

Tema 4. Control Neuronal

Luis Miguel Bergasa

Control Neuronal

Índice

- **Introducción**
- **Arquitecturas de Control Neuronal**
 - Control directo
 - Control inverso
- **Identificadores con RNs**
- **Control de sistemas SISO mediante RNs**
 - Controlador predictivo
 - Controlador NARMA-L2
 - Controlador con modelo de referencia
- **Control de sistemas MIMO mediante RNs**
 - Con modelo de referencia
- **Estabilidad**

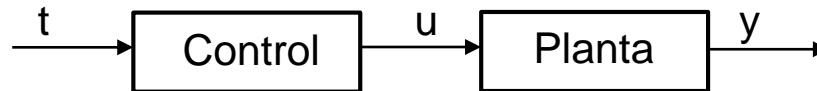
Introducción

Control Neuronal

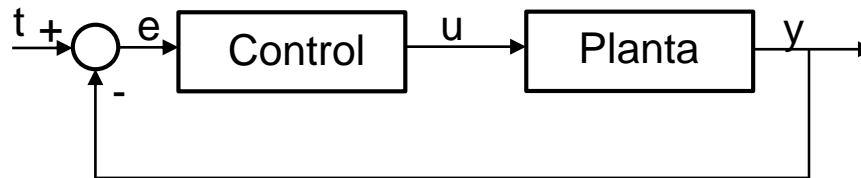
Introducción

- **Conceptos de Control Neuronal se basan en la teoría clásica de control**

- Control en lazo abierto



- Control en lazo cerrado
 - Menor conocimiento de la planta
 - Mejor comportamiento ante perturbaciones



Control Neuronal

Introducción

➤ Teoría básica de control

➤ Control clásico

- Todo o nada (“bang-bang”). Ej: termostado de T^a
- PID (Proporcional-Integral-Derivativo)
 - Se consigue más precisión
 - Problema: ajustar las constantes del controlador PID (K_I , K_D , K_P)

➤ Control moderno

- En el dominio del tiempo basado en VVEE
 - Control óptimo. Minimización de una función de coste
 - Alta carga computacional (off-line)
 - Se necesita un modelo de la planta y acceso a las variables a controlar
 - Control adaptativo. Combina la estimación y el control de parámetros en tiempo real
- Control Inteligente
 - **Control Neuronal**
 - Control borroso

Control Neuronal

Introducción

➤ **Control Neuronal**

- Sistemas de control que utilizan redes neuronales
- Se adaptan a sistemas dinámicos variantes en el tiempo implementando controles adaptativos
- Neurocontrolador = Controlador Neuronal

➤ **Propiedades**

➤ **Control de procesos complejos de los que se desconoce su modelos de comportamiento**

- Control e identificación de sistemas no lineales y variantes en el tiempo
 - En el control de sistemas lineales e invariantes (LTI) se utilizan técnicas de control clásico.
 - Control de procesos donde no se puede aplicar técnicas convencionales

➤ **Utiliza dos procesos de ajuste**

- Aprendizaje off-line (Entrenamiento): adquisición previa de conocimiento del sistema a controlar
- Aprendizaje on-line: adaptación dinámica a los cambios del sistema a controlar

Arquitecturas de Control Neuronal

Control Neuronal

Arquitecturas de Control Neuronal

- **Control directo**

- Control supervisado

- **Control inverso**

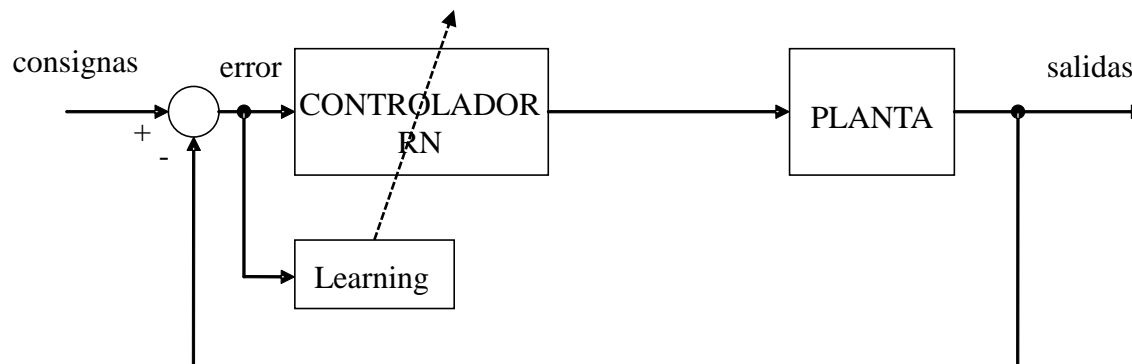
- No orientado a minimizar el error de control
 - Orientado a minimizar el error de control

Control Neuronal

Arquitecturas de Control Neuronal

➤ Control directo

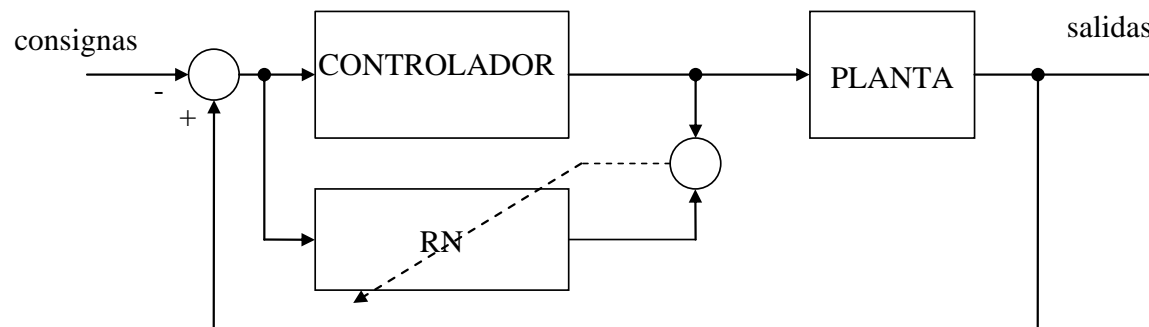
- La red neuronal implementa directamente el controlador
- Emplea datos numéricos de entrada/salida o un modelo matemático del sistema
- Objetivo: minimizar el error cuadrático de control $(\text{consignas} - \text{salidas})^2$



Control Neuronal

Arquitecturas de Control Neuronal

- **Caso práctico control directo -> control supervisado**
 - La red neuronal actúa como un aproximador universal copiando el comportamiento de un controlador previo
 - Entrenamiento *off-line* mediante un conjunto de entradas y salidas (*batch* de datos)
 - RN se coloca en paralelo con el controlador (clona el controlador)
 - Una vez entrenado se sustituye el controlador por la RN
 - Problema: El controlador final (RN) no es adaptativo



Control Neuronal

Arquitecturas de Control Neuronal

➤ Control inverso

1. Identifica la dinámica de la planta mediante una RN
2. Diseña un controlador neuronal a partir de la RN identificadora

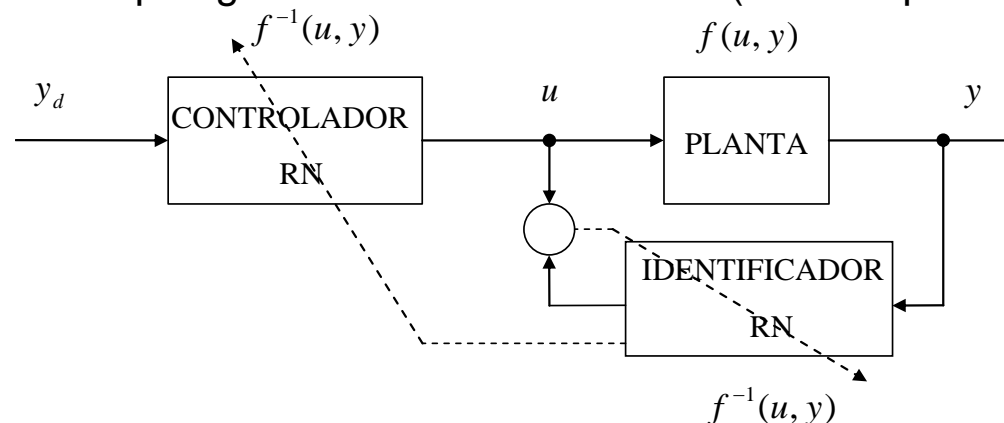
➤ Control inverso no orientado a minimizar el error de control

➤ RN identificadora

- Minimiza el error cuadrático de identificación de la planta (no el de control)
- Implementa la función inversa de la planta: $f^{-1}(u, y)$

➤ RN controladora

- Copia la topología de la RN identificadora (mismos pesos)

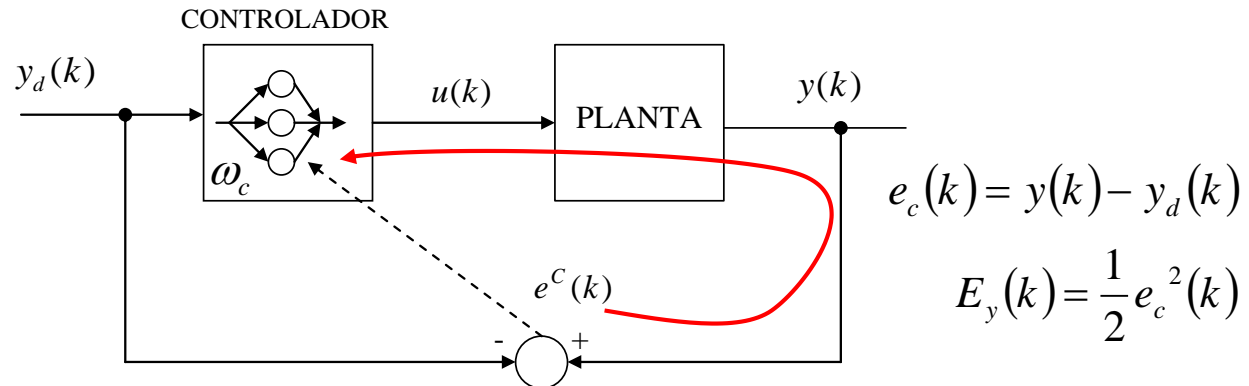


Control Neuronal

Arquitecturas de Control Neuronal

➤ Control inverso orientado a minimizar el error de control

- El controlador implementa la función inversa de la planta
- El controlador minimiza el error cuadrático de control (salida de la planta)
- Representa el esquema mas utilizado y de más altas prestaciones



➤ ¿Cómo se ajustan los pesos de la red controladora?

- Método de descenso por el gradiente

$$\omega_c(k+1) = \omega_c(k) + \Delta\omega_c \rightarrow \Delta\omega_c = -\alpha \cdot \frac{\partial E_y}{\partial \omega_c} = -\alpha \cdot \frac{\partial E_y}{\partial y} \cdot \frac{\partial y}{\partial u} \cdot \frac{\partial u}{\partial \omega_c} = -\alpha \cdot e_c \cdot J_{PLANTA} \cdot \frac{\partial u}{\partial \omega_c}$$

Depende de la topología de la RN

Control Neuronal

Arquitecturas de Control Neuronal

➤ Control inverso orientado a minimizar el error de control

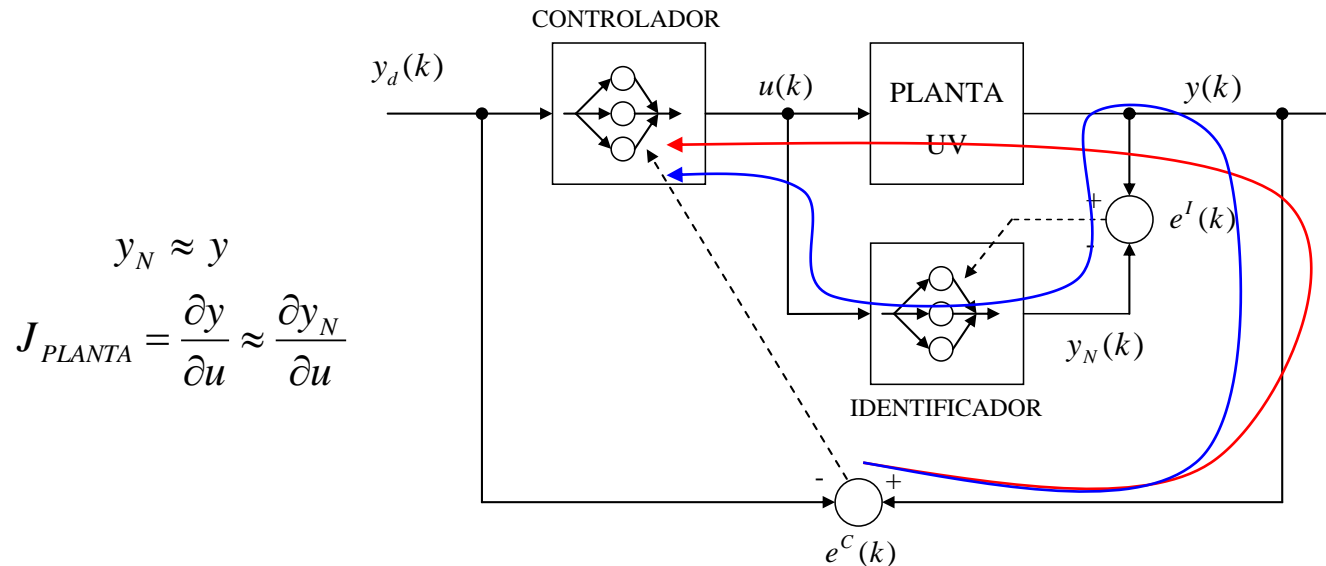
➤ Problema: El error está definido a la salida de la planta

➤ Pasar el error a la salida de la red neuronal exige el conocimiento del Jacobiano de la planta:

$$J_{PLANTA} = \frac{\partial y}{\partial u}$$

➤ ¿Cómo obtener el valor del Jacobiano de la planta J_{PLANTA} ?

➤ Utilizando una **RN identificadora**

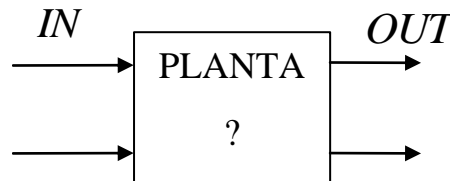


Identificadores con RNs

Identificadores con Redes Neuronales

➤ Identificadores neuronales

- Usar las RNs para modelar plantas desconocidas
 - Se basan en la capacidad de aprendizaje de las RNs para identificar sistemas (identificador universal)
 - Se usan un conjunto de entradas/salidas para el entrenamiento

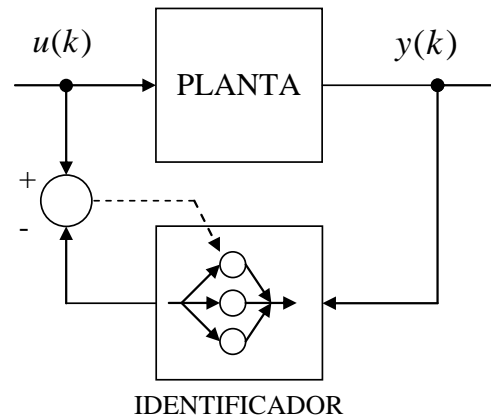


- No es necesario tener gran conocimiento de la planta
 - Modelo parametrizable de la planta física
 - Se ajustan los parámetros mediante una función de error entre la salida de la planta y del modelo y aplicando el algoritmo backpropagation

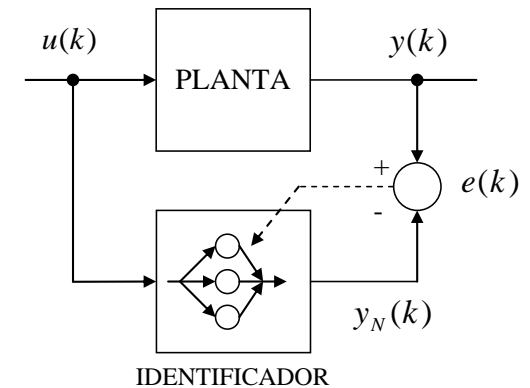
Identificadores con Redes Neuronales

➤ Configuraciones

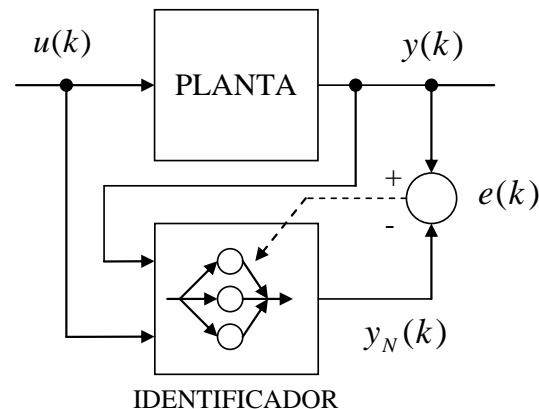
➤ Serie



➤ Paralelo



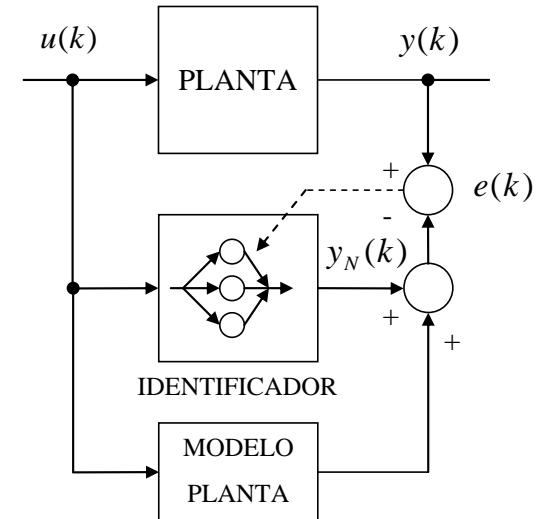
➤ Serie/Paralelo



Identificadores con Redes Neuronales

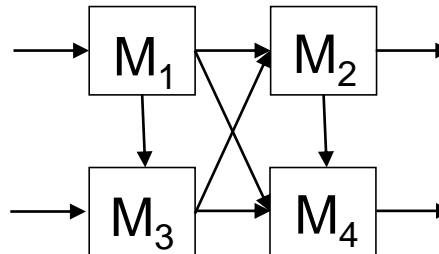
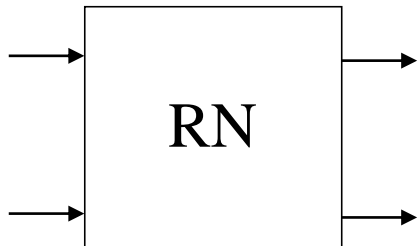
➤ Configuraciones

- Con conocimiento aproximado
 - La RN añade la parte que no recoge el modelo aproximado



➤ Redes modulares

- En la identificación de problemas complejos
 - Idea: dividir el problema en problemas sencillos



Control de sistemas SISO con RNs

Control de sistemas SISO mediante RNs

➤ Arquitecturas prácticas de control inverso

- A) Controlador predictivo
- B) Controlador NARMA-L2
- C) Controlador con modelo de referencia

➤ Controlador predictivo

- Identificación del sistema (off-line)
 - Usa una RN para predecir la salida de una planta no lineal
- Diseño del controlador (on-line)
 - El controlador calcula la entrada de control que optimiza el rendimiento de la planta sobre un horizonte temporal futuro
 - Utiliza el modelo neuronal de la planta previamente calculado en modo off-line

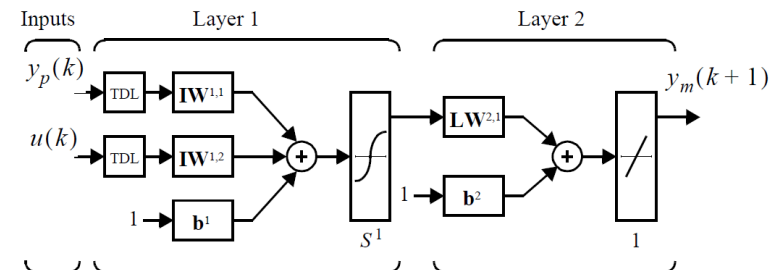
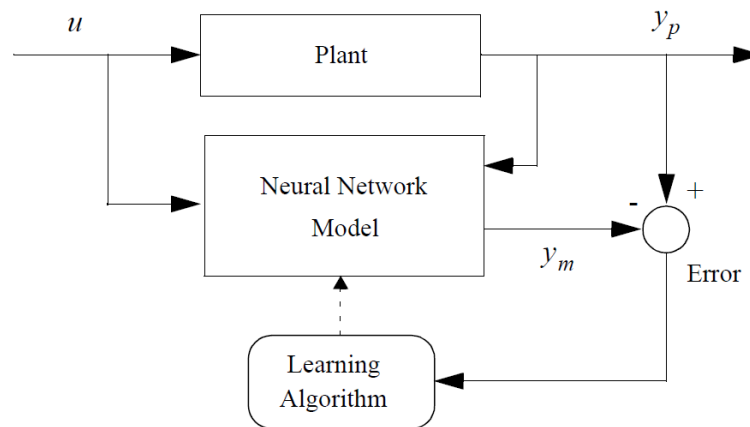
Control Neuronal

Control de sistemas SISO mediante RNs

➤ Controlador predictivo

➤ Identificación del sistema (off-line)

➤ Usa una RN para predecir la salida de una planta no lineal



➤ Modelo estándar usado en la identificación no lineal

➤ NARMA (Nonlinear Autoregressive-Moving Average)

$$y_p(k+d) = h[y_p(k), y_p(k-1), \dots, y_p(k-n+1), u(k), u(k-1), \dots, u(k-m+1)]$$

$$y_m(k+1) = \hat{h}[y_p(k), y_p(k-1), \dots, y_p(k-n+1), u(k), u(k-1), \dots, u(k-m+1); \mathbf{x}]$$

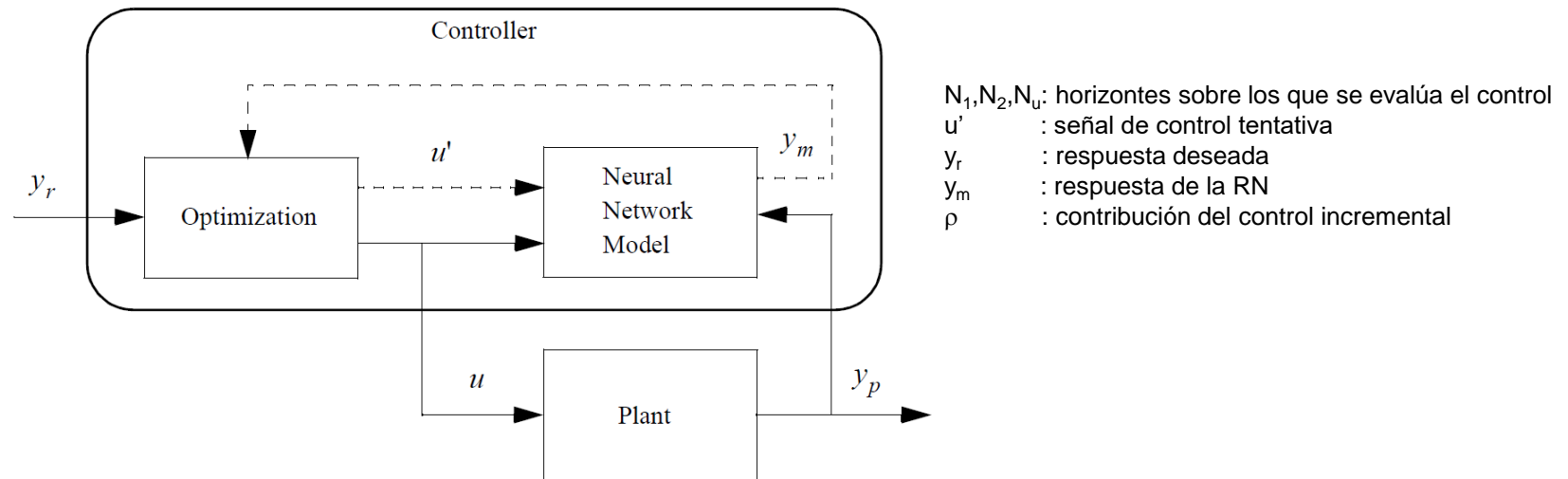
➤ Entrenamiento usando backpropagation estático (no hay realimentación) 20

Control Neuronal

Control de sistemas SISO mediante RNs

➤ Controlador predictivo

➤ Diseño del controlador (on-line)



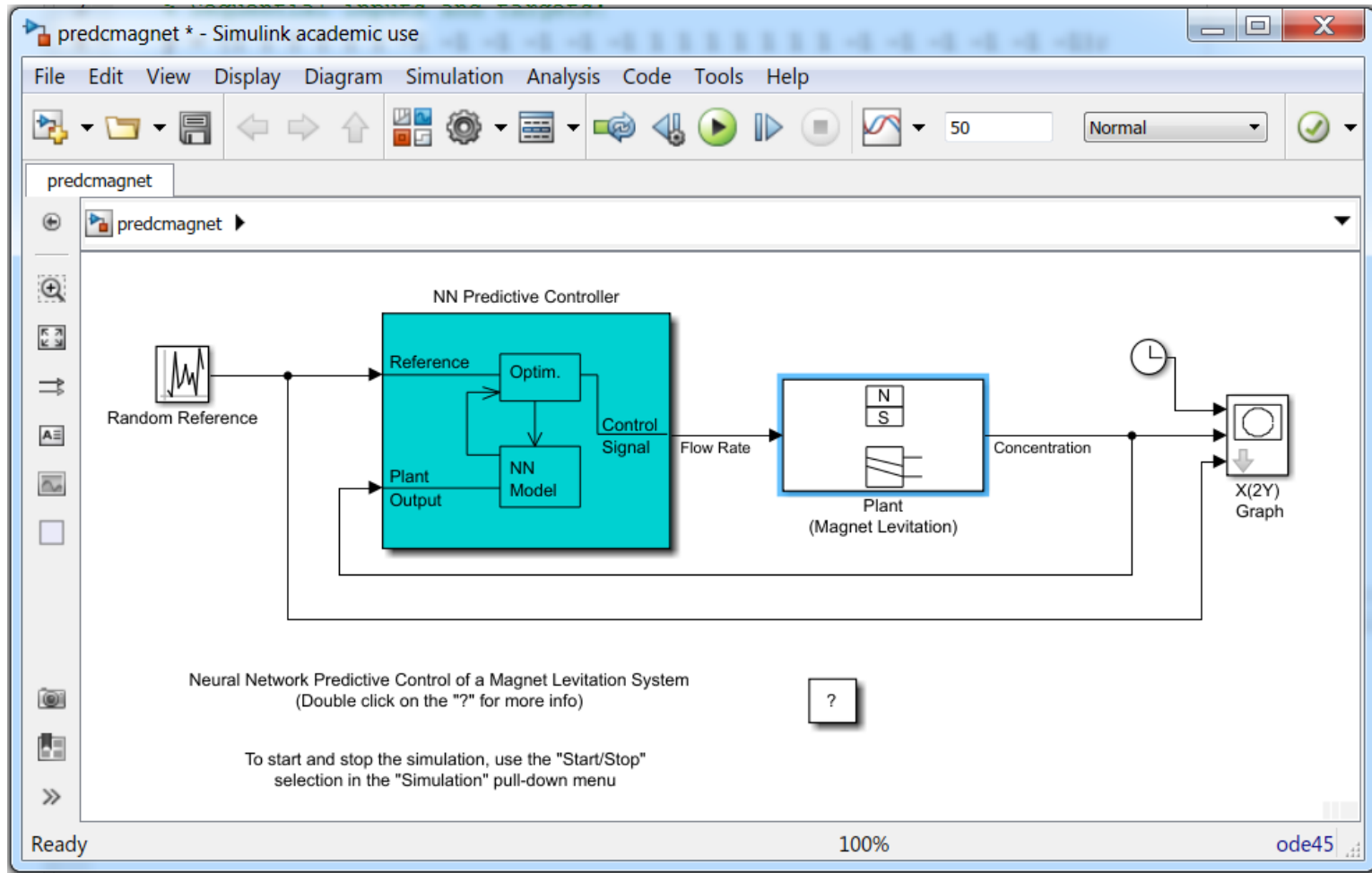
➤ Predicciones de la RN se usan por un módulo de optimización para minimizar J

$$J = \sum_{j=N_1}^{N_2} (y_r(k+j) - y_m(k+j))^2 + \rho \sum_{j=1}^{N_u} (u'(k+j-1) - u'(k+j-2))^2$$

➤ El módulo de optimización determina (u' , u) usando el algoritmo BFGS quasi-Newton (Broyden–Fletcher–Goldfarb–Shanno quasi-Newton)

Control Neuronal

Controlador predictivo. MATLAB



Control Neuronal

Controlador predictivo. MATLAB

The screenshot shows the 'Neural Network Predictive Control' dialog box. It has a menu bar with 'File', 'Window', and 'Help'. The main area contains several input fields and buttons. Callouts provide explanations for various parameters and buttons:

- File menu:** The File menu has several items, including ones that allow you to import and export controller and plant networks.
- Cost Horizon (N2):** The Cost Horizon N2 is the number of time steps over which the prediction errors are minimized. (Value: 7)
- Control Horizon (Nu):** The Control Horizon Nu is the number of time steps over which the control increments are minimized. (Value: 2)
- Control Weighting Factor (ρ):** The Control Weighting Factor multiplies the sum of squared control increments in the performance function. (Value: 0.05)
- Search Parameter (α):** This parameter determines when the line search. (Value: 0.001)
- Minimization Routine:** A dropdown menu currently showing 'csrchbac'.
- Iterations Per Sample Time:** A field for the number of iterations. (Value: 2)
- Buttons:** 'Plant Identification', 'OK', 'Cancel', and 'Apply'.
- Bottom text:** 'Perform plant identification before controller configuration.'

Additional callouts for the buttons:

- Plant Identification:** This button opens the Plant Identification window. The plant must be identified before the controller is used.
- OK/Apply:** After the controller parameters have been set, select OK or Apply to load the parameters into the Simulink model.
- Iterations:** This selects the number of iterations of the optimization algorithm to be performed at each sample time.

Control Neuronal

Controlador predictivo. MATLAB

The screenshot shows the 'Plant Identification' dialog box in MATLAB. It is divided into several sections: 'Network Architecture', 'Training Data', and 'Training Parameters'. Callouts provide detailed explanations for various fields and buttons.

Network Architecture:

- Size of Hidden Layer:** Set to 7. Callout: "The number of neurons in the first layer of the plant model network."
- No. Delayed Plant Inputs:** Set to 2. Callout: "You can define the size of the two tapped delay lines coming into the plant model."
- No. Delayed Plant Outputs:** Set to 2.
- Sampling Interval (sec):** Set to 0.2. Callout: "Interval at which the program collects data from the Simulink plant model."
- Normalize Training Data:** A checkbox that is currently unchecked. Callout: "You can normalize the data using the premmx function."

Training Data:

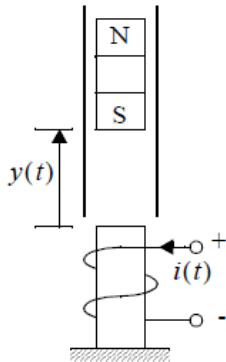
- Training Samples:** Set to 8000. Callout: "Number of data points generated for training, validation, and test sets."
- Limit Output Data:** A checked checkbox. Callout: "You can select a range on the output data to be used in training."
- Maximum Plant Input:** Set to 4.
- Minimum Plant Input:** Set to 0.
- Maximum Interval Value (sec):** Set to 20. Callout: "The random plant input is a series of steps of random height occurring at random intervals. These fields set the minimum and maximum height and interval."
- Minimum Interval Value (sec):** Set to 5.
- Simulink Plant Model:** Set to 'CSTR'. Callout: "Simulink plant model used to generate training data (file with .mdl extension)."
- Buttons:** 'Generate Training Data', 'Import Data', and 'Export Data'. Callout for 'Generate Training Data': "This button starts the training data generation."

Training Parameters:

- Training Epochs:** Set to 200. Callout: "Number of iterations of plant training to be performed."
- Training Function:** Set to 'trainlm'. Callout: "You can use any training function to train the plant model."
- Use Current Weights:** A checked checkbox. Callout: "Select this option to continue training with current weights. Otherwise, you use randomly generated weights."
- Use Validation Data:** A checked checkbox. Callout: "You can use validation (early stopping) and testing data during training."
- Use Testing Data:** An unchecked checkbox.
- Buttons:** 'Train Network', 'OK', 'Cancel', and 'Apply'. Callout for 'Train Network': "This button begins the plant model training. Generate or import data before training."
- Footer:** "Generate or import data before training the neural network."

Controlador predictivo. Ejemplo

➤ Sistema de levitación magnético



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

M: masa del imán	M=3
g: fuerza de la gravedad	g=9.8
β : coeficiente de fricción viscosa	$\beta=12$
α : coeficiente de fuerza de campo	$\alpha=15$

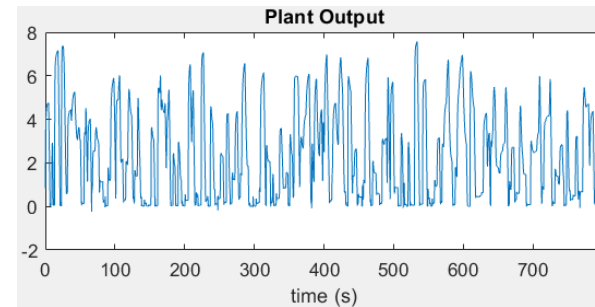
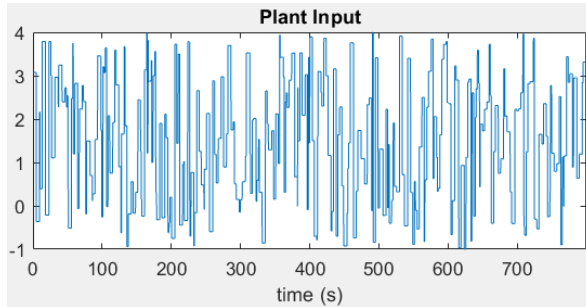
➤ Identificación de la planta

- $inp_{del}=3, out_{del}=2, y_m(k+1) = \hat{h}[y_p(k), y_p(k-1), y_p(k-2), i(k), i(k-1), i(k-2), i(k-3); \mathbf{x}]$
- 9 neuronas en la capa oculta, $f_s = 0.1 \text{ s}$
- Generación de datos de entrenamiento
 - Training samples: 8000
 - Plant input: [-1, 4]
 - Interval value: $0.5 < \tau < 5$
 - Plant output: [0, inf]
 - Plant model: ballrepel0

Control Neuronal

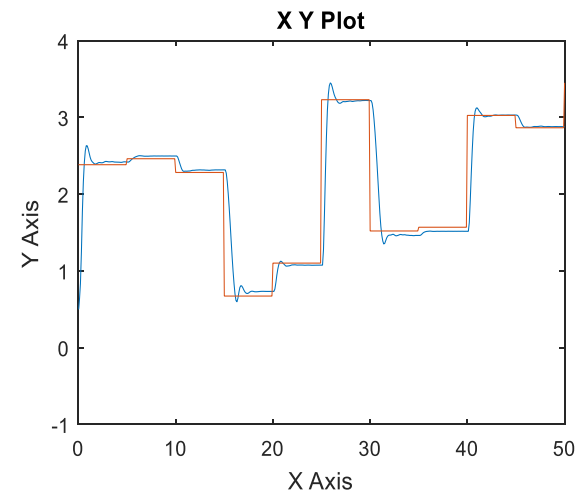
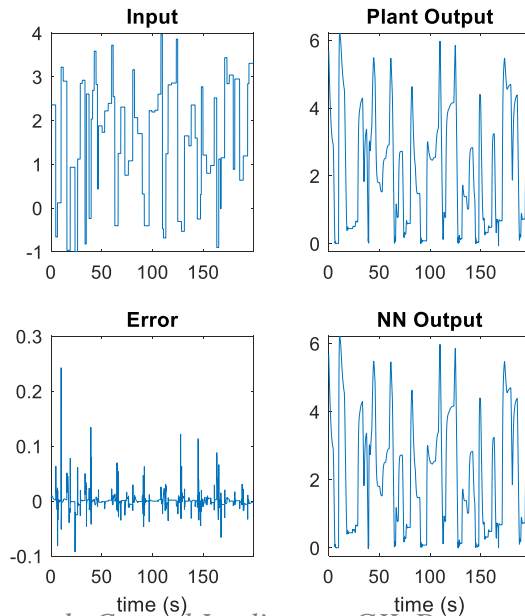
Controlador predictivo. Ejemplo

➤ Identificación de la planta



➤ Diseño del controlador

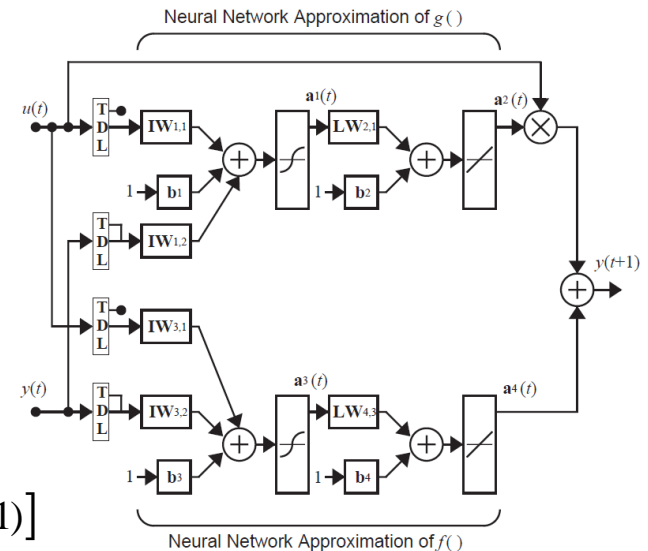
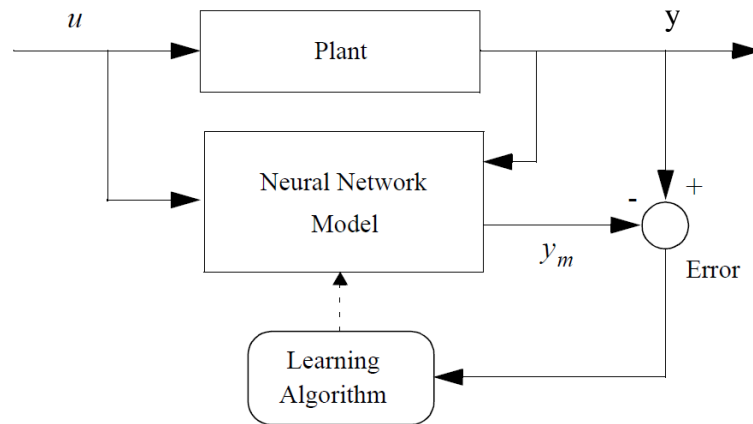
$$N_1 = 1, N_2 = 7$$
$$N_u = 3, \rho = 0.01$$



Control Neuronal

Control de sistemas SISO mediante RNs

- **Controlador NARMA-L2 o de linearización realimentada**
 - Identifica la planta mediante un modelo NARMA off-line
 - El modelo identificado se usa para desarrollar el controlador on-line
 - Transforma una planta no lineal en lineal mediante la cancelación de las no linealidades
- **Identificación del modelo NARMA-L2 (off-line)**



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

Control Neuronal

Control de sistemas SISO mediante RNs

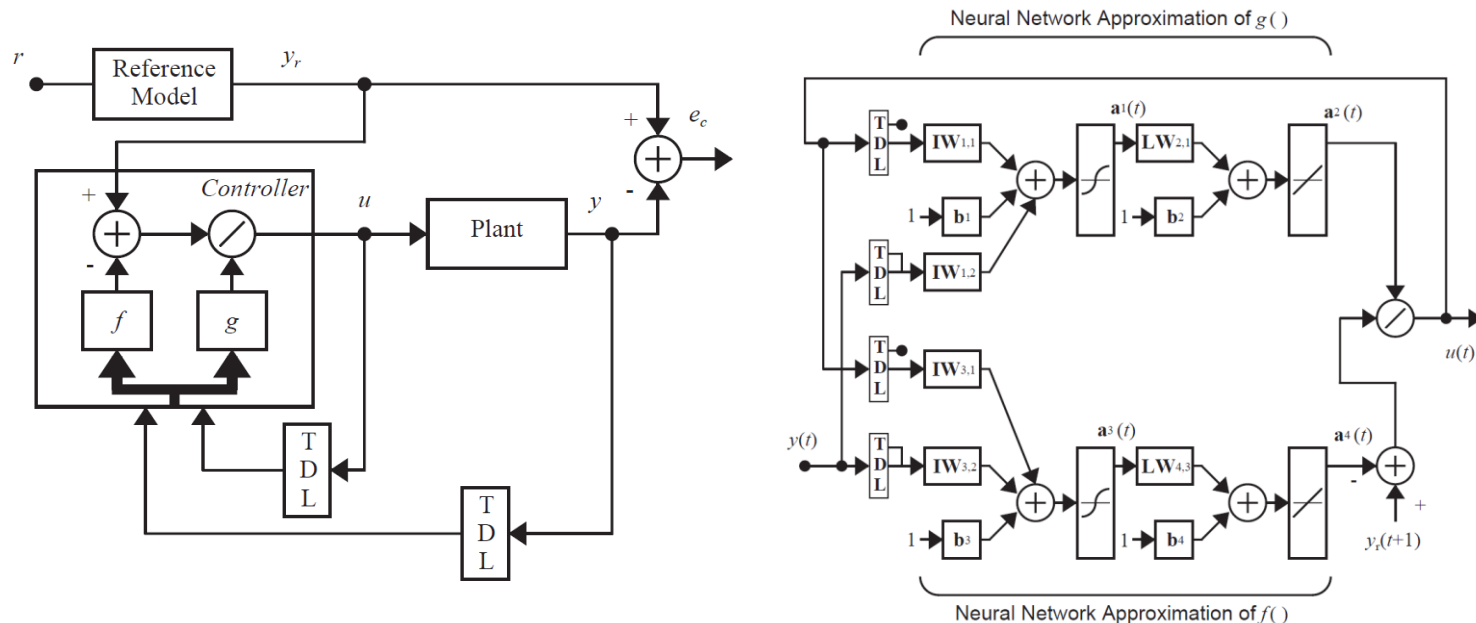
➤ Controlador NARMA-L2

➤ Diseño del controlador

- Considerando $y(k+d)=y_r(k+d)$

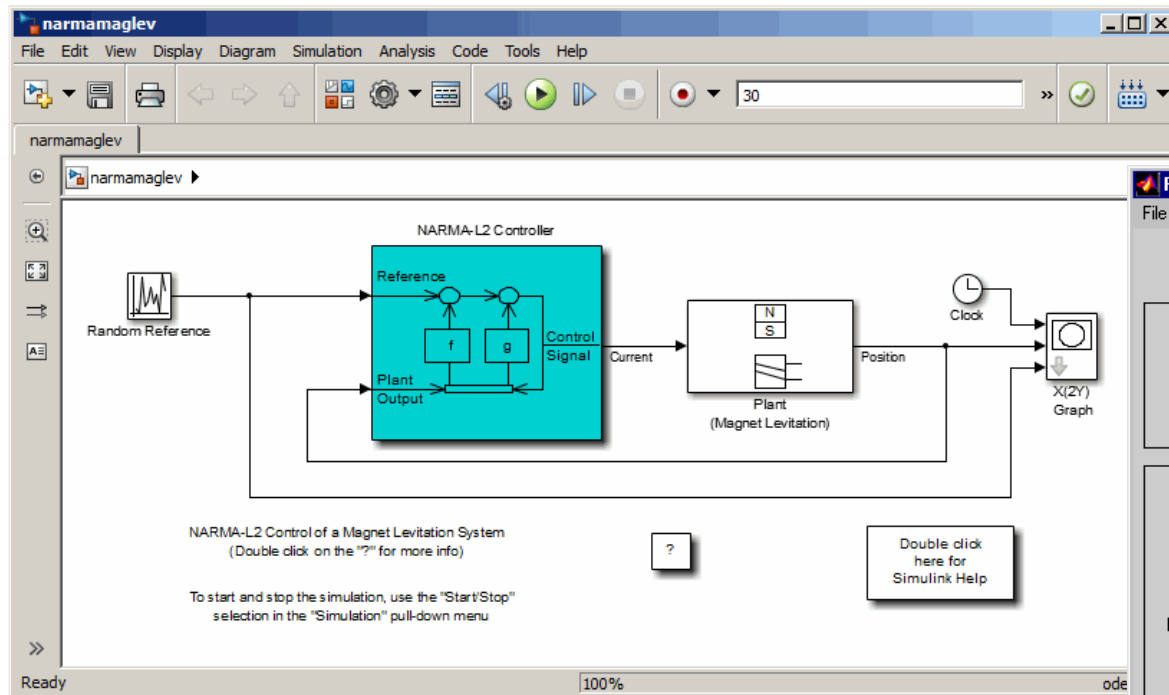
$$u(k) = \frac{y_r(k+d) - f[y(k), y(k-1), \dots, y(k-n+1), u(k-1), \dots, u(k-m+1)]}{g[y(k), y(k-1), \dots, y(k-n+1), u(k-1), \dots, u(k-m+1)]} \quad d \geq 1$$

- Entrenamiento usando backpropagation estático (no hay realimentación)



Control Neuronal

Control SISO NARMA-L2. MATLAB



Plant Identification - NARMA-L2

File Window Help

Plant Identification - NARMA-L2

Network Architecture

Size of Hidden Layer No. Delayed Plant Inputs

Sampling Interval (sec) No. Delayed Plant Outputs

☒ Normalize Training Data

Training Data

Training Samples ☒ Limit Output Data

Maximum Plant Input Maximum Plant Output

Minimum Plant Input Minimum Plant Output

Maximum Interval Value (sec) Simulink Plant Model:

Minimum Interval Value (sec)

Training Parameters

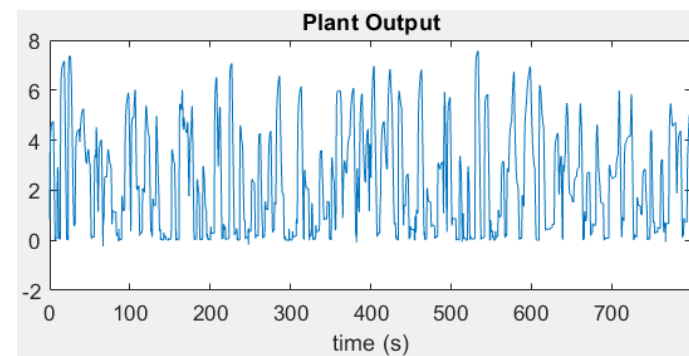
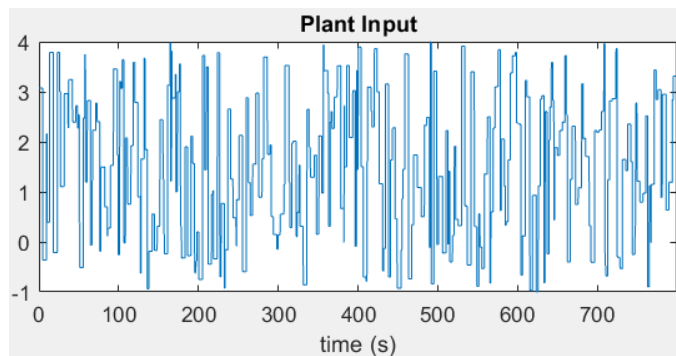
Training Epochs Training Function

☐ Use Current Weights ☒ Use Validation Data ☒ Use Testing Data

Generate or import data before training the neural network.

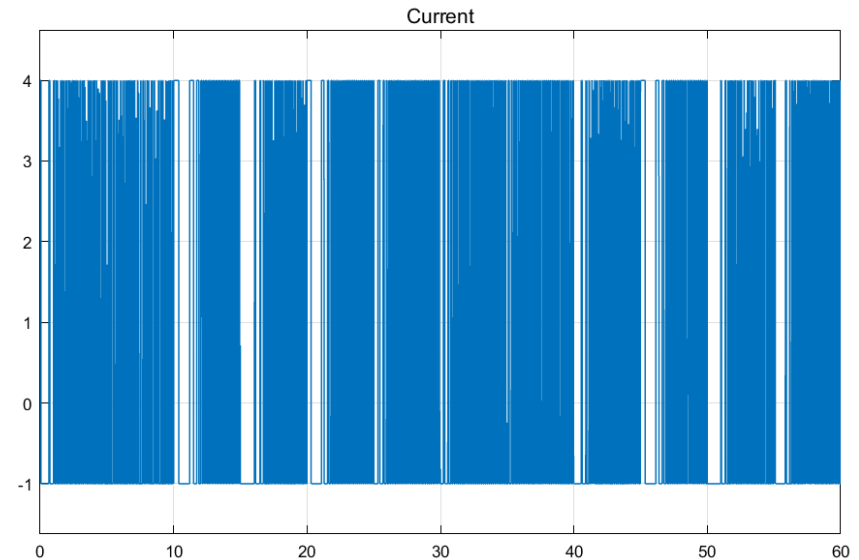
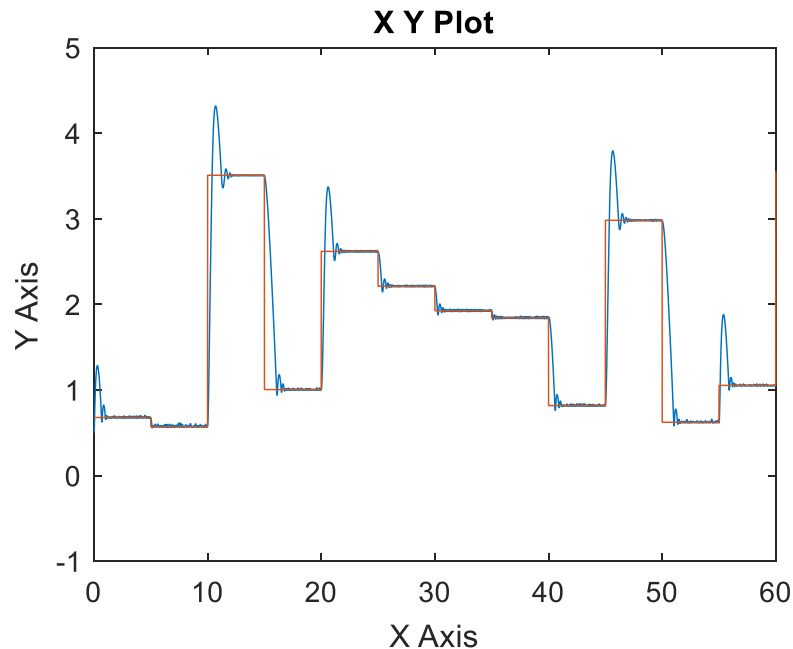
Control SISO NARMA-L2. Ejemplo

- Sistema de levitación magnético
 - Identificación de la planta (mismo que en el caso anterior)
 - $inp_{del}=3, out_{del}=2, y_m(k+1) = \hat{h}[y_p(k), y_p(k-1), y_p(k-2), i(k), i(k-1), i(k-2), i(k-3); \mathbf{x}]$
 - 9 neuronas en la capa oculta, $f_s = 0.1 \text{ s}$
 - Generación de datos de entrenamiento
 - Training samples: 8000
 - Plant input: $[-1, 4]$
 - Interval value: $0.5 < \tau < 5$
 - Plant output: $[0, \inf]$
 - Plant model: *ballrepel0*



Control SISO NARMA-L2. Ejemplo

- Diseño del controlador
 - La salida sigue la señal de referencia de forma precisa
 - NARMA-L2 produce más oscilaciones en la señal de control que los otros controladores
 - Estas oscilaciones se pueden reducir mediante un proceso de filtrado



Control de sistemas SISO mediante RNs

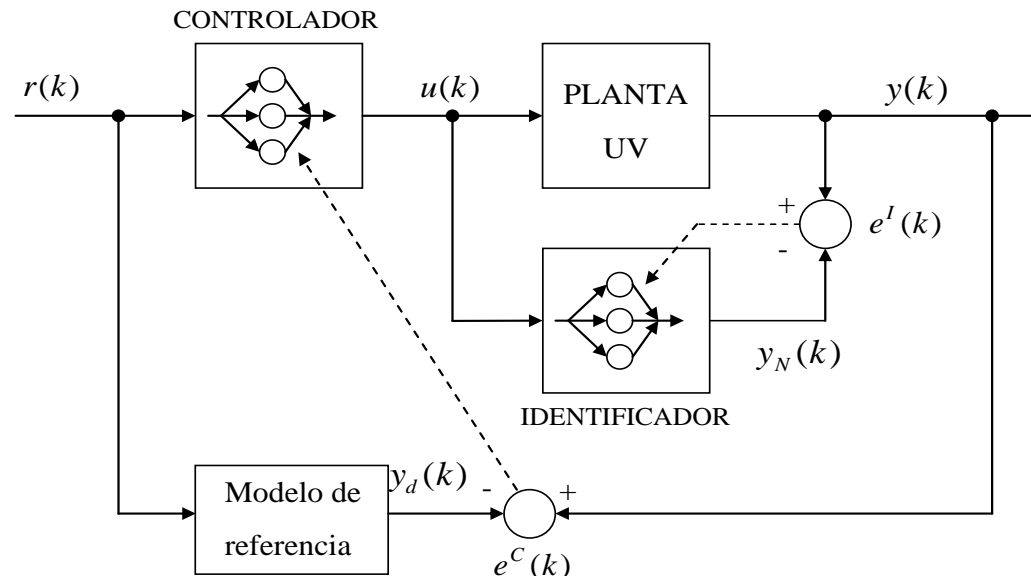
➤ Controlador con modelo de referencia

➤ Neuroidentificador:

- Es una RN que modela de forma *on-line* la planta
- La RN se ajusta tratando de minimizar el error cuadrático de identificación $e^I(k)$

➤ Neurocontrolador

- RN que genera la señal de control $u(k)$ en modo on-line
- Trata de minimizar el error cuadrático de control $e^C(k)$

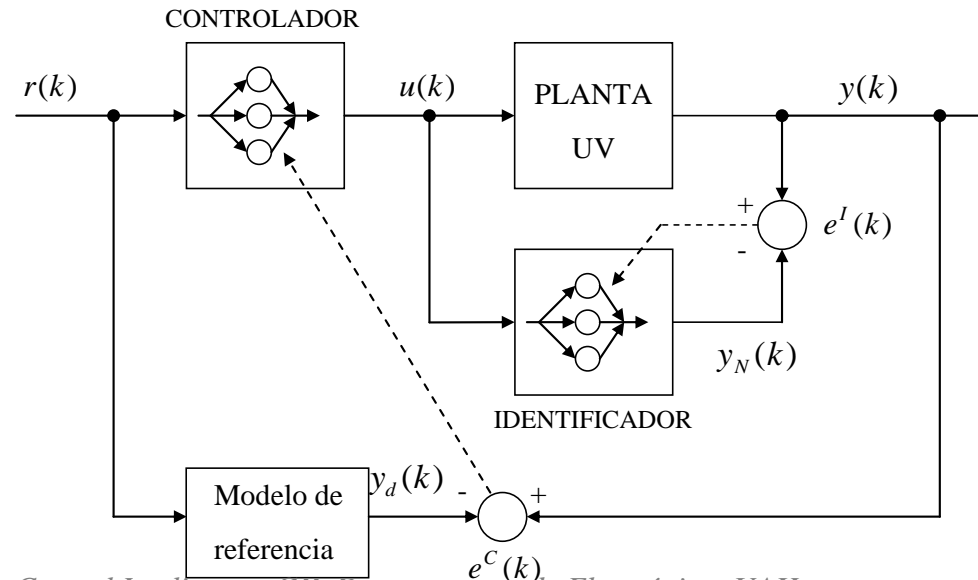


Control de sistemas SISO mediante RNs

➤ Con modelo de referencia

➤ Modelo de referencia

- Hace de filtro paso bajo para obtener una señal de referencia de entrada al controlador que pueda ser seguida por el sistema
- La constante de tiempo del modelo de referencia similar a la de la planta (versión linealizada de la planta)
- Debe ser estable. Condición adicional a las condiciones de estabilidad en el aprendizaje de las RNs individuales



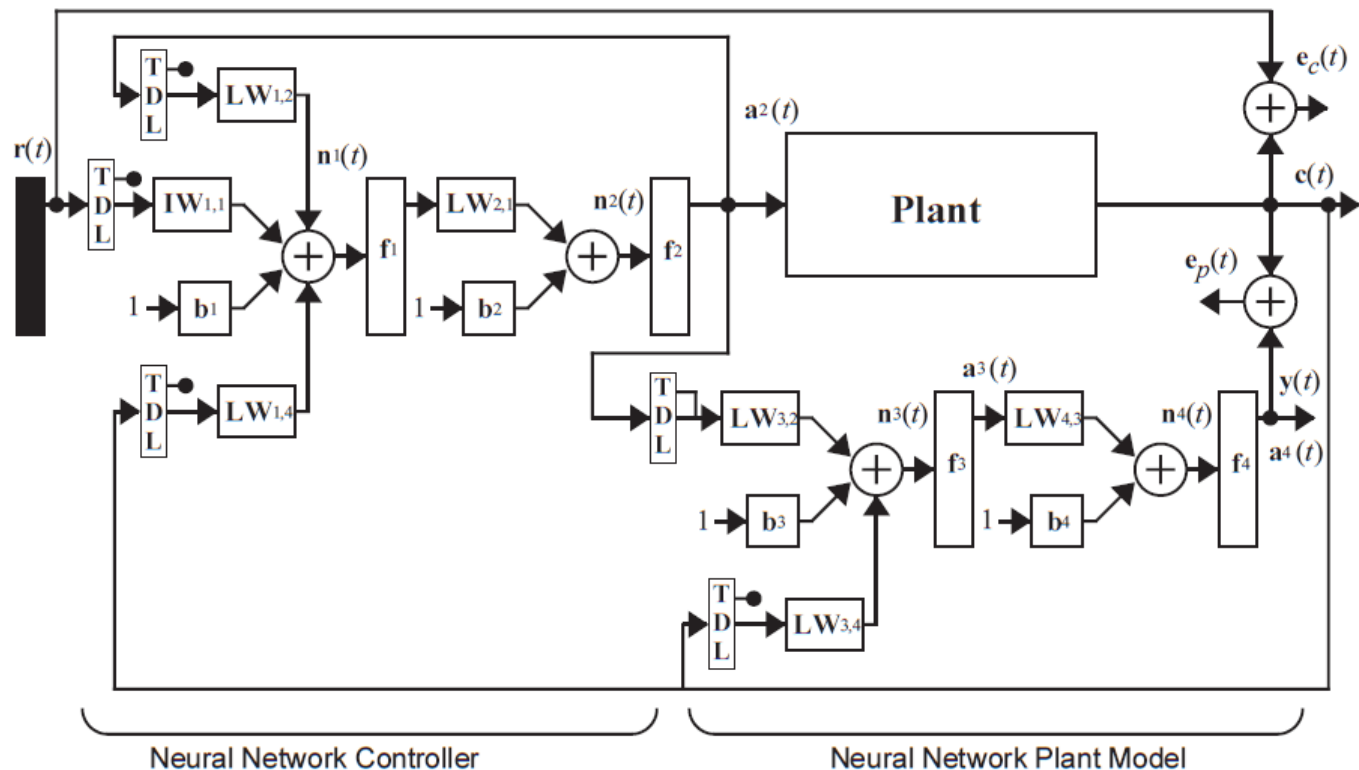
Control Neuronal

Control de sistemas SISO mediante RNs

➤ Con modelo de referencia

➤ Modelo MATLAB

- Red recurrente (realimentada). Entrenamiento usando backpropagation dinámica



Control Neuronal

Control de sistemas SISO mediante RNs

➤ Con mod. de referencia

$$E^I(k) = \frac{1}{2} (y(k) - y_N(k))^2 = \frac{1}{2} (e^I(k))^2$$

$$E^C(k) = \frac{1}{2} (y(k) - y_d(k))^2 = \frac{1}{2} (e^C(k))^2$$

$$J(k) = \frac{\partial y(k)}{\partial u(k)} \approx \frac{\partial y_N(k)}{\partial u(k)}$$

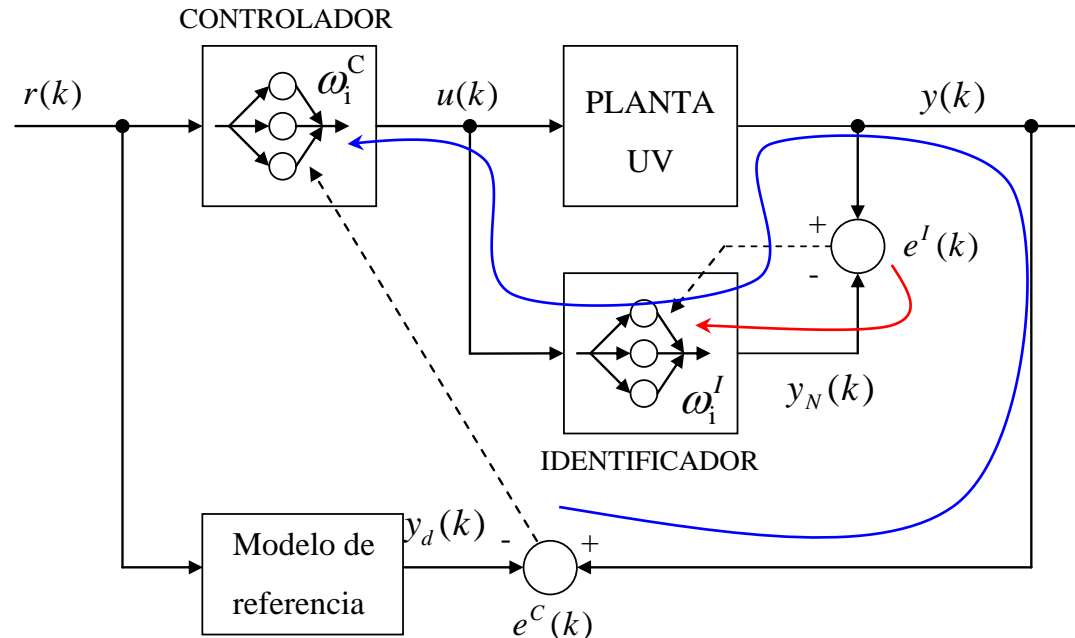
➤ Ajuste de pesos:

- Dos ratios de aprendizaje distintos α_1 (identificador) y α_2 (controlador)

$$\Delta \omega_i^I(k) = -\alpha_1 \frac{\partial E^I(k)}{\partial \omega_i^I(k)} = -\alpha_1 \cdot \frac{\partial E^I(k)}{\partial y_N(k)} \cdot \frac{\partial y_N(k)}{\partial \omega_i^I(k)} = -\alpha_1 \cdot (-e^I(k)) \cdot \frac{\partial y_N^I(k)}{\partial \omega_i^I(k)}$$

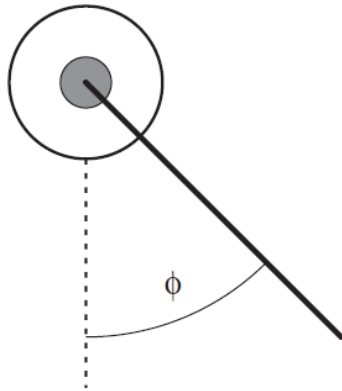
Depende de la topología de la RN

$$\Delta \omega_i^C(k) = -\alpha_2 \cdot \frac{\partial E^C(k)}{\partial \omega_i^C(k)} = -\alpha_2 \cdot \frac{\partial E^C(k)}{\partial y(k)} \cdot \frac{\partial y(k)}{\partial u(k)} \cdot \frac{\partial u(k)}{\partial \omega_i^C(k)} = -\alpha_2 \cdot e^C(k) \cdot J(k) \cdot \frac{\partial u(k)}{\partial \omega_i^C(k)}$$



Control con modelo de referencia. Ejemplo

➤ Control de un brazo robot



$$\text{Modelo planta} \rightarrow \frac{d^2\phi}{dt^2} = -10\sin\phi - 2\frac{d\phi}{dt} + u$$

$$\text{Modelo referencia} \rightarrow \frac{d^2y_r}{dt^2} = -9y_r - 6\frac{dy_r}{dt} + 9r$$

Φ : ángulo que forma el brazo con la vertical

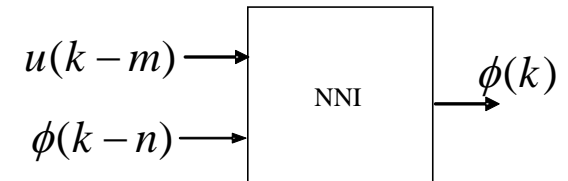
u : torsión aplicada por el motor DC

r : señal de referencia

y_r : salida del modelo de referencia

➤ Identificación de la planta

- Intervalo de muestreo: 0.05
- Anchura de pulsos de 0.1 a 2 s
- Amplitud de pulsos de -15 a 15 N-m
- Retardadores de la entrada: ($m=0,1,2$)
- Retardadores de la salida: ($n=1,2$)
- Topología de la red: 5-10-1



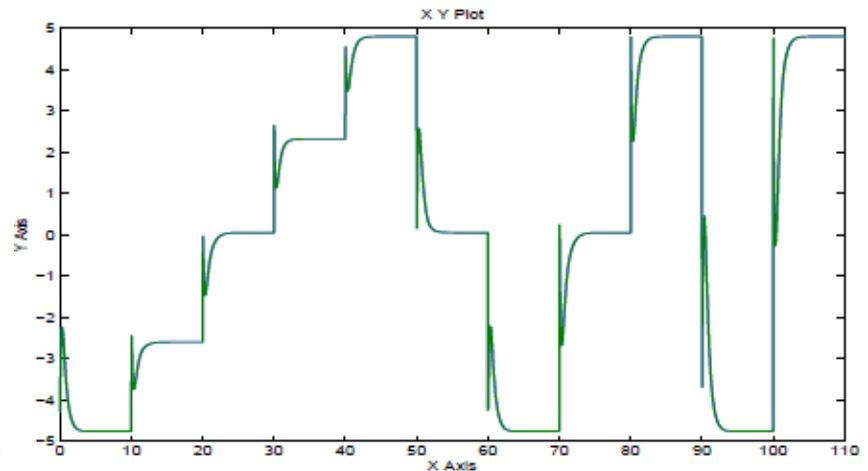
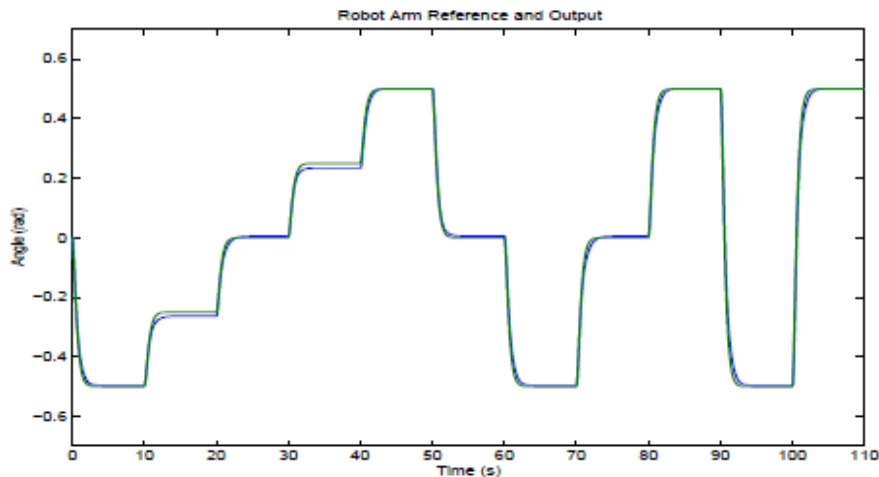
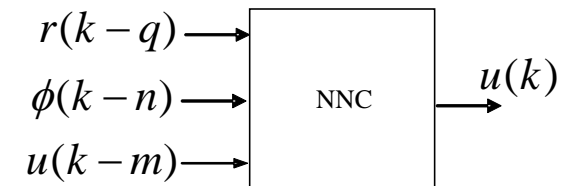
Control Neuronal

Control con modelo de referencia. Ejemplo

➤ Control de un brazo robot

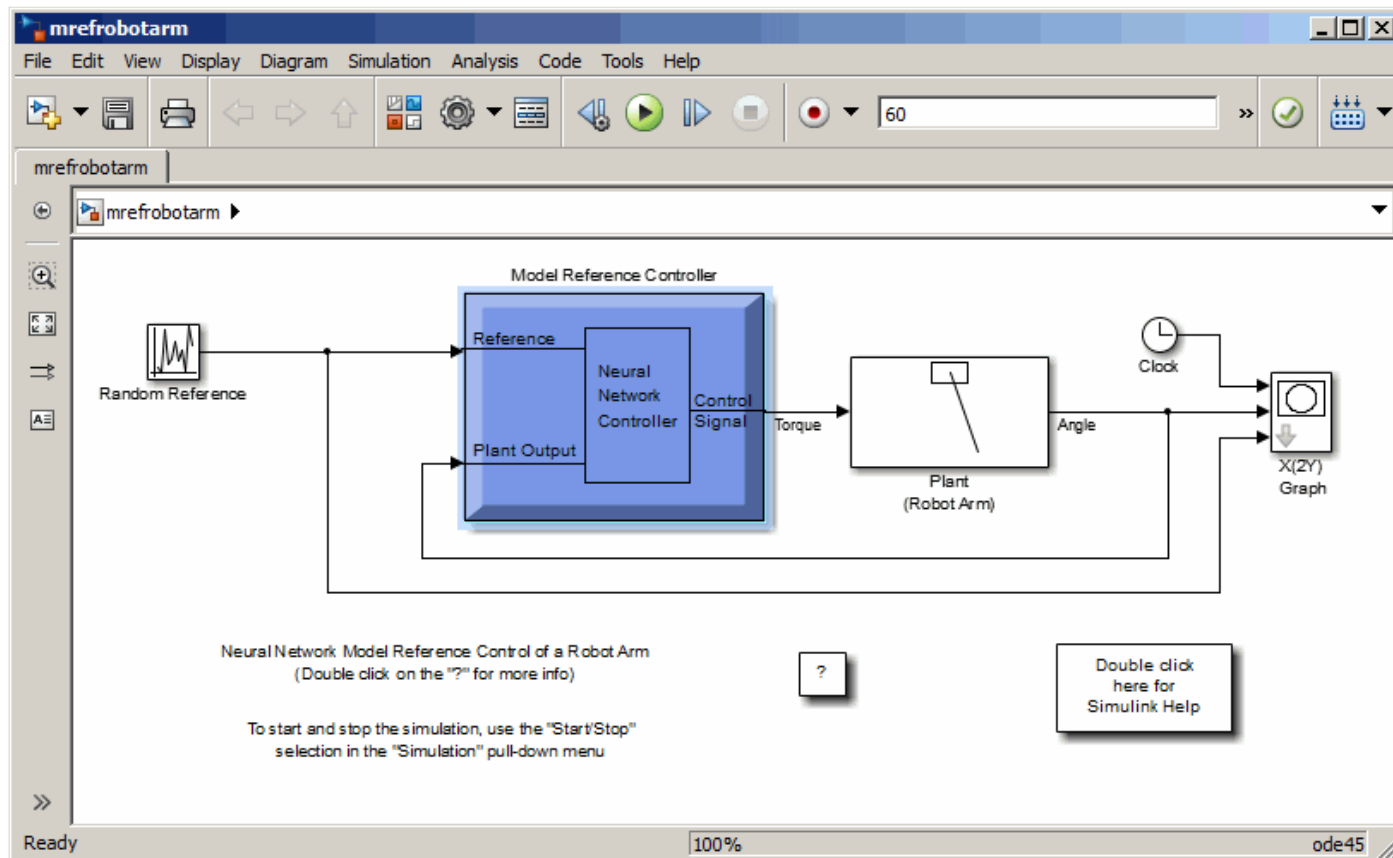
➤ Diseño del controlador

- Amplitud referencia: $-\pi/4$ a $\pi/4$
- Anchura pulsos de referencia: 0.1 a 2 s
- Retardadores: $(m=1)$, $(n=1,2)$, $(q=1,2)$
- Topología de la red: 5-13-1
- Entrenamiento usando BFGS quasi-Newton algorithm con backpropagation dinámica para calcular los gradientes.



Control Neuronal

Control con modelo de referencia. Ejemplo



Control Neuronal

Control con modelo de referencia. Ejemplo

The screenshot shows the 'Model Reference Control' dialog box with the following sections and callouts:

- File Menu:** The file menu has several items, including ones that allow you to import and export controller and plant networks.
- Network Architecture:** This block specifies the inputs to the controller. It includes fields for 'Size of Hidden Layer' (13), 'No. Delayed Reference Inputs' (2), 'Sampling Interval (sec)' (0.05), 'No. Delayed Controller Outputs' (1), and 'No. Delayed Plant Outputs' (2). There is a checkbox for 'Normalize Training Data'.
- Training Data:** You must specify a Simulink reference model for the plant to follow. This section includes 'Maximum Reference Value' (0.7), 'Minimum Reference Value' (-0.7), 'Maximum Interval Value (sec)' (2), 'Minimum Interval Value (sec)' (0.1), 'Controller Training Samples' (200), and a 'Reference Model' field with a 'Browse' button. Below these are buttons for 'Generate Training Data', 'Import Data', and 'Export Data'.
- Training Parameters:** The training data is broken into segments. Specify the number of training epochs for each segment. This section includes 'Controller Training Epochs' (10), 'Controller Training Segments' (2), a checkbox for 'Use Current Weights' (checked), and a checkbox for 'Use Cumulative Training' (unchecked). At the bottom are buttons for 'Plant Identification', 'Train Controller', 'OK', 'Cancel', and 'Apply'.
- Buttons:**
 - 'Plant Identification': This button opens the Plant Identification window. The plant must be identified before the controller is trained.
 - 'Train Controller': After the controller has been trained, select OK or Apply to load the network into the Simulink model.
 - 'OK' and 'Apply': If selected, segments of data are added to the training set as training continues. Otherwise, only one segment at a time is used.

A blue text instruction at the bottom of the dialog box states: **Perform plant identification before controller training.**

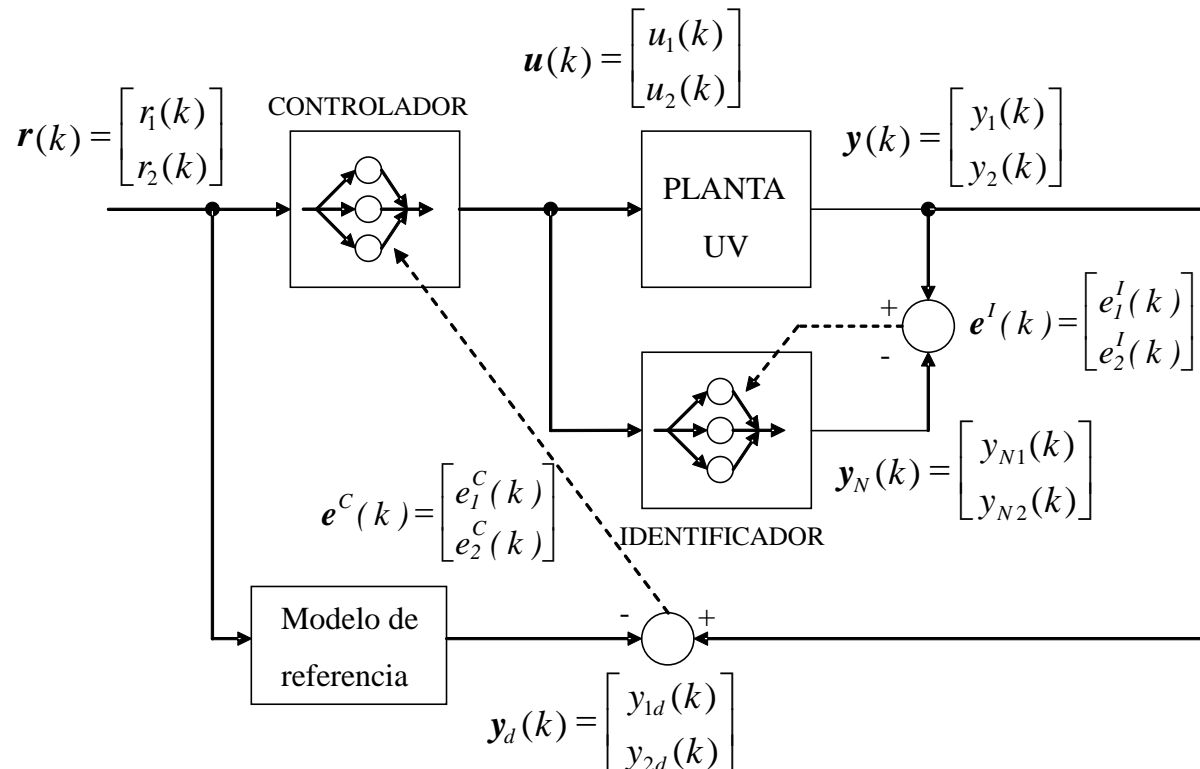
Control de sistemas MIMO con RNs

Control Neuronal

Control de sistemas MIMO mediante RNs

➤ Con modelo de referencia

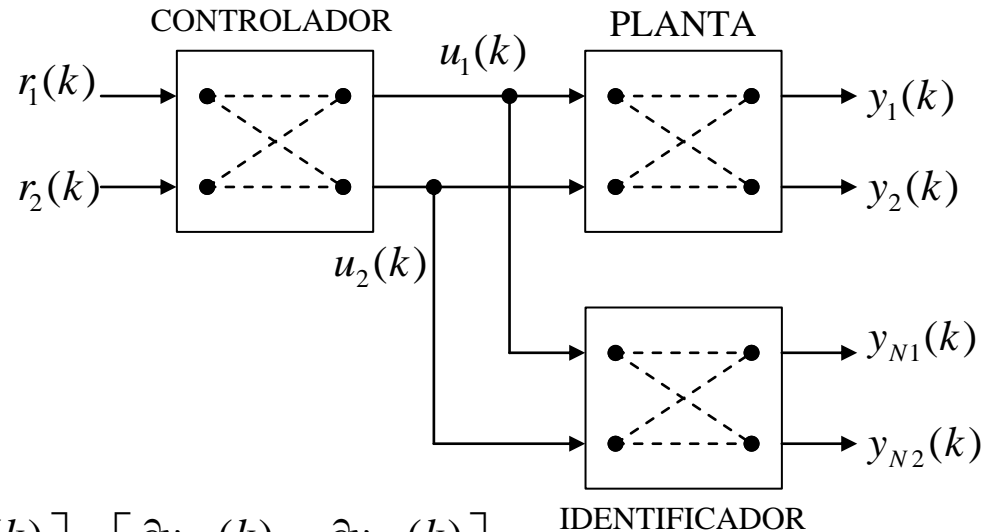
- Arquitectura
- $$E^I(k) = \frac{1}{2} (\mathbf{e}^I(k))^T (\mathbf{e}^I(k)) = \frac{1}{2} (y_1(k) - y_{N1}(k))^2 + \frac{1}{2} (y_2(k) - y_{N2}(k))^2 = \frac{1}{2} (e_1^I(k))^2 + \frac{1}{2} (e_2^I(k))^2$$
- $$E^C(k) = \frac{1}{2} (\mathbf{e}^C(k))^T (\mathbf{e}^C(k)) = \frac{1}{2} (y_1(k) - y_{d1}(k))^2 + \frac{1}{2} (y_2(k) - y_{d2}(k))^2 = \frac{1}{2} (e_1^C(k))^2 + \frac{1}{2} (e_2^C(k))^2$$



Control de sistemas MIMO mediante RNs

➤ Con modelo de referencia

➤ Arquitectura



$$J(k) = \begin{bmatrix} J_{11}(k) & J_{12}(k) \\ J_{21}(k) & J_{22}(k) \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1(k)}{\partial u_1(k)} & \frac{\partial y_1(k)}{\partial u_2(k)} \\ \frac{\partial y_2(k)}{\partial u_1(k)} & \frac{\partial y_2(k)}{\partial u_2(k)} \end{bmatrix} \approx \begin{bmatrix} \frac{\partial y_{N1}(k)}{\partial u_1(k)} & \frac{\partial y_{N1}(k)}{\partial u_2(k)} \\ \frac{\partial y_{N2}(k)}{\partial u_1(k)} & \frac{\partial y_{N2}(k)}{\partial u_2(k)} \end{bmatrix}$$

Aproximación

$$y_{N1}(k) \approx y_1(k)$$

$$y_{N2}(k) \approx y_2(k)$$

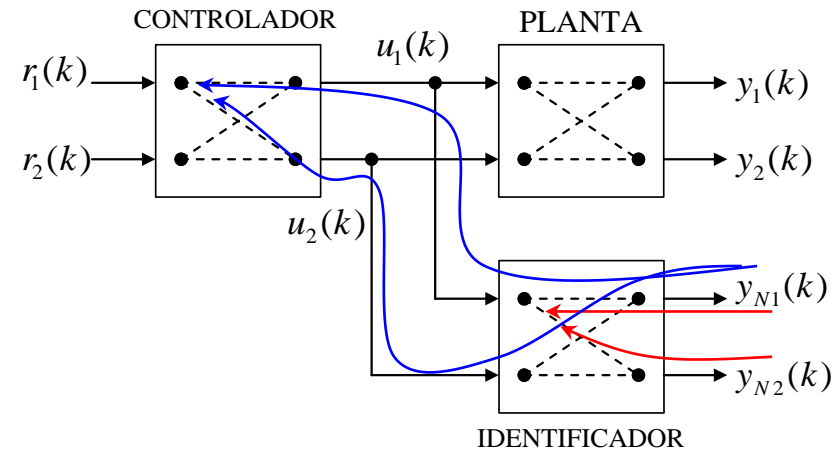
Control Neuronal

Control de sistemas MIMO mediante RNs

➤ Con modelo de referencia

➤ Ajuste de pesos

- Dos ratios de aprendizaje distintos
 α_1 (identificador) y α_2 (controlador)



$$\Delta \omega_i^I(k) = -\alpha_1 \frac{\partial E^I(k)}{\partial \omega_i^I(k)} = -\alpha_1 \cdot \left(\frac{\partial E^I(k)}{\partial y_{N1}(k)} \cdot \frac{\partial y_{N1}(k)}{\partial \omega_i^I(k)} + \frac{\partial E^I(k)}{\partial y_{N2}(k)} \cdot \frac{\partial y_{N2}(k)}{\partial \omega_i^I(k)} \right) = +\alpha_1 \cdot \left(e_1^I(k) \cdot \frac{\partial y_{N1}^I(k)}{\partial \omega_i^I(k)} + e_2^I(k) \cdot \frac{\partial y_{N2}^I(k)}{\partial \omega_i^I(k)} \right)$$

$$\Delta \omega_i^C(k) = -\alpha_2 \cdot \frac{\partial E^C(k)}{\partial \omega_i^C(k)} \approx \leftarrow \mathbf{y}_N(k) \approx \mathbf{y}(k)$$

$$-\alpha_2 \cdot \left(\frac{\partial E^C(k)}{\partial y_{N1}(k)} \left(\frac{\partial y_{N1}(k)}{\partial u_1(k)} \cdot \frac{\partial u_1(k)}{\partial \omega_i^C(k)} + \frac{\partial y_{N1}(k)}{\partial u_2(k)} \cdot \frac{\partial u_2(k)}{\partial \omega_i^C(k)} \right) + \frac{\partial E^C(k)}{\partial y_{N2}(k)} \left(\frac{\partial y_{N2}(k)}{\partial u_1(k)} \cdot \frac{\partial u_1(k)}{\partial \omega_i^C(k)} + \frac{\partial y_{N2}(k)}{\partial u_2(k)} \cdot \frac{\partial u_2(k)}{\partial \omega_i^C(k)} \right) \right) =$$

$$-\alpha_2 \cdot \left(e_1^C(k) \left(J_{11}(k) \cdot \frac{\partial u_1(k)}{\partial \omega_i^C(k)} + J_{12}(k) \cdot \frac{\partial u_2(k)}{\partial \omega_i^C(k)} \right) + e_2^C(k) \left(J_{21}(k) \cdot \frac{\partial u_1(k)}{\partial \omega_i^C(k)} + J_{22}(k) \cdot \frac{\partial u_2(k)}{\partial \omega_i^C(k)} \right) \right)$$

Estabilidad

Control Neuronal

Estabilidad

➤ Estabilidad en el **aprendizaje de las RNs individuales**

- **ADALINE:** el algoritmo Widrow-Hoff converge a un mínimo global para ratios de aprendizaje que cumplan la siguiente condición:

$$0 < \alpha < 1 / \lambda_{\max}$$

- λ_{\max} mayor autovalor de la matriz de correlación de entrada **R**

- **MADALINE:** el algoritmo backpropagation es más complejo. Hay muchos mínimos locales. No está garantizado el mínimo global. Se prueban varias condiciones iniciales. Las **variaciones del backpropagation** utilizan ratios de aprendizaje variables en función del error que **garantizan la estabilidad** del aprendizaje

➤ Estabilidad **global del sistema en lazo cerrado**

- Se comporta como el modelo de referencia. Estudio de **estabilidad del modelo de referencia**.
 - Modelo lineal -> 1º criterio de Lyapunov
 - Modelo no lineal -> 2º criterio de Lyapunov

Control Neuronal

Estabilidad

➤ Estabilidad en el **aprendizaje de RNs ADALINE**

➤ Recordemos el algoritmo LMS-Widrow-Hoff o regla delta:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + 2\alpha e(k)\mathbf{z}(k) \quad \mathbf{x} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} \quad \mathbf{z} = \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}$$

$$E[\mathbf{x}(k+1)] = E[\mathbf{x}(k)] + 2\alpha E[e(k)\mathbf{z}(k)] \quad e(k) = t(k) - \mathbf{x}^T(k)\mathbf{z}(k)$$

$$E[\mathbf{x}(k+1)] = E[\mathbf{x}(k)] + 2\alpha \{E[t(k)\mathbf{z}(k)] - E[(\mathbf{x}^T(k)\mathbf{z}(k))\mathbf{z}(k)]\} \quad \mathbf{x}^T(k)\mathbf{z}(k) = \mathbf{x}(k)\mathbf{z}^T(k)$$

$$E[\mathbf{x}(k+1)] = E[\mathbf{x}(k)] + 2\alpha \{E[t(k)\mathbf{z}(k)] - E[(\mathbf{z}(k)\mathbf{z}^T(k))\mathbf{x}(k)]\} \quad \mathbf{x}(k) \text{ independiente de } \mathbf{z}(k)$$

$$E[\mathbf{x}(k+1)] = E[\mathbf{x}(k)] + 2\alpha \{\mathbf{h} - \mathbf{R} E[\mathbf{x}(k)]\} = [\mathbf{I} - 2\alpha \mathbf{R}]E[\mathbf{x}(k)] + 2\alpha \mathbf{h}$$

➤ El sistema dinámico será estable si todos los autovalores de $[\mathbf{I} - 2\alpha \mathbf{R}]$ caen dentro del círculo unidad

$$1 - 2\alpha \lambda_i > -1, \lambda_i : \text{autovalores de } \mathbf{R}$$

$$\alpha < 1/\lambda_i, \forall i \rightarrow 0 < \alpha < 1/\lambda_{\max}$$

Control Neuronal

Estabilidad aprendizaje ADALINE. Ejemplo 1

- Calcular el máximo ratio de aprendizaje para la siguiente ecuación y los siguientes datos de entrada:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha e(k)\mathbf{p}^T(k) \quad \left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, t_1 = [-1] \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = [1] \right\}$$

- Asumiendo que los vectores se generan con igual probabilidad:

$$\mathbf{R} = E[\mathbf{p}\mathbf{p}^T] = \frac{1}{2}\mathbf{p}_1\mathbf{p}_1^T + \frac{1}{2}\mathbf{p}_2\mathbf{p}_2^T = \frac{1}{2}\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}\begin{bmatrix} 1 & -1 & -1 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}\begin{bmatrix} 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

- Los autovalores de \mathbf{R} son:

$$\lambda_1 = 1.0, \quad \lambda_2 = 0.0, \quad \lambda_3 = 2.0.$$

- El máximo ratio de aprendizaje será:

$$\alpha < \frac{1}{\lambda_{max}} = \frac{1}{2.0} = 0.5$$