

**DISEÑO DE ESTRATEGIAS DE CONTROL
NEURONAL Y SU APLICACIÓN AL CONTROL
DE UN SISTEMA DE NIVEL DE LÍQUIDO**

MARTÍN ALAYON

TESIS DE INGENIERÍA

**FACULTAD DE INGENIERIA
UNIVERSIDAD DE BUENOS AIRES**

(2004)

CONTROL NEURONAL POR MODELO INTERNO

Dos redes son utilizadas en este esquema de control directo. Ambas son entrenadas off-line utilizando datos empíricos de la planta. Una es un modelo neuronal directo de la planta y la otra es el modelo neuronal inverso. El sistema en cascada de estas dos redes, modelo inverso-modelo directo, equivale a un sistema identidad. Entonces, la señal que se presente a la entrada del modelo inverso debe ser repicada en la salida del modelo directo. Si los modelos son buenos, la salida del modelo directo y la salida de la planta deben ser similares y el sistema en cascada Modelo inverso-Planta también deberá ser similar al sistema identidad.

La realimentación del error definido por

$$er(k) = Yp(k) - Ym(k) \quad (3.1)$$

es necesaria para compensar las diferencias entre planta y modelo o simplemente perturbaciones del sistema.

El filtro digital F hace más flexible el diseño del controlador dando la opción de diseñar la respuesta de la planta a la referencia. El sistema controlador-planta responde a la referencia como lo haría el filtro F. Al elegir $F=1$ se tendrá un dead-beat controller (NØrgaard, et al., 2001).

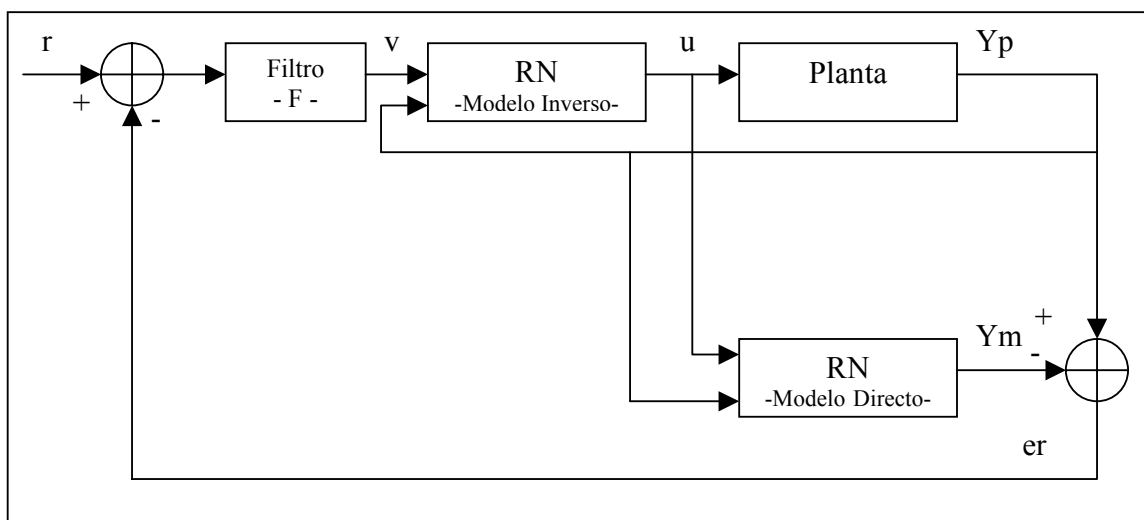


Figura 3-5: Esquema del control por modelo interno (IMC)

Modelo neuronal directo y modelo neuronal inverso

El **modelo neuronal directo** es un modelo NNARX construido con un perceptron de una sola capa. La señal de salida de este modelo es la predicción a un paso de la salida de la planta y es función de sus autoregresores.

$$Y_m(k) = g(Y_p(k-1), \dots, Y_p(k-na), u(k-1), \dots, u(k-nb)) \quad (3.2)$$

Para la construcción de este modelo se debe seleccionar una estructura, es decir los parámetros na y nb , y realizar el entrenamiento off-line a partir de datos empíricos para estimar los parámetros del modelo. El procedimiento para realizar esta tarea se detalla en el capítulo 2.

El **modelo neuronal inverso** es también un modelo NNARX[P3] construido con un perceptron de una sola capa. La salida de esta red es la fuerza de control y es función de los siguientes autoregresores:

$$u(k) = f(v(k), Y_p(k), \dots, Y_p(k-na), u(k-1), \dots, u(k-nb)) \quad (3.3)$$

Donde $v(k)$ es la salida del filtro F .

DISEÑO DEL CONTROLADOR

Los pasos a seguir para el diseño de un controlador por modelo interno se presentan en la figura 3-6. El primero consiste en obtener datos entrada salida de la planta a controlar. Es importante que este conjunto de datos posea la información acerca de la dinámica de la planta en todo el rango de operación para el cual se desea diseñar el controlador. En los siguientes dos pasos se utiliza estos datos para el entrenamiento de un modelo neuronal directo y de un modelo neuronal inverso. La selección de estructura, el entrenamiento y la validación de estos modelos se realiza siguiendo los procedimientos detallados en el capítulo 2.

El diseño del filtro digital F se realiza a partir de las especificaciones que el sistema debe cumplir. Si los modelos directo e inverso son buenos el sistema controlador-planta se comportará como lo haría el filtro.

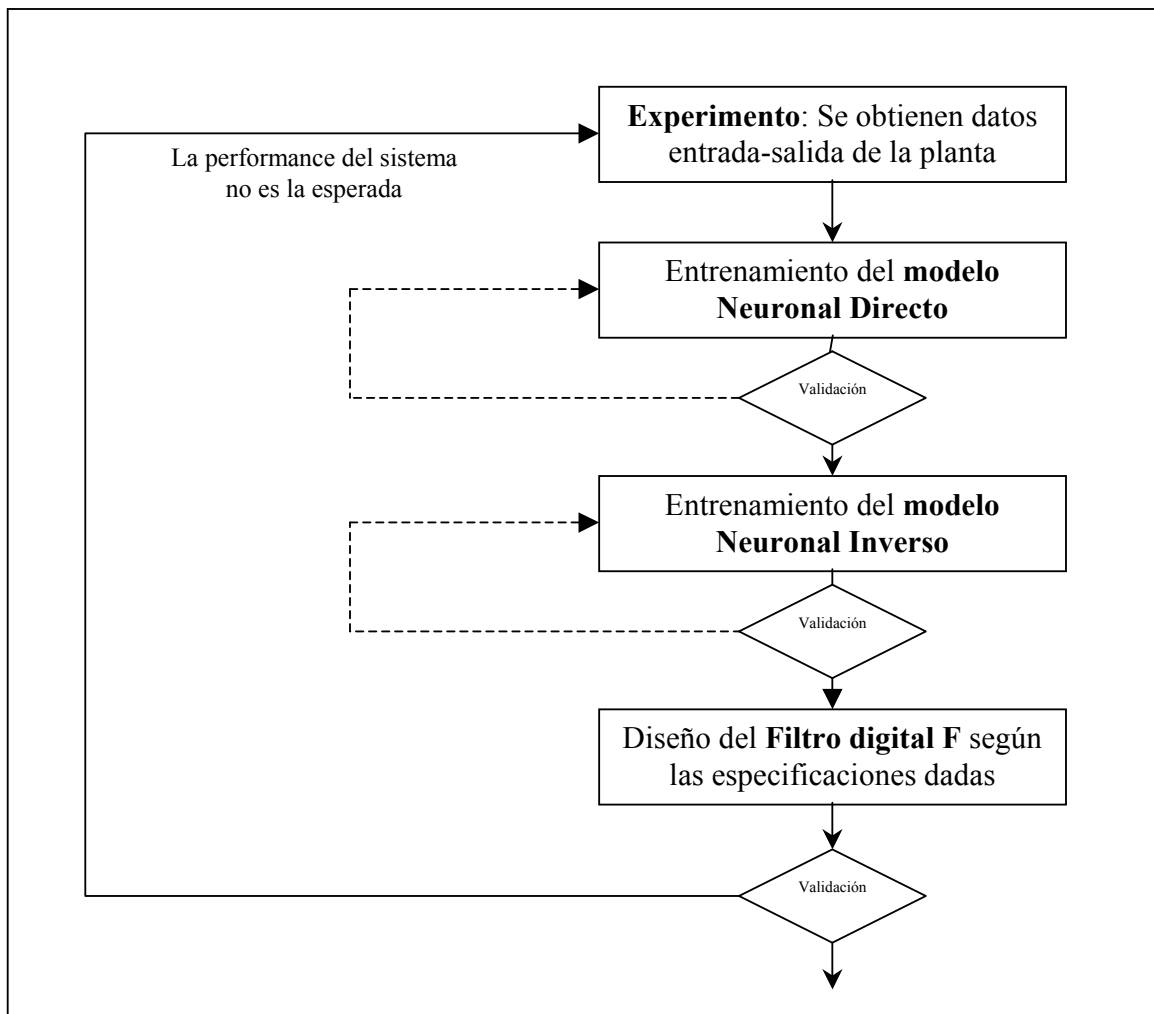


Figura 3-6: Diagrama de flujo para el diseño de un controlador por modelo interno

CONTROL NEURONAL POR MODELO DE REFERENCIA

Para este esquema de control de diseño directo se entrena de un red neuronal de modo tal que el sistema en cascada Red Neuronal-Planta se comporte como un sistema de referencia. Este debe ser elegido teniendo en cuenta las especificaciones que se deban cumplir. En la figura 3-1 se observa el esquema básico. Los parámetros del controlador neuronal son modificados en función de la diferencia entre la salida deseada y la salida actual de la planta. Durante el entrenamiento del controlador una función costo de esta diferencia, er , es minimizada. Este error viene dado por:

$$er(n) = Y_r(n) - Y_p(n) \quad (3.4)$$

La función costo es el error cuadrático:

$$J(n) = |er(n)|^2 \quad (3.5)$$

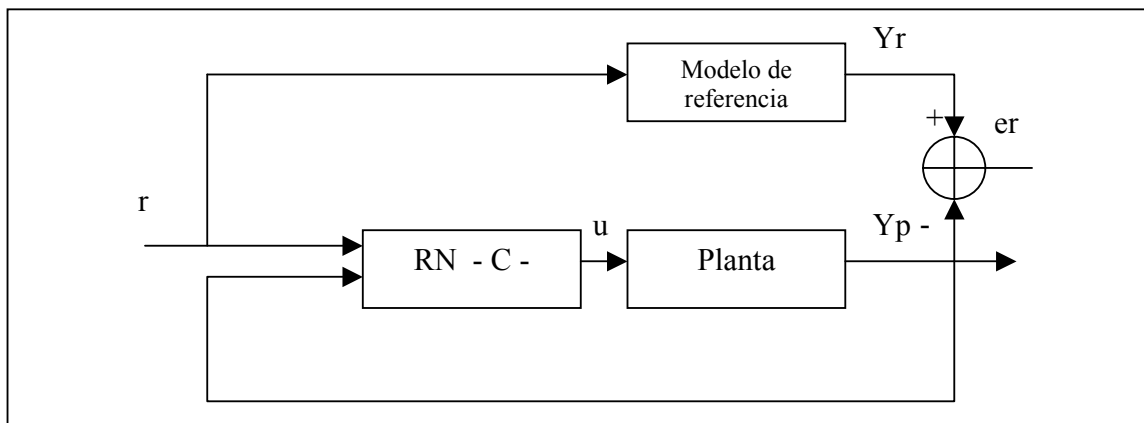


Figura 3-1: Esquema del control por modelo de referencia

Se propone estimar los parámetros θ del controlador utilizando el algoritmo Back-Propagation. Para esto es necesario computar el error de la fuerza de control, eu , a partir de el error de la salida, er . Dado que no se posee necesariamente el modelo fenomenológico de la planta, se utiliza una red neuronal entrenada off-line con a partir de datos entrada salida de la planta. Luego, se retropropaga el error, er , a través del modelo neuronal de la planta hasta obtener el error de la fuerza de control, eu . Con este error “virtual”, como lo denomina Nøgaard, se implementa el algoritmo Back-Propagation y se entrena al controlador.

Entonces, el esquema se modifica para el proceso de entrenamiento. En la figura 3-2 se observa que el error es retropropagado por el modelo neuronal de la planta sin modificar sus parámetros. Se obtiene el error de la fuerza de control, eu , que luego es retropropagado por el controlador neuronal modificando los pesos sinápticos.

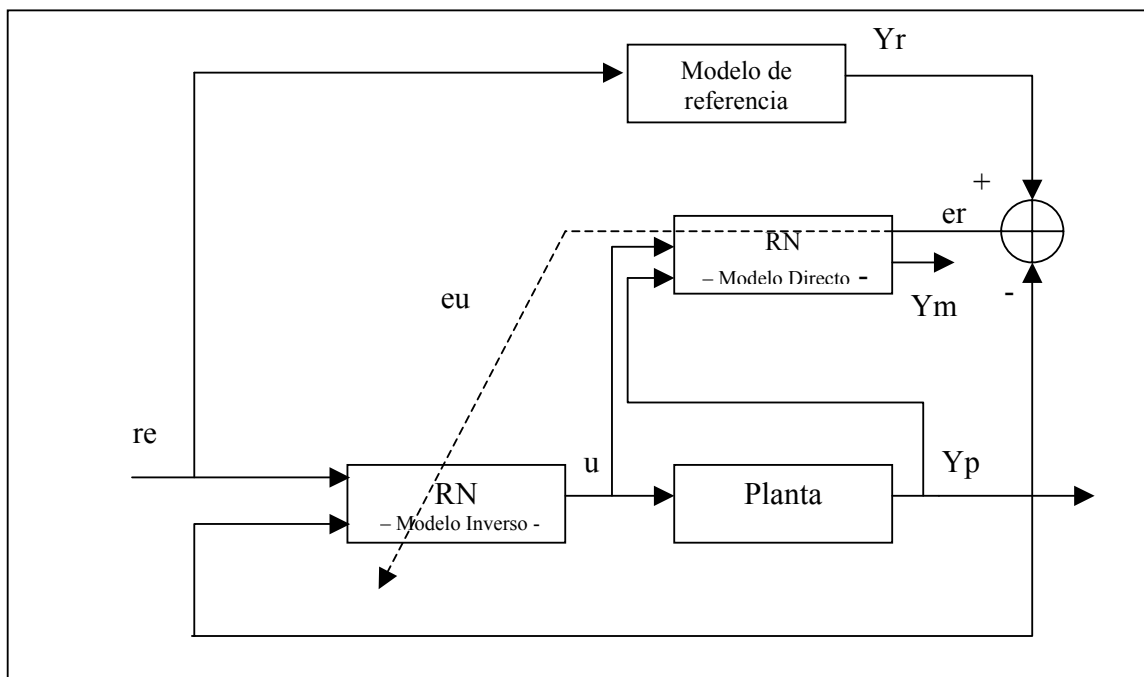


Figura 3-2: Esquema para el entrenamiento de un controlador por modelo de referencia

Modelo neuronal directo y modelo neuronal inverso

El **modelo neuronal directo** es un modelo NNARX construido con un perceptron de una sola capa. La señal de salida de este modelo es la predicción a un paso de la salida de la planta y es función de sus autoregresores.

$$Y_m(k) = g(Y_p(k-1), \dots, Y_p(k-na), u(k-1), \dots, u(k-nb)) \quad (3.6)$$

Para la construcción de este modelo se debe seleccionar una estructura, es decir los parámetros na y nb , y realizar el entrenamiento off-line a partir de datos empíricos. El procedimiento para realizar esto se detalla en el capítulo 2.

El objeto de obtener un modelo neuronal directo de la planta es poder obtener a partir de la señal error er la señal error eu (Ver figura 3-2). La señal error er se define de la siguiente manera:

$$er(k) = Y_r(k) - Y_p(k) \quad (3.7)$$

mientras que la señal eu está dada por:

$$eu(k-1) = ud(k-1) - u(k-1) \quad (3.8)$$

siendo $ud(k-1)$ la fuerza de control que reemplazada en la ecuación 3-3 hubiese dado como predicción $er(k) + Y_m(k)$.

$$er(k) + Y_m(k) = g(Y_p(k-1), \dots, Y_p(k-na), \mathbf{ud(k-1)}, u(k-2), \dots, u(k-nb)) \quad (3.9)$$

Además, si el modelo directo es bueno,

$$\mathbf{Y_r(k)} \cong er(k) + Y_m(k) \quad (3.10)$$

Para obtener $eu(k-1)$ se propone linearizar la función g en un tiempo k dado en función de $u(k-1)$. Entonces se obtiene una función lineal g dada por:

$$Ym(k) = \hat{g}(\hat{u}(k-1)) = \left. \frac{\partial g}{\partial u(k-1)} \right|_Q (\hat{u}(k-1) - u(k-1)) + g|_Q \quad (3.11)$$

Donde Q es el punto de operación en el tiempo k dado por el vector de autoregresos, $\varphi(k)$, para el tiempo k y donde $\hat{u}(k-1)$ es una variable.

$$Q \mapsto \varphi(k) = [Yp(k-1), \dots, Yp(k-na), u(k-1), \dots, u(k-nb)] \quad (3.12)$$

Si se evalúa la ecuación 3-8 para $\hat{u}(k-1)$ igual a $u(k-1)$ se obtiene $Ym(k)$ que debe ser similar a $Yp(k)$. Además, si se reemplaza $\hat{u}(k-1)$ por $ud(k-1)$ se obtiene una aproximación de $Yr(k)$.

$$Yp(k) \cong Ym(k) = \left. \frac{\partial g}{\partial u(k-1)} \right|_Q (u(k-1) - u(k-1)) + g|_Q = g|_Q \quad (3.13)$$

$$Yr(k) \cong Ym(k) = \left. \frac{\partial g}{\partial u(k-1)} \right|_Q (\hat{u}(k-1) - u(k-1)) + g|_Q \quad (3.14)$$

Luego reemplazando en 3-4

$$Yr(k) - Yp(k) = er(k) \cong \left. \frac{\partial g}{\partial u(k-1)} \right|_Q (ud(k-1) - u(k-1)) \quad (3.15)$$

y utilizando 3-5 llegamos a la siguiente expresión que relaciona er con eu

$$er(k) = \left. \frac{\partial g}{\partial u(k-1)} \right|_Q eu(k-1) \quad (3.16)$$

La derivada de la función g de la red neuronal es fácil de calcular. Aquí se presenta simplemente el resultado:

$$\left. \frac{\partial g}{\partial u(k-1)} \right|_Q = \sum_j W_j w_{ji} \left(1 - \tanh^2 \left[\sum_i w_{ji} \phi_i(k) + w_{j0} \right] \right) \quad (3.17)$$

donde W_j son los pesos de la neurona de salida; w_{ji} es el peso i de la neurona j de la capa oculta $\phi_i(k)$ es el autoregresor i

El **modelo neuronal inverso** es también un modelo NNARX construido con un perceptron de una sola capa. Este modelo luego del entrenamiento por modelo de referencia será el controlador. La salida es la fuerza de control y es función de los siguientes autoregresores:

$$u(k) = f(r(k+1), Y_p(k), \dots, Y_p(k-na), u(k-1), \dots, u(k-nb)) \quad (3.18)$$

Donde $r(k+1)$ es la referencia. Si bien la muestra $r(k+1)$ es de un tiempo futuro está presente en el instante k y representa cómo debe responder el sistema en $k+1$.

El modelo inverso es la condición inicial para el entrenamiento por modelo de referencia. Para construirlo se lo entrenará off-line con datos entrada salida tomados de la planta.

DISEÑO DEL CONTROLADOR

El procedimiento de diseño de un controlador por modelo de referencia requiere una serie de pasos. Estos se muestran en la figura 3-3. El primero consiste en diseñar un sistema de referencia que cumpla con las especificaciones. Luego, es necesario obtener un conjunto de datos entrada-salida de la planta para ser utilizados en los pasos posteriores. Es importante que los datos cubran todo el rango de operación en el cual la planta deberá funcionar. Con estos datos se entrenan un modelo neuronal directo y uno inverso utilizando las técnicas presentadas en el capítulo dos de este trabajo.

El modelo neuronal inverso se utiliza como condición inicial del controlador en la etapa de su entrenamiento.

Una vez validados los modelos se genera una señal de referencia (Y_{ref} . Ver figura 3-2). Para asegurarse la convergencia se debe elegir una señal que funcione al sistema dentro de los límites de operación utilizados en el experimento (paso 1). Es preferible una señal que mantenga al sistema dentro del rango de operación y además lejos de los límites de este rango.

Los pesos sinápticos serán modificados en una magnitud $\Delta\omega_{ij}$. Este valor puede ser regulado por una constante μ denominada en inglés “step size”. Un valor pequeño de esta constante puede aumentar el tiempo de convergencia del algoritmo. Esto se evidencia de modo que a medida que transcurren las iteraciones durante el entrenamiento el error cuadrático casi no varía. Un valor relativamente alto de esta constante puede causar que el entrenamiento no converja. Existe un valor óptimo entre ambos extremos. Se propone comenzar con valores pequeños y aumentarlos mientras que el algoritmo no diverja.

El número de iteraciones representa el número de veces que se utilizará la señal de referencia para realizar las actualizaciones de los parámetros del controlador. Para las primeras pruebas se recomienda utilizar una única iteración. Luego, para el entrenamiento definitivo el número de iteraciones debe ser incrementado hasta que el error cuadrático no disminuya significativamente.

El paso siguiente es el entrenamiento del controlador. Puede ocurrir que el algoritmo no converja. Es decir que el error cuadrático aumente en vez de disminuir o que no se modifique. Ambos casos no son deseados. El primero puede ser causado por: 1- El conjunto de datos entrada-salidas con el cual los modelos neuronales fueron entrenados no cubre el rango de operación exigido en el entrenamiento. 2 - La señal de referencia no mantiene al sistema dentro del rango de operación para el cual los modelos inverso y directo fueron entrenados. 3 - El valor de la constante μ es mayor que el óptimo.

Que el error cuadrático no disminuya es causado principalmente porque el valor de la constante μ es muy pequeño y las modificaciones de los parámetros del controlador ($\Delta\omega_{ij}$) no son significativas.

ENTRENAMIENTO

En la figura 3-4 se presenta un esquema del algoritmo de entrenamiento. Este consiste en un par de ciclos anidados. Durante el ciclo interior se toma una muestra de la señal de referencia y se computa la salida del modelo de referencia (Y_{ref}) más la salida del controlador (u). Con este último valor se calcula la salida de la planta (Y_p) y la salida del modelo neuronal de la planta. Luego se computa la diferencia $Y_p - Y_{ref}$ que es el error. Este se retropropaga a través del modelo neuronal directo y se obtiene el error de la fuerza de control, e_u . Y por último, este valor, es utilizado para modificar los pesos sinápticos del controlador. Este procedimiento se realiza para todos los puntos de la señal de referencia completándose el ciclo interior. Este ciclo se repite un número de veces dado por el *número de iteraciones* durante el ciclo exterior.

Es importante destacar que el proceso de entrenamiento debe ser realizado on-line. Es decir que se requiere tomar datos de la planta durante el entrenamiento. Cada vez que se computa la fuerza de control (paso A de la figura 3-4) se actúa sobre la planta con este valor y se obtiene el resultado. Esto es limitante, no sólo porque se exige ocupar la planta, sino también porque si la frecuencia con la que se toman los datos es alta se necesitará un computador suficientemente veloz que pueda realizar todos los cálculos (A, B, C, D y E en la figura 3-4) en un tiempo de muestreo.

Sin embargo el entrenamiento on-line permite que este esquema sea aplicado a un sistema variante en el tiempo (NØrgaard et al, 2001). En este caso el modelo directo de la planta deberá ser actualizado simultáneamente con el controlador. Nuevamente el tiempo de muestreo limita esta aplicación.

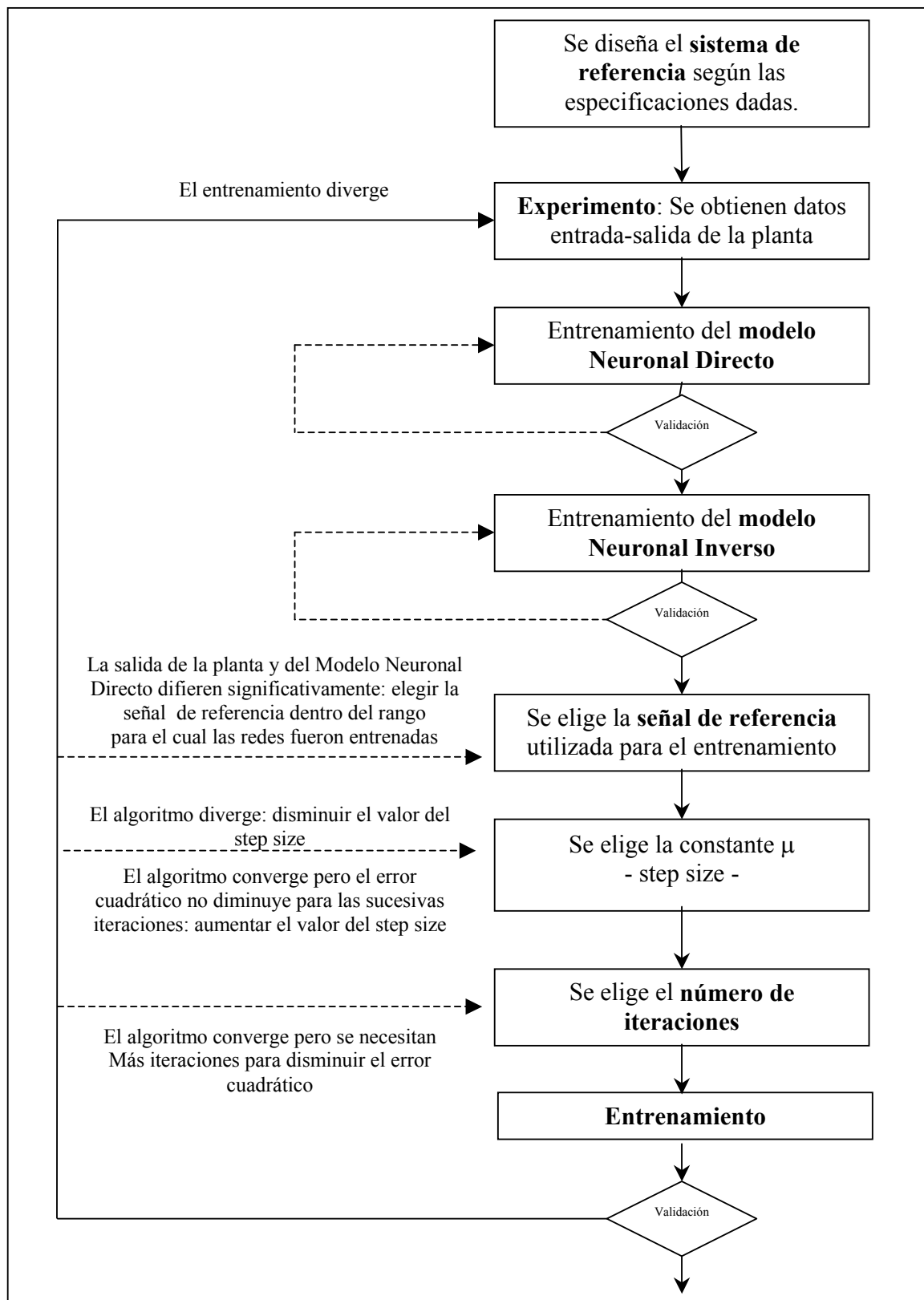


Figura 3-3:Diagrama de flujo para el diseño de un controlador por modelo de referencia

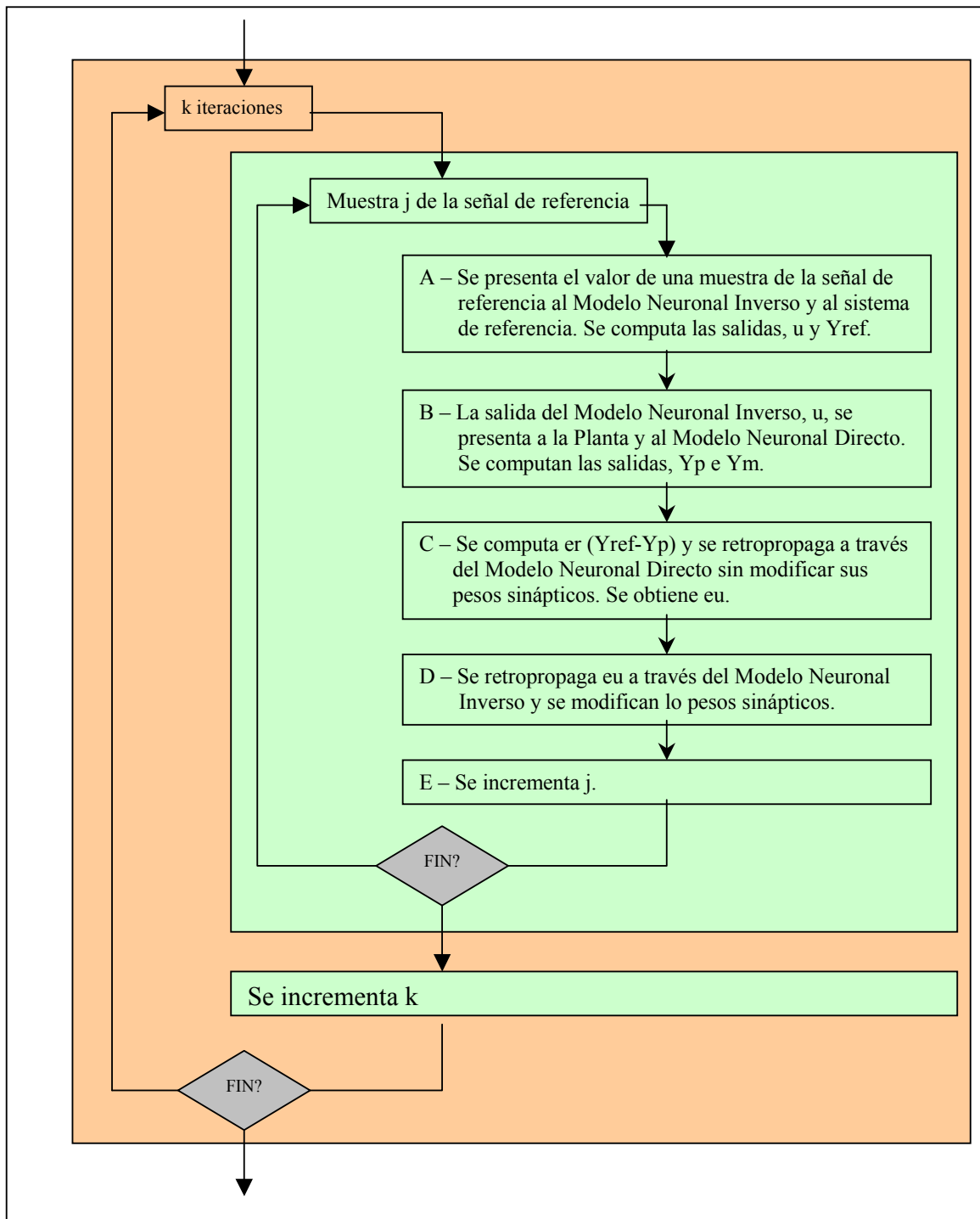


Figura 3-4: Diagrama de flujo del algoritmo de entrenamiento del control por modelo de referencia

CONTROL PREDICTIVO NEURONAL

El tipo de control predictivo analizado en este trabajo se basa en el esquema presentado por Clarke et al. (1978) y conocido como control predictivo generalizado (GPC: “Generalized Predictive control”). Este tipo de controlador computa la fuerza de control de modo tal de minimizar una función de costo definida de la siguiente manera.

$$J(k) = \sum_{i=N1}^{N2} [r(k+i) - Ym(k+i)]^2 + \rho \sum_{i=1}^{Nu} [\Delta u(k+i-1)]^2 \quad (3.19)$$

Donde

- r : Trayectoria deseada
- Ym : Predicción realizada por el modelo neuronal
- $N1$ y $N2$: Definen el rango de predicción
- u : Fuerza de control
- Δ : operador $1-q^{-1}$; siendo $q^{-1}x(t)=x(t-1)$
- Nu : Horizonte de control
- ρ : Factor que penaliza los cambios de la fuerza de control

El esquema del control predictivo se presenta en la figura 3-7. Para cada intervalo de muestreo el controlador computa los Nu valores de la fuerza de control futuros que minimizan la función costo. Luego, el primero de estos es entregado a la planta. Durante el siguiente instante de muestreo se repite el procedimiento. Esta tarea resulta exigente en términos de tiempo de computo. Dado que este disminuye sensiblemente con el valor del parámetro Nu se suele elegir su valor igual a la unidad (NØrgaard et al, 2001).

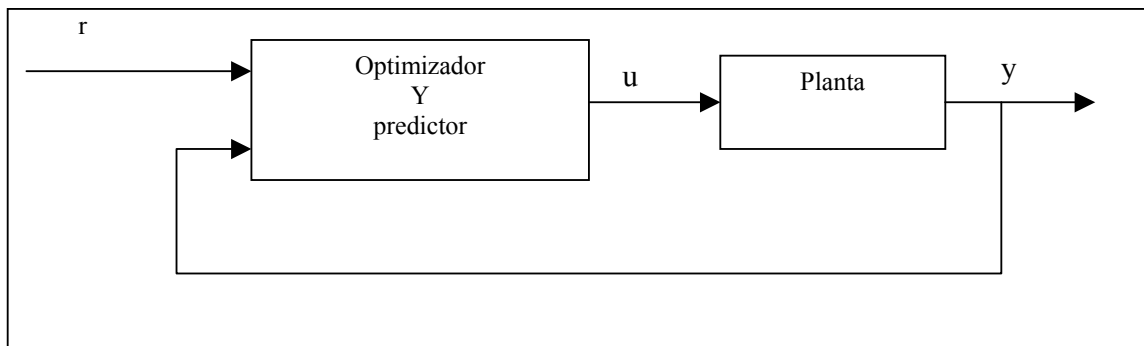


Figura 3-7: Esquema básico del control predictivo

Este esquema presenta un problema de optimización que debe ser resuelto para cada intervalo de muestreo. Para el caso de modelo lineales es posible encontrar una solución analítica. En cambio, para el caso de modelos no lineales, el problema de optimización debe ser resuelto numéricamente.

En orden de obtener el mínimo de la función de costo, el esquema de control predictivo neuronal requiere un modelo neuronal del sistema que pueda estimar la respuesta futura del sistema dentro del rango de predicción. Si bien el modelo neuronal es entrenado off-line para estimar la salida en el instante $k+1$, en este esquema, es forzado a estimar la salida n pasos en el futuro. La predicción a n pasos puede ser escrita de la siguiente manera:

$$Ym(k+n) = g(\varphi(k+n)) \quad (3.20)$$

Donde

$$N_1 \leq n \leq N_2 \quad (3.21)$$

Y

$$\begin{aligned} \varphi(k+n) = [& Ym(k+n-1), \dots, Ym(k+n-\min(n, na)), \\ & y(k-1), \dots, y(k-\max(na-n, 0)), \\ & u(k+n-1), \dots, u(k+n-nb)] \end{aligned} \quad (3.22)$$

A partir de estas ecuaciones es posible minimizar la función costo definida en 3.19 en función de la fuerza de control u . Para lograrlo se emplea un algoritmo de optimización. Algunos de los algoritmos adecuados se presentan en el capítulo 2. Estos son: el método del gradiente, el método de Newton, el método de Gauss-Newton y el método de Levenberg-Marquardt

CONTROL PREDICTIVO POR LINEALIZACIÓN INSTANTÁNEA

Resolver el problema de optimización de la función de costo definida 3.19 es un proceso costoso en términos de tiempo de cómputo. La teoría de control predictivo para sistemas lineales proporciona una solución analítica que reduce significativamente el tiempo de cómputo. Dado que para sistemas no lineales aún no hay una propuesta similar y que obligadamente se debe recurrir a algoritmos numéricos, se propone la siguiente aproximación. Si es posible obtener un modelo lineal para cada instante de muestro y este es un aproximación relativamente buena del sistema dentro del rango de predicción, es válido linealizar el modelo instante a instante y resolver el problema lineal. En la figura 3-8 se observa el esquema de esta propuesta

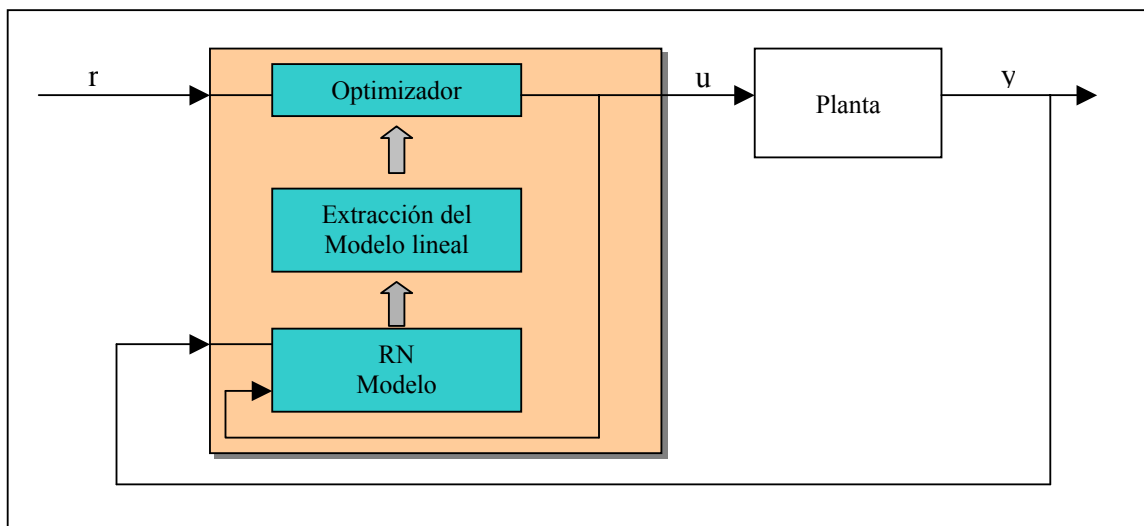


Figura 3-8: Esquema del control predictivo por linealización instantánea

Solución analítica: Ecuación diofántica

El control predictivo generalizado propuesto por Clark (1987) se diseña considerando el modelo ARIMAX dado por la siguiente ecuación:

$$A(q^{-1}).Y(k) = q^{-d} B(q^{-1})u(k) + \frac{e(k)}{\Delta} \quad (3.23)$$

$A(q^{-1})$ y $B(q^{-1})$ son polinomios de coeficientes constantes, d es el retardo del sistema, $e(k)$ es ruido blanco y Δ es el operador $1-q^{-1}$. La predicción óptima de este modelo que minimiza la función costo, J , está dado por

$$\hat{Y}(k+j) = G_j(q^{-1})\Delta u(k-d+j) + F_j(q^{-1})Y(k) \quad (3.24)$$

Donde $\hat{Y}(k+j)$ es la predicción del modelo j pasos en el futuro dentro del rango de predicción dado por N_1 y N_2 e $Y(k)$ es la salida medida de la planta en el instante k . Además $G_j(q^{-1})$ es un polinomio dado por

$$G_j(q^{-1}) = E_j(q^{-1})B(q^{-1}) \quad (3.25)$$

y $E_j(q^{-1})$ es otro polinomio que junto con $F_j(q^{-1})$ cumplen con la ecuación diofántica

$$1 = E_j(q^{-1}).A(q^{-1}).\Delta + q^{-j}F_j(q^{-1}) \quad (3.26)$$

Además, estos dos polinomios son únicos definidos por los polinomios A y B . El polinomio F_j es de orden uno y tiene la forma

$$F_j(q^{-1}) = f_{j,0} + f_{j,1}q^{-1} \quad (3.27)$$

mientras que el polinomio E_j es de orden $(j-1)$ y de la forma

$$E_j(q^{-1}) = 1 + e_{j,1}q^{-1} + \dots + e_{j,j-1}q^{-(j-1)} \quad (3.28)$$

Utilizando la ecuación diofántica y el par de ecuaciones 3.27 y 3.28 se obtienen los coeficientes de los polinomios F_j y E_j . Luego, utilizando estos resultados y la ecuación 3.25 se computan los coeficientes de G_j , ' $g_{j,i}$ '. Dado que

$$\begin{aligned} g_0 &= g_{1,0} = g_{2,0} = \dots = g_{N2,0} \\ g_1 &= g_{2,1} = \dots = g_{N2,1} \\ &\dots \\ g_{N2-1} &= g_{N2,N2-1} \end{aligned} \quad (3.29)$$

Se construye una matriz G de la forma

$$G = \begin{bmatrix} g_0 & 0 & \dots & 0 \\ g_1 & g_0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{Nu-1} & g_{Nu-2} & \dots & g_0 \\ \vdots & \vdots & & \vdots \\ g_{N2-1} & g_{N2-2} & \dots & g_{N2-Nu} \end{bmatrix}_{NxNu} \quad (3.30)$$

Y una matriz de la forma

$$f = \begin{bmatrix} f_{k+1} \\ f_{k+2} \\ \vdots \\ f_{k+N2} \end{bmatrix} = \begin{bmatrix} [G_1(q^{-1}) - g_0]\Delta u(k) + F_1(q^{-1})Y(k) \\ [G_1(q^{-1}) - g_1q^{-1} - g_0]\Delta u(k+1) + F_1(q^{-1})Y(k) \\ \vdots \\ [G_1(q^{-1}) - g_{N2-1}q^{-1} - \dots - g_1q^{-1} - g_0]\Delta u(k+1) + F_1(q^{-1})Y(k) \end{bmatrix} \quad (3.31)$$

La matriz f depende exclusivamente de términos conocidos hasta 'k' de 'u' y de 'Y'. Utilizando estas matrices puede escribirse el vector de predicciones óptimas de la siguiente manera

$$\hat{Y} = G\tilde{u} + f \quad (3.32)$$

Siendo

$$\tilde{u} = \{\Delta u(k), \Delta u(k+1), \dots, \Delta u(k+N-1)\}^T \quad (3.33)$$

un vector que agrupa los términos desconocidos a determinar. De reemplazar las predicciones óptimas (ec. 3.32) en la ecuación de costo J , se desprende el resultado analítico para el cómputo de la fuerza de control u . Este es:

$$u(k) = u(k-1) + H^T (ref - f) \quad (3.34)$$

donde 'ref' es el vector de valores futuros de la referencia y H^T es la primera fila de la matriz dada por $(G^T G + \lambda I)^{-1} G^T$.

Linealización del modelo Neuronal

La solución analítica propuesta requiere conocer los polinomios A y B del modelo lineal. Al linealizar el modelo neuronal se extraerán estos parámetros de los polinomios.

La predicción del modelo neuronal está dada por una función no lineal del vector de autorregresores.

$$Y_m(k) = g(\varphi(k)) \quad (3.35)$$

Para extraer un modelo lineal de la red neuronal en el instante k_0 se debe linealizar la ecuación 3.23 alrededor del punto de operación dado por.

$$\varphi(ko) = [Ym(ko-1), \dots, Ym(ko-na), u(ko-1), \dots, u(ko-nb)] \quad (3.36)$$

Luego, la aproximación lineal queda de la siguiente manera

$$\begin{aligned} Ym(k) - Ym(ko) &= \left. \frac{\partial g(\varphi(k))}{\partial \varphi_1(k)} \right|_{\varphi(k)=\varphi(ko)} \cdot (Ym(k-1) - Ym(ko-1)) + \dots \\ &+ \left. \frac{\partial g(\varphi(k))}{\partial \varphi_{na}(k)} \right|_{\varphi(k)=\varphi(ko)} \cdot (Ym(k-na) - Ym(ko-na)) + \\ &+ \left. \frac{\partial g(\varphi(k))}{\partial \varphi_{na+1}(k)} \right|_{\varphi(k)=\varphi(ko)} \cdot (u(k-1) - u(ko-1)) + \dots \\ &+ \left. \frac{\partial g(\varphi(k))}{\partial \varphi_{na+nb}(k)} \right|_{\varphi(k)=\varphi(ko)} \cdot (u(k-na) - u(ko-na)) \end{aligned} \quad (3.37)$$

El cómputo de las derivadas se realiza a partir de la expresión 2.15 propuesta para el análisis de sensibilidad de modelos. De modo de obtener una expresión más compacta se realizan los siguientes reemplazos

$$a_i = - \left. \frac{\partial g(\varphi(k))}{\partial \varphi_i(k)} \right|_{\varphi(k)=\varphi(ko)} ; 1 \leq i \leq na \quad b_i = \left. \frac{\partial g(\varphi(k))}{\partial \varphi_i(k)} \right|_{\varphi(k)=\varphi(ko)} ; na+1 \leq i \leq na+nb \quad (3.38)$$

$$\bar{Y}(k-j) = Ym(k-j) - Ym(ko-j) \quad \bar{u}(k-j) = u(k-j) - u(ko-j) \quad (3.39)$$

Luego se obtiene

$$\bar{Y}(k) = -a_1 \bar{Y}(k-1) - \dots - a_{na} \bar{Y}(k-na) + b_1 \bar{u}(k-1) + \dots + b_{nb} \bar{u}(k-nb) \quad (3.40)$$

Además, utilizando las ecuaciones 3.27 3.28 y reordenando algunos términos, se obtiene

$$A(q^{-1}).Ym(k) = q^{-1}B(q^{-1})u(k) + \varsigma(ko) \quad (3.41)$$

Donde $Y_m(k)$ es la aproximación lineal de la salida del modelo neuronal; los operadores $A(q^{-1})$ y $B(q^{-1})$ están dados por:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \dots + a_{na} q^{-na} \quad (3.42)$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + \dots + b_{nb} q^{-nb} \quad (3.43)$$

y

$$\begin{aligned} \zeta(ko) &= A(q^{-1})Y_m(ko) - q^{-1}B(q^{-1})u(ko) \\ &= Y_m(ko) + a_1 Y_m(ko-1) + \dots + a_{na} Y_m(ko-na) - b_0 u(ko-1) - \dots - b_{nb} u(ko-nb) \end{aligned} \quad (3.44)$$

EMULACIÓN DE UN CONTROLADOR PREDICTIVO CON UNA RED NEURONAL

Un controlador predictivo neuronal en cada instante de muestreo debe realizar una búsqueda del mínimo de la función de costo J en función de las fuerzas de control futuras. En el caso que la frecuencia de muestreo deba ser alta dado que la planta es rápida o si el computador que ejecuta el algoritmo de control no es lo suficientemente veloz, implementar un controlador predictivo neuronal puede resultar, en términos de tiempo computacional, ineficiente. Se propone, entonces, entrenar una red neuronal que emule el comportamiento de este tipo de controlador. De esta manera, el tiempo de cómputo de la señal de control se reduce significativamente.

Entrenar este controlador neuronal requiere de un conjunto de datos experimentales del controlador predictivo neuronal. El primer paso de esta estrategia de control es diseñar el controlador predictivo. Una vez hecho esto se realiza un experimento de modo de obtener el conjunto de datos experimentales. En la figura 3-9 se muestra un esquema de un posible experimento. El proceso de adquisición de datos se realiza a lazo abierto dado que se supone que el controlador predictivo no puede ser implementado en planta.

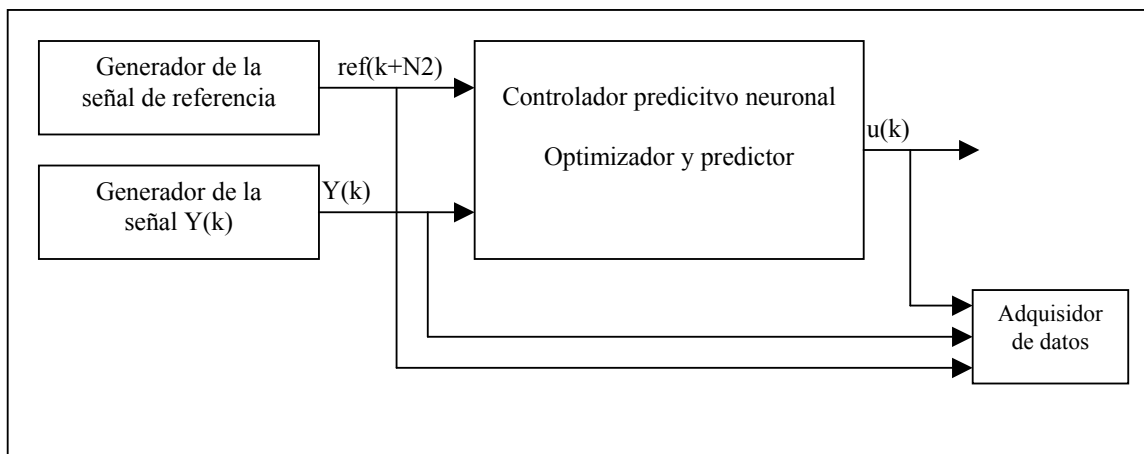


Figura 3-9: Experimento para la adquisición del conjunto de datos experimentales de un controlador predictivo neuronal

El controlador predictivo neuronal consiste en un bloque que computa la señal ‘u’ para el tiempo ‘k’ en función de dos señales de entrada: la referencia, ‘ref’, y la salida de la planta, ‘Y’. El sistema es de tipo MISO si se considera que el horizonte de control es igual a 1. Para el entrenamiento de la red neuronal se deben considerar todas las señales de entrada sin embargo esto no requiere modificación importantes de los algoritmos de entrenamiento. La siguiente ecuación describe la relación de la salida con las entradas.

$$u(k) = g(\text{ref}(k + N2), \dots, \text{ref}(k + N1), y(k - 1), \dots, y(k - na), \dots, y(k - nb)) \quad (3.45)$$

Donde $N2$ $N1$ definen el rango de predicción y na y nb son el número de autoregresores del modelo neuronal de la planta que utiliza el controlador predictivo para estimar las salidas futuras.