código 2: Cálculo de la matriz de Kalman con ctrb

```
from control import ctrb

Kalman_ctrb = ctrb(A,B.reshape(-1,1))

print(Kalman)
```

En ambos códigos el resultado es el siguiente:

Código 3: Matriz de Kalman resultante en ambos casos

```
[[ 2.00000000e-02 -4.00000000e-04  8.00000000e-06]
 [ 1.00000000e-01 -5.80000000e-03  3.44000000e-04]
 [ 0.00000000e+00  1.13333333e-03 -7.95555556e-05]]
```

Se verifica que son iguales. Calculando el rango de la matriz rg = 3, se verifica que el sistema es controlable, pues tiene rango completo.

2.2. Ejercicio 5

Usualmente es difícil conocer completamente las variables de estado ya que sólo podemos obtener observaciones imprecisas de estas. Por esto, en lo que sigue, supondremos que solamente observamos C_2 y C_3. Esto nos lleva a considerar un observador de la forma:

$$\vec{Y} = \mathcal{C}\vec{X}$$

Identifique $\mathcal{C}$ y utilice python (sin el paquete de control) para calcular la matriz de observabilidad del sistema (S) - (1). Compare con lo obtenido usando el comando obsv del paquete control de python. ¿Es el sistema observable?

Solución:
Consideramos un observador de los estados C_2, C_3 de la forma

$$Y = CX$$

La matriz C debe ser de la forma

$$C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Para que la observación sea

$$Y = \begin{bmatrix} C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X$$

Análogamente a lo anterior calculamos la matriz de observabilidad del sistema sin y con el comando **obsv**:

Código 4: Cálculo de la matriz de observabilidad sin el comando obsv

```
# Escribimos la matriz de Observabilidad
# Definir las matrices A y C
A = np.array([[-4/200, 0, 0], [1/100, -6/100, 0], [2/300, 0.01, -5/300]])
C = np.matrix([[0, 1, 0], [0, 0, 1]])
CA = np.dot(C, A)
CA2 = np.dot(C, np.dot(A, A))
Obs = np.concatenate((C, CA, CA2), axis=0)
print('Matriz de observabilidad:', Obs)
```

Código 5: Cálculo de la matriz de observabilidad con el comando obsv

```
from control import obsv
C = np.matrix([[0, 1, 0], [0, 0, 1]])
```

```
O = obsv(A,C)
print('Matriz de observabilidad:', O)
rg_O = np.linalg.matrix_rank(O)
print('El rango de la Matriz de Observabilidad es:', rg_O)
```

Se verifica la igualdad de ambos métodos obteniendo:

Código 6: Matriz de observabilidad

```
[[ 0.00000000e+00  1.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]
 [ 1.00000000e-02 -6.00000000e-02  0.00000000e+00]
 [ 6.66666667e-03  1.00000000e-02 -1.66666667e-02]
 [-8.00000000e-04  3.60000000e-03  0.00000000e+00]
 [-1.44444444e-04 -7.66666667e-04  2.77777778e-04]]
```

Para ver si el sistema es observable basta calcular el rango de la matriz $\mathcal{O}$, el cual es $rg(\mathcal{O}) = 3$, por lo que $\mathcal{O}$ tiene rango completo y así el sistema es observable.

2.3. Ejercicio 6

A partir de lo aprendido en clases, calcule la forma canónica de Brunovski de los sistemas del ejercicio anterior (sin utilizar el toolbox de Control). Compare con los resultados obtenidos al utilizar *canonical form*.

Solución:

Calculemos la forma canónica de Brunovski del sistema, para ello, debemos buscar el polinomio característico $p_A(\lambda) = det(A - \lambda I)$.

Computando vía subdeterminante en la primera fila

$$det(A-\lambda I) = \det\begin{bmatrix} -\frac{4}{200}-\lambda & 0 & 0 \\ \frac{1}{100} & -\frac{6}{100}-\lambda & 0 \\ \frac{2}{300} & \frac{3}{300} & -\frac{5}{300}-\lambda \end{bmatrix} = (-\frac{1}{50}-\lambda)\det\begin{bmatrix} -\frac{6}{100}-\lambda & 0 \\ \frac{1}{100} & -\frac{1}{60}-\lambda \end{bmatrix}$$

$$= (-\frac{1}{50}-\lambda)(\frac{6}{100}+\lambda)(\frac{1}{60}+\lambda)$$

$$= -(\lambda^3 + \frac{29}{300}\lambda^2 + \frac{19}{7500}\lambda + \frac{1}{50000})$$

Luego

$$p_A(\lambda) = \lambda^3 + a_1\lambda^2 + a_2\lambda + a_3$$

Tenemos $a_1 = -\frac{29}{300} \approx -0.0967$, $a_2 = -\frac{19}{7500} \approx -0.00253$, $a_3 = -\frac{1}{50000} \approx -0.00002$, con lo que se obtiene la forma de Brunovski

$$\hat{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.00002 & -0.00253 & -0.0967 \end{bmatrix} \ ; \ \hat{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Utilizamos la librería *control* junto al método *canonical form* para comparar los resultados obtenidos, teniendo que aplicar una sola observación pues se tienen problemas para sistemas observados por más de una variable.

Código 7: Cálculo con canonical form

```
import control as ctrl
# Definimos matrices
A = np.matrix([[-4/200, 0, 0], [1/100, -6/100, 0], [2/300, 0.01, -5/300]])
B = np.array([4/200, 10/100, 0]).reshape(-1, 1)
C = np.array([0, 1, 0])
# Creamos sistema de control (con una sola observacion)
sys = ctrl.ss(A,B,C,0)
A_Brunovski = ctrl.canonical_form(sys, form='reachable')
A_Brunovski[0]
```

Nos entrega el sistema canónico de la forma

$$\hat{A} = \begin{bmatrix} -0.0967 & -0.00253 & -0.00002 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \ ; \ \hat{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Que notamos representa un sistema teóricamente equivalente al propuesto con la forma vista en clases.

3. Parte C. Reguladores y estabilizadores

3.1. Ejercicio 7

Construya un estabilizador por feedback lineal, $\vec{U} = -K\vec{X}$ para una matriz K apropiada, para el sistema (S). Para esto, utilice los comandos *place* y *lqr* para obtener distintas matrices de ganancia K. Con el comando eig de numpy. linalg verifique si los sistemas son estabilizables. Compare los resultados obtenidos al utilizar ambos comandos. Simule las trayectorias obtenidas por estos controles.

Solución:
Veamos primero la construcción del estabilizador con *place*

Código 8: Construcción del estabilizador con place

```python
import control as ctrl
import numpy as np
# Estructura
# control.place(A, B, p)
# A (2D array_like) – Dynamics matrix
# B (2D array_like) – Input matrix
# p (1D array_like) – Desired eigenvalue locations
# Definimos matrices
A = np.matrix([[-4/200, 0, 0], [1/100, -6/100, 0], [2/300, 0.01, -5/300]]) #(3 x 3)
B = np.matrix([4/200, 10/100, 0]).T #(3 x 1)
# Construimos estabilizador por feedback lineal
def system_dynamics_linear_feedback(K):
    def system_dynamics_feedback(t, X):
        return np.dot(A, X.T) + np.dot(B, np.dot(-K,X.T))
    return system_dynamics_feedback
# Construcción de K - place (1 x 3)
K = ctrl.place(A, B, [-2, -4, -1])
K_ob = ctrl.place(A, B, [-2, -4, -1])
print("La matriz K=", K)
# Condiciones iniciales
X0 = np.array([200, 5, 100])
# Intervalo de tiempo
t_span = (0, 10)
solucion_feedback =  solve_ivp(system_dynamics_linear_feedback(K_ob), t_span, X0, method='RK45', t_eval=np.linspace(0, 10, 1000)) #Feedback
grafica_solucion(solucion_feedback, "de tipo feedback")
```

Lo que nos entrega:

```
La matriz K= [[-1024562.26750565   204981.48683446   699174.58330735]]
```

y el siguiente grafico de la solución:

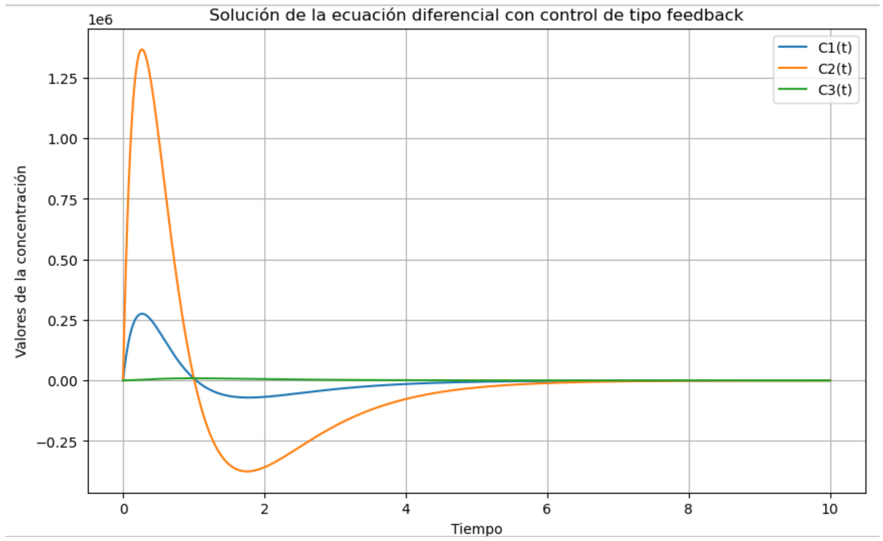

Figura 6: Sol. de la ec. dif. con control feedback.

Nota: Se respetan las condiciones iniciales del problema, pero debido a las magnitudes que toma la concentración de los tanques no se logra apreciar. Se encuentra en escala $\approx 10^7$, por lo que números del orden $\approx 10^3$ se ven muy cercanos al cero.

Observamos que el sistema es estabilizable usando *place*, notando además que $||x(t)|| \to 0$ cuando $t \to +\infty$. Recordar que $\bar{x} = 0$ es punto de equilibrio de la ecuación $\dot{x} = (A - BK)x$. Notamos además que toma valores muy altos a pesar de estabilizarse.

Veamos la construicción del estabilizador usando *lqr*. La función *lqr()* calcula el control óptimo de tipo feedback lineal $\vec{U} = -K\vec{X}$ que minimiza el costo cuadrático.

$$\min_{u(\cdot)} J(u) = \int_0^\infty (x^T Q x + u^T R u + 2x^T N u) dt$$

Para construir las matrices Q y R, que bajo la teoría deben ser definidas positivas, tomando $N = 0$ y juntando bien las dimensiones de las matrices a utilizar, escogemos

$$Q = I; R = 1$$

Código 9: Construcción del estabilizador usando lqr

```python
import control as ctrl
# Definimos matrices
A = np.matrix([[-4/200, 0, 0], [1/100, -6/100, 0], [2/300, 0.01, -5/300]]) #(3 x 3)
```

```
B = np.matrix([4/200, 10/100, 0]).T #(3 x 1)
Q = np.eye(3)
R = 2
K_lqr, S, E = ctrl.lqr(A, B, Q, R)
print("La matriz K=", K_lqr)
# Condiciones iniciales
X0 = np.array([5000, 0, 1000])
# Intervalo de tiempo
t_span = (0, 10000)
solucion_feedback = solve_ivp(system_dynamics_linear_feedback(K_lqr), t_span,
    ↪ X0, method='RK45', t_eval=np.linspace(0, 400, 10000)) #Feedback
grafica_solucion(solucion_feedback, "de tipo feedback")
```

Lo que nos retorna:

```
La matriz K= [[0.2208701  0.33782777 0.15650817]]
```

y el siguiente gráfico de la solución:

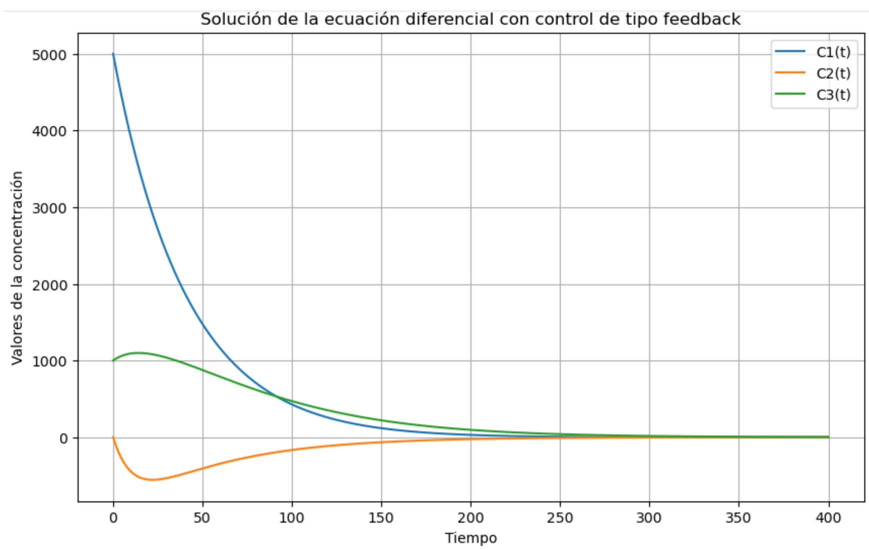

Figura 7: Sol. de la ec. dif. con control feedback.

Observamos que el sistema es estabilizable con la matriz entregada por el método *lqr*, sin embargo, es importante notar que el tiempo en que llega a cero el sistema, es para un $T > 0$ considerablente mayor (≈ 300) que para el tiempo con la matriz entregada por *place* ($\approx T = 6$).

Ahora verifiquemos que el sistema efectivamente es estabilizable para los K obtenidos con *place* y *lqr*, con el comando *eig*.

Código 10: Valores propios de A-BK con K dada por place

```python
# Definimos función que verifica si el sistema es estabilizable, esto es, A - BK es
    ↪ Hurwitz
def Hurwitz(M):
    eigvalues, eigvectors = np.linalg.eig(M)
    return eigvalues
M = A - B @ K
eig = Hurwitz(M)
print("Los valores propios de la matriz A-BK:", eig, ". Es Hurwitz!")
```

Lo que nos retorna:

```
Los valores propios de la matriz A-BK: [-4.00000008 -1.99999977 -1.00000015] . Es
    ↪ Hurwitz!
```

Además verificamos que los valores propios son aquellos que fueron impuestos para hacer la matriz Hurwitz.

Código 11: Valores propios de A-BK con K dada por lqr

```python
K_lqr, S, E = ctrl.lqr(A, B, Q, R)
N = A - B @ K_lqr
eig_lqr = Hurwitz(N)
print("Los valores propios de la matriz A-BK:", eig_lqr)
```

Lo que nos retorna:

```
Los valores propios de la matriz A-BK: [-0.09271624+0.j -0.0210753+0.00185212j
    ↪ -0.0210753-0.00185212j] . Es Hurwitz!
```

Verificamos que la matriz K es tal que $A - BK$ es Hurwitz, por lo que el sistema (A,B) es estabilizable.

3.2. Ejercicio 8

Construya el estimador de Luenberger asociado al sistema controlado y observado del ejercicio 5. Simule el estimador para otros tipos de observaciones $Y(\cdot)$ y controles $U(\cdot)$.

Solución:
Se construye el observador de Luenberger asociado al sistema controlado y observado.

$$\dot{X} = AX + BU$$

$$Y = CX$$

$$A = \begin{bmatrix} -\frac{4}{200} & 0 & 0 \\ \frac{1}{100} & -\frac{6}{100} & 0 \\ \frac{2}{300} & \frac{3}{300} & -\frac{5}{300} \end{bmatrix} \quad B = \begin{bmatrix} \frac{4}{200} \\ \frac{10}{100} \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Consideramos entonces el sistema

$$\dot{\hat{X}} = A\hat{X} + Bu + L(\hat{Y} - Y) \ ; \ \hat{X}(0) = \hat{X}_0 \text{ conocido}$$
$$\hat{Y} = C\hat{X}$$

Nos interesa escoger $L \in \mathbb{R}^{n\times p}$ de manera que la matriz $A + LC$ sea Hurwitz, de esta manera se tiene que el error del estimador $e_x = \hat{X} - X$ que sigue la dinámica

$$\dot{e_x}(t) = (A + LC)e_x(t) \ ; \ e_x(0) = \hat{X}_0 - X(0)$$

Sea tal que

$$e_x \to 0$$

Independiente del valor $X(0)$ para un tiempo suficientemente grande, se sabe de Álgebra Lineal que las matrices $A + LC$ y $(A + LC)^T = A^T + C^T L^T$ tienen mismos valores propios, pues transponer una matriz no altera estos, luego

$$A + LC \text{ es Hurwitz} \iff A^T + C^T L^T \text{ es Hurwitz}$$

Luego podemos usar el comando place para crear una matriz L^T adecuada usando el sistema de control transpuesto (A^T, C^T) y encontrar el L buscado transponiendo la matriz.

Recordar que para un sistema controlado (A, B), el comando *place* entrega K tal que $A - BK$ es Hurwitz, por lo tanto, consideramos el sistema controlado $(A^T, -C^T)$ de manera que el $K = L^T$ sea tal que

$$(A^T + C^T K) = (A^T + C^T L^T) \text{ es Hurwitz}$$

Código 12: Construcción de la matriz L

```
# Definir las matrices A y C
A = np.matrix([[-4/200, 0, 0], [1/100, -6/100, 0], [2/300, 0.01, -5/300]])
A_dual = A.T
C = np.matrix([[0, 1, 0], [0, 0, 1]])
C_dual = -C.T
# Construcción de L_dual - place (3 x 2)
poles = [-4, -1, -1]
L_dual = ctrl.place(A_dual, C_dual, poles)
L = L_dual.T
print("La matriz L calibrada=", L)
# Vemos si A+LC tiene dichos polos y verificar que es Hurwitz
```

```
O = A + L @ C
print('Se verifica que los polos de A + LC son:', Hurwitz(O), '. Es Hurwitz!')
```

Lo que retorna:

```
La matriz L calibrada= [[-270.02769231 -180.01846154]
 [  -3.69538462    -1.83692308]
 [  -1.84692308    -2.20794872]]
Se verifica que los polos de A + LC son: [-1. -4. -1.] . Es Hurwitz!
```

Consideramos entonces el sistema

$$\dot{\hat{X}} = A\hat{X} + BU + L(\hat{Y} - Y) \; ; \; \hat{X}(0) = \hat{X}_0 \text{ conocido}$$
$$\hat{Y} = C\hat{X}$$

Para el L recién encontrado, considerando distintas condiciones $\hat{X}_0$.

Código 13: Código para la dinámica del observador de Luenberger

```
from scipy import interpolate
from scipy.interpolate import krogh_interpolate
from scipy.interpolate import KroghInterpolator

# Creamos L que estabiliza el error
poles = [-3, -2, -3]
L_dual = ctrl.place(A_dual, C_dual, poles)
L = L_dual.T

# Interpolamos la solución X(t) para tener una función analítica, de la cual
    ↪ desprender la observación Y(t) = C X(t)
def Y_obsv_u_C(C, solucion):
    observed_ucos = solucion.y
    x_observed = solucion.t
    observed_interpolate = interpolate.interp1d(x_observed, observed_ucos) # X(t)
    def Y_obsv(t):
        return C @ observed_interpolate(t) # Y(t) = CX(t)
    return Y_obsv

# Definimos la dinámica para el observador de Luenberger
def system_obsv_u(u, C, Y, L):
    def system_obvs(t, hat_X):
        return np.dot(A, hat_X) + np.dot(B, u(t)) + L @ (C @ hat_X - Y(t)) #Y(t)
    ↪ = C X(t)
    return system_obvs
```

A continuación se muestran las soluciones de los sistemas de ecuaciones controlados (para distintos controles) vs el observador de Luenberger anteriormente encontrado.

Para el sistema con control de tipo bang-bang se obtuvo lo siguiente:

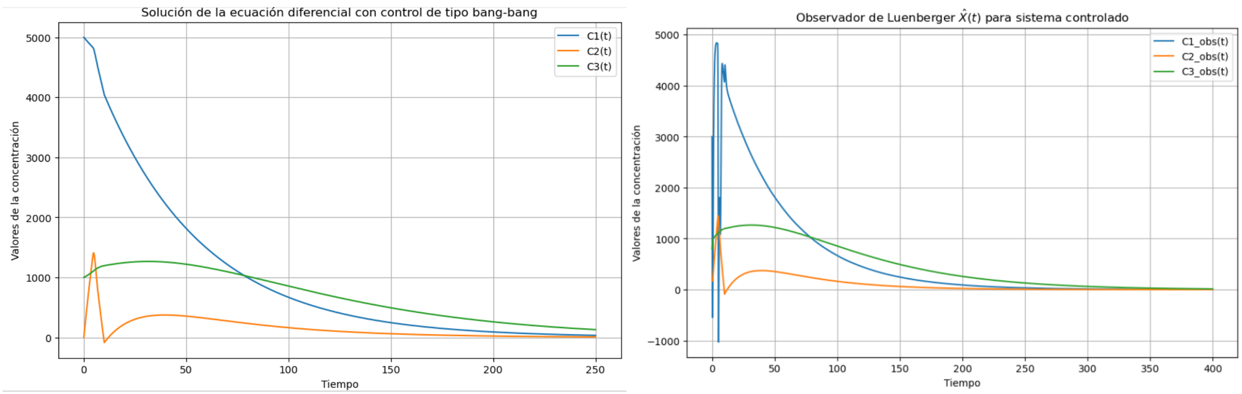

Figura 8: Original vs Observador de Luenberger caso bang-bang

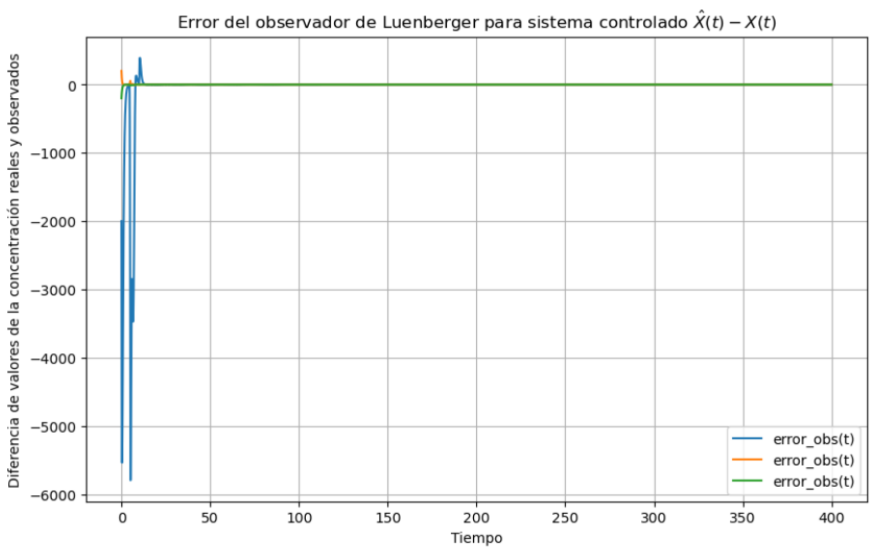

Figura 9: Error del observador de Luenberger caso bang-bang

Para el sistema con control nulo se obtuvo lo siguiente:

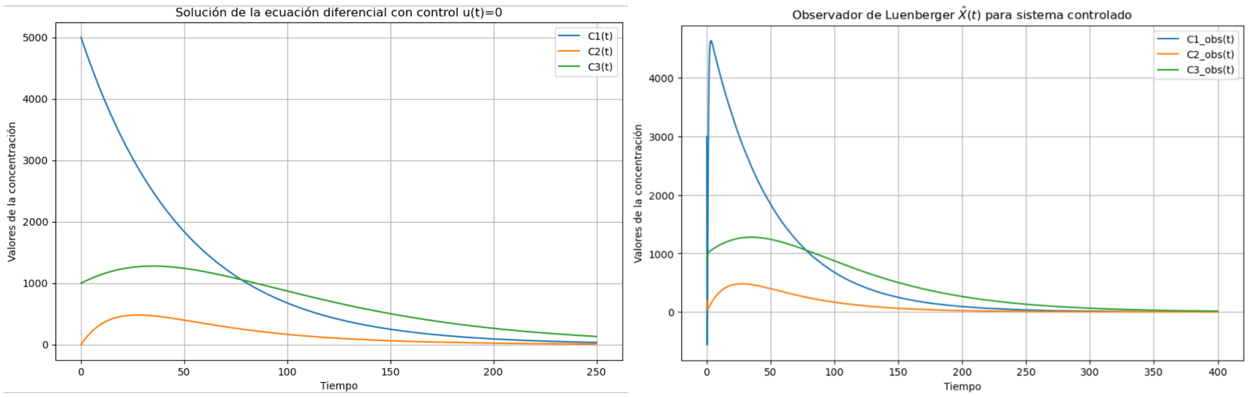

Figura 10: Original vs Observador de Luenberger caso nulo

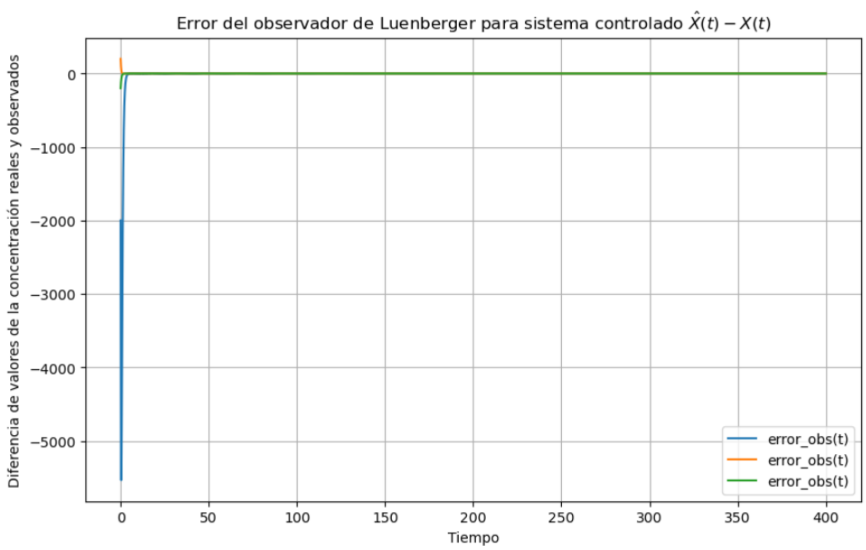

Figura 11: Error del observador de Luenberger caso nulo

Para el sistema con control constante se obtuvo lo siguiente:

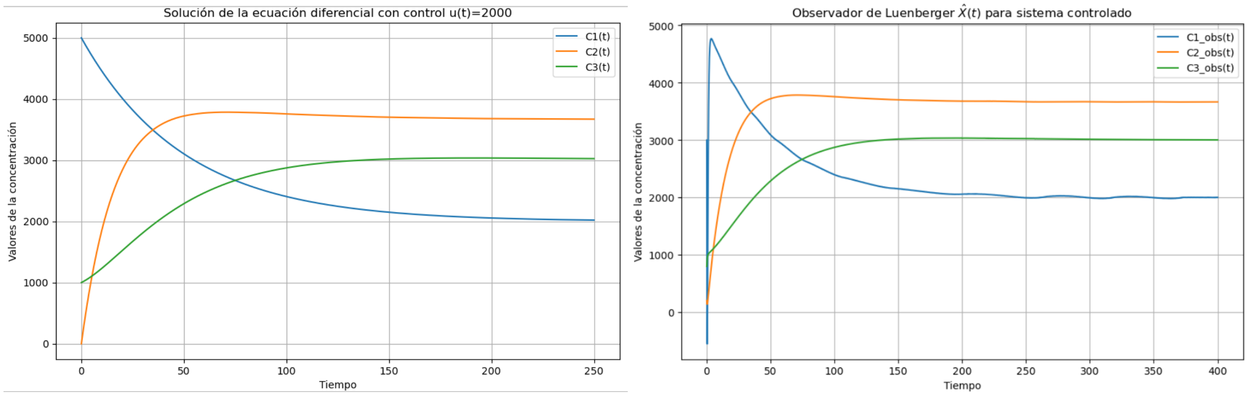

Figura 12: Original vs Observador de Luenberger caso constante

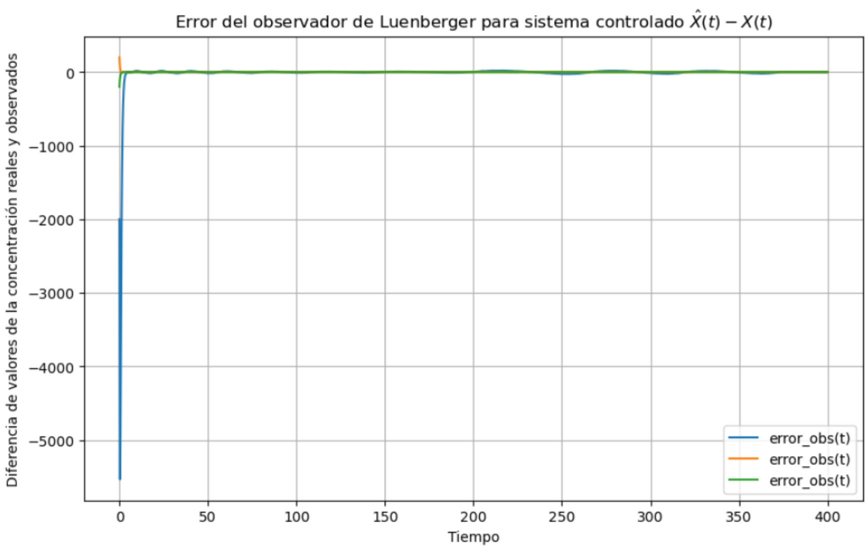

Figura 13: Error del observador de Luenberger caso constante

Para el sistema con control sinusoidal se obtuvo lo siguiente:

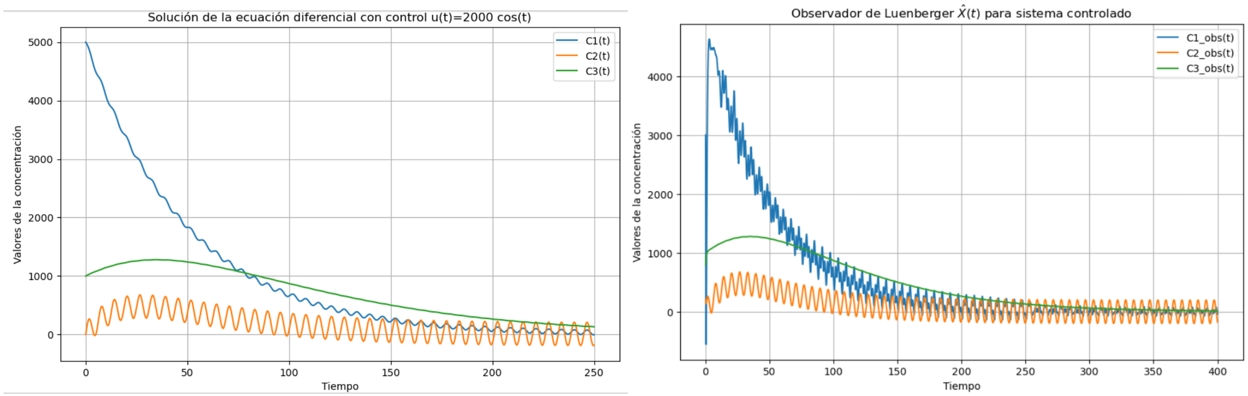

Figura 14: Original vs Observador de Luenberger caso sinusoidal

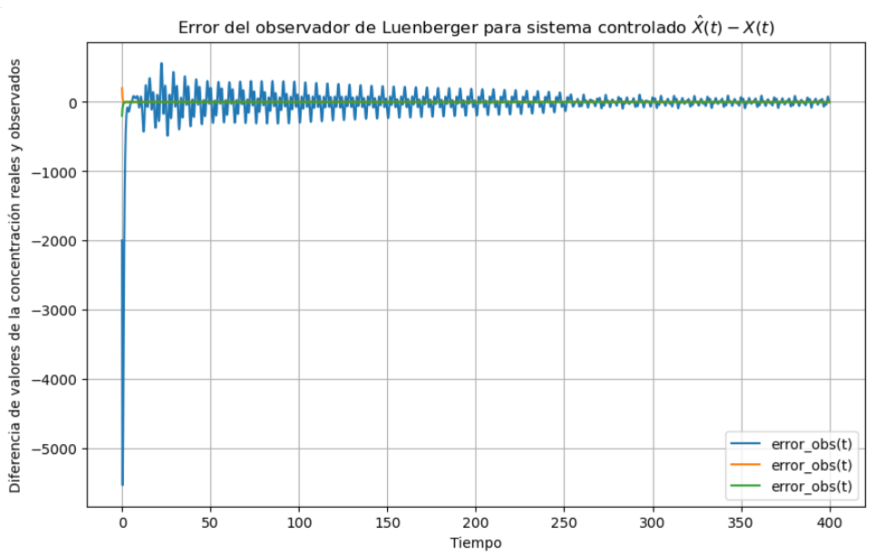

Figura 15: Error del observador de Luenberger caso sinusoidal

Notamos que en todos los casos las soluciones $\hat{X}$ que se obtienen apartir de la observación y la solución real X se acercan asintóticamente para $T > 0$ suficientemente largo. Probemos con otro tipo de observaciones $Y(\cdot)$.

Tomamos

$$Y(t) = \begin{bmatrix} C_2(t) - C_1(t) \\ 4C_2(t) + 2C_3(t) \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 4 & 2 \end{bmatrix} X$$

Primero construimos análogamente al anterior, un programa para este observador (disponible en el archivo .ipynb) obteniendo:

```
La matriz L calibrada= [[  56.42          22.078      ]
 [  53.75285714   21.40214286]
 [-110.30761905  -44.42238095]]
Se verifica que los polos de A + L C son: [-4. -1. -1.] . Es Hurwitz!
```

A continuación se muestran las soluciones de los sistemas de ecuaciones controlados (para distintos controles) vs este nuevo observador.

Para el sistema con control de tipo bang-bang se obtuvo lo siguiente:

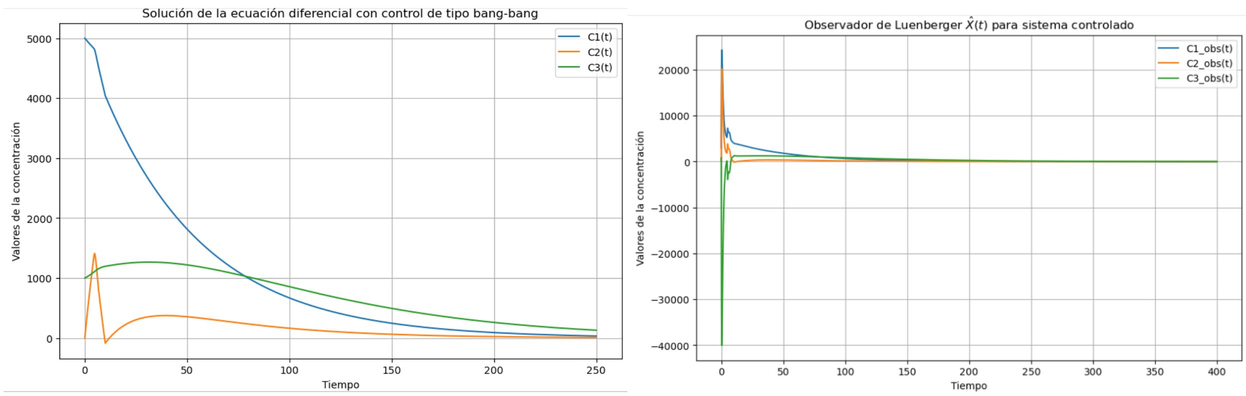

Figura 16: Original vs Nuevo observador de Luenberger caso bang-bang

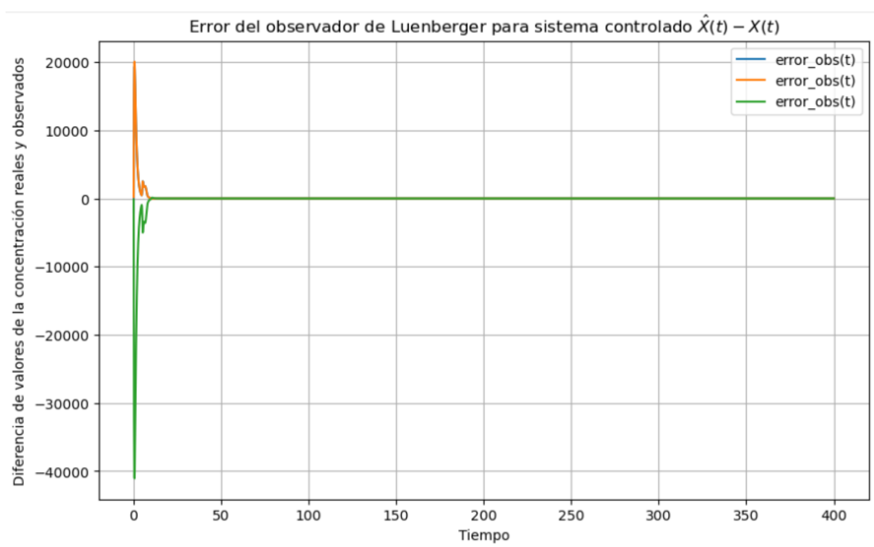

Figura 17: Error del nuevo observador de Luenberger caso bang-bang

Para el sistema con control sinusoidal se obtuvo lo siguiente:

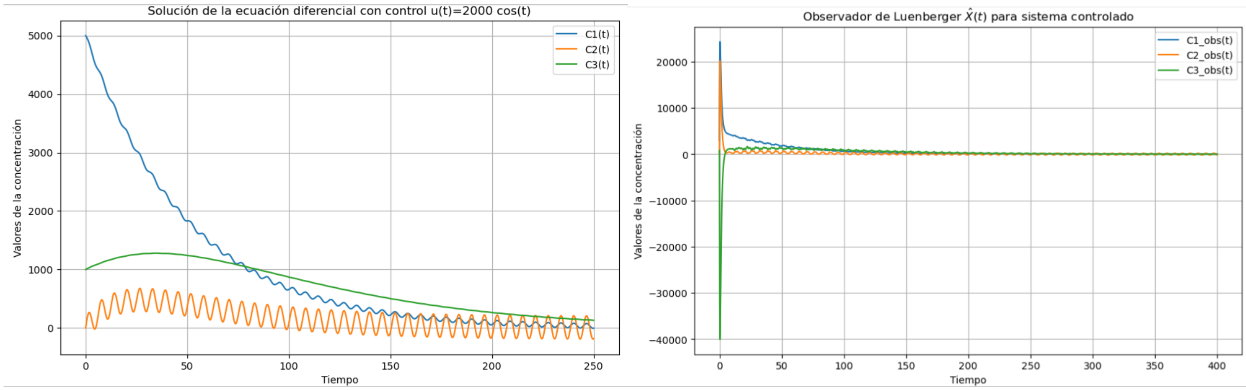

Figura 18: Original vs Nuevo observador de Luenberger caso sinusoidal

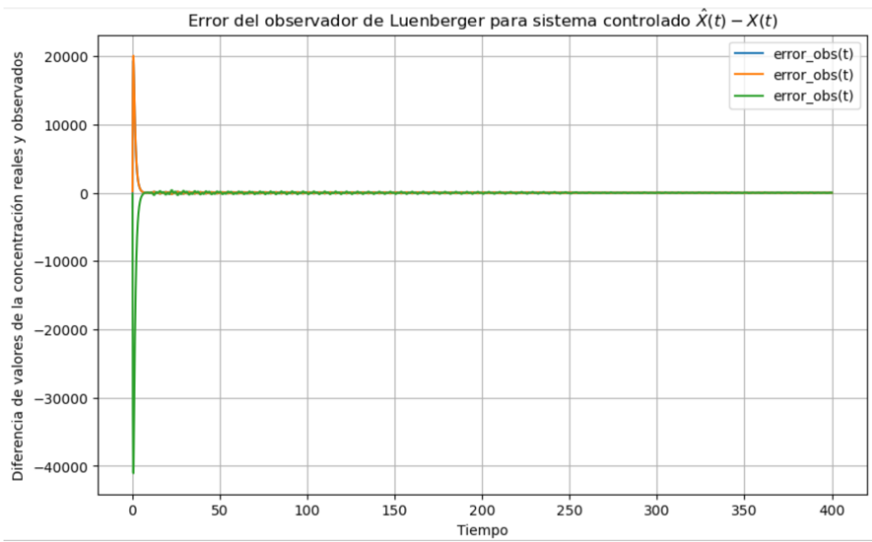

Figura 19: Error del nuevo observador de Luenberger caso sinusoidal

Se verifica que el observador de Luenberger sigue aproximando la solución para las observaciones

$$Y(t) = \begin{bmatrix} C_2(t) - C_1(t) \\ 4C_2(t) + 2C_3(t) \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 4 & 2 \end{bmatrix} X$$

Se aprecia además que para el error se toman diferencias más drásticas en los primeros tiempos de simulación y luego converge rápidamente a cero.

3.3. Ejercicio 9

Considere los sistemas lineales controlados y observados del ejercicio 5. A partir de lo aprendido en clases, construya un control estabilizador por feedback lineal a partir de la observación $\vec{Y}$. Simule lo obtenido y compare con las trayectorias del sistema original para distintas condiciones iniciales. Discuta los resultados numéricos obtenidos en base a la teoría vista en cátedra.

Solución:

Considerando los sistemas lineales controlados y observados del problema 5 se construye un control estabilizador por feedback lineal a partir de la observación $\vec{Y}$.

Tenemos las observaciones

$$Y(t) = \begin{bmatrix} C_2(t) \\ C_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} X$$

Consideramos el control estabilizador del sistema por feedback lineal de la forma

$$u(t) = KY(t) = KCX$$

donde $K \in \mathbb{R}^{1\times 2}$, dicho control entrega la dinámica

$$\dot{X}(t) = AX + BKCx = (A + BKC)X$$

El sistema anterior sabemos se estabiliza a cero si la matriz $A+BKC$ resulta ser Hurwitz, por lo que debemos tomar valores k_1 y k_2 de $K = [k_1, k_2]$ de tal manera que $A+BKC$ cumpla dicha propiedad, se buscan valores por inspección.

Código 14: Verificación de la inspección

```
# Definir las matrices A y B
A = np.matrix([[-4/200, 0, 0], [1/100, -6/100, 0], [2/300, 0.01, -5/300]])
B = np.matrix([4/200, 10/100, 0]).transpose()
C = np.matrix([[0, 1, 0], [0, 0, 1]])

k__1=-4
k__2=-5
K=np.matrix([k__1,k__2])

print('Valores propios de A +BKC:', Hurwitz(A + B @ (K @ C)))
print('¿'Re(\lambda__{{A+BKC}}) < 0 ?', (Hurwitz(A + B @ (K @ C)) < 0))
```

Lo que nos retorna:

```
Valores propios de A +BKC: [-0.02578699+0.0068186j -0.02578699-0.0068186j
```

```
    ↪ -0.44509268+0.j        ]
¿‘Re(\lambda_{{A+BKC}}) < 0 ? [ True  True  True]
```

Por lo que tenemos K que hace Hurwitz la matriz $A + BKC$ tomando $K = [-4, -5]$.

Finalmente modelamos el sistema con dicho control $u(t)$ y vemos si se estabiliza a partir de las observaciones Y para distintas condiciones iniciales.

Código 15: Programa para 3 condiciones iniciales distintas

```

# Condición inicial del problema
X0 = np.array([5000, 0, 1000]).T
X1 = np.array([2000, 400, 3000]).T
X2 = np.array([3000, 4000, 1000]).T

# Definimos la dinámica
def system_obsv_Y_feedback(K,C):
    def system_obsv_feedback(t,X):
        return np.dot(A + B @ (K @ C), X.T)
    return system_obsv_feedback

sol_obs_feedback_x0 = solve_ivp(system_obsv_Y_feedback(K,C), t_span, X0,
    ↪ method='RK45', t_eval=np.linspace(0, tf, discr))
sol_obs_feedback_x1 = solve_ivp(system_obsv_Y_feedback(K,C), t_span, X1,
    ↪ method='RK45', t_eval=np.linspace(0, tf, discr))
sol_obs_feedback_x2 = solve_ivp(system_obsv_Y_feedback(K,C), t_span, X2,
    ↪ method='RK45', t_eval=np.linspace(0, tf, discr))

grafica_sol_obvs(sol_obs_feedback_x0)
grafica_sol_obvs(sol_obs_feedback_x1)
grafica_sol_obvs(sol_obs_feedback_x2)
```

Lo que nos retorna los siguientes 3 gráficos:

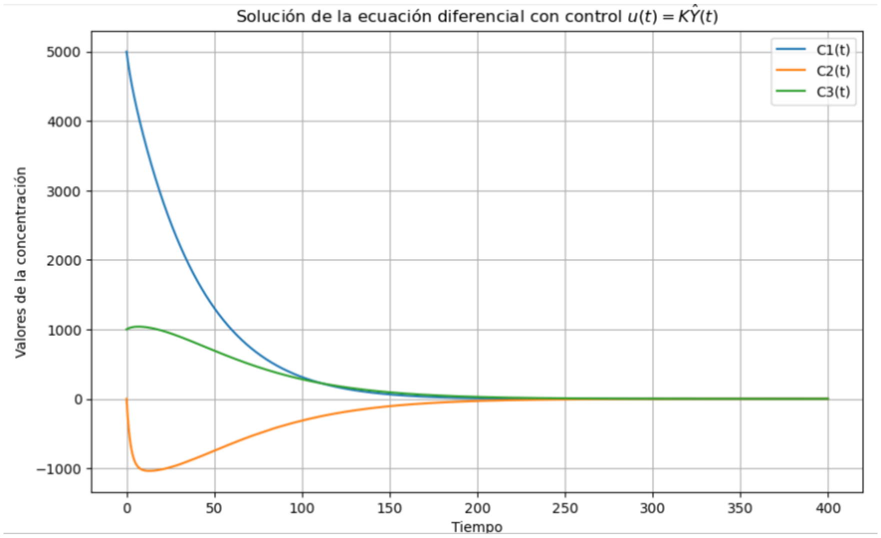

Figura 20: Solución del sistema con condición inicial X0

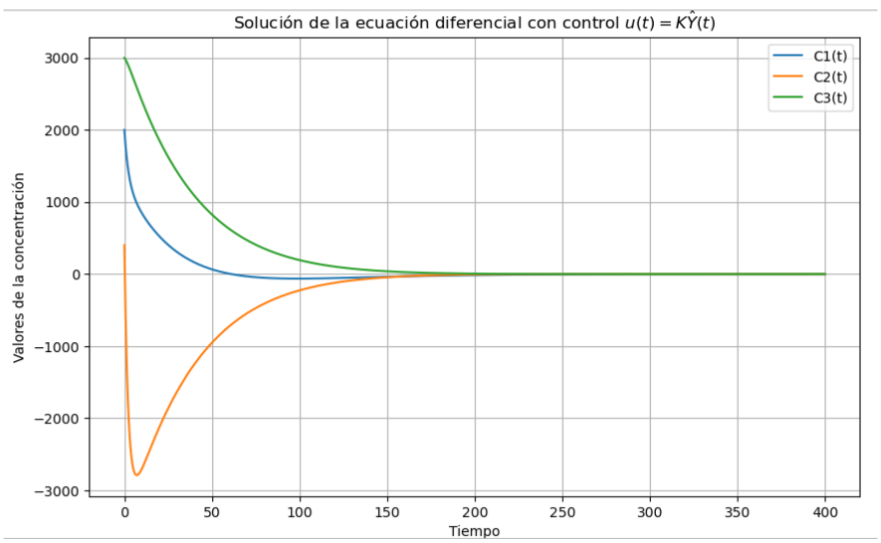

Figura 21: Solución del sistema con condición inicial X1

Y finalmente

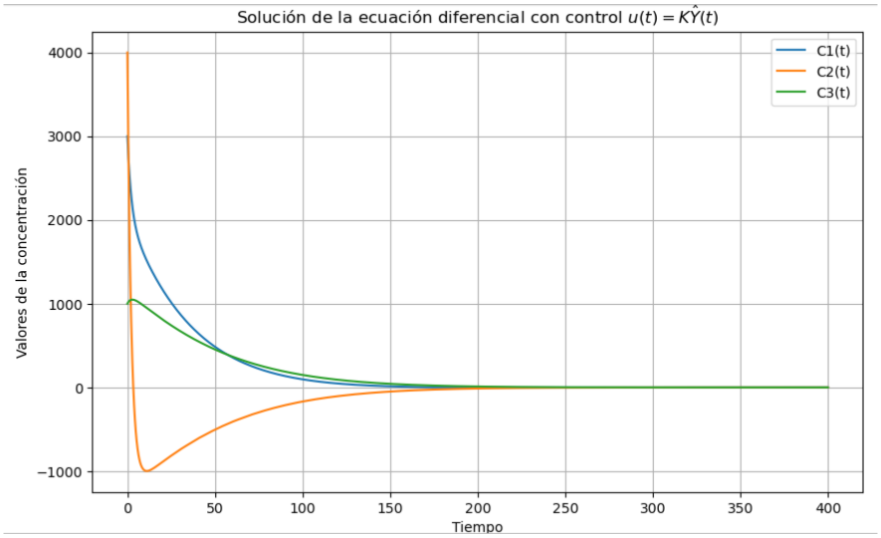

Figura 22: Solución del sistema con condición inicial X2

Verificamos que las soluciones para cada condición inicial se estabilizan en cero a partir del control por retroalimentación de las observaciones $u(t) = KY(t)$.