

MEMORIA DE PRÁCTICAS

PRÁCTICA DE CONTROL DE UN MOTOR DE CORRIENTE
CONTINUA



UNIVERSIDAD
COMPLUTENSE
MADRID

Alejandro Avellaneda Calderón

Curso 2024-2025. Primer Cuatrimestre

Grado en Física. Doble Grado en Matemáticas - Física
Universidad Complutense de Madrid

Índice

1. Introducción. Objetivo de las prácticas y metodología	2
2. Breve descripción del <i>Hardware</i> y <i>Software</i> del sistema	3
2.1. <i>Hardware</i> del sistema experimental	3
2.2. <i>Software</i> del sistema experimental	4
3. Manejo del motor de continua en lazo abierto. Modelo e identificación del motor	4
3.1. Manejo del motor de continua en lazo abierto	4
3.2. Modelo e identificación del motor	8
4. Control del motor por realimentación de estados estimados	10
4.1. Un modelo completo del motor	10
4.2. Diseño de un controlador de la posición del motor por realimentación de estados estimados	12
4.2.1. Controlador para el motor ideal	12
4.2.2. Discretización del controlador para control del motor real	19
4.3. Controlador discreto para el modelo completo del motor y el motor real	25
5. Conclusiones	28

1. Introducción. Objetivo de las prácticas y metodología

El objetivo de este documento es plasmar por escrito los procedimientos realizados en las prácticas de la asignatura de Sistemas Dinámicos y Realimentación, cursada durante el primer cuatrimestre del año 2024-2025. En estas sesiones se ha construido desde cero un sistema de control para la posición y la velocidad de un motor de corriente continua, y se han llevado a cabo una serie de prácticas relacionadas con la identificación, modelado y control del sistema. Las prácticas se han realizado en el Laboratorio de Sistemas Digitales del departamento de Arquitectura de Computadores y Automática de la Facultad de Ciencias Físicas, donde se ha proporcionado al alumno todos los elementos de *hardware* necesarios. Por otro lado, el *software* usado para modelar el sistema es Simulink.

En el Anexo A de este documento se puede consultar la distribución de prácticas en función de las fechas en las que se ha acudido al Laboratorio. Como algunas prácticas han llevado más tiempo que otras en realizarse, es muy útil consultar este anexo para ilustrar la duración de las diferentes sesiones. Además, en el Anexo B se diferencian los contenidos de cada práctica y se presenta un breve resumen de cada una de ellas, así como su localización en este documento. Por cuestiones puramente literarias en la redacción de esta memoria se explica el modelado y control del sistema de seguido, como si fuese una práctica sola, a pesar de estar dividida en diferentes sesiones.

Finalmente es necesario mencionar que en este documento se presentan única y exclusivamente los resultados más significativos e importantes de las sesiones de prácticas, y sólo se entra a estudiar y explicar en detalle los más importantes. Para solventar lo que a priori puede ser una falta de explicación o detalle, se adjunta junto a esta memoria todos los archivos de MATLAB y de SIMULINK desarrollados para hacer la práctica. La distribución es la siguiente:

- En los archivos **.mlx** se desarrollan todos los cálculos y desarrollos teóricos necesarios para las prácticas. Estos cálculos incluyen, por ejemplo, el diseño de las matrices K y L necesarias para la realimentación de estados estimados y acción integral, la derivación de la función de posición angular $\theta(t)$ o el proceso de discretización de un sistema continuo. Hay uno por práctica, excepto para la práctica 3 ya que no ha sido necesario.
- En los archivos **.slx** se incluyen los modelos de bloques desarrollados en Simulink para simular el motor de cc. Hay uno por práctica. A la hora de correr los modelos 4,5 y 6 se recomienda primero hacer correr el archivo **.mlx** correspondiente, ya que estos modelos cogen los datos (matrices y parámetros) directamente desde el Workspace.
- En los archivos **.m** se incluyen algunos valores usados para graficar, y en el **EXCEL** se muestra el ajuste realizado sobre los datos para sacar los parámetros k_e y p del motor a considerar.
- Finalmente, en la carpeta "Figuras" se muestran muchas de las figuras incluidas en este documento.

2. Breve descripción del *Hardware* y *Software* del sistema

2.1. *Hardware* del sistema experimental

A nivel de dispositivos, el sistema se compone de varios elementos vitales para su montaje y correcto funcionamiento. A continuación se presenta un breve resumen de cada uno de ellos:

- **Motor:** motor de corriente continua, alimentado mediante escobillas, pensado para trabajar a un voltaje nominal de 12V.
- **Reductora:** juego de ruedas dentadas que permiten reducir el número de vueltas que da el motor.
- **Encoder:** situado en la culata del motor, permite conocer el ángulo recorrido por el eje del motor.
- **Placa de expansión:** esta placa, modelo X_NUCLEO_IHM15A1, permite conectar entre sí multitud de puertos y dispositivos.
- **Microprocesador:** conectado a la placa de expansión, permite generar las señales de control necesarias para alimentar una serie de pines lógicos. Además, incluye un compilador C/C++ cruzado, así como un puerto USB con el que se conecta, vía cable, con el ordenador.
- **Fuente de alimentación:** proporciona una fuente de corriente continua, de valores desde 0V hasta 12V.

Así, siguiendo el guión de prácticas de estas sesiones, se configura el *hardware* del experimento; este montaje experimental puede consultarse en la figura 2.1.1

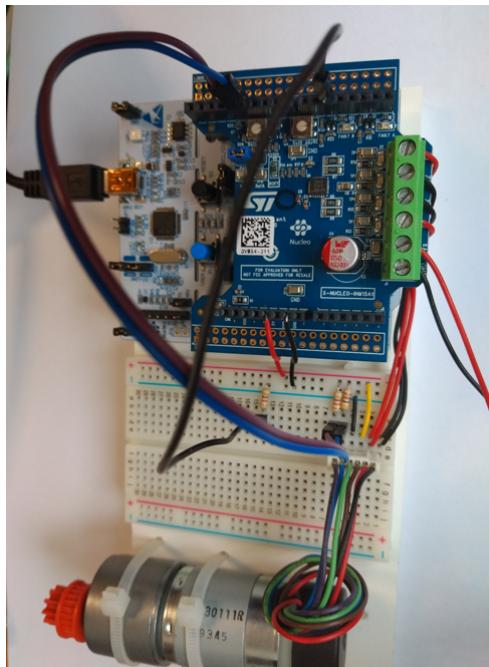


Figura 2.1.1: Montaje del *hardware* del motor.

2.2. Software del sistema experimental

El *software* elegido para desarrollar la práctica es el proporcionado por MATLAB/Simulink. Ambos programas son muy populares en la Academia y en la industria, y ofrecen soporte para la programación de placas de la familia NUCLEO. Concretamente, se emplea la librería *Simulink Coder Support Package for STMicroelectronics Nucleo Boards*; si no se tiene descargada, es necesario hacerlo vía la pestaña *Add-ons* de MATLAB.

Simulink permite crear modelos arrastrando bloques predefinidos (matemáticos, lógicos, de control, etc.) y conectándolos para representar relaciones y flujos de datos. Posteriormente, las simulaciones se ejecutan para visualizar el comportamiento de dicho modelo en el tiempo. Además, se pueden ajustar parámetros y probar diferentes escenarios en tiempo real, facilitando el diseño de sistemas complejos. Por ello, es una herramienta extraordinariamente útil para el diseño del sistema de control del motor. Los pasos a seguir para terminar de configurar el programa, así como para compilar y ejecutar el modelo, pueden consultarse en el guión de prácticas en el que se basa la memoria.

3. Manejo del motor de continua en lazo abierto. Modelo e identificación del motor

3.1. Manejo del motor de continua en lazo abierto

En esta primera parte de la práctica se construye un modelo de Simulink que permite suministrar al motor un valor fijo de voltaje, así como leer su posición y velocidad a través de los *encoders*. Debido a su extensión e importancia, esta primera parte puede también interpretarse como una práctica independiente en sí misma.

La figura 3.1.1 muestra una vista general del modelo. Como se ve en ella y se indica en el guión de prácticas, el modelo del motor en lazo abierto se compone de dos bloques de entrada, uno en escalón que regula la entrada de voltaje y uno deslizante que controla el sentido de giro del motor; uno central, llamado **bloque motor**; y dos bloques de salida o *scopes* que permiten observar las señales de posición y velocidad del motor. A la luz de esta figura, es obvio que el eje central del modelo es el bloque motor.

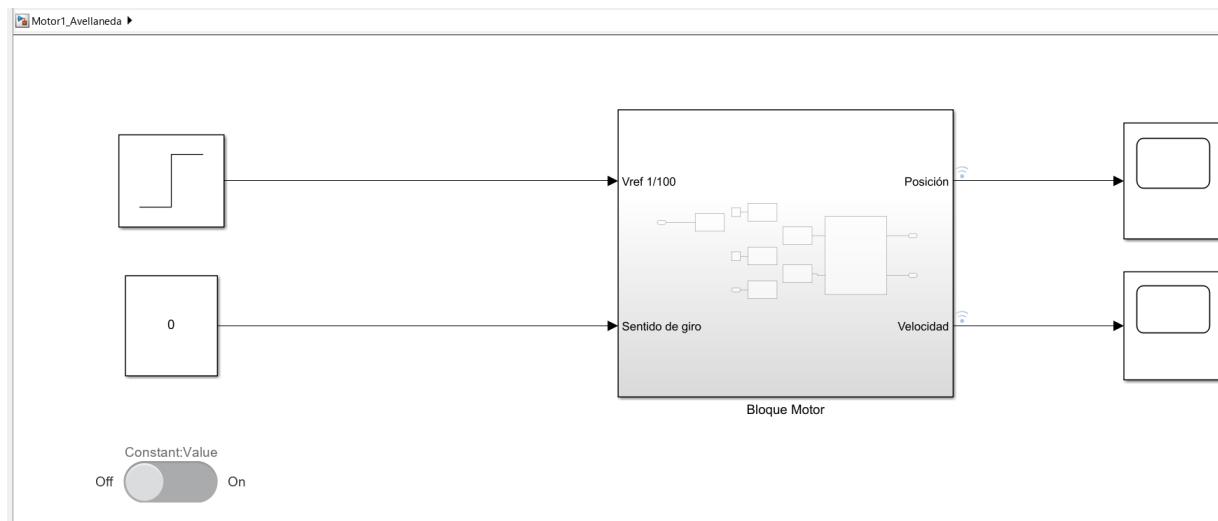


Figura 3.1.1: Vista general del modelo del motor en lazo abierto. Se observan los dos bloques de señales de entrada, el bloque motor como pieza central del modelo y los dos *scopes* de salida que permiten ver la visualización de los valores de posición y velocidad.

El bloque motor es, como se puede apreciar en la figura 3.1.2, una combinación de diferentes bloques y subsistemas cruciales para el buen funcionamiento del motor. Por un lado está formado por numerosos bloques *Digital Write* que recogen el valor del escalón descrito en el párrafo anterior y lo envían directamente a los pines indicados en los bloques. Además, cada bloque se encarga de habilitar un puente de la placa de expansión. Para estudiar en profundidad lo que hace cada bloque se recomienda consultar el guión de prácticas. Sin embargo, es necesario destacar los bloques “Encoder A” y “Encoder B”, que permiten leer un valor digital en el pin para el que está configurado y enviarlo al subsistema **Lector Encoder**. Este subsistema, ilustrado en la figura 3.1.3, transforma las lecturas de los Encoders en distancias angulares y velocidades angulares, ambas expresadas en grados/s.

Práctica de control

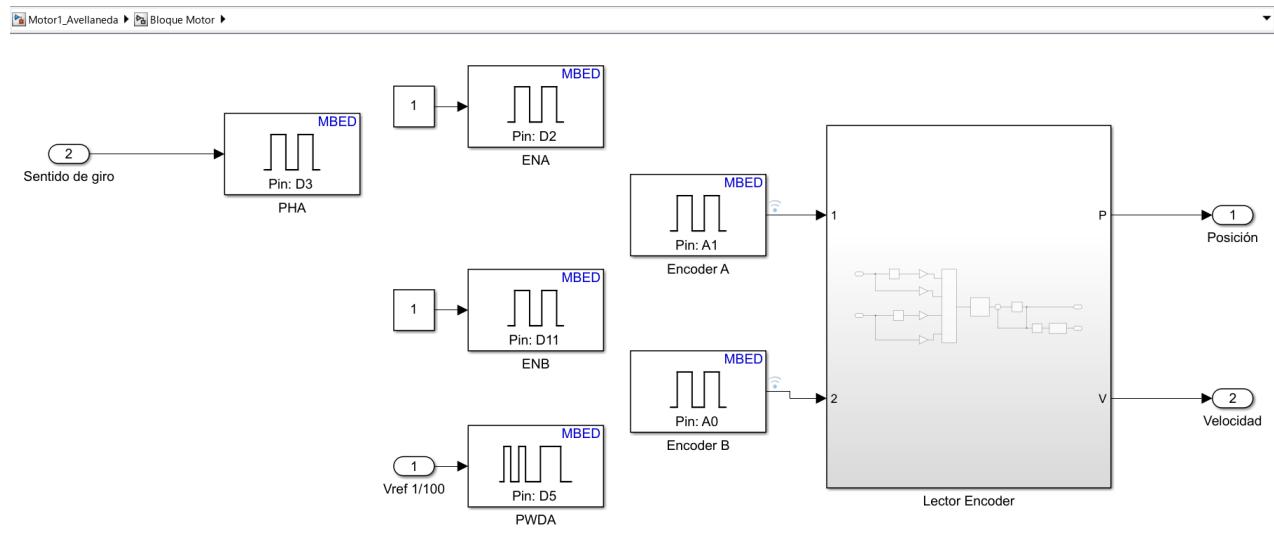


Figura 3.1.2: Vista del bloque motor por dentro. Se observan numerosos bloques *Digital Write* que recogen información y la envían a los diferentes pines. Se destacan los bloques “Encoders” y el subsistema “Lector Encoder”.

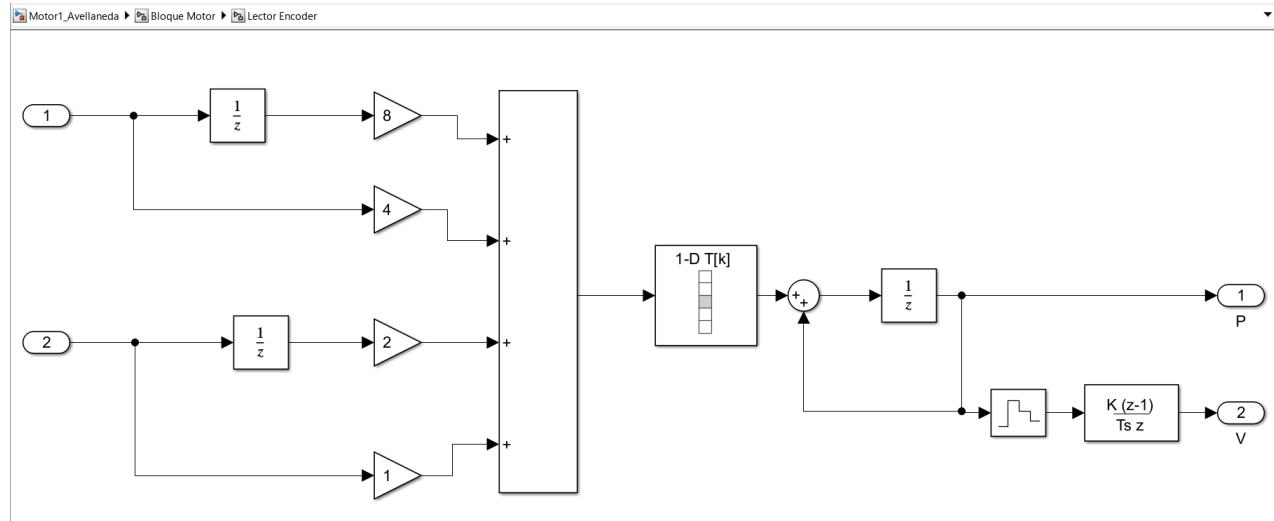


Figura 3.1.3: Vistazo interior al Lector Encoder. Este subsistema recoge las lecturas de los Encoders y las transforma en posiciones y velocidades angulares, ambas expresadas en grados/s.

Así, siguiendo el guión de prácticas se construye en Simulink un modelo de motor en lazo abierto; este modelo puede verse en el archivo “Motor1_Avellaneda” del archivo .zip en el que se incluye esta memoria.

Además, al simular este modelo se consiguen registrar datos de posición y velocidad, que varían en función del voltaje V suministrado por la fuente. Aunque se las medidas se hacen variando el voltaje suministrado en 2V, para un intervalo de 0 a 12V, por una cuestión de espacio se muestran en este documento solo las relativas a 6,10 y 12V. Estos valores de posición y medida registrados pueden observarse en las figuras 3.1.4, 3.1.5 y 3.1.6 respectivamente.

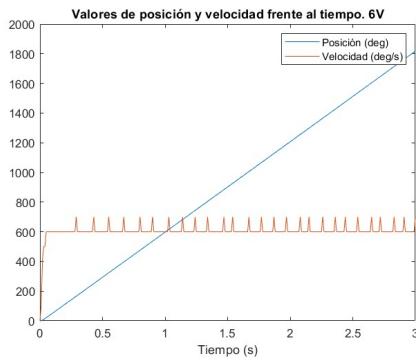


Figura 3.1.4: Valores de posición y velocidad registrados en el motor para un voltaje suministrado de 6V.

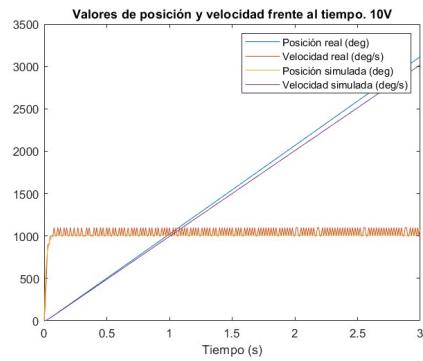


Figura 3.1.5: Valores de posición y velocidad registrados en el motor para un voltaje suministrado de 10V.

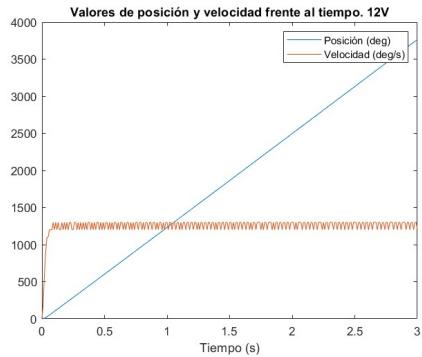


Figura 3.1.6: Valores de posición y velocidad registrados en el motor para un voltaje suministrado de 12V.

En los 3 casos se observa que si bien la posición crece de manera lineal (en todos los casos el tiempo de simulación es de 3s), la velocidad tiende a estabilizarse y oscilar en torno a un valor determinado, y que este valor es mayor cuanto mayor es el voltaje suministrado por la fuente.

Por otro lado es interesante reseñar la lectura de los *encoders*. Estos *encoders* devuelven una señal *en cuadratura*, es decir, que oscila entre 0 y 1 a intervalos definidos variables en función del voltaje suministrado a la fuente. Las figuras 3.1.7,3.1.8 muestran este comportamiento, para un voltaje suministrado de 6V y 12V respectivamente. A lo largo de este documento se estudiarán con más detalle la señal producida por los *encoders*, así como su validez.

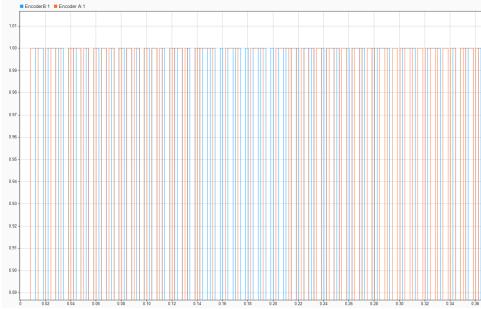


Figura 3.1.7: Lectura de los *encoders* para un voltaje suministrado de 6V.

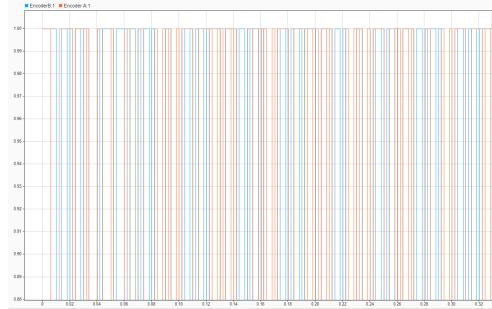


Figura 3.1.8: Lectura de los *encoders* para un voltaje suministrado de 12V.

3.2. Modelo e identificación del motor

Con los datos recogidos en la sección anterior nos es posible estimar de modo aproximado los parámetros k_e y p del motor, algo fundamental para lo sucesivo en la práctica. Para ello, primero hemos de derivar la expresión correspondiente a la posición angular de un motor de continua. Gracias al guion de prácticas sabemos que la posición θ y la velocidad ω vienen determinadas por una ecuación diferencial ordinaria,

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

donde k_e, p son los parámetros antes mencionados y $e(t)$ es una función que representa el voltaje suministrado, que en nuestro caso será constantemente V . Así, podemos calcular de manera numérica la función de posición $\theta(t)$ gracias a la potente herramienta de cálculo simbólico de MATLAB. A pesar de que este cálculo puede consultarse en el archivo "P2_pos.m", se adjunta aquí las líneas de código escritas para resolverlo.

```
syms k_e p V
syms w(t)
equation = diff(w,t) == -p*w + k_e*V;
inicio = w(0) == 0;
omega(t) = dsolve(equation, inicio);

syms theta(t)
equation2 = diff(theta,t) == omega(t);
inicio2 = theta(0) == 0;
theta(t) = dsolve(equation2, inicio2);
```

En consecuencia, aplicando el cálculo simbólico como se indica más arriba, se llega a que la posición angular $\theta(t)$ del motor puede modelarse como

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

La parte exponencial de la ecuación 3.2 corresponde al transitorio, y varía en función del voltaje suministrado. Por tanto, eligiendo correctamente el intervalo de tiempos, podemos despreciar este término y considerar un comportamiento lineal de $\theta(t)$, tal y como se ha indicado en la sección anterior. Así, sobre los datos de posición registrados para diferentes voltajes V se pueden despejar los parámetros k_e y p con métodos algebraicos básicos, y luego considerar estos parámetros como la media de todos ellos. Este ajuste, que se puede consultar en el EXCEL .ajuste”, devuelve unos valores de k_e y p de **6215.40** y **61.44**, respectivamente. En lo que sigue se tomarán estos valores como los parámetros de nuestro motor real, y se usarán para considerar su modelo ideal en Simulink.

Además, conocida la dinámica seguida por el motor (ecuación ??) y los parámetros k_e, p del motor real puede construirse en Simulink un modelo de motor ideal sobre el que trabajar. Este modelo, construido a base de bloques como se nos ha enseñado en las sesiones de laboratorio, puede observarse en la figura 3.2.9; para un análisis más detallado del modelo, consúltese el documento ”Motor2_Avellaneda.slx”.

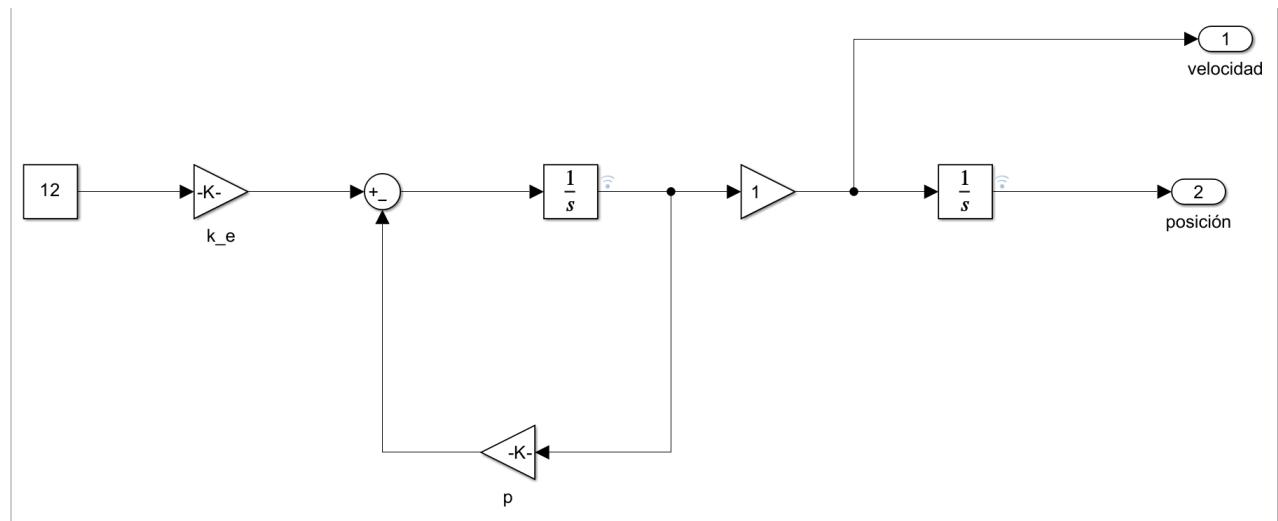


Figura 3.2.9: Modelo ideal de motor realizado en Simulink. El modelo ya incluye los valores k_e y p del motor real sobre el que se trabaja.

Lo esperable al simular este modelo de motor ideal es que devolviese valores similares a los del motor real a voltajes suministrados iguales. En efecto, las figuras 3.2.10, 3.2.11, 3.2.12 muestran una comparativa de los valores simulados y los valores reales obtenidos en la sección anterior, para unos voltajes suministrados de 6,10 y 12V respectivamente. Para voltajes altos podemos ver cómo las posiciones reales y simuladas se separan conforme pasa el tiempo, mientras que para un voltaje suministrado de 6V la separación es prácticamente nula. Además, en los 3 casos se observa que las velocidades reales y simuladas son prácticamente iguales en todo el tiempo de simulación. En consecuencia, podemos concluir que nuestro modelo ideal refleja con fidelidad el comportamiento del motor real.

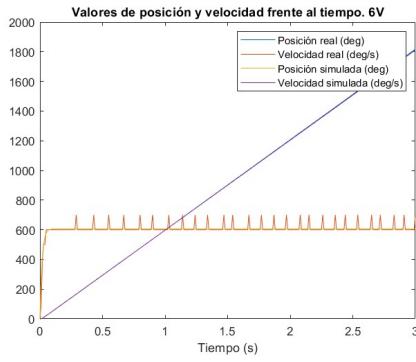


Figura 3.2.10: Comparativa de datos simulados y datos reales para un voltaje de 6V.

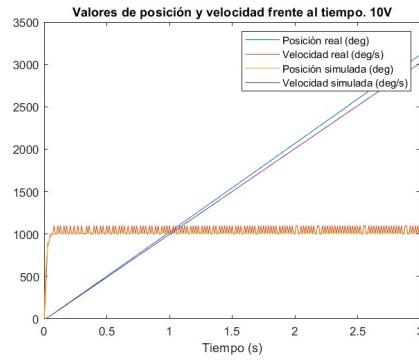


Figura 3.2.11: Comparativa de datos simulados y datos reales para un voltaje de 10V.

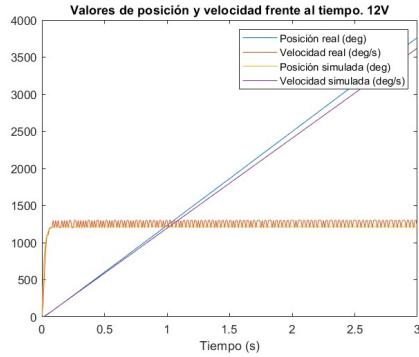


Figura 3.2.12: Comparativa de datos simulados y datos reales para un voltaje de 12V.

4. Control del motor por realimentación de estados estimados

4.1. Un modelo completo del motor

El siguiente paso de la práctica es, sobre el modelo de motor ideal construido en el capítulo anterior, construir un modelo de motor completo que refleje todos los componentes de nuestro motor real, no solo la fuente de corriente continua y el integrador sino también aspectos fundamentales como la señal PWM o las lecturas de los *encoders*. lo primero se consigue mediante un correcto modelado del sistema de alimentación, y lo segundo mediante un correcto modelado de dichos *encoders*.

Así, siguiendo los pasos establecidos en el guion de prácticas, se consigue construir sobre el modelo de motor ideal expuesto en la figura 3.2.9 un modelo completo de motor de corriente continua, en el que se incluyen los aspectos señalados más arriba. Este modelo puede consultarse en la figura 4.1.1; para un análisis más detallado del mismo, consúltese el archivo "Motor3_Avellaneda".

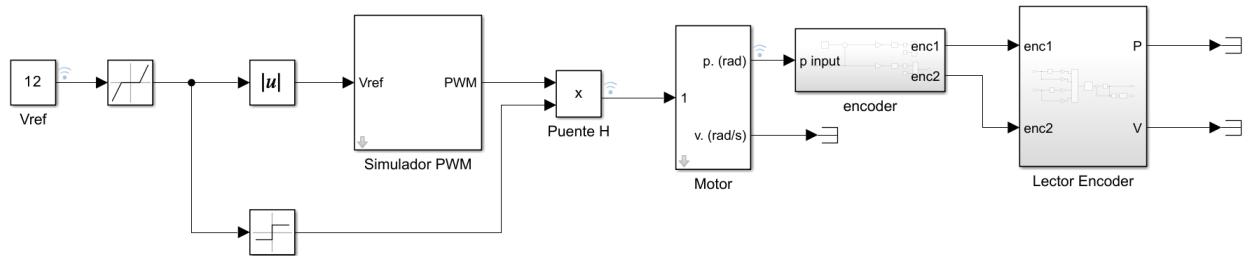


Figura 4.1.1: Modelo de motor completo desarrollado en Simulink.

Como se puede observar en la figura, el modelo contiene un bloque específico para modelar la señal PWM que emplea el motor para recibir un nivel de tensión variable, así como un subsistema que simula los *encoders* del motor y otro que simula su lectura. Además, entre el bloque simulador de PWM y el bloque motor (que contiene el modelo de motor ideal de la figura 3.2.9) se ha colocado un bloque que funciona a modo de puente H, es decir un circuito que permite invertir la polaridad de tensión aplicada a una carga, y que resulta fundamental para el correcto funcionamiento del motor. Todos los bloques se han construido siguiendo estrictamente lo indicado en el guion de prácticas.

Así, para diferentes voltajes V_{ref} de entrada se simula el modelo de motor completo desarrollado en esta parte de la práctica. Concretamente se simulan para unos voltajes suministrados de 6,10 y 12V, tal y como se ha ido haciendo a lo largo de toda la práctica. Los valores de posición y velocidad registrados para estos voltajes pueden consultarse en las figuras 4.1.2, 4.1.3 y 4.1.4, respectivamente.

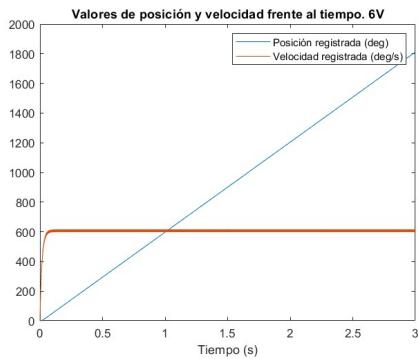


Figura 4.1.2: Valores registrados de posición y velocidad en el modelo completo de motor, para un voltaje suministrado de 6V.

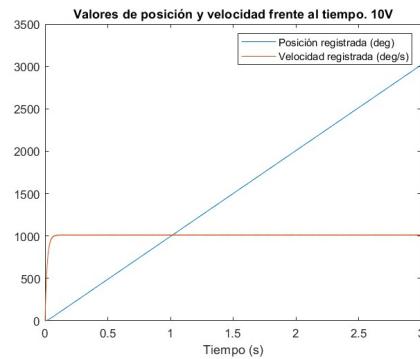


Figura 4.1.3: Valores registrados de posición y velocidad en el modelo completo de motor, para un voltaje suministrado de 10V.

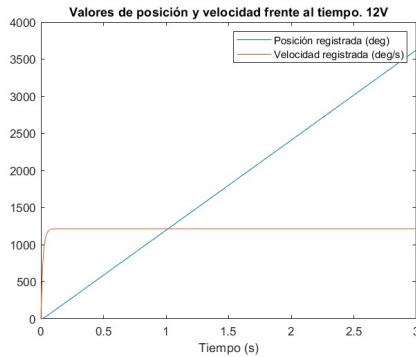


Figura 4.1.4: Valores registrados de posición y velocidad en el modelo completo de motor, para un voltaje suministrado de 12V.

En los 3 casos (si bien es más acusado en el caso de menor voltaje) se tiene un comportamiento similar al observado tanto en el motor ideal como en el real: mientras que la posición crece de una manera lineal, la velocidad rápidamente se estabiliza en torno a un valor fijo (y variable en función del voltaje considerado), para luego oscilar el resto de la simulación en torno a él. En consecuencia, podemos concluir que esta parte de la práctica se ha realizado con éxito

4.2. Diseño de un controlador de la posición del motor por realimentación de estados estimados

4.2.1. Controlador para el motor ideal

Una vez desarrollado en Simulink un modelo completo de motor, que simule a la perfección el motor real del laboratorio, el paso natural a seguir es claro: ¿podemos controlar ese motor de cc y conseguir que se pare donde nosotros queramos?. Este problema es equivalente a, dada una consigna prefijada θ_r , si podemos conseguir que en un tiempo de simulación determinado, la posición del motor alcance θ_r y que la velocidad sea $\omega_r = 0$. Para solventar este problema, en

esta parte de la práctica se diseña un controlador de la posición por realimentación de estados estimados, al que posteriormente se le añade acción integral. Sin embargo, por cuestiones de simplicidad en este documento se expone todo el diseño de manera simultánea.

En primer lugar, consideremos la matriz $C = [1 \ 0]$, y sean A, B las matrices de la ecuación 3.2 respectivamente. Consideramos así la matriz C porque queremos observar (y en consecuencia controlar) la posición, que es la primera entrada del vector x solución del sistema 3.2. Así, aplicando la teoría vista en clase, la matriz de observabilidad $O = obsv(A, C)$, que es $2x2$, tiene rango 2, y por tanto el sistema es observable. En consecuencia, se le puede aplicar control por realimentación de estados estimados.

El siguiente paso es construir la matriz L de ganancia, de forma que $A - LC$ sea una matriz de estabilidad. Para ello, se usa el método de Lyapunov de autovalores para colocar los polos del sistema estimado en $-1,2p$. El código de MATLAB escrito para obtener L , a pesar de que puede consultarse en "P4_realimentacion mlx" (de hecho, se recomienda mirarlo ahí porque está perfectamente explicado) es el siguiente:

```

clear;
clc;

% Definición de parámetros
p = 61.441545;
ke = 6215.396692;

% Matrices del sistema
A = [0 1; 0 -p];
B = [0; ke];
C = [1 0];

% Cálculo de la observabilidad
O = obsv(A, C);
rank(O)

% Obtención de L por Lyapunov
mu = 1.2 * p;
W = lyap(-mu * eye(2) - A', C' * C);
L = (C * W^-1)' / 2;

```

Una vez diseñado el controlador de posición por realimentación de estados estimados, el siguiente paso es construir un controlador de acción integral, que lleve al motor hasta una posición θ_r prefijada y lo pare ahí. Para ello, consideramos las matrices ampliadas $A_{amp} = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix}$ y $B_{amp} = \begin{bmatrix} B \\ 0 \end{bmatrix}$. Como la matriz de controlabilidad $control = ctrb(A_{amp}, B_{amp})$, que es de tipo $3x3$, tiene rango 3, se tiene que el par ampliado (A_{amp}, B_{amp}) es controlable y por tanto se le puede aplicar control integral a la posición. Así, empleando en esta ocasión el comando *acker* de MATLAB podemos construir la matriz K del controlador, así como la acción integral K_I .

En esta ocasión, se colocan los polos del sistema en $-0,5p$, para que así el sistema converja más despacio, y fijamos la acción integral en $-1,5p$. Nuevamente, aunque el código puede consultarse más detalladamente en el archivo antes mencionado, se adjunta aquí una copia del mismo relativo a la parte comentada.

```
% Matrices ampliadas
Amp = [A zeros(2,1); C 0];
Bamp = [B; 0];

% Verificación de controlabilidad
control = ctrb(Amp, Bamp);
rank(control);

% Cálculo de las ganancias
Kamp = acker(Amp, Bamp, [-p1 -p1 -1.5*p]); %Ultimo --> acc.integral
K_I = Kamp(3);
K = Kamp(1, 1:2);
```

Por último, también podemos calcular una ganancia de acción directa F_c que lleve el motor a la posición indicada por el escalón de entrada $y = \theta_r$. Esta ganancia (o *feedforward*) F_c tiene como ecuación

$$F_c = (C(BK - A)^{-1}B)^{-1} \quad (3)$$

Y en el caso de nuestro motor, toma un valor de $F_c = 0,1518$

Finalmente ya tenemos todos los ingredientes necesarios para construir nuestro sistema de control por realimentación de estados y acción integral. Para montarlo, se siguen las instrucciones del guion de prácticas paso a paso, y se desarrolla en Simulink un modelo de bloques sobre el motor ideal de la práctica 2, ilustrado en la figura 3.2.9. Este sistema de control puede consultarse tanto en la figura 4.2.5 como en el archivo "Motor4_Avellaneda.slx"

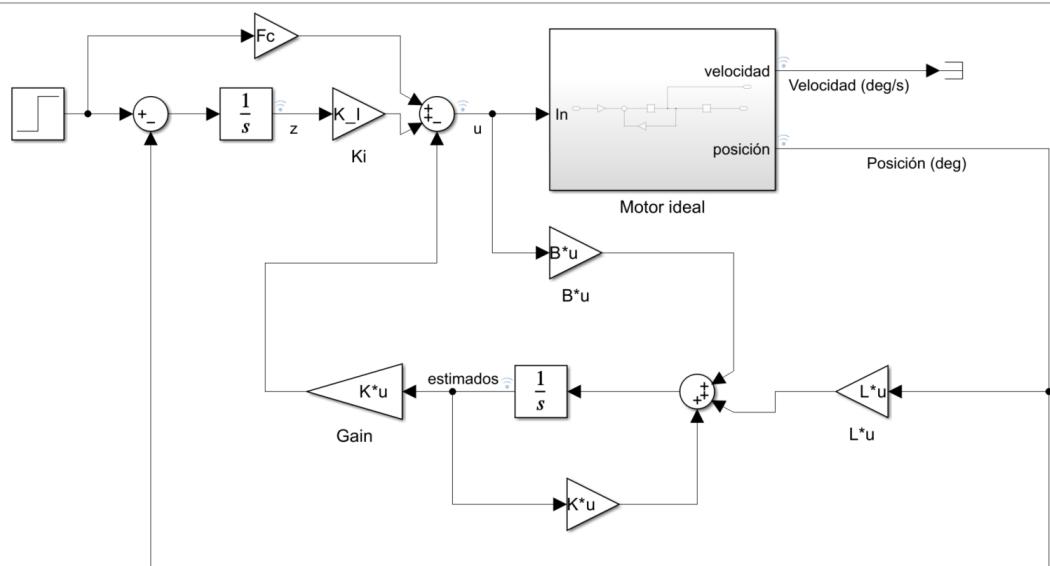


Figura 4.2.5: Sistema de control por realimentación de estados y acción integral sobre el motor ideal de la práctica 2.

Así, al hacer correr la simulación y fijar un $\theta_r = 360$ (el equivalente a una vuelta completa) en un tiempo de simulación de 3s, se comprueba que efectivamente el motor alcanza esa posición y se para (véase figuras 4.2.6 y 4.2.7)

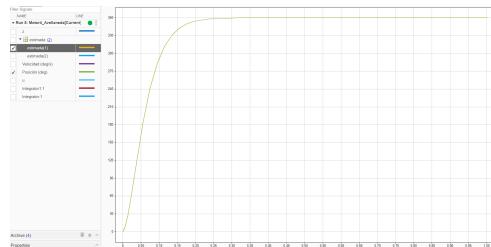


Figura 4.2.6: Datos de posición registrados para un control por realimentación de estados y acción integral.

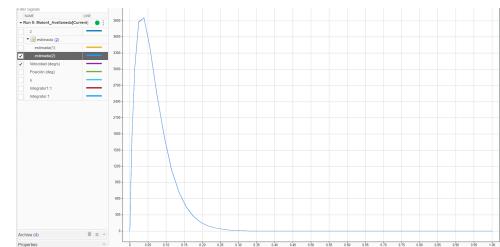
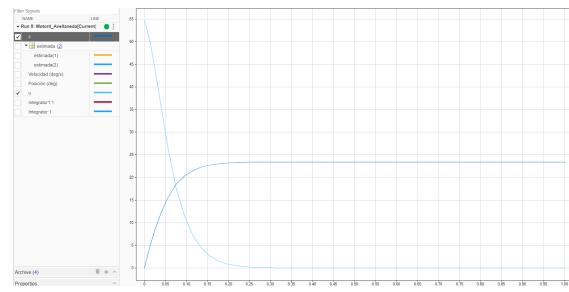


Figura 4.2.7: Datos de velocidad registrados para un control por realimentación de estados y acción integral.

Además, en las figuras se observa que la diferencia entre los valores reales y los valores estimados es prácticamente nula, lo que nos hace bien que el sistema de control está bien diseñado. Por otro lado, es interesante estudiar también la señal de entrada u y el efecto del error de posición inicial z sobre esta señal. La figura 4.2.8 muestra que si bien la entrada parte desde valores muy altos y rápidamente converge a 0, la señal de error tiende a estabilizarse en torno a un valor fijo.

Figura 4.2.8: Variación de la señal de entrada u y error z

Por otro lado, es interesante estudiar el efecto de la posición de los polos del sistema. En los casos anteriores se han colocado los polos en $-0,5p$, lo que ha provocado que el sistema convergiese más lentamente de lo que lo haría de forma natural. Sin embargo, si en lugar de desplazarnos hacia valores más positivos, los desplazamos hacia valores más negativos, el sistema debería converger más rápidamente. En efecto, si se colocan los polos en, por ejemplo, $-2p$, el sistema converge más rápidamente. Además, una rápida convergencia implicaría que la velocidad debería realizar una gran oscilación, alcanzando picos muy altos antes de bajar a 0. Sin embargo, esto implicaría que la señal de entrada u suministrada al sistema debiera ser muy alta, produciendo así un gran costo físico y energético. Estas tendencias se observan con claridad en las figuras 4.2.9, 4.2.10 y 4.2.11 respectivamente.

De manera inversa, colocar los polos del sistema en valores mas positivos (por ejemplo, en $-0,01p$) provocaría una convergencia mucho más lenta del sistema, pero a un coste energético mucho menor. Este comportamiento puede observarse en las figuras 4.2.12, 4.2.13 y 4.2.14 respectivamente.

Práctica de control

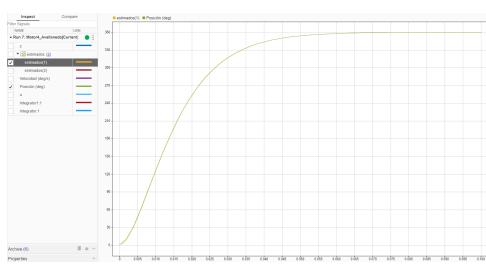


Figura 4.2.9: Datos de posición registrados para unos polos en $-2p$. Se observa una rápida convergencia al valor de posición requerido.

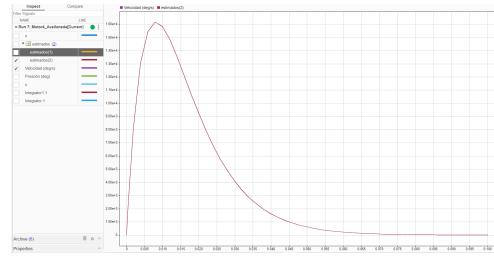


Figura 4.2.10: Datos de velocidad registrados para unos polos en $-2p$. Tras una rápida y amplia oscilación, la velocidad rápidamente tiende a 0 y el motor se para.

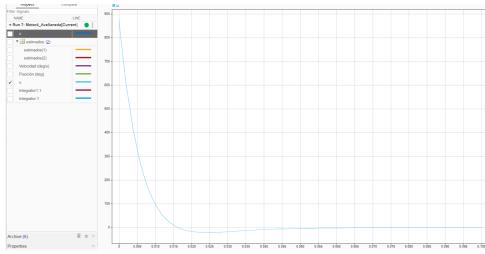


Figura 4.2.11: Datos de entrada registrados para unos polos en $-2p$. Si bien la señal de entrada rápidamente tiende a 0, inicialmente es muy alta, lo que indica el alto coste energético de esta rápida convergencia.

Práctica de control

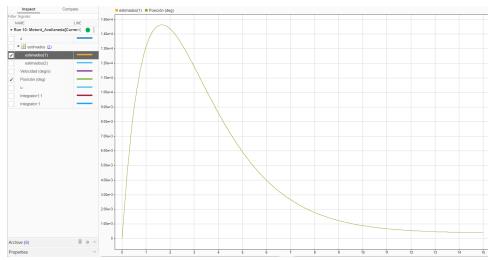


Figura 4.2.12: Datos de posición registrados para unos polos en $-0,01p$. Se observa una muy lenta convergencia al valor de posición requerido.

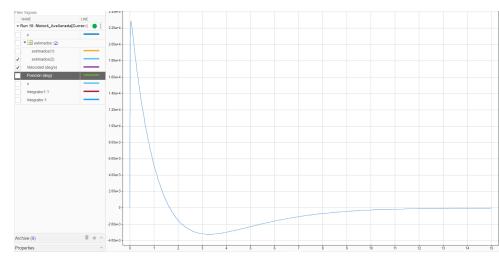


Figura 4.2.13: Datos de velocidad registrados para unos polos en $-0,01p$. Tras una oscilación, la velocidad lentamente tiende a 0 y el motor se para.

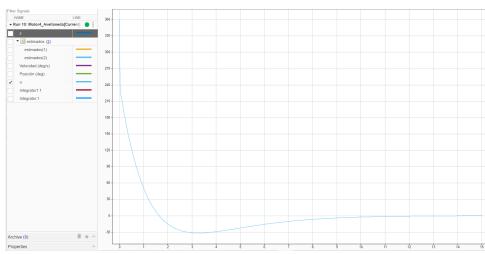


Figura 4.2.14: Datos de entrada registrados para unos polos en $-0,01p$. Si bien la señal de entrada tiende a 0 después de oscilar, inicialmente no es muy alta, lo que indica el bajo coste energético de esta convergencia.

Finalmente, es necesario comentar el efecto de la acción directa F_c sobre la convergencia del sistema. Hasta el momento todas las simulaciones se han hecho teniendo en cuenta esta acción directa e incluyéndola en el modelo. Esta acción directa, dependiente de la matriz K y por tanto de los polos del sistema, da un impulso extra a la convergencia del sistema y lo lleva directamente a la posición requerida, donde el motor se para. Sin embargo, al eliminar la acción directa, la dinámica del sistema cambia: en lugar de llevarlo directamente, ahora el sistema alcanza ahora un pico de posición superior al valor prefijado, y posteriormente baja y se estabiliza en torno a él. Recíprocamente, el comportamiento de la velocidad es similar: después de alcanzar un pico, la velocidad desciende hasta ser negativa (i.e. cambia el sentido de giro) y una vez se alcanza θ_r , esta se estabiliza. Este comportamiento puede observarse con claridad en las figuras 4.2.15 y 4.2.16 respectivamente.

Práctica de control

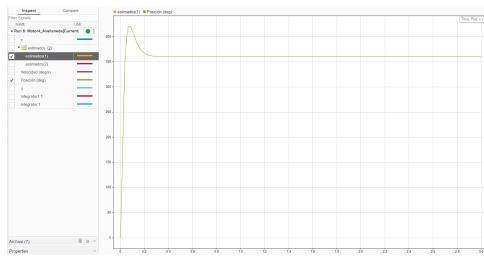


Figura 4.2.15: Datos de posición registrados para un control por realimentación de estados y acción integral, sin efecto de acción directa F_c .

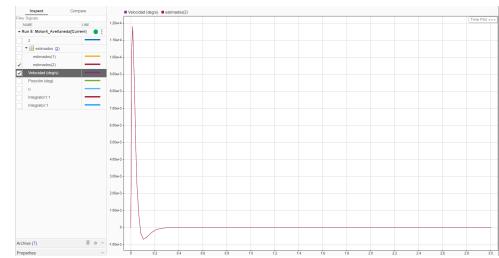


Figura 4.2.16: Datos de velocidad registrados para un control por realimentación de estados y acción integral, sin efecto de acción directa F_c .

Nuevamente, por cuestiones de espacio y de hacer esta memoria lo más clara y directa posible no se tienen en consideración más casos de convergencia en función de la variación de los polos del sistema. Si el Profesor o el lector desea estudiar más ejemplos, se recomienda modificar los valores de los polos en "P4_realimentacion mlx" luego hacer correr el modelo en "Motor4_Avellaneda.slx"

4.2.2. Discretización del controlador para control del motor real

Hasta ahora hemos considerado el motor (y su sistema de control) como algo continuo, cuando en la realidad esto no es así, y el motor es controlado y observado por un sistema digital. Esto significa que la señal de entrada (voltaje) que recibe el motor y la señal de salida (lectura de *encoders*) no se realiza de modo continuo, sino discreto, y marcado por el periodo de muestreo al que la placa ST Nucleo envía o recibe datos por sus pines. Por tanto, para modelar realmente el sistema y hacerlo lo más similar posible al motor real, hemos de transformar el sistema 3.2 en uno de tiempo discreto, con un periodo de muestreo fijado en $T = 0,0001\text{s}$. Las ecuaciones del modelo discreto son

$$\begin{aligned} x(n+1) &= Fx(n) + Gu(n) \\ y(n) &= Cx(n) + Du(n) \end{aligned} \quad (4)$$

donde F, G son las matrices discretizadas de A, B respectivamente, C se mantiene igual y D se toma como 0. Así, pese a que el proceso de discretización del sistema puede consultarse en detalle en el archivo "P5_discretizacion mlx", se adjunta aquí el código de MATLAB que discretiza el sistema.

```
clear;
clc;
p = 61.441545;
ke = 6215.396692;

A = [0 1; 0 -p];
B = [0; ke];
C = [1 0];
Fc = 0.1518;
p1 = 0.5 * p;
```

```

p_integral = 0.7 * p;

% Escogemos un tiempo de muestreo T
T = 0.0001;
sys = ss(A, B, C, zeros(1, 1));
sysd = c2d(sys, T);
F = sysd.A;
G = sysd.B;
C = sysd.C;
D = sysd.D;

```

El papel de las matrices A, B, C, D en el caso discreto es análogo al de las matrices A, B, C, D en el caso continuo, y podemos en consecuencia aplicar los mismos métodos y herramientas vistas en teoría y desarrolladas en la sección anterior. Sin embargo, previo a esto hay que tener en cuenta tres consideraciones: la discretización de los polos, la discretización de la acción directa F_c y la discretización de la acción integral.

En cuanto a la **discretización de los polos**, gracias al guion de prácticas sabemos que si λ_i es un polo en el caso continuo, en el caso discreto se transforma como $\Lambda_i = e^{\lambda_i T}$, donde T es el tiempo de muestreo. Así, las matrices de ganancias K y L en el caso discreto pueden emplearse con el comando *acker*, y quedan de la forma

$$K = \text{acker}(F, G, [\Lambda_{p1} \dots \Lambda_{pn}])$$

$$L = (\text{acker}(F^T, C^T, [\Lambda_{o1} \dots \Lambda_{on}]))^T$$

El proceso de discretización de los polos se muestra en las siguientes líneas de código:

```

Polos_realimentacion = [-p1 -p1 -p_integral]; %Último --> acc.integral
Polos_observador = [-p -p];

% Ahora, discretizamos
Polos_realimentacion_discreto = exp(Polos_realimentacion * T);
Polos_observador_discreto = exp(Polos_observador * T);

```

Por otro lado la **discretización de la acción directa** puede hacerse de dos formas: a partir de la acción directa en caso continuo F_c , mediante un cálculo matricial larguísimo y muy engorroso, o mediante la ecuación

$$f = \left[C (I - (F - GK))^{-1} G \right]^{-1} \quad (5)$$

Por cuestiones de sencillez se obtiene la acción directa vía ecuación 4.2.2; debido a su simplicidad, no se incluye aquí el código que devuelve f , aunque puede consultarse en el archivo antes mencionado.

Finalmente, la **discretización de la acción integral** viene de considerar la ecuación

$$\begin{bmatrix} x(n+1) \\ z(n+1) \end{bmatrix} = \begin{bmatrix} F & 0 \\ TC & 1 \end{bmatrix} \begin{bmatrix} x(n) \\ z(n) \end{bmatrix} + \begin{bmatrix} G \\ 0 \end{bmatrix} u(n) + \begin{bmatrix} 0 \\ -T \end{bmatrix} \theta_r \quad (6)$$

donde θ_r es, una vez más, la posición prefijada. Así, con esta información y la expuesta más arriba, podemos sacar las matrices F_{amp} y G_{amp} del sistema discreto, para posteriormente obtener tanto las matrices de ganancias K y L como la acción integral (discreta) K_I :

```

Famp = [F zeros(2,1); T*C 1];
Gamp = [G; 0];

Kdis = acker(Famp, Gamp, Polos_realimentacion_discreto);
K = Kdis(1, 1:2);
KI = Kdis(3);

L = acker(F', C', Polos_observador_discreto)';

```

Así, ya tenemos todos los ingredientes para construir el controlador discreto en Simulink. Para hacerlo, tomamos el modelo desarrollado en la práctica 3 (e ilustrado en la figura 4.2.5 y cambiamos las matrices del caso continuo por las del caso discreto. También se sustituyen los valores de la acción integral K_I y la acción directa F_c por los nuevos K_I, f . El modelo puede consultarse tanto en la figura 4.2.17 como en el archivo "Motor5_Avellaneda.slx" (aunque nuevamente se pide antes correr el archivo "P5_discretizacion mlx" para que el modelo coja los datos del *workspace*).

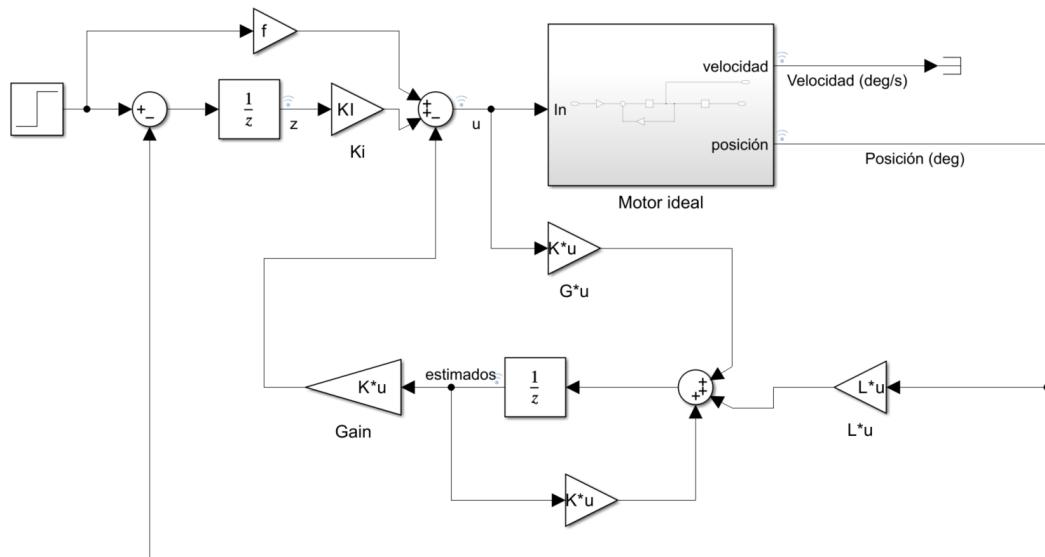


Figura 4.2.17: Sistema de control por realimentación de estados estimados y acción integral para un motor ideal. Caso discreto: se han sustituido los integradores por bloques *UnitDelay* con un tiempo de muestreo T .

Así, si simulamos este modelo y establecemos un valor de consigna de, por ejemplo 540° (el equivalente a vuelta y media del motor) se observa que después de tan solo 0.3s y amplias

Práctica de control

oscilaciones, tanto la posición y la velocidad tienden a estabilizarse en torno a los valores requeridos (540° y $0^\circ/\text{s}$ respectivamente). Además, se observa que la señal de entrada u también oscila, alcanzando un pico de aproximadamente 4000° , para finalmente acabar en 0. Finalmente, observamos también que la señal de error z tiende a 0, lo que significa que el error entre los valores reales y los estimados prácticamente se anula. Estos comportamientos quedan ilustrados en las figuras 4.2.18, 4.2.19 y 4.2.20.

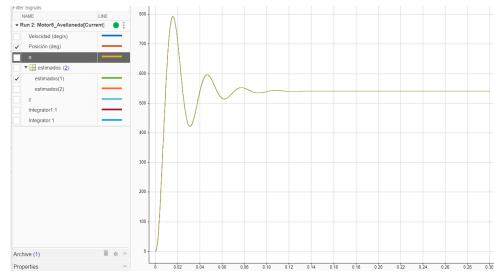


Figura 4.2.18: Datos de posición registrados para unos polos dados en el caso discreto.

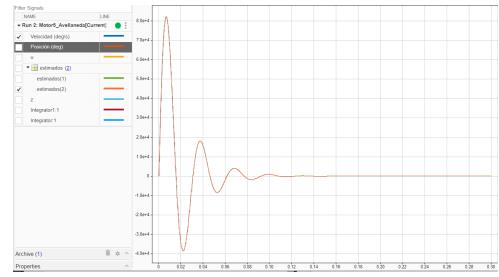


Figura 4.2.19: Datos de velocidad registrados para unos polos dados en el caso discreto.

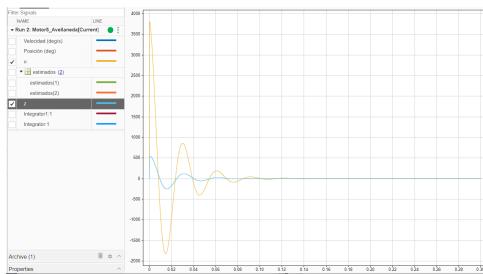


Figura 4.2.20: Datos de señal de entrada u y error z registrados para unos polos dados en el caso discreto.

Por otro lado es interesante observar el efecto de la acción directa f en la estabilización del sistema en torno al θ_r dado. Si se elimina el efecto de esta acción (esto es, fijar un 1 en el bloque *gain* que lleva la acción) se observaría cómo en lugar de llevar de manera "directa" la posición al valor fijado, el sistema daría muchas más oscilaciones ante de finalmente pararse en él. En efecto, al simular el modelo con el efecto de acción directa eliminado, estas tendencias quedan perfectamente reflejadas en los datos de posición, velocidad y entrada, y pueden consultarse en las figuras 4.2.21, 4.2.22 y 4.2.23 respectivamente.

Práctica de control

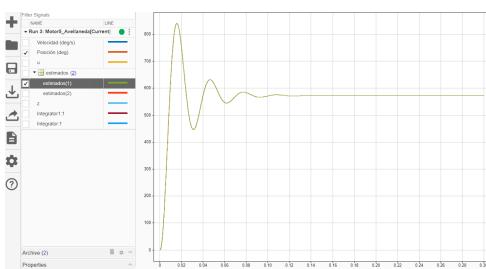


Figura 4.2.21: Datos de posición registrados para unos polos dados en el caso discreto, eliminada la acción directa f .

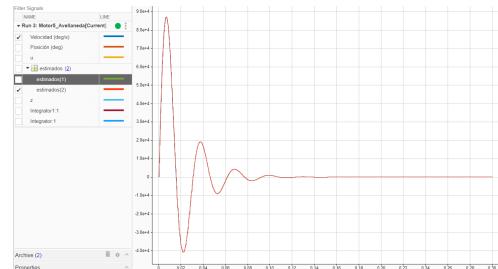


Figura 4.2.22: Datos de velocidad registrados para unos polos dados en el caso discreto, eliminada la acción directa f .

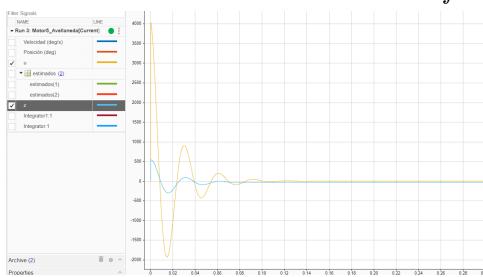


Figura 4.2.23: Datos de señal de entrada u y error z registrados para unos polos dados en el caso discreto, eliminada la acción directa f .

Finalmente, de manera análoga a lo realizado en la sección anterior conviene estudiar el ritmo de convergencia del sistema en función de la colocación de sus polos. Si en lugar de fijar los polos del sistema en $-0,5p$ se fijasen en valores más negativos, por ejemplo en $-2p$, se tendría una convergencia muchísimo más rápido del sistema al θ_r requerido, pero a un coste energético (valores de la señal de entrada u) increíblemente altos y oscilaciones mucho más violentas. Por el contrario, si se fijasen estos polos en valores más positivos y cercanos al 0, como por ejemplo en $-0,05p$, se tendría un coste energético más bajo a costa de un mayor tiempo de convergencia. Estos comportamientos quedan ilustrados a la perfección en las figuras 4.2.24, 4.2.25 y 4.2.26 (para polos más negativos) y en las figuras 4.2.27, 4.2.28 y 4.2.29 (para polos más positivos).

Práctica de control

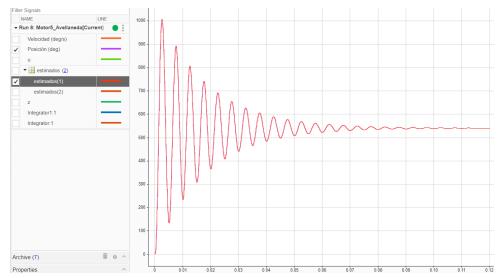


Figura 4.2.24: Datos de posición registrados para unos polos situados en $-2p$ en el caso discreto.

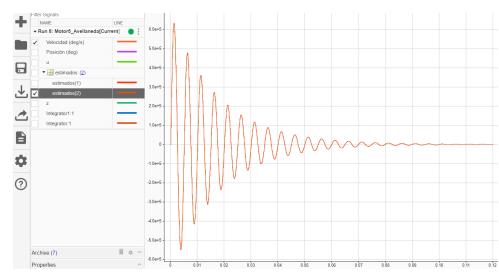


Figura 4.2.25: Datos de velocidad registrados para unos polos situados en $-2p$ en el caso discreto.

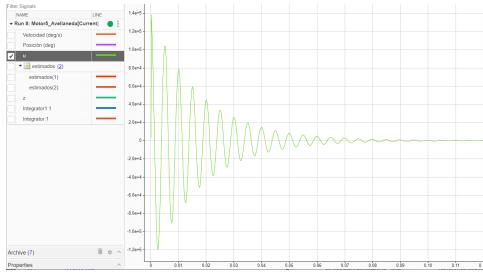


Figura 4.2.26: Datos de señal de entrada u registrados para unos polos situados en $-2p$ en el caso discreto.

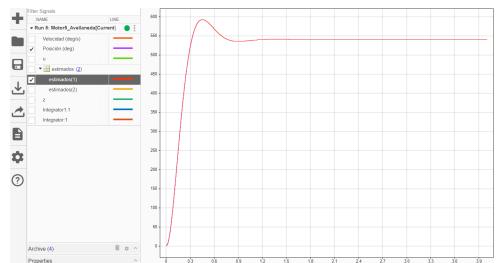


Figura 4.2.27: Datos de posición registrados para unos polos situados en $-0,05p$ en el caso discreto.

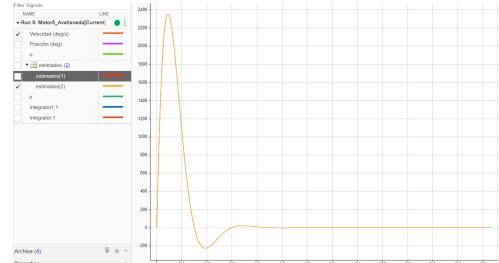


Figura 4.2.28: Datos de velocidad registrados para unos polos situados en $-0,05p$ en el caso discreto.

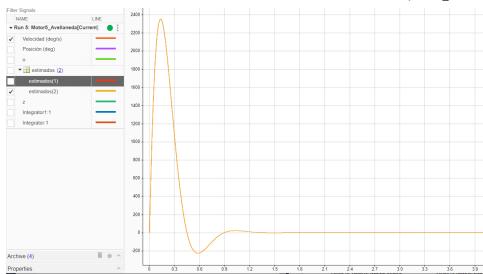


Figura 4.2.29: Datos de señal de entrada u registrados para unos polos situados en $-0,05p$ en el caso discreto.

4.3. Controlador discreto para el modelo completo del motor y el motor real

Finalmente tenemos todos los ingredientes necesarios para trasladar nuestro sistema de control (discreto) realizado sobre un motor ideal, al motor real que se usa en el laboratorio y cuyo modelado puede consultarse en la figura 3.1.1. Así tomando un tiempo de muestreo de $T = 0,0001s$ y un valor de consigna $\theta_r = 540$ (o vuelta y media del motor) es posible aplicar el sistema de control y de acción integral a un sistema físico real, trasladando las ecuaciones al mundo real. Además, para evitar un exceso de oscilaciones, que a la larga pueden dañar el sistema físico o desestabilizarlo (a este problema se le conoce como *wind-up*, se introduce un sistema ***anti wind-up***, consistente en un atenuador de la acción integral mientras el actuador está saturado, es decir, mientras el valor de la entrada demandada por el controlador supera el valor que éste realmente puede dar. Sin embargo, en este caso no se desconecta directamente la acción integral, sino que se altera la entrada del error de posición al integrador añadiendo un nuevo término proporcional a la diferencia entre la entrada u calculada por el controlador y valor máximo de la salida. Este valor de la constante - llamada *kamp* - se ajusta habitualmente en función de las características que se quieran dar a la salida.

Así, la figura 4.3.30 muestra el sistema de control y realimentación implementado sobre el motor real ilustrado en la figura 3.1.1. A este modelo, además, se le ha incluido un subsistema que simula el sistema *anti Wind-up* antes explicado. A pesar de que para construirlo se ha seguido estrictamente lo indicado en el guion de prácticas, si se desea un análisis más exhaustivo puede consultarse el modelo en "Motor6_Avellaneda.slx". Sin embargo, a la hora de correr el modelo, es preciso hacer correr antes el archivo "P6_implementacion mlx", ya que este modelo coge los datos del *workspace* una vez corrido el *live script*.

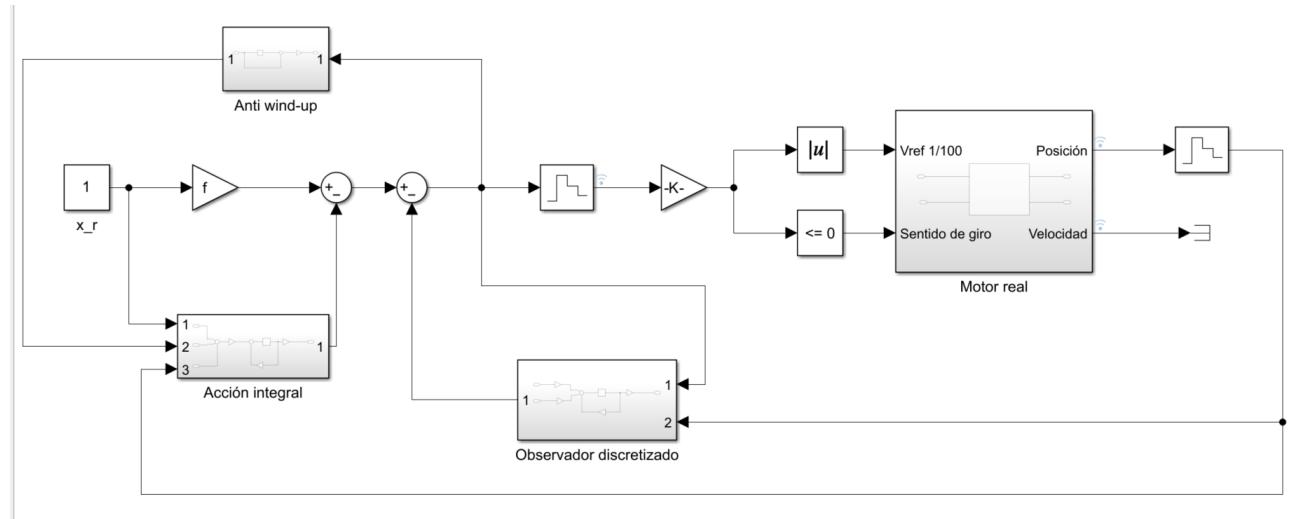


Figura 4.3.30: Modelo real del motor, con sistema de control por estados estimados, acción integral y acción directa ya incorporado. Se incluye también un subsistema *anti Wind-up* para dar más robustez al sistema físico.

Así, si se hace funcionar el motor real con el sistema de control implementado y se fijan de antemano unos valores de consigna de, por ejemplo, 540° y 720° (lo equivalente a vuelta y media y dos vueltas respectivamente) lo esperable es que tras un tiempo finito el motor alcance estos valores y se parase. En efecto, esto puede comprobarse que sucede consultando las figuras 4.3.31

Práctica de control

y 4.3.32

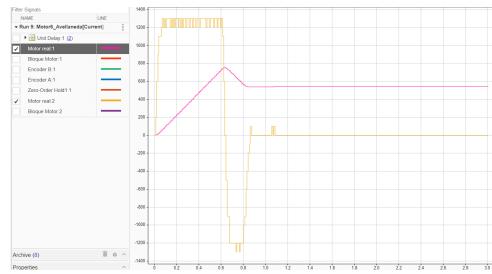


Figura 4.3.31: Datos de posición y velocidad registrados en un motor real, con un valor de consigna de 540° .

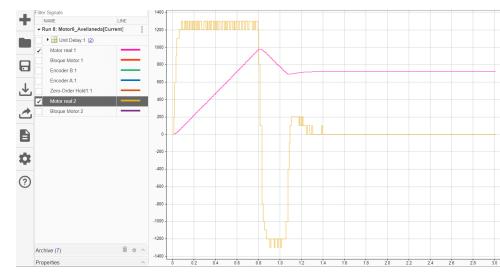


Figura 4.3.32: Datos de posición y velocidad registrados en un motor real, con un valor de consigna de 720° .

Mientras que la posición alcanza un pico y posteriormente baja al valor pedido (i.e., el motor gira en sentido contrario), se observa que la velocidad alcanza un máximo, para posteriormente bajar y cambiar de signo (pasando precisamente por el valor 0, es decir el de cambio de sentido, en el instante en que la posición alcanza el máximo). Además, es interesante observar que las oscilaciones de posición y velocidad no son excesivas. La frecuencia de estas oscilaciones está directamente relacionada con el valor de la constante $kamp$ del sistema *anti wind-up* integrado en el motor. A mayor valor de la constante (figura 4.3.33) se tendrán menos oscilaciones, y a menor valor de ella (figura 4.3.34), más.

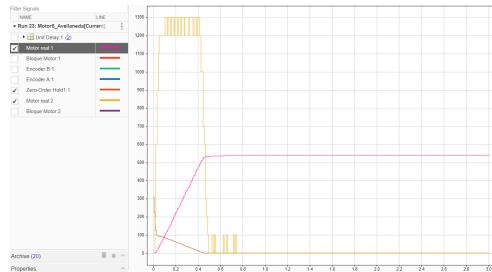


Figura 4.3.33: Datos de posición y velocidad registrados en un motor real, con un valor de consigna de 540° y un $kamp$ de 5.

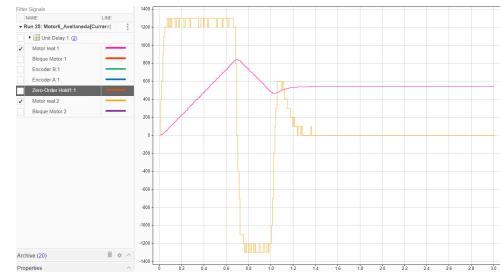


Figura 4.3.34: Datos de posición y velocidad registrados en un motor real, con un valor de consigna de 540° y un $kamp$ de 0.5.

Por otro lado, uno puede ir un paso más allá y preguntarse por el efecto que tienen tanto la acción directa f como la acción integral K_I en el control del sistema. Si se elimina el efecto de la acción directa (esto es, fijar como 1 el bloque *gain* del sistema que la contiene) se observa cómo la posición sigue más o menos la trayectoria seguida anteriormente (lo que indica que esta trayectoria depende en su mayoría de la acción integral) mientras que la velocidad oscila más que cuando se aplicaba f . Por el contrario, si se elimina la acción integral del sistema se observa cómo la velocidad, después de alcanzar un pico máximo, oscila fuertemente alrededor del valor 0, mientras que la posición se estabiliza más lentamente en torno a la consigna. Los efectos de eliminar la acción directa f y la acción integral K_I del sistema de control quedan reflejados en las figuras 4.3.35 y 4.3.36 respectivamente.

Práctica de control

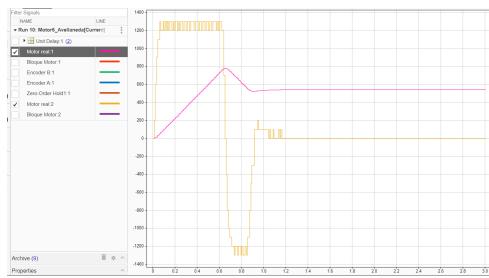


Figura 4.3.35: Datos de posición y velocidad registrados en un motor real, con un valor de consigna de 540° y sin efecto de la acción directa f .

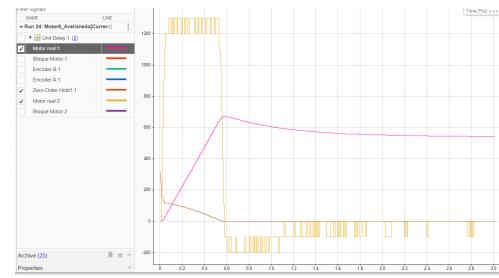


Figura 4.3.36: Datos de posición y velocidad registrados en un motor real, con un valor de consigna de 540° y sin efecto de la acción integral K_I .

Finalmente, y aunque no aparece en el guion de prácticas, uno puede hilar aún más fino y diseñar un sistema de control que optimice al máximo el valor de la señal de entrada u . Supongamos que, por ejemplo, no se quiere u supere el valor de 20 por su alto costo energético. En tal caso se puede diseñar el regulador cuadrático estableciendo la matriz de pesos \bar{R} según el criterio de Byson, y posteriormente resolver la ecuación de Riccati para obtener la matriz de ganancia K . Aunque este cálculo puede encontrarse en "P6_LQR.mlx", se adjuntan aquí las líneas de código empleadas para ello:

```

clear;
clc;

p = 61.441545;
ke = 6215.396692;

A = [0 1;0 -p];
B = [0;ke];
C = [1 0];

%Primero, discretizamos el sistema con T=0.0001s
T = 0.0001;
sys = ss(A,B,C,zeros(1,1));
sysd = c2d(sys,T);
Ad = sysd.A;
Bd = sysd.B;
Cd = sysd.C;
Dd = sysd.D;

%Establecemos las matrices de pesos

Qbar = eye(2); %no queremos limitadores en los estados
Rbar = 1/400; %Criterio Byson%
H = zeros(2,1);
G = eye(2);
rho = 2;

```

```
%Diseñamos LQR
Q = G'*Qbar*G;
R = H'*Qbar*H+rho*Rbar;
N = G'*Qbar*H;
[K,P,la]= dlqr(Ad,Bd,Q,R,N); %eq. de Riccati
```

Así, si se sustituye la matriz K empleada hasta ahora por la obtenida mediante el diseño del *LQR*, y se hace funcionar el motor, el valor u de la entrada no debería superar el 20. Por cuestiones de tiempo y problemas en el *hardware* (y a que por alguna razón no se ha iniciado el modelo) no ha dado tiempo a probar esta matriz K en el laboratorio, pero se invita a probarlo.

5. Conclusiones

A lo largo de todo el cuatrimestre, y no sin grandes esfuerzos, se ha conseguido diseñar un sistema de control para un motor de corriente continua, que consigue parar el motor en un ángulo previamente fijado. Para ello, partiendo de unas ecuaciones de estado puramente físicas se ha diseñado con Simulink un modelo de motor ideal que reflejase la dinámica del sistema. Posteriormente, sobre ese motor ideal se ha construido un sistema de control por realimentación de estados estimados y de acción directa e integral. Además, se ha estudiado el efecto de dicha acción integral y directa, así como la posición de los polos del sistema, en el ritmo de convergencia del mismo. Las simulaciones hechas sobre este modelo han devuelto lo buscado, estabilizar el motor en torno a un ángulo dado y una velocidad nula, con lo que el último paso ha sido implementar este sistema de control en un motor real. Para ello, primero se ha discretizado el sistema, estableciendo un tiempo de muestreo de $T = 0,0001s$, y finalmente se ha implementado en un modelo de motor real, diseñado también con Simulink y conectado a una placa ST Nucleo que controla un motor en el "mundo real". Con todo ello se ha conseguido llevar un motor real a un ángulo θ_r prefijado, y pararlo ahí.

En consecuencia, se puede concluir que la práctica se ha realizado con éxito.

Anexos

Anexo A: distribución de las prácticas en función de las sesiones de laboratorio

Se adjunta a continuación una tabla (1) en el que se muestran la distribución de las prácticas a lo largo de las sesiones de laboratorio. Las prácticas se han realizado en el Laboratorio de Sistemas Digitales de la segunda planta de la Facultad de CC Físicas.

En algunas sesiones aparecen más de dos prácticas hechas en la misma sesión. Esto se debe a que en muchas ocasiones ha sido posible terminar a la hora la práctica, y se ha tenido que terminar bien en casa o bien en tutorías. Por tanto, el trabajo en el laboratorio relativo a estas prácticas se ha basado únicamente en tomar datos del motor real, procesándolos y comparándolos en casa.

Además, la mayor parte del trabajo relativo a la construcción de los modelos también se ha hecho en casa y por tanto no se incluye en estas sesiones de prácticas.

Fecha de la sesión	Práctica realizada
30/09/2024	Práctica 1
14/10/2024	Práctica 1
28/10/2024	Práctica 1
04/11/2024	Prácticas 1 y 2
18/11/2024	Prácticas 3 y 4
25/11/2024	Práctica 5
09/12/2024	Práctica 6

Cuadro 1: Distribución de las prácticas en función de la sesión. No se incluye el trabajo en casa ni de tutorías (que ha sido mucho)

Anexo B: contenidos de cada práctica y localización en la memoria de prácticas

Práctica	Breve resumen	Localización	Archivos relacionados
1	- Identificación de los componentes - Familiarizarse con Simulink - Construcción del modelo de motor real - Primera toma de datos	- 2.1 y 2.2 - 3.1	- Motor1_Avellaneda.slx - datos_real.mat
2	- Derivación de la función de posición - Obtención de k_e y p - Construcción de modelo ideal - Comparativa de datos de modelo y motor	- 3.2	- ajuste.xlsx - Motor2_Avellaneda.slx - P2_comparativa.mat - P2_pos.mlx
3	- Construcción de un modelo ideal completo - Comparativa de datos	- 4.1	- Motor3_Avellaneda.slx - P3_comparativa.mat
4	- Diseño de un sistema de control - Estudio del efecto del sistema - Estudio del efecto de la acción directa F_c - Estudio de u y z	- 4.2.1	- Motor4_Avellaneda.slx - P4_realimentacion.mlx
5	- Discretización del sistema de control - Estudio del efecto del sistema - Estudio de f - Estudio de u y z	- 4.2.2	- Motor5_Avellaneda.slx - P5_discretizacion.mlx
6	- Implementación - Estudio del efecto f y K_I - Estudio del efecto del <i>anti wind-up</i>	- Sección 4.3	- Motor6_Avellaneda.slx - P6_implementacion.mlx - P6_LQR.mlx

Cuadro 2: Descripción del contenido de las prácticas, distribución de las mismas en el documento y archivos relacionados con cada una.