# ENTRENAMIENTO INDIRECTO DE MODELOS DE CAJA GRIS UTILIZANDO LEVENBERG-MARQUARDT EN MATLAB®

Francisco Cruz Naranjo
Gonzalo Acuña Leiva
fcruz@diinf.usach.cl, gacuna@usach.cl

Departamento de Ingeniería Informática
Universidad de Santiago de Chile

## RESUMEN

Los modelos de caja gris son aquellos que mezclan ecuaciones diferenciales, las cuales actúan como caja blanca y redes neuronales, utilizadas como caja negra para complementar el modelo fenomenológico. Estos modelos han sido utilizados en diversas investigaciones probándose su eficacia. El presente trabajo tiene por objetivo mostrar una variación del algoritmo de Levenberg-Marquardt que incluye Matlab® en el toolbox de redes neuronales, para poder trabajar con modelos de caja gris, en su modalidad de aprendizaje indirecto. El modelo neuronal de caja gris fue probado en la simulación de un proceso químico de reactor continuo agitado (CSTR) respondiendo satisfactoriamente a la simulación.

Palabras Claves: Modelo de caja gris, modelo híbrido, redes neuronales, identificación de parámetros variantes en el tiempo.

## 1. INTRODUCCIÓN

Los procesos industriales son altamente relevantes para un país, en especial para su economía, es por esto que surge la necesidad de monitorear y optimizar tales procesos. Sin embargo al tratarse de proceso complejos, como aquellos en los que intervienen gran cantidad de variables de entrada y salida y que además son representados por modelos no lineales y con parámetros variables en el tiempo, surgen importantes impedimentos.

Cuando los procesos a modelar son complejos, la determinación de variables o parámetros relevantes para el mejoramiento de éstos es una tarea ardua y difícil, ya sea por la no existencia de instrumentos o bien por el costo extremadamente alto que tendría construirlos. Ante esto surge la necesidad de estimar las variables que no se pueden medir directamente. Para esto se requiere un sensor virtual, el cual observa y mide directamente otras variables del sistema, con lo cual puede estimar las variables que no pueden ser medidas en línea.

Un problema adicional constituye un modelo que posee parámetros variantes en el tiempo, ya que se debe adoptar una estrategia para identificar tales parámetros en línea y tiempo real. Una metodología utilizada en estos casos, especialmente en el ámbito de procesos químicos y biotecnológicos, es utilizar los denominados modelos de caja gris, inicialmente denominados modelos híbridos en [1]. Estos modelos son aquellos que incluyen un modelo fenomenológico limitado, el cual se complementa con parámetros obtenidos mediante redes neuronales.

El problema, entonces, consiste en desarrollar un modelo neuronal de caja gris utilizando Matlab®, el cual constituye una herramienta capaz de apoyar y facilitar la creación y posterior entrenamiento de estos modelos.

## 2. MODELOS DE CAJA GRIS

Los modelos neuronales de caja gris son utilizados para sistemas donde existe algo de conocimiento a priori, es decir, se conocen algunas leyes físicas, pero sin embargo, algunos parámetros se deben determinar a partir de datos observados.

Los modelos de caja gris, inicialmente denominados redes neuronales híbridas o modelos híbridos neuronales, han sido estudiados hace más de una década. Psichogios y Ungar [1] desarrollaron en 1992 fundamentos de una primera red neuronal híbrida para modelado de procesos. En su estudio realizaron una comparación entre redes neuronales estándar y redes neuronales híbridas resultando éstas últimos con mejor desempeño al predecir parámetros, debido a poseer conocimiento a priori y el correspondiente aprovechamiento de éste. Además probaron que los modelos híbridos pueden ser identificados y entrenados con un grupo bastante menor de datos, con respecto a un modelo equivalente de caja negra (redes neuronales).

Dos años más tarde, en 1994, Thompson y Kramer [2] clasificaron los modelos híbridos en dos categorías. La primera es el modelo de caja gris en serie (Figura 1 (a)), el cual consistente en una red neuronal que soporta valores intermedios, que luego son introducidos en el modelo fenomenológico. La segunda categoría es el modelo de caja gris en paralelo (Figura 1 (a)), el cual consistente en una red neuronal que compensa el modelo fenomenológico, en el sentido de modelar el error.

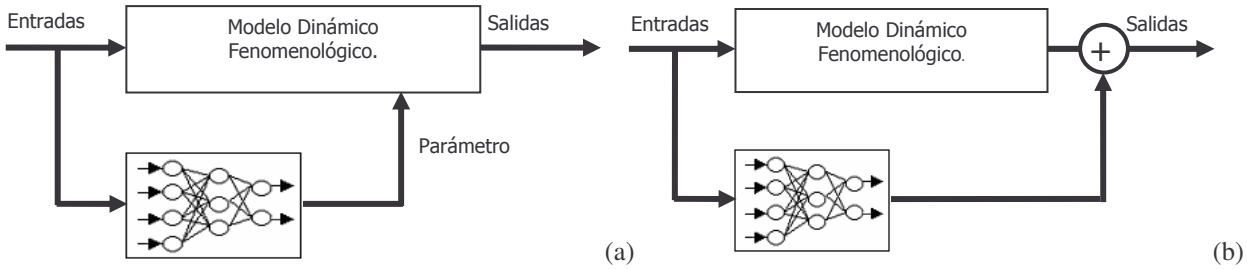


Figura 1: (a) Modelo de caja gris en serie. (b) Modelo de caja gris es paralelo

Luego, en 1996, Van Can, Hellinga, Luyben y Heijnen [3], demostraron que en general el esquema híbrido de caja gris en serie presenta mejores resultados, comparativamente con el esquema en paralelo, mostrando mejores rendimientos de modelos en serie para sistemas MISO (del inglés, *multiple input single output*). No obstante en modelos para sistemas SISO (del inglés, *single input single output*) los resultados mostrados son de características similares.

En 1999 Acuña, Cubillos, Thibault y Latrille [4], distinguieron entre dos formas de entrenamiento. La primera forma corresponde al entrenamiento directo (Figura 2 (a)), la cual utiliza el error originado a la salida de la red neuronal para la correcta determinación de sus pesos. La segunda forma es el entrenamiento indirecto (Figura 2 (b)), en donde se utiliza el error originado a la salida del modelo para efectos de aprendizaje de la red neuronal. El entrenamiento indirecto se puede realizar de dos formas, la primera es minimizando una función objetivo de variables de estado, por medio de una técnica de optimización no lineal y la segunda es retropropagando el error de salida en los pesos de la red neuronal a través de las ecuaciones del modelo fenomenológico.

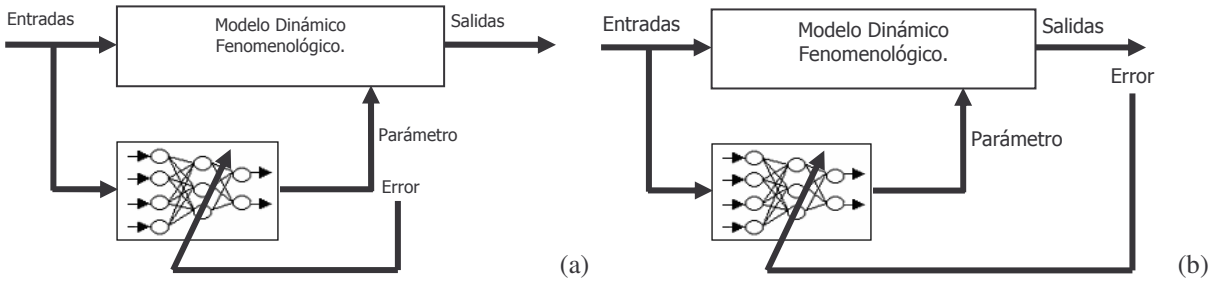


Figura 2: (a) Modelo de caja gris en serie con entrenamiento directo. (b) Modelo de caja gris con entrenamiento indirecto.

En 2004, Pinto y Acuña [5], desarrollaron una aplicación basada en Matlab® para la elaboración de modelos neuronales de caja gris utilizando una estrategia de entrenamiento directa, este trabajo tiene por objetivo realizar una implementación en Matlab®, de un algoritmo de entrenamiento basado en Levenberg-Marquardt con estrategia indirecta, retropropagando el error del modelo a través del modelo fenomenológico hacia la red neuronal.

## 3. IMPLEMENTACIÓN DE LA APLICACIÓN EN MATLAB®

La implementación desarrollada en Matlab® consta de dos distintas etapas, la primera de ellas consistió en la modificación del algoritmo de entrenamiento de Levenberg-Marquardt para permitirle poseer pesos fijos dentro de una red neuronal, esto se lleva a cabo con el objetivo de posteriormente modelar la parte fenomenológica a través de neuronas y pesos dentro de una red neuronal, en la cual los pesos entre estas neuronas no podrán ser modificados. De esta manera el error se propagará a través del modelo fenomenológico hacia la red neuronal.

El algoritmo de Levenberg-Marquardt es parte del toolbox de redes neuronales que incorpora Matlab®, con el cual se puede entrenar cualquier objeto de red neuronal. El utilizar el mismo algoritmo como base nos asegura un doble beneficio, por una parte la garantía de ser un algoritmo antes probado y, por otra parte, la total incorporación al entrenamiento de cualquier red, bajo los mismos comandos que el toolbox de redes neuronales ofrece, que podamos crear en la cual queramos darle la característica de poseer pesos fijos.

Una segunda etapa dentro del desarrollo de la aplicación, consiste en la completa comprensión del comando network, el cual funciona como primitiva de todos los tipos de redes neuronales que se pueden crear en Matlab® [6]. Es decir cada vez que creamos una red neuronal, el comando que siempre esta tras de esto es network. Sin embargo, network nos permite crear redes personalizadas a nuestra medida. La forma de utilizarlo es descrita a continuación:

Sintaxis

```
red = network
red = network(numeroDeEntradas, numeroDeCapas, conexionDeBias, conexionDeEntradas,
  conexionDeCapas, conexionDeSalidas, conexionDeTarget)
```

Descripción

Network posee los siguientes argumentos opcionales, los valores por defecto se muestran a continuación de la coma para cada argumento:

| | |
|---|---|
| numeroDeEntradas | : Número de entradas a la red, 0. |
| numeroDeCapas | : Número de capas de la red, 0. |
| conexionDeBias | : numeroDeCapas x 1, vector buliano, ceros. |
| conexionDeEntradas | : numeroDeCapas x numeroDeEntradas, matriz buliana, ceros. |
| conexionDeCapas | : numeroDeCapas x numeroDeCapas, matriz buliana, ceros. |
| conexionDeSalidas | : 1 x numeroDeCapas, vector buliano, ceros. |
| conexionDeTarget | : 1 x numeroDeCapas, vector buliano, ceros. |

Y retorna:

| | |
|---|---|
| red | : Nuevo objeto de red neuronal con determinados valores de propiedades. |

Una completa referencia del comando network, detallada además con ejemplos, puede ser encontrada en [6].

Crear una red personalizada es un trabajo bastante engorroso la primera vez, sin embargo, las siguiente ocasiones se torna considerablemente más sencillo, esto debido a la experiencia que comienza a poseer el diseñador de la red.

Una vez creada nuestra red debemos indicarle a Matlab® que para el entrenamiento utilice nuestra propio algoritmo de Levenberg-Marquardt con pesos fijos, el cual se puede fijar con *red.trainFcn = 'trainlmGris'*. Sin embargo, hasta el momento el algoritmo actuaría de la misma forma en que lo realiza Levenberg-Marquardt tradicional, ya que si no especificamos pesos fijos el algoritmo podrá modificar todos los pesos de la red. Para imponer pesos fijos al algoritmo debemos crear dos vectores de dimensiones 1 – por – número de pesos fijos. Los vectores deberán poseer los nombres de vectorDePesosFijos y ubicacionesPesosFijos, en donde se especificarán respectivamente los valores de los pesos fijos y la ubicación de éstos dentro de la red. Tales vectores deberán ser declarados como globales para así ser reconocidos por el algoritmo de entrenamiento, así como también los valores especificados en vectorDePesosFijos deben ser variables globales.

## 4. APLICACIÓN Y RESULTADOS

Para la aplicación del algoritmo creado se realizó la simulación de un proceso CSTR que utiliza una reacción exotérmica de primer orden [7]. La entrada al sistema corresponde a la temperatura de la camisa de enfriamiento y la salida del sistema corresponde al grado de avance de la reacción. Las ecuaciones de estado que describen al sistema son las siguientes:

$$x'_1 = -x_1 + D_a \cdot (1 - x_1) \cdot e^{\left( \frac{x_2}{1 + x_2 / \gamma} \right)} \tag{1}$$

$$x'_2 = -x_2 + B \cdot D_a \cdot (1 - x_1) \cdot e^{\left( \frac{x_2}{1 + x_2 / \gamma} \right)} + \beta \cdot (\mu - x_2) \tag{2}$$

$$y = x_1 \tag{3}$$

Donde:

$x_1$ : Grado de avance de la reacción.

$x_2$ : Temperatura adimensional del contenido del reactor.

$\mu$ : Entrada que corresponde a una tasa de flujo adimensional del fluido de transferencia de calor a través de la camisa de enfriamiento.

Para realizar la simulación se utilizaron para las constantes del modelo los siguientes valores:

$D_a = 0,072$

$\mathrm{B} = 8,0$

$\beta = 0,3$

$\gamma = 20,0$

Como los modelos neuronales de caja gris constan de dos partes, la primera compuesta por un modelo fenomenológico que es representado por ecuaciones diferenciales y la segunda compuesta por un modelo empírico que es representado por una red neuronal, por lo tanto, el modelo fenomenológico de caja gris es representado por las siguientes ecuaciones:

$$x'_1 = -x_1 + 0,072 \cdot (1 - x_1) \cdot \rho \tag{4}$$

$$x'_2 = -x_2 + 8 \cdot 0,072 \cdot (1 - x_1) \cdot \rho + 0,3 \cdot (\mu - x_2) \quad (5)$$

$$y = x_1 \quad (6)$$

Donde $\rho$ corresponde al parámetro con dificultad de obtener, el cual será estimado por la red neuronal. Sin embargo, para efectos experimentales y generación de datos, se utilizará la siguiente expresión:

$$\rho = e^{\left(\frac{x_2}{1+x_2/20}\right)}$$

Luego, el parámetro se asumirá completamente desconocido y se estimará utilizando la red neuronal entrenada en forma indirecta. A continuación, integrando las ecuaciones (4) y (5) en el intervalo de tiempo t y t+1, las ecuaciones discretas son las siguientes:

$$x_{1(t+1)} = x_{1(t)} + \left(-x_{1(t)} + 0,072 \cdot \left(1 - x_{1(t)}\right) \cdot \rho\right) \cdot \Delta t \quad (7)$$

$$x_{2(t+1)} = x_{2(t)} + \left(-x_{2(t)} + 8 \cdot 0,072 \cdot \left(1 - x_{1(t)}\right) \cdot \rho + 0,3 \cdot \left(u - x_{2(t)}\right)\right) \cdot \Delta t \quad (8)$$

Desarrollando las ecuaciones (7) y (8), éstas pueden ser descritas de la siguiente manera:

$$x_{1(t+1)} = x_{1(t)} - x_{1(t)}\Delta t + 0,072\rho\Delta t - 0,072\rho\Delta t x_{1(t)} \quad (9)$$

$$x_{2(t+1)} = x_{2(t)} - x_{2(t)}\Delta t + 8 \cdot 0,072\rho\Delta t - 8 \cdot 0,072\rho\Delta t x_{1(t)} + 0,3u\Delta t - 0,3x_{2(t)}\Delta t \quad (10)$$

Por lo que en definitiva las ecuaciones que constituyen la parte fenomenológica del modelo neuronal de caja gris son las siguientes:

$$x_{1(t+1)} = x_{1(t)} - x_{1(t)}\Delta t + 0,072\rho\Delta t - 0,072\rho\Delta t x_{1(t)} \quad (11)$$

$$x_{2(t+1)} = x_{2(t)} - x_{2(t)}\Delta t + 8 \cdot 0,072\rho\Delta t - 8 \cdot 0,072\rho\Delta t x_{1(t)} + 0,3u\Delta t - 0,3x_{2(t)}\Delta t \quad (12)$$

$$y = x_1 \quad (13)$$

Las ecuaciones de la parte fenomenológica pueden ser descritas en conjunto con la parte empírica del modelo, que es representada por una red neuronal. Para realizar dicha representación se diseña todo dentro de una gran red neuronal (Figura 3), la cual deberá poseer la característica de poder fijar pesos, de tal manera que éstos no sean alterados en la fase de entrenamiento. Los pesos en los cuales se indica un valor en la red neuronal corresponden a los pesos fijos, es decir, la parte fenomenológica del modelo, mientras los pesos que no muestran valores corresponden a la parte neuronal del modelo. Inicialmente los pesos modificables toman valores aleatorios con una distribución normal, media 0 y desviación estándar 0.5.

Para generar el modelo de caja gris dentro de una sola red neuronal, utilizando cuatro neuronas para la estimación del parámetro, se utiliza el comando network, y basado en la sintaxis anteriormente descrita, la red neuronal se debe crear de la siguiente manera para la simulación de un proceso CSTR:

network(3, 4, [1;1;0;0], [0 1 0;0 0 0;1 0 0;1 1 1], [0 0 0 0;1 0 0 0;0 1 0 0;0 1 1 0], [0 0 0 1], [0 0 0 1]);

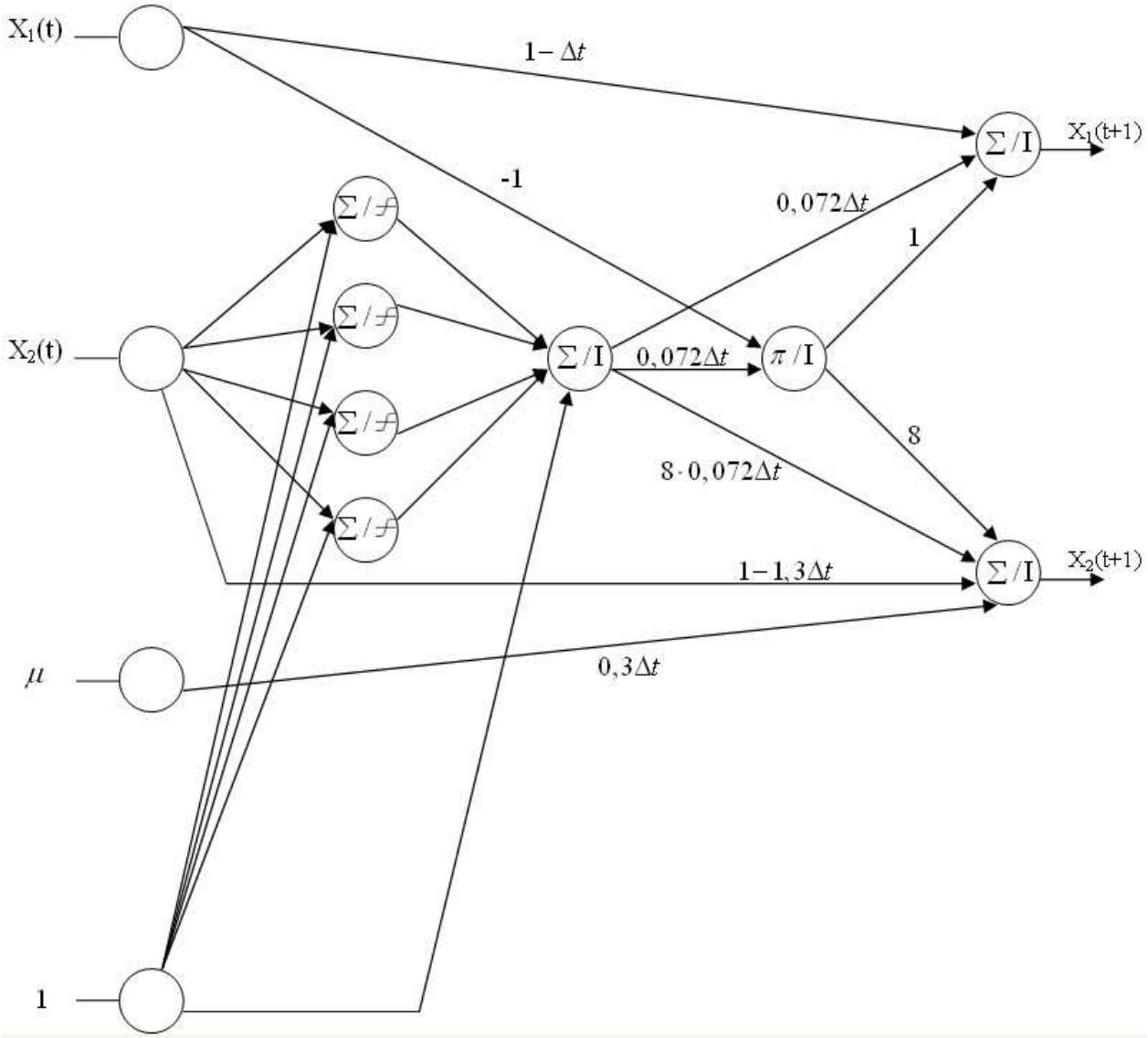


Figura 3: Red neuronal para proceso CSTR con pesos fijos.

Posteriormente, se debe indicar los pesos fijos, la cantidad de neuronas por capa, el tipo de función de activación, la función de transferencia y el rango de las entradas, la función de performance y el algoritmo de entrenamiento, básicamente como en cualquier objeto de red neuronal.

El la red se utiliza como función de *performance* el error cuadrático medio, que es calculado en la salida del modelo fenomenológico y desde ahí retropropagado hacia los pesos de la red que son modificables. En la fase de entrenamiento la entrada utilizada es una señal ruidosa (Figura 4 (a)), mientras que en la etapa de simulación la entrada utilizada es una señal sinusoidal (Figura 4 (b))

Para la parte neuronal del modelo se utilizó tangente hiperbólica como función de transferencia, mientras para la parte fenomenológica se utilizó la función identidad como función de transferencia. La función de activación utilizada fue la sumatoria de las ponderaciones de las entradas por los pesos, exceptuando una neurona, donde se utilizó una productoria entre la ponderación de las entradas por los pesos, esto debido a la forma de las ecuaciones fenomenológicas. La red neuronal híbrida o modelo de caja gris fue entrenada por 3.000 épocas. Es importante notar

que el número de épocas de entrenamiento aumenta en este tipo de modelos, ya que al poseer pesos fijos es una restricción que se añade al algoritmo de optimización.

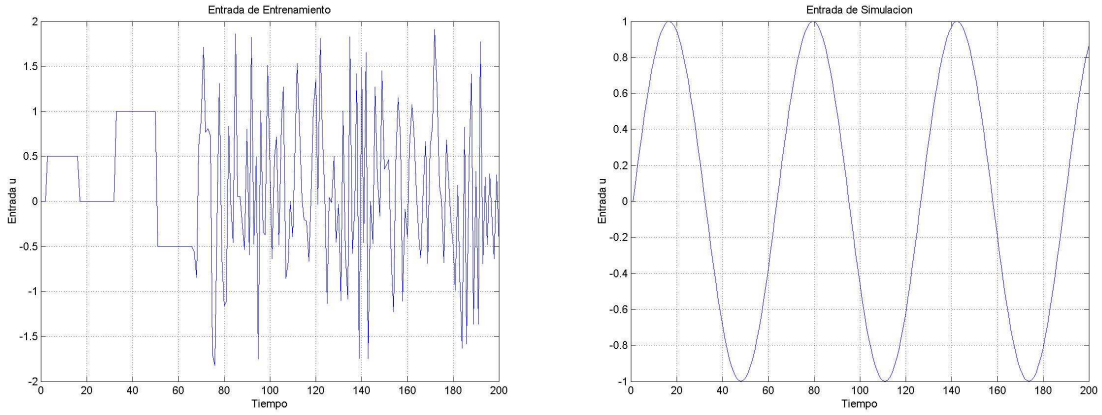


Figura 4: (a) Entrada utilizada para la etapa de entrenamiento. (b) Entrada utilizada para la etapa de simulación.

El valor utilizado para la simulación de $\Delta t$ fue de 0.005, mientras los valores iniciales para las variables de estado son los siguientes:

$$x_1 = 0,1$$
$$x_2 = 0,8$$

Luego del entrenamiento el error cuadrático medio tiene un valor de 0.000140477 (Figura 5 (a)), la validación obtenida del modelo neuronal de caja gris puede apreciarse en la Figura 5 (b).

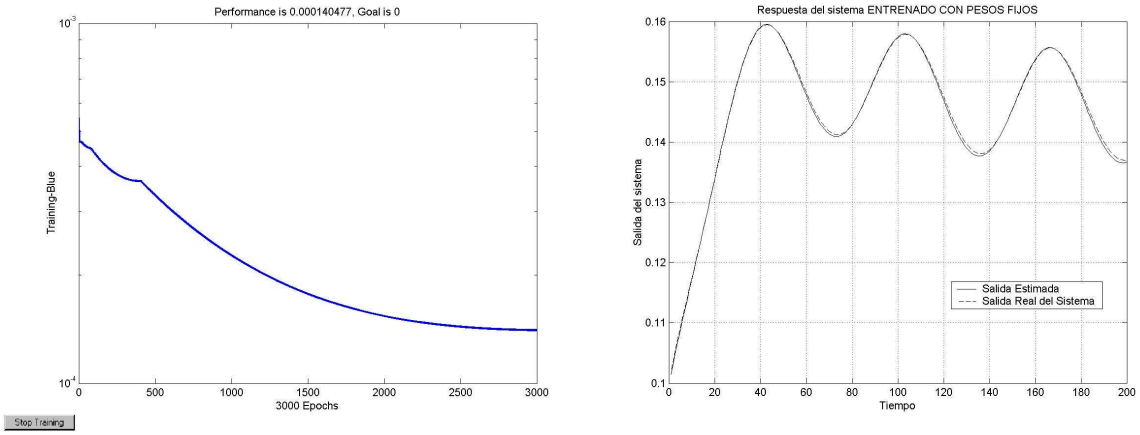


Figura 5: (a) Error cuadrático medio en la etapa de entrenamiento. (b) Validación de la salida real del sistema y la salida obtenida mediante el modelo neuronal de caja gris.

Finalmente posterior a la simulación el valor final del vector de estado y la salida en la iteración 200 es el siguiente:

$$X_t = \begin{bmatrix} 0.1369 \\ 0.8472 \end{bmatrix}$$
$$Y = \begin{bmatrix} 0.1369 \end{bmatrix}$$

En la validación del modelo neuronal se puede observar que prácticamente la estimación del modelo neuronal de caja gris se comporta de la misma forma que la salida real del sistema real, obteniendo un bajo valor de error.

## 5. CONCLUSIONES

Los modelos neuronales de caja gris poseen reales ventajas con respecto a los de caja negra, ya que permiten aprovechar el conocimiento a priori que se posea de un sistema. Además apoyan a los modelos de caja blanca estimando parámetros difíciles de obtener o bien que no pueden ser obtenidos directamente en línea.

El método de aprendizaje indirecto utilizado en este trabajo provee de una buena estimación sin la necesidad de poseer valores reales para los parámetros a estimar, solo bastando tener datos de las salidas reales del sistema, ya que el error es calculado en la salida del modelo de caja gris, y desde ahí retropropagado hacia el interior.

El error cuadrático medio obtenido posteriormente al entrenamiento ha sido bastante bajo, lo que se refleja en la posterior validación de la salida estimada por el modelo neuronal de caja gris en comparación con la salida real del sistema.

## AGRADECIMIENTOS

Los autores desean agradecer el financiamiento parcial del Proyecto Fondecyt 1040208

## REFERENCIAS

[ 1 ] Psichogios, D.; Ungar, L. (1992). "*A Hybrid Neural Network-First Principles Approach to Process Modeling*", Vol. 38, Nº10, páginas 1499-1511.

[ 2 ] Thompson, M.; Kramer, M. (1994). "*Modeling Chemical Processes Using Prior Knowledge and Neural Networks*", Vol. 40, Nº8, páginas 1328-1340.

[ 3 ] Van Can, H; Hellinga, C.; Luyben, K.; Heijnen, J. (1996). "*Strategy for Dynamic Process Modeling Based on Neural Network in Macroscopic Balances*", AIChE 42, páginas 3403-3418.

[ 4 ] Acuña, G.; Cubillos, F.; Thibault, J.; Latrille, E. (1999). "*Comparison of Methods for Training Grey-Box Neural Network Models*", Computers and Chemicals Engineering Supplement, páginas 561-564.

[ 5 ] E. Pinto, G Acuña, *"Desarrollo de un Toolbox Matlab para la elaboración de modelos neuronales de caja gris"*, Actas en CD del XVI Congreso de la Asociación Chilena de Control Automático, Universidad de las Américas, Santiago, Noviembre 2004.

[ 6 ] Demuth H.; Beale M. (1998). *"Neural Network Toolbox For Use with Matlab®"*

[ 7 ] Hernández, E.; Arkun, Y. (1992). "*Study of the Control-Relevant Properties of Backpropagation Neural Network Models of Nonlinear Dynamical Systems*", Computers Chemical Engineering, Volumen 16, número 4, páginas 227-240, Abril.