Tema 4. Control Neuronal

Luis Miguel Bergasa

Control Neuronal

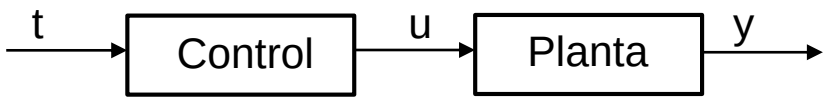
Índice

- **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**

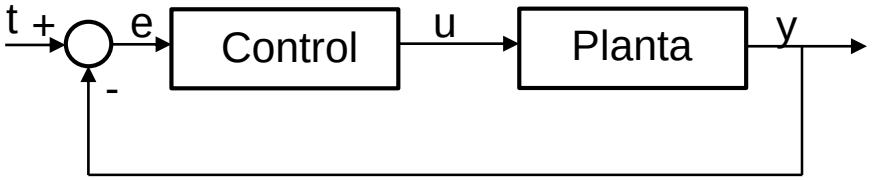
Introducción

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



Introducción

- **Teoría básica de control**
  - Control clásico
    - Todo o nada (“bang-bang”). Ej: termostado de Tª
    - 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

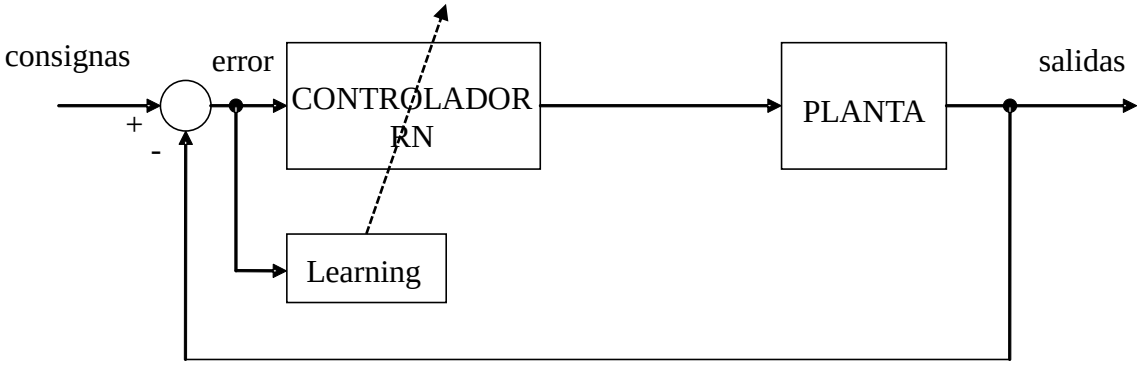
Arquitecturas de Control Neuronal

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

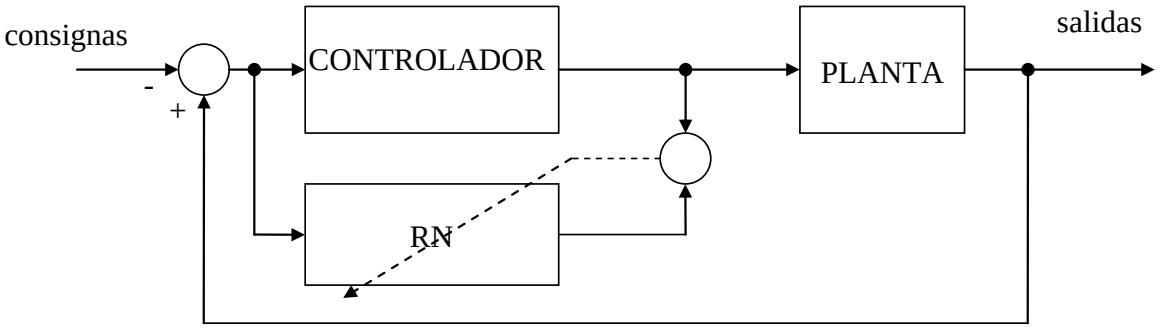
Arquitecturas de Control Neuronal

- **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 (consignas – salidas)^2



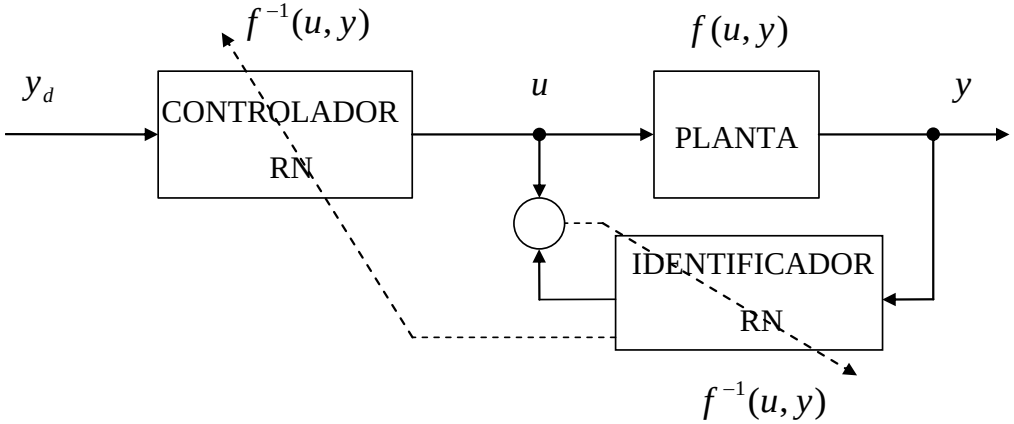
Arquitecturas de Control Neuronal

- **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



Arquitecturas de Control Neuronal

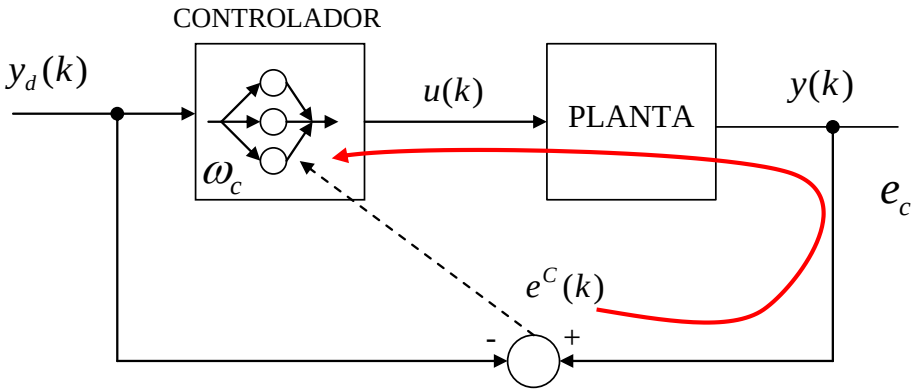
- **Control inverso**
  1. Identifica la dinámica de la planta mediante una RN
  2. Diseña un controlador neuronal a partir de la RN identificadora
- **Control inverso no orientado a minimizar el error de control**
  - RN identificadora
    - Minimiza el error cuadrático de identificación de la planta (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



$$e_c(k) = y(k) - y_d(k)$$

$$E_y(k) = \frac{1}{2} e_c{}^2(k)$$

- ¿Cómo se ajustan los pesos de la red controladora?
  - Método de descenso por el gradiente

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

J_{PLANTA} ? ← Depende de la topología de la RN

Arquitecturas de Control Neuronal

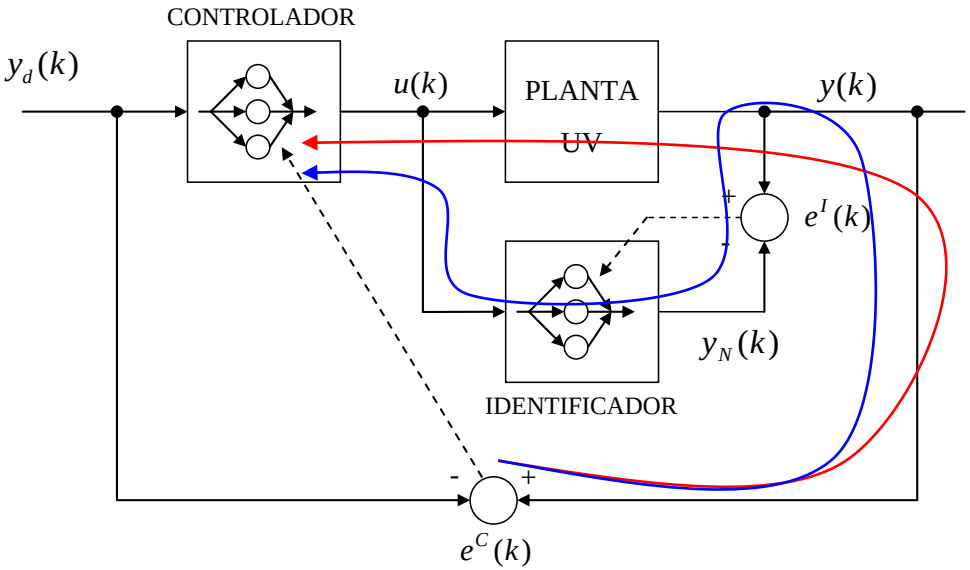
- **Control inverso orientado a minimizar el error de control**
  - Problema: El error está definido a la salida de la planta
    - Pasar el error a la salida de la red neuronal exige el conocimiento del Jacobiano de la planta:

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

  - ¿Cómo obtener el valor del Jacobiano de la planta J_{PLANTA}?
    - Utilizando una **RN identificadora**

$$y_N \approx y$$

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



Identificadores con RNs

Identificadores con Redes Neuronales

Identificadores neuronales

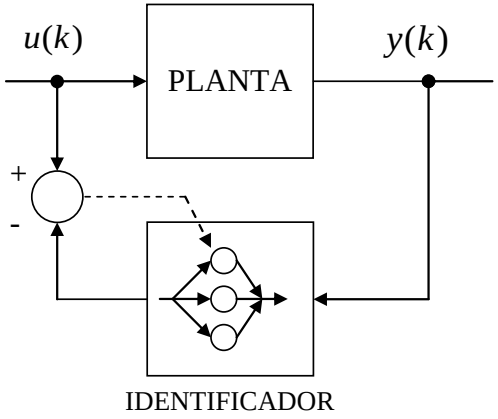
- Usar las RNs para modelar plantas desconocidas
  - Se basan en la capacidad de aprendizaje de las RNs para identificar sistemas (identificador universal)
  - Se usan un conjunto de entradas/salidas para el entrenamiento

- 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

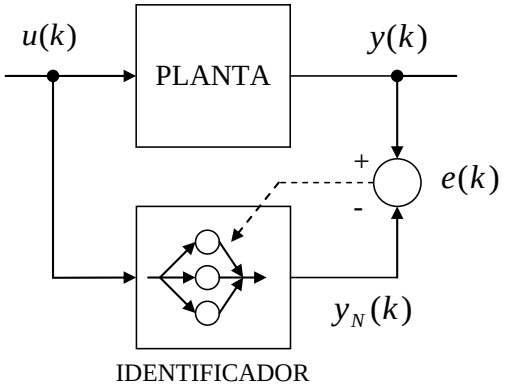
Identificadores con Redes Neuronales

Configuraciones

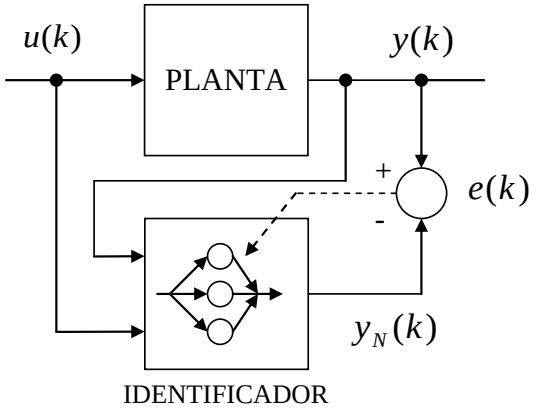
- Serie



- Paralelo

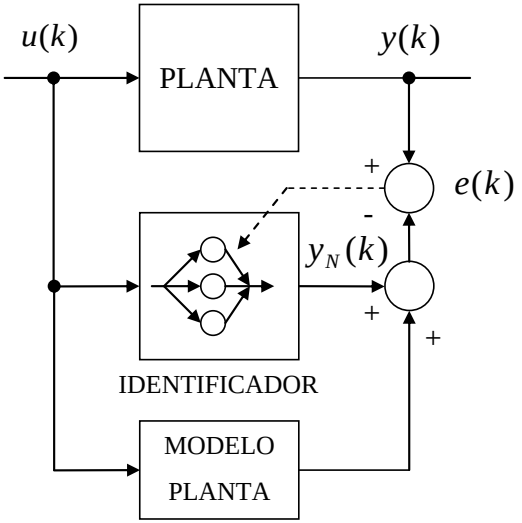


- Serie/Paralelo

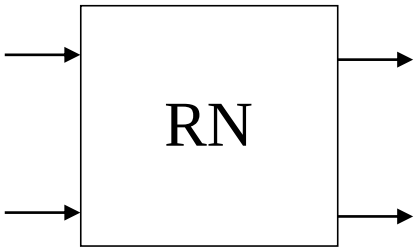


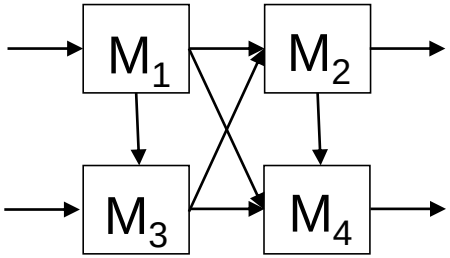
Identificadores con Redes Neuronales

- **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





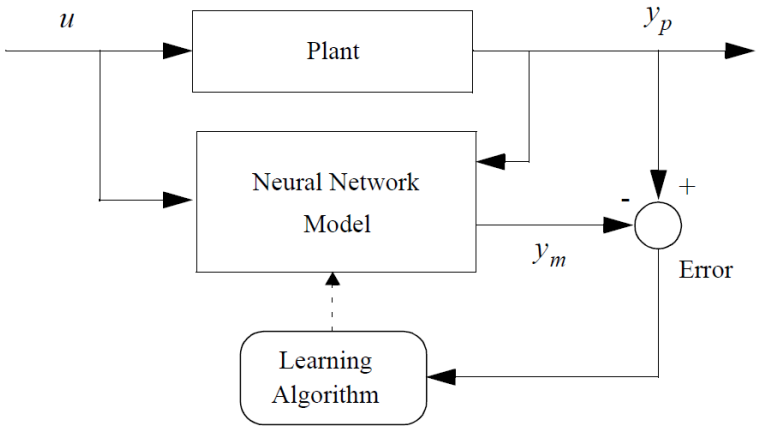
Control de sistemas SISO con RNs

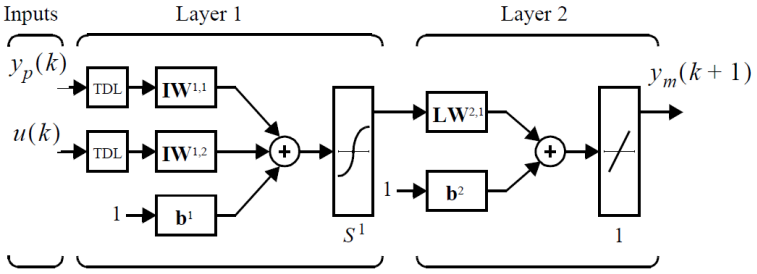
Control de sistemas SISO mediante RNs

- **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 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)

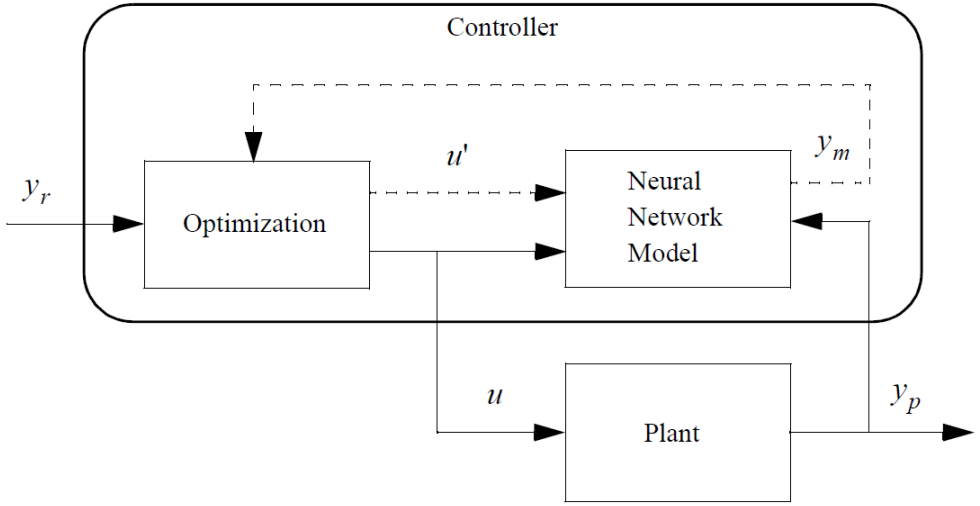
$$y_p(k+d) = h\left[y_p(k), y_p(k-1),..., y_p(k-n+1), u(k), u(k-1),..., u(k-m+1)\right]$$

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

    - Entrenamiento usando backpropagation estático (no hay realimentación)

Control de sistemas SISO mediante RNs

- **Controlador predictivo**
  - Diseño del controlador (on-line)



N_1,N_2,N_u: horizontes sobre los que se evalúa el control
u' : señal de control tentativa
y_r : respuesta deseada
y_m : respuesta de la RN
ρ : contribución del control incremental

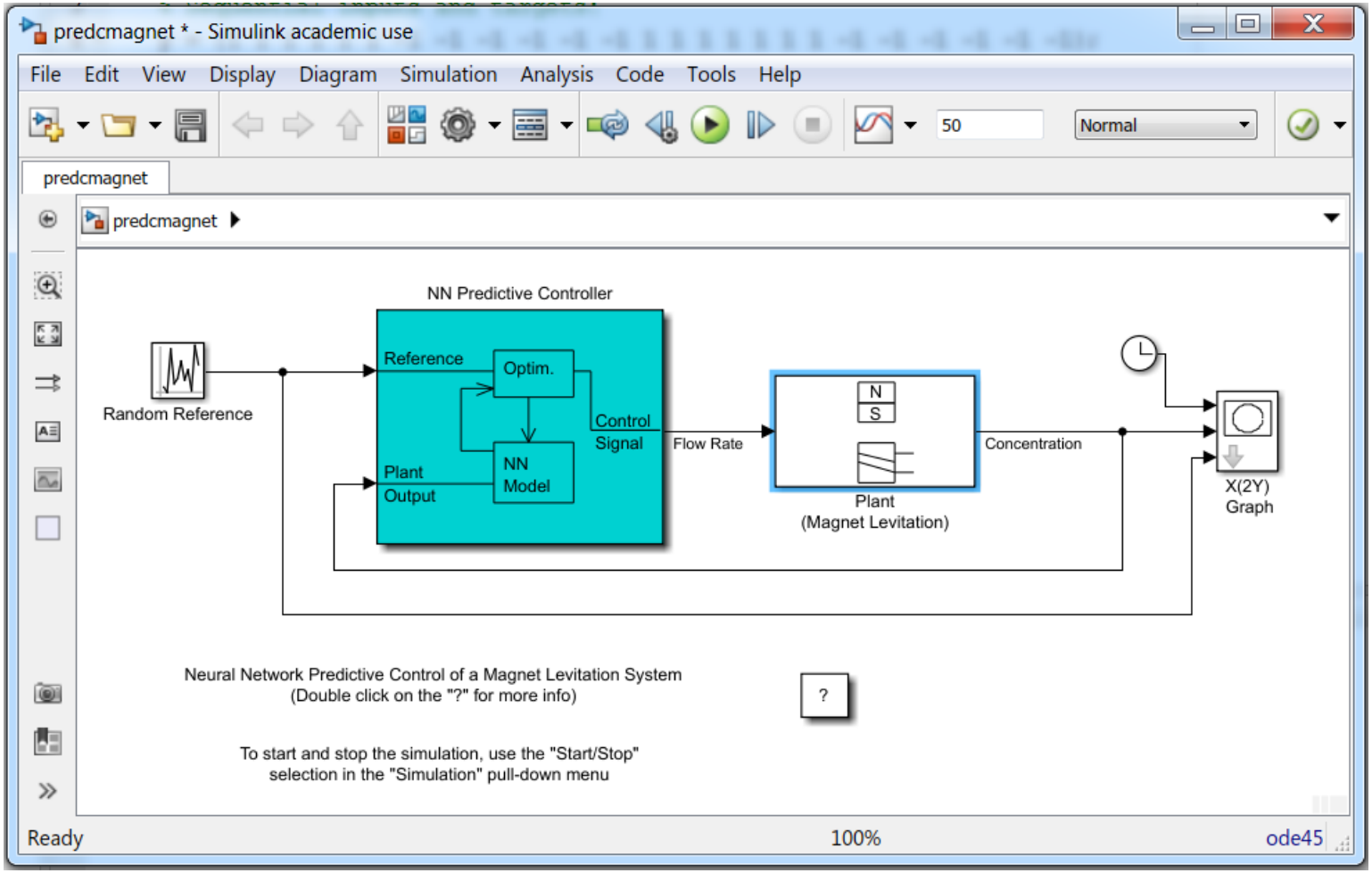
  - Predicciones de la RN se usan por un módulo de optimización para minimizar J

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

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

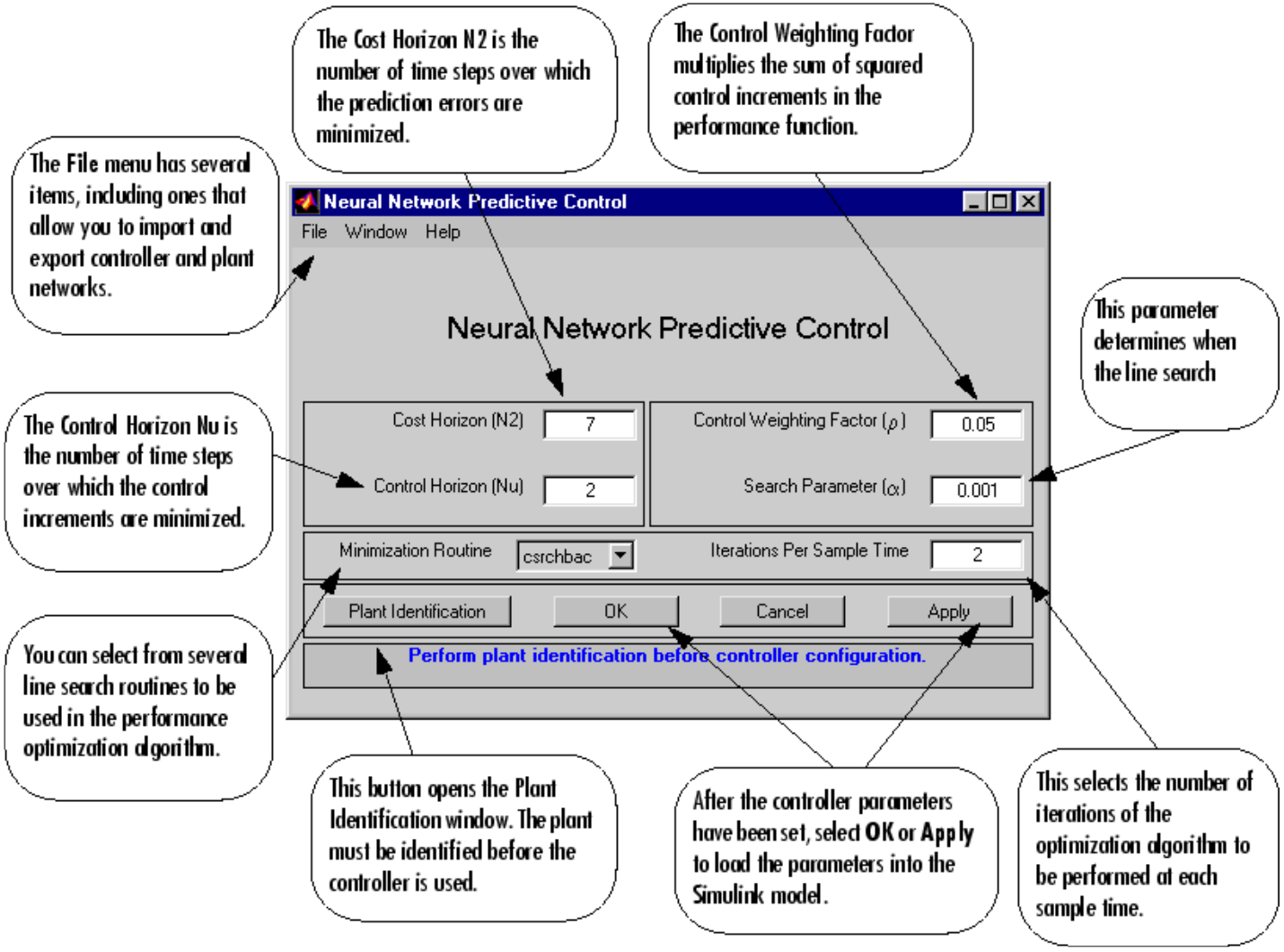
Control Neuronal

Controlador predictivo. MATLAB



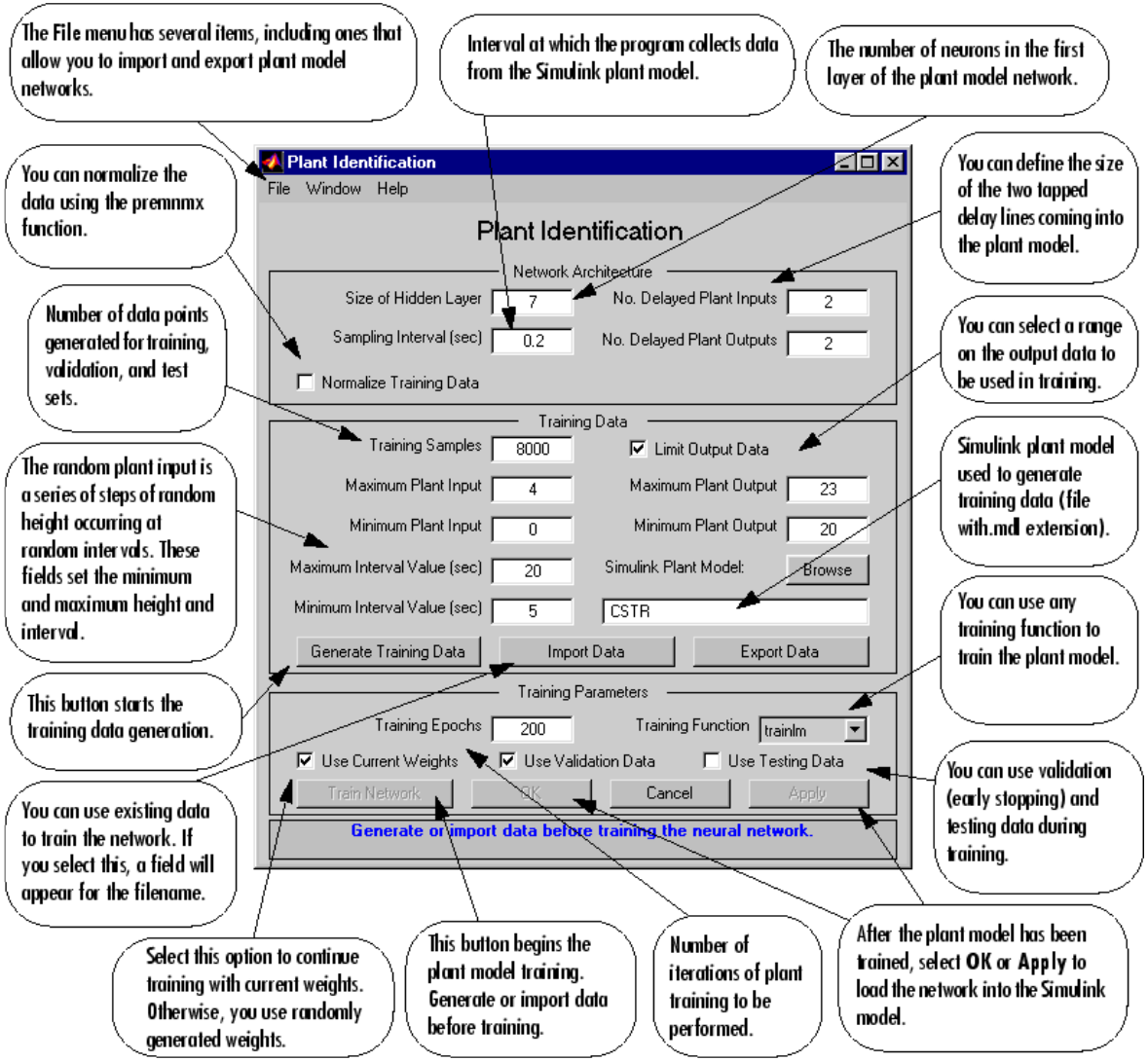
Control Neuronal

Controlador predictivo. MATLAB



Control Neuronal

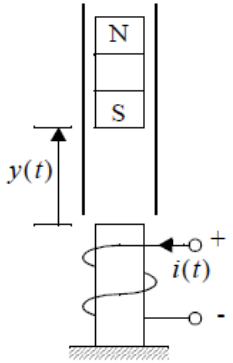
Controlador predictivo. MATLAB



Control Neuronal

Controlador predictivo. Ejemplo

- Sistema de levitación magnético



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

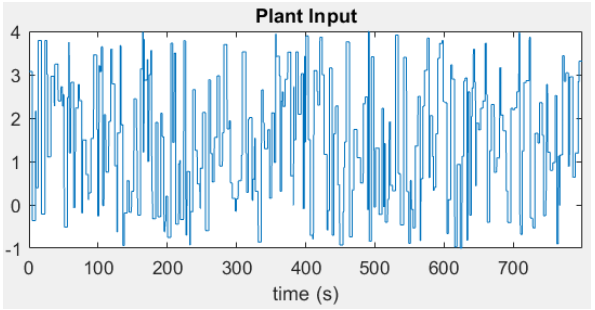
M: masa del imán	M=3
g: fuerza de la gravedad	g=9.8
β: coeficiente de fricción viscosa	β=12
α: coeficiente de fuerza de campo	α=15

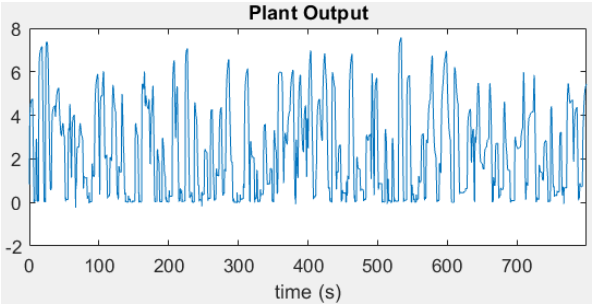
- Identificación de la planta
  - inp_{del}=3, out_{del}=2, $y_m(k+1) = \hat{h}[y_p(k), y_p(k-1), y_p(k-2), i(k), i(k-1), i(k-2), i(k-3); \mathbf{x}]$
  - 9 neuronas en la capa oculta, $f_s = 0.1\ s$
  - Generación de datos de entrenamiento
    - *Training samples: 8000*
    - *Plant input: [-1, 4]*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

Control Neuronal

Controlador predictivo. Ejemplo

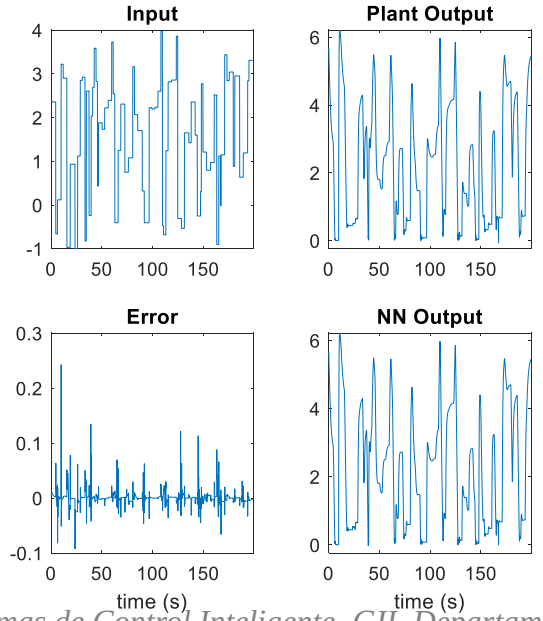
- Identificación de la planta

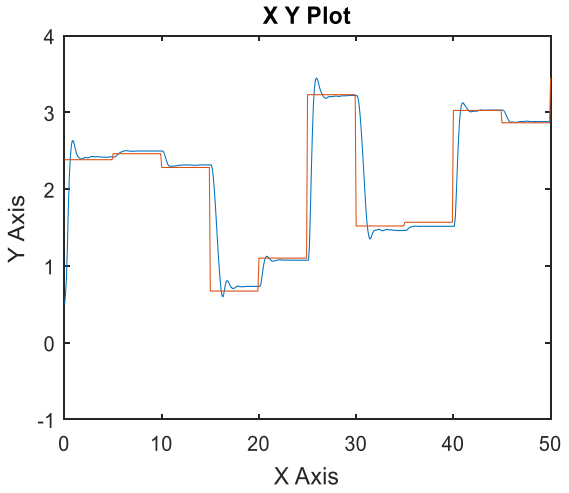




- Diseño del controlador

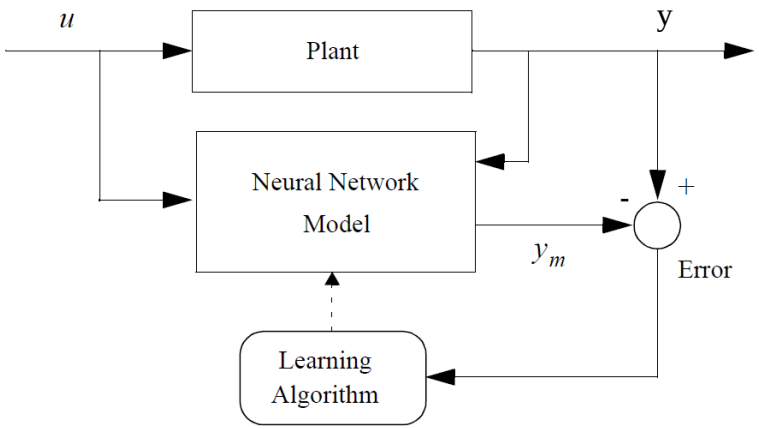
$N_1 = 1, N_2 = 7$
$N_u = 3, \rho = 0.01$

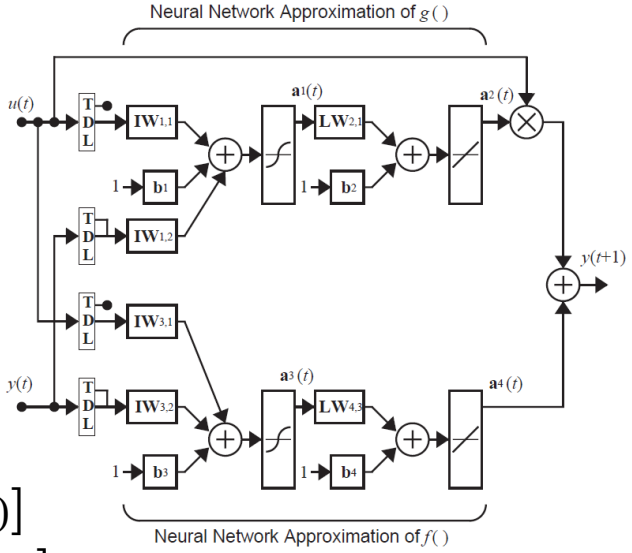




Control de sistemas SISO mediante RNs

- **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)**





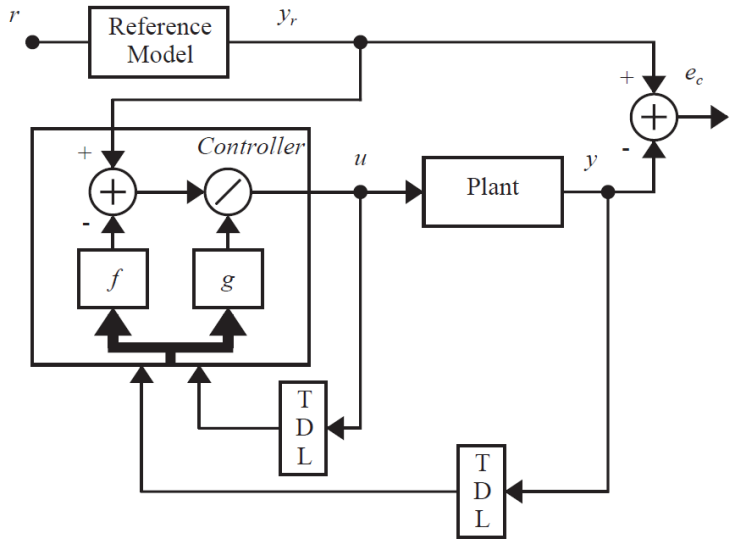
$$\hat{y}(k+d) = f\left[y(k), y(k-1),..., y(k-n+1), u(k-1),..., u(k-m+1)\right] + g\left[y(k), y(k-1),..., y(k-n+1), u(k-1),..., u(k-m+1)\right]u(k)$$

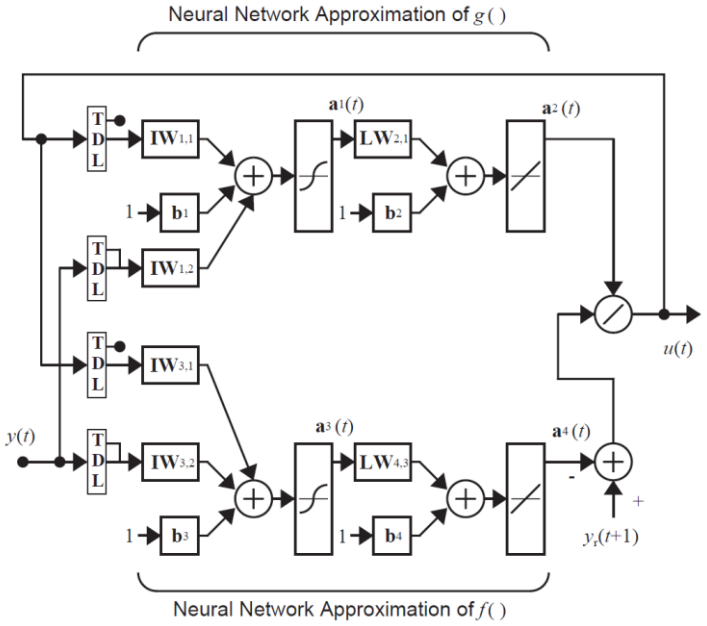
Control de sistemas SISO mediante RNs

- **Controlador NARMA-L2**
  - **Diseño del controlador**
    - Considerando $y(k+d)=y_r(k+d)$

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

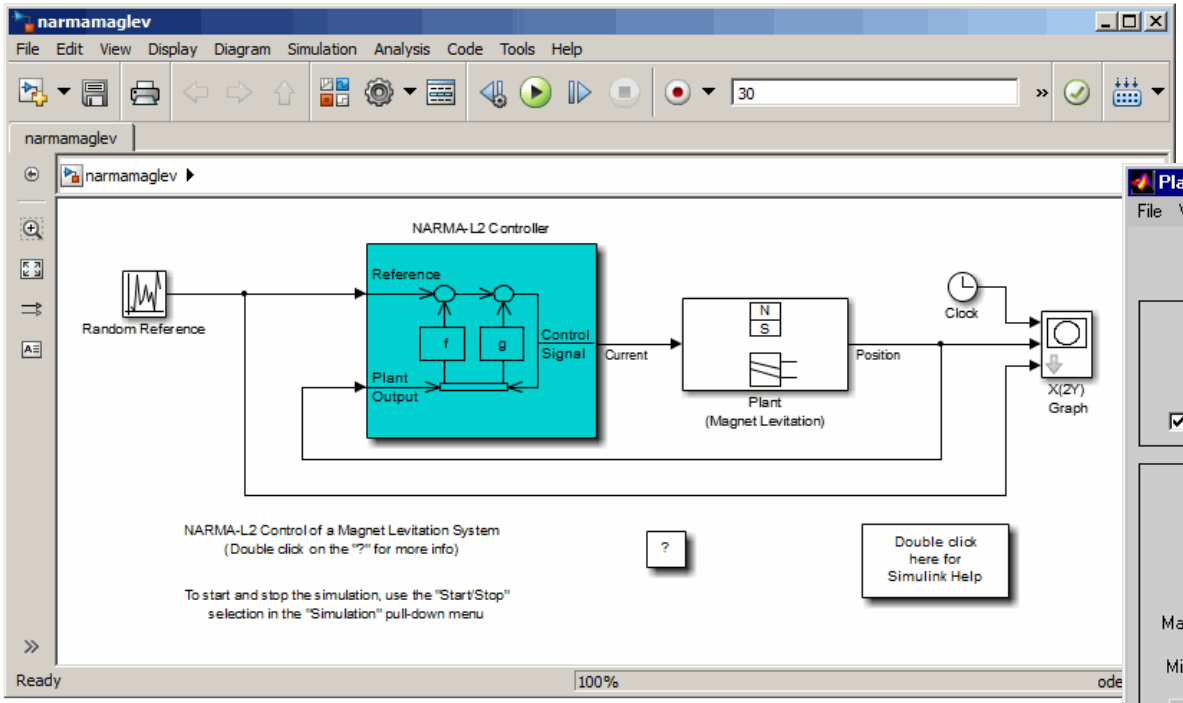
    - Entrenamiento usando backpropagation estático (no hay realimentación)

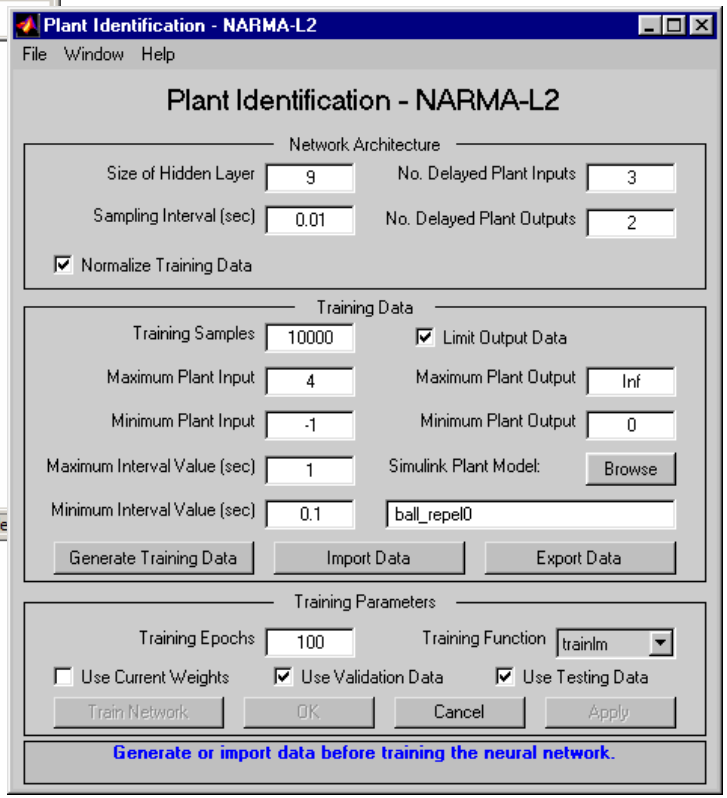




Control Neuronal

Control SISO NARMA-L2. MATLAB

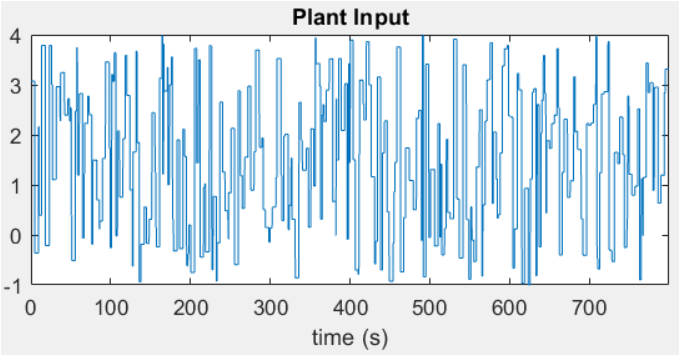


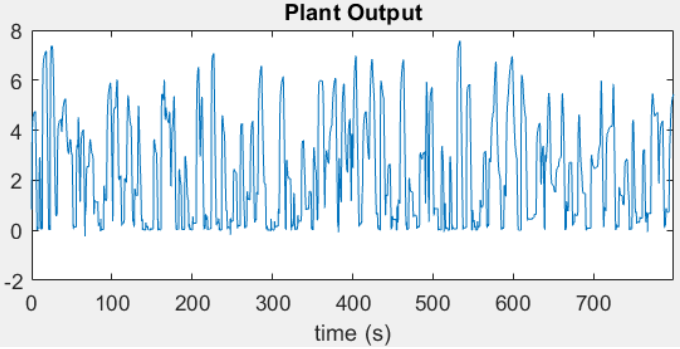


Control Neuronal

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, $f_s = 0.1\ s$
    - 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*

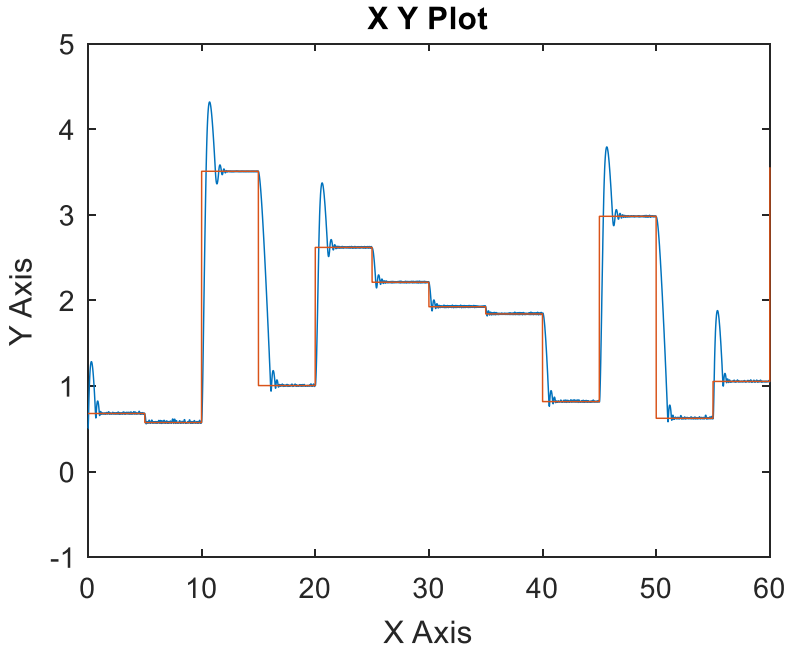


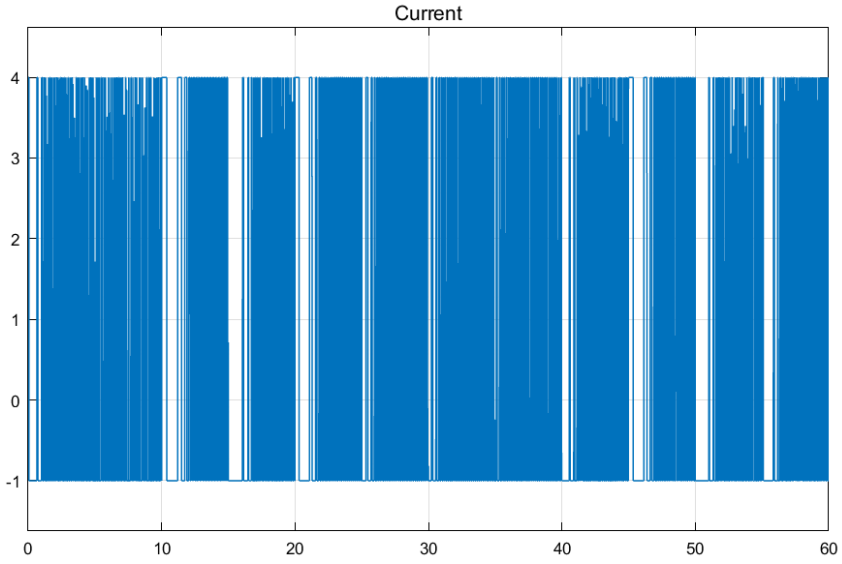


Control Neuronal

Control SISO NARMA-L2. Ejemplo

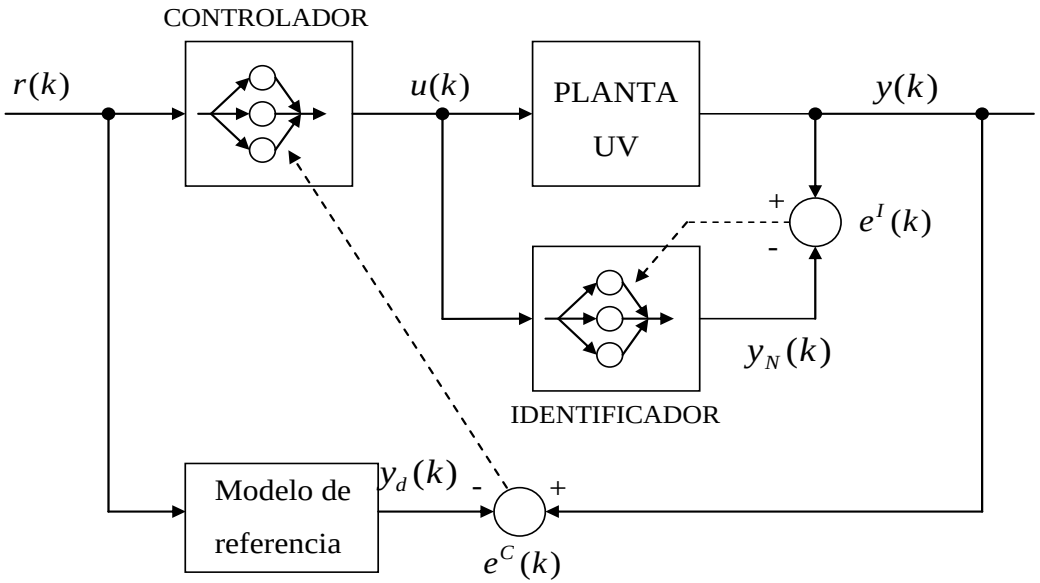
- Diseño del controlador
  - La salida sigue la señal de referencia de forma precisa
  - NARMA-L2 produce más oscilaciones en la señal de control que los otros controladores
  - Estas oscilaciones se pueden reducir mediante un proceso de filtrado





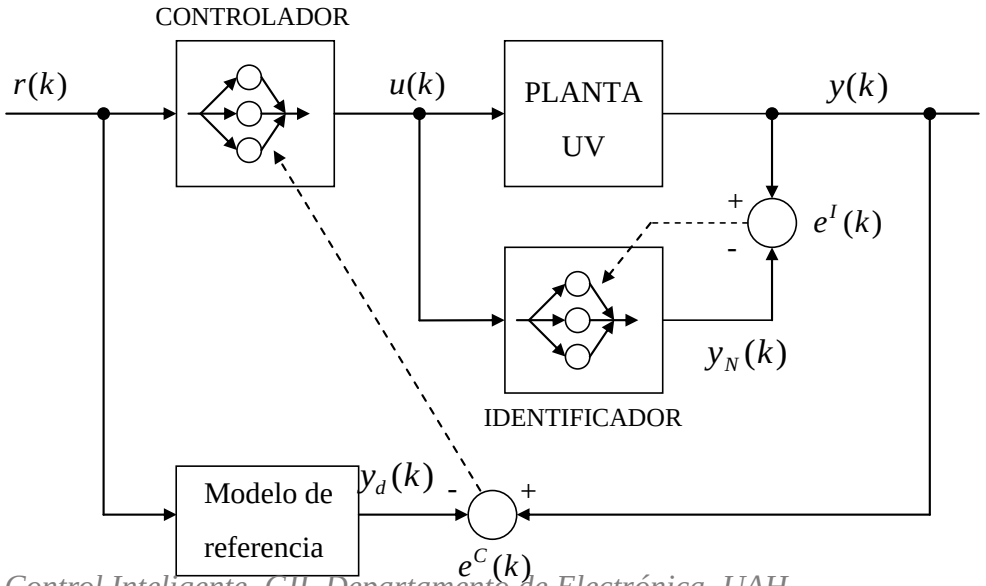
Control de sistemas SISO mediante RNs

- **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 $e^C(k)$



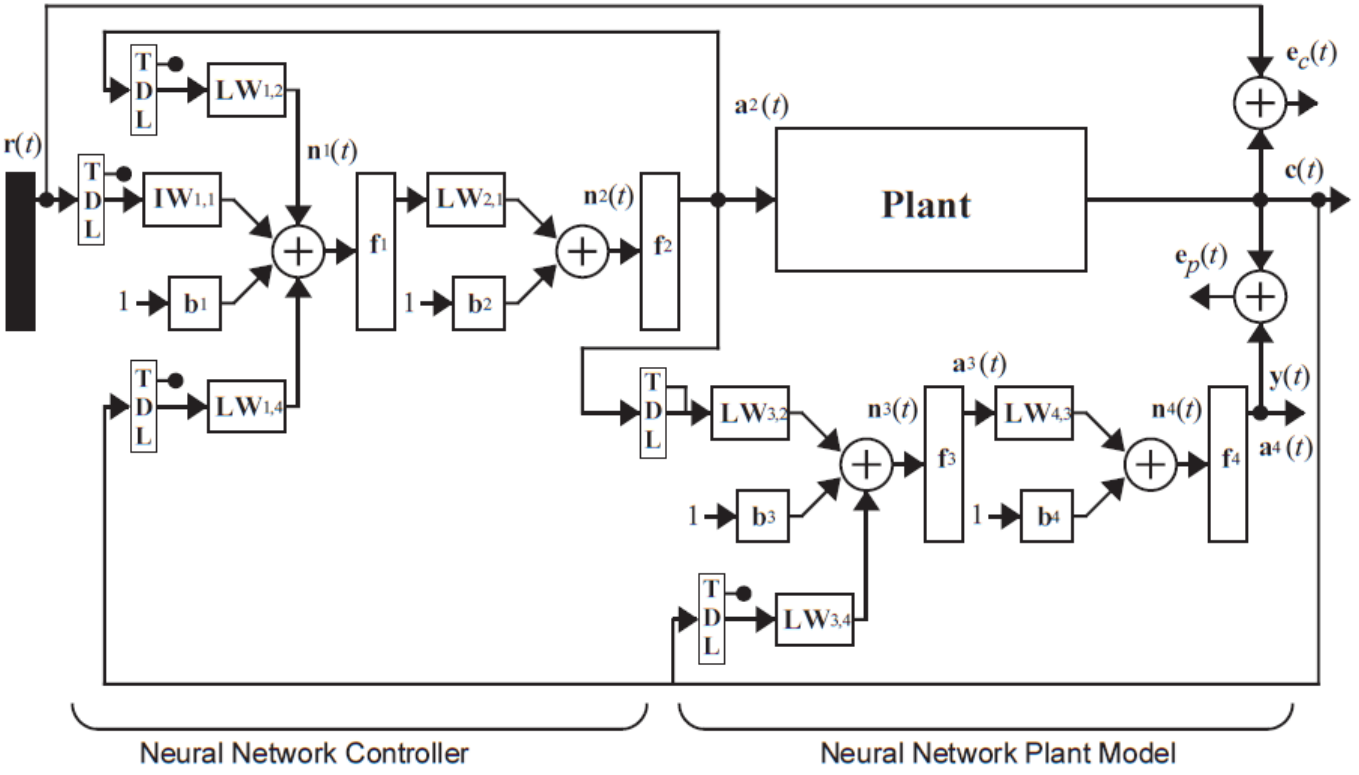
Control de sistemas SISO mediante RNs

- **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



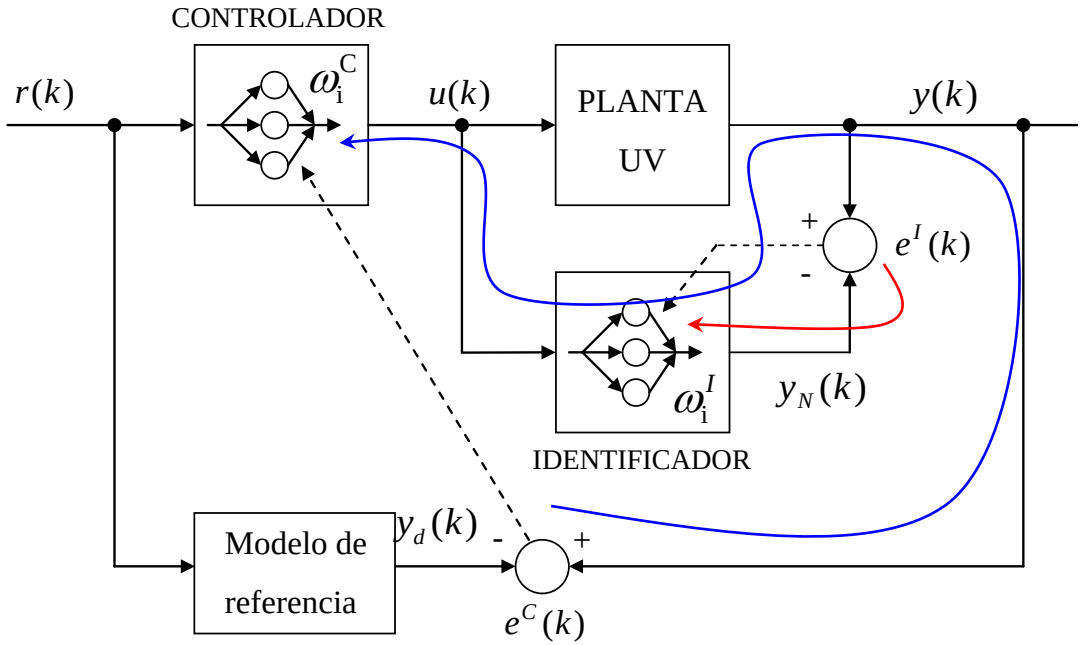
Control de sistemas SISO mediante RNs

- **Con modelo de referencia**
  - **Modelo MATLAB**
    - Red recurrente (realimentada). Entrenamiento usando backpropagation dinámica



Control de sistemas SISO mediante RNs

- **Con mod. de referencia**



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

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

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

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

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

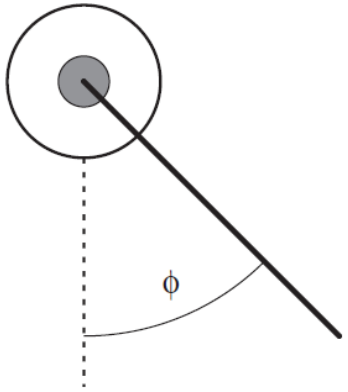
Depende de la topología de la RN

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

Control Neuronal

Control con modelo de referencia. Ejemplo

- Control de un brazo robot

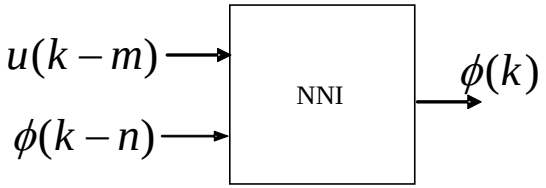


$$Modelo\ planta \rightarrow \frac{d^2\phi}{dt^2} = -10\sin\phi - 2\frac{d\phi}{dt} + u$$

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

Φ: ángulo que forma el brazo con la vertical
u: torsión aplicada por el motor DC
r: señal de referencia
yr: salida del modelo de referencia

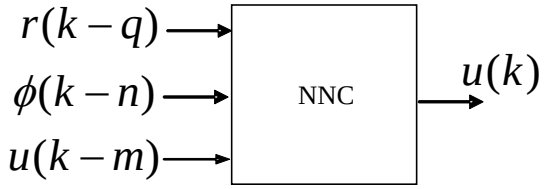
- Identificación de la planta
  - Intervalo de muestreo: 0.05
  - Anchura de pulsos de 0.1 a 2 s
  - Amplitud de pulsos de -15 a 15 N-m
  - Retardadores de la entrada: (m=0,1,2)
  - Retardadores de la salida: (n=1,2)
  - Topología de la red: 5-10-1

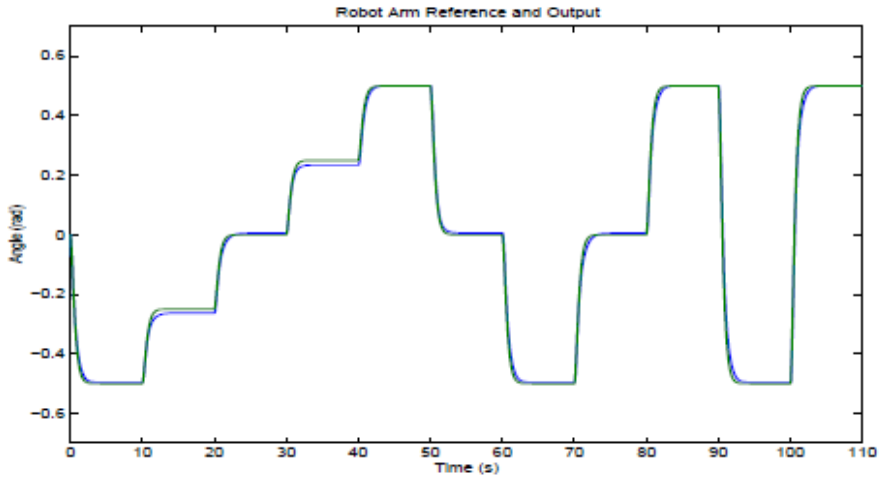


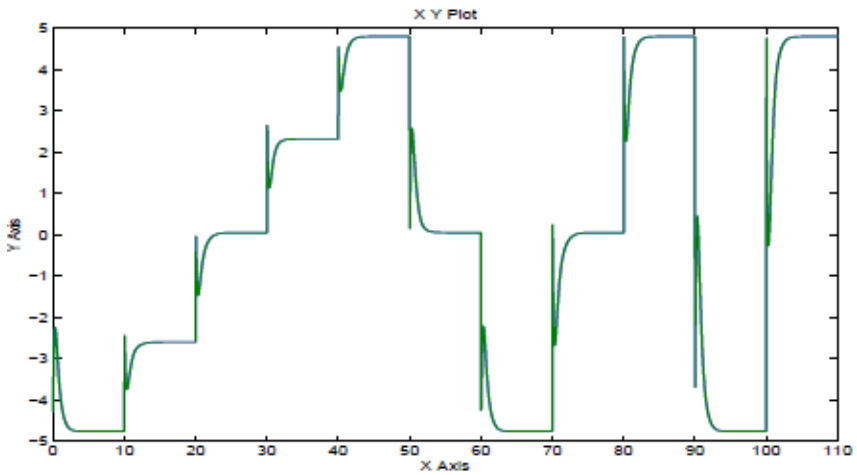
Control Neuronal

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.

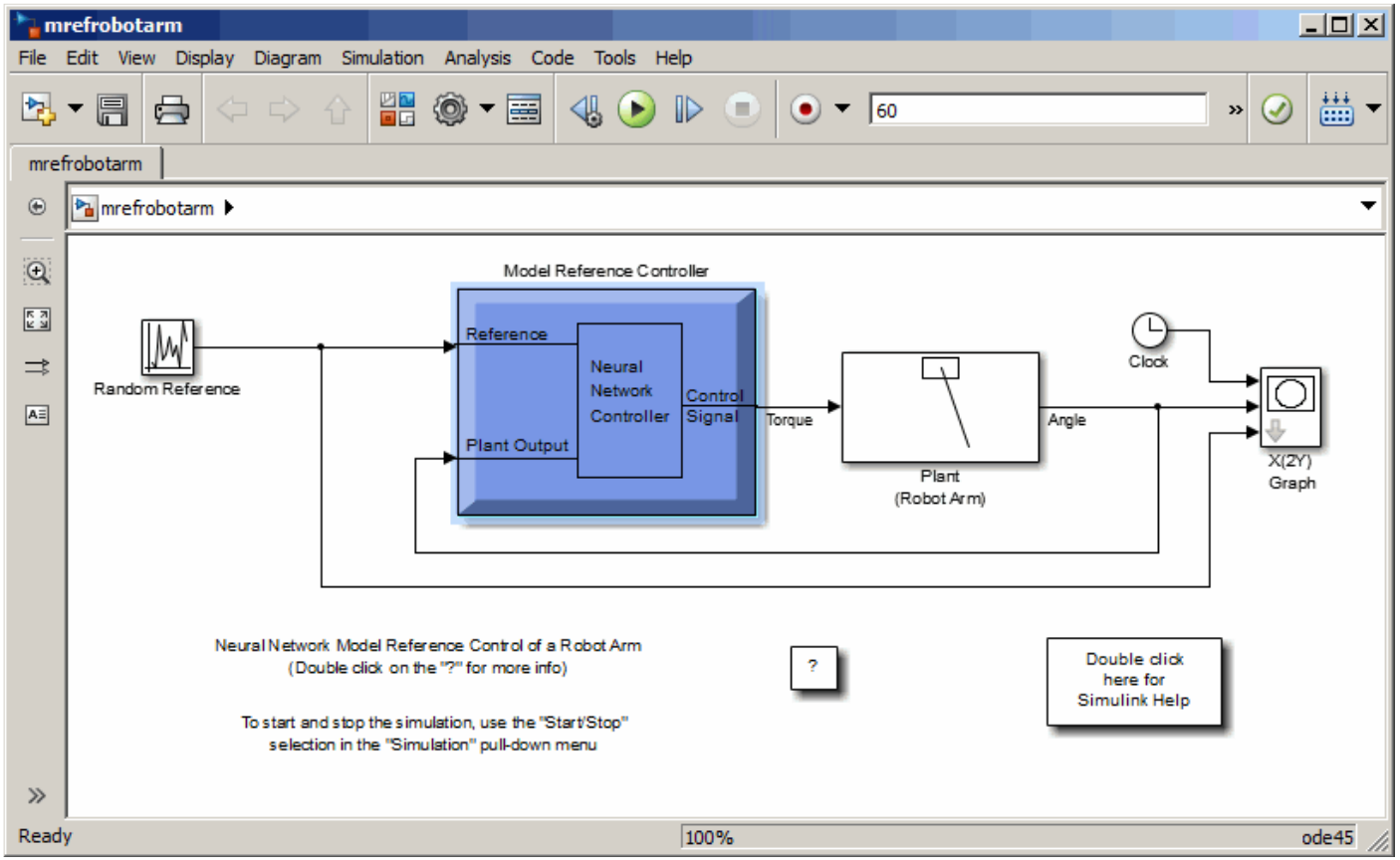






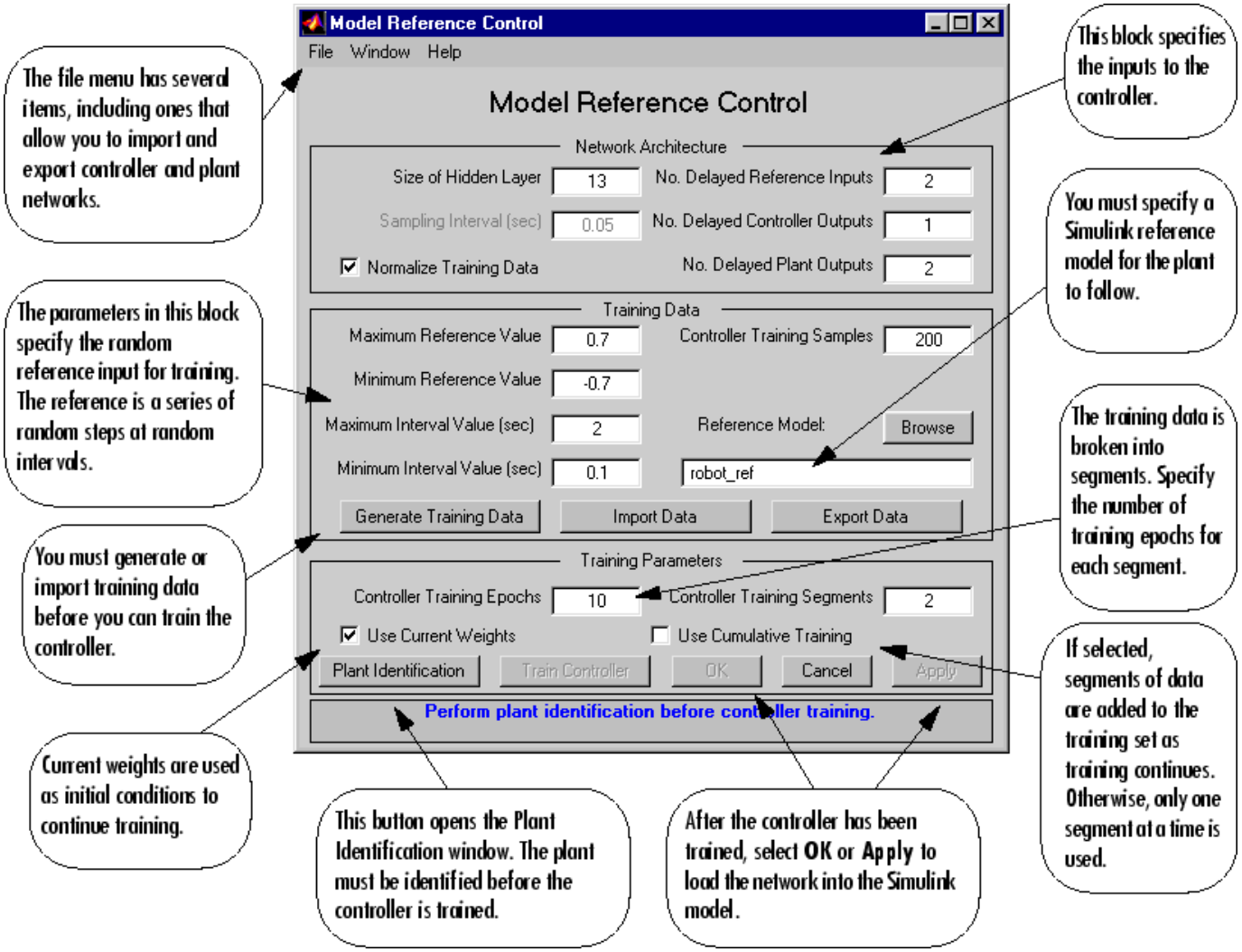
Control Neuronal

Control con modelo de referencia. Ejemplo



Control Neuronal

Control con modelo de referencia. Ejemplo



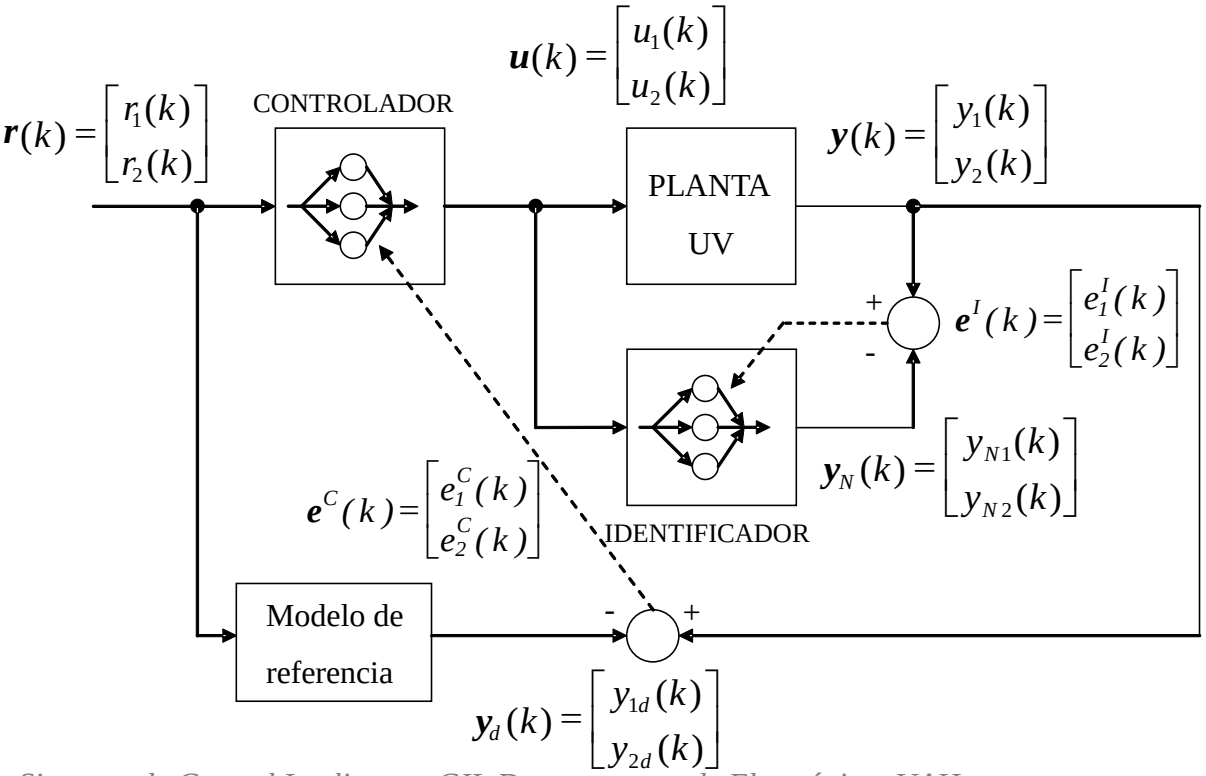
Control de sistemas MIMO con RNs

Control de sistemas MIMO mediante RNs

- **Con modelo de referencia**
  - Arquitectura

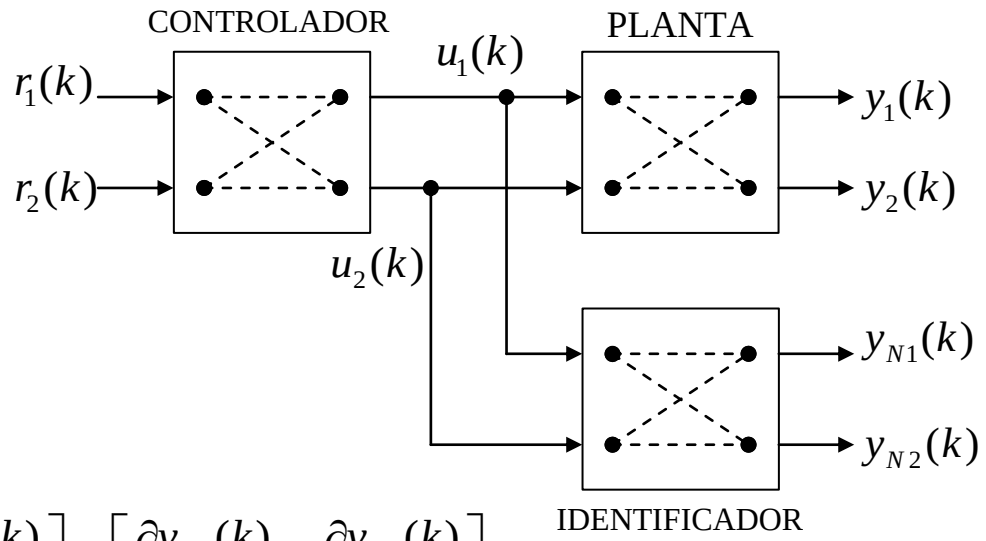
$$E^I(k)=\frac{1}{2}\left(\mathbf{e}^I(k)\right)^T\left(\mathbf{e}^I(k)\right)=\frac{1}{2}\left(y_1(k)-y_{N1}(k)\right)^2+\frac{1}{2}\left(y_2(k)-y_{N2}(k)\right)^2=\frac{1}{2}\left(e_1^I(k)\right)^2+\frac{1}{2}\left(e_2^I(k)\right)^2$$

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



Control de sistemas MIMO mediante RNs

- **Con modelo de referencia**
  - Arquitectura



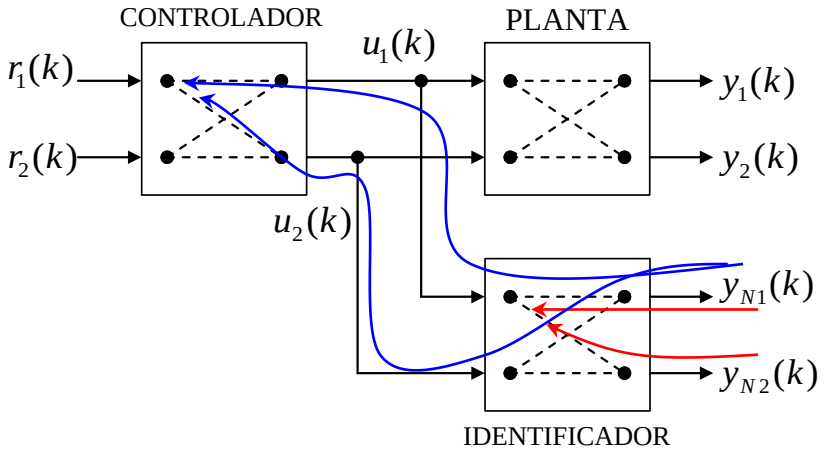
$$J(k)=\begin{bmatrix} J_{11}(k) & J_{12}(k) \\ J_{21}(k) & J_{22}(k) \end{bmatrix}=\begin{bmatrix} \frac{\partial y_1(k)}{\partial u_1(k)} & \frac{\partial y_1(k)}{\partial u_2(k)} \\ \frac{\partial y_2(k)}{\partial u_1(k)} & \frac{\partial y_2(k)}{\partial u_2(k)} \end{bmatrix}\approx\begin{bmatrix}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

Aproximación

$$y_{N1}(k)\approx y_1(k)$$
$$y_{N2}(k)\approx y_2(k)$$

Control de sistemas MIMO mediante RNs

- **Con modelo de referencia**
  - Ajuste de pesos
    - Dos ratios de aprendizaje distintos α_1 (identificador) y α_2(controlador)



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

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

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

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

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

Estabilidad

- Estabilidad en el **aprendizaje de RNs ADALINE**
  - Recordemos el algoritmo LMS-Widrow-Hoff o regla delta:

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

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

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

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

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

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

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

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

Estabilidad aprendizaje ADALINE. Ejemplo 1

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

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha e(k)\mathbf{p}^T(k)$$

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\-1\\-1\end{bmatrix}, t_1 = \begin{bmatrix}-1\end{bmatrix}\right\} \qquad \left\{\mathbf{p}_2 = \begin{bmatrix}1\\1\\-1\end{bmatrix}, t_2 = \begin{bmatrix}1\end{bmatrix}\right\}$$

- 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 **R** son:

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

- El máximo ratio de aprendizaje será:

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