TECNOLÓGICO
NACIONAL DE MÉXICO

cenidet®
Centro Nacional de Investigación
y Desarrollo Tecnológico

Técnicas Modernas de Identificación de Sistemas

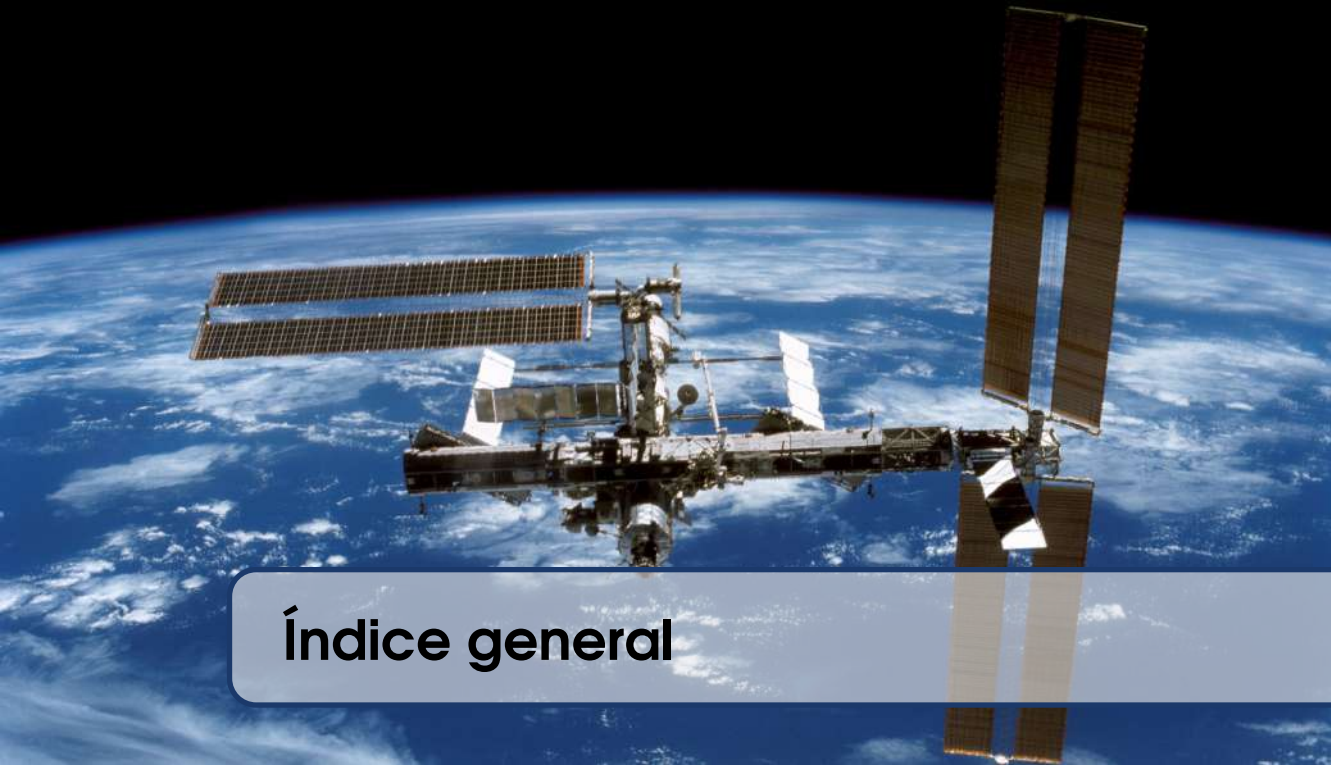
Una Guía Práctica

Víctor Manuel Alvarado Martínez

ORCID 0000-0003-1769-9607
Scopus Author ID 6603538550
Google Scholar ID user=SalhrYIAAAAJ

PUBLISHED BY TECNOLÓGICO NACIONAL DE MÉXICO

First printing, March 2023



Índice general

I Modelos lineales

1 Sistemas Lineales Invariantes en el Tiempo y Causales 3

1.1 Modelos en ecuaciones diferenciales 3
1.2 Modelos en ecuaciones en diferencias 6
1.3 Predictor de un paso 10
1.4 Mínimos cuadrados 13

2 Modelos en función de transferencia 17

2.1 Modelo Estócastico 17
2.2 Estructura ARX 22
2.3 Estructura ARMAX 24
2.4 Estructura OE 27
2.5 Estructura BJ 30
2.6 Otras Estructuras 32

3 Modelos para sistemas en lazo cerrado 35

3.1 Identificación en lazo Cerrado 35

3.2 Estructura CLOE 36
3.3 Algoritmo CLOE 40
3.4 CLOE + Perturbación extendida 42

II Modelos no lineales

4 Modelos usando redes neuronales 47
4.1 Arquitectura de redes neuronales 47
4.2 Red Neuronal - Ejemplo simple 50
4.3 Red Neuronal - Especial SYSID 54

5 Modelos a bloques: Hammerstein-Wiener 61
5.1 Modelos a Bloques 61
5.2 Modelo Hammerstein-Wiener 63
5.3 Hammerstein-Wiener: Sobreparametrización 64
5.4 Hammerstein-Wiener: Iterativo 67

Bibliografía 71
Conferencias 71
Artículos 72
Libros 73

Index 75

Índice de figuras

2.1 Diagrama a bloques del Modelo Estócastico. 20
2.2 Diagrama a bloques de la estructura ARX. 23
2.3 Diagrama a bloques de la estructura ARMAX. 26
2.4 Diagrama a bloques de la estructura OE. 28
2.5 Diagrama a bloques de la estructura BJ. 30
2.6 Diagrama a bloques de la estructura ARARMAX. 33

3.1 Diagrama a bloques de la estructura CLOE. 36

4.1 Neurona Artificial. 47
4.2 Red Neuronal de 2 capas. 51
4.3 Red Neuronal Artificial 2n-2-1. 55
4.4 Red Neuronal Artificial 2-2-1. 58
4.5 Red Neuronal Artificial 2-1. 58

5.1 Modelo Hammerstein. 61
5.2 Modelo Wiener. 61
5.3 Modelo Hammerstein-Wiener. 62
5.4 Modelo Hammerstein-Wiener con funciones. 62
5.5 Realización del Modelo Hammerstein-Wiener. 63

Dedicatoria

A mi amada esposa

Ma. Guadalupe López López

A mis adorados hijos

Froylán Antonio Alvarado López

Cristina Zayetsy Alvarado López

Agradecimientos

Libro elaborado durante el ejercicio del periodo Sabático autorizado por el TecNM

Contribución Académica

Esta obra aborda una temática útil a varias áreas de la ingeniería, cuando se necesita un modelo para cálculos, diseño o control de un sistema, planta o proceso.

Después de 30 años la teoría ya está bien probada pero las referencias básicas siguen solo en inglés. Existen algunos libros en español pero en realidad son tesis convertidas en libros de poca difusión. Contar con textos en español contribuirían a una mas rápida asimilación en diferentes ingenierías del TecNM. La temática es de aplicación directa y por eso en algunas carreras de ingeniería proceden a identificar sistemas mediante aplicaciones en software sin comprender bien el fondo, esta obra pretende suplir esa carencia.

En la Linea de Generación y Aplicación del Conocimiento de ***Control Automático*** del Departamento de Ingeniería Electrónica el curso de Identificación de Sistemas tiene 20 años impartiéndose, es una materia esencial en la formación de los estudiantes, disponer de un texto propio puede mejorar las condiciones de aprendizaje.

En cuanto a los objetivos educacionales de la obra se puede decir que contribuye con las bases matemáticas particulares a está técnica, a conocer 4 de las estructuras básicas de modelos y cuando se aplican, a encontrar una solución numérica para el calculo de los parámetros de los modelos y finalmente manejar el proceso de validación de los modelos obtenidos.

Modelos lineales

1 Sistemas Lineales Invariantes en el Tiempo y Causales 3

1.1 Modelos en ecuaciones diferenciales 3
1.2 Modelos en ecuaciones en diferencias 6
1.3 Predictor de un paso 10
1.4 Mínimos cuadrados 13

2 Modelos en función de transferencia . 17

2.1 Modelo Estócastico 17
2.2 Estructura ARX 22
2.3 Estructura ARMAX 24
2.4 Estructura OE 27
2.5 Estructura BJ 30
2.6 Otras Estructuras 32

3 Modelos para sistemas en lazo cerrado 35

3.1 Identificación en lazo Cerrado 35
3.2 Estructura CLOE 36
3.3 Algoritmo CLOE 40
3.4 CLOE + Perturbación extendida 42

1. Sistemas Lineales Invariantes en el Tiempo y Causales

En este capitulo se presentan los modelos de base a utilizar para el modelado de sistemas dinámicos. Primero en tiempo continuo a pesar de que en este texto no hay identificación de modelos en tiempo continuo pero es una referencia de base y los modelos en tiempo discreto son un símil de su contraparte.

1.1 Modelos en ecuaciones diferenciales

La representación matemática para sistemas dinámicos más común son las ecuaciones diferenciales, para modelos lineales se emplean las ordinarias de orden n. Para el caso de sistemas dinámicos lo más general es incluir también una entrada externa de orden m. Para que el sistema sea causal es necesario que $n \geq m$. La ecuación queda de la siguiente manera:

$$\begin{aligned} &\frac{dy^n(t)}{d^n t} + a_{n-1}\frac{dy^{n-1}(t)}{d^{n-1}t} + a_{n-2}\frac{dy^{n-2}(t)}{d^{n-2}t} + \cdots + a_1\frac{dy(t)}{dt} + a_0 y(t) = \\ &b_m\frac{du^m(t)}{d^m t} + b_{m-1}\frac{du^{m-1}(t)}{d^{m-1}t} + b_{m-2}\frac{du^{m-2}(t)}{d^{m-2}t} + \cdots + b_1\frac{du(t)}{dt} + b_0 u(t). \end{aligned} \tag{1.1}$$

Por convención para los sistemas dinámicos $y(t)$ representa la salida o variable dependiente del sistema y $u(t)$ representa la entrada o variable manipulada del sistema, ambas son función del tiempo. La transformada de Laplace de la ecuación 1.1, considerando condiciones iniciales cero, tiene la siguiente expresión:

$$s^nY(s)+a_{n-1}s^{n-1}Y(s)+a_{n-2}s^{n-2}Y(s)+\cdots+a_1sY(s)+a_0Y(s)= \quad (1.2)$$
$$b_ms^mU(s)+b_{m-1}s^{m-1}U(s)+b_{m-2}s^{m-2}U(s)+\cdots+b_1sU(s)+b_0U(s).$$

La función de transferencia es la representación más usada para los modelos de sistemas dinámicos, es la relación $Y(s)/U(s)$ de la ecuación 1.2

$$\frac{Y(s)}{U(s)}=\frac{s^n+a_{n-1}s^{n-1}+a_{n-2}s^{n-2}+\cdots+a_1s+a_0}{b_ms^m+b_{m-1}s^{m-1}+b_{m-2}s^{m-2}+\cdots+b_1s+b_0}. \quad (1.3)$$

A veces es necesario tener una representación temporal en vez de la variable compleja s de Laplace, en ese caso se utiliza el operador diferencial p que se define como $p=d/dt$, la ecuación 1.1 puede entonces escribirse como:

$$p^ny(t)+a_{n-1}p^{n-1}y(t)+a_{n-2}p^{n-2}y(t)+\cdots+a_1py(t)+a_0y(t)= \quad (1.4)$$
$$b_mp^mu(t)+b_{m-1}pu^{m-1}u(t)+b_{m-2}p^{m-2}u(t)+\cdots+b_1pu(t)+b_0u(t).$$

Factorizando $y(t)$ y $u(t)$, funciones del tiempo, es posible escribir la relación entrada-salida como:

$$y(t)=\frac{p^n+a_{n-1}p^{n-1}+a_{n-2}p^{n-2}+\cdots+a_1p+a_0}{b_mp^m+b_{m-1}p^{m-1}+b_{m-2}p^{m-2}+\cdots+b_1p+b_0}u(t). \quad (1.5)$$

La otra representación muy utilizada es conocida como *Espacio de Estados*. Primero se debe transformar la ecuación 1.1 en n ecuaciones ordinarias de primer orden y luego representarla de manera matricial. Para transformar existen varios métodos, uno muy empleado por su conversión directa es: seleccionar las variables de estado

$$x_1(t)=y(t), x_2(t)=y'(t), \cdots, x_n(t)=\tfrac{d^{n-1}y(t)}{dt^{n-1}}, \quad (1.6)$$

y por consiguiente

$$\dot{x}_1(t)=x_2(t), \dot{x}_2(t)=x_3(t), \cdots, \dot{x}_{n-1}(t)=x_n(t), \dot{x}_n(t)=\tfrac{d^ny(t)}{dt^n}. \quad (1.7)$$

Las variables de estado puedan agruparse en un vector

$$x(t)=[x_1(t)\,x_2(t)\,\cdots\,x_n(t)]^T \quad (1.8)$$

pudiendo entonces representar la ecuación diferencial ordinaria de orden n en la forma matricial:

$$\begin{aligned} \dot{x}(t) &= A\,x(t) + B\,u(t) \\ y(t) &= C\,x(t) + D\,u(t) \end{aligned} \tag{1.9}$$

donde A es una matriz cuadrada de coeficientes constantes con dimensión nxn, B es un vector de coeficientes constantes con dimensión nx1, C es un vector de coeficientes constantes con dimensión 1xn y D es un escalar.

Para el caso de la ecuación 1.1 la elección de las variables de estado, dado las derivadas en la entrada u, es el siguiente:

$$\begin{aligned} \dot{x}_1(t) &= x_2(t) + c_1 u(t) \\ \dot{x}_2(t) &= x_3(t) + c_2 u(t) \\ &\vdots\ \vdots\ \vdots \\ \dot{x}_{n-1}(t) &= x_n(t) + c_{n-1} u(t) \\ \dot{x}_n(t) &= -a_0 x_2(1) - a_1 x_2(t) - a_2 x_3(t) \cdots - a_{n-2} x_{n-1}(t) - a_{n-1} x_n(t) + c_n u(t). \end{aligned} \tag{1.10}$$

Entonces las matrices A y B quedan como sigue, la matriz $C = [1\,0\,\ldots\,0\,0]$ y la matriz $D = c_0$.

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix} \quad B = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} \tag{1.11}$$

Los coeficientes $c_0, \cdots, c_{n-1}$ deben ser calculados a partir de las variables de estado por simple despeje recursivo. Para entender como desenmarañar veamos el caso cuando $m = n$, la salida y su derivada quedan:

$$y(t) = x_1(t) + c_0 u(t) \tag{1.12}$$

$$\tag{1.13}$$

$$\frac{\mathrm{dy(t)}}{\mathrm{d}t} = \frac{\mathrm{dx_1(t)}}{\mathrm{d}t} + c_0 \frac{\mathrm{du(t)}}{\mathrm{d}t}. \tag{1.14}$$

Pero $\dot{x}_1(t)$ corresponde con la primera linea de la ecuación matricial, ya descrita arriba, sustituyendo en la derivada de la salida

$$\frac{\mathrm{dy(t)}}{\mathrm{d}t} = x_2(t) + c_0 \frac{\mathrm{du(t)}}{\mathrm{d}t} + c_1 u(t), \tag{1.15}$$

continuando con la segunda derivada de la salida y sustituyendo $\dot{x}_2(t)$ ahora por la segunda linea de la ecuación matricial

$$\frac{dy^2(t)}{d^2t} = x_3(t) + c_0\frac{du^2(t)}{d^2t} + c_1\frac{\mathrm{du(t)}}{\mathrm{d}t} + c_2u(t), \tag{1.16}$$

se vuelve evidente de continuar así hasta el orden $m = n$ recuperamos la ecuación original 1.1 y por simple igualación de coeficientes se puede despejar $c_0, \cdots, c_{n-1}$.

Veamos un caso concreto $m = n = 4$, igualando coeficientes las equivalencias son:

$$c_0 = b_4 \tag{1.17}$$
$$c_1 = b_3 - a_1c_0$$
$$c_2 = b_2 - a_1c_1 - a_0c_0$$
$$c_3 = b_1 - a_1c_2 - a_2c_1 - a_3c_0$$
$$c_4 = b_0 - a_1c_3 - a_2c_2 - a_3c_1 - a_4c_0. \tag{1.18}$$

Esto se puede generalizar como:

$$c_0 = b_m \tag{1.19}$$
$$c_k = b_{m-k} - \sum_{i=1}^{k} a_ic_{k-i} \mid k = 1,2,\cdots,n. \tag{1.20}$$

Para el caso en que $n > m$ todas las ecuaciones quedan igual simplemente los coeficientes b_i para $n > i > m$ son igual a cero. En particular la salida es igual a la primer variable de estado $y(t) = x_1(t)$ y $D = 0$.

1.2 Modelos en ecuaciones en diferencias

La identificación de sistemas tiene como principal característica que se trabaja con datos, es decir con variables discretas. Entonces se impone presentar el equivalente de los modelos destacados en la sección previa.

En discreto un sistema dinámico es modelado por una ecuación en diferencias de orden n equivalente en el tiempo continuo a la ecuación 1.1.

$$\begin{aligned} &y(t-n)+a_{n-1}y(t-(n-1))+a_{n-2}y(t-(n-2))+\cdots \\ &+a_1y(t-1)+a_0y(t)=b_mu(t-m)+b_{m-1}u(t-(m-1)) \\ &+b_{m-2}u(t-(m-2))+\cdots+b_1u(t-1)+b_0u(t). \end{aligned} \tag{1.21}$$

Muy importante notar que en el caso discreto la variable t que usamos para representar el tiempo toma valores discretos $t = 0,1,2,3,\ldots$. Tenemos una fuerte asociación con la variable t para representar el tiempo y dado que todos los modelos de este libro son discretos no es ventajoso utilizar una variable diferente. La identificación de modelos en tiempo continuo es otra temática, ver por ejemplo [8, 22].

Otra diferencia destacable con respecto a los sistemas continuos es que en discreto no siempre se tiene que n es mayor que m, en efecto, es posible que se presenten sistemas dinámicos para los cuales $m > n$ en discreto, pero son realmente pocos casos con algunas propiedades que salen del objeto de estudio del libro. Solo se consideran sistemas discretos para lo que $n \geq m$, igual al caso continuo.

De igual manera es posible una representación en una variable compleja en este caso mediante la Transformada z, la función de transferencia discreta queda:

$$\frac{Y(z)}{U(z)} = \frac{b_mz^m+b_{m-1}z^{m-1}+b_{m-2}z^{m-2}+\cdots+b_1z+b_0}{z^n+a_{n-1}z^{n-1}+a_{n-2}z^{n-2}+\cdots+a_1z+a_0}. \tag{1.22}$$

Por cuestiones de implementación numérica es mas conveniente representar la función de transferencia discreta en la variable z^{-1}. Normalmente esta representación en potencias negativas se obtiene directamente de la ecuación de diferencias ordinaria 1.21 pero también es posible obtenerla de la representación en potencias positivas 1.22 mediante manipulación algebraica. Tiene la siguiente forma:

$$\frac{Y(z^{-1})}{U(z^{-1})} = \frac{b_mz^{-m}+b_{m-1}z^{-(m-1)}+b_{m-2}z^{-(m-2)}+\cdots+b_1z^{-1}+b_0}{z^{-n}+a_{n-1}z^{-(n-1)}+a_{n-2}z^{-(n-2)}+\cdots+a_1z^{-1}+a_0} \tag{1.23}$$

Es muy importante notar que las ecuaciones 1.22 y 1.23 no son iguales, la ecuación 1.22 es la transformada z de 1.21 directamente. Mientras la ecuación

1.23 muestra simplemente que tiene la misma forma, en realidad solo es un asunto de arreglar los coeficientes para poderlas igualar.

También para el caso discreto es muy útil una representación temporal como alternativa a la variable compleja z. Así como en continuo se utiliza el operador p en discreto se usa el operador q conocido como *operador en adelanto* y se define como $qy(t) = y(t+1)$. La relación entrada-salida con el operador q equivalente a la función de transferencia discreta representada en la ecuación 1.22 es

$$y(t) = \frac{b_m q^m + b_{m-1} q^{m-1} + b_{m-2} q^{m-2} + \cdots + b_1 q + b_0}{q^n + a_{n-1} q^{n-1} + a_{n-2} q^{n-2} + \cdots + a_1 q + a_0} u(t). \qquad (1.24)$$

Como ya se dijo por cuestiones de implementación práctica es más común utilizar la representación con potencias negativas por eso se hace uso del *operador en atraso* q definido como $q^{-1}y(t) = y(t-1)$. Así llegamos a la representación más utilizada para un sistema dinámico, la relación entrada-salida en forma de cociente como una función de transferencia discreta

$$y(t) = \frac{b_m q^{-m} + b_{m-1} q^{-(m-1)} + b_{m-2} q^{-(m-2)} + \cdots + b_1 q^{-1} + b_0}{q^{-n} + a_{n-1} q^{-(n-1)} + a_{n-2} q^{-(n-2)} + \cdots + a_1 q^{-1} + a_0} u(t) \qquad (1.25)$$

La representación de estados en variables discretas se hace de la misma manera que en el caso continuo, en este caso en n ecuaciones de diferencia de primer orden con tiempo discreto $t = 0, 1, 2, 3, \ldots$

$$\begin{aligned} x(t+1) &= A\,x(t) + B\,u(t) \\ y(t) &= C\,x(t) + D\,u(t) \end{aligned} \qquad (1.26)$$

Existen varios procedimientos para obtenerla, en el caso continuo elegimos la primer variable de estado como la salida y así sucesivamente, luego recuperamos la ecuación diferencial ordinaria 1.1 de orden n y despejamos los coeficientes. En el caso discreto vamos a obtener la representación en espacio de estados a partir de la función de transferencia o relación entrada-salida 1.24.

Separamos en dos partes, una que contenga el numerador y otra el denominador mediante una variable instrumental $v(t)$

$$v(t) = \frac{1}{q^n + a_{n-1}q^{n-1} + a_{n-2}q^{n-2} + \cdots + a_1 q + a_0} u(t), \tag{1.27}$$

$$y(t) = \frac{b_m q^m + b_{m-1}q^{m-1} + b_{m-2}q^{m-2} + \cdots + b_1 q + b_0}{1} v(t). \tag{1.28}$$

Elegimos los estados

$$\begin{aligned}
x_1(t) &= v(t) \\
x_1(t+1) &= x_2(t) \\
x_2(t+1) &= x_3(t) \quad = x_1(t+2) \\
&\vdots \quad \vdots \qquad \vdots \\
x_{n-2}(t+1) &= x_{n-1}(t) = x_1(t+n-2) \\
x_{n-1}(t+1) &= x_n(t) \quad = x_1(t+n-1) \\
x_n(t+1) &= \qquad = x_1(t+n).
\end{aligned} \tag{1.29}$$

Ya solo falta $x_n(t+1) = x_1(t+n)$, lo podemos obtener de la ecuación 1.27 reescribiéndola y despejando $v(t+n) = x_1(t+n) = x_n(t+1)$

$$\begin{aligned}
v(t+n) = u(t) - a_{n-1}v(t+n-1) - a_{n-2}v(t+n-2) + \cdots \\
-a_1 v(t+1) - a_0 v(t).
\end{aligned} \tag{1.30}$$

Ya tenemos entonces la matriz A y el vector B de la representación matricial.

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{1.31}$$

Para la salida $y(t)$ reescribimos la ecuación 1.28

$$\begin{aligned}
y(t) = b_m v(t+m) + b_{m-1}v(t+m-1) + b_{m-2}v(t+m-2) + \\
\cdots + b_1 v(t+1) + b_0 v(t).
\end{aligned} \tag{1.32}$$

con esto ya tenemos el vector C y como vemos que la salida no depende directamente de la entrada el escalar D es cero, quedando completa la representación en variables de estado discreta.

$$C = \begin{bmatrix} b_0 & b_1 & \cdots & b_{m-2} & b_{m-1} & b_m & 0_{m+1} & \cdots & 0_n \end{bmatrix} \tag{1.33}$$

Notar que el vector C se complementa con ceros, ya que el vector de estados x es de dimensión n, para el caso general $n > m$ habrá i ceros para $n > i > m$. Cuando $m = n$ no habrá ceros que agregar, el último coeficiente de C será b_m. Esta forma de representación matricial de las ecuaciones en diferencia se conoce como forma canónica controlable. Para identificación en espacio de estados ver [17, 1, 10].

1.3 Predictor de un paso

Un predictor es una herramienta matemática que nos permite hacer una conjetura respecto al valor futuro de una variable. Tiene aplicación en muchas áreas del conocimiento, finanzas, econometría, control automático [24], clima, etc. Es un sujeto muy basto y complejo, resulta a veces en algoritmos pesados computacionalmente hablando y se sale del tópico del libro. Sin embargo, tenemos necesidad de uno muy sencillo, el que se resuelve cuando se tienen todos los datos y solo hay que predecir el próximo valor a la salida, se le conoce como el ***Predictor de un paso***.

El predictor será uso en el próximo capítulo para establecer el regresor que nos permita calcular parámetros de los modelos en función de transferencia.

Sea el siguiente modelo del sistema a identificar

$$y(t) = G(q)u(t) + H(q)e(t) = G(q)u(t) + v(t), \tag{1.34}$$

este modelo será ampliamente estudiado en el capítulo 2, por el momento el propósito es exclusivamente encontrar el predictor de un paso para este modelo.

$G(q)$ y $H(q)$ son las funciones de transferencia o representaciones de un sistema, típicamente obtenidas por transformada z y representadas ya sea en

z o en q. El modelo tiene dos partes en la primera la entrada es $u(t)$ que es la señal de entrada a la planta o proceso esa parte es determinista, no cambia, mientras que la segunda parte tiene como entrada $e(t)$ que es una señal aleatoria, esta segunda parte se dice probabilística y el conjunto de ambas partes se dice Modelo Estocástico.

Aunque $G(q)$ y $H(q)$ son del mismo tipo no son lo mismo en el sentido de que $G(q)$ representa al sistema, planta o proceso, depende de la física, su forma no la fijamos nosotros. Caso contrario $H(q)$ que es la función acompañamiento de la variable aleatoria y sirve para moldear la perturbación, es elegida, por tanto puedo poner las restricciones que se ajusten a mis necesidad. En particular deseo que sea invertible, lo garantizo, que sea estable y que sea mónico, todo esto sin perdida de generalidad.

Trabajemos primero con la perturbación $v(t)$, expresada en transformada z con variable q es así

$$v(t) = H(q)e(t) = \sum_{i=0}^{\infty} h(i)q^{-i}e(t) = \sum_{i=0}^{\infty} h(i)e(t-i). \tag{1.35}$$

Mónico quiere decir que el primer término sea uno, eso siempre se puede.

Estable significa que

$$\sum_{i=0}^{\infty} \left|h(i)\right| < \infty, \tag{1.36}$$

e invertible significa que $e(t) = H(q)^{-1}v(t)$, expresado en transformada

$$e(t) = \tilde{H}(q)v(t) = \sum_{i=0}^{\infty} \tilde{h}(i)v(t-i), \tag{1.37}$$

para asegurarse que sea invertible basta que

$$\sum_{i=0}^{\infty} \left|\tilde{h}(i)\right| < \infty. \tag{1.38}$$

Predictor para $v(t)$, en su forma de transformada quitamos el primer termino de la sumatoria y renombramos el resto

$$v(t) = \sum_{i=0}^{\infty} h(i)e(t-i) = e(t) + \sum_{i=i}^{\infty} h(i)e(t-i) = e(t) + m(t-1), \tag{1.39}$$

el nombre $m(t-1)$ hace referencia aque solo tiene valores hasta un instante anterior de tiempo puesto que la sumatoria comienza en 1 y no en el tiempo actual. La predicción condicional de $v(t)$ se denota $\hat{v}(t|t-1)$, significa la predicción de $v(t)$ en el tiempo t conociendo todos los valores anterior de $v(t)$ hasta un tiempo antes. La mejor predicción para $v(t)$ conocida como *Maxima Predicción a Posteriori*, MAP por sus siglas en inglés, es $m(t-1)$ ya que contiene toda la información de $v(t)$ hasta un instante antes y dado que la media o promedio de $e(t) = 0$, es decir el valor esperado de adición a la suma del único término que le falta es cero por lo tanto

$$\hat{v}(t|t-1) = m(t-1) = \sum_{i=i}^{\infty} h(i)e(t-i). \tag{1.40}$$

Algunas maneras útiles de expresar $\hat{v}(t|t-1)$ son

$$\begin{aligned}\hat{v}(t|t-1) &= \sum_{i=i}^{\infty} h(i)e(t-i) + e(t) - e(t) = H(q)e(t) - e(t) \\ &= [H(q)-1]e(t) = [1-H^{-1}(q)]v(t).\end{aligned} \tag{1.41}$$

La predicción condicional para la salida, a partir del modelo original de la ecuación 1.34 es

$$\hat{y}(t|t-1) = G(q)u(t) + \hat{v}(t|t-1), \tag{1.42}$$

recordar que la primera parte del modelo es determinística no hay predicción a hacer.

Sustituyendo $\hat{v}(t|t-1)$ de la ecuación 1.41

$$\hat{y}(t|t-1) = G(q)u(t) + [1-H^{-1}(q)]v(t), \tag{1.43}$$

sustituyendo $v(t)$ a partir del modelo original ecuación 1.34

$$\hat{y}(t|t-1) = G(q)u(t) + [1-H^{-1}(q)][y(t) - G(q)u(t)], \tag{1.44}$$

después de un arreglo algebraico finalmente tenemos la expresión del predictor de un paso para la salida

$$\hat{y}(t|t-1) = H^{-1}(q)G(q)u(t) + [1-H^{-1}(q)]y(t), \tag{1.45}$$

esta es la ecuación buscada que será ampliamente utilizada en el siguiente capitulo.

1.4 Mínimos cuadrados

La estimación de los parámetros de un modelo es una actividad esencial, típicamente cierra el procedimiento durante la identificación de sistemas. El método de estimación más utilizado es el de mínimos cuadrados, existen diferentes implementaciones o algoritmos, nosotros tenemos necesidad de uno que pueda actualizarse con cada nuevo dato que llegue, se les conoce como algoritmos recursivos de métodos cuadrados. Enseguida describimos el que será empleado en este libro, presentado entre otros por Landau en 2006 [18].

El arreglo básico consiste de dos vectores, un vector de parámetros θ y un vector de datos $\phi(t)$ también conocido como regresor. Notar que el vector de datos es variante en el tiempo puesto que las variables del sistema lo son también, en cambio el vector de parámetros es constante ya que se trata de los coeficiente constantes de un modelo.

$$\theta = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_{n-1} \end{bmatrix}^T \tag{1.46}$$

$$\phi = \begin{bmatrix} \phi_0 & \phi_1 & \dots & \phi_{n-1} \end{bmatrix}^T \tag{1.47}$$

El problema básico de mínimos cuadrados es encontrar unos coeficientes constantes para los datos del sistema tal que se aproximen a la salida predicha de ese sistema:

$$\hat{y}(t) = \theta^T \phi(t) = \phi^T(t)\theta. \tag{1.48}$$

El ajuste de predicción se hace para cada salida del sistema por eso es más apropiado un algoritmo recursivo, donde solo se actualiza la información a cada instante de tiempo y no se hacen cálculos con todo el histórico.

La predicción de la salida en el tiempo $t+1$ puede hacerse de dos maneras. La primera es considerando únicamente la información disponible hasta el tiempo t, en ese caso se le conoce como estimación *a priori* $\hat{y}^\circ(t+1)$. La segunda se hace con información ya del tiempo $t+1$ entonces se conoce como estimación *a posteriori* $\hat{y}(t+1)$. Consistentemente tenemos dos errores de predicción, el error *a priori* y el error *a posteriori* calculados respectivamente con la salida estimada correspondiente.

$$\begin{aligned} \varepsilon^\circ(t+1) &= y(t+1) - \hat{y}^\circ(t+1) \\ \varepsilon(t+1) &= y(t+1) - \hat{y}(t+1) \end{aligned} \tag{1.49}$$

La actualización del vector de parámetros se hace de manera clásica, el valor actual más una corrección. El factor de corrección tiene 3 términos: una ganancia de adaptación $F(t)$, los datos $\phi(t)$ y el error $\varepsilon(t+1)$. La ganancia nos da el tamaño del paso en la búsqueda, los datos contienen la información y el error nos señala que tan cerca estamos y la dirección.

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F(t)\phi(t)\varepsilon(t+1) \tag{1.50}$$

Para actualizar la ganancia de adaptación $F(t+1)$ se usa un factor de corrección cuadrático para garantizar una convergencia en descendencia convexa, solamente que se normaliza para hacerlo independiente de las magnitudes. Adicionalmente se le incluyen dos parámetros de ajuste $\lambda_1(t)$ y $\lambda_2(t)$.

$$F(t+1) = \frac{1}{\lambda_1(t)}\left[F(t) - \frac{F(t)\phi(t)\phi^T(t)F(t)}{\frac{\lambda_1(t)}{\lambda_2(t)} + \phi^T(t)F(t)\phi(t)}\right] \tag{1.51}$$

La ganancia de adaptación se inicializa con un valor positivo $F(0) > 0$. Un primer caso en dejar la ganancia constante $F(t+1) = F(t) = F(0)$ durante la evolución del algoritmo, eligiendo $\lambda_1(t)$ y $\lambda_2(t)$ también constantes, típicamente $F(0) = 1000$, $\lambda_1(t) = 1$ y $\lambda_2(t) = 0$. Esto da un algoritmo sencillo, fácil de implementar y que es suficiente para sistemas de bajo orden y poco ruido.

Un segundo caso es establecer una traza constante para la ganancia de adaptación $tr\,F(t+1) = tr\,F(t) = tr\,F(0)$ se aconseja para sistemas con parámetros variables en el tiempo.

Con los parámetros de ajuste $\lambda_1(t)$ y $\lambda_2(t)$ se puede por ejemplo fijar una ganancia decreciente o establecer un factor de olvido fijo o variable.

Si se quiere una ganancia decreciente en realidad hay que dejar fijos y con poco peso los parámetros de ajuste para que no afecten ya que por construcción la ganancia es decreciente, típicamente $\lambda_1(t) = 1$ y $\lambda_2(t) = 1$. Se recomienda para sistemas con parámetros constantes.

Para evitar que la búsqueda se quede estancada se suele utilizar un factor de olvido, así la nueva información tiene mas peso y actualiza la estimación. Por ejemplo se escogen $\lambda_2(t) = 1$ y $0 < \lambda_1(t) < 1$, se aconseja para sistemas con variación lenta de variables.

Ya por último es posible combinar por ejemplo traza constante y factor de olvido para aprovechar las mejores características de cada método. Aplica en casos difíciles por ejemplo cuando no se tiene información inicial de los parámetros. Para mas detalles en la configuración de los parámetros de ajuste del algoritmo de mínimos cuadrados con fines de identificación ver [18].

Por último en el algoritmo de mínimos cuadrados se actualiza el error *a posteriori* usando el error *a priori* solo que normalizado igual que la ganancia para hacerlos compatibles. Con esto se reinicia el ciclo para un nuevo tiempo t.

$$\varepsilon(t+1) = \frac{\varepsilon^{\circ}(t+1)}{1+\phi^{T}(t)F(t)\phi(t)} \tag{1.52}$$

2. Modelos en función de transferencia

Este capitulo presenta los modelos lineales más utilizados para sistemas de una entrada - una salida en función de transferencia discreta. Son 4 estructuras a considerar: ARX, ARMAX, OE y BJ. Además constituye la base para todo modelado por Identificación de Sistemas [23].

2.1 Modelo Estócastico

La función de transferencia es la relación entrada-salida más utilizada para los sistemas de una entrada - una salida, SISO por sus siglas en inglés. Para identificación multivariable ver por ejemplo [25]. En este libro los modelos empleados son discretos porque los datos obtenidos son muestreados y las soluciones numéricas bien establecidas son para modelos discretos. Para una discusión detallada ver por ejemplo [2].

Los modelos a identificar son los presentados en el capítulo anterior ecuaciones 1.23 y 1.25 . La ecuación 1.23 es una representación en el plano complejo z y la ecuación representa la relación temporal de la entrada-salida con el operador q, aquí la última para discutir sobre ella.

$$\begin{aligned} y(t) &= \frac{b_m q^{-m} + b_{m-1} q^{-(m-1)} + b_{m-2} q^{-(m-2)} + \cdots + b_1 q^{-1} + b_0}{q^{-n} + a_{n-1} q^{-(n-1)} + a_{n-2} q^{-(n-2)} + \cdots + a_1 q^{-1} + a_0} u(t) \\ &= G(q) u(t) \end{aligned} \tag{2.1}$$

$G(q)$ es la función de transferencia discreta para el sistema de entrada $u(t)$ y

salida $y(t)$, nótese el abuso de notación, normalmente debería ser $G(q^{-1})$, para fácil lectura es común encontrarla así y siendo consistentes no genera ningún problema.

Cuando se elabora un modelo por primeros principios y termina en una ecuación diferencial ordinaria, la función de transferencia 2.1 es el tipo de modelo que podemos obtener, dependiendo del sistema y sus condiciones de operación la precisión puede no ser muy buena.

Típicamente cuando se elabora un modelo por primeros principios nos damos cuenta que hay variables que pueden modificar la salida, pero que no puedo manipular, a esto se le conoce como *entradas conocidas no manipuladas*. Algunos ejemplos son: la temperatura ambiental en un sistema de regulación de temperatura en una casa, el viento en la conducción de un avión, la marea en la dirección de una barca, la fricción en los neumáticos de un auto en la conducción o la temperatura en un componente electrónico. Esta es la primera razón por la que un modelo del tipo $G(q)$ no es suficiente, hay entradas no consideradas, que pueden o no tener un impacto importante en la salida.

Estas entradas no manipuladas se clasifican en dos tipos: las que se miden y las que no. Es importante hacer esta distinción porque de alguna manera la información que arrojan las medibles no pueden dar un ajuste o calibración del modelo, es información valiosa que se puede incorporar *a posteriori* al modelado y con técnicas avanzadas se puede incorporar durante el proceso de modelado. En cambio las que no podemos medir solo pueden darnos una idea de su influencia en la salida pero no una cuantificación de su efecto. Dentro de las medibles encontramos temperatura ambiental, temperatura interior, presión atmosférica, velocidad del viento, voltaje de alimentación o irradiación solar. Aunque vale aclarar que depende de la envergadura del sistema en cuestión, así, aunque estas variables sea posible medirlas en la práctica no se realiza, por ejemplo, en los equipos de climatización no se incluye sensor de temperatura ambiental exterior o un sistema de paneles fotovoltaicos no incluye sensor de irradiación solar, mientras grandes equipos como un generador eólico vertical incluye medidores de temperatura, viento y presión atmosférica. Ejemplos de entradas no manipuladas ni medibles son: el oleaje en un barco, la turbulencias en un avión, las fluctuaciones en un flujo de entrada en un proceso industrial en estado estable o la fricción. Aunque sabemos que están ahí y tienen un efecto considerable a la salida no se miden por costoso, irrealizable o impráctico.

A parte de las entradas no manipuladas también otras variables que no en-

tran o no se propagan en el sistema pero que tienen una influencia en la salida, típicamente de manera general se les conoce como perturbaciones. El ruido eléctrico, el ruido acústico, las vibraciones en una planta o proceso, son ejemplos de perturbaciones. Muchas veces no tienen un gran impacto en la salida pero si son fuente de error para el modelado.

Otro aspecto a considerar son las dinámicas no modeladas. Cuando se hace el modelado por principios físicos o químicos frecuentemente hay dinámicas que se dejan de tomar en cuenta es parte del proceso normalmente por simplificación para evitar complejidades de poco aporte.

Por último las no linealidades que se dejan de lado. Por simpleza o comodidad en el manejo del modelo o simplemente porque se desea diseñar un control y se necesita un modelo lineal, algunas no linealidades se dejan de lado o bien se trabaja cerca de un punto de operación y se linealiza de tal manera que la no linealidad tenga poca influencia.

Por estas razones citadas es notoriamente insuficiente un modelo determinístico que solo tomo como influencia la entrada manipulable de la planta:

$$y(t) = G(q)u(t). \tag{2.2}$$

El reto es incorporar todos estos fenómenos al modelo, pero como se ve son de muy diversa índole, sería complejo el proceso y el modelo resultante, lo que se busca entonces es un compromiso entre simplicidad y eficiencia del modelado.

La idea original de ***Lennart Ljung*** [19] propuesta en 1987 es un modelado estocástico: una parte determinística y una parte probabilística de tal manera que se pudiera cubrir un largo espectro con una sola adición buscando dejar simple el modelo.

$$y(t) = G(q)u(t) + v(t) = G(q)u(t) + H(q)e(t) \tag{2.3}$$

donde $e(t)$ es una variable aleatoria, definida por su función de densidad de probabilidad, PDF por sus siglas en inglés, $H(q)$ es una función de transferencia elegida apropiadamente para moldear $e(t)$ de tal maneera que se puedan representar fenómenos aleatorios pero también un símil de uno determinístico y $v(t)$ engloba la representación conjunta de varios fenómenos y se le llama de manera genérica *perturbación*. Un diagrama a bloques que permite mejor visualizar esta idea se muestra en la figura 2.1, se aprecia claramente la superposición de los

dos sistemas, el determinístico y el probabilístico.

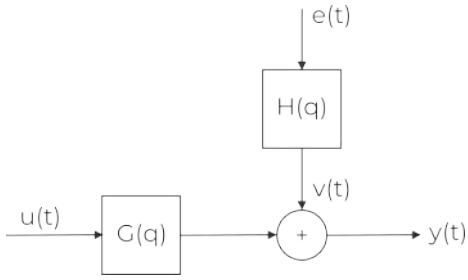


Figura 2.1: Diagrama a bloques del Modelo Estócastico.

Analizando esta idea se puede destacar, dado que nos manejamos dentro de un contexto lineal es perfectamente valido agregar simplemente por superposición $v(t)$, un termino que engloba todos los fenómenos no presentes en la función de transferencia entrada-salida. Elegir el mismo tipo de representación para $v(t)$, es decir una función de transferencia $H(q)$ con una función de excitación $e(t)$ aprovecha la riqueza de una función de transferencia para modelar fenómenos dinámicos lineales y aparte conserva la misma estructura que $G(q)$ esto permite no introducir otro tipo de términos que compliquen su solución.

Pero, ¿elegir de esta manera $v(t)$ realmente permite representar todo tipo de perturbación? la respuesta es un claro sí. Es posible elegir parejas de $H(q)$ y $e(t)$ tal que $v(t)$ tenga pseudo-forma de escalón, de pulso, de ruido, de exponencial o de señal periódica. Más aún en la práctica se elige $e(t)$ suficientemente genérico y solo se calcula la $H(q)$ que le acompaña. Tipicamente se elige $e(t)$ como un ruido blanco gaussiano (RBG) de media cero y varianza λ.

En conclusión el modelo propuesto para sistemas dinámicos que se usa en este capítulo queda de la siguiente forma:

$$\begin{aligned} y(t) &= G(q)u(t) + H(q)e(t) \\ e(t) &= RBG\left\{E(e(t)) = 0, E(e^2(t)) = \lambda\right\}. \end{aligned} \tag{2.4}$$

Dada la estructura, lo siguiente es ¿cómo encontramos $G(q)$ y $H(q)$? No hay una respuesta simple. Antes de la propuesta de ***Ljung*** la idea dominante era "encontrar" el "verdadero" modelo. Partamos de un ejemplo sencillo para entender: un circuito eléctrico RC en serie, resistencia-capacitor, su modelo por leyes de Kirchhoff y Ohm es conocido y sencillo, es una ecuación diferencial

ordinaria de primer orden, expresada como función de transferencia entre el voltaje de entrada $V_e(s)$ y como salida el voltaje en la resistencia $V_R(s)$

$$V_R(s) = \frac{RCs}{1+RCs} V_e(s), \tag{2.5}$$

como se ve en la función de transferencia, los parámetros, el de la resistencia R en *Ohms* y el del capacitor C en faradios F aparecen de manera explícita lo que es muy útil. Por esa razón durante mucho tiempo el interés era re-encontrar esos parámetros en el modelado a partir de datos conocido hay como identificación de sistemas. Eso era llamado encontrar el "verdadero" modelo lo cual es una falacia. Primero porque ese modelo por muy fácil lectura que tenga y respeto a la concepción física de la materia en realidad es solo una representación humana la naturaleza no viene con modelo y segundo solo son un puñado de sistemas para los que se puede plantear una ecuación simple como esta, para la mayoría de los sistemas físicos no tenemos una representación a usar de inmediato es parte de la valía de la identificación de sistemas.

Aceptado el hecho de que no vamos a encontrar parámetros físicos durante el modelado por identificación dejamos libre $G(q)$ y $H(q)$ para cualquier expresión. Recordando que el punto de partida son las ecuaciones de diferencia ordinarias de orden n y que su función de transferencia en realidad se obtiene mediante una transformación a la variable compleja z cuya definición es

$$G(z) = \sum_{k=-\infty}^{\infty} g(k) z^{-k}, \tag{2.6}$$

es decir, una sumatoria infinita de términos, algo poco práctico y que solo en algunos casos conduce a una expresión regular sin sumatorias. Utilizar entonces directamente la transformada z para encontrar el modelo no es explotable.

La propuesta es un compromiso entre la generalidad de la función transformada en z y un subconjunto menos basto pero con la posibilidad de ser representado por un numero finito de parámetros. Definimos θ como el vector de parámetros a encontrar, entonces el modelo propuesto es

$$\begin{aligned} y(t) &= G(q,\theta)u(t) + H(q,\theta)e(t) \\ e(t) &= RBG\left\{E(e(t)) = 0, E(e^2(t)) = \lambda\right\}, \end{aligned} \tag{2.7}$$

donde $G(q,\theta)$ y $H(q,\theta)$ son un conjunto de modelos dependientes de los parámetros θ. Todo el trabajo consiste en explorar dichos modelos en la búsqueda de uno que ajuste bien los datos. Por abuso de notación a veces encontramos el modelo como la ecuación 2.4 pero hace referencia al modelo parametrizado.

2.2 Estructura ARX

Particularizando el modelo general se obtiene la primer estructura, la más simple en construcción. Tomamos la ecuación 1.21 de diferencias ordinaria de orden n y agregamos la pertubación mas sencilla que podemos tener, es decir solo agregamos $e(t)$.

$$\begin{aligned} &y(t-n)+a_{n-1}y(t-(n-1))+a_{n-2}y(t-(n-2))+\cdots \\ &+a_1y(t-1)+a_0y(t)=b_mu(t-m)+b_{m-1}u(t-(m-1)) \\ &+b_{m-2}u(t-(m-2))+\cdots+b_1u(t-1)+e(t). \end{aligned} \tag{2.8}$$

Al estar trabajando con sistemas discretos en la práctica no puede haber respuesta instantánea, es decir la entrada $u(t)$ en el tiempo t tiene consecuencias a partir de $t+1$, nunca en el mismo valor de tiempo, se dice que los sistemas discretos presentan al menos un retardo de uno. Por esa razón no aparece el termino $b_0u(t)$. Aplicamos el operador de retraso q^{-1} y factorizamos la entrada y la salida

$$\begin{aligned} &y(t)[q^{-n}+a_{n-1}q^{n-1}+a_{n-2}q^{n-2}+\cdots+a_1q^{-1}+a_0]= \\ &u(t)[b_mq^{-m}+b_{m-1}q^{m-1}+b_{m-2}q^{m-2}+\cdots+b_1q^{-1}]+e(t). \end{aligned} \tag{2.9}$$

Creamos un vector que contenga todos los parámetros desconocidos

$$\theta=\begin{bmatrix} a_0 & a_1 & \dots & a_{n-2} & a_{n-1} & b_1 & b_2 & \dots & b_{m-1} & b_m \end{bmatrix}^T. \tag{2.10}$$

Definimos los polinomios $A(q,\theta)$ y $B(q,\theta)$

$$\begin{aligned} A(q,\theta)&=[q^{-n}+a_{n-1}q^{n-1}+a_{n-2}q^{n-2}+\cdots+a_1q^{-1}+a_0] \\ B(q,\theta)&=[b_mq^{-m}+b_{m-1}q^{m-1}+b_{m-2}q^{m-2}+\cdots+b_1q^{-1}]. \end{aligned} \tag{2.11}$$

Reescribimos la ecuación 2.9

$$A(q,\theta)y(t)=B(q,\theta)u(t)+e(t). \tag{2.12}$$

Puesta en forma del modelo general

$$y(t) = \frac{B(q,\theta)}{A(q,\theta)}u(t) + \frac{1}{A(q,\theta)}e(t) = G(q,\theta)u(t) + H(q,\theta)e(t), \quad (2.13)$$

donde

$$G(q,\theta) = \frac{B(q,\theta)}{A(q,\theta)} \quad H(q,\theta) = \frac{1}{A(q,\theta)}. \quad (2.14)$$

Este modelo representado en la ecuación 2.13 es conocido como modelo ARX, por sus siglas en inglés, auto regresivo con entrada externa. Es más fácil de observar sus partes en un diagrama de bloques, en este caso la figura 2.2. Notar que por construcción la variable externa tiene el mismo denominador es decir la misma dinámica que la planta, esta es la característica de esta arquitectura.

Dado que se tiene una tira de datos entrada salida, es decir tenemos a nuestra disposición todos los instantes de tiempo de muestreo se puede recurrir al predictor de un paso como herramienta para la estimación del vector de parámetros θ. Su forma general es la siguiente:

$$\hat{y}(t|\theta) = H^{-1}(q,\theta)G(q,\theta)u(t) + [1 - H^{-1}(q,\theta)]y(t). \quad (2.15)$$

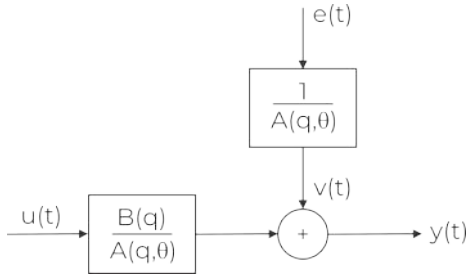


Figura 2.2: Diagrama a bloques de la estructura ARX.

Sustituyendo $G(q,\theta)$ y $H(q,\theta)$ de la ecuación 2.14 y simplificando, el estimador de la salida dependiente del vector θ tiene la forma

$$\hat{y}(t|\theta) = B(q,\theta)u(t) + [1 - A(q,\theta)]y(t). \quad (2.16)$$

Recordando que el polinomio $A(q,\theta)$ es mónico, ver ecuación 2.11, la diferencia en el corchete elimina los unos y nos quedamos unicamente con los coeficientes de $a_0 \dots a_{n-1}$ en negativo. Más los b_m coeficientes del polinomio $B(q,\theta)$, tenemos un total de $n+m$ coeficientes a estimar. Creamos un vector de datos del mismo tamaña para hacer corresponder con los términos de los polinomios:

$$\phi(t) = [u(t-m) \dots u(t-1) - y(t-n) \cdots - y(t-2) - y(t-1)]^T, \quad (2.17)$$

de esta manera la ecuación 2.16 puede reescribirse como:

$$\hat{y}(t|\theta) = \theta^T \phi(t) = \phi^T(t)\theta, \quad (2.18)$$

lo cual convierte nuestro problema de estimar los coeficientes del vector de parámetros θ en la resolución de un problema clásico de mínimos cuadrados. El lector curioso puede multiplicar ambos vectores $\phi(t)$ y θ para verificar que el resultado es idéntico a lo obtenido con la ecuación 2.16.

Tenemos una tira de datos de tamaño N donde $N >> n$ por lo que se puede ver $\phi(t)$ como una ventana en la tira de datos que se va recorriendo hasta pasarlos todos. Los datos en el vector $\phi(t)$ dependen del tiempo por lo que va tomando segmentos de la tira de datos. En realidad se tiene una matriz conformada por todos los valores de $\phi(t)$ para cada t por renglón. Se utiliza cualquier solver para calcular los valores de los coeficientes de θ que minimizan el error con respecto a los datos.

Una vez calculado θ con los coeficientes se forman los polinomios $A(q,\theta)$ y $B(q,\theta)$ y finalmente $G(q,\theta)$ y $H(q,\theta)$ para tener el modelo completo en formato estándar, tipo ecuación 2.13.

2.3 Estructura ARMAX

Para la primer estructura se eligió comenzar con la perturbación más sencilla en esta segunda la idea es poner un modelo completo a la perturbación, entonces la ecuación de diferencias queda:

$$\begin{aligned} &y(t-n) + a_{n-1}y(t-(n-1)) + a_{n-2}y(t-(n-2)) + \dots \\ &+a_1 y(t-1) + a_0 y(t) = b_m u(t-m) + b_{m-1}u(t-(m-1)) \\ &+b_{m-2}u(t-(m-2)) + \cdots + b_1 u(t-1) + c_0 e(t) + c_1 e(t-1) + \dots \\ &+c_{r-1}e(t-(r-1)) + e(t-r). \end{aligned} \quad (2.19)$$

Procedemos de la misma manera, aplicar el operador retardo q^{-1} y factorizar.

$$\begin{aligned} &y(t)[q^{-n}+a_{n-1}q^{n-1}+a_{n-2}q^{n-2}+\cdots+a_1q^{-1}+a_0] = \\ &u(t)[b_mq^{-m}+b_{m-1}q^{m-1}+b_{m-2}q^{m-2}+\cdots+b_1q^{-1}] \\ &+e(t)[q^{-r}+c_{r-1}q^{r-1}+\cdots+c_1q^{-1}+c_0]. \end{aligned} \tag{2.20}$$

Definimos un nuevo polinomio $C(q,\theta)$

$$C(q,\theta) = [q^{-r}+c_{r-1}q^{n-1}+c_{r-2}q^{r-2}+\cdots+c_1q^{-1}+c_0], \tag{2.21}$$

reescribimos la ecuación 2.20

$$A(q,\theta)y(t) = B(q,\theta)u(t)+C(q,\theta)e(t). \tag{2.22}$$

Puesta en forma del modelo general

$$y(t) = \frac{B(q,\theta)}{A(q,\theta)}u(t)+\frac{C(q,\theta)}{A(q,\theta)}e(t) = G(q,\theta)u(t)+H(q,\theta)e(t), \tag{2.23}$$

donde

$$G(q,\theta) = \frac{B(q,\theta)}{A(q,\theta)} \quad H(q,\theta) = \frac{C(q,\theta)}{A(q,\theta)}. \tag{2.24}$$

Este modelo representado en la ecuación 2.23 es conocido como modelo ARMAX, por sus siglas en inglés, auto regresivo (AR) de promedio movil (MA) con entrada externa (X). Es más fácil de observar sus partes en un diagrama de bloques, en este caso la figura 2.2. En este caso ya se tiene un modelo completo para la perturbación pero todavía comparte el mismo denominador es decir la misma dinámica que la planta, se considera para esta familia de arquitecturas como la mas completa aunque su limitación es obvia conviene para muchos sistemas en la práctica.

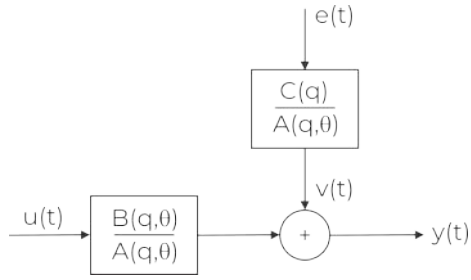


Figura 2.3: Diagrama a bloques de la estructura ARMAX.

Nuevamente se define el vector θ de parámetros para esta arquitectura

$$\theta = \begin{bmatrix} a_0 & \dots & a_{n-1} & b_1 & \dots & b_m & c_0 & \dots & c_{r-1} \end{bmatrix}^T. \tag{2.25}$$

Nuevamente recurrimos al predictor de un paso, ecuación 2.15, para estimar los valores de θ. Sustituyendo $G(q,\theta)$ y $H(q,\theta)$ de la ecuación 2.24 y simplificando, el estimador de la salida dependiente del vector θ tiene la forma

$$C(q,\theta)\hat{y}(t|\theta) = B(q,\theta)u(t) + [C(q,\theta) - A(q,\theta)]y(t). \tag{2.26}$$

Para proceder similar a la estructura ARX agregamos a cada lado de la ecuación $[1 - C(q,\theta)]\hat{y}(t|\theta)$ y después de algunas manipulaciones queda

$$\begin{aligned} \hat{y}(t|\theta) = {} & B(q,\theta)u(t) + [1 - A(q,\theta)]y(t) \\ & + [C(q,\theta) - 1][y(t) - \hat{y}(t|\theta)]. \end{aligned} \tag{2.27}$$

El estimador tiene 3 partes lo cual coincide con los 3 juegos de parámetros a, b y c a encontrar. De manera natural se piensa en un conjunto de datos con 3 tiras de valores, para hacerlos coincidir, pero nosotros solo registramos la entrada y la salida, nuestros datos solo tienen 2 tiras de valores. Este es el problema principal no es posible plantear un regresor lineal como la fue para el caso de ARX.

Lo que haremos es plantear un regresor pseudo-lineal, para esto creamos una variable instrumental $\varepsilon(t,\theta)$ definida como la diferencia entre la salida y su estimado

$$\varepsilon(t,\theta) = y(t) - \hat{y}(t|\theta), \tag{2.28}$$

esta variable constituye nuestra tercer tira de datos que junto a la entrada y a la salida proporcionan la información para resolver el estimador 2.27 y obtener θ.

Los polinomios $A(q,\theta)$ y $C(q,\theta)$ son mónicos pero si vemos el estimador 2.27 ambos pierden el uno por lo que solo quedan sus coeficientes a y c respectivamente, igual que para la estructura ARX los coeficientes a están en negativo. Con estas características podemos construir el vector renglón de datos $\phi(t,\theta)$ teniendo presente que ahora depende de θ porque su tercer tira de valores la variable $\varepsilon(t,\theta)$ así depende

$$\phi(t,\theta) = \begin{bmatrix} u(t-m) & \dots & u(t-1) & -y(t-n) & \dots \\ -y(t-1) & \varepsilon(t-r,\theta) & \dots & \varepsilon(t-1,\theta) & \end{bmatrix}^T. \quad (2.29)$$

Con esto podemos plantear el calculo del vector de parámetros θ como la minimización del error para una forma tipo mínimos cuadrados

$$\hat{y}(t|\theta) = \theta^T \phi(t) = \phi^T(t)\theta, \quad (2.30)$$

la igualdad con la ecuación 2.27 se puede verificar manualmente resolviendo el producto polinomial.

Dado que la tira de datos entrada - salida es de tamaño N donde $N >> n$ tenemos que calcular N valores de $\varepsilon(t-r,\theta)$ para tener las 3 tiras necesarias. Los datos en el vector $\phi(t,\theta)$ dependen del tiempo por lo que va tomando segmentos de la tira de datos. En realidad se tiene una matriz conformada por todos los valores de $\phi(t)$ para cada t por renglón. Se utiliza cualquier solver para calcular los valores de los coeficientes de θ que minimizan el error con respecto a los datos.

Una vez calculado θ con los coeficientes se forman los polinomios $A(q,\theta)$, $B(q,\theta)$ y $C(q,\theta)$, finalmente $G(q,\theta)$ y $H(q,\theta)$ para tener el modelo completo en formato estándar, tipo ecuación 2.23.

2.4 Estructura OE

Las estructuras ARX y ARMAX son lo mas utilizado para modelar pero ambas comparten la característica de que incluyen en la perturbación la dinámica de la planta y no siempre conviene esta asociación. Sin embargo, ARX y ARMAX fueron obtenidos a partir de la ecuación de diferencias, ese camino ya

está agotado necesitamos poder desasociar las dinámicas de la planta y la de la perturbación. Forcemos esta desasociación partamos al revés primero plantemos el diagrama a bloques.

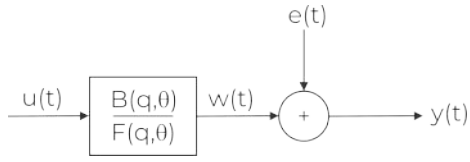


Figura 2.4: Diagrama a bloques de la estructura OE.

Comenzamos con la inclusión de perturbación más simple donde $H(q,\theta) = 1$ es decir directa sin función de transferencia. Dada esa característica a esta estructura se le conoce como *error de salida*, OE por sus siglas en inglés.

La perturbación y la planta no comparten dinámica y de ese hecho no podemos nombrar $A(q,\theta)$ al denominador de la planta ya que no se refleja directamente en la salida, nombramos un nuevo polinomio $F(q,\theta)$, mónico igual que los otros denominadores, sin perdida de generalidad.

$$F(q,\theta) = [q^{-l} + f_{l-1}q^{l-1} + f_{l-2}q^{l-2} + \cdots + f_1 q^{-1} + f_0] \tag{2.31}$$

La ecuación de este sistema es

$$y(t) = \frac{B(q,\theta)}{F(q,\theta)}u(t) + e(t) = G(q,\theta)u(t) + H(q,\theta)e(t), \tag{2.32}$$

expresa claramente la independencia buscada del sistema. Para poder plantearlo en ecuaciones de diferencia hacemos uso de una variable auxiliar $w(t)$ que corresponde a la salida del bloque determinístico antes de incorporar la perturbación.

$$w(t) = \frac{B(q,\theta)}{F(q,\theta)}u(t) \qquad y(t) = w(t) + e(t) \tag{2.33}$$

La ecuación de diferencias para la variable auxiliar es

$$\begin{aligned} &w(t-l) + f_{l-1}w(t-(l-1)) + f_{l-2}w(t-(l-2)) + \cdots \\ &+f_1 w(l-1) + f_0 w(t) = b_m u(t-m) + b_{m-1}u(t-(m-1)) \\ &+b_{m-2}u(t-(m-2)) + \cdots + b_1 u(t-1). \end{aligned} \tag{2.34}$$

Creamos el vector que contiene a todos los parámetros desconocidos

$$\theta = \begin{bmatrix} f_0 & f_1 & \dots & f_{l-2} & f_{n-1} & b_1 & b_2 & \dots & b_{m-1} & b_m \end{bmatrix}^T. \quad (2.35)$$

Planteamos la ecuación polinomial que involucra la entrada con la salida

$$F(q,\theta)y(t) = B(q,\theta)u(t) + F(q,\theta)e(t), \quad (2.36)$$

trasladable fácilmente a una ecuación de diferencias

$$\begin{aligned} &y(t-l) + f_{l-1}y(t-(l-1)) + f_{l-2}y(t-(l-2)) + \cdots \\ &+f_1 y(l-1) + f_0 y(t) = b_m u(t-m) + b_{m-1}u(t-(m-1)) \\ &+b_{m-2}u(t-(m-2)) + \cdots + b_1 u(t-1) + e(t-l) + f_{l-1}e(t-(l-1)) \\ &+f_{l-2}e(t-(l-2)) + \cdots + f_1 e(l-1) + f_0 e(t). \end{aligned} \quad (2.37)$$

Pero hay algo muy importante a considerar $w(t)$ no es observable, tenemos que estimarlo paramétricamente por lo que la ecuación 2.34 en realidad debemos escribirla como

$$\begin{aligned} &w(t-l,\theta) + f_{l-1}w(t-(l-1),\theta) + \cdots + f_1 w(l-1,\theta) \\ &+f_0 w(t,\theta) = b_m u(t-m) + b_{m-1}u(t-(m-1)) \\ &+b_{m-2}u(t-(m-2)) + \cdots + b_1 u(t-1). \end{aligned} \quad (2.38)$$

Para el predictor de un paso, ecuación 2.15, sustituyendo $H(q,\theta) = 1$ llegamos a

$$\hat{y}(t|\theta) = \frac{B(q,\theta)}{F(q,\theta)}u(t) = G(q,\theta)u(t) = w(t,\theta), \quad (2.39)$$

de manera natural $w(t,\theta)$ es el predictor natural de $\hat{y}(t|\theta)$, eso implica que los valores pasados de $\hat{y}(t|\theta)$ pueden ser tomados como $w(t,\theta)$ para la construcción de un regresor.

De la ecuación 2.38 se puede construir el vector renglón de datos

$$\phi(t,\theta) = \begin{bmatrix} u(t-m) & \dots & u(t-2) & u(t-1) \\ -w(t-l,\theta) & \dots & -w(t-2,\theta) & -w(t-1,\theta) \end{bmatrix}^T, \quad (2.40)$$

o lo que es lo mismo

$$\hat{y}(t|\theta) = \phi^T(t,\theta)\theta, \quad (2.41)$$

teniendo una vez mas un problema de mínimos cuadrados a resolver. Pero no medimos $w(t,\theta)$ sustituimos sus datos por el estimado de la salida por lo que es dependiente de θ y una vez más no tenemos una regresión lineal.

Para resolver igual que en los otros casos se forma un vector de datos para cada tiempo, un vector de datos por cada renglón conforman la matriz de datos de N renglones, minimizando para obtener los parámetros de θ con el menor error conforme a la ecuación 2.41.

Una vez calculado θ con los coeficientes se forman los polinomios $F(q,\theta)$ y $B(q,\theta)$, finalmente $G(q,\theta)$ ya que $H(q,\theta) = 1$. Con esto se construye el modelo completo en formato estándar, tipo ecuación 2.32.

2.5 Estructura BJ

Para esta estructura, la idea de partida es la misma que para la estructura OE, es decir, independencia de la planta determinística con la diferencia de que ahora vamos a utilizar un modelo completo para la perturbación. Procediendo de la misma manera empezamos por el diagrama a bloques.

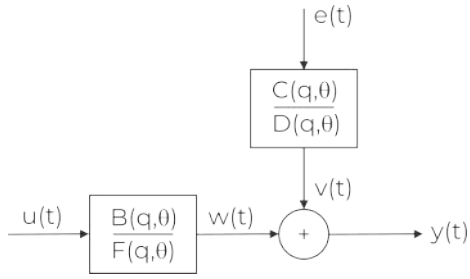


Figura 2.5: Diagrama a bloques de la estructura BJ.

Se destaca la función de transferencia completa en la perturbación y su no conexión con la planta. Este es el esquema mas completo y complejo a

utilizar para sistemas lineales SISO en este texto, se le conoce como Modelo Box-Jenkins (BJ) ya que fue propuesto desde 1970.

Igual que en el caso de OE, al no compartir dinámica la planta y la perturbación no podemos nombrar $A(q,\theta)$ al denominador de la planta ya que no se refleja directamente en la salida, usamos igual a $F(q,\theta)$. El único polinomio nuevo es $D(q,\theta)$, mónico igual que los otros denominadores, sin perdida de generalidad.

$$D(q,\theta) = [q^{-k} + d_{k-1}q^{k-1} + d_{k-2}q^{k-2} + \cdots + d_1 q^{-1} + d_0] \tag{2.42}$$

La ecuación para este sistema

$$y(t) = \frac{B(q,\theta)}{F(q,\theta)}u(t) + \frac{C(q,\theta)}{D(q,\theta)}e(t) = G(q,\theta)u(t) + H(q,\theta)e(t), \tag{2.43}$$

donde

$$G(q,\theta) = \frac{B(q,\theta)}{F(q,\theta)} \quad H(q,\theta) = \frac{C(q,\theta)}{D(q,\theta)}, \tag{2.44}$$

muestra lo completo para representar sistemas pero también la complejidad, ahora son 4 polinomios a estimar. Creamos el vector renglón que contiene a todos los parámetros desconocidos

$$\theta = \begin{bmatrix} f_0 & f_1 & \dots & f_{l-2} & f_{l-1} & b_1 & b_2 & \dots & b_{m-1} & b_m \\ c_0 & c_1 & \dots & c_{r-2} & c_{r-1} & d_0 & d_1 & \dots & d_{k-2} & d_{k-1} \end{bmatrix}^T. \tag{2.45}$$

Es claro que el numero de parámetros a calcular ha doblado y los datos siguen iguales solo la entrada y la salida. Sustituyendo $G(q,\theta)$ y $H(q,\theta)$ en el predictor de un paso, ecuación 2.15

$$\hat{y}(t|\theta) = \frac{D(q,\theta)B(q,\theta)}{C(q,\theta)F(q,\theta)}u(t) + \frac{C(q,\theta) - D(q,\theta)}{C(q,\theta)}y(t). \tag{2.46}$$

Para formar el vector de datos utilizamos el registro de entradas y salidas, pero también lo visto en las estructuras anteriores: el error de predicción y los estimados pasados de salida. En total se necesitan 4 tiras de datos para los

coeficientes de 4 polinomios. Para el nuevo polinomio $D(q,\theta)$ lo mas típico es utilizar una variable relacionada con el error por tratarse de la dinámica de la perturbación.

Esta vez estamos lejos de una regresión lineal y los algoritmos típicos de mínimos cuadrados con inversa frecuentemente fallan, se necesita un algoritmo de mínimos cuadrados robusto. Los programas como el Toolbox de Identificación de Sistemas de ***Matlab*** vienen preparados para esto y calculan frecuentemente bien los polinomios de BJ. Pero no es exclusivo un buen algoritmo de minimización que vienen en los libros de métodos numéricos es suficiente, ver por ejemplo [21].

Pero lo que no cambia es que hay que formar la matriz de datos de N renglones, el tamaño del renglón es igual a la suma de los coeficientes a calcular que corresponden al tamaño de θ.

Una vez calculado θ con los coeficientes se forman los polinomios $F(q,\theta)$, $B(q,\theta)$, $C(q,\theta)$ y $D(q,\theta)$, finalmente $G(q,\theta)$ y $H(q,\theta)$. Con esto se construye el modelo completo en formato estándar, tipo ecuación 2.43.

2.6 Otras Estructuras

Las 4 estructuras presentadas en este capítulo, ARX, ARMAX, OE y BJ son las únicas que son explicadas y explotadas en este libro. Son las más comunes por exitosas pero no son las únicas existen muchas más, sirva esta sección para mencionar algunas de ellas que aunque menos usadas tienen su nicho, es decir son muy buenas para algunas aplicaciones o bien representan retos tecnológicos interesantes.

Modelo **ARIMAX**. Es como la estructura ARMAX pero con una parte de integración mejorada. Es útil para representar sistemas con perturbaciones lentas, explicado en [15]. El modelo e obtenido remplazando la entrada y la salida por sus diferencias, ejemplo: $\Delta y(t) = y(t) - y(t-1)$.

Modelos **ARARX** y **ARARMAX**. Son de la misma familia que ARX y ARMAX la diferencia es la elección de modelos para la perturbación, para ARX y ARMAX se elijó un modelo auto regresivo mientras que para ARARX y ARARMAX se usa un promedio móvil. Con el diagrama a bloques se ve claramente como esta compuesta la estructura ARARMAX, ARARX es igual solo que el polinomio $C = 1$. Es una alternativa al modelado de perturbación

útil cuando tenemos problemas de ajuste. Ver por ejemplo [3].

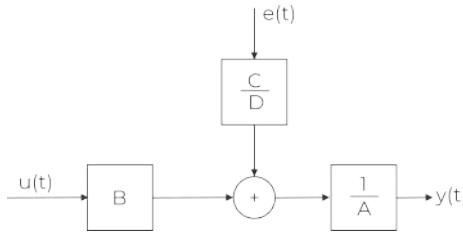


Figura 2.6: Diagrama a bloques de la estructura ARARMAX.

Modelo **General**. Siguiendo la lógica de representación por diagramas de bloques hemos visto que hay 5 polinomios en total del A al F, si hacemos las combinaciones serían 32 en total. El modelo que tiene todos los polinomios se le conoce como Modelo General, su ecuación queda así

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t). \tag{2.47}$$

Esta expresión es demasiado general para tener aplicación práctica, el modelo resultante es poco útil y eso sin contar en la gran complejidad para obtenerlo. Lo mas común es usar casos especiales o simplificados para situaciones particulares.

Modelo **FIR**. Es un tipo de modelo muy utilizado en análisis de señales como Acústica, finanzas, econometría, etc. Es muy simple ya que solo tiene un polinomio: B. Su ecuación es

$$y(t) = B(q)u(t). \tag{2.48}$$

Tiene dos ventajas importantes, es un regresor lineal por ser un caso especial de ARX y el error se refleja directamente en la salida, es un caso especial de OE. Esto significa que puede ser estimado eficientemente y es robusto contra el ruido.

Su desventaja principal es el numero de parámetros, no es raro decenas o hasta cientos de parámetros.

3. Modelos para sistemas en lazo cerrado

La identificación de sistemas en lazo cerrado es conveniente cuando la planta cuenta ya con un controlador y está operando o bien en el caso de una planta que en lazo abierto es inestable. De otra manera no es aconsejable utilizarla por la contaminación del error con el filtro del controlador, lo que numéricamente perturba la estimación de parámetros.

La ventaja de la identificación en lazo cerrado es que se pueden reajustar el controlador o el modelo con una segunda identificación.

Durante todo el capitulo los polinomios están dados en potencias negativas de q, a veces para facilitar la lectura en abuso de notación se omite en el nombre del polinomio expresar la potencia negativa, así se escribe $A(q)$ en vez de $A(q^{-1})$ pero todos son polinomios a potencias negativas.

3.1 Identificación en lazo Cerrado

Para poder identificar en lazo cerrado necesitamos conocer el controlador que está en operación. Sin perdida de generalidad supongamos un controlador tipo RST, donde R, S y T son polinomios definidos así:

$$\begin{aligned} S(q) &= [q^{-n_s} + s_{n_s-1}q^{n_s-1} + s_{n_s-2}q^{n_s-2} + \cdots + s_1 q^{-1} + s_0] \\ R(q) &= [r_{n_r}q^{-n_r} + r_{n_r-1}q^{n_r-1} + r_{n_r-2}q^{n_r-2} + \cdots + r_1 q^{-1} + r_0] \\ T(q) &= [t_{n_t}q^{-n_t} + t_{n_t-1}q^{n_t-1} + t_{n_t-2}q^{n_t-2} + \cdots + t_1 q^{-1} + t_0]. \end{aligned} \tag{3.1}$$

Un esquema general para identificación en lazo cerrado se muestra en la figura 3.1. La planta a ser identificada se indica como $G(t)$, notar que se incluye el controlador, está operando en lazo cerrado. En la misma figura se replica abajo en paralelo un sistema con el mismo controlador y con un modelo estimado de la planta $\hat{G}(t)$. La diferencia entre ambas salidas, llamado error en lazo cerrado ε_{LC}, sirve para ajustar el modelo mediante un Algoritmo de Adaptación Paramétrica (AAP).

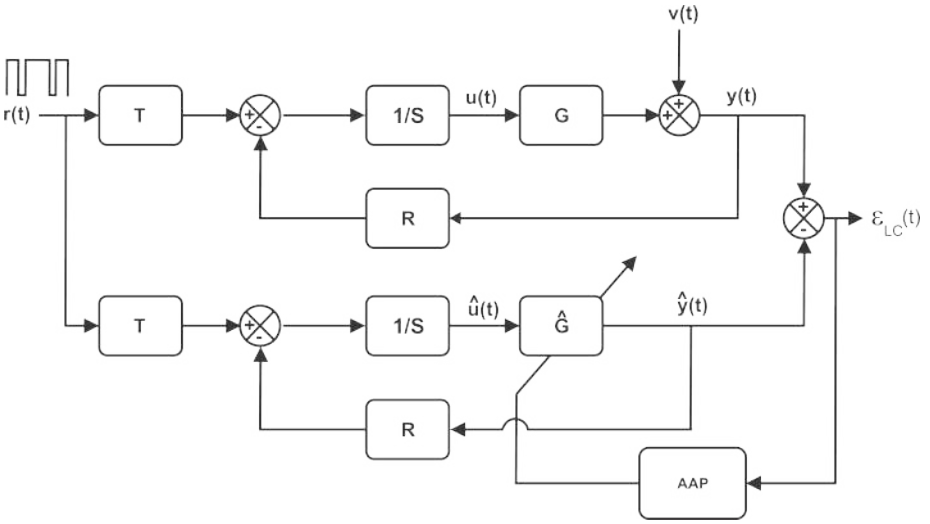


Figura 3.1: Diagrama a bloques de la estructura CLOE.

Las señales que vemos son: la entrada de control $u(t)$, la salida de la planta $y(t)$, la entrada al modelo estimado $\hat{u}$, la salida del modelo estimado $\hat{y}(t)$, la referencia para control $r(t)$ y la perturbación $v(t)$.

3.2 Estructura CLOE

La primera estructura por construcción obvia es la del Error de Salida en Lazo Cerrado, CLOE por sus siglas en inglés. Comenzamos por definir el error y el error de predicción a la salida como:

$$\begin{aligned} \varepsilon_{LC}(t) &= y(t) - \hat{y}(t) \\ \varepsilon_{LC}(t+1) &= y(t+1) - \hat{y}(t+1). \end{aligned} \tag{3.2}$$

La predicción de la salida *a priori*, basada en $\hat{\theta}(t)$ y la predicción de la salida *a posteriori* o sea basada en $\hat{\theta}(t+1)$ se definen en la ecuación 3.3. También se representan en su forma de mínimos cuadrados, ver ecuación 1.48.

$$\begin{aligned}\hat{y}^{\circ}(t+1) &= \hat{y}\left(t+1|\hat{\theta}(t)\right) = \hat{\theta}^{T}(t)\phi(t)\\ \hat{y}(t+1) &= \hat{y}\left(t+1|\hat{\theta}(t+1)\right) = \hat{\theta}^{T}(t+1)\phi(t)\end{aligned} \tag{3.3}$$

De la misma manera se definen los errores de predicción en lazo cerrado, *a priori* y *a posteriori*.

$$\begin{aligned}\varepsilon^{\circ}_{LC}(t+1) &= y(t+1) - \hat{y}^{\circ}(t+1)\\ \varepsilon_{LC}(t+1) &= y(t+1) - \hat{y}(t+1)\end{aligned} \tag{3.4}$$

Con estas definiciones y aplicando el algoritmo de ajuste paramétrico de mínimos cuadrados presentado en las ecuaciones 1.50-1.52, la estimación paramétrica queda como sigue:

$$\varepsilon_{LC}(t+1) = \frac{\varepsilon^{\circ}_{LC}(t+1)}{1+\phi^{T}(t)F(t)\phi(t)} \tag{3.5}$$

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F(t)\phi(t)\varepsilon_{LC}(t+1) \tag{3.5a}$$

$$F(t+1) = \frac{1}{\lambda_1(t)}\left[F(t) - \frac{F(t)\phi(t)\phi^{T}(t)F(t)}{\frac{\lambda_1(t)}{\lambda_2(t)} + \phi^{T}(t)F(t)\phi(t)}\right] \tag{3.5b}$$

El error *a posteriori* se calcula con el error *a priori*, ecuación 3.5, con este se actualiza el vector de parámetros 3.5a y finalmente se calcula la nueva ganancia de adaptación 3.5b, quedando todo listo para la nueva iteración.

Para el modelo de la planta usamos nuestro modelo de base, ecuación 2.1, en forma polinomial queda:

$$y(t) = G(t)u(t) = \frac{q^{-d}B(q^{-1})}{A(q^{-1})}u(t), \tag{3.6}$$

donde

$$\begin{aligned}d &= retardo\\ A(q) &= [1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_A-1}q^{-n_A-1} + a_{n_A}q^{-n_A}]\\ B(q) &= [b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_B-1}q^{-n_B-1} + b_{n_B}q^{-n_B}].\end{aligned} \tag{3.7}$$

En esta representación el retardo puro del sistema se hace explicito para evitar un vector con ceros lo que numéricamente no es deseable. Además el polinomio $A(q)$ es mónico y el polinomio $B(q)$ tiene un retardo unitario que es el tiempo mínimo de respuesta ante una excitación. Por esta razón es común encontrar una versión factorizada de los polinomios $A(q)$ y $B(q)$ que no tienen coeficientes constantes al inicio, ni ceros ni uno.

$$\begin{aligned} A(q) &= 1 + q^{-1}A^*(q^{-1}) \\ B(q) &= q^{-1}B^*(q^{-1}) \end{aligned} \tag{3.8}$$

De esta manera se tienen los polinomios alternativos que facilitan el calculo al no tener renglones o columnas que se anulen o sean constantes.

$$\begin{aligned} A^*(q) &= [a_1 + a_2 q^{-1} + \cdots + a_{n_A-1}q^{-n_A} + a_{n_A}q^{-n_A+1}] \\ B^*(q) &= [b_1 + b_2 q^{-1} + \cdots + b_{n_B-1}q^{-n_B} + b_{n_B}q^{-n_B+1}]. \end{aligned} \tag{3.9}$$

Los polinomios A^* y B^* contienen todos los parámetros del modelo por lo que es posible construir con ellos el vector de parámetros θ.

$$\theta^T = [a_1 \cdots a_{n_A} \quad b_1 \cdots b_{n_B}] \tag{3.10}$$

Dada las potencias de q que acompañan a los coeficientes y con la experiencia del capitulo anterior es fácil deducir el vector de datos de acompañamiento.

$$\phi^T(t) = [-y(t) \cdots -y(t-n_A+1) \quad u(t-d) \cdots u(t-n_B+1-d)] \tag{3.11}$$

Del modelo de la planta 3.6, pero considerando la perturbación como en el caso de *ARX*, la salida de la planta se calcula como:

$$A(q)y(t) = q^{-d}B(q)u(t) + A(q)v(t). \tag{3.12}$$

Sustituyendo A^* y B^* en los dos primeros términos:

$$(1+q^{-1}A^*(q))y(t) = q^{-d-1}B^*(q)u(t) + A(q)v(t).$$

Despejando la salida y avanzando un paso, es decir multiplicar por q toda la ecuación, llegamos a la expresión de la salida siguiente:

$$y(t+1) = -A^*(q)y(t) + B^*(q)u(t-d) + A(q)v(t+1), \tag{3.13}$$

como esperado tiene la misma forma que el vector de datos. Así, es posible escribir la misma ecuación como:

$$y(t+1) = \theta^T \phi(t) + A(q)v(t+1). \tag{3.14}$$

De manera similar se pude proceder con el estimado de la planta pero en este caso no hay perturbación, la salida del modelo estimado queda:

$$\hat{y}(t+1) = -\hat{A}^*(q)\hat{y}(t) + \hat{B}^*(q)\hat{u}(t-d) = \hat{\theta}^T \phi(t). \tag{3.15}$$

Con sus vectores de parámetros y de datos construidos de la misma manera.

$$\begin{aligned} \hat{\theta}^T &= [\hat{a}_1 \cdots \hat{a}_{n_A} \quad \hat{b}_1 \cdots \hat{b}_{n_B}] \\ \phi^T(t) &= [-\hat{y}(t) \cdots -\hat{y}(t-n_A+1) \quad \hat{u}(t-d) \cdots \hat{u}(t-n_B+1-d)] \end{aligned} \tag{3.16}$$

La entrada de control $u(t)$ dado el controlador RST es

$$u(t) = -\frac{R}{S}y(t) + \frac{T}{S}r(t). \tag{3.17}$$

Y la entrada de control para el modelo queda

$$\hat{u}(t) = -\frac{R}{S}\hat{y}(t) + \frac{T}{S}r(t). \tag{3.18}$$

3.3 Algoritmo CLOE

Para desarrollar un algoritmo de estimación de los parámetros del modelo la idea es expresar la salida en función del vector de datos y mediante mínimos cuadrados hacer la estimación.

La entrada de la planta $u(t)$ de la ecuación 3.17 la sustituimos en la ecuación de salida 3.13.

$$y(t+1) = \frac{q^{-d}B^*T}{S}r(t) - \left(A^* + \frac{q^{-d}B^*R}{S}\right)y(t) + Av(t+1)$$

Remplazamos $y(t)$ utilizando la ecuación 3.2.

$$y(t+1) = \frac{q^{-d}B^*T}{S}r(t) - \left(A^* + \frac{q^{-d}B^*R}{S}\right)\hat{y}(t) - \left(A^* + \frac{q^{-d}B^*R}{S}\right)\varepsilon_{LC}(t) + Av(t+1)$$

Sustituyendo la referencia de la ecuación 3.18 y cancelando un término de $\hat{y}(t)$.

$$y(t+1) = -A^*\hat{y}(t) + q^{-d}B^*\hat{u}(t) - \left(A^* + \frac{q^{-d}B^*R}{S}\right)\varepsilon_{LC}(t) + Av(t+1)$$

La primera parte de la ecuación puede ser producida por el producto del vector de parámetros θ y el vector de datos ϕ como se hizo para las estructuras en lazo abierto.

$$\theta^T\phi(t) = -A^*\hat{y}(t) + q^{-d}B^*\hat{u}(t) = -A^*\hat{y}(t) + B^*\hat{u}(t-d)$$

Finalmente la expresión para la salida queda:

$$y(t+1) = \theta^T\phi(t) - \left(A^* + \frac{q^{-d}B^*R}{S}\right)\varepsilon_{LC}(t) + Av(t+1). \qquad (3.19)$$

Con esto podemos calcular el error a posteriori de la ecuación 3.2 sustituyendo 3.19 y 3.15.

$$\varepsilon_{LC}(t+1) = \left(\theta - \hat{\theta}\right)^T \phi(t) - \left(A^* + \frac{q^{-d}B^*R}{S}\right)\varepsilon_{LC}(t) + Av(t+1)$$

Factorizando $\varepsilon_{LC}(t+1)$.

$$\varepsilon_{LC}(t+1)\left(\frac{AS + q^{-d}BR}{S}\right) = \left(\theta - \hat{\theta}\right)^T \phi(t) + Av(t+1)$$

Los polos en lazo cerrado son:

$$P = AS + q^{-d}BR.$$

Remplazando el error a posteriori queda de la siguiente manera:

$$\varepsilon_{LC}(t+1) = \frac{S}{P}\left(\theta - \hat{\theta}\right)^T \phi(t) + \frac{AS}{P}v(t+1). \tag{3.20}$$

La ecuación 3.20 es válida en un ambiente estocástico, para el caso determinístico (sin perturbación) tenemos:

$$\varepsilon_{LC}(t+1) = \frac{S}{P}\left(\theta - \hat{\theta}\right)^T \phi(t). \tag{3.21}$$

Con esto queda completo el algoritmo CLOE, para encontrar los coeficientes del modelo utilizando mínimos cuadrados con las ecuaciones 3.5, el vector de datos y el estimado de parámetros no sufren modificación, finalmente el error *a priori* queda:

$$\varepsilon_{LC}^{\circ}(t+1) = y(t+1) - \hat{\theta}^T(t)\phi(t).$$

3.4 CLOE + Perturbación extendida

Igual que se hizo para el caso de identificación en lazo abierto se procede para el lazo cerrado. Primero se estableció una estructura básica CLOE, ARX en el caso de lazo abierto y luego se mejoró la representación de las perturbaciones en el modelo. Esto dió origen a ARMAX en el lazo abierto y en el caso de lazo cerrado se presenta la estructura XCLOE.

Para el caso de la estructura XCLOE se considera un modelo de perturbación del tipo:

$$v(t) = \frac{C(q)}{A(q)}e(t).$$

Donde $e(t)$ es un ruido blanco gaussiano de media cero y varianza λ y $C(q)$ es un polinomio asintóticamente estable. Siguiedo el mismo procedimiento que para CLOE planteamos la ecuación de salida estimada como en 3.15.

$$\begin{aligned}\hat{y}(t+1) &= -\hat{A}^*(q)\hat{y}(t) + \hat{B}^*(q)\hat{u}(t-d) + \hat{H}^*\frac{\varepsilon_{LC}(t)}{S} \\ &= \hat{\theta}^T\phi(t) + \hat{H}^*\frac{\varepsilon_{LC}(t)}{S} \\ &= \hat{\theta}_e^T\phi_e(t) \end{aligned} \tag{3.22}$$

Donde:

$$\begin{aligned}\hat{H}^* &= q(CS - \hat{P}) \\ C(q^{-1}) &= 1 + C^*(q^{-1}) \\ C^*(q^{-1}) &= c_1 + c_2q^{-1} + c_3q^{-2} + \cdots + c_{n_C}q^{-n_C+1} \\ \hat{H}^*(q^{-1}) &= \hat{h}_1 + \hat{h}_2q^{-1} + \hat{h}_3q^{-2} + \cdots + \hat{h}_{n_H}q^{-n_H+1} \\ &= C^*S - \hat{A}^*S - q^{-d}\hat{B}^*R. \end{aligned} \tag{3.23}$$

Retomamos la expresión de la salida 3.19 y remplazamos los estimados que acabamos de presentar en las ecuaciones 3.22.

$$y(t+1) = \theta^T \phi(t) + \hat{H}^* \frac{\varepsilon_{LC}(t)}{S} - C^* \varepsilon_{LC}(t) + Ce(t+1). \tag{3.24}$$

Teniendo expresiones para la salida y salida estimada, ecuaciones 3.24 y 3.22, es posible establecer el error *a posteriori* para la estructura XCLOE.

$$\varepsilon_{LC}(t+1) = \frac{1}{C} \left(\theta_e - \hat{\theta}_e\right)^T \phi_e(t) + e(t+1) \tag{3.25}$$

Donde:

$$\begin{aligned} \theta_e^T &= \begin{bmatrix} \theta^T & h_1 & h_2 & \cdots & h_{n_H} \end{bmatrix} \\ \hat{\theta}_e^T &= \begin{bmatrix} \hat{\theta}^T & \hat{h}_1 & \hat{h}_2 & \cdots & \hat{h}_{n_H} \end{bmatrix} \\ \phi_e^T &= \begin{bmatrix} \phi^T(t) & \varepsilon_{LC_f}(t) & \cdots & \varepsilon_{LC_f}(t-n_H+1) \end{bmatrix} \\ \varepsilon_{LC_f}(t) &= \frac{1}{S} \varepsilon_{LC}(t). \end{aligned} \tag{3.26}$$

Para usar los mínimos cuadrados como algoritmo de ajuste paramétrico (AAP) en XCLOE queda de la siguiente manera: el vector de parámetros es $\hat{\theta}_e(t)$, el vector de datos es $\phi_e(t)$ y el error *a priori* esta dado en la ecuación 3.27.

$$\varepsilon_{LC}^{\circ}(t+1) = y(t+1) - \hat{\theta}_e^T(t) \phi_e(t) \tag{3.27}$$

II Modelos no lineales

4 Modelos usando redes neuronales . . . 47

4.1 Arquitectura de redes neuronales 47
4.2 Red Neuronal - Ejemplo simple 50
4.3 Red Neuronal - Especial SYSID 54

5 Modelos a bloques: Hammerstein-Wiener 61

5.1 Modelos a Bloques 61
5.2 Modelo Hammerstein-Wiener 63
5.3 Hammerstein-Wiener: Sobreparametrización . . 64
5.4 Hammerstein-Wiener: Iterativo 67

4. Modelos usando redes neuronales

Igual que en otras áreas de la Ingeniería la inspiración para nuevas ideas viene de la observación de la naturaleza. Las redes neuronales artificiales son un caso de éxito, han mostrado su utilidad en modelado y clasificación de la información en diferentes campos de aplicación: medicina, economía e ingeniería, entre otros.

4.1 Arquitectura de redes neuronales

Una neurona artificial, simil de una biológica puede ser representada como en la figura 4.1. Las entradas (u_i) reciben información externa o de otras neuronas, equivalente de las dendritas en las neuronas biológicas. La información recibida es ponderada por los pesos sinápticos (w_i), equivalente a la sinapsis. La suma de las entradas ponderadas y un nivel de sesgo, conocido como *bias* (θ) por su nombre en inglés, es procesada por una función dicha de activación (φ), típicamente lineal, sigmoidal o gaussiana. finalmente la salida (y), axón en una biológica, conduce la información a otras neuronas o al exterior.

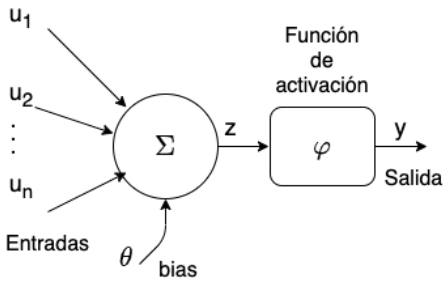


Figura 4.1: Neurona Artificial.

Siguiendo este esquema la representación matemática de la neurona artificial

presentada es la siguiente:

$$y = \varphi\left(\sum_{i=1}^{n} w_i u_i + \theta\right). \tag{4.1}$$

Una neurona artificial es la unidad de base, igual que en el caso biológico, es el agrupamiento de mucas neuronas lo que le confiere esa capacidad de procesamiento. Es posible agruparlas de muy diferentes maneras conocidas como arquitecturas de redes neuronales.

Ejemplos de arquitecturas simples son: red en adelanto, red con retorno y red celular. La red en adelanto consiste en colocar una neurona después de otra en cascada, es un tipo de red estática muy rápida. La red realimentada tiene la ventaja de la corrección que otorga el regreso de información. Una red celular consiste en una agrupación local de varias neuronas que se encuentran todas interconectada entre ellas.

En nuestro caso la producción de modelos con agrupamientos de redes neuronales artificiales presenta la ventaja de poder aproximar funciones no lineales con la elección de la función de activación mientras que los pesos sinápticos son ajustados por algún algoritmo de aprendizaje durante la fase de entrenamiento. La arquitectura más utilizada es una mezcla de redes de adelanto y realimentación que permite tener ambas propiedades. Típicamente se arreglan en capas, donde la 0 sería la capa de entrada, sucesivamente las capas posteriores $1,2,\ldots,n-1$ serían las capas ocultas y la última la capa $n-1$ es la capa de salida.

Tenemos dos parámetros: el numero de neuronas en cada capa y el numero de capas, entre mas grandes sean mayor capacidad de aproximación pero también se incrementa la complejidad y el costo computacional. Ha sido mostrado que una gran variedad de sistemas pueden ser representados con poco error con solamente dos o tres capas de neuronas por lo que son los valores de inicio más usados [7] y [9]. En cuanto al numero de neuronas hay una gran variedad de enfoques para determinar el mejor compromiso entre precisión y complejidad [20].

La elección de las funciones de activación le otorga a la red neuronal las capacidades para modelar sistemas complejos, la combinación de varios tipos es ampliamente utilizado al combinar sus ventajas, así por ejemplo es común que en las capas de entrada y la de salida se utilicen funciones de activación lineales mientras que en las capas ocultas se utilizan funciones de activación no

lineales [6]. En cuanto a las funciones de activación no lineales es típico usar funciones de saturación para evitar un crecimiento desbordado, un ejemplo muy usado es la función tanh por la ventaja de su derivada simple que se aprovecha para optimizar los valores de los pesos sinápticos con cualquier método de optimización basado en el gradiente.

Para ajustar los pesos sinápticos la red neuronal es sometida a un proceso llamado de entrenamiento. La red neuronal recibe como entrada un vector (regresor) de entradas pasadas de dimensión $[u(t-1),\ldots,u(t-n_b)]$ y con la salida se hace lo mismo, un regresor de salidas pasadas aplicado a las neuronas de entrada $[y(t-1),\ldots,y(t-n_a)]$, de tal manera que $n_a+n_b =$ al numero total de pesos sinápticos en la capa de entrada. La red estima el valor actual de la salida $\hat{y}(t)$ para el instante de tiempo t y comparado con el valor actual de la salida $y(t)$ podemos calcular el error de predicción $e(t)$.

Para ajustar los pesos sinápticos de la red ya definida se usa una algoritmo de aprendizaje que de una manera iterativa va ajustando los pesos minimizando una función objetivo muchas veces en función del error de predicción. La elección del algoritmo depende de factores deseados como la robustez, rápida convergencia, costo, insensibilidad al cambio de parámetros o tolerancia a errores de medición, entre otros. Para muchos casos el algoritmo mas simple, mínimos cuadrados es suficiente. Así para una tira de datos de tamaño N y un vector de pesos sinápticos w con error de predicción de $e(t,\hat{w}) = y(t) - \hat{y}(t,\hat{w})$ el criterio cuadrático queda:

$$E(w,u^N,y^N) = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{2}e^2(t,\hat{w}). \tag{4.2}$$

En una forma mas general un algoritmo para ajustar los pesos sinápticos w tiene la forma:

$$\hat{w}(t+1) = \hat{w}(t) - \eta[R]^{-1}\frac{\partial E}{\partial \hat{w}}. \tag{4.3}$$

R es para ajustar la dirección de busqueda, η es el paso del aprendizaje, $\frac{\partial E}{\partial \hat{w}}$ es el gradiente de la función objetivo y $\hat{w}(t)$ es el estimado de los pesos sinápticos en el instante t.

Eligiendo convenientemente R obtenemos algunos algoritmos conocidos como Gauss-Newton o Levenberg-Marquardt, en el caso más simple cuando

$R = 1$ recuperamos el algoritmo más sencillo utilizado en las redes neuronales, el algoritmo del gradiente descendente.

$$\hat{w}(t+1) = \hat{w}(t) - \eta \frac{\partial E}{\partial \hat{w}} \tag{4.4}$$

Dado un conjunto de datos de tamaño N, el estimado de los pesos sinápticos se calcula para cada t de 1 hasta N, cuando se llega a N se agotan los datos y se dice que se completó una época (γ). Los mismos datos se vuelven a utilizar para un nuevo entrenamiento (época) pero tomando como valores iniciales de los pesos sinápticos estimados los últimos de la época anterior. Así sucesivamente durante varias épocas, a cada pasada normalmente el error va disminuyendo de forma exponencial decreciente y solo queda decidir cuando parar, normalmente cuando ya no haya una disminución importante del error, a criterio del entrenador de la red. El parámetro η está relacionado con la velocidad de convergencia de la estimación, para valores pequeños es lenta y para valores grandes puede llegar más rápido pero también puede oscilar. Por esta razón hay diferentes propuestas para modificar η al pasar las etapas, en un principio un valor inicial que disminuye con el numero de épocas [16].

4.2 Red Neuronal - Ejemplo simple

En esta sección se presenta una red neuronal básica pero funcional para muchos sistemas sencillos, la idea es mostrar el modelo matemático sin generalidades para que sea fácil de leer y darle seguimiento. La red en cuestión tiene dos capas, en la entrada $n_0 = 2$ neuronas y una sola neurona en la capa de salida puesto que se tiene una sola salida ($n_1 = 1$). El tamaño del regresor en la entrada es elección del usuario, buscando el balance entre simpleza y precisión. La figura 4.2 muestra está red.

Para construir el modelo matemático partimos de la salida, de derecha a izquierda. Primero la función de activación de la segunda capa, luego la suma ponderada por los pesos sinápticos de la segunda capa y por último las dos sumas ponderadas de los regresores de entradas y salidas pasadas.

$$\hat{y}(t) = \varphi_3(T) \tag{4.5}$$

$$T = \sum_{p=1}^{n_0} z_p \varphi_1(r_p) \tag{4.6}$$

$$r_p = \sum_{i=1}^{n_b} (w_{p,i} u(t-i)) + \sum_{j=1}^{na} (w_{p,i+j} y(t-j)) \tag{4.7}$$

Donde $u(t-i)$ y $y(t-j)$ son la entrada y salida respectivamente para una tira de N datos, por tanto $1 \leq t \leq N$, dependiendo de cuantas entradas y salidas pasadas se quieran tomar, eso define el tamaño del regresor, $1 \leq i \leq n_b$ las entradas pasadas y $1 \leq j \leq n_a$ las salidas pasadas y, el numero de neuronas en la capa de entrada $1 \leq p \leq n_0$.

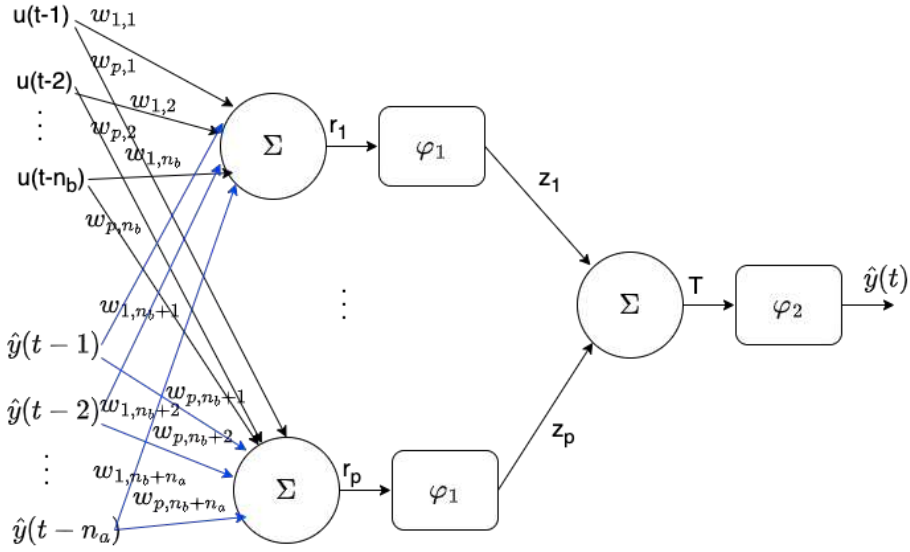


Figura 4.2: Red Neuronal de 2 capas.

Lo siguiente es elegir las funciones de activación, dependiendo de esta elección será el tipo y características del modelo obtenido. En este ejemplo vamos a mostrar 3 modelos diferentes. Para el primer modelo elegimos lo mas sencillo, ambas funciones de activación lineales $\varphi_1(x) = \varphi_2(x) = x$, modelo 4.5-4.7 queda:

$$\begin{aligned} \hat{y}(t) &= T \\ T &= \sum_{p=1}^{n_0} (z_p r_p) \\ r_p &= \sum_{i=1}^{n_b} (w_{p,i} u(t-i)) + \sum_{j=1}^{na} (w_{p,i+j} y(t-j)). \end{aligned} \tag{4.8}$$

Para el caso de una función de activación lineal en la capa de entrada ($\varphi_1(x) = x$) y una no lineal para la salida $\varphi_2(x) = nolineal$, el modelo 4.5-4.7 toma la forma:

$$\begin{aligned} \hat{y}(t) &= \varphi_2(T) \\ T &= \sum_{p=1}^{n_0} (z_p r_p) \\ r_p &= \sum_{i=1}^{n_b} (w_{p,i} u(t-i)) + \sum_{j=1}^{na} (w_{p,i+j} y(t-j)). \end{aligned} \tag{4.9}$$

La tercer forma es con una función de activación no lineal en la capa de entrada ($\varphi_1(x) = nolineal$) y lineal en la capa de salida ($\varphi_3(x) = x$).

$$\begin{aligned} \hat{y}(t) &= T \\ T &= \sum_{p=1}^{n_0} (z_p \varphi_1(r_p)) \\ r_p &= \sum_{i=1}^{n_b} (w_{p,i} u(t-i)) + \sum_{j=1}^{na} (w_{p,i+j} y(t-j)). \end{aligned} \tag{4.10}$$

Seamos específicos para mostrar el detalle, por ejemplo el tercer modelo ecuación 4.10 con $\varphi_1(x) = \tanh(x)$, dos neuronas en la primera capa ($n_0 = 2$) y un regresor con dos entradas pasadas ($n_b = 2$) y una salida pasada ($n_a = 1$). El modelo es el siguiente:

$$\begin{aligned} \hat{y}(t) &= T \\ T &= z_1 \tanh(r_1) + z_2 \tanh(r_2) \\ r_1 &= w_{1,1} u(t-1) + w_{1,2} u(t-2) + w_{1,3} y(t-1) \\ r_2 &= +w_{2,1} u(t-1) + w_{2,2} u(t-2) + w_{2,3} y(t-1). \end{aligned} \tag{4.11}$$

También es posible escribir el modelo con el regresor en este caso $J = [u(t-1) \quad u(t-2) \quad y(t-1)]$.

$$\begin{aligned} \hat{y}(t) &= T \\ T &= \sum_{p=1}^{2} z_p \tanh(r_p) \\ r_p &= \sum_{q=1}^{n_a+n_b} w_{p,q} J(q) \end{aligned} \tag{4.12}$$

Ya tenemos completamente definido el modelo, la siguiente etapa es el entrenamiento de la red para calcular los pesos sinápticos. Necesitamos definir una función objetivo, lo más sencillo es optimizar el error cuadrático $e(t) = y(t) - \hat{y}(t)$.

$$E(w_{p,q}, z_p) = \frac{1}{N}\sum_{t=1}^{N}\frac{1}{2}e^2(t, w_{p,q}, z_p) \tag{4.13}$$

Para el ajuste de los pesos sinápticos $w_{p,q}$ y z_p empleamos el algoritmo de gradiente descendente.

$$\begin{aligned} \hat{w}_{p,q}(t+1) &= \hat{w}_{p,q}(t) - \eta\frac{\partial E}{\partial \hat{w}_{p,q}} \\ \hat{z}_p(t+1) &= \hat{z}_p(t) - \eta\frac{\partial E}{\partial \hat{z}_p} \end{aligned} \tag{4.14}$$

Para η se elige un valor inicial (η_0) por el usuario y se hace diminuir exponencialmente después de cada época de entrenamiento. Las derivades parciales $\frac{\partial E}{\partial \hat{w}_{p,q}}$ y $\frac{\partial E}{\partial \hat{z}_p}$ se calculan con la regla de la cadena.

$$\begin{aligned} \frac{\partial E}{\partial \hat{w}_{p,q}} &= \frac{\partial E}{\partial \hat{e}}\frac{\partial e}{\partial \hat{\hat{y}}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial r_p}\frac{\partial r_p}{\partial \hat{w}_{p,q}} \\ \frac{\partial E}{\partial \hat{z}_p} &= \frac{\partial E}{\partial \hat{e}}\frac{\partial e}{\partial \hat{\hat{y}}}\frac{\partial \hat{y}}{\partial T}\frac{\partial T}{\partial z_p} \end{aligned} \tag{4.15}$$

En este caso el calculo de las derivadas arroja:

$$\begin{aligned} \frac{\partial E}{\partial \hat{e}} &= e(k) \\ \frac{\partial e}{\partial \hat{\hat{y}}} &= -1 \\ \frac{\partial \hat{y}}{\partial T} &= 1 \\ \frac{\partial T}{\partial r_p} &= z_p\ sech^2(r_p) \\ \frac{\partial r_p}{\partial \hat{w}_{p,q}} &= J(q) \\ \frac{\partial T}{\partial z_p} &= \tanh(r_p). \end{aligned}$$

Remplazando el calculo de las derivadas en la ecuación 4.14 la expresión para los pesos sinápticos es:

$$\begin{aligned} \hat{w}_{p,q}(t+1) &= \hat{w}_{p,q}(t) + \eta\ e(t)\ z_p\ sech^2(r_p)\ J(q) \\ \hat{z}_p(t+1) &= \hat{z}_p(t) + \eta\ e(t)\ \tanh(r_p) \end{aligned} \tag{4.16}$$

El costo computacional final depende del algoritmo de aprendizaje, en este caso gradiente descendente, la función objetivo elegida, en este caso el error cuadrático y de la arquitectura de la red, en este caso dos capas con dos neuronas en la entrada y una en la salida y, un regresor de dimensión 3.

4.3 Red Neuronal - Especial SYSID

En este sección se muestra una arquitectura de red neuronal especial para modelado tipo identificación de sistemas, fue presentada en [11, 12, 13]. Es una red de tres capas, la primera capa tiene $2n$ neuronas, la segunda capa, la oculta, tiene dos neuronas y la tercer capa tiene solo una neurona para la salida, por lo que la llamamos red neuronal 2n-2-1, se muestra en la figura 4.3. Esta estructura en apariencia sencilla puede producir diferentes tipos de modelos para sistemas lineales y no lineales, cubriendo un largo espectro de aplicaciones, además de que permite la reducción del costo computacional con una simplificación. El tamaño del regresor y la cantidad de neuronas en la primer capa son los parámetros de ajuste del usuario, en cuanto a la arquitectura, porque también es posible modificar la función objetivo y el algoritmo de ajuste de los pesos sinápticos.
De acuerdo con la figura 4.3 y siguiendo los mismos pasos que en la sección anterior el modelo es el siguiente:

$$\begin{aligned} \hat{y}(t) &= X\,\varphi_3(T) \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= \sum_{i=1}^{n} V_{b_i}\varphi_1(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{n} V_{a_i}\varphi_1(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.17}$$

El regresor para las entradas $J_u = [u(t-1)\ u(t-2)\ldots u(t-n_b)]$, el regresor para las salida $J_{\hat{y}} = [\hat{y}(t-1)\ \hat{y}(t-2)\ldots\hat{y}(t-n_a)]$, el vector de pesos sinápticos para la entrada $W_{b_i} = [W_{b_{i,1}}\ W_{b_{i,2}}\ldots W_{b_{i,n_b}}]^\top$ y el vector de pesos sinápticos para

la salida $W_{a_i} = [W_{a_{i,1}}\ W_{a_{i,2}} \ldots W_{a_{i,na}}]^\top$. La lista de todos los pesos sinápticos: X, Z_b, Z_a, V_{b_i}, V_{a_i}, W_{b_i}, W_{a_i}, Z_h para $1 \leq i \leq n$ neuronas.

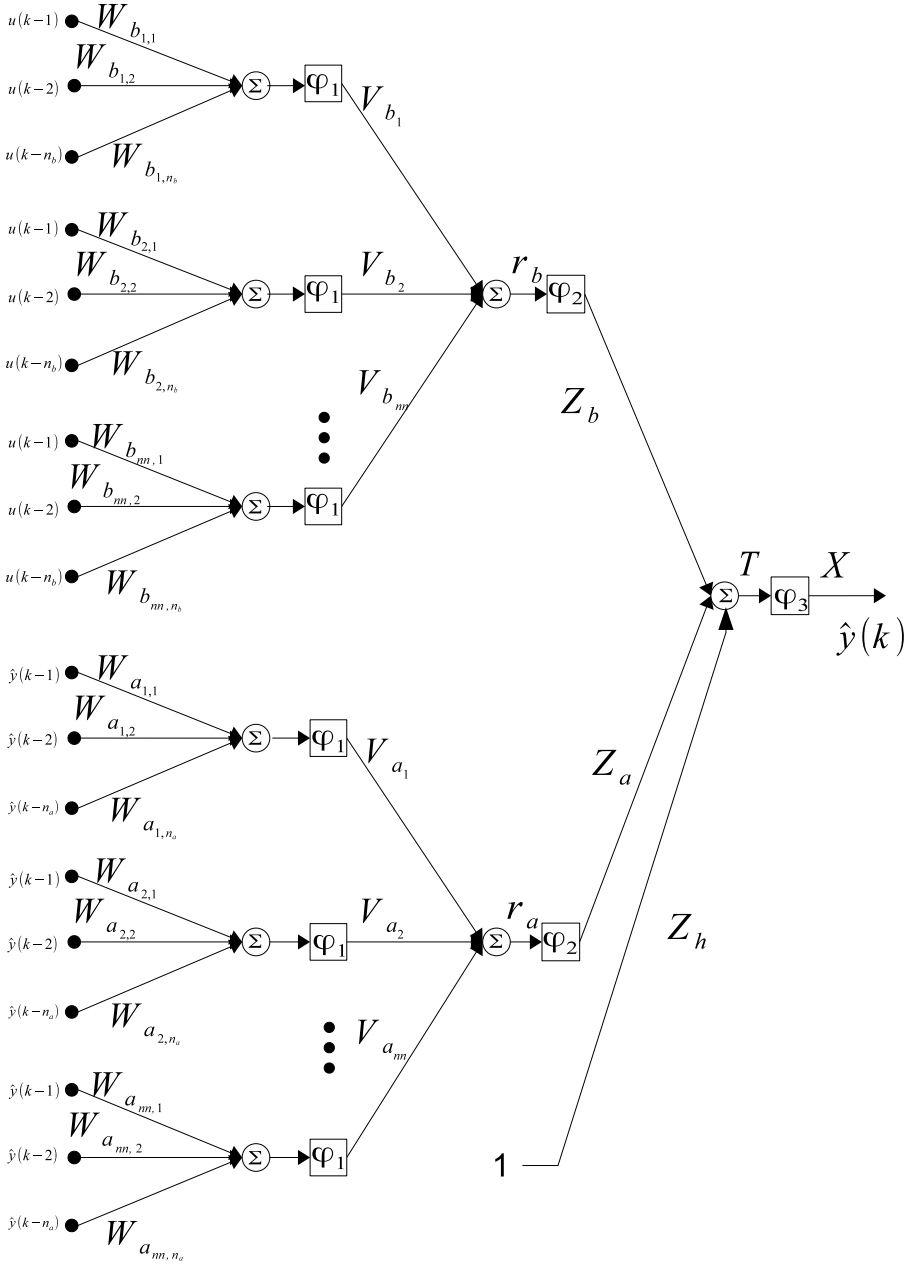


Figura 4.3: Red Neuronal Artificial 2n-2-1.

Eligiendo diferentes combinaciones de funciones de activación podemos tener diferentes tipos de modelos. Enseguida se muestran 4 familias de modelos que se pueden producir.

Primer modelo con funciones de activación no lineales en la capa de entrada y lineales en las otras dos $\varphi_3(z) = \varphi_2(z) = z$.

$$\begin{aligned} \hat{y}(t) &= X\ T \\ T &= Z_b\ r_b + Z_a\ r_a + Z_h \\ r_b &= \sum_{i=1}^{n} V_{b_i} \varphi_1(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{n} V_{a_i} \varphi_1(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.18}$$

Al tener la capa de entrada función de activación no lineal este tipo de modelo es apropiado para problemas de control no lineal.

El segundo tipo de modelo es el clásico con capa oculta no lineal, entonces las capas uno y tres son lineales $\varphi_3(z) = \varphi_1(z) = z$.

$$\begin{aligned} \hat{y}(t) &= X\ T \\ T &= Z_b \varphi_2(r_b) + Z_a \varphi_2(r_a) + Z_h \\ r_b &= \sum_{i=1}^{n} V_{b_i} (J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{n} V_{a_i} (J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.19}$$

Es la estructura clásica que se usa de manera general ya que la capa oculta no lineal le da una capacidad de ajuste de amplio espectro.

El tercer tipo de modelo se obtiene al considerar todas las funciones de activación lineales $\varphi_1(z) = \varphi_2(z) = \varphi_3(z) = z$ lo que viene a ser un modelo lineal tipo ARX.

$$\begin{aligned} \hat{y}(t) &= X\ T \\ T &= Z_b\ r_b + Z_a\ r_a + Z_h \\ r_b &= \sum_{i=1}^{n} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{n} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.20}$$

De esta manera es posible de tener modelos lineales en redes neuronales.

El cuarto tipo de modelos tiene una función de activación no lineal solo en la neurona de salida, es representativo de varios sistemas que presentan esta característica, electromecánicos y procesos por ejemplo, entonces la primera y segunda capa son lineales $\varphi_1(z) = \varphi_2(z) = z$.

$$\begin{aligned} \hat{y}(t) &= X\varphi_3(T) \\ T &= Z_b\ r_b + Z_a\ r_a + Z_h \\ r_b &= \sum_{i=1}^{n} V_{b_i}(J_u W_{b_i}) \\ r_a &= \sum_{i=1}^{n} V_{a_i}(J_{\hat{y}} W_{a_i}) \end{aligned} \tag{4.21}$$

Tomando como base la red neuronal propuesta 2n-2-1 es posible definir una versión simplificada cuyo costo computacional es significativamente menor conservando buenas características de ajuste. Presentamos la red neuronal 2-2-1 como se muestra en la figura 4.4, el numero de neuronas en la capa de entrada queda fijo en 2.

El modelo matemático es el siguiente:

$$\begin{aligned} \hat{y}(t) &= X\varphi_3(T) \\ T &= Z_b\varphi_2(r_b) + Z_a\varphi_2(r_a) + Z_h \\ r_b &= n \times V_{b_1}\varphi_1(J_u W_{b_1}) \\ r_a &= n \times V_{a_1}\varphi_1(J_{\hat{y}} W_{a_1}). \end{aligned} \tag{4.22}$$

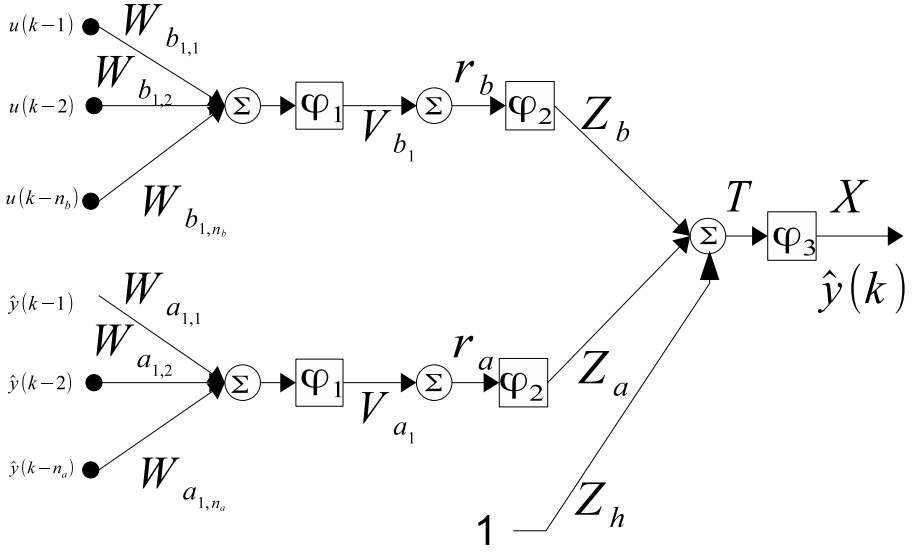


Figura 4.4: Red Neuronal Artificial 2-2-1.

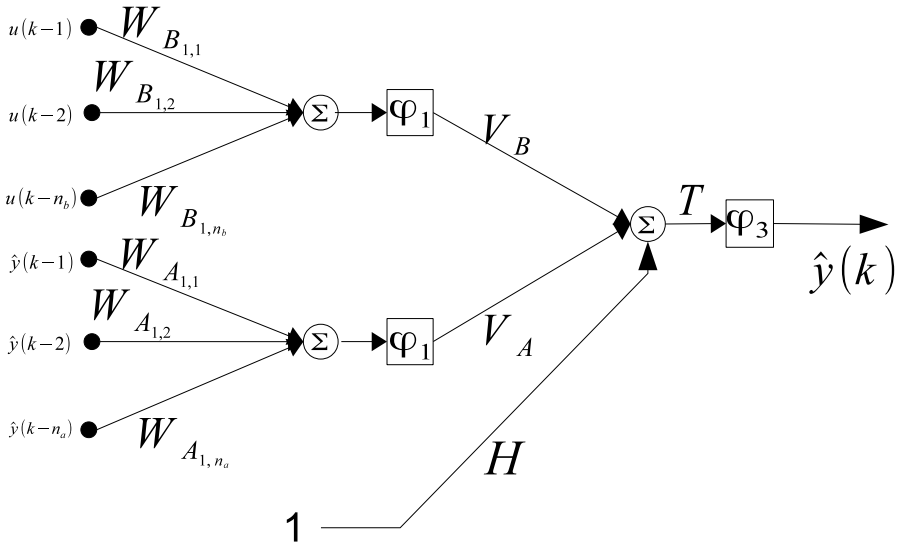


Figura 4.5: Red Neuronal Artificial 2-1.

Los pesos sinápticos son: $X, Z_b, Z_a, V_{b_1}, V_{a_1}, Z_h$ con $W_{b_1} = [W_{b_1}\ W_{b_2}\ \dots\ W_{b_{n_b}}]^\top$ y $W_{a_1} = [W_{a_1}\ W_{a_2}\ \dots\ W_{a_{n_a}}]^\top$.

Eligiendo apropiadamente las funciones de activación del modelo 4.17 es posible transformar el computo a esta red 2-2-1 guardando sus capacidades de representación pero con bajo costo computacional [11, 12, 13].

Finalmente es posible todavía reducir la complejidad y por tanto el costo computacional, en algunos casos una red sencilla es suficiente, la red 2-1 aquí representada en la figura x.

Esta red, compuesta unicamente de dos capas y tres neuronas, una para la salida otra se ocupa del regresor de las entradas pasadas y la última para el regresor de las salidas pasadas. Su modelos es como sigue:

$$\begin{aligned} \hat{y}(t) &= \varphi_3(T) \\ T &= V_B\varphi_1(J_u W_B) + V_A\varphi_1(J_{\hat{y}} W_A) + H \end{aligned} \tag{4.23}$$

Donde $W_B = [W_{B_{1,1}}\ W_{B_{1,2}}\ \dots\ W_{B_{1,n_b}}]^\top$ y $W_A = [W_{A_{1,1}}\ A_{1,2}\ \dots\ W_{A_{1,n_a}}]^\top$.

La velocidad de respuesta, la simplicidad de calculo hacen una buena opción de esta red para aplicaciones en linea.

En resumen la red neuronal propuesta en la ecuación 4.17 tiene una capacidad de aproximación buena para la mayoría de las aplicaciones, una versión que mejora considerablemente el costo computacional y en algunos casos sin perdida de poder de aproximación es la red 4.22 y por último una pequeña red muy simplificada pero útil por sus bajos requerimientos y aptitud para trabajar en tiempo real es la representada en 4.23.

5. Modelos a bloques: Hammerstein-Wiener

Una manera muy exitosa de tratar los sistemas no lineales es a través de los diagramas a bloques. Por un lado deja la parte no lineal como un componente estático y por otro lado facilita la estimación y uso del modelo. Construir un modelo puramente no lineal y variante en el tiempo puede ser un reto que demande una enorme cantidad de recursos para que al final tengamos un producto inexplotable.

5.1 Modelos a Bloques

Un modelo Hammerstein es un sistema de dos bloques que representa una familia de sistemas no lineales, aquellos representables por un bloque no lineal estático y un bloque lineal dinámico, como se muestra en la figura 5.1.

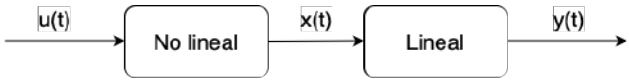


Figura 5.1: Modelo Hammerstein.

Existe también la estructura opuesta, es decir primero un bloque dinámico lineal y después un bloque estático no lineal, es conocido como Modelo Wiener, se muestra en la figura 5.2.

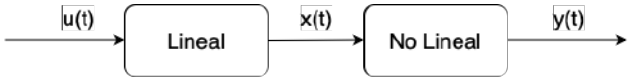


Figura 5.2: Modelo Wiener.

La combinación de ambos modelos se conoce como Hammerstein-Wiener, combina las capacidades de modelado pero se vuelve un modelo mas complejo, más difícil de manipular, representado en la figura 5.3.

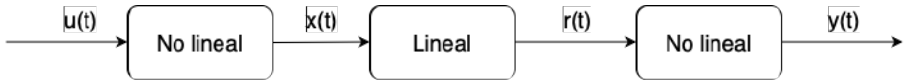


Figura 5.3: Modelo Hammerstein-Wiener.

Para identificar un modelo Hammerstein-Wiener se necesita en realidad construir tres partes o bloques por lo que no hay una manera única de hacerlo. Entre los métodos utilizados podemos listar: el enfoque iterativo, el enfoque por sobreparametrización, los métodos en el dominio de la frecuencia, los métodos por subespacios y los algoritmos estocásticos.

Construyendo el modelo bloque a bloque, el primero y el tercero son funciones de sus entradas únicamente, por lo que solo cuenta el valor presente en la entrada, no hay memoria, no hay dinámica temporal. En cambio el bloque de en medio es un sistema dinámico lineal típicamente representado por una función de transferencia o una descripción en espacio de estados, ver figura 5.4.

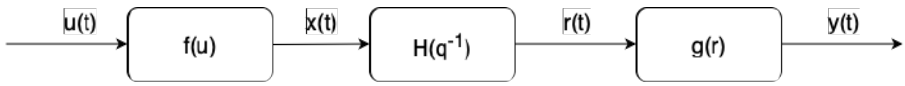


Figura 5.4: Modelo Hammerstein-Wiener con funciones.

El primer bloque es una función de la entrada $x(t) = f(u(t))$ y está función puede ser representada como una adición de funciones básicas:

$$x(t) = \sum_{t=1}^{m} c_l f_l(u(t)), \tag{5.1}$$

donde:
- m es el número de funciones básicas usadas.
- $u(t)$ es la señal de entrada en el tiempo t.
- c_l son los coeficientes de las funciones básicas.
- f_l son las funciones básicas con respecto a la entrada.

Para el bloque de salida hay que hacer lo mismo $y(t) = g(r(t))$ pero dado que los datos que tenemos son de la salida la función g debe ser invertible

$r(t) = g^{-1}(y(t))$ y es sobre esta que se hace la descomposición en funciones básicas.

$$r(t) = \sum_{l=1}^{p} d_l g_l(y(t)) \tag{5.2}$$

Donde:
- p es el número de funciones básicas usadas.
- $y(t)$ es la señal de salida en el tiempo t.
- d_l son los coeficientes de las funciones básicas.
- g_l son las funciones básicas con respecto a la salida.

Para el bloque de en medio, un sistema lineal invariante en el tiempo, usamos la representación de base que planteamos en el capitulo de sistemas lineales ecuación 2.1, en este caso la función de transferencia en operador q^{-1} es $H(q^{-1})$ y queda representado así:

$$H(q^{-1}) = \frac{b_o + b_1 q^{-1} + b_2 q^{-2} + ... + b_{n_b} q^{-n_b}}{a_o - a_1 q^{-1} - a_2 q^{-2} - ... - a_{n_a} q^{-n_a}} = \frac{B(q^{-1})}{A(q^{-1})}, \tag{5.3}$$

donde:
- n_a es el orden del denominador.
- n_b es el orden del numerador.

5.2 Modelo Hammerstein-Wiener

Para un modelo Hammerstein-Wiener no hay una manera única de identificar los tres bloques y es posible describirlo de muchas maneras es por eso que se tienen varios métodos de estimación para los parámetros, algunos ya citados arriba, la realización mostrada aquí es la presentada en [4, 5], separando el bloque lineal y considerando una fuente de ruido aditivo $v(t)$.

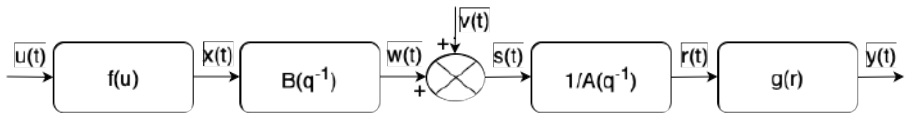


Figura 5.5: Realización del Modelo Hammerstein-Wiener.

El modelo para esta realización considerando las descomposición en funciones básicas como descrito arriba queda:

$$y(t) = \sum_{i=1}^{n_a} a_i \left\{ \sum_{l=1}^{p} d_l g_l[y(t-i)] \right\} + \sum_{j=0}^{n_b} b_j \left\{ \sum_{l=1}^{m} c_l f_l[u(t-j)] \right\} + v(t), \quad (5.4)$$

donde el primero de los dos bloques de sumatoria corresponde a las funciones de representación de la salida pero filtrada por el denominador de la función de transferencia y el segundo bloque sumador corresponde al bloque no lineal de la entrada que pasa a ser filtrado por numerador de la función de transferencia.

El modelo para esta estuctura es:

$$y(t) = g(H(q)f(u(t))), \quad (5.5)$$

claramente estamos ante un problema de optimización no lineal. De acuerdo con la figura 5.5 podemos partir en dos sistemas teniendo como señal intermedia $s(t)$, primero en función de la señal de entrada:

$$s(t) = \sum_{j=0}^{n_b} b_j \left\{ \sum_{l=1}^{m} c_l f_l[u(t-j)] \right\} + v(t), \quad (5.6)$$

luego en función de la salida:

$$s(t) = \sum_{l=1}^{p} a_0 d_l g_l[y(t)] + \sum_{i=1}^{n_a} a_i \left\{ \sum_{l=1}^{p} d_l g_l[y(t-i)] \right\}. \quad (5.7)$$

Igualando las ecuaciones 5.6 y 5.7, dejando todos los miembros de un solo lado y metiendo todos los términos a las sumatorias tenemos:

$$\sum_{j=0}^{n_b} \left\{ \sum_{l=1}^{m} b_j c_l f_l[u(t-j)] \right\} - \sum_{l=1}^{p} a_0 d_l g_l[y(t)] - \sum_{i=1}^{n_a} \left\{ \sum_{l=1}^{p} a_i d_l g_l[y(t-i)] \right\} + v(t) = 0. \quad (5.8)$$

Así vemos que se trata de una expresión bilineal para los parametros a, b, c y d.

5.3 Hammerstein-Wiener: Sobreparametrización

Un primer método que permite atacar la bilinealidad es conocido como sobreparametrización, aplicado a modelos Hammerstein-Wiener lo podemos ver en [5]. La ecuación 5.8 puede ser expresada en una forma sobreparametrizada, es decir en una forma matricial y lineal en los parámetros.

$$
\begin{aligned}
0 = &\sum_{j=0}^{n_b}\left\{\sum_{l=1}^{m} b_j c_l f_l[u(t-j)]\right\} - \sum_{l=1}^{p} a_0 d_l g_l[y(t)] \\
&- \sum_{i=1}^{n_a}\left\{\sum_{l=1}^{p} a_i d_l g_l[y(t-i)]\right\} + v(t) \\
0 = &\Phi(t)\theta + v(t)
\end{aligned} \tag{5.9}
$$

Donde $\Phi(t)$ y θ quedan de la siguiente manera:

$$
\Phi(t) = \begin{bmatrix}\Phi_1(t) & \Phi_2(t) & \Phi_3(t) & \Phi_4(t)\end{bmatrix}
$$

y

$$
\theta = \begin{bmatrix}
b_0c_1 & b_0c_2 & \cdots & b_0c_m & b_1c_1 & b_1c_2 & \cdots & b_1c_m \\
b_{n_b}c_1 & b_{n_b}c_2 & \cdots & b_{n_b}c_m & a_0d_1 & a_0d_2 & \cdots & a_0d_p \\
a_1d_1 & a_1d_2 & \cdots & a_1d_p & a_{n_a}d_1 & a_{n_a}d_2 & \cdots & a_{n_a}d_p
\end{bmatrix}^T.
$$

Desarrollando $\Phi(t)$ tenemos:

$$
\Phi_1(t) = \begin{bmatrix} f_1(u(t)) & f_2(u(t)) & \cdots & f_m(u(t))\end{bmatrix}, \tag{5.10}
$$

$$
\Phi_2(t) = \begin{bmatrix}
f_1(u(t-1)) & f_2(u(t-1)) & \cdots & f_m(u(t-1)) \\
f_1(u(t-2)) & f_2(u(t-2)) & \cdots & f_m(u(t-2)) \\
\cdots & \cdots & \cdots & \cdots \\
f_1(u(t-n_b)) & f_2(u(t-n_b)) & \cdots & f_m(u(t-n_b))],
\end{bmatrix} \tag{5.11}
$$

$$
\Phi_3(t) = \begin{bmatrix} -g_1(y(t)) & -g_2(y(t)) & \cdots & -g_p(y(t))\end{bmatrix} \tag{5.12}
$$

y

$$
\Phi_4(t) = \begin{bmatrix}
-g_1(y(t-1)) & -g_2(y(t-1)) & \cdots & -g_p(y(t-1)) \\
-g_1(y(t-2)) & -g_2(y(t-2)) & \cdots & -g_p(y(t-2)) \\
\cdots & \cdots & \cdots & \cdots \\
-g_1(y(t-n_a)) & -g_2(y(t-n_a)) & \cdots & -g_p(y(t-n_a))].
\end{bmatrix} \tag{5.13}
$$

Los parametros θ se estiman con la descomposición en valores singulares (SVD):

$$
\begin{aligned}
\Phi &= U\Sigma V^T = \sum_{l=1}^{L} \sigma_l u_l v_l^T \\
L &= (n_b+1)m + (n_a+1)p \\
\Sigma &= diag\begin{bmatrix}\sigma_1 & \sigma_2 & \cdots & \sigma_L\end{bmatrix}.
\end{aligned} \tag{5.14}
$$

La matriz Σ contiene los valores singulares, siendo σ_L el mas pequeño. u_L es el L-ésimo vector singular izquierdo y v_L es el L-ésimo vector singular derecho. Los parámetros buscados están en $\theta = v_L$ con norma euclidiana igual a 1 lo que nos indica que es una solución única. Ya con los parámetros estimados θ se forman las siguientes matrices re-acomodando los parámetros.

$$\theta_{bc} = \begin{bmatrix} b_0c_1 & b_1c_1 & \cdots & b_{n_b}c_1 \\ b_0c_2 & b_1c_2 & \cdots & b_{n_b}c_2 \\ \vdots & \vdots & \ddots & \vdots \\ b_0c_m & b_1c_m & \cdots & b_{n_b}c_m \end{bmatrix} \tag{5.15}$$

$$\theta_{ad} = \begin{bmatrix} a_0d_1 & a_1d_1 & \cdots & a_{n_a}d_1 \\ a_0d_2 & a_1d_2 & \cdots & a_{n_a}d_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_0d_p & a_1d_p & \cdots & a_{n_a}d_p \end{bmatrix} \tag{5.16}$$

A estas matrices θ_{bc} y θ_{ad} se les aplica nuevamente la descomposición SVD para calcular $\hat{a}$, $\hat{b}$, $\hat{c}$ y $\hat{d}$.

$$\theta_{bc} = \sum_{l=1}^{\min(n_b,m)} \delta_l \beta_l \gamma_l^T \tag{5.17}$$

Donde β contiene los vectores singulares izquierdos, γ los vectores singulares derechos y los valores propios en δ.

$$\theta_{ad} = \sum_{l=1}^{\min(n_a,p)} \varphi_l \lambda_l \omega_l^T \tag{5.18}$$

Donde λ contiene los vectores singulares izquierdos, ω los vectores singulares derechos y los valores propios en φ.

Finalmente los parámetros $\hat{a}$, $\hat{b}$, $\hat{c}$, $\hat{d}$ se obtienen con el siguiente calculo:

$$\begin{aligned} \hat{a} &= \varphi_1 \omega_1, \\ \hat{b} &= \delta_1 \gamma_1, \\ \hat{c} &= \beta_1, \\ \hat{d} &= \lambda_1. \end{aligned}$$

Enseguida hay que normalizar esos parámetros mediante la norma euclidiana.

$$\alpha = \begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix}$$
$$a = s_a \frac{\hat{a}}{\|\alpha\|} \sqrt{\frac{\|\hat{d}\|}{\|\hat{c}\|}}$$
$$b = s_a s_c s_d \frac{\hat{b}}{\|\alpha\|} \sqrt{\frac{\|\hat{c}\|}{\|\hat{d}\|}}$$
$$c = s_c \frac{\hat{c}}{\|\hat{c}\|}$$
$$d = s_d \frac{\hat{d}}{\|\hat{d}\|}$$

Donde s_a, s_c, s_d denotan el signo del primer elemento no cero de $\hat{a}$, $\hat{c}$, $\hat{d}$, respectivamente.

Es posible realizar una comprobación de los parámetros aprovechando su norma unitaria.

$$\begin{aligned} \|c\| &= 1, & c &= [c_1 \quad c_2 \quad \cdots \quad c_m]^T \\ \|d\| &= 1, & d &= [d_1 \quad d_2 \quad \cdots \quad d_p]^T \\ \left\| \begin{bmatrix} a \\ b \end{bmatrix} \right\| &= 1, & a &= [a_0 \quad a_1 \quad \cdots \quad a_{n_a}]^T \\ & & b &= [b_o \quad b_1 \quad \cdots \quad b_{n_b}]^T \end{aligned}$$

5.4 Hammerstein-Wiener: Iterativo

Tomando como base la representación del modelo Hammerstein-Wiener de la figura 5.4, las ecuaciones estáticas de los bloque no lineales son:

$$\begin{aligned} x(t) &= f(u(t)), \\ y(t) &= g(r(t)). \end{aligned} \tag{5.19}$$

Las funciones no lineales son representadas mediante polinomios:

$$\begin{aligned} x(t) &= f(u(t)) = \sum_{l=1}^{m} f_l u^l(t), \\ y(t) &= g(r(t)) = \sum_{l=1}^{p} g_l r^l(t). \end{aligned} \tag{5.20}$$

El bloque dinámico lineal se modela por una función de transferencia $H(q^{-1})$ en operador en atraso, $r(t)$ es la salida y $x(t)$ es la entrada.

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2} + \cdots + a_{n_a} q^{-n_a}$$
$$B(q^{-1}) = b_0 + b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}$$

$$H(q^{-1}) = \frac{B(q^{-1})}{A(q^{-1})} \tag{5.21}$$

Para el bloque lineal, la estimación de la salida se hace mediante el predictor de un paso, ecuación 1.45.

$$\hat{r}(t) = B(q^{-1})x(t) + [1 - A(q^{-1})]r(t) \tag{5.22}$$

La idea es producir una ecuación recursiva con el predictor de un paso para estimar los parámetros de manera recursiva, la idea original fue presentada en [14]. Un paso clave es representar la ecuación de salida polinomial 5.20 en una forma alternativa sacando el primer termino de la sumatoria:

$$y(t) = g_1 r(t) + g'(r(t)). \tag{5.23}$$

Sustituimos $H(q^{-1})$ por sus polinomios en el predictor:

$$\hat{r}(t) = b_o x(t) + [B(q^{-1}) - b_o]x(t) + [1 - A(q^{-1})]r(t). \tag{5.24}$$

Sustituyendo el predictor 5.24 en la ecuación de salida 5.23:

$$\begin{aligned} y(t) = g_1[b_o f(u(t)) + [B(q^{-1}) - b_o]x(t) \\ + [1 - A(q^{-1})]r(t)] + g'(r(t)). \end{aligned} \tag{5.25}$$

Sin perdida de generalidad se pueden elegir $g_1 = 1$ y $b_o = 1$, tal que el manejo de la ecuación 5.25 sea mas sencillo y sustituyendo los equivalentes polinomiales.

$$\begin{aligned} y(t) = \sum_{l=1}^{m} f_l u^l(t) + [B(q^{-1}) - 1]x(t) \\ + [1 - A(q^{-1})]r(t) + \sum_{l=2}^{p} g_l r^l(t) \end{aligned} \tag{5.26}$$

El predictor en forma vectorial queda:

$$\hat{y}(t) = \Phi(t)\theta, \tag{5.27}$$

el vector de parametros es

$$\theta = \begin{bmatrix} f_1 & f_2 & \cdots & f_m \\ b_1 & b_2 & \cdots & b_{n_b} \\ a_1 & a_2 & \cdots & a_{n_a} \\ g_2 & g_3 & \cdots & g_p \end{bmatrix}^T$$

y el vector de datos:

$$\Phi(t) = \begin{bmatrix} u(t) & u^2(t) & \cdots & u^m(t) \\ x(t-1) & x(t-2) & \cdots & x(t-n_b) \\ -r(t-1) & -r(t-2) & \cdots & -r(t-n_a) \\ r^2(t) & r^3(t) & \cdots & r^p(t) \end{bmatrix}.$$

Hay dos variables internas, no medibles, $x(t)$ y $r(t)$ aquí es donde se hace necesario un enfoque iterativo, donde las variables internas serán actualizadas luego de cada iteración. Sustituyendo los polinomio $A(q^{-1})$ $B(q^{-1})$ en el predictor nos queda:

$$\hat{r}(t) = x(t) + \sum_{i=1}^{n_b} b_i x(t-i) - \sum_{i=1}^{n_a} a_i r(t-i). \tag{5.28}$$

Es con esta ecuación que se hace la actualización de las variables internas para cada instante de tiempo t minimiando el error de predicción:

$$e(t) = y(t) - \hat{y}(t). \tag{5.29}$$

Bibliografía

Conferencias

[1] C Lyzell, M Enqvist y Lennart Ljung. “Handling Certain Structure Information in Subspace Identification”. En: *15th IFAC Symposium on System Identification*. Saint-Malo, France, 2009, páginas 90-95 (véase página 10).

[2] Brett Ninness. “Some System Identification Challenges and Approaches”. En: *15th IFAC Symposium on System Identification*. 3. Saint-Malo, France, 2009, páginas 1-20 (véase página 17).

[3] Nabiha Touijer y Samira Kamoun. “Robust self-tuning regulation for the stochastic systems described by ARARMAX mathematical models”. En: *14th International Conference on Sciences and Techniques of Automatic Control & Computer Engineering-STA’2013*. IEEE. 2013, páginas 104-109 (véase página 33).

Artículos

[4] Er-Wei Bai. "An optimal two-stage identification algorithm for Hammerstein–Wiener nonlinear systems". En: *Automatica* 34.3 (1998), páginas 333-338 (véase página 63).

[5] Er-Wei Bai. "A blind approach to the Hammerstein–Wiener model identification". En: *Automatica* 38.6 (2002), páginas 967-979 (véanse páginas 63, 64).

[6] Silvia Curteanu y Hugh Cartwright. "Neural networks applied in chemistry. I. Determination of the optimal topology of multilayer perceptron neural networks". En: *Journal of Chemometrics* 25 (2011), páginas 527-549. DOI: 10.1002/cem.1401 (véase página 49).

[7] T. Fukuda y T. Shibata. "Theory and application of neural networks for industrial control systems". En: *Industrial Electronics, IEEE Transactions* 39.6 (1992), páginas 472-489 (véase página 48).

[8] Rolf Johansson. "An Algorithm for Continuous-Time State Space Identification". En: December (1995), páginas 721-722 (véase página 7).

[9] R. Lippman. "An introduction to computing with neural nets". En: *IEEE ASSP Magazine* 4 (1987) (véase página 48).

[10] S Joe Qin, Weilu Lin y Lennart Ljung. "A novel subspace identification approach with enforced causal models". En: *Automatica* 41.12 (2005), páginas 2043-2053. ISSN: 00051098. DOI: 10.1016/j.automatica.2005.06.010. URL: http://linkinghub.elsevier.com/retrieve/pii/S0005109805002591 (véase página 10).

[11] Hector M. Romero Ugalde et al. "Neural network design and model reduction approach for black box nonlinear system identification with reduced number of parameters". En: *Neurocomputing* 101 (2013), páginas 170-180. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2012.08.013. URL: https://www.sciencedirect.com/science/article/pii/S0925231212006522 (véanse páginas 54, 59).

[12] Hector M. Romero Ugalde et al. "Balanced simplicity–accuracy neural network model families for system identification". En: *Neural Comput & Applic* 26 (2015), páginas 171-186. DOI: https://doi.org/10.1007/s00521-014-1716-8 (véanse páginas 54, 59).

[13] Hector M. Romero Ugalde et al. "Computational cost improvement of neural network models in black box nonlinear system identification". En: *Neurocomputing* 166 (2015), páginas 96-108. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2015.04.022. URL: https://www.

sciencedirect.com/science/article/pii/S0925231215004695 (véanse páginas 54, 59).

[14] Jozef Vörös. "An iterative method for Hammerstein-Wiener systems parameter identification". En: *Journal of electrical engineering* 55.11-12 (2004), páginas 328-331 (véase página 68).

Libros

[15] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015 (véase página 32).

[16] A. Cichocki y R. Unbehauen. *Neural Networks for Optimization and Signal Processing*. 1ra. John Wiley y Sons Ltd, 1993 (véase página 50).

[17] Tohru Katayama et al. *Subspace methods for system identification*. Volumen 1. Springer, 2005 (véase página 10).

[18] Ioan D Landau y Gianluca Zito. *Digital control systems: design, identification and implementation*. Volumen 130. Springer, 2006 (véanse páginas 13, 15).

[19] Lennart Ljung. *System Identification*. Editado por Thomas Kailath. second edition. United States of America: Prentice hall, 1999, página 603 (véase página 19).

[20] M. Noorgard et al. *Neural Networks for Modelling and Control of Dynamic Systems*. 1st. Great Britain: Springer-Verlag London Berlin Heidelberg, 2000. ISBN: 1-85233-227-1 (véase página 48).

[21] William H Press et al. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007 (véase página 32).

[22] Naresh Kumar Sinha y Ganti Prasada Rao. *Identification of continuous-time systems: Methodology and computer implementation*. Volumen 7. Springer Science & Business Media, 2012 (véase página 7).

[23] Arun K Tangirala. *Principles of system identification: theory and practice*. Crc Press, 2018 (véase página 17).

[24] Steen Tøffner-Clausen, Michael J Grimble y Michael A Johnson. *System identification and robust control: A case study approach*. Springer, 1996 (véase página 10).

[25] Yucai Zhu y Ton Backx. *Identification of multivariable industrial processes: for simulation, diagnosis and control*. Springer Science & Business Media, 2012 (véase página 17).

Índice alfabético

A

Algoritmo CLOE 40
Algoritmo XCLOE 42

B

Bibliografía 71

I

Identificación lazo cerrado 35
Iterativo 67

M

Modelos
- ARMAX 24
- ARX 22
- BJ 30
- Bloques 61
- CLOE 36
- Diferencias (Ecuaciones en) . 6
- Estócastico 17
- Función de transferencia . . . 17
- Hammerstein Wiener 61
- Hammerstein-Wiener 63
- Lazo Cerrado 35
- Lineales 3
- No lineales 47
- ODE 3
- OE 27
- Otros 32
- Redes Neuronales 47

Mínimos cuadrados 13

P

Predictor de un paso 10

R

Red neuronal 2n-2-1 54
Red neuronal básica 50

S

SLIT 3
Sobreparametrización 64