UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO.

FACULTAD DE INGENIERÍA.

ENTRENAMIENTO DE RED NEURONAL PARA PREDECIR PARÁMETROS DE ROBÓTICA BÍPEDA MEDIANTE CAMINATA HUMANA.

TESIS
QUE PARA OBTENER EL TÍTULO DE
INGENIERO MECATRÓNICO.

PRESENTA:
MAURICIO GARCÍA HERNÁNDEZ.

DIRECTOR DE TESIS:
DR. JOSÉ LUIS FERNÁNDEZ ZAYAS.

Ciudad Universitaria, febrero 2014.

AGRADECIMENTOS

La primera mención va dirigida a la que siempre será sin posibilidad de duda alguna la máxima casa de estudios, la Universidad Nacional Autónoma de México, la cual ha sido mi formadora desde hace ya 8 años cuando por mis méritos fui admitido en su bachillerato, hoy que soy Ingeniero es preciso externar mi más profundo agradecimiento y admiración a una institución tan honorable y noble, que brinda a sus integrantes el ambiente propicio para su constante evolución, institución que ha tenido un impacto fundamental en mi persona, en ser quien soy ahora, que ha marcado mi vida y que siempre estará presente en mi corazón.

Agradezco a mi familia, que a través de sus innumerables lecciones y apoyo incondicional me brindó todas las herramientas necesarias para ser una persona que marca diferencias. A mi hermano, quien siempre ha sido y será una persona que tiene y tendrá mi confianza le agradezco su paciencia y alegría. A mi padre por guiarme siempre por el camino del honor, el carácter, la inteligencia y las personas ejemplares. A mi madre por su constante dedicación y esfuerzo.

A mi novia, quien desde el día que entró en mi vida es la fuente de mi inspiración, le doy las gracias por obsequiarme su visión del mundo, su entusiasmo, su confianza, su sinceridad, su infinito apoyo y su amor.

A todos los que son mis verdaderos amigos, personas por las que siento gran afecto, poseedoras de cualidades que admiro, que sin importar su edad o donde se encuentren he tenido la fortuna de contar con su presencia constante en mi vida les agradezco su lealtad, amistad y lecciones.

Todas las personas que han contribuido a mi desarrollo tendrán siempre mi gratitud, ideal sería poder mencionarlas a todas, sin embargo al ser el espacio limitado me permito mencionar a:

-El entrenador del equipo de fútbol rápido de la Universidad Nacional Autónoma de México Juan Rodríguez Contreras, quien lleno de cualidades siempre con una excelsa fraternidad me ha brindado su amistad, además de lecciones de gran importancia en la vida que contribuyeron a mi formación integral como ser humano y como un verdadero universitario de sangre azul y de piel dorada.

-El Doctor José Luis Fernández Zayas, que siempre ha tomado la iniciativa en el impulso de la ingeniería mexicana y de su juventud, persona de gran profesionalismo y elocuencia, cuya vida y trayectoria sirve de ejemplo para una gran variedad de personas, quien me brindó su incondicional y completo apoyo.

¡México, Pumas, Universidad!

“Por mi raza hablará el espíritu”
Universidad Nacional Autónoma de México.

Contenido

Índice de figuras

Resumen

Ésta tesis se genera en un marco de cooperación entre la Universidad Nacional Autónoma de México mediante su Dirección General de Cooperación e Internacionalización, apoyada por el Departamento de Movilidad Estudiantil y la Secretaria de Servicios Académicos de su Facultad de Ingeniería con la Universidad Carlos III de Madrid.

El presente trabajo consiste en la generación de una red neuronal artificial y su entrenamiento para la predicción de valores originados por un sensor inercial, brinda el contexto fundamentalmente necesario para comprender el concepto de robótica humanoide y redes neuronales; a través de la selección de un software de trabajo se desarrolla un modelo de red capaz de cumplir con los requisitos de diseño brindados por el laboratorio de robótica de la Universidad Carlos III de Madrid, el cual está llevando a cabo la construcción de un robot humanoide conocido como RH-2 que será capaz interactuar con su entorno.

Para la propuesta de la solución final se analizan diferentes arquitecturas de red capaces de predecir valores en el tiempo, se contempla también la funcionalidad de los algoritmos de aprendizaje y reglas de error comúnmente utilizadas.

Los datos a predecir se generaron mediante la medición de parámetros en la caminata y caída humana hacía el frente. Tales datos en combinación con los que sean generados en un futuro por humanos, o cuando el RH-2 sea capaz de desplazarse por su propia cuenta se utilizarán empleando la red neuronal artificial generada por éste trabajo de tesis como experiencia por el robot; a través de la experiencia el autómata conocerá las aceleraciones a las que podría someterse si se desplazara por alguna de sus posibles trayectorias de movimiento.

Capítulo 1: Introducción

1.1 Contexto del desarrollo

El ser humano con el objetivo constante de mejorar sus condiciones de vida y sobrevivir descubrió la necesidad de crear herramientas generadoras de progreso, también comprendió su capacidad para producir herramientas automáticas, su meta final es crear herramientas totalmente autónomas, actualmente se encuentra en la frontera tecnológica entre las herramientas automáticas y semiautónomas.

De acuerdo con la definición de la Real Academia Española la automática es la ciencia que trata de sustituir en un proceso el operador humano por dispositivos mecánicos o electrónicos (RAE, 2014), en éste sentido el nivel de la tecnología actual en el ámbito de las maquinas-herramientas es evidente, el siguiente paso evolutivo consiste en construir herramientas autónomas, generando así dispositivos capaces de tomar decisiones por si mismos para cumplir con la tarea asignada. Las herramientas al llevar a cabo su función dentro de un ambiente humano deben cumplir con requisitos antropomórficos específicos, donde la herramienta ideal sería aquella que pudiera desempeñar la totalidad de las funciones de un ser humano; es necesaria la creación de máquinas que remplacen completamente al hombre en el desempeño de un abanico de actividades que comprende desde los actos de rescate hasta labores productivas. En consecuencia es primordial la construcción de máquinas con forma humana que no dependan totalmente del hombre, con el objetivo de mejorar la condición de vida del mismo.

En dicho contexto distintos campos del conocimiento han encontrado directa o indirectamente un campo fértil de desarrollo, surgiendo de ésta tendencia la robótica, la cual ha experimentado un crecimiento sostenidamente acelerado en las últimas décadas, al ser un área del conocimiento relativamente nueva y en constante expansión coexiste en una sinergia entre las investigaciones científicas y el empleo de conocimientos previos existentes.

Los paradigmas que genera la propuesta de nuevos desarrollos toman forma cuando es preciso generar soluciones a problemas inéditos, por ser de ésta condición generalmente no presentan un resultado satisfactorio al ser analizados con los modelos existentes. En el caso de la robótica bípeda uno de los principales obstáculos es la caminata estable, existen diversos enfoques orientados a la solución de tal contratiempo, las compañías y las universidades que realizan investigación en la materia se encuentran inmersas en la carrera de la locomoción bípeda estable.

En la ingeniería se observa que cuando el sistema a gobernar presenta una complejidad superior las técnicas de control empleadas para gobernarlo son de una sofisticación mayor. Si un sistema es capaz de percibirse a sí mismo y actuar en consecuencia en base a una tarea asignada se dice que es “inteligente”, una forma de control inteligente son las redes neuronales artificiales (RNA); inspiradas en el proceso de pensamiento humano pretenden emular el aprendizaje basado en experiencias pasadas, evidentemente tal enfoque es ampliamente compatible con la robótica.

1.2 Planteamiento del problema

La Universidad Carlos III de Madrid a través del laboratorio de robótica (RoboticsLab) perteneciente al departamento de Automatización e Ingeniería en Sistemas, cuenta con una línea de investigación en robótica humanoide iniciada en 2001 con el desarrollo del robot Leroy de la figura 1.1, a partir del 2002 se llevaron a cabo un gran número de proyectos de investigación para desarrollar la serie RH de robots humanoides, desde un enfoque mecatrónico los logros principales del programa fueron la generación del RH-0 y el RH-1, actualmente el RoboticsLab se encuentra desarrollando un nuevo prototipo denominado RH-2 para implementar investigaciones relacionadas con la locomoción, percepción y control del comportamiento durante la ejecución de tareas (Martínez de la Casa Díaz, 2012).

Figura 1. 1 Robot Leroy (UC3M, 2014).

Para que el RH-02 cuya maqueta virtual se muestra en la figura 1.2 tenga la capacidad de desplazarse en su entorno de trabajo necesita conocer diversos

parámetros, entre ellos la aceleración de su propio cuerpo respecto a un marco de referencia, la finalidad es poder controlar sus movimientos en consecuencia. Sin embargo no basta conocer los valores actuales de éstos parámetros, es necesario conocer también los valores futuros de dichas aceleraciones para poder diferenciar y seleccionar el mejor próximo movimiento posible, de ésta forma si el robot se encuentra en una trayectoria que generará una pérdida de estabilidad puede modificarla, asimismo es necesario que aprenda de tales experiencias.

Figura 1.2 Maqueta virtual del RH-02 (Martínez de la Casa Díaz, 2012).

Por lo tanto es imprescindible la predicción de valores futuros de las aceleraciones generados por el sensor a bordo del robot humanoide, además del aprendizaje de los mismos. Evaluando las condiciones del problema el RoboticsLab ha propuesto la generación de la solución mediante el empleo de una red neuronal, el error de generalización de la red neuronal no debe exceder un valor del 2% y debe ser un modelo confiable.

Dado que el RH-2 buscará emular la forma de caminar del hombre, el RoboticsLab ha decidido generar los parámetros iniciales de entrenamiento para el robot censando los valores que se generan durante la caminata y la caída al frente humana, tal es el origen de los datos de entrenamiento a ser empleados.

1.3 Alcance y objetivos

Objetivo General

- Entrenar una red neuronal capaz de cumplir la predicción de los parámetros de aceleración del sensor a bordo del RH-2 basada en el aprendizaje de experiencias pasadas.

Objetivos específicos

- La red neuronal generada por el entrenamiento no debe tener un error de generalización mayor al 2%.
- El modelo generado debe predecir los valores de forma confiable.
- La solución final debe ser capaz de aprender de experiencias pasadas, en concreto por los datos generados por el sensor MTi-28A53G53 a través de la caminata humana.

Alcance

- El alcance de éste trabajo es teórico, su finalidad es entrenar una red neuronal que cumpla con las características de diseño solicitadas utilizando los datos de entrenamiento provistos por el Laboratorio de Robótica de la Universidad Carlos III de Madrid. A través del diseño modular éste trabajo servirá para futuros desarrollos dentro del proyecto RH-2, por ejemplo el cálculo del ZMP del robot, su aprendizaje de los patrones de aceleraciones que desencadenan una caída, la dotación del sentido del equilibrio, etcétera.

Capítulo 2: Estado del Arte

2.1 Las redes neuronales artificiales

El objetivo primordial de la Ingeniería de Control es desarrollar sistemas que sean capaces de gobernar los procesos deseados mientras interactúan con las variables de su entorno de operación, dentro de estos desarrollos existen diversos enfoques, uno de ellos consiste en generar sistemas que sean capaces de ejecutar sus tareas con alta eficiencia sin la intervención humana al aprender a través de experiencias pasadas, es decir sistemas inteligentes.

Una aproximación a este enfoque son las denominadas redes neuronales artificiales, las cuales de igual forma que muchas otras soluciones a problemas brindadas por la ciencia están inspiradas en modelos observados en la naturaleza, la manifestación de dicha inspiración está presente en múltiples objetos de la vida cotidiana, por ejemplo aeronaves inspiradas en las aves, materiales más resistentes y ligeros que aquellos convencionales inspirados en el padrón en el que las abejas construyen sus panales, trenes bala inspirados en el Martín pescador, etcétera. Las redes neuronales artificiales buscan emular el funcionamiento de las redes neuronales biológicas en el cerebro humano y su forma en desencadenar reacciones asociadas a los diversos estímulos a las que están sujetas

2.1.1 Las redes neuronales biológicas

El sistema nervioso humano contiene estimativamente 10^{11} neuronas con más de 10^{14} conexiones (Sadava, Heller, Orians, Purvens, & Hillis, 2009) , una red neuronal biológica es la unión de un gran número de neuronas interconectadas entre sí.

Podemos comenzar entonces brindando la definición de neurona, las neuronas son el componente básico del sistema nervioso, son células que están compuestas de tres elementos principales: las dendritas, el cuerpo de la célula o soma y el axón. Las dendritas son el elemento receptor; son las partes que cargan las señales eléctricas al cuerpo de la célula; el cuerpo realiza la suma de esas señales y el axón lleva la señal desde el cuerpo de la célula hacia otras neuronas (Stevens, 1979), el axón se compone a su vez de un recubrimiento aislante denominado vaina de mielina y por los nodos de Ranvier (Simpson, 1990).

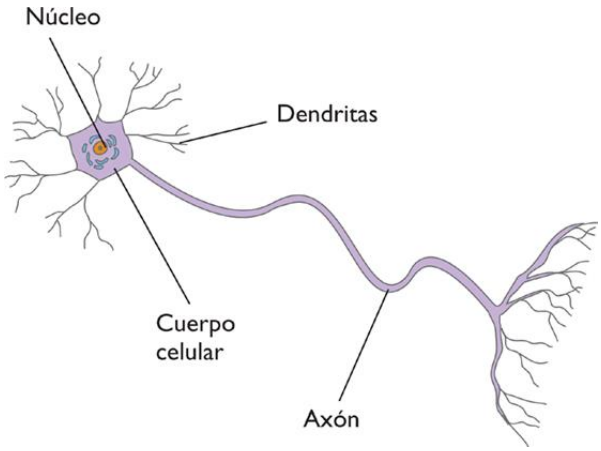


Figura 2.1 Esquema de una neurona (Estevam, 2014).

2.1.2 Sinapsis, comunicación entre las neuronas

Existe un proceso de flujo de señales eléctricas entre dichas células, este proceso se denomina sinapsis, se da entre una neurona que envía una señal a través de su axón y otra que recibe la señal a través de su dendrita, dichas neuronas son denominadas presináptica y postsináptica respectivamente (Martín del Brío & Sanz, 1997).

Las señales referidas se manifiestan en impulsos eléctricos, para que la señal fluya desde la neurona presináptica se genera un cambio en la conductancia de la membrana del axón, que abre y cierra los canales de sodio y de potasio. La membrana en reposo presenta un gradiente de potencial eléctrico de -70 [mV]. La transmisión de la señal a través del axón es el resultado también de una serie de despolarizaciones que ocurren en los nodos de Ranvier. Cuando uno de los nodos de despolariza se desencadena la despolarización del siguiente nodo, por lo tanto el potencial de acción viaja a lo largo de la fibra de forma discontinua. Una vez que un potencial de acción ha pasado por cierto punto, éste no puede volver a ser excitado durante aproximadamente 1 [ms], que es el tiempo que le toma en volver a su potencial de reposo, evidentemente estos periodos necesarios de reposo limitan la frecuencia de la transmisión de los impulsos nerviosos en las neuronas (Stevens, 1979).

El siguiente paso en la comunicación neuronal ocurre cuando el impulso llega al botón sináptico, de acuerdo a la forma de contacto entre dos neuronas que presentan sinapsis se pueden diferenciar dos tipos, la sinapsis eléctrica y la sinapsis química. En la sinapsis química la llegada del impulso al botón sináptico ocasiona la liberación de neurotransmisores por parte de la célula presináptica, y la absorción de éstos por la célula postsináptica en la hendidura sináptica, mientras tanto en la sinapsis eléctrica el botón sináptico entra en contacto directo con la célula postsináptica, y el potencial de acción pasa directamente a través de canales que conectan el citoplasma de las dos neuronas en una conexión física directa (Bustamante, 2007).

Así en la sinapsis se convierte una señal eléctrica de la neurona presináptica en un proceso químico, que después se convierte en una señal eléctrica en la neurona postsináptica. También cabe señalar que existen dos tipos de neurotransmisores: los excitatorios, que incrementan el potencial de membrana, y los inhibitorios, que decrementan dicho potencial (Stevens, 1979).

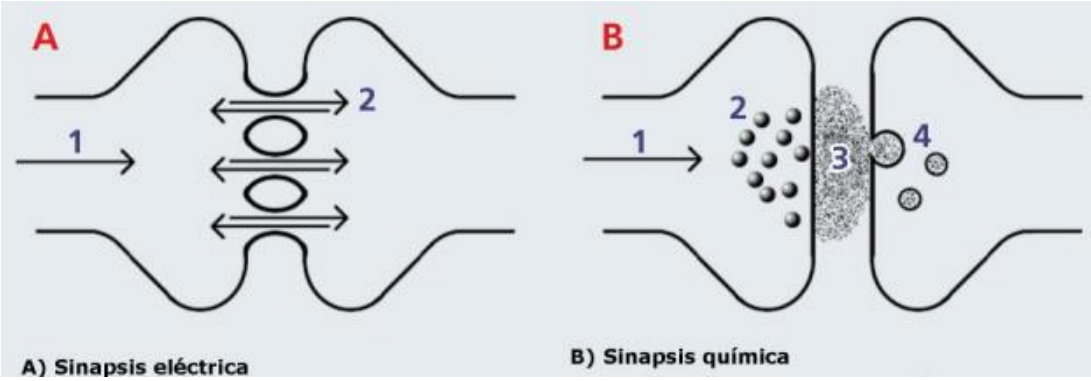


Figura 2.2 Esquema de la sinapsis eléctrica (A) y sinapsis química (B) (Cortés, 2014) .

2.2 Las redes neuronales artificiales

Una red neuronal artificial es un modelo matemático que intenta emular una red neuronal biológica a través de la interconexión de unidades de proceso denominadas neuronas. Con el objetivo de tener un comportamiento similar a sus pares biológicas, las redes neuronales artificiales poseen diversas características como son la auto organización, la generalización y el aprendizaje, las cuales les permiten resolver diversos problemas.

Para explicar matemáticamente éstas unidades de proceso fundamentales es necesario comenzar por definir gráficamente una neurona usando el esquema de la figura 2.3, en él podemos observar cuatro componentes principales:

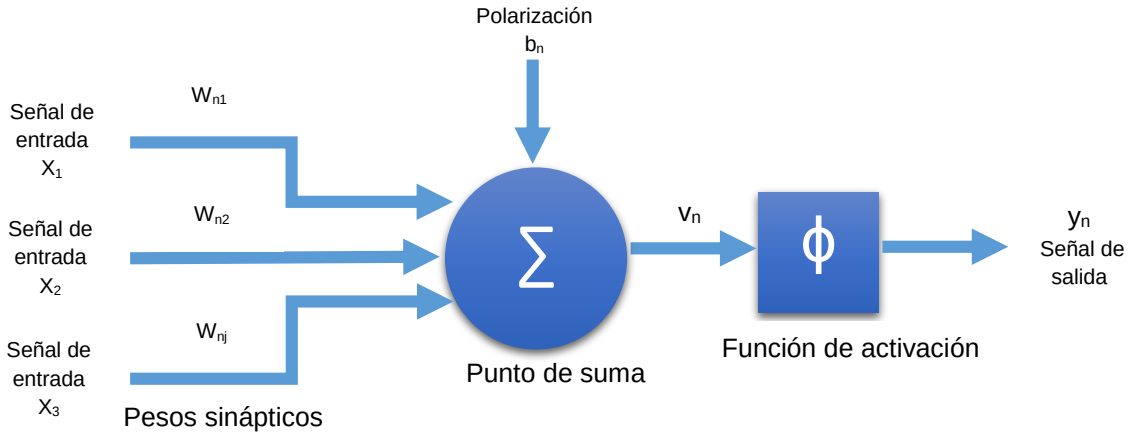


Figura 2.3 Esquema de una neurona artificial.

-Señales de entrada con sus enlaces al punto suma*:* Las señales de entrada x_1, x_2,…, x_n llegan al puno suma después de ser multiplicadas por los pesos sinápticos w_{nj} donde el subíndice n indica la neurona receptora y el subíndice j indica la neurona emisora.

-Sumador ∑: Suma los componentes de las señales de entrada multiplicadas por su peso correspondiente.

-Función de activación: Brinda la salida de la neurona en función del potencial de activación v_n.

-Umbral: Desplaza la entrada.

2.2.1 Definición matemática de las neuronas artificiales

En términos matemáticos, es posible describir la neurona *n* mostrada en la figura 2.3 por el siguiente par de ecuaciones.

$$u_n = \sum_{j=1}^{m} w_{nj}\, x_j \tag{2.1}$$

$$y_n = \phi(u_n + b_n) \tag{2.2}$$

u_n es la combinación lineal de las entradas ponderadas por los pesos sinápticos; b_n es la polarización o umbral; ϕ es la función de activación; y_n representa la señal de salida de la neurona.

La polarización es un parámetro externo de la neurona *n*, pero es posible considerarla como parte de las señales de entrada, de tal forma que si se combinan (2.1) y (2.2) se tiene:

$$v_n = \sum_{j=0}^{m} w_{nj}\, x_j \tag{2.3}$$

$$y_n = \phi(v_n) \tag{2.4}$$

A v_n se le denomina potencial de activación. En la ecuación (2.3) se ha agregado una nueva sinapsis. Su entrada:

$$x_0 = +1 \tag{2.5}$$

Y el peso correspondiente es:

$$w_{n0} = b_n \tag{2.6}$$

(Sánchez & Alanís, 2006).

2.2.2 Funciones de activación de las redes neuronales artificiales

La función de activación brinda la salida de la neurona relacionada dado el potencial de activación presente. Existen diversos tipos de funciones de activación que permiten generar la salida y_n, las cuales se diferencian unas de otras por sus distintos umbrales de activación; una característica deseable de las funciones de activación es el ser diferenciable ya que algunas reglas de aprendizaje presentan ese requisito, las funciones de activación más comunes son:

Función Escalón

La ecuación (2.7) describe la gráfica de la figura 2.4.

$$\phi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases} \tag{2.7}$$

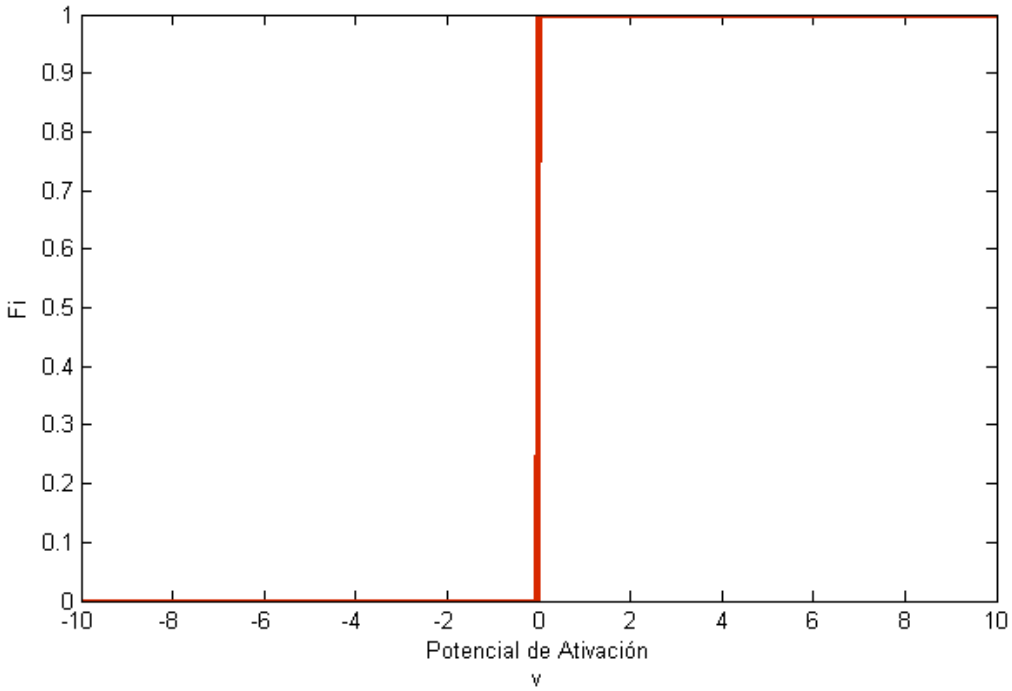


Figura 2.4 Función Escalón.

Función lineal a tramos

La gráfica de la figura 2.5 queda descrita por la ecuación (2.8).

$$\phi(v) = \begin{cases} 1, & v \geq 4 \\ v/4, & 4 > v > -4 \\ 0, & v \leq -4 \end{cases} \tag{2.8}$$

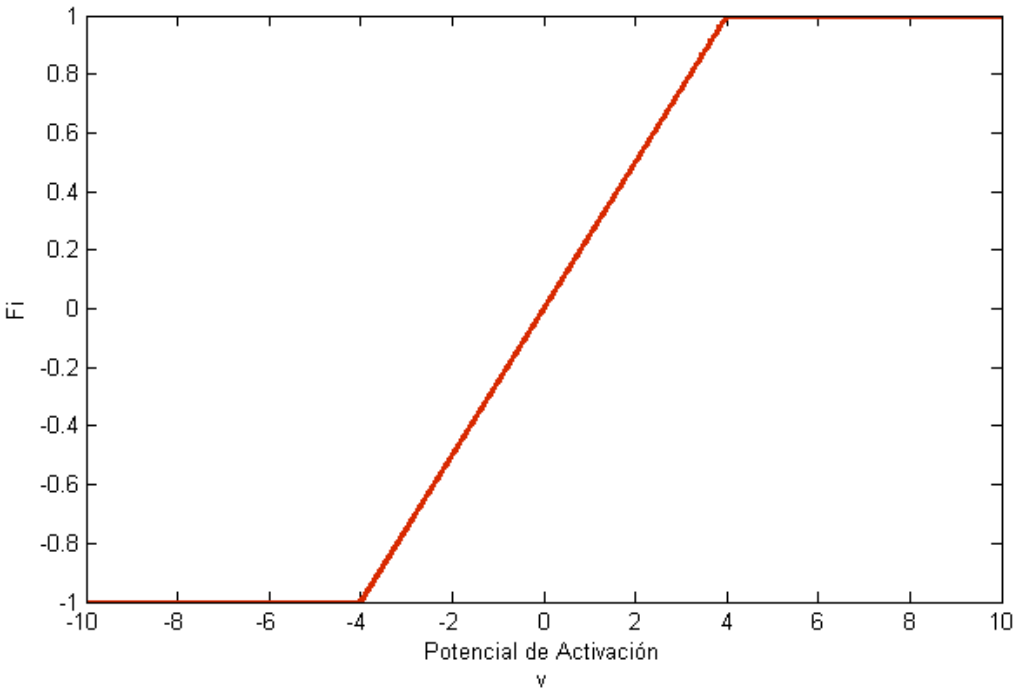


Figura 2.5 Función lineal a tramos.

Función sigmoidal

Las funciones de tipo sigmoidal son las más comúnmente utilizadas en las redes neuronales artificiales debido a que su umbral de activación es una curva que presenta una trayectoria más suave que otras, la ecuación (2.9) describe la gráfica de la figura 2.6.

$$\phi(v) = \tanh(v) \tag{2.9}$$

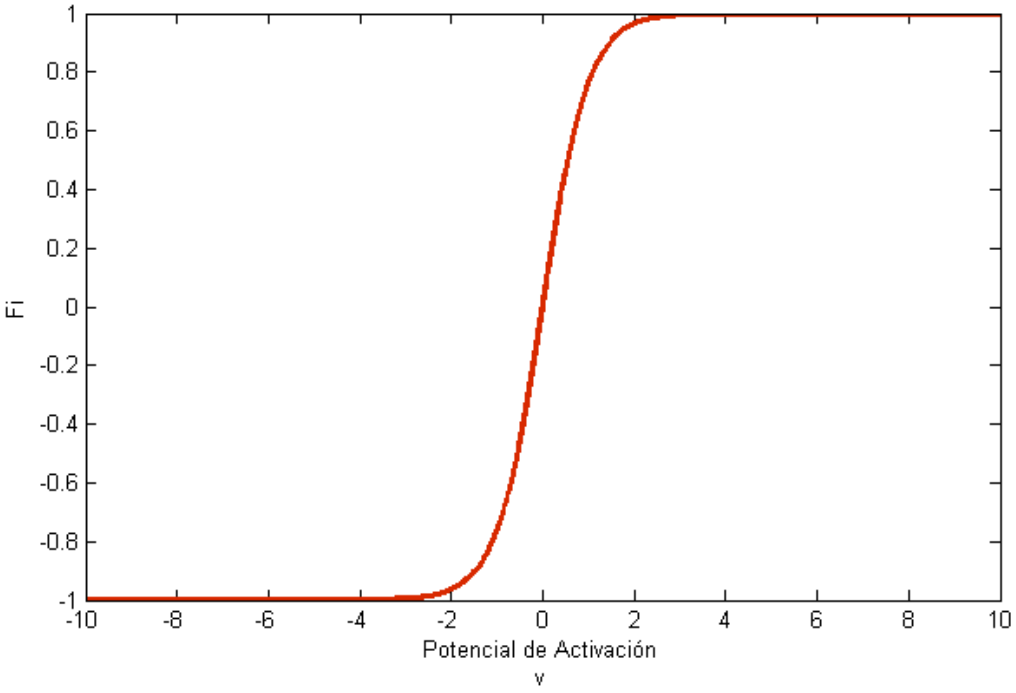


Figura 2.6 Función sigmoidal.

2.2.3 Arquitecturas de las redes neuronales artificiales

Se denomina arquitectura a la topología, estructura o patrón de conexionado de una red neuronal. En una red neuronal artificial, los nodos se conectan por medio de sinapsis; está estructura de conexiones sinápticas determina el comportamiento de la red. Las conexiones sinápticas son direccionales, es decir, la información solamente puede propagarse en un único sentido. En general, las neuronas se suelen agrupar en unidades estructurales que se denominan capas, el conjunto de una o más capas constituye una red neuronal.

Se distinguen tres tipos de capas: de entrada, de salida y ocultas. Una capa de entrada o sensorial está compuesta por neuronas que reciben datos o señales procedentes del entorno, por ejemplo proporcionadas por sensores. Una capa de salida es aquella cuyas neuronas proporcionan la respuesta de la red neuronal, sus neuronas pueden estar conectadas a actuadores. Una capa oculta es aquella que no tiene una conexión directa con el entorno, es decir, que no se conecta

directamente ni a órganos sensores ni a actuadores (Martín del Brío & Sanz, 1997).

Si se consideran las interconexiones de una red neuronal, sus pesos y las funciones de activación más comunes, se entiende que los pesos w_{nj} y sus variaciones fortalecen o debilitan la comunicación entre neuronas presinápticas y postsinápticas.

Redes neuronales unicapa

En una red neuronal unicapa, la capa de entrada se conecta directamente a la capa de neuronas de salida por medio de la sinapsis. Se designa como unicapa porque sólo tiene una capa con nodos computacionales; para ésta designación no se toma en cuenta la capa que contiene los nodos de entrada (Sánchez & Alanís, 2006), éste concepto se ejemplifica en la figura 2.7.

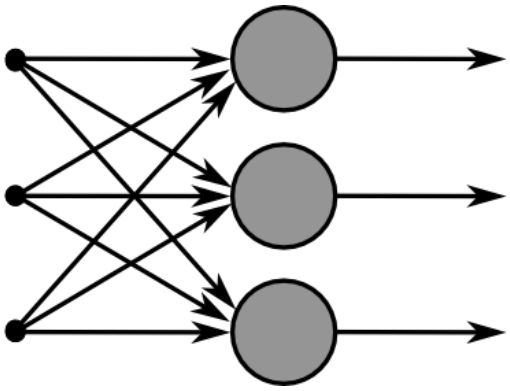


Figura 2.7 Red Neuronal Unicapa (Chrislb, SingleLayerNeuralNetwork, 2014).

Redes neuronales multicapa

Una capa oculta es aquella que se compone de neuronas que no tienen conexión con el entorno, las redes neuronales multicapa reciben ese nombre porque poseen una o más capas ocultas. Si todos los nodos de una capa se encuentran conectados a todos los nodos de la capa siguiente la red se denomina totalmente conectada; en el caso en que falte alguna conexión la red se denomina parcialmente conectada. La figura 2.8 muestra el esquema de una red neuronal totalmente conectada que tiene una capa oculta.

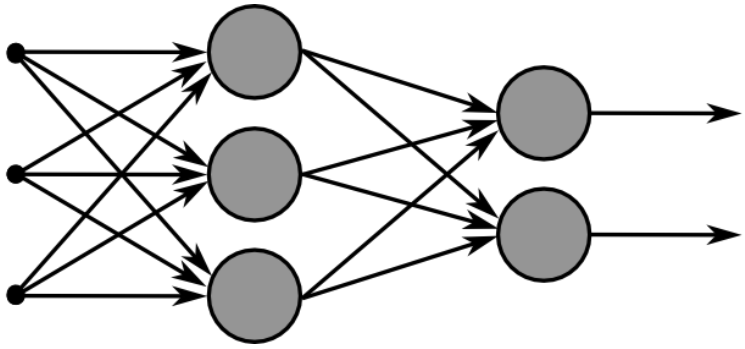


Figura 2.8 Red Neuronal Multicapa (Chrislb, MultiLayerNeuralNetwork, 2014).

Redes neuronales recurrentes.

Las redes neuronales recurrentes se distinguen de las anteriores en que por lo menos tienen una sinapsis de retroalimentación. Por ejemplo, una red recurrente puede consistir en una sola capa oculta con cada una de sus neuronas retroalimentando las señales de salida a la entrada de otras neuronas. La figura 2.9 muestra el esquema de una red neuronal recurrente donde la sinapsis de retroalimentación ésta indicada por la letra D (Sánchez & Alanís, 2006).

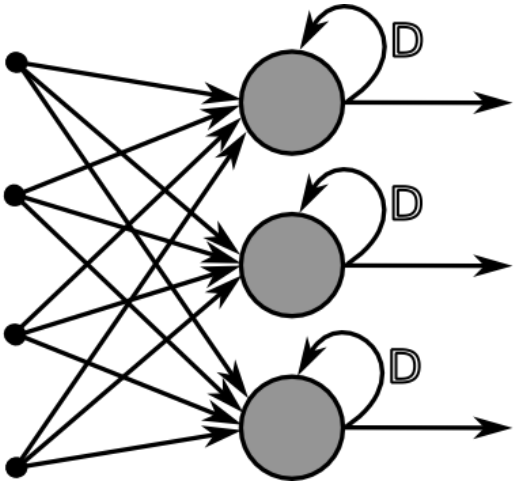


Figura 2.9 Red Neuronal Recurrente (Chrislb, RecurrentLayerNeuralNetwork, 2014).

2.2.4 El aprendizaje en las redes neuronales artificiales

Las redes neuronales pueden tener memoria, es decir, pueden memorizar un comportamiento o pueden aprender, sin embargo el primer escenario no es de amplío interés ya que es posible memorizar patrones a través de modelos más simples, como es el caso de las funciones polinómicas de orden superior; el segundo escenario es de primordial importancia, ya que uno de los objetivos principales de las redes neuronales es realizar un trabajo determinado después de haber aprendido del entorno.

La red neuronal aprende variando sus parámetros, en particular los pesos sinápticos y el umbral de polarización. El aprendizaje es un proceso por el cual los parámetros se adaptan por la interacción continua con el medio ambiente (Haykin, 1999). La forma de aprender de la red neuronal se manifiesta en el procedimiento de adaptación de dichos parámetros. Usualmente el proceso de aprendizaje consiste en determinar utilizando diversos algoritmos el conjunto de pesos w_{nj} que permita que la red generalice.

Cuando se construye un sistema neuronal se parte de un cierto modelo de neurona y de una determinada arquitectura de red, estableciéndose generalmente los pesos sinápticos iniciales como nulos o aleatorios. Para que la red resulte operativa es necesario entrenarla, lo que constituye el modo de aprendizaje. La forma convencional del entrenamiento o aprendizaje es el modelado de las sinapsis, consiste en modificar los pesos sinápticos siguiendo una cierta regla de aprendizaje construida normalmente a partir de la optimización de una función de error, enfocada en la eficacia actual de la operación de la red. Si denominamos $w_{nj}(k)$ al peso que conecta la neurona presináptica j con la postsináptica n en la iteración k, el algoritmo de aprendizaje, en función de las señales que en dicha iteración llegan procedentes del entorno, proporcionará el valor $\Delta w_{nj}(k)$ que da la modificación que se debe incorporar en dicho peso (Martín del Brío & Sanz, 1997), el cual quedará actualizado como se muestra en la ecuación (2.10).

$$\Delta w_{nj}(k+1) = w_{nj}(\mathrm{k}) + \Delta w_{nj}(k+1) \qquad (2.10)$$

Los algoritmos de aprendizaje son el conjunto de reglas para resolver el problema de la determinación del mejor valor de $\Delta w_{nj}(k)$ algunos métodos son:

-Aprendizaje Hebbeliano.

-Perceptrón unicapa.
-Perceptrón multicapa.

2.2.5 Aprendizaje Hebbeliano

Modelo de Aprendizaje Hebbeliano

El aprendizaje Hebbeliano también se conoce como asociador lineal, es un modelo que consta únicamente de una capa de neuronas lineales con la función identidad como función de activación, sus elementos se definen en las ecuaciones (2.10), (2.11) y (2.12).

$$W = (w_1\ w_2\ \dots\ w_m)^T \tag{2.10}$$

$$y = Wx \tag{2.11}$$

$$y_n = \sum_{j=1}^{m} w_{nj}\, x_j \tag{2.12}$$

El objetivo de éste modelo es que los pesos w_{nj} adquieran los valores que permitan que entradas semejantes produzcan salidas similares, lo cual es posible aplicando la regla de Hebb.

La regla de aprendizaje de Hebb

La regla de Hebb es el punto de partida de muchos algoritmos más complejos, para su entendimiento se proporciona en ésta investigación la definición completa brindado por Martín del Brío & Sanz en su libro “Redes neuronales y sistemas borrosos”.

Por lo general, se conoce como aprendizaje Hebbeliano a los tipos de aprendizaje que involucran el cambio de los pesos w_{nj} proporcional al producto de la entrada x_j por la salida y_n de la neurona.

$$\Delta w_{nj} = \varepsilon y_{nj} x_j \tag{2.13}$$

Siendo Ɛ el denominado ritmo de aprendizaje, que suele ser una cantidad entre 0 y 1. Ésta expresión puede considerarse la representación matemática del modelo de aprendizaje descrito por Hebb.

Consideremos el asociador lineal. La regla de Hebb que permite aprender a asociar "*p*" pares de entrada-salida $\{(x^{\mu}, k^{\mu})/1 \leq \mu \leq p\}$, ajustando sus pesos W de modo que ante un cierto patrón de entrada x^{μ} responda con k^{μ}, y ante entradas similares $(x^{\mu}+\varepsilon)$ responda con salidas también próximas $(k^{\mu}+\delta)$ con ε y δ cantidades pequeñas, se expresa en éste caso particular de la siguiente manera

$$\Delta w_{nj}^{\mu} = k_n^{\mu} x_j^{\mu} \tag{2.14}$$

Y por lo tanto

$$w_{nj}^{new} = w_{nj}^{old} + \Delta w_{nj}^{\mu} \tag{2.15}$$

Si los pesos de partida son nulos, el valor final de W para las *"p"* asociaciones será

$$W = k^1 x^{1K} + k^2 x^{2K} + \cdots + k^p x^{pK} \tag{2.16}$$

Empleando la regla de Hebb para el entrenamiento del asociador lineal, si los vectores de entrada $\{x^1, x^2, \ldots, x^p\}$ son ortonormales (ortogonales y de longitud unidad) se cumple

$$\begin{aligned} W x^{\mu} &= (k^1 x^{1K} + k^2 x^{2K} + \cdots + k^p x^{pK}) \cdot x^{\mu} \\ &= k^1 (x^{1K} \cdot x^{\mu}) + \cdots + k^p (x^{pK} \cdot x^{\mu}) = k^{\mu} \end{aligned} \tag{2.17}$$

Y por tanto, ante la entrada x^{μ} se produce a respuesta aprendida k^{μ}; es decir, la regla de Hebb ha conseguido en este caso que la red aprenda a realizar las asociaciones deseadas. El problema reside en que las condiciones son muy restrictivas, ya que para que las asociaciones sean correctas, los patrones de entrada deben ser ortonormales. Por ello, si la dimensión del espacio de entrada es *n*, solamente podrá aprender hasta *n* asociaciones. Para almacenar más pares de entrada-salida será preciso utilizar otras estrategias.

Eliminando la condición de ortogonalidad, se tienen (manteniendo el requisito de vectores de longitud 1)

$$Wx^{\mu} = k^{1}(x^{1K} \cdot x^{\mu}) + k^{2}(x^{2K} \cdot x^{\mu}) + \cdots + k^{p}(x^{pK} \cdot x^{\mu}) \\ = k^{\mu} + \sum_{v \neq \mu} k^{v}(x^{vK} x^{\mu}) = k^{\mu} + \delta \tag{2.18}$$

Expresión denominada expansión señal-ruido, pues proporciona la salida deseada más un término adicional, que interpretamos como el ruido superpuesto a la señal.

Regla del error cuadrático y la matriz seudoinversa.

Usualmente se deducen los algoritmos de aprendizaje con el objetivo de generar los pesos ideales para la red neuronal, dichos algoritmos buscan obtener el menor error posible, el error de una red neuronal se puede conocer utilizando diversas medidas, una de las más comunes es el error cuadrático medio, el cual aplicado a la red neuronal anterior se define de la siguiente manera

$$E\{w_{nj}\} = \frac{1}{p}\sum_{\mu=1}^{p}(Wx^{\mu} - y_n)^2 \tag{2.19}$$

Donde si además definimos los vectores x^{μ} e y_n de forma matricial y aplicamos el concepto de norma cuadrática

$$X = \begin{bmatrix} x^{1K} \\ x^{2K} \\ \vdots \\ x^{pK} \end{bmatrix} \tag{2.20}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.21}$$

La ecuación (2.19) queda de la forma

$$E\{w_{nj}\} = \frac{1}{p}\|XW - y\|^2 \tag{2.22}$$

Para minimizar el error es necesario aplicar el operador gradiente a la ecuación (2.22) con lo que se obtiene

$$\nabla E\{w_{nj}\} = \frac{2}{p} X^T (XW - y) = 0 \tag{2.23}$$

Y el valor ideal de W viene dado por

$$X^T XW = X^T y$$
$$W = (X^T X)^{-1} X^T y \tag{2.24}$$

(Abu-Mustafa, 2012)

Donde el producto $(X^T X)^{-1} X^T$ se conoce como la matriz seudoinversa (X^+) de X. Al utilizar la matriz seudoinversa se consiguen almacenar hasta *n* vectores linealmente independientes en vez de solo *n* vectores ortogonales.

2.2.6 El perceptrón unicapa

Modelo de aprendizaje del perceptrón unicapa

Rosenblatt fue el autor de la propuesta del perceptrón, el cual es la forma más simplificada de una red neuronal, se utiliza para la clasificación de patrones linealmente separables (patrones que se localizan en los lados opuestos de un hiperplano, también conocido como dicotomía) usa la función escalón como función de activación y es una red unidireccional sin capas ocultas. La ecuación para *n* neuronas de entrada y *m* de salida para una red con ésta arquitectura es

$$y_i(t) = H(\sum_{j=1}^{n} w_{ij} x_j - \theta_i), \forall\, i, 1 \leq i \leq m \tag{2.25}$$

Si las variables de entrada al perceptrón están originadas por dos clases linealmente separables, sea X1 el subconjunto de vectores de entrenamiento x1(1), x1(2),… tal que pertenecen a C_1, y sea X2 el subconjunto de vectores de entrenamiento x2(1), x2(2),… tal que pertenecen a C_2; la unión de X1 y X2 es el

conjunto de entrenamiento completo. Dados los conjuntos de vectores X1 y X2 para el entrenamiento del clasificador, el proceso de entrenamiento involucra el ajuste del vector de pesos W de tal modo que las dos clases C_1 y C_2 sean separables por un hiperplano. Entonces existe un vector de pesos W tal que:

-Wx ≥ 0 para cualquier vector de entrada x que pertenezca a la clase C_1.
-Wx < 0 para cualquier vector de entrada x que pertenezca a la clase C_2.

(Martín del Brío & Sanz, 1997).

La regla de aprendizaje del perceptrón unicapa

Existe un tipo de algoritmos que modifican los pesos de la red respecto a la salida que se requiere a través de minimizar el error actual de la red, los algoritmos que funcionan de ésta manera se conocen “algoritmos por corrección de errores”.

Para explicar cómo funciona la regla de aprendizaje del perceptrón unicapa el siguiente desarrollo ésta basado en el libro de Martín del Brío & Sanz: Redes neuronales y sistemas borrosos.

Sea un conjunto *p* de patrones x^{μ}, μ=1,…,p, con sus salidas deseadas t^{μ}, ante la presentación del patrón μ-ésimo, si la respuesta que proporciona el perceptrón es correcta, no se actualizan los pesos; si es incorrecta, los modificaremos según la regla

$$\Delta w_{ij}^{\mu}(t) = \varepsilon(t_i^{\mu} - y_i^{\mu})x_j^{\mu} \qquad (2.26)$$

Que es la forma habitual de expresar la regla del perceptrón; el parámetro ε no debe ser demasiado pequeño, ya que implicaría un aprendizaje muy lento, si es demasiado grande puede introducir variaciones en los pesos excesivamente grandes que conduzcan oscilaciones en el entrenamiento.

La operación neuronal funciona de la siguiente manera:

$$y_i^{\mu}(t) = signo\left(\sum_{j=1}^{n} w_{ij}^{\mu}\, x_j^{\mu} - \theta_i^{\mu}\right) = signo(w_i \cdot x^{\mu}) = signo(\|w_i\|\, \|x^{\mu}\| \cos(\rho))$$

Que representa la neurona *i* cuyo vector de pesos es w_i con un patrón de entrada x^{μ}, y salida objetivo t_i^{μ}.

2.3 Robótica y robots humanoides

En la sociedad actual hay una creciente necesidad de realizar tareas de manera eficiente y precisa, existen tareas donde la acción humana es difícil, arriesgada o incluso imposible. Para ejecutarlas es necesaria la presencia de dispositivos (robots) que las realicen sin riesgo alguno. Pfeifer cita un ejemplo de tarea que puede ser perjudicial para la salud humana: el censado de parámetros en volcanes; para resolver el problema creó prototipos de robots utilizando las piezas de Lego (Pfeifer, 2006).

2.3.1 Definición de Robótica

La robótica es el área que se encarga del desarrollo de éste tipo de dispositivos. Multidisciplinar y en constante evolución, busca el desarrollo y la integración de técnicas y algoritmos para la construcción de máquinas denominadas robots.

Pio define la robótica como

"La conexión inteligente entre percepción y acción. Generar desarrollos en robótica significa estudiar, diseñar e implementar sistemas o dispositivos que posean cierto grado de percepción y cierta inteligencia, que sean útiles en la realización de una tarea en particular, predefinida o no, que implique la interacción física entre el sistema (o dispositivo) y el medio donde se realiza la tarea" (Pio, 2006).

La Robótica comprende el estudio de la ingeniería mecánica, la ingeniería eléctrica y la inteligencia artificial, entre otros temas. Hoy en día existen robots en diferentes ámbitos de la sociedad: se encuentran los que desempeñan alguna interacción social, los que tienen el propósito de la investigación científica y la educación e incluso los trabajadores que se instalaron en las fábricas y fueron responsables de la "Segunda Revolución Industrial" .

La automatización industrial y la robótica son dos campos estrechamente relacionadas. Se considera a la robótica como una forma de automatización industrial que utiliza tecnologías de robots en la producción y control. Una definición formal sería "una ciencia de la ingeniería aplicada, tomada como una combinación de máquinas operativas y la ciencia de la computación". Por lo tanto, el principal instrumento utilizado en la robótica es el robot. (Redel, 2014)

2.3.2 Definición de Robot

Una de las mayores fantasías del hombre es la construcción de una máquina con inteligencia artificial capaz de actuar y pensar como él. La palabra "robot" se origina de la palabra checa Robotnik, que significa "siervo". El término fue utilizado inicialmente por Karel Capek en 1923, época en que la idea de un "hombre mecánico" parecía pertenecer a cualquier obra de ciencia ficción.

No es solamente un deseo del hombre moderno construir robots; algunos hechos históricos demuestran que la idea no es nueva, existen relatos sobre algunas animaciones mecánicas hechas por Leonardo da Vinci, como el león animado de la figura 2.10 y sus esfuerzos para crear máquinas que reprodujeran el vuelo de las aves. Sin embargo, estos dispositivos eran muy limitados ya que no podían realizar más de una tarea o un conjunto reducido de ellas (Rosário, 2007).

Figura 2.10 Juguete automático de cuerda hecho por Leonardo da Vinci (André, 2014).

La Robotics Industries Association (RIA) propone su definición de robótica de la de la siguiente manera: "Un robot es un manipulador reprogramable y multifuncional diseñado para manejar los materiales, piezas, herramientas o dispositivos especiales mediante movimientos programados para llevar a cabo una variedad de tareas" (Redel, 2014) .

Otros conceptos importantes que ayudan a definir un robot son la capacidad de adaptación y versatilidad. La diferencia entre un ordenador y un robot se distingue en que el robot posee medios suficientes para poder interactuar con el mundo. El ordenador no se inicia sin la operación humana. Por lo tanto, un robot es considerado también un mecanismo inteligente que funciona de forma independiente (Murphy, 2000) .

2.3.3 Antecedentes históricos

El primer autómata del que se tiene conocimiento data del año 1500 a.c., siendo una estatua del rey de Etiopía que emite sonidos al amanecer cuando los rayos del sol la iluminaban. Entre el 300 y el 270 a.c se construyó la denominada clepsidra o reloj de agua, la cual se muestra en la figura 2.11a, además de un órgano que emitía sonidos por impulsos de agua. En los años siguientes Herón de Alejandría realizó un tratado de autómatas donde se describen múltiples juguetes como los de la figura 2.11b capaces de moverse por si solos de forma repetida (Bolaños, 2013).

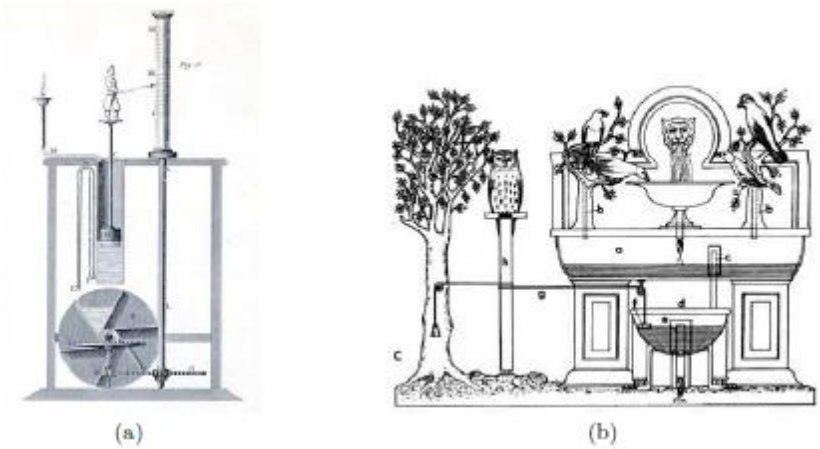


Figura 2.11 (a) Clepsidra o Reloj de agua (Taufiqurrakhman, 2014)*, (b) Aves de Hero de Alejandría que gorjean y beben* (Sánchez F. M., 2007)*.*

Durante la Edad Media y el Renacimiento se generaron esquemas para la construcción de sistemas mecánicos antropomorfos. Leonardo Da Vinci creó un león mecánico en el año 1500 que se abría del pecho y mostraba el escudo de armas del rey, también diseñó un dispositivo mecánico similar a un caballero mecánico (que nunca fue construido), capaz de incorporarse, agitar los brazos, mover la cabeza (a través de un cuello flexible) y apretar la mandíbula abierta, la figura 2.12a muestra una maqueta del caballero.

Figura 2.12 (a) Caballero mecánico de Leonardo Da Vinci (Torres, 2014)*, (b) Pato de Jacques de Vaucanson* (Tamburro, 2014)*.*

En el siglo XVIII, se crearon autómatas que intentaban replicar movimientos del ser humano realizando una serie de movimientos simples. Estas máquinas fueron asumiendo la tarea de ayudar al hombre, razón por la cual acabaron repercutiendo en su concepción del mundo y de los seres automáticos. Uno de los más famosos y completos constructores de androides automatizados, Jacques de Vaucanson realizó un autómata autista capaz de tocar melodías siguiendo una partitura, además del pato mecánico de la figura 2.12b que contiene más de 400 piezas móviles, capaz de graznar, comer de la mano de una persona, y completar de forma total la digestión (Bolaños, 2013).

En los años 50, el neurofisiólogo Walter (Walter, 1963) realizó una investigación pionera sobre los robots móviles autónomos, su objetivo fue estudiar las bases de las acciones simples generadas por los reflejos y poner a prueba su teoría de que

el comportamiento complejo puede surgir de las conexiones neuronales simples. Dicho científico logro crear robots en forma de tortuga como el mostrado en la figura 2.13, a los que llamó "Elsie" y "Elmer", a través de los cuales generó una gran influencia en el nacimiento de la ciencia de la cibernética. Asimismo escribió el libro "The Living Brain", el cual tuvo una gran repercusión en la época.

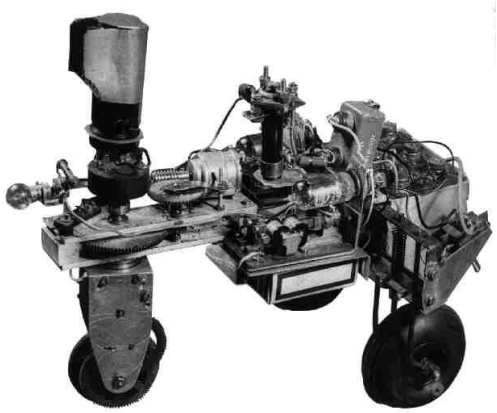


Figura 2.13 Fotografía original de Elsie, apodada la "tortuga" (Pieruccini, 2014).

El uso de la robótica como herramienta para el aprendizaje en la educación básica comenzó a finales del 60 por Papert, quien inspirado por las tortugas de Walter construyó su propio robot en forma de tortuga y desarrolló el lenguaje Logo para controlarlo (Papert, 1988).

Los primeros robots se comenzaron a utilizar con el objetivo de sustituir a los seres humanos en tareas que involucraban riesgo de muerte o condiciones desagradables como son los altos niveles de calor, ruido, presencia de gases tóxicos o esfuerzo físico extremo asimismo en tareas tediosas, monótonas o que no podía llevar a cabo. Dos tendencias en los últimos 20 años garantizaron la evolución de los robots; el extraordinario avance en la industria tecnológica y la necesidad de reducir el número de trabajadores asociada a los costos de producción.

La idea de la construcción de robots comenzó a volverse fuerte a principios del siglo XX debido a la necesidad de aumentar la productividad industrial y mejorar la calidad de los productos. En esa época el robot industrial encontró sus primeras aplicaciones; George Devol el cual entre sus primeros clientes tuvo a la General Motors, puede ser considerado el padre de la robótica industrial por su robot de

gran éxito en la época el "Unimate", figura 2.14, lanzado en 1962, cuya finalidad era automatizar algunas tareas industriales (Rosário, 2007).

Figura 2.14 Robot Unimate (Betts, 2014).

En 1966 la empresa Artificial Intelligence Lab de Stanford U.S.A desarrolló el robot llamado "shakey" que fue el primer robot capaz de reaccionar a sus propias acciones, en 1969 fue creado el primer brazo robótico, y en 1970 el primer robot capaz de seguir una línea, conocido como " Stanford Cart".

La compañía Lego lanzó en 1986 robots construidos utilizando piezas de Lego programados en el lenguaje Logo, en 1996 un robot se encargó de recopilar información acerca de los océanos y la vida marina, posteriormente en 1998 fue introducida en el mercado la serie de Lego Mindstorms que permite construir robots reprogramables de una forma relativamente sencilla.

Los robots industriales se clasifican cronológicamente en tres generaciones: la primera generación está conformada por los robots no programables, integrados por dispositivos que actúan como esclavos mecánicos del hombre; la segunda generación se compone por los robots programados para ejecutar sus tareas (movimientos) por medio de servomecanismos, la tercera generación; se integra por robots programables que trabajan en función del ambiente en el que se encuentran y cuentan con la capacidad de interactuar con él (Capelli, 2002).

2.3.4 Robot Humanoide

Básicamente los robots se pueden dividir en dos grupos: los manipuladores y los móviles (Craig, 1986). Los robots móviles como su nombre sugiere poseen la capacidad de desplazarse a través de su espacio de trabajo, esto les permite tener una gama de aplicaciones sumamente amplía. Los robots móviles de dos piernas (bípedos) pueden funcionar como la base para la construcción de robots humanoides, los cuales presentan semejanzas físicas con el cuerpo humano, ya que en algunos casos posen partes similares a las piernas, los brazos, la cabeza u otros rasgos semejantes.

Por lo tanto, un robot humanoide es aquel robot que presenta semejanzas físicas con el ser humano, algunas características que pueden presentar los robots humanoides son las siguientes:

• Locomoción bípeda estable;

• Brazos, que en algunos casos terminan en manos, pinzas articuladas o algún otro accesorio que depende del propósito del robot;

• Un cuerpo, el cual puede albergar diversos sistemas;

• Una cabeza con sensores de visión y sonido, sensores para orientarse y desplazarse por el entorno de trabajo;

• Autonomía en la toma decisiones;

• Desempeño de sus funciones en ambientes estructurados abarcando un amplio espectro de posibilidades (Fernández, 2011).

Con el fin de construir robots más similares a los seres humanos en el futuro es necesario aún el desarrollo y la evolución de diversas tecnologías. No basta construir robots que caminan, que tocan un instrumento o danzan, también es necesario el desarrollo de sistemas robóticos, es decir, sistemas que reproduzcan otros aspectos complejos con los que la naturaleza ha dotado al hombre:

• El Razonamiento.

• El habla;

• Articulaciones sofisticadas, como aquellas con las que cuentan las manos, los brazos y las piernas,

• Empatía emocional;

Como se puede observar en la figura 2.15 en la Universidad de Colorado en los Estados Unidos se desarrollaron manos mecánicas con juntas similares a las del ser humano (Souza, 2014).

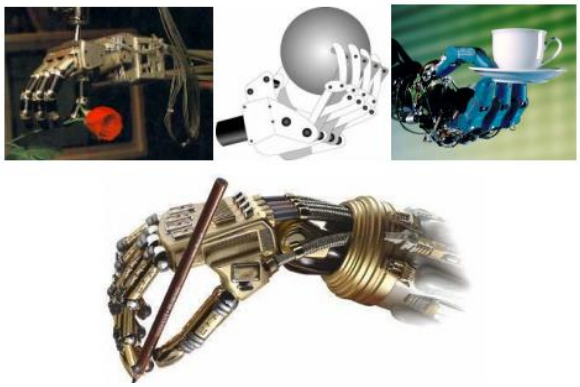


Figura 2.15 Mano mecánica desarrollada en la Universidad de Colorado en U.S.A (Fernández, 2011).

La sociedad se enfrenta a una creciente necesidad de máquinas (robots) que auxilien o sustituyan al hombre en el desarrollo de diversas tareas en entornos como oficinas, casas, hospitales y espacios abiertos, dichos robots deben ser capaces de realizar éstas tareas en entornos "humanos", la cual es una razón primordial por la que se desarrollan robots que posean una forma similar a la fisiología humana (Huang, Li, Nakamura, & Tanie, 2001).

Para realizar un conjunto de tareas no especificadas previamente en un entorno prediseñado para seres humanos, el robot humanoide se convierte en la opción ideal, los robot humanoides tienen una mayor posibilidad de interactuar con éste tipo de entorno al contar con diversas características del cuerpo humano.

Por ejemplo, un robot humanoide puede subir escaleras, abrir puertas, recoger un objeto, etc. Entre el abanico de las posibles aplicaciones que un robot humanoide podrá desarrollar en el futuro se encuentran: el mantenimiento de las plantas industriales, el trabajo cooperativo en el espacio abierto, la seguridad, las labores del hogar, los rescates y la tele operación de máquinas de construcción.

Uno de los problemas de la robótica móvil es la navegación, en un robot móvil puede ser descrita como el proceso utilizado por él mismo para moverse en un entorno de trabajo que generalmente contiene obstáculos, en el cual se busca llegar desde una posición y orientación inicial a una posición y orientación final. En éste proceso el robot debe recorrer una trayectoria obedeciendo sus restricciones cinemáticas y dinámicas, esquivando obstáculos cuando sea necesario, siendo capaz de conocer constantemente su posición y orientación en el ambiente de trabajo. En el caso de los robots con piernas ésta tarea se vuelve aún más compleja, dado que existe la posibilidad de perder el equilibrio y caer, por lo que los investigadores estudian también éste proceso. Una característica deseable en la navegación de dichos robots es el funcionamiento con el menor consumo de energía mientras se desplazan lo más rápido posible (Nogueira, 2005).

2.4 Ejemplos de robots humanoides

El robot Cog

Uno de los robots humanoides más famosos construidos por el MIT es el robot “Cog” (MIT, 2014), reproduce un torso humano con cabeza como se muestra en la figura 2.16. Éste robot tiene 21 grados de libertad que tienen la intención de aproximar sus movimientos a los movimientos humanos, está provisto de una amplia variedad de sistemas sensoriales, integrando señales visuales, auditivas y táctiles.

Figura 2.16 O Robot Cog (MIT, 2014).

Detrás del control computacional del Cog hay una red heterogénea de diversos tipos de procesadores, los cuales operan a distintos niveles de una jerarquía de control basada en microcontroladores enfocados en gobernar desde las articulaciones hasta las redes de DSPs (procesado de señal digital) empleadas para el procesamiento audiovisual del robot. El " cerebro" del Cog ha sido objeto de varias revisiones, la primera versión estaba constituida por una red de microcontroladores Motorola 68332 de 16 [MHz], conectados a memorias RAM de doble puerto en placas personalizadas.

El sistema visual del Cog fue diseñado para imitar algunas de las capacidades del sentido humano de la vista, incluyendo la visión binocular y la percepción de la profundidad del espacio. Cada ojo puede girar alrededor de eje vertical y del eje horizontal.

El robot Cog fue utilizado en experimentos de atención compartida, interacción social (imitación de comportamiento, selección de objetivos, etc.), reproducción de los movimientos del brazo, el torso y la cabeza humanos. También fue utilizado como una plataforma para verificar teorías sobre la mente y en experimentos relacionados con el área cognitiva humana (Scassellati, 2001).

El Robonauta 2

La NASA desarrolló un humanoide metálico al que llamó Robonauta 2 (figura 2.17), con el objetivo de ser utilizado en el espacio y poder reparar satélites y estaciones espaciales. A diferencia de los robots construidos hasta ahora, el Robonauta 2 tiene el tamaño de un astronauta con traje espacial, tiene brazos y manos más flexibles que los humanos, e inclusive sus muñecas tienen una capacidad de giro mayor con el objetivo de manipular las herramientas de trabajo. Su cuerpo robusto y tiene una cabeza equipada con dos cámaras que funcionan como ojos (NASA, 2014).

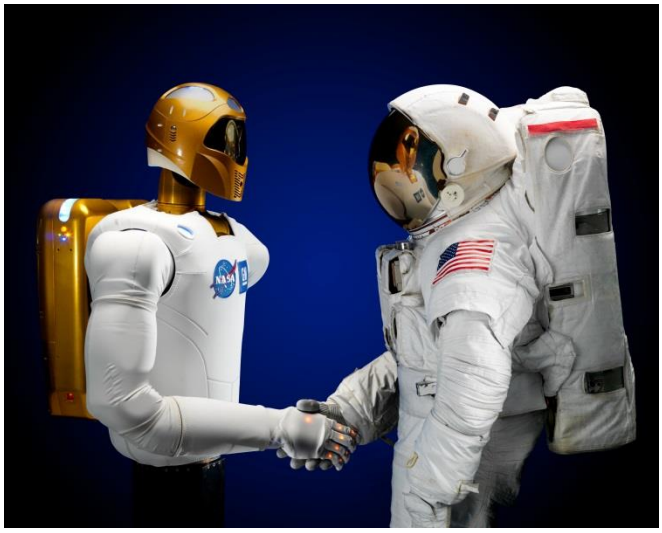


Figura 2.17 Robot humanoide Robonauta 2 (NASA, 2014).

El robonauta 2 ha sido diseñado para parecer una persona, se encuentra actualmente operando en la Estación Espacial Internacional. Puede usar las mismas herramientas que un astronauta, tiene dos modos de funcionamiento; el primero le permite ser controlado a distancia, el segundo le permite recibir una instrucción y automáticamente deducir como llevar a cabo la tarea solicitada (NASA, 2014).

El robot Honda Asimo

El robot Honda Asimo es el resultado de toda una línea de desarrollo de robots humanoides de la compañía Honda (Hirai, 1997), comenzando con la línea de E0-E6, seguida de la P1- P3 que culminó con el Asimo, ilustrado en la figura 2.18. Basándose en la inspiración de la ciencia ficción, la concepción del Asimo está orientada a la construcción de un robot capaz de convivir con los seres humanos en una vida social. Tales robots deben ser capaces de desplazarse en los ambientes domésticos esquivando objetos y subiendo escaleras.

Figura 2.18 O Robot Asimo da Honda (Honda, 2014).

La investigación de la empresa Honda estuvo enfocada en la capacidad de caminar de los robots bípedos, contempló el análisis de diversos factores, inicialmente se estudió el posicionamiento de las articulaciones en las patas del robot y su relación con la estructura de las articulaciones en los seres humanos, posteriormente se analizó el grado de movimiento de las articulaciones, a continuación se estudió el tamaño ideal para los segmentos de las patas y su

impacto en el peso y el centro de gravedad del robot, finalmente se determinó el torque necesario en las articulaciones durante la marcha y los diversos sensores necesarios para llevar a cabo éste proceso.

Para el impacto de las fuerzas durante el andar se consideró que los humanos poseen diversos recursos para la amortiguación del impacto, como son los tejidos, la curvatura de los arcos de los pies y el movimiento de sus dedos, por tanto en el Asimo se utilizaron materiales especiales que permiten la absorción de impactos para conseguir un andar estable, influyendo en que el robot no cayera al ser empujado, ni cuando se encontrara en superficies de apoyo inclinadas o accidentadas (Hirai, 1997).

2.5 Robot Humanoide RH-2

El RH-2 es un proyecto de vanguardia que busca acelerar la inevitable introducción de los robots en la vida cotidiana, éste tipo de proyectos impulsan cambios de notoria trascendencia en el mundo. El proyecto de robots humanoides RH de la Universidad Carlos III de Madrid (UC3M) consta de dos prototipos ya construidos: el RH-0 en 2004 y su versión mejorada el RH-1 en 2007, mostrados en la figura 2.19.

Ambos han sido desarrollados tomando como modelos los prototipos más avanzados existentes en aquel momento, como eran el ASIMO de Honda y el HRP-2P de Kawada el cual quedado obsoleto con el lanzamiento del modelo HRP-3P.

El RH-1 suponía simplemente la modificación o sustitución de algunos elementos hardware y rediseños mecánicos del RH-0, excepto algunos cambios estructurales se mantuvo el tamaño y los grados de libertad (Fernández, 2011).

Figura 2.19 A la izquierda el RH y a la derecha el RH1 (Fernández, 2011)

Para el desarrollo del RH-02 se partió desde cero cuando se inició el proyecto, el RH-02 es un proyecto novedoso que pretende una mayor cantidad de movimientos en las articulaciones, además de que se introduce un nuevo grado de libertad en el tronco del robot, que permite controlar mejor el balanceo. El robot

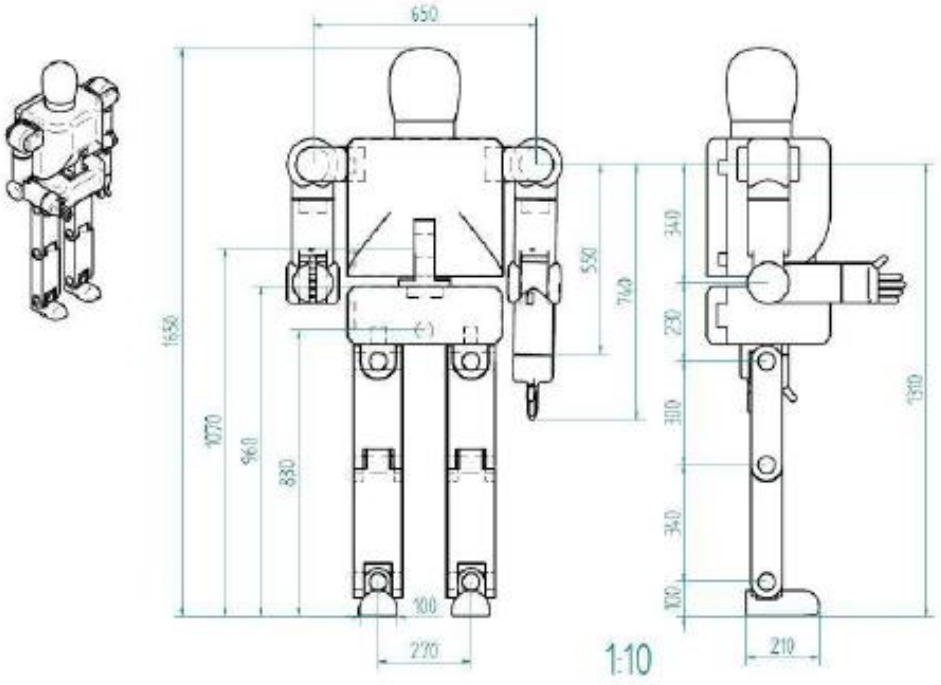
RH-2 dispone de 24 grados de libertad y se distribuyen de la siguiente forma por sus extremidades:

• **Piernas:** Cada una dispone de 6 GDL's distribuidos entre el tobillo, la rodilla y la cadera. La cadera posee 3 GDL's, uno en el plano sagital, otro en el frontal y el tercero en el plano transversal. La rodilla tiene 1 GDL en el plano sagital, el tobillo posee 2 GDL's, uno en el plano sagital para adaptar el pie al suelo, y otro en el plano frontal que permite el balanceo en combinación con el de la cadera para mantener el equilibrio.

• **Brazos:** Dispone cada uno de 5 GDL's distribuidos entre el hombro, el codo y la muñeca. En el hombro existen 2 GDL's en los planos sagital y frontal, en el codo hay 2 GDL's en plano frontal y transversal. En la muñeca existe únicamente 1 GDL en el plano transversal. Ésta distribución permite la manipulación de objetos, e intenta que la movilidad sea la de un brazo humano.

• **Tronco:** El tronco posee 2 GDL's en el plano transversal, que le permite el giro en ese plano sin tener que mover las piernas y otro en plano frontal, que le permite regular su inclinación

Se estima un peso de 60 Kg y una velocidad de 1 Km/h durante la caminata. Se calcula que podrá transportar objetos de 2 Kg de peso e incluso sentarse. Su altura aumenta de 1.2 m a 1.65 m, dotando al robot de un tamaño más acorde al de un humano, como se ve na figura 2.20.



.

Figura 2.20 Medidas del RH2 (Fernández, 2011).

La estructura mecánica del robot ha sido diseñada mediante el programa de diseño 3D, Catia. El programa está desarrollado para proporcionar apoyo desde la concepción del diseño, hasta la producción y el análisis de productos. Inicialmente fue desarrollado para servir a la industria aeronáutica, pero se ha hecho un gran hincapié en el manejo de superficies complejas.

El montaje del robot en su totalidad es una tarea compleja y es difícil que todo funcione perfectamente. Por eso, primero se realizó la parte inferior del RH-2 (las piernas y la cadera), que es la responsable de que el robot pueda andar, y una vez que se consiga una caminata estable se dará paso a la segunda fase, la cual consistirá en la que se acoplará del torso, los brazos y la cabeza (Fernández, 2011).

Capítulo 3: Metodología

3.1 Selección del software para el entrenamiento de la red neuronal

MATLAB fue el programa seleccionado para realizar el diseño ya que posee un asistente de generación de redes neuronales, a diferencia de lenguajes de programación como C++ o Java MATLAB posee una interfaz gráfica especializada para dichos desarrollos.

3.2 Datos

Los datos de entrenamiento generados por el RoboticsLab de la Universidad Carlos III de Madrid fueron generados utilizando el sensor MTi-28A53G53 del fabricante Xsens. El experimento consistió en colocar al sujeto de prueba el sensor en la cabeza y pedirle que diera un paso completo.

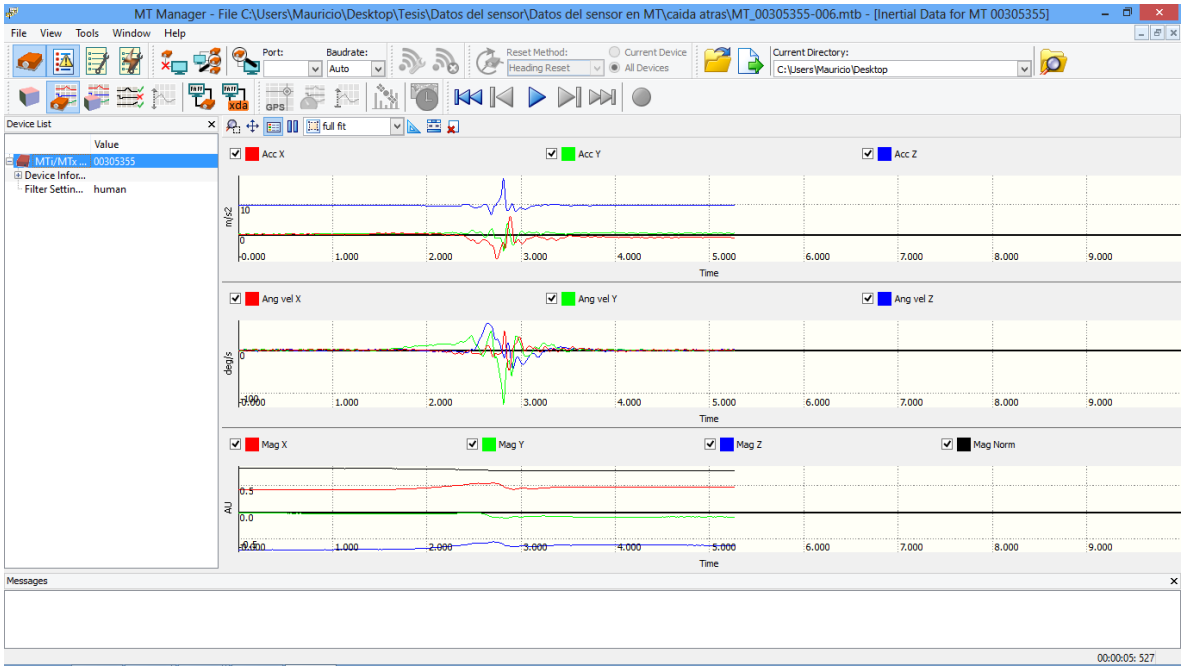


Figura 3.1 Datos del sensor en el programa nativo.

3.2.1 Entrenamiento

Los datos de entrenamiento serán aquellos que los diversos sujetos de prueba generen al colocar el sensor en su cabeza y caminar, posteriormente cuando el

robot sea capaz de moverse por sí mismo se agregaran todos los datos de los sujetos de prueba y los que el RH-2 genere.

3.2.2 Objetivo

Los datos objetivo son los generados al simular una caída al frente por parte de los sujetos de prueba.

3.3 Selección de modelo

La predicción es un problema en el que los valores pasados de una o más series de tiempo predicen valores futuros, para éste tipo de problemas el programa MATLAB versión 2013b brinda tres modelos diferentes posibles.

- No lineal Entrada-Salida: Predice series y(t) dados valores pasados de series x(t).
- No lineal Autoregresivo (NAR): Predice series y(t) dados “d” valores pasados de y(t).
- No lineal Autoregresivo con Entrada Exógena (NARX): Predice series y(t) dados “d” valores pasados de y(t) y otra serie x(t).

Dadas las múltiples combinaciones posibles de parámetros en cada estructura de red neuronal combinada con la versatilidad del asistente de creación de redes neuronales, se determinará cual es el mejor modelo posible de red neuronal a través de entrenar los 3 tipos diferentes de arquitectura, se emplearan los mismos parámetros para los tres casos y se realizará un análisis comparativo de las respuestas de los mismos.

Los parámetros de entrenamiento serán los establecidos por defecto en el programa MATLAB:

- Una capa oculta de 10 neuronas con la función tanh(x) como función de activación.
- Un retraso de dos valores.
- Se utilizaran 70% de los datos para el entrenamiento de la red; la red es ajustada de acuerdo al error calculado en éstos datos.
- 15% de los datos se emplearán para validación; se utilizan para medir la capacidad de generalización, el entrenamiento se detiene cuando la generalización deja de mejorar.

- Para la evaluación se emplearan el 15% de los datos; proporcionan una medida independiente del desempeño durante y después del entrenamiento.
- La regla de entrenamiento será la propuesta por el algoritmo de Levenberg-Marquard.
- La función de error estará definida por el Minimun Squared Error.

En la ventana de comandos el asistente de generación de redes neuronales se accede a través del comando “ntstool”.

3.3.1 Tratamiento de datos

Los datos de entrenamiento fueron importados al programa MATLAB desde el software MT Manager que el proveedor brinda para el manejo del sensor MTi-28A53G53, los offsets generados por el software fueron eliminados, dichas series de intervalos de 5 segundos fueron acopladas una a continuación de la otra, la gráfica generada por la caminata queda como se muestra en la figura 3.2, la cual representa el conjunto de datos con los que se dispone para entrenar la red neuronal.

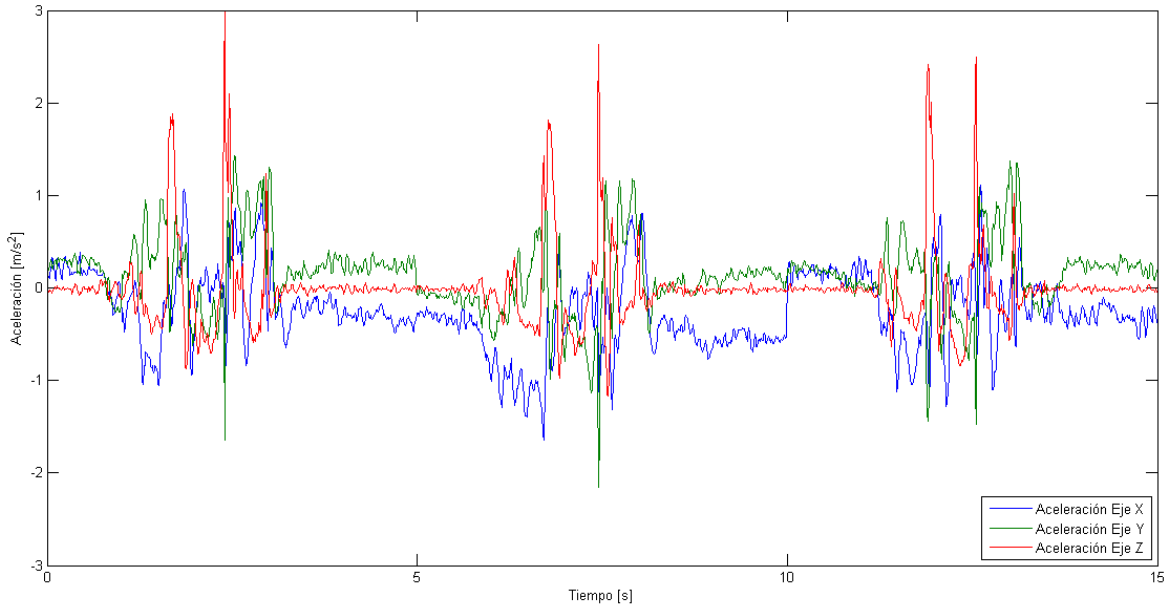


Figura 3.2 Gráfica de las aceleraciones obtenidas en el sensor generadas después de tres pasos simples.

Seguir éste acoplamiento de datos permitirá conjuntar las futuras muestras generadas por los pasos de los distintos sujetos de prueba que se empleen para el experimento.

Los datos generados por la caída fueron acoplados de la misma manera que los valores obtenidos en la caminata, la gráfica resultante se muestra en la figura 3.3.

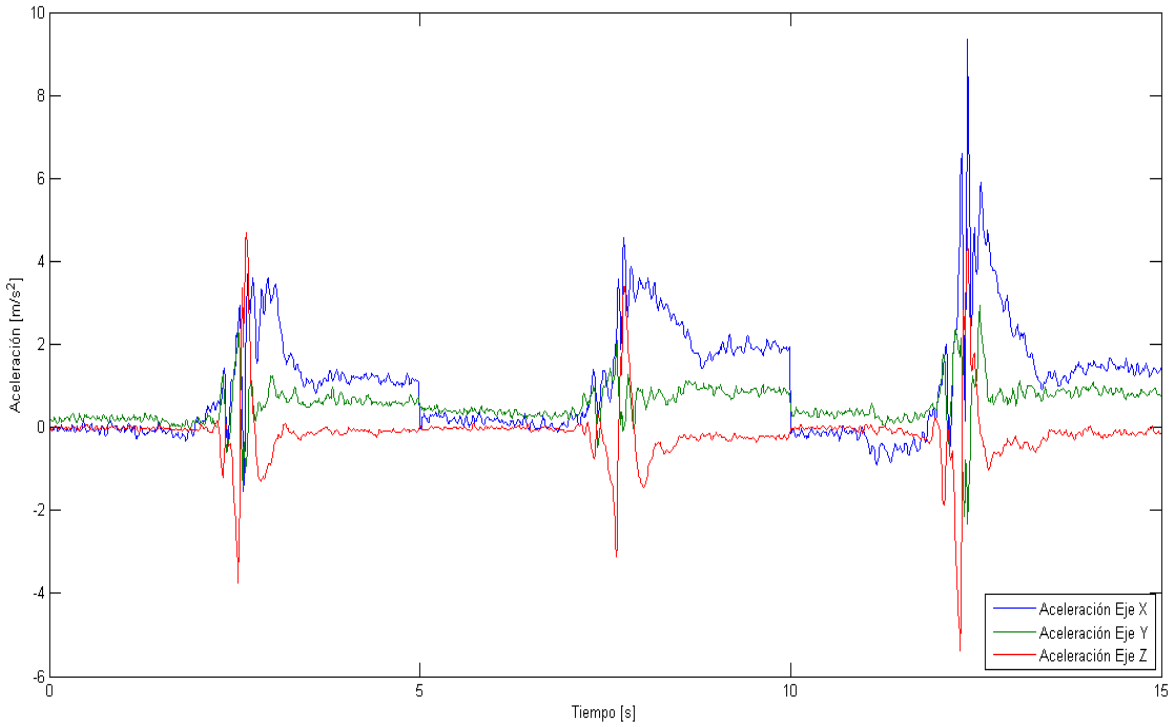


Figura 3.3 Gráfica de las aceleraciones obtenidas en el sensor generadas después de tres simulaciones de caída.

3.3.2 Desempeño y Selección de arquitectura de red neuronal

Para los datos de entrada y datos objetivo de la figura 3.2 y 3.3 empleando los parámetros de entrenamiento predefinidos en la sección 3.3 el desempeño fue el siguiente:

Arquitectura no lineal Entrada-Salida

Ésta arquitectura suele utilizarse para predecir parámetros cuando los valores pasados de y(t) no estarán disponibles durante el desarrollo real de los eventos, de los tres tipos de arquitectura de red que proporciona el asistente de MATLAB es el único que no es retroalimentado, su esquema se muestra en la figura 3.4.

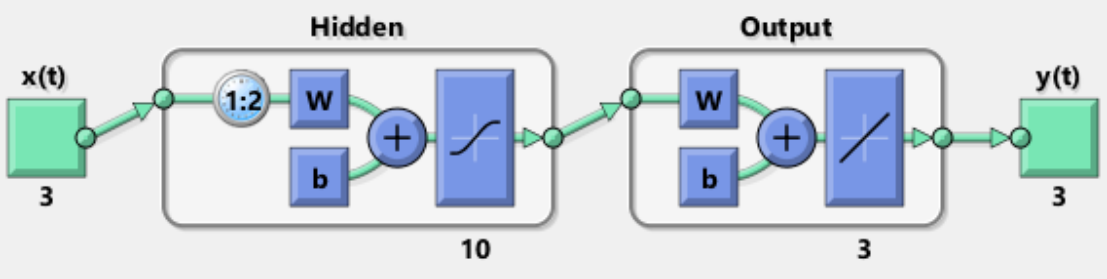


Figura 3.4 Esquema de la red neuronal No lineal Entrada-Salida.

La respuesta del modelo se muestra en la gráfica 3.5

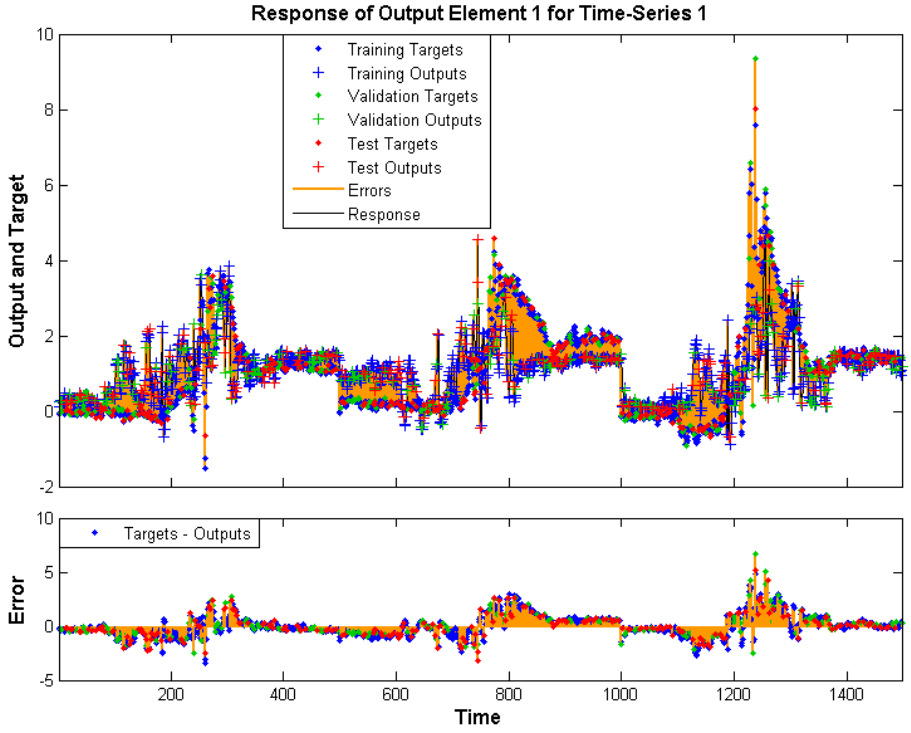


Figura 3.5 Gráfica de la respuesta al entrenamiento de la arquitectura No lineal Entrada-Salida.

En la figura 3.6 se observa una densidad considerable de líneas amarillas, las cuales representan el error de estimación, el error toma valores superiores al 5%, se observa que la mayor parte de los puntos representa un error asociado.

El mejor desempeño de validación viene dado por la gráfica 3.6.

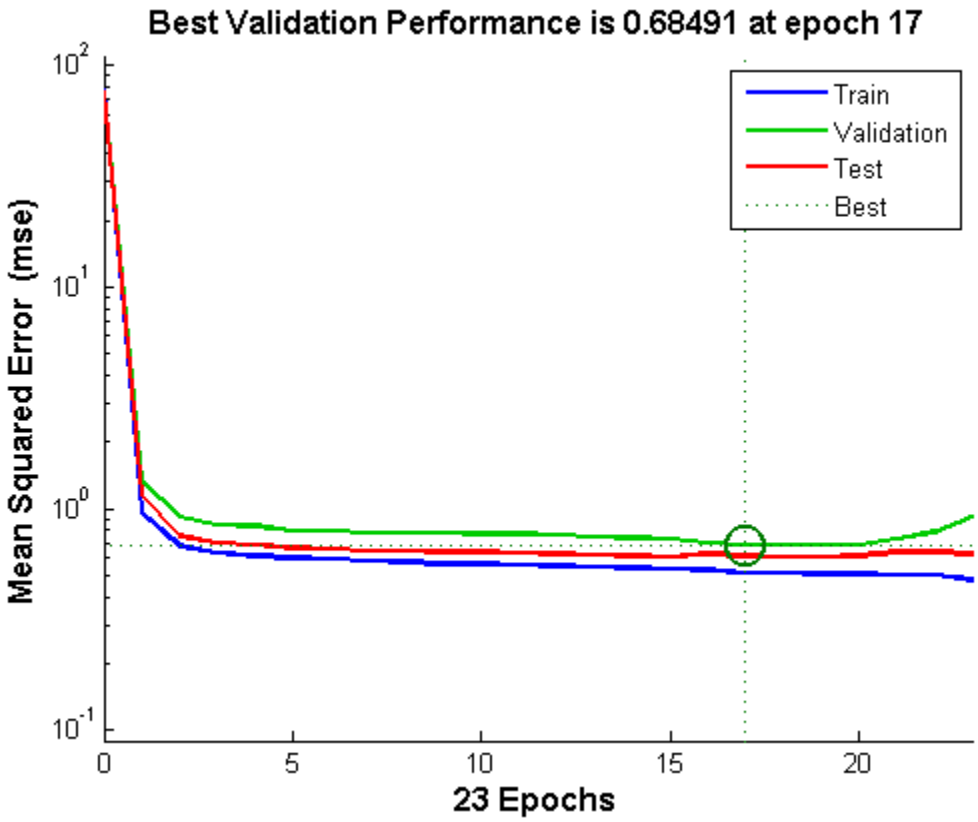


Figura 3.6 Mejor Desempeño de Validación de la arquitectura No lineal Entrada-Salida.

El mejor valor que se pudo obtener para el Mean Squared Error ocurrió en el ciclo de evaluación computacional número 27, el valor obtenido fue 0.68491, éste valor muestra que no se pudo alcanzar una generalización aceptable por parte de la red ya que es considerablemente mayor que cero.

La regresión de las variables se muestra en la gráfica 3.7.

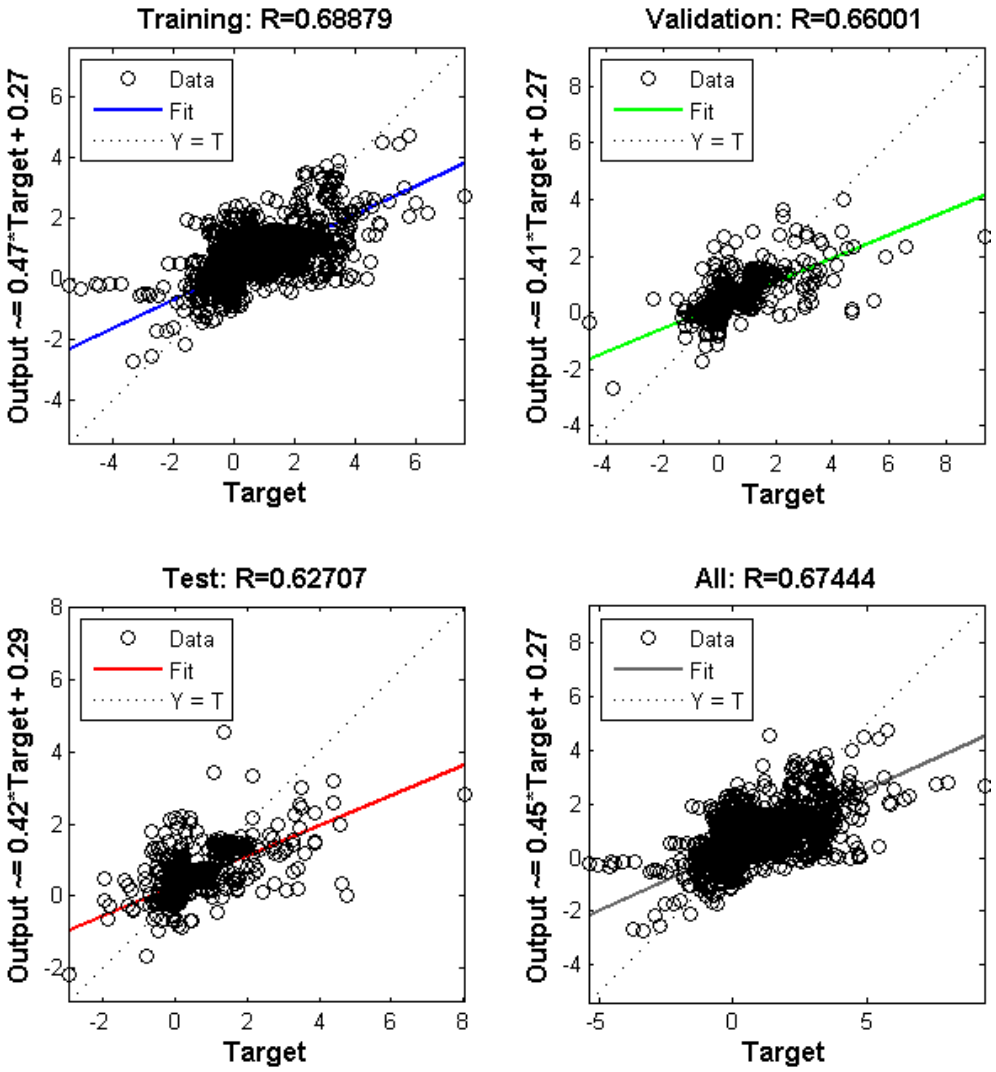


Figura 3.7 Curvas de regresión de la arquitectura No lineal Entrada-Salida.

Como se puede observar la curva Y=T es bastante diferente a la curva de ajuste generada por la red neuronal, esto demuestra su poca capacidad de predicción, además los puntos en forma de nube sugieren que la red no generalizó adecuadamente.

En la gráfica 3.8 se presenta el histograma del error.

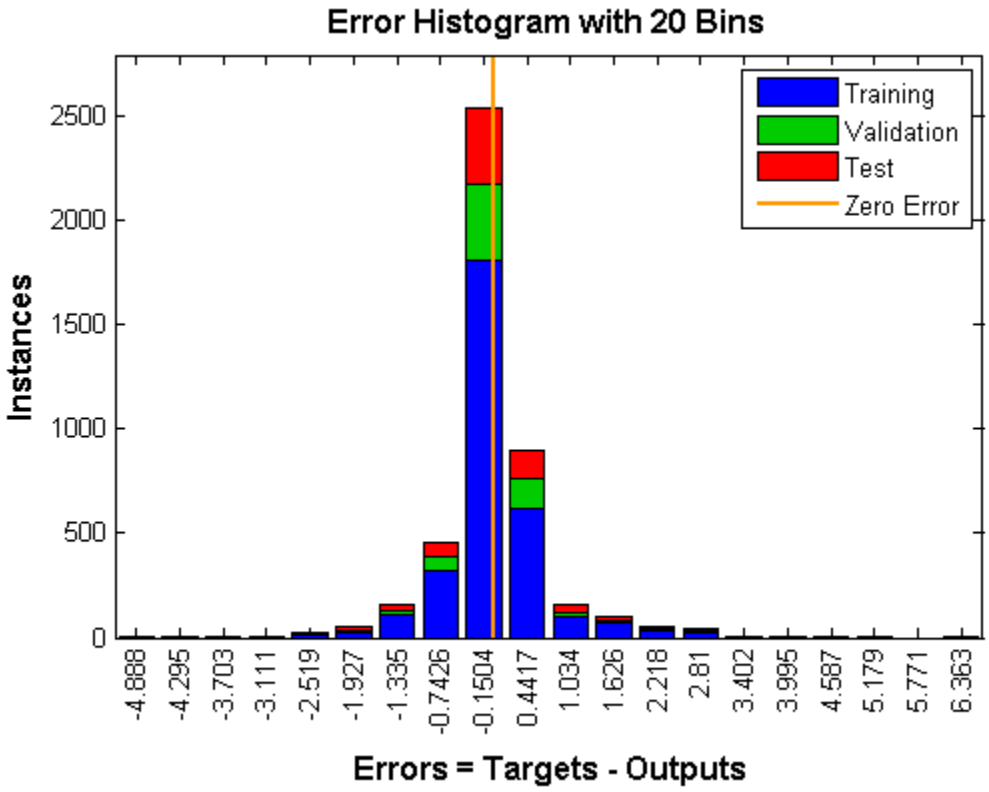


Figura 3.8 Histograma del error de la arquitectura No lineal Entrada-Salida.

Se muestra que la mayoría de los valores de desviación respecto al punto ideal es -0.1504, un valor que demuestra poca fidelidad.

Arquitectura No lineal Autoregresiva

El esquema de la red neuronal se muestra en la figura 3.9, éste tipo de red se utiliza cuando se contará con los valores de salida de la función.

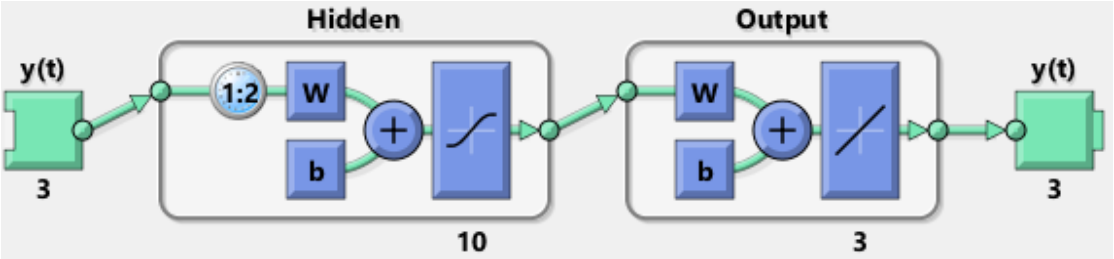


Figura 3.9 Arquitectura de la red neuronal No Lineal Autoregresiva.

La respuesta del modelo se muestra en la gráfica 3.10

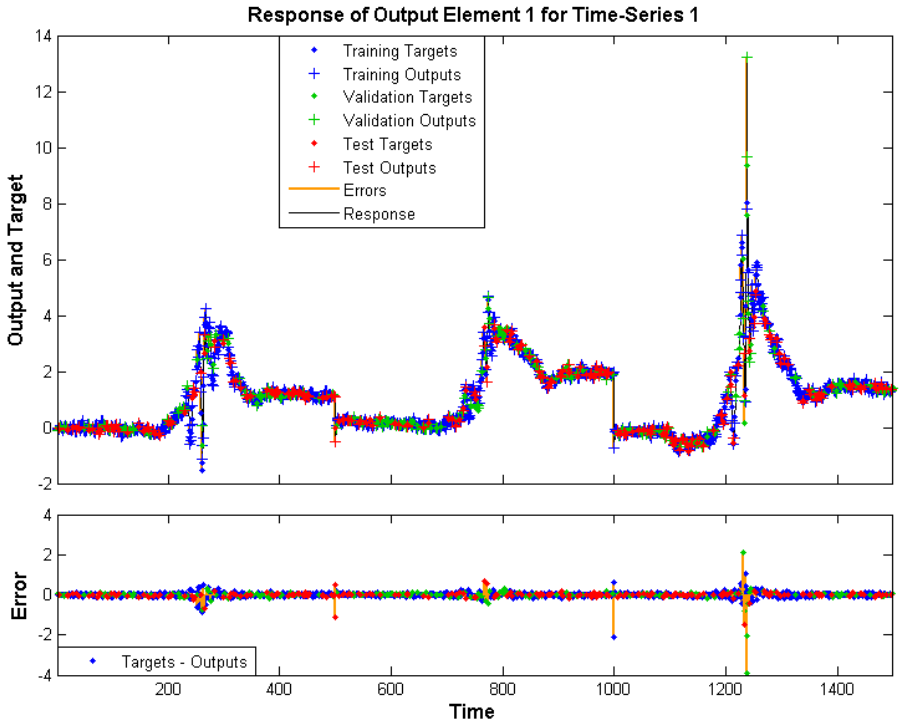


Figura 3.10 Gráfica de la respuesta al entrenamiento de la arquitectura No lineal Autoregresiva.

El error máximo en éste tipo de arquitectura es del 4%, aunado a la considerable disminución de líneas amarillas con respecto al modelo anterior se demuestra que éste tipo de arquitectura es más fiable.

El mejor desempeño de validación viene dado por la gráfica 3.11

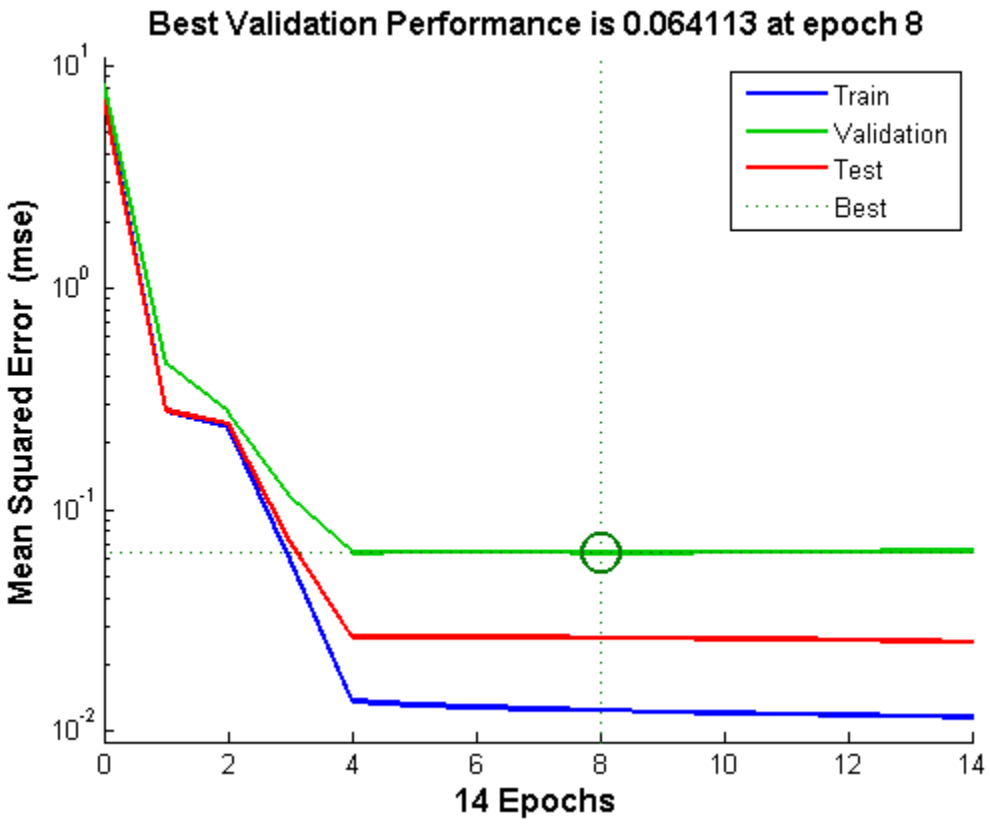


Figura 3.12 Mejor Desempeño de Validación de la arquitectura No lineal Autorregresiva.

El mejor valor que se pudo obtener para el Mean Squared Error ocurrió en el ciclo de evaluación computacional número8, el valor obtenido fue 0.068491, representa una mejora en eficiencia y precisión remarcable respecto al modelo anterior.

En la gráfica 3.12 se muestra la regresión de las variables.

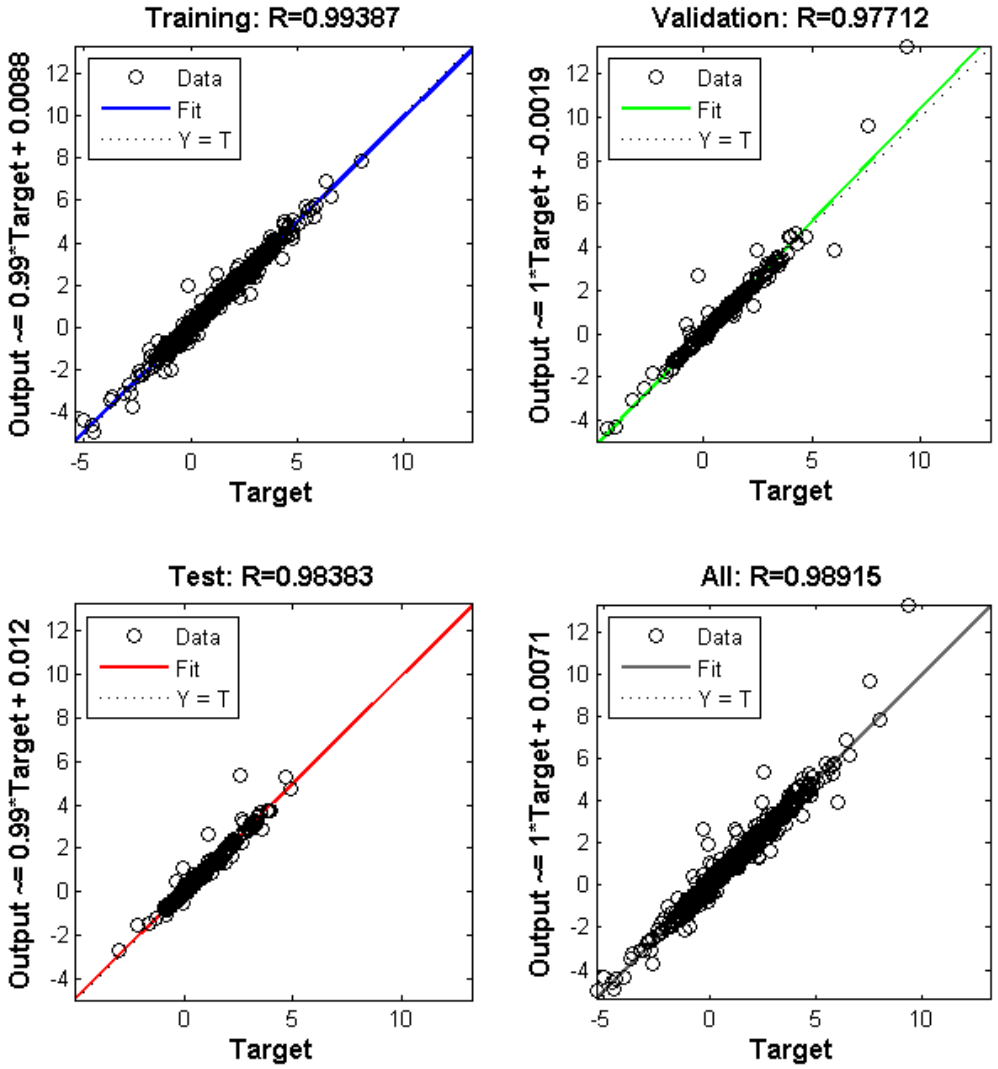


Figura 3.12 Curvas de regresión de la arquitectura No lineal Autoregresiva.

Se observa que la curva generada por la red neuronal es casi la curva ideal, esto es un indicio de que hay buena capacidad de generalización, los puntos en la distribución no presentan una desviación demasiado grande lo que refleja que se tiene poca incertidumbre en la mayoría de los mismos.

En la gráfica 3.13 se presenta el histograma del error.

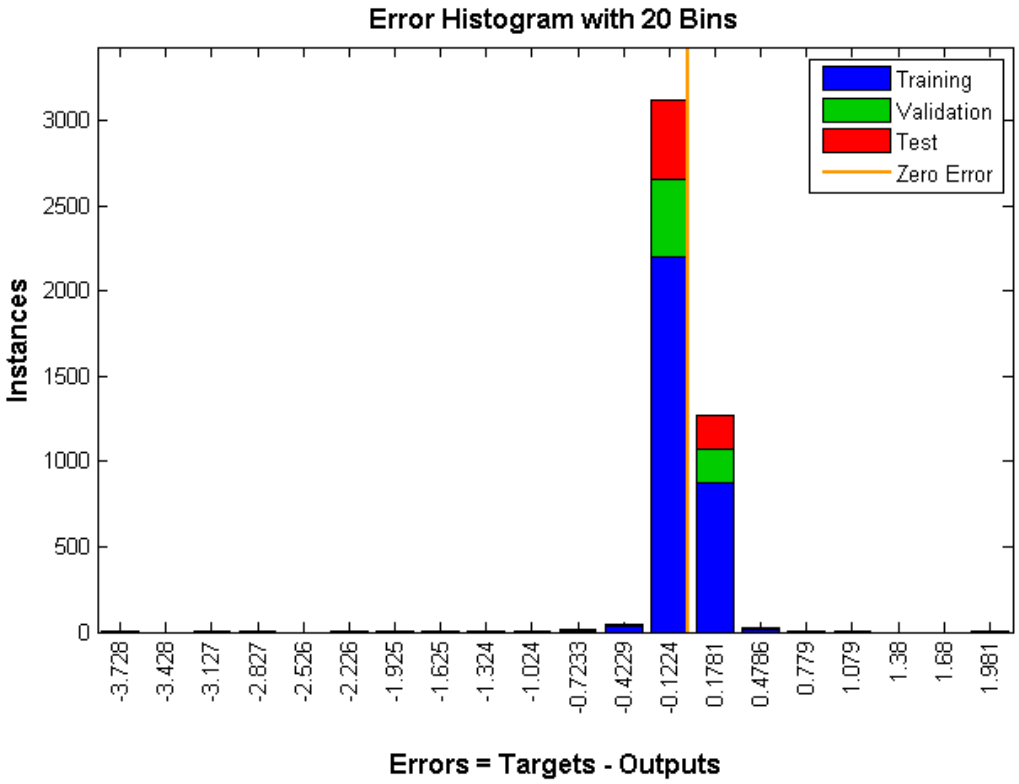


Figura 3.13 Histograma del error de la arquitectura No lineal Autoregresiva.

La figura 3.13 demuestra que la magnitud en valor del error respecto al modelo anterior disminuye, más no de forma considerable, por tanto la capacidad de generalización aumentó pero la magnitud del error no es lo suficientemente pequeña.

Arquitectura No lineal Autoregresiva con Entrada Exógena

El esquema de la red neuronal se muestra en la figura 3.14, éste esquema se utiliza cuando se puede acceder a los datos generados por la función y además se puede accesar a los datos de entrenamiento.

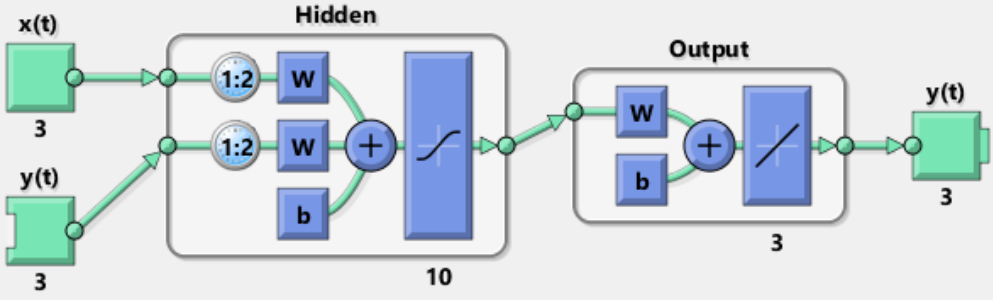


Figura 3.14 Esquema de la red neuronal No lineal Autoregresiva con Entrada Exógena.

La respuesta del modelo se muestra en la gráfica 3.15

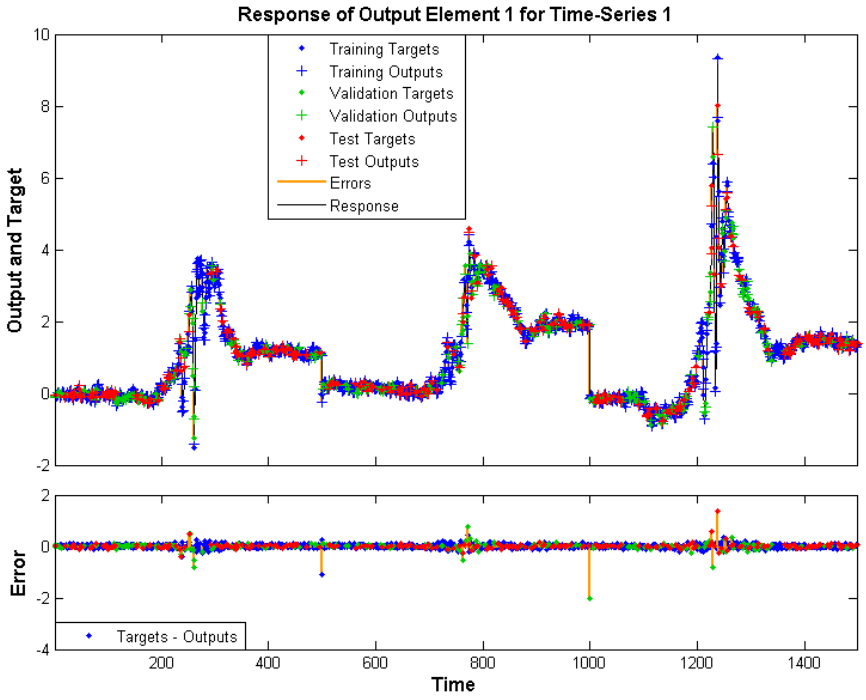


Figura 3.15 Gráfica de la respuesta al entrenamiento de la arquitectura No lineal Autoregresiva con Entrada Exógena.

Se observa que el error máximo está cerca del 2%, por tanto éste modelo presenta un error menor respecto a las otras dos arquitecturas.

El mejor desempeño de validación viene dado por la gráfica 3.16

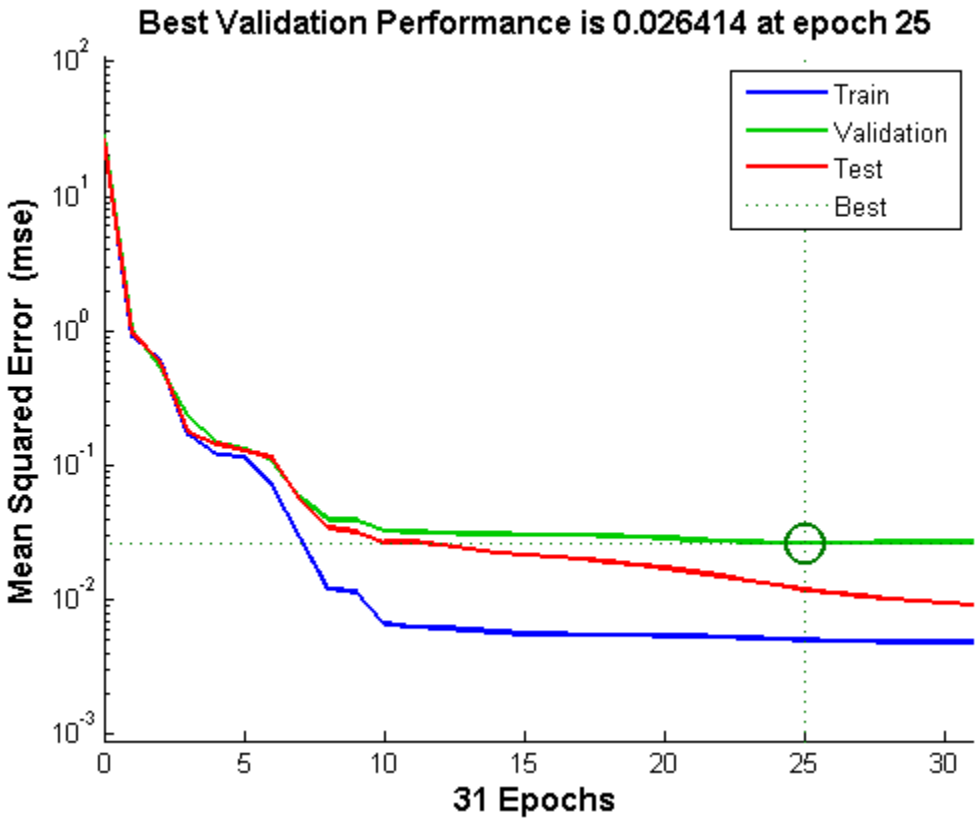


Figura 3.16 Mejor Desempeño de Validación de la arquitectura No lineal Autorregresiva con Entrada Exógena.

Comparando el gradiente de la figura 3.16 con el generado por la red No lineal de Entrada-Salida es dramáticamente pequeño, además de representar el mejor valor de todos los modelos.

En la gráfica 3.17 se muestra la regresión de las variables.

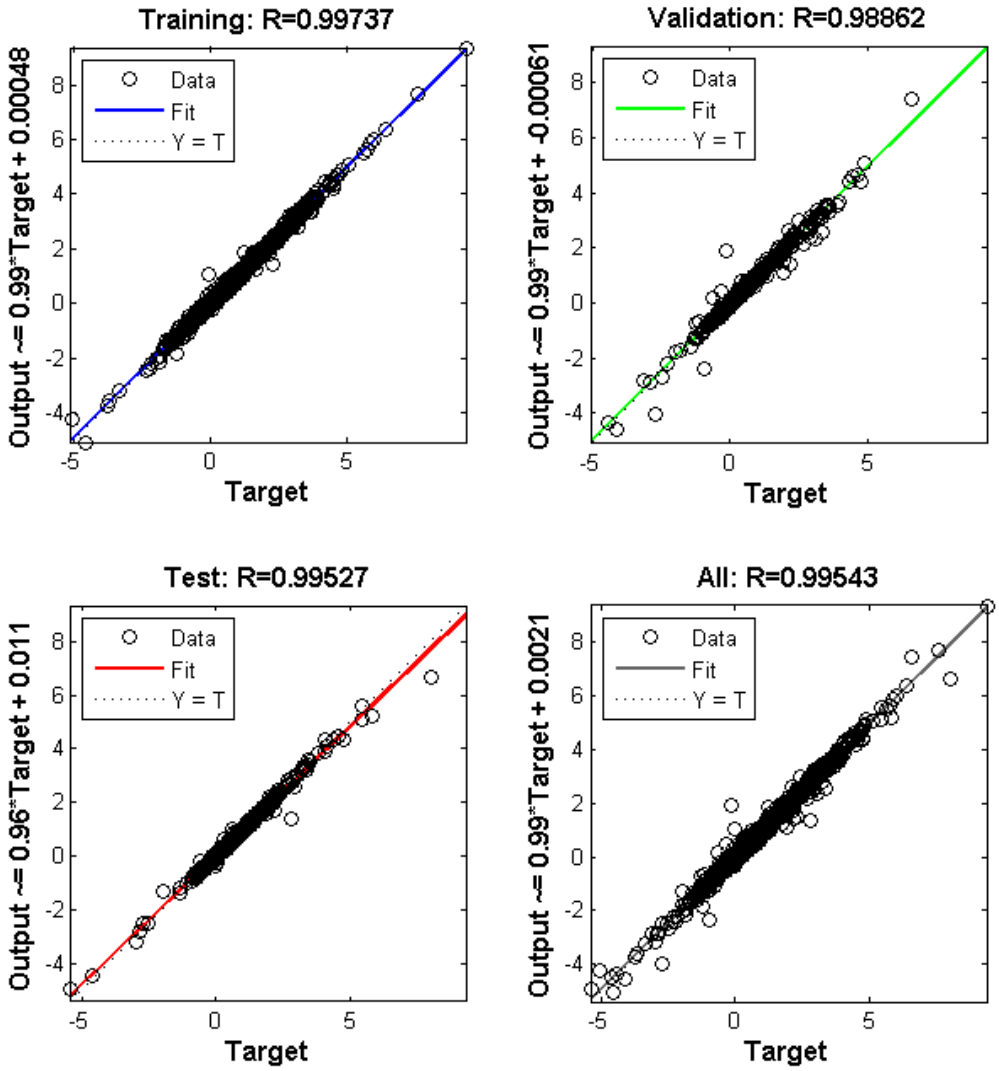


Figura 3. 17 Curvas de regresión de la arquitectura No lineal Autoregresiva con Entrada Exógena.

Podemos observar que los valores se distribuyen uniformemente cerca o casi exactamente sobre la curva de relación ideal, esto representa la capacidad superior de la red NARX para predecir parámetros.

En la gráfica 3.18 se presenta el histograma del error.

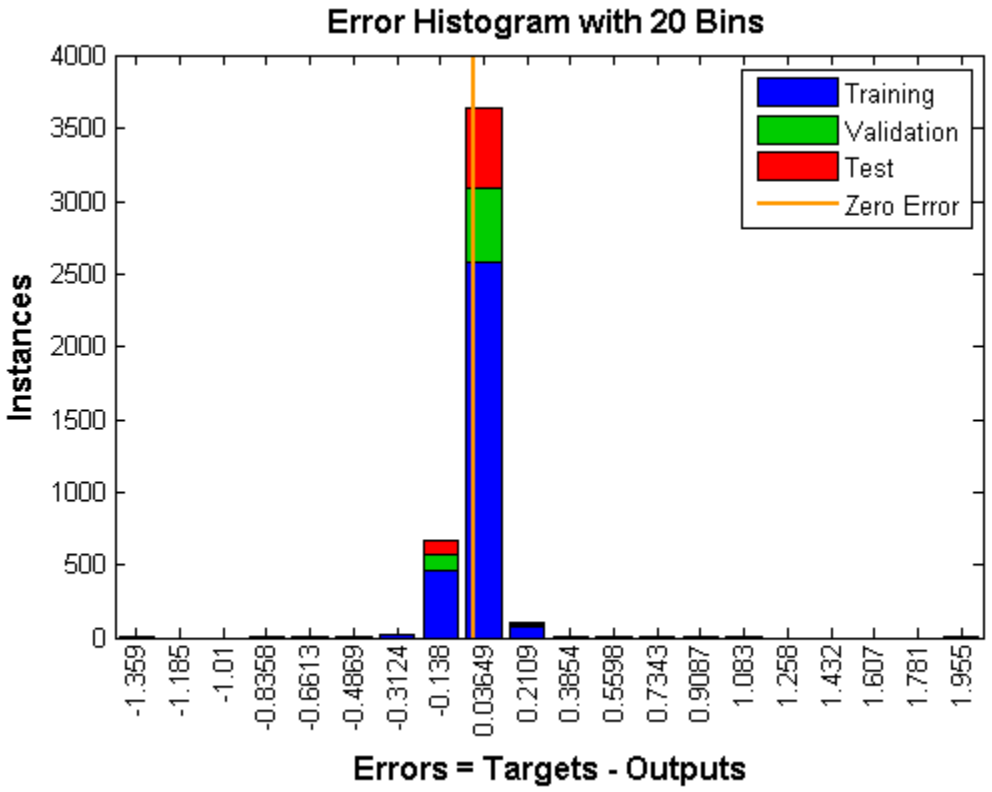


Figura 3. 18 Histograma del error de la arquitectura No lineal Autoregresiva con Entrada Exógena.

La relación de eficiencia entre la red NARX y la red NAR en términos de la magnitud del error es tan destacable que supera el 70%, brindando magnitudes de incertidumbre aceptables.

Después de los estudios realizados se plantea utilizar la red de arquitectura NARX por sus capacidades superiores de predicción y de brindar errores cuya magnitud es pequeña y aceptable.

Capítulo 4: Resultados

4.1 Red neuronal final propuesta

El esquema de la red neuronal final se muestra en la figura 4.1.

Se obtuvieron los parámetros acertados después de un proceso iterativo de evaluación, la red que cumple con los requisitos se presenta bajo el siguiente esquema:

- Una capa oculta de 130 neuronas con la función tanh(x) como función de activación.
- Un retraso de dos valores.
- Se utilizaran 80% de los datos para el entrenamiento de la red; la red es ajustada de acuerdo al error calculado en éstos datos.
- 5% de los datos se emplearán para validación; se utilizan para medir la capacidad de generalización, el entrenamiento se detiene cuando la generalización deja de mejorar.
- Para la evaluación se emplearan el 15% de los datos; proporcionan una medida independiente del desempeño durante y después del entrenamiento.
- La regla de entrenamiento será la propuesta por el algoritmo de Levenberg-Marquard.
- La función de error estará definida por el Minimun Squared Error.

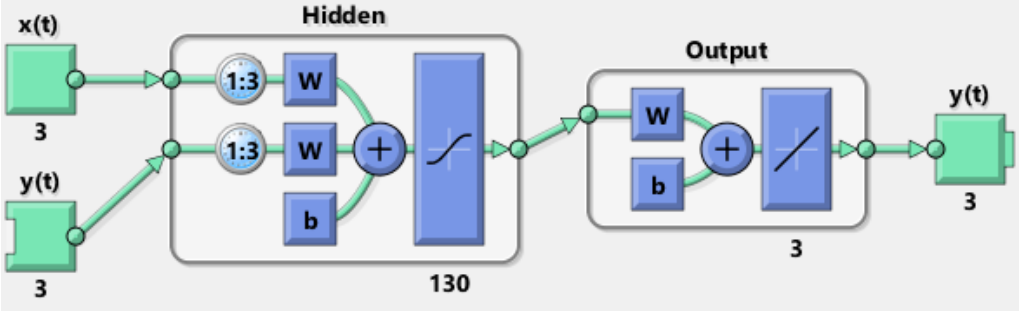


Figura 4.1 Esquema de la red neuronal No lineal Autoregresiva con Entrada Exógena Final.

La respuesta del modelo se muestra en la gráfica 4.2.

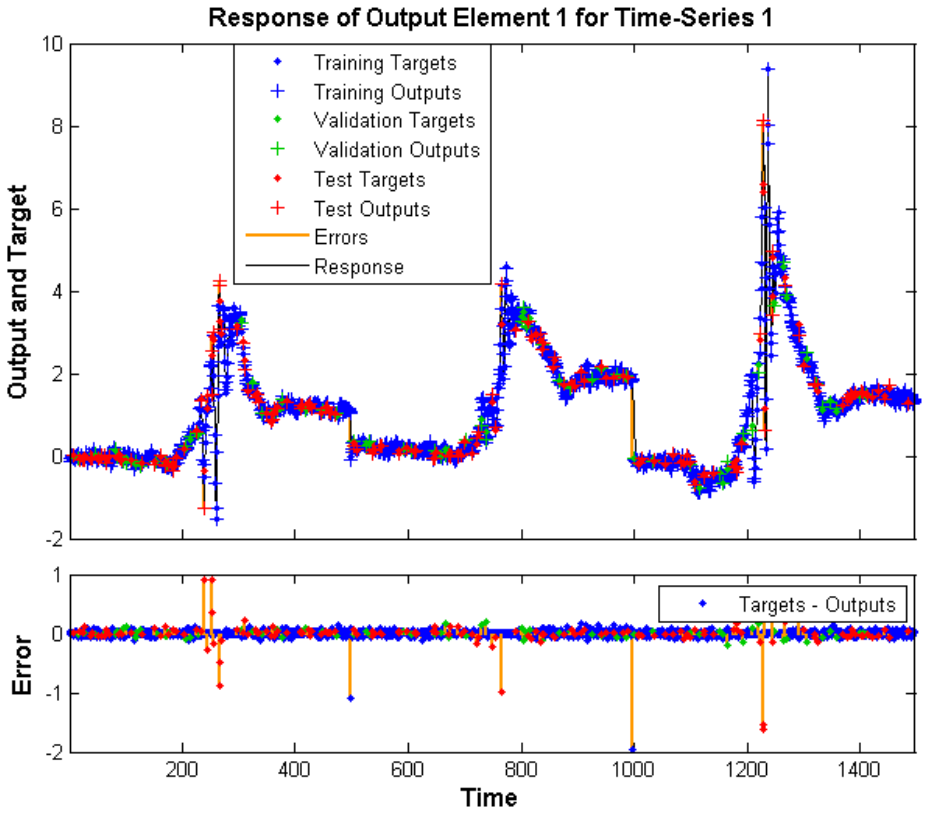


Figura 4.2 Gráfica de la respuesta al entrenamiento de la arquitectura No lineal Autoregresiva con Entrada Exógena Final.

Se observa que a través de la selección anterior de parámetros para entrenamiento se cumple el criterio de diseño de un error menor al 2%.

El mejor desempeño de validación viene dado por la gráfica 4.3

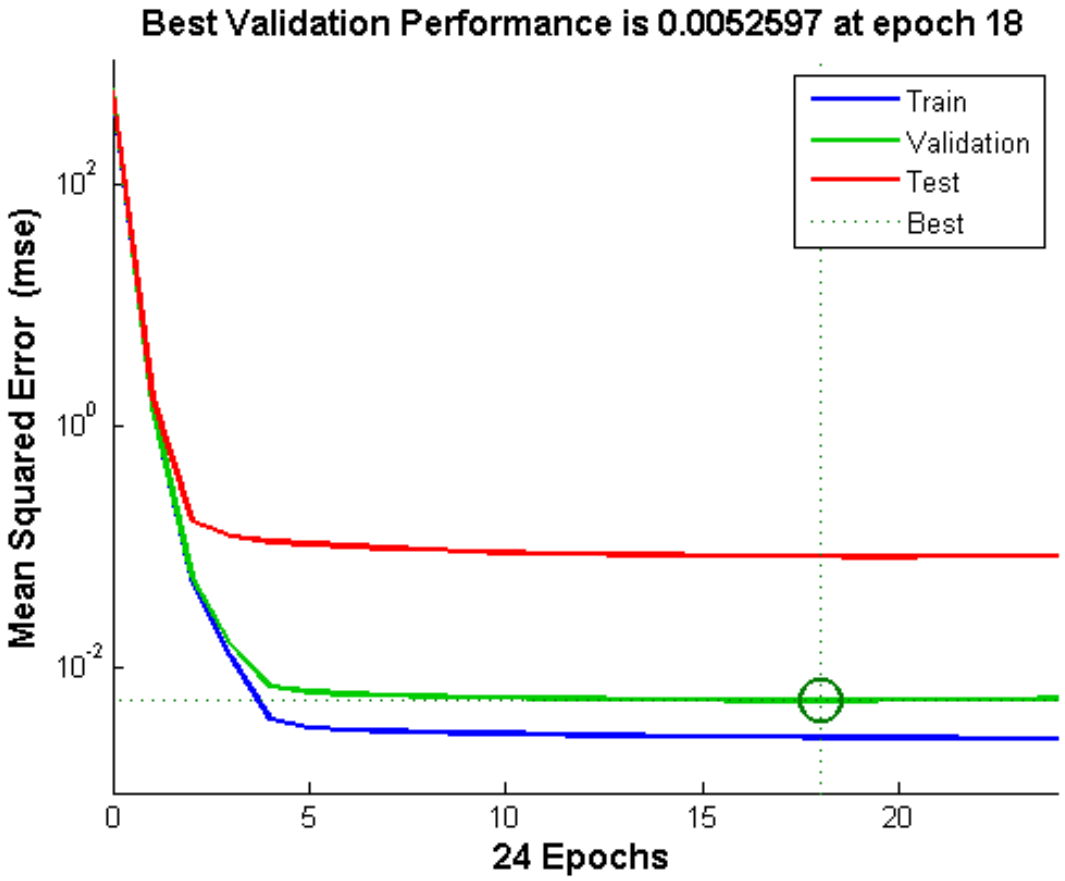


Figura 4.3 Mejor Desempeño de Validación de la arquitectura No lineal Autorregresiva con Entrada Exógena Final.

Comparada con los modelos te selección el gradiente del error es remarcablemente pequeño, implica una mejoría del 300% aproximadamente respecto a la red de 10 neuronas.

En la gráfica 4.4 se muestra la regresión de las variables.

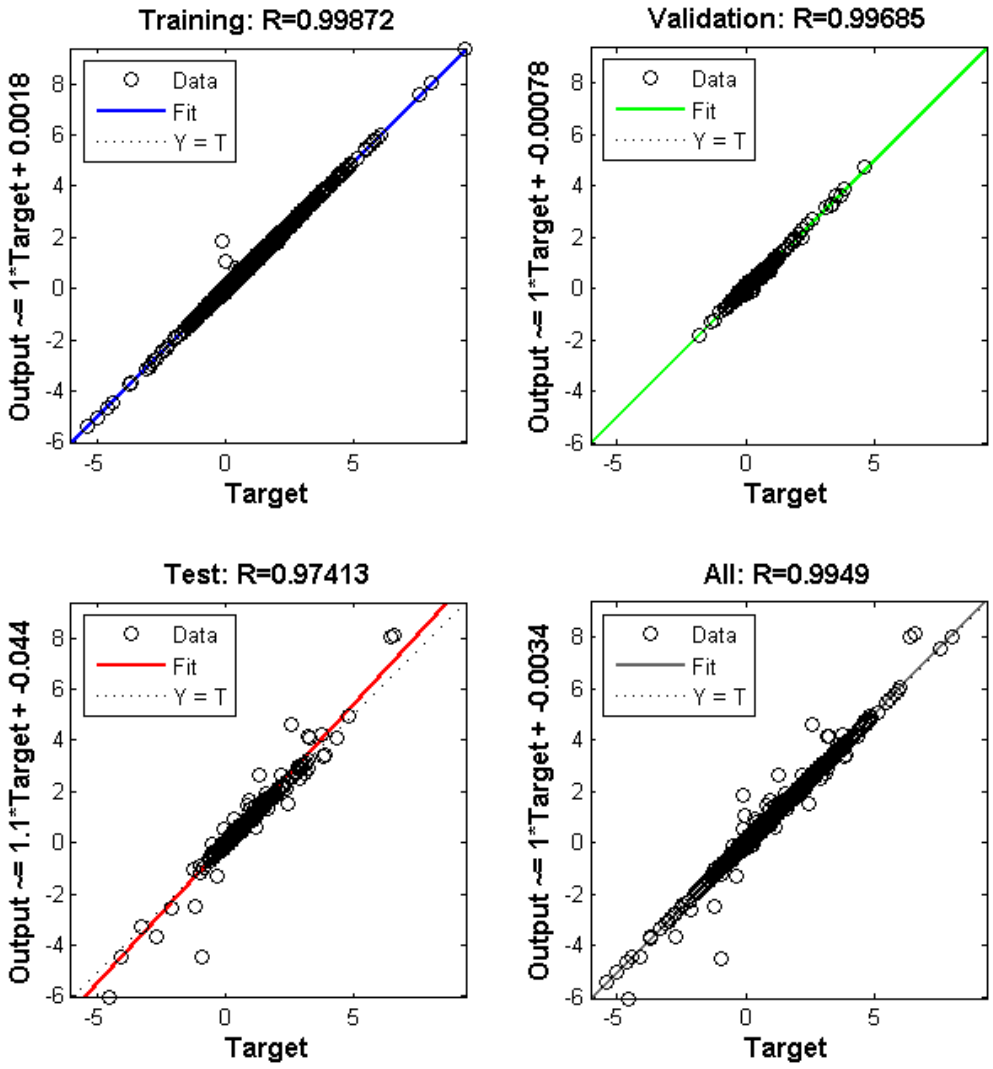


Figura 4.4 Curvas de regresión de la arquitectura No lineal Autoregresiva con Entrada Exógena Final.

La figura 4.4 muestra la reafirmación de lo conocido a través de las gráficas anteriores, el modelo presenta una desviación menor, dado que los puntos utilizados para el "Test" de la red neuronal no fueron los utilizados para entrenarla es de esperar que presenten la mayor desviación de todo el conjunto de datos.

En la gráfica 4.5 se presenta el histograma del error.

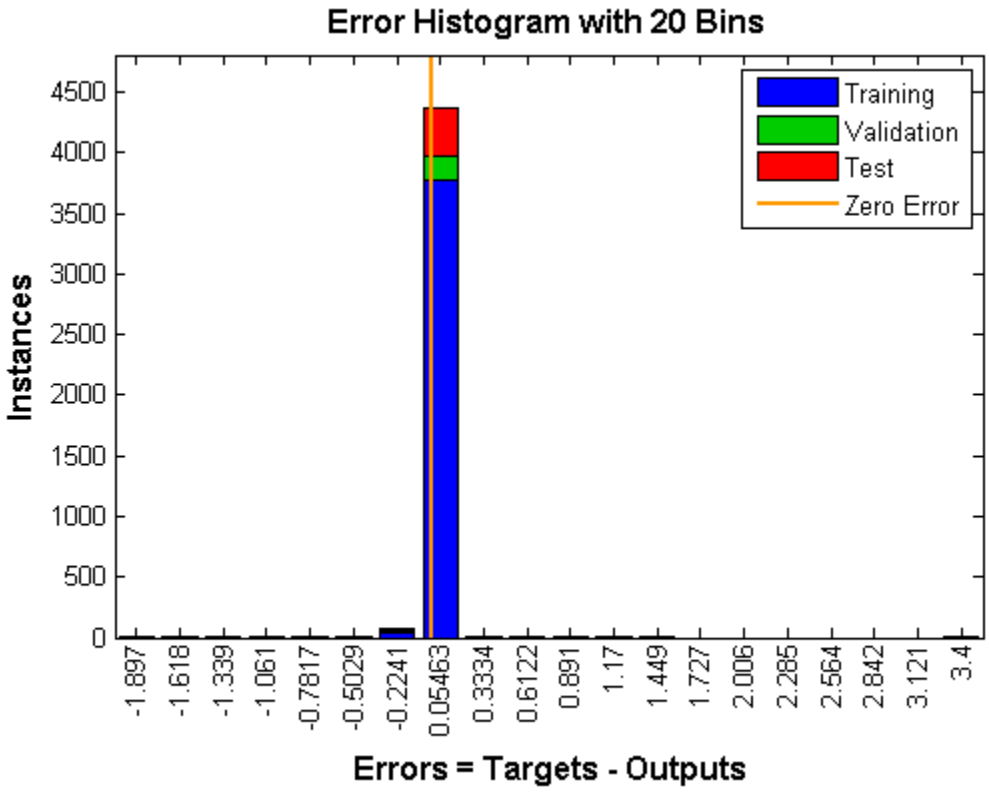


Figura 4. 5 Histograma del error de la arquitectura No lineal Autoregresiva con Entrada Exógena Final.

El histograma demuestra que el error para prácticamente la mayoría de puntos de la función presenta valores muy cercanos a cero.

En la gráfica 4.6 se muestra el error de autocorrelación.

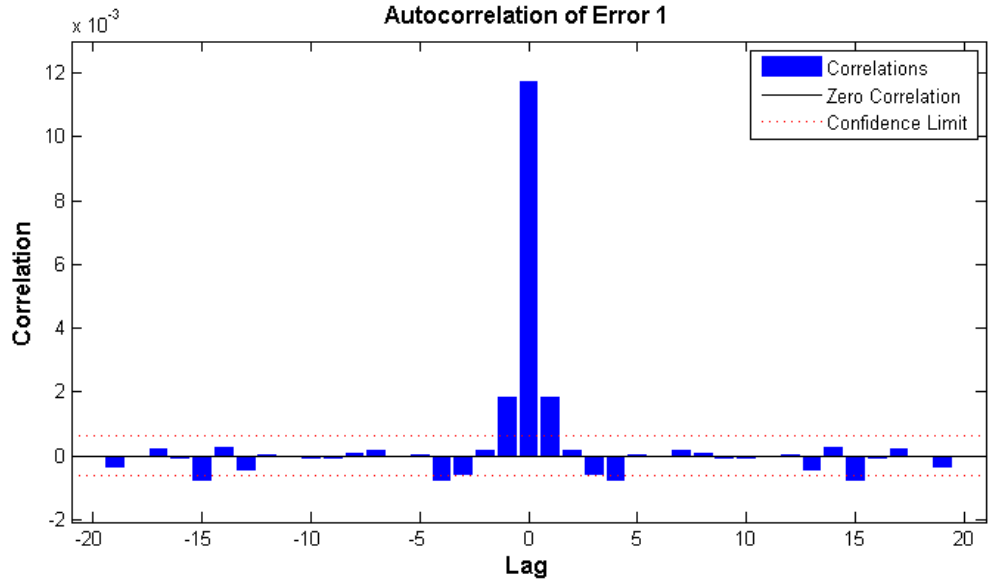


Figura 4.6 Error de autocorrelación de la arquitectura No lineal Autoregresiva con Entrada Exógena Final.

El error de autocorrelación demuestra que las medidas son confiables ya que lejos del lag 0 la gráfica no sobrepasa los niveles de confianza.

En la gráfica 4.7 se muestra la correlación existente entre la entrada y el error.

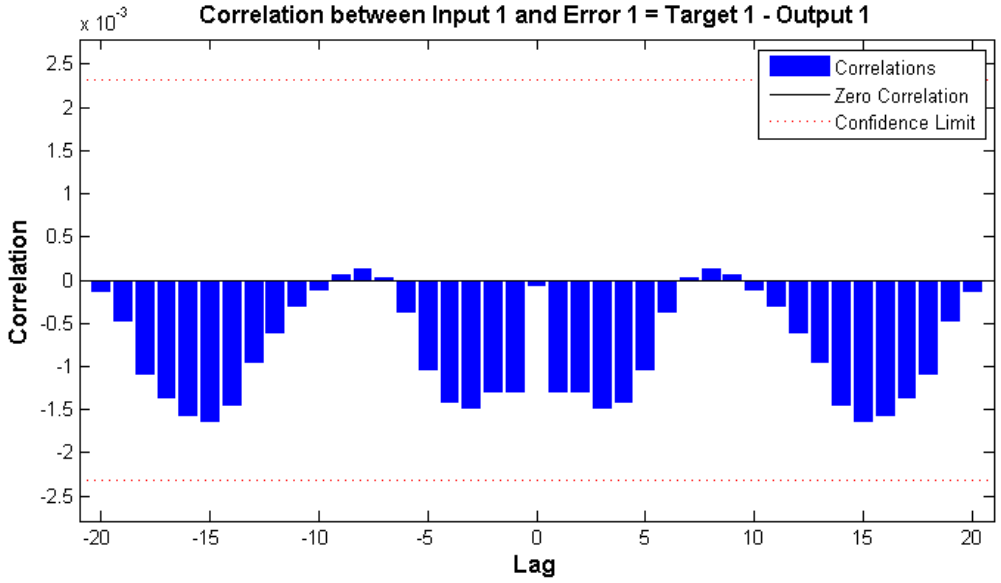


Figura 4.7 Correlación del error de la arquitectura No lineal Autoregresiva con Entrada Exógena Final.

El error de entre la entrada y la salida demuestra que las medidas son confiables ya que lejos del lag 0 la gráfica no sobrepasa los niveles de confianza. Por lo tanto se puede afirmar que no hay correlación y que el modelo es funcional.

Capítulo 5: Conclusiones

El aprendizaje es una selección de la función del error y un algoritmo de la actualización de los pesos. La selección del modelo implica intrínsecamente escoger el modelo más simple que cumpla con los parámetros deseados.

Cuando se reporta la solución final se brinda el modelo seleccionado (incluyendo el número de neuronas), en su caso, los retrasos seleccionados, la representación de los datos, pero los valores de vital importancia son los pesos generados después del entrenamiento de la red, ya que el objetivo principal del entrenamiento es ajustar los pesos de la red neuronal de la forma en que obtengamos la salida necesaria.

Si se genera un numero excesivamente grande de neuronas en la capa oculta la red neuronal sufre del conocido problema de “overfitting”, que ocurre cuando la hipótesis describe la curva de la función objetivo de una manera demasiado cercana, por lo tanto la hipótesis se ésta ajustando también al ruido de la señal, lo que significa que nuestra aproximación será errónea, presentara un error de la muestra bajo sin duda, pero el error de evaluación será tan grande que nos hará rechazar un modelo.

Los objetivos planteados se cubrieron a cabalidad, el error de aproximación de la red neuronal no supera el 2%. Las medidas relacionadas con el Minimum Squared Error asumen que los errores asociados a cada punto de la gráfica son independientes, por tanto si la autocorrelación del error hubiera rebasado los límites de confianza no se podría afirmar que la generalización de la red neuronal es funcional, al no haber rebasado los límites de confianza el error de autocorrelación las estimaciones de la red neuronal son confiables.

Capítulo 7: Bibliografia

- Abu-Mustafa, Y. S. (2 de Abril de 2012). Learning from data, machine learning course - recorded at a live broadcast from Caltech. California, Pasadena, U.S.A.
- André. (20 de Febrero de 2014). *ceticismo.net*. Obtenido de O leão mecânico de Leonardo da Vinci: http://ceticismo.net/2009/08/16/o-leao-mecanico-de-leonardo-da-vinci/
- Betts, D. E. (21 de Febrero de 2014). *Industrial Robot*. Obtenido de http://www.madehow.com/Volume-2/Industrial-Robot.html
- Bolaños, D. A. (2013). *Aprendizaje por refuerzo seguro para enseñar a un robot humanoide a caminar más rápido.* Madrid: Universidad Carlos III de Madrid.
- Bustamante, E. (2007). *El sistema nervioso, desde las neuronas hasta el cerebro humano.* Medellín: Universidad de Antioquia.
- Capelli, A. (2002). Robótica industrial. *Revista Mecatrônica Industrial*, 72.
- Chrislb. (02 de Febrero de 2014). *MultiLayerNeuralNetwork*. Obtenido de http://commons.wikimedia.org/wiki/File:MultiLayerNeuralNetwork.png
- Chrislb. (26 de Febrero de 2014). *RecurrentLayerNeuralNetwork*. Obtenido de http://commons.wikimedia.org/wiki/File:RecurrentLayerNeuralNetwork.png
- Chrislb. (26 de Febrero de 2014). *SingleLayerNeuralNetwork*. Obtenido de http://commons.wikimedia.org/wiki/File:SingleLayerNeuralNetwork.png
- Cortés, T. (19 de febrero de 2014). *Comunicación neuronal*. Obtenido de http://comunicacionneuronal.blogspot.pt/
- Craig, J. J. (1986). *Introduction to Robotics: Mechanics and Control.* Addison-Weley Publiching Company.
- Estevam, A. L. (19 de Febrero de 2014). *Cérebro e Memória: Mistérios Elétricos*. Obtenido de http://www.rc.unesp.br/biosferas/mat0002.php
- Fernández, E. M. (2011). *Diseño del sistema de conexionado electrónico para los brazos del robot humanoide RH2.* Madrid: Universidad Carlos III.
- Haykin, S. (1999). *Neural Networks: A comprensive Foundation.* New Jersey: Prentice Hall.
- Hirai, k. (1997). Current and Future Perspective of Honda Humanoid Robot. *International Conference on Intelligent Robots and Systems*, (págs. 500-508).
- Honda. (28 de Febrero de 2014). *Asimo*. Obtenido de http://world.honda.com/ASIMO/

- Huang, Q., Li, K., Nakamura, Y., & Tanie, K. (2001). Analysis of physical capability of a biped humanoid: Walkink speed and actuator specifications. *Internacional Conference on Intelligent Robots and Systems,*. Maui.
- Martín del Brío, B., & Sanz, A. (1997). *Redes neuronales y sistemas borrosos.* Madrid: RA-MA.
- Martínez de la Casa Díaz, S. (2012). *Human inspired humanoid robots control architecture.* Madrid: Universidad Carlos III de Madrid: Department of Systems Engineering and Automation, PhD Thesis.
- MIT, A. I. (22 de Febrero de 2014). *Brooks, Rodney Allen: Brooks and Cog*. Obtenido de http://www.britannica.com/EBchecked/media/56161/Rodney-Brooks-and-Cog-the-robot
- Murphy, R. R. (2000). *Introduction to ai robotics.* Cambridge: The Mit Press.
- NASA. (28 de Febrero de 2014). *What is Robonaut?* Obtenido de http://www.nasa.gov/audience/forstudents/k-4/stories/what-is-robonaut-k4.html#.UxEsl_l5P0Y
- Nogueira, M. B. (2005). *posicionamento e movimentação de um Robô Humanóide utilizando imagens de uma câmara móvel externa.* Rio Grande: Universidade Federal do Rio Grande do Norte Centro de Tecnologia .
- Papert, S. (1988). *Logo: computadores e educação.* São Paulo: Brasiliense.
- Pfeifer, E. a. (2006). Coleta de lixo médico utilizando protótipos de lego. *CONGRESSO BRASILEIRO DE COMPUTAÇÃO, XXVI* (págs. 75 - 82). Campo Grande: Anais do Enri - III Encontro de Robótica Inteligente.
- Pieruccini, A. (21 de Febrero de 2014). Obtenido de http://www.extremenxt.com/elsie.jpg
- Pio, J. e. (2006). A Robótica Móvel como Instrumento de Apoio à Aprendizagem de Computação. *SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, XVII* (págs. 8 - 10). Brasília: SBC.
- RAE. (24 de 02 de 2014). *Real Academia Española*. Obtenido de Diccionario de la lengua española: http://lema.rae.es/drae/?val=automatico
- Redel, R. e. (20 de Febrero de 2014). *Implementação de simuladores de robô com o uso da tecnologia de realidade virtual*. Obtenido de www.niee.ufrgs.br/cbcomp/cbcomp2004
- Rosário, J. M. (2007). *Princípios de mecatrônica.* São Paulo: Pearson.

- Sadava, D., Heller, H., Orians, G., Purvens, W., & Hillis, D. (2009). *Vida la ciencia de la biología.* México: Médica Panamericana.
- Sánchez, E. N., & Alanís, A. Y. (2006). *Redes neuronales, conceptos fundamentales y aplicaciones a Control Automático.* Madrid: Pearson Educación.
- Sánchez, F. M. (2007). *Historia de la robótica: de Arquitas de Tarento al robot Da Vinci (Parte II). .* Obtenido de http://digital.csi
- Scassellati, B. (2001). *Foundations for a Theory of Mind for a Humanoid Robot.* Cambridge: Department of Electrical Engineering and Computer Science, PhD Thesis.
- Simpson, P. (1990). *Artificial Neural Systems: Foundation, Paradogms, Applications and Implementations.* New York: Pergamon Press.
- Souza, F. J. (21 de Febrero de 2014). *Robótica.* Obtenido de Robôs sociais: http://webx.ubi.pt/~felippe/texts5/robotica_cap7.pdf
- Stevens, C. (1979). The neuron. *Sientific American*, 54-65.
- Tamburro, J. (20 de Febrero de 2014). Obtenido de http://janeentrelinhas.blogspot.pt/search/label/Me%20Myself%20and%20I
- Taufiqurrakhman, M. (21 de Febrero de 2014). *Galería de autómatas*. Obtenido de http://crosser.byethost18.com/gallery.html
- Torres, L. (20 de Febrero de 2014). Obtenido de http://culturacolectiva.com/armas-para-la-locura-bestial-da-vinci/
- UC3M. (16 de Febrero de 2014). *Humanoids* . Obtenido de http://roboticslab.uc3m.es/roboticslab/gallery.php?albumname=humanoids
- Walter, W. (1963). *The Living Brain.* New York: Norton Library.

ANEXOS

ANEXO 1 Códigos de programación y reporte de pesos de la red de la solución propuesta

PROGRAMA DE ENTRENAMIENTO LA RED NEURONAL NO LINEAL ENTRADA-SALIDA

```
% Solve an Input-Output Time-Series Problem with a Time Delay Neural Network
% Script generated by NTSTOOL.
% Created Sun Mar 02 13:05:32 CST 2014
%
% This script assumes these variables are defined:
%
%   MPasos - input time series.
%   MCaidaF - target time series.

X = tonndata(MPasos,true,false);
T = tonndata(MCaidaF,true,false);

% Create a Time Delay Network
inputDelays = 1:2;
hiddenLayerSize = 10;
net = timedelaynet(inputDelays,hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};

% Prepare the Data for Training and Simulation
% The function PREPARETS prepares timeseries data for a particular network,
% shifting time by the minimum amount to fill input states and layer states.
% Using PREPARETS allows you to keep your original time series data unchanged,
while
% easily customizing it for networks with differing numbers of delays, with
% open loop or closed loop feedback modes.
[x,xi,ai,t] = preparets(net,X,T);

% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'time';  % Divide up every value
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm';  % Levenberg-Marquardt

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','plotresponse', ...
  'ploterrcorr', 'plotinerrcorr'};
```

```
% Train the Network
[net,tr] = train(net,x,t,xi,ai);

% Test the Network
outputs = net(x,xi,ai);
errors = gsubtract(t,y);
performance = perform(net,t,y)

% Recalculate Training, Validation and Test Performance
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotresponse(t,y)
%figure, ploterrcorr(e)
%figure, plotinerrcorr(x,e)

% Early Prediction Network
% For some applications it helps to get the prediction a timestep early.
% The original network returns predicted y(t+1) at the same time it is given
x(t+1).
% For some applications such as decision making, it would help to have predicted
% y(t+1) once x(t) is available, but before the actual y(t+1) occurs.
% The network can be made to return its output a timestep early by removing one
delay
% so that its minimal tap delay is now 0 instead of 1.  The new network returns
the
% same outputs as the original network, but outputs are shifted left one
timestep.
nets = removedelay(net);
[xs,xis,ais,ts] = preparets(nets,X,T);
ys = nets(xs,xis,ais);
earlyPredictPerformance = perform(net,ts,ys)

% Deployment
% Change the (false) values to (true) to enable the following code blocks.
% See the help for each generation function for more information.
if (false)
  % Generate MATLAB function for neural network for application deployment
  % in MATLAB scripts or with MATLAB Compiler and Builder tools, or simply
  % to examine the calculations your trained neural network performs.
  genFunction(net,'myNeuralNetworkFunction');
  y = myNeuralNetworkFunction(x,xi,ai);
end
if (false)
  % Generate a matrix-only MATLAB function for neural network code
  % generation with MATLAB Coder tools.
  genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
  x1 = cell2mat(x(1,:));
  xi1 = cell2mat(xi(1,:));
  y = myNeuralNetworkFunction(x1,xi1);
```

```
end
if (false)
  % Generate a Simulink diagram for simulation or deployment with.
  % Simulink Coder tools.
  gensim(net);
end
```

PROGRAMA DE SIMULACIÓN LA RED NEURONAL NO LINEAL ENTRADA-SALIDA

```
function [y1,xf1] = RedNIO(x1,xi1)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 02-Mar-2014 13:05:03.
%
% [y1,xf1] = myNeuralNetworkFunction(x1,xi1) takes these arguments:
%   x = 3xTS matrix, input #1
%   xi1 = 3x2 matrix, initial 2 delay states for input #1.
% and returns:
%   y = 3xTS matrix, output #1
%   xf1 = 3x2 matrix, final 2 delay states for input #1.
% where TS is the number of timesteps.

  % ===== NEURAL NETWORK CONSTANTS =====

  % Input 1
  x1_step1_xoffset = [-1.648446;-2.160223;-1.165047];
  x1_step1_gain = [0.724022262236519;0.55746440659447;0.481916098407267];
  x1_step1_ymin = -1;

  % Layer 1
  b1 = [3.543496561031104;2.5153204391827959;-1.6911977820433008;-
1.1313345102626555;0.14450451456156377;-
0.85340685101661695;1.2665557132230494;1.559183877855741;-
2.173445500308639;1.7799207899155147];
  IW1_1 = [-1.9240998043284563 2.0896218613784594 2.1647822746224965 -
1.774872706098253 -1.0636993989303085 1.9476649091879816;-0.20584853008635626 -
2.0567293635953989 -0.28792799972523819 -0.44994542264732534 -0.77254358751590968
-0.68813126649708423;-0.57059223840806939 1.9770742448678305 0.28657396560266063
1.8275645832355631 1.7397484211518914 1.4591191893054918;-1.1792741490470209
0.46883777828265011 -1.4126358862505715 -0.66504387665641684 0.75934799659081265
-0.99478297115033176;-0.6817635839513978 -0.23191843887596703 0.50627867152682338
-0.61659668207757967 1.2371792138019941
0.00087226526330027448;0.73775868119482246 0.51687737356821783 -
1.1027899204542058 0.75227861961383702 -0.64960501094125811 0.97685822622990393;-
1.3118846015043519 1.9317073416986676 0.39375985446751427 -1.5377200321040663
3.1774879256056892 1.3903281782603298;1.3954790627054692 -0.57954024968578555
3.0133618838613505 1.750355524031701 -1.5409566993571819 0.13050462059947404;-
1.2235675552246874 1.5432704878132457 -0.050813647719274352 1.7952504253966335 -
0.26324536977191065 0.6582553167414984;-0.46030133841857207 1.3486054358369042
1.4706404026953479 -0.62741506057910823 1.1568579604115858 1.68072395245271381;

  % Layer 2
  b2 = [0.39456215059345351;0.46669402538229676;-0.95520208066995183];
  LW2_1 = [-0.35428640013238966 0.095517249290511821 0.27546488601275099 -
0.55938847530562452 -0.91088122011158257 -1.0944245908811481 -0.48107311775485095
-0.70362510953601565 1.2836907788011978 0.76080181591192775;-0.028728844123295048
-0.65777248758910878 -0.19698837433662131 -0.47745084488357636 -
0.16867480955112307 -0.0031708512224032127 -0.19690076032066928 -
```

```
0.47774411211566226 -0.10294223696312897 0.2838070006275718;-0.020057141045257337
0.67566263815083483 0.3046631511365967 0.27911043876205022 -0.33009904874944429 -
0.38283447752706296 -0.12209739806144912 0.17344629874832015 -0.3519103756792954
0.24150536240075099];

  % Output 1
  y1_step1_ymin = -1;
  y1_step1_gain = [0.18373480306477;0.379388145562126;0.196861089303277];
  y1_step1_xoffset = [-1.526312;-2.33664;-5.391398];

  % ===== SIMULATION ========

  % Dimensions
  TS = size(x1,2); % timesteps

  % Input 1 Delay States
  xd1 = mapminmax_apply(xi1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);
  xd1 = [xd1 zeros(3,1)];

  % Allocate Outputs
  y1 = zeros(3,TS);

  % Time loop
  for ts=1:TS

    % Rotating delay state position
    xdts = mod(ts+1,3)+1;

    % Input 1
    xd1(:,xdts) =
mapminmax_apply(x1(:,ts),x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

    % Layer 1
    tapdelay1 = reshape(xd1(:,mod(xdts-[1 2]-1,3)+1),6,1);
    a1 = tansig_apply(b1 + IW1_1*tapdelay1);

    % Layer 2
    a2 = b2 + LW2_1*a1;

    % Output 1
    y1(:,ts) =
mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
  end

  % Final delay states
  finalxts = TS+(1: 2);
  xits = finalxts(finalxts<=2);
  xts = finalxts(finalxts>2)-2;
  xf1 = [xi1(:,xits) x1(:,xts)];
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
  y = bsxfun(@minus,x,settings_xoffset);
  y = bsxfun(@times,y,settings_gain);
  y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
```

```
  a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
  x = bsxfun(@minus,y,settings_ymin);
  x = bsxfun(@rdivide,x,settings_gain);
  x = bsxfun(@plus,x,settings_xoffset);
end
```

PROGRAMA DE ENTRENAMIENTO LA RED NEURONAL NO LINEAL AUTOREGRESIVA

```
% Solve an Autoregression Time-Series Problem with a NAR Neural Network
% Script generated by NTSTOOL
% Created Sun Mar 02 10:37:56 CST 2014
%
% This script assumes this variable is defined:
%
%   MCaidaF - feedback time series.

T = tonndata(MCaidaF,true,false);

% Create a Nonlinear Autoregressive Network
feedbackDelays = 1:2;
hiddenLayerSize = 10;
net = narnet(feedbackDelays,hiddenLayerSize);

% Choose Feedback Pre/Post-Processing Functions
% Settings for feedback input are automatically applied to feedback output
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};

% Prepare the Data for Training and Simulation
% The function PREPARETS prepares timeseries data for a particular network,
% shifting time by the minimum amount to fill input states and layer states.
% Using PREPARETS allows you to keep your original time series data unchanged,
while
% easily customizing it for networks with differing numbers of delays, with
% open loop or closed loop feedback modes.
[x,xi,ai,t] = preparets(net,{},{},T);

% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'time';  % Divide up every value
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Choose a Training Function
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm';  % Levenberg-Marquardt

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','plotresponse', ...
  'ploterrcorr', 'plotinerrcorr'};


% Train the Network
[net,tr] = train(net,x,t,xi,ai);
```

```
% Test the Network
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)

% Recalculate Training, Validation and Test Performance
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotresponse(t,y)
%figure, ploterrcorr(e)
%figure, plotinerrcorr(x,e)

% Closed Loop Network
% Use this network to do multi-step prediction.
% The function CLOSELOOP replaces the feedback input with a direct
% connection from the outout layer.
netc = closeloop(net);
[xc,xic,aic,tc] = preparets(netc,{},{},T);
yc = netc(xc,xic,aic);
perfc = perform(net,tc,yc)
% Multi-step Prediction
% Sometimes it is useful to simulate a network in open-loop form for as
% long as there is known data T, and then switch to closed-loop to perform
% multistep prediction. Here The open-loop network is simulated on the known
% output series, then the network and its final delay states are converted
% to closed-loop form to produce predictions for 5 more timesteps.
[x1,xio,aio,t] = preparets(net,{},{},T);
[y1,xfo,afo] = net(x1,xio,aio);
[netc,xic,aic] = closeloop(net,xfo,afo);
[y2,xfc,afc] = netc(cell(0,5),xic,aic);
% Further predictions can be made by continuing simulation starting with
% the final input and layer delay states, xfc and afc.


% Step-Ahead Prediction Network
% For some applications it helps to get the prediction a timestep early.
% The original network returns predicted y(t+1) at the same time it is given
y(t+1).
% For some applications such as decision making, it would help to have predicted
% y(t+1) once y(t) is available, but before the actual y(t+1) occurs.
% The network can be made to return its output a timestep early by removing one
delay
% so that its minimal tap delay is now 0 instead of 1.  The new network returns
the
% same outputs as the original network, but outputs are shifted left one
timestep.
nets = removedelay(net);
[xs,xis,ais,ts] = preparets(nets,{},{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(net,ts,ys)
```

```
% Deployment
% Change the (false) values to (true) to enable the following code blocks.
% See the help for each generation function for more information.
if (false)
  % Generate MATLAB function for neural network for application deployment
  % in MATLAB scripts or with MATLAB Compiler and Builder tools, or simply
  % to examine the calculations your trained neural network performs.
  genFunction(net,'myNeuralNetworkFunction');
  y = myNeuralNetworkFunction(x,xi,ai);
end
if (false)
  % Generate a matrix-only MATLAB function for neural network code
  % generation with MATLAB Coder tools.
  genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
  x1 = cell2mat(x(1,:));
  xi1 = cell2mat(xi(1,:));
  y = myNeuralNetworkFunction(x1,xi1);
end
if (false)
  % Generate a Simulink diagram for simulation or deployment with.
  % Simulink Coder tools.
  gensim(net);
end
```

PROGRAMA DE SIMULACIÓN LA RED NEURONAL NO LINEAL AUTOREGRESIVA

```
function [y1,xf1] = RedNAR(x1,xi1)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 02-Mar-2014 10:36:15.
%
% [y1,xf1] = myNeuralNetworkFunction(x1,xi1) takes these arguments:
%   x = 3xTS matrix, input #1
%   xi1 = 3x2 matrix, initial 2 delay states for input #1.
% and returns:
%   y = 3xTS matrix, output #1
%   xf1 = 3x2 matrix, final 2 delay states for input #1.
% where TS is the number of timesteps.

  % ===== NEURAL NETWORK CONSTANTS =====

  % Input 1
  x1_step1_xoffset = [-1.526312;-2.33664;-5.391398];
  x1_step1_gain = [0.18373480306477;0.379388145562126;0.196861089303277];
  x1_step1_ymin = -1;

  % Layer 1
  b1 = [-1.9972892716896415;-2.2880260528461407;0.72429525970028308;-
0.86908741670416279;0.37161058055252666;-0.045749535640488115;-
0.0017119523821677034;-0.86812957356867992;0.88396775275860551;-
2.1083512910156545];
  IW1_1 = [1.0743675672779474 -0.19776950691505746 1.1315416142201264
0.26166340580967429 -1.1624275282125172 -0.57003970232102652;0.9085426008731986 -
0.80876786241389098 -0.92105758984074293 0.66245862879369144 -0.94359926102133507
-0.092407042106084275;-1.5328113259254736 0.20759953528653585 0.546164830940156
1.3200327247912611 -0.17948870777688405 -0.4801801204411757;-0.69142293098027641
-0.90993090131485588 -0.20499711793485684 -0.51995875720469842 1.2509513539168911
-0.48465284277087117;0.10916482344640872 0.048719278319020405 1.5971625454487961
```

```
-0.028475258359822064 0.11887957302801709 -1.2529483873063358;-1.3578594507718043
0.13897082767835239 -1.0866715765824868 0.0072315133669818019 0.28654720691987978
-0.91035899819719057;-0.33108389585213388 1.2783282356555021 -0.37934988557222854
0.50693537637829056 -0.48831335055919628 0.10630051324608861;-1.2769409182545608
-0.28865137468095542 -1.0537578854572618 -0.093524241808214489 1.1725959887437774
-0.59899717166544864;0.45471611144683716 -0.71604959616027219 1.1131096259641053
0.11145574732114912 0.27876970448261051 -1.5497159790468205;-0.30753264358974036
-0.56652632778501544 -0.72778957026224123 -0.88643582269555821 -
0.65692137443383103 1.242399863446078];

  % Layer 2
  b2 = [0.093794576474571387;0.54856358225406843;0.047193634042788486];
  LW2_1 = [0.54277131450743954 -0.63951158684052078 -1.3457617465765837 -
0.3244879002694761 -0.031295076957853739 -0.19843116882636558 0.48773438009094322
0.057906254239698307 0.72425615089423356 0.081412535516278098;-
0.27181993083397565 0.35721584536715784 -0.33229498269731994 -
0.090189754409207068 0.10079271091937528 -0.023331499982398651
0.86775088935062228 0.063337687423314207 -0.58462707884873599 -
0.14782528135788331;0.013623474467992653 0.29715642811874499 0.30047751105304493
-0.4614055655154104 1.6577886489219125 -0.099249923651843486 -0.54283442436333262
0.22520723691644509 -0.64549335416949172 0.21995728297790437];

  % Output 1
  y1_step1_ymin = -1;
  y1_step1_gain = [0.18373480306477;0.379388145562126;0.196861089303277];
  y1_step1_xoffset = [-1.526312;-2.33664;-5.391398];

  % ===== SIMULATION ========

  % Dimensions
  TS = size(x1,2); % timesteps

  % Input 1 Delay States
  xd1 = mapminmax_apply(xi1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);
  xd1 = [xd1 zeros(3,1)];

  % Allocate Outputs
  y1 = zeros(3,TS);

  % Time loop
  for ts=1:TS

    % Rotating delay state position
    xdts = mod(ts+1,3)+1;

    % Input 1
    xd1(:,xdts) =
mapminmax_apply(x1(:,ts),x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

    % Layer 1
    tapdelay1 = reshape(xd1(:,mod(xdts-[1 2]-1,3)+1),6,1);
    a1 = tansig_apply(b1 + IW1_1*tapdelay1);

    % Layer 2
    a2 = b2 + LW2_1*a1;

    % Output 1
    y1(:,ts) =
mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
  end

  % Final delay states
```

```
  finalxts = TS+(1: 2);
  xits = finalxts(finalxts<=2);
  xts = finalxts(finalxts>2)-2;
  xf1 = [xi1(:,xits) x1(:,xts)];
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
  y = bsxfun(@minus,x,settings_xoffset);
  y = bsxfun(@times,y,settings_gain);
  y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
  a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
  x = bsxfun(@minus,y,settings_ymin);
  x = bsxfun(@rdivide,x,settings_gain);
  x = bsxfun(@plus,x,settings_xoffset);
end
```

PROGRAMA DE ENTRENAMIENTO LA RED NEURONAL NO LINEAL AUTOREGRESIVA CON ENTRADA EXÓGENA

```
% Solve an Autoregression Problem with External Input with a NARX Neural Network
% Script generated by NTSTOOL
% Created Sun Mar 02 13:47:41 CST 2014
%
% This script assumes these variables are defined:
%
%   MPasos - input time series.
%   MCaidaF - feedback time series.

X = tonndata(MPasos,true,false);
T = tonndata(MCaidaF,true,false);

% Create a Nonlinear Autoregressive Network with External Input
inputDelays = 1:2;
feedbackDelays = 1:2;
hiddenLayerSize = 20;
net = narxnet(inputDelays,feedbackDelays,hiddenLayerSize);

% Choose Input and Feedback Pre/Post-Processing Functions
% Settings for feedback input are automatically applied to feedback output
% For a list of all processing functions type: help nnprocess
% Customize input parameters at: net.inputs{i}.processParam
% Customize output parameters at: net.outputs{i}.processParam
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.inputs{2}.processFcns = {'removeconstantrows','mapminmax'};

% Prepare the Data for Training and Simulation
% The function PREPARETS prepares timeseries data for a particular network,
% shifting time by the minimum amount to fill input states and layer states.
% Using PREPARETS allows you to keep your original time series data unchanged,
while
% easily customizing it for networks with differing numbers of delays, with
% open loop or closed loop feedback modes.
[x,xi,ai,t] = preparets(net,X,{},T);

% Setup Division of Data for Training, Validation, Testing
% The function DIVIDERAND randomly assigns target values to training,
% validation and test sets during training.
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
% The property DIVIDEMODE set to TIMESTEP means that targets are divided
% into training, validation and test sets according to timesteps.
% For a list of data division modes type: help nntype_data_division_mode
net.divideMode = 'value';  % Divide up every value
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% Customize training parameters at: net.trainParam
net.trainFcn = 'traingd';  % Levenberg-Marquardt

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
% Customize performance parameters at: net.performParam
```

```
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
% Customize plot parameters at: net.plotParam
net.plotFcns = {'plotperform','plottrainstate','plotresponse', ...
  'ploterrcorr', 'plotinerrcorr'};

% Train the Network
[net,tr] = train(net,x,t,xi,ai);

% Test the Network
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)

% Recalculate Training, Validation and Test Performance
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotregression(t,y)
%figure, plotresponse(t,y)
%figure, ploterrcorr(e)
%figure, plotinerrcorr(x,e)

% Closed Loop Network
% Use this network to do multi-step prediction.
% The function CLOSELOOP replaces the feedback input with a direct
% connection from the outout layer.
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
view(netc)
[xc,xic,aic,tc] = preparets(netc,X,{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(netc,tc,yc)
% Multi-step Prediction
% Sometimes it is useful to simulate a network in open-loop form for as
% long as there is known output data, and then switch to closed-loop form
% to perform multistep prediction while providing only the external input.
% Here all but 5 timesteps of the input series and target series are used to
% simulate the network in open-loop form, taking advantage of the higher
% accuracy that providing the target series produces:
numTimesteps = size(x,2);
knownOutputTimesteps = 1:(numTimesteps-5);
predictOutputTimesteps = (numTimesteps-4):numTimesteps;
X1 = X(:,knownOutputTimesteps);
T1 = T(:,knownOutputTimesteps);
[x1,xio,aio] = preparets(net,X1,{},T1);
[y1,xfo,afo] = net(x1,xio,aio);
% Next the the network and its final states will be converted to closed-loop
% form to make five predictions with only the five inputs provided.
```

```
x2 = X(1,predictOutputTimesteps);
[netc,xic,aic] = closeloop(net,xfo,afo);
[y2,xfc,afc] = netc(x2,xic,aic);
multiStepPerformance = perform(net,T(1,predictOutputTimesteps),y2)
% Alternate predictions can be made for different values of x2, or further
% predictions can be made by continuing simulation with additional external
% inputs and the last closed-loop states xfc and afc.


% Step-Ahead Prediction Network
% For some applications it helps to get the prediction a timestep early.
% The original network returns predicted y(t+1) at the same time it is given
y(t+1).
% For some applications such as decision making, it would help to have predicted
% y(t+1) once y(t) is available, but before the actual y(t+1) occurs.
% The network can be made to return its output a timestep early by removing one
delay
% so that its minimal tap delay is now 0 instead of 1.  The new network returns
the
% same outputs as the original network, but outputs are shifted left one
timestep.
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
[xs,xis,ais,ts] = preparets(nets,X,{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)

% Deployment
% Change the (false) values to (true) to enable the following code blocks.
% See the help for each generation function for more information.
if (false)
  % Generate MATLAB function for neural network for application deployment
  % in MATLAB scripts or with MATLAB Compiler and Builder tools, or simply
  % to examine the calculations your trained neural network performs.
  genFunction(net,'myNeuralNetworkFunction');
  y = myNeuralNetworkFunction(x,xi,ai);
end
if (false)
  % Generate a matrix-only MATLAB function for neural network code
  % generation with MATLAB Coder tools.
  genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
  x1 = cell2mat(x(1,:));
  x2 = cell2mat(x(2,:));
  xi1 = cell2mat(xi(1,:));
  xi2 = cell2mat(xi(2,:));
  y = myNeuralNetworkFunction(x1,x2,xi1,xi2);
end
if (false)
  % Generate a Simulink diagram for simulation or deployment with.
  % Simulink Coder tools.
  gensim(net);
end
```

PROGRAMA DE SIMULACIÓN LA RED NEURONAL NO LINEAL AUTOREGRESIVA CON ENTRADA EXÓGENA

```
function [y1,xf1,xf2] = RedNARX(x1,x2,xi1,xi2)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 02-Mar-2014 13:41:39.
%
% [y1,xf1,xf2] = myNeuralNetworkFunction(x1,x2,xi1,xi2) takes these arguments:
%   x = 3xTS matrix, input #1
%   1 = 3xTS matrix, input #2
%   xi1 = 3x2 matrix, initial 2 delay states for input #1.
%   xi2 = 3x2 matrix, initial 2 delay states for input #2.
% and returns:
%   y = 3xTS matrix, output #1
%   xf1 = 3x2 matrix, final 2 delay states for input #1.
%   xf2 = 3x2 matrix, final 2 delay states for input #2.
% where TS is the number of timesteps.

  % ===== NEURAL NETWORK CONSTANTS =====

  % Input 1
  x1_step1_xoffset = [-1.648446;-2.160223;-1.165047];
  x1_step1_gain = [0.724022262236519;0.55746440659447;0.481916098407267];
  x1_step1_ymin = -1;

  % Input 2
  x2_step1_xoffset = [-1.526312;-2.33664;-5.391398];
  x2_step1_gain = [0.18373480306477;0.379388145562126;0.196861089303277];
  x2_step1_ymin = -1;

  % Layer 1
  b1 = [0.62808214741960289;1.2695872030007578;-0.42459387561607087;-
0.15260207083337141;0.043613218052830666;0.033111758305040762;-
0.44261872297007576;-0.15198373244998237;-1.1319322946257524;-
0.76606254491426573];
  IW1_1 = [0.046532097695152208 0.1803429194245468 0.15865164022605233 -
0.032817113726513358 -0.2551773541295288 -0.1700180126560934;-0.61159763100362641
-1.0869707474407921 -2.3651794512994653 -0.47562919209605919 0.28900514642924646
1.9881337510639232;0.58035591820089016 0.21387434634616123 0.14734421361159894 -
0.49466137345276046 -0.35383529217080014 0.022462742378059716;-0.18425077072506686
-0.46075798992170691 0.050143055842242407 0.020303971790931918
0.71867572246357558 -0.21642071503014523;-0.37623442366872978 0.23765345526291781
0.33646531422402332 0.40085537763321238 -0.061020379025012679 -
0.4591918360547666;0.35794122345858237 -0.13694674259236717 -0.13144942220864739
-0.31189436106721979 0.075218482571747264 0.27114811327130828;-
0.31970103593943244 0.22079410531462021 -0.26794440357020111 0.42449150263813479
-0.13624903986976655 0.33614271771648224;0.45899039929231927 -0.50625710526291945
-0.48054482543446664 -0.41502320356000588 0.46317032393877045
0.70248746203257628;-0.70045164010605354 0.74409869419697039 -0.33905962812301854
1.3612379210294308 -0.64640647750592484 2.4526627368998959;-0.51381478739916175
0.20661310688033252 0.11285138812016304 0.5016386954224209 -0.40218120045103534 -
0.35083826041644689];
  IW1_2 = [-0.55660610606587335 -0.1808064445684627 1.202633908435613
0.75594335094016951 0.014307556556390825 -0.94742562899727545;3.6450279847142042
0.64900018136317206 2.7972673081041921 -1.5468338593913231 -1.1920653560226402 -
1.1106923571595981;0.10243684463143671 0.82604693871989965 2.5302030730457865 -
0.03150137122760064 -0.38192487599408326 -3.2233899599970659;1.6795958214435893 -
0.88681631850008469 -2.3033195369031354 -1.5810510378287586 1.5392948694783424
```

```
1.6117728809515504;-0.20145584248837056 0.27887211568230086 -1.1585888896236876
0.34502036879268194 0.64080127524220643 1.8133696629800016;0.1841921258244992
0.77906630036041935 0.10458795031149143 -0.087396064120614889 -
0.59097080664872914 -0.60012281608185469;0.75415370166372619 0.54323630273334977
0.8026956414822396 -0.38555774264134973 -0.44833216874232623
0.88477429491305259;0.36332436458697348 1.1445644153035606 0.072067481648406456 -
0.83056151507436371 -1.1145941289583305 -0.27734928352977256;0.9982106425772006
0.042342496445984881 0.32683453987956945 1.2574007759798591 0.65637030672336727
5.7267404618286175;4.0248145546281728 1.7102905721753476 -0.090353583756340353 -
3.7709723772582482 -2.5141028040300735 -0.23819545503384537];

  % Layer 2
  b2 = [-0.56840312049028496;0.27147118448197305;-0.66101171108497825];
  LW2_1 = [1.2776402372300635 -0.008446548977597515 -0.51622104087637044
0.60310085839846139 -0.38811198559945853 2.0778892446599699 0.81182440593825433 -
0.7824925058469504 -0.074445290562020094 0.2736468059098639;-0.55591393708272141
0.092362328909258815 0.20093436449817736 -0.25927373839266549 1.089218488178356
0.43262605512735386 -0.35363937107098636 0.28576969679703224 0.067185038609554493
0.12681982993417767;2.0181210401223724 0.011999486688657038 0.30918936449862694
0.3112391204112066 0.17254362864489173 -1.1095709229061408 0.36322606304402005
0.89600759228523008 -0.027090116723946036 0.047654302626148552];

  % Output 1
  y1_step1_ymin = -1;
  y1_step1_gain = [0.18373480306477;0.379388145562126;0.196861089303277];
  y1_step1_xoffset = [-1.526312;-2.33664;-5.391398];

  % ===== SIMULATION ========

  % Dimensions
  TS = size(x1,2); % timesteps

  % Input 1 Delay States
  xd1 = mapminmax_apply(xi1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);
  xd1 = [xd1 zeros(3,1)];

  % Input 2 Delay States
  xd2 = mapminmax_apply(xi2,x2_step1_gain,x2_step1_xoffset,x2_step1_ymin);
  xd2 = [xd2 zeros(3,1)];

  % Allocate Outputs
  y1 = zeros(3,TS);

  % Time loop
  for ts=1:TS

    % Rotating delay state position
    xdts = mod(ts+1,3)+1;

    % Input 1
    xd1(:,xdts) =
mapminmax_apply(x1(:,ts),x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

    % Input 2
    xd2(:,xdts) =
mapminmax_apply(x2(:,ts),x2_step1_gain,x2_step1_xoffset,x2_step1_ymin);

    % Layer 1
    tapdelay1 = reshape(xd1(:,mod(xdts-[1 2]-1,3)+1),6,1);
    tapdelay2 = reshape(xd2(:,mod(xdts-[1 2]-1,3)+1),6,1);
    a1 = tansig_apply(b1 + IW1_1*tapdelay1 + IW1_2*tapdelay2);
```

```
    % Layer 2
    a2 = b2 + LW2_1*a1;

    % Output 1
    y1(:,ts) =
mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
  end

  % Final delay states
  finalxts = TS+(1: 2);
  xits = finalxts(finalxts<=2);
  xts = finalxts(finalxts>2)-2;
  xf1 = [xi1(:,xits) x1(:,xts)];
  xf2 = [xi2(:,xits) x2(:,xts)];
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
  y = bsxfun(@minus,x,settings_xoffset);
  y = bsxfun(@times,y,settings_gain);
  y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
  a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
  x = bsxfun(@minus,y,settings_ymin);
  x = bsxfun(@rdivide,x,settings_gain);
  x = bsxfun(@plus,x,settings_xoffset);
end
```

PROGRAMA DE ENTRENAMIENTO LA RED NEURONAL SOLUCIÓN NO LINEAL AUTOREGRESIVA CON ENTRADA EXÓGENA

Los pesos reportados en ésta sección son los que se consideran como parte de la solución del modelo final seleccionado.

```
% Solve an Autoregression Problem with External Input with a NARX Neural Network
% Script generated by NTSTOOL
% Created Sun Mar 02 19:46:31 CST 2014
%
% This script assumes these variables are defined:
%
%   MPasos - input time series.
%   MCaidaF - feedback time series.

X = tonndata(MPasos,true,false);
```

```
T = tonndata(MCaidaF,true,false);

% Create a Nonlinear Autoregressive Network with External Input
inputDelays = 1:3;
feedbackDelays = 1:3;
hiddenLayerSize = 130;
net = narxnet(inputDelays,feedbackDelays,hiddenLayerSize);

% Choose Input and Feedback Pre/Post-Processing Functions
% Settings for feedback input are automatically applied to feedback output
% For a list of all processing functions type: help nnprocess
% Customize input parameters at: net.inputs{i}.processParam
% Customize output parameters at: net.outputs{i}.processParam
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.inputs{2}.processFcns = {'removeconstantrows','mapminmax'};

% Prepare the Data for Training and Simulation
% The function PREPARETS prepares timeseries data for a particular network,
% shifting time by the minimum amount to fill input states and layer states.
% Using PREPARETS allows you to keep your original time series data unchanged,
while
% easily customizing it for networks with differing numbers of delays, with
% open loop or closed loop feedback modes.
[x,xi,ai,t] = preparets(net,X,{},T);

% Setup Division of Data for Training, Validation, Testing
% The function DIVIDERAND randomly assigns target values to training,
% validation and test sets during training.
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
% The property DIVIDEMODE set to TIMESTEP means that targets are divided
% into training, validation and test sets according to timesteps.
% For a list of data division modes type: help nntype_data_division_mode
net.divideMode = 'value';  % Divide up every value
net.divideParam.trainRatio = 85/100;
net.divideParam.valRatio = 5/100;
net.divideParam.testRatio = 10/100;

% Choose a Training Function
% For a list of all training functions type: help nntrain
% Customize training parameters at: net.trainParam
net.trainFcn = 'traingd';  % Levenberg-Marquardt

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
% Customize performance parameters at: net.performParam
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
% Customize plot parameters at: net.plotParam
net.plotFcns = {'plotperform','plottrainstate','plotresponse', ...
  'ploterrcorr', 'plotinerrcorr'};

% Train the Network
[net,tr] = train(net,x,t,xi,ai);

% Test the Network
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)
```

```
% Recalculate Training, Validation and Test Performance
trainTargets = gmultiply(t,tr.trainMask);
valTargets = gmultiply(t,tr.valMask);
testTargets = gmultiply(t,tr.testMask);
trainPerformance = perform(net,trainTargets,y)
valPerformance = perform(net,valTargets,y)
testPerformance = perform(net,testTargets,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotregression(t,y)
%figure, plotresponse(t,y)
%figure, ploterrcorr(e)
%figure, plotinerrcorr(x,e)

% Closed Loop Network
% Use this network to do multi-step prediction.
% The function CLOSELOOP replaces the feedback input with a direct
% connection from the outout layer.
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
view(netc)
[xc,xic,aic,tc] = preparets(netc,X,{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(netc,tc,yc)
% Multi-step Prediction
% Sometimes it is useful to simulate a network in open-loop form for as
% long as there is known output data, and then switch to closed-loop form
% to perform multistep prediction while providing only the external input.
% Here all but 5 timesteps of the input series and target series are used to
% simulate the network in open-loop form, taking advantage of the higher
% accuracy that providing the target series produces:
numTimesteps = size(x,2);
knownOutputTimesteps = 1:(numTimesteps-5);
predictOutputTimesteps = (numTimesteps-4):numTimesteps;
X1 = X(:,knownOutputTimesteps);
T1 = T(:,knownOutputTimesteps);
[x1,xio,aio] = preparets(net,X1,{},T1);
[y1,xfo,afo] = net(x1,xio,aio);
% Next the the network and its final states will be converted to closed-loop
% form to make five predictions with only the five inputs provided.
x2 = X(1,predictOutputTimesteps);
[netc,xic,aic] = closeloop(net,xfo,afo);
[y2,xfc,afc] = netc(x2,xic,aic);
multiStepPerformance = perform(net,T(1,predictOutputTimesteps),y2)
% Alternate predictions can be made for different values of x2, or further
% predictions can be made by continuing simulation with additional external
% inputs and the last closed-loop states xfc and afc.


% Step-Ahead Prediction Network
% For some applications it helps to get the prediction a timestep early.
% The original network returns predicted y(t+1) at the same time it is given
y(t+1).
% For some applications such as decision making, it would help to have predicted
% y(t+1) once y(t) is available, but before the actual y(t+1) occurs.
```

```
% The network can be made to return its output a timestep early by removing one
delay
% so that its minimal tap delay is now 0 instead of 1.  The new network returns
the
% same outputs as the original network, but outputs are shifted left one
timestep.
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
view(nets)
[xs,xis,ais,ts] = preparets(nets,X,{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)

% Deployment
% Change the (false) values to (true) to enable the following code blocks.
% See the help for each generation function for more information.
if (false)
  % Generate MATLAB function for neural network for application deployment
  % in MATLAB scripts or with MATLAB Compiler and Builder tools, or simply
  % to examine the calculations your trained neural network performs.
  genFunction(net,'myNeuralNetworkFunction');
  y = myNeuralNetworkFunction(x,xi,ai);
end
if (false)
  % Generate a matrix-only MATLAB function for neural network code
  % generation with MATLAB Coder tools.
  genFunction(net,'myNeuralNetworkFunction','MatrixOnly','yes');
  x1 = cell2mat(x(1,:));
  x2 = cell2mat(x(2,:));
  xi1 = cell2mat(xi(1,:));
  xi2 = cell2mat(xi(2,:));
  y = myNeuralNetworkFunction(x1,x2,xi1,xi2);
end
if (false)
  % Generate a Simulink diagram for simulation or deployment with.
  % Simulink Coder tools.
  gensim(net);
end
```

PROGRAMA DE SIMULACIÓN LA RED NEURONAL FINAL REPORTADA NO LINEAL AUTOREGRESICA CON ENTRADA EXÓGENA

```
function [y1,xf1,xf2] = NARXF(x1,x2,xi1,xi2)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 02-Mar-2014
19:44:49.
%
% [y1,xf1,xf2] = myNeuralNetworkFunction(x1,x2,xi1,xi2) takes these
arguments:
%   x = 3xTS matrix, input #1
%   1 = 3xTS matrix, input #2
%   xi1 = 3x3 matrix, initial 3 delay states for input #1.
```

```
%   xi2 = 3x3 matrix, initial 3 delay states for input #2.
% and returns:
%   y = 3xTS matrix, output #1
%   xf1 = 3x3 matrix, final 3 delay states for input #1.
%   xf2 = 3x3 matrix, final 3 delay states for input #2.
% where TS is the number of timesteps.

  % ===== NEURAL NETWORK CONSTANTS =====

  % Input 1
  x1_step1_xoffset = [-1.648446;-2.160223;-1.165047];
  x1_step1_gain = [0.724022262236519;0.55746440659447;0.481916098407267];
  x1_step1_ymin = -1;

  % Input 2
  x2_step1_xoffset = [-1.526312;-2.33664;-5.391398];
  x2_step1_gain = [0.18373480306477;0.379388145562126;0.196861089303277];
  x2_step1_ymin = -1;

  % Layer 1
  b1 = [1.7987405865286279;1.9556498280935708;1.6564221159673589;-
1.8481458830257296;1.8667774131252814;-1.6664878981246849;-
1.654654448737392;-1.575324352572582;-
1.5422809041873242;1.5479500324763718;-1.5831725829964824;-
1.4602089383361363;1.2779710915072771;-
0.96306270678996186;1.7174480150006139;-
1.2774179001105035;1.2981406887806404;-1.1873907145895655;-
1.0249430226174727;-1.4478458832949401;-
1.2550619769330971;1.3320759495113321;1.1364629552601548;1.37016319996186
35;1.063130145121387;-1.5141602547826356;1.0096899296570541;-
0.62674006404471772;-
1.0643382468779292;1.1656337280630071;0.93002959274516295;1.2950295974190
564;-0.51051884736302566;-1.0955073625422276;-1.0050418518484652;-
0.66829605647597179;0.52919403859589165;0.85282260396501475;0.72810384559
945673;0.77546525048130643;0.86475341505552106;0.93474011859185768;-
0.68806830939985808;0.60440681172704436;0.45525897849442282;0.57790164678
885036;0.61152139800504235;0.6336069612891867;0.18360963970595395;0.52312
033683189085;0.21981137362957581;0.3027188003514536;-
0.36198210074004517;0.32149478181845997;0.60100877469425862;0.46473899034
028976;0.4992928141618112;-0.20174435117812081;-0.1800104596028623;-
0.35377423837447924;-0.1515885004869596;-0.30045178612480733;-
0.47723253087820394;0.22833829377242967;-0.26445922628785301;-
0.16737905704086664;-
0.083395424075863783;0.31837857493768534;0.37159984574101568;-
0.18829328647305713;-
0.46369005708081928;0.35348277463446265;0.17098340099557102;0.07354274935
6452788;0.38470424292425609;0.27384585124660737;-
0.36061367789217674;0.40404212925000377;-0.51264413036064171;-
0.31128420075621244;-
0.5710370252263437;0.63424876038031019;0.49264405784022919;-
0.66328559547745491;-0.37458587898382362;-
0.84105598953315597;0.6518660753526625;-
1.0858625203556698;0.67996171883508494;0.95730015038321226;-
0.93974756374237067;-
0.97460756371083224;0.67459353125048982;0.90679984936732183;1.14782479526
```

```
34568;0.77022688971859687;-0.86650995009845522;0.6393515068604636;-
1.0679125359056183;1.1095211090914776;-
0.99287579459269248;1.15156570026912;1.4898161680498963;1.183263422469952
5;-1.2792098701948593;-1.1178430265319648;-1.1468271927872886;-
1.2882507370614058;-1.3377381115433469;1.0863368196661034;-
0.80106099405971554;1.1344258584376601;-
1.1941706435032351;1.8479994770774408;1.4210057959645308;1.18618114482312
39;-1.4197760754004718;-1.3234605238539141;1.650153135126917;-
1.8843279021576063;-
1.3987842740244982;1.7640979202104285;1.7483115327949057;1.55424837642884
37;1.753646589998541;1.6715743922828796;-1.6229469257718185;-
1.7373065745658056;1.7830337983444466;-1.8871730968711249];
  IW1_1 = [-0.0032869727359862893 -0.61986915914397878 -
0.5833209957506148 0.5879690311126704 0.14826852457124445
0.30198978754043149 0.57454648165881461 -0.10809104924211443 -
0.29329952245512342;0.014938911258838058 -0.042509838460594696
0.067002353845695486 -0.037012738132312442 0.17673775400561875 -
0.62878874349832381 -0.47818129986707514 -0.42167759819290729 -
0.60380647319892178;-0.088265823649316816 -0.58560309511582487 -
0.053665615249973042 0.22890348545600522 -0.52115258033151457
0.32757031847775697 0.51869570725080838 -0.036012446929900604 -
0.072635915047367591;0.24652315772676026 0.40250638872928907
0.17348409935433556 -0.25593219795478778 0.36687836529884721 -
0.12717944151905072 0.36982005129884704 -0.26815320885349625 -
0.17772482989766891;-0.33030517847040552 0.52429846852197914
0.31174177587938445 0.24263672609404061 0.29566839039513926
0.57053299878004182 -0.13042332565967005 -0.50484439257849056
0.069627675341292286;0.57329927685882931 0.060640396797548207 -
0.49098964263346839 0.15595025558086004 -0.10225806817726457
0.53811496332288178 -0.63140908633514592 -0.20727593032993538
0.39183936108937178;0.46222346310880708 0.088368691286043133
0.83230081208500828 0.23703532730116858 -0.56219732254615717 -
0.55595218433764637 0.5522435351544589 -0.42080032658589173
0.8676421939900455;0.14681537503944869 0.20913508111348422
0.77438933152367184 -0.034375263269251541 -0.38166727276134593
0.27891392958165562 -0.041617706926858178 -0.63697935347394108 -
0.78028181602853008;0.61844518846647212 0.92305817442500437
0.56645730287347984 -0.28870337531955115 -0.29999638948617496 -
0.57237948471987954 -0.65651423843701495 0.084625792799255334
0.26120910911588502;-0.3491026184280111 0.38972562208226635 -
0.4555325759181223 0.62754790484738343 -0.33741324810137824
0.53199822227878835 -0.062634987935953887 -0.44819842631227524 -
0.016271290548371189;0.46335250946315581 0.17206610683837875 -
0.052293420355994873 0.15196960849110067 -0.51701599781493834 -
0.10779620665656106 0.45857695333449833 -0.42786524394137965
0.73529093552742397;0.55187558481808807 0.030792718500461079 -
0.27175410492482938 -0.076424904949447564 0.17450892906869334 -
0.62279568031045596 -0.18809776113893173 -0.27092758194513289 -
0.63960463698833225;-0.18243330712799349 0.20567216509981809
0.008981552846068646 0.25032695904892049 -0.18066104701094049 -
0.5136070442601629 -0.1128873376177493 -0.53314571558237545
0.38974190623850652;0.51328018628714878 0.39698287365112978 -
0.54717336172158781 0.50464128034949651 -0.51906164583362313
0.095925751429687553 -0.12904167283841239 0.52447338131757193 -
0.20489479871022209;-0.38901925170340462 0.050417826147198495 -
0.014502067458420266 0.89867389129230357 -0.12380208216778525 -
```

0.36816998241892002 0.64423845114886091 0.33377277291653396 -
0.076594853064975793;0.18464392297141841 -0.3069128264475976 -
0.35953640850295471 -0.02240642230509618 -0.55018860303201811
0.26511006113443109 0.10096817718095208 -0.71229592501163241 -
0.34447926573653798;0.16411205764064643 -0.29633914286220497
0.74011184200073743 -0.57211112172501155 0.41696543309390016
0.16651570412946409 0.33416111805693277 -0.36121453966064609 -
0.078366780267436875;0.32865484123413569 0.22956911778889946 -
0.16682843399079875 0.071687167823020273 -0.19949670833746899
0.037831050187833247 0.4897977438842297 0.40690996614260277
0.5140104244964625;0.14994545217868566 -0.63370527976693092 -
0.28381816894273781 0.23911662792381044 0.19668838286013932 -
0.18141401181221706 0.073033778210118874 -0.12302758638053238 -
0.63405530607043026;0.39374818358951746 -0.1127581175838807 -
0.51785408002915201 0.10287410918117196 -0.50242066758746684
0.024074359552293763 1.008789602473084 -0.81548208543439871
0.19647268589062594;0.45972225683274287 0.039129345663266826 -
0.5907531303545811 -0.16999407739921751 -0.77154110071824611 -
0.57392116006458216 0.55062110149714993 -0.28067705952930833 -
0.054569042209769464;-0.57012173746969408 -0.1579821709066549 -
0.5434670103804683 0.45608871079541058 0.15182507152852567
0.55385793614918832 -0.43882145088796987 0.28276873490586341
0.67206146202058836;-0.2152183670825242 -0.33024296705365647 -
0.50257249482912869 0.36462097930910065 0.081815078434615343
0.008428787565879484 -0.056039973424241635 0.08911536764644204
0.65486853606258377;-0.41748246379695408 0.36762019544541719
0.21064850869968058 0.037973086553876642 -0.37603478506733551
0.069049093402547279 -0.5521980334479426 -0.10622357252818754
0.52388974725694637;-0.53247338561873592 -0.65049691257700526 -
0.71780681052872508 -0.011684735237006503 0.430057870665146
0.33209721118742525 -0.58160623168588743 0.39182471336296842 -
0.36283492731122391;-0.04127409321390503 0.84016892933287113
0.2571295447172679 0.30164769488487225 -0.03114174550259086
0.033295946999209847 0.28340600080911116 0.39665520972385065
0.35222167179362251;-0.26123187315379254 -0.3297155433458005 -
0.51357199333761927 0.61879343801971409 0.46303122587166495 -
0.093379017807720219 -0.43466586268214874 0.63859008745221801 -
0.31987592620741828;0.2511955766497978 0.40572054387809653 -
0.29690065752252709 -0.69875656727509328 0.48212816806472353
0.43425471136792337 -0.67377816110950872 -0.3586798950526634 -
0.26597842187655779;0.54385963906758772 -0.69257373998735527 -
0.16865804572161711 -0.44020032783385266 -0.00079226584045922121
0.54607240737028895 -0.36931490246379106 0.1717545847361229 -
0.61519433741769625;-0.0078103149427021486 0.20978785232574493 -
0.15930812037652112 0.43998251530468857 -0.22910014880627214
0.29230400159639641 0.16440827251694676 -0.62761749590074156 -
0.043189134925611029;-0.4137152325887935 0.20037900318994367 -
0.76027257864597875 0.16370269779539287 0.27589492521960418
0.29268248062646662 0.19776767026612768 0.0015874127164112715 -
0.029387597972605966;-0.188021128655843 0.26843594616684568
0.44614288792109719 0.37563282919480373 0.44625474430599754
0.32632488046451313 0.0075531285072217034 -0.22170320407168831
0.0048049666084220618;0.57882570347408302 0.29822732091607101
0.91258940621868423 0.082676360858873629 -0.074347492667717049 -
0.89359119699528167 0.38153526613117456 -0.38537467620691351
0.4560036483882951;0.089917570526708845 0.10052999770766192 -

```
0.41451637236536587 -0.66644104967240891 0.61341020260534107 -
0.48486391196142131 -0.34387008150881937 -0.32138531013025529
0.53497676704372676;0.73475151064071875 0.31688128133613369
0.31816978435946386 0.04101647292720164 -0.37037801572399198
0.48244075721424062 0.036950529364679213 -0.38027649699291161
0.22315629791101083;0.021193932797296852 0.41007892552438313
0.10033752750131068 0.5720795451332118 0.19158013509291771
0.43668669361127199 -0.32195474254362616 0.1913300488269071 -
0.40842048595269248;-0.2875059436835059 -0.33013810243517733
0.38162628723459818 0.02403141230673091 -0.27668547937933879
0.17389082948555601 -0.23314248013998301 0.2846066435836766 -
0.06257129123963126;0.26010478643155222 -0.1063872922732905 -
0.12289087329370034 0.74815498874279496 0.043629452600641633
0.014329955891593471 -0.73821875543243265 0.52587565005807513 -
0.79364192103668185;-0.28359850533972686 0.50743920924480002
0.15094202289762831 0.294874121672141 0.23257519199443935
0.20471717455342536 0.29978545321915312 0.69203527580659252 -
0.40934425771079425;-0.26536927817060596 -0.68747320423279834
0.39865141725138187 0.24577992054678671 0.32018010775019451
0.16349593030543821 0.95509309816295573 -0.38458870722415095 -
0.31433179480926521;-0.021653495505547331 0.30873159624674335 -
0.19200304740929441 0.27128684036285461 -0.087952141237167145 -
0.48817800415485607 -0.44443530283070437 -0.30740001663872174 -
0.59875532566260226;-0.54701670312428496 0.20168365791364251
0.38284166446616286 -0.33899588258454871 0.53237818382697799 -
0.34074116367019613 -0.13151639754146338 0.60985828813398568 -
0.25822057921245878;0.19054475061553677 0.49052104727299478
0.033606080469399002 -0.56256675724726968 -0.11262177501418862 -
0.13123690119258735 -0.45467213546239854 -0.36918326507318922 -
0.26419950323805952;-0.81497848930013639 -0.18172674523633425 -
0.084245566970957003 -0.55755218606758428 0.031049297560945387 -
0.60477114655532072 0.41343982040029781 0.0080077175141726378 -
0.37643916577497982;-0.20168844861455115 -0.37781797511826942 -
0.30190766687629339 0.13006740603370603 -0.35671521627860675
0.21845093644462149 -0.47216634144609898 0.37329268843320029
0.44563405463775413;-0.79607434951549982 0.39467356080991467
0.19575776564521047 -0.17010993305428501 0.24223434676601202 -
0.27158748093797075 0.32624652340099186 -0.18450689683693147 -
0.22077607493324752;-0.060219568912499979 -0.33629364384433746
0.14261258459258835 -0.0091724973308701628 -0.17791972074062257
0.28357161541372028 0.35403661743806158 0.32756546289849364
0.062717415036529736;-0.3870015449901173 0.24564693636511498
0.60594416649160476 0.12235354032137168 0.36560246931934243 -
0.28933893743306899 -0.18285643983551148 0.048837534704488553
0.16942398144993764;-0.33234199647730051 0.28434921586536432 -
0.39583133418667632 -0.30200888905485329 -0.28521431626470733
0.49545660322778984 -0.0015973281870923003 -0.040368863240044149
0.32699973002727423;-0.66076643676223801 0.4143943736255869 -
0.60461235160561511 -0.20920864855923141 -0.10549483483132953 -
0.32420023600156872 -0.22180880617855595 0.62581003604147689
0.0038034793422104763;-0.21647583911193921 0.0036358541708388474
0.29145569737516852 -0.049885463719476782 -0.089105020224022072
0.14718971267853401 -0.16158501198367237 -0.51890347527170866 -
0.035491458528273026;-0.26827305233156579 -0.43155432896472312
0.80613118068115941 0.16794274993832761 0.48811009264557487
0.27064481028317083 0.4203351351115952 -0.40388381836383713
```

0.57686410088414708;-0.14409858624025068 0.41699663694533629
0.13198714794342703 0.65961582850919021 -0.56852646984880395 -
0.025579714271132776 -0.33818319973659688 0.14656867176048269 -
0.68313511079571221;-0.19746011841074657 -0.029389857651067372 -
0.5453113197238918 0.62276634210310688 -0.57652618302897862 -
0.4972160349616126 0.036221261394826516 -0.045898430439845768 -
0.54119175853753265;-0.067820425254630737 -0.667570822712971520
0.3364926960684449 0.090424329412943649 0.38862335119511499
0.47688217933285754 0.36304510999341716 0.092804981963278896
0.484203506040154;-0.12075243141343693 0.51112173599069255 -
0.32106496064487627 0.066570957346961154 -0.62720455874866132 -
0.43749874759506591 0.41113061696431441 0.40435889223889021 -
0.53010602566037868;0.17453279506252753 0.36084372530216469 -
0.24953355689856732 -0.53065987495888667 0.24583283993126834
0.64509976131514635 0.39719594637012279 -0.068282993581349524 -
0.29100067612711372;0.17145991276202854 0.53075078799144959
0.27519802859965387 -0.41054172493338464 -0.33754228173911582 -
0.33637576768037614 0.48697061213575532 0.38042770278847554
0.38021052028997182;-0.45794635738151979 -0.31114498276628189 -
0.52288072404607977 -0.8141992704671680 0.016523349300224543
0.57481760955868377 0.40031315768225323 -0.089332835583285128
0.42948715871037824;0.38026548038205504 0.2654616691198553
0.6115741863836951 -0.48285324973344074 0.29676772079752001
0.18426875724869507 -0.26235339515124367 -0.17960814027197089 -
0.11882274854915888;0.22859081993838345 -0.14222937192200755
0.6405398734475154 0.40897696019952456 0.086799945381086568 -
0.38358195583791577 -0.025943921751121802 -0.78795175398647388
0.096397890582028189;0.39045665670906488 0.36321136413945665
0.51389090961597639 0.3540983771585835 0.24728523106439573 -
0.75332439748722846 0.22387513755923311 -0.30507623420291341
0.2488890885163442;0.52533420958537147 0.60647713313431006 -
0.30852017086134081 0.17152838014996927 0.056888739482641386
0.54602285886491109 0.43370699421400466 -0.342820983057809
0.20298831440668008;0.18921857469632225 -0.37265544758933111
0.11367030142509008 -0.6091784997077333 -0.45906662720158264 -
0.06081686084376646 -0.67400249588390759 0.22335546270792231
0.36652935993791125;0.47784634351014466 0.72639525276656247
0.16062129668928501 0.14158831010607331 -0.3545625307228496 -
0.31797739753549387 0.56830784656167321 0.40791915449244687 -
0.46352994025663974;-0.28349469401009292 0.32034835765607084
0.56619256188678824 0.2086503220178377 -0.39405050246976958 -
0.28905398272935429 -0.2262591051141718 -0.658295094489811O5 -
0.2011023916126651;-0.068838446234189651 0.12530983008052271
0.41046774255997875 0.21747082988901872 -0.74011373382897483
0.032449467929610773 0.2405630830061834 0.40935674575414588 -
0.61506363215943205;-0.59053204087711952 0.44711003224330531
0.0061060469691152121 0.3324387428846779 -0.573014769067137 -
0.63555900922377817 -0.012956802770622305 0.26154095145163891 -
0.72055261143040938;0.1554726505008612 0.033485830068734575
0.56713957633937895 -0.40885559908221863 0.46203721241350582 -
0.40402131795197888 -0.20160817196398811 0.76357306770977607
0.5689531222850378;-0.092946142336705881 -0.45866735874536013
0.35831993364442499 -0.28481267609011979 0.73944265596488046 -
0.38939530693840763 0.21296603144502393 -0.20874323720690474 -
0.26533521640802454;-0.08388240686984226 0.36724297013513935
0.17175713361732703 -0.43801909778204812 -0.53102425473503456 -

0.075489641037034663 0.051143490768087593 -0.32948904209250751
0.28613542264549385;0.10222298273352015 -0.22491118670365776 -
0.069828349507617224 0.28438273526302188 0.73715798867519988 -
0.30733125708584003 -0.40324045397035402 0.023562077417409538
0.41718304623953067;0.054541982995279487 0.0072574575578343041 -
0.2866112226663427 0.40522579988472773 -0.37536315958428657 -
0.10943041293852146 -0.052693255909187262 -0.2357377610631424 -
0.41019147378649495;-0.56155238718753919 0.22329185807123234
0.38123924900370393 -0.36531520785077282 0.28388396855767656 -
0.6403097534924671 -0.15571966292824174 0.2652588442577688
0.63132912871924296;0.62186719706774318 0.084227239656897004
0.30325412051756451 -0.0021340005716299379 0.18126188499050436 -
0.45847893831793995 0.37636372555482861 -0.097117301243371867
0.36245595323749236;0.27554544710745565 0.48313637939455351
0.38487711115057666 0.095366205911837559 0.24791210989299628
0.50369135477447136 0.38133262913143873 -0.19928705559248772
0.053627846225846966;-0.12064483193187057 -0.15689972047484654
0.4728045441228324 -0.26916706925811795 0.43640120807400812 -
0.57430776550710372 0.0046445598626287388 0.468753360176225 93
0.074098425943366214;0.40214817140197118 -0.39653320600405306 -
0.067909511579707552 0.1606253606577957 0.43759210977642915
0.54695900285245513 -0.60011962072558822 0.16562776223973227
0.097987776604399945;-0.37691379760256227 0.0089524573004742449
0.36038926434059704 -0.43611305778606585 0.031118541551617196
0.61597591261271478 0.55651663650716887 -0.23787201425088836
0.7597129636756137;-0.60835241554066355 -0.36399796367857484
0.26342582160643879 0.55529120745439386 -0.63666595270661963
0.16989252513944864 -0.29557752974611118 -0.60919082253445889
0.69209965071867297;-0.29790498046451697 -0.13731095492686984 -
0.16922165685535875 -0.059041703502974878 -0.20197421382328268 -
0.045018978991221162 -0.060038443303712712 0.51606621921033258
0.64469751711516143;0.55981525235346552 0.3028557279151059
0.014540505660778337 0.50744014187568975 -0.40555669849752801
0.43553154466010507 -0.5720526268673034 -0.23152600882452012 -
0.16081086532455463;0.53408985177990476 0.69594490806672271 -
0.018940455188671868 -0.34451095383978447 0.47057885370381691
0.14860457641945288 0.40298456947114747 -0.07663229141866626 -
0.46497026719234957;-0.61172945382220378 -0.48966353686363973 -
0.40580001644143415 0.1686370111809235 0.0436741341186241 14 -
0.18860285001048249 -0.1950547041557488 -0.38327824821061984
0.51250642499110155;0.17761237792107207 0.32315377040052001
0.41723021730518961 -0.01458982286472001 -0.54551484240720316 -
0.32301815703990699 0.22667334011930623 -0.10463732984592998 -
0.54206338913332797;-0.0089584975829499099 -0.029648204645840979
0.45097521652269246 -0.2505141474899768 -0.3129545440389368
0.20825284745597886 0.52189928549117837 -0.50613715460008324 -
0.32481414973832984;-0.28944114051798819 -0.30806546306010374
0.61935335259253776 -0.28391067807744924 0.053372829595943652 -
0.55453867123352407 -0.57449233477799766 0.013434912319389763 -
0.068824539499359821;-0.50059352595297313 -0.12522879153842534 -
0.4272114552224599 -0.079195609385649679 0.46070270793351603
0.24376436397172765 -0.71844512907457847 -0.21059263637760856 -
0.0091045096548569848;0.35127922659222588 -0.49036987642959545
0.13870314845339304 -0.24735610065939534 0.62457751894990332 -
0.3024422674530674 0.42766416685921343 -0.42016229328407873
0.034435412306387338;0.26177095024216906 -0.42291869012143696

0.011889198656989421 -0.083816354226259368 -0.0035837527875674444
0.15547214329074227 0.33947063131031291 -0.076005150091948193
0.36691554577599711;0.014443530860458612 -0.43677350141813315
0.79972666280811411 0.15328935010553127 0.28919711864304176 -
0.029955936214480276 0.59408464968610653 -0.74159102572889946 -
0.69475849742043172;-0.17855501375063679 -0.084824776389422291
0.527929368189894 -0.031596651588454171 0.21801683837478528 -
0.22908044969688951 -0.23594553201196619 0.83130699727326518
0.89208204154938819;-0.021423091638476291 -0.29053653518927469
0.199385203471101 -0.68753756614697115 -0.48355388491536411 -
0.018032450991329489 0.55188168839934126 0.0893814610445752
0.040212679284114747;-0.038206126962304139 0.79717386216634489
0.29325676851748411 -0.28877134272053645 -0.060061552451413862
0.31859652147407203 -0.31268858534981991 0.15772784295317685
0.1506917474294692;0.14687035638453239 0.46635001619568833
0.16525367203777996 -0.26713832881860478 0.064756630259424255 -
0.58461836447520288 0.16837434749875288 0.15146457330103497
0.71599740161189296;0.460941274318108 -0.10393968784738275 -
0.10975329325250562 0.038125372780741712 -0.36815719402501423
0.41250254999340696 -0.052773524584514939 -0.37572792300003466
0.17787324089584325;-0.40842440023978221 0.22564848741500174
0.03881412901709249 -0.33782218127733554 -0.13213663762492087 -
0.21338714687862659 -0.62681980519402114 0.41154440558788952 -
0.30797903246538527;0.36648350156416765 -0.43605390939611782 -
0.42411765463065254 0.31967072470899599 0.31799023485906436 -
0.35259194191618121 -0.0057865696708946524 -0.43953676617831105 -
0.52052489587011752;-0.40787296868362238 -0.46621403207197387 -
0.56946580933681668 -0.20675114119660146 -0.34971005821459711
0.10849837248000232 -0.33342062935864447 -0.75913439292265161
0.35636913076072663;0.3593771295062661 0.10105157184083766
0.72537885103338107 0.089175395424713949 0.09249431989577897
0.18029288310454317 -0.43178962363171081 -0.4086378714281953
0.223313105118320 62;-0.56970876079998423 0.17304761471920357 -
0.0091755238094085069 -0.39706003472614709 0.26202925186491871 -
0.59294273254937557 -0.54447720729883953 0.59945328757259408 -
0.036729951549550555;0.41961132603673207 -0.27232250236459959 -
0.32994534604785308 -0.55339486131793469 -0.33367135181355767 -
0.68078478861816427 -0.49620426477814333 0.062588549640336735
0.24161576338995983;0.3368086588094748 -0.5719236013085256 -
0.13790939499498681 0.033793443601123185 0.22225971828320223
0.66095832328451698 0.2878527747699941 0.35919227369158235 -
0.56868525548251869;0.33415793983803921 -0.77468976393866695 -
0.40371709792015736 0.63484582241818355 0.36106803123429926
0.67848260323576037 0.52356080938054428 -0.30163416168425361
0.61945289380364899;-0.30337026844506904 0.15438233499535831 -
0.082195461314261434 0.68924118872281537 -0.41121525246255958 -
0.17196423846564843 0.51277043097395913 -0.36259131961594171 -
0.32979726716310659;-0.079732270316407422 -0.15770912595018627
0.046263559782851495 -0.69435013845640881 -0.053377135765733885
0.33344181976706694 0.35811038656283295 -0.60592207350390714 -
0.58498329593320075;-0.37661509541913735 0.098545211083118034 -
0.27691025480621101 -0.011728019840071163 0.65175986909495187 -
0.60776377773076085 -0.4448724927773281 -0.099360020421768169 -
0.37696877546899515;-0.25954698085313560.0027022647555138835 -
0.38030220662717845 0.56396959339702002 -0.20863450274114742 -
0.81991479994893857 -0.70338123131798469 0.1190673848610407

0.37543880589145878;-0.3122824950975836 -0.011496696505008629 -
0.57072790495314751 -0.44177174503180278 0.31491546897095252 -
0.53615388748189508 0.42346256528039367 -0.63185589037313095
0.43974813989328354;-0.14847972853676589 -0.57151106779981053
0.30108503186592989 0.27668288467587077 0.32295200199361163
0.34192306947854473 -0.95167772620981728 -0.015992436688859746
0.54626620290268246;-0.24670454989923485 0.51107566317102582 -
0.10580761909121959 -0.30586310840637287 -0.79305805175141364 -
0.26383348499996939 0.53683977655331283 -0.39176344910483613
0.27228296544199748;0.34088779126186225 -0.098595205986212631
0.2464709189099967 -0.19650620468874866 -0.23639612166818413 -
0.33370034652419728 -0.18681410125818215 -0.65967419914994296 -
0.38938120043435026;-0.82064160427476074 0.67463713423340077 -
0.21156347713226148 -0.16369681931546495 0.42669463075132036 -
0.34735774886335158 0.6852129466982203 0.197121667764148 -
0.35694991251504382;0.17137817853323939 0.095930269613370506 -
0.20259877040340568 -0.18330384517073842 0.51055610380584615 -
0.48440633933840921 -0.59182888930204236 -0.24395638365566305
0.3808733946141265;0.12274644399343092 -0.027782252655243887 -
0.66569438601956832 0.13805030836095802 -0.64473997817054984 -
0.49455405692273019 -0.43290278689738437 0.089372103733495509 -
0.60388641467102822;0.080568109235957738 -0.21653928755751786 -
0.39748597585161044 -0.15905701077838891 -0.1534111286755338 -
0.096453161394136872 0.54723840088523101 -0.33249177930723689 -
0.27492403240341523;-0.51570726316258242 0.38621273913220305 -
0.45456251853879215 -0.73298701621283935 0.15550247471144099 -
0.66495213502066852 -0.31448599948823935 0.46274362763256188 -
0.23312646345041191;-0.30739084046520432 0.65639220665080589 -
0.26026241459960603 0.49792059111038006 -0.3455748696269994 -
0.54173892002503155 0.37672875447554344 0.40565356677781944 -
0.45876758993291006;0.35908366541975451 -0.26967287926498329
0.49348440903951873 -0.19422777007181302 0.62591512047127884
0.31016320934114244 -0.55267035072997639 -0.034181241308567756 -
0.41247050376144717;-0.37241079689915435 -0.22930486640200404 -
0.32290420407606368 0.48176665065913105 0.3002090766935831
0.64771560768848857 0.35922249994933619 -0.098412861665623741 -
0.16589480548792362;-0.61211470032813053 -0.29738328454952162 -
0.61465099814084034 -0.28333843584679674 0.7381927377861498 -
0.26134127968480481 -0.17620417403027674 0.26954029771021726
0.37309536327508053;0.67015264084627413 -0.44930227968056047
0.32033076051797832 -0.51170284274514 -0.42180498296252883
0.27326150261448301 -0.45535880805253087 -0.43457280340761661
0.36343274074469434;0.36131214397424033 0.73629474747710921 -
0.12015640162272226 0.073904710526020323 0.39131828312987754 -
0.010364854795439164 0.4916991871211972 0.60611051442683894 -
0.17995498356044518;0.24188919836815354 0.37560635220290051
0.56409336925823306 -0.40514014588280761 0.035410168451617575
0.045660807559023062 -0.20459952044848304 -0.12217435748645913
0.64428510259063854;0.16275854436724702 0.1308476093731602 -
0.41733932533387913 -0.055221603500070549 -0.15070626026331402
0.57481861610841722 -0.64129575968192898 0.78603524102011313 -
0.36600132808257929;0.81117209159313197 0.43487627993329703
0.54330605044003599 -0.23208887801207881 0.25977290041012757
0.50569693993919185 -0.71242684054359628 -0.51020326231021162
0.44016778196630851;-0.16446278790943333 0.11426133375898993
0.50182670092084669 -0.15103433803332736 -0.13112132190149436

```
0.29099671053210696 1.0157416412613225 0.092067906823393802 -
0.16352415610255444;-0.19801785342933501 0.50435636244396886 -
0.30821976935436779 -0.019420169015006858 -0.030285972457425561
0.16897724310656786 -0.27544259622832379 0.58091180453812619 -
0.14668099785464178;-0.0084263312412291708 0.0072148518049691794 -
0.49625531901855424 -0.20001880135065489 -0.16547065254093926
0.38465523546676761 -0.2312487105574379 -0.38804845381989989 -
0.280914562497938;-0.42354243241063289 0.071345982155199009
0.43381377574074764 0.62851833390930689 -0.50115261155417257 -
0.15868822697823773 -0.64358561183783647 0.50706748304932214
0.23156628559264994];
  IW1_2 = [-0.31548579033258856 0.0015952434615628557 -
0.72796745754139569 -0.72104260051886226 -0.11254670255833338
0.057826295863864126 -0.43827031021963203 0.27568934610199819 -
0.27491113612736262;-0.49098463562740202 0.57533613219168855
0.52642299461536024 -0.54710838688806118 -0.37029193661848914
0.010585849337838594 -0.26088219481394226 0.71288797021923855 -
0.39902486734934212;-0.3802819459741929 -0.75061176547031461
0.50503305511691521 -0.71421710070454314 0.39031673138144385 -
0.1396551179778639 -0.28461690299882197 -0.48269355100626821
0.090329153219928321;-0.48391011644569915 -0.42519280845387436 -
0.17604017287267962 -0.4207169881334849 -0.30708469881673583 -
0.2038957080663524 0.40541446366986328 -0.85365159122359902 -
0.55865090396641381;-0.57014230595834203 0.07524835204288266
0.47424403050384401 -0.51389156719619333 -0.097483831625587075 -
0.18141637504618607 -0.26173301854280129 -0.32758893474596218
0.53203888259363785;-0.50341126122992996 -0.5456457312075913
0.12049135746955229 -0.71850988272882854 0.21650756629116175 -
0.28457052712772446 0.74837878497700705 -0.51159718460067316 -
0.37961669976116558;0.14585181133921366 -0.56663776983353553
0.20813116730190695 0.18369545421026065 0.25108417203273042
0.22319322269579012 0.029516649932462769 -0.27346720731127239
0.26392608088842023;-0.063483539233730971 -0.089848415622886452
0.26858919871909703 0.86530188053501678 0.47729097063090803 -
0.52880396960645637 -0.12032008572335431 -0.27075833348107425
0.55596173582967812;-0.35768862581566507 -0.28800294849261399
0.263943270076476 0.70232863566349246 -0.2140292597393757 -
0.32083744538154868 -0.22011897750647397 0.017894776893919533
0.39340881304890835;0.52567229299111373 0.028587725617070004 -
0.088106204289341322 -0.37737058603139556 0.59647243162114616
0.42628986729343726 -0.56142794131619678 0.65350293013785044
0.46290933729769385;-0.53611182269884894 -0.16139161599023299 -
0.3551636139896448 -0.36363505883252911 -0.53891289186604385
0.70264345924172 -0.18120774921773497 -0.62254020997535453
0.52954670016650995;0.33357742046202427 -0.59216299085064072 -
0.19470067251446424 0.87912133324259467 -0.1234225959647347
0.75253500430669662 0.72221834080232061 0.54550370008818838
0.23720204600088879;0.3494924938784108 0.3212946663197746 -
0.21381535966797388 0.38889159219040781 -0.26475147392614184 -
0.33789403159485037 0.74264323050687742 -0.055633665885815628
0.13297649846197615;-0.044715344296507122 -0.075613799779572902
1.1704099309599914 0.6182614698117197 -0.56081387837851304 -
0.3163920524598044 -0.066852947568823803 0.24876668679246833 -
0.46255000954161063;0.27389856099950527 -0.4746330919005301 -
0.10925277914117708 0.23514332091360937 0.32212037656314618 -
0.70505197235608585 0.25717734401209819 0.38234040347257342 -
```

0.16137099034862021;0.41699009994989422 -0.15291129443624768
0.99084186720079304 0.094915000941220157 0.41255715582116792 -
0.066623330119578369 -0.22004083383667822 0.22557591588011464 -
0.10117399355490979;-0.063180140559693232 0.032882594917041749
0.082357278306925741 0.26331249564086623 0.15071375952113261 -
0.21290524612958922 0.460923740702467 -0.096340127301456052 -
0.29356021729606474;0.15941976503061239 -0.53686393748641936 -
0.28063053379673902 0.47270949254029287 -0.0011572749222637604
0.75779315200994912 -0.15347441997949288 -0.078574424187496328 -
0.73098454552889625;-0.27628379344744558 0.30603906429369448
0.73801379613782891 -0.22503329726944088 0.63472509149213074
0.24431062582351312 0.13308636701266396 -0.2787986134985212
0.34160862897634497;0.48344764390053202 -0.52316702212332789
0.013726413404778662 0.36596966989271773 0.61129737460319977 -
0.33354341390212905 -0.11831333752450295 0.96008468858843909 -
0.060460817527708384;0.22779274935009786 0.43565802798768394 -
0.59167537840348317 -0.35960672279330619 -0.30135042761450614 -
0.80621291837198239 -0.42476689694260922 -0.060637847474936633 -
0.14356448463599275;0.067310242680338001 0.55942864557749683 -
0.69731057451382372 -0.59007770567676654 -0.1355463895026231
0.010111981616619978 -0.078912912774600469 0.088002195258060689 -
0.99450337294381996;0.77417412457016865 -0.36771464091463724 -
0.20505883384137444 -0.26512379904240951 0.075237581209813414
0.31738423688828415 -0.56120330781305572 -0.26373050069348086 -
0.5851060855962843;-0.079723839943826008 0.064519285914293775 -
0.51661094474243563 -0.053246538414149704 0.6473033650548442 -
0.22061996786127175 0.062447033696062644 0.35749612506844525 -
0.58110675099814124;-0.58081358872509781 0.53418528828862266
0.43314122002777145 -0.05679582683607344 0.4341764473648419
0.1108120387365826 -0.51510790282694374 -0.044201186124001687
0.41844977963598112;0.13079154666878989 -0.52814418720623868
0.20672186715350849 -0.015996740704257863 0.38800895880619207
0.50739717821306429 0.44613098963296782 0.064857892743237652 -
0.62461354785062317;-0.60105920072495744 -0.68184206879762221 -
0.27483901132761557 0.034015179619838617 -0.24560884565316349 -
0.62498889559199844 -0.28622765274842221 0.19607172209458462 -
0.38467221265386187;0.22060937012150214 0.23831409953169561 -
0.062345642215156143 -0.23788913452616178 0.32454261268463586 -
0.36380241853790141 0.21710997028651527 -0.70701663464379216 -
0.20082616768507072;0.44620924704123077 0.13053223250593177 -
0.35678664924499354 0.0067301542543530242 0.29765546980478852 -
0.11770401099516335 0.32665790765258168 -0.54095889631037841
0.61322123066789835;-0.58694901356515949 0.56843441370595993
0.68568955088951555 0.71606975968910802 0.83896385111723348
0.13617088671407582 -0.41433405711427679 0.56603502989478272
0.4057132670353627;-0.81175837510213922 -0.17341560891771923 -
0.060562781089576148 -0.1034579196874806 0.92655855599978676 -
0.32013743577367587 -0.51679978519233649 -0.26121443756472268 -
0.058301452997192589;0.048465568117115124 -0.53647441450478017 -
0.40492343691969795 0.030143687413513202 0.43342448030499686
0.33504083204946139 -0.66575687887448531 0.10853554148578397 -
0.94343082774385112;-0.20239996057846754 -0.70316597650194179
0.53827887553118692 -0.67334973974528112 -0.12035787363914441
0.11523585415391731 0.14555368673354971 0.1996379476264899 -
0.31566293080916225;-0.88843768542675161 0.030326581119619438 -
0.73897751231574993 0.086642798805405374 0.17325699337788913 -

0.17185056168889734 0.31709947510321701 -0.20300116688024647 -
0.3180213869580007;0.14772106912163024 -0.240070083843587 -
0.54852221642984444 -0.086549609066206359 -0.57701122019039275
0.10362714644406082 0.092012167366463962 0.42124487261097815 -
0.7045067167450646;-0.23145390602927834 0.30189955469515678 -
0.4594031648045962 0.20149117837607458 -0.25689083898180254
0.48845334787026873 0.49650027259312934 0.81567523900567651
0.1083179594464289;0.072694774936454737 0.084717966079298573 -
0.26411173582676339 0.80751081052760831 0.24466977726576503 -
0.64409778753430647 -0.2076677172598641 -0.25901792069152546 -
0.69722894199467778;-0.21092722213870099 -0.2704412406994921
0.22697351521999204 -0.8693580230761091 0.30279720521447784
0.48708937212768888 0.44480743370396103 0.25115113853126653
0.23553493895193534;-0.88599912064061193 0.45987799120460698 -
0.38223940137967882 0.35289409266013938 -0.075797946435321478
0.11899754807529832 -0.91342407846131946 -0.60359364990890918
0.48984839065049629;-0.21687204410850894 0.48938145500142993 -
0.2603743206012325 0.52742149455293053 0.1245181363640387 -
0.15313564376629113 -0.39083577755177046 0.3023202931864763
0.2810144792294077;0.69427642190469685 0.51112026218952433
0.3941456088239601 -0.27532138886648921 0.046346380222607314 -
0.25030637296447567 0.40675802855089821 -0.27758321610200143
0.53618062121564913;-0.60472157070153176 -0.17169642927905335
0.81446257808290801 -0.58972870251225817 -0.5535065450161053
0.65617525667843613 0.43733399139021861 -0.034315673262245917
0.74472721743551096;-0.17842384550433302 -0.71122347221413973 -
0.56649864852523746 -0.36831001308060091 -0.46346349798248071
0.39931755935640778 0.070281553622150009 0.59017599732122783
0.012690556571798801;0.49538647124752144 -0.21879678355890619
0.3161536153832214 0.6873068149437378 0.31363045845576754 -
0.026609951597558176 -0.0090666541035637257 -0.49402751118643001 -
0.050366141591235966;-0.047707230797432103 -0.17337779817923277 -
0.0050435847732820012 0.14628776248771796 0.24567998989870707
0.38404408908401932 0.64763236080287845 0.10896139251291818
0.60561585626145698;-0.64685487024054478 -0.72789219776696723 -
0.24141405239482364 0.011453128657306584 0.44081496901789385
0.40053860489324333 -0.19434462258062676 0.092008493382497475
0.53339765747321455;0.66737025357780644 0.44795447865661858
0.43337100285580027 -0.42888491860411121 0.011080885202961471
0.20533618160290942 0.044792266382204945 0.60182604583913546
0.58763993085602484;-0.10014616471360049 -0.25084012181532489 -
0.49918170926366923 -0.56448483353879941 -0.041060843974620335
0.38526564728007906 0.21329066948022385 -0.61483332697786675 -
0.054093297331432477;-0.22212528691235908 -0.16249174996493534 -
0.28499006832754692 0.24208223710354443 -0.30465857108113426 -
0.37220427395069283 -0.68995077067763733 -0.55798686305510903
0.58995674891682204;0.082523617990420295 -0.66188253243661777 -
0.2093907850433924 -0.092818103419271186 0.50717290835208495
0.064369673155393659 -0.071745676188123053 -0.0094520624754043762 -
0.027242896882675401;-0.21091903476112475 0.33592923924977536
0.39600096485105368 0.074396692330199674 0.70541279634568832
0.5970347196048098 -0.31243261060300398 0.42447080017347338 -
0.599185127971256;-0.34375183278238902 -0.37239911020967825 -
0.39541649763577208 0.65375326224957631 0.60202147190843824 -
0.27617836502989063 -0.18132748733419335 0.052596899322685658 -
0.15026780980905874;0.58093025994236003 0.18971588788293653 -

0.81284383207432198 0.54860523668534555 -0.16483450633114005
0.67599218193369492 -0.24498755209399642 0.46146966432922137
0.2581230622709923;-0.28671056578479698 -0.44458164551842871 -
0.31541269371565811 -0.57218196377942609 0.41480331022267924
0.23581015304643335 0.074145325685180702 -0.56169455475917884 -
0.52305017698557887;0.081607233179330871 0.25906803900443109 -
0.46595030136812793 -0.14160182335887075 -0.49322597389229411
0.4491324442336696 -0.55996594152537027 0.48881032634848959
0.03387578676380594;-0.6309435870551876 -0.017461771800174906
0.090338951512412391 -0.19590618102946394 -0.082862736803112527 -
0.26998618405780089 -0.50077052731686955 -0.32785168324874331 -
0.004401385678662573;0.087373286656888441 -0.40463187348100582 -
0.73143454233623117 0.33087331962513972 -0.19735907344478226
0.62593136610916078 -0.36884043744448725 -0.61604415295018478
0.38387815471093778;0.24915725059721366 -0.59667481018106461
0.55200116168946001 -0.55980673382073465 0.012967112877788536 -
0.55176206716091303 -0.063581478511484255 0.31476334231742209
0.079512990950691947;0.26131824919752256 -0.078232329681522023 -
0.060388281609096317 -0.0058362132731842563 -0.32612164591820009 -
0.46731360721057319 -0.067941076771206796 -0.43347589626214461
0.66371122698708707;-0.30222505644027331 -0.53376119227514729 -
0.21610623300017645 -0.46729837359579895 0.038781722257572759 -
0.64781974431664824 0.19941455164154401 -0.76419644797862385 -
0.54309706022126891;0.68716408846851973 0.29735728410809659 -
0.33246053765206446 0.246064631130644 -0.38963087204477914 -
0.05147198981490636 -0.052866893794039069 -0.28771693166295587
0.70521311124010544;-0.31530738423136662 -0.48625105071358993
0.33909267216145667 -0.41907794127188491 -0.78542605415387001 -
0.20658926490544863 0.5909414093173343 0.47418910913069434
0.0051920158654737182;-0.16776119715669408 -0.11023612743308685 -
0.53523857674320274 -0.23144726580031857 0.62852980046458684
0.57189180376360427 0.58190987037999953 0.057870576079501107
0.4825669264367245;-0.56821262456249255 0.65231840887160586
0.094111001767367725 0.044722883496892969 0.71523319540295183 -
0.47941475517275217 -0.30651291412213361 -0.41555759283789279 -
0.098707643614301233;0.42681357843125683 0.29566427662413663
0.41114871662994013 0.82727574279931571 0.2282807296087368
0.60357516885775131 0.095764162268744929 -0.26078909842871123 -
0.012530535602330277;0.36563022482417029 -0.28425641532946572 -
0.48996444635757613 0.3426388595903837 -0.32536604470195613
0.81740173845474617 0.37922285395231964 0.41816773333512658 -
0.24766649881123318;0.46654817157878214 -0.71814093002722834 -
0.25590945403060233 -0.14675161734583531 -0.017860812828917244
0.10570516816248372 -0.043170293885534596 0.75208985926405281
0.023580756987084516;-0.60619174102141171 0.47365267033869135
0.51567118970282122 -0.8360405304314602 0.1192998056331435 -
0.16069934635101718 -0.7356913475621758 -0.088214301049790514
0.37105106938486632;0.098957069641920539 -0.50066515547223933 -
0.1005396204479999 -0.43144596284850345 0.48067333846895882
0.3744210901783252 0.19058992982406539 0.40937600460292128 -
0.13596135693161596;-0.69123284149855446 -0.63809341205019987 -
0.65993765223867396 0.16255912844808879 0.067354958673626689
0.11217811084434051 0.74754081745920231 -0.92296350467036292 -
0.21586747276460685;0.62235971978215998 -0.017099557672210532
0.11166182931577674 -0.58307004730459366 0.4106268730763305 -
0.084551087684077306 0.49970773275951769 -0.1079666183945838 -

0.20525160331747572;-0.84947275356006302 0.038655277492914954 -
0.19923608294789433 0.25806375626051964 0.48701743717762813 -
0.79107832506736986 0.052991600778281196 0.48082475444559902 -
0.43046598367763156;-0.87290282333315405 -0.13283972932409033
0.12303146748821599 0.7009909549672374 -0.55247321753209433
0.40347442909052555 0.24163114007473366 0.36333941214605747
0.039566364267588014;-0.046440858237681279 -0.71343071278034709 -
0.90257476509828327 0.63174518166941318 0.2169099382391779
0.045962216144724892 -0.090258429682107705 0.22718662036637635 -
0.34427881790055437;-0.053499156952179591 0.09294232186282067
0.26300114898527793 -0.10850099544296274 -0.16249559851200907 -
0.014423257610585875 0.31241841532731379 0.13302648905693798
0.01308039286975464;-0.18096070785470242 0.69370655499361988
0.13397835481545331 -0.20238331895953876 0.22664300283407707
0.59581095329070988 -0.38857733953801837 0.41525191942518158
0.31650683003011348;-0.13074123748936656 0.95669126366588186
0.063958221690180644 -0.30688592077590759 -0.41606804336259673 -
0.63748006026866977 -0.25218106648147037 0.49656984035431839
0.32421909058319393;0.085222608056375646 -0.36783659006287656
0.41066439960623025 -0.74859123920901183 0.089249692655297697 -
0.56029519214897494 0.57309691318354272 0.42739875924877241 -
0.6574823961179872;0.071004653121806052 0.16001540433073214 -
0.39757609584666981 0.037683584537409769 -0.025975860258243649
0.4528080074840326 -0.34471758707754513 0.1194365328473513 -
0.33683351848253296;-0.50590268605792887 -0.14116550813427936
0.51450566249299978 0.48096765208673908 0.173477905532778
0.12723803744651321 0.042062535731229228 -0.20548337868458652
0.18720803181769155;0.12942921332962415 0.052002490501715855 -
0.70192545679232865 0.054137442189241922 0.67693841656038178
0.4283544383138968 0.73522801713040653 0.45912554088605223 -
0.51188912127326291;0.55831946317513648 -0.0020457715043282008 -
0.50423159500201298 -0.24838831427065314 -0.2411868807931043 -
0.12805545046680503 0.26387012947547556 0.42514339229762904 -
0.14326535346113639;0.22472648939492967 0.12914776265009192 -
0.0070471782608854819 0.3771116893285299 -0.52433727693794474 -
0.34866984517155591 -0.3838931496306105 0.56914965220701219 -
0.38948867481096733;0.12834965814187668 -0.37031068945148432 -
0.32548495676312894 0.53812960911255059 -0.35306797842770271
0.3907213370347119 0.074140062312690228 0.22712117304949642
0.39441904713245962;0.44292809050252319 0.22448400780019154
0.46063606285663555 -0.53027446306616133 -0.020708883779717399 -
0.3319651299060527 -0.27784697524483354 0.23110785421312216 -
0.6144462635463801;-0.16739194293797865 -0.24564715121860717
0.066446274721058893 0.45768052913080753 -0.54341348520092436
0.22072646517770431 -0.51535132677288331 -0.42076008572639406 -
0.11386635449549229;0.71447958195308003 0.27662033695446431 -
0.19566430239369184 -0.2067275330561221 -0.70143756546843883 -
0.3412867717685687 -0.96707579527412557 0.15834836737161989
0.23698789911215076;0.071868495252735379 -0.31526059854094973
0.31579754984094888 -0.3183538853349781 -0.44742683949441098
0.37944306428161206 -0.071753070659360901 -0.33364152062839664
0.54865370449268858;-0.80940742506104002 -0.10647233253643958
0.47910917825368299 0.24711161361836456 -0.13256369232621154
0.33018699696824338 0.67359579967178718 -0.65218756230477237
1.0130344935664297;0.51373689238545472 0.71857207544246848
0.18935899562337574 0.25359001680793547 0.71389332274117623

```
0.26740719528850188 -0.55062411580154402 -0.25078279589527108
0.83877352195654675;0.21039957235250847 -0.67227647974147842
0.49517953502207929 0.28942221556572678 0.091376051013659088 -
0.21829266826098057 -0.3580944406321161 -0.59859672618855386 -
0.068075069520836778;0.087269360372170493 0.58175694102609143 -
0.25482077317584217 -0.52447848729238278 0.52812364270973755 -
0.63549111476863329 0.045287030089755424 0.62172483417876068 -
0.56093667221592536;-0.7295907424680167 -0.61588025971284621
0.016339998419240317 0.50899796899190064 -0.13012429353048699 -
0.44281829347144946 -0.15456100715096144 -0.72085513401967338 -
0.4987127604472697;0.89314842928488758 -0.018015760887119321 -
0.24909596453154068 -0.4699624551461638 0.25796935228046791
0.23233292158108565 -0.076380521911422458 -0.32525033128197306 -
0.87622852036593224;-0.41907929872611926 -0.85542764507095026 -
0.31931506621384587 0.42979700551477956 0.39955810205666681
0.33150286929684197 0.034839860672082244 0.24051300701540204
0.37488115555736162;-0.68290523315580953 0.18858940910999217 -
0.37899735461993755 0.2285845378413382 -0.73445356222929148 -
0.47407590959895551 -0.28436273337145795 -0.13104745548768656 -
0.40178522085807683;-0.16352919715906752 -0.025185754816654277 -
0.40877632948662701 -0.73412238680565045 -0.23210288970516263
0.052052245916526664 0.1500890834752551 -0.2872559464720314
0.47128662336887439;-0.053028996901237899 -0.17154726181347738
0.26211879863851933 -0.17176672731521866 -0.003936466059124173 -
0.036561122585867495 0.26831802426128898 -0.53763071193279766
0.97697268241504798;-0.023827934921700673 -0.0028706391150212368 -
0.67191962584141651 0.49653273337751652 -0.0042900625497597177 -
0.50457612544646624 0.12049379150619191 -0.21390226741166848 -
0.0499565861120932;0.28433169433249367 -0.053972765126688645
0.29695739487547712 -0.481098049440223 -0.67914669797960381 -
0.21413404964333119 -0.063921279763827901 0.20471659382702742 -
0.51712306595336477;0.60909753415286882 0.35273311811490998 -
0.84340620062701632 0.064764301561650847 -0.61427564511157029 -
0.2429550793144431 0.13712645867634443 -0.40623443142372279
0.37220580300784051;-0.17359915601515111 0.14167613010639074 -
0.26951485641478806 -0.61006844148606953 -0.33840417112658139
0.4506967256820153 -0.72247619399629803 -0.49652472533493591 -
0.39592310944399872;0.057733927080772199 -0.14696000323394012 -
0.49101733973930295 -0.69723542270168914 -0.335301288365213 -
0.32692473507442799 0.0474325642767459 0.060833646047400158 -
0.88653705727793386;0.45790900839257809 0.55702472827877707
0.18421249683591503 0.78734331760639653 0.17601613765238955 -
0.41202413437719354 -0.35794223129707542 0.111074396377863
0.32518090012006295;-0.57818685250805113 -0.96554907027396464 -
0.30255568602987382 -0.13852356243849887 0.12409098659558303
0.018451602177366671 -0.22031518997136335 -0.37431047112818927
0.38264157824618172;0.42815185839654624 -0.31643994742021758 -
0.66558296165451059 0.72133093692215378 0.40106037513452819 -
0.59403209620018083 0.45896251543647792 -0.43989575419391486 -
0.046751379862131084;-0.14960067523837894 -0.48967764840236266
0.18049117194569894 0.7786570696810331 0.53821966274390465
0.53453895864222567 -0.43725650161563434 0.51458997307040044 -
0.41422346581523845;-0.1699473461645285 0.44466870341272979 -
0.43297134407871285 -0.19842404237848871 0.44232427070997615
0.67355302127646521 0.8582644400630316 -0.74665795896952569
0.48192509229748109;-0.33855704961826782 -0.10937898146811274 -
```

```
0.1181221440370701 0.71476362820861217 -0.74831753994370787
0.12934553094444931 0.37427938828079371 -0.011321611123356005
0.41638276469591523;0.28428112825017915 -0.12330643799413973 -
0.64782193503089625 -0.6382903325812237 0.11929112819110875
0.3953402101224458 -0.39703931655145508 0.32269221359174699 -
0.53552617972994021;0.62681182980161498 -0.0046701323491480772
0.56183819655041545 -0.52050506435113542 -0.53144900012608665
0.12162167796667155 -0.57080643533103825 -0.29764458768228147 -
0.12144144412898801;0.35302063490086283 0.24616111908796748
0.67500031039285291 0.40503048727928437 -0.036051451074245303 -
0.0078816274393882467 0.37982002171621182 -0.36268223208264155 -
0.24933703623668102;0.53833061416056616 0.34899303172082458 -
0.53524365983643774 0.33884154939231459 -0.39378563639441977
0.081237952997862878 0.20211683796838167 -0.0015386527536870049 -
0.57345030586050605;0.31070871944782158 -0.39604515785847499
0.254246852418267 -0.21041863046348333 0.1477240802520616
0.27940329218280763 -0.50892139416570026 0.61385745038551398
0.54073707783000291;-0.17526869072190374 -0.18614487511316952
0.093634055350265644 -0.53801515771801511 0.088214127206958218
0.35514186761780669 -0.8071039724324458 0.43331104014451022 -
0.65040028746117939;-0.58289422829297788 -0.57402823102850598
0.30151290088080462 0.61578558635243663 -0.75918603167510512 -
0.23222159214317981 0.53950628604853079 -0.20711521192834273
0.73056602005430671;-0.19422971236155992 0.74678375528058871 -
0.28952770391262261 0.0049245800743687859 -0.56313550742118057
0.1263796517571337 -0.64738743264895326 -0.11067745706715237 -
0.029171049550278563;0.1675567051681752 -0.2747777249509874
0.72910031602113101 0.69283915905167481 -0.10185606125298617 -
0.2004168964417628 -0.2602950217708308 0.18038040109942971
0.68769926774586432;0.51020189568090935 -0.37231809366791924
0.38530887098719024 -0.27244795876000527 -0.28355251574888946 -
0.59141861171946364 -0.40981718058916589 -0.49481153558183877
0.31432912902901078;-0.26102491960205804 0.4361178348345558
0.30459068842532255 -0.2818116067148369 0.39657871834244179 -
0.58995106402852315 0.75752110201872447 -0.27963662576554238
0.26967455804481749;0.20826614264646817 -0.36691268776549124
0.85781984782733922 -0.32192091658430505 -0.00016869671857726309
0.6789242080417669 -0.39446331348664027 -0.6153033331426625 -
0.19865901379261977;-0.44809370766760054 0.36758068148186207
0.041595401977076382 0.60810921979481458 0.45819842392114812 -
0.76894314360473437 0.14349907607684775 -0.54376286405592533 -
0.4060752135167966;-0.59602649346910186 -0.35657167732082207
0.30004544469022892 -0.034062263112104869 0.25625598862545734 -
0.32300970037187854 0.43437452875342031 0.4412623657922346 -
0.21611048159117305;-0.55215163604855133 0.60455882313427134
0.78154034109556203 0.53390696653332992 0.75627385443493245
0.59619667539608823 0.62561704759871284 -0.01731718297297493
0.11686250888549936;0.23921315036874638 -0.61202460124646685 -
0.26913593282640602 -0.015388817393501958 0.46825578107332799 -
0.11243390941010321 -0.44033646613582916 -0.51857228552348 -
0.53077964094296759;0.34560382784884763 -0.08505365970517241
0.30451777553614429 0.24199363442070443 0.46729120065457791
0.26875262466798183 -0.047657795000750826 -0.0081307396675274757 -
0.67591123498949734;0.79894346278498107 0.26288517213574419
0.22314135445992897 -0.202842466496852 -0.68288941269595238 -
0.45361306210230046 -0.59850721662523287 0.23104919278034894 -
```

```
0.28384176972458114;0.71672970725933394 0.16773945644478236 -
0.48813693032309352 0.41522263875561433 0.3225639817412147 -
0.50461217601595709 -0.42635074954408758 -0.41776091766032236
0.077017746082077551;-0.080545742361315509 -0.38093392557741967 -
0.6062500190369905 0.73215700583928711 -0.92900429093525516 -
0.18268725855461343 0.092092966713053934 0.66508279288708883
0.31919204946223639;0.02772763489963025 0.21321287845573952
0.37811948586336303 0.031670521617627449 -0.69422178468464335 -
0.33429722002599693 0.53994527080996413 0.37699432829887342
0.53883343392885108];

  % Layer 2
  b2 = [-0.75907876506333949;0.34542629046981443;-0.71379748063282611];
  LW2_1 = [-0.30403752354337521 0.55361159049201603 -0.34102992500636736
0.49140581058155042 0.60482145482015581 0.35254235136380585
0.17581497032085142 -0.3612592878948892 -0.40064086836706414
0.44215839229732595 -0.26939157746720505 -0.4669786741111302 -
0.26994999081406035 0.58249807038370693 0.29905854820452282
0.32860762700232737 0.50476080573403315 0.37719514652811714 -
0.26020471224282521 -0.097322136540224119 -0.3462168092155487 -
0.76105139418132728 -0.60195316213612005 0.65715167122225393 -
0.6345068684078442 -0.78968289024767424 0.027895518634182798 -
0.99978121330161962 0.31392591323114211 0.24842574463794476
0.33492734841318639 0.736844797698745 -0.13695368532588803
0.68174560303266363 0.62693316107416841 -0.33716336151994641 -
0.51560370330673078 0.46567029586458941 -0.58284014498549186 -
0.53090849580042199 -0.29669966324082375 0.26818044453063106 -
0.5963669836407135 -0.15123930098337401 -0.0080521787333029445 -
0.38607781498316573 0.67892741969171988 0.59389348155167321
0.34315483679016323 0.69783801373321053 -0.068642616118421415
0.051588989452124943 0.15115863305418303 -0.5325281086834851 -
0.69960187339126667 -0.067387167701523323 0.098142422913178873 -
0.3001752747512656 -0.27928237497825853 -0.22672277376117952
0.82745631500124539 -0.4984865659592132 -0.46265136481842589 -
0.38763727649251206 -0.27442369539803779 -0.34556640029134217 -
0.22970377258559255 0.44679093439023204 -0.15137065760301421
0.82099450288151099 0.050608789102179551 -0.30925251007054078 -
0.72411478381395977 0.55582327751765004 0.65326365260372099
0.028488756604483769 0.33844854120584272 0.28806011666593551 -
0.12574977511249549 -0.097908186882772821 0.45561718469896184
0.6617092677824733 -0.25727610913207316 -0.104221598402875
0.47914901431504731 0.057466928128849029 -0.42543961002510666
0.91487390670290869 -0.68732274191170095 0.497149663499104 -
0.69961427083804062 -1.0484792940370691 -0.37523421950588715 -
0.9349814237350883 -0.32408641197440202 -0.082261350300648697 -
0.55534421936631806 -0.092720684644889598 0.093405867383875893 -
0.84135544299016485 0.46246851135671357 -0.19817318435086886
0.63349842275447743 -0.41142205039821544 -0.21064551450665717
0.5888409179066536 -0.33903589843429294 -0.76433176496851518
0.050179044903596676 0.10305453665697159 0.77632532357386064
0.43059277641956689 -0.33413250265370381 0.4349254209170284 -
0.18670543452285876 -0.40739293630692025 0.10095428349549181 -
0.21383591692216694 -0.51322115733602769 -0.63013585713856757 -
0.45779530808816904 -0.4644787820251749 0.43951013210984691 -
0.67107995739814907 0.089759491977649569 0.64216567069310748
```

```
0.77324053827166872 -0.92518192434153201 -0.42782012531296021
0.65177382974423326;0.47191306092344315 -0.26128523599167075
0.26632962619725759 0.45032565030076327 0.32289744014827754
1.0603232476271893 0.6646477570656063 0.85678418677611767 -
0.16973993398653195 0.55609716637638451 -0.50843780452160769 -
0.21804182576102918 0.37443274492361178 -0.37535145656715069
0.64137324437279097 -0.56813231266345321 -0.36116117872853121 -
0.29446444968493546 -0.33137892523962897 0.75029822393350243
0.37933504311593574 -0.57184892467547055 -0.15557703675818174 -
0.30370193453130689 -0.74889952348444455 0.80486696406329128 -
0.09147659710139977 0.10003475638250799 -0.95684623949568448 -
0.91486399876783353 0.55762678114917574 -0.78893455173700777 -
1.0093971351150781 0.10827392250742307 -0.33018262876940152 -
0.56391573514751903 0.52123978589893016 0.26638886446967114 -
0.31278868925785236 -0.39170296940405019 -0.19199958449735721
0.60612192390495712 -0.076327851279647974 -0.13793601128452337
0.51194927596145667 -0.044993515711519043 0.31545652815993913
0.026752792609562728 0.26994154229622125 -0.58550503953156974
0.44117505817485153 -0.43911056595116632 1.0561018505979773
0.34736212321805826 0.56275697153429516 -0.060403572100157815
0.56349803814285304 0.1289125837055414 -0.24127006190069503 -
0.063315102747480215 0.47162975492724651 0.71693497064086731 -
0.12517963449650654 0.056159388592635709 0.32381767139998152 -
0.022902619519824446 -0.51416371954486606 -0.23103383901338975
0.29297771692263685 0.25948049022943537 0.049450896608601726 -
0.26062362026369407 0.10017767766082976 -0.2381468122805063 -
0.19243722939127908 0.20858658481652889 0.84938634173952254
0.83534558776207801 0.53910592253945411 -0.25379229658299779
0.18011658037261769 -0.84818955817433761 0.25272733335433711
0.049890418444577603 0.5699317494645445 0.02688331749285237 -
0.63514971345256133 0.25669084515382995 0.68779705306377281
0.46900556970951729 -0.49720295962950817 -0.012764937347231135
0.036985691026785741 -0.065679227203537097 -0.79875375081728339
0.75646204339928447 -0.52424250704356645 0.050259804014506082
0.60808093389288376 -0.52492617112610884 -0.072826578442532347
0.25345987881863685 -0.27998068615324267 -0.45552205290616604
0.50248807020916519 0.25339383405116273 0.49351521127321896
0.26436483935771393 -0.095597675400210719 0.96493359553195601
0.71050489502821379 0.2621704763847103 0.51258583820521997
0.5112812408566656 0.85486546149555298 -0.83509565224046822 -
0.31431655831059707 0.12120644648675183 0.41179875296046814
0.036725636955795796 -0.63911185897364231 -0.17999965347391467
0.56250854008636542 0.10817433807433059 0.38335750563387272
0.23339494532053762 -0.68450224993329811 -0.16958696212211036
0.24434427008465048 0.27498052512401328;0.37187537299263673
0.80473064820674467 -0.33857041265862137 -0.467116238635615536 -
0.52177407955030486 -0.12499877254351838 0.066671099094745986 -
0.30571184221285402 -0.041662684511677713 -0.89021718455341459
0.066357921332609648 -0.26157907887604781 -0.47175667874621557
0.602574612626535 -0.16498758880953185 0.54524899621755629
0.49162234364690732 0.53881480723993491 -0.50393609200738954 -
0.37439487335883365 -0.49052452310530753 -0.13601706022156429
0.70961706057749674 0.53244293509526708 0.099367824426338708
0.0080454765700633604 0.5707791377821545 -0.28407351825905897
0.64525895766952612 0.40108641401256995 -0.40702896224990698
0.76238685208773427 0.030576304095824753 0.058621910934008092 -
```

```
0.17400725401172507 0.38728807085330141 0.12575278068615375 -
0.6063155157564557 -0.0088177754930173526 0.066122916634209064 -
0.095679359896561234 -0.20976039242978012 -0.082472218991149246
0.034010273190667836 0.47269946849812511 0.81905869666330633 -
0.60108358723512634 -0.46124003208842163 0.1730024889643933 -
0.50390813684726532 -0.12657580363660104 -0.04397075440709531 -
0.2987116918513506 0.12976159676984031 -0.27692787181013839 -
0.39189966939573673 -0.67183914574812553 0.26455954722315173 -
0.12632214611657275 -0.22288826239563014 -0.30767000820868834 -
0.085354880796873936 0.45700542362899599 0.010618717037130267
0.018878846824327899 0.31728581915820064 -0.57984205701156355 -
0.010795064089853115 -0.5109933991603437 0.221683780934089 -
0.598276514287574 -0.88543346198886308 -0.33774073409878075 -
0.05475562834956417 0.38163880040391601 0.25934186407717152
0.03555219244713971 0.46158716564964641 -0.04708592723513623 -
0.029132716551416769 -0.31682152523804535 -0.35581170011426627
0.29784702244098316 -0.28305950055616724 0.73552670643668305 -
1.0474885968102579 0.81385126009677944 0.60142808144204518 -
0.32911075525471672 0.26757980402378923 0.38058023669297192 -
0.43709199923865333 -0.17731932857762214 -0.41112306476297616
0.88416659935923714 -0.34384935432672231 0.41944103465528015 -
0.69648185800910634 0.012881859116802117 -0.42589060061199807
0.12310953170129754 -0.18171538063052098 0.21395586359667307
0.15234178942680426 0.1034887167129043 -0.19567292713812537 -
0.20180485671071943 0.29308269997813863 0.34836111445442464 -
0.39582002843786807 0.062393951871106468 0.39327580175894217 -
0.25442679353840941 -0.76322154391335295 0.26441117427151251
0.34725324584360767 -0.00026625483776232788 -0.39705680670521953
0.3377684079437227 0.92105294354148559 -0.3402836389366577 -
0.30068408250205891 0.20147623430947403 -0.33084413639930621 -
0.23434475672328789 0.039780929537788605 -0.44822822800749473 -
0.60771930422635589 -0.26369670305466125 -0.34008784819703058];

  % Output 1
  y1_step1_ymin = -1;
  y1_step1_gain = [0.18373480306477;0.379388145562126;0.196861089303277];
  y1_step1_xoffset = [-1.526312;-2.33664;-5.391398];

  % ===== SIMULATION ========

  % Dimensions
  TS = size(x1,2); % timesteps

  % Input 1 Delay States
  xd1 =
mapminmax_apply(xi1,x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);
  xd1 = [xd1 zeros(3,1)];

  % Input 2 Delay States
  xd2 =
mapminmax_apply(xi2,x2_step1_gain,x2_step1_xoffset,x2_step1_ymin);
  xd2 = [xd2 zeros(3,1)];

  % Allocate Outputs
```

```
  y1 = zeros(3,TS);

  % Time loop
  for ts=1:TS

    % Rotating delay state position
    xdts = mod(ts+2,4)+1;

    % Input 1
    xd1(:,xdts) =
mapminmax_apply(x1(:,ts),x1_step1_gain,x1_step1_xoffset,x1_step1_ymin);

    % Input 2
    xd2(:,xdts) =
mapminmax_apply(x2(:,ts),x2_step1_gain,x2_step1_xoffset,x2_step1_ymin);

    % Layer 1
    tapdelay1 = reshape(xd1(:,mod(xdts-[1 2 3]-1,4)+1),9,1);
    tapdelay2 = reshape(xd2(:,mod(xdts-[1 2 3]-1,4)+1),9,1);
    a1 = tansig_apply(b1 + IW1_1*tapdelay1 + IW1_2*tapdelay2);

    % Layer 2
    a2 = b2 + LW2_1*a1;

    % Output 1
    y1(:,ts) =
mapminmax_reverse(a2,y1_step1_gain,y1_step1_xoffset,y1_step1_ymin);
  end

  % Final delay states
  finalxts = TS+(1: 3);
  xits = finalxts(finalxts<=3);
  xts = finalxts(finalxts>3)-3;
  xf1 = [xi1(:,xits) x1(:,xts)];
  xf2 = [xi2(:,xits) x2(:,xts)];
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y =
mapminmax_apply(x,settings_gain,settings_xoffset,settings_ymin)
  y = bsxfun(@minus,x,settings_xoffset);
  y = bsxfun(@times,y,settings_gain);
  y = bsxfun(@plus,y,settings_ymin);
end

% Sigmoid Symmetric Transfer Function
function a = tansig_apply(n)
  a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
```

```
function x =
mapminmax_reverse(y,settings_gain,settings_xoffset,settings_ymin)
  x = bsxfun(@minus,y,settings_ymin);
  x = bsxfun(@rdivide,x,settings_gain);
  x = bsxfun(@plus,x,settings_xoffset);
end
```