Actividad Práctica Nº3: Diseño de controladores considerando la dinámica del error y la magnitud de la acción de control en sistemas no lineales multivariables

Alumno: Villar, Federico Ignacio
Profesor: Pucheta, Julián Antonio
Fecha de entrega: 14 de junio de 2023

Contenido

Introducción

En el siguiente informe se detalla la resolución de la tercera actividad práctica propuesta por el profesor Julián Pucheta para la materia de Sistemas de Control II. Se explican brevemente los procedimientos utilizados para el desarrollo de los controladores solicitados, así como también se adjuntan las gráficas que hacen a la interpretación de resultados obtenidos.

Este trabajo práctico requirió de diversas simulaciones para comprobar funcionamientos y determinar límites admisibles, por lo que se optó por no agregar todas las gráficas al informe, de modo de no hacerlo demasiado extenso. Es por ello que todas las simulaciones desarrolladas en Live Script de Matlab se cargan en el repositorio del proyecto en GitHub.

Consigna

Caso 1. Sistema de 3 variables de estado.

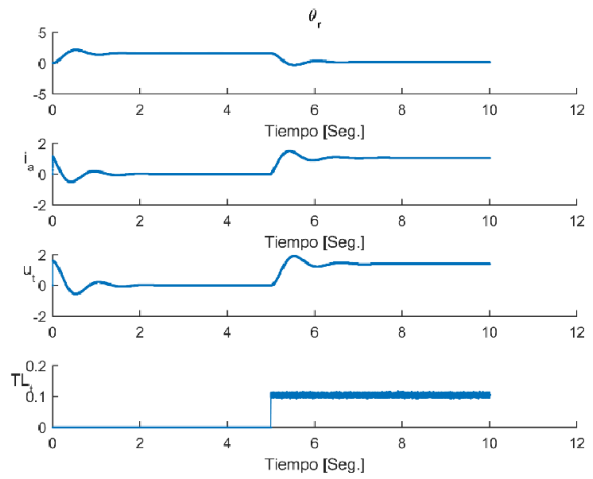


Fig. 1. Evolución del ángulo cuando el controlador tiene perturbaciones en su operación.

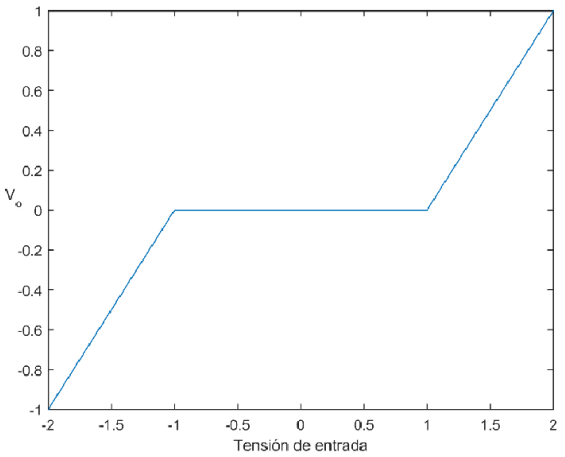


Fig. 2. Relación de entrada salida que tiene el actuador lineal con zona muerta de 1V.

Dadas las ecuaciones del motor de corriente continua con torque de carga TL máximo de 1.5Nm, con los parámetros:

- LAA=0.56 10-3;
- J=0.0019;
- Ra=1.35;
- Bm= 0.000792;
- Ki=.1;
- Km=.1

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

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

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

Item [1] Implementar un sistema en variables de estado que controle el ángulo del motor, para consignas de $\pi/2$ y $-\pi/2$ cambiando cada 5 segundos y que el TL de 1,5 Nm aparece sólo para $\pi/2$, para $-\pi/2$ es nulo. Hallar el valor de integración Euler adecuado.

Objetivo: acelerar la dinámica del controlador que muestra la Fig. 1 aunque ésta se muestre en otra escala de tiempo verificando el resultado con las curvas del archivo xlsx adjunto.

-Evitando que la tensión supere los 24Volts en valor absoluto, especificar el tiempo de muestreo necesario para el controlador cumpla el objetivo.

-Asumiendo que no puede medirse directamente la corriente, proponer un controlador que logre el objetivo.

-Determinar el efecto de la no linealidad en la acción de control, descripta en la Fig. 2, y verificar cuál es el máximo valor admisible de ésa no linealidad.

Caso 2. Sistema lineal de cuatro variables de estado.

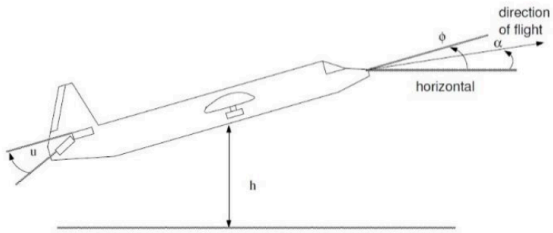


Fig. 3. Modelo de sistema de altitud en un avión.

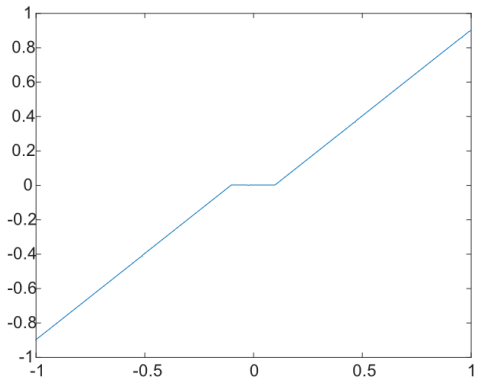


Fig. 4. Relación de entrada salida que tiene el actuador lineal con zona muerta de 0.5.

Para el caso de la Fig. 3, modelo válido sólo para pequeños ángulos, se tiene

$$\begin{cases} \dot{\alpha} = a(\phi - \alpha) \\ \ddot{\phi} = -\omega^2(\phi - \alpha - b \cdot u) \\ \dot{h} = c\alpha \end{cases} \quad (4)$$

donde $\omega>0$ representa la frecuencia natural, y los coeficientes a b son constantes positivas, u es la variable manipulada y es proporcional a la posición de los elevadores, ϕ (ángulo de cabeceo) en radianes, vuela a c metros por segundo, su trayectoria de vuelo forma un ángulo con la horizontal (si $\alpha>0$ sube, si $\alpha<0$ desciende).

Ítem [2] Para el caso del avión, emplear un tiempo de integración por Euler adecuado y un tiempo de simulación de 70seg. Los parámetros son a=0.07; =9; b=5; c=150, hallar un controlador para que los polos de lazo cerrado se ubican en i=-15 ±15j; -0.5± 0.5j, para referencias de 100 y -100 metros en altura, ambas con alturas iniciales de -500 y 500.

-Proponer un controlador en tiempo discreto en variables de estado para que el proceso evolucione en los rangos de validez del modelo, es decir donde los ángulos y el valor de la acción de control en valor absoluto son menores a la unidad.

-Asumiendo que no puede medirse el ángulo , pero sí el ángulo y la altura, proponer un esquema que permita lograr el objetivo de control.

-Establecer el valor del tiempo de muestreo más adecuado para implementar el diseño en un sistema micro controlado.

-Determinar el efecto de la no linealidad en la acción de control, descripta en la Fig. 4, y verificar cuál es el máximo valor admisible de la no linealidad.

Caso de estudio 3. Sistema no lineal de cuatro variables de estado.

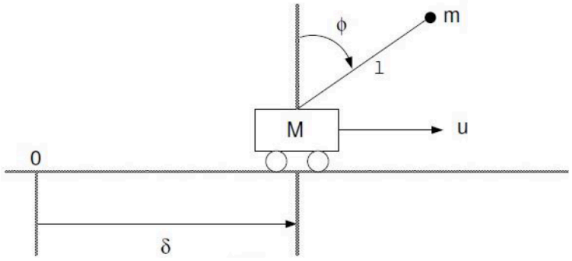


Fig. 5. Sistema del péndulo.

Para el caso del esquema del péndulo invertido de la Fig. 5 donde el modelo es,

$$\begin{cases} (M+m)\ddot{\delta} + ml\ddot{\phi}\cos\phi - ml\dot{\phi}^2 sen\phi + F\dot{\delta} = u \\ l\ddot{\phi} - gsen\phi + \ddot{\delta}\cos\phi = 0 \end{cases} \tag{5}$$

Con las variables de estado $x = \left[\delta \quad \dot{\delta} \quad \phi \quad \dot{\phi}\right]^T$, y los valores de los coeficientes de m=0,1; F=0,1; l=1,6; g=9,8; M=1,5 y $\Delta t = 10^{-4}$ seg, tomando un tiempo de simulación de 15 segundos.

Ítem [3] Calcular sistema controlador que haga evolucionar al péndulo en el equilibrio estable.

Objetivo de control: partiendo de una condición inicial nula en el desplazamiento y el ángulo en π, hacer que el carro se desplace a 10 metros evitando las oscilaciones de la masa m, considerando que es una grúa. Una vez que δ=10 modificar a m a un valor 10 veces mayor y volver al origen evitando oscilaciones.

-Considerar que sólo puede medirse el desplazamiento δ y el ángulo .

-Especificar el rango posible para el tiempo de muestreo para implementar el sistema en un microcontrolador.

-Determinar el efecto de la no linealidad en la acción de control, descripta en la Fig. 4, y verificar cuál es el máximo valor admisible de ésa no linealidad.

Desarrollo del primer caso

Partiendo de las ecuaciones diferenciales que modelan al motor de corriente continua, para comenzar con las simulaciones y el diseño del controlador es necesario conocer la representación matricial en variables de estado del sistema. Para ello, se parte de las siguientes matrices:

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

Luego, con las matrices valuadas en Matlab, se obtiene un objeto sistema a partir de la representación en espacio de estados. Luego de esto, se analizan las dinámicas del sistema obtenido, y con ello se plantea el tiempo de integración, que nuevamente, al igual que en las actividades anteriores, se lo tomó menor a la dinámica más rápida, y el tiempo de simulación utilizado es de 10 segundos, tal como aparecen los datos de simulación de Excel. El período de integración escogido es de $h = 1.10^{-5}$. El período de muestreo debe ser mayor al período de integración, así que se opta por utilizar un orden de magnitud superior $T_S = 1.10^{-4}$.

Una vez declarado período de muestreo se procede a discretizar el sistema con ese tiempo y bajo la utilización de un retentor de orden cero. Para el control del sistema se amplían las matrices A y B, de modo de contar con la posibilidad de un integrador que elimine el error en estado de régimen. Con las matrices ampliadas, se diseña un controlador DLQR, cuya matriz K está dada como:

$$K = [4.8058 \quad 14.3855 \quad 146.2666]$$

Y KI:

$$KI = 1.9207$$

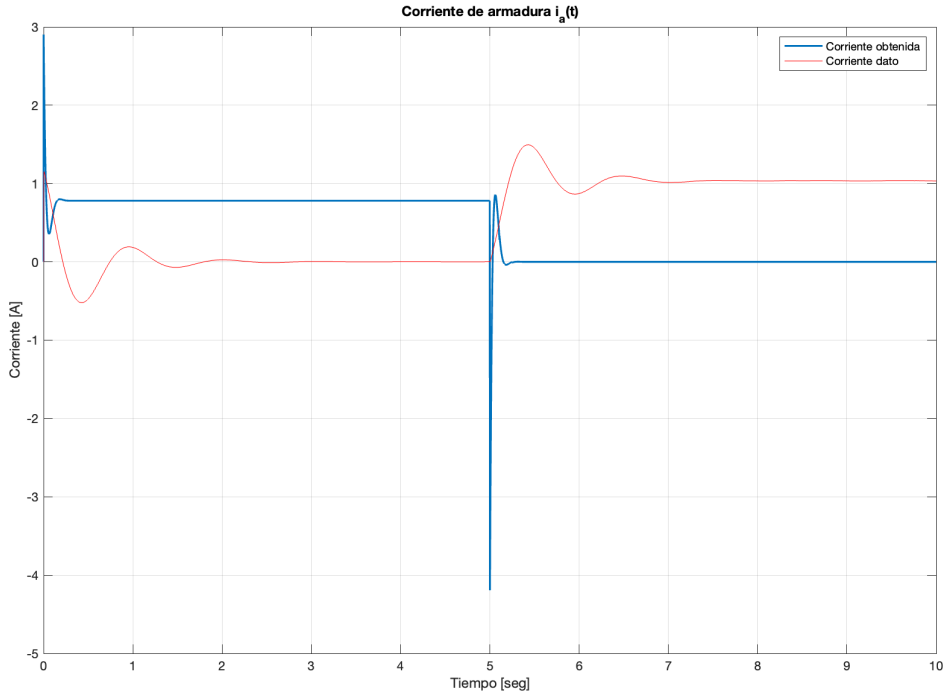
Como la corriente en el sistema no es medible, se accede a su valor mediante el uso de un observador y se la controla de nuevo con un método de DLQR. Su matriz es:

$$K_O = \begin{bmatrix} 0 & 0 & 0.9902 \\ -0.0084 & 0.9902 & 0.0001 \end{bmatrix}$$

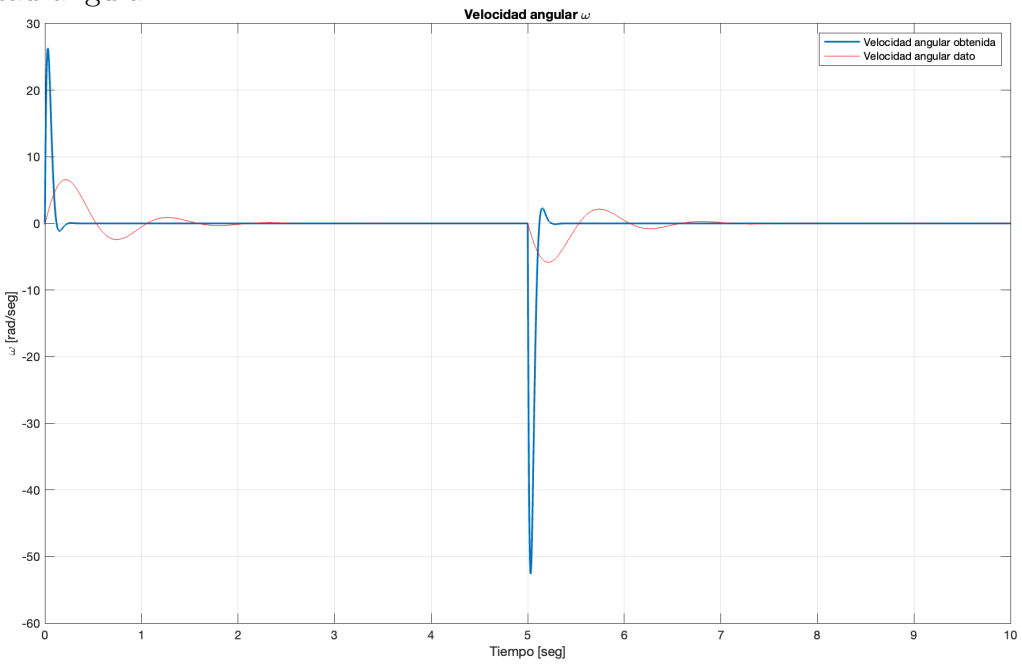
Otro aspecto a tener en cuenta en el motor es que el actuador posee una no linealidad, dada como una zona muerta, esa alinealidad generará una acción de control que a priori no es la deseada, sin embargo, con las correcciones propias del sistema, existirán límites admisibles para ella.

Primeros resultados de simulación

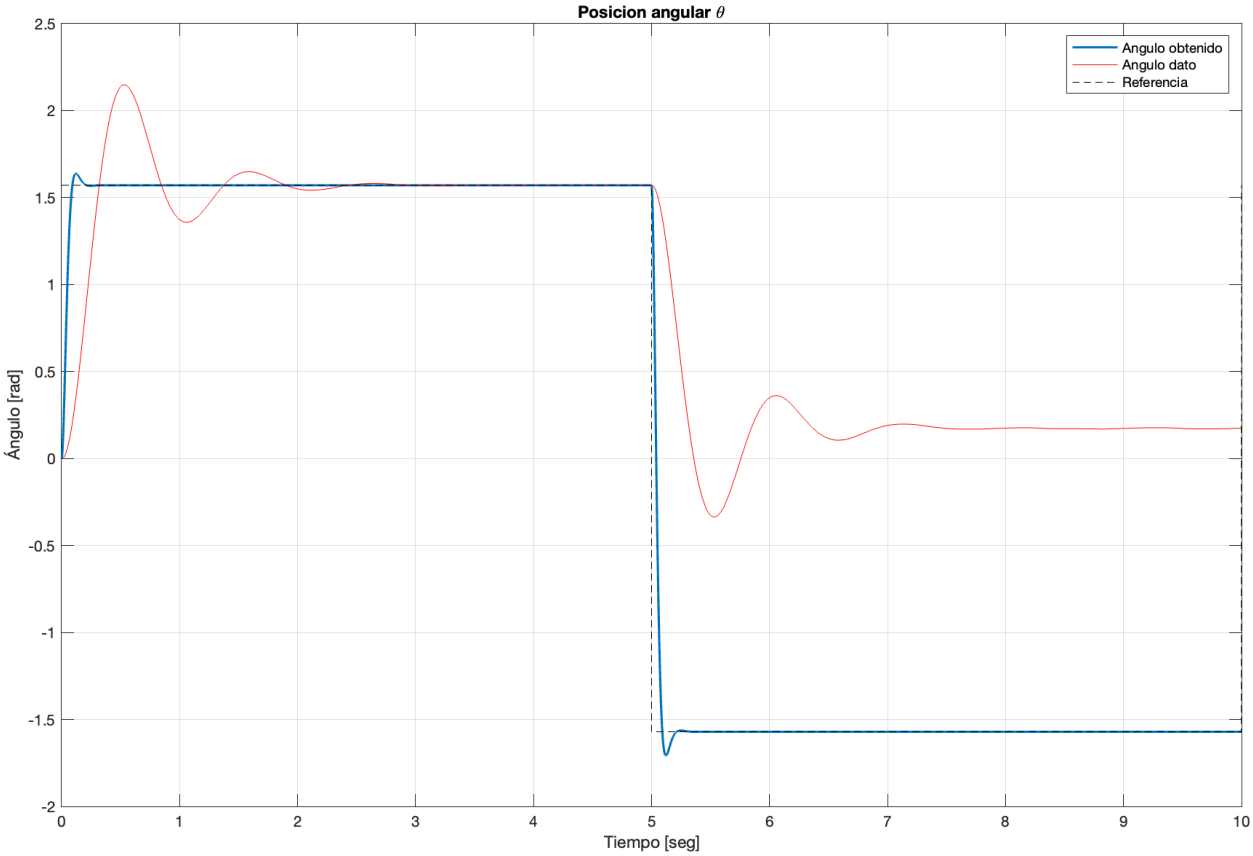
La primera simulación del programa se hace con una alinealidad unitaria. Para ver los resultados, y comparar, en el Live Script desarrollado, se hace uso de un slider que permite variar rápidamente la no linealidad. En un principio, se obtienen los resultados a continuación. Cuando en la leyenda se habla de “dato”, se hace referencia a los datos que brinda el Excel con las curvas medidas del motor controlado mediante un controlador del tipo P.



Se puede ver que en la corriente, el sistema controlado de la forma planteada adquiere valores superiores a los medidos inicialmente en la curva del motor obtenida del Excel. Esto tiene que ver con el control diseñado, además de que, ahora es posible adquirir un estado en donde se pueda seguir a la referencia, y no como antes que para -π/2 el sistema no respondía correctamente. A continuación, se adjunta la gráfica de la velocidad angular.

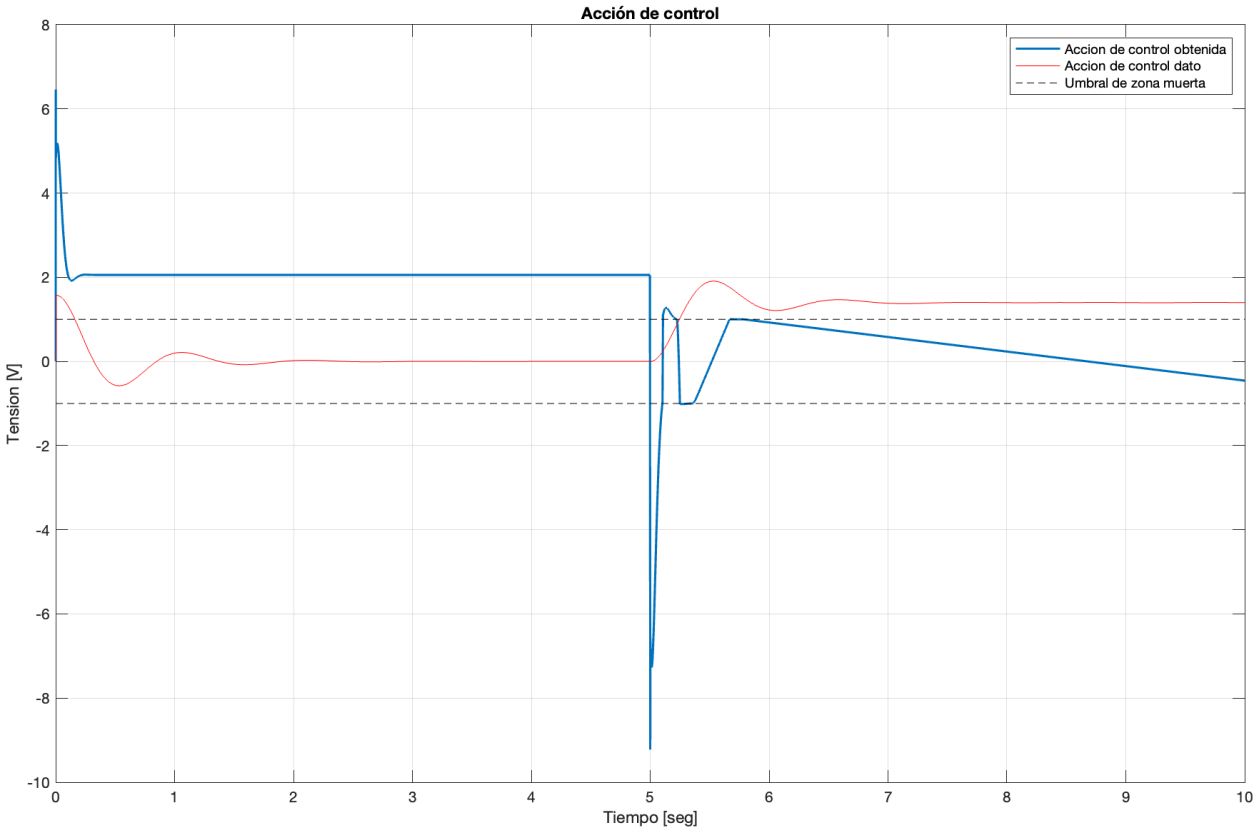


En la figura anterior se aprecia una gran diferencia en la velocidad angular del eje del motor, esto en gran medida se debe a la disminución del tiempo de establecimiento del sistema.



En la gráfica superior puede verse cómo el nuevo sistema se ajusta rápidamente a la referencia planteada, y además, posee un sobrepasamiento de menor magnitud, lo que está implicando una mejora en el control del mismo. Se puede apreciar también que el sistema ahora puede ajustarse a la referencia de -π/2, cosa que antes no se daba. Hasta ahora, el efecto de la no linealidad unitaria no parece afectar a los requerimientos del usuario de controlador del motor, por lo que se está logrando el objetivo de control propiamente dicho.

De las gráficas restaría analizar el comportamiento de la acción de control, ya que esta tiene límites en su actuador, se hablaba de 24 Volts, además de que posee una no linealidad, ya mencionada anteriormente. Se mostrará a continuación la comparación de las curvas, y se delimita el umbral de la zona muerta, de modo de apreciar el efecto que esta introduce.

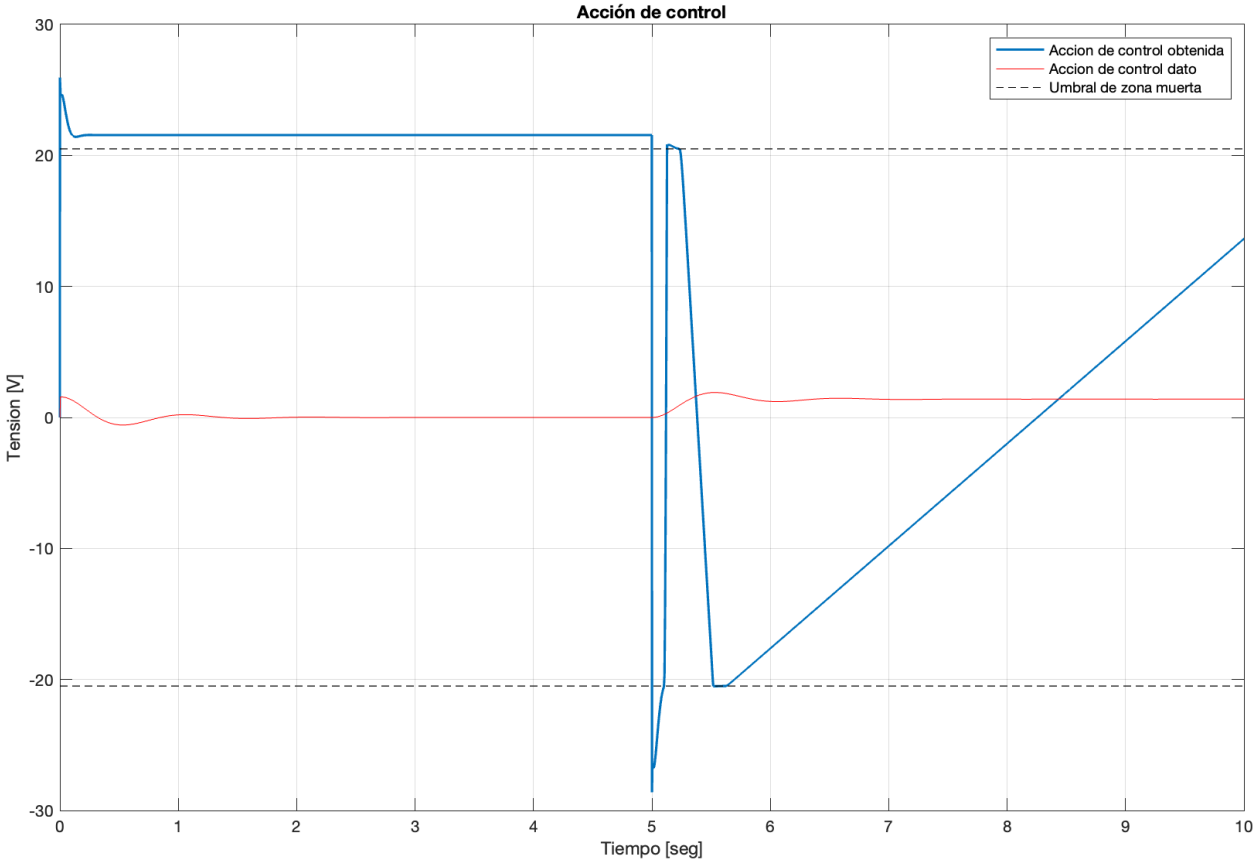


Puede comprobarse entonces, que, según la gráfica, el sistema respeta los límites de actuación del actuador (valga la redundancia), y que la no linealidad afecta el transitorio, pero sin embargo, el sistema logra compensar dentro de los límites establecidos, por lo que se logra el objetivo, como se dijo anteriormente.

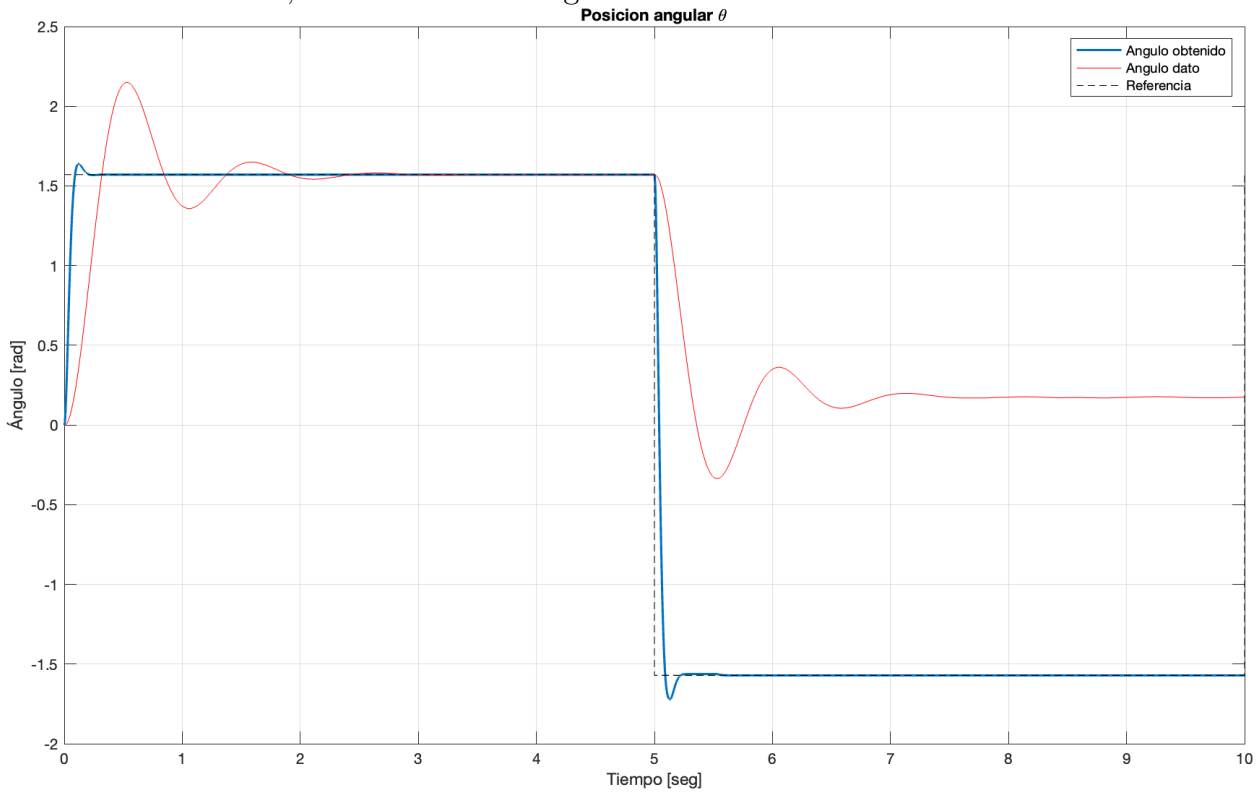
Límites de la zona muerta

La zona muerta del actuador introduce una no linealidad que hace que el control del sistema no pueda operar en ciertos rangos. Se pudo apreciar en la última gráfica adjunta que la acción de control supera en un valor de aproximadamente 4 Volts. Entonces, a simple vista se puede suponer que una no linealidad de aproximadamente 19 Volts recién será problema para el límite de saturación del actuador. Es por eso que se fueron simulando distintos valores, hasta que se encontró que la no linealidad que afecta al sistema es de 20.5 Volts, lo que corrobora la hipótesis inicial.

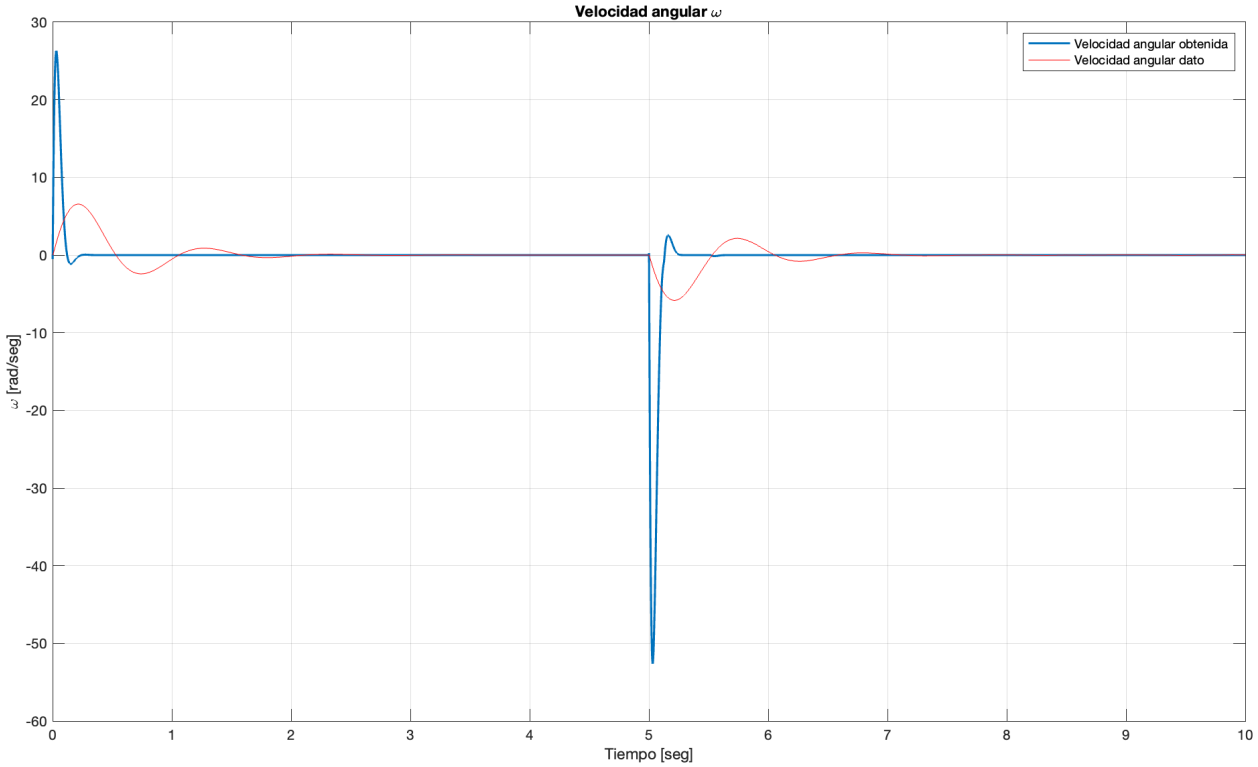
El resultado de la simulación con el valor obtenido de zona muerta en la acción de control es el siguiente.



Con esta simulación, la dinámica del ángulo del motor es:



De esta forma, se puede ver que el control aún sigue manejando la referencia de forma adecuada, simplemente varía el valor de la velocidad angular, por el hecho de que la acción de control debe ser más fuerte. Esto se aprecia en la siguiente gráfica.


Velocidad angular ω
Velocidad angular obtenida
Velocidad angular dato
ω [rad/seg]
Tiempo [seg]
30
20
10
0
-10
-20
-30
-40
-50
-60
0
1
2
3
4
5
6
7
8
9
10

Desarrollo del segundo caso

Para el sistema lineal de cuatro variables de estado del avión, se parte de su modelo en espacio de estados con las siguientes matrices:

$$A = \begin{bmatrix} -a & a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \omega^2 & -\omega^2 & 0 & 0 \\ c & 0 & 0 & 0 \end{bmatrix}; \; B = \begin{bmatrix} 0 \\ 0 \\ b\omega^2 \\ 0 \end{bmatrix}$$

En donde:

- ω es la frecuencia natural.
- a, b son coeficientes positivos

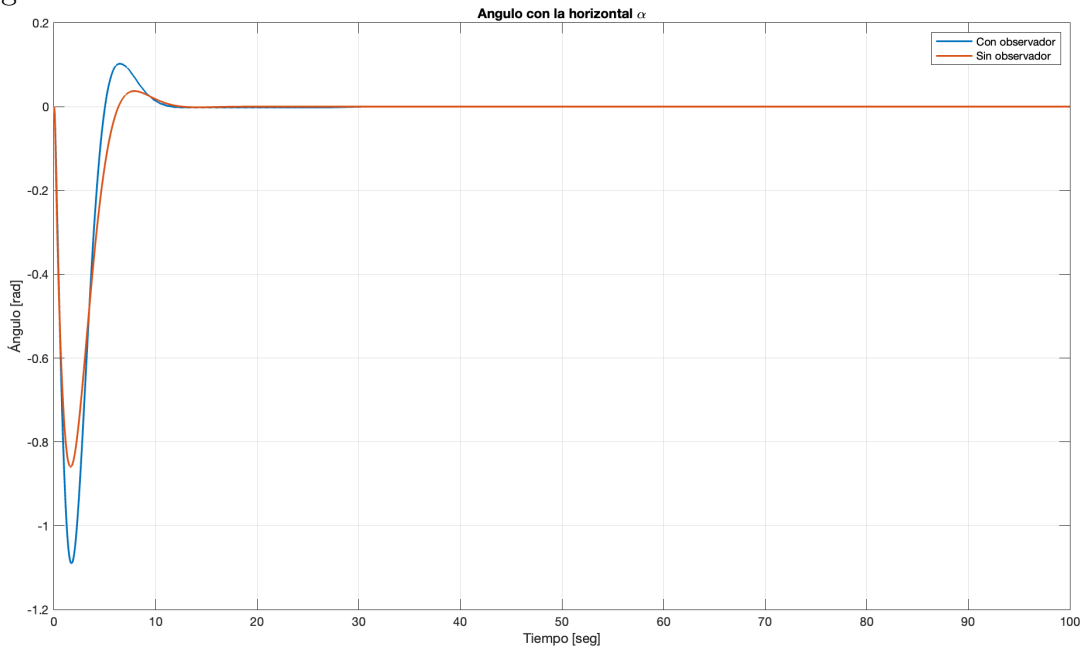
La consigna pide 4 polos a lazo cerrado determinados. Para ello, se hace uso de la función **acker** de Matlab y se obtiene un controlador. El controlador observador se plantea en un inicio a partir de un controlador del tipo LQR, lográndose así la matriz:

$$K_O = \begin{bmatrix} 0.01 & 0.0036 & -0.0118 & 1.7615 \\ 0.0001 & 0.0524 & -0.0036 & 0.0036 \end{bmatrix}$$

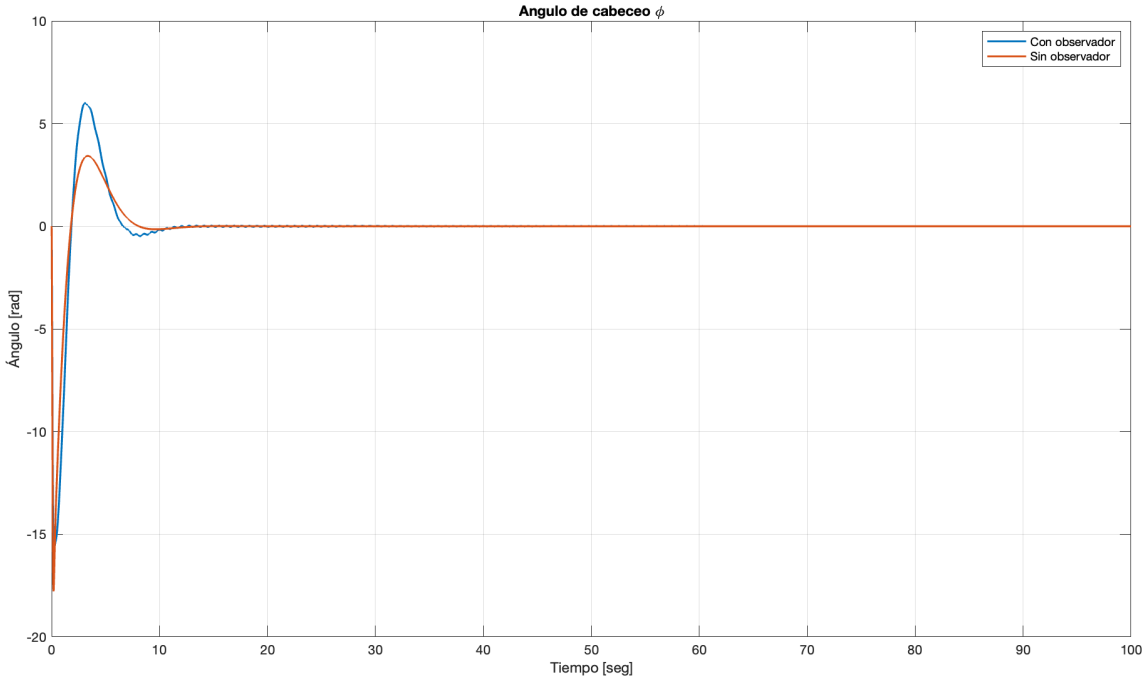
Con esa matriz, y el controlador diseñado mediante la asignación de polos con la fórmula de Ackerman, se logra una primera aproximación.

Simulación con método de Ackerman para el controlador

Al simular el control con método de asignación de polos por la fórmula de Ackerman, se logra un control que lleva a la referencia. Sin embargo, puede apreciarse que no se cumple el requisito de diseño para la acción de control y el ángulo de cabeceo. Las gráficas obtenidas son entonces:

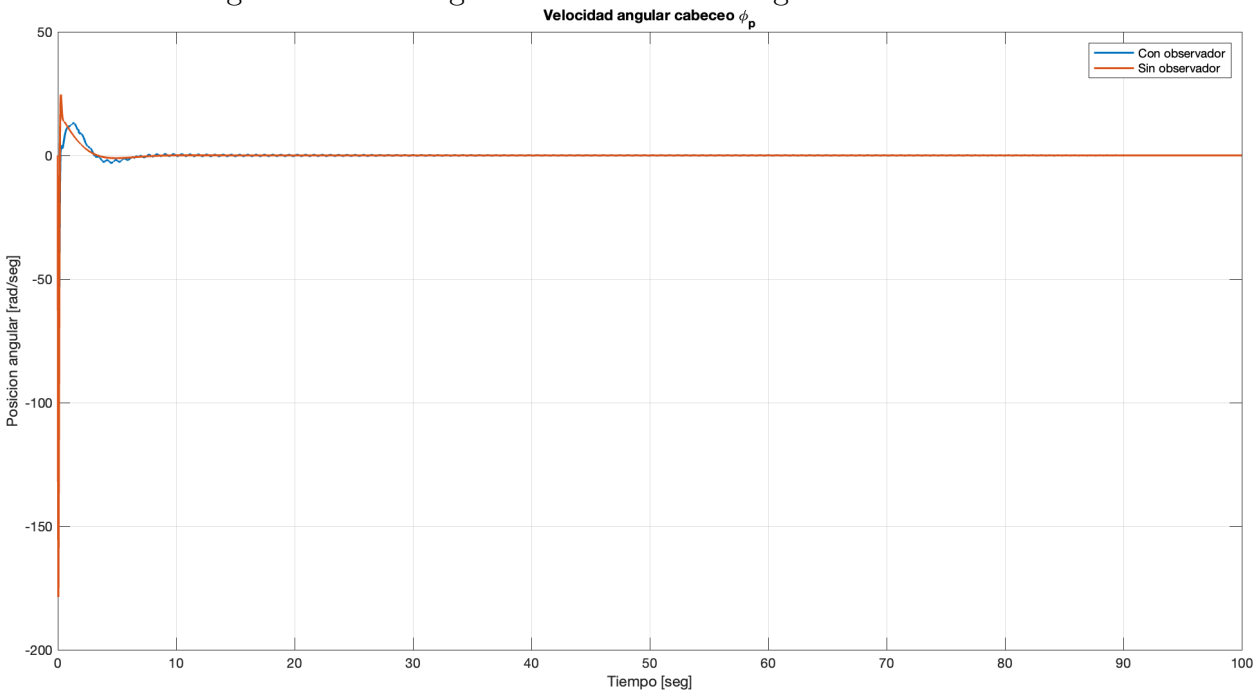


Se puede ver que, en el caso del controlador para estados medibles, se tiene una respuesta que cumple con los requisitos, sin embargo, la introducción de un observador hace que no se cumpla el requisito de diseño.

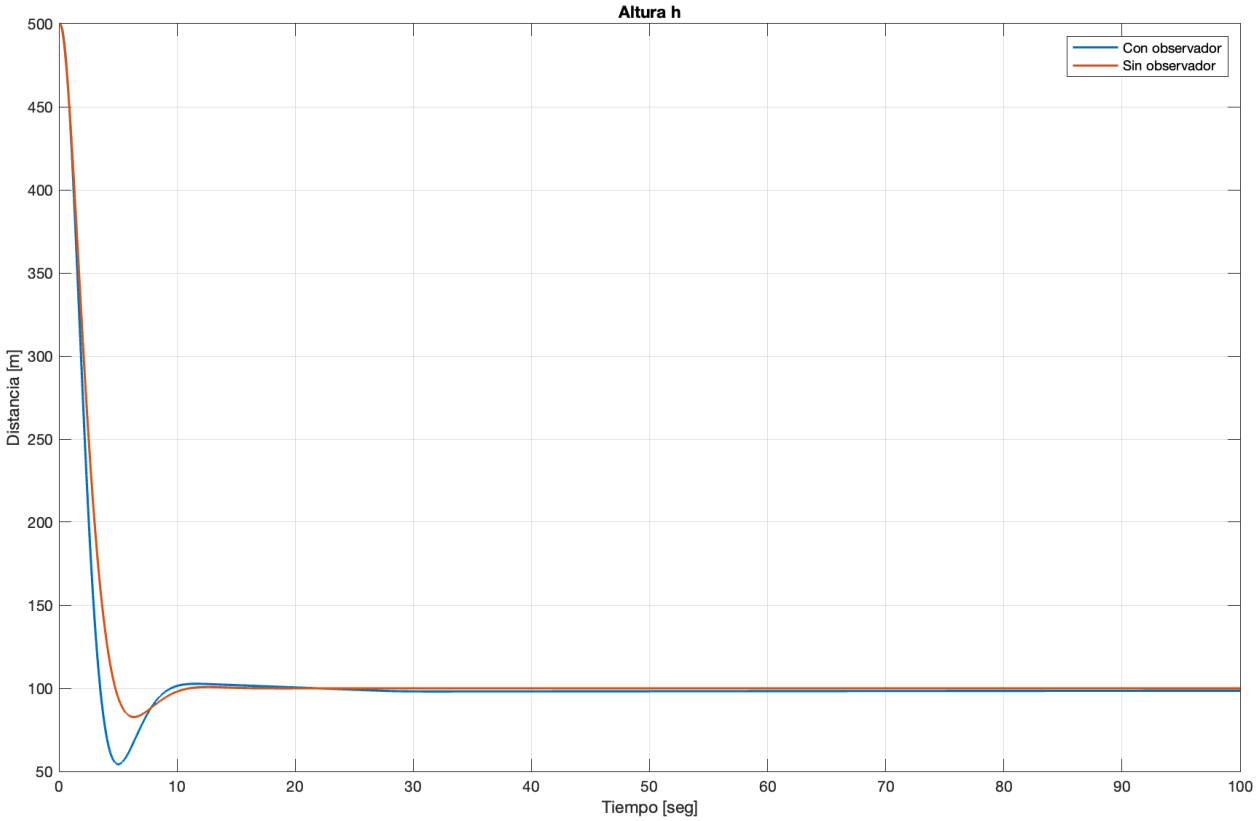


Ahora, en el ángulo de cabeceo, puede apreciarse que se supera el límite unitario, por lo que tampoco se está cumpliendo con el objetivo del cliente.

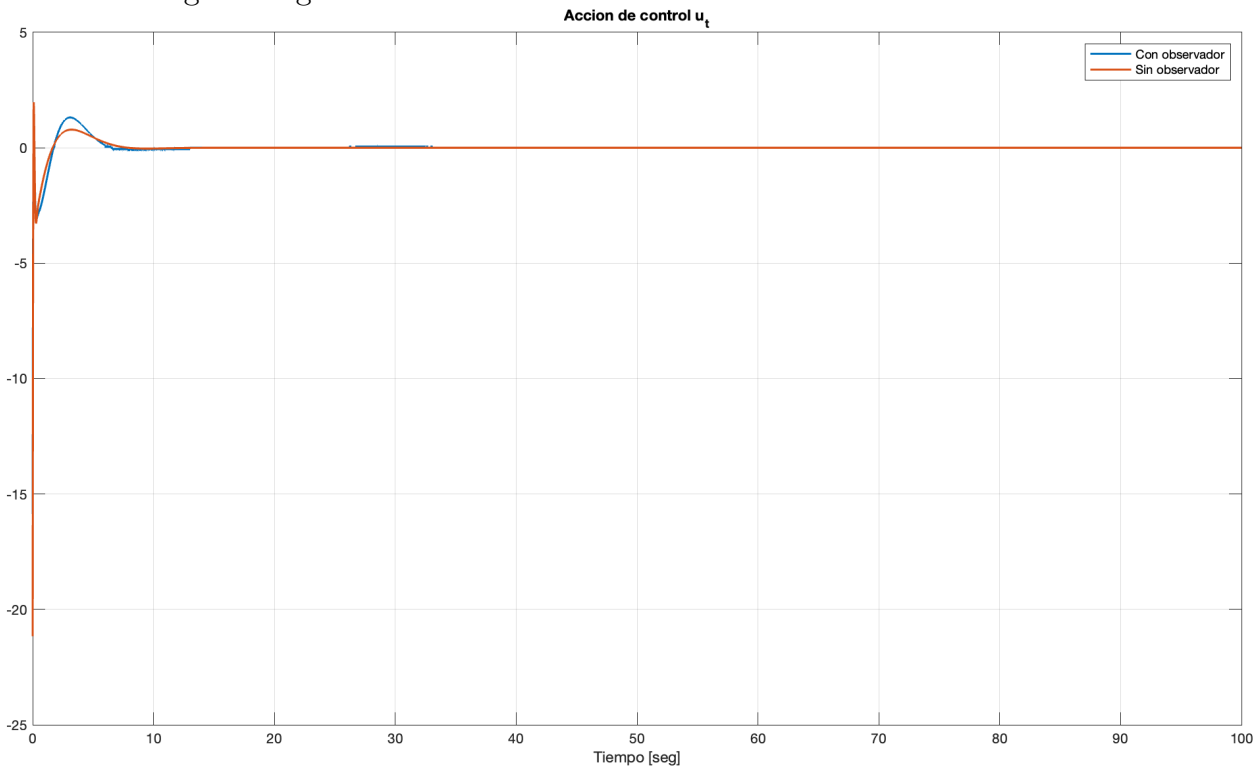
La velocidad angular de este ángulo de cabeceo es la siguiente:



La referencia de la altura puede alcanzarse perfectamente con el observador y el controlador sin el observador.



Cómo se dijo, la acción de control tampoco cumple con el objetivo de diseño. Eso puede verse en el siguiente gráfico.



Estas gráficas obtenidas de la simulación inicial se obtuvieron a partir de la altura inicial en 500 metros y con una referencia de 100 metros. Sin embargo, los requisitos de diseño no se cumplen con este controlador, se pretende de esta forma

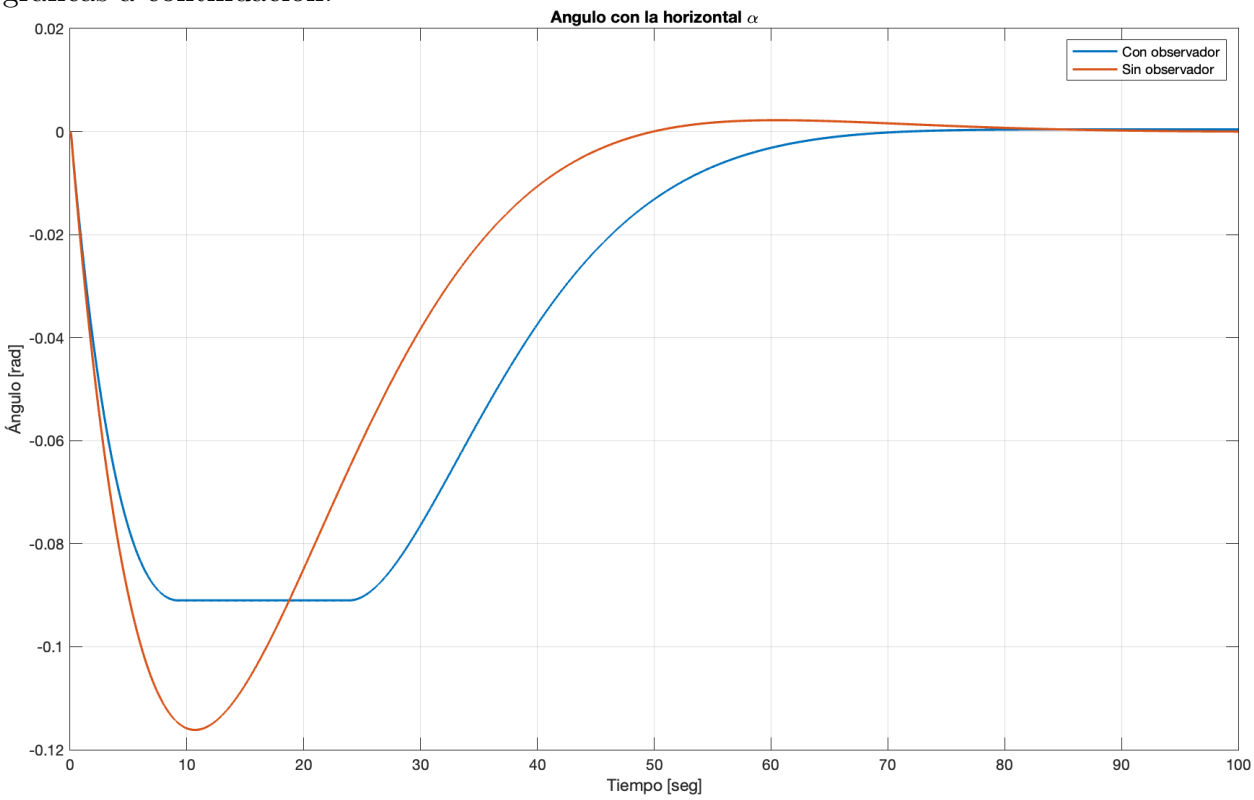
entonces, no llenar de gráficas el documento informe. Todas las gráficas se pueden ver en el Live Script presente en el repositorio GitHub del proyecto.

Simulación con el controlador LQR

El controlador LQR obtenido para el sistema tiene la siguiente forma:

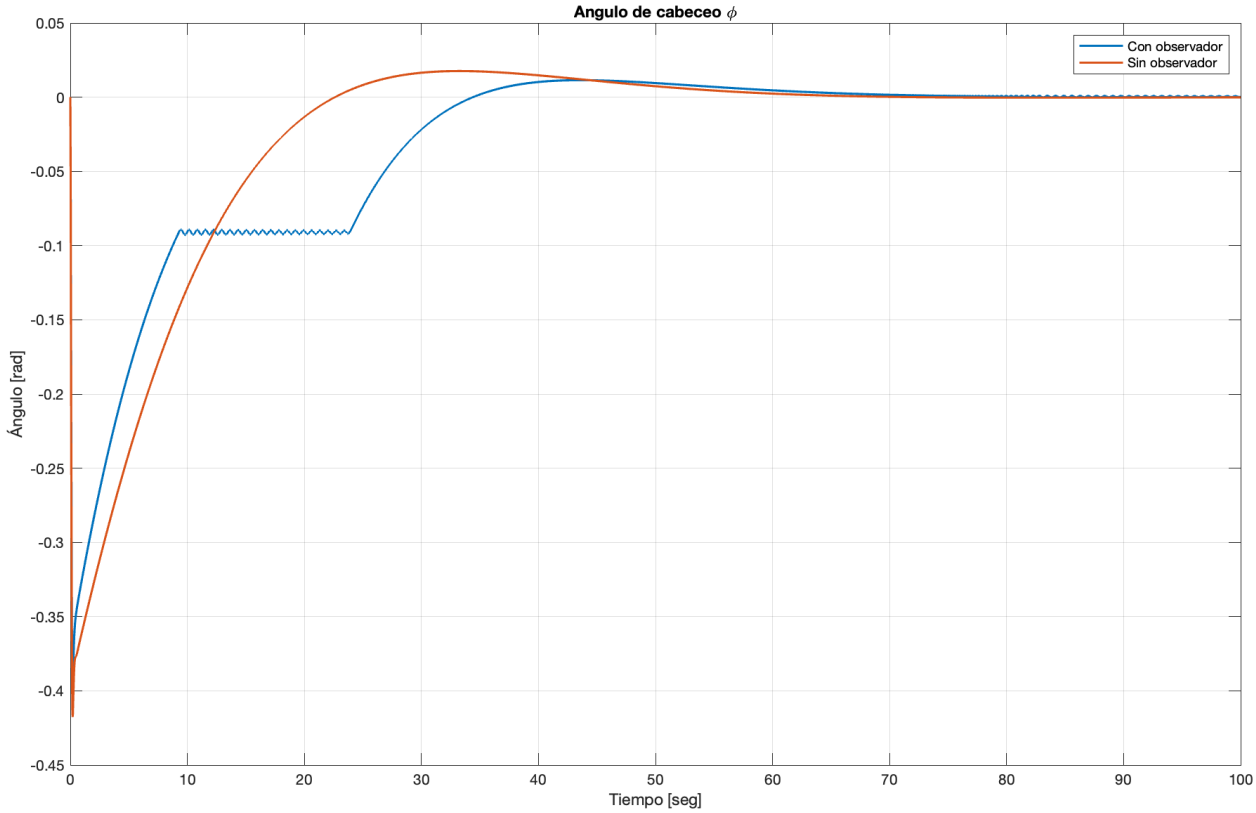
$$K = [1.5013 \quad 0.8255 \quad 0.0639 \quad 0.0010]$$

Nuevamente, de modo de no llenar de gráficas el documento, se adjuntan los resultados de la simulación de una única situación. Esta situación será la misma que se planteó en la primera simulación, de modo de poder comparar. De esa forma se obtienen las gráficas a continuación.

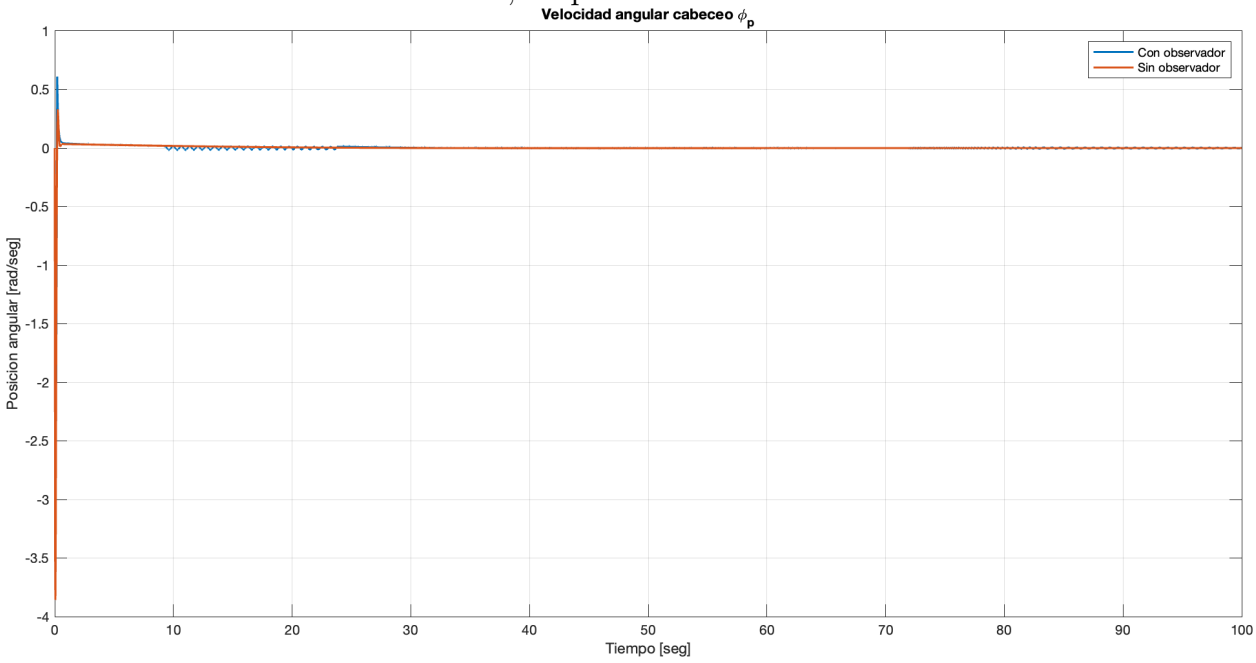


Se puede ver que, si bien el observador no sigue perfectamente la dinámica real, el sistema en ambos casos cumple con el objetivo de control, es decir, un ángulo menor a 1.

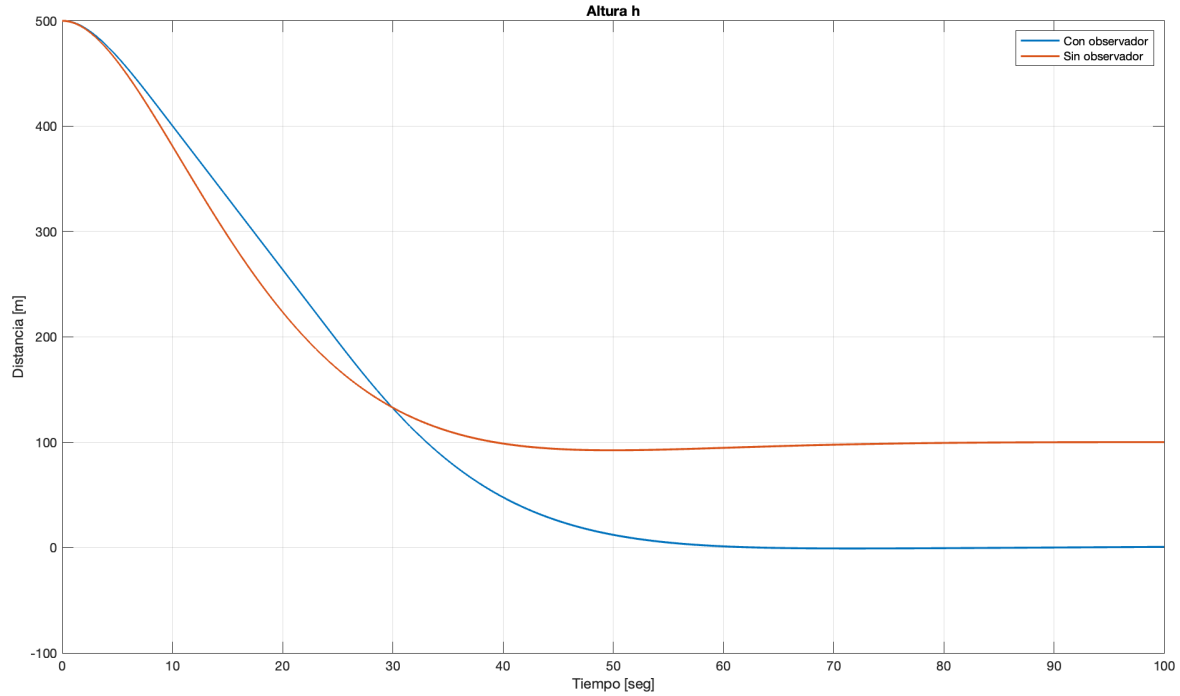
Ahora, para el caso del ángulo de cabeceo del avión, se obtiene una gráfica que cumple con el objetivo planteado por el cliente. Se podrá apreciar nuevamente un retraso en el seguimiento por parte del observador, pero la dinámica es acotada nuevamente en cuanto al ángulo.



La velocidad angular disminuye con respecto a la dinámica de la primera simulación. Si bien no se tenía un límite en este caso, una menor velocidad angular implica que el avión se mueve de forma más suave, lo que es deseado.

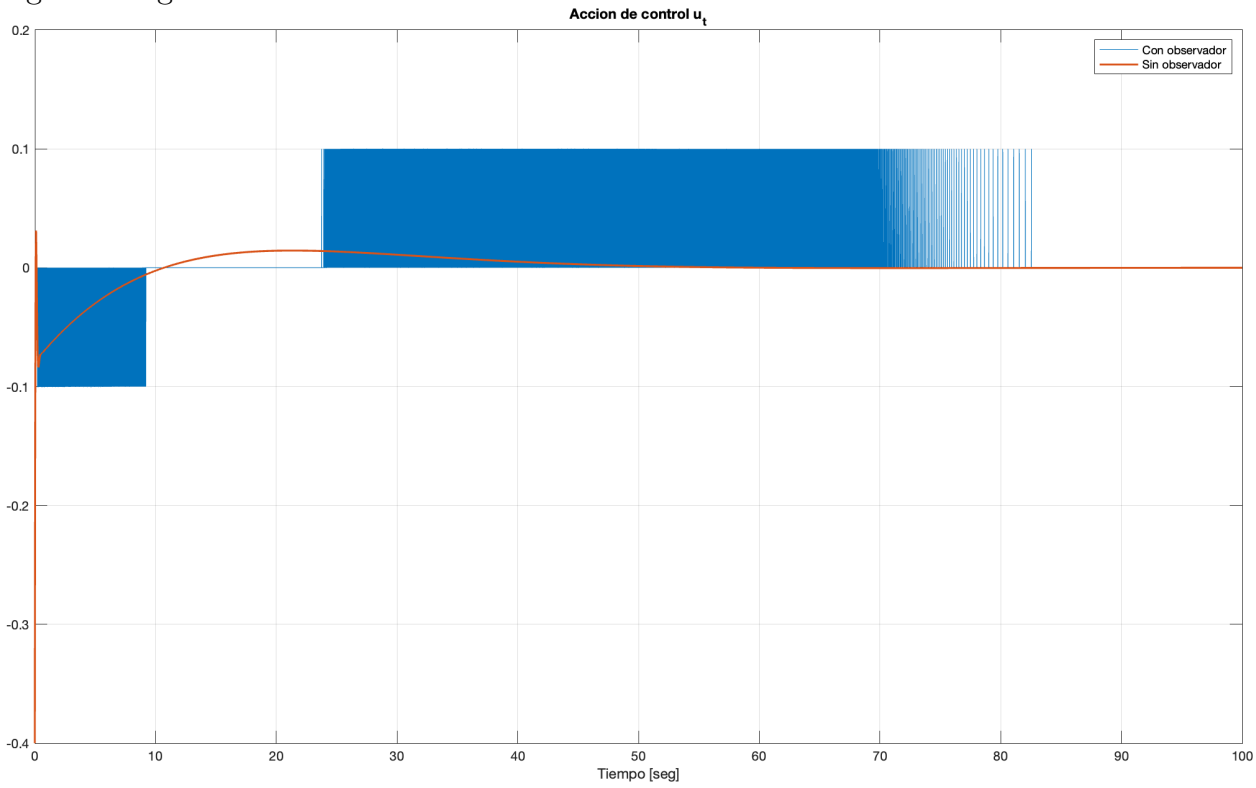


Donde aparece un problema, es en el observador de la altura. Como puede verse a continuación, existirá un error en estado estable de aproximadamente 100 metros. Si bien esto puede corregirse con un integrador, en la práctica no es problema porque el parámetro h es medible, lo que implica que no es necesario implementar el observador para ese estado.



Como las variables observadas son el ángulo con respecto a la horizontal y el ángulo de cabeceo, y, se cumple el requerimiento de control para esos estados, entonces se toma como válido el resultado obtenido.

Ahora, la acción de control cumple con el objetivo de diseño, puede verse en la siguiente figura.



Se puede apreciar también que, si bien se cumple el objetivo de control, en el momento en que se está en el transitorio, el observador posee una oscilación. La oscilación no es

deseada, pero aparece en las otras observaciones también. Lo importante a cumplir es la cota superior e inferior en el diseño.

Desarrollo del tercer caso

Para diseñar el controlador del sistema del péndulo invertido se hace uso de las matrices de la aproximación lineal. Sin embargo, las simulaciones deben hacerse de forma no lineal, es decir, aproximando Euler con las ecuaciones diferenciales.

En el sistema a controlar se tienen dos situaciones de masa. Para ello se planteará dos veces el sistema a controlar, modificando el parámetro de la masa en las ecuaciones de espacio de estados, y de esa forma se hará un switch de controladores en el instante en que se alcance la referencia de desplazamiento. Se hará uso de integradores nuevamente. El primer controlador tiene la siguiente forma:

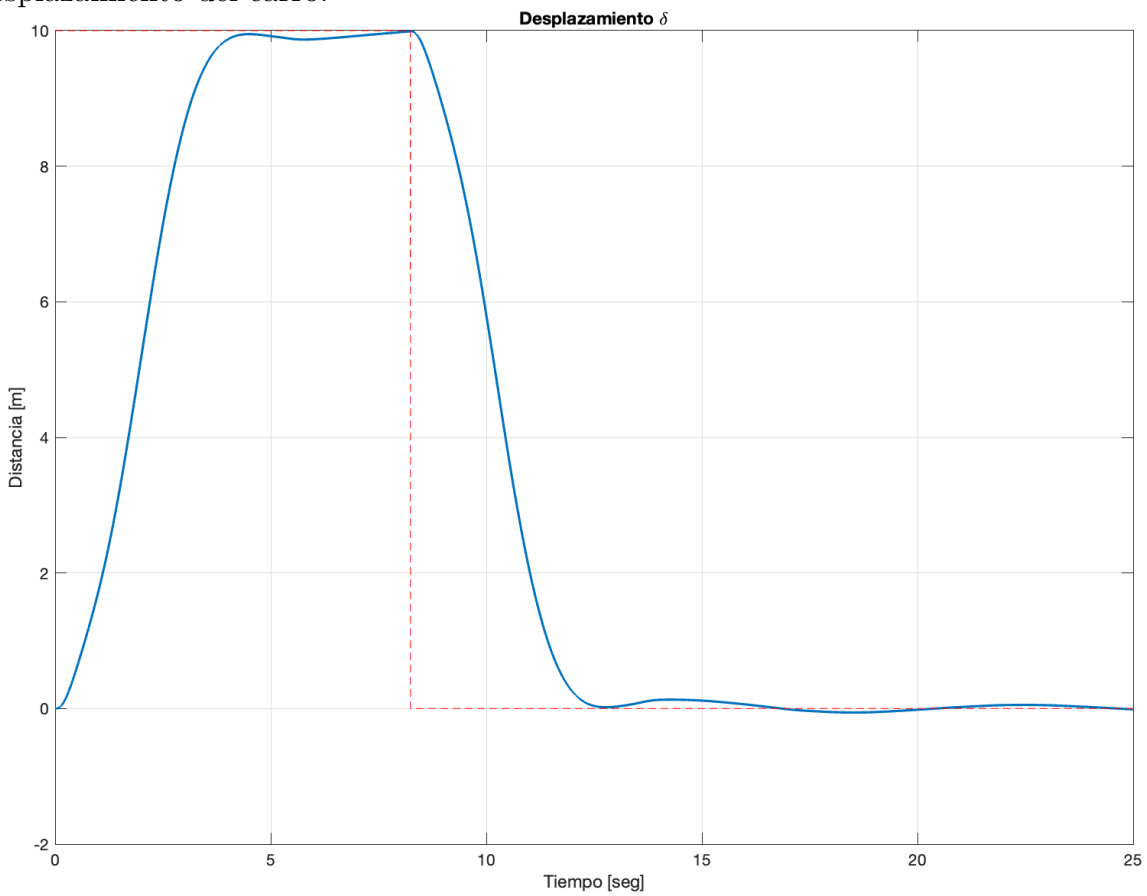
$$K_{amp} = [40.7242 \quad 29.6109 \quad 78.8271 \quad -12.0997 \quad -0.2047]$$

El segundo controlador tiene la forma:

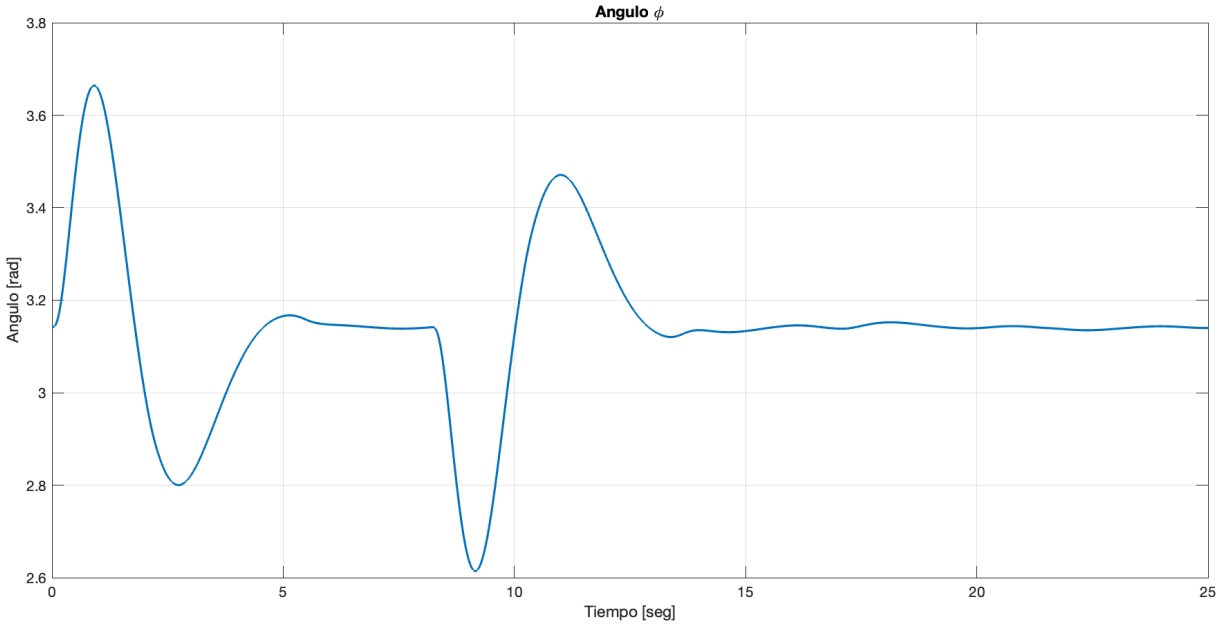
$$K_{amp} = [26.2505 \quad 19.7487 \quad 42.3199 \quad -6.6219 \quad -0.1292]$$

Resultados de la primera simulación

Con la primera simulación se obtienen la siguiente respuesta en cuanto a desplazamiento del carro.



Se aprecia que el sistema se establece en aproximadamente 13 segundos. La simulación es de 25 segundos solo a modo de verificar el correcto establecimiento de este. La siguiente gráfica muestra el ángulo del carro durante el transitorio.



Se puede ver que las oscilaciones con respecto a π son pequeñas, por lo que no existen grandes oscilaciones de la carga.

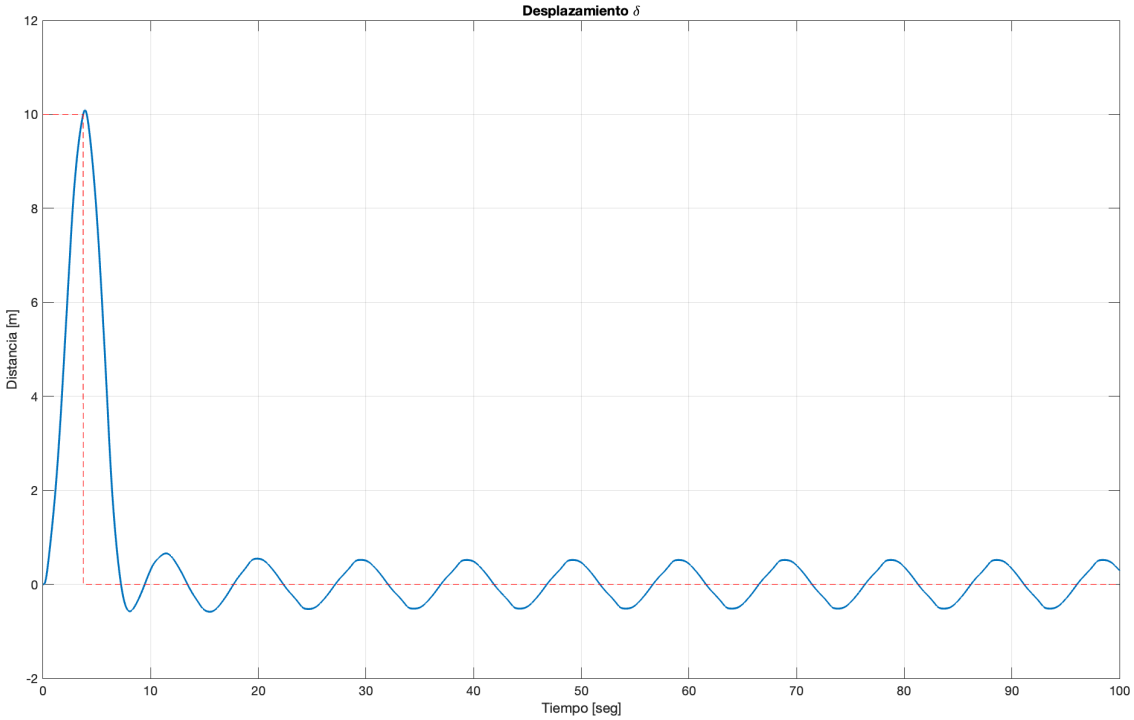
Este control obtenido es lo suficientemente viable como para lograr el objetivo de control planteado en la consigna.

Pero, además de lograr un control que lleve el sistema a las referencias, es necesario analizar el efecto de la variación en el período de muestreo del sistema.

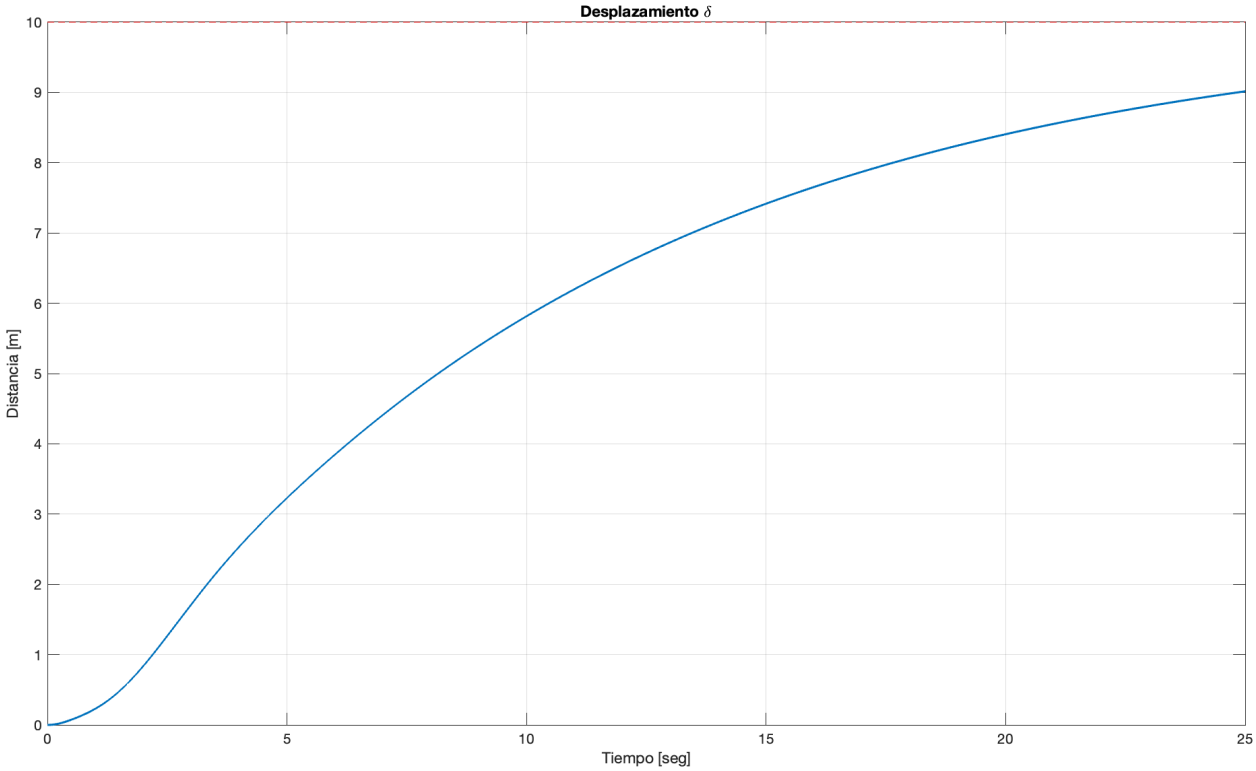
Fue posible verificar lo siguiente:

- La disminución del período de muestreo (acercándolo a Δt) hace que el sistema sea mucho más rápido, pues se acerca el control a uno en tiempo continuo, sin embargo, esto hace que, como los controladores no están diseñados para trabajar con ese T_S, existan oscilaciones, y los transitorios no cumplan con el objetivo de control. Esto puede corregirse rediseñando los controladores.
- Al aumentar el período de muestreo el sistema se vuelve muy lento, no pudiendo así traer mover la grúa, y trasladar la carga al origen.
- La no linealidad presente en el sistema genera una zona muerta en el actuador. Si esta linealidad está por debajo de 10, que es aproximadamente el valor pico de la acción de control, no afecta demasiado al transitorio del sistema. Sin embargo, al aumentar esta no linealidad superando la mitad del valor pico (5), comienzan a apreciarse ciclos límite. Estos ciclos aparecen porque el sistema planteado con esa no linealidad grande tienden a aproximarse a un control todo o nada con relé, por ejemplo, en donde el ancho de histéresis es demasiado grande. Este es un efecto no deseado.
- Se pudo determinar en base a las simulaciones que, el rango posible de período de muestreo es aproximadamente: $8.10^{-3} \leq T_S \leq 1.10^{-2}$ en segundos.

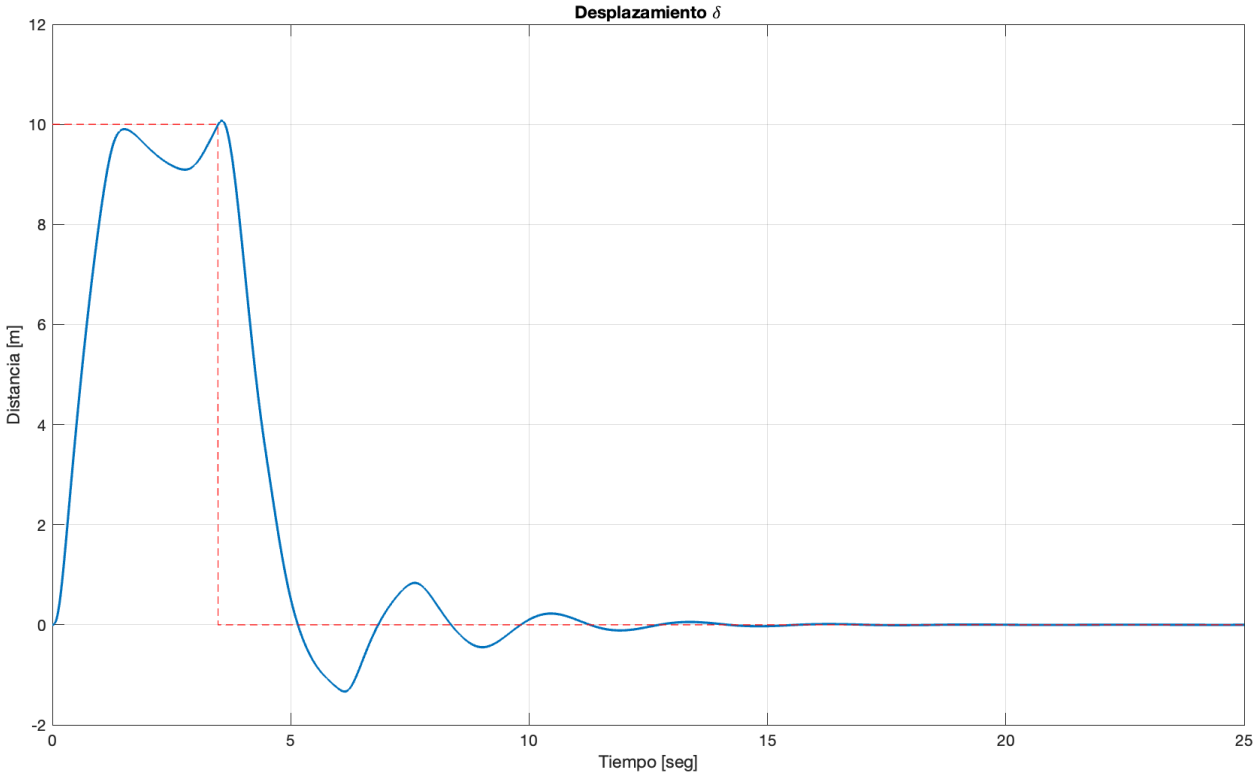
Se muestran los efectos a continuación:



En la gráfica superior se aprecia el ciclo límite introducido por el aumento de la no linealidad del sistema actuador. Ahora, se muestra el efecto del aumento del período de muestreo.



Como se dijo anteriormente, el Sistema se enlentece, no logrando así la referencia intermedia dentro de un tiempo de simulación adecuado. En cambio, si se disminuyera el período de muestreo, ocurre lo siguiente.


Desplazamiento δ
Distancia [m]
Tiempo [seg]
12
10
8
6
4
2
0
-2
0
5
10
15
20
25

Observaciones

Generalidades

- Si se pretende trabajar con sucesivas modificaciones de valores en el sistema, como por ejemplo lo son las alinealidades, períodos de muestreo o masas, se puede facilitar mucho el proceso si se trabaja en Live Script de Matlab con sliders.
- El período de muestreo de un sistema siempre debe ser menor al período de integración de Euler, de modo que el sistema siga evolucionando, aún cuando no está siendo muestreado.
- El controlador para sistemas no lineales, debe diseñarse partiendo de un modelo linealizado, así se pueden utilizar las matrices de estado, y con ello plantear una realimentación de estados en el control.
- El profesor Laboret nombró en clase que la función de **acker** de Matlab es más robusta que **place** al momento de diseñar controladores por asignación de polos, por lo que se la utilizó en este trabajo.
- Es importante tener una noción de los valores de las constantes intervinientes en un sistema, de modo de poder detectar un valor inusual, y consultar con bibliografía si es factible. En este trabajo pudo simularse desde cero con una constante $B_m = 82$. Sin embargo, la carga computacional, sumado a que ya se tenía como referencia un set de mediciones sobre el motor a comparar, hicieron que lo óptimo sea disminuir el valor de la constante en cuestión. Con ello se logró un control comparable al de la consigna. Al usar 82 como valor, el sistema tardaba mucho en establecerse, y era necesario aumentar el tiempo de simulación, lo que no hacía que la respuesta obtenida sea comparable a la que se tenía como dato desde un principio.

Indicadores de logro

El alumno considera alcanzados los siguientes indicadores de logro planteados en el aula virtual:

- 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.

- Diseñar controladores de procesos considerando la dinámica y la magnitud de las acciones de control.
- Diseñar controladores para sistemas no lineales.
- Relacionar controladores digitales con el muestreo y la reconstrucción de señales de procesos de tiempo continuo.
- Determinar funciones de transferencia discretas de sistemas con muestreadores y retentores de orden cero.
- Comprender la equivalencia entre planos s y z y los lugares de tiempo de establecimiento, frecuencia natural amortiguada y no amortiguada y coeficiente de amortiguamiento constante.
- Digitalizar controladores de tiempo continuo.
- Expresar el modelo lineal multivariable de un proceso real en un punto de operación.
- Comprender conceptos de equilibrio, ciclo límite y espacio de estados.
- Analizar sistemas lineales a tramos o de estructura variable.

Conclusiones

Luego de la realización de esta actividad práctica es posible decir que se adquirió un conocimiento integral de los temas planteados como indicadores de logro. En este caso se hizo hincapié en el control de sistemas en tiempo discreto, a partir de su representación continua. Se pudo comprender cómo plantear límites en las acciones de control o dinámicas de ciertos sistemas a partir de los controladores (en este caso LQR).

Durante el desarrollo del trabajo surgieron ciertos inconvenientes en cuanto a carga computacional de algunos cálculos, pero se entiende desde el lado del alumno que fueron resueltos, y se pudo lograr el resultado esperado tanto en simulaciones como en conocimiento adquirido.

Fue posible en este trabajo comparar nuevamente la performance de controladores por fórmula de Ackerman o asignación directa de polos con LQR. Se puede ver que sintonizando ciertos valores en R o Q para estos últimos controladores, se puede afectar de una forma u otra el comportamiento del sistema, pudiendo lograr con unas pocas iteraciones e interpretación del efecto de los pesos un sistema que se ajuste a los requerimientos del cliente.

Finalmente, a modo de resumen, lo principal que se pudo comprender en este trabajo práctico fue:

- Discretización de sistemas reales.
- Relación entre períodos de integración y muestreo para controladores reales.
- Diseño de controladores por LQR y DLQR.
- Implementación de integradores con matrices ampliadas.
- Desarrollo e implementación de observadores.
- Relación entre sistemas continuos y su discretización para controlarlos.

Anexo: códigos de simulación

Código del modelo del motor de CC

```
clear all;
close all;
clc;
%%
% Primero se abre el archivo de Excel con las curvas medidas, de forma de poder
% comparar posteriormente los resultados obtenidos.

file = '/Users/federicovillar/Documents/GitHub/TPsControl2/actividad3/motor.xls';
sheet = 'Hoja1';
[num, txt, raw] = xlsread(file, sheet);
timeExcel = num(:, 1);
angExcel = num(:, 2);
omegaExcel = num(:, 3);
iaExcel = num(:, 4);
vExcel = num(:, 5);
torqueExcel = num(:, 6);
%%
% Se declaran ahora los parametros del motor.

Laa = 0.56e-3;
J = 0.0019;
Ra = 1.35;
Bm = 0.000792;
Ki = 0.1;
Km = 0.1;
%%
% El modelado en espacio de estados para el sistema en tiempo continuo es:

A = [-Ra/Laa -Km/Laa 0; Ki/J -Bm/J 0; 0 1 0]
B = [1/Laa; 0; 0]
C = [0 0 1; 0 1 0]
D = [0]
sys = ss(A,B,C,D)
pole(sys)
%%
% Los tiempos de integración y muestreo son:

Ts = 1e-4;
h = 1e-5;
%%
% Se discretiza el sistema.

dSys = c2d(sys,Ts,'zoh')
A = dSys.A;
B = dSys.B;
C = dSys.C;
%%
% Para eliminar el error en estado estable se amplian las matrices y se implementa
% un integrador.

Aamp = [A,zeros(3,1);-C(1,:)*A, 1]
Bamp = [B;-C(1,:)*B]
Camp = [C(1,:) 0]
%%
% El controlador DLQR queda declarado como a continuacion:

Q = diag([1 1 1 0.1]);
R = 0.001;
Kamp = dlqr(Aamp,Bamp,Q,R)
K = Kamp(1:3)
Ki = -Kamp(4)
%%
% Como no puede medirse la corriente, se implementa un observador para controlarla.

```

```
Ao = A';
Bo = C';
Co = B';
Qo = 1*diag([10 10 10]);
Ro = 0.1;
Ko = dlqr(Ao,Bo,Qo,Ro)
%%
% Para poder simular el sistema, los parametros temporales son:

T = 10;
h = 1e-5;
Kmax = T/Ts;
t = 0:h:(T);
%%
% La referencia y el torque aplicado, son respectivamente:

thetaRef = (pi/2)*square(2*pi*(1/10)*t);
torqueRef = ((1.5)/2)*square(2*pi*(1/10)*t)+((1.5)/2);
%%
% La alinealidad del controlador como zona muerta es:

deadZone = 1.2
%%
% Luego, vector de estados y vector observado.

x(1,1) = 0;
x(2,1) = 0;
x(3,1) = 0;
x(4,1) = 0;
xObs(1,1) = 0;
xObs(2,1) = 0;
xObs(3,1) = 0;
%%
% Para simular:

xTs = x((1:3),1);
vTs = x(4,1);
z = 1;
for i=1:1:Kmax
    xK = xTs;
    vK = vTs;
    u = -K(1:3)*xObs(1:3)+Ki*vK;
    uPrev = u;
%%
% El actuador llega a la alinealidad.

    if abs(u)<deadZone
        u = 0;
    else
        u = sign(u)*(abs(u)-deadZone);
    end
    yS = C*x(1:3,z);
    for j = 1:1:Ts/h
        prevU(z) = uPrev;
        x1P = -Ra*x(1,z)/Laa-Km*x(2,z)/Laa+u/Laa;
        x2P = Ki*x(1,z)/J-Bm*x(2,z)/J-torqueRef(z)/J;
        x3P = x(2,z);
        xP = [x1P; x2P; x3P];
        x((1:3),z+1) = x((1:3),z)+h*xP;
        z = z+1;
    end
    yHat = C*xObs;
    xObs = A*xObs+B*u+Ko'*(yS-yHat);
    vTs = vTs+thetaRef(z)-C(1,:)*xTs;
    xTs = x((1:3),z);
end
%%
% Para comparar, se obtienen las graficas:

plot(t(1:length(x(1,:))),x(1,:),'LineWidth',1.5);
```

```
hold on;
plot(timeExcel,iaExcel,'r');
hold off
grid;
xlim([0 T]);
legend('Corriente obtenida','Corriente dato');
title('Corriente de armadura i_a(t)');
xlabel('Tiempo [seg]');
ylabel('Corriente [A]');
plot(t(1:length(x(2,:))),x(2,:),'LineWidth',1.5);
hold on;
plot(timeExcel,omegaExcel,'r');
hold off;
grid;
xlim([0 T]);
legend('Velocidad angular obtenida','Velocidad angular dato');
title('Velocidad angular \omega');
xlabel('Tiempo [seg]');
ylabel('\omega [rad/seg]');
plot(t(1:length(x(3,:))),x(3,:),'LineWidth',1.5);
hold on;
plot(timeExcel,angExcel,'r');
plot(t(1:length(thetaRef)),thetaRef,'k--','LineWidth',0.5);
hold off;
grid;
xlim([0 T]);
legend('Angulo obtenido','Angulo dato','Referencia');
title('Posicion angular \theta');
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
%%
% Finalmente, se compara la accion de control.

dead = ones(2,length(prevU));
dead(1,:) = dead(1,:)* deadZone;
dead(2,:) = dead(2,:)* (-1)*deadZone;
plot(t(1:length(prevU)),prevU,'LineWidth',1.5);
hold on;
plot(timeExcel,vExcel,'r');
plot(t(1:length(prevU)),dead,'k--');
hold off
xlim([0 T]);
grid;
title('Acción de control');
xlabel('Tiempo [seg]');
ylabel('Tension [V]');
legend('Accion de control obtenida','Accion de control dato','Umbral de zona muerta');
%%
% El plano de fases es el siguiente:

plot(x(3,:),x(2,:),'LineWidth',1.5);
hold on;
plot(angExcel,omegaExcel,'r');
hold off;
grid
xlabel('Posicion angular');
ylabel('Velocidad angular');
title('Planos de fase');
legend('Plano obtenido', 'Plano dato')

```

Código del modelo de avión

```
clear all;
close all;
clc
%%
% Primero se declaran los parámetros del sistema.
```

```
 6.
 7. a = 0.07;
 8. b = 5;
 9. c = 150;
10. w = 9;
11. %%
12. % Su modelado en el espacio de estados es el siguiente:
13.
14. A = [-a a 0 0; 0 0 1 0; w^2 -w^2 0 0; c 0 0 0]
15. B = [0; 0; b*w^2; 0]
16. C = [0 0 0 1;0 1 0 0]
17. D = [0]
18. %%
19. % La consigna pide unos determinados polos a lazo cerrado:
20.
21. p1 = -15+15i;
22. p2 = -15-15i;
23. p3 = -0.5+0.5i;
24. p4 = -0.5-0.5i;
25. poles = [p1 p2 p3 p4];
26. K = acker(A,B,poles)
27. Ap = A-B*K
28. sysCL = ss(Ap,B,C,D)
29. pole(sysCL)
30. %%
31. % La ganancia de preamplificación:
32.
33. G = -inv(C(1,:)*inv(Ap)*B)
34. %%
35. % Ahora, se plantea el siguiente observador:
36.
37. Ao = A'
38. Bo = C'
39. Co = B'
40. Qo = diag([100 10000 100 100000]);
41. Ro = 1000000;
42. Ko = lqr(Ao,Bo,Qo,Ro)
43. %%
44. % Lo siguiente es para poder simular:
45.
46. T = 100;
47. h = 1e-4;
48. t = 0:h:(T-h);
49. ref = 100;
50. alpha(1) = 0;
51. phi(1) = 0;
52. phiP(1) = 0;
53. high(1) = 500;
54. u(1) = 0;
55. uu(1) = 0;
56. stateVector = [alpha(1);phi(1);phiP(1);high(1)];
57. xOp = [0;0;0;0];
58. xOp = [alpha(1);phi(1);phiP(1);high(1)];
59. alphaO(1) = 0;
60. phiO(1) = 0;
61. phiPO(1) = 0;
62. highO(1) = 400;
63. xObs = [alphaO(1);phiO(1);phiPO(1); highO(1)];
64. obsStateVector = [alphaO(1);phiO(1);phiPO(1);highO(1)];
65. deadZone = 0.1
66. %%
67. % Ahora, la simulación con observador:
68.
69. for i=1:T/h
70.     u(i) = -K*obsStateVector+G*ref;
71.     if (abs(u(i)) < deadZone)
72.         uu(i)=0;
73.     else
74.         uu(i)=sign(u(i))*(abs(u(i)));
75.     end
```

```
    alpha(i) = xOp(1);
    phi(i) = xOp(2);
    phiP(i) = xOp(3);
    high(i) = xOp(4);
    xP = A*xOp+B*uu(i);
    xOp = xOp+h*xP;
    alphaO(i)= xObs(1);
    phiO(i)= xObs(2);
    phiPO(i)= xObs(3);
    highO(i)= xObs(4);
    yOutObs = C*obsStateVector;
    yOut   = C*stateVector;
    xPrevP = A*xObs+B*uu(i)+Ko'*(yOut-yOutObs);
    xObs = xObs + xPrevP*h;
    stateVector=[alpha(i);phi(i);phiP(i);high(i)];
    obsStateVector=[alphaO(i);phiO(i);phiPO(i);highO(i)];
end
%%
% La simulación sin observador:

uSO(1)=0;
stateVectorSO=[alpha(1);phi(1);phiP(1);high(1)];
xSO=[alpha(1);phi(1);phiP(1); high(1)];
for i=1:T/h
    uSO(i) = -K*stateVectorSO+G*ref;
    alphaSO(i) = xSO(1);
    phiSO(i) = xSO(2);
    phiPSO(i) = xSO(3);
    highSO(i) = xSO(4);
    xp_so = A*xSO+B*uSO(i);
    xSO = xSO+h*xp_so;
    stateVectorSO = [alphaSO(i);phiSO(i);phiPSO(i);highSO(i)];
end
%%
% Se obtienen las siguientes gráficas:

plot(t,alpha,'LineWidth',1.5);
hold on;
plot(t,alphaSO,'LineWidth',1.5);
hold off
title('Angulo con la horizontal \alpha');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
plot(t,phi,'LineWidth',1.5);
hold on;
plot(t,phiSO,'LineWidth',1.5);
hold off;
title('Angulo de cabeceo \phi');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
plot(t,phiP,'LineWidth',1.5);
hold on;
plot(t,phiPSO,'LineWidth',1.5);
hold off
title('Velocidad angular cabeceo \phi_p');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Posicion angular [rad/seg]');
grid;
plot(t,high,'LineWidth',1.5);
hold on;
plot(t,highSO,'LineWidth',1.5);
hold off
title('Altura h');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
```

```
ylabel('Distancia [m]');
grid;
plot(t,uu,'LineWidth',1.5);
hold on;
plot(t,uSO,'LineWidth',1.5);
hold off
title('Accion de control u_t');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
grid;
%%
% Se aprecia en las figuras que se cumple el requerimiento de la magnitud en
% los ángulos, pero no en la acción de control, por lo que se implementará un
% controlador del tipo LQR.
%%
% Ahora, las simulaciones con el nuevo controlador:

clear all;
close all;
clc
%%
% Primero se declaran los parámetros del sistema.

a = 0.07;
b = 5;
c = 150;
w = 9;
%%
% Su modelado en el espacio de estados es el siguiente:

A = [-a a 0 0; 0 0 1 0; w^2 -w^2 0 0; c 0 0 0]
B = [0; 0; b*w^2; 0]
C = [0 0 0 1;0 1 0 0]
D = [0];
%%
% Diseño del LQR:

Q = diag([1 1000000 1 1]);
R = 1000000;
K = lqr(A,B,Q,R)
%%
% La ganancia de preamplificación:

G = -inv(C(1,:)*inv(A-B*K)*B)
%%
% Ahora, se plantea el siguiente observador:

Ao = A'
Bo = C'
Co = B'
Qo = diag([100 1000 100 1000]);
Ro = 5000000;
Ko = lqr(Ao,Bo,Qo,Ro);
%%
% Lo siguiente es para poder simular:

T = 100;
h = 1e-4;
t = 0:h:(T-h);
ref = 100;
alpha(1) = 0;
phi(1) = 0;
phiP(1) = 0;
high(1) = 500;
u(1) = 0;
uu(1) = 0;
stateVector = [alpha(1);phi(1);phiP(1);high(1)];
xOp = [0;0;0;0];
xOp = [alpha(1);phi(1);phiP(1);high(1)];
alphaO(1) = 0;
```

```
phiO(1) = 0;
phiPO(1) = 0;
highO(1) = 500;
xObs = [alphaO(1);phiO(1);phiPO(1); highO(1)];
obsStateVector = [alphaO(1);phiO(1);phiPO(1);highO(1)];
deadZone = 0.1;
%%
% Ahora, la simulación con observador:

for i=1:T/h
    u(i) = -K*obsStateVector+G*ref;
    if (abs(u(i)) < deadZone)
        uu(i)=0;
    else
        uu(i)=sign(u(i))*(abs(u(i)));
    end
    alpha(i) = xOp(1);
    phi(i) = xOp(2);
    phiP(i) = xOp(3);
    high(i) = xOp(4);
    xP = A*xOp+B*uu(i);
    xOp = xOp+h*xP;
    alphaO(i)= xObs(1);
    phiO(i)= xObs(2);
    phiPO(i)= xObs(3);
    highO(i)= xObs(4);
    yOutObs = C*obsStateVector;
    yOut   = C*stateVector;
    xPrevP = A*xObs+B*uu(i)+Ko'*(yOut-yOutObs);
    xObs = xObs + xPrevP*h;
    stateVector=[alpha(i);phi(i);phiP(i);high(i)];
    obsStateVector=[alphaO(i);phiO(i);phiPO(i);highO(i)];
end
%%
% La simulación sin observador:

uSO(1)=0;
stateVectorSO=[alpha(1);phi(1);phiP(1);high(1)];
xSO=[alpha(1);phi(1);phiP(1); high(1)];
for i=1:T/h
    uSO(i) = -K*stateVectorSO+G*ref;
    alphaSO(i) = xSO(1);
    phiSO(i) = xSO(2);
    phiPSO(i) = xSO(3);
    highSO(i) = xSO(4);
    xp_so = A*xSO+B*uSO(i);
    xSO = xSO+h*xp_so;
    stateVectorSO = [alphaSO(i);phiSO(i);phiPSO(i);highSO(i)];
end
%%
% Se obtienen las siguientes gráficas:

plot(t,alpha,'LineWidth',1.5);
hold on;
plot(t,alphaSO,'LineWidth',1.5);
hold off
title('Angulo con la horizontal \alpha');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
plot(t,phi,'LineWidth',1.5);
hold on;
plot(t,phiSO,'LineWidth',1.5);
hold off;
title('Angulo de cabeceo \phi');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
```

```
plot(t,phiP,'LineWidth',1.5);
hold on;
plot(t,phiPSO,'LineWidth',1.5);
hold off
title('Velocidad angular cabeceo \phi_p');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Posicion angular [rad/seg]');
grid;
plot(t,high,'LineWidth',1.5);
hold on;
plot(t,highSO,'LineWidth',1.5);
hold off
title('Altura h');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Distancia [m]');
grid;
plot(t,uu,'LineWidth',0.5);
hold on;
plot(t,uSO,'LineWidth',1.5);
hold off
title('Accion de control u_t');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
grid;
%%
% Ahora mejora la dinámica, sin embargo, existe un error en estado estacionario
% para el observador de la altura. Sin embargo, esto no es problema por el hecho
% de que la altura es medible, no observada. Las variables observadas en la realidad
% son $\alpha \;$ y $\phi \;$.
%
% Se procede con estos controladores entonces, a simular las otras 3 situaciones
% de referencia y altura inicial.

clear all;
close all;
clc
a = 0.07;
b = 5;
c = 150;
w = 9;
A = [-a a 0 0; 0 0 1 0; w^2 -w^2 0 0; c 0 0 0];
B = [0; 0; b*w^2; 0];
C = [0 0 0 1;0 1 0 0];
D = [0];
Q = diag([1 1000000 1 1]);
R = 1000000;
K = lqr(A,B,Q,R);
G = -inv(C(1,:)*inv(A-B*K)*B);
Ao = A';
Bo = C';
Co = B';
Qo = diag([100 1000 100 1000]);
Ro = 10000000000;
Ko = lqr(Ao,Bo,Qo,Ro);
T = 100;
h = 1e-4;
t = 0:h:(T-h);
ref = -100;
alpha(1) = 0;
phi(1) = 0;
phiP(1) = 0;
high(1) = -500;
u(1) = 0;
uu(1) = 0;
stateVector = [alpha(1);phi(1);phiP(1);high(1)];
xOp = [0;0;0;0];
xOp = [alpha(1);phi(1);phiP(1);high(1)];
alphaO(1) = 0;
```

```
phiO(1) = 0;
phiPO(1) = 0;
highO(1) = 400;
xObs = [alphaO(1);phiO(1);phiPO(1); highO(1)];
obsStateVector = [alphaO(1);phiO(1);phiPO(1);highO(1)];
deadZone = 0.1;
for i=1:T/h
    u(i) = -K*obsStateVector+G*ref;
    if (abs(u(i)) < deadZone)
        uu(i)=0;
    else
        uu(i)=sign(u(i))*(abs(u(i)));
    end
    alpha(i) = xOp(1);
    phi(i) = xOp(2);
    phiP(i) = xOp(3);
    high(i) = xOp(4);
    xP = A*xOp+B*uu(i);
    xOp = xOp+h*xP;
    alphaO(i)= xObs(1);
    phiO(i)= xObs(2);
    phiPO(i)= xObs(3);
    highO(i)= xObs(4);
    yOutObs = C*obsStateVector;
    yOut   = C*stateVector;
    xPrevP = A*xObs+B*uu(i)+Ko'*(yOut-yOutObs);
    xObs = xObs + xPrevP*h;
    stateVector=[alpha(i);phi(i);phiP(i);high(i)];
    obsStateVector=[alphaO(i);phiO(i);phiPO(i);highO(i)];
end
uSO(1)=0;
stateVectorSO=[alpha(1);phi(1);phiP(1);high(1)];
xSO=[alpha(1);phi(1);phiP(1); high(1)];
for i=1:T/h
    uSO(i) = -K*stateVectorSO+G*ref;
    alphaSO(i) = xSO(1);
    phiSO(i) = xSO(2);
    phiPSO(i) = xSO(3);
    highSO(i) = xSO(4);
    xp_so = A*xSO+B*uSO(i);
    xSO = xSO+h*xp_so;
    stateVectorSO = [alphaSO(i);phiSO(i);phiPSO(i);highSO(i)];
end
%%
% Se obtienen las siguientes gráficas:

plot(t,alpha,'LineWidth',1.5);
hold on;
plot(t,alphaSO,'LineWidth',1.5);
hold off
title('Angulo con la horizontal \alpha');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
plot(t,phi,'LineWidth',1.5);
hold on;
plot(t,phiSO,'LineWidth',1.5);
hold off;
title('Angulo de cabeceo \phi');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
plot(t,phiP,'LineWidth',1.5);
hold on;
plot(t,phiPSO,'LineWidth',1.5);
hold off
title('Velocidad angular cabeceo \phi_p');
legend('Con observador','Sin observador')
```

```
xlabel('Tiempo [seg]');
ylabel('Posicion angular [rad/seg]');
grid;
plot(t,high,'LineWidth',1.5);
hold on;
plot(t,highSO,'LineWidth',1.5);
hold off
title('Altura h');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Distancia [m]');
grid;
plot(t,uu,'LineWidth',1.5);
hold on;
plot(t,uSO,'LineWidth',1.5);
hold off
title('Accion de control u_t');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
grid;
%%
%

clear all;
close all;
clc
a = 0.07;
b = 5;
c = 150;
w = 9;
A = [-a a 0 0; 0 0 1 0; w^2 -w^2 0 0; c 0 0 0];
B = [0; 0; b*w^2; 0];
C = [0 0 0 1;0 1 0 0];
D = [0];
Q = diag([1 1000000 1 1]);
R = 1000000
K = lqr(A,B,Q,R);
G = -inv(C(1,:)*inv(A-B*K)*B);
Ao = A';
Bo = C';
Co = B';
Qo = diag([100 1000 100 1000]);
Ro = 10000000000;
Ko = lqr(Ao,Bo,Qo,Ro);
T = 100;
h = 1e-4;
t = 0:h:(T-h);
ref = -100;
alpha(1) = 0;
phi(1) = 0;
phiP(1) = 0;
high(1) = 500;
u(1) = 0;
uu(1) = 0;
stateVector = [alpha(1);phi(1);phiP(1);high(1)];
xOp = [0;0;0;0];
xOp = [alpha(1);phi(1);phiP(1);high(1)];
alphaO(1) = 0;
phiO(1) = 0;
phiPO(1) = 0;
highO(1) = 400;
xObs = [alphaO(1);phiO(1);phiPO(1); highO(1)];
obsStateVector = [alphaO(1);phiO(1);phiPO(1);highO(1)];
deadZone = 0.1;
for i=1:T/h
    u(i) = -K*obsStateVector+G*ref;
    if (abs(u(i)) < deadZone)
        uu(i)=0;
    else
        uu(i)=sign(u(i))*(abs(u(i)));
```

```
    end
    alpha(i) = xOp(1);
    phi(i) = xOp(2);
    phiP(i) = xOp(3);
    high(i) = xOp(4);
    xP = A*xOp+B*uu(i);
    xOp = xOp+h*xP;
    alphaO(i)= xObs(1);
    phiO(i)= xObs(2);
    phiPO(i)= xObs(3);
    highO(i)= xObs(4);
    yOutObs = C*obsStateVector;
    yOut   = C*stateVector;
    xPrevP = A*xObs+B*uu(i)+Ko'*(yOut-yOutObs);
    xObs = xObs + xPrevP*h;
    stateVector=[alpha(i);phi(i);phiP(i);high(i)];
    obsStateVector=[alphaO(i);phiO(i);phiPO(i);highO(i)];
end
uSO(1)=0;
stateVectorSO=[alpha(1);phi(1);phiP(1);high(1)];
xSO=[alpha(1);phi(1);phiP(1); high(1)];
for i=1:T/h
    uSO(i) = -K*stateVectorSO+G*ref;
    alphaSO(i) = xSO(1);
    phiSO(i) = xSO(2);
    phiPSO(i) = xSO(3);
    highSO(i) = xSO(4);
    xp_so = A*xSO+B*uSO(i);
    xSO = xSO+h*xp_so;
    stateVectorSO = [alphaSO(i);phiSO(i);phiPSO(i);highSO(i)];
end
%%
% Se obtienen las siguientes gráficas:

plot(t,alpha,'LineWidth',1.5);
hold on;
plot(t,alphaSO,'LineWidth',1.5);
hold off
title('Angulo con la horizontal \alpha');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
plot(t,phi,'LineWidth',1.5);
hold on;
plot(t,phiSO,'LineWidth',1.5);
hold off;
title('Angulo de cabeceo \phi');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
plot(t,phiP,'LineWidth',1.5);
hold on;
plot(t,phiPSO,'LineWidth',1.5);
hold off
title('Velocidad angular cabeceo \phi_p');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Posicion angular [rad/seg]');
grid;
plot(t,high,'LineWidth',1.5);
hold on;
plot(t,highSO,'LineWidth',1.5);
hold off
title('Altura h');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Distancia [m]');
grid;
```

```
plot(t,uu,'LineWidth',1.5);
hold on;
plot(t,uSO,'LineWidth',1.5);
hold off
title('Accion de control u_t');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
grid;
clear all;
close all;
clc
a = 0.07;
b = 5;
c = 150;
w = 9;
A = [-a a 0 0; 0 0 1 0; w^2 -w^2 0 0; c 0 0 0];
B = [0; 0; b*w^2; 0];
C = [0 0 0 1;0 1 0 0];
D = [0];
Q = diag([1 1000000 1 1]);
R = 1000000
K = lqr(A,B,Q,R);
G = -inv(C(1,:)*inv(A-B*K)*B);
Ao = A';
Bo = C';
Co = B';
Qo = diag([100 1000 100 1000]);
Ro = 10000000000;
Ko = lqr(Ao,Bo,Qo,Ro);
T = 100;
h = 1e-4;
t = 0:h:(T-h);
ref = 100;
alpha(1) = 0;
phi(1) = 0;
phiP(1) = 0;
high(1) = -500;
u(1) = 0;
uu(1) = 0;
stateVector = [alpha(1);phi(1);phiP(1);high(1)];
xOp = [0;0;0;0];
xOp = [alpha(1);phi(1);phiP(1);high(1)];
alphaO(1) = 0;
phiO(1) = 0;
phiPO(1) = 0;
highO(1) = 400;
xObs = [alphaO(1);phiO(1);phiPO(1); highO(1)];
obsStateVector = [alphaO(1);phiO(1);phiPO(1);highO(1)];
deadZone = 0.1;
for i=1:T/h
    u(i) = -K*obsStateVector+G*ref;
    if (abs(u(i)) < deadZone)
        uu(i)=0;
    else
        uu(i)=sign(u(i))*(abs(u(i)));
    end
    alpha(i) = xOp(1);
    phi(i) = xOp(2);
    phiP(i) = xOp(3);
    high(i) = xOp(4);
    xP = A*xOp+B*uu(i);
    xOp = xOp+h*xP;
    alphaO(i)= xObs(1);
    phiO(i)= xObs(2);
    phiPO(i)= xObs(3);
    highO(i)= xObs(4);
    yOutObs = C*obsStateVector;
    yOut   = C*stateVector;
    xPrevP = A*xObs+B*uu(i)+Ko'*(yOut-yOutObs);
    xObs = xObs + xPrevP*h;
```

```
    stateVector=[alpha(i);phi(i);phiP(i);high(i)];
    obsStateVector=[alphaO(i);phiO(i);phiPO(i);highO(i)];
end
uSO(1)=0;
stateVectorSO=[alpha(1);phi(1);phiP(1);high(1)];
xSO=[alpha(1);phi(1);phiP(1); high(1)];
for i=1:T/h
    uSO(i) = -K*stateVectorSO+G*ref;
    alphaSO(i) = xSO(1);
    phiSO(i) = xSO(2);
    phiPSO(i) = xSO(3);
    highSO(i) = xSO(4);
    xp_so = A*xSO+B*uSO(i);
    xSO = xSO+h*xp_so;
    stateVectorSO = [alphaSO(i);phiSO(i);phiPSO(i);highSO(i)];
end
%%
% Se obtienen las siguientes gráficas:

plot(t,alpha,'LineWidth',1.5);
hold on;
plot(t,alphaSO,'LineWidth',1.5);
hold off
title('Angulo con la horizontal \alpha');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
plot(t,phi,'LineWidth',1.5);
hold on;
plot(t,phiSO,'LineWidth',1.5);
hold off;
title('Angulo de cabeceo \phi');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Ángulo [rad]');
grid;
plot(t,phiP,'LineWidth',1.5);
hold on;
plot(t,phiPSO,'LineWidth',1.5);
hold off
title('Velocidad angular cabeceo \phi_p');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Posicion angular [rad/seg]');
grid;
plot(t,high,'LineWidth',1.5);
hold on;
plot(t,highSO,'LineWidth',1.5);
hold off
title('Altura h');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
ylabel('Distancia [m]');
grid;
plot(t,uu,'LineWidth',1.5);
hold on;
plot(t,uSO,'LineWidth',1.5);
hold off
title('Accion de control u_t');
legend('Con observador','Sin observador')
xlabel('Tiempo [seg]');
grid;
```

Código del modelo del péndulo

```
clear all;
close all;
clc;
%%
```

```
% Se definen los parámetros del sistema:

m = 0.1;
F = 0.1;
l = 1.6;
g = 9.8;
M = 1.5;
%%
% Las matrices del sistema en tiempo continuo son:

Ac = [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]
Bc = [0; 1/M; 0; 1/(l*M)]
Cc = [1 0 0 0; 0 0 1 0];
Dc = [0];
%%
% Los tiempos que se manejarán son:

Ts = 1e-2;
T = 25;
At = 1e-4;
Kmax = T/Ts;
%%
% Se obtiene el sistema de tiempo continuo para luego discretizar.

sys1 = ss(Ac, Bc, Cc, Dc)
dSys1 = c2d(sys1,Ts,'zoh')
A = dSys1.A
B = dSys1.B
%%
% Se implementa un integrador para medir el desplazamiento:

Cref = Cc(1,:);
Aamp1 = [A,zeros(4,1);-Cref*A,eye(1)]
Bamp1 = [B;-Cref*B]
Q1 = diag([.1 1e-2 1 .1 0.0000093])
R1 = 1.9e-4;
K1 = dlqr(Aamp1,Bamp1,Q1,R1)
Kp1 = K1(1:4)
Kint1 = -K1(5)
%%
% Ahora, para cuando la masa aumente en 10 veces, se plantea otro controlador.

m1 = 10*m;
Ac2 = [0 1 0 0; 0 -F/M -m1*g/M 0; 0 0 0 1; 0 -F/(l*M) -g*(m1+M)/(l*M) 0]
sys2 = ss(Ac2,Bc,Cc,Dc)
dSys2 = c2d(sys2,Ts,'zoh')
A2 = dSys2.A
B2 = dSys2.B
Aamp2 = [A2,zeros(4,1);-Cref*A2,eye(1)]
Q2 = diag([.1 1e-2 1 .1 0.0000093])
R2 = 0.0005;
K2 = dlqr(Aamp2,Bamp1,Q2,R2)
Kp2 = K2(1:4)
Kint2 = -K2(5)
%%
% El observador, a continuacion:

Ao = A'
Bo = Cc'
Co = B'
Qo = diag([0.001 1000 0.5 0.0001])
Ro = diag([80 10000])
Ko = dlqr(Ao,Bo,Qo,Ro);
Ko = Ko'
%%
% Para poder simular, se plantean las siguientes variables:

phi(1) = pi;
x = [0;0;phi(1);0];
delta = x(1);
```

```
deltaP = x(2);
phi = x(3);
omega = x(4);

phiPP(1) = 0;
h = Ts/20;
i = 1;
deltaRef = 10;
bool = 0;
v(1) = 0;
xHat = [0;0;pi;0];
xOp=[0 0 pi 0]';
reference(1) = 10;
%%
% Simulacion:

K = Kp1;
KI = Kint1;
for index=1:Kmax
    yOut = Cc*x;
    yOutObs = Cc*(xHat-xOp);
    v(index+1) = v(index)+deltaRef-yOut(1);
    u1(index) = -K*(x-xOp)+KI*v(index+1);
    deadZone = 1;
    if(abs(u1(index)) < deadZone)
        u1(index) = 0;
    else
        u1(index)=sign(u1(index))*(abs(u1(index))-deadZone);
    end
    for j=1:Ts/h
        u(i) = u1(index);
       p_pp = (1/(M+m))*(u(i)-m*l*phiPP*cos(phi(i))+m*l*omega(i)^2*sin(phi(i))-F*deltaP(i));
        phiPP = (1/l)*(g*sin(phi(i))-p_pp*cos(phi(i)));
        deltaP(i+1) = deltaP(i)+h*p_pp;
        delta(i+1) = delta(i)+h*deltaP(i);
        omega(i+1) = omega(i)+h*phiPP;
        phi(i+1) = phi(i)+h*omega(i);
        if(delta(i) >= 9.99)
            if(bool == 0)
                deltaRef = 0;
                m = m*10;
                bool = 1;
                K = Kp2;
                KI = Kint2;
            end
        end
        i = i+1;
        reference(i) = deltaRef;
    end
    x = [delta(i-1); deltaP(i-1); phi(i-1); omega(i-1)];
    xHat = A*xHat+B*u1(index)+Ko*(yOut-yOutObs)+xOp;
end
u(i) = u1(index);
t = 0:h:T;
%%
% Se muestran las gráficas obtenidas:

plot(t,delta,'LineWidth',1.5);
hold on;
plot(t,reference,'r--')
hold off;
grid;
title('Desplazamiento \delta');
xlabel('Tiempo [seg]')
ylabel('Distancia [m]')
plot(t,deltaP,'LineWidth',1.5);
grid;
title('Velocidad de desplazamiento')
xlabel('Tiempo [seg]')
ylabel('Velocidad [m/s]')
```

```
plot(t,phi,'LineWidth',1.5);
grid;
title('Angulo \phi')
xlabel('Tiempo [seg]')
ylabel('Angulo [rad]')
plot(t,phi/pi*100-100,'LineWidth',1.5);
grid;
title('Angulo \phi con respecto a \pi')
xlabel('Tiempo [seg]')
ylabel('Error [%]')
plot(t,omega,'LineWidth',1.5);
grid;
title('Velocidad angular')
xlabel('Tiempo [seg]')
ylabel('Velocidad angular [rad/s]')
%%
% Los planos de fase son los siguientes:

plot(phi,omega,'LineWidth',1.5);
title('Ángulo vs Velocidad angular');
xlabel('Ángulo \phi [rad]');
ylabel('Velocidad angular \omega [rad/seg]');
grid;
plot(delta,deltaP,'LineWidth',1.5);
title('Distancia vs velocidad');
xlabel('Distancia [m]');
ylabel('Velocidad [m/seg]');
grid;
```