Universidad Nacional de Córdoba
Facultad de Ciencias Exactas, Físicas y Naturales
Departamento de Ingeniería Electrónica

Informe Actividad Práctica Nº2

Trabajo Práctico Sistemas de Control II

Autor: Villar, Federico Ignacio
Profesor: Pucheta, Julián Antonio
Fecha de entrega: 27 de mayo de 2023
Córdoba, Argentina

Resumen

En el siguiente documento se detalla la resolución de la segunda actividad práctica propuesta por el profesor Julián Pucheta en la materia de Sistemas de Control II. Se desarrollarán los procedimientos utilizados, así como también se adjuntarán los diferentes códigos de Python, Julia y Matlab utilizados para la compleción de la actividad.

Al final del informe, se añaden algunos enlaces de utilidad para la investigación de fuentes y futuras lecturas de este informe como bibliografía. Se trata de que las conclusiones sean cortas, concisas y representativas del trabajo realizado.

Todas las gráficas de funciones aduntas en el documento son de elaboración propia haciendo uso de código en los lenguajes antes mencionados. Cabe destacar que las herramientas de software utilizadas fueron:

- Visual Studio Code
- Matlab
- Scilab
- Github Desktop

Todo este trabajo se encuentra publicado en un repositorio de Github y mantenido por el alumno autor del informe. Se adjunta junto con los links anteriormente mencionados el enlace del repositorio online. Además, en la entrega se adjuntan los códigos utilizados en formatos: .m, .ipynb, .mlx, .py, junto con los archivos de Live Script exportados en formado .pdf.

Índice de Contenidos

Índice de Figuras

Índice de Códigos

1. Consigna

1.1. Caso de estudio 1. Sistema de tres variables de estado

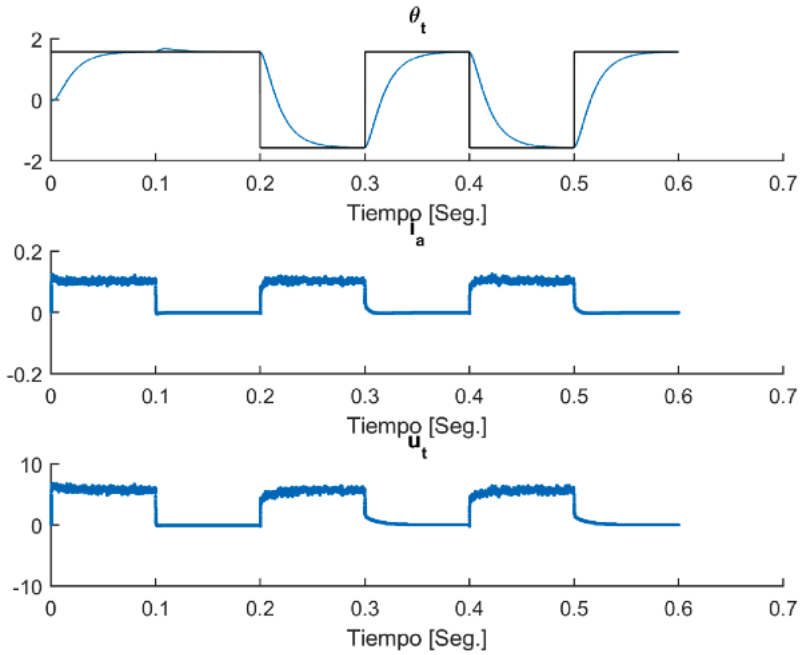

Figura 1: Evolución del ángulo cuando el controlador en variables de estado tiene perturbaciones en su operación.

Dadas las ecuaciones del motor de corriente continua con torque de carga T_L no nulo, con los parámetros:

- $L_{AA} = 5\ 10^{-3}$
- $J = 0.004$
- $R_A = 0.2$
- $B = 0.005$
- $K_i = 6.5\ 10^{-5}$
- $K_m = 0.055$

$$\frac{di_a}{dt} = -\frac{R_A}{L_{AA}} i_a - \frac{K_m}{L_{AA}} \omega_r + \frac{1}{L_{AA}} v_a$$

$$\frac{d\omega_r}{dt} = \frac{K_i}{J} i_a - \frac{B_m}{J} \omega_r - \frac{1}{J} T_L$$

$$\frac{d\theta_t}{dt} = \omega_r$$

1. Implementar un sistema en variables de estado que controle el ángulo del motor, para consignas de $\pi/2$ y $-\pi/2$ cambiando cada 2 segundos y que el de T_L de 1.15 10^{-3} aparece sólo para $\pi/2$, para $-\pi/2$ es nulo. Hallar el valor de integración Euler adecuado. El objetivo es mejorar la dinámica del controlador que muestra la figura 1.

2. Considerar que no puede medirse la corriente y sólo pueda medirse el ángulo, por lo que debe implementarse un observador. Obtener la simulación en las mismas condiciones que en el punto anterior, y superponer las gráficas para comparar.

1.2. Caso de estudio 2. Sistema no lineal de cuatro variables de estado

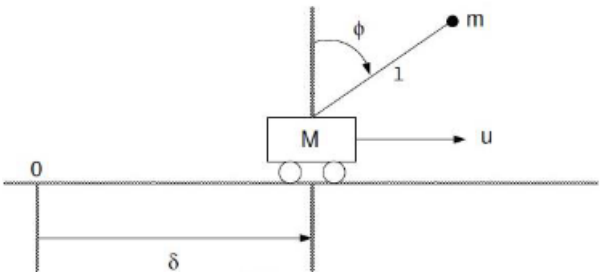

Figura 2: Sistemas para modelar, extraído de "Sontag. Mathematical Control Theory. 1998. Pag. 104."http://www.sontaglab.org.

Para el caso del esquema del péndulo invertido de la figura 2 donde el modelo es:

$$(M+m)\ddot{\delta} + ml\ddot{\phi}cos(\phi) - mlp\dot{h}i^2 sin(\phi) + F\dot{\delta} = u$$

$$l\ddot{\phi} - gsin(\phi) + \ddot{\delta}cos(\phi) = 0$$

Con las variables de estado $x = \begin{bmatrix}\delta & \dot{\delta} & \phi & \dot{\phi}\end{bmatrix}^T$, y los valores de los coeficientes de:

- $m = 0.1$
- $F = 0.1$
- $g = 9.8$
- $M = 1.5$

Determinar Δt y el tiempo de simulación adecuados.

1. Calcular un controlador que haga evolucionar al péndulo en el equilibrio inestable, partiendo de una condición inicial nula en el desplazamiento y termine en -10 metros manteniendo la vertical. Determinar el ángulo máximo que puede alejarse de la vertical en $t = 0$ para que el sistema cumpla el objetivo de control.

2. Incorporar un observador para el caso en que sólo puedan medirse el desplazamiento δ y el ángulo ϕ, repetir las simulaciones para las condiciones anteriores y graficar los resultados en gráficas superpuestas.

3. Calcular un controlador que haga evolucionar al péndulo en el equilibrio estable, partiendo de una condición inicial nula en el desplazamiento y el ángulo en π que termine en 2 metros evitando las oscilaciones de la masa m, considerando que es una grúa. Una vez que que $\delta = 2$ modificar a m a un valor 10 veces mayor y volver al origen evitando oscilaciones.

4. Incorporar un observador para el caso en que sólo puedan medirse el desplazamiento δ y el ángulo ϕ, repetir las simulaciones para las condiciones anteriores y graficar los resultados en las gráficas superpuestas para el equilibrio estable.

2. Resolución del caso 1

Partiendo de las ecuaciones diferenciales que modelan al motor de corriente continua, para comenzar la simulación y diseño del controlador es necesario conocer la representación matricial en variables de estado del motor. Para ello, en un principio de se realizó un despeje partiendo de las ecuaciones de dato. Esto llevó a la siguiente representación:

$$A = \begin{bmatrix} -\frac{R_A}{L_{AA}} & -\frac{K_m}{L_{AA}} & 0 \\ \frac{K_i}{J} & -\frac{B_m}{J} & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{L_{AA}} & 0 \\ 0 & -\frac{1}{J} \\ 0 & 0 \end{bmatrix}$$

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

$$D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Ahora, reemplazando con los valores de las constantes proporcionadas por las consignas, se obtienen las siguientes matrices que representan al sistema:

$$A = \begin{bmatrix} -40 & -11 & 0 \\ 0.0162 & -1.25 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 200 & 0 \\ 0 & -250 \\ 0 & 0 \end{bmatrix}$$

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

$$D = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Luego de tener estas matrices declaradas en Matlab, se obtuvo la representación en espacio de estados y luego las funciones de transferencia que modelaban al sistema, todo esto haciendo uso de las siguientes dos líneas de código.

Código 1: Espacio de estados, y funciones de transferencia para el motor.

```
sys = ss(A,B,C,D)
Gtheta = tf(sys)
```

Para esta representación en variables de estado planteadas, el vector x propiamente dicho es:

$$x = \begin{bmatrix} i_a \\ \omega \\ \theta \end{bmatrix}^T$$

Con las primeras definiciones dadas, se parte al diseño del controlador para la referencia dada.

2.1. Diseño del controlador solicitado

2.1.1. Primera aproximación

Para cumplir con las condiciones que pone la consigna, en un principio se planteó el desarrollo de un controlador por el método de asignación de polos.

Los polos se ubicaron décadas por debajo de los obtenidos con el modelo del motor, para de esa forma mejorar el tiempo de establecimiento del sistema. Al llegar el momento de la simulación, se puedo ver una notoria mejora en la disminución de los transitorios, por lo que a priori, la consigna estaba cumplida, pero cuando se analizó la evolución de la corriente de armadura, se pudo apreciar que esta estaba en el orden de magnitud de 10^7, algo que claramente indicaba que el control no era muy eficiente, ya que se requería de mucha corriente para lograr el objetivo del setpoint.

Luego, para disminuir la corriente, se planteó una frecuencia de cambio de referencia menor, esta vez cada 10 segundos. Con este cambio disminuyó la magnitud de la corriente, pero siguió siendo alta. Se muestra a continuación la evolución de la corriente y el ángulo θ del eje del motor.

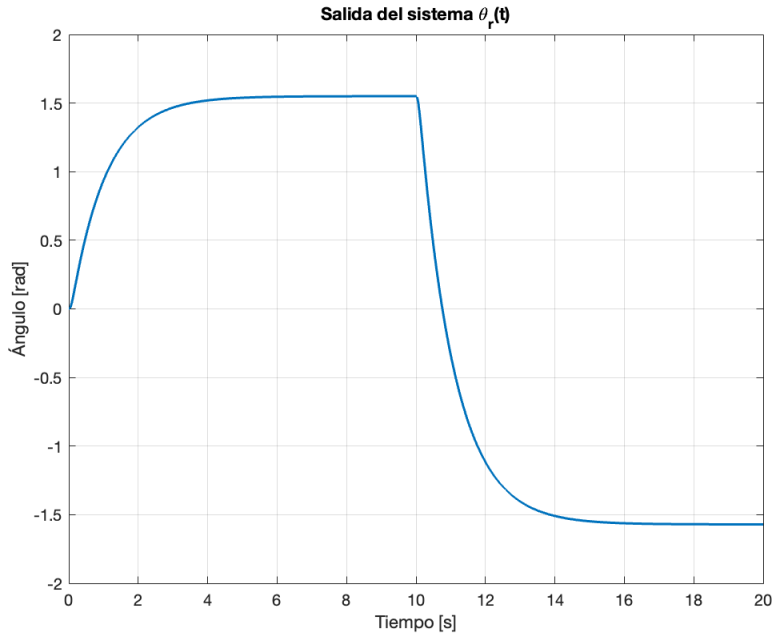

Figura 3: Dinámica del ángulo del sistema para el método de asignación de polos.

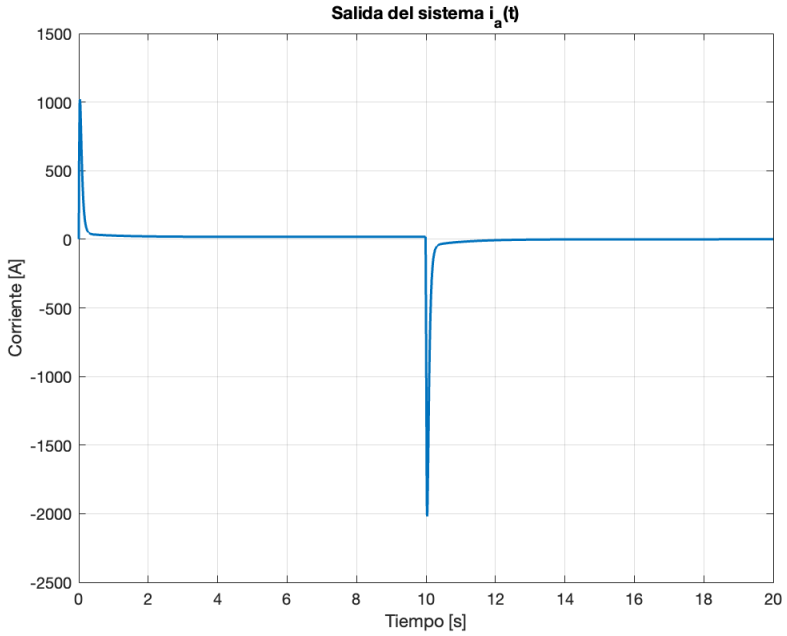

Figura 4: Dinámica de la corriente del sistema para el método de asignación de polos.

En la figura del ángulo no parece apreciarse problema alguno, pero, al momento de analizar la corriente de armadura en el sistema pueden apreciarse picos de 1000 A y -200 A. Estos valores no

son adecuados para el motor, ya que este debería contar con protecciones y componentes sobredimensionados en el circuito.

2.1.2. Diseño de controlador LQR

Para evitar los problemas anteriormente vistos, se plantea la posibilidad de controlar el sistema haciendo uso del método de LQR y una conmutación de la referencia a los 25 segundos.

Para el diseño del LQR es necesario declarar la matriz Q, en donde se colocan los pesos que se le asignan a cada variable. En este caso, la última variable cuenta con un peso considerablemente superior a las demás, por el hecho de que se pretende controlar el ángulo del eje del motor. La matriz resultante es:

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 100000 \end{bmatrix}$$

Y, el valor de R es 100. Ahora, para obtener la matriz K del controlador es necesario declarar matrices ampliadas del sistema, que son las siguientes:

$$A_{amp} = \begin{bmatrix} -40 & -11 & 0 & 0 \\ 0.0162 & -1.25 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$$B_{amp} = \begin{bmatrix} 200 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Luego, mediante el siguiente comando de Matlab se obtiene la matriz del controlador K.

Código 2: Obtención del controlador LQR para el ángulo del motor de corriente continua.

```
K = lqr(Aamp,Bamp,Q,R)
```

Se obtiene:

$$K = \begin{bmatrix} 0.0116 & 23.2442 & 50.0305 & -31.6228 \end{bmatrix}$$

Para simular el sistema mediante una integración por Euler, es necesario tener un paso definido. Este paso se lo obtiene analizando la dinámica del sistema. Se obtienen los polos del sistema a lazo cerrado, y se busca la respuesta más rápida, ahora, con el tiempo de establecimiento para esa respuesta, llamado t_r, se obtiene un paso de simulación de Euler que debe ser menor a $t_r = 0.0013$. En este caso se decide utilizar $h = 0.0001$. Recordando lo mencionado anteriormente, el cambio de referencia a los 25 segundos, entonces se simula por 50 segundos.

Con estos últimos 2 valores se declara un vector de tiempo, y luego, las condiciones iniciales del sistema se plantean haciendo uso de vectores que inician en 0 en su primer valor, para luego,

haciendo uso del método de integración de Euler ir añadiéndole valores que responden a la dinámica del sistema planteado. Las condiciones iniciales que se plantean son:

Código 3: Condiciones iniciales para la simulación del motor.

```
h = 1e-4;
simTime = 50;
t = 0:h:(simTime-h);
ia(1) = 0;
theta(1) = 0;
omega(1) = 0;
stateVector = [ia(1) omega(1) theta(1)]';
xop = [0 0 0]';
x = [ia(1) omega(1) theta(1)];
zeta(1) = 0;
integ(1) = zeta(1);
```

Con los valores anteriores declarados, se procede a realizar la simulación:

Código 4: Integración de Euler para la simulación del motor controlado

```
for i = 1:(simTime/h)
  zetaP = reference(i)-C*stateVector;
  zeta(i) = integ+zetaP*h;
  u(i) = -K(1:3)*stateVector-K(4)*zeta(i);
  ia(i) = x(1);
  omega(i) = x(2);
  theta(i) = x(3);
  x1P = -Ra*x(1)/Laa-Km*x(2)/Laa+u(i)/Laa;
  x2P = Ki*x(1)/J-Bm*x(2)/J-torque(i)/J;
  x3P = x(2);
  xP = [x1P x2P x3P]';
  x = x+h*xP;
  stateVector = [ia(i) omega(i) theta(i)]';
  integ = zeta(i);
  end
save('iaReal.mat','ia');
```

La simulación arroja los siguientes resultados:

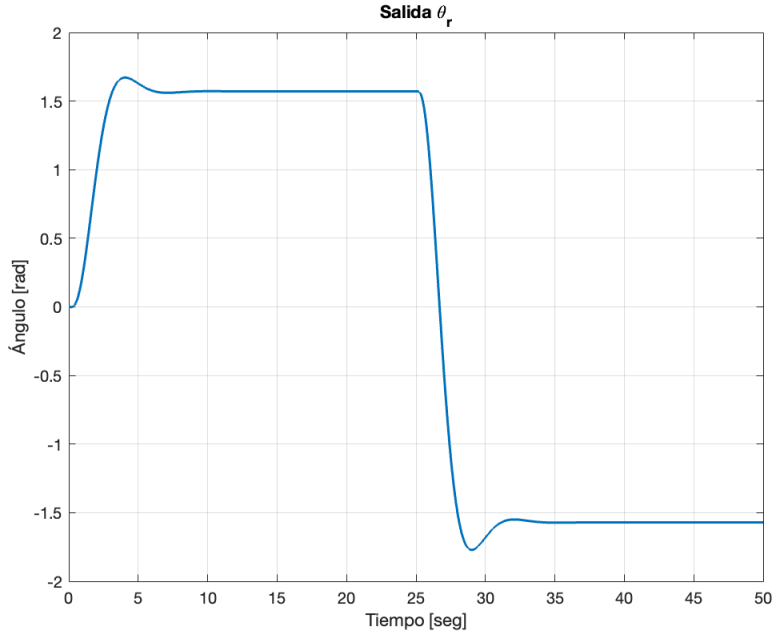

Figura 5: Evolución del ángulo con controlador LQR.

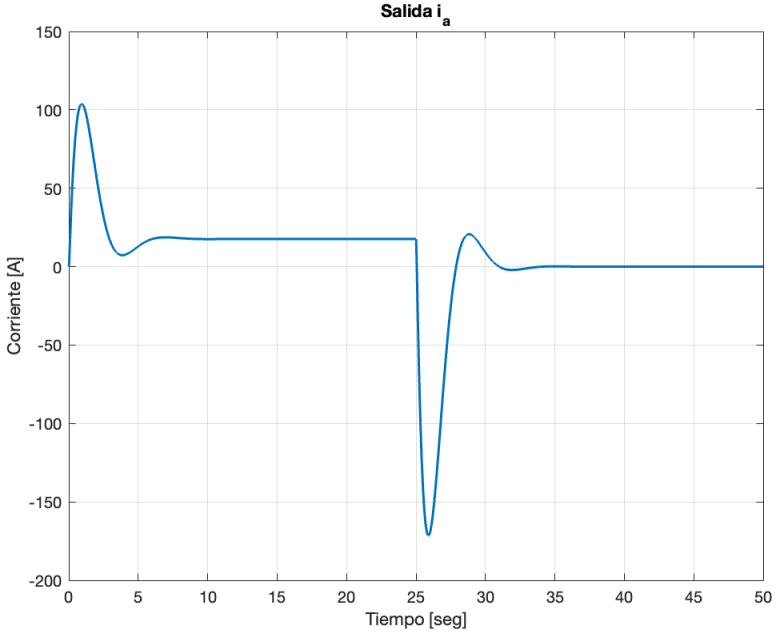

Figura 6: Evolución de la corriente de armadura con controlador LQR.

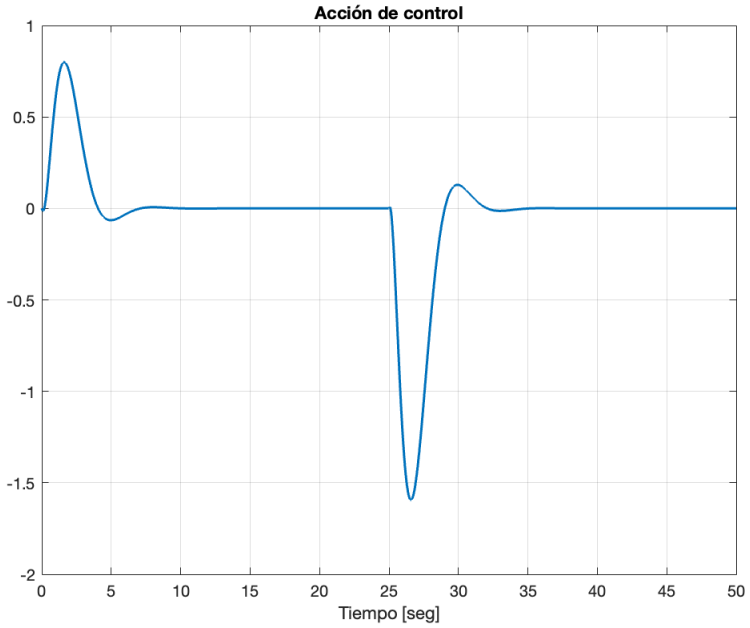

Figura 7: Acción de control LQR.

Puede apreciarse ahora que los valores de la corriente son más coherentes que los obtenidos anteriormente. Si bien a priori parecen elevados, hay que tener en cuenta que los parámetros que se tienen como dato sugieren que el motor es de dimensiones considerables, no es similar a los planteados anteriormente.

2.1.3. Observador

En la consigna se pide la implementación de un observador para el caso en que únicamente pueda medirse el ángulo de salida del motor. Para ello, se plantean los mismos controladores que se vieron anteriormente, pero para el caso del observador, se modo de observar (valga la redundancia) la diferencia entre el modelo real y el observado para una misma situación de controlabilidad. El código que declara las variables necesarias es el mismo que se planteó anteriormente. Se adjunta a continuación:

Código 5: Declaración de variables para el observador.

```
Laa = 5e-3;
J = 0.004;
Ra = 0.2;
Bm = 0.005;
Ki = 6.5e-5;
Km = 0.055;
A = [-Ra/Laa -Km/Laa 0; Ki/J -Bm/J 0; 0 1 0];
B = [1/Laa 0 0]';
C = [0 0 1];
D = [0 0];
Q = diag([0.1 0.1 0.1 100000]);
R = 100;
```

```
Aamp = [A zeros(3,1); -C 0];
Bamp = [B(:,1); 0];
Camp = [C 0];
K = lqr(Aamp,Bamp,Q,R);
h = 1e-4;
simTime = 50;
t = 0:h:(simTime-h);
reference = (pi/2)*square(2*pi*(1/50)*t);
torque = ((1.15e-3)/2)*square(2*pi*(1/50)*t)+((1.15e-3)/2);
ia(1) = 0;
theta(1) = 0;
omega(1) = 0;
stateVector = [ia(1) omega(1) theta(1)]';
xop = [0 0 0]';
x = [ia(1) omega(1) theta(1)]';
zeta(1) = 0;
integ(1) = zeta(1);
```

Hasta este punto es todo similar, pero, para el cálculo del controlador LQR en este sistema no se trabaja con las matrices iniciales, sino que es necesario realizar un par de cambios. Se muestra a continuación el código, y luego las matrices obtenidas.

Código 6: Obtención del observador para el motor.

```
Ao = A';
Bo = C';
Qo = diag([1 0.01 0.01]);
Ro = 10;
Ko = lqr(Ao,Bo,Qo,Ro);
```

Las matrices:

$$A_o = \begin{bmatrix} -40 & 0.0162 & 0 \\ -11 & -1.25 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B_o = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$Q_o = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$$

$$K_o = \begin{bmatrix} -0.0001 & 0.0003 & 0.0402 \end{bmatrix}$$

Con las matrices declaradas, para simular se utiliza el siguiente código.

Código 7: Simulación del observador para el motor CC.

```
obsStateVector = [ia(1) omega(1) theta(1)]';
```

```
  xObs = [0 0 0]';
  for i = 1:(simTime/h)
    zetaP = reference(i)-Camp(1:3)*stateVector-Camp(4)*integ;
    zeta(i) = integ+zetaP*h;
    u(i) = -K(1:3)*obsStateVector-K(4)*zeta(i);
    ia(i) = x(1);
    omega(i) = x(2);
    theta(i) = x(3);
    x1P = -Ra*x(1)/Laa-Km*x(2)/Laa+u(i)/Laa;
    x2P = Ki*x(1)/J-Bm*x(2)/J-torque(i)/J;
    x3P = x(2);
    xP = [x1P x2P x3P]';
    x = x+xP*h;
    iaO(i)= xObs(1);
    omegaO(i)= xObs(2);
    thetaO(i)= xObs(3);
    yO(i) = C*obsStateVector;
    y(i) = Camp(1:3)*stateVector+Camp(4)*integ;
    xTP = A*xObs+B*u(i)+Ko*(y(:,i)-yO(:,i));
    xObs = xObs+xTP*h;
    stateVector = [ia(i) omega(i) theta(i)]';
    integ = zeta(i);
    obsStateVector =[iaO(i) omegaO(i) thetaO(i)]';
  end
```

El objetivo del uso del observador era la medición de la corriente, ya que únicamente se tenía acceso al ángulo del motor. La simulación realizada se comparó con la señal real obtenida anteriormente. Se adjunta el resultado a continuación:

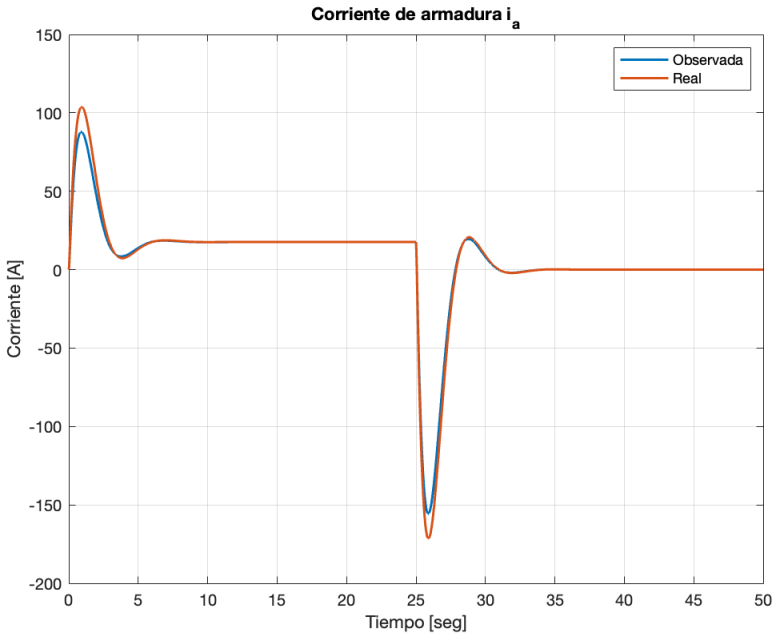

Figura 8: Comparación de corriente real y observada.

El resultado sugiere una aproximación que tiene sentido. Puede verse que el máximo error apreciado es de alrededor de 10 %. Y, como en ingeniería se suele tomar a 10 % como un valor mucho más pequeño que otro, se lo toma como válido.

2.2. Sintonización de controladores

2.2.1. Planteo del LQR

Para la sintonización de los controladores LQR aplicados en la resolución del primer caso de la consigna de la actividad práctica se trabajó con los pesos en la matriz Q. La matriz esta permite ponderar los costos a minimizar, es por ello, que, como se nombró anteriormente, se le asignó un mayor peso a las variables que se pretende controlar. El uso de este tipo de controladores permite tener un mayor manejo de la acción de control. Esto es algo sobre lo que no tiene todo el control el uso del método de asignación de polos.

Si se plantea en el sentido estricto el procedimiento, aquí no se trabajó de forma directa con la matriz P, ni se obtuvo de forma analítica la solución de la ecuación de Ricati en tiempo continuo. Todo esto se simplifica en programación simplemente haciendo uso de la función **lqr()**. La bibliografía de la cátedra ya predice esto diciendo que: "la solución de la ecuación matricial de Ricati se encuentra numéricamente por cómputo matricial, teniendo como dato los valores numéricos de A, B, R y Q", siendo esto el uso del algoritmo de Schur.

Para el diseño del controlador óptimo cuadrático, una vez formulado el problema, se debe resolver la ecuación reducida de Ricati con respecto a P verificando que $A - BK$ sea estable.

2.2.2. Planteo del observador

Para plantear el observador en el motor, cabe destacar que la matriz diagonal del mismo, es de dimensión $3x3$, a diferencia de la del controlador para el sistema real, que es de $4x4$. Esto viene

dado por la limitación que tiene el sistema en la medición de la corriente de armadura. Siguiendo esta limitación es que también se modifican los pesos de la matriz Q_o. La precisión del observador depende de los valores de sintonización utilizados. En el planteo en este ejercicio se realizaron de forma manual, con algunas correcciones hechas mediante pruebas de simulación. Esto permitió llegar a una performance aceptable, pero no necesariamente es la óptima.

3. Resolución del caso 2

3.1. Aproximación Lineal

El caso del péndulo invertido se trata de un sistema no lineal. Para poder implementar y simular, lo ideal es hacerlo mediante un análisis directo de las ecuaciones diferenciales que modelan su comportamiento. Sin embargo, como primera aproximación, se linealizaron las experiones para ángulo pequeño, esto es, planteando lo siguiente:

$$cos(\phi) = 1, \ sin(\phi) = \phi$$

De esa forma se logra la siguiente representación en matrices para el espacio de estados:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{F}{M} & -\frac{mg}{M}0 & \\ 0 & 0 & 0 & 1 \\ 0 & \frac{F}{Ml} & \frac{g(m+M)}{Ml} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \frac{l}{M} \\ 0 \\ \frac{l}{Ml} \end{bmatrix}$$

Con las matrices declaradas, se puede realizar una simulación en donde, con los parámetros del sistema se puede llevar al carro de una posición inicial a una referencia de $-10\ m$. El siguiente script de Python realiza la simulación lineal.

Código 8: Simulación lineal del péndulo invertido.

```
# Librerías necesarias
import numpy as np
from control.matlab import *
import matplotlib.pyplot as plt
# Variables del sistema
m = 0.1
F = 0.1
l = 1
g = 9.8
M = 1.5
# Matrices
A = np.array([[0, 1, 0, 0],
              [0, -F/M, -m*g/M, 0],
              [0, 0, 0, 1],
              [0, F/(l*M), g*(M+m)/(l*M), 0]])
B = np.array([[0],
              [1/M],
              [0],
              [-1/(l*M)]])
C = np.array([[1, 0, 0, 0],
              [0, 0, 1, 0]])
```

```
D = np.array([[0],
              [0]])
# Creación del modelo en espacio de estados
olSystem = ss(A, B, C, D)
# Polos a lazo abierto
olPoles = np.linalg.eigvals(A)
# Matriz de controlabilidad
M = ctrb(A, B)
# Variables para LQR
Q = np.dot(C.T, C)
R = 1
K, _, _ = lqr(A, B, Q, R)
# Nuevas matrices a implementar
Ac = A - np.dot(B, K)
Bc = B
Cc = C
Dc = D
# Sistema a lazo cerrado
clSys = ss(Ac, Bc, Cc, Dc)
# Tiempo de integración y simulación
h = 1e-4
simTime = 50
t = np.arange(0, simTime, h)
# Referencia del sistema
r = 10 * np.ones_like(t)
# Simulación
y, t, x = lsim(clSys, r, t)
delta = y[:, 0]
phi = y[:, 1]
# Gráfica
fig, ax = plt.subplots()
ax.plot(t, delta, label='Posición del carro [m]')
ax.set_xlabel('Tiempo [seg]')
ax.set_ylabel('Posición del carro [m]')
ax2 = ax.twinx()
ax2.plot(t, phi, 'r', label='Ángulo del péndulo [rad]')
ax2.set_ylabel('Ángulo del péndulo [rad]')
plt.title('Respuesta a entrada -10m con LQR')
plt.grid(True)
plt.show()
```

El código anterior arroja la siguiente gráfica, en donde el trazo rojo es el ángulo del péndulo $\phi(t)$, cuya referencia en el eje Y se aprecia a la derecha. Y, por la izquierda se tiene la referencia en Y de la posición $\delta(t)$ del carro del péndulo, gráfica que tiene el trazo en azul.

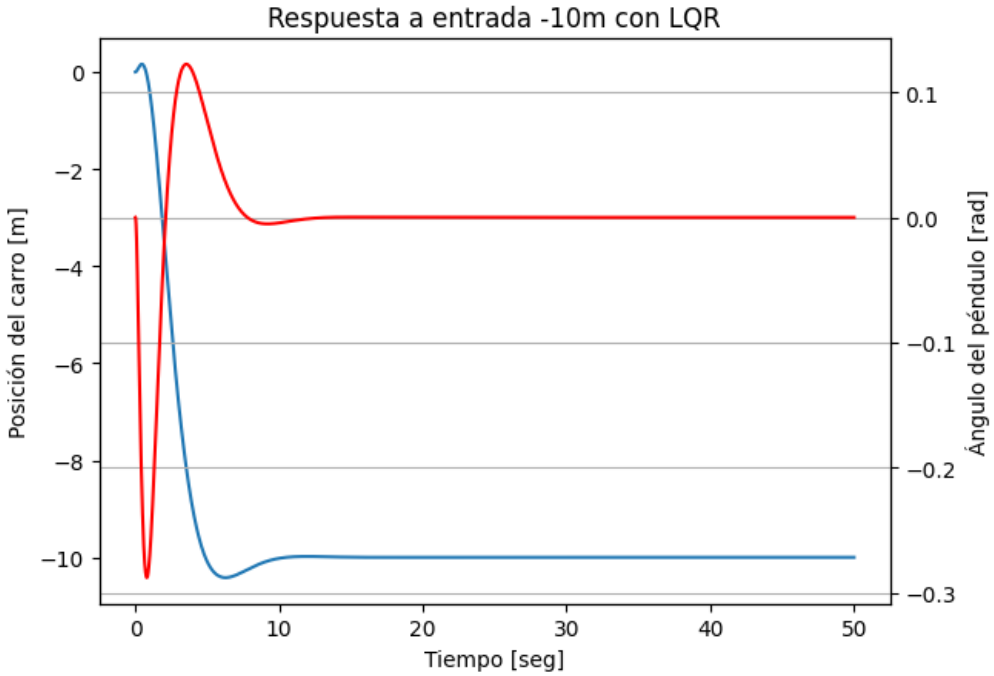

Figura 9: Simulación del péndulo linealizado para ángulo pequeño.

El observador se implementó de la siguiente forma:

Código 9: Observador para el péndulo linealizado.

```
# Planteo del observador
obsPoles = np.array([-6, -7, -8, -9])
# Matriz L
L = place_poles(A.T, C.T, obsPoles).gain_matrix.T
# Matrices de estado ampliadas
Aamp = np.block([[A - B @ K, B @ K], [np.zeros_like(A), A - L @ C]])
Bamp = np.block([[B], [np.zeros_like(B)]])
Camp = np.block([[Cc, np.zeros_like(Cc)]])
Damp = np.array([[0], [0]])
# Sistema a lazo cerrado
obsSysCL = StateSpace(Aamp, Bamp, Camp, Damp)
# Simulación lineal del observador
t1, y1, x1 = obsSysCL.output(U=r, T=t)
# Gráfica del sistema observado
fig, ax1 = plt.subplots()
ax2 = ax1.twinx()
ax1.plot(t1, y1[:, 0])
ax2.plot(t1, y1[:, 1], 'r')
ax1.set_ylabel('Posición del carro observado [m]')
ax2.set_ylabel('Ángulo del péndulo observado [rad]')
ax1.set_xlabel('Tiempo [seg]')
plt.title('Respuesta a entrada -10m con LQR observada')
plt.grid(True)
```

```
24 plt.show()
```

El código del observador arroja la siguiente gráfica:

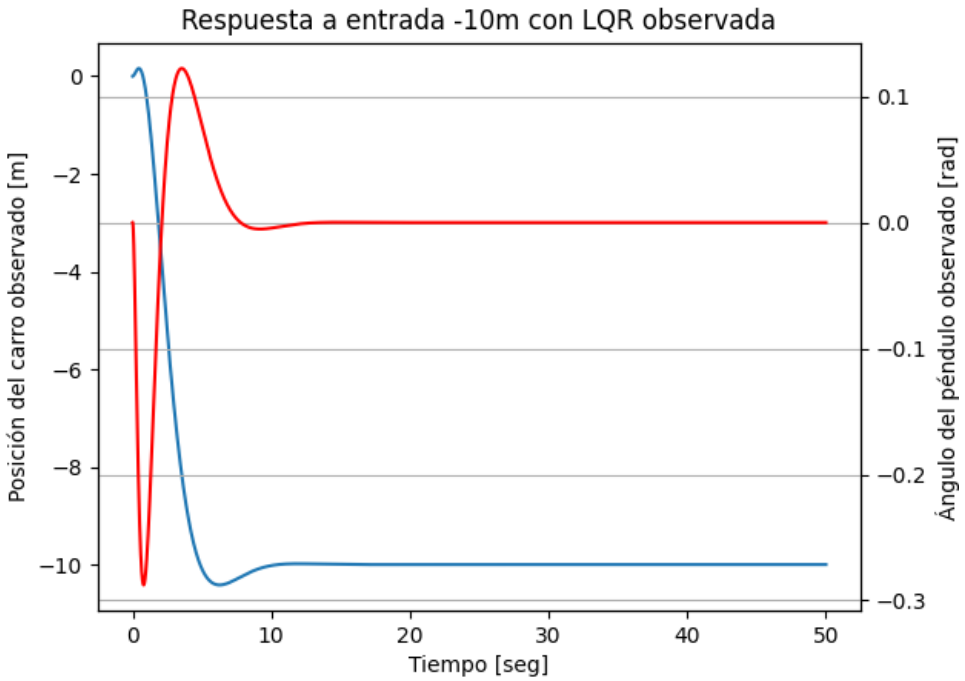

Figura 10: Simulación del observador para el péndulo linealizado para ángulo pequeño.

3.2. Simulación para sistema no lineal (masa constante)

Ahora, para simular el sistema no lineal, se planteó una metodología similar a la aplicada al caso del motor de corriente continua. Es necesario aclarar que para poder obtener los tiempos de simulación y período de integración para Euler fue necesario utilizar el modelo linealizado para analizar la función de transferencia y con ella diseñar los controladores también. Se muestra a continuación las matrices del sistema de estado linealizadas obtenidas a partir de evaluar las matrices declaradas en la sección 3.1 con los valores provistos en el ejercicio.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.0667 & -0.6533 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.0667 & 10.4533 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0.6667 \\ 0 \\ -0.6667 \end{bmatrix}$$

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

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Para el uso del controlador LQR, se planteó:

Código 10: LQR para item 3.

```
Q = C'*C;
R = 1;
K = lqr(A,B,Q,R);

```

Lo que dio lugar a la siguiente matriz:

$$K = \begin{bmatrix} -1 & -2.5237 & -44.1772 & -13.7953 \end{bmatrix}$$

Con esa matriz, el polo más rápido del sistema a lazo cerrado es $\lambda_1 = -32.2344 + j0.1031$. Lo que lleva a:

$$t_r = \frac{ln(0.95)}{-3.3403} = 0.0154$$

Como el tiempo de integración debe ser menor a t_r se decidió por utilizar 1 10^{-4}, sin embargo, un tiempo más grande hubiera sido válido también.

Para el tiempo de simulación, se analizó el polo más cercano al origen, dando un $t_l = 16.6615$, lo que hizo que se simule por 3 segundos. Con eso en mente, se simuló el sistema. Se usó un vector **reference** que almacenaba los datos de la referencia (posición y ángulo) y las condiciones iniciales de simulación. El código es el que se adjunta a continuación:

Código 11: Simulación del péndulo no lineal.

```
G = -inv(C*inv(A-B*K)*B);
h = 1e-4;
simTime = 50;
t = 0:h:(simTime-h);
reference = [-10 0];
delta(1) = 0;
deltaP(1) = 0;
phi(1) = 0;
phiP(1) = 0;
stateVector = [delta(1) deltaP(1) phi(1) phiP(1)]';
xop = [0 0 0 0]';
x = [delta(1) deltaP(1) phi(1) phiP(1)]';
phiPP = 0;
for i = 1:(simTime/h)
   u(i) = -K*stateVector+reference(1)*G;
   delta(i) = x(1);
   deltaP(i) = x(2);
   phi(i) = x(3);
```

```
    phiP(i) = x(4);
    deltaPP = (u(i)-F*x(2)-m*l*phiPP*cos(x(3)-xop(3))+m*l*sin(x(3)-
    xop(3))*x(4)^2)/(M+m);
    phiPP = (g*sin(x(3)-xop(3))-deltaPP*cos(x(3)-xop(3)))/l;
    x1P = x(2);
    x2P = deltaPP;
    x3P = x(4);
    x4P = phiPP;
    xP = [x1P x2P x3P x4P]';
    x = x+xP*h;
    stateVector = [delta(i) deltaP(i) phi(i) phiP(i)]';
  end
```

El resultado de la simulación se puede apreciar en la siguiente gráfica, en donde se incluyen cada una de las variables y su evolución.

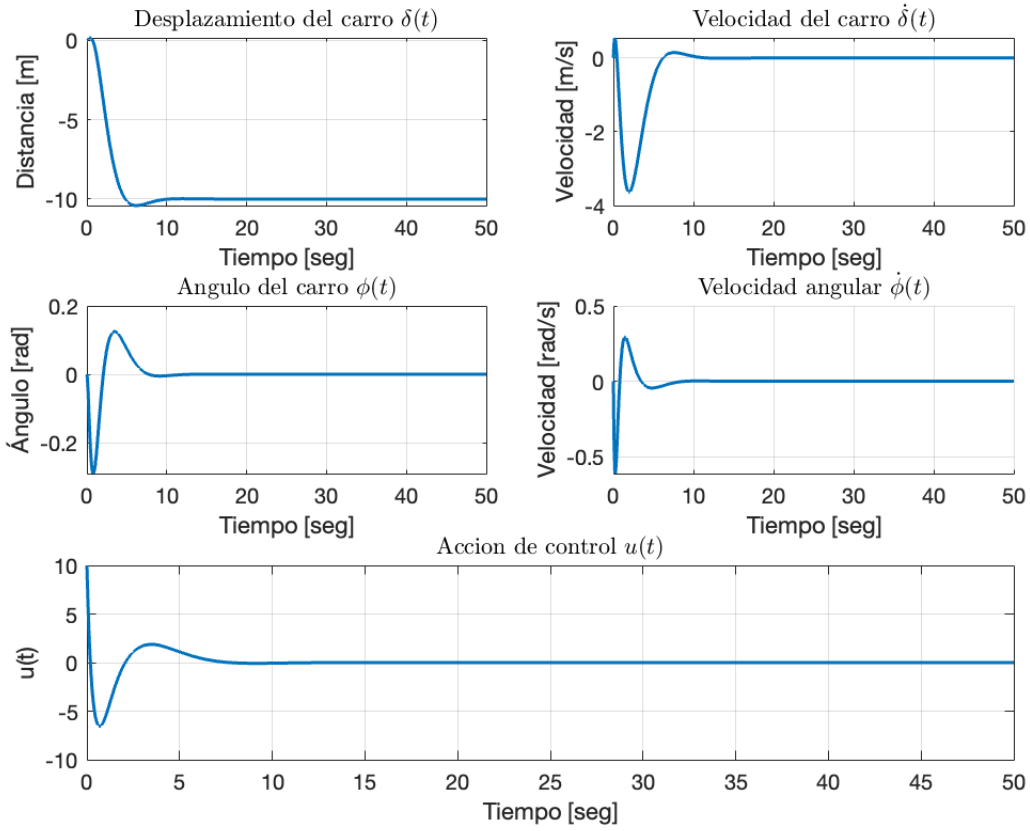

Figura 11: Simulación del péndulo no linealizado para setpoint de 10 m.

Se pide a continuación incorporar un observador para el caso en que solo pueden medirse el desplazamiento δ y el ángulo ϕ.

Para poder implementar ese observador se declara la siguiente matriz:

$$Q_o = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$$

Que junto con un $R = 10$ y las matrices para el observador obtenidas a partir de transponer las propias del sistema, permiten obtener la siguiente matriz con la función **lqr()**.

$$K_o = \begin{bmatrix} 0.3983 & 0.1231 & -0.4332 & -1.1117 \\ -0.4332 & -1.2615 & 5.0807 & 12.9505 \end{bmatrix}$$

Ahora, con esta matriz, y el planteo de las diferentes condiciones iniciales, se tiene el código a continuación, que simula el sistema observado.

Código 12: Simulación del observador para péndulo con masa constante.

```
m = 0.1;
F = 0.1;
l = 1.6;
g = 9.8;
M = 1.5;
A = [0 1 0 0; 0 -F/M -m*g/M 0; 0 0 0 1; 0 F/(l*M) g*(M+m)/(l*M) 0];
B = [0; 1/M; 0; -1/(l*M)];
C = [1 0 0 0; 0 0 1 0];
Q = C'*C;
R = 1;
K = lqr(A,B,Q,R);
G = -inv(C(1,:)*inv(A-B*K)*B)
Qo = diag([1 0.01 1 0.01])
Ro = 10;
Ao = A';
Bo = C';
Ko = lqr(Ao,Bo,Qo,Ro)
h = 1e-4;
simTime = 50;
t = 0:h:(simTime-h);
reference = [-10 0];
delta(1) = 0;
deltaP(1) = 0;
phi(1) = 0;
phiP(1) = 0;
stateVector = [delta(1) deltaP(1) phi(1) phiP(1)]';
obsStateVector = [delta(1) deltaP(1) phi(1) phiP(1)]';
xop = [0 0 0 0]';
x = [delta(1) deltaP(1) phi(1) phiP(1)];
xObs = [0 0 0 0]';
phiPP = 0;
for i = 1:(simTime/h)
  u(i) = -K*stateVector+reference(1)*G;
  delta(i) = x(1);
```

```
    deltaP(i) = x(2);
    phi(i) = x(3);
    phiP(i) = x(4);
    deltaPP = (u(i)-F*x(2)-m*l*phiPP*cos(x(3)-xop(3))+m*l*sin(x(3)-xop(3))*x(4)^2)/(M+m);
    phiPP = (g*sin(x(3)-xop(3))-deltaPP*cos(x(3)-xop(3)))/l;
    x1P = x(2);
    x2P = deltaPP;
    x3P = x(4);
    x4P = phiPP;
    xP = [x1P x2P x3P x4P]';
    x = x+xP*h;
    deltaO(i) = xObs(1);
    deltaPO(i) = xObs(2);
    phiO(i) = xObs(3);
    phiPO(i) = xObs(4);
    yO = C*obsStateVector;
    y = C*stateVector;
    error = y-yO;
    xPrevP = A*xObs+B*u(i)+Ko'*error;
    xObs = xObs + xPrevP*h;
    stateVector = [delta(i) deltaP(i) phi(i) phiP(i)]';
    obsStateVector = [deltaO(i) deltaPO(i) phiO(i) phiPO(i)]';
end
```

Se comparan entonces las dinámicas observadas con las reales del sistema.

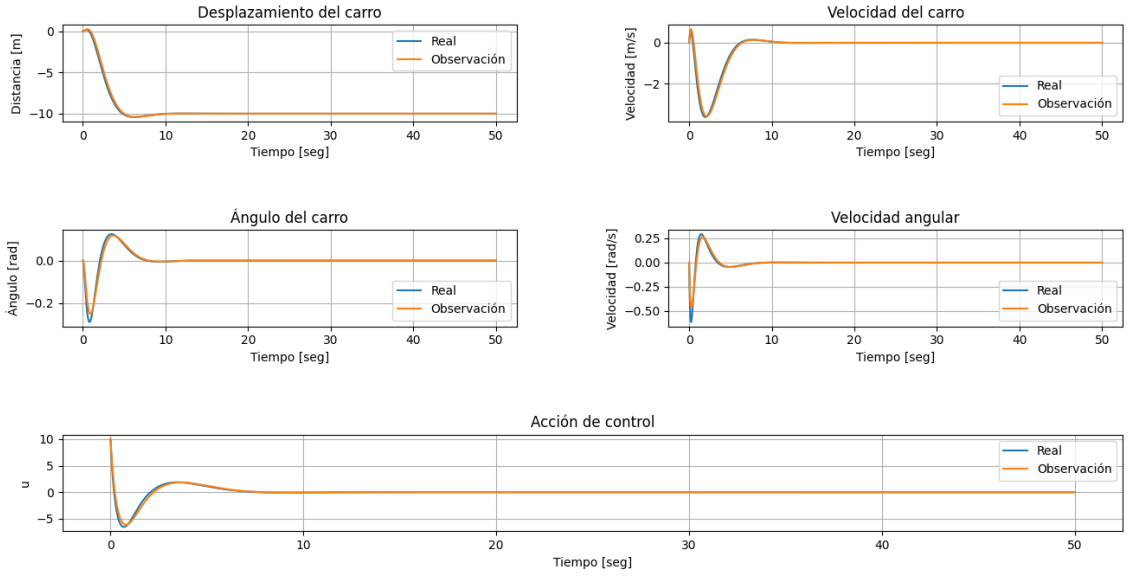

Figura 12: Análisis de la observación para sistema con masa constante.

Es posible apreciar que la aproximación realizada mediante la implementación del observador es representativa de la realidad, ya que apenas se nota diferencia entre las magnitudes.

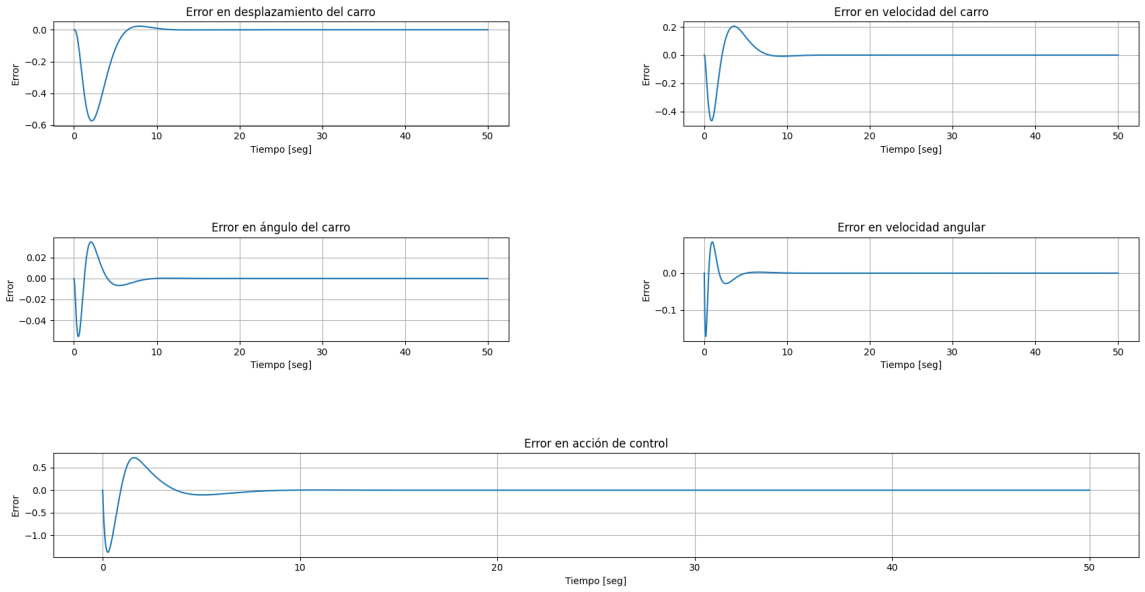

Figura 13: Errores de observación a masa constante.

3.3. Simulación para sistema no lineal (caso de la grúa)

Para este último inciso, se debe implementar un controlador que lleve el péndulo a una referencia de 2 m y cuando llegue a ese punto, se debe cambiar la masa por una de 10 veces el peso inicial. Esto viene a representar el caso en que una grúa enganche una cierta carga, por lo que en el camino de ida y vuelta no se tendrá el mismo sistema. Para ello, se implementarán 2 controlares, uno para cada caso de masa. Como se busca que la carga no sufra oscilaciones, al momento de realizarle el tuning al LQR se le asignará mucho peso a la variable del ángulo en la declaración de la matriz Q. Con esto se podrá apreciar la eliminación de movimientos espasmódicos.

Con el valor de masa inicial y los parámetros físicos del sistema, que permanecen invariantes, se tienen las mismas matrices A, B, C y D que en el inciso anterior. Como se cuenta con otra referencia, se plantean las matrices siguientes para el controlador LQR:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 10000 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{bmatrix}, \; R = 100$$

$$K = \begin{bmatrix} 1.2232 & 2.2498 & 6.7212 & 2.7929 & -0.3162 \end{bmatrix}$$

En el código se divide la matriz K para controlar diferentes instantes de la situación a simular. Esto se da par ambos casos de masa. Se muestra a continuación las matrices que se utilizaron para controlar el péndulo invetido cuando aumenta la masa.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.0667 & -6.5333 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.0417 & -10.2083 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 100000 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \; R = 100$$

$$K = \begin{bmatrix} 0.9550 & 4.4103 & 31.3625 & 27.1855 & -0.1 \end{bmatrix}$$

Con estas matrices, las condiciones iniciales todas nulas excepto el ángulo que vale π, se simula el sistema. El período de integración por Euler es el mismo que se usó anteriormente, y además, se simula por 100 segundos, el doble de tiempo que antes.

Código 13: Simulación de la grúa.

```
h = 1e-4;
simTime = 100;
t = 0:h:(simTime-h);
refDelta = square(2*pi*(1/100)*t)+1;
reference = [refDelta pi];
mass = -0.45*square(2*pi*(1/100)*t)+0.55;
delta(1) = 0;
deltaP(1) = 0;
phi(1) = pi;
phiP(1) = 0;
stateVector = [delta(1) deltaP(1) phi(1) phiP(1)]';
xop = [0 0 pi 0]';
x = [delta(1) deltaP(1) phi(1) phiP(1)]';
zeta(1) = 0;
integ(1) = zeta(1);
phiPP = 0;
K = Kint;
KI = Kint1;
for i = 1:(simTime/h)
  if(mass(i) > 0.1)
    K = KintCar;
    KI = KintCar1;
  end
  zetaP = refDelta(i)-C1(1,:)*(stateVector-xop);
  zeta(i) = integ+h*zetaP;
  u(i) = -K*(stateVector(1:4)-xop(1:4))+KI*zeta(i);
  delta(i) = x(1);
  deltaP(i) = x(2);
  phi(i) = x(3);
  phiP(i) = x(4);
```

```
    deltaPP = (u(i)-F*x(2)-mass(i)*l*phiPP*cos(x(3))
    +mass(i)*l*sin(x(3))*x(4)^2)/(M+mass(i));
    phiPP = (g*sin(x(3))-deltaPP*cos(x(3)))/l;
    x1P = x(2);
    x2P = deltaPP;
    x3P = x(4);
    x4P = phiPP;
    xP = [x1P x2P x3P x4P]';
    x = x+xP*h;
    stateVector = [delta(i) deltaP(i) phi(i) phiP(i)]';
    integ = zeta(i);
  end
```

El código arroja los siguientes resultados:

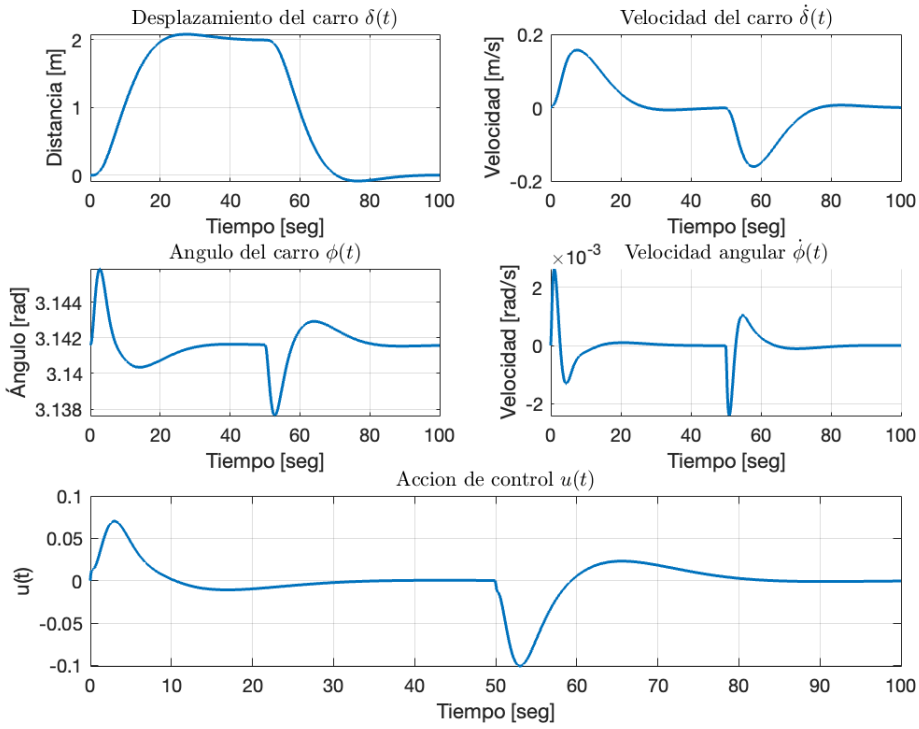

Figura 14: Simulación de la grúa.

Ahora se debe implementar un observador, al igual que en el inciso anterior. Para ello se plantearon las siguientes matrices K. Se ve para el caso de masa menor, y luego para la masa aumentada.

$$K = 10^5 * \begin{bmatrix} 0.0069 & 1.1158 & 0.0009 & 0.3251 \\ 0.0010 & 0.3717 & 0.0071 & 1.1833 \end{bmatrix}$$

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, K = \begin{bmatrix} 1.9137 & 1.3970 & 0.3632 & 1.2106 \\ 0.3632 & -0.1964 & 0.8791 & -0.0477 \end{bmatrix}$$

El código utilizado para simular es el siguiente:

Código 14: Simulación del observador para la grúa.

```
deltaO(1) = 0;
deltaPO(1) = 0;
phiO(1) = pi;
phiPO(1) = 0;
states = [deltaO(1) deltaPO(1) phiO(1) phiPO(1)]';
xop = [0 0 pi 0]';
x = [deltaO(1) deltaPO(1) phiO(1) phiPO(1)]';
zetaObs(1) = 0;
integObs(1) = zetaObs(1);
phiPP = 0;
xObs = [deltaO(1) deltaPO(1) phiO(1) phiPO(1)]';
obsStateVector = [deltaO(1) deltaPO(1) phiO(1) phiPO(1)]';
K = Kint;
KI = Kint1;
Ko = Kob;
for i = 1:(simTime/h)
  if(mass(i) > 0.1)
    K = KintCar;
    KI = KintCar1;
    Ko = KoCar;
  end
  zetaPObs = refDelta(i)-C1(1,:)*(obsStateVector-xop);
  zetaObs(i) = integObs+h*zetaPObs;
  u(i) = -K*(obsStateVector-xop)+KI*zetaObs(i);
  deltaO(i) = x(1);
  deltaPO(i) = x(2);
  phiO(i) = x(3);
  phiPO(i) = x(4);
  deltaPP = (u(i)-F*x(2)-mass(i)*l*phiPP*cos(x(3))
  +mass(i)*l*sin(x(3))*x(4)^2)/(M+mass(i));
  phiPP = (g*sin(x(3))-deltaPP*cos(x(3)))/l;
  x1P = x(2);
  x2P = deltaPP;
  x3P = x(4);
  x4P = phiPP;
  xP = [x1P x2P x3P x4P]';
  x = x+xP*h;
  deltaObs(i) = xObs(1);
  deltaPObs(i) = xObs(2);
  phiObs(i) = xObs(3);
  phiPObs(i) = xObs(4);
  yObs = C1*obsStateVector;
```

```
    y = C1*states;
    error = y-yObs;
    xPrevP = A*(xObs-xop)+B*u(i)+Ko'*error;
    xObs = xObs+h*xPrevP;
    states = [deltaO(i) deltaPO(i) phiO(i) phiPO(i)]';
    obsStateVector = [deltaObs(i) deltaPObs(i) phiObs(i) phiPObs(i)]';
    integObs = zetaObs(i);
end
```

Y se muestra a continuación la comparación con el sistema real.

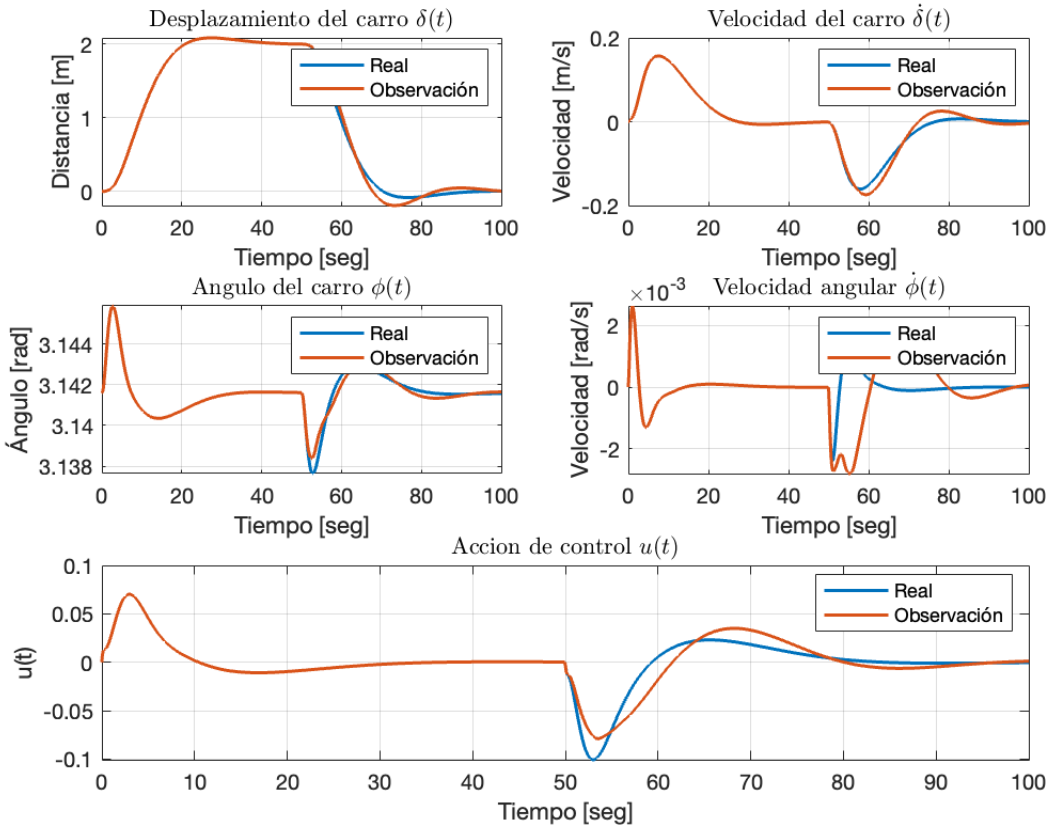

Figura 15: Comparación de observación y realidad del sistema grúa.

Se aprecia que el mayor error de observación aparece en la velocidad angular. El resto de las observaciones se mantienen dentro de lo esperado luego de comprobar la performance de las simulaciones anteriores de la actividad.

3.4. Sintonizado de LQR

Al igual que en el ejercicio en donde se analizaba el control de un motor de corriente continua. El sintonizado del controlador LQR se hizo mediante sucesivas simuaciones, comprobando el verdadero funcionamiento del sistema bajo las referencias planteadas. Cabe destacar que para el caso en que

simplemente se llevaba el péndulo de una posición a otra, se planteó una matriz Q diagonal de forma diferente a como se hizo en el resto de la actividad. Esto significa que se trabajó con una matriz identidad simplemente, asignándole el mismo peso a cada uno de los parámetros ($Q = C'C$ y $R = 1$).

Luego, para poder sintonizar el observador se utilizó una táctica de comprobación de los resultados de la simulación para diferentes situaciones (variación de Q y R).

4. Observaciones

4.1. Generalidades

En el desarrollo del trabajo práctico aparecieron diferentes complicaciones que hicieron que se deba profundizar más el conocimiento en ciertos temas, así como también fue necesario ampliar el repertorio de funciones que se maneja para el uso de simulaciones en el software. A continuación se enumerarán algunas cuestiones que surgieron durante el desarrollo de la actividad, con explicaciones que sirven como futuras recomendaciones.

- **Simulación del motor de corriente continua**. Al momento de plantear la simulación del motor de corriente continua luego de diseñar un controlador por colocación de polos, apareció el inconveniente de la magnitud de la corriente de armadura para poder simular un cambio de referencia cada 2 segundos. Esto se corrigió en cierta medida gracias al espaciamiento entre los cambios de referencia, sin embargo, la magnitud de la corriente seguía siendo alta. Para ello se planteó bajar la frecuencia del cambio de referencia e implementar un controlador LQR. Luego de diversas pruebas, se comprobó que el método utilizado sirvió para lograr valores razonables en las simulaciones, y, además, parece ser que la gráfica de la consigna se corresponde con valores de parámetros diferentes a los enumerados en la consigna. Se entiende que por esto último es que aparecieron las complicaciones al intentar mejorar la dinámica del sistema. Y es ese factor, junto con el de la idealidad de las simulaciones en el software, que no ponen un límite inicial a diferentes magnitudes, ya que simplemente representan cálculos numéricos para la computadora. Otro punto importante a recalcar con el tema de cómo implementar los controladores viene por el lado de que se debe analizar bien en qué momento vale diseñar po método de asignación de polos, fórmula de Ackermann o LQR, ya que lo válido para un sistema o una determinada referencia puede no serlo para otro.

- **Implementación de controladores LQR**. A lo largo del desarrollo del trabajo práctico se trabajó con todos controladores del tipo LQR, salvo en un observador para el último ejercicio. Esto aparece porque en el primer inciso fue necesario, y luego por inercia se terminó trabajando con ese método. Sin embargo, tal y como se dijo en la oración anterior, el uso de controladores por asignación de polos es válido, y esto pudo comprobarse en el item de observadores para el ejercicio del péndulo invertido. La implementación de controladores u observadores debe realizarse a partir de sistemas linealizados, ya que se plantea un análisis de los estados del sistema. Y, para ello, es necesario contar con una representación matricial en variables de estado. Esta representación debe obtenerse a partir del análisis de las ecuaciones diferenciales que modelan el comportamiento del sistema, y del punto de operación del mismo.

- **Linealización del péndulo**. Para el diseño de controladores LQR fue necesario trabajar con expresiones linealizadas que modelen los sistemas en juego (como se indicó en el ítem anterior). Pero luego, al momento de simular, lo estrictamente correcto es plantear una simulación no lineal, haciendo uso de las ecuaciones diferenciales que expresan el comportamiento del sistema del péndulo por ejemplo. Esto introduce una dificultad extra, sin embargo, el error en la aproximación realizada en la segunda clase es leve, y esto pudo verse al simular el sistema mediante un bucle con Euler o haciendo uso de una simulación lineal con la función lsim(). Existe una función de Matlab ($ode45()$) y de Python ($solve_ivp()$) que permite trabajar con ecuaciones diferenciales ordinarias, y permitirían realizar simulaciones a sistemas de la índole del péndulo

invertido sin la necesidad de aplicar un bucle for, por ejemplo. Se optó por realizar las simulaciones con bucles ya que de esa forma se enseñaron en la materia de **Métodos Numéricos**, y en los apuntes de clase aparecen códigos ilustrativos que utilizan este tipo de algoritmos.

- **Sintonización de LQR**. Se pudo comprobar, mediante fuerza bruta (esto se realizó simplemente por la posibilidad de contar con un software de simulación), que el aumento de la ponderación de cada variable de estado en la matriz Q permite que el control sobre la misma posea una dinámica más suave. Esto fue vital en el desarrollo del último controlador para el inciso del péndulo, en donde se necesitaba que la masa grande no presente oscilaciones. En las gráficas a priori puede parecer que existen grandes oscilaciones, pero al analizar la escala a la que ocurren, porcentajes menores al 1 % con respecto a π, se puede ver que son prácticamente inapreciables.

4.2. Indicadores de logros

En la consigna del aula virtual se planteó la necesidad de indicar las lecciones aprendidas relacionadas con los indicadores de logros. Se enumeran a continuación los que se consideran aprendidos:

- Diseñar, proyectar y calcular sistemas de automatización y control para brindar soluciones óptimas de acuerdo a las condiciones definidas por el usuario.
- Analizar la factibilidad de controlar un proceso real conociendo el modelo.
- Diseñar controladores con realimentación de estados para obtener una dinámica estable considerando la dinámica y la magnitud de las acciones de control.
- Diseñar observadores de procesos linealizados para controlarlos cuando no es viable medir sus variables.
- Diseñar controladores para obtener una dinámica estable del proceso linealizado.
- Analizar el comportamiento de procesos reales mediante su representación matemática no lineal multivariable.

4.3. Repositorio de Github

En el repositorio de GitHub del proyecto, se pueden encontrar los archivos de Matlab en formatos Live Script y el estándar (.m). Además de estos archivos, aparecen algunos en formato .mat, que almacenan la información de las simulaciones realizadas. Estos archivos son utilizados en un script de Python que compara simulaciones de procesos reales con observadas y además provee una visualización de los errores de observación presentes. Existe también un pequeño script en Python que realiza una simulación de un péndulo lineal.

A modo de ilustración, y, porque se lo consideró complementario, se adjunta un par de scripts de Julia que muestran el control de un sistema de péndulo invertido, con linelización y control PID. Esto, si bien no representa lo solicitado por la consigna, sirve a modo de ejemplo para comprobar cómo se pueden realizar diferentes simulaciones haciendo uso de un lenguaje que no sea Matlab ni Python. Estos scripts fueron extraídos de los ejemplos de la librería RobustAndOptimalControl (https://juliacontrol.github.io/RobustAndOptimalControl.jl/dev/).

4.4. Enlaces de utilidad

- https://github.com/all5one-sudo/TPsControl2
- https://python-control.readthedocs.io/en/0.9.3.post2/
- https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=ControlStateSpace
- https://ctms.engin.umich.edu/CTMS/index.php?aux=Animations_Invgui
- https://www.mathworks.com/matlabcentral/fileexchange/97187-inverted-pendulum-swing-up-lqr-control-design-simulation
- https://repository.udistrital.edu.co/bitstream/handle/11349/4020/GonzalezUribeChristianDavid2016.pdf;jsessionid=8B4E35737B442413678CBC95869BA7B9?sequence=1
- https://biblat.unam.mx/hevila/Revistapolitecnica/2012/no15/5.pdf
- http://www.me.unlv.edu/Undergraduate/coursenotes/control/IP01_Upright_Pendulum.pdf

5. Conclusiones

Luego de la realización de este trabajo práctico, es posible decir que se adquirió un conocimiento integral de los aspectos del modelado en variables de estado, así como del control de procesos modelados con esta técnica. Se pudo comprender la importancia del análisis de la magnitud de la acción de control y de variables del sistema que no son las referenciadas de una forma directa (como lo fue la corriente de armadura para el caso del motor).

Si bien durante el desarrollo del trabajo surgieron ciertos inconvenientes, estos sirvieron para comprender mejor qué camino tomar en cuanto al diseño de controladores, así como se pudo discernir también que es necesario realizar un análisis del sistema a modelar, el modelo del mismo, y comprender si es necesario realizar alguna linealización. Se pudo ver además, que para el caso del motor de corriente continua apareció una respuesta que no era la esperada, que, se infiere que puede deberse a los valores de los parámetros indicados. Es decir, no es que estén mal, sino que a priori, al intentar diseñar un controlador por asignación de polos, o por fórmula de Ackermann, se llega a un resultado no óptimo, y esto hizo que se tenga que investigar y familiarizar con el método de LQR para luego terminar implementándolo en el resto del trabajo práctico.

Finalmente, se entiende que el tema fue comprendido en su totalidad por el alumno, lográndose así cumplir los siguientes objetivos:

- Modelado en espacio de estados y análisis del control de los mismos.
- Diseño de controladores en variables de estado.
- Análisis de la acción de control.
- Diseño de controladores por método de asignación de polos, fórmula de Ackermann y LQR.
- Implementación de simulaciones para sistemas no lineales.
- Desarrollo e implementación de observadores.

Referencias

[1] Modelación de un sistema monovariable de orden n. Material de clases del Profesor Pucheta.

[2] Análisis en el espacio de estados. Material de clases del Profesor Pucheta.

[3] Diseño de control lineal en variables de estados en tiempo continuo. Material de clases del Profesor Pucheta.

[4] Controladores - Técnicas Modernas. Material de clases del Profesor Pucheta.

[5] Inverted Pendulum - System Modelling. https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling

[6] DC Motor Position: System Analysis. https://ctms.engin.umich.edu/CTMS/index.php?example=MotorPosition&section=SystemAnalysis

[7] Control of an Inverted Pendulum. https://www.control.isy.liu.se/student/tsrt03/files/invpendpmenglish.pdf

[8] Mathematical Control Theory, Eduardo D. Sontag. http://www.sontaglab.org/FTPDIR/sontag_mathematical_control_theory_springer98.pdf