3º INGENIERÍA INDUSTRIAL
AUTÓMATAS Y SISTEMAS DE CONTROL

PRÁCTICA DE SISTEMAS DE CONTROL, SESIÓN 20

CONTROL DISCRETO DE UN MOTOR DE CC EN VELOCIDAD

1. OBJETIVOS

Los objetivos de esta práctica son:

- Mostrar las posibilidades de control de sistemas reales mediante computador (control discreto o digital).
- Diseñar e implementar un controlador discreto para realizar el control de velocidad de un motor de corriente continua.

2. INTRODUCCIÓN

En esta práctica se realizará el control discreto en velocidad de un servomotor de corriente continua. Los pasos que se seguirán en la práctica son los pasos típicos en el diseño de un controlador discreto:

1. **Identificación del sistema.** En este primer paso se obtendrá un modelo matemático del sistema (función de transferencia). Si el diseño del controlador se realiza directamente en el plano discreto, deberá obtenerse una función de transferencia discreta (en la variable *z*). Si el diseño se realiza en continuo, es suficiente con obtener una función de transferencia continua para el sistema (en la variable *s*), para posteriormente discretizar el controlador obtenido. *En esta práctica realizaremos el diseño directamente usando funciones de transferencia discretas*, por lo que obtendremos un modelo discreto del servomotor, $G(z)$.

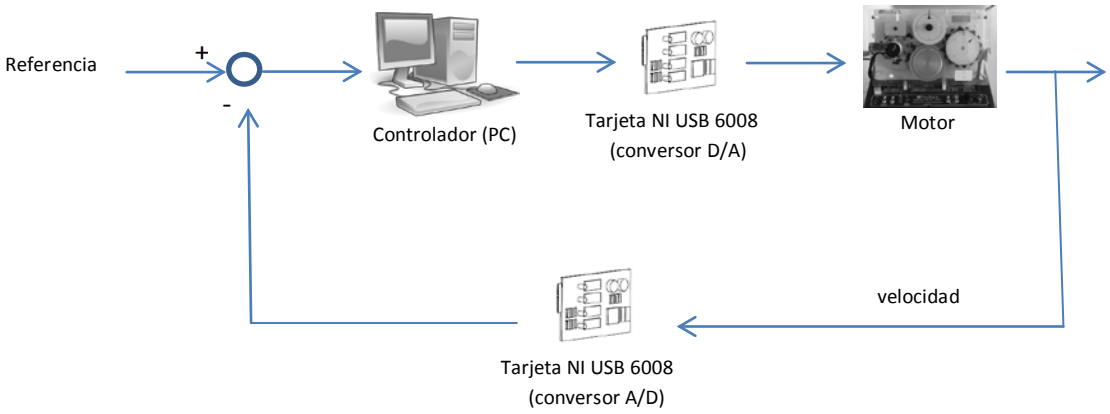
2. **Diseño del controlador.** En este paso se obtiene la función de transferencia discreta del controlador, $G_R(z)$, aplicando cualquiera de los métodos vistos en clase (PIDs por el lugar de las raíces, asignación de polos, tiempo mínimo, tiempo finito, etc.).

3. **Prueba del controlador en simulación.** Antes de probar el controlador diseñado sobre el sistema real, es conveniente realizar una simulación usando el modelo del sistema, para comprobar su funcionamiento y corregir posibles errores o ajustar el comportamiento.

4. **Prueba del controlador con el sistema real.** Por último se probará el controlador sobre el servomotor real, ajustando el comportamiento final si es necesario.

Antes de realizar el diseño del controlador discreto deberá decidirse qué periodo de muestreo usar. Para elegir el periodo de muestreo más adecuado pueden analizarse las características frecuenciales del sistema o bien elegir un periodo de muestreo lo suficientemente pequeño (por ejemplo, el mínimo que nos permita nuestro sistema de adquisición de datos). En nuestro caso consideraremos que el periodo de muestreo adecuado para el análisis y control del sistema es **T = 0.01 s**, y es el que usaremos en esta práctica.

La tarjeta de adquisición de datos que se usará es la NI USB-6008, ya usada en la práctica de identificación. En el esquema de control en bucle cerrado, esta tarjeta actuará como convertidor D/A para enviar la señal de control al sistema, y como conversor D/A para leer la señal de velocidad. En la figura siguiente se muestra el esquema de control discreto que se implementará en esta práctica.



3. REPASO: IDENTIFICACIÓN DEL SISTEMA

La identificación de los servomotores ya se hizo en una práctica previa. En esta práctica conviene volver a realizar la identificación del servomotor concreto que se utilice, ya que cada servomotor se comporta de una forma ligeramente diferente. Recordemos que la identificación la realizamos estudiando la respuesta del sistema en bucle abierto frente a una entrada escalón. Para realizar las pruebas de esta práctica puede hacerse la identificación para un escalón de, por ejemplo, 4 V. El sistema puede aproximarse a uno de primer orden cuya función de transferencia es

$$G(s) = \frac{k}{1 + \tau s}$$

donde:

- k es la ganancia, que puede obtenerse como el cociente

$$k = \frac{\text{valor final}}{\text{magnitud del escalón aplicado}}$$

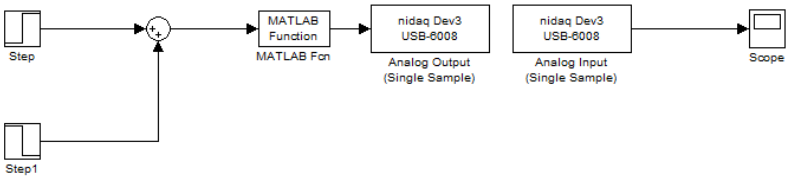
- τ es la constante de tiempo y puede obtenerse como el tiempo transcurrido desde el inicio del escalón hasta que la salida alcanza el 63% del valor final

La identificación la realizaremos con la tarjeta **NI USB-6008**, tal y como se hizo en la práctica de identificación. Recordemos que, para la identificación del servo en *velocidad*, las conexiones son las siguientes:

Servomotor (tarjeta ADAM 3937)		**Tarjeta NI USB-6008 (conectada al PC)**	
Amplificador: entrada positiva	**Pin 33**	**Pin 14**	Salida analógica, canal 0
Fuente de alimentación: 0 V	**Pin 11**	**Pin 16**	Masa salidas analógicas
Tacogenerador: señal positiva	**Pin 35**	**Pin 2**	Entrada analógica, canal 0 (AI0+)
Fuente de alimentación: 0 V	**Pin 11**	**Pin 3**	Entrada analógica, canal 0 (AI0-)

Nota: las conexiones deben realizarse con la fuente de alimentación desconectada, repasando las mismas antes de conectarla. Además, siempre que no se esté ejecutando un programa sobre el motor, la fuente de alimentación debe permanecer desconectada.

La identificación la realizaremos con el servomotor *sin freno* (palanca hacia arriba). Una vez realizadas las conexiones dibujaremos el esquema Simulink que se muestra a continuación para leer los datos necesarios:

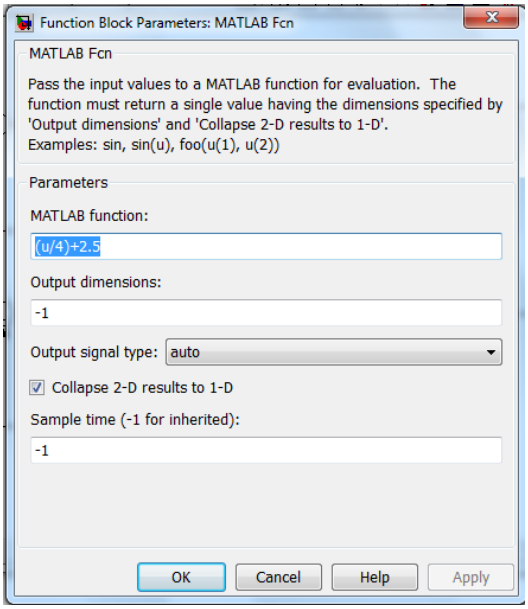


Recordemos los parámetros de configuración de los bloques de este esquema:

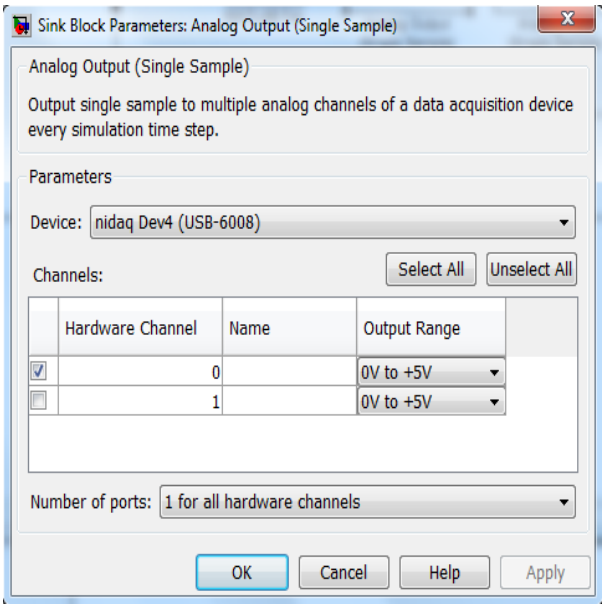
- El primer escalón (`Step`) se corresponde con la referencia de velocidad (expresada en voltios). Introduciremos un escalón de, por ejemplo, 4 V (parámetro `Final value` = 4), dejando un tiempo de 2 segundos para permitir que se inicialice el servo (parámetro `Step time` = 2). También puede indicarse el periodo de muestreo de 0.01 s en el campo `Sample time`.

- El segundo escalón sirve para detener el servo al final de la simulación. Para ello, suponiendo una duración de la simulación de 10 s, podemos detenerlo en el instante t=8 s (parámetro `Step time` = 8). En el parámetro `Final value` indicaremos un escalón de signo opuesto al anterior (-4 en nuestro caso). También podemos indicar el periodo de muestreo 0.01 s en el parámetro `Sample time`.

- El bloque `Matlab Fcn` que aparece a continuación de los escalones sirve para realizar la transformación entre los valores de salida calculados por el controlador en Simulink (que están en el rango [-10,10] voltios) y las tensiones que se admite el bloque '`Analog Output (Single Sample)`' de salida hacia la tarjeta (que están en el rango [0,5] voltios). En este bloque deberá introducirse la expresión `(u/4)+2.5` en el parámetro '`MATLAB function`' (ver la figura siguiente). Recuérdese que este bloque se encuentra en la librería "User-Defined Functions" de simulink.

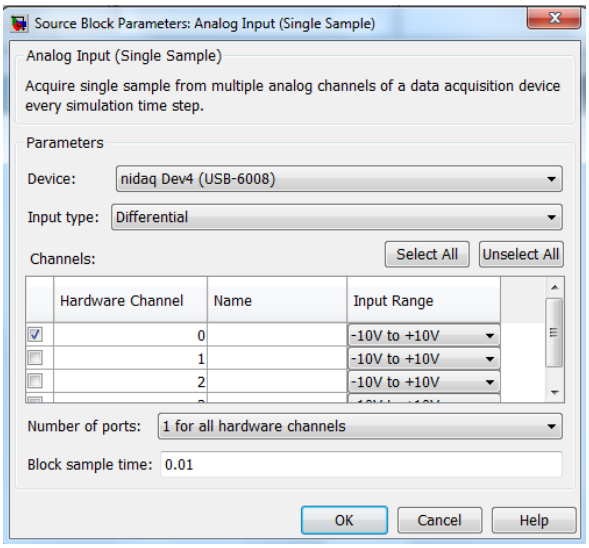


- Bloque de salida (`Analog Output (Single Sample)`). Este bloque se encuentra en la librería 'Data Acquisition Toolbox', y los parámetros a introducir aparecen en la ventana siguiente:



Recuérdese que, aunque en este bloque el rango de la salida es [0,5] voltios, las tarjetas están montadas sobre una placa con electrónica adicional para transformar la salida monopolar disponible en la tarjeta NI USB-6008 a la salida bipolar en el rango [-10,10] voltios.

- Bloque de entrada (`Analog Input (Single Sample)`). También se encuentra en la librería 'Data Acquisition Toolbox', y los parámetros a introducir aparecen en la ventana siguiente:



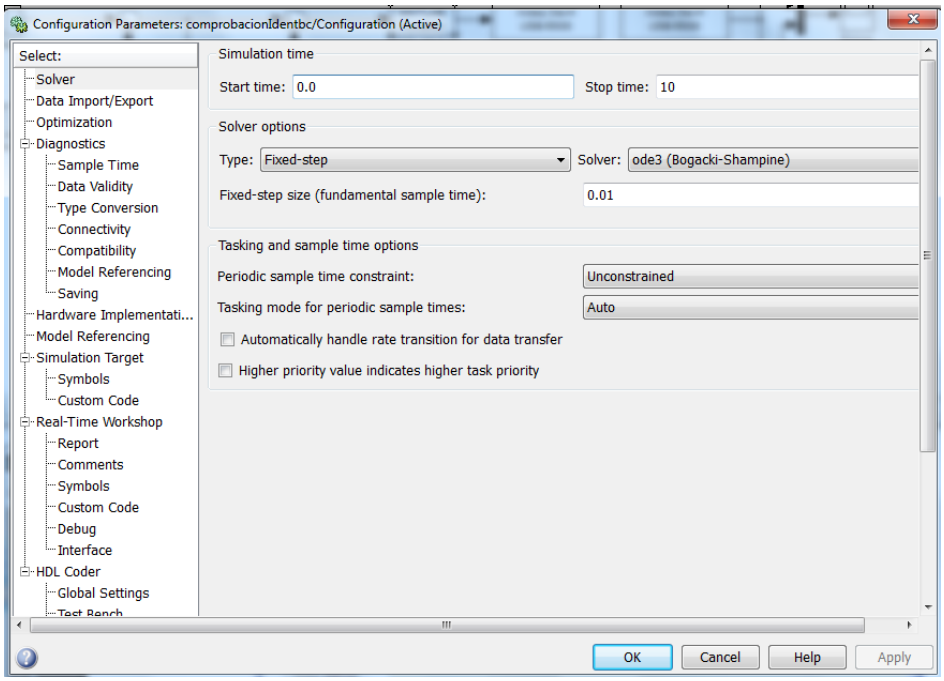
En las dos ventanas anteriores el identificador de la tarjeta puede ser distinto al mostrado aquí, dependiendo de la tarjeta concreta que se esté usando. Este identificador lo detecta automáticamente Matlab (parámetro '`Device`' en las ventanas anteriores).

- El bloque `Scope` se usará para representar la salida del sistema y realizar las medidas del valor final y la constante de tiempo. También puede usarse un bloque 'To Workspace' para almacenar la salida en una variable de Matlab y realizar los cálculos con más exactitud, como se hizo en la práctica de identificación.

- Para asegurarnos un correcto funcionamiento del esquema es conveniente modificar los siguientes parámetros de simulación (puede accederse a la ventana de configuración desde la opción de menú '**Simulation->Configuration Parameters**'):

    - ***Solver Options -> Type*** se configurará como 'Fixed *Step'*.
    - ***Fixed Step size:*** se fijará a **0.01** s (que coincide con el periodo de muestreo.

En la figura siguiente se muestra la ventana de configuración.



Una vez que hemos obtenido la función de transferencia continua, deberá obtenerse el equivalente discreto para el periodo de muestro T=0.01 s. Para ello, si suponemos que los valores de la ganancia y la constante de tiempo están almacenadas en las variables de Matlab k y `tau`, puede usarse la función `c2dm` con los comandos siguientes de Matlab:

```
num = [k];
den = [tau 1];
[numd, dend] = c2dm(num,den,0.01,'zoh')
```

De esta forma, en las variables `numd` y `dend` estarán contenidos el numerador y denominador de la función de transferencia discretizada usando un bloqueador de orden cero (`zoh` en Matlab). Esta función será la que usaremos a continuación para diseñar el controlador discreto.

4. DISEÑO DEL CONTROLADOR EN VELOCIDAD

Para realizar el control del sistema debemos calcular la función de transferencia del controlador a utilizar. Como sabemos, esta función de transferencia vendrá determinada por las especificaciones tanto dinámicas como estáticas que deseemos que presente el sistema.

EJERCICIO 1

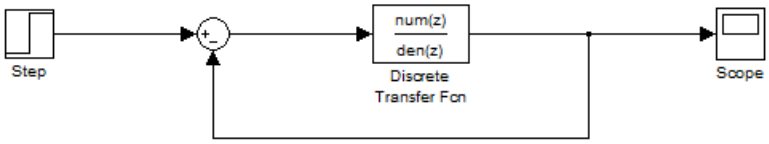
Estudiar el comportamiento del sistema en bucle cerrado sin controlador (equivalente a un controlador con ganancia K_p=1), tanto en simulación (usando el modelo discretizado obtenido en la identificación) como con el sistema real.

(1) Comprobar si el comportamiento en simulación y con el sistema real es similar.
(2) Comprobar que el sistema en bucle cerrado *no es capaz* de seguir la referencia de entrada ya que es de tipo cero (es decir, si la referencia es un escalón de 4 V, el sistema no será capaz de estabilizarse en este valor en régimen permanente).
(3) Calcular experimentalmente la sobreoscilación, tiempo de establecimiento y tiempo de subida del sistema en bucle cerrado. Puede usarse para ello la salida obtenida en Simulink, el comando `step`, o la herramienta `rltool`. Recuérdese que la llamada a `rltool` con el sistema discretizado se puede hacer de la forma siguiente:

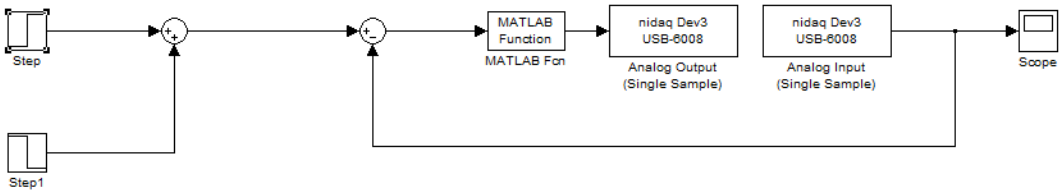
```
sistemad = tf(numd,dend,0.01);
rltool(sistemad)
```

Los esquemas en Simulink que pueden usarse en este ejercicio son los siguientes:

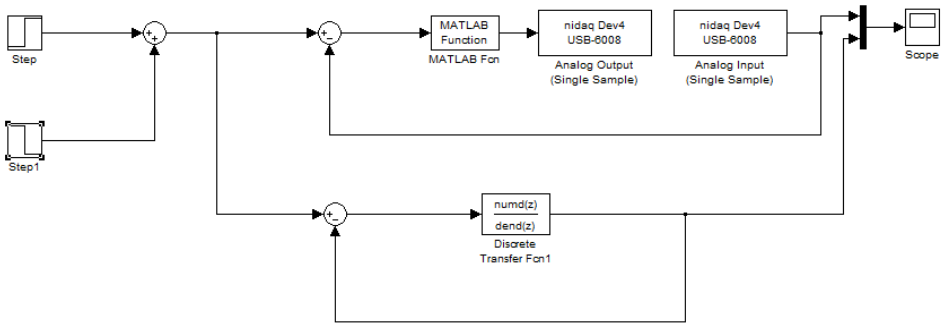
(a) Esquema en simulación (`numd` y `dend` son el numerador y denominador del modelo discreto del sistema, obtenidos previamente).



(b) Esquema con el sistema real



(c) Esquema para comparar el modelo con el sistema real.



Con las pruebas realizadas en el ejercicio 1 se puede concluir que es necesario diseñar un regulador para conseguir que el error en régimen permanente sea cero o, al menos, esté por debajo de un valor especificado.

EJERCICIO 2

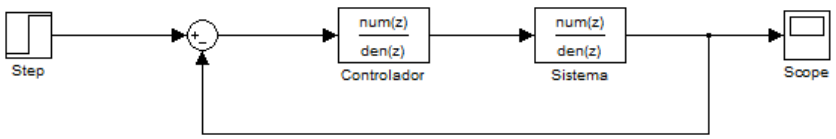
Diseñar el regulador más sencillo posible de tipo PID de tal forma que

- Conserve de forma aproximada el comportamiento dinámico del sistema en bucle cerrado que se ha observado en el ejercicio 1 (sobreoscilación, tiempo de establecimiento, tiempo de subida).
- Elimine el error en régimen permanente ante entrada escalón.

Se recomienda utilizar la herramienta `rltool` de Matlab como ayuda para realizar el diseño.

5. PRUEBA DEL CONTROLADOR EN SIMULACIÓN

Una vez que se ha diseñado el controlador discreto es conveniente comprobar su funcionamiento en simulación. Esto podemos hacerlo directamente desde `rltool` o bien puede usarse el siguiente esquema en Simulink:



donde en el bloque 'Controlador' deberemos introducir el numerador y denominador de la función de transferencia del controlador diseñado.

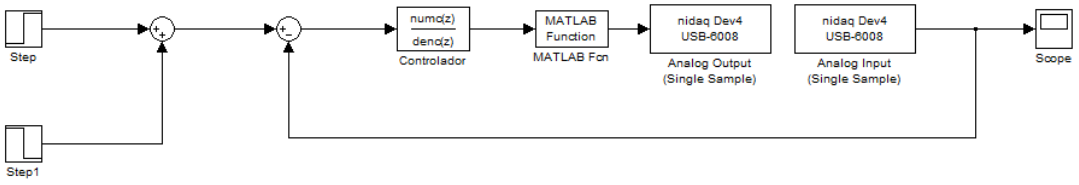
EJERCICIO 3

Comprobar el funcionamiento en simulación del controlador PID diseñado en el ejercicio anterior, usando `rltool` o el esquema Simulink anterior.

(1) ¿Existe error en régimen permanente?
(2) Comprobar si la sobreoscilación, tiempo de establecimiento y el tiempo de subida son parecidos a los del sistema en bucle cerrado sin el regulador, tal y como se pedía en las especificaciones.
(3) Obtener la evolución de la señal de control. ¿Se hace mayor que 10 V en algún instante?

6. PRUEBA DEL CONTROLADOR CON EL SISTEMA REAL

El último paso en el diseño del controlador consiste en probar su funcionamiento sobre el sistema real. Para ello debemos utilizar la tarjeta de adquisición de datos que se encargará de leer la señal del sensor de posición y enviar al motor la señal de control calculada por el PID diseñado. El esquema en Simulink que puede usarse para probar el controlador es el siguiente:



donde en el bloque 'Controlador' deberemos introducir la función de transferencia del controlador.

EJERCICIO 4

Comprobar sobre el sistema real el funcionamiento del controlador diseñado.

(1) ¿Existe error en régimen permanente?
(2) Comprobar si la sobreoscilación, tiempo de establecimiento y el tiempo de subida son parecidos a los del sistema en bucle cerrado sin el regulador, tal y como se pedía en las especificaciones.
(3) Realizar un experimento en el que se aplique el freno sobre el motor cuando ha alcanzado el régimen permanente. ¿Se comporta el sistema de forma satisfactoria? ¿Qué ocurre si se quita el freno después de haberlo aplicado?
(4) Comprobar el funcionamiento del controlador para valores del escalón de referencia distintos de 4 V.

Nota: dependiendo del motor que se esté usando, es posible que para algunos valores del escalón no se alcance la referencia deseada debido a que el motor es incapaz de alcanzarla. En este caso el controlador estaría enviando el máximo voltaje que admite el motor. Esto puede ocurrir, especialmente, si está el freno activado.

EJERCICIO 5 (optativo)

Diseñar un controlador discreto por asignación de polos que asigne los polos en bucle cerrado en z=0.8 y elimine el error en régimen permanente ante una entrada escalón. Comprobar el funcionamiento en simulación y sobre el sistema real. Analizar las respuestas obtenidas y la acción de control.