Práctica 3: Control de un motor en de corriente continua

Marcial-Antonio Barajas Martín

Práctica 1.

1.- Identificar todos los componentes de la planta. Placa de desarrollo, Placa de expansión, Motor, Fuente de alimentación, etc.

El motor es un EMG30, del que se compone del motor propiamente, una reductora y un encoder, el encoder se compone de sensores de efecto hall y un disco con imanes permanentes. La placa de desarrollo es una NUCLEO-411RE que hace la comunicación con Simulink, la placa de expansión es compatible con Arduino y la fuente de alimentación es una genérica de 12V que se comunica mediante dos cables a nuestro motor.

2.- Comprobar que todo el Hardware está configurado de acuerdo con la descripción de estas notas.

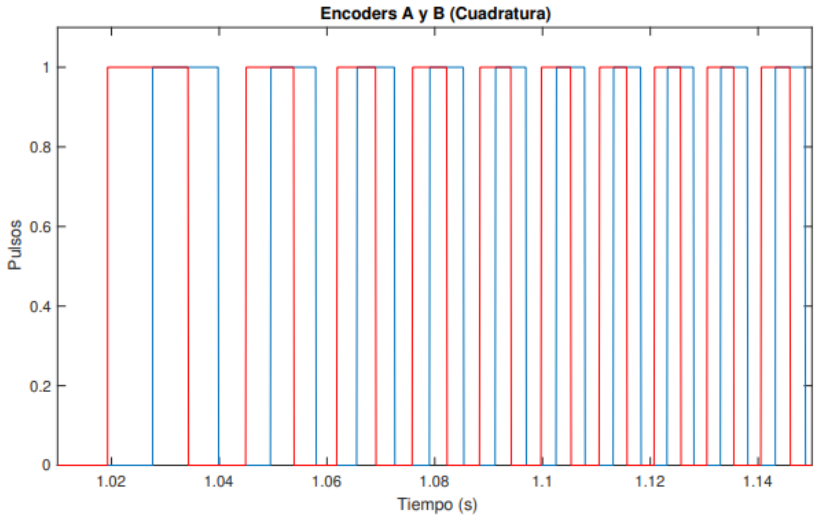
El montaje fue hecho en el propio laboratorio por los técnicos, visto a la hora del funcionamiento

3.- Montar el modelo de Simulink para el manejo del motor en lazo abierto. Configurarlo y comprobar que funciona correctamente, haciendo que el motor se mueva.

El modelo está en el fichero *motor_real_Practica1.slx*

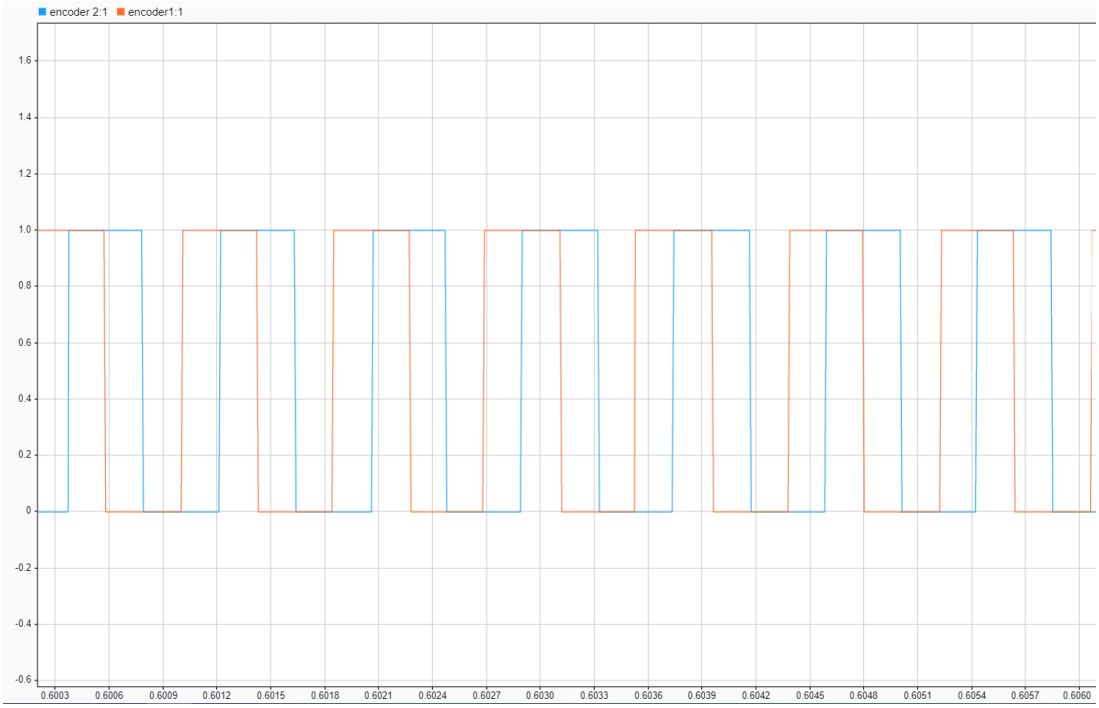
4.- Recoger datos de velocidad y posición para diversos valores del voltaje de entrada. Observar también las señales de los encoders y discutir su validez.

Podemos usar la figura 3.14 del guion:



Encoders A y B en el modelo real

De acuerdo a las gráficas y los colores de ambas, vemos que en el modelo real los pulsos se juntan más a medida que va avanzando el tiempo



Lectura de los encoders de la simulación

En este caso, el encoder está hecho en Simulink (*modelo_ideal_practica3.slx*) vemos que para el ideal los pulsos tienen la misma separación

Práctica 2.

1.- Obtener a partir de cálculo simbólico la expresión correspondiente a la posición angular de un motor de continua (ec. 3.14). Obtenerla a partir del modelo en variables de estado y a partir de la función transferencia. Calcular la expresión correspondiente a la velocidad angular.

$$\dot{\theta}(t) = \omega(t)$$

$$\dot{\omega}(t) = -\frac{1}{J}\left(\mu + \frac{a^2}{R + R_0}\right)\omega(t) + \frac{1}{J}\frac{a\,K_a}{(R + R_0)}e(t) - \frac{1}{J}T_L(t)$$

Partiendo de estas ecuaciones, sabemos que podemos dejar todo descrito en función de dos variables, que llamamos k_e y p:

$$k_e = \frac{1}{J}\frac{a\,K_a}{R + R_0}\,, p = \left(\mu + \frac{a^2}{R + R_0}\right)\frac{1}{J}\,, k_m = \frac{1}{J}$$

En el propio guion propone que consideremos un par de carga nulo, por lo que $T_L(t) = 0$, desarrollando las ecuaciones de estado obtenemos:

$$\dot{\theta}(t) = \omega(t)$$

$$\dot{\omega}(t) = -p\ \omega(t) + p\ e(t)$$

Considerando que la función $e(t)$ como un escalón de valor V, resolviendo el sistema de ecuaciones diferenciales obtenemos la ecuación:

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

De una manera análoga se podría hacer mediante las funciones de transferencia

$$\theta(s) = \frac{1}{s}\frac{\frac{1}{J}\frac{aK_a}{R+R_0}}{s+\left(\mu+\frac{a^2}{R+R_0}\right)\frac{1}{J}}e(s) - \frac{1}{s}\frac{\frac{1}{J}}{s+\left(\mu+\frac{a^2}{R+R_0}\right)\frac{1}{J}}T_L(s)$$

Si procedemos de manera análoga, con el par de carga nulo $T_L(s) = 0$, obtenemos:

$$\theta(s) = \frac{1}{s}\frac{k_e}{(s+p)}\ e(s)$$

2.- Construir un modelo del motor de cc en Simulink. Se puede hacer a partir de las ecuaciones de estado o de funciones de transferencia, pero de modo que se pueda obtener tanto la posición como la velocidad angular. El modelo deberá obtener los parámetros ke y p del workspace de Matlab. Emplear el modelo de Simulink para el control del motor, construido en la práctica anterior, en lazo abierto para identificar el motor real. Es decir, para obtener sus parámetros ke y p. Para ello:

- **Obtener la respuesta del motor para distintos voltajes (por ejemplo incrementando el valor de la entrada de dos voltios en dos voltios desde 0 a 12V) y un tiempo de simulación en todos los casos de 3s**

  El archivo está incluido en la carpeta con el nombre *medidas_motor.mat y representado en la tabla del siguiente apartado.*

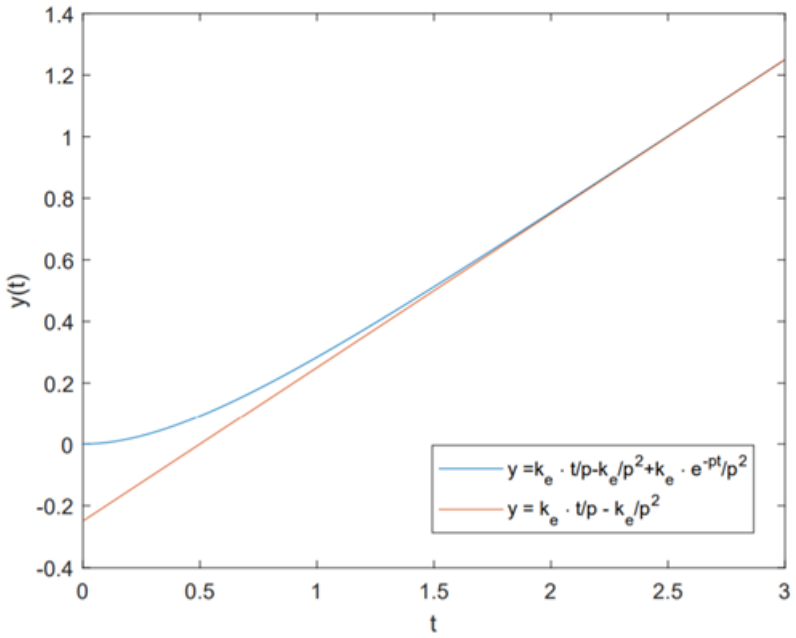
- **Calcular por regresión lineal la pendiente y el término independiente de la respuesta en estado estacionario**

  Realizamos las medidas escalando en sumas de 1.2V hasta los 12V

Voltaje (V)	Pendiente	Ordenada	Ke	p
1.2	1688,9	17.8	-159180	-94.30
2.4	1538,9	-25.3	93977	61.06
3.6	1377,7	56.7	-33475	-24.29
4.8	1218,9	15,4	-96475	-79.14
6	1055,6	-8,4	132500	125.59
7.2	897,94	12.33	-658397	-72.8189
8.4	749,71	28,64	-19622	-26.1732
9.6	591,14	9,69	-66063	-61.0052
10.8	429,36	11,58	-15920	-37.077
12	269,20	11,41	-6350	-23.59

- **Estimar los valores de Ke y p para cada valor del voltaje de entrada ¿Qué se puede concluir?**

  Para este cálculo, hemos eliminado los primeros 100 valores de cada medida, de tal manera que así podemos eliminar el término exponencial, vemos que los valores de la pendiente van en decrecimiento a medida que subimos el voltaje en el medidor de Simulink.



Gráfica posición-tiempo

Al quitar el término exponencial nos queda la ecuación:

$$\theta = V\frac{k_e}{p}\,t - V\frac{k_e}{p^2}$$

Los valores de la pendiente están incluidos en la tabla anterior, lo hemos sacado usando el comando polyfit de Matlab para extraerlo. Para los valores incrementados de voltaje, el valor de la pendiente es decreciente mientras que la ordenada en el origen se mantiene de manera alterna.

Para este cálculo, hemos eliminado los primeros 100 valores de cada medida, de tal manera que así podemos eliminar el término exponencial , vemos que los valores de la pendiente van en decrecimiento a medida que subimos el voltaje en el medidor de Simulink, ambién, de los valores de ke y p, podemos decir que son valores bastante dispares ya que estamos intentando linealizar un sistema no lineal, por comodidad a la hora de realizar las mediciones para los estimadores y el controlador del sistema simulado, tomaremos las medidas de ke y p de la medida intermedia, ya que la consideraremos la más fiable (V=6V, ke=132500, p=125,59)

Práctica 3

1.- Construir un modelo completo del motor, a partir del modelo de motor ideal construido en la práctica

El modelo está en el archivo *modelo_ideal_practica3.slx*

2. Añadir el mismo bloque de lectura de encoders construido para leer los encoders del motor real.

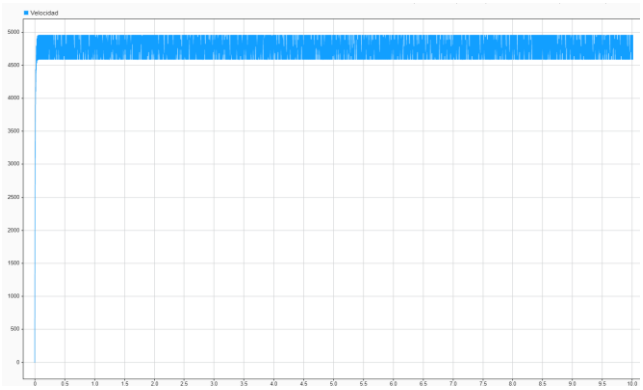
El modelo está descrito en el archivo *modelo_ideal_practica3.slx*, que aparece con los valores establecidos en el guion, las gráficas se detallan a continuación con los valores V=12V, k_e =132500, p=129.5, las medidas también están detalladas en el archivo de workspace *modelo_ideal_Practica_3.mat*

2.- Simular el comportamiento del modelo para distintos valores del voltaje de entrada Vref.

Las gráficas las se han separado ya que en la representación del inspector no cuadraban en el mismo gráfico, para solventar esto, se ha usado el valor de la gráfica de posición (en el inspector) con la salida del encoder 1 (en un scope) y la velocidad en el inspector y la salida del encoder 2 en otro scope. Se puede ver fácilmente en el archivo de Simulink adjunto a este apartado

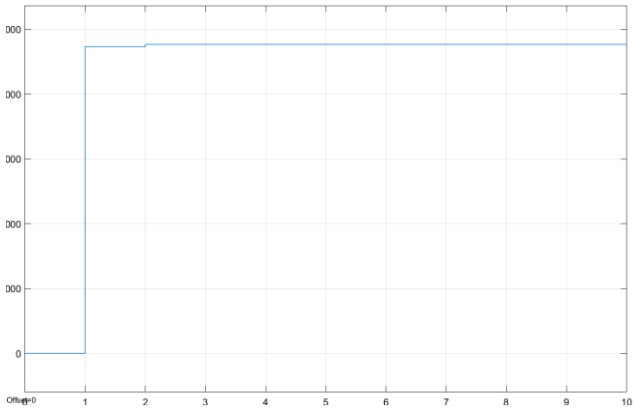
-Comparar la salida directa (posición y velocidad) del motor con la que se obtiene a partir de la lectura de los encoders

Para la representación de la velocidad en el visor obtenemos:



Gráfica velocidad-tiempo medida en el visor

Y la gráfica relativa al encoder de la velocidad (encoder 2):

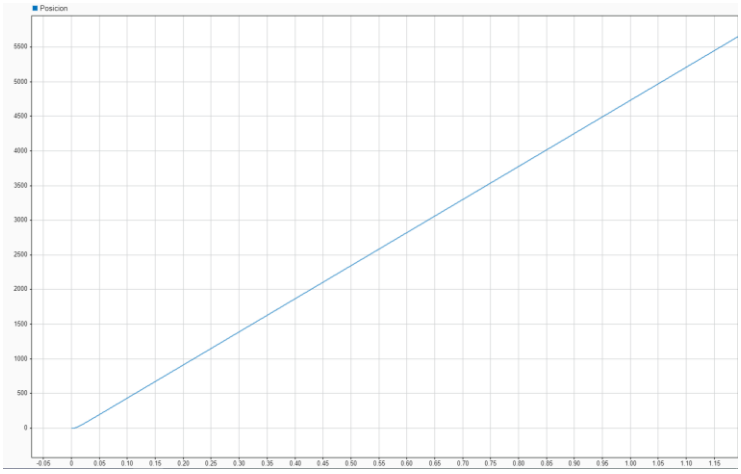


Gráfica velocidad-tiempo medida en el encoder

Vemos que las gráficas se asemejan a lo descrito en el guion pero el problema es que la escala del encoder es mucho más pequeña que la de velocidad.

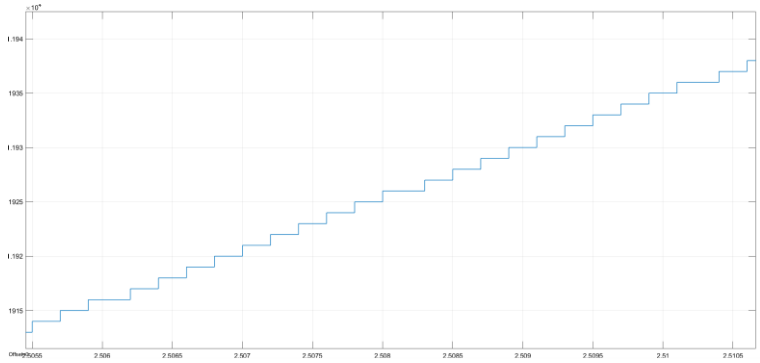
-Comparar los valores de las lecturas de velocidad y posición obtenidas a través de los encoders para el motor real y para el modelo.

Hacemos una representación hecha por la lectura de la posición frente al tiempo primero y luego leemos el encoder.



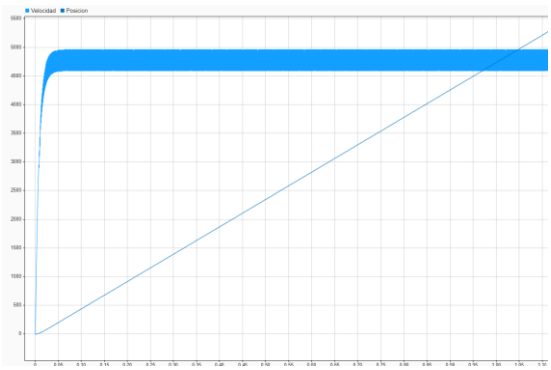
Gráfica posición tiempo

Posición medida en el encoder:



Gráfica posición-tiempo medida del encoder

Y por último, hacemos una medida de posición – velocidad



Gráfica Velocidad y Posición frente a tiempo

Y comprobamos que es idéntica a la figura que aparece en el guion, la velocidad alcanza su posición máxima en un tiempo muy reducido de tiempo.

Práctica 4.

1.- Construir un modelo en Simulink, que incluya el motor ideal y un sistema de control por realimentación de estados estimados y acción integral.

El archivo está en el fichero *modelo_practica4.slx* y las medidas usadas están en el fichero worskpace

Modelo_ideal_practica4.workspace.mat

2.- Comprobar el efecto de la posición de los polos del sistema y el estimador sobre la velocidad de respuesta del sistema y el valor de la señal de control u. (incrementarlos y disminuirlos en potencias de 10)

Para los polos, hemos usado los descritos en el guion: k=[0.5p 0.5p], usando sobre el sistema descrito por:

$$A = \begin{pmatrix} 0 & 1 \\ 0 & -p \end{pmatrix}$$

$$B = \begin{pmatrix} 0 \\ k_e \end{pmatrix}$$

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

Usando estas matrices, podemos describir los polos K con el comando

K = acker(A,B,[0.5p 0.5p])

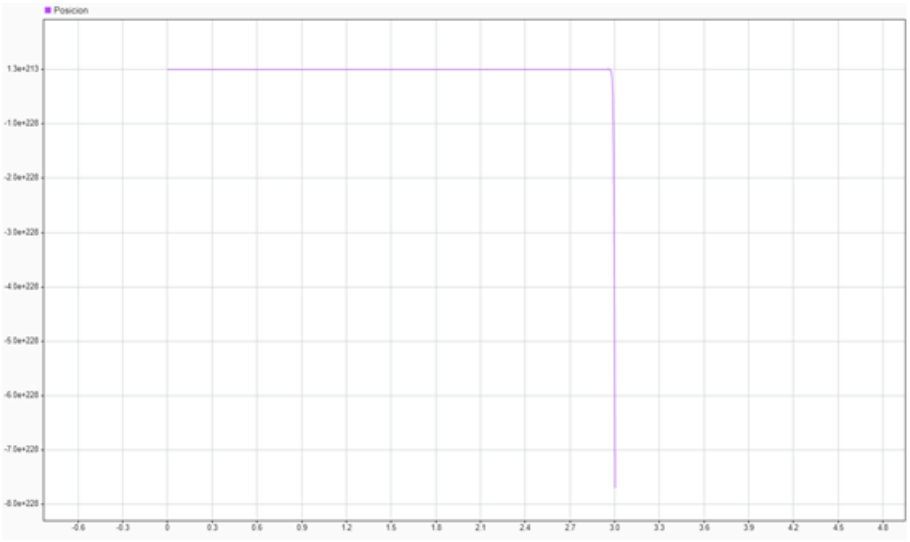
El resultado de la medida también influye con el estimador L que hemos construido y que está situado en =[1.2p 1.2p]^T, la medida de este estimador lo podemos hacer con el comando:

L=acker(A',C',[1.2p 1.2p])

Para hallar L y K usamos el comando *acker* en vez de *place*, ya que con este último comando no podríamos sacar los valores ya que el polo está repetido.

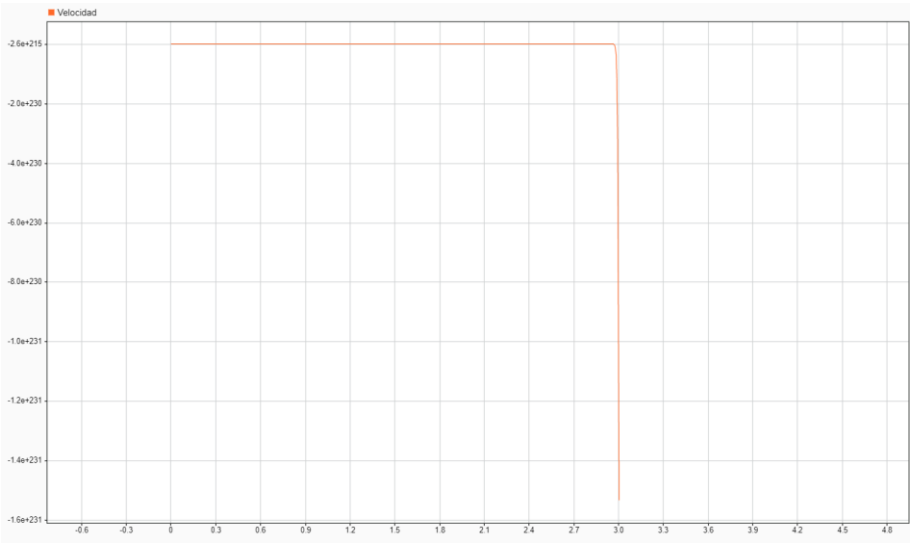
Las gráficas obtenidas para estos modelos son:

Para la posición



Gráfica Posición-Tiempo

Y para la velocidad:



Gráfica Velocidad-Tiempo

Observamos que hay algo que no está yendo bien, ya que la velocidad del motor baja de manera muy abrupta cuando se acerca a t=3 s.

Para las cuestiones 3 y 4 del guion, al no poder observar bien la medida del modelo, no se puede observar el efecto de la ganancia ni el error de posición inicial sobre u.

Práctica 5

1.- Diseñar un sistema de control basado en realimentación de estados para el motor, empleando los parámetros Ke y p identificados en la práctica 2.

El Modelo se describe en el archivo *modelo_real_practica5.slx* del directorio y su workspace en *modelo_real_practica5_workspace.mat*

2.- Estudiar el efecto de la posición de los polos y los valores de f en la respuesta obtenida para entradas escalón. Examinar el efecto sobre la entrada del sistema u.

Con el sistema descrito, podemos usar los valores de las matrices que están descritas en el guion, el valor de T=0.005 s. y tenemos que recalcular los valores de K y L para emplearlos con el estimador discreto

Entonces, obtenemos:

$$F = e^{AT} \quad G = e^{AT}A^{-1}[I - e^{-AT}]\,B$$

$$C = C, \quad D = D$$

Con estos valores y los obtenidos usando la discretización de los polos, que resulta ser:

$$\Lambda_i = e^{\lambda_i T}$$

Usamos los comandos:

K = acker(F,G,[Λ_{p1} Λ_{p2}])

L = acker(F',C',[Λ_{p1} Λ_{p2}])'

Siendo Λ_{p1} Λ_{p2} el valor de los polos discretizados.

Para simular este modelo hemos usado una posición inicial de x_r = 5, el polo diseñado para la acción integral lo situamos en -20.

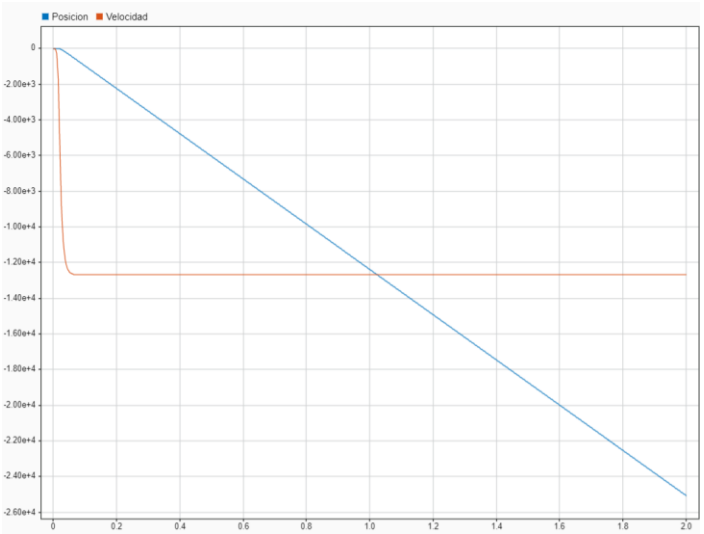
Práctica 6

1.- Construir los bloques de Simulink del estimador acción directa y acción integral discreta y añadirlos al modelo de motor completo desarrollado en la práctica 3.

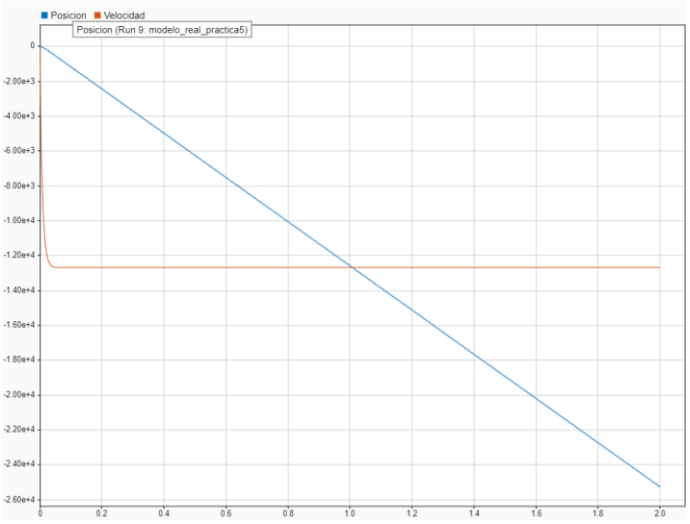
El modelo está diseñado en el archivo *motor_ideal_discretizado.slx*

2. Estudiar la respuesta del sistema para los sistemas de control diseñados en la práctica 5. Comprobar el efecto de eliminar la acción integral Kiz = 0 o la acción directa f=0

Usando el modelo *modelo_real_practica5.slx*, vamos a cambiar los dos valores para representarlo.



Sistema con f=0, ki=-20.



Sistema con Kiz=-20, f=0,

Vemos que hay algo que no está yendo bien ya que, para ambos valores, de manera cambiada, sigue dando el mismo resultado

3.- Añadir al modelo un bloque anti wind-up, analizar la salida para distintos valores de la constante kamp.

El modelo está realizado en el archivo *motor_realista_windup.slx*

El modelo está hecho siguiendo los pasos del guion pero no funciona, da un error de ejecución a los 0.70 segundos de simulación del sistema, por lo que no podemos determinar nada, se ha probado a cambiar dos veces los valores de *kamp*, pero sigue dando el error de ejecución en el mismo tiempo (0.75s de la simulación).