UNIVERSIDAD
CARLOS III DE MADRID

CONTROL INTELIGENTE

PRÁCTICA 3

Control de un motor de cc utilizando Redes Neuronales

La Toolbox Deep Learning de Matlab contiene herramientas para el diseño, implementación, visualización y simulación de redes neuronales. Las redes neuronales son usadas en aplicaciones donde un análisis formal es difícil o imposible, como reconocimiento de patrones o identificación y control de sistemas no lineales. La Toolbox contiene: redes feedforward, redes de base radial, redes dinámicas, mapas auto-organizados (mapas de Kohonen), etc.

NARMA-L2 (Feedback Linearization) Control

Este es un tipo de control neuronal que se suele llamar de dos formas distintas: control por linealización de la realimentación o control NARMA-L2. Se le llama linealización de la realimentación cuando el modelo de la planta tiene una forma particular (*companion form*: contiene los coeficientes de un polinomio característico correspondiente en una de sus filas o columnas laterales, siendo el resto una matriz diagonal). Se le llama NARMA-L2 cuando el modelo de la planta puede ser aproximado por la misma forma. La idea central de este tipo de control es transformar sistemas dinámicos no lineales en sistemas dinámicos lineales cancelando las no linealidades.

Se empieza presentando el modelo del sistema y demostrando como se puede usar una red neuronal para identificar el modelo. Luego se describe como el modelo de la red neuronal identificado puede ser usado para desarrollar un controlador.

El primer paso es identificar el sistema que va a ser controlado. Se entrena la red neuronal para representar la dinámica hacia delante del sistema. Hay que elegir que estructura usar para el modelo. Un modelo estándar que es usado para representar sistemas no lineales discretos es el modelo Nonlinear AutoRegressive-Moving Average (NARMA):

$$y(k+d) = N[y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-n+1)]$$

donde $u(k)$ es la entrada de sistema y la salida es $y(k)$. Para la fase de identificación, se puede entrenar la red neuronal para aproximar una función no lineal N. Este es el procedimiento de identificación seguido por la parte predictiva del controlador neuronal.

Si se quiere que la salida del sistema siga alguna trayectoria de referencia $y(k + d) = yr(k + d)$, el siguiente paso consiste en desarrollar un controlador no lineal de la forma:

$$u(k) = G[y(k), y(k-1), \ldots, y(k-n+1), y_r(k+d), u(k-1), \ldots, u(k-m+1)]$$

El problema usando este controlador es que si se quiere entrenar una red neuronal para crear una función G que minimice el error por mínimos cuadrados, es necesario usar propagación hacia atrás dinámica. Esto puede ser muy lento. Una solución consiste en usar modelos aproximados para representar el sistema. El controlador usado aquí está basado en el modelo aproximado NARMA-L2:

$$\hat{y}(k+d) = f[y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-m+1)] + g[y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-m+1)] \cdot u(k)$$

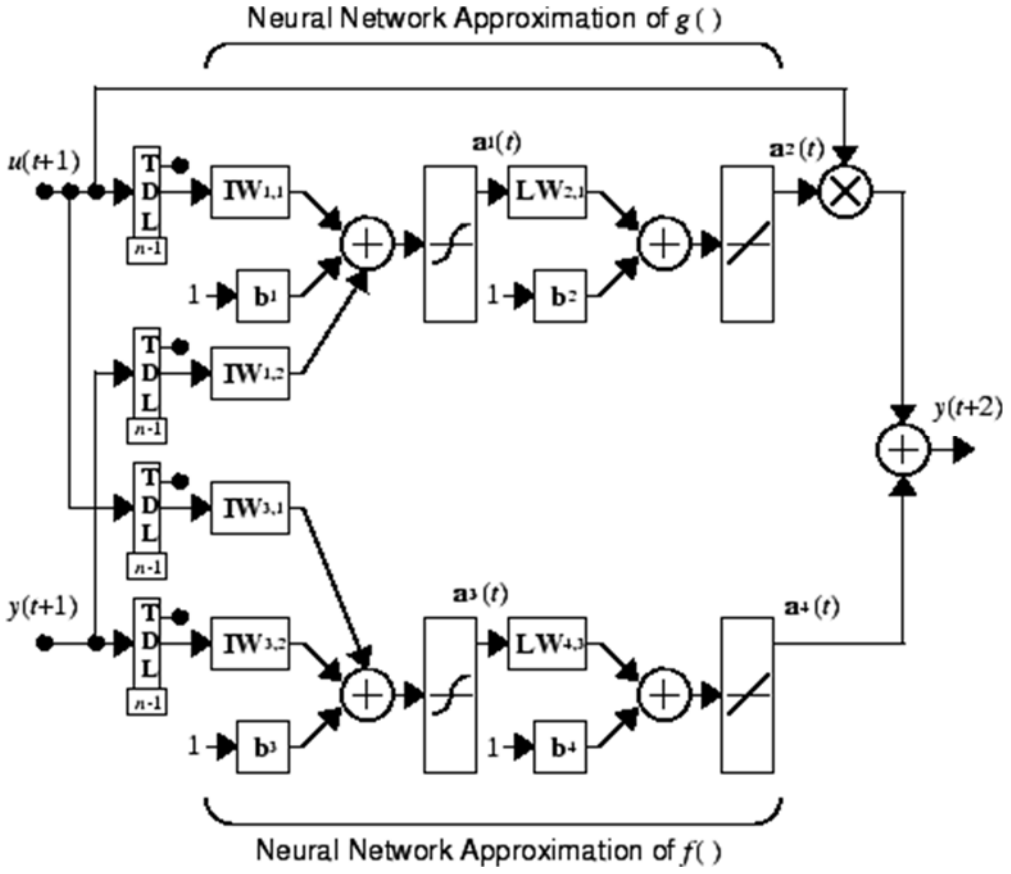
En este modelo la siguiente entrada del controlador $u(k)$ no está contenida dentro de la no linealidad. La ventaja es que se puede obtener la entrada de control que hace que el sistema siga la referencia $y(k + d) = yr(k + d)$. El controlador resultante tiene la forma siguiente:

$$u(k) = \frac{y_r(k+d) - f[y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-n+1)]}{g[y(k), y(k-1), \ldots, y(k-n+1), u(k-1), \ldots, u(k-n+1)]}$$

Usar esta ecuación directamente puede causar problemas de realización porque hay que determinar la entrada de control $u(k)$ basándonos en la salida en el mismo instante de tiempo, $y(k)$. Por lo tanto, el modelo que se usa es el siguiente:

$$y(k+d) = f[y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-n+1)] + g[y(k), \ldots, y(k-n+1), u(k), \ldots, u(k-n+1)] \cdot u(k+1)$$

donde $d \geq 2$. La siguiente figura ilustra la estructura de una red neuronal:

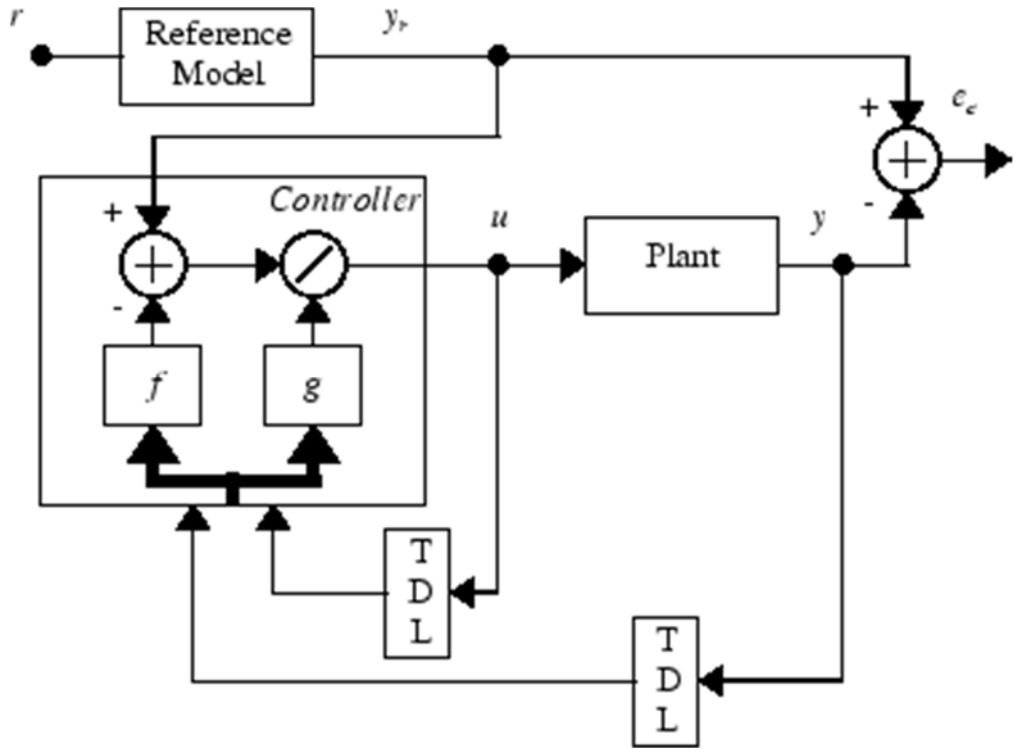


Controlador NARMA-L2:

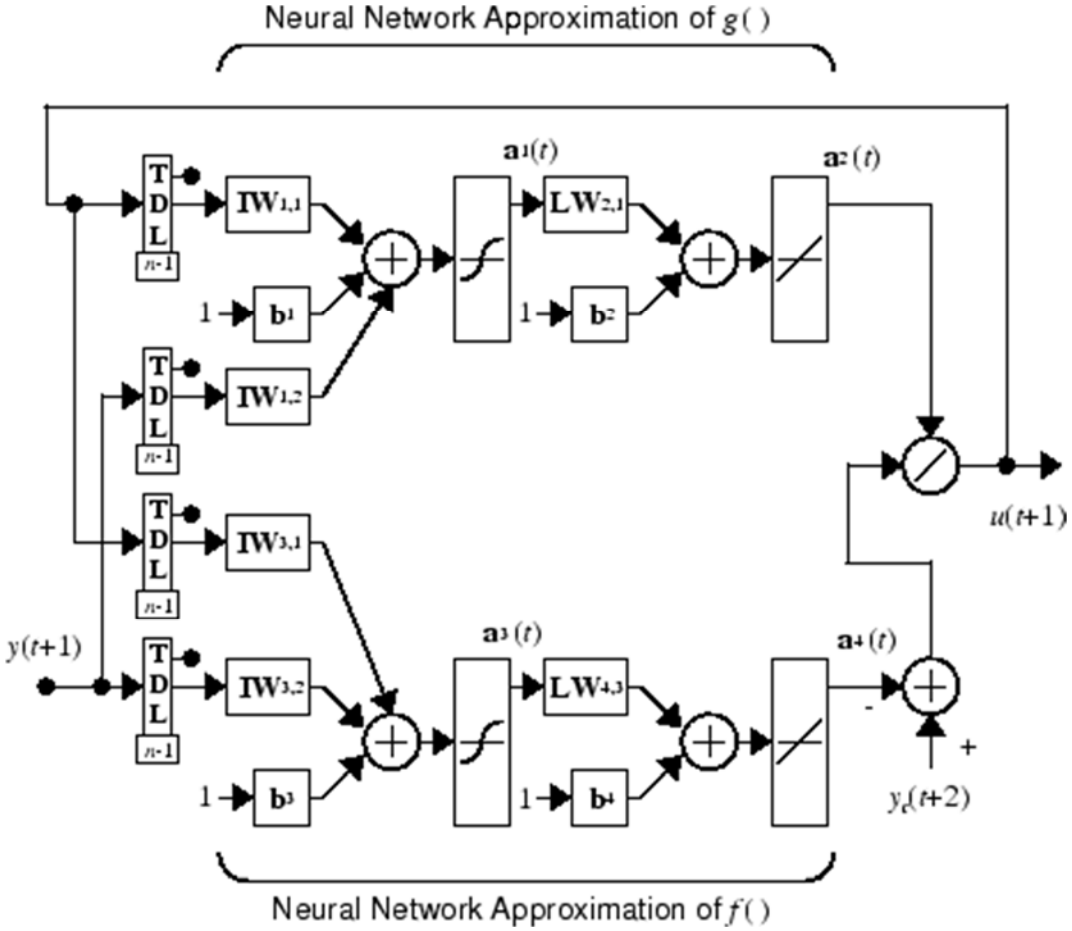
Usando el modelo NARMA-L2, se obtiene el controlador:

$$u(k+1)=\frac{y_r(k+d)-f[y(k),\ldots,y(k-n+1),u(k),\ldots,u(k-n+1)]}{g[y(k),\ldots,y(k-n+1),u(k),\ldots,u(k-n+1)]}$$

que es realizable para $d \geq 2$. La siguiente figura es un diagrama de bloques del controlador NARMA-L2:



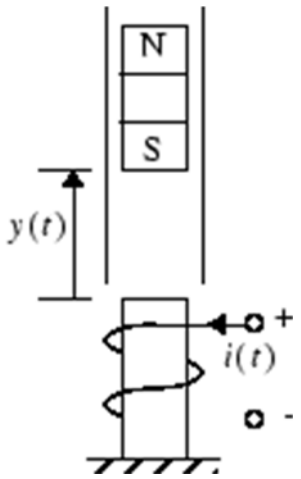
Este controlador puede ser implementado con el modelo de la planta previamente identificado NARMA-L2, tal y como se muestra en la siguiente figura:



EJEMPLO:

Con este ejemplo se muestra como el controlador NARMA-L2 es entrenado. El primer paso para poder usarlo. es copiar el bloque de Simulink NARMA-L2 Controller de la Toolbox Neural Network. Si no está seguro de cómo hacerlo consultar la ayuda de Simulink en Matlab. Este paso se salta en esta demostración.

El objetivo del ejemplo es controlar la posición de un imán suspendido sobre un electroimán, donde el imán está limitado para solo poder moverse en la dirección vertical, tal y como se muestra en la figura:



La ecuación de movimiento de este tipo de sistema es:

$$\frac{d^2y(t)}{dt^2} = -g + \frac{\alpha}{M}\frac{i^2(t)}{y(t)} - \frac{\beta}{M}\frac{dy(t)}{dt}$$

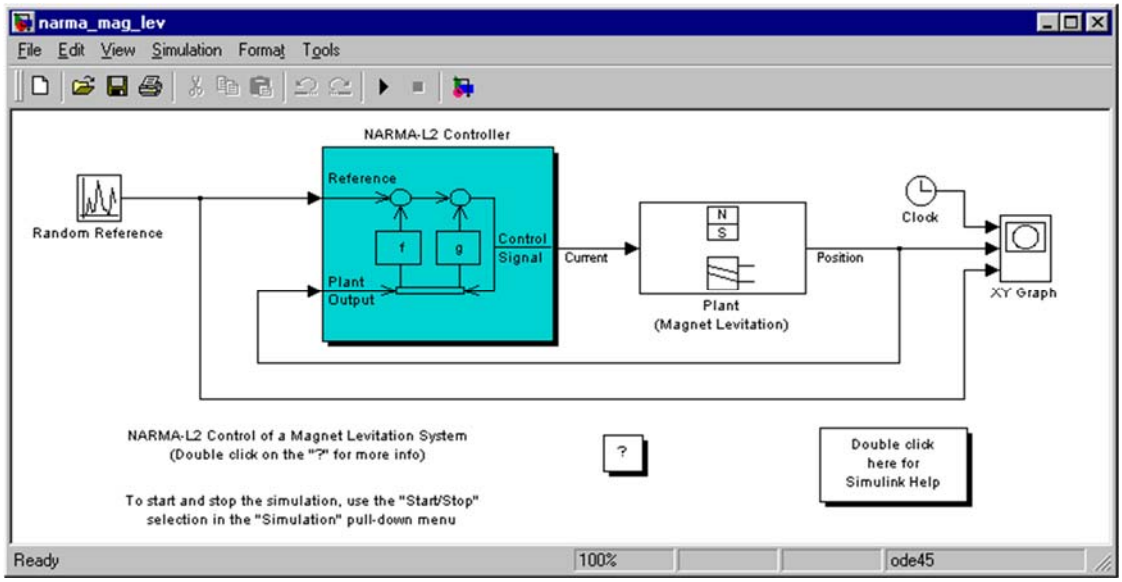
donde $y(t)$ es la distancia del imán respecto al electroimán, $i(t)$ es la corriente que circula por el electroimán, M es la masa del imán, y g es la constante gravitacional. El parámetro β es el coeficiente de fricción viscosa que es determinado por el material por el que se mueve el imán, y α es la constante de fuerza del campo que es determinado por el número de vueltas del cable sobre electroimán y la fuerza del imán.

Pasos:

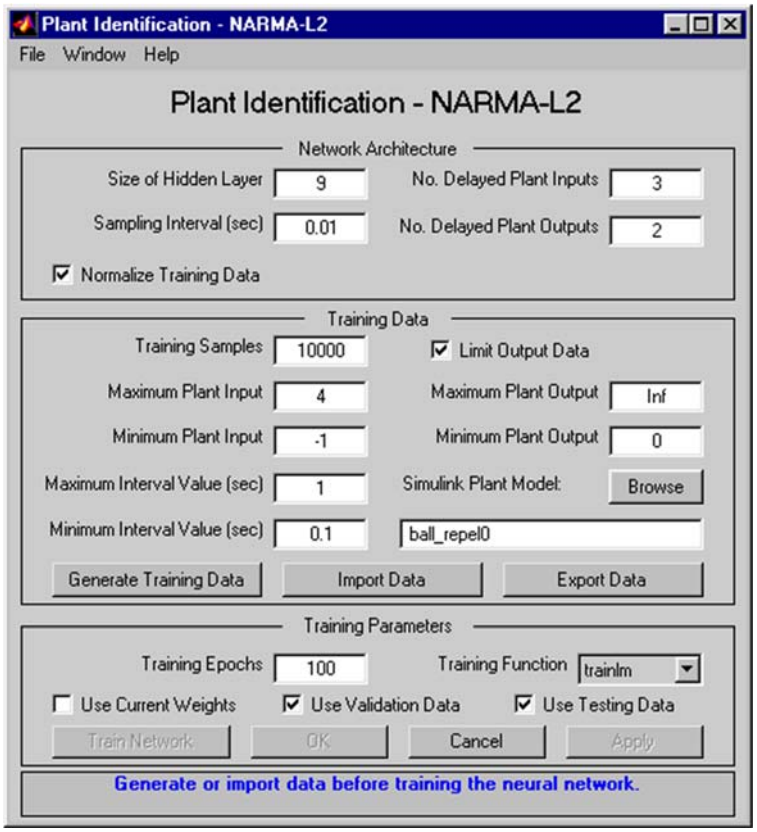
1. Iniciar MATLAB.

2. Escribir *narmamaglev* en el editor de comandos. Este comando inicia el simulink y crea el siguiente modelo, que ya contiene el bloque de control NARMA-L2.

>> *narmamaglev*

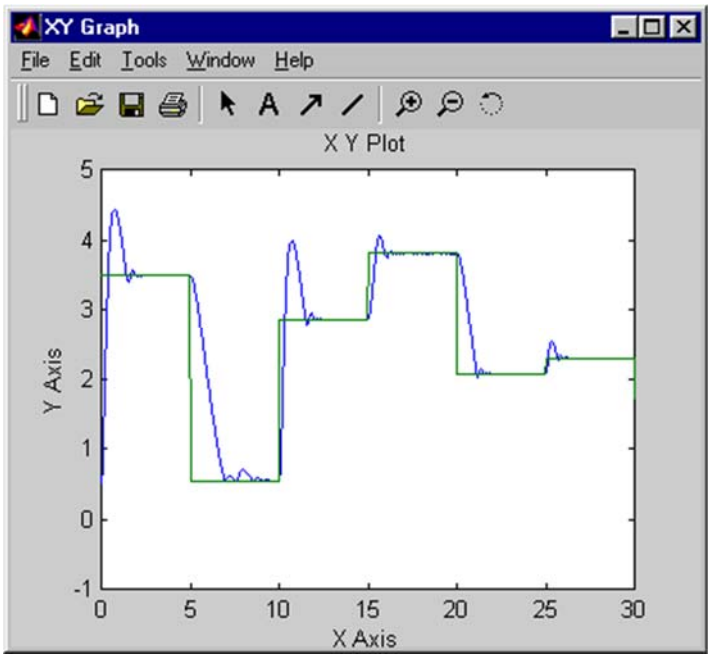


3. Hacer doble click en el bloque de control NARMA-L2. Se abre la siguiente ventana, que le permite entrenar el modelo NARMA-L2. No hay una ventana separada para el controlador porque el controlador es determinado directamente desde el modelo, no como el modelo de control predictivo.



4. Ahora hay que generar los datos de entrenamiento (Generate Training Data) y luego entrenarlo (Train Network). Esta ventana trabaja igual que otras ventanas de identificación de planta, luego el proceso de entrenamiento no es repetido. En lugar de eso, simular el controlador NARMA-L2.

5. Volver al modelo de Simulink e iniciar la simulación presionando Start dentro del modelo de Simulación. A medida que la simulación progresa, la salida de la planta y la señal de referencia son mostradas, como en la figura siguiente:

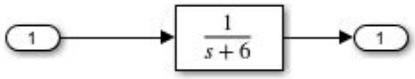


<u>Trabajo a realizar: Control de un motor de corriente continua a cuál función de transferencia en velocidad es:</u>

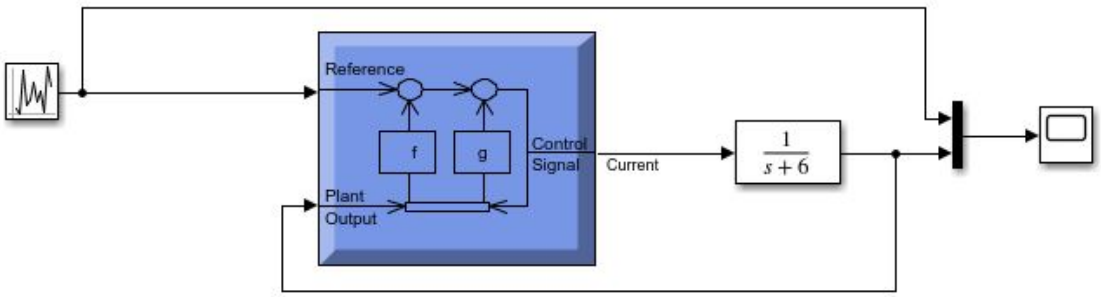
$$G(s) = \frac{1}{s+6}$$

Una vez hemos aprendido cómo funciona el controlador NARMA-L2, se va a proceder a controlar el motor de corriente continúa utilizando esta red neuronal. Para ello,

1. El primer paso consiste en crear un modelo Simulink que contiene la planta, en nuestro caso la función de transferencia G(s), guardándolo con el nombre de ***tf2.slx*** .

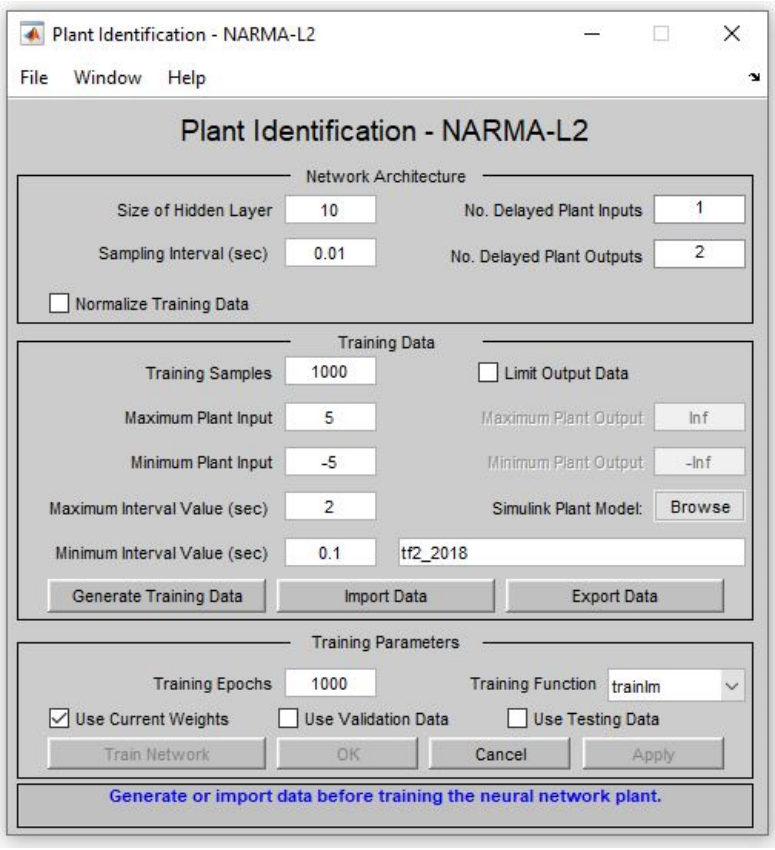


2. Se diseña el esquema de control basado en el bloque NRAMA-L2, y se guarda con el nombre por ejemplo ***tf2_narma.slx.***



Tanto el modelo ***tf2.slx*** como ***tf2_narma.slx*** deberían encontrarse en la misma carpeta en el directorio corriente de Matlab.

3. Configuramos el bloque NRAMA-L2 por ejemplo:



Antes de iniciar la simulación hay que identificar el sistema: generar los datos de entrenamiento (Generate Training Data) y luego entrenarlo (Train Network).

Una vez alcanzado un resultado aceptable, intentar ajustar el controlador de tal forma que se consiga un comportamiento óptimo.

Se pide al alumno que modifique los parámetros del controlador de tal forma que se controle de forma adecuada el motor.

Comparar los resultados con los obtenidos en las prácticas anteriores utilizando control borroso y el PID convencional, señalando ventajas en inconvenientes.

Si los resultados no son adecuados, no somos capaces de entrenar correctamente la red, empezar de nuevo modificando alguna de las siguientes variables.

- Size of hidden layer. Defecto: 10.
- Training Samples. Defecto: 1000.
- Training epochs. Defecto: 1000.

Repetir el mismo procedimiento para las siguientes funciones de transferencias, analizando los resultados:

a) $G(s) = \frac{1}{0.51s^2+s+1}$

b) $G(s) = \frac{1}{0.51s^2+s}$