

# Práctica de control de un motor de corriente continua

Sistemas dinámicos y realimentación

Miguel Quiroga González

Grado en Física  
18 de enero de 2024



UNIVERSIDAD  
**COMPLUTENSE**  
MADRID

# 1. Manejo en lazo abierto

El motor empleado en esta sección de la práctica es el mismo que se utilizará en las secciones restantes. Lo primero que se realiza con el motor elegido es comprobar su correcto funcionamiento. Esto se consigue mediante su manejo en lazo abierto, sin realimentación de ningún tipo. El modelo de Simulink utilizado para esta función se presenta en las figuras a continuación.

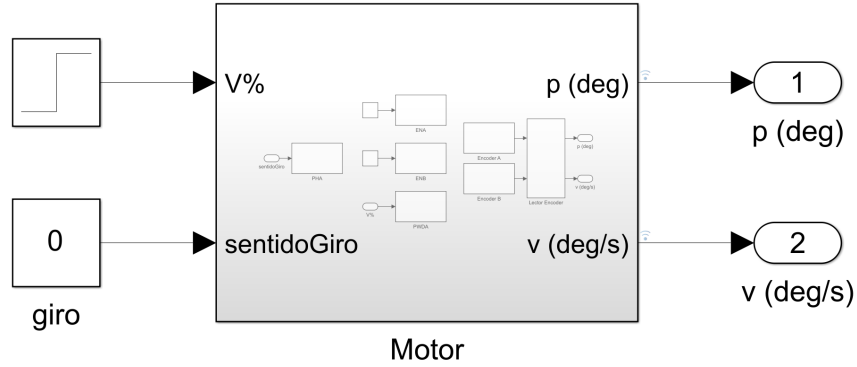


Figura 1: Superficie del modelo de Simulink para control en lazo abierto.

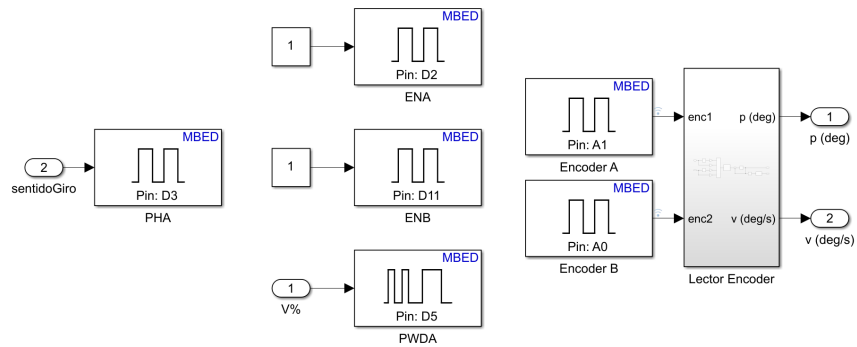


Figura 2: Dirección de pines y lectura de *encoders*.

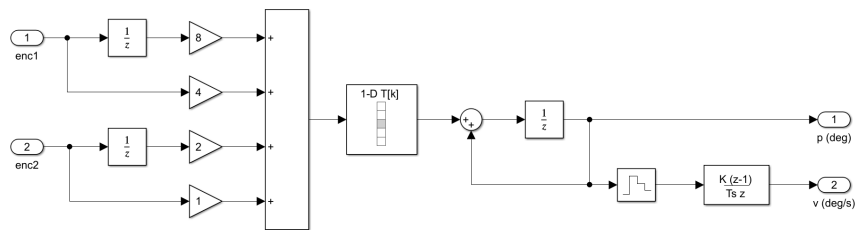


Figura 3: Lector de los *encoders* del motor.

En este modelo se introduce una señal de escalón a la entrada denominada **V%**, la cual representa el porcentaje de voltaje suministrado al sistema, siendo un 0% 0 voltios y un 100% 12 voltios. En las figuras 4 y 5 se observan los datos tomados de la posición y velocidad del motor, respectivamente. Se puede apreciar como la medida de la posición es predecible, pero la de la velocidad salta continuamente entre dos valores. Esto es debido al método de toma de datos mediante

los sensores en cuadratura. Estos únicamente permiten obtener con fiabilidad la posición, por lo que la velocidad se ha de extraer de la misma posición.

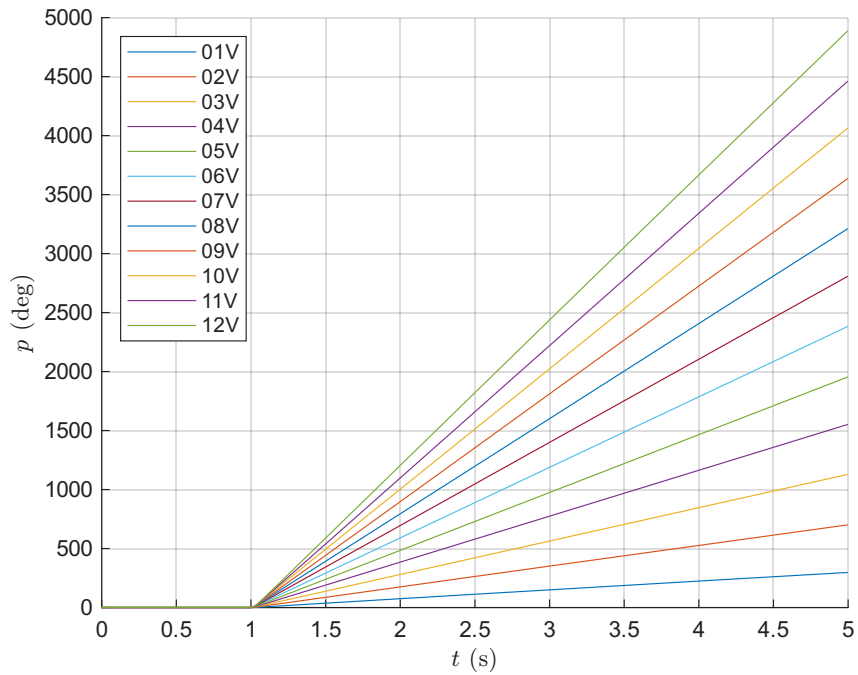


Figura 4: Posición medida según el voltaje.

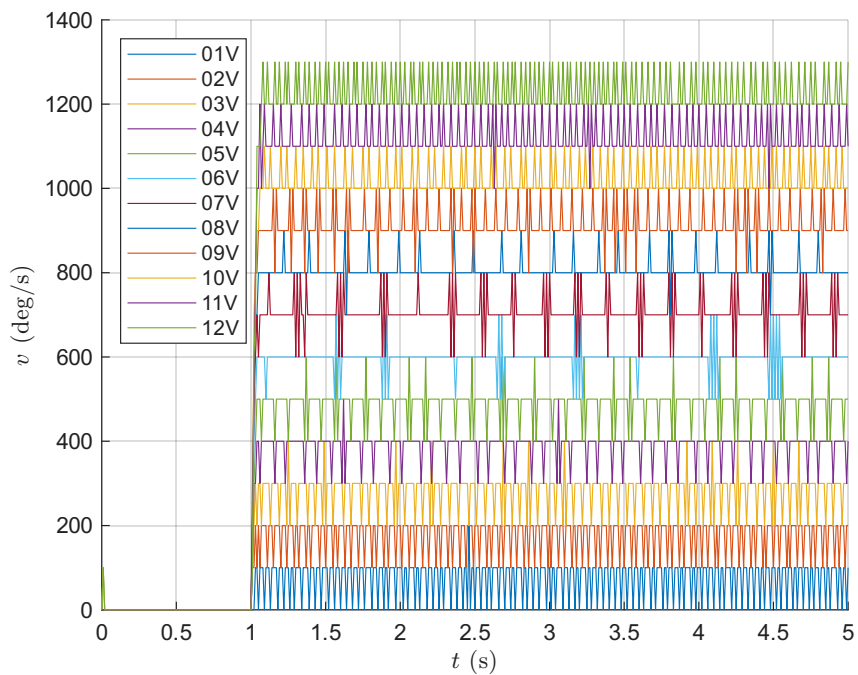


Figura 5: Velocidad medida según el voltaje.

Respecto a los sensores en cuadratura, en la figura 6 se puede apreciar cómo, incluso a velocidad constante, a veces éstos se saltan medidas. Esto realmente no es un problema muy grave debido a que el sistema evoluciona mucho más rápido que los pulsos individuales de estos encoders. Pese a saltarse algunas medidas, el sistema no se ve gravemente afectado.

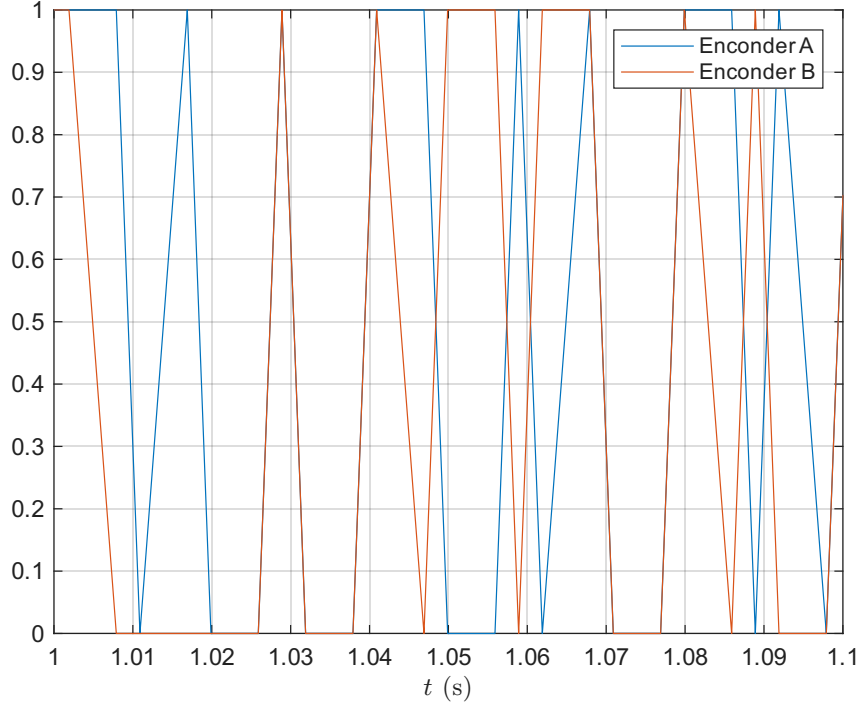


Figura 6: Señal de los encoders.

## 2. Modelado e identificación

Cuando se aplica una señal de escalón  $e(t) = V$  al sistema, la respuesta de la posición angular del mismo es

$$\theta(t) = V \frac{k_e}{p^2} e^{-pt} + V \frac{k_e}{p} t - V \frac{k_e}{p^2}, \quad (1)$$

y la velocidad angular será por lo tanto

$$\omega(t) = \dot{\theta}(t) = -V \frac{k_e}{p} e^{-pt} + V \frac{k_e}{p}. \quad (2)$$

El término con la exponencial corresponde al *wind-up* del motor, es decir, una respuesta no estacionaria. En el caso de este sistema ésta respuesta es muy rápida y se puede eliminar a la hora de ajustar los datos para la identificación del motor. Esto permite realizar un simple ajuste lineal. Además, permite modelar el sistema mediante matrices del siguiente modo:

$$\begin{bmatrix} \dot{\theta} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -p \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ k_e \end{bmatrix} e(t); \quad (3)$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \theta \\ \omega \end{bmatrix}. \quad (4)$$

Comparando este sistema con un sistema lineal arbitrario, es evidente que

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -p \end{bmatrix}; \quad B = \begin{bmatrix} 0 \\ k_e \end{bmatrix}; \quad C = \begin{bmatrix} 1 & 0 \end{bmatrix}; \quad D = 0. \quad (5)$$

Una vez se tiene un modelo matemático del comportamiento del sistema, se procede a hacer la identificación del motor empleado en la práctica. Para ello se emplean los datos presentados en las figuras 4 y 5. Antes de proceder al ajuste de los datos, se realiza una depuración de los mismos, eliminando el primer segundo de inacción y escogiendo únicamente los primeros dos segundos de operación. Esto evita la incorporación de errores en los parámetros del sistema. Evidentemente también se ignoran los datos de la respuesta a 0 V, puesto que el motor se halla estático.

En la tabla 1 se presentan los datos obtenidos para cada voltaje tras realizar un ajuste lineal siguiendo la expresión 1.

Cuadro 1: Valores de los parámetros del sistema

$V$ (V)	$k_e \cdot 10^3$ (deg V <sup>-1</sup> s <sup>-2</sup> )	$p$ (s <sup>-1</sup> )
1	5.2899	69.9906
2	8.3536	94.6228
3	8.3763	88.7427
4	7.9683	81.8434
5	8.2836	84.5199
6	8.7220	87.6272
7	8.1479	81.0227
8	7.8191	77.6825
9	6.9450	68.5878
10	7.1908	70.6311
11	5.6286	55.3149
12	4.9612	48.3772

Observando los datos de la tabla se puede apreciar como el motor encuentra su respuesta más rápida y precisa para los voltajes medios. Esto es principalmente debido al término  $p$ , que corresponde al *wind-up* del motor.

Con el objetivo de llegar a un único valor de los parámetros, se realiza una simple media de los valores de la tabla y se consigue que  $k_e = 7,3072 \cdot 10^3$  deg V<sup>-1</sup> s<sup>-2</sup> y  $p = 75,7469$  s<sup>-1</sup>. Con estos valores, se procede a construir un modelo de Simulink para simular el motor con sus parámetros. Realmente este es un modelo ideal puesto que no tiene en cuenta el *wind-up*.

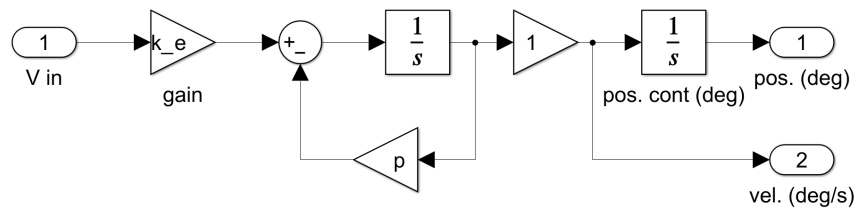


Figura 7: Modelo del motor en Simulink.

Simulando este sistema para los mismos voltajes que en la sección 1, se obtienen resultados

similares. Éstos pueden ser vistos en las figuras 8 y 9. La diferencia más notable es en las velocidades angulares. Al ser esto una simulación, los datos no se han de extraer de dos sensores en cuadratura, y por lo tanto los resultados son más razonables.

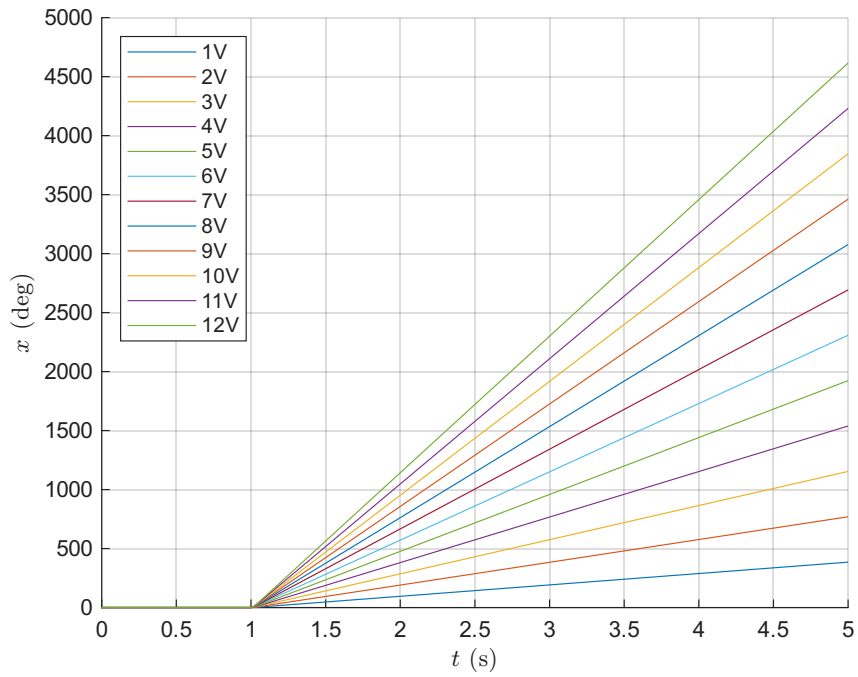


Figura 8: Posición simulada con el modelo ideal.

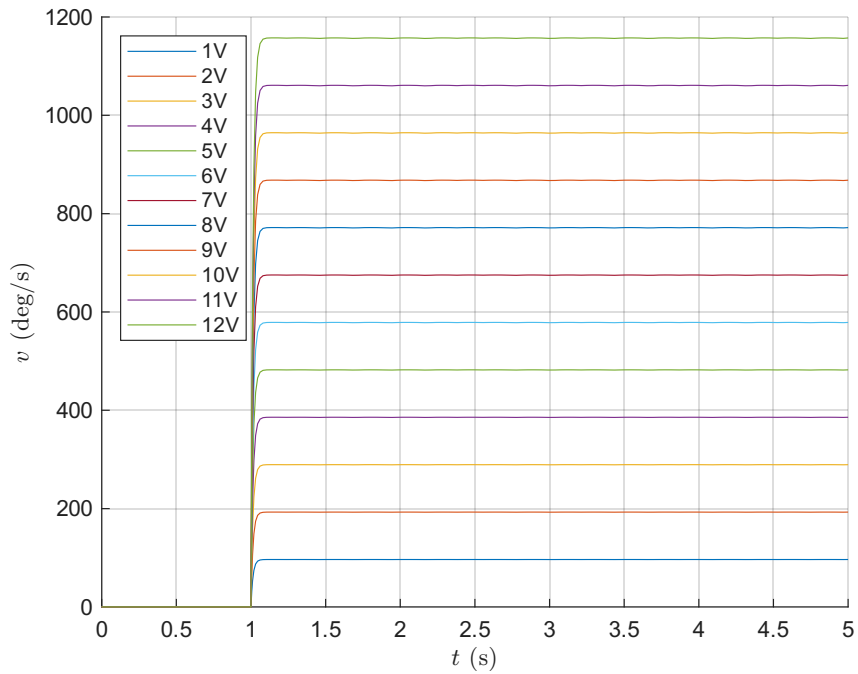


Figura 9: Velocidad simulada con el modelo ideal.

### 3. Simulación completa del motor

Con el objetivo de poder simular el motor ideal con el sistema de la figura 7, se han de implementar varios comportamientos que tiene el motor real. Estos son sus *encoders* y el lector correspondiente, así como una alimentación mediante señal PWM. Esto último además deberá incluir el puente H, para poder girar el motor en ambos sentidos. La señal PWM se consigue mediante el circuito de Simulink presentado en la figura 10. El puente H lo componen los bloques que rodean el subsistema PWM en la figura 12, que consisten en extraer el signo del valor suministrado y volverlo a aplicar a la señal una vez generada la señal PWM. Adicionalmente se añade un bloque previo de *deadzone* que ignora voltajes menores a 0.5 V.

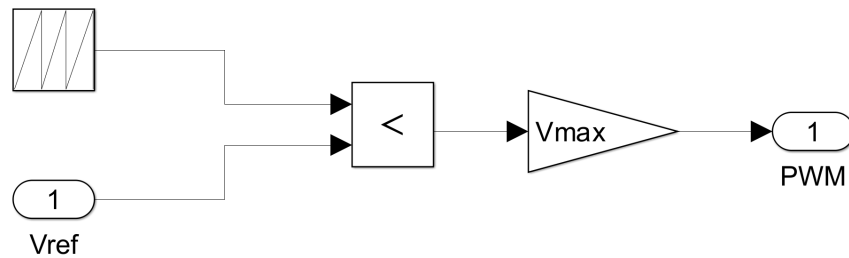


Figura 10: Simulación de la señal PWM.

Como se ha mencionado antes, también se necesita un sistema que codifique la posición y velocidad del motor tal y como lo hacen los sensores en cuadratura del motor real. De este modo se podrá usar posteriormente el sistema de la figura 3 para leer los *encoders*. El sistema que realiza esto se encuentra en la figura 11. La ganancia `rad2deg` que aparece en el modelo se emplea únicamente si el motor se configura para proveer los valores de posición y velocidad en radianes y radianes por segundo, respectivamente. Al no ser el caso, esta ganancia se deja en 1.

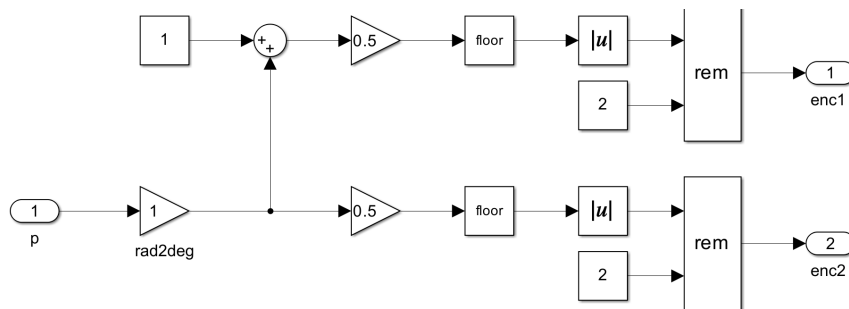


Figura 11: Codificador de las señales del motor para simular sensores en cuadratura.

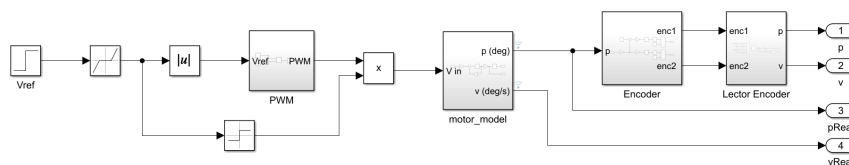


Figura 12: Sistema de simulación ideal completo.

Finalmente se tiene el modelo completo para simular el motor en lazo abierto (fig. 12). La primera comprobación a realizar es que los valores de posición y velocidad que proporciona el lector de los *encoders* sean iguales a las posiciones y velocidades reales simuladas. En la figura 13 se puede observar como el lector de posición encaja a la perfección con la posición del motor. Sin embargo lo mismo no es cierto para la velocidad, la cual tiene altibajos en torno al valor real que debería tener. Esto es algo que se tendrá en cuenta para próximas secciones de la práctica, puesto que también era un problema con los datos del motor real.

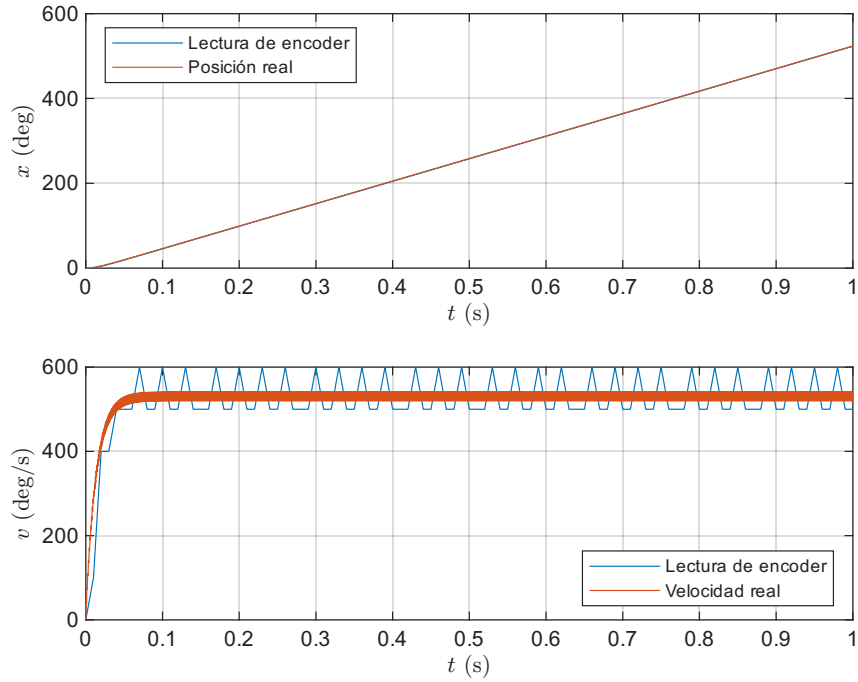


Figura 13: Comprobación de la validez del lector de *encoders*.

Además, se puede obtener una gráfica con la respuesta del sistema a cada valor de voltaje desde 1 a 12 V aumentando en la unidad (fig. 14 y 15), del mismo modo que en las secciones anteriores. Se puede apreciar como la respuesta es similar a las gráficas previas.



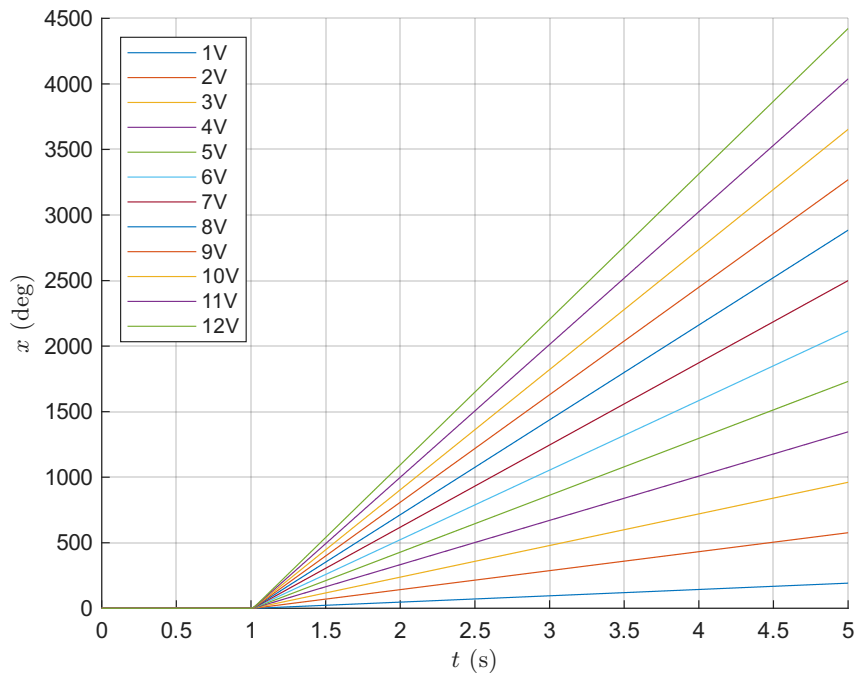


Figura 14: Posición simulada según el *encoder*.

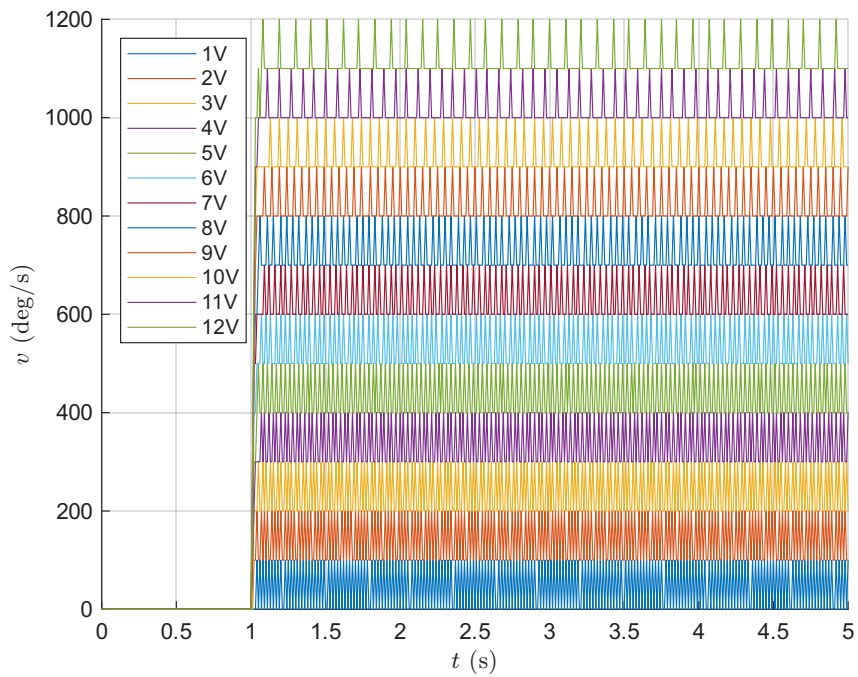


Figura 15: Posición simulada según el *encoder*.

#### 4. Diseño de un controlador por realimentación de estados estimados

Volviendo al motor ideal sin imitar su comportamiento real, se va a diseñar un sistema que permita controlar el motor y llevarlo a una posición de consigna elegida. Esto se hará mediante

una realimentación de estados y un control integral. Sin embargo, como se explicó previamente, los datos de la velocidad del motor no son de fiar, por lo que también se empleará un estimador de estados. Es por esto que en la ecuación 5 se eligió ese valor para  $C$ .

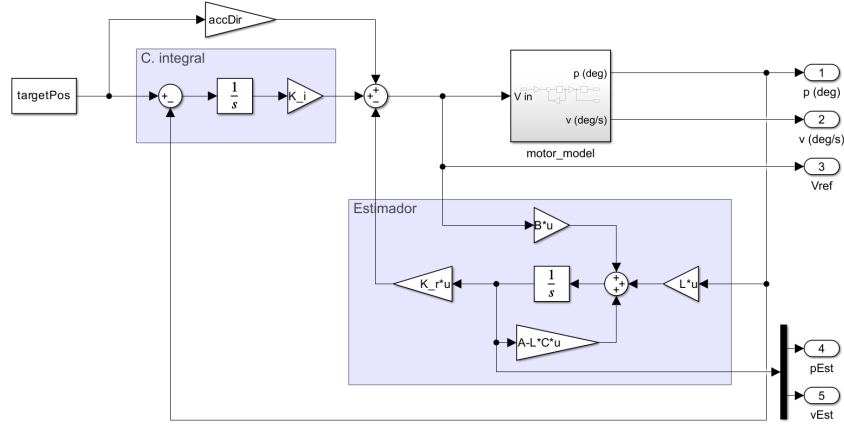


Figura 16: Sistema del controlador con el motor ideal.

El modelo completo se presenta en la figura 16. Para poder emplear el comando de `place` de MATLAB, se han elegido unos polos del estimador tal que  $\lambda_e = [-0,65, -0,6] \cdot p$ . El vector de polos aplicado a la matriz ampliada a la hora de calcular las ganancias de realimentación y control integral es  $\lambda_{ri} = [-0,3, -0,35, -0,4] \cdot p$ . Los resultados de simulación con estos polos se pueden ver en la figura 17. En la figura se aprecia que se alcanza con éxito un valor de consigna de  $5^\circ$ . Lo que también se llega a apreciar sin problemas en la figura es el efecto que tiene la diferencia inicial entre el estimador y los estados reales. Esta diferencia provoca un pico en la acción del sistema, que se termina relajando según converge.

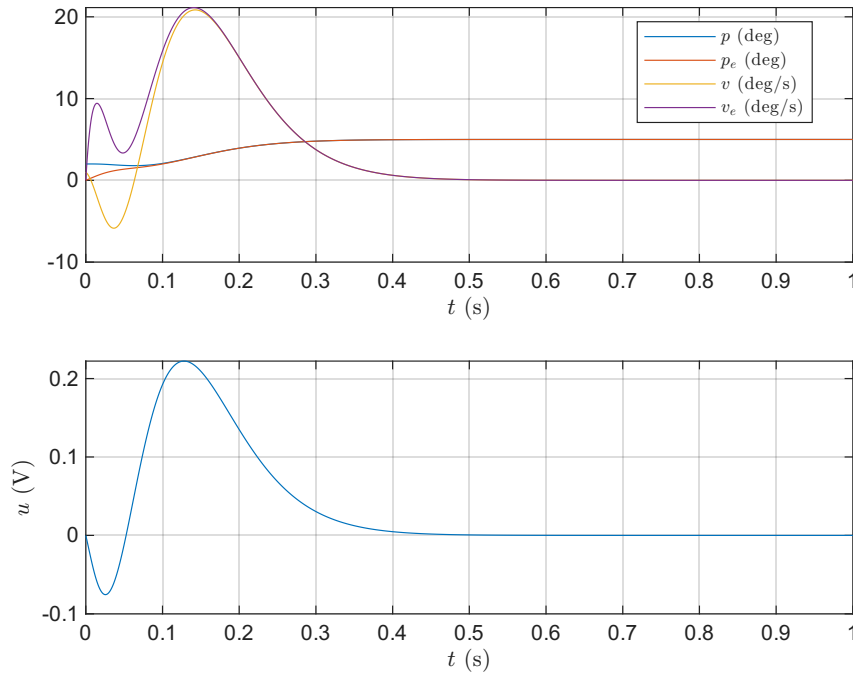


Figura 17: Simulación con  $\lambda_e = [-0,65, -0,6] \cdot p$  y  $\lambda_{ri} = [-0,3, -0,35, -0,4] \cdot p$ .

En las figuras a continuación (fig. 18 y 19) se observa el efecto que tiene en el sistema la colocación de los polos tanto del estimador como de la realimentación. En la figura 18 se observa una respuesta lenta debido a unos polos mucho más pequeños. En este caso el sistema aporta todo el voltaje que puede para estabilizarlo, lo cual realmente no sería necesario si conociera la posición real. En la figura 19 se observa todo lo contrario, una respuesta muy rápida, que produce un pico de tensión muy alta también, y donde los estados llegan a tomar valores muy elevados. En el motor real, si la tensión de alimentación requerida supera 12 V únicamente lleva a una saturación y eventualmente alcanzará el valor deseado, por lo que lo ideal sería una respuesta más rápida que lenta.

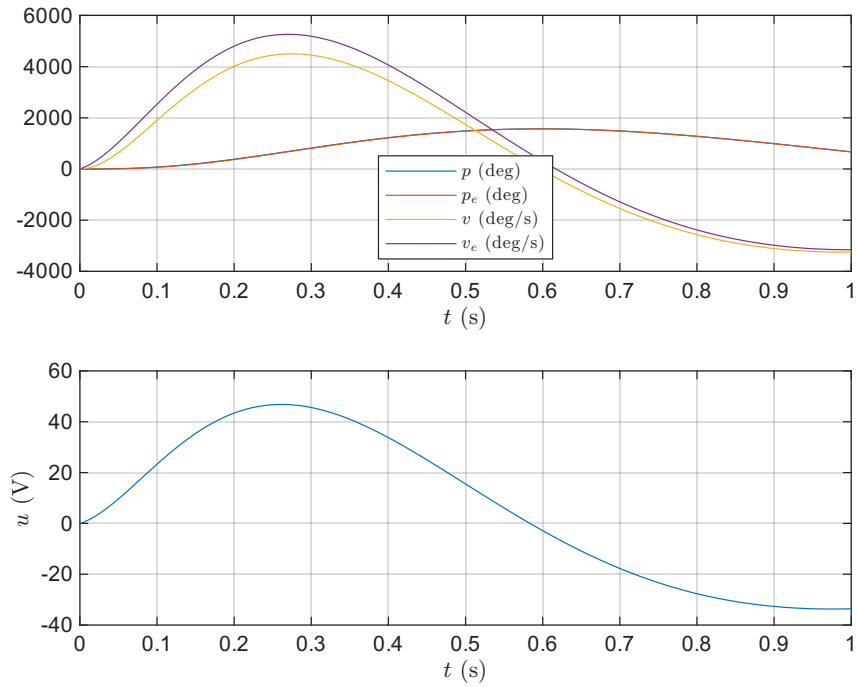


Figura 18: Simulación con  $\lambda_e = [-0,065, -0,06] \cdot p$  y  $\lambda_{ri} = [-0,03, -0,035, -0,04] \cdot p$ .

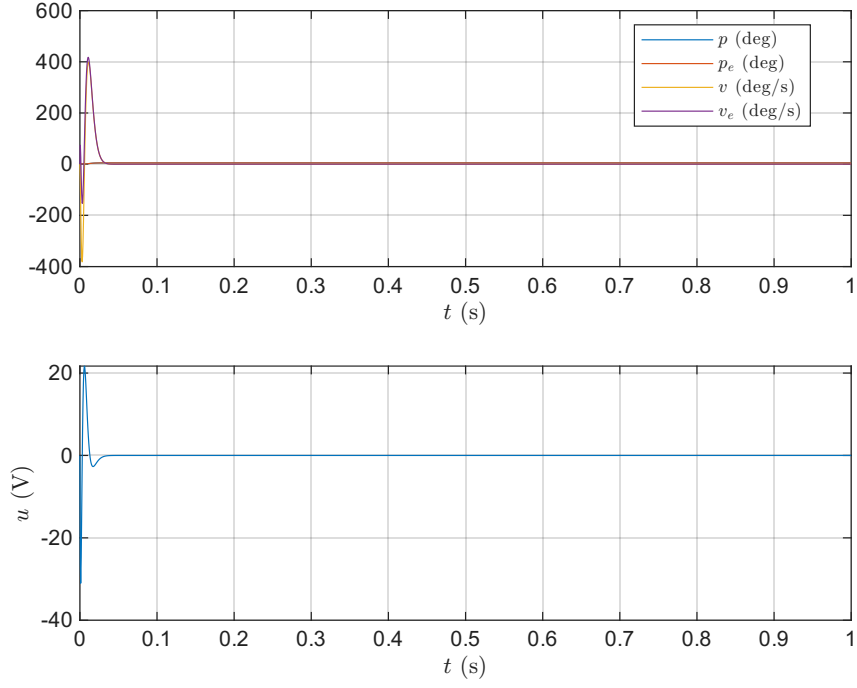


Figura 19: Simulación con  $\lambda_e = [-65, -6] \cdot p$  y  $\lambda_{ri} = [-3, -3, 5, -4] \cdot p$ .

La ganancia denominada `accDir` es un remanente del control sin integrador. Si se ajusta esta ganancia sin presencia de un control integral, se puede forzar un valor de consigna manualmente. Pero como ya se ha mencionado, al estar empleando un control integral esta ganancia es innecesaria y se puede dejar 0. Ajustarla llevaría a un valor de

$$F_c = \left[ -C[A - BK]^{-1}B \right]^{-1}. \quad (6)$$

En la figura 20 se puede observar una simulación con los mismos polos que la figura 17 y se aprecia como, pese a que el voltaje requerido es mayor, se alcanza la consigna más rápido.

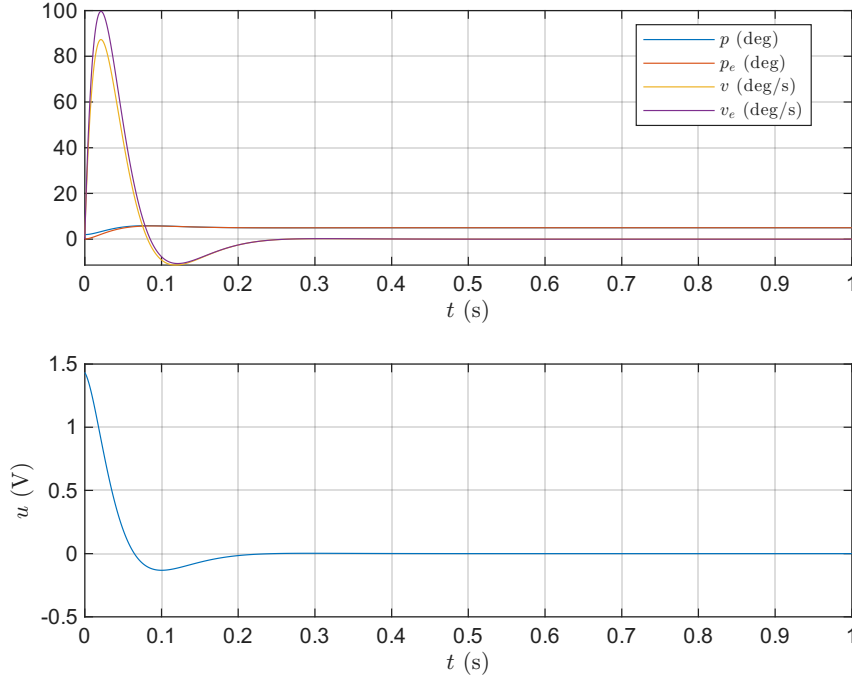


Figura 20: Simulación con la acción directa ajustada.

## 5. Discretización del controlador

Para poder llegar a operar con el motor real se ha de tener en cuenta que todo se hace a través de un sistema digital: las operaciones no se hacen de modo continuo sino discreto. Es por esto que es necesario discretizar nuestro sistema. Para ello se empleará el bloque *Zero-Order Hold* de Simulink. Este bloque muestrea las señales que le lleguen siguiendo un periodo proporcionado, el cual en este caso será  $T = 0,1$  ms. Este periodo será el mismo que se emplea a la hora de simular el sistema, empleando pasos fijos de tamaño  $T$ . Sin embargo, al discretizar el sistema las matrices que lo describen son alteradas de tal manera que

$$A \rightarrow F = e^{AT}; \quad B \rightarrow G = e^{AT} \left[ \int_0^T e^{-\tau} d\tau \right] B. \quad (7)$$

Gracias a estar usando MATLAB, no es necesario realizar ninguno de estos cálculos manualmente –programándolos, evidentemente–. Para hallar las nuevas matrices basta con definir un sistema continuo empleando la función `ss` y posteriormente crear la versión discretizada usando `c2d` pasando como argumentos el sistema continuo y el periodo  $T$  elegido. Al haber cambiado las matrices del sistema también se han de recalcular los polos. La transformación de unos polos arbitrarios es

$$\lambda \rightarrow \Lambda = e^{\lambda T}, \quad (8)$$

y finalmente el cálculo de ganancias de realimentación se realiza igual que en el sistema continuo, reemplazando adecuadamente las matrices y los polos.

Al igual que con el modelo en variables continuas, existe una ganancia de acción directa,  $f$ , para el sistema discretizado. Del mismo modo, realmente no es necesario ajustar esta ganancia

debido a la acción integral de la cual se dispone. Sin embargo, se puede ajustar igualmente esta ganancia de la siguiente manera:

$$f = \left[ C [I - (F - GK)]^{-1} G \right]^{-1}, \quad (9)$$

la cual no siempre produce exactamente los mismos resultados que la ganancia  $F_c$  si no fue ajustada. Para conseguir el mismo efecto en ambas,  $f$  se puede alterar libremente para que su acción coincida con la del modelo continuo. Esto es gracias a la presencia de acción integral. Ahora  $f$  toma el valor

$$f = - \left[ C [I - (F - GK)]^{-1} G \right]^{-1} C [A - BK]^{-1} B F_c. \quad (10)$$

Simulando el sistema sin acción integral pero con la ganancia de acción directa ajustada se puede apreciar cómo el sistema también consigue alcanzar la consigna rápidamente (fig. 21).

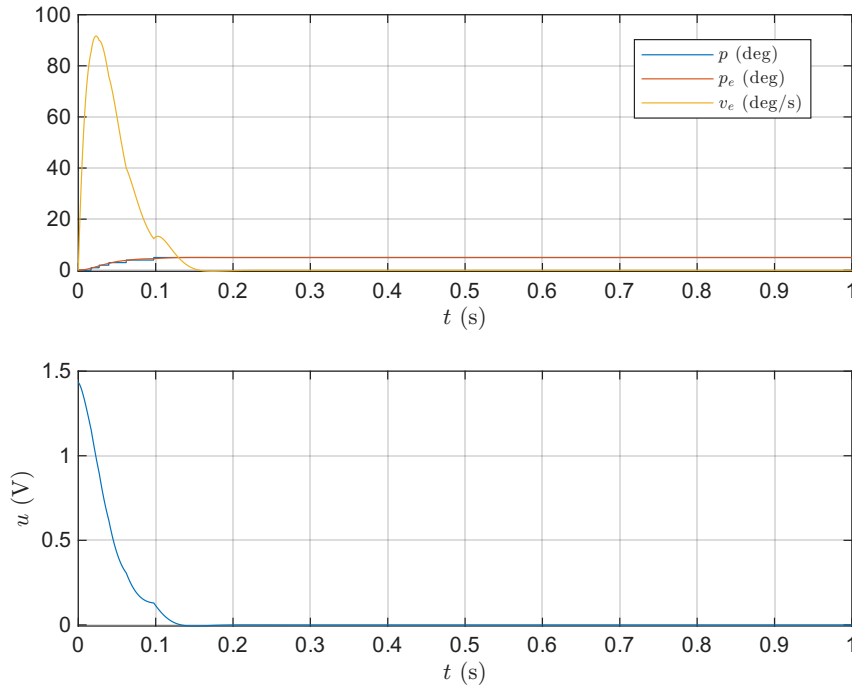


Figura 21: Sistema simulado sin acción integral.

En la figura 22 se presenta la simulación correspondiente al valor ajustado de la acción directa. Comparando con la figura 23, con  $f = 0$ , se observa cómo las diferencias son las mismas que para el control continuo. Salvando errores numéricos, la forma de la realimentación del sistema con acción directa es muy similar a la de la figura 20, tal y como se esperaba. En ambos sistemas se da una situación que previamente no se daba, que es que la posición estimada no termina de converger exactamente con la real, y además el voltaje suministrado no termina en 0, sino en un valor muy cercano. Esto es debido al bloque de *deadzone* que se introdujo previamente para emular los voltajes bajos en los cuales el motor no tiene suficiente fuerza como para empezar a moverse.

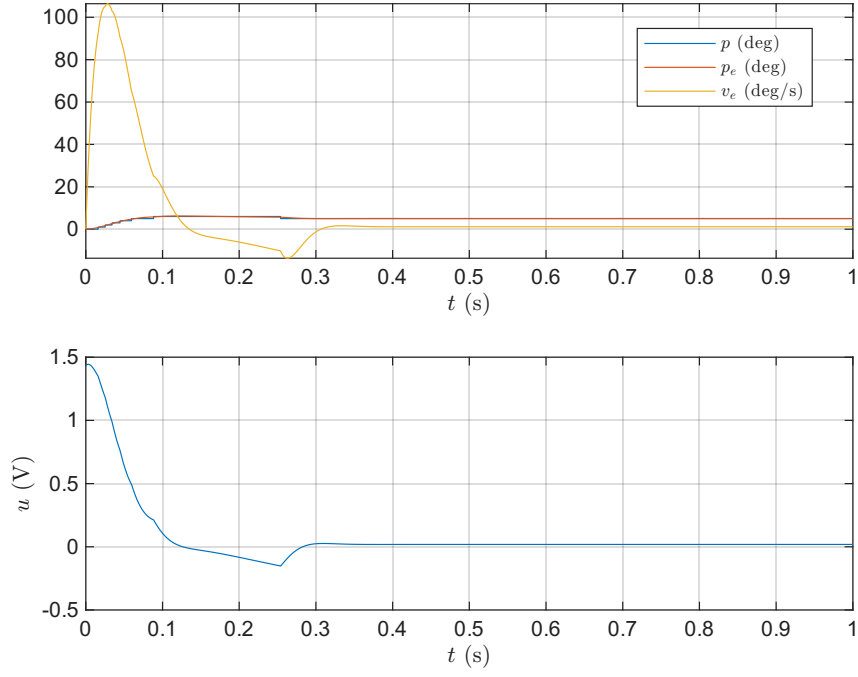


Figura 22: Simulación con acción directa y polos discretizados para  $\lambda_e = [-0,65, -0,6] \cdot p$  y  $\lambda_{ri} = [-0,3, -0,35, -0,4] \cdot p$ .

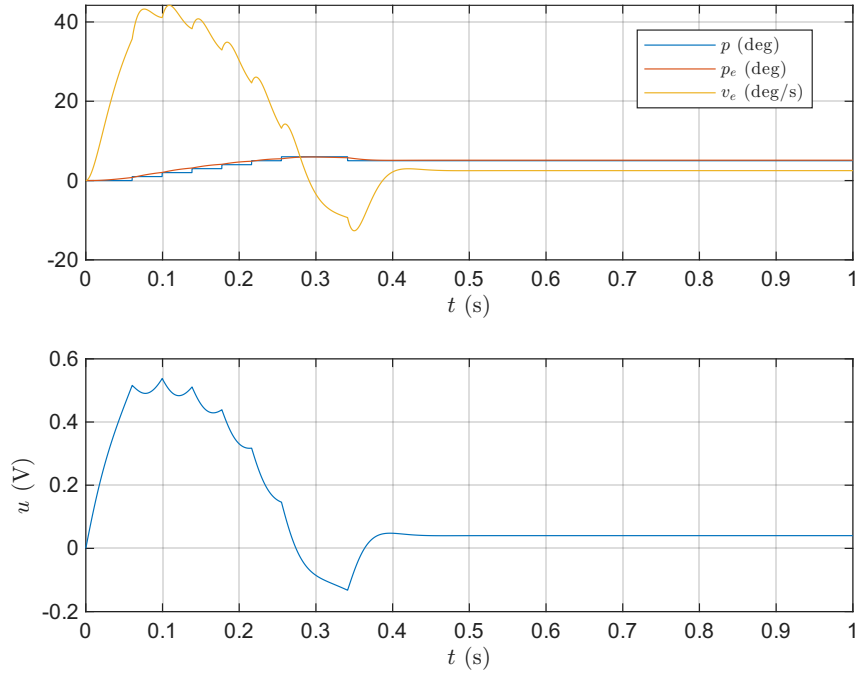


Figura 23: Simulación sin acción directa y polos discretizados para  $\lambda_e = [-0,65, -0,6] \cdot p$  y  $\lambda_{ri} = [-0,3, -0,35, -0,4] \cdot p$ .

Del mismo modo que para el sistema continuo, se puede forzar una respuesta muy rápida al sistema (fig. 24) o una respuesta lenta (fig. 25).

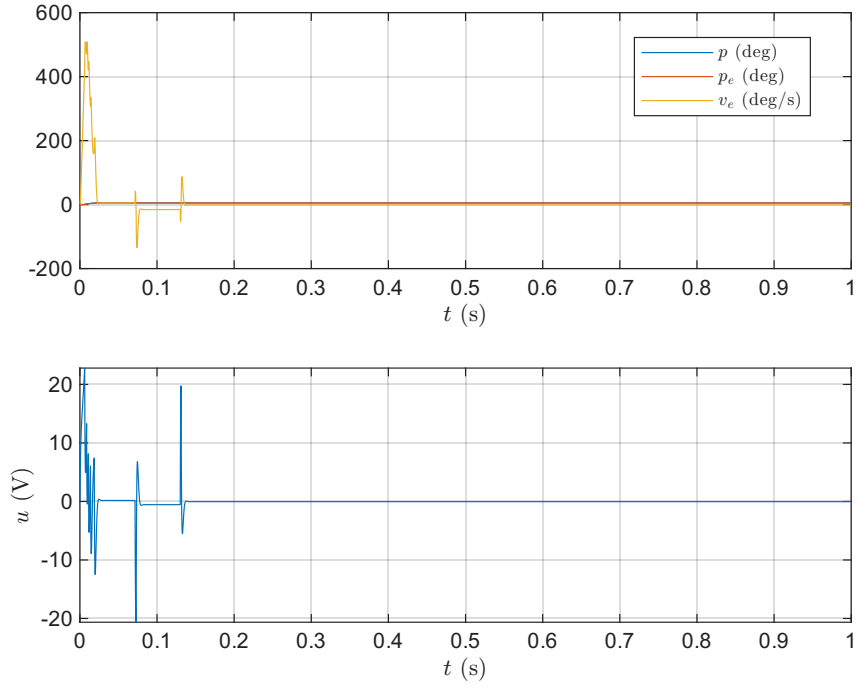


Figura 24: Simulación sin acción directa y polos discretizados para  $\lambda_e = [-65, -6] \cdot p$  y  $\lambda_{ri} = [-3, -3, 5, -4] \cdot p$ .

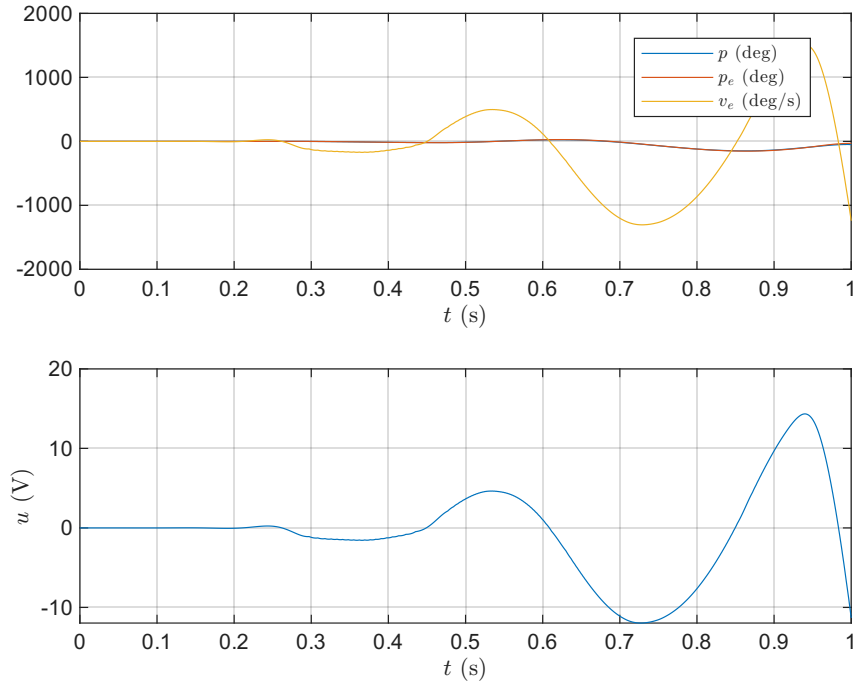


Figura 25: Simulación sin acción directa y polos discretizados para  $\lambda_e = [-0,065, -0,06] \cdot p$  y  $\lambda_{ri} = [-0,03, -0,035, -0,04] \cdot p$ .

Por último se añade al modelo un subsistema que intenta solucionar el problema del *wind-up* –cuando la entrada al sistema satura–. El sistema completo se encuentra en la figura 26.



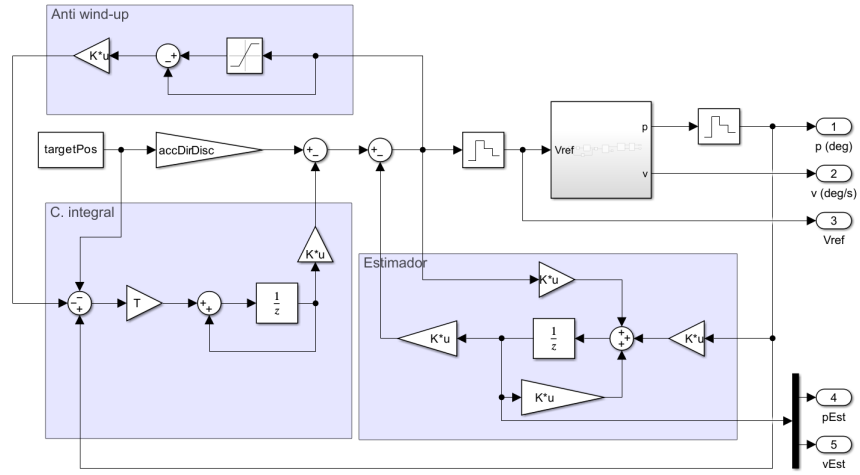


Figura 26: Sistema completo.

Si se discretizan los polos situados en  $\lambda_e = [-5, -5, 1] \cdot p$  y  $\lambda_{ri} = [-4, -4, 1, -4, 2] \cdot p$ , se fuerza al sistema a saturar la señal pasados los 12 V. Con esto se puede comprobar el efecto que tiene la constante de anti *wind-up* (fig. 27-29). Se puede apreciar como para constantes demasiado altas se crean oscilaciones inducidas en la respuesta del sistema. Con esto se concluye que lo mejor es establecer la constante a un nivel relativamente bajo cercano a 0.1.

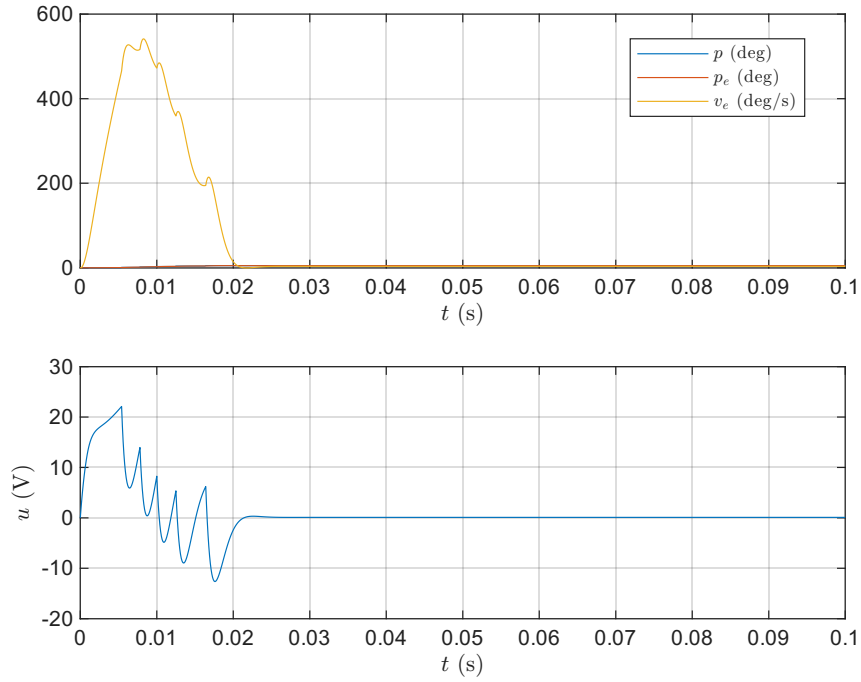


Figura 27: Anti *wind-up* con  $k = 0,1$ .

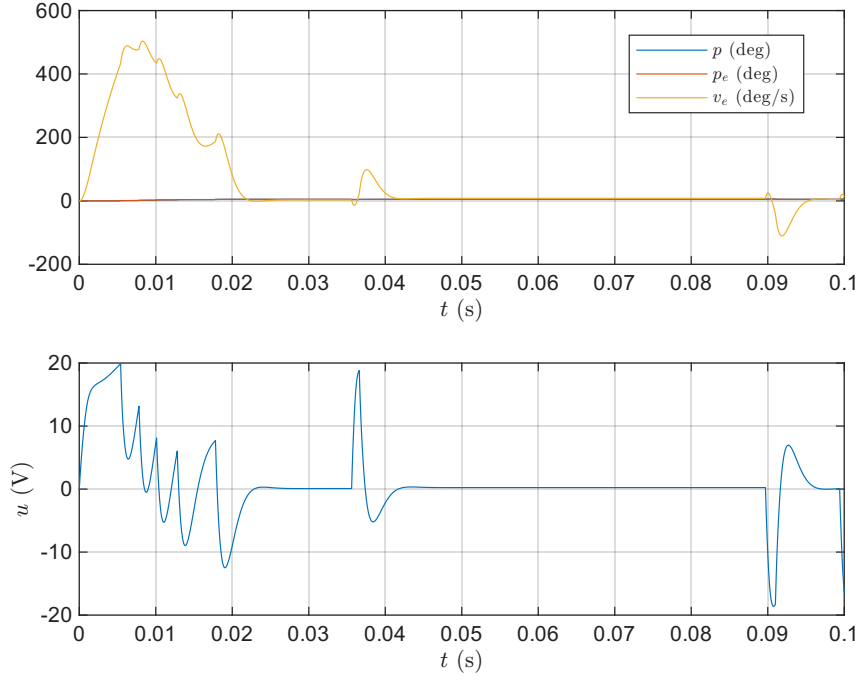


Figura 28: Anti *wind-up* con  $k = 0,2$ .

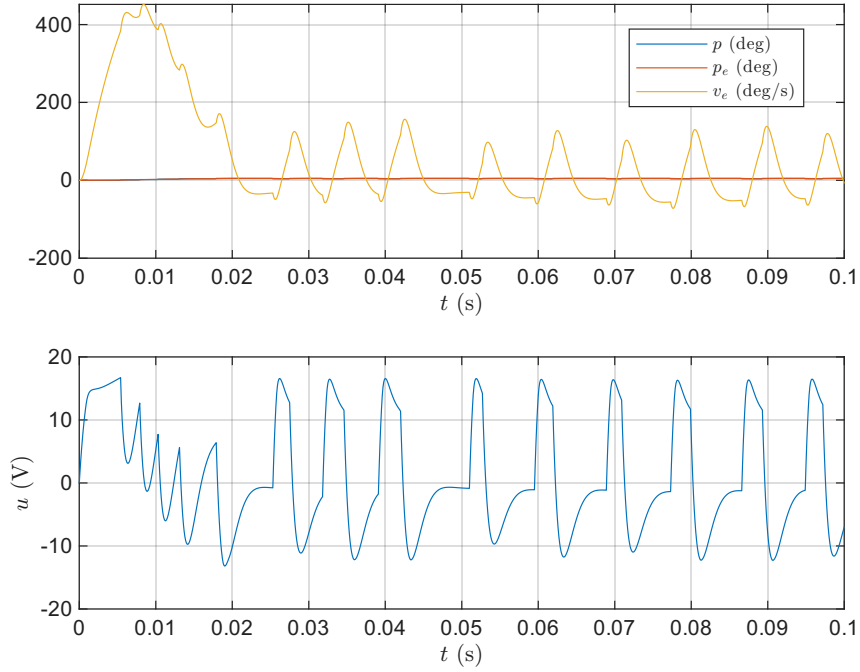


Figura 29: Anti *wind-up* con  $k = 0,5$ .

## 6. Acción sobre el motor real

Con el controlador discreto terminado, queda sustituir el modelo del motor (fig 12) del controlador de la figura 26 por el motor real (fig. 1). Además, es importante añadir un par de bloques más a la entrada del motor real. Éste toma señales de entrada normalizadas a 100, de manera que

12 V sea equivalente a 100. Esto se arregla con una ganancia de  $\frac{100}{12}$ . Además, hay que especificar el sentido de giro por separado del valor de la señal, de manera que se toma el valor absoluto para la señal, y el sentido de giro se determina con una comparación con 0. Estos cambios se pueden ver reflejados en la figura 30.

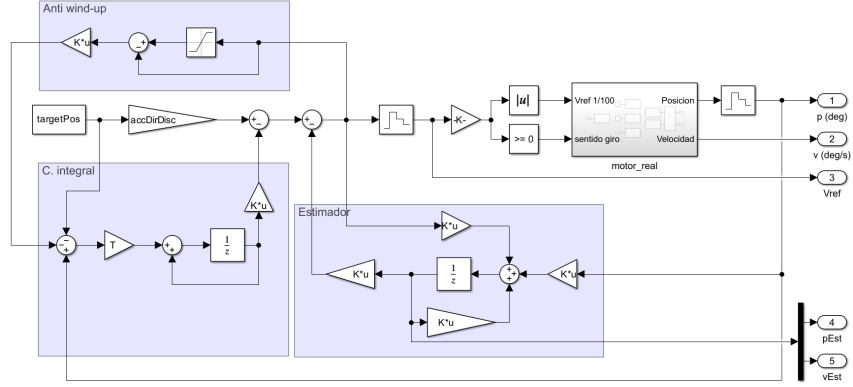


Figura 30: Controlador con el motor real implementado.

Finalmente ya están listos todos los preparativos para realizar medidas y aplicar el controlador en el motor real. Para comprobar el correcto funcionamiento de todos los subsistemas, se selecciona una consigna de  $\pm 180^\circ$ . En las figuras 31 y 32 se pueden ver los resultados del controlador. Tal y como se esperaba, se alcanza la consigna perfectamente. Además, se puede apreciar cómo la señal de entrada se satura, pasando los 12 V, y se activa el anti *wind-up*. Como se predijo, la saturación del sistema no le evite eventualmente alcanzar la consigna determinada. Además, se observa cómo el sentido de giro se transmite correctamente, alcanzando la consigna con el signo adecuado.

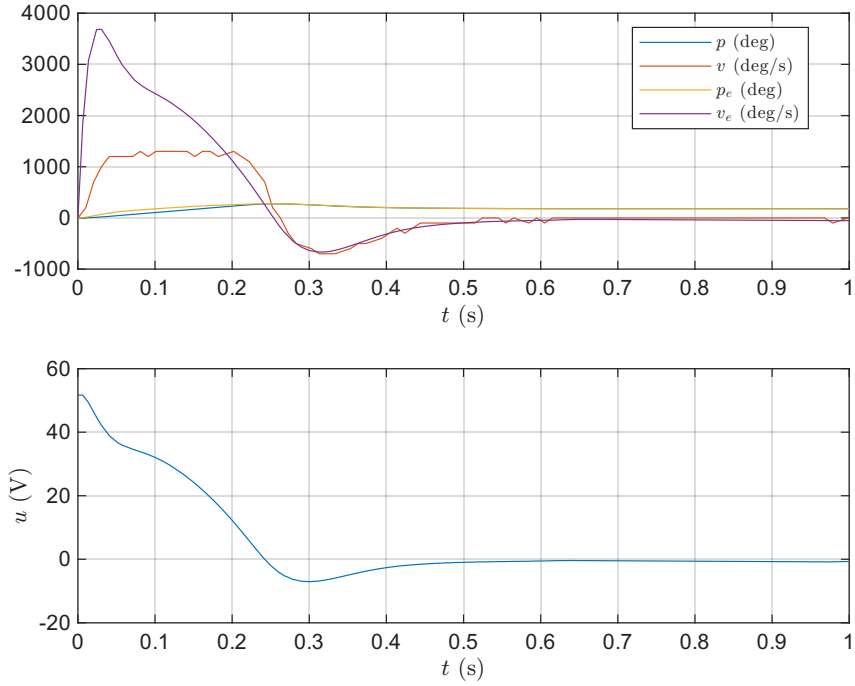


Figura 31: Control real del motor con una consigna positiva de  $180^\circ$ .

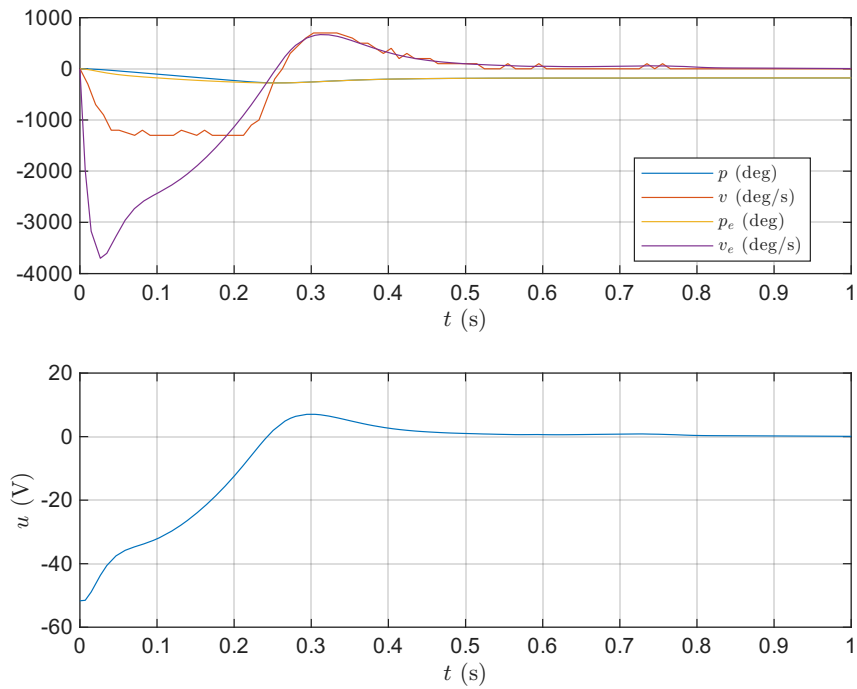


Figura 32: Control real del motor con una consigna negativa de  $-180^\circ$ .

## Anexo I

El código empleado se presenta a continuación, y también está disponible en el siguiente repositorio de GitHub: [https://github.com/n0rb/Realimentados/tree/cambios-miguel/practica\\_motor/realizacion](https://github.com/n0rb/Realimentados/tree/cambios-miguel/practica_motor/realizacion).

```

1  % Cargamos los datos medidos
2  motor_data = load('motor_data\all_data.mat')
3
4  % Posiciones
5  figure
6  hold on
7  for i = 1:12
8      % Extracción de datos
9      % Los archivos tienen el formato motorXXV, siendo XX el voltage
10     % suministrado al motor
11     voltage = pad(num2str(i), 2, "left", '0');
12     extracted = motor_data.(strcat('motor',voltage,'V')).getElement('Motor:1');
13     positions = extracted.Values.Data;
14     time = extracted.Values.Time;
15     % Representación
16     plot(time, positions, DisplayName=strcat(voltage,'V'))
17     grid on
18 end

```

```

19 legend(Location="northwest")
20 xlabel('$t$ (s)', Interpreter='latex')
21 ylabel('$p$ (deg)', Interpreter='latex')
22 exportgraphics(gcf,'figuras\posicionLazoAbierto.pdf')
23
24 % Velocidades
25 figure
26 hold on
27 for i = 1:12
28     % Extracción de datos
29     voltage = pad(num2str(i), 2, "left", '0');
30     extracted = motor_data.(strcat('motor',voltage,'V')).getElement('Motor:2');
31     positions = extracted.Values.Data;
32     time = extracted.Values.Time;
33     % Representación
34     plot(time, positions, DisplayName=strcat(voltage,'V'))
35     grid on
36 end
37 legend(Location="northwest")
38 xlabel('$t$ (s)', Interpreter='latex')
39 ylabel('$v$ (deg/s)', Interpreter='latex')
40 xlim([0 5])
41 exportgraphics(gcf,'figuras\velocidadLazoAbierto.pdf')
42
43 % Comprobación de encoders
44 encoder_data = load('motor_data\encoders_raw.mat')
45 encoderA = encoder_data.data.getElement('Encoder A:1').Values.Data;
46 timeA = encoder_data.data.getElement('Encoder A:1').Values.Time;
47 encoderB = encoder_data.data.getElement('Encoder B:1').Values.Data;
48 timeB = encoder_data.data.getElement('Encoder B:1').Values.Time;
49 figure
50 plot(timeA, encoderA(:), DisplayName='Encoder A')
51 hold on
52 plot(timeB, encoderB(:), DisplayName='Encoder B')
53 grid on
54 legend
55 xlim([1 1.1])
56 xlabel('$t$ (s)', Interpreter='latex')
57 exportgraphics(gcf,'figuras\encodersCheck.pdf')
58
59
60 % Modelado
61 syms V k_e p t
62 angularPos = V*k_e/p^2*exp(-p*t) + V*k_e/p*t - V*k_e/p^2

```

```

63 angularVel = diff(angularPos,t)
64
65 % Identificación - cargado de datos
66 motor_data = load('motor_data\all_data.mat')
67
68 % Ajuste lineal de los datos
69 % Se preasignan los tamaños de las variables
70 lin_fit = ones([12 2]);
71 ke_mat = ones([12 1]);
72 p_mat = ones([12 1]);
73 for i = 1:12
74     % Extracción de datos
75     voltage = pad(num2str(i), 2, "left", '0');
76     extracted = motor_data.(strcat('motor',voltage,'V')).getElement('Motor:1');
77     positions = extracted.Values.Data(401:1200); % sólo 2s del motor encendido
78     time = extracted.Values.Time(1:800); % tiempos re-centrados en 0
79     % Ajuste lineal.
80     lin_fit(i,:) = polyfit(time, positions, 1);
81     ke_mat(i) = -lin_fit(i,1)^2 / (lin_fit(i,2) * i);
82     p_mat(i) = -lin_fit(i,1) / lin_fit(i,2);
83 end
84 % Los parámetros del sistema serán
85 ke_mat, p_mat
86 ke = mean(ke_mat)
87 p = mean(p_mat)
88 % Construimos las matrices del sistema
89 A = [0 1; 0 -p]
90 B = [0 ke] '
91 C = [1 0]
92 D = 0;
93
94 % Comprobación
95 open('motor_ideal.slx')
96 figure(1)
97 clf
98 hold on
99 figure(2)
100 clf
101 hold on
102 for i = 1:12
103     voltage = i;
104     simOut = sim('motor_ideal.slx');
105     time = simOut.tout;
106     angPos = simOut.yout{1}.Values.Data;

```

```

107     angVel = simOut.yout{2}.Values.Data;
108     figure(1)
109     plot(time, angPos, DisplayName=strcat(num2str(voltage), 'V'))
110     figure(2)
111     plot(time, angVel, DisplayName=strcat(num2str(voltage), 'V'))
112 end
113 figure(1)
114 grid on
115 legend(Location="northwest")
116 xlabel('$t$ (s)', Interpreter='latex')
117 ylabel('$x$ (deg)', Interpreter='latex')
118 xlim([0 5])
119 exportgraphics(gcf, 'figuras\posicionIdeal.pdf')
120 figure(2)
121 grid on
122 legend(Location="northwest")
123 xlabel('$t$ (s)', Interpreter='latex')
124 ylabel('$v$ (deg/s)', Interpreter='latex')
125 xlim([0 5])
126 exportgraphics(gcf, 'figuras\velocidadIdeal.pdf')
127
128
129 % Simulación ideal
130 % Definición de parámetros para la simulación
131 tPWM = 1e-3;
132 Vmax = 12;
133
134 % Comprobación de encoders
135 voltage = 6;
136 tf = 1;
137 stepTime = 0;
138 open('simIdeal.slx')
139 simOut = sim('simIdeal.slx');
140 timeReal = simOut.tout;
141 timePos = simOut.yout{1}.Values.Time;
142 angPos = simOut.yout{1}.Values.Data;
143 timeVel = simOut.yout{2}.Values.Time;
144 angVel = simOut.yout{2}.Values.Data;
145 angPosReal = simOut.yout{3}.Values.Data;
146 angVelReal = simOut.yout{4}.Values.Data;
147 figure
148 subplot(2,1,1)
149 plot(timePos, angPos)
150 hold on

```

```

151 plot(timeReal,angPosReal)
152 grid on
153 legend('Lectura de encoder','Posición real',Location="northwest")
154 xlabel('$t$ (s)', Interpreter='latex')
155 ylabel('$x$ (deg)', Interpreter='latex')
156 xlim([0 1])
157 subplot(2,1,2)
158 plot(timeVel,angVel)
159 hold on
160 plot(timeReal,angVelReal)
161 grid on
162 legend('Lectura de encoder','Velocidad real',Location="southeast")
163 xlabel('$t$ (s)', Interpreter='latex')
164 ylabel('$v$ (deg/s)', Interpreter='latex')
165 xlim([0 1])
166 exportgraphics(gcf,'figuras\comprobacionEncoders.pdf')
167
168 % Comparación con los datos reales
169 tf = 5;
170 stepTime = 1;
171 figure(27)
172 clf
173 hold on
174 figure(34)
175 clf
176 hold on
177 for i = 1:12
178     voltage = i;
179     simOut = sim('simIdeal.slx');
180     timePos = simOut.yout{1}.Values.Time;
181     angPos = simOut.yout{1}.Values.Data;
182     timeVel = simOut.yout{2}.Values.Time;
183     angVel = simOut.yout{2}.Values.Data;
184     figure(27)
185     plot(timePos, angPos, DisplayName=strcat(num2str(voltage),'V'))
186     figure(34)
187     plot(timeVel, angVel, DisplayName=strcat(num2str(voltage),'V'))
188 end
189 figure(27)
190 grid on
191 legend(Location="northwest")
192 xlabel('$t$ (s)', Interpreter='latex')
193 ylabel('$x$ (deg)', Interpreter='latex')
194 xlim([0 5])

```



```

195 exportgraphics(gcf,'figuras\posicionIdealEncoder.pdf')
196 figure(34)
197 grid on
198 legend(Location="northwest")
199 xlabel('$\theta$ (s)', Interpreter='latex')
200 ylabel('$\dot{\theta}$ (deg/s)', Interpreter='latex')
201 xlim([0 5])
202 exportgraphics(gcf,'figuras\velocidadIdealEncoder.pdf')
203
204
205 % Ganancia del estimador
206 poles_e = [-0.65*p -0.6*p];
207 L = place(A',C',poles_e)';
208 % Ganancias de realimentación de estados y del control integral
209 A_ri = [A zeros(2,1); C 0];
210 B_ri = [B; 0];
211 poles_ri = [-0.3*p -0.35*p -0.4*p];
212 K_ri = place(A_ri,B_ri,poles_ri);
213 K_r = K_ri(1:2);
214 K_i = K_ri(3);
215 % Condiciones iniciales del sistema
216 p0 = 2;
217 v0 = 1;
218 accDir = (-C*(A-B*K_r)^-1*B)^-1;
219 % Posición de consigna
220 targetPos = 5;
221
222 % Se simula con la acción directa correcta
223 open('controller.slx')
224 simOut = sim('controller.slx');
225 time = simOut.tout;
226 angPos = simOut.yout{1}.Values.Data;
227 angVel = simOut.yout{2}.Values.Data;
228 voltage = simOut.yout{3}.Values.Data;
229 angPosEst = simOut.yout{4}.Values.Data;
230 angVelEst = simOut.yout{5}.Values.Data;
231 figure
232 subplot(2,1,1)
233 plot(time,angPos)
234 hold on
235 plot(time,angPosEst)
236 plot(time,angVel)
237 plot(time,angVelEst)
238 grid on

```

```

239 legend('$p$ (deg)', '$p_e$ (deg)', '$v$ (deg/s)', '$v_e$ (deg/s)', ...
240         Location="best", Interpreter='latex')
241 xlabel('$t$ (s)', Interpreter='latex')
242 xlim([0 1])
243 subplot(2,1,2)
244 plot(time,voltage)
245 grid on
246 xlabel('$t$ (s)', Interpreter='latex')
247 ylabel('$u$ (V)', Interpreter='latex')
248 xlim([0 1])
249 exportgraphics(gcf,'figuras\accionDirecta.pdf')
250
251 % Se simula el sistema para diferentes polos
252 accDir = 0;
253 simOut = sim('controller.slx');
254 time = simOut.tout;
255 angPos = simOut.yout{1}.Values.Data;
256 angVel = simOut.yout{2}.Values.Data;
257 voltage = simOut.yout{3}.Values.Data;
258 angPosEst = simOut.yout{4}.Values.Data;
259 angVelEst = simOut.yout{5}.Values.Data;
260 figure
261 subplot(2,1,1)
262 plot(time,angPos)
263 hold on
264 plot(time,angPosEst)
265 plot(time,angVel)
266 plot(time,angVelEst)
267 grid on
268 legend('$p$ (deg)', '$p_e$ (deg)', '$v$ (deg/s)', '$v_e$ (deg/s)', ...
269         Location="best", Interpreter='latex')
270 xlabel('$t$ (s)', Interpreter='latex')
271 xlim([0 1])
272 subplot(2,1,2)
273 plot(time,voltage)
274 grid on
275 xlabel('$t$ (s)', Interpreter='latex')
276 ylabel('$u$ (V)', Interpreter='latex')
277 xlim([0 1])
278 exportgraphics(gcf,'figuras\controlJusto.pdf')
279
280 poles_e = [-0.065*p -0.06*p];
281 L = place(A',C',poles_e)';
282 poles_ri = [-0.03*p -0.035*p -0.04*p];

```

```

283 K_ri = place(A_ri,B_ri,poles_ri);
284 K_r = K_ri(1:2);
285 K_i = K_ri(3);
286 simOut = sim('controller.slx');
287 time = simOut.tout;
288 angPos = simOut.yout{1}.Values.Data;
289 angVel = simOut.yout{2}.Values.Data;
290 voltage = simOut.yout{3}.Values.Data;
291 angPosEst = simOut.yout{4}.Values.Data;
292 angVelEst = simOut.yout{5}.Values.Data;
293 figure
294 subplot(2,1,1)
295 plot(time,angPos)
296 hold on
297 plot(time,angPosEst)
298 plot(time,angVel)
299 plot(time,angVelEst)
300 grid on
301 legend('$p$ (deg)', '$p_e$ (deg)', '$v$ (deg/s)', '$v_e$ (deg/s)', ...
302         Location="best", Interpreter='latex')
303 xlabel('$t$ (s)', Interpreter='latex')
304 xlim([0 1])
305 subplot(2,1,2)
306 plot(time,voltage)
307 grid on
308 xlabel('$t$ (s)', Interpreter='latex')
309 ylabel('$u$ (V)', Interpreter='latex')
310 xlim([0 1])
311 exportgraphics(gcf,'figuras\controlLento.pdf')
312
313 poles_e = [-6.5*p -6*p];
314 L = place(A',C',poles_e)';
315 poles_ri = [-3*p -3.5*p -4*p];
316 K_ri = place(A_ri,B_ri,poles_ri);
317 K_r = K_ri(1:2);
318 K_i = K_ri(3);
319 simOut = sim('controller.slx');
320 time = simOut.tout;
321 angPos = simOut.yout{1}.Values.Data;
322 angVel = simOut.yout{2}.Values.Data;
323 voltage = simOut.yout{3}.Values.Data;
324 angPosEst = simOut.yout{4}.Values.Data;
325 angVelEst = simOut.yout{5}.Values.Data;
326 figure

```

```

327 subplot(2,1,1)
328 plot(time,angPos)
329 hold on
330 plot(time,angPosEst)
331 plot(time,angVel)
332 plot(time,angVelEst)
333 grid on
334 legend('$p$ (deg)', '$p_e$ (deg)', '$v$ (deg/s)', '$v_e$ (deg/s)', ...
335         Location="best", Interpreter='latex')
336 xlabel('$t$ (s)', Interpreter='latex')
337 xlim([0 1])
338 subplot(2,1,2)
339 plot(time,voltage)
340 grid on
341 xlabel('$t$ (s)', Interpreter='latex')
342 ylabel('$u$ (V)', Interpreter='latex')
343 xlim([0 1])
344 exportgraphics(gcf, 'figuras\controlRapido.pdf')
345
346
347 % Discretizamos el sistema continuo
348 motorCont = ss(A,B,C,D);
349 T = 1e-4;
350 motorDisc = c2d(motorCont,T);
351 F = motorDisc.A;
352 G = motorDisc.B;
353 % Recalculamos los polos de la realimentación, el integrador y el estimador
354 F_ri = [F zeros(2,1); T*C 1];
355 G_ri = [G; 0];
356 poles_ri_disc = exp([-0.3*p -0.35*p -0.4*p]*T);
357 K_ri_disc = place(F_ri,G_ri,poles_ri_disc);
358 K_r_disc = K_ri_disc(1:2);
359 K_i_disc = K_ri_disc(3);
360 poles_e_disc = exp([-0.65*p -0.6*p]*T);
361 L_disc = place(F',C',poles_e_disc)';
362
363 % Constantes
364 k_awp = 0.1;
365 accDirDisc = ( C*( eye(2)-(F-G*K_r_disc))^-1 * G ) )^-1;
366 accInt = 1;
367
368 % Se simula con la acción directa correcta y acción integral
369 open('controller_disc.slx')
370 simOut = sim('controller_disc.slx');

```

```

371 time = simOut.tout;
372 angPos = simOut.yout{1}.Values.Data;
373 voltage = simOut.yout{3}.Values.Data;
374 angPosEst = simOut.yout{4}.Values.Data;
375 angVelEst = simOut.yout{5}.Values.Data;
376 figure
377 subplot(2,1,1)
378 plot(time,angPos)
379 hold on
380 plot(time,angPosEst)
381 plot(time,angVelEst)
382 grid on
383 legend('$p$ (deg)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
384         Location="best", Interpreter='latex')
385 xlabel('$t$ (s)', Interpreter='latex')
386 xlim([0 1])
387 subplot(2,1,2)
388 plot(time,voltage)
389 grid on
390 xlabel('$t$ (s)', Interpreter='latex')
391 ylabel('$u$ (V)', Interpreter='latex')
392 xlim([0 1])
393 exportgraphics(gcf,'figuras\accionDirectaDisc.pdf')
394
395 % Sin acción integral
396 accInt = 0;
397 simOut = sim('controller_disc.slx');
398 time = simOut.tout;
399 angPos = simOut.yout{1}.Values.Data;
400 voltage = simOut.yout{3}.Values.Data;
401 angPosEst = simOut.yout{4}.Values.Data;
402 angVelEst = simOut.yout{5}.Values.Data;
403 figure
404 subplot(2,1,1)
405 plot(time,angPos)
406 hold on
407 plot(time,angPosEst)
408 plot(time,angVelEst)
409 grid on
410 legend('$p$ (deg)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
411         Location="best", Interpreter='latex')
412 xlabel('$t$ (s)', Interpreter='latex')
413 xlim([0 1])
414 subplot(2,1,2)
415 plot(time,voltage)

```

```

416 grid on
417 xlabel('$t$ (s)', Interpreter='latex')
418 ylabel('$u$ (V)', Interpreter='latex')
419 xlim([0 1])
420 exportgraphics(gcf,'figuras\sinAccionIntegral.pdf')
421
422 % Se simula el sistema para diferentes polos
423 accDirDisc = 0;
424 accInt = 1;
425 simOut = sim('controller_disc.slx');
426 time = simOut.tout;
427 angPos = simOut.yout{1}.Values.Data;
428 voltage = simOut.yout{3}.Values.Data;
429 angPosEst = simOut.yout{4}.Values.Data;
430 angVelEst = simOut.yout{5}.Values.Data;
431 figure
432 subplot(2,1,1)
433 plot(time,angPos)
434 hold on
435 plot(time,angPosEst)
436 plot(time,angVelEst)
437 grid on
438 legend('$p$ (deg)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
439         Location="best", Interpreter='latex')
440 xlabel('$t$ (s)', Interpreter='latex')
441 xlim([0 1])
442 subplot(2,1,2)
443 plot(time,voltage)
444 grid on
445 xlabel('$t$ (s)', Interpreter='latex')
446 ylabel('$u$ (V)', Interpreter='latex')
447 xlim([0 1])
448 exportgraphics(gcf,'figuras\controlJustoDisc.pdf')
449
450 poles_ri_disc = exp([-0.03*p -0.035*p -0.04*p]*T);
451 K_ri_disc = place(F_ri,G_ri,poles_ri_disc);
452 K_r_disc = K_ri_disc(1:2);
453 K_i_disc = K_ri_disc(3);
454 poles_e_disc = exp([-0.065*p -0.06*p]*T);
455 L_disc = place(F',C',poles_e_disc)';
456 simOut = sim('controller_disc.slx');
457 time = simOut.tout;
458 angPos = simOut.yout{1}.Values.Data;
459 voltage = simOut.yout{3}.Values.Data;
460 angPosEst = simOut.yout{4}.Values.Data;

```

```

461 angVelEst = simOut.yout{5}.Values.Data;
462 figure
463 subplot(2,1,1)
464 plot(time,angPos)
465 hold on
466 plot(time,angPosEst)
467 plot(time,angVelEst)
468 grid on
469 legend('$p$ (deg)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
470         Location="best", Interpreter='latex')
471 xlabel('$t$ (s)', Interpreter='latex')
472 xlim([0 1])
473 subplot(2,1,2)
474 plot(time,voltage)
475 grid on
476 xlabel('$t$ (s)', Interpreter='latex')
477 ylabel('$u$ (V)', Interpreter='latex')
478 xlim([0 1])
479 exportgraphics(gcf,'figuras\controlLentoDisc.pdf')
480
481 poles_ri_disc = exp([-3*p -3.5*p -4*p]*T);
482 K_ri_disc = place(F_ri,G_ri,poles_ri_disc);
483 K_r_disc = K_ri_disc(1:2);
484 K_i_disc = K_ri_disc(3);
485 poles_e_disc = exp([-6.5*p -6*p]*T);
486 L_disc = place(F',C',poles_e_disc)';
487 simOut = sim('controller_disc.slx');
488 time = simOut.tout;
489 angPos = simOut.yout{1}.Values.Data;
490 voltage = simOut.yout{3}.Values.Data;
491 angPosEst = simOut.yout{4}.Values.Data;
492 angVelEst = simOut.yout{5}.Values.Data;
493 figure
494 subplot(2,1,1)
495 plot(time,angPos)
496 hold on
497 plot(time,angPosEst)
498 plot(time,angVelEst)
499 grid on
500 legend('$p$ (deg)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
501         Location="best", Interpreter='latex')
502 xlabel('$t$ (s)', Interpreter='latex')
503 xlim([0 1])
504 subplot(2,1,2)

```

```

505 plot(time,voltage)
506 grid on
507 xlabel('$t$ (s)', Interpreter='latex')
508 ylabel('$u$ (V)', Interpreter='latex')
509 xlim([0 1])
510 exportgraphics(gcf,'figuras\controlRapidoDisc.pdf')
511
512 % Con diferentes constantes de anti wind-up
513 poles_ri_disc = exp([-4*p -4.1*p -4.2*p]*T);
514 K_ri_disc = place(F_ri,G_ri,poles_ri_disc);
515 K_r_disc = K_ri_disc(1:2);
516 K_i_disc = K_ri_disc(3);
517 poles_e_disc = exp([-5*p -5.1*p]*T);
518 L_disc = place(F',C',poles_e_disc)';
519 accInt = 1;
520 accDirDisc = 0;
521 k_awp = 0.1;
522 simOut = sim('controller_disc.slx');
523 time = simOut.tout;
524 angPos = simOut.yout{1}.Values.Data;
525 voltage = simOut.yout{3}.Values.Data;
526 angPosEst = simOut.yout{4}.Values.Data;
527 angVelEst = simOut.yout{5}.Values.Data;
528 figure
529 subplot(2,1,1)
530 plot(time,angPos)
531 hold on
532 plot(time,angPosEst)
533 plot(time,angVelEst)
534 grid on
535 legend('$p$ (deg)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
536         Location="best", Interpreter='latex')
537 xlabel('$t$ (s)', Interpreter='latex')
538 xlim([0 0.1])
539 subplot(2,1,2)
540 plot(time,voltage)
541 grid on
542 xlabel('$t$ (s)', Interpreter='latex')
543 ylabel('$u$ (V)', Interpreter='latex')
544 xlim([0 0.1])
545 exportgraphics(gcf,'figuras\antiWindUpFlojo.pdf')
546
547 k_awp = 0.2;
548 simOut = sim('controller_disc.slx');

```



```

549 time = simOut.tout;
550 angPos = simOut.yout{1}.Values.Data;
551 voltage = simOut.yout{3}.Values.Data;
552 angPosEst = simOut.yout{4}.Values.Data;
553 angVelEst = simOut.yout{5}.Values.Data;
554 figure
555 subplot(2,1,1)
556 plot(time,angPos)
557 hold on
558 plot(time,angPosEst)
559 plot(time,angVelEst)
560 grid on
561 legend('$p$ (deg)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
562         Location="best", Interpreter='latex')
563 xlabel('$t$ (s)', Interpreter='latex')
564 xlim([0 0.1])
565 subplot(2,1,2)
566 plot(time,voltage)
567 grid on
568 xlabel('$t$ (s)', Interpreter='latex')
569 ylabel('$u$ (V)', Interpreter='latex')
570 xlim([0 0.1])
571 exportgraphics(gcf, 'figuras\antiWindUpMedio.pdf')
572
573 k_awp = 0.5;
574 simOut = sim('controller_disc.slx');
575 time = simOut.tout;
576 angPos = simOut.yout{1}.Values.Data;
577 voltage = simOut.yout{3}.Values.Data;
578 angPosEst = simOut.yout{4}.Values.Data;
579 angVelEst = simOut.yout{5}.Values.Data;
580 figure
581 subplot(2,1,1)
582 plot(time,angPos)
583 hold on
584 plot(time,angPosEst)
585 plot(time,angVelEst)
586 grid on
587 legend('$p$ (deg)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
588         Location="best", Interpreter='latex')
589 xlabel('$t$ (s)', Interpreter='latex')
590 xlim([0 0.1])
591 subplot(2,1,2)
592 plot(time,voltage)
593 grid on

```

```

594 xlabel('$t$ (s)', Interpreter='latex')
595 ylabel('$u$ (V)', Interpreter='latex')
596 xlim([0 0.1])
597 exportgraphics(gcf,'figuras\antiWindUpFuerte.pdf')
598
599
600 % Con consigna positiva
601 data_pos = load('motor_data\motor_real_control.mat')
602 p = data_pos.data.getElement('p (deg):1').Values.Data;
603 pTime = data_pos.data.getElement('p (deg):1').Values.Time;
604 v = data_pos.data.getElement('v (deg//s):1').Values.Data;
605 vTime = data_pos.data.getElement('v (deg//s):1').Values.Time;
606 pEst = data_pos.data.getElement('pEst:1').Values.Data;
607 pEstTime = data_pos.data.getElement('pEst:1').Values.Time;
608 vEst = data_pos.data.getElement('vEst:1').Values.Data;
609 vEstTime = data_pos.data.getElement('vEst:1').Values.Time;
610 Vref = data_pos.data.getElement('Vref:1').Values.Data;
611 VrefTime = data_pos.data.getElement('Vref:1').Values.Time;
612 % Representación
613 figure
614 subplot(2,1,1)
615 plot(pTime,p)
616 hold on
617 plot(vTime,v)
618 plot(pEstTime,pEst)
619 plot(vEstTime,vEst)
620 grid on
621 legend('$p$ (deg)', '$v$ (deg/s)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
622         Location="best", Interpreter='latex')
623 xlabel('$t$ (s)', Interpreter='latex')
624 xlim([0 1])
625 subplot(2,1,2)
626 plot(VrefTime,Vref)
627 grid on
628 xlabel('$t$ (s)', Interpreter='latex')
629 ylabel('$u$ (V)', Interpreter='latex')
630 xlim([0 1])
631 exportgraphics(gcf,'figuras\controlRealPos.pdf')
632
633 % Con consigna negativa
634 data_neg = load('motor_data\motor_real_control_neg.mat')
635 p = data_neg.data.getElement('p (deg):1').Values.Data;
636 pTime = data_neg.data.getElement('p (deg):1').Values.Time;
637 v = data_neg.data.getElement('v (deg//s):1').Values.Data;

```

```

638 vTime = data_neg.data.getElement('v (deg//s):1').Values.Time;
639 pEst = data_neg.data.getElement('pEst:1').Values.Data;
640 pEstTime = data_neg.data.getElement('pEst:1').Values.Time;
641 vEst = data_neg.data.getElement('vEst:1').Values.Data;
642 vEstTime = data_neg.data.getElement('vEst:1').Values.Time;
643 Vref = data_neg.data.getElement('Vref:1').Values.Data;
644 VrefTime = data_neg.data.getElement('Vref:1').Values.Time;
645 % Representación
646 figure
647 subplot(2,1,1)
648 plot(pTime,p)
649 hold on
650 plot(vTime,v)
651 plot(pEstTime,pEst)
652 plot(vEstTime,vEst)
653 grid on
654 legend('$p$ (deg)', '$v$ (deg/s)', '$p_e$ (deg)', '$v_e$ (deg/s)', ...
655         Location="best", Interpreter='latex')
656 xlabel('$t$ (s)', Interpreter='latex')
657 xlim([0 1])
658 subplot(2,1,2)
659 plot(VrefTime,Vref)
660 grid on
661 xlabel('$t$ (s)', Interpreter='latex')
662 ylabel('$u$ (V)', Interpreter='latex')
663 xlim([0 1])
664 exportgraphics(gcf,'figuras\controlRealNeg.pdf')

```