PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

ESCUELA DE POSGRADO

MODELADO Y CONTROL BASADO EN REDES NEURONALES ARTIFICIALES DE UNA PLANTA PILOTO DE DESALINIZACIÓN DE AGUA DE MAR POR ÓSMOSIS INVERSA

Tesis para optar por el grado de
Magíster en Ingeniería de Control y Automatización

Autor:
NEIL NEIZER CARRASCO BANDA

Asesor:
D.Sc RAÚL RIVAS PÉREZ

Miembros del Jurado:
PhD. Juan Javier Sotomayor Moriano
PhD. Antonio Manuel Morán Cárdenas
MSc. Pedro Moisés Crisóstomo Romero

Lima - Perú
2016

DEDICATORIA

A mis padres Herlinda y Clemente, personas ejemplares, llenas de bondad. A mi hermano Willian, quien siempre está pendiente de cuidarme. A Víctor Aguilar, el amigo incondicional y sincero que siempre me ha motivado para seguir creciendo personal y profesionalmente.

AGRADECIMIENTOS

El autor de este trabajo de tesis, Neil Neizer Carrasco Banda, agradece el apoyo del Programa Nacional de Innovación para la Competitividad y Productividad (Innóvate Perú) entidad que financió el proyecto 207-FINCyT-IA-2013, en el marco del cual se desarrolló la presente tesis: “Modelado y Control Basado en Redes Neuronales Artificiales de una Planta Piloto de Desalinización de Agua de Mar por Ósmosis Inversa”.

RESUMEN

En este trabajo de tesis se parte de la hipótesis de que: es posible aumentar la eficiencia de las plantas desalinizadoras de agua de mar utilizando sistemas de control inteligente, conformados por controladores multivariables fundamentados en redes neuronales artificiales y modelos matemáticos multivariables basados en las leyes de conservación de la materia y energía.

Se empieza describiendo las tecnologías empleadas a nivel industrial para la desalinización de agua, y se continúa con una revisión del estado del arte del modelado y control de las plantas desalinizadoras que se basan en ósmosis inversa.

Luego, teniendo en cuenta la polarización de la concentración y utilizando el modelo de solución-difusión para cuantificar el transporte de las sales y del solvente a través de la membrana, se plantean las ecuaciones de balance de materia y energía en la unidad de ósmosis inversa de una planta desalinizadora de agua de mar, obteniéndose un modelo matemático no lineal multivariable. Al validar el modelo con datos experimentales se verifica un buen grado de ajuste.

Más adelante, se linealiza el modelo obtenido (alrededor de sus condiciones normales de operación) para conseguir una representación en el espacio de estados, que es utilizada en el diseño de un neurocontrolador dinámico. Se valida el desempeño del controlador diseñado frente a cambios en la referencia, presencia de ruido y rechazo a perturbaciones, encontrándose en todos los casos un desempeño satisfactorio.

Posteriormente se realiza un estudio comparativo del desempeño del controlador neuronal diseñado frente a controladores PID, verificándose con bastante claridad la superioridad del controlador neuronal, especialmente en las situaciones en las que cambian significativamente las dos variables controladas a la vez.

Se concluye la memoria de tesis haciendo una propuesta de implementación práctica del sistema de control diseñado, que sugiere el uso de una aplicación cliente/servidor OPC con el controlador neuronal implementado en Simulink (en una PC) y la comunicación con la planta a través de un PLC.

ÍNDICE

LISTA DE SÍMBOLOS PARA LAS VARIABLES DEL MODELO MATEMÁTICO

Símbolo	Variable	Unidad
A	Área superficial de la membrana	m^2
C	Concentración	mg/kg (= ppm)
C_f	Concentración de la corriente de alimentación	mg/kg (= ppm)
C_p	Concentración de la corriente de permeado	mg/kg (= ppm)
C_r	Concentración de la corriente de rechazo	mg/kg (= ppm)
C_{m1}	Concentración en la capa límite del lado rechazo	mg/kg (= ppm)
C_{m2}	Concentración en la capa límite del lado permeado	mg/kg (= ppm)
$Cond$	Conductividad	uS/cm
$Cond_p$	Conductividad eléctrica de la corriente de permeado	uS/cm
F_f	Flujo másico de la corriente de alimentación	Kg/s
F_p	Flujo másico de la corriente de permeado	Kg/s
F_r	Flujo másico de la corriente de rechazo	Kg/s
F_{m1}	Flujo másico a través de la membrana desde la capa límite del lado rechazo	Kg/s
F_{m2}	Flujo másico desde la membrana hacia la capa límite del lado permeado	Kg/s
pH_f	medida del contenido de iones H^+ en la corriente de alimentación	adimensional
k	Selectividad de la membrana	bar^{-1}
k_A	Constante de permeabilidad al soluto	m/s
k_B	Constante de permeabilidad al solvente	$Kg/(m^2.s.bar)$
m_P	Masa de fluido dentro del volumen de control del subsistema permeado	Kg
m_r	Masa de fluido dentro del volumen de control del subsistema rechazo	Kg

P_f	Presión de la corriente de alimentación	Bar
P_p	Presión de la corriente de permeado	Bar
P_r	Presión de la corriente de rechazo	Bar
P_{m1}	Presión en la capa límite del lado rechazo	Bar
P_{m2}	Presión en la capa límite del lado permeado	Bar
$Ploss_P$	Pérdidas de presión en el lado permeado	Bar
$Ploss_r$	Pérdidas de presión en el lado rechazo	Bar
T_f	Temperatura de la corriente de alimentación	°C
T_p	Temperatura de la corriente de permeado	°C
T_r	Temperatura de la corriente de rechazo	°C
T_{m1}	Temperatura en la capa límite del lado rechazo	°C
T_{m2}	Temperatura en la capa límite del lado permeado	°C
π	Presión osmótica	bar
θ	Corte	adimensional
Λ	Conductividad molar	S.cm2.mol-1

INTRODUCCIÓN

Actualmente, debido al progresivo crecimiento demográfico e industrial, la demanda de agua potable ha aumentado significativamente en casi todos los países del planeta. Para satisfacer esta demanda, se necesitan fuentes de agua de fácil acceso y económicamente aprovechables, que sin embargo; como consecuencia de la contaminación ambiental, son cada vez más escasas y de menor calidad (Rivas-Perez, 1990; Wilf y Bartels, 2005).

En el Perú, aunque, en general el agua no constituye un recurso escaso, este no se encuentra disponible de forma natural en el espacio y tiempo en el que se necesita, por ejemplo, mientras que en la selva el agua es abundante, muchas poblaciones y actividades económicas de la costa sufren de continua escasez (Moncada-Valerio et al., 2012). Además, al ritmo de desarrollo actual, con el paso del tiempo, esta situación de escasez se volverá más crítica; hecho que exige la necesidad de promover la utilización de procesos alternativos para el abastecimiento de agua potable, entre los que se encuentra la desalinización del agua de mar con tecnologías eficientes y económicamente atractivas.

La ósmosis inversa (O.I.) constituye desde hace algunos años la tecnología más eficiente para la desalinización de agua de mar, pero aún no lo suficientemente atractiva desde el punto de vista económico (Voutchkov, 2013). En este trabajo se parte de la hipótesis de que es posible aumentar la eficiencia de las plantas desalinizadoras de agua de mar utilizando sistemas de control inteligente, conformados por controladores multivariables fundamentados en redes neuronales artificiales (R.N.A.) y modelos matemáticos multivariables basados en las leyes de conservación de la materia y energía.

El presente trabajo de tesis tiene como objetivo general: obtener un modelo matemático multivariable de una planta desalinizadora de agua de mar por O.I. empleando las leyes de conservación de la materia y energía; así como, diseñar un controlador multivariable basado en R.N.A. para las variables críticas de dicha planta, con el fin de aumentar su eficiencia.

Para cumplir con este objetivo general, se deben lograr los siguientes objetivos específicos:

- Estudiar los fundamentos teóricos de la desalinización de agua de mar mediante O.I.
- Obtener un modelo matemático multivariable de la unidad de O.I. de una planta piloto de desalinización de agua de mar.
- Diseñar un controlador multivariable basado en R.N.A para las variables críticas de la planta objeto de estudio.
- Realizar una propuesta de implementación práctica del sistema de control diseñado.

El capítulo uno se empieza hablando sobre la problemática de la escasez de agua potable en el mundo, luego se describen las principales tecnologías utilizadas en la desalinización de agua. A continuación se presentan los modelos estáticos, dinámicos, teóricos y experimentales que se han desarrollado a lo largo del tiempo para el proceso de ósmosis inversa. Se termina el capítulo haciendo una revisión del estado del arte de los sistemas automáticos de control para las plantas industriales de desalinización mediante O.I.

El capítulo dos se empieza describiendo cada una de las etapas que se llevan a cabo en una planta industrial de desalinización de agua por O.I., luego se habla de las membranas, su estructura, sus características y su configuración; se continúa el capítulo mencionando los parámetros más importantes de un proceso de desalinización por O.I., y se describe la planta piloto de desalinización de agua de la PUCP. Posteriormente, se aplican las ecuaciones de balance de materia y energía para obtener un modelo matemático no lineal multivariable de la unidad de O.I. de una planta piloto de desalinización de agua de mar, el cual posteriormente se valida con datos experimentales.

El capítulo tres se empieza repasando la teoría básica sobre redes neuronales artificiales, especialmente aquella relacionada con la arquitectura de las redes y sus mecanismos de aprendizaje, luego se habla de los neurocontroladores y sus algoritmos de entrenamiento. Se propone una generalización del algoritmo Dynamic Back Propagation para diseñar neurocontroladores dinámicos multivariables y se desarrolla un procedimiento de carácter general para el escalamiento de las de señales de entrada y salida a un

neurocontrolador y para la transformación lineal del modelo matemático utilizado en su entrenamiento. Posteriormente se explican las razones por las que conviene diseñar un neurocontrolador para la planta objeto de estudio. Se obtiene una representación de la planta en el espacio de estados, con la cual se entrena el neurocontrolador que posteriormente se implementa en Simulink mediante un bloque MATLAB-Function para controlar el modelo no lineal de la unidad de O.I. de una planta piloto de desalinización de agua de mar. Finalmente se evalúa el desempeño del neurocontrolador diseñado en diversos escenarios: seguimiento de trayectorias, rechazo a perturbaciones y presencia de ruido.

El capítulo cuatro se empieza realizando un análisis comparativo del desempeño del controlador neuronal diseñado frente a controladores PID y se termina haciendo una propuesta de implementación práctica del sistema de control diseñado.

1. ESTADO DEL ARTE DEL MODELADO Y CONTROL DE PLANTAS DE DESALINIZACIÓN DE AGUA DE MAR POR ÓSMOSIS INVERSA

1.1. Introducción

Actualmente, debido a la sobrepoblación de las zonas urbanas, la sobre explotación de las principales cuencas fluviales, la contaminación de los acuíferos subterráneos y el alarmante calentamiento de las aguas oceánicas (Calderon-Valdez et al., 2015), la escasez de agua potable constituye un grave problema para todos los países (Kovalenko et al., 1989; Pedregal et al., 2009; Rivas-Perez et al. 2014a, 2014b).

Solo un pequeño porcentaje del agua de la Tierra es dulce y de este menos del 1% es de fácil acceso para el uso humano (Watkins et al., 2006). De acuerdo a la última publicación de la OMS referente a saneamiento y agua potable, el 91% de la población mundial utiliza fuentes segura de agua potable, quedando aun 663 millones de personas que carecen de este acceso (UNICEF, 2015), además, se estima que en el 2025, la mitad de la población vivirá en zonas afectadas por la escasez de agua (WHO, 2013). Esto afectará la producción agrícola e industrial, aumentando en consecuencia los precios de los productos industriales y de los alimentos (Feliu-Batlle et al., 2008, 2005).

Esta realidad sugiere que, hay una alta probabilidad de que estallen guerras por el agua potable si no se encuentra una solución factible y fiable para la escasez en todos los países del planeta (Kovalenko et al., 1990; Rivas-Perez et al., 2011, 2003).

Muchos científicos consideran que la desalinización del agua de mar es la solución que resolverá la futura crisis de agua en el planeta (Moncada-Valerio et al., 2012; Rivas-Perez et al., 2014c), sin embargo; advierten que esta solución no es simple porque requiere de tecnologías costosas y consumidoras de grandes cantidades de energía.

1.2. Tecnologías para la desalinización de agua de mar

En muchos lugares la necesidad de desalinizar el agua es incluso más importante y urgente que la producción de alimentos.

Desalinizar el agua significa separarla de la solución salina en la que se encuentra. El proceso más antiguo para este propósito es el de destilación, en el cual, primero se evapora el agua agregándole calor o reduciendo su presión de vapor, y luego se lo condensa sobre una superficie fría. Las tres tecnologías principales del tipo destilación que se utilizan a nivel industrial para desalinizar el agua son: destilación flash multietapa (MSF), evaporación multiefecto (ME) y compresión de vapor (VC).

Hasta los años 80, la tecnología más utilizada era la MSF, pero actualmente los procesos con membranas dominan el mercado. En estos procesos, la separación ocurre gracias a la naturaleza selectiva de la membrana, la cual, bajo la influencia de una fuerza motriz externa, permite el paso preferencial ya sea del agua o de los iones, pero no de ambos.

Destilación flash multietapa (MSF). En este proceso, la condensación del vapor producido precalienta la corriente de alimentación (agua de mar) a medida que esta fluye sucesivamente en una serie de etapas. La corriente de alimentación se calienta aún más con una fuente de calor externa antes de ingresar a las cámaras de evaporación flash (etapas), en las que, mediante la disminución súbita y sucesiva de su presión se convierte en el vapor que se utiliza en el precalentamiento del agua de alimentación y que después de condensarse constituirá el producto final. La gran mayoría de plantas que utilizan esta tecnología están instaladas en Medio Oriente y en general son de grandes capacidades (Wilf, 2004). Un esquema simplificado de una planta de desalinización de agua de mar que utiliza el proceso MSF se muestra en la Figura 1.1.

Evaporación multiefecto (ME). En este proceso, se utiliza el calor latente de condensación del vapor producido en un efecto para evaporar agua líquida en el siguiente efecto; logrando de esta manera una gran integración energética, y por ende, una alta eficiencia. Las plantas ME se diseñan principalmente de dos tipos: multiefecto de tubos horizontales (HTME) o evaporador de tubos verticales (VTE). En un evaporador multiefecto de tubos verticales (Figura 1.2), el agua salina que fluye hacia abajo dentro de los tubos se evapora con la energía suministrada por la condensación (en la parte exterior de los tubos) del vapor proveniente de un efecto de mayor temperatura.

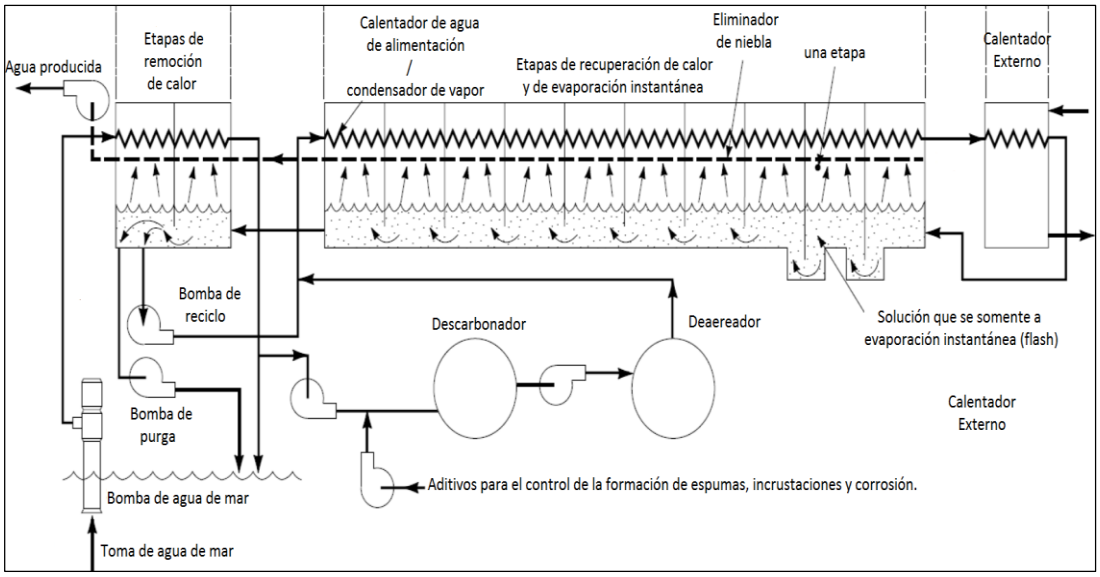

Figura 1.1. Diagrama de flujo del proceso de evaporación flash multietapa (MSF), (Kirk-Othmer, 2004).

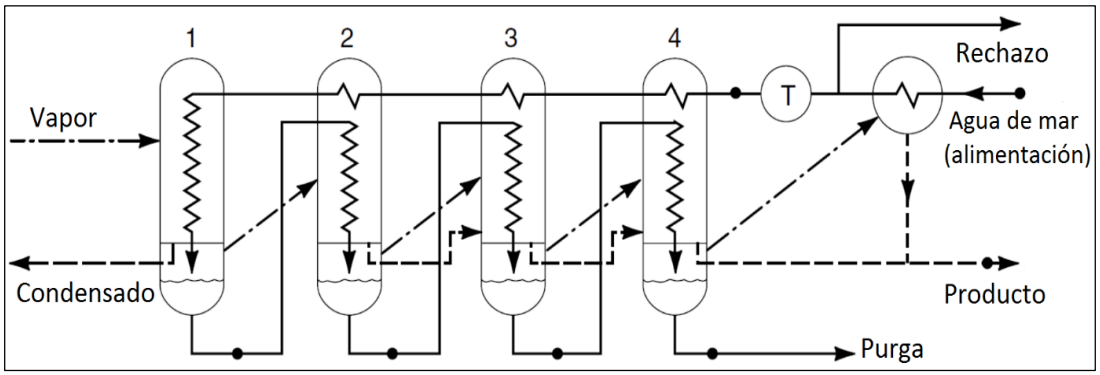

Figura 1.2. Diagrama de flujo simplificado de una planta de desalinización con cuatro efectos de tubos verticales, (Kirk-Othmer, 2004).

Destilación por compresión de vapor (VC). En este proceso, se vaporiza el agua de mar precalentada rociándola sobre un banco de tubos en cuyo interior se condensa vapor. El mismo vapor producido de esta manera, es el que luego se comprime y envía a condensar en el interior del banco de tubos para suministrar el calor necesario. El compresor puede ser accionado por una turbina de vapor, un motor de combustión interna o un motor eléctrico (Figura 1.3a). La compresión del vapor también puede conseguirse con un eyector (Figura 1.3b). Esta última configuración mejora la confiabilidad y disponibilidad de la planta gracias a su simplicidad y ausencia de partes móviles; no obstante, tiene menos eficiencia.

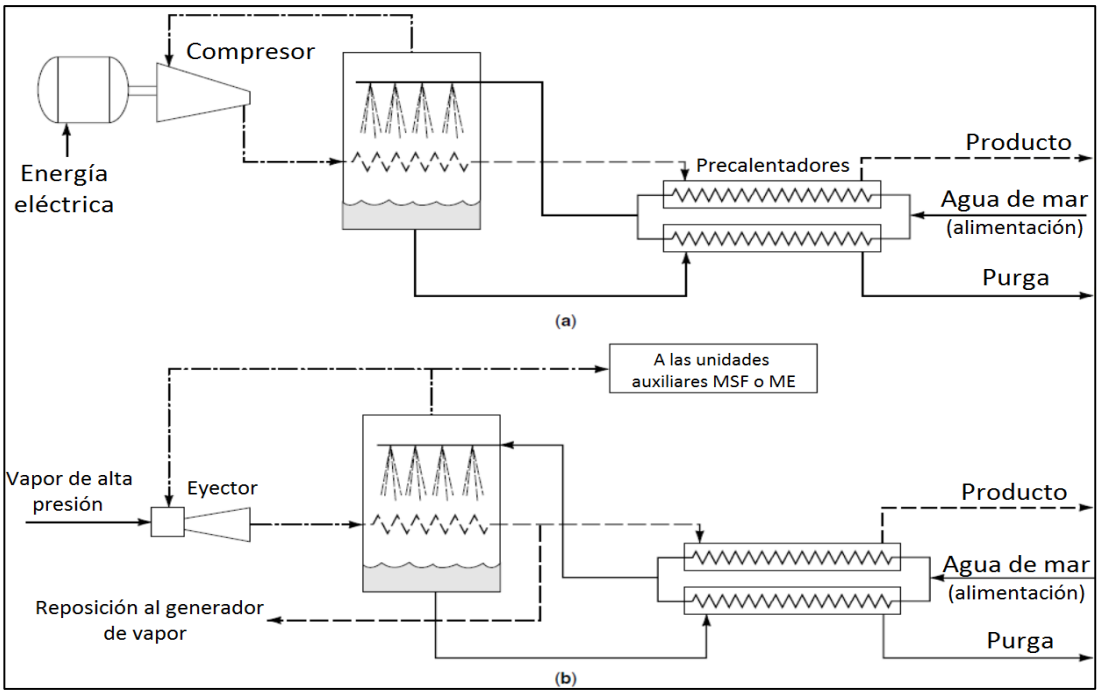

Figura 1.3. Diagrama de flujo esquemático de una planta de desalinización por compresión de vapor de tubos horizontales: (a) con compresor accionado por un motor eléctrico, (b) con un eyector (Kirk-Othmer, 2004).

Electrodiálisis (ED). En este proceso, la alimentación (agua de mar) es conducida por el espacio entre dos membranas, una permeable solo a cationes y la otra solo a aniones. Un campo eléctrico externo dirige los aniones en dirección al cátodo y los cationes hacia el ánodo. Debido a la selectividad de las membranas, estos iones solo pueden atravesar un compartimiento en su camino hacia el electrodo respectivo quedando atrapados inmediatamente en el siguiente compartimiento después de atravesar una membrana. Esto permite reducir manera la concentración de sales en algunos compartimientos e incrementarla en otros. Un esquema simplificado de una planta de desalinización por electrodiálisis se muestra en la Figura 1.4.

Ósmosis inversa (O.I.). En este proceso, la desalinización está basada en el fenómeno de transferencia de materia más común de las células vivientes (la ósmosis directa). Se aplica una presión mayor que la presión osmótica en lado de la solución más concentrada en sales para forzar el paso del agua a través de una membrana que es permeable únicamente al solvente. La Figura 1.5 muestra la configuración típica de un sistema de desalinización de agua de mar mediante O.I.

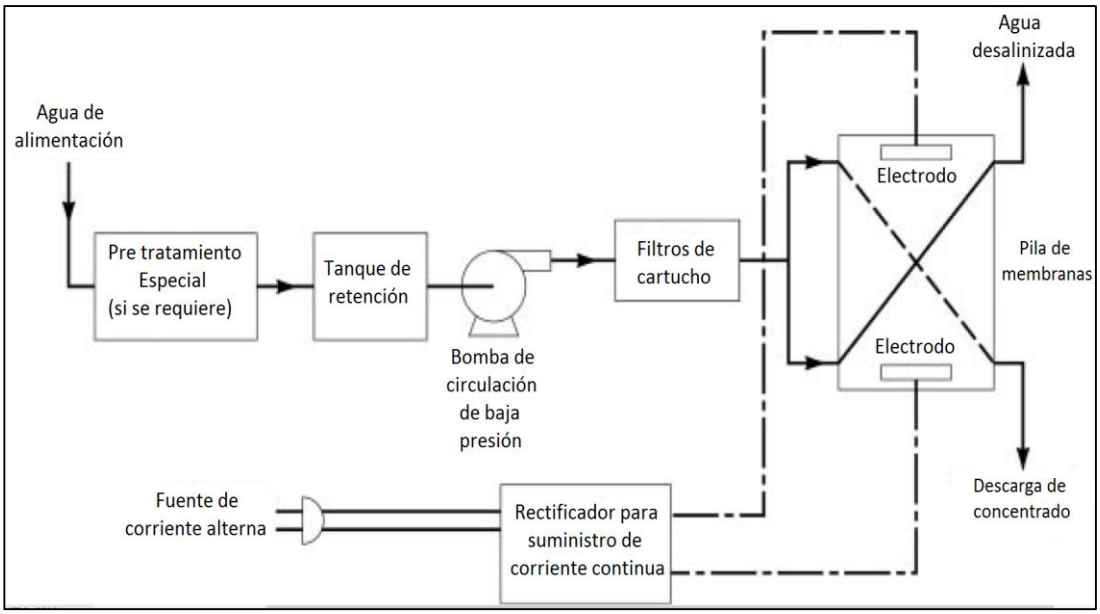

Figura 1.4. Diagrama de flujo simplificado de una planta de desalinización por electrodiálisis (Kirk-Othmer, 2004).

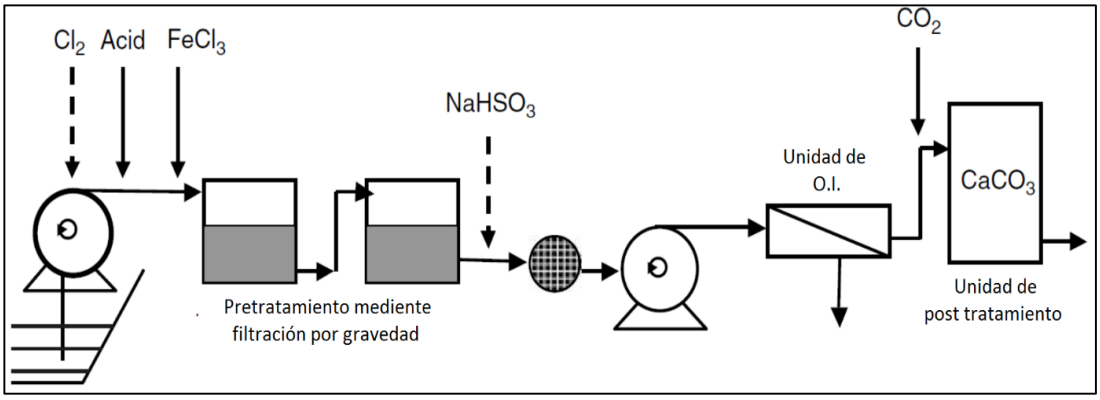

Figura 1.5. Esquema simplificado de una planta de desalinización de agua de mar mediante O.I. (Kirk-Othmer, 2004).

1.3. Estado del arte del modelado de las plantas desalinizadoras de agua de mar que utilizan ósmosis inversa

1.3.1. Introducción

De entre todas las tecnologías para desalinizar el agua de mar, descritas en la sección 1.2, se particulariza en el proceso basado en el fenómeno de la ósmosis inversa, y se hace una revisión del estado del arte de su modelado matemático.

La desalinización de agua utilizando ósmosis inversa, empezó a investigarse en los años 60 y se utilizó comercialmente a partir de la siguiente década (Alatiqi el al., 1989). Aunque la mayor parte de los costos fijos y variables para la instalación de plantas industriales de desalinización por O.I. dependen principalmente de la industria de fabricación de las membranas, el control automático juega un papel fundamental en la optimización energética, pues, mientras mejor sea el sistema de control de la planta, será posible operarla en forma segura, en puntos de operación más favorables y con mayor eficiencia. Se empieza estudiando el principio termodinámico que rige el fenómeno osmótico que se produce en un sistema formado por dos disoluciones de diferentes concentraciones separadas por una membrana semipermeable. Este principio (también llamado fuerza motriz) es la diferencia de los potenciales químicos que el solvente tiene en cada disolución, debido a la cual pasará espontáneamente a través de la membrana hacia la solución más concentrada (Figura 1.6a). Esta transferencia de materia continuará de manera espontánea hasta que el potencial químico del solvente sea el mismo en ambas disoluciones.

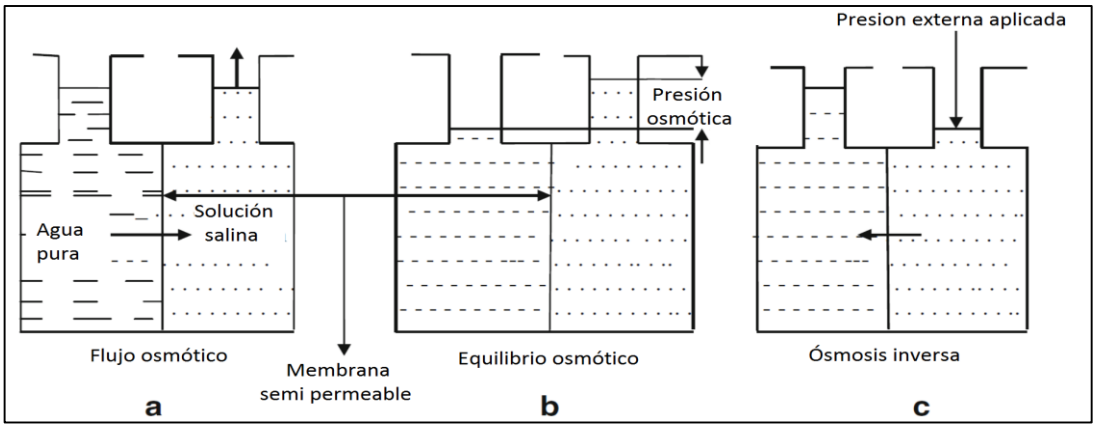

Figura 1.6. Procesos de difusión a través de membranas.

En variables medibles, la igualdad de potenciales químicos se alcanza cuando la presión de la solución más concentrada evita que más flujo de solvente se transfiera desde la solución diluida. A este fenómeno se le conoce como equilibrio osmótico y a la presión de equilibrio como presión osmótica (Figura 1.6b). Cuando se aplica una presión externa a la solución concentrada para forzar el flujo de solvente en la dirección opuesta a lo que normalmente sucedería se produce la ósmosis inversa (Figura 1.6c).

Es importante mencionar que los modelos matemáticos pueden clasificarse como estáticos o dinámicos y teóricos o experimentales. En el presente trabajo se utilizará la clasificación teórico – experimental; entendiendo por teóricos, a los modelos basados en las ecuaciones de conservación de la materia y energía; y por experimental, a los modelos obtenidos mediante técnicas de identificación de sistemas. La clasificación anterior implica que, entre los modelos teóricos se estudiarán los modelos estáticos y dinámicos de ese tipo, y que se hará lo mismo para los modelos experimentales.

1.3.2. Modelos basados en las ecuaciones de balance de materia y energía

1.3.2.1. Modelos estáticos

La gran mayoría de modelos teóricos encontrados en la literatura para el proceso de O.I. son modelos estáticos, que se utilizan principalmente en el diseño de las plantas desalinizadoras en virtud de que describen de forma adecuada el desempeño de las membranas. Estos importantes modelos, se subdividen en tres categorías:

- Modelos de termodinámica irreversible
- Modelos no porosos o de membrana homogénea
- Modelos porosos

- Modelos de termodinámica irreversible

En estos modelos, las ecuaciones que predicen el flujo de los solutos y del solvente a través de la membrana se derivan suponiendo que el sistema a su alrededor está en equilibrio termodinámico. La densidad de flujo (flux) del solvente J_w a través de la membrana viene dada por:

$$J_w = L_P(\Delta P - \sigma\Delta\pi), \tag{1.1}$$

donde P es la presión, σ el coeficiente de reflexión, L_P una constante y π la presión osmótica, que se calcula con la ecuación de Van't Hoff:

$$\pi = \emptyset\left(\frac{n}{V}\right)RT, \tag{1.2}$$

donde $\emptyset$ es el coeficiente de presión osmótica, n el número de moles de solutos disueltos, V el volumen de la disolución, T la temperatura absoluta y R la constante universal de gases ideales.

De manera similar, el flux de los solutos J_s se obtiene de la ecuación:

$$J_s = \omega\Delta\pi + (1-\sigma)(C_m)_{avg}J_w, \quad (1.3)$$

donde ω es una constante y $(C_m)_{avg}$ es la media logarítmica de la concentración de solutos en la membrana.

Este modelo, desarrollado por Kedem y Katchalsky (1958) fue mejorado por Spiegler y Kedem (1966), y ha alcanzado gran aplicabilidad a pesar de no describir el mecanismo de transporte a través de la membrana.

- Modelos no porosos

Londsdale et al. (1965) superaron los inconvenientes de los modelos de termodinámica irreversible al suponer que el soluto y solvente primero se disuelven en la membrana antes de difundirse a través de ella. Este modelo llamado modelo de solución-difusión no toma en cuenta las características de la membrana. Fue mejorado por Soltaniesh y Gill (1981), y posteriormente por Bhattacharyya y Willians (1992c), quienes asumieron que:

- La membrana de ósmosis inversa no tiene poros y su capa superficial es homogénea.
- El solvente y los solutos se disuelven en la capa superficial de la membrana y luego se difunden a través de ella.
- La difusión del soluto es independiente de la del solvente y es exclusivamente debida al gradiente de potencial químico.
- Los gradientes de potencial químico son consecuencia de las diferencias de presiones y concentraciones en ambos lados de la membrana.

Estos modelos emplean la ley de Fick para calcular las densidades de flujo de solvente (J_W) y de soluto (J_S) a través de la membrana con:

$$J_W = -D_{Wm}\frac{dC_{Wm}}{dz} \quad (1.4)$$

$$J_S = -D_{Sm}\frac{dC_m}{dz}, \quad (1.5)$$

donde D_{Wm}, D_{Sm} son las difusividades en la fase membrana y dC_{Wm}/dz, dC_{Wm}/dz los gradientes de concentración del solvente y soluto respectivamente.

Las ecuaciones (1.4) y (1.5) usualmente se utilizan en sus formas simplificadas:

$$J_w = K_w(\Delta P - \Delta\pi) \quad (1.6)$$

$$J_s = K_B(C_F - C_P) \quad (1.7)$$

donde C_f y C_p son las concentraciones de la alimentación y del permeado respectivamente.

En las ecuaciones (1.6) y (1.7) se observa que solamente se requieren dos parámetros para caracterizar el sistema, a saber: la permeabilidad al solvente K_W y la permeabilidad al soluto K_B. Por su simplicidad, este modelo ha sido ampliamente utilizado en procesos de ósmosis inversa con solutos orgánicos e inorgánicos (Sobana y Panda, 2011).

Soltaniesh y Gill (1981) observaron que el modelo de solución-difusión da buenas predicciones solamente en membranas con bajo contenido de agua. Posteriormente, Mazid (1984) observó que el modelo de solución-difusión no predice bien el flujo de solutos ni de solvente para muchas combinaciones de membranas y solutos (particularmente orgánicos).

Luego Burgoff et al. (1988) verificaron que el modelo de solución-difusión no explica la disminución ocurrida en el flujo de solvente cuando la solución contiene algunos compuestos orgánicos disueltos; y para superar este inconveniente propusieron un modelo que toma en cuenta la influencia de la presión sobre el potencial químico del soluto, y una modificación a la ecuación de transporte que asigna un efecto significativo a los solutos orgánicos pero despreciable a los inorgánicos. Estas expresiones para el potencial químico μ_s y para la densidad de flujo de soluto J_s están dadas por:

$$\mu_s = RTln\left(\frac{C_F}{C_P}\right) + V\Delta P \quad (1.8)$$

$$J_s = \frac{D_{Sm}K_{Sm}}{\delta}(C_F - C_P) + L_{SP}\Delta P, \quad (1.9)$$

donde V es el volumen de la disolución, D_{Sm} el coeficiente de difusión del soluto en la membrana, K_{Sm} el coeficiente de distribución del soluto, δ el espesor de la membrana y L_{SP} el parámetro responsable del transporte de sales debido a la diferencia de presión entre ambos lados de la membrana.

Asimismo, Burgoff et al. (1988) reconocen que este modelo no describe bien el transporte de algunos solutos orgánicos, ni predice el decremento sustancial del flujo de agua cuando el sistema contiene compuestos orgánicos disueltos. Por estas razones el modelo no es muy aplicado en la práctica.

- Modelos porosos

Suponiendo que el transporte de solutos y solvente se puede llevar a cabo a través de las imperfecciones (poros) de las membranas, se formularon otros modelos (Sherwood, 1967). Según estos, el flux total de solvente N_w y de soluto N_s a través de una membrana vienen dados por:

$$N_w = J_w + K_2 \Delta P \tag{1.10}$$

$$N_s = J_s + K_2 \Delta P \tag{1.11}$$

Aunque se ha demostrado que estas ecuaciones se ajustan bastante bien a los datos experimentales (Soltaniesh y Gill, 1981), no se usan con frecuencia en la práctica porque se necesitan tres parámetros para caracterizar el sistema, parámetros que usualmente son función de la presión y la concentración, y que deben ser determinados mediante regresión no lineal. Este modelo también falla cuando el sistema contiene algunos compuestos orgánicos disueltos, pues predice un flujo de solvente mucho mayor al real.

Posteriormente, Sourirajan (1970) presentó el modelo de difusión porosa, en el cual se asume que la membrana es microporosa (con una estructura capilar) y que el mecanismo de la separación es determinado por el transporte a través de los poros y por un fenómeno superficial. Según este modelo, el flux de agua J_w y el flux de los solutos J_s se calculan con:

$$J_w = K_w(\Delta P - [\pi(X_F) - \pi(X_P)]) \tag{1.12}$$

$$J_s = \frac{D_{SP} K_B C_T}{\delta}(X_F - X_P), \tag{1.13}$$

donde X_F y X_P son las fracciones molares del soluto en la alimentación y en el permeado respectivamente.

Este modelo tampoco puede explicar la disminución del flujo de solvente debido a la presencia de solutos orgánicos.

Merten (1966), Johnson y Boesen (1975) asumieron que: la membrana es microporosa, que el transporte del solvente es causado por un esfuerzo cortante en la dirección flujo dentro los poros y que el transporte de los solutos es llevado a cabo tanto por difusión como por convección, para proponer el modelo de poros finos, que luego fue mejorado por Souriraraja y Matsura (1985), quienes obtuvieron la expresión para el flux de los solutos a través de una membrana, dada por:

$$J_S^{Poros} = \frac{-RT}{X_{Sw}*b}\frac{dC_{poros}}{dz} + \frac{u*C_{poros}}{b}, \quad (1.14)$$

donde u es la velocidad del soluto a través de los poros, X_{Sw} representa la fuerza de fricción entre el solvente y la membrana, y b es definido como la relación entre la fuerza de fricción que actúa sobre el soluto que se mueve a través de los poros de la membrana y la fuerza de fricción que experimenta el soluto en la solución libre.

Souriraraja y Matsura (1985) consideraron que la ecuación para el transporte del solvente J_W se obtiene al igualar la presión transmembrana con la fuerza de fricción en las paredes de los poros:

$$J_W = \frac{\varepsilon R_P^2 \Delta P}{8\eta\tau\delta}\left[\frac{1}{1+\frac{R_P^2 X_{Sm} C_P}{8\eta}}\right], \quad (1.15)$$

donde ε es la porosidad de una membrana de espesor $\tau\delta$, Rp es el radio de los poros y X_{Sm} representa la fuerza de fricción entre la membrana y el soluto. A pesar de los esfuerzos realizados, este modelo no puede predecir correctamente el flujo del solvente.

Mehdizadeh (1990) modificó el modelo de poros finos considerando que los poros son de un menor tamaño e introdujo un término adicional para tomar en cuenta el componente difusivo del flujo en la salida de los poros, obteniendo en base a ello que la ecuación para el flux de soluto está dada por:

$$J_S^{poros} = -R\frac{T}{X_{Sw}b}\frac{dC_{poros}}{dz}\bigg|_{z=\tau\delta} + \frac{uC_{poros}}{b}\bigg|_{z=\tau\delta} \quad (1.16)$$

Luego, como el balance de masa en estado estacionario para el soluto en el interior de los poros da:

$$\frac{dJ_S^{poros}}{dz} = 0 \quad (1.17)$$

Se combinan las ecuaciones (1.14), (1.16) y (1.17) para obtener:

$$\frac{d^2C_{poros}}{dz} - \frac{uX_{Sw}}{RT}\frac{dC_{poros}}{dz} = 0 \quad (1.18)$$

La ecuación (1.18) se puede resolver con las condiciones de frontera:

$$C_{poros}(0) = K_D C_F \quad (1.19)$$

$$C_{poros}(\tau\delta) = K_D C_P \quad (1.20)$$

Para dar (Mehdizadeh, 1990):

$$C_P = C_F - (C_P - C_F)\left[\frac{1-exp\left(\frac{\mu X_{Sw} z}{RT}\right)}{1-exp\left(\frac{\mu\tau\delta X_{Sw}}{RT}\right)}\right] \quad (1.21)$$

Aunque la ecuación (1.21) puede resolverse mediante prueba y error, se prefiere partir de la ecuación (1.18) y obtener una solución explícita para C_P haciendo una reformulación de las condiciones de frontera (Williams, 2013). Mehdizadeh (1990) verificó que este modelo predice mayores concentraciones de permeado que el modelo original de poros finos.

Más adelante, Matsura y Sourirajan (1981), Dickson (1988) modificaron sus modelos para tener en cuenta la variación de la concentración de los solutos en la dirección radial dentro de los poros de la membrana, consiguiendo con ello un nuevo modelo que fue llamado modelo de flujo fuerza superficial – poro (Bhattacharyya y Willians, 1992c), en el cual se asume que:

- El transporte del solvente en la membrana ocurre a través de sus poros como consecuencia de un esfuerzo cortante.
- El transporte de los solutos se lleva a cabo por difusión y convección dentro de los poros de la membrana.
- El transporte del agua y los solutos a través de los poros de las membranas es determinado por las fuerzas de interacción, las fuerzas de fricción y los gradientes de potencial químico.
- Los poros son cilíndricos y atraviesan la membrana de lado a lado.
- La distribución de los solutos dentro de los poros es controlada por un campo potencial.

En el 2002 Wiley y Fletcher (2002) validan un modelo formado por ecuaciones diferenciales parciales que toma en cuenta la variación espacial de las propiedades (concentración, presión, temperatura, flujo, etc.) de las corrientes del sistema. Este modelo que fue resuelto con un software de dinámica de fluidos computacional mostró buen ajuste a los datos experimentales.

En el 2013 Sano y Nakayama (2013) propusieron un modelo de transporte para membranas de fibra hueca formado por tres ecuaciones diferenciales ordinarias de primer orden escritas en términos de los valores promedio de la velocidad, presión y concentración de la corriente de rechazo. Este modelo que necesita del empleo adicional de cualquiera de los modelos existentes para el transporte de solvente y solutos a través la membrana, toma en cuenta la polarización de la concentración y fue validado con datos experimentales mostrando un excelente grado de correspondencia.

1.3.2.2. Modelos dinámicos

Estos modelos sirven para simular el comportamiento dinámico de las plantas de desalinización por ósmosis inversa y para diseñar sistemas de control automático. Una revisión bibliográfica exhaustiva ha permitido verificar que son muy pocos los modelos existentes que sean teóricos y dinámicos a la vez.

Gambier et al. (2007) emplean las ecuaciones de conservación de la materia, energía y cantidad de movimiento para obtener un modelo de parámetros concentrados que luego utilizan para detección y diagnóstico de fallas. Se han encontrado errores e inconsistencias de fundamento en las ecuaciones de balance, por lo que no se recomienda el uso de este modelo.

Belkacem (2008) modela una planta experimental de desalinización por O.I. en la que la corriente de rechazo se recircula al tanque de alimentación. Se observa que los bastidores de ósmosis inversa son modelados con ecuaciones algebraicas (es decir, no se toma en cuenta su dinámica). Las ecuaciones diferenciales parciales (EDP) que forman el modelo son resueltas con el método de elementos finitos (MEF).

Al-haj et al. (2009) desarrollaron un modelo dinámico teórico en el que cada elemento de membrana es dividido en un conjunto de tuberías en serie (de modo que la salida de una tubería sea la entrada de la siguiente). El modelo está formado por un sistema de ecuaciones diferenciales ordinarias y la validación con datos experimentales muestra un buen grado de exactitud.

Sobana y Panda (2013) reportan modelos teóricos para las diferentes secciones de una planta completa de desalinización por O.I., que luego validan con datos experimentales. Este modelo tiene el mismo inconveniente que el de Absar et al. (2008).

Jiang et al. (2014), basados en la teoría de solución-difusión, obtienen un modelo de parámetros distribuidos para la simulación del proceso de desalinización por O.I. Los autores resuelven el sistema de EDP que constituyen el modelo utilizando el MEF y realizan la validación con los datos experimentales de Abbas (2005).

Palacín (2014), obtiene un modelo matemático multivariable escrito en términos de ecuaciones diferenciales parciales que implementa en el software EcosimPro para propósitos de simulación y control.

1.3.3. Modelos basados en datos experimentales

Los modelos más utilizados para propósitos de control en plantas desalinizadoras de agua de mar por O.I. se han obtenido mediante las técnicas de identificación de sistemas. Con estas herramientas, a diferencia del uso de los balances de materia y energía, los modelos se consiguen a partir de datos experimentales de entrada-salida de la planta y tienen la desventaja de que solo sirven en el proceso particular para el cual se tomaron los datos (Rivas-Perez-et al., 1994).

Alatiqi et al. (1989) mediante identificación de sistemas estimaron la estructura MIMO de una planta piloto de desalinización de agua de mar por O.I.

Assef et al. (1995) y Robertson et al. (1996), obtuvieron modelos con matrices de transferencia multivariable, en las que cada función de transferencia individual es de segundo orden con un cero en el numerador.

Fkirin y Al Madhair (1997) realizaron una identificación óptima en línea para determinar los parámetros de un modelo con estructura ARMAX utilizando el método recursivo de mínimos cuadrados.

Zilouchian y Jafar (2001) identificaron una planta de desalinización por O.I. que funciona con energía fotovoltaica, obteniendo una matriz de transferencia y una representación en el espacio de estados.

Saengrung et al. (2007) modelan dos plantas de desalinización por O.I. mediante identificación de sistemas y redes neuronales artificiales obteniendo una representación en el espacio de estados, modelos individuales con estructura ARX y un modelo neuronal.

Riverol y Pilipovik (2005) identifican en tiempo discreto a una planta de desalinización por O.I. y luego convierten el modelo a tiempo continuo.

Abbas y Al-Bastaki (2005) obtienen un modelo neuronal con tres entradas (presión, temperatura y concentración de la alimentación), una capa oculta y una salida (flujo de permeado) para una planta experimental.

Chaaben et al. (2011) obtienen la matriz de transferencia MIMO de un pequeño sistema de ósmosis inversa que funciona con energía fotovoltaica.

Sobana y Panda (2013) reportan modelos con estructura ARX de primer orden más tiempo muerto para relacionar las diferentes variables del modelo MIMO de una planta real de desalinización por O.I. También presentan un modelo no lineal en el dominio del tiempo obtenido mediante regresión.

Dhaifallah y Nisar (2014) utilizan la técnica de identificación no lineal basada en máquinas de soporte vectorial por mínimos cuadrados para identificar los modelos de Wiener de un proceso de desalinización por O.I. Los resultados obtenidos mostraron un ajuste de 96% a los datos experimentales.

Este estudio del estado del arte muestra que, las técnicas de modelado matemático para la obtención de modelos multivariables del comportamiento dinámico de plantas desalinizadoras de agua de mar por O.I., han sido muy poco utilizadas, y que; posiblemente los únicos modelos dinámicos teóricos válidos son el de Jiang et al. (2014) y el de Palacín (2014).

A estas alturas del desarrollo de la industria de la desalinización, se debería contar con más modelos dinámicos teóricos que (dependiendo de la aplicación) puedan emplearse rápidamente en el diseño de sistemas efectivos de control.

Por ello, en la presente tesis se propone un modelo matemático de parámetros concentrados obtenido a partir las ecuaciones de conservación de la materia y energía.

1.4. Estado del arte de los sistemas de control de plantas desalinizadoras de agua de mar que utilizan ósmosis inversa

1.4.1. Introducción

En 1999, Alatiqi y colaboradores (Alatiqi et al., 1999) estudian el estado del arte de los sistemas de control e instrumentación utilizados en plantas desalinizadoras de agua de mar por O.I., encontrando que, por su simplicidad y facilidad para ajustar los parámetros de sintonía, los controladores más utilizados son los de tipo PID, y en particular, los controladores PI.

En el 2013, Sobana y Panda (2013) resumen el estado del arte del control automático de plantas desalinizadoras por O.I., y resaltan que hasta la fecha, los controladores PI siguen dominando la industria, mientras que; los controladores avanzados se han implementado principalmente en módulos experimentales.

En esta sección se resumen los trabajos reportados en la literatura sobre las principales estrategias de control aplicadas en plantas desalinizadoras de agua de mar.

1.4.2. Control clásico

Inicialmente, Mindler y Epstein (1986) proponen un sistema de control on/off que trabaja bien, pero que tiene la desventaja de necesitar grandes instalaciones de almacenamiento de permeado para compensar los cambios en la demanda, y de ese modo, evitar el frecuente encendido y apagado de la planta.

Posteriormente, Alatiqi y colaboradores (Alatiqi et al., 1989) proponen un sistema de control PID multilazo, con un controlador de presión y un controlador de pH. La sintonía de los controladores desacoplados se realiza utilizando el método BLT.

Riverol y Pilipovik (2005) diseñaron controladores PID perfectamente desacoplados para una planta de desalinización por O.I. La implementación en la planta real muestra que el sistema de control no se desempeña tan bien como teóricamente se espera.

Kim et al. (2009), diseñan controladores PID sintonizados mediante algoritmos inmuno-genéticos. Posteriormente verifican que los controladores diseñados se desempeñan mejor que controladores sintonizados con la técnica de Zieglers-Nichols.

Rathore et al. (2013), utilizan un algoritmo de optimización por enjambre de partículas para sintonizar los controladores PID de una planta de desalinización por ósmosis inversa, y muestran que, los controladores sintonizados de esta manera tienen un mejor performance que aquellos sintonizados con la técnica de Zieglers-Nichols.

Es bien conocido que para el control de plantas multivariables complejas, la aplicación de controladores PID convencionales no conduce a resultados satisfactorios (Rivas-Perez et al. 1987), y por ello, diversos investigadores han propuesto el uso de controladores avanzados.

1.4.3. Control avanzado

Robertson et al. (1996) utilizan el modelo desarrollado por Alatiqi para proponer la implementación de un controlador de matriz dinámica (DMC) capaz de monitorear y controlar cinco variables del sistema: la temperatura y el pH de la alimentación, y el flujo, la conductividad y la presión del permeado.

Assef et al. (1999) implementan un controlador predictivo basado en modelo con restricciones (CMPC) para una planta experimental de desalinización por O.I. El modelo para el diseño del controlador es obtenido mediante una prueba escalón. El sistema de control tiene dos entradas (flujo de ácido, flujo de rechazo) y cuatro salidas (pH de la alimentación, conductividad del permeado, flujo de permeado y presión transmembrana). Los resultados de campo muestran un desempeño superior a controladores PID convencionales.

Burden et al. (2001) diseñan un controlador CMPC y comparan su desempeño frente a un controlador estándar proporcional-integral (PI) en un módulo experimental de ósmosis inversa con membranas de fibra hueca B-9 Permasep.

Zilouchian y Jafar (2001) diseñan un sistema de control inteligente de tipo neurodifuso para el control multivariable de una planta prototipo de desalinización de agua de mar mediante O.I.

Abbas (2006) diseña un controlador de matriz dinámica para el modelo matemático presentado por Alatiqui et al. (1989). El controlador diseñado mostró ser superior a los controladores PID convencionales.

Gambier et al. (2006) utilizan la técnica de optimización multiobjetivo basada en Pareto para sintonizar de forma simultánea los controladores interactuantes de un sistema de control óptimo de una planta experimental de desalinización de agua de mar por O.I.

McFall et al. (2008) diseñan un controlador no lineal basado en Lyapunov para el modelo matemático de un proceso de desalinización por O.I. que opera con alta recuperación. La evaluación del sistema de control muestra su capacidad para compensar las perturbaciones en la concentración de la alimentación.

Bartman et al. (2009), proponen un controlador robusto que produce una respuesta adecuada a las variaciones en el flujo del agua que ingresa como alimentación a la planta experimental estudiada.

Al-haj et al. (2010) diseñan un controlador CMPC para el modelo matemático que desarrollaron previamente (Al-haj et al., 2009). Se resalta que las variables manipuladas son el flujo y la presión de la alimentación en vez de: el pH y la presión, pues esta última combinación de variables no guarda relación con lo que sucede realmente en la práctica. Los resultados de simulación muestran buen performance y robustez del controlador diseñado.

Gambier y Badreddin (2011), utilizan el enfoque de optimización paramétrica multiobjetivo para obtener parámetros de sintonía que hacen más robustos a los lazos de control.

Moncada (2012) diseña un controlador predictivo basado en modelo para el modelo matemático multivariable de Alatiqi et al. (1989). Los resultados de simulación muestran un mejor desempeño que los controladores PID con los cuales se compara.

Madaeni et al. (2015) realizaron el modelado y control basado en redes neuronales artificiales de la unidad de tratamiento de agua por O.I. de una planta de generación de energía eléctrica.

Es importante observar que, aunque se reportan algunas aplicaciones de controladores basados en redes neuronales artificiales para plantas de desalinización de agua, aún existen muchos problemas de carácter teórico y práctico que no se abordan en los trabajos referidos.

1.5. Objetivos de la tesis

1.5.1. Objetivo general

Obtener un modelo matemático multivariable de una planta desalinizadora de agua de mar por O.I. empleando las ecuaciones de conservación de la materia y energía; así como, diseñar un controlador multivariable basado en R.N.A. para las variables críticas de dicha planta con el fin de aumentar su eficiencia.

1.5.2. Objetivos específicos

- Estudiar los fundamentos teóricos de la desalinización de agua de mar mediante O.I.
- Obtener un modelo matemático multivariable de la unidad de O.I. de una planta piloto de desalinización de agua de mar.
- Diseñar un controlador multivariable basado en R.N.A para las variables críticas de la planta objeto de estudio.
- Realizar una propuesta de implementación práctica del sistema de control diseñado

2. MODELADO MATEMÁTICO DE LA UNIDAD DE O.I. DE UNA PLANTA PILOTO DE DESALINIZACIÓN DE AGUA DE MAR

2.1. Fundamentos de la desalinización de agua de mar por O.I.

2.1.1. Introducción

En una planta industrial de desalinización por O.I., el agua de mar primero pasa por una serie de etapas de pretratamiento, las cuales normalmente son: filtración, coagulación, floculación, sedimentación, flotación por aire disuelto, filtración por medio granular y/o con filtros de cartucho, dosificación de inhibidores de incrustaciones y dosificación de biosidas. Una vez pre tratada, el agua marina es bombeada a alta presión hacia los bastidores de ósmosis inversa, donde ocurre la desalinización propiamente dicha. Posteriormente, tiene lugar el postratamiento del producto, el cual busca acondicionarlo a las especificaciones de calidad establecidas por la legislación vigente o en los estándares internacionales (Wang, 2008). La Figura 2.1 muestra un diagrama de flujo simplificado de una planta industrial de desalinización de agua de mar mediante O.I.

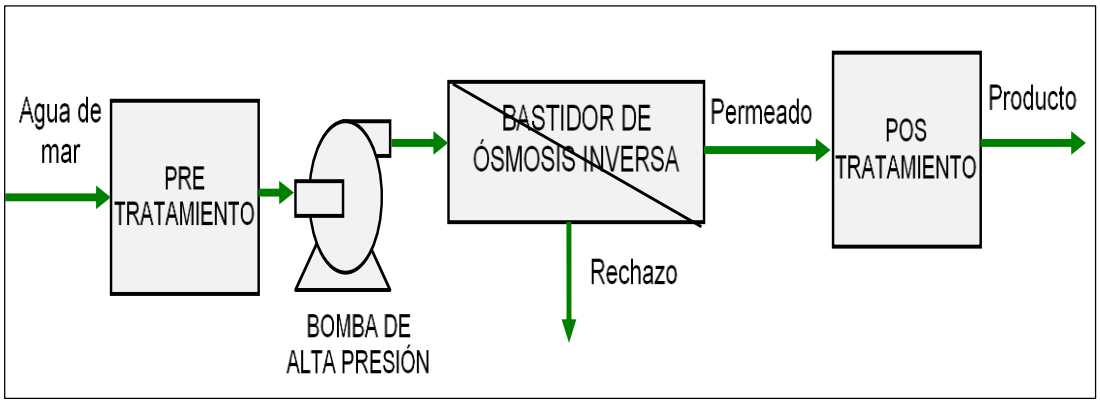

Figura 2.1. Diagrama de flujo simplificado de una planta industrial de desalinización de agua de mar por O.I.

2.1.2. Etapas del proceso de obtención agua desalinizada por O.I.

Las principales etapas para la obtención de agua desalinizada utilizando el proceso basado en el fenómeno de la ósmosis inversa son las siguientes (Voutchkov, 2013):

- Captación y bombeo.

La captación del agua de mar es un componente clave en toda planta de desalinización, pues, el tipo y localización de las tomas tiene un impacto directo sobre la calidad del agua fuente y los costos de producción.

Actualmente las instalaciones de captación de agua fuente para plantas de desalinización se clasifican en dos categorías: tomas abiertas y tomas sub superficiales.

Las tomas abiertas colectan el agua fuente directamente desde la superficie del cuerpo de agua, vía una estructura costa adentro o costa afuera y mediante tuberías que interconectan estas estructuras con la planta de desalinización. Las tomas sub superficiales son utilizados comúnmente para colectar el agua salina desde algún acuífero cercano a la costa.

Hoy en día las tomas abiertas en el océano son las tecnologías más utilizadas en todo el mundo para captar agua de mar porque pueden instalarse prácticamente en cualquier lugar y construirse de cualquier tamaño (Kucera, 2014). Aunque las tomas abiertas son adecuadas para plantas de desalinización de cualquier tamaño, su efectividad-costo depende de algunos factores relacionados con su ubicación, entre los cuales se pueden mencionar: el tamaño de la planta, la profundidad y geología del fondo del océano y el impacto de las potenciales fuentes de contaminación (aguas residuales, tráfico de embarcaciones, puertos con grandes actividades industriales, etc.).

Las tomas abiertas son comúnmente clasificadas como estructuras costa adentro o costa afuera. Las tomas costa adentro (Figura 2.2) han encontrado aplicación principalmente en plantas de desalinización térmicas o híbridas y normalmente consisten de canales grandes y profundos que terminan en una cámara de concreto desde donde se bombea el agua hacia las unidades de pretratamiento; y las tomas abiertas costa afuera comúnmente consisten de una estructura de entrada con tapa de velocidad (Figura 2.3), uno o más conductos para el agua captada, rejas separadoras de basura, filtros finos y estaciones de bombeo.

Las tomas sub superficiales son preferidas sobre las tomas abiertas debido a que el agua salina que colectan usualmente tiene una mejor calidad en términos de sólidos, arcilla, aceites y grasas, contenido de algas, contaminación orgánica natural y microorganismos acuáticos (Lior, 2013) Los tipos más comunes de tomas sub superficiales son: pozos verticales (Figura 2.4), pozos horizontales direccionalmente perforados (Figura 2.5), pozos horizontales tipo Ranney (Figura 2.6) y galerías de infiltración (Figura 2.7).

Figura 2.2. Planta desalinizadora de la bahía de Tampa (Voutchkov, 2013).

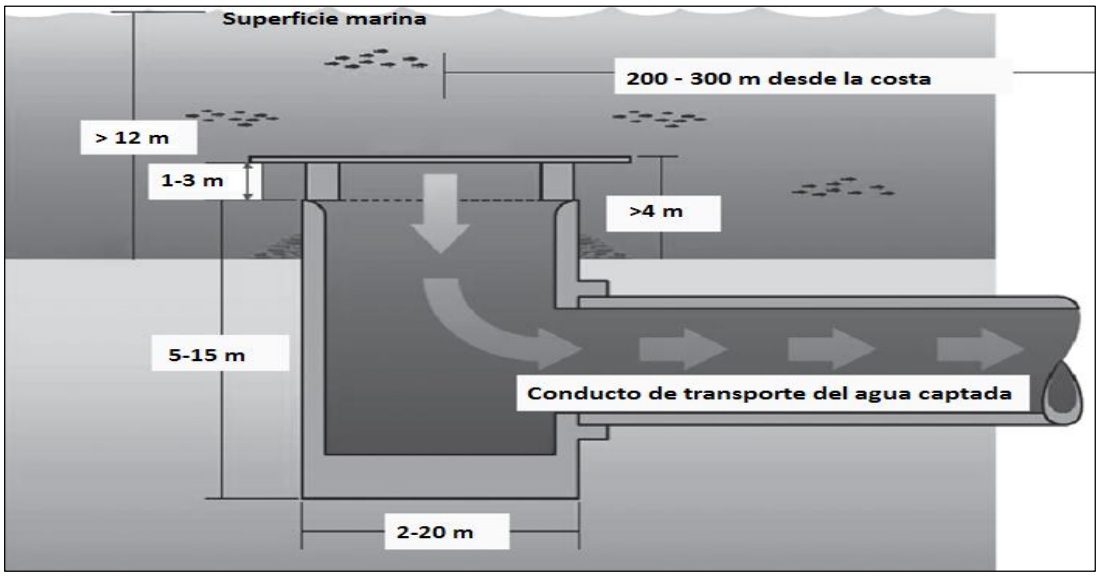

Figura 2.3. Toma abierta de captación de agua de mar tipo tapa de velocidad (Voutchkov, 2013).

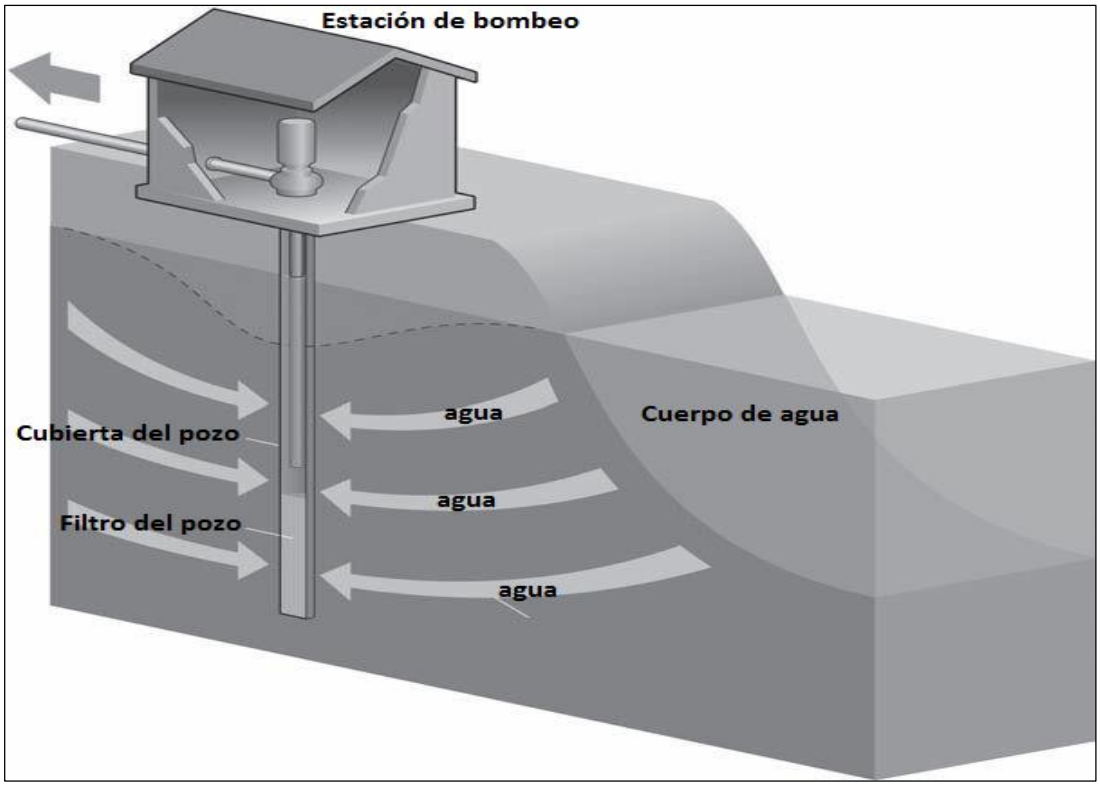

Figura 2.4. Captación de agua de mar con un pozo vertical (Cotruvo, 2010).

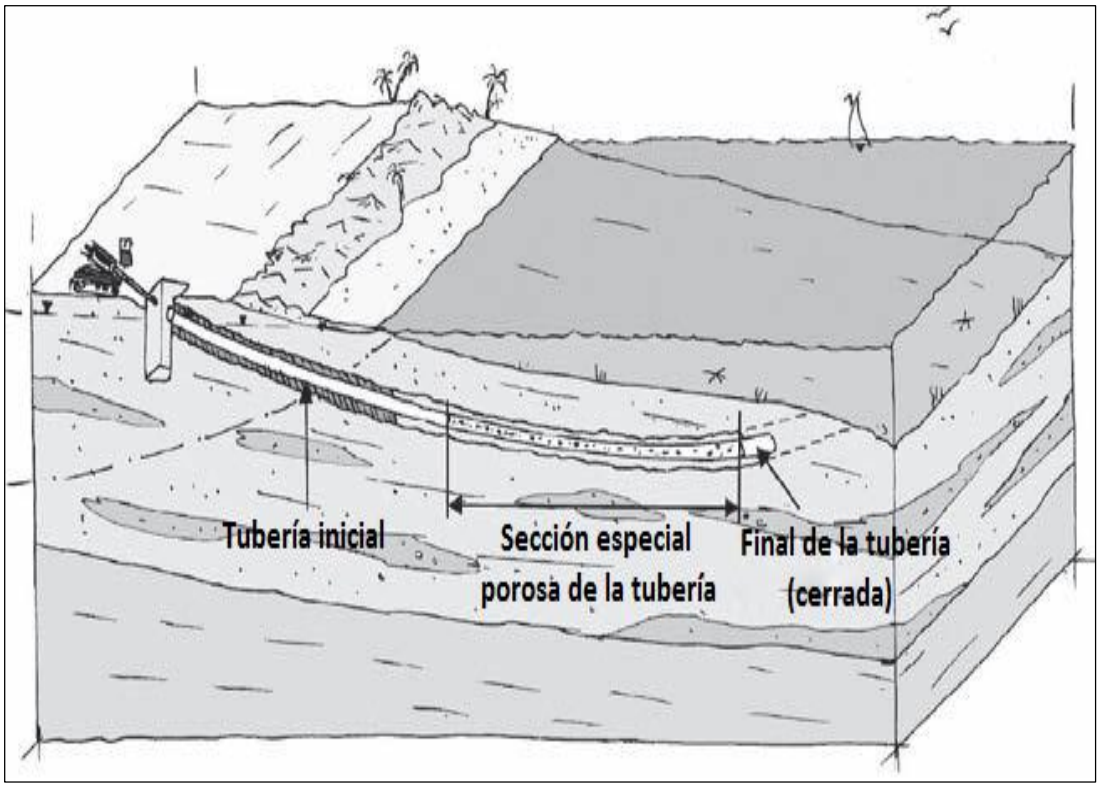

Figura 2.5. Captación de agua de mar con un pozo horizontal direccionalmente perforado (Voutchkov, 2013).

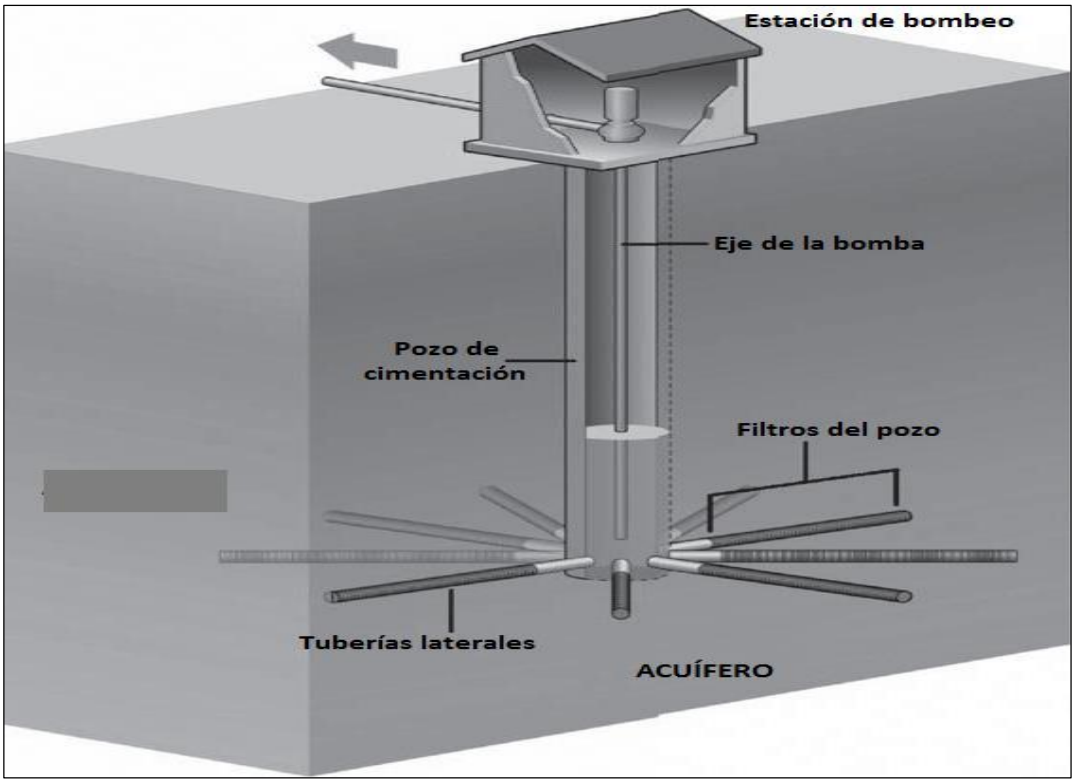

Figura 2.6. Captación de agua de mar con un pozo horizontal tipo Ranney (Cotruvo, 2010).

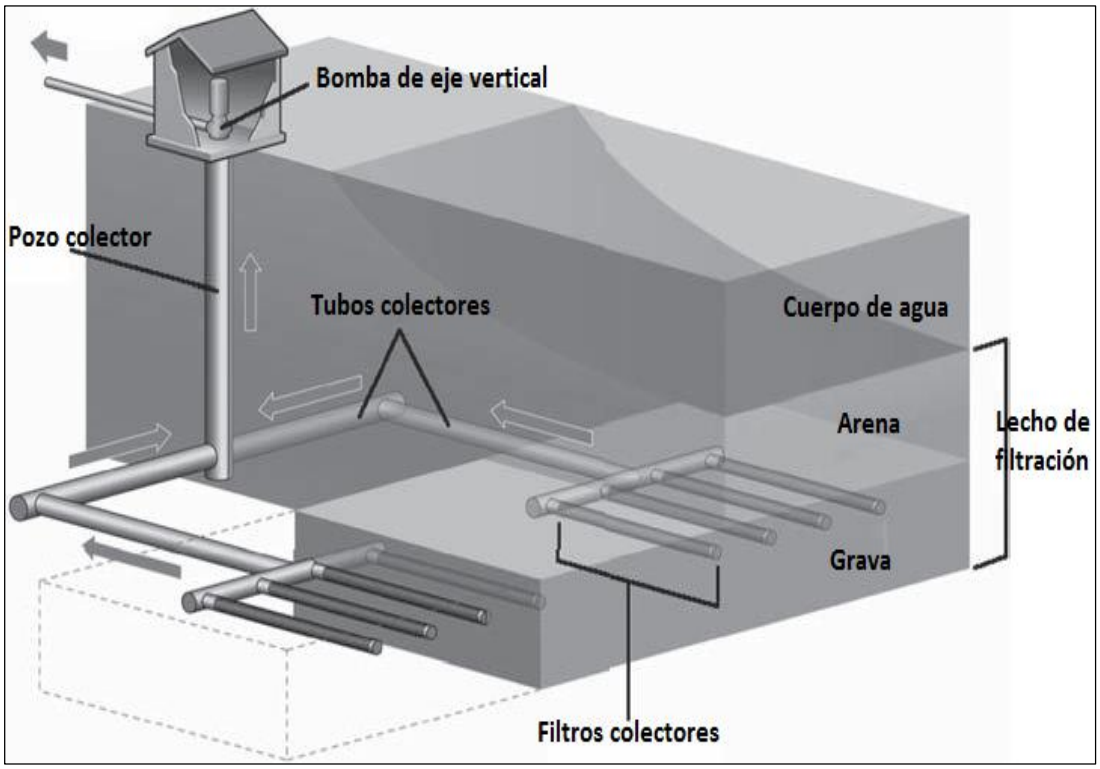

Figura 2.7. Captación de agua de mar mediante galerías de infiltración (Cotruvo, 2010).

- Pretratamiento.

Los procesos de pretratamiento son fundamentales para garantizar una calidad adecuada del agua de alimentación a los bastidores de ósmosis inversa. No existe un conjunto estándar de operaciones de pretratamiento a realizar en una planta de desalinización, pues, estas dependen del tipo y la cantidad de contaminantes presentes en el agua fuente, sin embargo; en una planta típica de desalinización de agua de mar por O.I., normalmente se lleva a cabo el siguiente pretratamiento: (1) se dosifica hipoclorito de sodio en la cámara de entrada para que actúe como biocida y agente oxidante, (2) se dosifica cloruro férrico en la cámara de floculación como agente floculante de los sólidos en suspensión y coloides, (3) se eliminan los flóculos formados en los tanques de flotación por aire disuelto, (4) se eliminan las partículas más pequeñas (aquellas que no pudieron flocularse) en filtros de arena y filtros de medio granular, (5) se realiza una purificación adicional en filtros de carbón activado, filtros de cartucho o membranas de nanofiltración, (6) se agregan agentes anti incrustantes para prevenir la formación de incrustaciones en la superficie de las membranas, y (7) se agrega bisulfito de sodio para proteger a las membranas de los agentes oxidantes, en especial del cloro libre.

- Bombeo a alta presión y desalinización por O.I. En esta etapa, se utilizan bombas de alta presión para enviar el agua pre tratada hacia los bastidores de ósmosis inversa, en los que tiene lugar el proceso de desalinización. Un bastidor de ósmosis inversa es el conjunto de recipientes a presión, módulos de membrana y tuberías que permiten llevar a cabo el proceso de O.I. (Figura 2.8).

Figura 2.8. Bastidores de ósmosis inversa (Cipollina, 2009).

- Recuperación de energía. Se utilizan ruedas Pelton, turbinas Francis o cámaras isobáricas para transferir la energía en forma de presión de la corriente de rechazo a la corriente de alimentación a los bastidores. Esto permite aumentar la eficiencia energética de la planta.

- Postratamiento. Se remineraliza el permeado con dióxido de carbono, calcita e hipoclorito de sodio para obtener un producto con las especificaciones de calidad deseadas.

2.1.3. Estructuras, materiales y configuración de las membranas de O.I.

En 1971 la Dow Chemical Company fabricó membranas con triacetato de celulosa en la configuración de fibra hueca (Figura 2.9). Estas membranas fueron las primeras en ser comercializadas (Johnson, 2009).

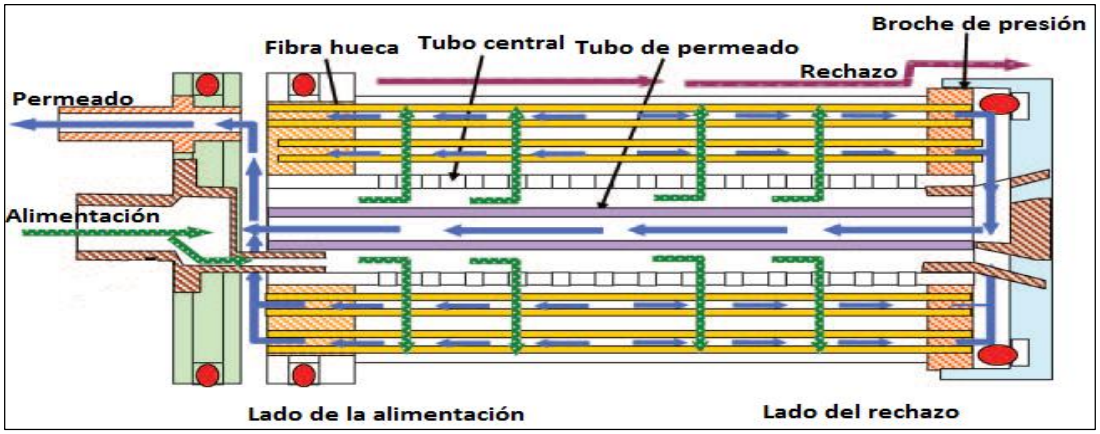

Figura 2.9. Módulo de membrana de fibra hueca (Lior, 2013).

Una de las principales ventajas de las membranas de fibra hueca, es la compensación de su bajo flux por medio de una gran área de transferencia, lo cual reduce el riesgo de polarización de la concentración. Y una de sus grandes desventajas es la dificultad para remover la suciedad e incrustaciones que tienden a formarse en su superficie.

A pesar de la ventajosa capacidad de las fibras de triacetato de celulosa para tolerar el cloro libre, Dow preocupado por las limitaciones intrínsecas presentes en la naturaleza química de las membranas y por las dificultades de fabricación de los módulos de fibra hueca, en 1985 compró la corporación FilmTec y de eso modo ganó acceso a la tecnología de membranas de poliamida y de construcción de elementos enrollados en espiral.

Independientemente de su configuración cilíndrica, un módulo de ósmosis inversa enrollado en espiral es un dispositivo formado esencialmente por hojas planas, donde el flujo es de tipo cruzado, pues, mientras el agua de alimentación pasa axialmente a través del módulo, el permeado se mueve en una trayectoria espiral en dirección radial hacia el tubo colector.

Actualmente, la ingeniería de módulos de ósmosis inversa enrollados en espiral es conducida por la necesidad de reducir costos y por el deseo de obtener el máximo provecho de esta última tecnología de membranas (Johnson, 2009). Las principales partes de estos elementos son: el espaciador de alimentación, el espaciador del permeado, el tubo de permeado y la tapa lateral (Figura 2.10).

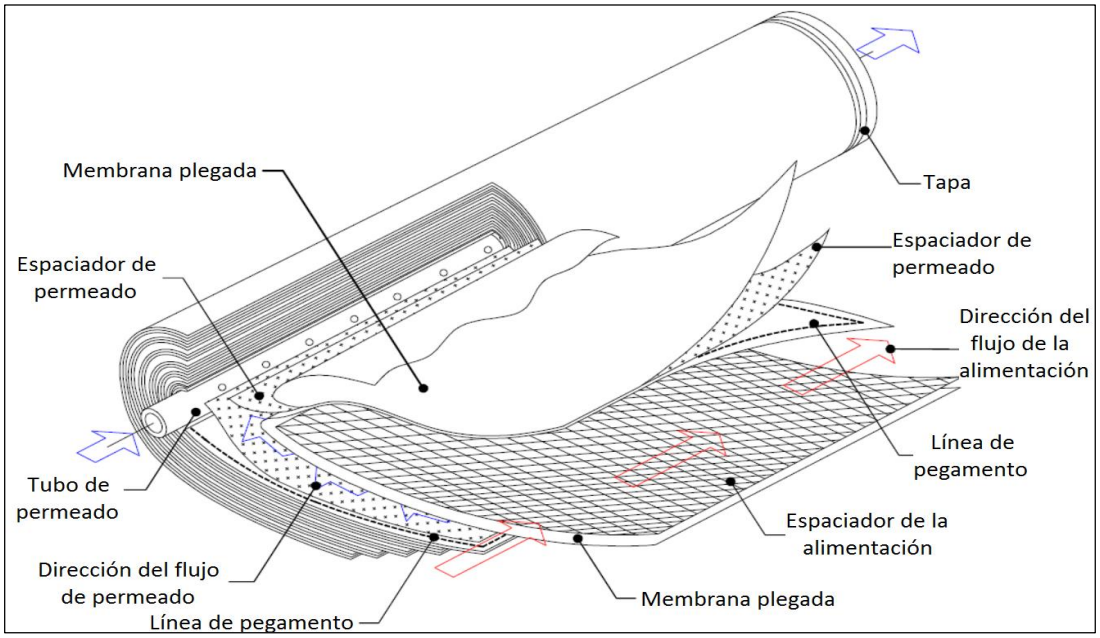

Figura 2.10. Configuración de una membrana de ósmosis inversa enrollada en espiral (Johnson, 2009).

Los elementos de la membrana se instalan en serie dentro de un recipiente llamado recipiente de presión (Figura 2.11). En plantas de mediana capacidad comúnmente se utilizan módulos estándar de 8 pulgadas de diámetro por 40 pulgadas de longitud, en una cantidad de seis a ocho elementos por recipiente de presión (Voutchkov, 2013). Un análisis detallado de costo-beneficio (Wilf et al., 2008) demostró que la instalación de ocho elementos en vez de siete o seis resulta más ventajoso económicamente para plantas medianas y grandes.

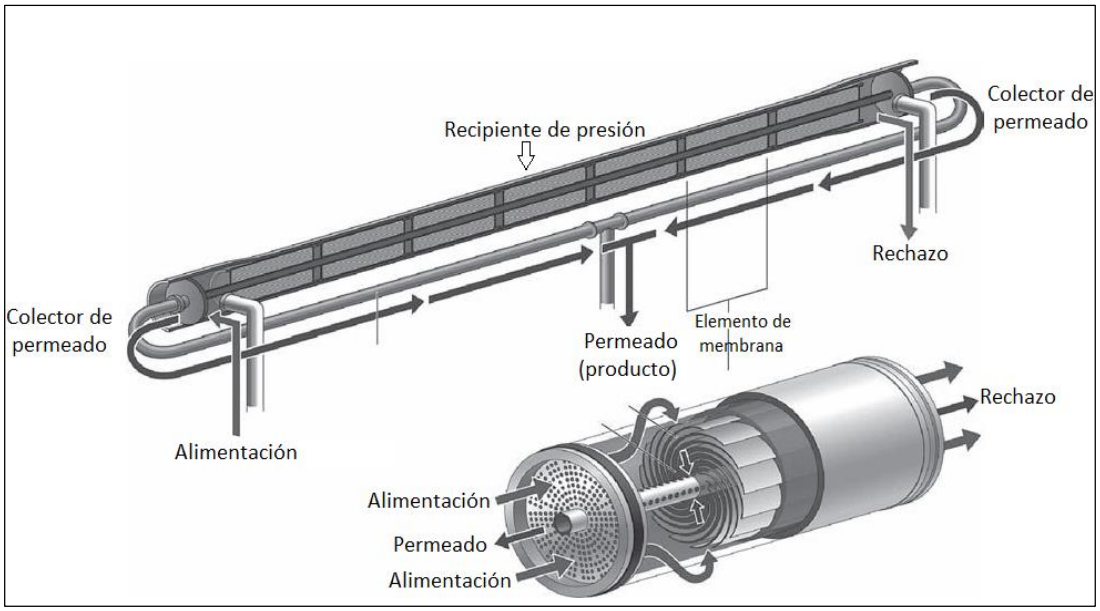

Figura 2.11. Configuración de los elementos de la membrana dentro de un recipiente de presión (Voutchkov, 2013).

2.1.4. Parámetros del proceso de desalinización por O.I.

- Rechazo de sales (RS)

Es una medida relativa de cuanto de la sal que estaba inicialmente presente en la alimentación es retenida y rechazada por la membrana. Su valor depende del proceso de fabricación de la membrana y de los materiales de construcción utilizados. Se calcula utilizando la relación:

$$RS = (1 - Cp/Ca) \times 100, \quad (2.1)$$

donde Cp y Ca son las concentraciones del permeado y de la alimentación respectivamente.

- Recuperación (Recovery)

Es la cantidad (expresada en porcentaje) de permeado producido a partir del agua de alimentación. Su valor depende de la formación de incrustaciones sobre la superficie de la membrana, de la polarización de la concentración y de las restricciones propias de los equipos e instalaciones. Para membranas enrolladas en espiral la recuperación típica es de 10%, y para membranas de fibra hueca este valor llega hasta 60% (Cipollina, 2009). Se calcula con:

$$Recovery = (Fp/Fa) \times 100, \quad (2.2)$$

donde Fp es el flujo de permeado y Fa el flujo de la alimentación.

- Presión osmótica $(\boldsymbol{\pi})$

Es la presión mínima que debe tener la corriente más concentrada en sales para evitar que el solvente de la corriente de menor concentración fluya espontáneamente hacia ella. Puede estimarse con la ecuación de Van't Hoff:

$$\pi = RT \sum x_i, \tag{2.3}$$

Donde R es la constante universal de los gases ideales, T la temperatura absoluta y $\sum x_i$ la suma de las fracciones molares de todos los solutos disueltos en el agua.

- Presión transmembrana o presión impulsora neta (NDP)

Es la presión real que dirige el transporte de solvente desde el lado de la alimentación hacia el lado del permeado. Este parámetro toma diferentes valores en la dirección axial de los elementos de membrana, y para cálculos aproximados, suele utilizarse la NDP promedio, la cual es definida como la diferencia entre la presión de la alimentación (P_f) y todas las fuerzas que se oponen al movimiento del permeado a través de la membrana. Estas fuerzas están constituidas por: la presión osmótica promedio (π), la presión promedio del permeado (P_P) y la caída de presión promedio de la corriente de alimentación $(0.5P_d)$. Es decir:

$$NDP = P_f - (\pi + P_P + 0.5P_d) \tag{2.4}$$

- Permeabilidad al soluto $(\boldsymbol{k_A})$

Es una medida de la facilidad con la que los solutos pueden atravesar una membrana. En general tiene un valor pequeño porque las membranas de desalinización por O.I. se diseñan para evitar el paso de las sales. Tiene unidades de longitud/tiempo.

- Permeabilidad al solvente $(\boldsymbol{k_B})$

Es la facilidad con la que la membrana permite el tránsito del solvente a través de ella. Depende exclusivamente del material de fabricación de la membrana y puede entenderse como la habilidad de una membrana para transportar más solvente que otra membrana a través de la misma área superficial y bajo la misma presión aplicada. Tiene unidades de flujo/(área*presión).

- Polarización de la concentración

Es el término utilizado para describir la acumulación de los solutos rechazados en la superficie de la membrana. Este fenómeno tiene los siguientes efectos

negativos: (1) disminución del flux del solvente debido al incremento de la presión osmótica en la capa límite formada en la superficie de la membrana en contacto con la corriente de alimentación, (2) aumento del flux de soluto a través de la membrana debido al incremento del gradiente de concentración entre ambos lados de esta, (3) precipitación en la superficie de la membrana de los solutos que exceden su límite de solubilidad dando lugar a la formación de incrustaciones, (4) aumento del ensuciamiento de la membrana debido al material coloidal y particulado que se adhiere a su superficie.

La polarización de la concentración dificulta el modelado del proceso de desalinización porque es difícil cuantificarla. En el caso particular del flujo a través de membranas de hojas planas, se puede partir de la ecuación de difusión-convección de Navier-Stokes para calcular el perfil de concentraciones (C) de los solutos en la capa límite del lado de la alimentación (Bhattacharyya y Williams, 1992c):

$$U\frac{\partial C}{\partial z} + V\frac{\partial C}{\partial y} - D_{Sw}\left(\frac{\partial^2 C}{\partial z^2} + \frac{\partial^2 C}{\partial y^2}\right) = 0, \tag{2.5}$$

donde $U\ y\ V$ son velocidades.

Si además se hace la suposición de que la capa límite está estancada $\xi(z) = \xi$ y que la concentración no cambia en la dirección z (Figura 2.12), la ecuación (2.5) se reduce a:

$$V\frac{\partial C}{\partial y} = D_{Sw}\frac{\partial^2 C}{\partial y^2} \tag{2.6}$$

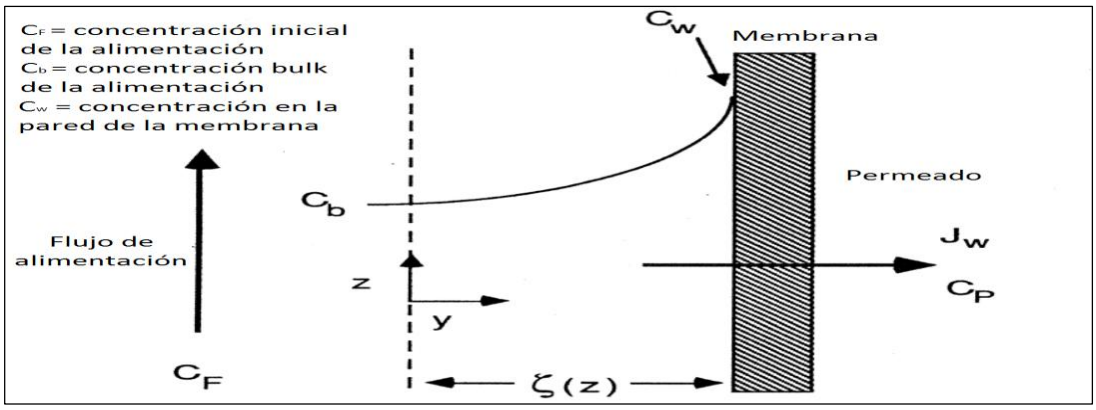

Figura 2.12. Polarización de la concentración.

La ecuación (2.6) puede integrarse fácilmente para dar la muy conocida y ampliamente utilizada teoría de la película de Brian (Brian, 1966):

$$\frac{C_W - C_P}{C_F - C_P} = exp\left(\frac{V_W \xi}{D_{Sw}}\right) \tag{2.7}$$

2.2. Descripción de la planta piloto de desalinización de agua de la PUCP

En el laboratorio de Control Avanzado de la Pontificia Universidad Católica del Perú se ha instalado una planta piloto para la desalinización de agua con la tecnología de ósmosis inversa (Figura 2.13). Esta planta tiene los siguientes componentes:

- un tanque para el agua salada de alimentación,
- un tanque para el permeado,
- un tanque para el rechazo,
- un tanque para el ácido con el cual se modifica el pH de la alimentación,
- dos recipientes de presión con dos módulos de membrana cada uno,
- una bomba de alta presión para la corriente de alimentación a la planta,
- sensores de flujo, de presión, de temperatura y de pH,
- una válvula de control, y
- un tablero de control.

Figura 2.13. Planta piloto para desalinización de agua de la PUCP.

2.3. Modelado matemático de la unidad de O.I. de una planta piloto de desalinización de agua de mar

2.3.1. Selección del modelo de transporte a emplear

En esta sección se deduce un modelo matemático no lineal multivariable para simular el comportamiento dinámico del proceso que se lleva a cabo en los bastidores de ósmosis inversa de una planta piloto de desalinización.

Las relaciones entre las variables de proceso, y en particular entre las variables de control, se consiguen mediante balances de materia y energía.

Se considera que el transporte de solvente y solutos a través de la membrana se realiza por el mecanismo de solución-difusión (Bhattacharyya y Willians, 1992c), en cuyo modelo se asume que: (1) la membrana de ósmosis inversa no tiene poros y su capa superficial es homogénea. (2) El solvente y los solutos se disuelven en la capa superficial de la membrana y luego se difunden a través de ella. (3) La difusión del soluto es independiente de la del solvente y es exclusivamente debida al gradiente de potencial químico. (4) Los gradientes de potencial químico son consecuencia de las diferencias de presiones y concentraciones en ambos lados de la membrana.

Para tener en cuenta la polarización de la concentración, se utilizan los resultados presentados por Armijo y Condorhuaman (2012), los cuales están basados en la ley de Fick y en las ecuaciones de transferencia de materia.

2.3.2. Definición de los subsistemas permeado, rechazo y membrana

Suponiendo que la membrana es plana y que los flujos de alimentación, permeado y rechazo siguen las trayectorias indicadas por las flechas de color verde en la Figura 2.14, se caracteriza cada corriente por sus propiedades (flujo, composición, presión, temperatura, etc.).

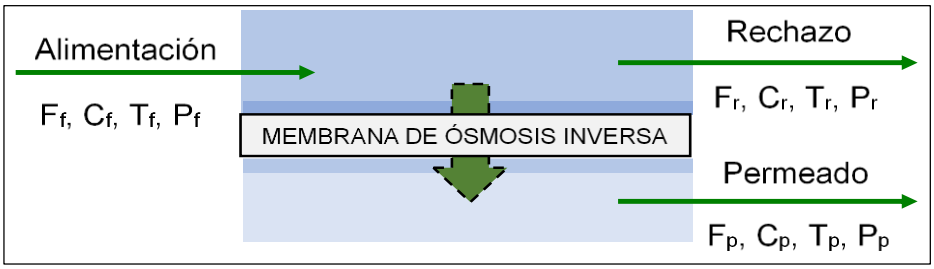

Figura 2.14. Esquema del proceso de desalinización por O.I.

Aunque los flujos, presiones, temperaturas y composiciones de todas las corrientes van cambiado a medida que estas avanzan en la dirección axial de cada elemento de membrana, en el presente trabajo se considera que estas variables están concentradas en un punto del espacio y que son independientes de la posición del fluido al cual están asociados. Esta simplificación permite obtener un modelo matemático de parámetros concentrados que puede escribirse utilizando ecuaciones diferenciales ordinarias (Bequette, 2003).

Bajo la suposición anterior, la dinámica del proceso solo podrá observarse definiendo un volumen de control para el lado del rechazo y otro para el lado del permeado, por lo que se definen los subsistemas rechazo, permeado y membrana de la Figura 2.15.

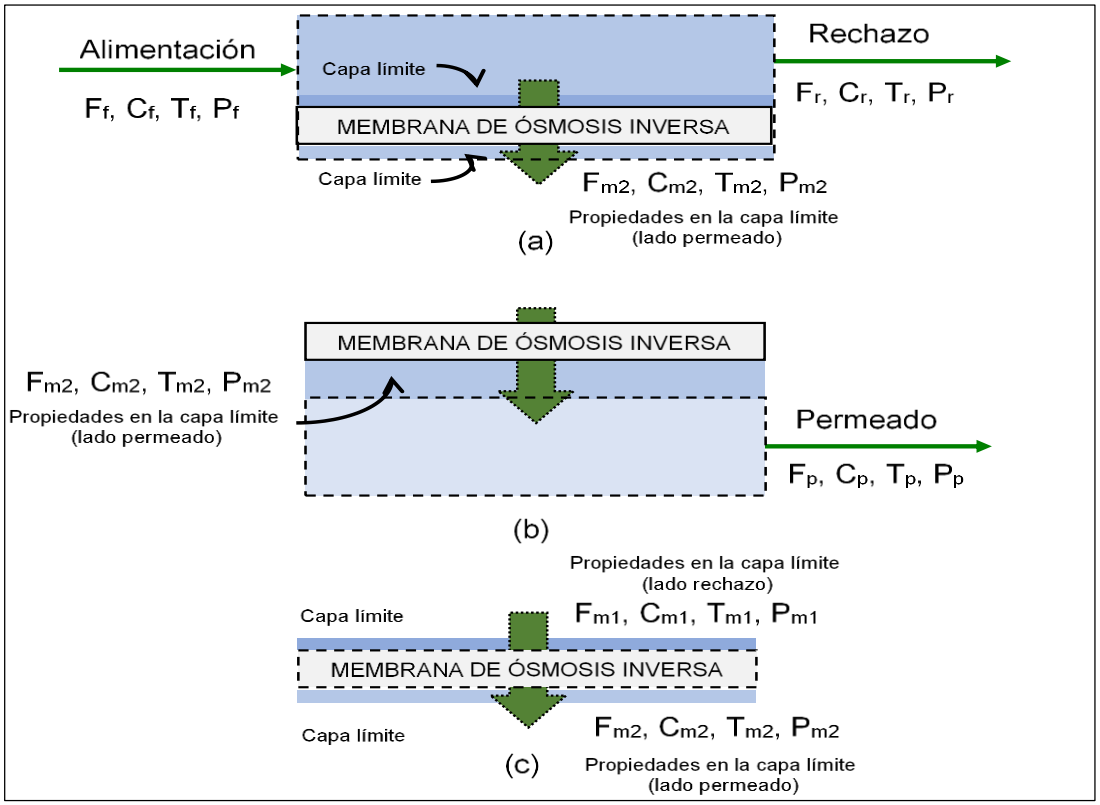

Figura 2.15. Definición de subsistemas y variables para el planteamiento de las ecuaciones de balance de materia y energía (las líneas a trazos definen cada subsistema). (a) Subsistema rechazo, (b) subsistema permeado y (c) subsistema membrana.

Considerando además que la transferencia de calor con el exterior es despreciable, se escriben las ecuaciones de balance.

2.3.3. Balances de materia, energía y relación entre presiones

2.3.3.1. Ecuaciones de balance en el subsistema rechazo

- Balance global de materia

$$\frac{dm_r}{dt} = F_f - F_r - F_{m2}, \tag{2.8}$$

donde m_r es la masa de la solución presente en el subsistema rechazo en cada instante de tiempo (holdup del subsistema rechazo), F_f es el flujo másico de alimentación, F_r el flujo másico de rechazo y F_{m2} el flujo másico de solvente que pasa a través de la membrana hacia el lado permeado.

Como el agua es un fluido incompresible, la ecuación (2.8) se reduce a:

$$0 = F_f - F_r - F_{m2} \tag{2.9}$$

- Balance parcial de materia (balance de sales)

$$\frac{dC_r}{dt} = \frac{1}{m_r}\left[F_f\left(C_f - C_r\right) - F_{m2}(C_{m2} - C_r)\right], \tag{2.10}$$

donde C_r es la concentración de la solución presente en el subsistema rechazo en cada instante de tiempo, C_f la concentración de la alimentación y C_{m2} la concentración en la capa límite del lado permeado.

- Balance de energía

$$\frac{dT_r}{dt} = \frac{1}{m_r}\left[F_f\left(T_f - T_r\right) - F_{m2}(T_{m2} - T_r)\right], \tag{2.11}$$

donde T_r es la temperatura de la solución presente en el subsistema rechazo en cada instante de tiempo, T_f la temperatura de la alimentación, y T_{m2} la temperatura en la capa límite del lado permeado.

La suposición de parámetros concentrados implica que dentro del subsistema rechazo no hay gradientes de temperatura, por lo que $T_{m2} = T_r$ y la ecuación 2.11 se reduce a:

$$\frac{dT_r}{dt} = \frac{1}{m_r}\left[F_f\left(T_f - T_r\right)\right] \tag{2.12}$$

- Relación entre presiones

$$P_r = P_f - Ploss_r, \tag{2.13}$$

donde P_r es la presión de la corriente de rechazo, P_f la presión de la alimentación y $Ploss_r$ las pérdidas de presión por fricción en el lado rechazo.

2.3.3.2. Ecuaciones de balance en el subsistema permeado

- Balance global de materia

$$\frac{dm_p}{dt} = F_{m2} - F_p, \tag{2.14}$$

donde m_p es la cantidad de materia (masa) presente en el subsistema permeado en cada instante de tiempo (holdup del subsistema permeado), F_{m2} el flujo másico de solvente que pasa a través de la membrana desde el lado rechazo hacia el lado permeado y F_p el flujo másico de permeado.

Como la incompresibilidad del agua impide la acumulación de materia en el volumen de control del subsistema permeado, siempre se cumplirá que $dm_p/dt = 0$, y con ello, la ecuación (2.14) se simplifica a:

$$0 = F_{m2} - F_p \tag{2.15}$$

- Balance parcial de materia (balance de sales)

$$\frac{dC_p}{dt} = \frac{1}{m_p}\left[F_{m2}\left(C_{m2} - C_p\right)\right], \tag{2.16}$$

donde C_p es la concentración del subsistema permeado en cada instante de tiempo y C_{m2} la concentración en la capa límite del lado permeado.

Como la polarización de la concentración en el lado permeado es pequeña, la ecuación (2.16) puede simplificarse a:

$$0 = C_{m2} - C_p \tag{2.17}$$

- Balance de energía

$$\frac{dT_p}{dt} = \frac{1}{m_p}\left[F_{m2}\left(T_{m2} - T_p\right)\right], \tag{2.18}$$

donde T_p es la temperatura del subsistema permeado en cada instante de tiempo y T_{m2} la temperatura en la capa límite del lado permeado.

Como dentro del subsistema rechazo no hay gradientes de temperatura, la ecuación (2.18) se reduce a:

$$0 = T_{m2} - T_p \tag{2.19}$$

- Relación entre presiones

$$P_p = P_{m2} - Ploss_p \tag{2.20}$$

donde P_p es la presión del permeado, P_{m2} la presión en la capa límite del lado permeado y $Ploss_p$ son las pérdidas por fricción en el lado permeado.

2.3.3.3. Ecuaciones de balance en el subsistema membrana

Se usa el modelo de solución-difusión (Bhattacharyya y Willians, 1992c) para predecir los flujos de solvente (F_{m2}) y solutos (F_{Am2}) a través de la membrana:

$$F_{m2} = k_B A(\Delta P - \Delta\pi) \tag{2.21}$$

$$F_{Am2} = k_A A(C_{m1} - C_{m2}) \tag{2.22}$$

donde k_B es la permeabilidad al solvente, A el área de la membrana, ΔP la presión transmembrana, $\Delta\pi$ la diferencia de presiones osmóticas, k_B la permeabilidad al soluto, C_{m1} y C_{m2} las concentraciones en las capas límite del lado rechazo y permeado respectivamente.

Se considera que no hay acumulación de materia ni de energía en el subsistema membrana y se aplican las ecuaciones de balance para obtener:

$$0 = F_{m1} - F_{m2} \tag{2.23}$$

$$0 = T_{m1} - T_{m2} \tag{2.24}$$

Para relacionar C_{m2} con C_{m1}, se parte de las definiciones de concentración de solutos $C_{Am2} = m_A/m_{total}$ y concentración de solvente $C_{Bm2} = m_B/m_{total}$ en la capa límite del lado permeado, y se escribe:

$$\frac{m_A}{C_{Am2}} = \frac{m_B}{C_{Bm2}} \tag{2.25}$$

Como C_{Am2} y C_{Bm2} son constantes, la derivada respecto al tiempo de ambos lados de la ecuación (2.25) da:

$$\frac{F_A m2}{C_{Am2}} = \frac{F_{m2}}{C_{Bm2}} \tag{2.26}$$

Luego de reemplazar las ecuaciones (2.21) y (2.22) en la ecuación (2.26) y agrupar convenientemente los términos se obtiene:

$$0 = C_{m1} - C_{m2}[1 + k(\Delta P - \Delta\pi)], \tag{2.27}$$

donde la selectividad de la membrana k es la relación k_B/k_A.

La polarización de la concentración se tiene en cuenta utilizando la ecuación presentada por Armijo y Condorhuaman (2012), la cual relaciona la concentración de la capa límite del lado permeado con la de la alimentación:

$$0 = C_{m2} - \frac{C_f}{1+k(1-\theta)(\Delta P - \Delta\pi)}, \tag{2.28}$$

Donde el corte θ un parámetro adimensional definido como:

$$\theta = \frac{F_p}{F_f} \tag{2.29}$$

Adicionalmente, al considerar a la presión como un parámetro concentrado, se consigue la relación:

$$0 = P_{m1} - P_r \tag{2.30}$$

Finalmente, se utiliza la ecuación presentada por Lystern y Cohen (2007) para relacionar la presión osmótica $(\pi, \text{en bar})$ con la concentración $(C, en\ \text{mmol/L})$.

$$\pi = 4.572 \times 10^{-2} C - 1.797 \times 10^{-6} C^2 + 4.631 \times 10^{-9} C^3 \tag{2.31}$$

2.3.4. Relación entre la concentración de sales y la conductividad

Dado que en la práctica se cuantifica el contenido de sales de las corrientes de proceso midiendo la conductividad (en uS/cm) en vez de la concentración (en ppm), es necesario establecer un procedimiento fiable para convertir los valores de una unidad de medida a otra, pues se encontró por ejemplo, que las correlaciones de García (García, 2009) no dan buenos resultados en el rango de concentraciones típicas de una corriente de permeado.

Kohlrausch (Ball, 2015) demostró que, las conductividades molares (Λ) a bajas concentraciones de los electrolitos pueden calcularse utilizando la siguiente ecuación (ley de Kohlrausch):

$$\Lambda = \Lambda_0 - \mathrm{B}\sqrt{C}, \tag{2.32}$$

donde, Λ_0 es la conductividad molar límite (a dilución infinita) y B una constante que depende de la naturaleza del electrolito, de la temperatura y del disolvente. Para el cloruro de sodio (NaCl), Λ_0 es 126.45 S.cm^2.mol^{-1} y B puede aproximarse con:

$$B = 60.32 + 0.2289\Lambda_0 \tag{2.33}$$

Si se considera que el agua a desalinizar contiene únicamente al NaCl como electrolito, su conductividad $(Cond, en\ uS/cm)$ en función de su concentración $(C, en\ ppm)$ estará dada por:

$$Cond = \left[\frac{126.45 - 0.3692523\sqrt{C}}{58.440}\right] C \tag{2.34}$$

2.3.5. Relación entre el pH y la conductividad

El pH tiene una influencia directa sobre la carga eléctrica de la superficie de la membrana, y por ello, sobre el paso de sales a través de esta. A bajos pHs (pHs ácidos), el ion hidrógeno ataca a los grupos carboxilo de la superficie de la membrana y neutraliza su carga, reduciendo con ello su capacidad para rechazar a los iones negativos, y más aun, la relación entre el pH y la carga superficial de la membrana es lineal en un amplio rango de valores (Franks et al., 2009).

Dado que los datos experimentales presentados por Alatiqi (Alatiqi et al., 1989) muestran que el cambio de la conductividad por unidad de cambio de pH es prácticamente constante:

$$\frac{d(Cond_p)}{d(pH)} \approx Constante1 \tag{2.35}$$

En el presente trabajo se considera que los cambios en la carga superficial de la membrana debidos al pH se traducen como cambios en el valor de la constante de permeabilidad al soluto.

Para cuantificar el efecto del cambio de pH sobre la permeabilidad al soluto, se parte de la ecuación de transporte de sales escrita para dos condiciones: (1) antes del cambio de pH; y (2) después del cambio:

$$F1_{sal} = k1_A A\left(C1_{m1} - C1_p\right) \tag{2.36}$$

$$F2_{sal} = k2_A A\left(C2_{m1} - C2_p\right) \tag{2.37}$$

Como el pH no afecta al flujo de permeado y $F_{sal} = F_p \times C_p$, las ecuaciones (2.36) y (2.37) pueden escribirse convenientemente:

$$C1_p = k1_A \frac{A}{F_p}\left(C1_{m1} - C1_p\right) \tag{2.38}$$

$$C2_p = k2_A \frac{A}{F_p}\left(C2_{m1} - C2_p\right) \tag{2.39}$$

Puesto que $C1_p$ y $C2_p$ son pequeños en comparación con $C1_{m1}$ y $C2_{m1}$; puede considerarse sin pérdida de generalidad que $\frac{(C1_{m1}-C1_p)}{(C2_{m1}-C2_p)} \approx 1$, para que al dividir la ecuación (2.38) entre la (2.39) se obtenga:

$$\frac{C1_p}{C2_p} = \frac{k1_A}{k2_A} \tag{2.40}$$

Esto significa que k_A y C_p son directamente proporcionales, matemáticamente:

$$\frac{d(k_A)}{d(C_p)} = Constante2 \tag{2.41}$$

Puede demostrarse que $Constante2 = k_A/C_p$.

Por otro lado, la aplicación de la ley de Kohlrausch a una disolución electrolítica de NaCl a bajas concentraciones da como resultado:

$$Cond_p \approx 2C_p \tag{2.42}$$

Finalmente, se combinan las ecuaciones (2.35), (2.41) y (2.42), para obtener la relación entre la constante de permeabilidad al solvente (k_A) y el pH de la corriente de alimentación a los bastidores de ósmosis inversa:

$$\frac{d(k_A)}{d(pH_f)} = \frac{Constante1}{2} * \left(\frac{k_A}{C_p}\right) \tag{2.43}$$

2.4. Validación del modelo matemático obtenido

El modelo matemático obtenido está formado por las ecuaciones algebraicas y diferenciales que se presentan en forma resumida en la Tabla 2.1.

Para utilizar este modelo solo se necesita el valor de los parámetros: A, k_A, k_B, $\mathrm{Constante1}$ y los valores iniciales de las variables m_p y m_r.

Se valida el modelo comparando sus resultados con los datos experimentales presentados por Alatiqi et al. (1989), quienes utilizan una membrana de fibra hueca B-10 Permasep, que tiene un área superficial de 152 m^2 (Al-Bastaki et al., 1999). Los parámetros k_A y k_B son aquellos que hacen que la salida del modelo sea igual a la salida experimental en el punto normal de operación:

$F_f = 315.45\ g/s$

$P_f = 63.07\ bar$

$T_f = 25.00\ °C$

$C_f = 3000\ ppm$

$pH_f = 6.45$

$P_p = 1.01\ bar$

Tabla 2.1. Ecuaciones del modelo matemático del proceso de desalinización por O.I.

ECUACIÓN	SUBSISTEMA RECHAZO	SUBSISTEMA PERMEADO
Balance global de materia	$\frac{dm_r}{dt} = F_f - F_r - F_{m2}$	$\frac{dm_p}{dt} = F_{m2} - F_p$
Balance parcial de materia	$\frac{dC_r}{dt} = \frac{1}{m_r}\left[F_f\left(C_f - C_r\right) - F_{m2}(C_{m2} - C_r)\right]$	$\frac{dC_p}{dt} = \frac{1}{m_p}\left[F_{m2}\left(C_{m2} - C_p\right)\right]$
Balance de energía	$\frac{dT_r}{dt} = \frac{1}{m_r}\left[F_f\left(T_f - T_r\right) - F_{m2}(T_{m2} - T_r)\right]$	$\frac{dT_p}{dt} = \frac{1}{m_p}\left[F_{m2}\left(T_{m2} - T_p\right)\right]$
Relación entre presiones	$P_r = P_f - Ploss_r$	$P_p = P_{m2} - Ploss_p$

ECUACIÓN	SUBSISTEMA MEMBRANA
Flujo de solvente a través de la membrana	$F_{m2} = k_B A(\Delta P - \Delta\pi)$
Balance global de materia	$0 = F_{m1} - F_{m2}$
Parámetro θ	$\theta = \frac{F_p}{F_f}$
Relación entre C_{m2} y C_f	$0 = C_{m2} - \frac{C_f}{1 + k(1-\theta)(\Delta P - \Delta\pi)}$
Relación entre C_{m1} y C_{m2}	$0 = C_{m1} - C_{m2}[1 + k(\Delta P - \Delta\pi)]$
Balance de energía	$0 = T_{m1} - T_{m2}$
Relación entre presiones	$0 = P_{m1} - P_r$
Presión osmótica	$\pi = 4.572 \times 10^{-2}C - 1.797 \times 10^{-6}C^2 + 4.631 \times 10^{-9}C^3$

RELACIÓN ENTRE	ECUACIÓN
Concentración y conductividad	$Cond = \left[\frac{126.45 - 0.3692523\sqrt{C}}{58.440}\right]C$
Permeabilidad al solvente y pH de la alimentación	$\frac{d(k_A)}{d(pH_f)} = \frac{Constante1}{2} * \left(\frac{k_A}{C_p}\right)$

Ejecutando el programa p02AlatiqiInititalConditionsBL.m del anexo A se determinan los siguientes valores para k_A y k_B:

$k_A = 2.4500 \times 10^{-3} \quad \mathrm{m/s}$

$k_B = 1.3605 \times 10^{-2} \quad g/(s * m^2 * bar)$

El parámetro $Constante1$ es la pendiente de la ecuación que resulta al realizar una regresión lineal con los valores experimentales de $Condp$ vs pHf. Para este caso particular el valor de $Constante1$ es:

$Constante1 = -58.164 \ \mathrm{uS/cm}$

Debido a la naturaleza incompresible del agua, los valores de m_p y m_r permanecen constantes e iguales a sus valores iniciales. Por su definición estas variables pueden entenderse como la cantidad de materia contenida en los subsistemas permeado y rechazo respectivamente. Esto significa que son muy importantes en la dinámica del proceso porque representan la capacitancia del sistema. Para calcular m_p y m_r se debe conocer la geometría de los recipientes de presión y de los elementos de membrana, así como del volumen ocupado por las membranas, los espaciadores y el tubo colector. Un cálculo aproximado permite obtener los siguientes valores:

$m_r = 860 \ g$

$m_p = 700 \ g$

Para validar el modelo matemático propuesto se ejecutan los programas p03AlatiqiBLComparacionPFC.m y p04AlatiqiBLComparacionpHC.m del anexo A. Los resultados obtenidos se presentan en las Figuras 2.16, 2.17 y 2.18.

Como se observa en la figura 2.16, el modelo tiene un ajuste de 92.37% a los datos experimentales de la dependencia del flujo de permeado con la presión de la alimentación, mientras que, para la conductividad del permeado en función de la presión de la alimentación, el ajuste es solo de 74.57% (Figura 2.17). Y para la conductividad del permeado en función del pH de la alimentación, el grado de ajuste es de 78.71% (Figura 2.18)

Estos resultados señalan que el modelo se aproxima bastante bien a los datos experimentales y por ende puede ser utilizado en el diseño de sistemas efectivos de control para la planta objeto de estudio.

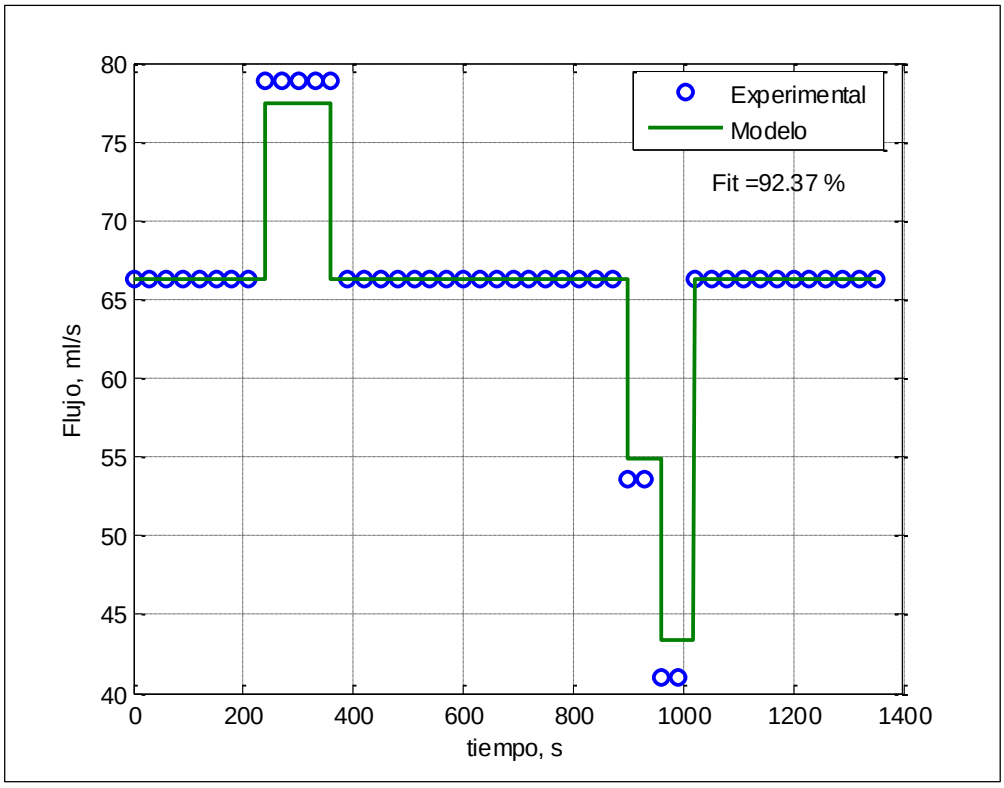

Figura 2.16. Resultados de la validación del modelo matemático. Flujo de permeado frente a cambios escalón en la presión de la corriente de entrada a la unidad de ósmosis inversa.

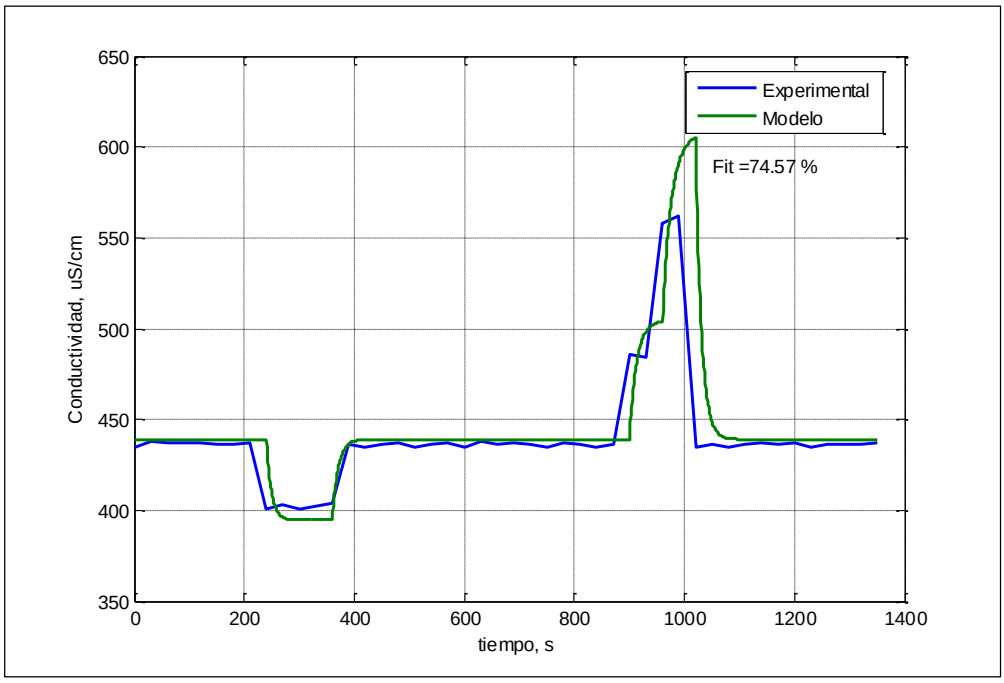

Figura 2.17. Resultados de la validación del modelo matemático. Conductividad del permeado frente a cambios escalón en la presión de la corriente de entrada a la unidad de ósmosis inversa.

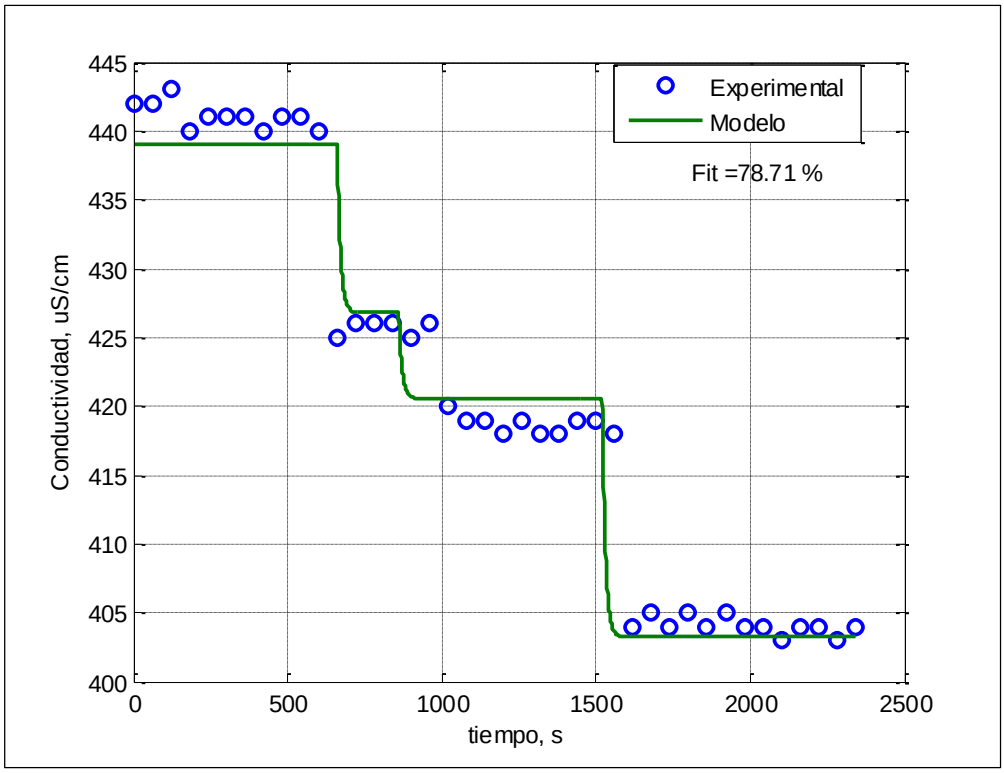

Figura 2.18. Resultados de la validación del modelo matemático. Conductividad del permeado frente a cambios escalón en el pH de la corriente de entrada a la unidad de ósmosis inversa.

2.5. Conclusiones parciales

- Se dedujo un modelo matemático no lineal multivariable para la unidad de ósmosis inversa de una planta desalinizadora de agua.
- Los resultados de validación frente a datos reales de operación de una planta piloto muestran un elevado grado de correspondencia. El modelo se ajusta con un 92.37% a los datos experimentales de la dependencia del flujo de permeado con la presión de la alimentación. Con un 74.57% a los datos experimentales de la conductividad del permeado en función de la presión de la alimentación. Y un 78.71% para la conductividad del permeado en función del pH de la alimentación.
- El modelo matemático obtenido puede ser utilizado de forma fiable en el diseño de sistemas efectivos de control para la planta objeto de estudio.

3. DISEÑO DE UN CONTROLADOR MULTIVARIABLE BASADO EN R.N.A. PARA LA UNIDAD DE O.I DE UNA PLANTA PILOTO DE DESALINIZACIÓN DE AGUA DE MAR

3.1. Introducción

Debido a la naturaleza cambiante de la demanda de agua desalinizada, todas las plantas industriales cuentan con sistemas automáticos de control para las variables principales del proceso, en particular para el flujo y la conductividad del permeado, que se controlan manipulando convenientemente la presión y el pH de la corriente de alimentación a los bastidores de O.I.

En la búsqueda de un mejor desempeño de las plantas de desalinización por O.I. se han propuesto diversas estrategias de control, que van desde el simple control on/off hasta controladores predictivos y robustos. En el estudio del estado del arte (capítulo 1) se mencionaron los diversos controladores que han sido diseñados e implementados en estas plantas, observándose que las aplicaciones basadas en R.N.A no han sido suficientemente investigadas.

En este capítulo se diseña un controlador multivariable basado en R.N.A. para controlar el flujo y la conductividad del permeado de una planta piloto de desalinización de agua de mar por O.I.

3.2. Teoría básica sobre redes neuronales artificiales

3.3. Redes neuronales artificiales

Las R.N.A. son estructuras para el procesamiento de información que imitan la arquitectura y el modo de funcionamiento del sistema nervioso biológico. Toda red neuronal artificial es un conjunto de procesadores elementales (neuronas) que están interconectados y operan en paralelo recibiendo y transmitiendo señales entre ellos o con el entorno con el fin de realizar cierta tarea de procesamiento (Cirstea, 2002).

3.3.1. Arquitectura de una R.N.A.

La organización y disposición de las neuronas en la red es lo que constituye la arquitectura de una red neuronal. Esta se caracteriza por cuatro parámetros importantes: el número de capas, la cantidad de neuronas por capa, la fuerza de la conexión entre las neuronas y el tipo de conexión (Flórez y Fernández, 2009).

Suele hacerse una primera clasificación en base a su agrupación en capas, hablándose de redes monocapa y redes multicapa. Asimismo, dependiendo de la dirección en la que la información se transmite, se habla de redes unidireccionales o prealimentadas (feedforward) y de redes recurrentes o realimentadas (feedback).

En las redes prealimentadas, la información se transmite solo en el dirección desde la capa de entrada hacia la de salida, mientras que, en las redes recurrentes, la información circula entre las capas en cualquier dirección. En estas redes la primera capa es llamada capa de entrada porque recibe información del exterior (por ejemplo, las señales de los sensores) y la última capa es llamada capa de salida porque envía información hacia afuera de la red (por ejemplo, hacia los actuadores) y las demás son conocidas como capas intermedias o capas ocultas porque no se contactan con el exterior.

En la R.N.A. prealimentada de la Figura 3.1, las señales x_i, i = {1,2,..,n}, son las entradas, los círculos son las neuronas, las líneas son la información que se transmite desde las neuronas de una capa a la siguiente, y las señales u_j, j = {1,2,3…,m}, son las salidas de la red. Observe que cada neurona de una capa intermedia se encuentra conectada a todas las neuronas de las capas adyacentes. Esta forma de interconexión es la que le da robustez a las R.N.A. (Rumehart y McClelland, 1986).

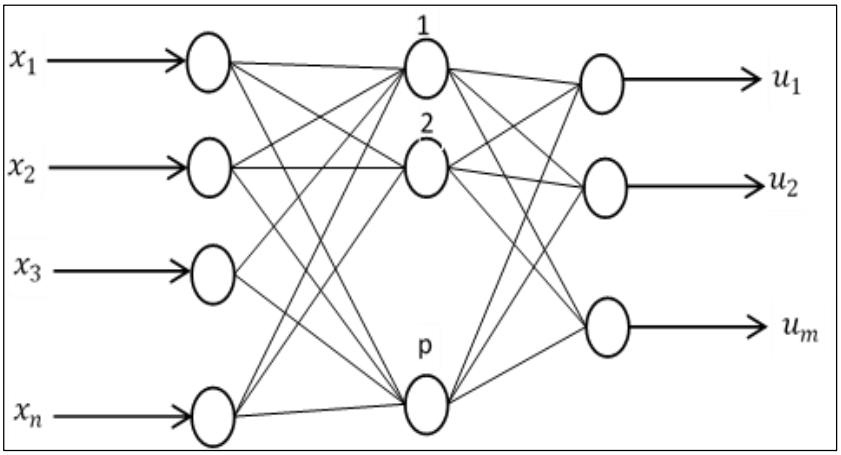

Figura 3.1. R.N.A. prealimentada con una capa oculta.

Matemáticamente, a cada neurona de una capa intermedia se asocian las señales que se detallan en la Figura 3.2.

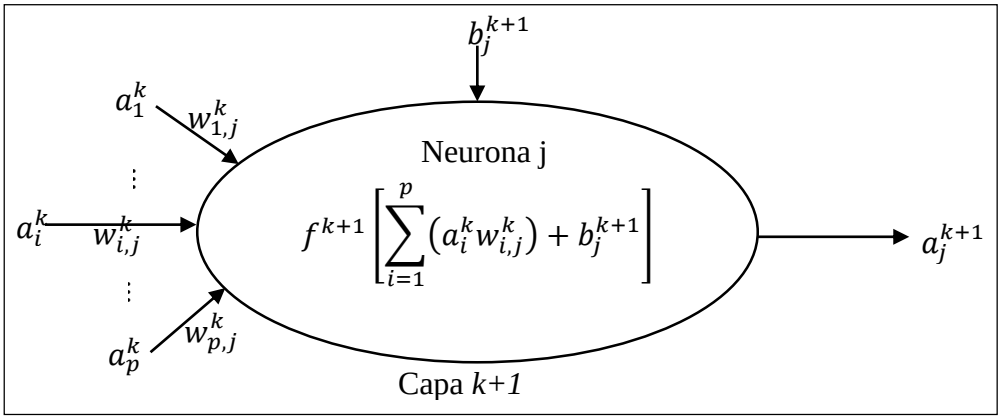

Figura 3.2. Señales asociadas a la neurona j en la capa $k+1$ de una R.N.A.

De acuerdo con la Figura 3.2, a la neurona ingresa la señal b_j^{k+1} (una constante conocida como bias) y las señales a_i^k multiplicadas por $w_{i,j}^k$, siendo a_i^k la salida de la neurona i de la capa k y $w_{i,j}^k$ el peso de la interconexión entre la neurona i de la capa k y la neurona j de la capa $k+1$. En resumen, la entrada neta a esta neurona es:

$$\varphi_j^{k+1} = \sum_{i=1}^{p}(a_i^k w_{i,j}^k) + b_j^{k+1} \tag{3.1}$$

Al evaluar la ecuación (3.1) en la función de transferencia f, se obtiene la salida (estado de activación) de la neurona j:

$$a_j^{k+1} = f^{k+1}(\varphi_j^{k+1}) \tag{3.2}$$

Las funciones de transferencia más empleadas en redes multicapa son: la sigmoidea y la tangente hiperbólica (Galushkin, 2005):

Función de activación sigmoidea (Figura 3.3):

$$f^{k+1}(\varphi_j^{k+1}) = \frac{1}{1+e^{-\varphi_j^{k+1}}} \tag{3.3}$$

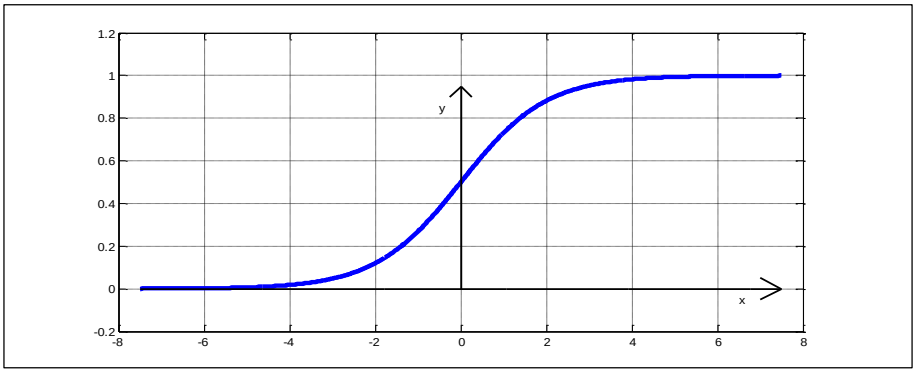

Figura 3.3. Función de activación sigmoidea.

Función de activación tangente hiperbólica (Figura 3.4):

$$f^{k+1}\left(\varphi_j^{k+1}\right) = \frac{2}{1+e^{-2\varphi_j^{k+1}}} - 1 \tag{3.4}$$

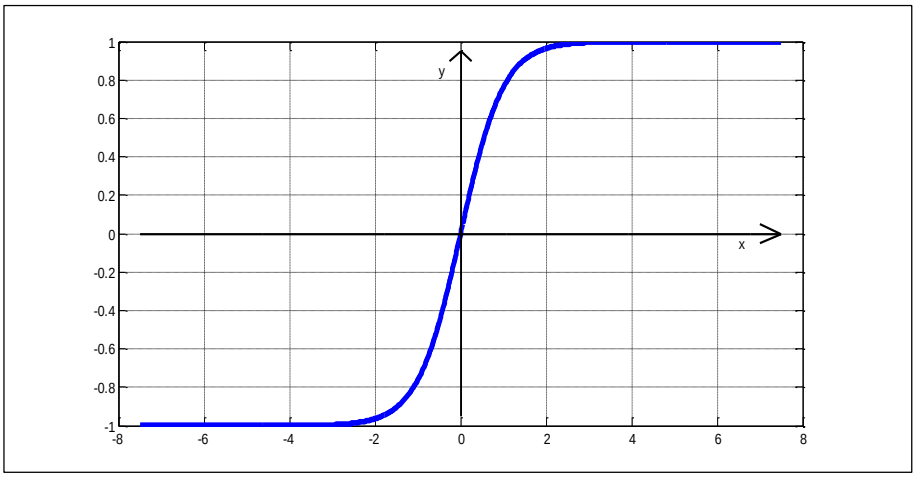

Figura 3.4. Función de activación tangente hiperbólica.

Aunque puede pensarse que una R.N.A. con muchas capas intermedias garantizará mejores resultados, Cybenko (1989) demostró (para el caso particular de funciones de activación sigmoideas) que una R.N.A. unidireccional con una capa intermedia y un número finito de neuronas en ella, permite aproximar cualquier función continua en un subconjunto compacto de R^n (teorema de la aproximación universal).

3.3.2. Tipos de aprendizaje de una R.N.A.

Una R.N.A. aprende para encontrar el valor exacto de los pesos de todas sus conexiones, y de este modo quedar capacitada para resolver de forma eficiente cualquier problema.

El aprendizaje de las R.N.A. se puede categorizar como aprendizaje con un profesor y aprendizaje sin un profesor (Haykin, 2009).

En el aprendizaje con un profesor (también llamado aprendizaje supervisado), el profesor indica a la red neuronal cual es la salida que debe conseguir (salida deseasa) para cada ejemplo con el que se realiza el entrenamiento (Figura 3.5), y mientras exista una diferencia (error) entre salida de la red y la salida deseada (con una tolerancia establecida), los pesos de las conexiones de las neuronas de la red se modificarán convenientemente.

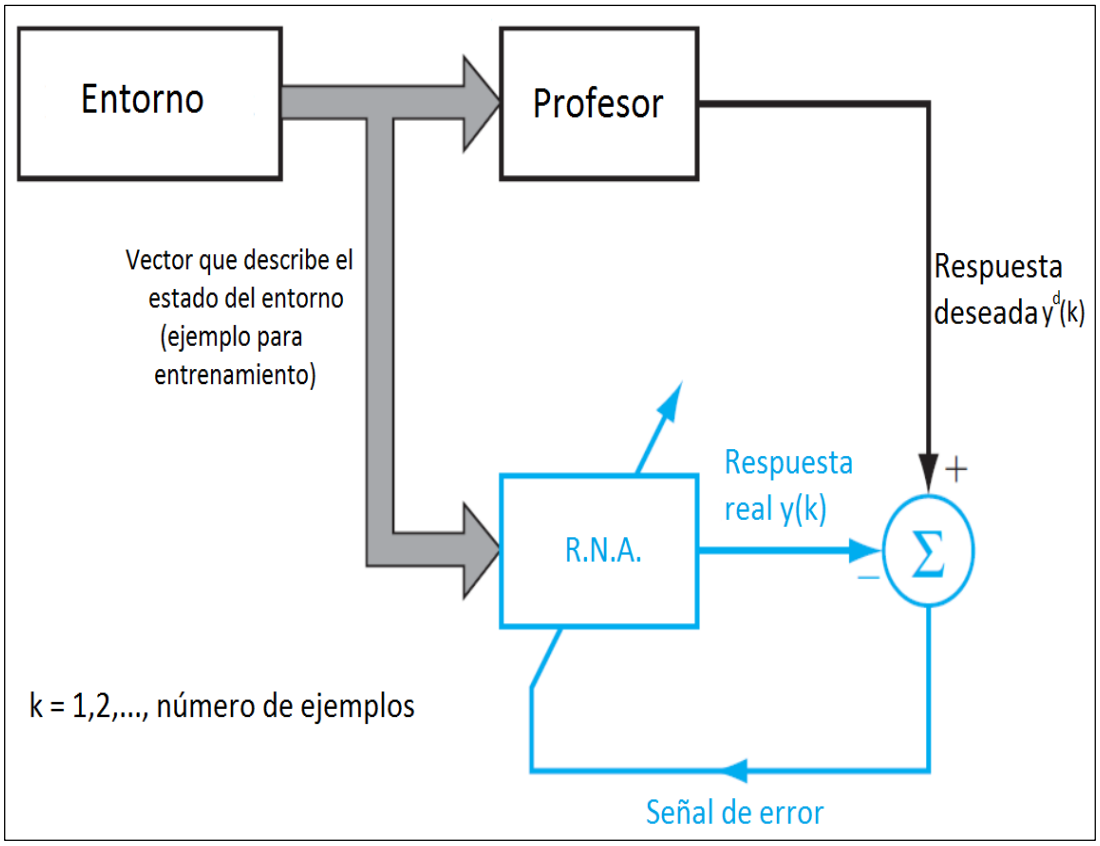

Figura 3.5. Esquema del aprendizaje supervisado (Haykin, 2009).

Usualmente la modificación de los pesos se realiza procurando minimizar la función de costo dada por la sumatoria del cuadrado de los errores:

$$J(k) = \sum_{j=1}^{p} \left[y_j^d(k) - y_j(k) \right]^2, \tag{3.5}$$

donde p es el número de salidas y y_j la j-ésima salida de la red para el ejemplo de entrenamiento k.

El aprendizaje sin un profesor se clasifica en dos tipos: aprendizaje por reforzamiento y aprendizaje no supervisado.

El aprendizaje por reforzamiento de una relación entrada-salida se lleva a cabo mediante un proceso de interacción continua con el entorno que busca minimizar un índice de costo escalar. Esta forma de aprendizaje se clasifica en tres tipos básicos: no asociativo, asociativo y secuencial (Omidvar y Elliott, 1997). La figura 3.6 muestra los componentes principales de un sistema de aprendizaje por reforzamiento asociativo, en el cual, un crítico convierte la señal de reforzamiento primaria recibida del entorno en una señal de reforzamiento heurística (señal de más alta calidad) mediante el análisis de varios ejemplos (Haykin, 2009).

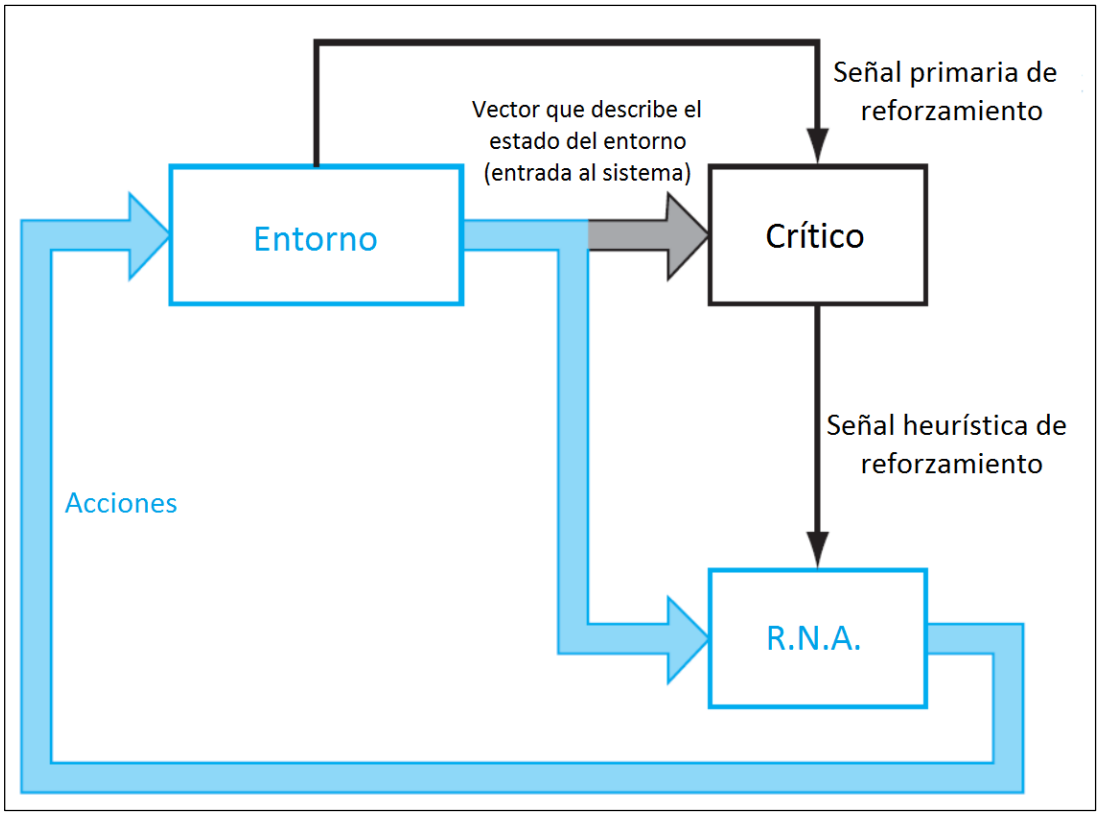

Figura 3.6. Esquema del aprendizaje por reforzamiento (Haykin, 2009).

En el aprendizaje no supervisado o auto organizativo (Figura 3.7), no está presente ni un crítico ni un profesor para dirigir el proceso de aprendizaje, sino más bien, los parámetros de la red son optimizados en base a una medida independiente de la calidad de la representación que se requiere que la red aprenda (Becker, 1991). Para que la red pueda aprender se debe utilizar una regla de aprendizaje competitivo que funcione en lazo abierto, donde el ajuste de los parámetros sea función exclusiva de los ejemplos de entrada (Haykin, 2009).

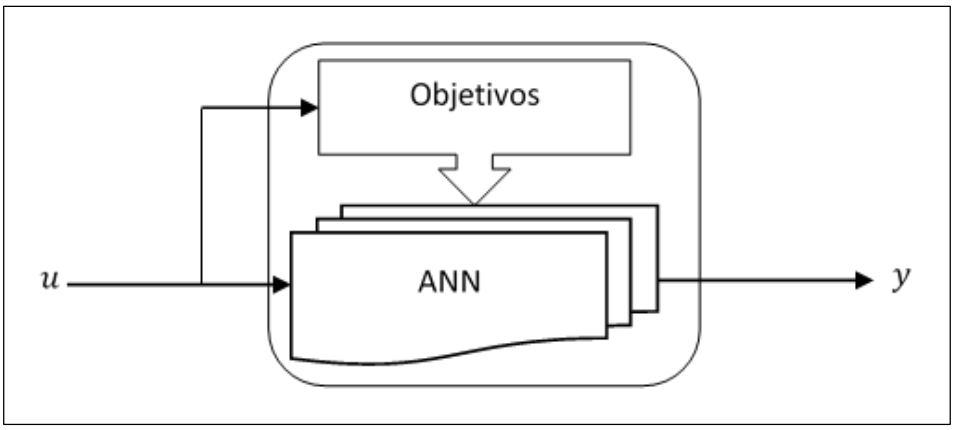

Figura 3.7. Esquema del aprendizaje no supervisado (Sarmiento, 2006).

3.3.3. Algoritmos de entrenamiento de una R.N.A.

Son el conjunto de instrucciones con las que se modifican los pesos de las conexiones de una R.N.A. hasta que alcancen sus valores apropiados. A continuación, se presentan algunas de las reglas más conocidas:

- Regla de Hebb: Históricamente es la primera regla de aprendizaje que fue formulada. Establece que el peso de la conexión entre dos neuronas aumenta si ambas neuronas son activadas simultáneamente y disminuye si lo hacen por separado (Rabuñal y Dorado, 2006).

- Regla Delta: También llamada método del gradiente descendente. Es una modificación de la regla de Hebb que establece que: para reducir la diferencia entre la salida de la R.N.A. y la salida deseada, el peso de las conexiones entre neuronas cambia según la dirección de máxima pendiente de la función error. En esta regla es común inicializar los pesos de forma aleatoria.

- Regla Backpropagation: También conocida como regla delta generalizada. Es uno de los métodos más utilizados para el entrenamiento de R.N.A. prealimentadas. El proceso de entrenamiento tiene dos fases: (1) se estimula a las neuronas de la capa de entrada a la red con el primer ejemplo, los efectos de este estímulo se continúan transmitiéndose a través de las capas siguientes hasta producir ciertas salidas que se comparan con las salidas deseadas para calcular el error en cada neurona de la capa de salida. (2) Las señales de error se transmiten hacia atrás a todas las neuronas que contribuyen directamente a la salida. Esta propagación hacia atrás continúa capa tras capa hasta que todas las neuronas de la red reciban una señal que describa su contribución relativa al error total. Finalmente, en base a las señales recibidas, se actualizan los pesos de todas las conexiones. Se repite este procedimiento con los siguientes ejemplos hasta que el error sea cero.

- Random Activation Weight Neural Net: Este algoritmo se aplica a redes unidireccionales con una sola capa oculta. Sugiere utilizar valores aleatorios para los pesos de las conexiones entre la capa de entrada y la capa oculta, reduciendo la tarea de diseño a calcular solo los pesos entre la capa oculta y la de salida (Te Braake y Van Straten, 1995). La R.N.A. se entrena en una única iteración, y por ello resulta conveniente para aplicaciones en tiempo real.

3.4. Bases teóricas para el diseño de controladores basados en R.N.A.

3.4.1. Neurocontroladores

El uso de neurocontroladores es una alternativa que ofrece el control inteligente para cumplir con las tareas de estabilización, regulación o tracking en un sistema dinámico. En general, los neurocontroladores se clasifican como estáticos y dinámicos.

- Neurocontroladores estáticos

Se llaman estáticos porque el entrenamiento de la red neuronal que constituye el controlador es estático. Esto significa que se utiliza un conjunto de datos de entrada y salida previamente obtenidos del sistema dinámico (en lazo cerrado) para entrenar la red neuronal (Figura 3.8).

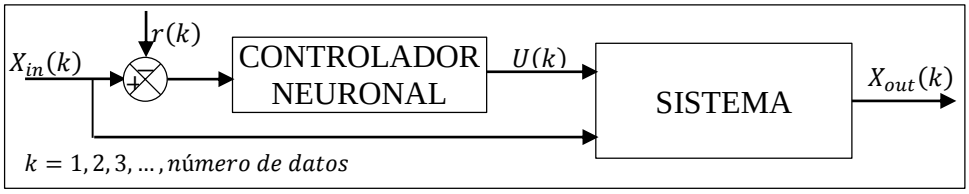

Figura 3.8. Esquema para el entrenamiento de un neurocontrolador estático.

En el entrenamiento con el algoritmo de retropropagación de errores (Rumelhart et al., 1986), se parte del estado inicial del sistema $X_{in}(1)$ y se calcula el error respecto al valor deseado $r(1)$. Este error ingresa a la red neuronal (que inicialmente tiene pesos arbitrarios) para generar una señal de control $U(1)$ que se envía a la planta junto con $X_{in}(1)$. Estas dos señales producen el estado $X_{out}(1)$, que es comparado con el valor de la referencia para ese instante de tiempo y el error obtenido es retropropagado hacia todas las neuronas de la red para hacer una primera actualización de los pesos de todas las conexiones (entrenamiento patrón). Luego ingresa $X_{in}(2)$ - $r(2)$ a la red neuronal y se repite en proceso. Se continúa de esta manera hasta utilizar todos los ejemplos disponibles para el entrenamiento. Se puede repetir varias veces todo el proceso, empezando cada vez con los últimos valores obtenidos para los pesos, hasta conseguir la convergencia deseada, la cual normalmente se mide en términos de una tolerancia para el error de seguimiento.

- Neurocontroladores dinámicos

Estos neurocontroladores se entrenan realimentando la salida de la planta, y por eso se dice que la red neuronal que hace las veces de controlador es dinámica (Figura 3.9).

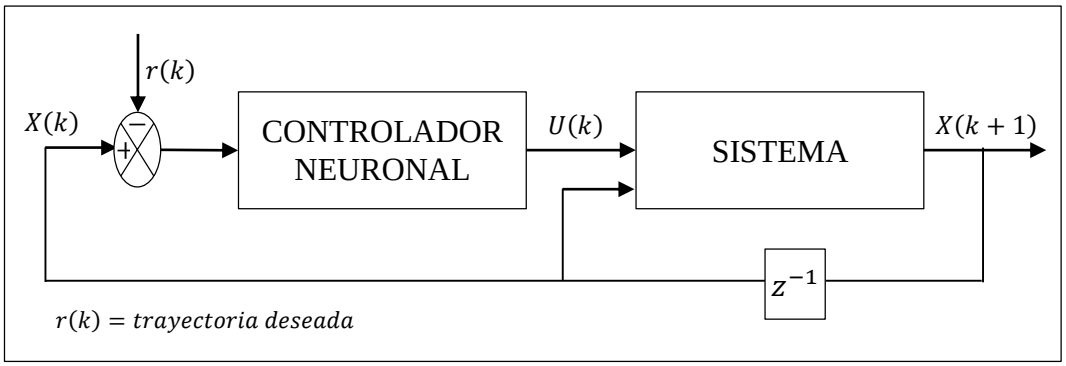

Figura 3.9. Esquema de entrenamiento de un neurocontrolador dinámico.

Los algoritmos de entrenamiento más utilizados son el Back Propagation Through Time (BPTT) y el Dynamic Back Propagation (DBP).

En ambos casos se parte de la condición inicial para los estados del sistema $X(1)$ y se calcula su diferencia (error) respecto a los valores deseados $r(1)$. Esta señal de error ingresa a la red neuronal (que inicialmente tiene pesos arbitrarios) para producir una señal de control $U(1)$ que se envía a la planta junto con $X(1)$. Estas dos señales generan el estado del sistema en el siguiente instante de tiempo $X(2)$, que se realimenta a la red neuronal para producir $U(2)$, el cual junto con $X(2)$ generarán a $X(3)$ y así sucesivamente hasta calcular una cantidad de estados previamente fijada. Durante cada ciclo (realimentación) se van haciendo los cálculos de las variables necesarias para realizar una primera actualización de pesos. Obtenidos los nuevos valores de los pesos se repite todo el proceso hasta conseguir la salida deseada.

Es importante mencionar que el algoritmo DBP es una forma recursiva del algoritmo BPTT, que además ofrece la ventaja de permitir el uso ya sea de un modelo neuronal o de un modelo en el espacio de estados para la planta, mientras que, el algoritmo BPTT solo permite utilizar un modelo neuronal.

En el entrenamiento del controlador neuronal multivariable para la planta objeto de estudio del presente trabajo de tesis, se utiliza el algoritmo recursivo DBP en virtud de sus ventajas computacionales sobre el algorimo BPTT.

3.4.2. Ecuaciones para el diseño de controladores basados en R.N.A.

En esta sección se deducen las ecuaciones para el diseño y entrenamiento de un neurocontrolador dinámico multivariable con n entradas, m salidas y nm neuronas en la capa oculta (Figura 3.10).

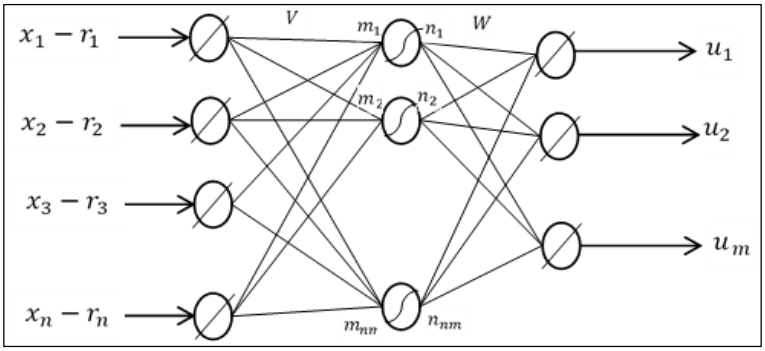

Figura 3.10. Estructura del controlador neuronal multivariable a diseñar.

Teniendo en cuenta que v_{ij} es el peso de la conexión entre la neurona i de la capa de entrada y la neurona j de la capa intermedia, se calcula la entrada a la neurona j de la capa intermedia utilizando la regla de propagación:

$$m_j = \sum_{i=1}^{n} v_{ij}(x_{\mathrm{i}} - r_{\mathrm{i}}) \tag{3.6}$$

Que es equivalente a:

$$m_j = v_{1j}(x_1 - r_1) + v_{2j}(x_2 - r_2) + \cdots + v_{nj}(x_n - n_n) \tag{3.7}$$

La ecuación (3.7) se puede utilizar para calcular la entrada a cada neurona de la capa intermedia y escribir el resultado en forma vectorial:

$$\begin{bmatrix} m_1 \\ m_2 \\ \cdots \\ m_n \end{bmatrix} = \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{n1} \\ v_{12} & v_{22} & & v_{n2} \\ \vdots & & \ddots & \vdots \\ v_{1nm} & v_{2nm} & \cdots & v_{nnm} \end{bmatrix} * \begin{bmatrix} x_1 - r_1 \\ x_2 - r_2 \\ \cdots \\ x_n - r_n \end{bmatrix} \tag{3.8}$$

Si se define el vector de entradas a la capa intermedia $m = [m_1 \ m_2 \ \ldots \ m_{nm}]'$, el vector de estados del sistema $x = [x_1 \ x_2 \ \ldots \ x_n]'$, el vector de valores deseados para cada estado $r = [r_1 \ r_2 \ \ldots \ r_n]'$, y la matriz de pesos para las conexiones entre las neuronas de la capa de entrada y de la capa intermedia:

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1nm} \\ v_{21} & v_{22} & & v_{2nm} \\ \vdots & & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nnm} \end{bmatrix} \tag{3.9}$$

La ecuación (3.8) se podrá escribir como:

$$m = V' * (X - r) \tag{3.10}$$

Considerando que el vector de salidas $n = [n_1 \ n_2 \ \dots \ n_{nm}]'$ y el vector entradas m a la capa intermedia, se relacionan según la función de activación:

$$n = f(m) \tag{3.11}$$

Y que w_{jk} es el peso de conexión entre la neurona j de la capa intermedia y la neurona k de la capa de salida; se puede demostrar, con un procedimiento similar al realizado para obtener la ecuación (3.10), que:

$$U = W' * n, \tag{3.12}$$

donde W, la matriz de pesos para las conexiones entre las neuronas de la capa intermedia y de la capa de salida de la red neuronal, se define como:

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & & w_{2m} \\ \vdots & & \ddots & \vdots \\ w_{nm1} & v_{nm2} & \cdots & v_{nmm} \end{bmatrix} \tag{3.13}$$

Para el objetivo de control: minimizar la diferencia (error) entre la salida de un sistema $X(k+1)$ y la trayectoria de referencia $r(k+1)$, se plantea la función de costo (N es el número de ejemplos utilizados en el entrenamiento):

$$J = \frac{1}{2}\sum_{k=0}^{N-1}[X(k+1) - r(k+1)]'[X(k+1) - r(k+1)] \tag{3.14}$$

y se utiliza el método del gradiente para actualizar el valor de los pesos de todas las conexiones de la red neuronal:

$$W = W - \eta \frac{\partial J}{\partial W} \tag{3.15}$$

$$V = V - \eta \frac{\partial J}{\partial V} \tag{3.16}$$

Para un peso en particular, la regla de la cadena permite obtener:

$$\frac{\partial J}{\partial w_{ij}} = \sum_{k=0}^{N-1}[X(k+1) - r(k+1)]' \frac{\partial X(k+1)}{\partial w_{ij}} \tag{3.17}$$

A continuación se deduce una fórmula recursiva para la derivada $\frac{\partial X(k+1)}{\partial w_{ij}}$.

Para el caso particular de k=0, se tiene:

$$\frac{\partial X(1)}{\partial w_{ij}} = \frac{\partial X(1)}{\partial w_{ij}} \tag{3.18}$$

Como $X(2)$ es función de $X(1)$ y $U(1)$ (Figura 3.9), y a la vez $X(1)$ y $U(1)$ son funciones de w_{ij} (Figura 3.11), la regla de la cadena para k= 1 da:

$$\frac{\partial X(2)}{\partial w_{ij}} = \frac{\partial X(2)}{\partial U(1)}\frac{\partial U(1)}{\partial w_{ij}} + \frac{\partial X(2)}{\partial U(1)}\frac{\partial U(1)}{\partial X(1)}\frac{\partial X(1)}{\partial w_{ij}} + \frac{\partial X(2)}{\partial X(1)}\frac{\partial X(1)}{\partial w_{ij}} \tag{3.19}$$

La ecuación (3.19) puede reescribir como:

$$\frac{\partial X(2)}{\partial w_{ij}} = \frac{\partial X(2)}{\partial U(1)}\frac{\partial U(1)}{\partial w_{ij}} + \left[\frac{\partial X(2)}{\partial U(1)}\frac{\partial U(1)}{\partial X(1)} + \frac{\partial X(2)}{\partial X(1)}\right]\frac{\partial X(1)}{\partial w_{ij}} \tag{3.20}$$

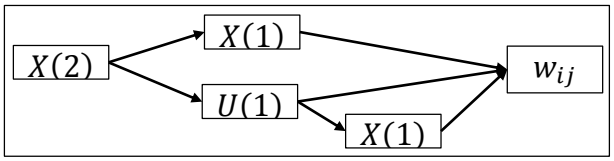

Figura 3.11. Diagrama de árbol para calcular $\frac{\partial X(2)}{\partial w_{ij}}$ con la regla de la cadena

Si se continúa calculando las derivadas y empleando la regla de la cadena, se obtiene la fórmula recursiva conocida como algoritmo de entrenamiento DBP:

$$\frac{\partial X(k+1)}{\partial w_{ij}} = \frac{\partial X(k+1)}{\partial U(k)}\frac{\partial U(k)}{\partial w_{ij}} + \left[\frac{\partial X(k+1)}{\partial U(k)}\frac{\partial U(k)}{\partial X(k)} + \frac{\partial X(k+1)}{\partial X(k)}\right]\frac{\partial X(k)}{\partial w_{ij}} \tag{3.21}$$

Para el caso particular de un sistema dinámico lineal e invariante en el tiempo representado en el espacio de estados por:

$$X_{k+1} = AX_k + BU_k \tag{3.22}$$

Las derivadas $\frac{\partial X(k+1)}{\partial X(k)}$ y $\frac{\partial X(k+1)}{\partial U(k)}$ son inmediatas:

$$\frac{\partial X(k+1)}{\partial X(k)} = A \tag{3.23}$$

$$\frac{\partial X(k+1)}{\partial U(k)} = B \tag{3.24}$$

Y permiten reducir la ecuación (3.21) a:

$$\frac{\partial X(k+1)}{\partial w_{ij}} = B\,\frac{\partial U(k)}{\partial w_{ij}} + \left[B\,\frac{\partial U(k)}{\partial X(k)} + A\right]\frac{\partial X(k)}{\partial w_{ij}} \tag{3.25}$$

Observe que, definiendo las derivadas (matrices jacobianas):

$$\frac{\partial X(k)}{\partial W(:,j)} = \begin{bmatrix} \frac{\partial x_1}{\partial w_{1j}} & \frac{\partial x_1}{\partial w_{2j}} & \cdots & \frac{\partial x_1}{\partial w_{nmj}} \\ \frac{\partial x_2}{\partial w_{1j}} & \frac{\partial x_2}{\partial w_{2j}} & & \frac{\partial x_2}{\partial w_{nmj}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial x_n}{\partial w_{1j}} & \frac{\partial x_n}{\partial w_{2j}} & \cdots & \frac{\partial x_n}{\partial w_{nmj}} \end{bmatrix} \tag{3.26}$$

$$\frac{\partial U(k)}{\partial W(:,j)} = \begin{bmatrix} \frac{\partial u_1}{\partial w_{1j}} & \frac{\partial u_1}{\partial w_{2j}} & \cdots & \frac{\partial u_1}{\partial w_{nmj}} \\ \frac{\partial u_2}{\partial w_{1j}} & \frac{\partial u_2}{\partial w_{2j}} & & \frac{\partial u_2}{\partial w_{nmj}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial u_m}{\partial w_{1j}} & \frac{\partial u_m}{\partial w_{2j}} & \cdots & \frac{\partial u_m}{\partial w_{nmj}} \end{bmatrix} \tag{3.27}$$

La ecuación (3.25) puede generalizarse a:

$$\frac{\partial X(k+1)}{\partial W(:,j)} = B\,\frac{\partial U(k)}{\partial W(:,j)} + \left[B\,\frac{\partial U(k)}{\partial X(k)} + A\right]\frac{\partial X(k)}{\partial W(:,j)} \tag{3.28}$$

Y si se define:

$$\frac{\partial X(k)}{\partial W} = \left[\frac{\partial X(k)}{\partial W(:,1)} \quad \frac{\partial X(k)}{\partial W(:,2)} \quad \cdots \quad \frac{\partial X(k)}{\partial W(:,m)}\right] \tag{3.29}$$

$$\frac{\partial U(k)}{\partial W} = \left[\frac{\partial U(k)}{\partial W(:,1)} \quad \frac{\partial U(k)}{\partial W(:,2)} \quad \cdots \quad \frac{\partial U(k)}{\partial W(:,m)}\right] \tag{3.30}$$

Se obtiene una ecuación más general que la ecuación (3.28).

$$\frac{\partial X(k+1)}{\partial W} = B\,\frac{\partial U(k)}{\partial W} + \left[B\,\frac{\partial U(k)}{\partial X(k)} + A\right]\frac{\partial X(k)}{\partial W} \tag{3.31}$$

Observe que las ecuaciones (3.26), (3.27), (3.29) y (3.30) están bien definidas para la multiplicación matricial en las ecuaciones (3.28) y (3.31).

Como una salida $u_k$cualquiera de la red neuronal viene dada por:

$$u_k = \sum_{j=1}^{nm} w_{jk} n_j \tag{3.32}$$

La derivada $\frac{\partial u_k}{\partial w_{jk}}$ será:

$$\frac{\partial u_k}{\partial w_{jk}} = n_j \tag{3.33}$$

Entonces, utilizando la definición (3.27), se obtiene:

$$\frac{\partial U(k)}{\partial W(:,1)} = \begin{bmatrix} n_1 & n_2 & \cdots & n_{nm} \\ 0 & 0 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{m\times nm} = \begin{bmatrix} n' \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{m\times nm} \tag{3.34}$$

Y con la definición (3.30), se consigue:

$$\frac{\partial U(k)}{\partial W} = \begin{bmatrix} n' & 0 & \cdots & 0 \\ 0 & n' & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & n' \end{bmatrix}_{m\times(nm*m)} \tag{3.35}$$

Por otro lado, al derivar la ecuación (3.12) y usar la regla de la cadena:

$$\frac{\partial U(k)}{\partial X(k)} = \frac{\partial U(k)}{\partial n}\frac{\partial n}{\partial m}\frac{\partial m}{\partial X(k)} \tag{3.36}$$

Utilizando las ecuaciones (3.10) y (3.12), la ecuación (3.36) se reduce a:

$$\frac{\partial U(k)}{\partial X(k)} = W' * \frac{\partial n}{\partial m} * V' \tag{3.37}$$

De manera similar que para la matriz de pesos W, se puede demostrar que:

$$\frac{\partial X(k+1)}{\partial V} = B\frac{\partial U(k)}{\partial V} + \left[B\frac{\partial U(k)}{\partial X(k)} + A\right]\frac{\partial X(k)}{\partial V} \tag{3.38}$$

Con $\frac{\partial U(k)}{\partial V}$ y $\frac{\partial U(k)}{\partial V(:,j)}$ definidos por:

$$\frac{\partial U(k)}{\partial V} = \begin{bmatrix} \frac{\partial U(k)}{\partial V(:,1)} & \frac{\partial U(k)}{\partial V(:,2)} & \cdots & \frac{\partial U(k)}{\partial V(:,nm)} \end{bmatrix} \tag{3.39}$$

$$\frac{\partial U(k)}{\partial V(:,j)} = \begin{bmatrix} \frac{\partial u_1}{\partial v_{1j}} & \frac{\partial u_1}{\partial v_{2j}} & \cdots & \frac{\partial u_1}{\partial v_{nj}} \\ \frac{\partial u_2}{\partial v_{1j}} & \frac{\partial u_2}{\partial v_{2j}} & & \frac{\partial u_2}{\partial v_{nj}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial u_m}{\partial v_{1j}} & \frac{\partial u_m}{\partial v_{2j}} & \cdots & \frac{\partial u_m}{\partial v_{nj}} \end{bmatrix} \tag{3.40}$$

A partir de las ecuaciones (3.10), (3.11) y (3.12) se obtiene:

$$U = W' * f\big(V' * (X - r)\big) \tag{3.41}$$

Lo cual significa que una salida $u_k$cualquiera de la red neuronal es:

$$u_k = \sum_{j=1}^{nm} w_{jk} f\big(\sum_{i=1}^{n} v_{ij}(x_i - r_i)\big) n_j \tag{3.42}$$

Utilizando la regla de la cadena se calcula la derivada:

$$\frac{\partial u_k}{\partial v_{ij}} = w_{jk} \frac{\partial n_j}{\partial m_j}(x_i - r_i) \tag{3.43}$$

En consecuencia, utilizando la definición (3.39), se tiene:

$$\frac{\partial U(k)}{\partial V(:,j)} = \begin{bmatrix} w_{j1}\frac{\partial n_j}{\partial m_j}(x_1 - r_1) & w_{j1}\frac{\partial n_j}{\partial m_j}(x_2 - r_2) & \cdots & w_{j1}\frac{\partial n_j}{\partial m_j}(x_n - r_n) \\ w_{j2}\frac{\partial n_j}{\partial m_j}(x_1 - r_1) & w_{j2}\frac{\partial n_j}{\partial m_j}(x_2 - r_2) & & w_{j2}\frac{\partial n_j}{\partial m_j}(x_n - r_n) \\ \vdots & & \ddots & \vdots \\ w_{jm}\frac{\partial n_j}{\partial m_j}(x_1 - r_1) & w_{jm}\frac{\partial n_j}{\partial m_j}(x_2 - r_2) & \cdots & w_{jm}\frac{\partial n_j}{\partial m_j}(x_n - r_n) \end{bmatrix} \tag{3.44}$$

Con una notación abreviada, la ecuación (3.44) se escribe:

$$\frac{\partial U(k)}{\partial V(:,j)} = \begin{bmatrix} w_{j1}[x - r]' \\ w_{j2}[x - r]' \\ \vdots \\ w_{jm}[x - r]' \end{bmatrix} \frac{\partial n_j}{\partial m_j}_{m \times n} \tag{3.45}$$

Y con ello, la ecuación (3.39) será:

$$\frac{\partial U(k)}{\partial V} = \left[\begin{bmatrix} w_{11}[x - r]' \\ w_{12}[x - r]' \\ \vdots \\ w_{1m}[x - r]' \end{bmatrix} \frac{\partial n_1}{\partial m_1} \quad \begin{bmatrix} w_{21}[x - r]' \\ w_{22}[x - r]' \\ \vdots \\ w_{2m}[x - r]' \end{bmatrix} \frac{\partial n_2}{\partial m_2} \quad \cdots \quad \begin{bmatrix} w_{nm1}[x - r]' \\ w_{nm2}[x - r]' \\ \vdots \\ w_{nmm}[x - r]' \end{bmatrix} \frac{\partial n_{nm}}{\partial m_{nm}} \right]_{m \times (n * nm)} \tag{3.46}$$

Luego, si se escoge la función de activación tangente hiperbólica:

$$n = \frac{2}{1 + e^{-\left(\frac{m - c}{a}\right)}} - 1, \tag{3.47}$$

donde vector de centros $c = [c_1 \;\; c_2 \;\; \ldots \;\; c_{nm}]'$ y el vector de inclinaciones $a = [a_1 \;\; a_2 \;\; \ldots \;\; a_{nm}]'$, también se optimizan utilizando el método del gradiente:

$$c = c - \eta \frac{\partial J}{\partial c} \tag{3.48}$$

$$a = a - \eta \frac{\partial J}{\partial a}, \tag{3.49}$$

y las derivadas $\frac{\partial J}{\partial c}$ y $\frac{\partial J}{\partial a}$ se calculan a partir de la función de costo, ecuación (3.14):

$$\frac{\partial J}{\partial c} = \sum_{k=0}^{N-1} [X(k+1) - r(k+1)]' \frac{\partial X(k+1)}{\partial c} \tag{3.50}$$

$$\frac{\partial J}{\partial a} = \sum_{k=0}^{N-1} [X(k+1) - r(k+1)]' \frac{\partial X(k+1)}{\partial a} \tag{3.51}$$

También se puede demostrar que:

$$\frac{\partial X(k+1)}{\partial c} = B \frac{\partial U(k)}{\partial c} + \left[B \frac{\partial U(k)}{\partial X(k)} + A\right] \frac{\partial X(k)}{\partial c} \tag{3.52}$$

$$\frac{\partial X(k+1)}{\partial a} = B \frac{\partial U(k)}{\partial a} + \left[B \frac{\partial U(k)}{\partial X(k)} + A\right] \frac{\partial X(k)}{\partial a} \tag{3.53}$$

De la ecuación (3.12):

$$\frac{\partial U(k)}{\partial c} = W' \frac{\partial n}{\partial c} \tag{3.54}$$

$$\frac{\partial U(k)}{\partial a} = W' \frac{\partial n}{\partial a} \tag{3.55}$$

El elemento de la posición (j, l) de la matriz $\frac{\partial n}{\partial c}$ se calcula de la ecuación (3.47):

$$\frac{\partial n_j}{\partial c_l} = \frac{n_j^2 - 1}{2a} \tag{3.56}$$

De manera similar, el elemento de la posición (j, l) de la matriz $\frac{\partial n}{\partial a}$ es:

$$\frac{\partial n_j}{\partial a_l} = \frac{m-c}{2a_l^2} \left(n_j^2 - 1\right) \tag{3.57}$$

Y el elemento de la posición (j, l) de la matriz $\frac{\partial n}{\partial m}$ es:

$$\frac{\partial n_j}{\partial m_l} = -\frac{n_j^2 - 1}{2a} \tag{3.58}$$

3.4.3. Escalamiento de las señales de entrada y salida del neurocontrolador

Debido a que las salidas de las funciones tangente hiperbólica que se usan como funciones de activación de las neuronas de la capa intermedia del controlador están limitadas al rango [-1, 1], y que las señales de control no necesariamente son de este orden de magnitud, resulta conveniente utilizar factores de escalamiento apropiados entre la salida del controlador y la entrada a la planta. Observe que no es estrictamente necesario realizar el escalamiento, ya que los pesos de las conexiones ente las neuronas de la capa intermedia y de la capa de salida pueden tomar los valores adecuados para manejar cualquier orden de magnitud en la salida, sin embargo; cuando las salidas son de diferentes órdenes de magnitud, algunos pesos necesitan tomar valores pequeños y otros mucho más grandes, y como se parte de valores aleatorios y se utiliza el mismo ratio de aprendizaje, no se alcanzará simultáneamente la convergencia de todos los pesos de la red.

Sucede algo parecido para las entradas al neurocontrolador, pues la función tangente hiperbólica es más sensible a valores de entradas cercanos al origen (por ejemplo, entre -2 y 2). Esto sugiere que también se deben escalar las señales de entrada a la red neuronal para facilitar el entrenamiento.

Si $me_1, me_2, \dots, me_n$ son los factores de escalamiento de los errores $e_1, e_2, \dots, e_n$ de los estados del sistema (Figura 3.12), se pueden escribir las ecuaciones:

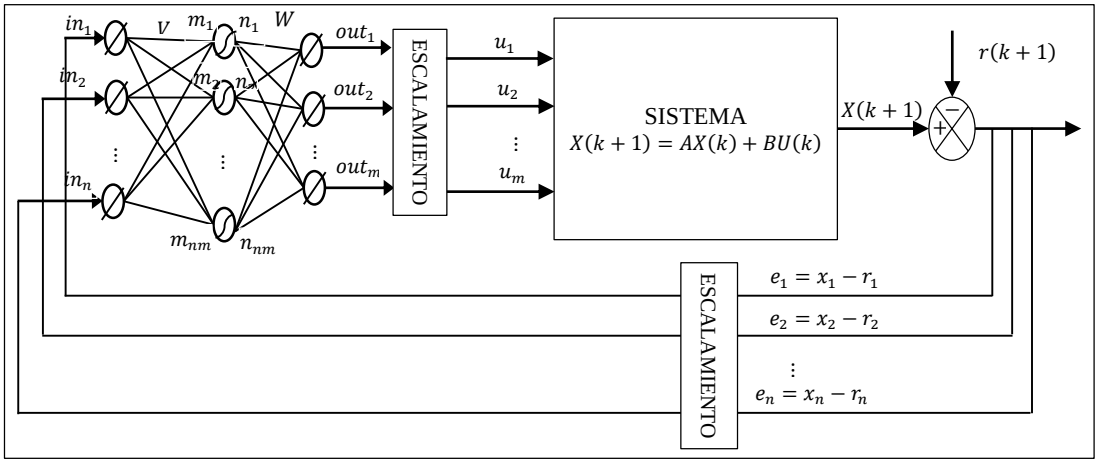

Figura 3.12. Esquema para el escalamiento de las señales de entrada y salida del controlador neuonal.

$$\begin{aligned} in_1 &= me_1 * e_1 \\ in_2 &= me_2 * e_2 \\ &\vdots \\ in_n &= me_n * e_n \end{aligned} \tag{3.59}$$

Y al definir el vector de entradas $in = [in_1, in_2, \dots, in_n]'$, el vector de errores $e = [e_1, e_2, \dots, e_n]'$ y la matriz de transformación:

$$ME = \begin{bmatrix} me_1 & 0 & \cdots & 0 \\ 0 & me_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & me_n \end{bmatrix} \tag{3.60}$$

La ecuación (3.59) se podrá escribir como:

$$in = ME * e \tag{3.61}$$

De manera similar, si $ms_1, ms_2, \dots, ms_m$ son los factores de escalamiento de las salidas $out_1, out_2, \dots, out_m$ del controlador neuronal, es decir:

$$\begin{aligned} u_1 &= ms_1 * out_1 \\ u_2 &= ms_2 * out_2 \\ &\vdots \\ u_m &= ms_m * out_m \end{aligned} \tag{3.62}$$

Y si se define el vector de salidas $out = [out_1, out_2, \dots, out_m]'$, el vector de entradas a la planta $u = [u_1, u_2, \dots, u_m]'$ y la matriz de transformación:

$$MS = \begin{bmatrix} ms_1 & 0 & \cdots & 0 \\ 0 & ms_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & ms_m \end{bmatrix} \tag{3.63}$$

La ecuación (3.62) se podrá escribir como:

$$u = MS * out \tag{3.64}$$

Puesto que en la deducción de las ecuaciones para el diseño y entrenamiento del neurocontrolador (sección 3.4.2) se ha supuesto que las salidas y las entradas de la planta son las que se usan directamente como entradas y salidas respectivas de la red neuronal, habrá que hacer una transformación a la representación en espacio de estados del sistema (planta).

Para el sistema dinámico representado por:

$$X(k+1) = A * X(k) + B * U(k) \tag{3.65}$$

$$Y(k) = C * X(k) + D * U(k) \tag{3.66}$$

Se define la transformación lineal:

$$Z = ME * X \tag{3.67}$$

Que se reemplaza en la ecuación (3.65) para obtener:

$$ME^{-1} * Z(k+1) = A * ME^{-1} * Z(k) + B * MS * out(k) \tag{3.68}$$

O equivalentemente:

$$Z(k+1) = ME * A * ME^{-1} * Z(k) + ME * B * MS * out(k) \tag{3.69}$$

De manera similar, la ecuación de salida de la representación en espacio de estados, ecuación (3.66), se escribe:

$$Y(k) = C * ME^{-1} * Z(k) + D * MS * out(k) \tag{3.70}$$

Por conveniencia se definen las matrices:

$$AA = ME * A * ME^{-1} \tag{3.71}$$

$$BB = ME * B * MS \tag{3.72}$$

$$CC = C * ME^{-1} \tag{3.73}$$

$$DD = D * MS \tag{3.74}$$

Para escribir las ecuaciones:

$$Z(k+1) = AA * Z(k) + BB * out(k) \tag{3.75}$$

$$Y(k) = CC * Z(k) + DD * out(k) \tag{3.76}$$

Que son la nueva representación en el espacio de estados del sistema.

3.5. Razones que justifican el diseño de controladores basados en R.N.A. para la planta objeto de estudio

En base a los fundamentos teóricos presentados en las secciones 3.2 y 3.3, las razones que justifican el uso de un controlador basado en R.N.A. para la planta objeto de estudio son las siguientes:

- La superioridad de las R.N.A. frente a otros esquemas (polinomios, etc.) para modelar sistemas no lineales.
- La capacidad de aprendizaje y auto organización de una R.N.A. que alivia al usuario de la tarea de obtener un modelo del sistema.
- La facilidad con la que las R.N.A. pueden aplicarse en el modelado y control de plantas MIMO.
- La capacidad de las R.N.A. para responder bien ante señales de entrada no presentadas durante el entrenamiento.
- La flexibilidad de las R.N.A para tolerar cambios no relevantes en la entrada, como por ejemplo señales con ruido.
- La capacidad de las R.N.A. para tolerar fallos y continuar respondiendo de manera aceptable frente a daños parciales, en virtud de que contienen información almacenada de forma redundante.
- La disponibilidad de software y hardware apropiado para ser utilizados con R.N.A. en tiempo real.

3.6. Diseño del neurocontrolador

3.6.1. Introducción

En una planta de desalinización por ósmosis inversa siempre se necesita controlar la cantidad (flujo) y la calidad (conductividad) del producto obtenido (permeado). Este control se realiza en los bastidores de ósmosis inversa, y se consigue manipulando convenientemente la presión y el pH de la corriente de alimentación. En base a lo anterior, las variables de control serán:

- Variables controladas:

  - Flujo de permeado, ml/s

  - Conductividad del permeado, uS/cm

- Variables manipuladas

  - Presión de la corriente de alimentación, bar

  - pH de la corriente de alimentación.

Esto significa que en esencia el controlador neuronal tendrá dos entradas y dos salidas, sin embargo, dependiendo del modelo utilizado para entrenar la red neuronal, el número de entradas puede ser diferente.

Debido a que para diseñar un neurocontrolador dinámico se necesita un modelo matemático de la planta, ya sea neuronal o en el espacio de estados, en esta sección se empieza obteniendo dicho modelo, luego se entrena y valida el neurocontrolador, que finalmente se implementa en Simulink.

El modelo matemático para entrenar la red neuronal es una representación en el espacio de estados. El neuroncontrolador se entrena utilizando el algoritmo Dynamic Back Propagation y su implementación en Simulink se hace mediante un bloque MATLAB-Function.

3.6.2. Obtención de un modelo matemático en el espacio de estados de la planta

Se realiza una identificación no paramétrica en el dominio del tiempo del modelo no lineal de la unidad de O.I. de una planta de desalinización de agua de mar para obtener la representación en el espacio de estados requerida para el entrenamiento del controlador neuronal.

Como la planta es MIMO, las funciones de transferencia que relacionan sus entradas y salidas se obtienen utilizando el siguiente procedimiento:

- Se determina la respuesta temporal en lazo abierto del sistema frente a un cambio escalón en una sola de las entradas (manteniendo constante las demás).
- Se guardan los valores de los perfiles temporales de la entrada en cuestión y de todas las salidas afectadas por esta.
- Se utiliza el System Identification Toolbox de Matlab para obtener el modelo matemático a partir de los datos guardados.
- Se repiten los pasos anteriores hasta agotar todas las entradas de la planta.

Se parte de las condiciones de operación en el estado estacionario inicial del sistema (Alatiqi et al. 1989): $F_f = 315.45\ ml/s$, $P_f = 63.07\ bar$, $T_f = 25.00\ °C$, $C_f = 3000\ ppm$, $pH_f = 6.45$, $P_p = 1.01\ bar$, para obtener los datos de entrada y salida (requeridos para la identificación) utilizando los programas p01ROGeneracionDatosPFC.m y p02ROGeneracionDatospHC.m (anexo B). Los cambios escalón aplicados son del 10% para la presión y -10% para el pH de la alimentación.

Un análisis visual de los datos permite verificar que, sin pérdida de generalidad, se pueden utilizar modelos de función de transferencia de segundo orden, sin ceros ni tiempo muerto para relacionar todas las variables de control de entrada y salida.

Después de realizar la identificación del sistema mediante el System Identification Toolbox de Matlab, se obtienen los siguientes resultados:

- Relación: presión de la alimentación $(\boldsymbol{P})$ – flujo de permeado $(\boldsymbol{F})$:

$$\frac{F(s)}{P(s)} = \frac{1.6151}{3.8228\times10^{-4}s^2+3.3141s+1} \tag{3.77}$$

Se definen las variables de estado x_1= flujo de permeado debido a la presión de la alimentación, x_2= tasa de cambio del flujo de permeado debido a la presión de la alimentación, y la variable de entrada u_1= presión de la corriente de alimentación, para obtener:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -2615.9 & -8669.2\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 4225.0\end{bmatrix}u_1 \tag{3.78}$$

$$y_1 = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} \tag{3.79}$$

- Relación: presión de la alimentación $(\boldsymbol{P})$ – conductividad del permeado $(\boldsymbol{C})$:

$$\frac{C(s)}{P(s)} = \frac{-6.5250}{1.0403\times10^{-2}s^2+9.1671s+1} \tag{3.80}$$

Se definen las variables de estado x_3 = conductividad del permeado debido a la presión de la alimentación, x_4= tasa de cambio de la conductividad del permeado debido a la presión de la alimentación, para obtener:

$$\begin{bmatrix}\dot{x}_3\\ \dot{x}_4\end{bmatrix} = \begin{bmatrix}0 & 1\\ -96.13 & -881.21\end{bmatrix}\begin{bmatrix}x_3\\ x_4\end{bmatrix} + \begin{bmatrix}0\\ -627.23\end{bmatrix}u_1 \tag{3.81}$$

$$y_{2P} = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_3\\ x_4\end{bmatrix} \tag{3.82}$$

- Relación: pH de la alimentación (pH) – conductividad del permeado (C):

$$\frac{C(s)}{pH(s)} = \frac{-60.992}{75.3670\,s^2+18.1480s+1} \tag{3.83}$$

Se definen las variables de estado x_5= conductividad del permeado debido al pH de la alimentación, x_6= tasa de cambio de la conductividad del permeado debido al pH de la alimentación, y la variable de entrada u_2 = pH de la corriente de alimentación para obtener:

$$\begin{bmatrix}\dot{x}_5\\ \dot{x}_6\end{bmatrix} = \begin{bmatrix}0 & 1\\ -0.0133 & -0.2408\end{bmatrix}\begin{bmatrix}x_5\\ x_6\end{bmatrix} + \begin{bmatrix}0\\ -0.8093\end{bmatrix}u_2 \tag{3.84}$$

$$y_{2pH} = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_5\\ x_6\end{bmatrix} \tag{3.85}$$

La correspondencia entre el modelo lineal y los datos del modelo no lineal se verifica en las figuras 3.13, 3.14 y 3.15.

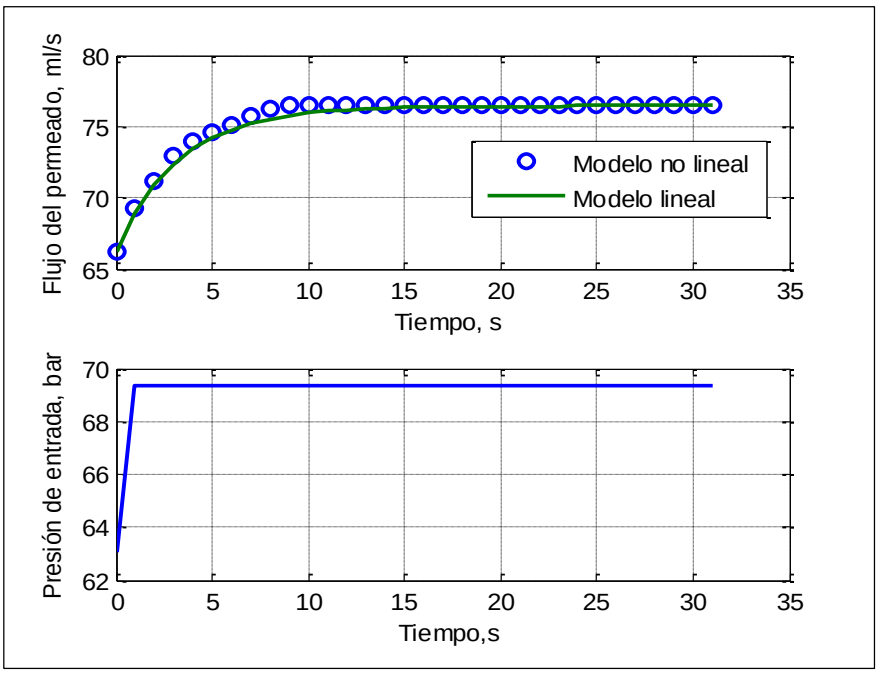

Figura 3.13.- Ajuste del modelo lineal a los datos del modelo no lineal: P – F.

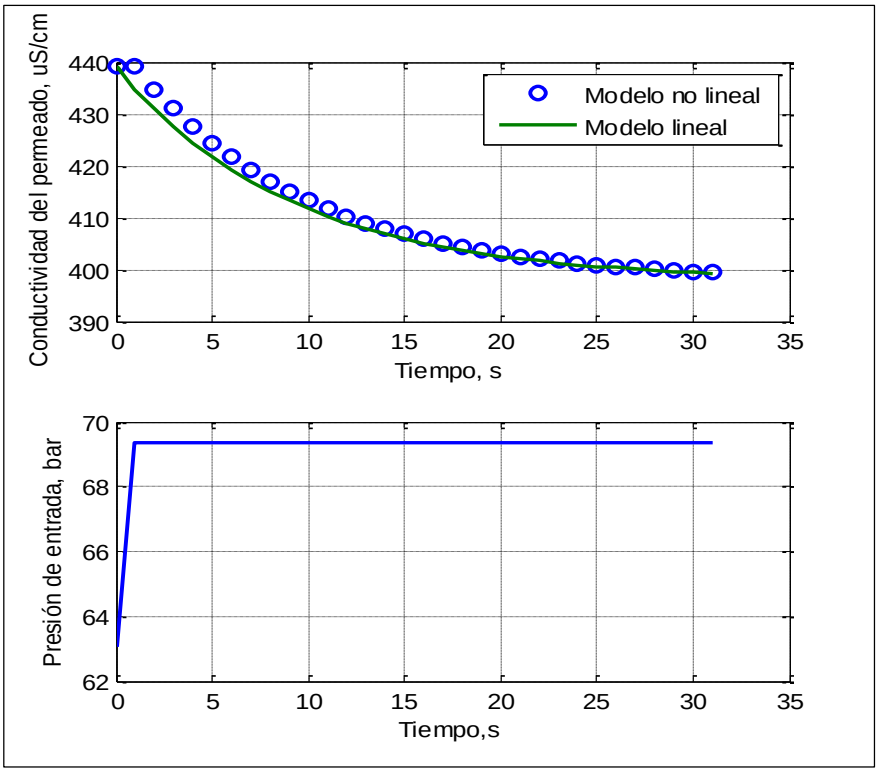

Figura 3.14.- Ajuste del modelo lineal a los datos del modelo no lineal: P - C

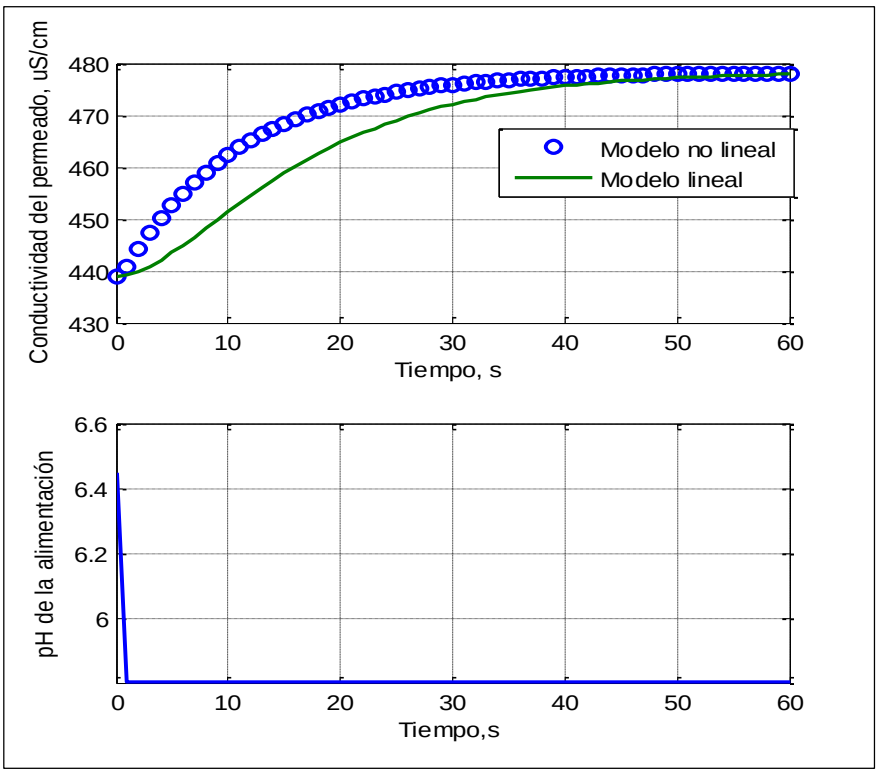

Figura 3.15.- Ajuste del modelo lineal a los datos del modelo no lineal: pH – C.

A partir de las representaciones en el espacio de estado individuales, ecuaciones (3.78), (3.79), (3.81), (3.82), (3.84) y (3.85), se escribe la representación general en el espacio de estados del modelo lineal de la planta:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\\ \dot{x}_4\\ \dot{x}_5\\ \dot{x}_6\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\\ -2615.9 & -8669.2 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & -96.13 & -881.21 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & -0.0133 & -0.2408\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\end{bmatrix}+\begin{bmatrix}0 & 0\\ 4225.0 & 0\\ 0 & 0\\ -627.23 & 0\\ 0 & 0\\ 0 & -0.8093\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{3.86}$$

$$\begin{bmatrix}y_1\\ y_2\end{bmatrix}=\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\\ x_4\\ x_5\\ x_6\end{bmatrix} \tag{3.87}$$

Es importante mencionar que todas las variables de las funciones de transferencia son las variables de desviación respecto a sus valores de estado estacionario, y debido a esto, los estados también son variables de desviación.

Se discretiza la ecuación (3.86), con un tiempo de muestreo de un segundo, para obtener el modelo discreto lineal de la unidad de O.I. de una planta de desalinización de agua:

$$X(k+1)=A*X(k)+B*U(k) \tag{3.88}$$

$$Y(k)=C*X(k), \tag{3.89}$$

con:

$$A=\begin{bmatrix}0.73954 & 8.53\mathrm{e}-05 & 0 & 0 & 0 & 0\\ -0.22316 & -2.57e-05 & 0 & 0 & 0 & 0\\ 0 & 0 & 0.89675 & 0.00102 & 0 & 0\\ 0 & 0 & -0.09784 & -0.00011 & 0 & 0\\ 0 & 0 & 0 & 0 & 0.99387 & 0.88675\\ 0 & 0 & 0 & 0 & -0.01177 & 0.78035\end{bmatrix}$$

$$B=\begin{bmatrix}0.42067 & 0\\ 0.36043 & 0\\ -0.67369 & 0\\ -0.63837 & 0\\ 0 & -0.37362\\ 0 & -0.71762\end{bmatrix}$$

$$C=\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 1 & 0\end{bmatrix}$$

3.6.3. Entrenamiento del neurocontrolador

En base a las ecuaciones de diseño deducidas en la sección 3.4.2, se escribe el programa p05MultivariableNeuralController.m (anexo C) para entrenar la red neuronal dinámica que constituirá el controlador multivariable.

De acuerdo a la representación en el espacio de estados de la planta, ecuación (3.86), el neurocontrolador tendrá seis (6) entradas y dos (2) salidas.

Se utiliza la función tangente hiperbólica como función de activación para todas las neuronas de la capa intermedia, y se mantienen fijos sus centros en el valor de cero y sus inclinaciones en el valor de uno.

En las condiciones normales de operación de la planta la magnitud del flujo de permeado está alrededor de 60 – 70 ml/s, la conductividad del permeado está entre 420 – 460 uS/cm, la presión de la alimentación entre 60 – 70 bar y el pH entre 6.2 – 6.6 (Alatiqi et. al, 1989). Suponiendo además que:

- La tasa de cambio del flujo de permeado es a lo mucho +- 0.5 ml/s^2.
- La tasa de cambio de la conductividad del permeado debida a la presión es a lo mucho +- 5 uS/cm/s.
- La tasa de cambio de la conductividad del permeado debida al pH es a lo mucho +- 5 uS/cm/s.

Se utilizan los siguientes factores de escalamiento:

$$me_1 = 0.10$$
$$me_2 = 2.00$$
$$me_3 = 0.02$$
$$me_4 = 0.20$$
$$me_5 = 0.02$$
$$me_6 = 0.20$$

para que las entradas a la red neuronal estén en el rango [-1, 1]. Y los factores de escalamiento:

$$ms_1 = 5.00$$
$$ms_2 = 0.50$$

para llevar las salidas del neurocontrolador desde valores alrededor de [-1, 1] hasta sus órdenes de magnitud correspondientes.

Se entrena el controlador neuronal para llevar el sistema desde múltiples condiciones iniciales hasta su punto normal de operación.

En el proceso de entrenamiento se observa que, inicializando los pesos de las matrices V y W de forma aleatoria en valores menores a 0.025 y utilizando un ratio de aprendizaje de 0.0005, se consiguen resultados satisfactorios para diferente número de neuronas en la capa intermedia (por ejemplo nm = 3, 8).

Cuando el ratio de aprendizaje es mayor que 0.0005 no se consigue convergencia, y para valores menores, el entrenamiento es muy lento.

Luego de varias etapas de entrenamiento consistentes de cinco mil iteraciones cada una, se consiguen los resultados buscados. Para el caso particular en el que el número de neuronas en la capa intermedia es de tres (3), las matrices de pesos obtenidas son:

$$V = \begin{bmatrix} 0.0081 & 0.0135 & -0.0215 \\ -0.0206 & 0.0167 & 0.0195 \\ -0.0452 & -0.0210 & 0.0077 \\ 0.0467 & -0.0150 & -0.0058 \\ -0.0150 & -0.0057 & -0.0264 \\ 0.0025 & 0.0381 & -0.0071 \end{bmatrix}$$

$$W = \begin{bmatrix} 0.0098 & -0.0199 \\ -0.0416 & -0.0025 \\ -0.0048 & 0.0077 \end{bmatrix}$$

$$c = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$a = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

A continuación se valida el entrenamiento del controlador neuronal diseñado, evaluando su desempeño en el control del modelo lineal de la planta frente a un cambio escalón de 10% en el punto de consigna del flujo de permeado. Observe que, como la presión afecta a la conductividad, el controlador debe ser capaz de compensar el cambio de esta variable modificando el pH de la alimentación. Los resultados obtenidos al ejecutar el programa p06MultivariableNeuralControllerValidation.m del anexo C se presentan en las figuras 3.16, 3.17, 3.18 y 3.19.

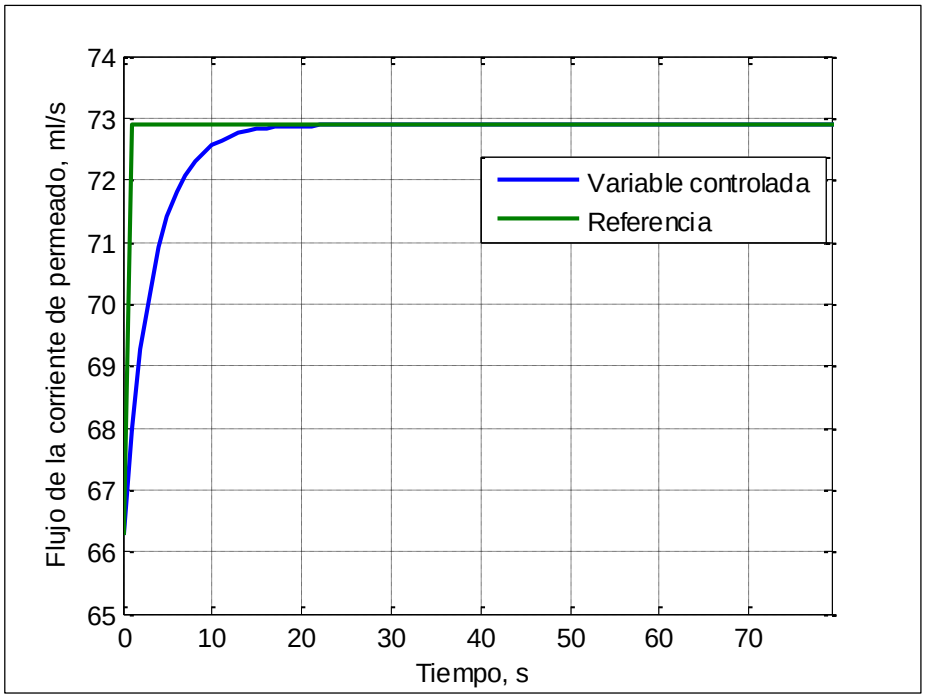

Figura 3.16. Respuesta en lazo cerrado, del sistema de control neuronal para el modelo lineal de la planta, frente a un cambio escalón de 10% en el set point del flujo de permeado.

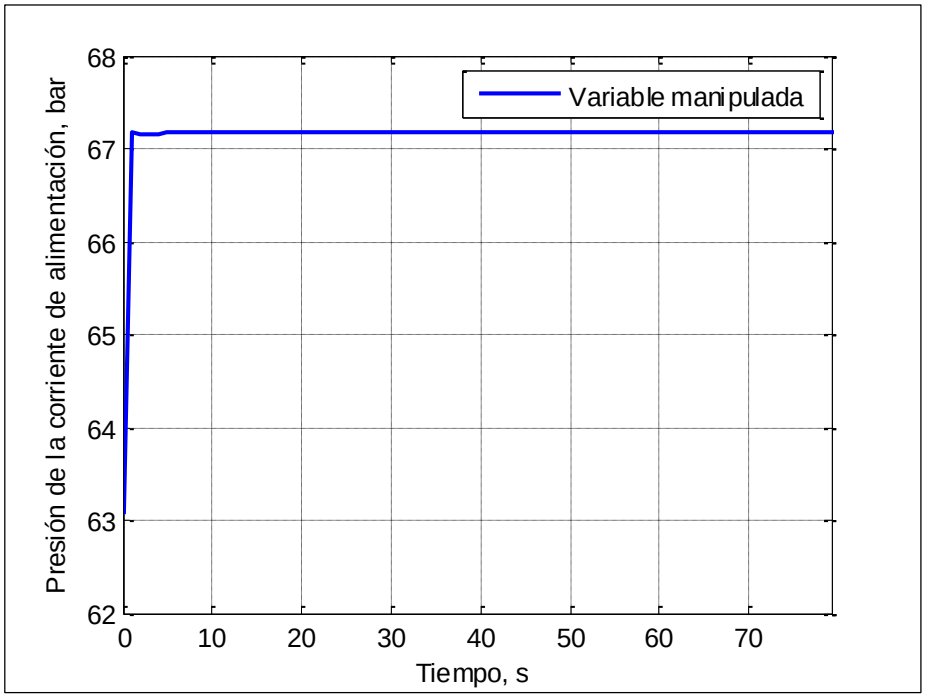

Figura 3.17. Respuesta en lazo cerrado, del sistema de control neuronal para el modelo lineal de la planta, frente a un cambio escalón de 10% en la referencia del flujo de permeado.

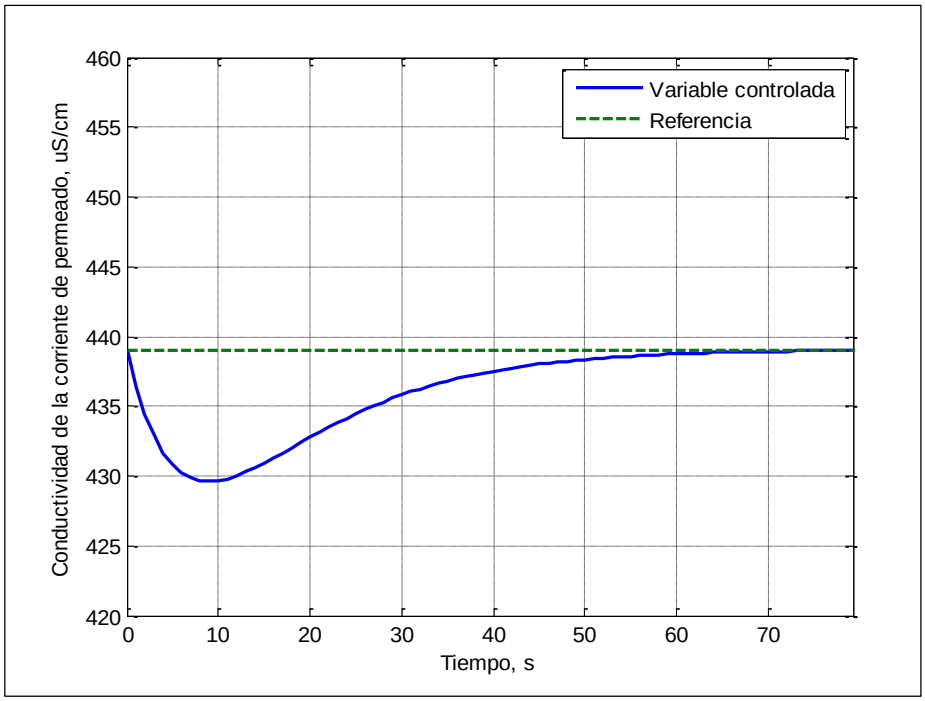

Figura 3.18. Respuesta en lazo cerrado, del sistema de control neuronal para el modelo lineal de la planta, frente a un cambio escalón de 10% en la referencia del flujo de permeado.

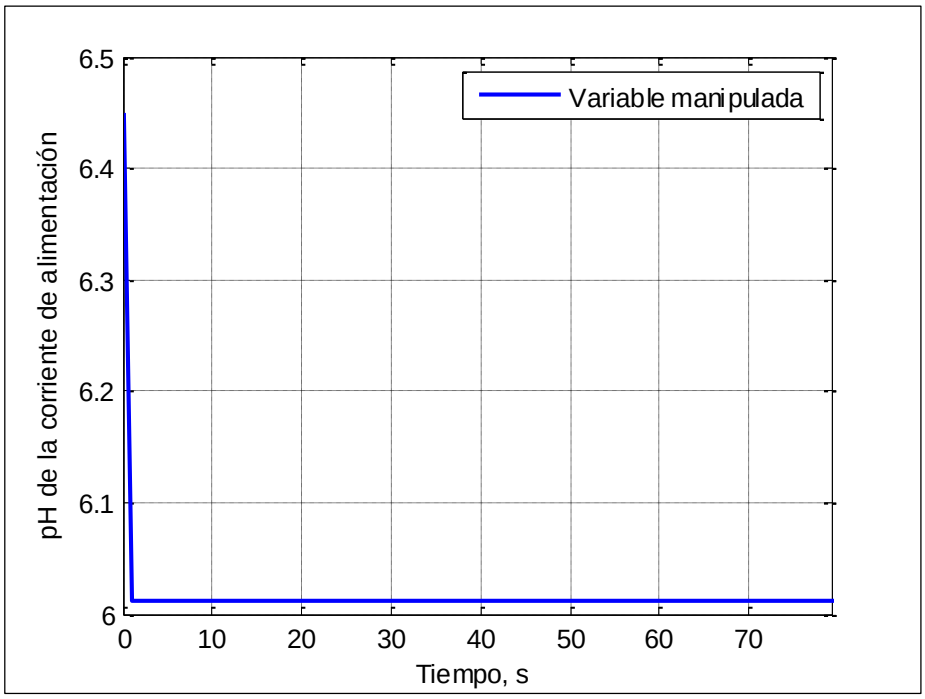

Figura 3.19. Respuesta en lazo cerrado, del sistema de control neuronal para el modelo lineal de la planta, frente a un cambio escalón de 10% en la referencia del flujo de permeado.

3.6.4. Implementación del neuroncontrolador en Simulink

El neurocontrolador multivariable diseñado en la sección 3.5.3 se implementa en Simulink mediante un bloque MATLAB-Function para controlar el modelo no lineal de la unidad de O.I. de una planta piloto de desalinización de agua de mar. El algoritmo que constituye el controlador neuronal es el programa ControladorNeuronal.m del anexo C. La estructura del sistema de control resultante se presenta en la Figura 3.20.

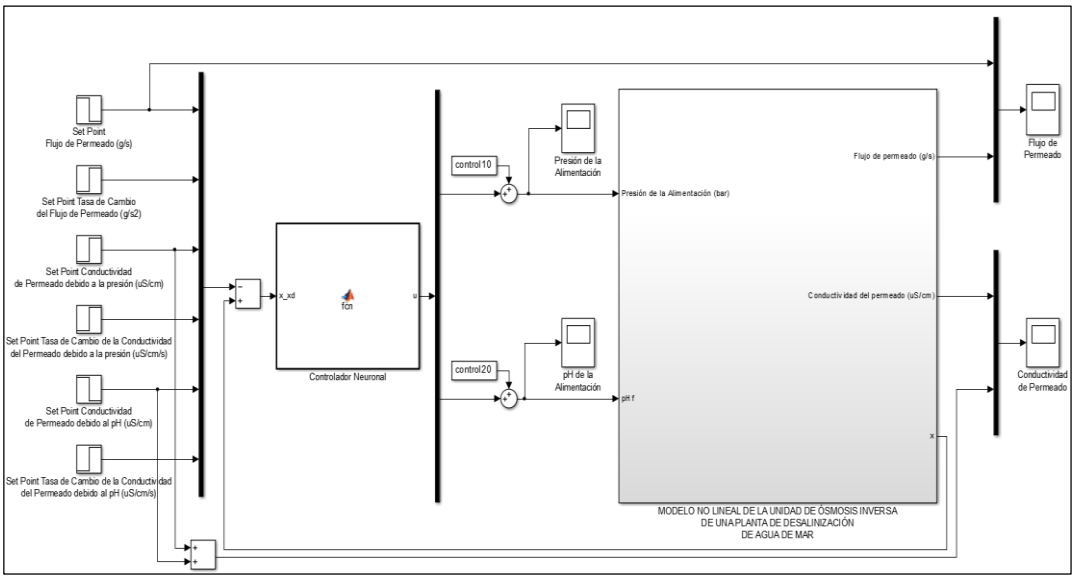

Figura 3.20. Interfaz del sistema de control multivariable basado en R.N.A para el modelo no lineal de la unidad de O.I. de una planta desalinizadora de agua de mar.

3.7. Evaluación del desempeño del neurocontrolador diseñado

Dado que en la práctica es bastante natural encontrar situaciones en las que se desea cambiar la cantidad de la producción sin afectar su calidad, se evalúa el desempeño del neurocontrolador diseñado (e implementado en Simulink para el control del modelo no lineal de la planta), frente a un cambio escalón de -5% en la referencia del flujo de permeado, considerando la presencia de ruido blanco en los sensores (de aplitud máxima igual al 0.5% de la magnitud de la variable medida) y de una perturbación de -5% en la concentración de las sales de la corriente de alimentación. Los resultados obtenidos después de ejecutar el programa p01controlNeuronalnoLineal.m del anexo D se presentan en las figuras 3.21, 3.22, 3.23 y 3.24.

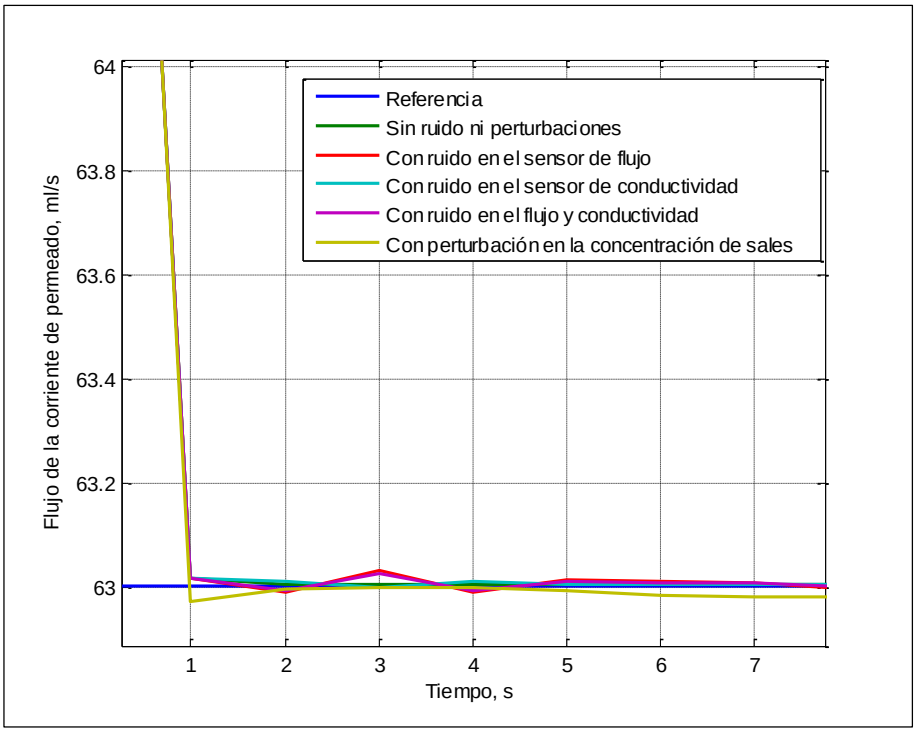

Figura 3.21. Respuesta en lazo cerrado, del sistema de control neuronal del modelo no lineal de la planta, frente a un cambio escalón de -5% en la referencia del flujo de permeado. Variable controlada: Flujo de permeado.

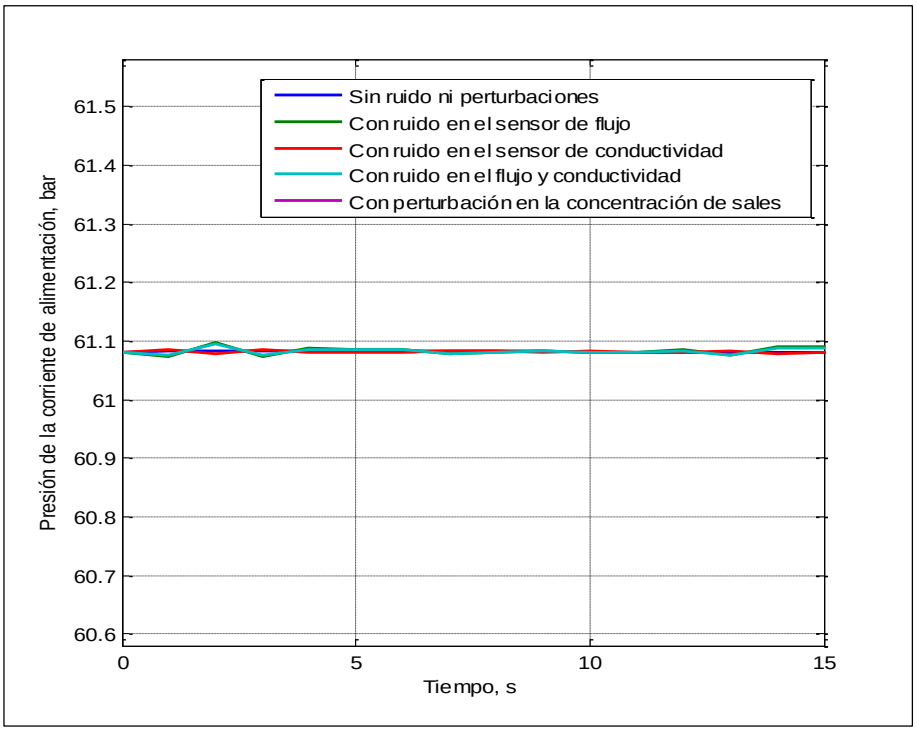

Figura 3.22. Variaciones de la variable manipulada: Presión de la corriente de alimentación.

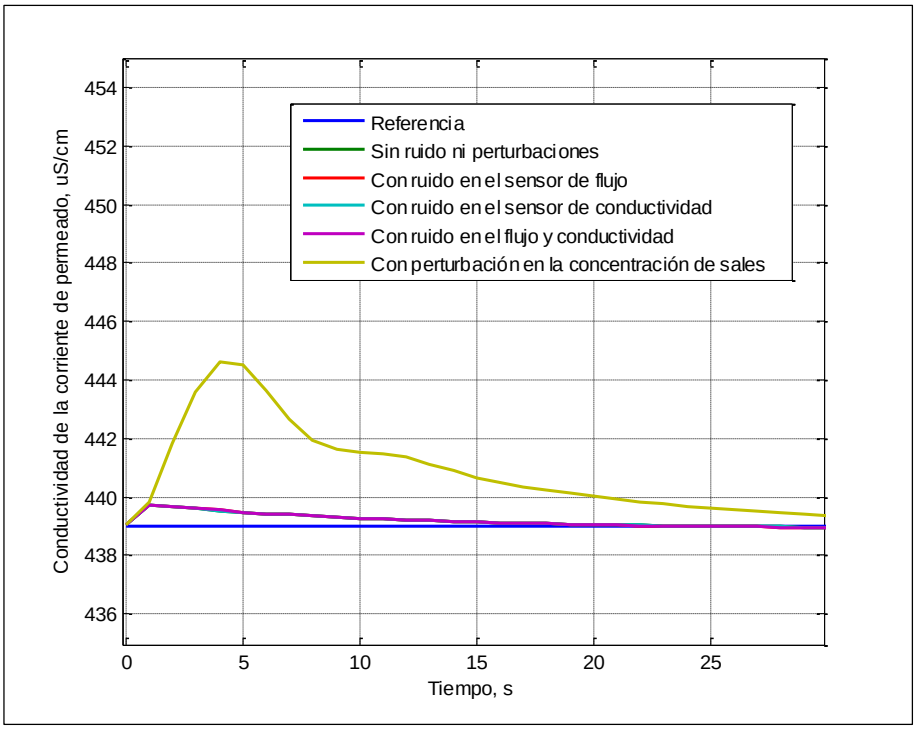

Figura 3.23. Respuesta en lazo cerrado, del sistema de control neuronal del modelo no lineal de la planta, frente a un cambio escalón de -5% en la referencia del flujo de permeado. Variable controlada: Conductividad del permeado.

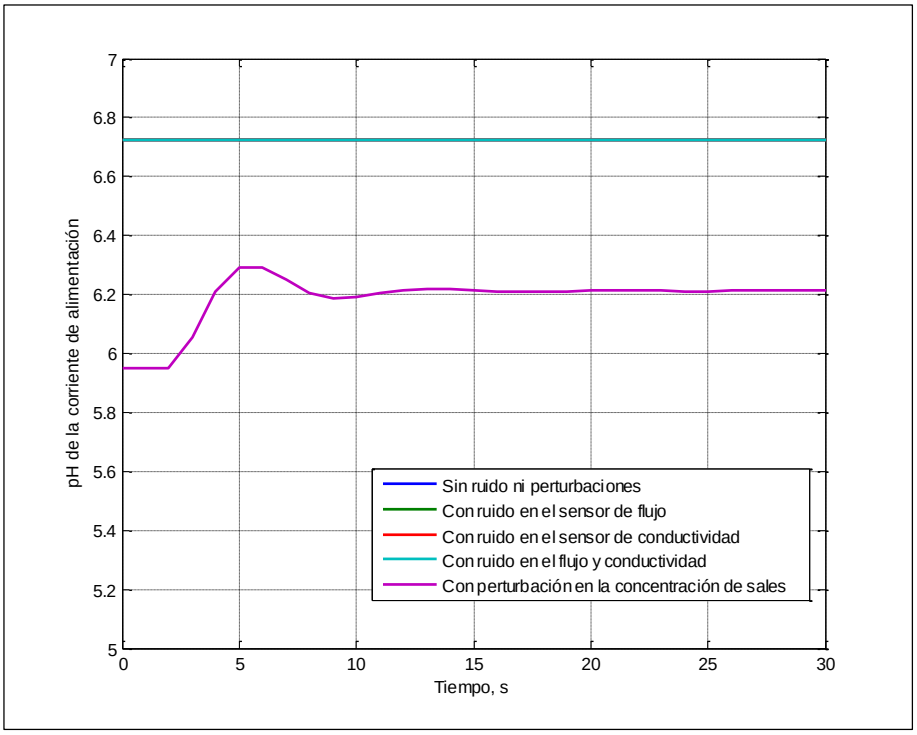

Figura 3.24. Variaciones de la variable manipulada: pH de la corriente de alimentación.

3.8. Análisis de resultados

Aunque parecen resultados simples, las figuras presentadas en la sección 3.6 son producto de muchas horas de esfuerzo y dedicación. Primero se tuvo que deducir una expresión general del algoritmo de entrenamiento Dynamic Back Propagation para que pueda utilizarse en el diseño de neurocontroladores con múltiples entradas y múltiples salidas, luego fue necesario establecer un procedimiento, también de carácter general, para el escalamiento de las señales de entrada y salida al neurocontrolador; posteriormente se linealizó el modelo matemático de la planta para obtener una representación en el espacio de estados, después se programó el algoritmo para el diseño del controlador que posteriormente se entrenó teniendo en cuenta la cantidad de neuronas en la capa intermedia, los ratios de aprendizaje de las matrices de pesos, el número de etapas de entrenamiento, las diferentes condiciones iniciales utilizadas para el entrenamiento, entre otros factores. Finalmente se implementa en simulink el neurocontrolador diseñado.

En la sección 3.4.2 se desarrolla una generalización del algoritmo Dynamic Back Propagation que permite diseñar controladores neuronales multivariables. El resultado fue implementado en el programa p05MultivariableNeuralController.m del anexo C, verificándose en este la efectividad del algoritmo DBP para entrenar esta clase de controladores.

En la sección 3.4.3 se deduce un procedimiento de carácter general para la transformación lineal del modelo matemático empleado para el entrenamiento del neurocontrolador y también para el escalamiento de sus señales de entrada y salida. Vale mencionar que en el presente trabajo se intentó entrenar la red neuronal sin hacer escalamiento de las señales y no se pudieron conseguir resultados satisfactorios, mientras que, luego de escalar adecuadamente las variables, el entrenamiento fue inmediato.

Se resalta además que, aunque el controlador fue diseñado para una linealización del modelo matemático obtenido en el capítulo 2, este se desempeña bastante bien en el control del modelo no lineal base, especialmente alrededor de las condiciones para las cuales se hizo la linealización. Resultados adicionales de simulación muestran que, este buen desempeño se empobrece a medida que el sistema se aleja más de su condición normal de operación. Debido a lo anterior, resulta interesante

pensar que se pueden obtener mejores resultados durante el control si el entrenamiento se realiza utilizando un modelo neuronal de la planta en vez del modelo en el espacio de estados.

En la sección 3.5.2 se obtiene una representación en el espacio de estados del sistema mediante la identificación no parámetrica en el dominio del tiempo de la planta (modelo no lineal) alrededor de sus condiciones normales de operación (cambios de +- 10% en las señales de entrada). Los resultados de las figuras 3.10, 3.11 y 3.12 muestran un buen ajuste del modelo lineal al no lineal en el rango considerado.

El entrenamiento del neurocontrolador se realizó teniendo en cuenta la cantidad de neuronas en la capa intermedia, los ratios de aprendizaje de las matrices de pesos, el número de etapas de entrenamiento, las diferentes condiciones iniciales utilizadas para el entrenamiento, el valor máximo aceptable para la suma de los cuadrados de los errores y muchos factores más que afortunadamente permitieron realizar el entrenamiento de forma satisfactoria y conseguir los resultados de la sección 3.5.3.

Aparentemente se necesitan seis referencias para el neurocontrolador, sin embargo, tres de ellas (las que se definieron como tasas de cambio), siempre son cero en las condiciones de operación en estado estacionario. Esto significa que en realidad solo se necesitan tres set ponits, a saber, la cantidad requerida del producto (flujo) y su calidad (conductividad) debida a la presión y debida al pH.

Observe que la implementación, en una planta real, del controlador diseñado en la sección 3.5.3, requerirá del conocimiento de los estados x_2, x_4 y x_6, los cuales no son medidos directamente. Esto requiere que en dichas plantas se emplee algún observador de estados, que afortunadamente no es necesario para nuestro caso, pues en el modelo matemático no lineal, tales estados pueden ser obtenidos fácilmente. Esta facilidad del modelo matemático no lineal, para simular a una planta real, sugiere que podría utilizarse como una especie de observador y más aún como un “predictor” para algún otro algoritmo de control. Esta tendencia de utilizar modelos matemáticos teóricos para anticiparse y tomar acción en la optimización del desempeño de plantas industriales es una práctica cada vez más común en la industria de los procesos químicos (Rangaiah y Kariwala, 2012).

Es muy importante resaltar que el controlador neuronal diseñado de por si ya tiene naturaleza desacoplada, y que la interacción entre las variables pasa prácticamente desapercibida, permitiendo un control rápido, suave, sin oscilaciones, ni sobreimpulso.

Observe además que el controlador es prácticamente insensible al ruido de los sensores, aunque si presenta dificultades frente a perturbaciones, reflejando con ello la necesidad de realizar más entrenamiento de la red neuronal para estas situaciones.

3.9. Conclusiones parciales

- Se obtuvo un modelo lineal multivariable en el espacio de estados para la unidad de ósmosis inversa de una planta piloto de desalinización de agua de mar, el cual presenta un buen grado de ajuste al modelo no lineal desarrollado en el capitulo 2.
- Se diseñó un controlador multivariable basado R.N.A. para el control de las variables críticas de la planta de desalinización de agua de mar objeto de estudio.
- Los resultados de simulación del sistema de control con el controlador diseñado mostraron un buen seguimiento de la referencia, con una respuesta rápida, suave, sin oscilaciones, ni sobreimpulsos, evidenciándose con ello la naturaleza desacoplada del controlador neuronal.
- A pesar de que con el controlador diseñado se logra controlar a la planta objeto de estudio dentro de sus rangos de operación nominal, los resultados obtenidos muestran que se necesita un mayor entrenamiento de la R.N.A. para mejorar las respuestas temporales del sistema de control en términos de rechazo a perturbaciones.

4. PROPUESTA DE IMPLEMENTACIÓN PRÁCTICA DEL SISTEMA DE CONTROL DISEÑADO

4.1. Introducción

Aunque en el capítulo 3 se demostró que el controlador basado en R.N.A. diseñado permite controlar de forma satisfactoria la planta objeto de estudio, es necesario comparar su desempeño frente a otros controladores, en particular frente a controladores PID, para evaluar su efectividad y sus ventajas. Esta tarea se realiza en el presente capítulo. Posteriormente se hace una propuesta de implementación práctica del sistema de control diseñado.

4.2. Análisis comparativo del sistema de control neuronal diseñado vs. un sistema de control con controladores PID

Se diseñan controladores PID en el dominio de la frecuencia utilizando márgenes de fase y de ganancia (Rivas-Perez et al., 2014e). Se encuentra que la acción derivativa no ofrece mejoras significativas en el control del flujo, y por ello se selecciona un controlador PI que tiene los siguientes parámetros de sintonía:

$$K_C = 0.01, \qquad T_I = 0.15$$

De manera similar, para el control de la conductividad, se determinó que basta con un controlador PI con los siguientes parámetros de sintonía:

$$K_C = -7.5 \times 10^{-4}, \qquad T_I = -1.15 \times 10^{-7}$$

En la Figura 4.1 se muestra la estructura del sistema de control PID diseñado. Se evalúa el desempeño del controlador PI diseñado para el flujo del permeado frente a un escalón de 5% en el setpoint. También, se evalúa su robustez frente a perturbaciones, para lo cual, se consideraron escalones de -5% y 10% en la concentración y temperatura de la corriente de alimentación a los 200 y 300 segundos de iniciada la simulación. Los resultados obtenidos al ejecutar el programa p01ROFlowControl.m del anexo E se presentan en las Figuras 4.2 y 4.3.

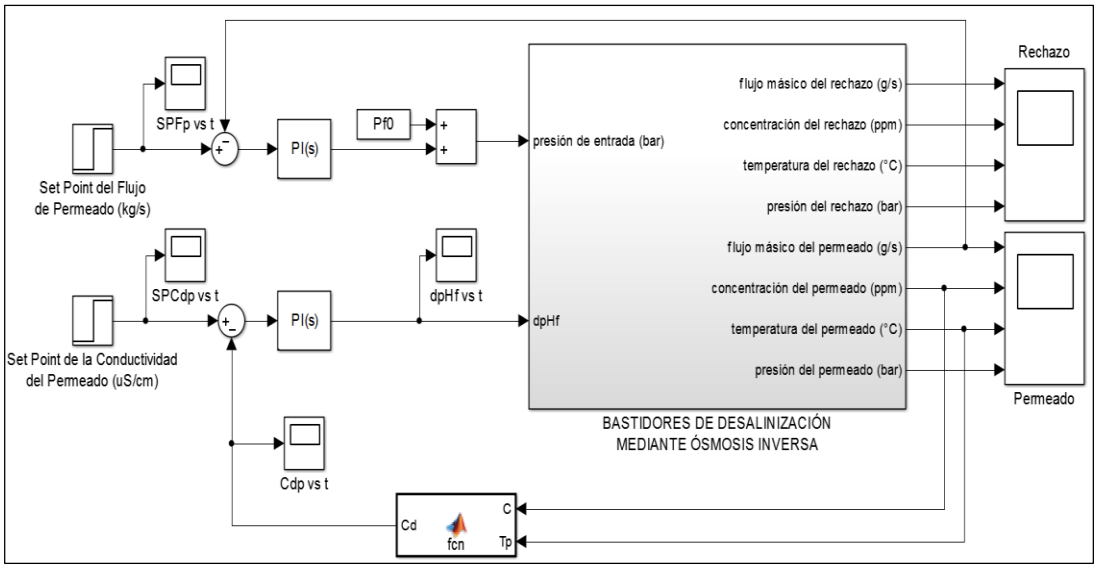

Figura 4.1. Estructura del sistema de control con los controladores PI diseñados.

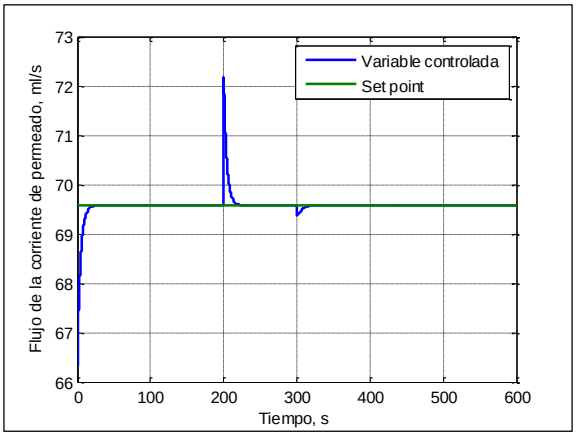

Figura 4.2. Respuesta temporal del sistema de control del flujo del permeado con el controlador PI diseñado.

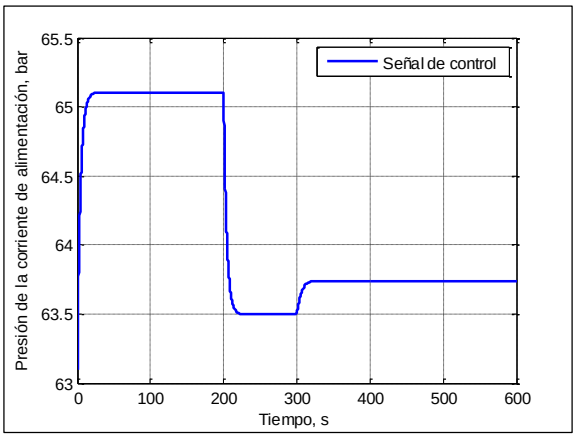

Figura 4.3. Variaciones de la variable manipulada: Presión

Luego, se evalúa el desempeño del controlador PI diseñado para la conductividad del permeado frente a un escalón de -10% en el setpoint. Los resultados obtenidos al ejecutar el programa p02ROConductivityControl.m del anexo E se presentan en las Figuras 4.4 y 4.5.

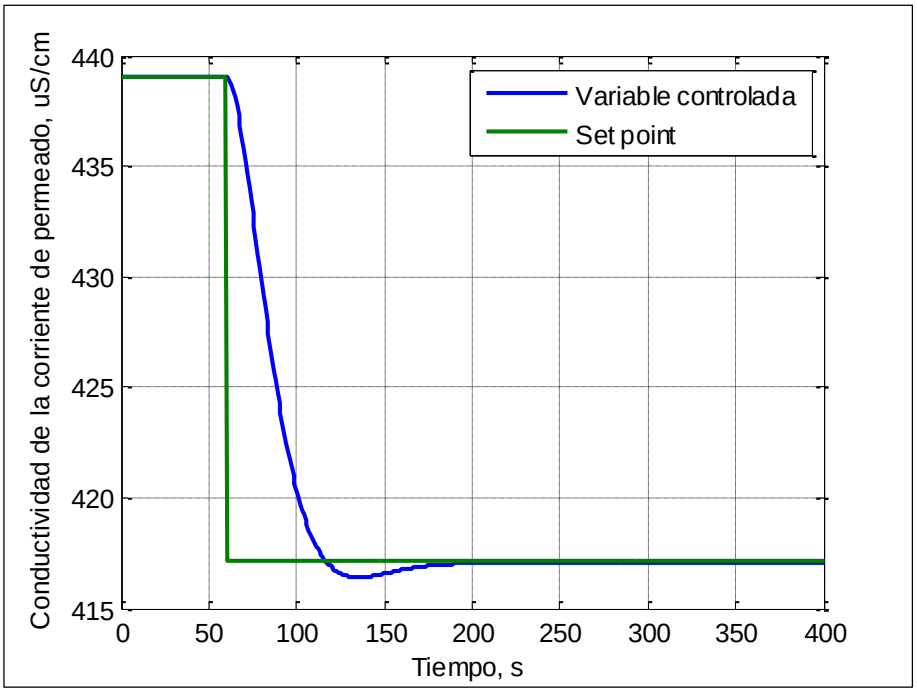

Figura 4.4. Respuesta temporal del sistema de control de la conductividad del permeado con el controlador PI diseñado.

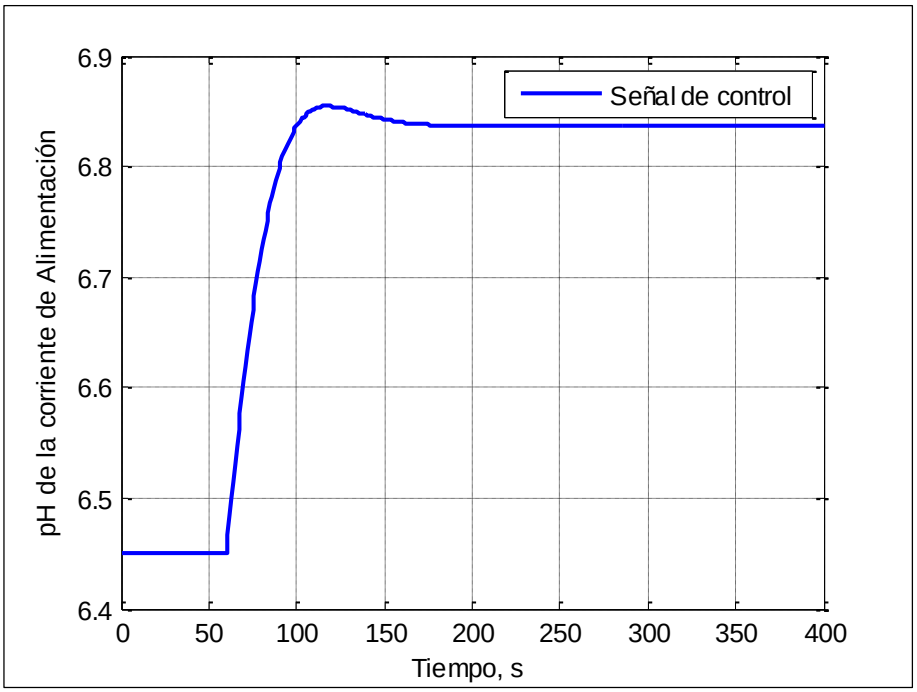

Figura 4.5. Variaciones de la variable manipulada: pH

Finalmente se comparan las respuestas temporales de los sistemas de control PI y neuronal diseñados para la planta objeto de estudio. Los resultados obtenidos al ejecutar el programa p01controlNeuronalvsPInoLineal.m del anexo E se presentan en las Figuras 4.6, 4.7, 4.8 y 4.9.

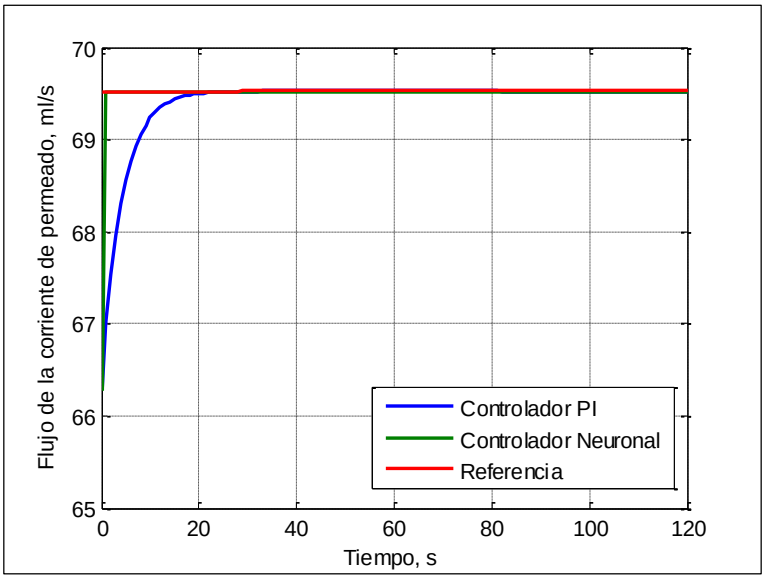

Figura 4.6. Comparación de las respuestas temporales del sistema de control con controladores PID vs. neuronal del flujo del permeado.

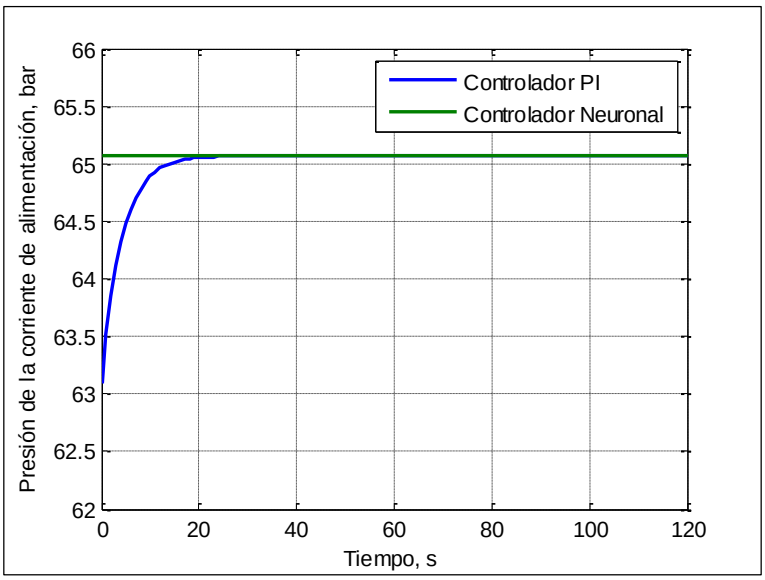

Figura 4.7. Comparación de las señales de control del sistemas de control del flujo del permeado con controladores PI vs. neuronal.

En la Figura 4.6, se observa que el sistema de control del flujo del permeado con el controlador neuronal responde más rápidamente.

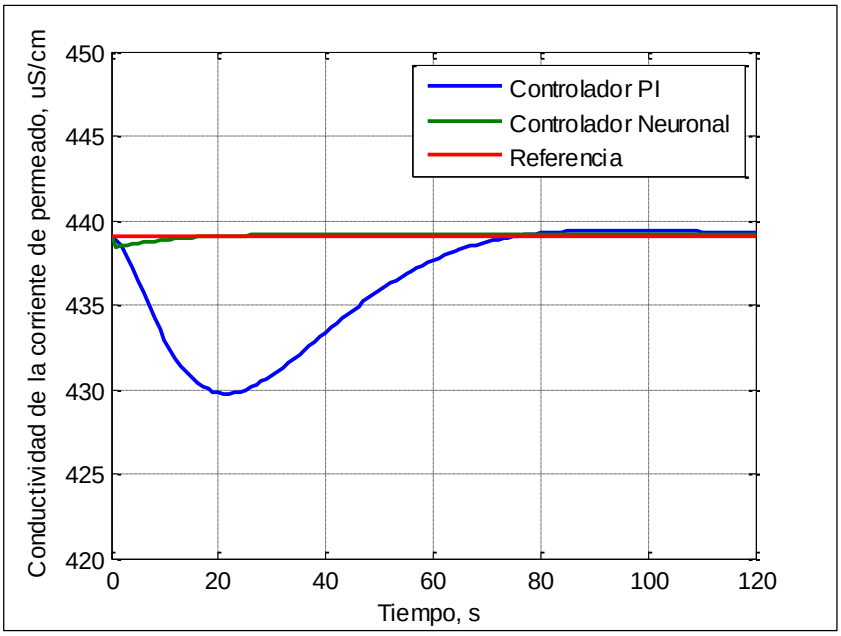

Figura 4.8. Comparación de las respuestas temporales del sistema de control de la conductividad del permeado con controladores PID vs. neuronal.

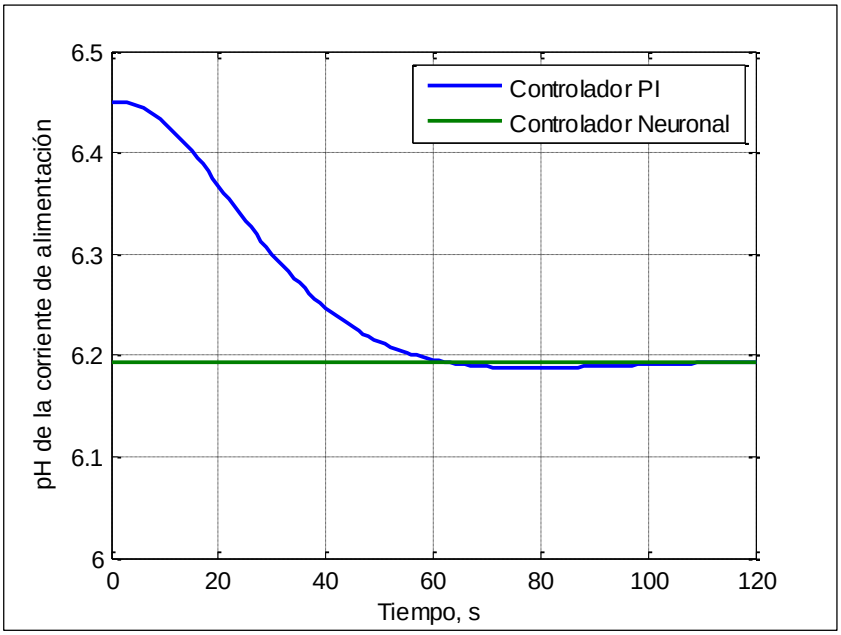

Figura 4.9. Comparación de las señales de control del sistemas de control de la conductividad del permeado con controladores PI vs. neuronal.

En las Figuras 4.8 y 4.9 se observa que el neurocontrolador tiene una tendencia más suave y es menos sensible a la interación presentada entre las variables.

En la Tabla 4.1 se presentan las especificaciones de las respuestas temporales obtenidas del sistema de control con los controladores PI vs. neuronal diseñados. De la tabla es evidente que los mejores resultados en las respuestas temporales del sistema de control se obtienen con el controlador neuronal.

Tabla 4.1. Especificaciones de las respuestas temporales del sistema de control neuronal diseñados vs. controladores PI.

CONTROLADOR	**ESPECIFICACIÓN**		
	Error de estado estacionario	**Tiempo de establecimiento (s)**	**Sobreimpulso Mp (%)**
	Variable: Flujo del permeado		
PI	0	20	0
Neuronal	0	2	0
	Variable: Conductividad del permeado		
PID	0	72	0
Neuronal	0	16	0

Se utiliza la integral del cuadrado del error (ISE) como índice de desempeño para cuantificar el rendimiento de los sistemas de control diseñados, el cual se define como (Rivas-Perez et al., 2014d; Smith y Corripio, 1997):

$$ISE = \int_0^\infty e^2(t)dt, \tag{4.1}$$

donde e(t) es la señal de error en el dominio del tiempo.

Los resultados obtenidos de cálculo del ISE se muestran en la Tabla 4.2.

Tabla 4.2. Índice de desempeño de los sistemas de control diseñados.

Controlador	**ISE Flujo**	**ISE conductividad**
PID	16.6741	2.3754e+03
Neuronal	10.5522	3.0106

De la Tabla 4.2 es posible observar que el menor índice de desempeño del sistema de control se obtiene con el controlador neuronal, significando que este posibilita controlar de forma efectiva a la planta objeto de estudio.

4.3. Propuesta de implementación práctica del controlador diseñado

Para la implementación práctica del sistema de control basado en R.N.A. diseñado para la planta objeto de estudio, se propone utilizar un cliente/servidor OPC, el cual puede ser programado en Matlab. Esta solución posibilita utilizar el controlador neuronal diseñado en en una PC (Simulink) para controlar la planta a través de un PLC. En la Figura 4.10, se muestra un diagrama de esta propuesta de implementación práctica.

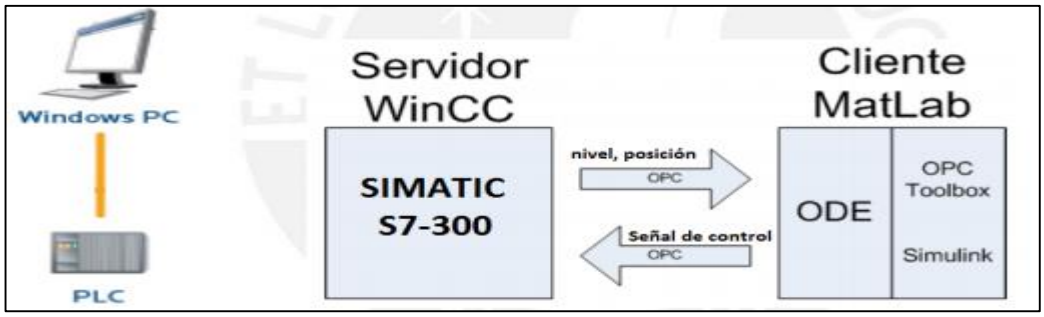

Figura 4.10. Diagrama de comunicación cliente/servidor OPC y un PLC.

Los estándares de comunicación OPC (OLE for Process Control) soportan las tecnologías OLE, COM, DCOM y ActiveX creadas por Microsoft (Sabin y González, 2010) permitiendo el intercambio de datos y la interoperabilidad entre dispositivos de campo, sistemas de control y los software de gestión de la producción en la supervisión y control de procesos industriales (Zheng y Nakagawa, 2002). La comunicación se realiza mediante una aplicación cliente/servidor, donde el servidor OPC es el que contiene la interfaz común para la comunicación y permite que cualquier aplicación pueda escribir/leer sobre/a cualquier variable que de este (Zamarreño, 2001).

Para el control de la planta de desalinización por O.I. objeto de estudio, las mediciones de las variables controladas (flujo y conductividad del permeado) recolectadas a través del controlador local (SIMATIC S7-300) se envían a la PC (cliente), donde se encuentra activa la aplicación del controlador neuronal en Simulink diseñado en el presente trabajo. La señal de control calculada por el controlador neuronal se envía a través del PLC a la planta.

De esta manera puede realizarse la implementación práctica y económica del sistema de control diseñado en la planta pilot de desalinización de la PUCP.

4.4. Conclusiones parciales

- Se diseñaron dos controladores PI desacoplados para el control de las variables críticas de la planta objeto de estudio.
- Los resultados de simulación del sistema de control con los controladores PI diseñados mostraron un buen desempeño frente a cambios en la referencia, así como frente a perturbaciones externas, evidenciando con ello que el diseño de estos controladores se realizó correctamente.
- Las comparaciones de las respuestas temporales del sistema de control de la planta objeto de estudio con los controladores neuronal y PI diseñados mostraron que, el mejor desempeño del sistema de control se obtiene con el controlador neuronal.
- Se desarrolló una propuesta simple y económica de implementación práctica del controlador neuronal multivariable diseñado utilizando un cliente/servidor OPC.

CONCLUSIONES

Aplicando las leyes de conservación de la materia y la energía, se obtuvo un modelo matemático multivariable para la unidad de ósmosis inversa de una planta desalinizadora de agua de mar.

Los resultados satisfactorios alcanzados en la validación del modelo matemático con datos reales de operación de una planta piloto permitieron verificar que el modelo describe de forma adecuada el comportamiento dinámico de dicha planta, ya que se ajusta con un 92.37% a los datos experimentales de la dependencia del flujo de permeado con la presión de la alimentación. Con un 74.57% a los datos experimentales de la conductividad del permeado en función de la presión de la alimentación. Y un 78.71% para la conductividad del permeado en función del pH de la alimentación.

Se obtuvo un modelo lineal multivariable en el espacio de estados mediante la identificación no paramétrica en el dominio del tiempo del modelo no lineal desarrollado en el capítulo 2. Este modelo presenta un buen grado de ajuste al modelo no lineal alrededor de sus condiciones normales de operación.

Para el control efectivo de la planta objeto de estudio se diseñó un controlador multivariable basado en R.N.A. Los resultados de simulación del sistema de control mostraron un buen seguimiento de la referencia, así como una considerable robustez frente a las perturbaciones externas y a la presencia de ruido en los sensores.

Se diseñaron dos controladores PI desacoplados para la evaluación comparativa del sistema de neurocontrolador diseñado. La comparación de las respuestas temporales del sistema de control de la planta objeto de estudio con los controladores PI y neuronal diseñados, mostraron que el mejor desempeño del sistema de control se obtiene cuando se aplica el controlador multivariable basado en R.N.A.

Se realizó una propuesta de implementación práctica del controlador multivariable neuronal diseñado, basada en la aplicación de un cliente/servidor OPC programado en una PC y un PLC SIMATIC S7-300 para la comunicación con la planta.

RECOMENDACIONES

Validar el modelo matemático obtenido con los datos experimentales de la planta piloto de desalinización por O.I. de la PUCP.

Utilizar el modelo matemático no lineal obtenido como observador de estados y/o algoritmo predictor en otras estrategias avanzadas de control.

Utilizar un modelo neuronal de la planta en vez del modelo en el espacio de estados para entrenar el controlador neuronal.

Realizar un mayor entrenamiento del neurocontrolador diseñado para mejorar su desempeño frente a perturbaciones externas.

Implementar el neurocontrolador diseñado en la planta piloto de desalinización por O.I. de la PUCP.

BIBLIOGRAFÍA

- Abbas A. (2006). Model predictive control of a reverse ósmosis desalination unit. Desalination, 194(1-3), 268-280.
- Abbas A. and Al-Bastaki N. (2005). Modeling of an RO water desalination unit using neural networks. Chemical Engineering Journal, 114(1-3), pp.139-143.
- Al-Bastaki N. and Abbas A. (1999). Modeling an industrial reverse ósmosis unit. Desalination, 126(1-3), 33-39.
- Alatiqi I.M., Ghabris A.H., and Ebrahim S. (1989). System identification and control of reverse ósmosis desalination. Desalination, 75(1-3), 119-140.
- Al-haj Ali, Ajbar A., Ali E. and Alhumaizi K. (2009). Modeling the transient behavior of an experimental reverse osmosis tubular membrane. Desalination, 245(1-3), pp.194-204.
- Al-haj Ali, Ajbar A., Ali E. and Alhumaizi K. (2010). Robust model-based control of a tubular reverse-osmosis desalination unit. Desalination, 255(1-3), pp.129-136.
- Alexiadis A., Wiley D., Vishnoi A., Lee R., Fletcher D. and Bao J. (2007). CFD modelling of reverse ósmosis membrane flow and validation with experimental results. Desalination, 217(1-3), 242-250.
- Armijo y Condorhuamán (2012). Simulación dinámica de sistemas de ósmosis inversa. Rev. Per. Quím. Ing. Quím. Vol. 15 N.º 1 , 2012. Págs. 21-37.
- Assef J.Z., Watters J.C., Desphande P.B., and Alatiqi I.M. (1995). Advanced control of a reverse ósmosis desalination unit. In: Proceedings of International Desalination Association (IDA) World Congress, 5, 174-188, Abu Dhabi.
- Assef J.Z., Watters J.C., Deshpande P.B., and Alatiqi I.M. (1997). Advanced control of a reverse ósmosis desalination unit. Journal of Process Control, 7(4), 283-289.
- Ball (2015). Physical Chemistry. Cengage Learning, Stamford.
- Bartman A., McFall C., Christofides P. and Cohen Y. (2009). Model-predictive control of feed flow reversal in a reverse osmosis desalination process. Journal of Process Control, 19(3), pp.433-442.

- Becker S. (1991). “Unsupervised learning procedures for neural networks,” International Journal of Neural Systems, vol. 2, pp. 17–33.
- Belkacem A. (2008). Mathematical modeling of reverse osmosis process by the orthogonal collocation on finite element method. Asian Journal of Applied Sciences, pp.1-18.
- Bequette B. (2003). Process control. Upper Saddle River, N.J.: Prentice Hall PTR.
- Bhattacharyya D, and Willians M.E. (1992c). Separation of hazardous organics by low pressure reverse ósmosis membrane phase II. Final Report, EPA Report, EPA/600/2-91/045.
- Brian P. (1966). "Mass Transport in Reverse Osmosis", in Desalination by Reverse Osmosis, U. Merten, ed., pp. 161-202, MIT Press, Cambridge, MA.
- Burden A.C., Deshpande P.B., and Watters J.C. (2001). Advanced control of a B-9 Permasep permeator desalination pilot plant. Desalination, 133, 271-283.
- Burghoff H., Lee H., and Pusch W. (1980). Characterization of transport across cellulose acetate membrane in the presence of strong solute-membrane interactions. Journal of Applied Polymers Science, 25, 323.
- Burghoff H., Lee H., and Pusch W. (1988). Characterization of transport across cellulose acetate fibre reverse ósmosis system. Desalination, 68, 11-28.
- Calderon-Valdez S.N., Feliu-Batlle V., and Rivas-Perez R. (2015). Fractional-order mathematical model of an irrigation main canal pool. Spanish Journal of Agricultural Research, 13(3), e0212.
- Calderon Valdez S.N., Rivas-Perez R., Ruiz Torija M.A., Feliu-Batlle V. (2009). Fractional PI controller design with optimized robustness to time delay changes in main irrigation canals. 14th IEEE Conference on Emerging Technologies and Factory Automation, ETFA’2009, 1411-1417.
- Castillo-Garcia F.J., Feliu-Batlle V., and Rivas-Perez R. (2013a). Frequency specification regions of fractional order PI controller for first order plus time delay processes. Journal of Process Control, 23(4), 598-612.

- Castillo-Garcia F.J., Feliu-Batlle V., and Rivas-Perez R. (2013b). Time domain tuning of fractional order controllers combined with a Smith predictor for automation of water distribution in irrigation main channel pools. Asian Journal of Control, 15(3), 819-833.
- Castillo-Garcia F.J, Feliu-Batlle V., Rivas-Perez R., and Sanchez-Rodriguez L. (2011). Time domain tuning of a fractional order PIa controller combined with a Smith predictor for automation of water distribution in irrigation main channel pools. IFAC Proceedings Volumes (IFAC-PapersOnline), 18(Part 1), 15049-15054.
- Castillo Garcia F., Rivas-Perez R., Feliu Batlle V. (2008). Fractional IIalpha controller combined with a Smith predictor for effective water distribution in a main irrigation canal pool. IFAC Proceedings Volumes (IFAC-PapersOnline), 17(Part 1), 14558-14563.
- Chaaben A., Andoulsi R., Sellami A and Mhiri R. (2011). MIMO Modeling Approach for a Small Photovoltaic Reverse Osmosis Desalination System. INSAT, Tunis.
- Cipollina A., Micale G. and Rizzuti L. (2009). Seawater desalination. Heidelberg: Springer.
- Cirstea M. (2002). Neural and fuzzy logic control of drives and power systems. Oxford [England]: Newnes.
- Cotruvo J. (2010). Desalination technology. Boca Raton: CRC Press/Taylor & Francis.
- Cuenca-Tinitana J., Rivas-Perez R. (2012). Desarrollo de un sistema de control predictivo para la distribución de agua en un tramo de un canal principal de riego. Revista Estudios Universitarios Universidad Nacional de Loja, 119-130.
- Cybenko, G. (1989). Approximation by Superpositions of a Sigmoidal Function. Mathematics of Control, Signals, and Systems, 2(4), 303-314.
- Dhaifallah M. and Nisar K. (2014). Support Vector Machine Identification of Subspace Hammerstein Models. IJCTE, 7(1), pp.9-15.
- Elliott D. and Omidvar O. (1997). Neural Systems for Control. Academic Press.
- Franks R., Bartels C. and Nagghappan L. (2009). Performance of RO when Reclaiming High pH - High Temperature Wastewater. Hydranautics.

- Feliu-Batlle V., Rivas-Perez R., Castillo-Garcia F.J., Sanchez-Rodriguez L., and Linares Saez A. (2011). Robust fractional order controller for irrigation main canal pools with time-varying dynamical parameters. Computers and Electronics in Agriculture, 76(2), 205-217.
- Feliu Batlle V., Rivas-Perez R., Sanchez Rodriguez L., Castillo Garcia F., and Linarez Saez A. (2008). Robust fractional order PI controller for a main irrigation canal pool. IFAC Proceedings Volumes (IFAC-PapersOnline), 17(Part 1), 15535-15540.
- Feliu Batlle V., Rivas P.R., and Castillo F. (2005). Fractional robust control to delay changes in main irrigation canals. IFAC Proceedings Volumes (IFAC-PapersOnline), 16(Part 1), 28-33.
- Feliu Batlle V., Rivas-Perez R., Gorostiaga Canepa L., and Sanchez Rodriguez L. (2003). Fractional control for open irrigation canal. Proceedings of VI Inter-Regional Conference on Environment-Water. Land and Water Use Planning and Management - Envirowater 2003. Albacete, Spain.
- Flórez López R. and Fernández Fernández J. (2008). Las redes neuronales artificiales. Oleiros, La Coruña: Netbiblo.
- Fritzmann C., Lowenberg J., Wintgens T. and Melin, T. (2007). State-of-the-art of reverse ósmosis desalination. Desalination, 216(1-3), 1-76.
- Galushkin A. (2007). Neural networks theory. Berlin: Springer.
- Gambier A. and Badreddin E. (2011). A Robust Control Approach for a Reverse Osmosis Desalination Plant, Powered by Renewable Energy Sources. U Heidelberg CEST2011
- Gambier A., Wellenreuther A. and E. Badreddin E (2006). Optimal control of a reverse osmosis desalination plant using multi-objective optimization. Gambier. 2006. ICCA-IEEE.
- Gambier A., Krasnik A. and Badreddin E. (2007). Dynamic modeling of a simple reverse osmosis desalination plant for advanced control purposes. Proceedings of the 2007 American Control Conference. IEEE.
- García (2002). Aplicación de la ósmosis inversa y la nanofiltración en el acondicionamiento de aguas para calderas. UNIOVI.
- Haykin S. (2009). Neural networks and learning machines. New York: Prentice Hall/Pearson.

- Hernandez-Lopez Y., Rivas-Perez R., and Lorenzo Galván G. (2015). Robust proportional integral control of water level in an irrigation main canal. Revista Cubana de Ingeniería, 6(2), 25-34.
- Isasi P., Galván I. (2004). Redes neuronales artificiales. Un enfoque práctico, Pearson Prentice Hall, Madrid.
- Jiang A., Ding Q., Wang J., Jiangzhou S., Cheng W. and Xing C. (2014). Mathematical Modeling and Simulation of SWRO Process Based on Simultaneous Method. Journal of Applied Mathematics, Article ID 908569, 1-11.
- Johnson and Busch (2009). Engineering aspects of reverse ósmosis module design. Lenntech.
- Kedem O. and Katchalsky A. (1958). Thermodynamic analysis of the permeability of biological membranes to non-electrolytes. Biochimica et Biophysica Acta, 27, pp.229-246.
- Kim G., Park J., Kim J., Lee H. and Heo H. (2009). PID control of reverse osmosis desalination plant using immune-genetic algorithm. ICROS-SICE
- Kirk R., Othmer D. and Seidel A. (2007). Kirk-Othmer encyclopedia of chemical technology. Hoboken, NJ: Wiley-Interscience.
- Kovalenko P.I., Matzeliuk E.M., and Rivas-Perez R. (1990). Adaptive control of water distribution in main irrigation canals with variable time delay. Scientific Researches in Hydrotechnique and Land Reclamation, 32-40.
- Kovalenko P.I., Pichuguin E.D., and Rivas-Perez R. (1989). Compensation of nonlinear characteristics with dead-zone in water distribution control systems. Construction and Exploitation of Land Reclamation Systems, 37-49.
- Kucera J. (2014). Desalination. Hoboken, N.J.: John Wiley & Sons.
- Linares A., Rivas-Perez R., and Feliu V. (2008). New advanced management and control strategies applied to hydraulic Infrastructures for water transport and distribution. Abengoa, No December, 41-42.
- Lior N. (2013). Advances in water desalination. Hoboken, N.J.: Wiley
- Lonsdale H., Merten U., and Riley R. (1965). "Transport Properties of Cellulose Acetate Osmotic Membranes", Journal of Applied Polymer Science, 9, 1341.

- Madaeni S., Shiri M. and Kurdian A. (2015). Modeling, optimization, and control of reverse osmosis water treatment in kazeroon power plant using neural network. Chemical Engineering Communications, 202(1), pp.6-14.
- McFall C., Bartman A., Christofides P. and Cohen Y. (2008). Control and Monitoring of a High Recovery Reverse Osmosis Desalination Process. Industrial & Engineering Chemistry Research, 47(17), pp.6698-6710.
- Mason E.A., and Lonsdale H. (1990). Statistical-mechanical theory of membrane transport. Journal of Membrane Science 51(1-2), 1-81.
- Matsurra T., and Sourirajan S. (1981). Reverse ósmosis transport through capillary pores under the influence of surface forces. Industrial & Engineering Chemistry Process Design and Development, 20, 273. paginas
- Matthiasson E., and Sivik B. (1980). Concentration polarization and fouling. Desalination, 35, 59.
- Mazid M. (1984). Mechanisms of transport through reverse ósmosis membranes. Separation Science and Technology 19, 357.
- Mehdizadeh H. (1990). Modelling of transport phenomena in reverse ósmosis membranes. Dissertation. Department of Chemical Engineering, McMaster University, Hamilton, Ontario, Canada.
- Merten U. (1966), "Transport Properties of Osmotic Membranes", in Desalination by Reverse Osmosis, pp. 15-54, MIT Press, Cambridge, Mass.
- Mindler A. and Epstein A. (1986). Chapter 2.6 Measurements and control in reverse osmosis desalination. Desalination, 59, pp.343-379.
- Moncada (2012). Desarrollo de un Sistema de control predictivo de un bastidor de ósmosis inversa de una planta desalinizadora de agua de mar. Mg. PUCP
- Moncada-Valerio J., Rivas-Perez R., and Sotomayor Moriano J. (2012). Control predictivo multivariable de un bastidor de ósmosis inversa de una planta desalinizadora de agua de mar. Memorias del XV Congreso Latinoamericano de Control Automático “CLCA12”. Lima, Perú.
- Nakayama A., and Sano Y. (2013). An application of the Sano-Nakayama membrane transport model in hollow fiber reverse ósmosis desalination systems. Desalination, 311, 95-102.

- Paritosh F. and Murad S. (1996). Molecular simulations of ósmosis and reverse ósmosis in aqueous electrolyte solutions. AIChE J., 42(10), 2984-2986.
- Palacín L. (2014). Modelling, simulation and advanced control of small-scale reverse osmosis desalination plants. PhD. Universidad de Valladolid.
- Pedregal D.J., Rivas-Perez R., Feliu V., Sanchez L., and Linares A. (2009). A non-linear forecasting system for the Ebro River at Zaragoza, Spain. Environmental Modelling & Software, 24(4), 502-509.
- Phuntsho S., Vigneswaran S., Kandasamy J., Hong S., Lee, S. and Shon H. (2012). Influence of temperature and temperature difference in the performance of forward ósmosis desalination process. Journal of Membrane Science, 415-416, 734-744.
- Pusch W. (1977). Determination of transport parameters of synthetic membranes by hyper filtration experiments. Ber Bunsenges Phys Chem 81(9), 269.
- Pusch W. (1986). Measurement techniques of transport through membranes. Desalination, 59, 105.
- Qin J., Lay W. and Kekre K. (2012). Recent developments and future challenges of forward ósmosis for desalination: a review. Desalination and Water Treatment, 39(1-3), 123-136.
- Rabuñal J. and Dorado J. (2006). Artificial neural networks in real-life applications. Hershey PA: Idea Group Pub.
- Rangaiah G. and Kariwala V. (2012). Plantwide Control: Recent Developments and Applications. John Wiley & Sons.
- Rathore N., Kundariya N., Narain A. (2013). PID Controller Tuning in Reverse Osmosis System based on Particle Swarm Optimization. Rathore. IJSRP, Volume 3, Issue 6.
- Riverol C. and Pilipovik V. (2005). Mathematical Modeling of Perfect Decoupled Control System and Its Application: A Reverse Osmosis Desalination Industrial-Scale Unit. Journal of Automated Methods and Management in Chemistry, 2005(2), pp.50-54.
- Rivas-Pérez (1990). Automatic control of water distribution in irrigation systems. DSc. Ukrainian Academy of Agricultural Sciences.

- Rivas-Perez R., Feliu-Batlle V., Castillo-Garcia F.J., and Linares-Saez A. (2014a). Mathematical model for robust control of an irrigation main canal pool. Environmental Modelling & Software, 51(1), 207-220.
- Rivas-Perez R., Feliu-Batlle V., Castillo-Garcia F.J., Sanchez-Rodriguez L., and Linares Saez A. (2014b). Robust fractional order controller implemented in the first pool of the Imperial de Aragon main canal. Tecnología y Ciencias del Agua, 5(1), 23-42.
- Rivas-Perez, R., and Sotomayor-Moriano, J. (2014c). Control predictivo generalizado de las variables críticas de una unidad de ósmosis inversa. Memorias del XVI Congreso Latinoamericano de Control Automático, CLCA 2014, 642-647. Cancún, México.
- Rivas-Perez R., Castillo-Garcia F.J., Sotomayor-Moriano J., Feliu-Batlle V. (2014d). Robust fractional order control of steam pressure in steam drum of bagasse boiler. Revista Iberoamericana de Automática e Informática Industrial, 11(1), 20-31
- Rivas-Perez R., Feliu-Batlle V., Castillo-Garcia F.J., Benites Gonzalez O.I. (2014e). Temperature control of a crude oil preheating furnace using a modified Smith predictor improved with a disturbance rejection term. IFAC Proceedings Volumes (IFAC-PapersOnline), 19(PART 1), 5760-5765.
- Rivas-Perez R., Feliu-Batlle V., Castillo-Garcia F.J., Sanchez-Rodriguez L., and Linares-Saez A. (2011). Control oriented model of a complex irrigation main canal pool. IFAC Proceedings Volumes (IFAC-PapersOnline), 18(Part 1), 2919-2924.
- Rivas-Perez R., Feliu-Batlle V., Sanchez Rodriguez L., Pedregal Tercero D.J., Linares Saez A., Aguilar Mariñosa J.V., and Langarita Garcia P. (2008a). Identification of the first pool of the Imperial de Aragon main irrigation canal. Hydraulic Engineering in Mexico, 23(1), 71-87.
- Rivas-Perez R., Feliu Batlle V., Castillo Garcia F., and Linarez Saez A. (2008b). System identification for control of a main irrigation canal pool. IFAC Proceedings Volumes (IFAC-PapersOnline), 17(Part 1), 9649-9654.
- Rivas-Perez R., Peran Gonzalez J.R., Pineda Reyes B., and Perez Pereira S. (2003). Distributed control under centralized intelligent supervision in the Güira de Melena irrigation system. Hydraulic Engineering in Mexico, 18(2), 53-68.

- Rivas-Perez R., Prada Moraga C., Peran Gonzalez J.R., and Kovalenko P.I. (2002). Robust adaptive predictive control of water distribution in irrigation canals. IFAC Proceedings Volumes (IFAC-PapersOnline), 15(Part 1), 97-102.
- Rivas-Perez R., Aref Ghraizi R., Peran Gonzalez J.R., and Cesar Sanchez E. (2000). Industrial boilers. Integral automatic control system. Automática e Instrumentación, 308, 79-84.
- Rivas-Perez R., Herranz J., Llanes O., and Cartaya L. (1994). Modelo matemático dinámico de generadores de vapor. Revista de Ingeniería Electrónica, Automática y Comunicaciones, 15(3), 45-54.
- Rivas-Perez R. (1990). Automatic control of water distribution in irrigation systems. D.Sc thesis, All Russia Research Institute of Hydraulic Engineering and Land Reclamation A.N. Kostyakov (NIIG&M), Moscow, Russia.
- Rivas-Perez R., Cao T.G., Franco Parellada C., Prokofiev V.E. (1987). Automatic control system of time delay plants. Control, Cibernética y Automatización, 20(1), 24-26.
- Riverol C, Pilipovik V (2005). Mathematical modeling of perfect decoupled control system and its application: a reverse ósmosis desalination industrial-scale unit. J Autom Methods Manag Chem, 2, 50-54.
- Robertson M.W., Watters J.C., Desphande P.B., Assef J.Z., and Alatiqi I.M. (1996). Model based control for reverse ósmosis desalination processes. Desalination, 104(1–2), 59–68.
- Rumelhart D. and McClelland J. (1986). Parallel distributed processing. Cambridge, Mass.: MIT Press.
- Sarmiento (2006). Diseño e implementación de un controlador basado en redes neuronales con entrenamiento rápido para sistemas de control 2x2. Mg. Universidad del Norte.
- Sabín D., González A. (2010). Aplicación cliente-servidor MatLab®-WinCC® empleando comunicación OPC. RIELAC, Vol 31, No(1) 2010, pag 1-10.
- Saengrung A., Abtahi A. and Zilouchian A. (2007). Neural network model for a commercial PEM fuel cell system. Journal of Power Sources, 172(2), pp.749-759.

- Senthil K.R., Kumaraswamy S., and Mani A. (2006). Experimental investigation on a two phase jet pump used in desalination systems. Desalination 204(1–3), 437-447.
- Sherwood T., Brian P., and Fisher R. (1967). "Desalination by Reverse Osmosis", Industrial and Engineering Chemistry Fundamentals, 6, 2
- Smith C., Corripio A. (1997). Principles and practice of automatic process control. John Wiley & Sons, New York.
- Sobana S. and Panda R. (2011). Review on modelling and control of desalination system using reverse ósmosis. Reviews in Environmental Science and Bio/Technology, 10(2), 139-150.
- Sobana and Panda (2013). Identification, modeling and control of continuous desalination process using reverse osmosis. PhD. Anna University.
- Soltaniesh M., and Gill W.N. (1981). Review of reverse ósmosis membranes and transport model. Chem Eng Commun, 12, 279.
- Sourirajan S. (1970). Reverse ósmosis. Academic Press, New York.
- Souriraraja S., and Matsuara T. (1985). Reverse ósmosis/ultra filtration principles. National Research Council of Canada, Ottawa.
- Spiegler K.S., and Kedem O. (1966). Thermodynamics of hyper filtration reverse ósmosis: criteria for efficient membranes. Desalination, 1, 311.
- Sundaramoorthy S., Srinivasan G. and Murthy D. (2011). An analytical model for spiral wound reverse ósmosis membrane modules: Part II: Experimental validation. Desalination, 277(1-3), 257-264.
- Sundaramoorthy S., Srinivasan G. and Murthy D. (2011). An analytical model for spiral wound reverse ósmosis membrane modules: Part I:" Model development and parameter estimation. Desalination, 277(1-3), 230-256.
- Te Braake, H. and Van Straten, G. (1995). Random activation weight neural net (RAWN) for east non-iterative training. Engineering Applications of Artificial Intelligence, 8(1), pp.71-80.
- UNICEF (2015). Progress on Sanitation and Drinking Water; 2015 Update and MDG Assessment. WHO
- Voutchkov, N. (2013). Desalination engineering. New York, McGraw-Hill.
- Wang, L. (2008). Membrane and desalination technologies. New York, Humana Press.

- Watkins K., Carvajal L., Coppard D., Fuentes R., Ghosh A., Giamberar-dini C., Johansson C., Seck P., Ugaz C and Yaqub S. (2006). Beyond scarcity: power, poverty and the global water crisis. Human Development Report 2006. UNDP
- Wiley D. and Fletcher D. (2002). Computational fluid dynamics modelling of flow and permeation for pressure-driven membrane processes. Desalination, 145(1-3), pp.183-186.
- Williams (2013). A review of reverse osmosis theory. EET Corporation and Williams Engineering Services Company, Inc.
- Wilf (2004). Water Desalination (Kirk-Othmer encyclopedia of chemical technology). Hoboken, NJ: Wiley-Interscience.
- Wilf M. and Bartels C. (2005). Optimization of seawater RO systems design. Desalination, 173(1), pp.1-12.
- Wilf M., Awerbuch L., Bartels C., Mickley M., Pearce G., and Voutchkov N. (2007). The Guidebook To Membrane Desalination Technology: Reverse Osmosis, Nanofiltration And Hybrid Systems Process Design, Applications And Economics. Rehovot: Balaban Publishers.
- Zamarreño (2001). Entorno de comunicaciones OPC para EcosimPro. 1ª Reunión de Usuarios de EcosimPro, UNED.
- Zheng L. and Nakagawa H. (2002). OPC (OLE for process control) specification and its developments. SICE 2002. Volume 2, pag 917 - 920.
- Zilouchian A. and Jafar M. (2001). Automation and process control of reverse osmosis plants using soft computing methodologies. Desalination, 135(1-3), pp.51-59.

ANEXO A

PROGRAMAS PARA LA ESTIMACIÓN DE PARÁMETROS Y VALIDACIÓN DEL MODELO NO LINEAL MULTIVARIABLE OBTENIDO

A.1. p02AlatiqiInititalConditionsBL.m

```
%Determinación del estado estacionario inicial
(condiciones %iniciales) del sistema.
%Este programa sirve también para calcular los valores
reales de %kA y kB mediante prueba y error. Los valores
reales de estos %parámetros serán aquellos que hacen que
la salida del modelo %sea igual a la salida de la planta
real (datos experimentales).
clc; close all; clear all;

%%Datos
%Corriente de alimentación
Ff=315.45; %g/s (5 gpm)
Pf=(900+14.7)/14.50373; %bar
Tf=25.0;  %°C
Cf=30000; %ppm
pHf=6.45;
Cdf=C2Cd(Cf); %uS/cm
dPb=0; %bar
dPp=0; %bar
%permeado
Pp=1.01325; %bar
%membrana
kA=2.4500*10^-3; %m/s
kB=1.3605*10^-2; %g/(s*m2*bar)
A=152; %m2
k=kB/kA; %bar-1
%soluto
Mmolar=58.44; %g/gmol
%Condiciones iniciales para solución del sistema de EDO's
mb0=43/50*1000; %g
mp0=35/50*1000; %g
Tb0=0; %°C
Tp0=0; %°C
Cb0=0; %ppm
Cp0=0; %ppm
%%Simulación del modelo no lineal
open('AlatiqiInitialConditionsBL');
simOut=sim('AlatiqiInitialConditionsBL');
```

A.2 p03AlatiqiBLComparacionPFC.m

```
%Comparación de los resultados del modelo no lineal con
los %valores experimentales de Alatiqi et al. (1989).
%Efecto de la presión de la alimentación sobre el flujo y
la %conductividad del permeado.
clc; close all; clear all; format short g;

%Se ejecuta el programa AlatiqiInititalConditionsBL.m para
%determinar el valor de todas las variables en el estado
%estacionario inicial y se asignan estos valores a sus
%variables %respectivas
p02AlatiqiInititalConditionsBL
%Condiciones iniciales de la corriente de alimentación
Ff0=Ff; %g/s
Pf0=Pf; %bar
Tf0=Tf; %°C
Cf0=Cf; %ppm
pHf0=pHf;
%Condiciones iniciales sistema
mb0=mb(end); %g
mp0=mp(end); %g
Fb0=Fb(end); %g/s
Fp0=Fp(end); %g/s
Cb0=Cb(end); %ppm
Cp0=Cp(end); %ppm
Tb0=Tb(end); %°C
Tp0=Tp(end); %°C
Pb0=Pb(end); %bar
Pp0=Pp(end); %bar

%Se simula el sistema en lazo abierto frente a los mismos
%cambios realizados en la referencia. Luego se presentan
los %resultados en forma gráfica.
open('AlatiqiBLROSimulationRVarP');
simOut=sim('AlatiqiBLROSimulationRVarP');
tsim=retentate.time;
Fretentate=retentate.signals(1,1).values;
Cretentate=retentate.signals(1,2).values;
Tretentate=retentate.signals(1,3).values;
Pretentate=retentate.signals(1,4).values;
% clear 'retentate'
Fpermeate=permeate.signals(1,1).values;
Cpermeate=permeate.signals(1,2).values;
Tpermeate=permeate.signals(1,3).values;
Ppermeate=permeate.signals(1,4).values;
% clear 'permeate'

%Se definen las variables para comparar gráficamente
%resultados.
Pf=Pfdata.signals.values; %bar
```

```
Fp=Fpermeate; %g/s
Condp=Kohlrausch(Cpermeate).*(1+0.0212*(Tpermeate-
25));%uS/cm,
%Se calculan las variables de desviación los resultados
de %simulación
Pfn=Pf-Pf(1);
Fpn=Fp-Fp(1);
Condpn=Condp-Condp(1);

%%Valores experimentales de la referencia.
tA=30*(0:45)';%s
PfA=[900 900 900 900 900 900 900 900 1000 1000 1000 1000
1000 900 900 900 900 900 900 900 900 900 900 900 900 900
900 900 900 900 800 800 700 700 900 900 900 900 900 900
900 900 900 900 900 900]';%psi
FpA=[1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.25 1.25
1.25 1.25 1.25 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05
1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 0.85 0.85
0.65 0.65 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05
1.05 1.05 1.05]';%gpm
CondpA=[435 438 437 437 437 436 436 437 401 403 401 402
404 436 435 436 437 435 436 437 435 438 436 437 436 435
437 436 435 436 486 484 558 562 435 436 435 436 437 436
437 435 436 436 436 437]';%uS/cm
disp('        [tiempo(s) presión(psi) flujo(gpm)
conductividad(uS/cm)]')
disp([tA PfA FpA CondpA])
%se escriben los datos experimentales en las unidades del
%presente trabajo
PfA=(PfA+14.7)/14.50373;
FpA=FpA/0.01585033;
%Se calculan las variables de desviación de los datos
%experimentales.
PfAn=PfA-PfA(1);
FpAn=FpA-FpA(1);
CondpAn=CondpA-CondpA(1);

%Se toman los valores de simulación del modelo
%correspondientes a los mismos tiempos que los datos
%disponibles en la referencia.
puntos=tA+1;
tcomp=tsim(puntos);
Fpcomp=Fp(puntos);
Condpcomp=Condp(puntos);
%Se calcula el error cuadrático medio (MSE) y la bondad
de %ajuste (Fit)
errorFp=Fpcomp-FpA;
MSEFp=sum(errorFp.^2)/(length(errorFp)-1)
fitFp=goodnessOfFit(Fpcomp,FpA,'NMSE');
fitFptext=strcat(num2str(round(fitFp*10000)/100),' %')
errorCondp=Condpcomp-CondpA;
```

```
MSECondp=sum(errorCondp.^2)/(length(errorCondp)-1)
fitCondp=goodnessOfFit(Condpcomp,CondpA,'NMSE');
fitCondp=0.7457;%este valor es obtenido con el programa
%AlatiqiBLValidacionPFC.m
fitCondptext=strcat(num2str(round(fitCondp*10000)/100),'
%')

%%Comparación de resultados
%gráficas en unidades reales
figure(1);
plot(tA,FpA,'o','LineWidth',2);
hold all;
plot(tsim,Fp,'LineWidth',2);
xlabel('tiempo, s');
ylabel('Flujo, ml/s');
% title('Flujo de permeado ante cambios escalón en la
presión %de la alimentación');
legend('Experimental','Modelo');
text(1050,72.5,strcat('Fit = ',fitFptext));
grid on;
figure(2);
plot(tA,CondpA,'LineWidth',2);
hold all;
plot(tsim,Condp,'LineWidth',2);
xlabel('tiempo, s');
ylabel('Conductividad, uS/cm');
% title('Conductividad del permeado ante cambios escalón
en la %presión de la alimentación');
legend('Experimental','Modelo');
text(1050,590,strcat('Fit = ',fitCondptext));
grid on;
```

A.3 p04AlatiqiBLComparacionpHC

```
%Comparación de los resultados del modelo no lineal con
los %valores experimentales de Alatiqi et al. (1989).
Efecto del %pH %de la alimentación sobre la conductividad
del permeado.
clc; close all; clear all; format short g;

%Se ejecuta el programa AlatiqiInititalConditionsBL.m
p02AlatiqiInititalConditionsBL
%Condiciones iniciales de la corriente de alimentación
Ff0=Ff; %g/s
Pf0=Pf; %bar
Tf0=Tf; %°C
Cf0=Cf; %ppm
pHf0=pHf;
%Condiciones iniciales sistema
mb0=mb(end); %g
mp0=mp(end); %g
Fb0=Fb(end); %g/s
Fp0=Fp(end); %g/s
Cb0=Cb(end); %ppm
Cp0=Cp(end); %ppm
Tb0=Tb(end); %°C
Tp0=Tp(end); %°C
Pb0=Pb(end); %bar
Pp0=Pp(end); %bar

%Se simula el sistema en lazo abierto frente a los mismos
%cambios realizados en la referencia.
open('AlatiqiBLROSimulationRVarpH');
simOut=sim('AlatiqiBLROSimulationRVarpH');
tsim=retentate.time;
Fretentate=retentate.signals(1,1).values;
Cretentate=retentate.signals(1,2).values;
Tretentate=retentate.signals(1,3).values;
Pretentate=retentate.signals(1,4).values;
% clear 'retentate'
Fpermeate=permeate.signals(1,1).values;
Cpermeate=permeate.signals(1,2).values;
Tpermeate=permeate.signals(1,3).values;
Ppermeate=permeate.signals(1,4).values;
% clear 'permeate'

%Se definen las variables para comparar gráficamente
%resultados.
pHf=pHfdata.signals.values;
Condp=Kohlrausch(Cpermeate).*(1+0.0212*(Tpermeate-25));
%Se calculan las variables de desviación los resultados
de %simulación
pHfn=pHf-pHf(1);
```

```
Condpn=Condp-Condp(20*60);
%%Valores experimentales de la referencia.
tA=60*(0:39)';%s
pHfA=[6.45 6.45 6.45 6.45 6.45 6.45 6.45 6.45 6.45 6.66
6.66 6.66 6.66 6.66 6.77 6.77 6.77 6.77 6.77 6.77 6.77
6.77 6.77 6.77 6.77 7.08 7.08 7.08 7.08 7.08 7.08 7.08
7.08 7.08 7.08 7.08 7.08 7.08 7.08 7.08]';%-log([H+])
FpA=[1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05
1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05
1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05
1.05 1.05 1.05 1.05 1.05 1.05 1.05 1.05]';%gpm
CondpA=[442 442 443 440 441 441 441 440 441 441 440 425
426 426 426 425 426 420 419 419 418 419 418 418 419 419
418 404 405 404 405 404 405 404 404 403 404 404 403
404]';%uS/cm
disp('        [tiempo(s)     pH      flujo(gpm)
conductividad(uS/cm)]')
disp([tA pHfA FpA CondpA])
%se escriben los datos experimentales en las unidades del
%presente trabajo
FpA=FpA/0.01585033;
%Se calculan las variables de desviación con los valores
de la %referencia.
pHfAn=pHfA-pHfA(1);
CondpAn=CondpA-CondpA(20);

%Se toman los valores de simulación del modelo en los
mismos %tiempos que los datos disponibles en la
referencia.
puntos=tA+1;
tcomp=tsim(puntos);
pHfcomp=pHf(puntos);
%Se calcula el error cuadrático medio (MSE) y la bondad
de %ajuste (Fit)
errorpHf=pHfcomp-pHfA;
MSEpHf=sum(errorpHf.^2)/(length(errorpHf)-1)
fitpHf=goodnessOfFit(pHfcomp,pHfA,'NRMSE');
fitpHptext=strcat(num2str(round(fitpHf*10000)/100),' %')
%%Comparación de resultados
%gráficas en unidades reales
figure(1);
plot(tA,CondpA,'o','LineWidth',2);
hold all;
plot(tsim,Condp,'LineWidth',2);
xlabel('tiempo, s');
ylabel('Conductividad, uS/cm');
% title('Conductividad del permeado ante cambios escalón
en el pH de la alimentación');
legend('Experimental','Modelo');
text(1810,437,strcat('Fit = ',fitpHptext));
grid on;
```

ANEXO B

PROGRAMAS PARA LA IDENTIFICACIÓN EN EL DOMINIO DEL TIEMPO Y LA VALIDACIÓN DEL MODELO LINEAL OBTENIDO

B.1 p01ROGeneracionDatosPFC.m

```
%Simulación en lazo abierto y obtención de datos para la
%identificación del sistema en el dominio del tiempo.
Efecto de %la presión de la alimentación sobre el flujo y
la conductividad %del permeado
clc; close all; clear all;

%%Determinación del Estado Estacionario Inicial
disp('---Determinación del Estado Estacionario Inicial---
');
%Se ejecuta el programa AlatiqiInititalConditionsBL.m para
determinar las variables en el estado estacionario inicial
AlatiqiInititalConditionsBL
%%Generación de datos a partir del Estado Estacionario
Inicial
disp('-Efecto de la presión sobre el flujo y
conductividad-');
%Condiciones iniciales de la corriente de alimentación
Ff0=Ff; %g/s
Pf0=Pf; %bar
Tf0=Tf; %°C
Cf0=Cf; %ppm
pHf0=pHf;
%Condiciones iniciales sistema
kA0=kA(end); %g/(s*m2)
mb0=mb(end); %g
mp0=mp(end); %g
Fb0=Fb(end); %g/s
Fp0=Fp(end); %g/s
Cb0=Cb(end); %ppm
Cp0=Cp(end); %ppm
Tb0=Tb(end); %°C
Tp0=Tp(end); %°C
Pb0=Pb(end); %bar
Pp0=Pp(end); %bar
Condp0=Kohlrausch(Cp0);%uS/cm
%simulación en lazo abierto
open('ROIdentificationRVarP');
simOut=sim('ROIdentificationRVarP');
tsim=permeate.time;
Fpermeate=permeate.signals(1,1).values;
Cpermeate=permeate.signals(1,2).values;
Tpermeate=permeate.signals(1,3).values;
Ppermeate=permeate.signals(1,4).values;
%cálculos
Pfeed=Pretentate;
Condretentate=C2Cd(Cretentate);
Condpermeate=Kohlrausch(Cpermeate);
save datosPFC tsim Pfeed Fpermeate Condpermeate
```

B.2 p02ROGeneracionDatospHC.m

```
%Simulación en lazo abierto y obtención de datos para la
%identificación del sistema en el dominio del tiempo.
Efecto %del pH de la alimentación sobre la conductividad
del %permeado.
clc; close all; clear all;

%%Determinación del Estado Estacionario Inicial
disp('--Determinación del Estado Estacionario Inicial--');
%Se ejecuta el programa AlatiqiInititalConditionsBL.m para
%determinar el valor de todas las variables en el estado
%estacionario inicial.
AlatiqiInititalConditionsBL

%%Generación de datos a partir del Estado Estacionario
Inicial
disp('-----Efecto del pH sobre la conductividad-----');
%Condiciones iniciales de la corriente de alimentación
Ff0=Ff; %g/s
Pf0=Pf; %bar
Tf0=Tf; %°C
Cf0=Cf; %ppm
pHf0=pHf;
%Condiciones iniciales sistema
kA0=kA(end); %g/(s*m2)
mb0=mb(end); %g
mp0=mp(end); %g
Fb0=Fb(end); %g/s
Fp0=Fp(end); %g/s
Cb0=Cb(end); %ppm
Cp0=Cp(end); %ppm
Tb0=Tb(end); %°C
Tp0=Tp(end); %°C
Pb0=Pb(end); %bar
Pp0=Pp(end); %bar
Condp0=Kohlrausch(Cp0);%uS/cm
%simulación en lazo abierto
open('ROIdentificationRVarpH');
simOut=sim('ROIdentificationRVarpH');
tsim=permeate.time;
Fpermeate=permeate.signals(1,1).values;
Cpermeate=permeate.signals(1,2).values;
Tpermeate=permeate.signals(1,3).values;
Ppermeate=permeate.signals(1,4).values;
clear 'permeate'
%cálculos
pHfeed=pHfdata.signals(1,1).values;
Condretentate=C2Cd(Cretentate);
Condpermeate=Kohlrausch(Cpermeate);
save datospHC tsim pHfeed Condpermeate
```

B.3 p03PreparacionDatosIdenPF.m

```
%Se calculan los valores de las variables de desviación a
%utilizar en la identificación no paramétrica en el dominio
%del tiempo. La identificación se realiza utilizando el
System %Identification Toolbox de Matlab.
close all; clear all; clc;
%Se cargan los datos de la simulación del modelo de la
planta %almacenados en el archivo datosPFC.mat
load('datosPFC.mat');
time=tsim;
input=Pfeed;
output=Fpermeate;
%Estos datos de entrada y salida corresponden a la
respuesta del sistema a una entrada escalón del 10% en la
presión de la corriente de alimentación.
figure(1)
subplot(2,1,1)
plot(time,input,'LineWidth',2)
ylabel('Presión de la alimentación, bar');
grid on;
subplot(2,1,2)
plot(time,output,'LineWidth',2)
xlabel('Tiempo, s');
ylabel('Flujo de permeado, g/s');
grid on;

%Se determinan los valores de las variables de desviación
dtime=time;
dtime=dtime-dtime(30);
dtime(1:29)=[];
dinput=input;
dinput=dinput-dinput(30);
dinput(1:29)=[];
doutput=output;
doutput=doutput-doutput(30);
doutput(1:29)=[];
%Se utilizan las variables de desviación para la
identificación del sistema en el dominio del tiempo con el
System Identification Toolbox de MATLAB.
ident
%En el se importan los datos de entrada (dinput) y salida
(doutput) con un starting time = 0 y sampling inteval = 1.
Luego se ajustan a un modelo de segundo orden, sin ceros,
ni tiempo muerto, que se exportan al Workspace de Matlab
con el nombre MPF2orden, y finalmente, se guarda todo en
forma %manual el archivo PFident.mat usando la línea de
código:
save PFident
```

B.4 p04PreparacionDatosIdenPC.m

```
%Se calculan los valores de las variables de desviación a
%utilizar en la identificación no paramétrica en el
dominio %del tiempo. La identificación se realiza
utilizando el System %Identification Toolbox de Matlab.
close all; clear all; clc;
%Se cargan los datos de la simulación del modelo de la
planta %almacenados en el archivo datosPFC.mat
load('datosPFC.mat');
time=tsim;
input=Pfeed;
output=Condpermeate;
%Estos datos de entrada y salida corresponden a la
respuesta %del sistema a una entrada escalón del 10% en
la presión de la %corriente de alimentación
figure(1)
subplot(2,1,1)
plot(time,input,'LineWidth',2)
ylabel('Presión de la alimentación, bar');
grid on;
subplot(2,1,2)
plot(time,output,'LineWidth',2)
xlabel('Tiempo, s');
ylabel('Conductividad del permeado, uS/cm');
grid on;

%Se determinan los valores de las variables de desviación
dtime=time;
dtime=dtime-dtime(30);
dtime(1:29)=[];
dinput=input;
dinput=dinput-dinput(30);
dinput(1:29)=[];
doutput=output;
doutput=doutput-doutput(30);
doutput(1:29)=[];
%Se utilizan las variables de desviación para la
%identificación del sistema en el dominio del tiempo con
el %System Identification Toolbox de MATLAB.
ident
%En el se importan los datos de entrada (dinput) y salida
%(doutput) con un starting time = 0 y sampling inteval =
1. %Luego se ajustan a un modelo de segundo orden, sin
ceros, ni %tiempo muerto, que se exportan al Workspace de
Matlab con el %nombre MPC2orden, y finalmente, se guarda
todo en forma
%manual el archivo PCident.mat usando la línea de código:
save PCident
```

B.5 p05PreparacionDatosIdenpHC.m

```
%Se calculan los valores de las variables de desviación a
%utilizar en la identificación no paramétrica en el
dominio %del tiempo. La identificación se realiza
utilizando el System %Identification Toolbox de Matlab.
close all; clear all; clc;
%Se cargan los datos de la simulación del modelo de la
planta %almacenados en el archivo datosPFC.mat
load('datospHC.mat');
time=tsim;
input=pHfeed;
output=Condpermeate;
%Estos datos de entrada y salida corresponden a la
respuesta %del sistema a una entrada escalón del 10% en
la presión de la %corriente de alimentación
figure(1)
subplot(2,1,1)
plot(time,input,'LineWidth',2)
ylabel('pH de la alimentación, bar');
grid on;
subplot(2,1,2)
plot(time,output,'LineWidth',2)
xlabel('Tiempo, s');
ylabel('Conductivad del permeado, uS/cm');
grid on;

%Se determinan los valores de las variables de desviación
dtime=time;
dtime=dtime-dtime(1);
% dtime(1:1)=[];
dinput=input;
dinput=dinput-dinput(1);
% dinput(1:1)=[];
doutput=output;
doutput=doutput-doutput(1);
% doutput(1:1)=[];

%Se utilizan las variables de desviación para la
%identificación del sistema en el dominio del tiempo con
el %System Identification Toolbox de MATLAB.
ident
%En el se importan los datos de entrada (dinput) y salida
%(doutput) con un starting time = 0 y sampling inteval =
1. %Luego se ajustan a un modelo de segundo orden, sin
ceros, ni %tiempo muerto, que se exportan al Workspace de
Matlab con el %nombre MpHC2orden, y finalmente, se guarda
todo en forma %manual el archivo PFident.mat usando la
línea de código:
save pHCident
```

B.6 p06PFidentSim.m

```
%Se discretiza y lleva al espacio de estados el modelo
obtenido y también se realiza la simulación en lazo
abierto.
close all; clear all; clc;

load('PFident.mat');
%modelo de segundo orden con dos polos, sin ceros ni
tiempo muerto
Kp = MPF2orden.Kp; Tp1 = MPF2orden.Tp1; Tp2 =
MPF2orden.Tp2; Td = 0; Tz = 0; %Td = MPF2orden.Td; Tz =
MPF2orden.Tz;

%Modelo lineal continuo del sistema en el dominio de s
PFc1=tf([Kp*Tz Kp],[Tp1*Tp2 Tp1+Tp2 1],'iodelay',Td)
[num den]=tfdata(PFc1,'v')
PFb1=num(2)/den(1);%para que la ecuación característica
sea un polinómio mónico
PFb0=num(3)/den(1);
PFa1=den(2)/den(1);
PFa0=den(3)/den(1);
save PFGs PFb0 PFa0 PFa1
%Modelo lineal continuo en el espacio de estados (primera
forma, desarrollo propio)
%X=[x1 x2]'; x1p=x2
Ac1=[0 1; -PFa0 -PFa1]
Bc1=[0; PFb0]
Cc1=[1 0]
Dc1=[0]
PFc2=ss(Ac1,Bc1,Cc1,Dc1)
save PFsss2 PFc2

%Se discretiza el modelo
Ts=1;%1/100000*round(min(10000*[Tp1 Tp2]));%Ts= 1/10 de
la menor constante de tiempo
%Modelo lineal discreto del sistema en el dominio de z
PFz=c2d(PFc1,Ts,'zoh')
[numd dend]=tfdata(PFz,'v')
PFdb1=numd(2)/dend(1);%para que la ecuación
característica sea un polinómio mónico
PFdb0=numd(3)/dend(1);
PFda1=dend(2)/dend(1);
PFda0=dend(3)/dend(1);
%Modelo lineal discreto en el espacio de estados (primera
forma, a partir de la f.t. z)
Ad1=[-PFda1 1; -PFda0 0]
Bd1=[PFdb1; PFdb0]
Cd1=[1 0]
Dd1=[0]
PFd1=ss(Ad1,Bd1,Cd1,Dd1,Ts)
```

```
%respuesta al escalón del sistema continuo
U=dinput;
yGs=step(U(end)*PFc1,dtime);
figure(1);
subplot(2,1,1);
plot(dtime,doutput+Fpermeate(1),'o','LineWidth',2);
hold all;
plot(dtime,yGs+Fpermeate(1),'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Flujo del permeado, ml/s');
legend('Modelo no lineal','Modelo lineal');
grid on;
subplot(2,1,2);
plot(dtime,dinput+Pfeed(1),'LineWidth',2);
xlabel('Tiempo,s');
ylabel('Presión de entrada, bar');
% axis([0 600 60 90]);
grid on;

%Comparación de las respuestas de los modelos discretos
%se presenta el modelo continuo y no lineal para
comparación
figure(2);
plot(dtime,doutput+Fpermeate(1),'o','LineWidth',2);
hold all;
plot(dtime,yGs+Fpermeate(1),'LineWidth',2);
xlabel('Tiempo,s');
ylabel('Flujo del permeado, g/s');
grid on;
%respuesta al escalón del modelo discreto en z
t=dtime(1):Ts:dtime(end);
U=dinput;
yGz=step(U(end)*PFz,t);
plot(t,yGz+Fpermeate(1),'d','LineWidth',2);
%respuesta al escalón de los modelos discretos en el
espacio de estados
%primera forma
yGz=step(U(end)*PFd1,t);
plot(t,yGz+Fpermeate(1),'--','LineWidth',2);
%segunda forma
x = [ 0;  0];
k = 1;
u = [0 linspace(U(end),U(end),length(t)-1)];
nn=length(t);
for n = 1:nn
    x1(k,1) = x(1,1);
    x2(k,1) = x(2,1);
    x = (Ad2)*x + Bd2*u(k);
    k = k + 1;
end
plot(t,x1+Fpermeate(1),'s','LineWidth',2);
```

B.7 p07PCidentSim.m

```
%Se discretiza y lleva al espacio de estados el modelo
obtenido y también se realiza la simulación en lazo
abierto.
close all; clear all; clc;

load('PCident.mat');
%modelo de segundo orden con dos polos, sin ceros ni
tiempo muerto
Kp = MPC2orden.Kp; Tp1 = MPC2orden.Tp1; Tp2 =
MPC2orden.Tp2; Td = 0; Tz = 0; %Td = MPC2orden.Td; Tz =
MPC2orden.Tz;

%Modelo lineal continuo del sistema en el dominio de s
PCc1=tf([Kp*Tz Kp],[Tp1*Tp2 Tp1+Tp2 1],'iodelay',Td)
[num den]=tfdata(PCc1,'v')
PCb1=num(2)/den(1);%para que la ecuación característica
sea un polinómio mónico
PCb0=num(3)/den(1);
PCa1=den(2)/den(1);
PCa0=den(3)/den(1);
save PCGs PCb0 PCa0 PCa1
%Modelo lineal continuo en el espacio de estados (primera
forma, desarrollo propio)
%X=[x1 x2]'; x1p=x2
Ac1=[0 1; -PCa0 -PCa1]
Bc1=[0; PCb0]
Cc1=[1 0]
Dc1=[0]
PCc2=ss(Ac1,Bc1,Cc1,Dc1)
save PCsss2 PCc2

%Se discretiza el modelo
Ts=1;%1/10000*round(min(1000*[Tp1 Tp2]));%Ts= 1/10 de la
menor constante de tiempo
%Modelo lineal discreto del sistema en el dominio de z
PCz=c2d(PCc1,Ts,'zoh')
[numd dend]=tfdata(PCz,'v')
PCdb1=numd(2)/dend(1);%para que la ecuación
característica sea un polinómio mónico
PCdb0=numd(3)/dend(1);
PCda1=dend(2)/dend(1);
PCda0=dend(3)/dend(1);
%Modelo lineal discreto en el espacio de estados (primera
forma, a partir de la f.t. z)
Ad1=[-PCda1 1; -PCda0 0]
Bd1=[PCdb1; PCdb0]
Cd1=[1 0]
Dd1=[0]
PCd1=ss(Ad1,Bd1,Cd1,Dd1,Ts)
```

```
%respuesta al escalón del sistema continuo
U=dinput;
yGs=step(U(end)*PCc1,dtime);
figure(1);
subplot(2,1,1);
plot(dtime,doutput+Condpermeate(1),'o','LineWidth',2);
hold all;
plot(dtime,yGs+Condpermeate(1),'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Conductividad del permeado, uS/cm');
legend('Modelo no lineal','Modelo lineal');
grid on;
subplot(2,1,2);
plot(dtime,dinput+Pfeed(1),'LineWidth',2);
xlabel('Tiempo,s');
ylabel('Presión de entrada, bar');
grid on;

%Comparación de las respuestas de los modelos discretos
%se presenta el modelo continuo y no lineal para
comparación
figure(2);
plot(dtime,doutput+Condpermeate(1),'o','LineWidth',2);
hold all;
plot(dtime,yGs+Condpermeate(1),'LineWidth',2);
xlabel('Tiempo,s');
ylabel('Conductividad del permeado, uS/cm');
grid on;
%respuesta al escalón del modelo discreto en el dominio
de z
t=dtime(1):Ts:dtime(end);
U=dinput;
yGz=step(U(end)*PCz,t);
plot(t,yGz+Condpermeate(1),'d','LineWidth',2);
%respuesta al escalón de los modelos discretos en el
espacio de estados
%primera forma
yGz=step(U(end)*PCd1,t);
plot(t,yGz+Condpermeate(1),'--','LineWidth',2);
%segunda forma
x = [ 0;  0];
k = 1;
u = [0 linspace(U(end),U(end),length(t)-1)];
nn=length(t);
for n = 1:nn
    x1(k,1) = x(1,1);
    x2(k,1) = x(2,1);
    x = (Ad2)*x + Bd2*u(k);
    k = k + 1;
end
plot(t,x1+Condpermeate(1),'s','LineWidth',2);
```

B.8 p08pHCidentSim.m

```
%Se discretiza y lleva al espacio de estados el modelo
obtenido y también se realiza la simulación en lazo
abierto.
close all; clear all; clc;

load('pHCident.mat');
%modelo de segundo orden con dos polos, sin ceros ni
tiempo muerto
Kp = MpHC2orden.Kp; Tp1 = MpHC2orden.Tp1; Tp2 =
MpHC2orden.Tp2; Td = 0; Tz = 0; %Td = MpHC2orden.Td; Tz =
MpHC2orden.Tz;

%Modelo lineal continuo del sistema en el dominio de s
pHCc1=tf([Kp*Tz Kp],[Tp1*Tp2 Tp1+Tp2 1],'iodelay',Td)
[num den]=tfdata(pHCc1,'v')
pHCb1=num(2)/den(1);%para que la ecuación característica
sea un polinómio mónico
pHCb0=num(3)/den(1);
pHCa1=den(2)/den(1);
pHCa0=den(3)/den(1);
save pHCGs pHCb0 pHCa0 pHCa1
%Modelo lineal continuo en el espacio de estados (primera
forma, desarrollo propio)
%X=[x1 x2]'; x1p=x2
Ac1=[0 1; -pHCa0 -pHCa1]
Bc1=[0; pHCb0]
Cc1=[1 0]
Dc1=[0]
pHCc2=ss(Ac1,Bc1,Cc1,Dc1)
save pHCsss2 pHCc2

%Se discretiza el modelo
Ts=1;%1/10000*round(min(1000*[Tp1 Tp2]));%Ts= 1/10 de la
menor constante de tiempo
%Modelo lineal discreto del sistema en el dominio de z
pHCz=c2d(pHCc1,Ts,'zoh')
[numd dend]=tfdata(pHCz,'v')
pHCdb1=numd(2)/dend(1);%para que la ecuación
característica sea un polinómio mónico
pHCdb0=numd(3)/dend(1);
pHCda1=dend(2)/dend(1);
pHCda0=dend(3)/dend(1);
%Modelo lineal discreto en el espacio de estados (primera
forma, a partir de la f.t. z)
Ad1=[-pHCda1 1; -pHCda0 0]
Bd1=[pHCdb1; pHCdb0]
Cd1=[1 0]
Dd1=[0]
pHCd1=ss(Ad1,Bd1,Cd1,Dd1,Ts)
```

```
%respuesta al escalón del sistema continuo
U=dinput;
yGs=step(U(end)*pHCc1,dtime);
figure(1);
subplot(2,1,1);
plot(dtime,doutput+Condpermeate(1),'o','LineWidth',2);
hold all;
plot(dtime,yGs+Condpermeate(1),'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Conductividad del permeado, uS/cm');
legend('Modelo no lineal','Modelo lineal');
grid on;
subplot(2,1,2);
plot(dtime,dinput+pHfeed(1),'LineWidth',2);
xlabel('Tiempo,s');
ylabel('pH de la alimentación');
% axis([0 600 60 90]);
grid on;

%Comparación de las respuestas de los modelos discretos
%se presenta el modelo continuo y no lineal para
comparación
figure(2);
plot(dtime,doutput+Condpermeate(1),'o','LineWidth',2);
hold all;
plot(dtime,yGs+Condpermeate(1),'LineWidth',2);
xlabel('Tiempo,s');
ylabel('Conductividad del permeado, uS/cm');
grid on;
%respuesta al escalón del modelo discreto en z
t=dtime(1):Ts:dtime(end);
U=dinput;
yGz=step(U(end)*pHCz,t);
plot(t,yGz+Condpermeate(1),'d','LineWidth',2);
%respuesta al escalón de los modelos discretos en el
espacio de estados
%primera forma
yGz=step(U(end)*pHCd1,t);
plot(t,yGz+Condpermeate(1),'--','LineWidth',2);
%segunda forma
x = [ 0;  0];
k = 1;
u = [0 linspace(U(end),U(end),length(t)-1)];
nn=length(t);
for n = 1:nn
    x1(k,1) = x(1,1);
    x2(k,1) = x(2,1);
    x = (Ad2)*x + Bd2*u(k);
    k = k + 1;
end
plot(t,x1+Condpermeate(1),'s','LineWidth',2);
```

ANEXO C

PROGRAMAS PARA EL DISEÑO DEL NEUROCONTROLADOR Y SU IMPLEMENTACIÓN EN SIMULINK

C.1 p05MultivariableNeuralController.m

```
%=========================================================
%Entrenamiento de un Neurocontrolador Dinámico para el
%control del Flujo y la Conductividad del Permeado en una
%Planta de Desalinización por O.I.
%Memoria de Tesis: "Modelado y Control Basado en Redes
%Neuronales Artificiales de una Planta Piloto de
%Desalinización de Agua de Mar por Ósmosis Inversa"
%=========================================================

%Se entrena el controlador neuronal para varias
condiciones %iniciales utilizando el algoritmo de
entrenamiento: Dynamic Back Propagation (DBP)
%=========================================================
clear all; close all; clc;
disp('ENTRENAMIENTO DEL CONTROLADOR NEURONAL
MULTIVARIABLE, ALGORITMO DBP')
disp(' ')

% PARA EL SISTEMA DISCRETO EN EL ESPACIO DE ESTADOS:
% x_(k+1) = Ak*x_k + Bk*u_k
% y_k = Ck*x_k
% los estados, las entradas y las salidas son:
% estados
% x1 = [FPP-FPPss]
% x2 = [FPPp],  x2ss = 0
% x3 = [CPP-CPPss]
% x4 = [CPPp],  x4ss = 0
% x5 = [CPpH-CPpHss]
% x6 = [CPpHp], x6ss = 0
% entradas
% u1 = [PA-PAss]
% u2 = [pHA-pHAss]
% salidas
% y1 = [FP-FPss]
% y2 = [CP-CPss]
% con:
% FP = Flujo de la corriente de permeado, ml/s
% CP = Conductividad de la corriente de permeado, uS/cm
% PA = Presión de la corriente de alimentación, bar
% pHA = pH de la corriente de alimentación, adimensional
% FPP: Flujo de permeado debido a la presión de la
alimentación
% FPPp: Tasa de cambio del Flujo de permeado debido a la
presión
% CPP: Conductividad del permeado debido a la presión de
la alimentación
% CPPp: Tasa de cambio de la Conductividad del permeado
debido a la presión
```

```
% CPpH: Conductividad del permeado debido al pH de la
alimentación
% CPpHp: Tasa de cambio Conductividad del permeado debido
al pH
%-------------------------------------------------------
% Además, las matrices correspondientes son:
load('SS2system.mat');
Ts=1;
[Ak Bk]=c2d(A,B,Ts)
Ck = C
Dk = D

%Para entrenar el controlador neuronal multivariable se
utilizan las mismas condiciones de operación (inicial y
final) usadas para evaluar el desempeño de los
controladores PID. Es decir, se partirá de la condición:
%entradas
PAss  = 63.06654;%bar
pHAss = 6.45;
%salidas
FPss = 66.264;%ml/s
CPss = 439.06;%uS/cm

%Por comodidad, para el proceso: las variables de entrada
se llamarán control,las de salida se llamarán respuesta,
y las variables con las que se definen los estados se
llamarán variables_e. En estado estacionario, todas estas
%variables tienen los valores:
control1ss = PAss;
control2ss = pHAss;
controlss  = [control1ss; control2ss]
respuesta1ss = FPss;
respuesta2ss = CPss;
respuestass  = [respuesta1ss; respuesta2ss]
variable_e1ss = respuesta1ss;
variable_e2ss = 0;
variable_e3ss = respuesta2ss;%32 es la contribución del
pH a la conductividad
variable_e4ss = 0;           %desde pH 7 hasta 6.35
variable_e5ss = 0;
variable_e6ss = 0;
variables_ess = [variable_e1ss; variable_e2ss;
variable_e3ss; variable_e4ss; variable_e5ss;
variable_e6ss]
%observe que, en el modelo no lineal obtenido, en
correspondencia con la referencia (Alatiqi et al., 1989)
se considera que la contribución a la conductividad del
pH, a pH=6.45, es cero.
%Observe además que, independientemente de la elección
del estado estacionario inicial, todas las variables de
```

```
desviación en ese punto son cero, es decir, las entradas,
los estados y las salidas del modelo son cero.
u1ss = 0;
u2ss = 0;
uss  =[u1ss; u2ss];
y1ss = 0;
y2ss = 0;
yss  =[y1ss; y2ss];
x1ss = 0;
x2ss = 0;
x3ss = 0;
x4ss = 0;
x5ss = 0;
x6ss = 0;
xss  = [x1ss; x2ss; x3ss; x4ss; x5ss; x6ss];
%Se llegará a la condición en la que, el flujo de
permeado sea el 105% de su valor inicial manteniendose la
conductividad en el mismo valor inicial.

%Lo anterior significa que el estado final deseado
resulta de resolver el sistema de ecuaciones:
%y_k = Ck*x_k, con  y = [y1; y2], x = [x1; x2;...;x6]
%x_k = Ak*x_k + Bk*u_k, con u = [u1; u2]
%La primera ecuación matricial permite obtener:
%0.05*respuesta1 = x1d;
%               0 = x3d + x5d;
%y la segunda se puede escribir de forma conveniente
como:
%x_k = inv(eye(6)-Ak)*Bk*u_k
%al definir AA = inv(eye(6)-Ak)*Bk, se encuentra:
%x1d = AA(1,1)*u1d + AA(1,2)*u2d
%x2d = AA(2,1)*u1d + AA(2,2)*u2d
%x3d = AA(3,1)*u1d + AA(3,2)*u2d
%x4d = AA(4,1)*u1d + AA(4,2)*u2d
%x5d = AA(5,1)*u1d + AA(5,2)*u2d
%x6d = AA(6,1)*u1d + AA(6,2)*u2d
%sistema de ecuaciones que se reduce aún más:
%x1d = AA(1,1)*u1d
%x2d = AA(2,1)*u1d
%x3d = AA(3,1)*u1d
%x4d = AA(4,1)*u1d
%x5d = AA(5,2)*u2d
%x6d = AA(6,2)*u2d
%con la información disponible, se resuelve y obtiene:
AA = inv(eye(6)-Ak)*Bk
y1d = 0.05*respuesta1ss
x1d = y1d
u1d = 1/AA(1,1)*x1d
x2d = AA(2,1)*u1d
x3d = AA(3,1)*u1d
x4d = AA(4,1)*u1d
```

```
y2d = 0
x5d = -x3d
u2d = 1/AA(5,2)*x5d
x6d = AA(6,2)*u2d
%A partir de los valores anteriores, los estados deseados
son:
xd  = [x1d; x2d; x3d; x4d; x5d; x6d]
%y los valores deseados para las señales de control y
salidas "y":
ud  = [u1d; u2d]
yd  = [y1d; y2d]

%El conjunto de condiciones iniciales utilizadas para el
entrenamiento es:
cond_ini = [xss];

% Se realiza el escalamiento de las señales de entrada y
salida a la red neuronal al rango de [- 1, 1] para
facilitar el entrenamiento.
% Para las variables de entrada a la red neuronal se
considera lo siguiente:
% x1 - x1d, Flujo de permeado - Flujo deseado de permeado
= +- 10 ml/s alrededor del punto de operación
% x2 - x2d, Tasa de cambio del delta de flujo de permeado
+- 0.5 ml/s2 alrededor del punto de operación
% x3 - x3d, Conductividad del permeado debida a la
presión - Conductividad deseada del permeado debida a la
presión = +- 50 uS/cm alrededor del punto de operación
% x4 - x4d, Tasa de cambio del delta de Conductividad del
permeado debida a la presión +- 5 uS/cm/s alrededor del
punto de operación
% x5 - x5d, Conductividad del permeado debida al pH -
Conductividad deseada del permeado debida al pH = +- 50
uS/cm alrededor del punto de operación
% x6 - x6d, Tasa de cambio del delta de Conductividad del
permeado debida al pH +- 5 uS/cm/s alrededor del punto de
operación
% Para la variable de salida de la red neuronal se
considera lo siguiente:
% u1 - u1d, Presión de la alimentación - Presión deseada
de la alimentación = +- 5 bar alrededor del punto de
operación
% u - ud, pH de la alimentación - pH deseado de la
alimentación = +- 0.5ME alrededor del punto de operación
% Haciendo las correspondencias correctas, se tendrán los
factores de escalamiento m siguientes:
% Para x1:
me1=(1 - (-1))/(10 - (-10));
% Para x2:
me2=(1 - (-1))/(0.5 - (-0.5));
% Para x3:
```

```
me3=(1 - (-1))/(50 - (-50));
% Para x4:
me4=(1 - (-1))/(5 - (-5));
% Para x5:
me5=(1 - (-1))/(50 - (-50));
% Para x6:
me6=(1 - (-1))/(5 - (-5));
% Para u1:
ms1=(5 - (-5))/(1 - (-1));
% Para u2:
ms2=(0.5 - (-0.5))/(1 - (-1));
me=[me1;me2;me3;me4;me5;me6]
ms=[ms1;ms2]
%y las matrices de transformación
ME=diag(me)
MS=diag(ms)
save M ME MS

%definidos estos factores de escalamiento, se calculan
los valores escalados de todas las variables y se asignan
a las mismas variables (esto con el fin de no usar mas
variables en el programa)

Ak=ME*Ak*inv(ME)
Bk=ME*Bk*MS
Ck=Ck*inv(ME)

xss=ME*xss
uss=inv(MS)*uss
yss=Ck*xss

xd=ME*xd
ud=inv(MS)*ud
yd=Ck*xd

cond_ini=ME*cond_ini

ne = 6; %número de neuronas en la capa de entrada
nm = 3; %número de neuronas en la capa intermedia
ns = 2; %número de neuronas en la capa de salida

%Posibilidad de cargar los pesos guardados en
entrenamientos previos o usar valores iniciales
aleatorios
resp = menu('Desea utilizar los pesos guardados en
entrenamientos previos','SI','NO');
if resp == 1
load pesosMVnm3.mat
[ne_old,nm_old] = size(v);
if nm_old ~= nm
```

```
disp('Usted no puede utilizar los pesos guardados porque
estos corresponden a un número diferente de neuronas en
la capa intermedia. El número de neuronas con el que se
entrenó previamente la red es: ');
nm_old
break
end
else
v = 0.025*randn(ne,nm); %valores iniciales aleatorios
para los pesos v
w = 0.025*randn(nm,ns); %valores iniciales aleatorios
para los pesos w
c = zeros(nm,1); %centros de las funciones de activación
a = ones(nm,1);  %inclinaciones de las funciones de
activación
end

eta  = 0.0005;%ratio de aprendizaje de los pesos v y w
etac = 0.0000;%ratio de aprendizaje de los centros c
(ceros)
etaa = 0.0001;%ratio de aprendizaje de la inclinación a

ndata = 50; %número de veces que se integra la ecuación
de estado con cada combinación de pesos
n_max_iter_tot = 5000;%número máximo de iteraciones
totales a realizar, número de actualizaciones de pesos
%esto significa que n_max_iter_tot/ndata será el número
de veces que actualizan los pesos
error_tol = 0.01;%tolerancia para el error, una vez que
el error sea menor que error_tol, se aceptan los pesos y
se detiene el entrenamiento

r = [0 0 0 0 0 0]';
x_xd_buscado = ones(ndata,ne)*diag(r);%diferencia buscada
entre x y xd
dt = Ts; t=0;
niter = 1;%inicialización del contador del número de
iteraciones realizadas en el entrenamiento
n_ini = 1;%número de condiciones iniciales usadas para el
entrenamiento
error_max_per = 10000000;%se inicializa el error máximo
posible en un valor suficientemente grande
error = error_max_per;
while( (error > error_tol) & (niter < n_max_iter_tot) )
    for c_ini=1:n_ini
    ersum2 = zeros(ne,1);
    dx1dw_t = zeros(nm,ns);
    dx2dw_t = zeros(nm,ns);
    dx3dw_t = zeros(nm,ns);
    dx4dw_t = zeros(nm,ns);
    dx5dw_t = zeros(nm,ns);
```

```
    dx6dw_t = zeros(nm,ns);
    dx1dv_t = zeros(ne,nm);
    dx2dv_t = zeros(ne,nm);
    dx3dv_t = zeros(ne,nm);
    dx4dv_t = zeros(ne,nm);
    dx5dv_t = zeros(ne,nm);
    dx6dv_t = zeros(ne,nm);
    dx1dc_t = zeros(nm,1);
    dx2dc_t = zeros(nm,1);
    dx3dc_t = zeros(nm,1);
    dx4dc_t = zeros(nm,1);
    dx5dc_t = zeros(nm,1);
    dx6dc_t = zeros(nm,1);
    dx1da_t = zeros(nm,1);
    dx2da_t = zeros(nm,1);
    dx3da_t = zeros(nm,1);
    dx4da_t = zeros(nm,1);
    dx5da_t = zeros(nm,1);
    dx6da_t = zeros(nm,1);
    dJdw_t = zeros(nm,ns);
    dJdv_t = zeros(ne,nm);
    dJdc_t = zeros(nm,1);
    dJda_t = zeros(nm,1);
    %Condiciones inciales
    x = cond_ini(:,c_ini);
    t = 0;
        for k = 1:ndata-1
        %Entrada a la red neuronal, valores escalados
        in_red = x-xd;
        %Cálculos internos de la red neuronal
        m = v'*in_red;
        n = 2.0./(1 + exp(-(m-c)./a)) - 1;%función de
        activación sigmoidea 2
        out_red = w'*n;
        %Señal de control, desescalamiento y saturación
        u = out_red + ud;
        ureal = MS*u;
        if ureal(1) >  5; ureal(1) =  5; end
        if ureal(1) < -5; ureal(1) = -5; end
        if ureal(2) >  0.5; ureal(2) =  0.5; end
        if ureal(2) < -0.5; ureal(2) = -0.5; end
        u = inv(MS)*(ureal);
        control(k,:,c_ini) = u';%en variables de
desviación
        escaladas
        estado(k,:,c_ini) = x';%en variables de
desviación
        escaladas
        y = Ck*(x);%en variables de desviación reales
        salida(k,:,c_ini) = y';
        tsim(k,:,c_ini)=t;
```

```
        t=t+dt;
        %Cálculo de derivadas para la actualización de
los
        pesos de la red neuronal
        % Jacobiano = dX(k+1)/dX(k)
        jacob = Ak;
        % dxdu = dX(k+1)/dU(k)
        dxdu  = Bk;
        x = Ak*(x) + Bk*(u);
        % dU(k)/dW
        dndm = diag((1 - n.*n)./(2*a));
        dudx = w'*dndm*v';
        dudw_s = n;
        dudv_s1 = in_red*w(:,1)'*dndm;% s1 = salida 1
        dudv_s2 = in_red*w(:,2)'*dndm;
        dndc = ((n.*n-1)./(2.0.*a));
        dudc_s = [dndc.* w(:,1) dndc.* w(:,2)];
        dnda = ((n.*n-1).*(m-c)./(2*a.*a));
        duda_s = [dnda.* w(:,1) dnda.* w(:,2)];
        jacob_t = dxdu*dudx + jacob;
        %Actualizacion de derivadas en la Red Neuronal
        dx1dw_t = [dxdu(1,1).*dudw_s dxdu(1,2).*dudw_s] +
jacob_t(1,1).*dx1dw_t + jacob_t(1,2).*dx2dw_t +
jacob_t(1,3).*dx3dw_t + jacob_t(1,4).*dx4dw_t +
jacob_t(1,5).*dx5dw_t + jacob_t(1,6).*dx6dw_t;
        dx2dw_t = [dxdu(2,1).*dudw_s dxdu(2,2).*dudw_s] +
jacob_t(2,1).*dx1dw_t + jacob_t(2,2).*dx2dw_t +
jacob_t(2,3).*dx3dw_t + jacob_t(2,4).*dx4dw_t +
jacob_t(2,5).*dx5dw_t + jacob_t(2,6).*dx6dw_t;
        dx3dw_t = [dxdu(3,1).*dudw_s dxdu(3,2).*dudw_s] +
jacob_t(3,1).*dx1dw_t + jacob_t(3,2).*dx2dw_t +
jacob_t(3,3).*dx3dw_t + jacob_t(3,4).*dx4dw_t +
jacob_t(3,5).*dx5dw_t + jacob_t(3,6).*dx6dw_t;
        dx4dw_t = [dxdu(4,1).*dudw_s dxdu(4,2).*dudw_s] +
jacob_t(4,1).*dx1dw_t + jacob_t(4,2).*dx2dw_t +
jacob_t(4,3).*dx3dw_t + jacob_t(4,4).*dx4dw_t +
jacob_t(4,5).*dx5dw_t + jacob_t(4,6).*dx6dw_t;
        dx5dw_t = [dxdu(5,1).*dudw_s dxdu(5,2).*dudw_s] +
jacob_t(5,1).*dx1dw_t + jacob_t(5,2).*dx2dw_t +
jacob_t(5,3).*dx3dw_t + jacob_t(5,4).*dx4dw_t +
jacob_t(5,5).*dx5dw_t + jacob_t(5,6).*dx6dw_t;
        dx6dw_t = [dxdu(6,1).*dudw_s dxdu(6,2).*dudw_s] +
jacob_t(6,1).*dx1dw_t + jacob_t(6,2).*dx2dw_t +
jacob_t(6,3).*dx3dw_t + jacob_t(6,4).*dx4dw_t +
jacob_t(6,5).*dx5dw_t + jacob_t(6,6).*dx6dw_t;

        dx1dv_t = (dxdu(1,1).*dudv_s1 +
dxdu(1,2).*dudv_s2) + jacob_t(1,1).*dx1dv_t +
jacob_t(1,2).*dx2dv_t + jacob_t(1,3).*dx3dv_t +
jacob_t(1,4).*dx4dv_t + jacob_t(1,5).*dx5dv_t +
jacob_t(1,6).*dx6dv_t;
```

```
        dx2dv_t = (dxdu(2,1).*dudv_s1 +
dxdu(2,2).*dudv_s2) + jacob_t(2,1).*dx1dv_t +
jacob_t(2,2).*dx2dv_t + jacob_t(2,3).*dx3dv_t +
jacob_t(2,4).*dx4dv_t + jacob_t(2,5).*dx5dv_t +
jacob_t(2,6).*dx6dv_t;
        dx3dv_t = (dxdu(3,1).*dudv_s1 +
dxdu(3,2).*dudv_s2) + jacob_t(3,1).*dx1dv_t +
jacob_t(3,2).*dx2dv_t + jacob_t(3,3).*dx3dv_t +
jacob_t(3,4).*dx4dv_t + jacob_t(3,5).*dx5dv_t +
jacob_t(3,6).*dx6dv_t;
        dx4dv_t = (dxdu(4,1).*dudv_s1 +
dxdu(4,2).*dudv_s2) + jacob_t(4,1).*dx1dv_t +
jacob_t(4,2).*dx2dv_t + jacob_t(4,3).*dx3dv_t +
jacob_t(4,4).*dx4dv_t + jacob_t(4,5).*dx5dv_t +
jacob_t(4,6).*dx6dv_t;
        dx5dv_t = (dxdu(5,1).*dudv_s1 +
dxdu(5,2).*dudv_s2) + jacob_t(5,1).*dx1dv_t +
jacob_t(5,2).*dx2dv_t + jacob_t(5,3).*dx3dv_t +
jacob_t(5,4).*dx4dv_t + jacob_t(5,5).*dx5dv_t +
jacob_t(5,6).*dx6dv_t;
        dx6dv_t = (dxdu(6,1).*dudv_s1 +
dxdu(6,2).*dudv_s2) + jacob_t(6,1).*dx1dv_t +
jacob_t(6,2).*dx2dv_t + jacob_t(6,3).*dx3dv_t +
jacob_t(6,4).*dx4dv_t + jacob_t(6,5).*dx5dv_t +
jacob_t(6,6).*dx6dv_t;

        dx1dc_t = (dxdu(1,1).*dudc_s(:,1) +
dxdu(1,2).*dudc_s(:,2)) + jacob_t(1,1).*dx1dc_t +
jacob_t(1,2).*dx2dc_t + jacob_t(1,3).*dx3dc_t +
jacob_t(1,4).*dx4dc_t + jacob_t(1,5).*dx5dc_t +
jacob_t(1,6).*dx6dc_t;
        dx2dc_t = (dxdu(2,1).*dudc_s(:,1) +
dxdu(2,2).*dudc_s(:,2)) + jacob_t(2,1).*dx1dc_t +
jacob_t(2,2).*dx2dc_t + jacob_t(2,3).*dx3dc_t +
jacob_t(2,4).*dx4dc_t + jacob_t(2,5).*dx5dc_t +
jacob_t(2,6).*dx6dc_t;
        dx3dc_t = (dxdu(3,1).*dudc_s(:,1) +
dxdu(3,2).*dudc_s(:,2)) + jacob_t(3,1).*dx1dc_t +
jacob_t(3,2).*dx2dc_t + jacob_t(3,3).*dx3dc_t +
jacob_t(3,4).*dx4dc_t + jacob_t(3,5).*dx5dc_t +
jacob_t(3,6).*dx6dc_t;
        dx4dc_t = (dxdu(4,1).*dudc_s(:,1) +
dxdu(4,2).*dudc_s(:,2)) + jacob_t(4,1).*dx1dc_t +
jacob_t(4,2).*dx2dc_t + jacob_t(4,3).*dx3dc_t +
jacob_t(4,4).*dx4dc_t + jacob_t(4,5).*dx5dc_t +
jacob_t(4,6).*dx6dc_t;
        dx5dc_t = (dxdu(5,1).*dudc_s(:,1) +
dxdu(5,2).*dudc_s(:,2)) + jacob_t(5,1).*dx1dc_t +
jacob_t(5,2).*dx2dc_t + jacob_t(5,3).*dx3dc_t +
jacob_t(5,4).*dx4dc_t + jacob_t(5,5).*dx5dc_t +
jacob_t(5,6).*dx6dc_t;
```

```
        dx6dc_t = (dxdu(6,1).*dudc_s(:,1) +
dxdu(6,2).*dudc_s(:,2)) + jacob_t(6,1).*dx1dc_t +
jacob_t(6,2).*dx2dc_t + jacob_t(6,3).*dx3dc_t +
jacob_t(6,4).*dx4dc_t + jacob_t(6,5).*dx5dc_t +
jacob_t(6,6).*dx6dc_t;

        dx1da_t = (dxdu(1,1).*duda_s(:,1) +
dxdu(1,2).*duda_s(:,2)) + jacob_t(1,1).*dx1da_t +
jacob_t(1,2).*dx2da_t + jacob_t(1,3).*dx3da_t +
jacob_t(1,4).*dx4da_t + jacob_t(1,5).*dx5da_t +
jacob_t(1,6).*dx6da_t;
        dx2da_t = (dxdu(2,1).*duda_s(:,1) +
dxdu(2,2).*duda_s(:,2)) + jacob_t(2,1).*dx1da_t +
jacob_t(2,2).*dx2da_t + jacob_t(2,3).*dx3da_t +
jacob_t(2,4).*dx4da_t + jacob_t(2,5).*dx5da_t +
jacob_t(2,6).*dx6da_t;
        dx3da_t = (dxdu(3,1).*duda_s(:,1) +
dxdu(3,2).*duda_s(:,2)) + jacob_t(3,1).*dx1da_t +
jacob_t(3,2).*dx2da_t + jacob_t(3,3).*dx3da_t +
jacob_t(3,4).*dx4da_t + jacob_t(3,5).*dx5da_t +
jacob_t(3,6).*dx6da_t;
        dx4da_t = (dxdu(4,1).*duda_s(:,1) +
dxdu(4,2).*duda_s(:,2)) + jacob_t(4,1).*dx1da_t +
jacob_t(4,2).*dx2da_t + jacob_t(4,3).*dx3da_t +
jacob_t(4,4).*dx4da_t + jacob_t(4,5).*dx5da_t +
jacob_t(4,6).*dx6da_t;
        dx5da_t = (dxdu(5,1).*duda_s(:,1) +
dxdu(5,2).*duda_s(:,2)) + jacob_t(5,1).*dx1da_t +
jacob_t(5,2).*dx2da_t + jacob_t(5,3).*dx3da_t +
jacob_t(5,4).*dx4da_t + jacob_t(5,5).*dx5da_t +
jacob_t(5,6).*dx6da_t;
        dx6da_t = (dxdu(6,1).*duda_s(:,1) +
dxdu(6,2).*duda_s(:,2)) + jacob_t(6,1).*dx1da_t +
jacob_t(6,2).*dx2da_t + jacob_t(6,3).*dx3da_t +
jacob_t(6,4).*dx4da_t + jacob_t(6,5).*dx5da_t +
jacob_t(6,6).*dx6da_t;
        %Cálculo del error total acumulado
        out_des = x_xd_buscado(k+1,:);%diferencia entre x
y xd
        en cada instante
        out_des = out_des';
        er = (in_red - out_des);
        erJ = (in_red - out_des).^1;
        dJdw_t = dJdw_t + erJ(1,1).*dx1dw_t +
erJ(2,1).*dx2dw_t + erJ(3,1).*dx3dw_t + erJ(4,1).*dx4dw_t
+ erJ(5,1).*dx5dw_t + erJ(6,1).*dx6dw_t;
        dJdv_t = dJdv_t + erJ(1,1).*dx1dv_t +
erJ(2,1).*dx2dv_t + erJ(3,1).*dx3dv_t + erJ(4,1).*dx4dv_t
+ erJ(5,1).*dx5dv_t + erJ(6,1).*dx6dv_t;
```

```
        dJdc_t = dJdc_t + erJ(1,1).*dx1dc_t +
erJ(2,1).*dx2dc_t + erJ(3,1).*dx3dc_t + erJ(4,1).*dx4dc_t
+ erJ(5,1).*dx5dc_t + erJ(6,1).*dx6dc_t;
        dJda_t = dJda_t + erJ(1,1).*dx1da_t +
erJ(2,1).*dx2da_t + erJ(3,1).*dx3da_t + erJ(4,1).*dx4da_t
+ erJ(5,1).*dx5da_t + erJ(6,1).*dx6da_t;
        ersum2 = ersum2 + er.^2;
        end
    %Se calcula el promedio de todas las derivadas
    dJdw_t = dJdw_t/k;
    dJdv_t = dJdv_t/k;
    dJdc_t = dJdc_t/k;
    dJda_t = dJda_t/k;
    %Se almacenan las derivadas de cada condición inicial
en
    las variables
    %DJw, DJv, DJa, DJb
    DJw1(:,c_ini) = dJdw_t(:,1);
    DJw2(:,c_ini) = dJdw_t(:,2);
    DJv1(c_ini,:) = dJdv_t(1,:);
    DJv2(c_ini,:) = dJdv_t(2,:);
    DJv3(c_ini,:) = dJdv_t(3,:);
    DJv4(c_ini,:) = dJdv_t(4,:);
    DJv5(c_ini,:) = dJdv_t(5,:);
    DJv6(c_ini,:) = dJdv_t(6,:);
    DJa(:,c_ini) = dJda_t(:,1);
    DJc(:,c_ini) = dJdc_t(:,1);
    end
    %Se promedian las derivadas obtenidas para cada
condición
    inicial
    for aux=1:nm
    DJW1(aux,1) = sum(DJw1(aux,:))/n_ini;
    DJW2(aux,1) = sum(DJw2(aux,:))/n_ini;
    DJV1(1,aux) = sum(DJv1(:,aux))/n_ini;
    DJV2(1,aux) = sum(DJv2(:,aux))/n_ini;
    DJV3(1,aux) = sum(DJv3(:,aux))/n_ini;
    DJV4(1,aux) = sum(DJv4(:,aux))/n_ini;
    DJV5(1,aux) = sum(DJv5(:,aux))/n_ini;
    DJV6(1,aux) = sum(DJv6(:,aux))/n_ini;
    DJA(aux,1) = sum(DJa(aux,:))/n_ini;
    DJC(aux,1) = sum(DJc(aux,:))/n_ini;
    end
%Se actualizan los pesos
dw = [DJW1 DJW2];
dv = [DJV1; DJV2; DJV3; DJV4; DJV5; DJV6];
da = DJA;
dc = DJC;
w = w - eta*dw;
v = v - eta*dv;
c = c - etac*dc;
```

```
a = a - etaa*da;
error = sum(ersum2)
niter = niter + 1;
error_max_per = error;
end
%valores de los estados (variables de desviación) reales
estado=estado*inv(ME);
x1=estado(:,1);
x2=estado(:,2);
x3=estado(:,3);
x4=estado(:,4);
x5=estado(:,5);
x6=estado(:,6);
%señales de control (variables de desviación) reales
control=control*MS;
u1=control(:,1);
u2=control(:,2);
%señales de salida (variables de desviación) reales
y1=salida(:,1);
y2=salida(:,2);

Fpermeate = y1 + respuesta1ss;
SPFpermeate = [respuesta1ss
linspace(y1d+respuesta1ss,y1d+respuesta1ss,length(y1)-
1)];
Pfeed = u1 + control1ss;

Condp = y2 + respuesta2ss;
SPCondp = [respuesta2ss
linspace(y2d+respuesta2ss,y2d+respuesta2ss,length(y2)-
1)];
pHff = u2 + control2ss;

%variable controlada y variable manipulada
figure(1);
plot(tsim,Fpermeate,'LineWidth',2);
hold all;
plot(tsim,SPFpermeate,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Flujo de la corriente de permeado, ml/s');
legend('Variable controlada','Referencia');
axis([0 t 65 70])
grid on;

figure(2);
plot(tsim,Pfeed,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Presión de la corriente de alimentación, bar');
legend('Variable manipulada');
axis([0 t 62 66])
grid on;
```

```
figure(3);
plot(tsim,Condp,'LineWidth',2);
hold all;
plot(tsim,SPCondp,'--','LineWidth',2);
xlabel('Tiempo, s');
ylabel('Conductividad de la corriente de permeado,
uS/cm');
legend('Variable controlada','Referencia');
axis([0 t 400 500])
grid on;

figure(4);
plot(tsim,pHff,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('pH de la corriente de alimentación');
legend('Variable manipulada');
axis([0 t 6 7])
grid on;
```

C.2 p06MultivariableNeuralControllerValidation.m

```
%=========================================================
% Validación del entrenamiento de un Neurocontrolador
Dinámico para el Flujo y la Conductividad del Permeado en
una Planta de Desalinización por O.I.
% Memoria de Tesis: "Modelado y Control Basado en Redes
Neuronales Artificiales de una Planta Piloto de
Desalinización de Agua de Mar por Ósmosis Inversa"
%=========================================================
clear all; close all; clc;
disp('VALIDACIÓN DEL ENTRENAMIENTO DEL CONTROLADOR
NEURONAL MULTIVARIABLE')
disp(' ')

% PARA EL SISTEMA DISCRETO EN EL ESPACIO DE ESTADOS:
% x_(k+1) = Ak*x_k + Bk*u_k
% y_k = Ck*x_k
% los estados, las entradas y las salidas son:
% estados
% x1 = [FPP-FPPss]
% x2 = [FPPp],  x2ss = 0
% x3 = [CPP-CPPss]
% x4 = [CPPp],  x4ss = 0
% x5 = [CPpH-CPpHss]
% x6 = [CPpHp], x6ss = 0
% entradas
% u1 = [PA-PAss]
% u2 = [pHA-pHAss]
% salidas
% y1 = [FP-FPss]
% y2 = [CP-CPss]
% con:
% FP = Flujo de la corriente de permeado, ml/s
% CP = Conductividad de la corriente de permeado, uS/cm
% PA = Presión de la corriente de alimentación, bar
% pHA = pH de la corriente de alimentación, adimensional
% FPP: Flujo de permeado debido a la presión de la
alimentación
% FPPp: Tasa de cambio del Flujo de permeado debido a la
presión
% CPP: Conductividad del permeado debido a la presión de
la alimentación
% CPPp: Tasa de cambio de la Conductividad del permeado
debido a la presión
% CPpH: Conductividad del permeado debido al pH de la
alimentación
% CPpHp: Tasa de cambio Conductividad del permeado debido
al pH
%---------------------------------------------------------
```

```
% Además, las matrices correspondientes son:
load('SS2system.mat');
Ts=1;
[Ak Bk]=c2d(A,B,Ts)
Ck = C
Dk = D

%Se partirá de la condición:
%entradas
PAss  = 63.06654;%bar
pHAss = 6.45;
%salidas
FPss = 66.264;%ml/s
CPss = 439.06;%uS/cm

%En estado estacionario se tienen los valores:
control1ss = PAss;
control2ss = pHAss;
controlss  = [control1ss; control2ss]
respuesta1ss = FPss;
respuesta2ss = CPss;
respuestass  = [respuesta1ss; respuesta2ss]
variable_e1ss = respuesta1ss;
variable_e2ss = 0;
variable_e3ss = respuesta2ss;
variable_e4ss = 0;
variable_e5ss = 0;
variable_e6ss = 0;
variables_ess = [variable_e1ss; variable_e2ss;
variable_e3ss; variable_e4ss; variable_e5ss;
variable_e6ss]

controlss = pHAss;
respuestass = CPss;
variable_e1ss = respuestass;
variable_e2ss = 0;
variables_ess = [variable_e1ss; variable_e2ss]
%Como el estado inicial es estacionario, todas las
variables %de desviación en ese punto son cero, es decir,
las entradas, %los estados y las salidas del modelo son
cero.
u1ss = 0;
u2ss = 0;
uss  =[u1ss; u2ss];
y1ss = 0;
y2ss = 0;
yss  =[y1ss; y2ss];
x1ss = 0;
x2ss = 0;
x3ss = 0;
x4ss = 0;
```

```
x5ss = 0;
x6ss = 0;
xss  = [x1ss; x2ss; x3ss; x4ss; x5ss; x6ss];
%Se llegará a la condición en la que, la conductividad
del permeado sea el 105% de su valor inicial.

%Lo anterior significa que el estado final deseado
resulta de %resolver el sistema de ecuaciones:
%y_k = Ck*x_k, con  y = [y1; y2], x = [x1; x2;...;x6]
%x_k = Ak*x_k + Bk*u_k, con u = [u1; u2]
%La primera ecuación matricial permite obtener:
%0.05*respuesta1 = x1d;
%               0 = x3d + x5d;
%y la segunda se puede escribir de forma conveniente
como:
%x_k = inv(eye(6)-Ak)*Bk*u_k
%al definir AA = inv(eye(6)-Ak)*Bk, se encuentra:
%x1d = AA(1,1)*u1d + AA(1,2)*u2d
%x2d = AA(2,1)*u1d + AA(2,2)*u2d
%x3d = AA(3,1)*u1d + AA(3,2)*u2d
%x4d = AA(4,1)*u1d + AA(4,2)*u2d
%x5d = AA(5,1)*u1d + AA(5,2)*u2d
%x6d = AA(6,1)*u1d + AA(6,2)*u2d
%sistema de ecuaciones que se reduce aún más:
%x1d = AA(1,1)*u1d
%x2d = AA(2,1)*u1d
%x3d = AA(3,1)*u1d
%x4d = AA(4,1)*u1d
%x5d = AA(5,2)*u2d
%x6d = AA(6,2)*u2d
%con la información disponible, se resuelve y obtiene:
AA = inv(eye(6)-Ak)*Bk
y1d = 0.10*respuesta1ss
x1d = y1d
u1d = 1/AA(1,1)*x1d
x2d = AA(2,1)*u1d
x3d = AA(3,1)*u1d
x4d = AA(4,1)*u1d
y2d = 0
x5d = -x3d
u2d = 1/AA(5,2)*x5d
x6d = AA(6,2)*u2d
%A partir de los valores anteriores, los estados deseados
son:
xd  = [x1d; x2d; x3d; x4d; x5d; x6d]
%y los valores deseados para las señales de control y
salidas "y":
ud  = [u1d; u2d]
yd  = [y1d; y2d]
```

```
%El conjunto de condiciones iniciales utilizadas para la
validación es:
cond_ini = [xss];
load('M.mat')

Ak=ME*Ak*inv(ME)
Bk=ME*Bk*MS
Ck=Ck*inv(ME)

xss=ME*xss
uss=inv(MS)*uss
yss=Ck*xss

xd=ME*xd
ud=inv(MS)*ud
yd=Ck*xd

cond_ini=ME*cond_ini

load pesosMVnm3.mat
c_ini=1;
k=1;
dt = Ts; t=0;
x = cond_ini(:,c_ini);
u = uss;
y = yss;
while( ( k < 100 ) )
estado(k,:,c_ini) = x';%variables de desviación escaladas
control(k,:,c_ini)= u';%variables de desviación escaladas
salida(k,:,c_ini) = y';%en variables de desviación reales
tsim(k,:,c_ini)=t;
%Entrada a la red neuronal, valores escalados
in_red = x-xd;
%Cálculos internos de la red neuronal
m = v'*in_red;
n = 2.0./(1 + exp(-(m-c)./a)) - 1;%función de activación
out_red = w'*n;
%Señal de control, desescalamiento y saturación
u = out_red + ud;
ureal = MS*u;
    if ureal(1) >  5; ureal(1) =  5; end
    if ureal(1) < -5; ureal(1) = -5; end
    if ureal(2) >  0.5; ureal(2) =  0.5; end
    if ureal(2) < -0.5; ureal(2) = -0.5; end
u = inv(MS)*(ureal);
%Se calcula el nuevo estado
x = Ak*(x) + Bk*(u);
y = Ck*(x);
t=t+dt;
k=k+1;
end
```

```
%señales de control (variables de desviación) reales
control=control*MS;
u1=control(:,1);
u2=control(:,2);
%señales de salida (variables de desviación) reales
y1=salida(:,1);
y2=salida(:,2);

Fpermeate = y1 + respuesta1ss;
SPFpermeate = [respuesta1ss
linspace(y1d+respuesta1ss,y1d+respuesta1ss,length(y1)-
1)];
Pfeed = u1 + control1ss;

Condp = y2 + respuesta2ss;
SPCondp = [respuesta2ss
linspace(y2d+respuesta2ss,y2d+respuesta2ss,length(y2)-
1)];
pHff = u2 + control2ss;
t=t/1.25
%variable controlada y variable manipulada
figure(1);
plot(tsim,Fpermeate,'LineWidth',2);hold all;
plot(tsim,SPFpermeate,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Flujo de la corriente de permeado, ml/s');
legend('Variable controlada','Referencia');
axis([0 t 65 74]);grid on;

figure(2);
plot(tsim,Pfeed,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Presión de la corriente de alimentación, bar');
legend('Variable manipulada');
axis([0 t 62 68]);grid on;

figure(3);
plot(tsim,Condp,'LineWidth',2);hold all;
plot(tsim,SPCondp,'--','LineWidth',2);
xlabel('Tiempo, s');
ylabel('Conductividad de la corriente de permeado,
uS/cm');
legend('Variable controlada','Referencia');
axis([0 t 400 500]);grid on;

figure(4);
plot(tsim,pHff,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('pH de la corriente de alimentación');
legend('Variable manipulada');
axis([0 t 6 6.5]);grid on;
```

C.3 ControladorNeuronal.m

```
function u = fcn(x_xd)
ESCALAMIENTO=load('M.mat');
ME=ESCALAMIENTO.ME;
MS=ESCALAMIENTO.MS;
PESOS=load('pesosMVnm3.mat');
v=PESOS.v;
w=PESOS.w;
c=PESOS.c;
a=PESOS.a;
CONTROL_DESEADO=load('valores_deseados_no_linealp.mat');
ud=CONTROL_DESEADO.ud;
ud=[1 0;0 0.99]*ud;
% u2d=CONTROL_DESEADO.u2d;
in_red = ME*x_xd;
%Cálculos internos de la red neuronal
m = v'*in_red;
n = 2.0./(1 + exp(-(m-c)./a)) - 1;%función de activación
sigmoidea 2
out_red = w'*n;
%Señal de control, desescalamiento y saturación
u = out_red + MS\ud;
ureal = MS*u;
    if ureal(1) >  5; ureal(1) =  5; end
    if ureal(1) < -5; ureal(1) = -5; end
    if ureal(2) >  0.5; ureal(2) =  0.5; end
    if ureal(2) < -0.5; ureal(2) = -0.5; end
u = ureal;
u = u;
```

ANEXO D

PROGRAMAS PARA EL CONTROL NEURONAL DEL MODELO NO LINEAL DE LA UNIDAD DE ÓSMOSIS INVERSA DE UNA PLANTA PILOTO DE DESALINIZACIÓN DE AGUA DE MAR

D.1 p01controlNeuronalnoLineal.m

```
%Control neuronal del flujo y la conductividad del
permeado en %el modelo no lineal de una planta de
desalinización por O.I.
close all; clear all; clc;

%Determinación del Estado Estacionario Inicial
disp('-Determinación del Estado Estacionario Inicial-');
%Se ejecuta el programa InititalConditionsBL.m para
determinar %el valor de todas las variables en el estado
estacionario %inicial y se asignan estos valores a sus
variables %respectivas
InititalConditionsBL
%Condiciones iniciales de la corriente de alimentación
Ff0=Ff; %ml/s
Pf0=Pf; %bar
Tf0=Tf; %°C
Cf0=Cf; %ppm
pHf0=pHf;
%Condiciones iniciales sistema
kA0=kA(end); %ml/(s*m2)
mb0=mb(end); %ml
mp0=mp(end); %ml
Fb0=Fb(end); %ml/s
Fp0=Fp(end); %ml/s
Cb0=Cb(end); %ppm
Cp0=Cp(end); %ppm
Tb0=Tb(end); %°C
Tp0=Tp(end); %°C
Pb0=Pb(end); %bar
Pp0=Pp(end); %bar
Condp0=Kohlrausch(Cp0);%uS/cm

control10 = Pf0;
salida10 = Fp0;
control20 = pHf0;
salida20 = Condp0;

load('valores_deseados_no_linealp.mat');
x1d = xd(1);
x2d = xd(2);
x3d = xd(3);
x4d = xd(4);
x5d = xd(5);
y1d = yd(1);
y2d = yd(2);

Ts=1;%tiempo de integración
open('ROcontrolNeuronalnoLineal');
simOut=sim('ROcontrolNeuronalnoLineal');
```

```
%Se obtienen las variables de control para graficarlas
%variable de entrada
tsim=pHfdata.time;
PfeedNN=Pfdata.signals.values;
pHfeedNN=pHfdata.signals.values;

%variable de salida
FpermeateNN=Fpdata.signals.values(:,2);
CondpermeateNN=Condpdata.signals.values(:,1);
SPFpermeate = [salida10
linspace(y1d+salida10,y1d+salida10,length(FpermeateNN)-
1)];
SPCondpermeate = [salida20
linspace(y2d+salida20,y2d+salida20,length(CondpermeateNN)
-1)];
t=tsim(end)/2;

%gráficas en unidades reales
%Flujo de permeado
figure(1);
plot(tsim,SPFpermeate,'LineWidth',2);
hold all;
plot(tsim,FpermeateNN,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Flujo de la corriente de permeado, ml/s');
axis([0 t 66 70]);grid on;

figure(2);
plot(tsim,PfeedNN,'LineWidth',2);
hold all;
xlabel('Tiempo, s');
ylabel('Presión de la corriente de alimentación, bar');
legend('Variable manipulada');
axis([0 t 62 66]);grid on;
%Conductividad del permeado
figure(3);
plot(tsim,CondpermeateNN,'LineWidth',2);
hold all;
plot(tsim,SPCondpermeate,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Conductividad de la corriente de permeado,
uS/cm');
axis([0 t 430 450]);grid on;

figure(4);
plot(tsim,pHfeedNN,'LineWidth',2);
hold all;
xlabel('Tiempo, s');
ylabel('pH de la corriente de alimentación');
legend('Variable manipulada');
axis([0 t 6.1 6.4]);grid on;
```

ANEXO E

PROGRAMAS PARA LA SIMULACIÓN DEL SISTEMA DE CONTROL PID Y PARA LA COMPARACIÓN DE LOS CONTROLADORES PID vs. NEURONAL

E.1 p01ROFlowControl.m

```
%Control PID del Flujo de Permeado en una Planta de
Desaliniziación por O.I.
clc; close all; clear all;

%%Determinación del Estado Estacionario Inicial
disp('-Determinación del Estado Estacionario Inicial-');
%Se ejecuta el programa InititalConditionsBL.m para
determinar %el valor de todas las variables en el estado
estacionario %inicial y se asignan estos valores a sus
variables %respectivas
InititalConditionsBL

%%Control del Flujo de Permeado Manipulando la presión de
%Alimentación
disp('-----Control del Flujo de Permeado-----');
%Condiciones iniciales de la corriente de alimentación
Ff0=Ff; %g/s
Pf0=Pf; %bar
Tf0=Tf; %°C
Cf0=Cf; %ppm
pHf0=pHf;
%Condiciones iniciales sistema
kA0=kA(end); %g/(s*m2)
mb0=mb(end); %g
mp0=mp(end); %g
Fb0=Fb(end); %g/s
Fp0=Fp(end); %g/s
Cb0=Cb(end); %ppm
Cp0=Cp(end); %ppm
Tb0=Tb(end); %°C
Tp0=Tp(end); %°C
Pb0=Pb(end); %bar
Pp0=Pp(end); %bar
Condp0=Kohlrausch(Cp0);%uS/cm
%Simulación del sistema de control
open('ROPermeateFlowControl');
simOut=sim('ROPermeateFlowControl');
tsim=retentate.time;
Fretentate=retentate.signals(1,1).values;
Cretentate=retentate.signals(1,2).values;
Tretentate=retentate.signals(1,3).values;
Pretentate=retentate.signals(1,4).values;
clear 'retentate'
Fpermeate=permeate.signals(1,1).values;
Cpermeate=permeate.signals(1,2).values;
Tpermeate=permeate.signals(1,3).values;
Ppermeate=permeate.signals(1,4).values;
clear 'permeate'
SPFp=SPFpdata.signals.values;
```

```
%cálculos
Pfeed=Pretentate;
Condretentate=C2Cd(Cretentate).*(1+0.0212*(Tretentate-
25));
Condpermeate=Kohlrausch(Cpermeate).*(1+0.0212*(Tpermeate-
25));%incluye corrección por temperatura

%Variable controlada y variable manipulada
figure(11);
plot(tsim,Fpermeate,'LineWidth',2);
hold all;
plot(tsim,SPFp,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Flujo de la corriente de permeado, ml/s');
legend('Variable controlada','Referencia');
grid on;
figure(12);
plot(tsim,Pretentate,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Presión de la corriente de alimentación, bar');
legend('Señal de control');
grid on;

save FPIdata tsim Fpermeate SPFp Pretentate
```

E.2 p02ROConductivityControl.m

```
%Control PID de la Conductividad del Permeado en una
Planta de %Desaliniziación por O.I.
clc; close all; clear all;

%%Determinación del Estado Estacionario Inicial
disp('-Determinación del Estado Estacionario Inicial-');
%Se ejecuta el programa InititalConditionsBL.m
InititalConditionsBL

%%Control de la Conductividad del Permeado Manipulando el
pH %de la Alimentación
disp('--Control de la Conductividad del Permeado--');
%Condiciones iniciales de la corriente de alimentación
Ff0=Ff; %g/s
Pf0=Pf; %bar
Tf0=Tf; %°C
Cf0=Cf; %ppm
pHf0=pHf;
%Condiciones iniciales sistema
kA0=kA(end); %g/(s*m2)
mb0=mb(end); %g
mp0=mp(end); %g
Fb0=Fb(end); %g/s
Fp0=Fp(end); %g/s
Cb0=Cb(end); %ppm
Cp0=Cp(end); %ppm
Tb0=Tb(end); %°C
Tp0=Tp(end); %°C
Pb0=Pb(end); %bar
Pp0=Pp(end); %bar
Condp0=Kohlrausch(Cp0);%uS/cm
%simulación del sistema de control
open('ROPermeateConductivityControl');
simOut=sim('ROPermeateConductivityControl');
tsim=retentate.time;
Fretentate=retentate.signals(1,1).values;
Cretentate=retentate.signals(1,2).values;
Tretentate=retentate.signals(1,3).values;
Pretentate=retentate.signals(1,4).values;
clear 'retentate'
Fpermeate=permeate.signals(1,1).values;
Cpermeate=permeate.signals(1,2).values;
Tpermeate=permeate.signals(1,3).values;
Ppermeate=permeate.signals(1,4).values;
clear 'permeate'
SPCondp=SPCondpdata.signals.values;
Condp=Condpdata.signals.values;
pHff=pHfdata.signals.values;
```

```
%cálculos
Pfeed=Pretentate;
Condretentate=C2Cd(Cretentate).*(1+0.0212*(Tretentate-
25));
Condpermeate=Kohlrausch(Cpermeate).*(1+0.0212*(Tpermeate-
25));%incluye corrección por temperatura

%variable controlada y variable manipulada
figure(11);
plot(tsim,Condp,'LineWidth',2);
hold all;
plot(tsim,SPCondp,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Conductividad de la corriente de permeado,
uS/cm');
legend('Variable controlada','Referencia');
grid on;
figure(12);
plot(tsim,pHff,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('pH de la corriente de Alimentación');
legend('Señal de control');
grid on;

save CondPIdata tsim Condp SPCondp pHff
```

E.3 p01controlNeuronalvsPInoLineal.m

```
%Comparación de los controladores Neuronal y PI para el
flujo %y la conductividad del permeado en el modelo no
lineal de una %planta de desalinización por O.I.
close all; clear all; clc;

%Determinación del Estado Estacionario Inicial
disp('-Determinación del Estado Estacionario Inicial-');
%Se ejecuta el programa InititalConditionsBL.m
InititalConditionsBL
%Condiciones iniciales de la corriente de alimentación
Ff0=Ff; %ml/s
Pf0=Pf; %bar
Tf0=Tf; %°C
Cf0=Cf; %ppm
pHf0=pHf;
%Condiciones iniciales sistema
kA0=kA(end); %ml/(s*m2)
mb0=mb(end); %ml
mp0=mp(end); %ml
Fb0=Fb(end); %ml/s
Fp0=Fp(end); %ml/s
Cb0=Cb(end); %ppm
Cp0=Cp(end); %ppm
Tb0=Tb(end); %°C
Tp0=Tp(end); %°C
Pb0=Pb(end); %bar
Pp0=Pp(end); %bar
Condp0=Kohlrausch(Cp0);%uS/cm

control10 = Pf0;
salida10 = Fp0;
control20 = pHf0;
salida20 = Condp0;

load('valores_deseados_no_lineal.mat');
x1d = xd(1);x2d = xd(2);x3d = xd(3);x4d = xd(4);x5d =
xd(5);
y1d = yd(1);y2d = yd(2);

Ts=1;%tiempo de integración
open('ROcontrolNeuronalvsPInoLineal');
simOut=sim('ROcontrolNeuronalvsPInoLineal');
%Se obtienen las vaiables de control para graficarlas
%variable de entrada
tsim=Pfdata3.time;
PfeedPI=Pfdata3.signals.values(:,1);
PfeedNN=Pfdata3.signals.values(:,2);
pHfeedPI=pHfdata3.signals.values(:,1);
pHfeedNN=pHfdata3.signals.values(:,2);
```

```
%variable de salida
FpermeatePI=Fpdata3.signals.values(:,1);
FpermeateNN=Fpdata3.signals.values(:,2);
CondpermeatePI=Condpdata3.signals.values(:,1);
CondpermeateNN=Condpdata3.signals.values(:,2);
SPFpermeate = [salida10 linspace(y1d+salida10,
y1d+salida10,length(FpermeateNN)-1)];
SPCondpermeate = [salida20 linspace(y2d+salida20,
y2d+salida20,length(CondpermeateNN)-1)];

%gráficas en unidades reales
%Flujo de permeado
figure(1);
plot(tsim,FpermeatePI,'LineWidth',2);hold all;
plot(tsim,SPFpermeate,'LineWidth',2);
plot(tsim,FpermeateNN,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Flujo de la corriente de permeado, ml/s');
legend('Controlador PI','Controlador
Neuronal','Referencia');
axis([0 tsim(end) 65 70]);grid on;

figure(2);
plot(tsim,PfeedPI,'LineWidth',2);hold all;
plot(tsim,PfeedNN,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Presión de la corriente de alimentación, bar');
legend('Controlador PI','Controlador Neuronal');
axis([0 tsim(end) 62 66]);grid on;

%Conductividad del permeado
figure(3);
plot(tsim,CondpermeatePI,'LineWidth',2);hold all;
plot(tsim,CondpermeateNN,'LineWidth',2);
plot(tsim,SPCondpermeate,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('Conductividad de la corriente de permeado,
uS/cm');
legend('Controlador PI','Controlador
Neuronal','Referencia');
axis([0 tsim(end) 420 450]);grid on;

figure(4);
plot(tsim,pHfeedPI,'LineWidth',2);hold all;
plot(tsim,pHfeedNN,'LineWidth',2);
xlabel('Tiempo, s');
ylabel('pH de la corriente de alimentación');
legend('Controlador PI','Controlador Neuronal');
axis([0 tsim(end) 6.0 6.5]);
grid on;
```