CENTRO DE INGENIERIA Y DESARROLLO INDUSTRIAL

Especialidad de Tecnólogo en Mecatrónica

Periodo de Entrenamiento Industrial

Miguel Fabian Santoyo Esparza

Proyecto:

"APLICACIÓN DE REDES NEURONALES ARTIFICIALES PARA EL CÁLCULO DEL TORQUE Y DE LAS EMISIONES DE NO_X DE UN MOTOR DE COMBUSTIÓN INTERNA DIESEL"

Asesor:

Dr. Fernando Hernández Rosales

QUERETARO, QRO. A 5 DE SEPTIEMBRE DE 2011

ÍNDICE

I. INTRODUCCIÓN

En el presente trabajo se lleva a cabo la aplicación de las RNA para la el cálculo de ciertos parámetros de salida de la combustión del motor diesel como lo son el torque y las emisiones de NO_X, tomando la velocidad del motor y del consumo de combustible como variables de entrada.

Primeramente se explica el proyecto, sus antecedentes, la justificación de su realización, los objetivos que se pretenden alcanzar y el alcance que se tiene contemplado para este periodo de prácticas de la especialidad.

Después se procede a presentar el marco teórico que sirvió de base para la realización de este proyecto, como lo son la combustión interna del motor diesel, la teoría sobre RNA, el problema del control de la velocidad de un motor, así como el Estado del Arte en lo que se refiere al control de la combustión del motor, parte donde se explican algunos desarrollos con RNA, así como la forma en que actualmente se lleva a cabo el control de los motores, que es a través de la PCM (Power-Train Control Module).

Entrando ya en materia con el proyecto, se procede a describir las actividades realizadas durante todo el transcurso de la etapa de prácticas, para posteriormente describir el desarrollo de la RNA para este proyecto, explicando la forma en que se utilizó Matlab para la creación, configuración, entrenamiento, validación y pruebas de la red.

Finalmente se presentan los resultados de la creación de la red, su capacidad para predicción de las salidas, así como se concluye brevemente sobre los resultados y futuras aplicaciones de RNA al control de motores de combustión interna.

II. ANTECEDENTES

Los motores de combustión interna son ampliamente utilizados en casi todos los aspectos del quehacer humano, destacándose su uso en la industria automotriz, donde casi la totalidad de los vehículos que se producen aun utilizan este tipo de motores para la generación de movimiento del vehículo. Como se sabe, estos dispositivos convierten energía química en energía mecánica, a través de la conversión del movimiento alternativo del pistón en movimiento giratorio del cigüeñal.

Ahora se ha vuelto indispensable para el medio ambiente y la sustentabilidad, el hacer que los vehículos, y más propiamente los MCI (Motores de Combustión Interna) sean cada vez más eficientes, haciendo un mejor aprovechamiento del combustible, mientras que se disminuyan las emisiones de gases contaminantes, sobre todo aquellos como el CO (monóxido de carbono) y los NO_x (óxidos de nitrógeno).

Existen sistemas de control de vehículos basados en Ingeniería de Control, que controlan variables y dispositivos que intervienen en la combustión del motor con el objetivo de hacer más eficiente su funcionamiento, pero estos sistemas de control requieren de tener conocimiento del sistema a modelar, además de que una vez implementados, resulta muy complicado el modificarlos, en caso de que alguna de las condiciones del sistema se llegaren a cambiar, lo que con el paso del tiempo y el desgaste del vehículo es inevitable que suceda; además de que otros factores que pueden afectar el funcionamiento del vehículo como la humedad, temperatura, entre otros resultan ser muy difíciles de modelar mediante las técnicas de control convencionales.

III. DESCRIPCIÓN DEL PROYECTO

Una red neuronal se puede crear y entrenar para predecir arbitrariamente bien la salida o salidas de un sistema cualquiera dado un conjunto de variables de entrada, sin importar la cantidad de variables o las relaciones que están puedan tener (que sean lineales o no lineales). Las redes neuronales artificiales son por tanto, aproximadores universales de funciones. Las RNA se utilizan para modelación de sistemas altamente no lineales, multivariables y cambiantes en el tiempo.

En el presente trabajo se utilizará las herramientas de Matlab para crear una red neuronal artificial feedforward, se entrenará mediante el algoritmo de aprendizaje supervisado Backpropagation, y se probará con datos diferentes a los utilizados en el entrenamiento, con el objetivo de comprobar su ajuste en diferentes situaciones.

Como variables de entrada se tiene el consumo de combustible y la velocidad del motor diesel, y sus correspondientes datos objetivo de torque y emisiones de NO_x, se entrenará para que las salidas que entregue esta red sean los más cercanas posible a los datos objetivo presentados, hasta que se cumpla un criterio de error previamente establecido en la configuración de la red. Y ya entrenada, se le presentarán datos de consumo de combustible y de velocidad del motor (variables de entrada) y podrá predecir con gran exactitud el torque y las emisiones de NO_x (variables de salida) que tendrá el motor.

Esto quiere decir que la RNA aproximará una función de tal forma que:

$$(y_1, y_2) = f(x_1, x_2) \qquad \text{Ec. 3.1}$$

Dónde:

x_1 = Consumo de combustible
x_2 = Velocidad del motor
y_1 = Torque
y_2 = Emisiones de NO_X

Definitivamente no se podrá conocer la función que se aproximó, ya que las RNA funcionan como una 'caja negra', donde no se puede saber con certeza el proceso que se lleva a cabo como un todo, pero si cuáles son sus entradas y sus salidas y sobre todo monitorear el desempeño de la red en la predicción de esas salidas.

Con la predicción de estos datos, es posible entonces llevar a cabo cierto control de la combustión del motor, y así mejorar el rendimiento del combustible y minimizar el impacto ambiental por la emisión de gases nocivos al medio ambiente, aunque cabe señalar que este aspecto esta fuera del alcance de este proyecto.

IV. JUSTIFICACIÓN

Como ya se mencionó con anterioridad, actualmente existen sistemas de control para motores de combustión interna (MCI) dentro de la industria automotriz (que es a donde se enfoca este tipo de control) los cuales son creados básicamente utilizando técnicas de control convencionales, frecuentemente se utiliza un control repetitivo, como un PID.

El control del MCI se lleva a cabo en lo que se conoce como PCM (Power-Train Control Module). El PCM es una computadora pre-programada integrada en el vehículo, que regula el tiempo de ignición, la inyección de combustible, los dispositivos de control de emisiones, el sistema de carga, ciertas propiedades de la transmisión, control de la velocidad del motor, el ralentí (idle speed). El PCM puede adaptar su programación para controlar el MCI en condiciones variables de operación, lo cual hasta ahora ha dado resultados aceptables, dadas también las mejoras que a través del tiempo han surgido en la ingeniería de control.

Sin embargo, la tendencia actual es la implementación de sistemas inteligentes, que incorporen mayor flexibilidad, mientras que son capaces de incluir mayor cantidad de variables, independientemente del comportamiento de estas variables.

Las gráficas de la figura 4.1 muestran el comportamiento de las variables a caracterizar por la RNA. Para mayor simplicidad, se grafica cada variable de entrada con cada variable de salida, las variables de entrada en el eje *x* y las de salida en el eje *y*.

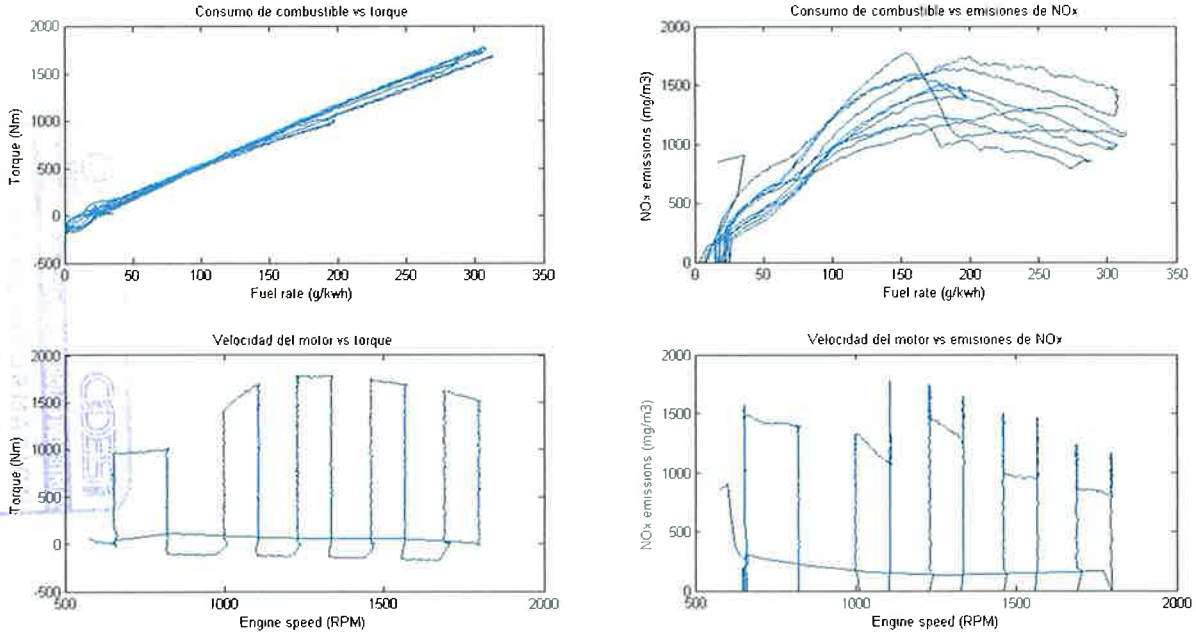

Fig. 4.1 *Comportamiento de las variables a caracterizar del motor diesel.*

Como se puede observar, solamente la relación consumo de combustible vs torque muestra comportamiento lineal, mientras que para las otras sería aún muy complicado modelar su comportamiento con el control convencional, y claramente se nota que sería muy inexacto tratar de aproximarla mediante métodos lineales.

De aquí la búsqueda de la aplicación de redes neuronales artificiales (RNA) en este tipo de control, que es multivariable, de comportamiento altamente no lineal y variable con el tiempo. Al poderse utilizar las RNA como aproximadores universales de funciones, y poder caracterizar cualquier sistema de *n* variables de entrada con *m* variables de salida, conviene entonces buscar la aplicación de esta relativamente nueva área del conocimiento en el control de la combustión de los vehículos automotores y más específicamente en los Motores de Combustión interna.

V. OBJETIVOS

GENERAL

Crear y configurar una RNA para el cálculo del torque y las emisiones de NO_x de un MCI, tomando como variables de entrada la velocidad del motor y el consumo de combustible.

PARTICULARES

- Investigar sobre el funcionamiento de los motores de combustión interna operados con diesel.
- Revisar el tema de RNA Feedforward y el algoritmo de entrenamiento Backpropagation.
- Conocer el software Matlab y sus herramientas para la creación y configuración de la RNA.
- Obtener los datos de la combustión del motor para el entrenamiento de la RNA.
- Crear y entrenar la RNA con los datos de combustión del motor.
- Crear nuevos datos, introducirlos a la red creada y compararlos con los datos objetivo, determinando la exactitud de la red en la predicción de los datos.

VI. ALCANCE

En este trabajo se emplean datos obtenidos de la base de datos de Matlab para la creación y entrenamiento de la RNA, y se logra conocer un poco más acerca de este tema que se plantea será de gran aplicación en los años venideros.

Se realizará el entrenamiento de una red Feedforward para responder al sistema suponiendo que este no tuviera cambios considerables, lo cual sirve hasta este punto para observar el buen desempeño de las RNA en el ajuste de cualquier función y en el cálculo de valores de variables de cualquier sistema.

VII. FUNDAMENTO TEÓRICO

ESTADO DEL ARTE

Control de la combustión del motor: PCM

Actualmente lo que predomina es el control de los vehículos mediante la PCM. La unidad de control de motor (PCM) es una unidad de control electrónico que administra varios aspectos de la operación de combustión interna del motor. Las unidades de control de motor más simples sólo controlan la cantidad de combustible que es inyectado en cada cilindro en cada ciclo de motor. Las más avanzadas controlan el punto de ignición, el tiempo de apertura/cierre de las válvulas, el nivel de impulso mantenido por el turbocompresor, y control de otros periféricos.

Las unidades de control de motor determinan la cantidad de combustible, el punto de ignición y otros parámetros monitorizando el motor a través de sensores. Estos incluyen: sensor MAP, sensor de posición del acelerador, sensor de temperatura del aire, sensor de oxígeno y muchos otros. Frecuentemente esto se hace usando un control repetitivo (como un controlador PID).

Antes de que las unidades de control de motor fuesen implantadas, la cantidad de combustible por ciclo en un cilindro estaba determinada por un carburador o por una bomba de inyección.

Funciones

Control de la inyección de combustible

Para un motor con inyección de combustible, una ECU determinará la cantidad de combustible que se inyecta basándose en un cierto número de parámetros. Si el acelerador está presionado a fondo, el ECU abrirá ciertas entradas que harán que la entrada de aire al motor sea mayor. La ECU inyectará más combustible según la cantidad de aire que esté pasando al motor. Si el motor no ha alcanzado la temperatura suficiente, la cantidad de combustible inyectado será mayor (haciendo que la mezcla sea más rica hasta que el motor esté caliente).

Control del tiempo de inyección

Un motor de ignición de chispa necesita para iniciar la combustión una chispa en la cámara de combustión. Una ECU puede ajustar el tiempo exacto de la chispa (llamado tiempo de ignición) para proveer una mejor potencia y un menor gasto de combustible. Si la ECU detecta un picado de bielas en el motor, y "analiza" que esto se debe a que el tiempo de ignición se está adelantando al momento de la compresión, ralentizará (retardará) el tiempo en el que se produce la chispa para prevenir la situación.

Una segunda, y más común causa que debe detectar este sistema es cuando el motor gira a muy bajas revoluciones para el trabajo que se le está pidiendo al coche. Este caso se resuelve impidiendo a los pistones moverse hasta que no se haya producido la

chispa, evitando así que el momento de la combustión se produzca cuando los pistones ya han comenzado a expandir la cavidad.

Pero esto último sólo se aplica a vehículos con transmisión manual. La ECU en vehículos de transmisión automática simplemente se encargará de reducir el movimiento de la transmisión.

Control de la distribución de válvulas

Algunos motores poseen distribución de válvulas. En estos motores la ECU controla el tiempo en el ciclo de motor en el que las válvulas se deben abrir. Las válvulas se abren normalmente más tarde a mayores velocidades que a menores velocidades. Esto puede optimizar el flujo de aire que entra en el cilindro, incrementando la potencia y evitando la mala combustión de combustible.

Control de arranque

Una relativamente reciente aplicación de la Unidad de Control de Motor es el uso de un preciso instante de tiempo en el que se producen una inyección e ignición para arrancar el motor sin usar un motor de arranque (típicamente eléctrico conectado a la batería). Esta funcionalidad proveerá de una mayor eficiencia al motor, con su consecuente reducción de combustible consumido.

Unidades programables

Una categoría especial de unidades de control de motor son aquellas que son programables. Estas unidades no tienen un comportamiento prefijado, y pueden ser reprogramadas por el usuario.

Las ECUs programables son requeridas en situaciones en las que las modificaciones después de la venta son importantes para el comportamiento final del motor. Entre estas situaciones se incluyen la instalación o cambio del turbocompresor, intercooler, tubo de escape, o cambio a otro tipo de combustible. Como consecuencia de estos cambios, la antigua ECU puede que no provea de un control apropiado con la nueva configuración. En estas situaciones, una ECU programable es la solución. Éstas pueden ser programadas/mapeadas conectadas a un computadora portátil mediante un cable USB, mientras el motor está en marcha.

La unidad de control de motor programable debe controlar la cantidad de combustible a inyectar en cada cilindro. Esta cantidad varía dependiendo en las RPM del motor y en la posición del pedal de aceleración (o la presión del colector de aire). El controlador del motor puede ajustar esto mediante una hoja de cálculo dada por el portátil en la que se representan todas las intersecciones entre valores específicos de las RPM y de las distintas posiciones del pedal de aceleración. Con esta hoja de cálculo se puede determinar la cantidad de combustible que es necesario inyectar.

Modificando estos valores mientras se monitoriza el escape utilizando un sensor de oxígeno (o sonda lambda) se observa si el motor funciona de una forma más eficiente o

no, de esta forma encuentra la cantidad óptima de combustible a inyectar en el motor para cada combinación de RPM y posición del acelerador. Este proceso es frecuentemente llevado a cabo por un dinamómetro, dándole al manejador del combustible un entorno controlado en el que trabajar.

Otros parámetros que son usualmente mapeados son:

- Ignición: Define cuando la bujía debe disparar la chispa en el cilindro.
- Límite de revoluciones: Define el máximo número de revoluciones por minuto que el motor puede alcanzar. Más allá de este límite se corta la entrada de combustible.
- Correcta temperatura del agua: Permite la adicción de combustible extra cuando el motor está frio (estrangulador).
- Alimentación de combustible temporal: Le dice a la ECU que es necesario un mayor aporte de combustible cuando el acelerador es presionado.
- Modificador de baja presión en el combustible: Le dice a la ECU que aumente el tiempo en el que actúa la bujía para compensar una pérdida en la presión del combustible.
- Sensor de oxígeno (sensor lambda): Permite que la ECU posea datos permanentes del escape y así modifique la entrada de combustible para conseguir una combustión ideal.

Algunas de las unidades de carreras más avanzadas incluyen funcionalidades como control de salida, limitación de la potencia del motor en la primera marcha para evitar la rotura de éste, etc. Otros ejemplos de funciones avanzadas son:

- Control de pérdidas: Configura el comportamiento del waste gate del turbo, controlando el boost.
- Inyección Banked: Configura el comportamiento del doble de inyectores por cilindro, usado para conseguir una inyección de combustible más precisa y para atomizar en un alto rango de RPM.
- Tiempo variable de levas: Le dice a la ECU como controlar las variables temporales en las levas de entrada y escape.
- Control de marchas.

Una ECU de carreras frecuentemente se equipa con un dispositivo de almacenamiento que graba los valores de todos los sensores para un posterior análisis usando un software especial en un ordenador. Esto puede ser muy útil para la puesta a punto del vehículo y se consigue con la observación de los datos buscando anomalías en los datos o comportamientos de las ECUs. El almacenamiento de estos dispositivos que graban los datos suele rondar entre los 0.5 y 16 megabytes.

Para conseguir la comunicación con el conductor, una ECU de carreras puede estar conectada a un "pila de datos", que es un pequeño guión de a bordo en el que el conductor puede ver las actuales RPM, velocidad y otros datos básicos del motor. Estas

zonas de almacenamiento, son mayoritariamente digitales, y se comunican con la ECU utilizando uno de los muchos protocolos entre los que se encuentran RS232, CANbus.

ECU flashing

Muchos coches recientes (fabricados en 1996 o posteriores) usan Ecus OBD-II, que son capaces de cambiar su programación a través de un puerto OBD. Entusiastas del motor con coches modernos aprovechan las ventajas de esta tecnología modificando sus motores. En lugar de utilizar un nuevo sistema de control de motor, uno puede utilizar el software apropiado para ajustar la antigua ECU. Haciendo esto, es posible mantener todas las funciones y el cableado mientras se utilizan ciertos programas de modificación de parámetros. Esto no debe ser confundido con el chip tuning, en el que el propietario tiene una ECU ROM físicamente remplazada por una distinta - este caso no requiere modificación de hardware (normalmente), aunque un equipamiento especial si es necesario.

Los sistemas de control del motor de fábrica frecuentemente poseen las mismas funcionalidades que unidades que no vienen de serie creadas para carreras, como por ejemplo tiempo tridimensional y mapas de control de combustible. Generalmente no tienen la habilidad de controlar dispositivos extras auxiliares, como el control de distribución de válvulas si el coche de fábrica tenía una geometría fija en el árbol de levas o si el control de arranque no poseía turbocompresor.

Historia

Diseño híbrido digital

El modelo híbrido digital fue popular en la mitad de los años 1980. Éste utilizaba técnicas analógicas para tomar medidas y procesaba los parámetros de entrada del motor, luego usaba una tabla almacenada en una memoria de solo lectura para obtener los valores de salida. Sistemas posteriores procesarían estas salidas dinámicamente. Este tipo de sistemas con memoria de solo lectura son fáciles de modificar si uno conoce bien el sistema. La desventaja de estos sistemas es que los valores preprocesados son sólo óptimos para un nuevo motor ideal. Este sistema no tiene la eficiencia de un sistema basado en una unidad central de procesamiento.

Los sistemas de control de motor sofisticados reciben entradas de otras fuentes, y controlan más partes del motor; como por ejemplo, los sistemas de control del tiempo de distribución de válvulas son controlados electrónicamente así como el funcionamiento del turbocompresor. Éstos además se deben comunicar con las unidades de control de transmisión o directamente con la interfaz que controla la transmisión de forma automática, sistemas de control de tracción y más sistemas con funciones similares. El cable CAN (controller area network) es frecuentemente utilizado para conseguir la comunicación entre estos dispositivos.

Unidades modernas

ECUs modernas utilizan un microprocesador que puede procesar las entradas de los sensores del motor en tiempo real. Una unidad de control electrónico contiene el hardware y el software (firmware). El hardware consiste en un conjunto de componentes electrónicos que van sobre una placa (PCB). El principal componente de este circuito en tabla es un chip microcontrolador. El software está almacenado en el microcontrolador o en otros chips de la PCB, generalmente en memorias EPROM o en memorias flash; es por ello que la CPU puede ser reprogramada actualizando el software de estas o cambiando los circuitos integrados.

Las unidades de control de motor modernas a veces incluyen control de velocidad.

Otras aplicaciones

Algunos sistemas que se usan en algunos motores de combustión también pueden tener otras aplicaciones. Como por ejemplo en aeronáutica, en los sistemas conocidos como "FADEC" (full authority digital engine controls). Este tipo de control electrónico es menos común en aeroplanos de motor de pistones que en automóviles, debido al alto coste que requieren los certificados que permiten que estas piezas puedan ser usadas para la aviación, a esto se le añade una baja demanda, y la consecuente innovación tecnológica del mercado. Además, un motor de carburador con una ignición magnética y un sistema de alimentación de combustible basado en la gravedad no requiere ninguna potencia electrónica para funcionar, lo que es un bonus en el tema de seguridad

No obstante el predominio del control convencional, existen algunos desarrollos basados en RNA para el control de los MCI, de los cuales se ahondará a continuación.

Algunos desarrollos con RNA

A continuación se describen brevemente algunos trabajos en lo que se refiere a la incorporación del uso de las redes neuronales artificiales para el control de la combustión del motor.

Predicción de factores múltiples del motor diesel con el uso de las RNA

Este trabajo fue realizado por Wenyong Xiao, investigador de Shanghai Jiao Tong University, School of Mechanical Engineering, publicado en septiembre de 2010, y en el cual se llevó a cabo la predicción de una serie de factores como velocidad del motor, torque, potencia y desempeño del combustible y del motor, basado en velocidad del motor, presión del múltiple de admisión y ángulo de avance de la bujía como variables de entrada. Se utilizó una RNA dinámica adaptativa, con entrenamiento en línea, lo cual arrojo resultados muy exactos al compararse con los reales como se muestra en las siguientes gráficas.

Datos:

1. Potencia (Kw)
2. Torque (Nm)
3. Desempeño del motor (kg/h)
4. Desempeño del combustible (g/kwh)

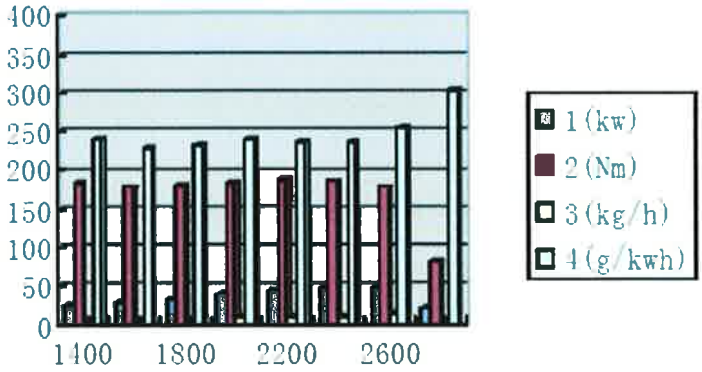

Fig. 7.1 *Datos reales obtenidos mediante medición*

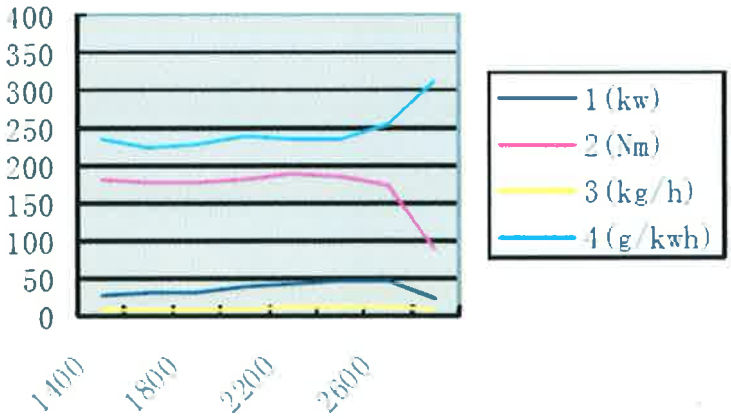

Fig. 7.2 *Datos predichos por la RNA.*

Como se puede observar el ajuste de la RNA es muy bueno en esta aplicación. Al poderse aplicar en línea, puede ser usada para controlar la combustión del motor en

tiempo real. Las desviaciones entre los valores reales y estimados se consideran despreciables en este experimento.

Evaluación del desempeño de las RNA en la predicción de las emisiones de NO_X en los motores Diesel.

Este experimento realizado por los investigadores O. Obodeh y C. I. Ajuwa del Depto. De Ingeniería Mecánica de la Universidad Ambrese Alli en Nigeria, desarrollaron una RNA para la predicción de las emisiones de NO_X, utilizando como variables de entrada la carga y la velocidad del motor, y como variables de salida se tienen las emisiones de NO_X, consumo de combustible y potencia. En laboratorio y haciendo un arreglo para el monitoreo de las variables del motor, se pudieron obtener los datos y hacer la comparación de datos experimentales contra los predichos por la RNA.

Los resultados arrojaron un error medio de 0.68 % a 3.34 % de las emisiones de NO_X. A continuación se muestra la gráfica de resultados experimentales y predichos de las emisiones para las diferentes condiciones tomadas.

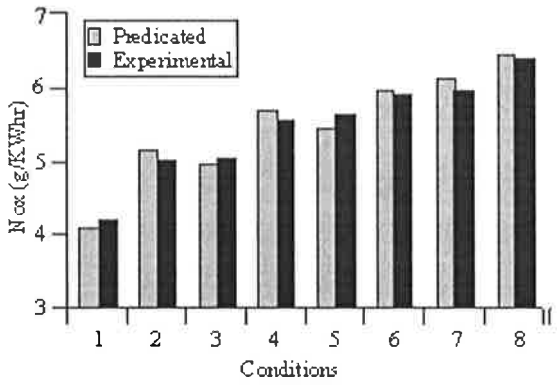

Fig. 7.3 *Datos experimentales vs predichos por la RNA.*

En este experimento las RNA muestran exactitud en la predicción de valores de la combustión en los motores.

Control inteligente del motor basado en RNA para reducción de partículas inquemadas y emisiones de NO_X del motor de combustión interna diesel.

Este trabajo fue llevado a cabo en la Universidad de West Virginia por el investigador C. Atkinson en 1997 aprox. En este un sistema de predicción de emisiones y eficiencia del combustible, basado en RNA se llevó a cabo tanto para un motor de ignición por chispa y como para uno basado en compresión, la RNA puede predecir en tiempo real la salida de potencia del motor, consumo de combustible y las emisiones usando parámetros del motor rápidamente leídos. Este sistema consistió en un modelo predictivo del motor que

fue diseñado para correr en un microprocesador en paralelo con el motor en tiempo real. tomando señales de entrada de los mismos sensores como del mismo motor. Ya en el campo el sistema es capaz de actualizar en tiempo real las nuevas relaciones entre las variables de acuerdo a cambios en el combustible, desgaste de los componentes del motor o cambios en el ambiente.

Las variables de entrada son:

- Temperatura del aire en el múltiple de admisión
- Presión del empuje en el múltiple de admisión
- Temperatura del enfriador del motor
- Temperatura de los gases de escape
- Velocidad del motor
- Temperatura y presión de combustible

Y como variables de salida la red creada es capaz de predecir:

- Salida de potencia o torque instantáneo del motor
- Consumo de combustible
- Temperatura de los gases de escape
- Emisiones en los gases de escape.

Las siguientes graficas muestran la comparación de valores medidos contra los predichos por la RNA (datos virtuales).

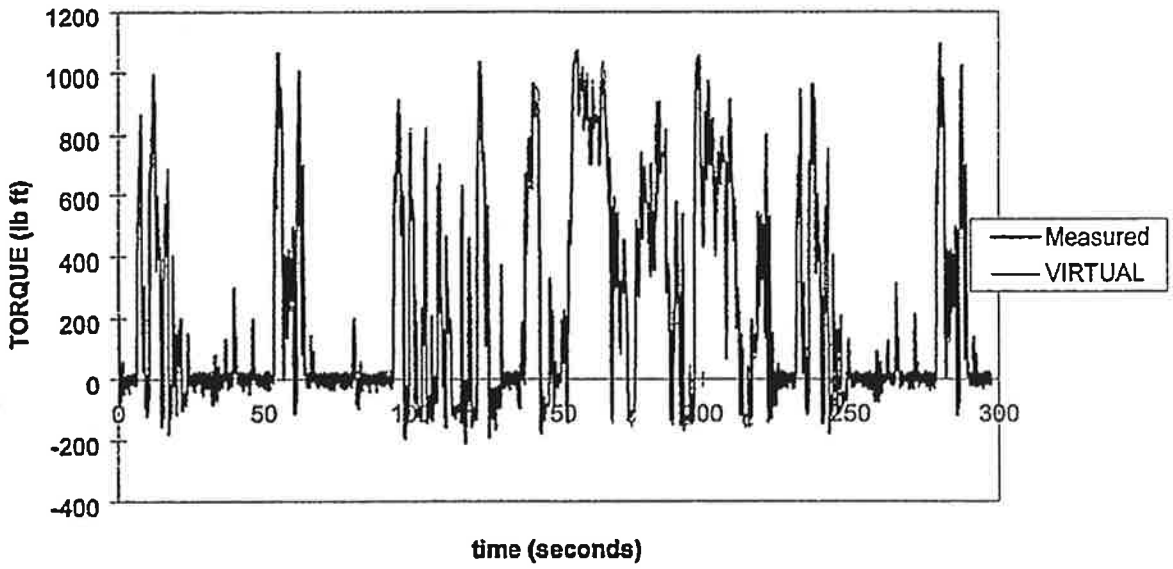

Fig. 7.4 *Datos de torque medidos vs virtuales (predichos por la RNA).*

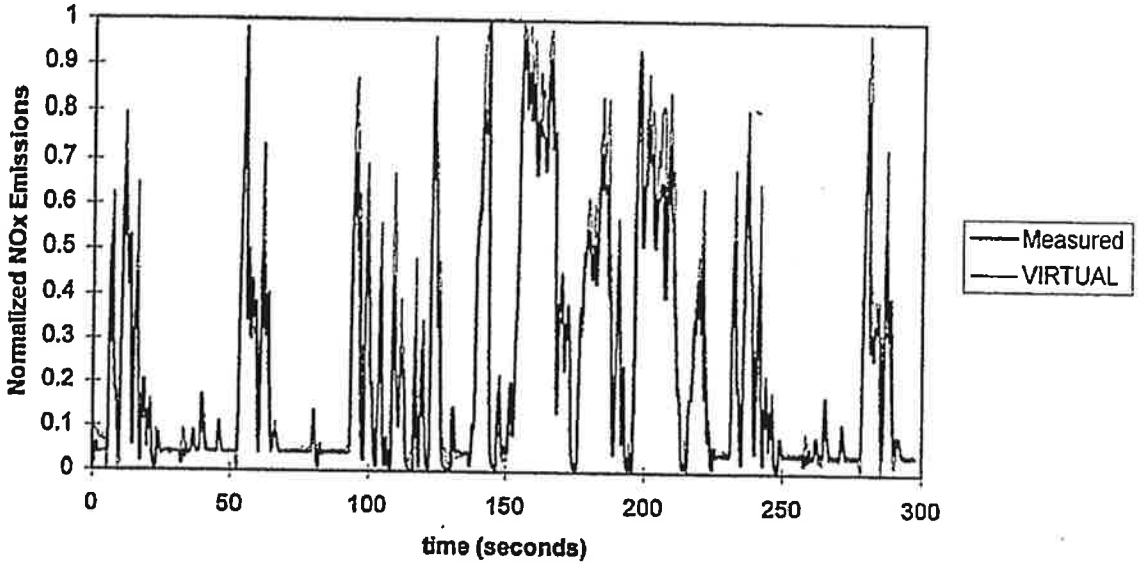

Fig. 7.5 *Datos de emisiones de NO_X medidos vs virtuales (predichos por la RNA).*

Los resultados muestran que la red creada es muy hábil para predecir estos datos, que para nuestro estudio son relevantes el torque y las emisiones de NO_X. Las RNA se conceptualizan aquí como *'sensores virtuales'* ya que sustituyen los sensores reales como predictores de los variables de salida del motor, lo que además de agilizar el procesamiento de los datos, hace mucho más económica la fabricación de vehículos.

Para este caso se realiza un entrenamiento de 30 minutos cada cierto tiempo, lo que actualizará los parámetros de la RNA para responder de forma adecuada, adaptándose a los posibles cambios sufridos por el sistema de combustión, incluso cambios en el entorno. No obstante aún se complica la aplicación de esta herramienta, ya que los requerimientos de computabilidad para la operación de la red son relativamente altos. Sin embargo, se hacen esfuerzos por alcanzar el nivel desempeño deseado por las computadoras de los vehículos.

MOTOR DE COMBUSTIÓN INTERNA DIESEL

En términos mecánicos, la construcción interna de un motor Diesel es similar a su contraparte de gasolina – componentes como pistones, bielas y un cigüeñal están presentes en ambos. Así como un motor de gasolina, un motor Diesel puede operar en un ciclo de cuatro tiempos (similar al ciclo Otto de las unidades de gasolina), o a un ciclo de dos tiempos, aunque con significativas diferencias. En ambos casos, la principal diferencia yace en el manejo de aire y combustible y el método de ignición.

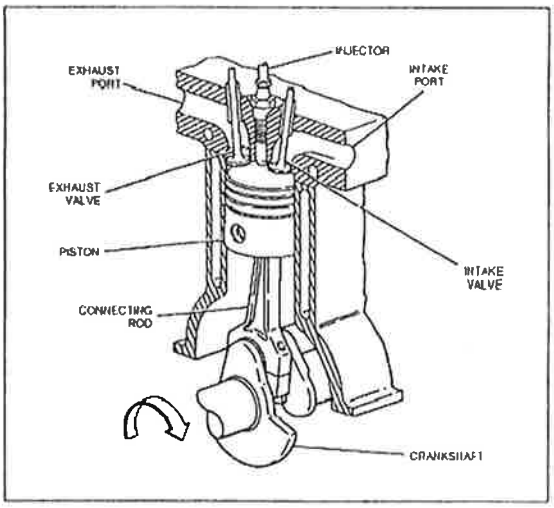

Fig. 7.6 *Motor diesel de cuatro tiempos.*

Motor Diesel de cuatro tiempos.

Un motor Diesel utiliza la compresión como ignición para quemar su combustible, en lugar de la chispa de la bujía en un motor de gasolina. Si el aire es comprimido a un alto nivel, su temperatura se incrementará a un punto tal que el combustible se quemará cuando entren en contacto. Este principio es usado por el motor Diesel de dos y cuatro tiempos para generar energía mecánica.

A diferencia de un motor de gasolina, al cual se le alimenta una mezcla de aire/combustible al cilindro durante la etapa de admisión, el motor Diesel aspira aire solamente. Enseguida de la admisión el cilindro es sellado y la carga de aire se comprime a altas presiones para calentarlo a la temperatura requerida para la ignición. Mientras en un motor de gasolina la relación de compresión raramente es mayor de 11:1 para evitar perjudicar la pre-ignición, una relación de compresión del Diesel es usualmente entre 16:1 y 25:1. Este nivel extremadamente alto de compresión ocasiona que la temperatura del aire se incremente un intervalo de 700 a 900 grados Celsius (1300 a 1650 grados Fahrenheit). Si un pedazo de acero fuera calentado a esa temperatura tomaría un rojo cereza fulgurante.

Cuando el pistón se aproxima al punto muerto superior (TDC), el combustible es inyectado al cilindro a alta presión, ocasionando que el combustible sea atomizado. A causa de la alta temperatura del aire dentro del cilindro, la ignición ocurre instantáneamente, causando un rápido y considerable incremento en la temperatura y presión en el cilindro. El pistón es impulsado hacia abajo con gran fuerza, empujando la biela y girando el cigüeñal.

Cuando el pistón está cerca de punto muerto inferior (BDC) los gases generados son expulsados desde el cilindro para el próximo ciclo. En muchos casos, los gases de escape serán usados para manejar una turbina de alimentación, la cual incrementa el volumen de la carga de aire en la admisión, resultando en una combustión más limpia y eficiente.

La secuencia de arriba generalmente describe como el motor Diesel opera. Sin embargo, hay notables diferencias entre el motor de cuatro tiempos y la versión de dos tiempos:

Motor de Combustión Interna Diesel de cuatro tiempos

El ciclo comienza con la etapa de admisión, la cual empieza cuando el pistón está en el punto muerto superior. La válvula de admisión es abierta, permitiendo el paso desde el exterior del motor (generalmente a través de un filtro de aire) hasta el puerto de admisión en la cabeza del cilindro y al cilindro mismo. Cuando el pistón va hacia el punto muerto inferior, un vacío parcial se desarrolla, causando que el aire entre al cilindro. En el caso de un motor con turbocompresor, el aire es compactado dentro del cilindro a presiones más altas que la atmosférica. Cuando el pistón pasa por el punto muerto inferior, la válvula de admisión cierra, sellando el cilindro.

La etapa de compresión, empieza cuando el pistón está en el punto muerto inferior y comienza a subir. La compresión continuará hasta que el pistón se aproxime al punto muerto superior.

La etapa de expansión, ocurre cuando el pistón alcanza el punto muerto superior al final de la etapa de compresión. En este tiempo, la inyección de combustible ocurre, resultando la combustión y la producción de trabajo útil.

El paso final es la etapa de escape, la cual empieza cuando el pistón se aproxima al punto muerto inferior después de la ignición. La válvula de escape en la cabeza del cilindro es abierta y cuando el pistón empieza a subir, los gases generados en la combustión son forzados a salir del cilindro. Cerca de punto muerto superior la válvula de admisión empezará a abrir antes de que la válvula de escape esté completamente cerrada, una condición conocida como válvula superpuesta. La superposición produce un flujo refrigerante de aire admitido sobre la válvula de escape, prolongando su vida. Una vez completada la etapa de escape el ciclo vuelve a empezar nuevamente.

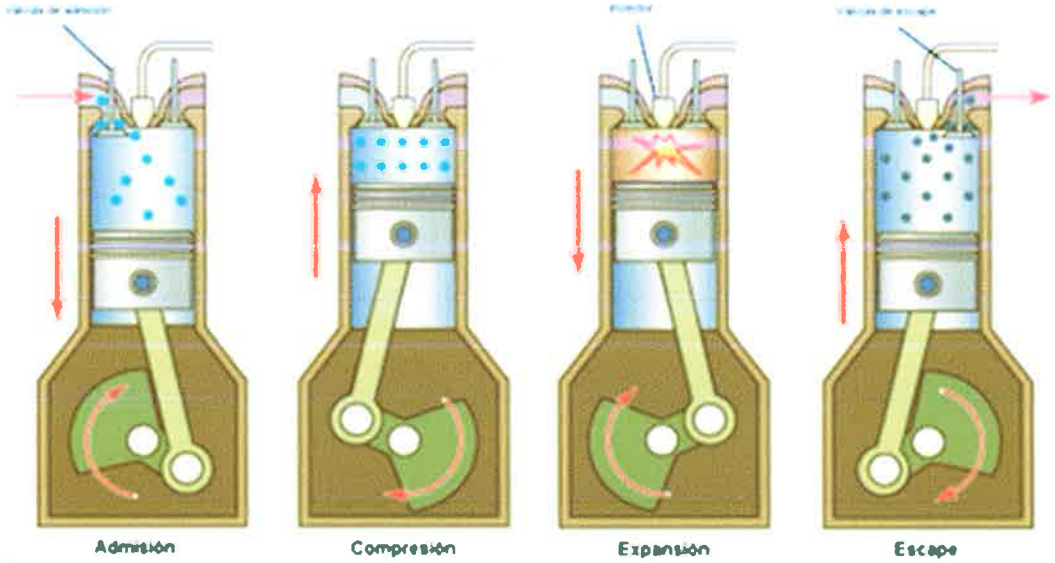

Fig. 7.7 *Ciclo de un motor Diesel de cuatro tiempos.*

Motor de Combustión Interna Diesel de dos tiempos

La admisión empieza cuando el pistón está cerca del punto muerto inferior. El aire es admitido en el cilindro a través del puerto en la pared del cilindro (no hay válvula de admisión). Puesto que el pistón se está moviendo hacia abajo en este momento, la aspiración debida a la presión atmosférica no es posible. Por lo tanto un soplador mecánico o un turbocompresor híbrido (un turbocompresor que es mecánicamente manejado por el cigüeñal a bajas velocidades del motor) es empleado para alimentar el cilindro con aire. En la fase cercana a la admisión, la carga de aire es también usada para forzar hacia fuera algún remanente de los gases de combustión de la previa etapa de potencia, un proceso conocido como limpieza. Cuando el pistón pasa por el punto muerto inferior, la válvula de escape será cerrada, y debido a la presión generada por el soplador o turbocompresor, el cilindro se llenará con aire. Una vez que el pistón empieza a moverse hacia arriba, el puerto de admisión de aire ubicado en la pared del cilindro será cubierto, sellando el cilindro. En este punto, la compresión empieza. Note que el escape y la admisión ocurre en una misma etapa, en el periodo durante el cual el pistón está cerca de la parte inferior del cilindro.

Cuando el pistón sube, la compresión toma lugar y cerca del punto muerto superior, la inyección de combustible se lleva a cabo, resultando la combustión, impulsando el pistón hacia abajo. Cuando el pistón se mueve hacia abajo en el cilindro alcanzará un punto donde la válvula de escape será abierta para expulsar los gases de combustión. Continuando el movimiento del pistón permitirá la entrada del aire por el puerto de admisión en la pared del cilindro, y el ciclo volverá a empezar de nuevo. Note que el

cilindro explotará en cada revolución, en oposición al motor de cuatro tiempos, en el cual el cilindro explota cada dos revoluciones (Hooley´s, 2000).

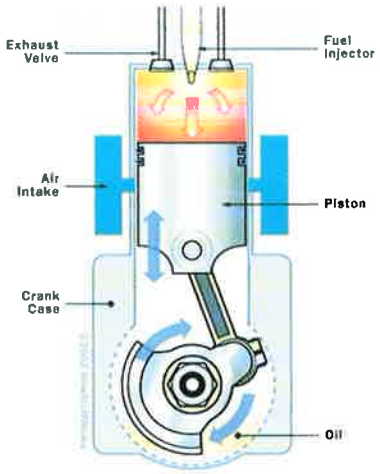

Fig. 7.8 *Ciclo de un motor Diesel de dos tiempos.*

Formación de gases de escape

El proceso de combustión, si se lleva a cabo de forma completa, genera Nitrógeno gaseoso (N_2), agua (H_2O) y bióxido de Carbono (CO_2,) sin embargo en la realidad se forman además una serie de gases contaminantes en la fase de emisión, para objeto de nuestro estudio solo consideraremos la generación de Óxidos de Nitrógeno (NO_X).

NOx

Los compuestos Oxido de Nitrógeno (NO) y Bióxido de Nitrógeno (NO_2) se denominan genéricamente como NOx. El NO es el compuesto que principalmente se produce la combustión del Diesel, cuando escapa a la atmósfera y reacciona con la luz solar y forma NO_2, que a su vez produce óxido de dinitrógeno (N_2O).

Como consecuencia, el modelado de NOx en el motor se suele simplificar restringiéndose a la predicción de NO. Los mecanismos químicos responsables de la formación de NO en los procesos de combustión son los siguientes:

Mecanismo térmico. Es el responsable principal de la formación de NO en motores. Este mecanismo se debe a la oxidación del nitrógeno del aire, y recibe su nombre de la alta temperatura de activación necesaria para romper el triple enlace del N2. Las dos primeras ecuaciones de este mecanismo fueron propuestas por Zeldovich (1946) y la tercera fue incorporada por Lavoie [Lovoie (1970)], formando el denominado "Mecanismo extendido de Zeldovich":

$$N + NO \Leftrightarrow N_2 + O$$

$$N + O_2 \Leftrightarrow NO + O$$

$$N + OH \Leftrightarrow NO + H$$

Mecanismo debido al N_2O intemedio. El NO aportado por este mecanismo es debido a que el nitrógeno molecular del aire puede reaccionar con átomos de oxígeno para formar N_2O en zonas pobres (dosados relativos menores de 0.8) y a bajas temperaturas (menores que 1500 K). Debido a su inestabilidad, casi no se mide N_2O en el escape, pero es una vía intermedia para formar NO al reaccionar con átomos de oxígeno e hidrógeno. Las reacciones más importantes que tienen en cuenta este compuesto son las siguientes:

$$N_2O + H \Leftrightarrow N_2 + OH$$

$$N_2O + O \Leftrightarrow N_2 + O_2$$

$$N_2O + O \Leftrightarrow NO + NO$$

Mecanismo prompt o súbito. Debe su nombre a la rápida aparición de NO en zonas ricas. Bajo condiciones de dosado rico, el nitrógeno molecular reacciona con los fragmentos de hidrocarburos formando compuestos intermedios como aminas o cianos, que al reaccionar con elementos oxigenados forman NO.

Mecanismo debido al nitrógeno del combustible. Este mecanismo puede contribuir de forma importante a la formación de NOx en las aplicaciones que usen combustibles fósiles de alto contenido de nitrógeno (como el carbón), pero este no es el caso de los combustibles Diesel. En el caso de los motores Diesel, donde se alcanzan altas presiones en un rango amplio de temperaturas, las fuentes principales de formación de NO son el mecanismo de formación debido la vía térmica y al N_2O intermedio.

SISTEMA ESTÁNDAR PARA CONTROL DE LA VELOCIDAD DE UN MOTOR

Los elementos básicos del sistema de control de velocidad para un motor que son de la clase que resulta en el desarrollo de reguladores son:

- Una válvula con cual cambiar la cantidad de combustible alimentada al motor
- Una maquina reciproca
- Un eje de salida con un volante y carga para la maquina

La válvula de entrada es una parte muy importante de este sistema. Una válvula es un elemento no lineal con una zona muerta en su modelo matemático de entrada. La zona muerta de la válvula de entrada, normalmente ocasionada por la fricción estática, es una característica muy importante para el diseño de gobernadores de velocidad centrífugos. Dicha zona muerta se debe incluir en el diseño del control del sistema. La válvula de entrada de dicho sistema esta motorizada de manera que un entrada constante de control en la válvula causa un rango constante de cambio en la posición de la válvula. La función característica que relaciona el flujo de combustible contra la posición de la válvula es también no lineal, de tal forma que controlar la entrada de combustible al sistema es un problema de control por sí mismo.

El siguiente diagrama muestra el sistema estándar de control de velocidad para el motor.

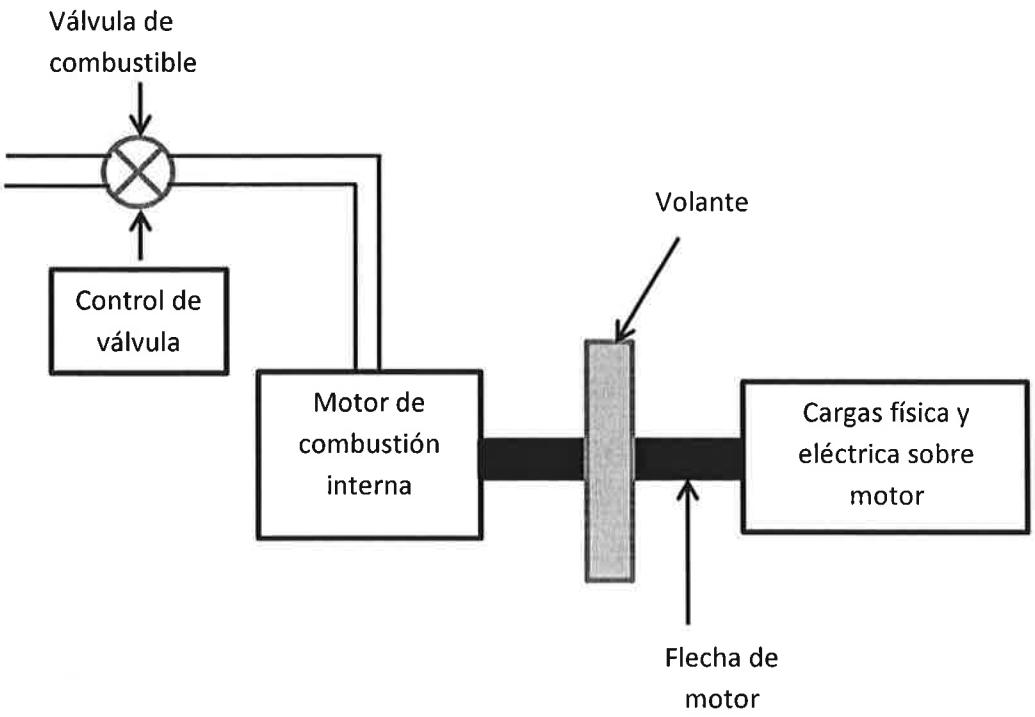

Fig. 7.9 *Control del motor.*

Problemas de control de un sistema estándar de control de velocidad de motor

El problema estándar del control de motor como se muestra en la figura tiene dos componentes principales, que pueden también encontrarse en muchas situaciones de control, y lo son

- La necesidad de controlar un actuador de entrada no lineal (como la válvula de control de flujo de combustible).
- El requerimiento para regular la velocidad de la maquina cuando la carga en el eje de salida cambia.

La forma de enfrentar estos problemas es cotidianamente mediante el uso del control convencional, con la compensación de la zona muerta en el actuador, el control en cascada y la predicción de la carga. Estos y otros métodos de control son incorporados en la PCM para los vehículos de combustión interna.

REDES NEURONALES ARTIFICIALES

El modelo Biológico

Se estima que el cerebro humano contiene más de cien mil millones de neuronas; estudios sobre la anatomía del cerebro humano concluyen que hay más de 1000 sinapsis la entrada y a la salida de cada neurona. Es importante notar que aunque el tiempo de conmutación de la neurona (*unos pocos milisegundos*) es casi un millón de veces menor que en los actuales elementos de las computadoras, ellas tienen una conectividad miles de veces superior que las actuales supercomputadoras [*6*].

Las neuronas y las conexiones entre ellas (sinapsis) constituyen la clave para el procesado de la información.

Algunos elementos a destacar de su estructura histológica son:

Las dendritas, que son la vía de entrada de las señales que se combinan en el cuerpo de la neurona. De alguna manera la neurona elabora una señal de salida a partir de ellas.

El axón, que es el camino de salida de la señal generada por la neurona.

Las sinapsis, que son las unidades funcionales y estructurales elementales que median entre las interacciones de las neuronas. En las terminaciones de las sinapsis se encuentran unas vesículas que contienen unas sustancias químicas llamadas neurotransmisores, que ayudan a la propagación de las señales electroquímicas de una neurona a otra.

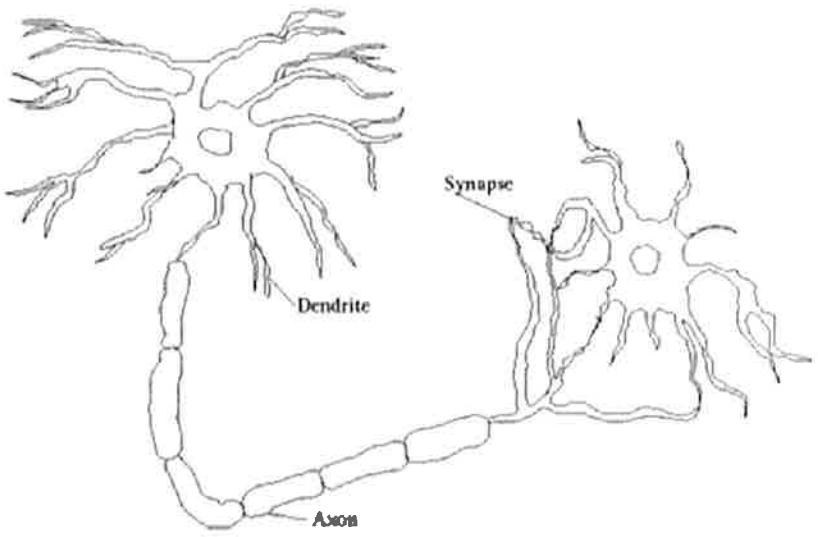

Fig. 7.10 *La neurona biológica*

Lo que básicamente ocurre en una neurona biológica es lo siguiente: la neurona es estimulada o excitada a través de sus entradas (inputs) y cuando se alcanza un cierto umbral, la neurona se dispara o activa, pasando una señal hacia el *axón*.

Posteriores investigaciones condujeron al descubrimiento de que estos procesos son el resultado de eventos electroquímicos. Como ya se sabe, el pensamiento tiene lugar en el cerebro, que consta de millones de neuronas interconectadas. Así, el secreto de la "inteligencia" -sin importar como se defina- se sitúa dentro de estas neuronas interconectadas y de su interacción.

La forma que dos neuronas interactúan no está totalmente conocida, dependiendo además de cada neurona. En general, una neurona envía su salida a otras por su axón. El axón lleva la información por medio de diferencias de potencial, u ondas de corriente, que depende del potencial de la neurona.

Este proceso es a menudo modelado como una regla de propagación representada por la función de red u(.). La neurona recoge las señales por su sinapsis sumando todas las influencias excitadoras e inhibidoras. Si las influencias excitadoras positivas dominan, entonces la neurona da una señal positiva y manda este mensaje a otras neuronas por sus sinapsis de salida. En este sentido la neurona puede ser modelada como una simple función escalón f(.). Como se muestra en la próxima figura, la neurona se activa si la fuerza combinada de la señal de entrada es superior a un cierto nivel, en el caso general el valor de activación de la neurona viene dado por una función de activación f(.).

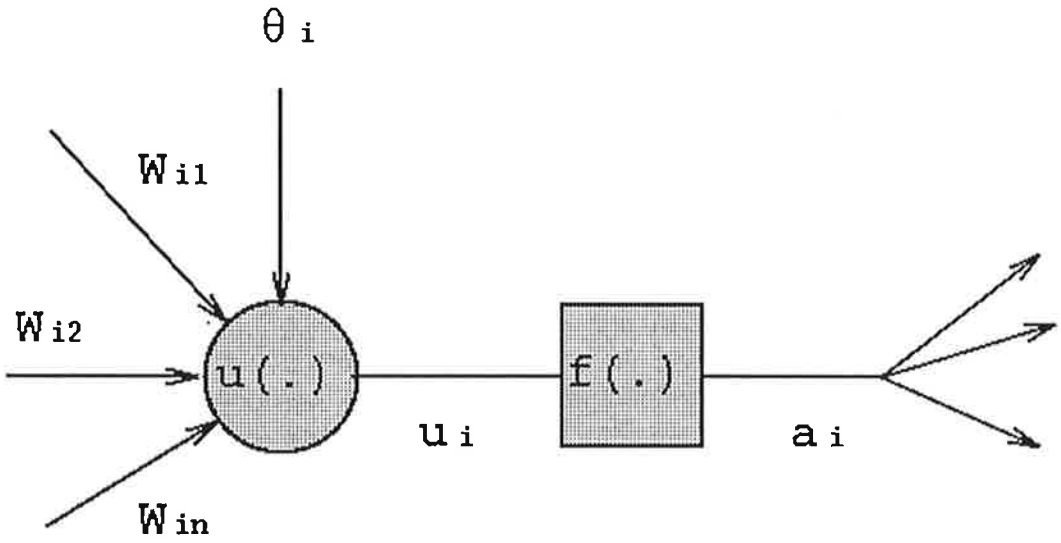

Fig. 7.11 *Neurona artificial*

Otras definiciones de Redes Neuronales

Una red neuronal es un procesador masivamente paralelo distribuido que es propenso por naturaleza a almacenar conocimiento experimental y hacerlo disponible para su uso. Este mecanismo se parece al cerebro en dos aspectos:

- El conocimiento es adquirido por la red a través de un proceso que se denomina aprendizaje.
- El conocimiento se almacena mediante la modificación de la fuerza o peso sináptico de las distintas uniones entre neuronas.

Una red neuronal es un **modelo computacional** con un conjunto de propiedades específicas, como son la habilidad de adaptarse o aprender, generalizar u organizar la información, todo ello basado en un procesamiento eminentemente paralelo.

Elementos de una Red Neuronal Artificial

A continuación se puede ver en la siguiente figura, un esquema de una red neuronal:

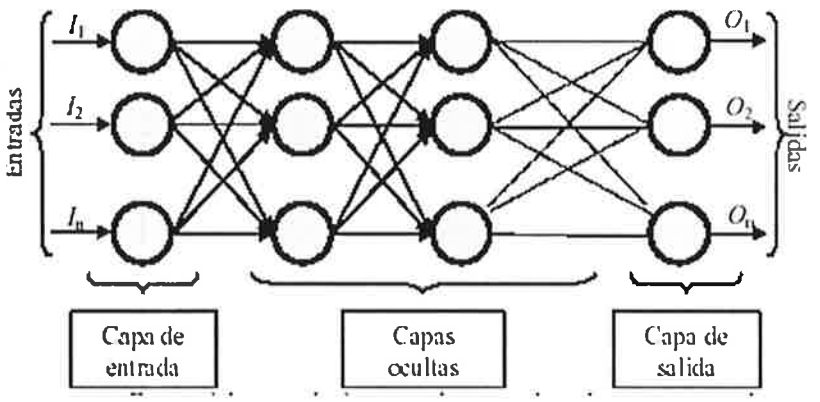

Fig. 7.12 *Red neuronal artificial*

La misma está constituida por neuronas interconectadas y arregladas en tres capas (esto último puede variar). Los datos ingresan por medio de la "capa de entrada", pasan a través de la "capa oculta" y salen por la "capa de salida". Cabe mencionar que la capa oculta puede estar constituida por varias capas.

En la siguiente figura se compara una neurona biológica con una neurona artificial. En la misma se pueden observar las similitudes entre ambas (tienen entradas, utilizan pesos y generan salidas).

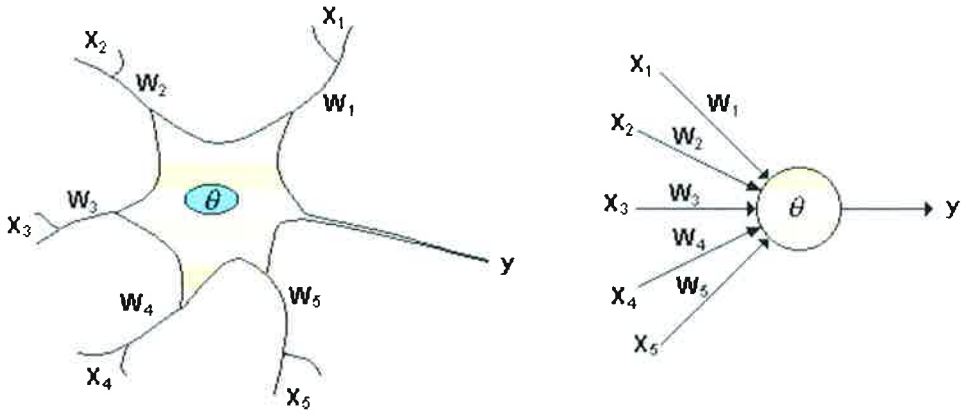

Fig. 7.13 *Comparación de la Neurona biológica vs la neurona artificial*

Unidades de proceso: La neurona artificial

Si se tienen N unidades (neuronas), podemos ordenarlas arbitrariamente y designar la i-ésima unidad como U_i. Su trabajo es simple y único, y consiste en recibir las entradas de las células vecinas y calcular un valor de salida, el cual es enviado a todas las células restantes.

Cada neurona i-ésima está caracterizada en cualquier instante por un valor numérico denominado valor o estado de activación $a_i(t)$; asociado a cada unidad, existe una función de salida, f_i, que transforma el estado actual de activación en una señal de salida. Dicha señal es enviada a través de los canales de comunicación unidireccionales a otras unidades de la red; en estos canales la señal se modifica de acuerdo con la sinapsis (el peso, w_{ji}) asociada a cada uno de ellos según determinada regla. Las señales moduladas que han llegado a la unidad j-ésima se combinan entre ellas, generando así la entrada total Net_j.

$$Net_j = \sum_i y_i w_{ji} \qquad \text{Ec. 7.1}$$

Una función de activación, F, determina el nuevo estado de activación $a_j(t+1)$ de la neurona, teniendo en cuenta la entrada total calculada y el anterior estado de activación $a_j(t)$.

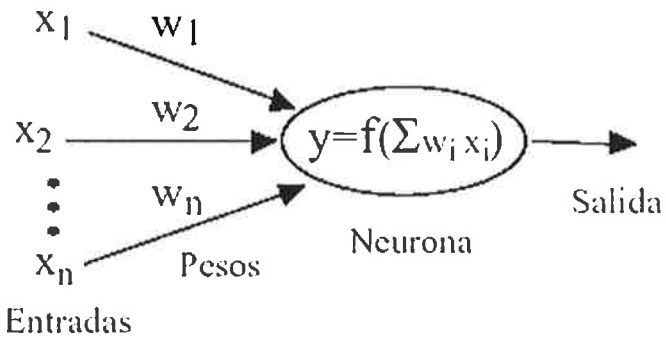

Fig. 7.14 *Calculo de la salida de la neurona artificial*

En cualquier sistema que se esté modelando, es útil caracterizar tres tipos de unidades: entradas, salidas y ocultas. Las unidades de entrada reciben señales del entorno, éstas pueden ser provenientes de censores o de otros sectores del sistema. Las unidades de salida envían la señal fuera del sistema; éstas pueden controlar directamente potencias u otros sistemas. Las unidades ocultas son aquellas cuyas entradas y salidas se encuentran dentro del sistema; es decir no tienen contacto con el exterior.

Se conoce como nivel o capa a un conjunto de neuronas cuyas entradas provienen de la misma fuente, y cuyas salidas se dirigen a un mismo destino.

Estado de Activación

Junto al conjunto de unidades, la representación necesita los estados del sistema en un tiempo *t.* Esto se especifica en un vector de N números reales A(t), que representa el *estado de* activación del conjunto de unidades de procesamiento. Cada elemento del vector representa la activación de una unidad en el tiempo *t.* La activación de una unidad U_i en el tiempo *t* se designa por $a_i(t)$; es decir:

$$A(t) = (a_1(t),.....a_i(t),.....a_N(t)) \qquad \text{Ec. 7.2}$$

El procesamiento que realiza la red se ve como la evolución de un patrón de activación en el conjunto de unidades que lo componen a través del tiempo.

Todas las neuronas que componen la red se hallan en cierto estado. Podemos decir que hay dos posibles estados, *reposo* y *excitado*, a los que denominaremos *estados de activación* y a cada uno de los cuales se le asigna un valor. Los valores de activación pueden ser continuos o discretos. Además pueden ser limitados o ilimitados. Si son discretos, suelen tomar un conjunto pequeño de valores o bien valores binarios. En notación binaria, un estado activo se indicaría por un 1, y se caracteriza por la emisión de un impulso por parte de la neurona (potencial de acción), mientras que un estado pasivo se indicaría por un 0. En otros modelos se considera un conjunto continuo de

estados de activación, en cuyo caso se asigna un valor entre [0,1] o en el intervalo [-1,1], generalmente siguiendo una función sigmoidal.

Los criterios o reglas que siguen las neuronas para alcanzar tales estados dependen de dos factores:

Dado que las propiedades macroscópicas de las redes neuronales no son producto de actuación de elementos individuales, es necesario tener idea del mecanismo de interacción entre las neuronas. El estado de activación estará fuertemente influenciado por tales interacciones ya que el efecto que producirá una neurona sobre otra será proporcional a la fuerza, peso de la conexión entre ambas.

La señal que envía cada una de las neuronas a sus vecinas dependerá de su propio estado de activación.

Función de transferencia

Así como es necesaria una regla que combine las entradas de una neurona con los pesos de las conexiones, también se requiere una regla que combine las entradas con el estado actual de la neurona para producir un nuevo estado de activación. Esta función F produce un nuevo estado de activación en una neurona a partir del estado (a_i) que existía y la combinación de las entradas con los pesos de las conexiones (net_i).

Dado el estado de activación $a_i(t)$ de la unidad U_i y la entrada total que llega, Net_i , el estado de activación siguiente, $a_i(t+1)$, se obtiene aplicando una función F, llamada *función de activación.*

$$a_i(t+1) = F(a_i(t), Net_i) \qquad \text{Ec. 7.3}$$

En la mayoría de los casos la función F es la *función identidad*, por lo que el estado de activación de la neurona en *t+1* coincidirá con el Net de la misma t. En este caso, el parámetro que se le pasa a la función de salida f , de la neurona será directamente el Net. Es estado de activación anterior no se tiene en cuenta. Según esto, la salida de una neurona $i(y_i)$ quedará según la expresión:

$$y_i(t+1) = f(Net_i) = f(\sum_{j=1}^{N} w_{ji} y_j(t)) \qquad \text{Ec. 7.4}$$

Por tanto, y en lo sucesivo, consideraremos únicamente la función f , que denominaremos de transferencia o de activación. Además, la misma no está centrada en el origen del eje que representa el valor de entrada neta sino que existe cierto desplazamiento debido a las características internas de la neurona y que no es igual en todas ellas. Este valor se denota como θ_i y representa el umbral de activación de la neurona *i.*

$$y_i(t+1) = f(Net_i - \theta_i) = f(\sum_{j=1}^{N} w_{ji} y_j(t) - \theta_i) \qquad \text{Ec.7.5}$$

La salida se obtiene en una neurona para las diferentes forma de la función f serán:

Función escalón

Si el conjunto de los estados de activación es E ={0,1}, tenemos que:

$$y_i(t+1) = \begin{cases} 1 \; si \; [Net_i > \theta_i] \\ y(t) \; si \; [Net_i = \theta_i] \\ 0 \; si \; [Net_i < \theta_i] \end{cases} \qquad \text{Ec. 7.6}$$

Si el conjunto es E = {-1,1}, tendremos que:

$$y_i(t+1) = \begin{cases} 1 \; si \; [Net_i > \theta_i] \\ y(t) \; si \; [Net_i = \theta_i] \\ -1 \; si \; [Net_i < \theta_i] \end{cases} \qquad \text{Ec. 7.7}$$

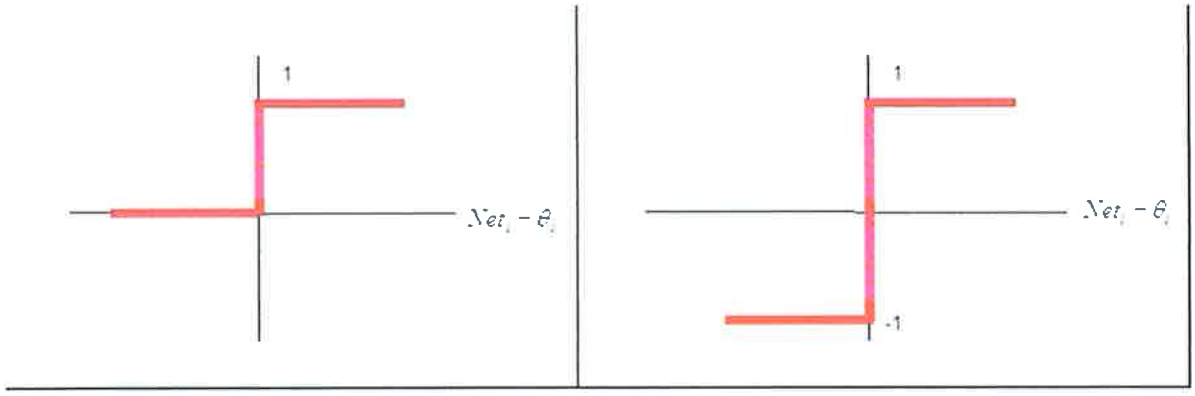

Fig. 7.15 *Función de Transferencia escalón*

Función lineal o identidad

El conjunto de estados E puede contener cualquier número real; el estado de activación coincide con la entrada total que ha llegado a la unidad.

$$y_i(t+1) = Net_i - \theta_i$$ Ec. 7.8

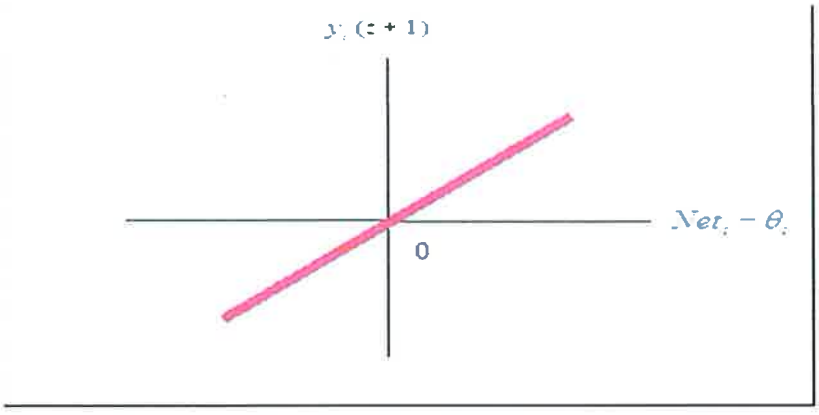

Fig. 7.16 *Función de Transferencia lineal*

Función lineal-mixta

$$y_i(t+1) = \begin{cases} b & si\ [Net_i <= b + \theta_i] \\ Net_i - \theta_i si & b + \theta_i < Net_i < B + \theta_i \\ B & si\ [Net_i >= B] \end{cases}$$ Ec. 7.9

Con esta función, el estado de activación de la unidad está obligado a permanecer dentro de un intervalo de valores reales prefijados.

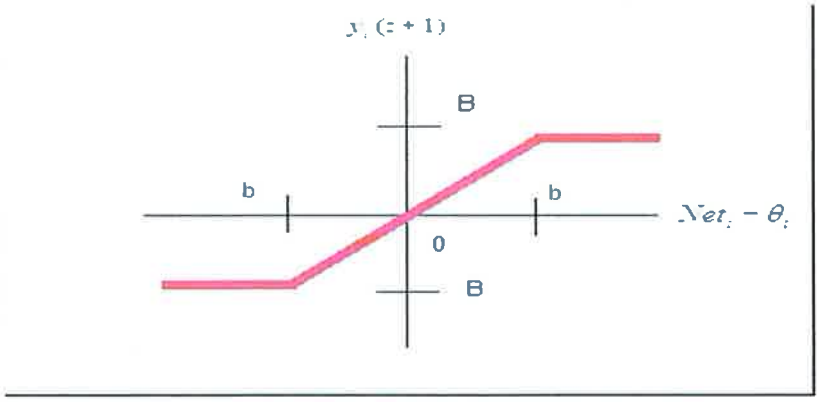

Fig. 7.17 *Función de Transferencia lineal mixta*

Función sigmoidal

Es una función continua, por tanto el espacio de los estados de activación es un intervalo del eje real.

$$y_i(t+1) = \frac{1}{(1 + e^{-(Net_i - \theta_i)})} \qquad \text{Ec. 7.10}$$

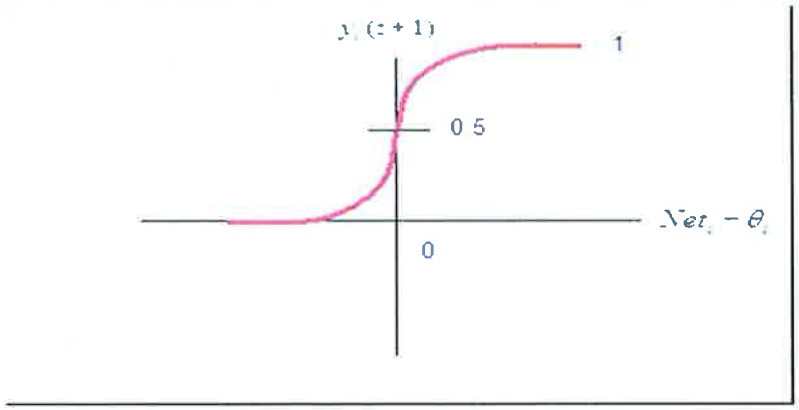

Fig. 7.18 *Función de Transferencia sigmoidal*

Para simplificar la expresión de la salida de una neurona *i,* es habitual considerar la existencia de una neurona ficticia, con valor de salida unidad, asociada a la entrada de cada neurona *i* mediante una conexión con peso de valor $-\theta_i$.De esta forma la expresión de salida quedará:

$$y_i(t+1) = f(\sum_{j=1}^{N} w_{ji} y_j(t) - \theta_i * 1) = f(\sum_{j=1}^{N} w_{ji} y_j(t)) = f(Net_i) \qquad \text{Ec. 7.11}$$

Redes Neuronales Artificiales Feedforward

En las últimas décadas las Redes Neuronales Artificiales (RNA) han recibido un interés particular, puesto que ofrece los medios para modelar de manera efectiva y eficiente problemas grandes y complejos. Las RNA son un método de resolver problemas, de forma individual o combinadas con otros métodos, para aquellas tareas de clasificación, identificación, diagnóstico, optimización o predicción en las que el balance datos/conocimiento se inclina hacia los datos y donde, adicionalmente, puede haber la necesidad de aprendizaje en tiempo de ejecución y de cierta tolerancia a fallos. A través de un algoritmo de aprendizaje supervisado o no supervisado, las RNA ajustan su arquitectura y parámetros de manera de poder minimizar alguna función de error que indique el grado de ajuste a los datos y la capacidad de generalización de las RNA.

Existe un gran número de arquitecturas neuronales, sin embargo, en este proyecto mostraremos solo la arquitectura más ampliamente utilizada y que será la que se utilizará en este trabajo, ya que es la indicada para la aplicación de aproximación de funciones: las redes feedforward.

El Perceptron Multicapa

El interés por la investigación en redes multicapa parte de los trabajos de Rosenblatt (1962) sobre Perceptrones y los de Widrow y sus alumnos sobre Madalines (1962). Los Madalines estaban constituidos por muchas unidades de entrada y muchos elementos Adalides en la primera capa, y con varios dispositivos lógicos (AND, OR,...) en la segunda capa. Sin embargo, como hemos visto, el Perceptrón simple es capaz de resolver problemas de clasificación e implementar funciones lógicas, como por ejemplo, la función OR, pero es incapaz de implementar la función lógica XOR. Sobre estas limitaciones, Minsky y Papert (1969) publicaron un libro titulado "Perceptrons" que supuso el abandono por parte de muchos científicos de la investigación en redes neuronales, pues no se encontraba un algoritmo de aprendizaje capaz de implementar funciones de este tipo. Las limitaciones de las redes de una sola capa hicieron que se plantease la necesidad de implementar redes en las que se aumentase el número de capas, es decir, introducir capas intermediarias o capas ocultas entre la capa de entrada y la capa de salida de manera que se pudiese implementar cualquier función con el grado de precisión deseado, es decir, que las redes multicapa fuesen aproximadores universales. Por ejemplo, con un Perceptrón con dos capaz se puede implementar la función lógica XOR. Al tener estas redes una topología más complicada, también se complicó la forma para encontrar los pesos correctos, ya que el proceso de aprendizaje es el que decide qué características de los patrones de entrada son representadas por la capa oculta de neuronas. En 1986 se abrió un nuevo panorama en el campo de las redes neuronales con el redescubrimiento por parte de Rumerlhard, Hinton y Williams del algoritmo de retropropagación. La idea básica de retropropagación fue descubierta por Werbos en su tesis doctoral (1974). Asimismo, algoritmos similares fueron desarrollados independientemente por Bryson y Ho (1969), Parker (1985) y LeCum

(1985). El algoritmo de retropropagación del error es un método eficiente para el entrenamiento de un Perceptron Multicapa. Se puede decir que puso fin al pesimismo que sobre el campo de las redes neuronales se había puesto en 1969 con la aparición del libro citado de Minsky y Papert.

Arquitectura

El Perceptron multicapa es una red de alimentación hacia adelante (feedforward) compuesta por una capa de unidades de entrada (sensores), otra capa de unidades de salida y un número determinado de capas intermedias de unidades de proceso, también llamadas capas ocultas porque no tienen conexiones con el exterior. Cada sensor de entrada está conectado con las unidades de la segunda capa, y cada unidad de proceso de la segunda capa está conectada con las unidades de la primera capa y con las unidades de la tercera capa, así sucesivamente. Las unidades de salida están conectadas solamente con las unidades de la última capa oculta, como se muestra en la figura 4.

Con esta red se pretender establecer una correspondencia entre un conjunto de entrada y un conjunto de salidas deseadas, de manera que

$$(x_1, x_2, \dots, x_N) \in R^N \rightarrow (y_1, y_2, \dots, y_M) \in R^M \qquad \text{Ec. 7. 12}$$

Para ello se dispone de un conjunto de con p patrones de entrenamiento, de manera que sabemos perfectamente que al patrón de entrada

$$(x_1^k, x_2^k, \dots, x_N^k)$$

Que le corresponda la salida

$$(y_1^k, y_2^k, \dots, y_M^k), k = 1, 2, \dots, p$$

Es decir, conocemos dicha correspondencia para p patrones. Así, nuestro conjunto de entrenamiento será:

$$\{(x_1^k, x_2^k, \dots, x_N^k) \rightarrow (y_1^k, y_2^k, \dots, y_M^k) : k = 1, 2, \dots, p\} \qquad \text{Ec. 7.13}$$

Para implementar dicha relación, la primera capa (sensores de entrada) tendrá tantos sensores como componentes tenga el patrón de entrada, es decir, N; la capa de salida tendrá tantas unidades de proceso como componentes tengan las salidas deseadas, es decir, M, y el número de capas ocultas y su tamaño dependerán de la dificultad de la correspondencia a implementar.

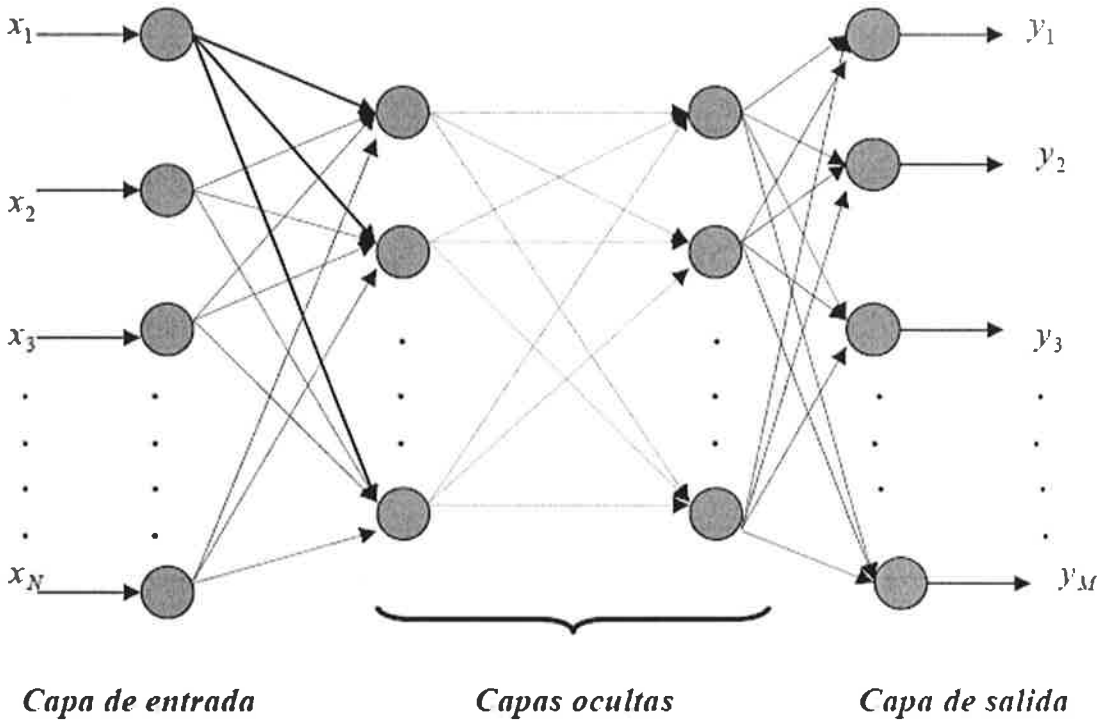

Fig. 7.19 *Topología de una Red Perceptron Multicapa*

Dinámica de la computación

Como las entradas a las unidades de proceso de una capa son las salidas de las unidades de proceso de la capa precedente, el Perceptron multicapa con sólo una capa oculta implementa la siguiente función:

Ec. 7.14

donde w_{ij} es el peso sináptico de la conexión entre la unidad de salida i y la unidad de proceso j de la capa oculta; g_1 es la función de transferencia de las unidades de salida, que puede ser una función logística, una función tangente hiperbólica o la función identidad; t_{jr} es el peso sináptico que conecta la unidad de proceso j de la capa oculta con el sensor de entrada r y g_2 es la función de transferencia de las unidades de la capa oculta, que puede ser también una función logística, una función tangente hiperbólica o

la función identidad. Una vez que hemos establecido la topología de la red, y su dinámica de la computación, la determinación de los pesos sinápticos nos llevará al diseño completo de la red. Para ello vamos a seguir un proceso de entrenamiento, mediante el cual vamos introduciendo cada uno de los patrones y evaluando el error que se comente entre las salidas obtenidas por la red y las salidas deseadas; entonces se irán modificando los pesos sinápticos según el error cometido, como vamos a ver a continuación.

Estructura y aprendizaje de la red *backpropagation*

En una red Backpropagation existe una capa de entrada con *n* neuronas y una capa de salida con *m* neuronas y al menos una capa oculta de neuronas internas. Cada neurona de una capa (excepto las de entrada) recibe entradas de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto las de salida). No hay conexiones hacia atrás *feedback* ni laterales entre las neuronas de la misma capa.

La aplicación del algoritmo tiene dos fases, una hacia delante y otra hacia atrás. Durante la primera fase el patrón de entrada es presentado a la red y propagado a través de las capas hasta llegar a la capa de salida. Obtenidos los valores de salida de la red, se inicia la segunda fase, comparándose éstos valores con la salida esperada para obtener el error. Se ajustan los pesos de la última capa proporcionalmente al error. Se pasa a la capa anterior con una retropropagación del error, ajustando los pesos y continuando con este proceso hasta llegar a la primer capa. De esta manera se han modificado los pesos de las conexiones de la red para cada patrón de aprendizaje del problema, del que conocíamos su valor de entrada y la salida deseada que debería generar la red ante dicho patrón.

El algoritmo *Backpropagation* requiere que la función de transferencia sea continua, y por lo tanto, diferenciable. Generalmente, la función utilizada será del tipo sigmoidal.

Pasos para aplicar el algoritmo de entrenamiento

Paso 1: Inicializar los pesos de la red con valores pequeños aleatorios.

Paso 2: Presentar un patrón de entrada y especificar la salida deseada que debe generar la red.

Paso 3: Calcular la salida actual de la red. Para ello presentamos las entradas a la red y vamos calculando la salida que presenta cada capa hasta llegar a la capa de salida, ésta será la salida de la red. Los pasos son los siguientes:

Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada. Para una neurona *j* oculta:

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h} \qquad \text{Ec. 7.15}$$

en donde el índice *h* se refiere a magnitudes de la capa oculta; el subíndice *p*, al *p-ésimo* vector de entrenamiento, y *j* a la *j-ésima neurona oculta. El término* θ puede ser opcional, pues actúa como una entrada más.

Se calculan las salidas de las neuronas ocultas:

$$y_{pj} = f_{j}^{h}(net_{pj}^{h}) \qquad \text{Ec. 7.16}$$

Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida:

$$net_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} y_{pj} + \theta_{k}^{0} \qquad \text{Ec. 7.17}$$

$$y_{pk} = f_{k}^{0}(net_{pk}^{o}) \qquad \text{Ec. 7.18}$$

Paso 4: Calcular los términos de error para todas las neuronas.

Si la neurona *k* es una *neurona de la capa de salida*, el valor de la *delta* es:

$$\delta_{pk}^{0} = (d_{pk} - y_{pk}) f_{k}^{o'}(net_{pk}^{o}) \qquad \text{Ec. 7.19}$$

La función *f* debe ser derivable:

Si la *neurona j no es de salida*, entonces la derivada parcial del error no puede ser evaluada directamente, por tanto se obtiene el desarrollo a partir de valores que son conocidos y otros que pueden ser evaluados.

La expresión obtenida en este caso es:

$$\delta_{pj}^{h} = f_{j}^{h'}(net_{pj}^{h}) \sum_{k} \delta_{pk}^{0} w_{kj}^{0} \qquad \text{Ec. 7.20}$$

Donde observamos que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término *propagación hacia atrás (Backpropagation).*

Paso 5: Actualización de los pesos: para ello utilizamos un algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la siguiente forma:

Para los pesos de las neuronas de la capa de salida:

$$w_{kj}^{0}(t+1) = w_{kj}^{o}(t) + \Delta w_{kj}^{o}(t+1) \qquad \text{Ec. 7.21}$$

$$\Delta w_{kj}^{o}(t+1) = \alpha \delta_{pk}^{o} y_{pj} \qquad \text{Ec. 7.22}$$

Para los pesos de las neuronas de la capa oculta:

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \Delta w_{ji}^{h}(t+1) \qquad \text{Ec. 7.23}$$

$$\Delta w_{ji}^{h}(t+1) = \alpha \delta_{pj}^{h} x_{pi} \qquad \text{Ec. 7.24}$$

En ambos casos, para acelerar el proceso de aprendizaje se puede añadir un término *momento.*

Paso 6: El proceso se repite hasta que el término de error:

$$E_p = \frac{1}{2} \sum_{k=1}^{M} \delta_{pk}^{2} \qquad \text{Ec. 7.25}$$

Resulta aceptablemente pequeño para cada uno de los patrones aprendidos.

Consideraciones sobre el algoritmo de aprendizaje

El algoritmo encuentra un valor mínimo de error (local o global) mediante una aplicación de pasos (gradiente) descendentes. Cada punto de la superficie de la función corresponde a un conjunto de valores de los pesos de la red. Con el gradiente descendente, siempre que se realiza un cambio en todos los pesos de la red, se asegura el descenso por la superficie del error hasta encontrar el valle más cercano, lo que puede hacer que el proceso de aprendizaje se detenga en un mínimo local de error.

Uno de los problemas del algoritmo es que en busca de minimizar la función de error, puede caer en un mínimo local o en algún punto estacionario, con lo cual no se llega a encontrar el mínimo global de la función de error. Sin embargo, no tiene porqué alcanzarse el mínimo global en todas las aplicaciones, sino que puede ser suficiente con un error mínimo preestablecido.

Control de Convergencia

En las técnicas de gradiente decreciente es conveniente avanzar por la superficie del error con incrementos de pesos pequeños. Esto se debe a que tenemos una información

local de la superficie y no se sabe lo lejos o cerca que se está del punto mínimo. Con incrementos grandes se corre el riesgo de pasar por encima del punto mínimo sin conseguir estacionarse en él. Con incrementos pequeños, aunque se tarda más en llegar, se evita que ocurra esto.

El incremento del paso adecuado influye en la velocidad de convergencia del algoritmo. La velocidad se controla con la tasa de aprendizaje α. Normalmente α, debe ser un número pequeño (del orden de 0,05 a 0,25), para asegurar que la red llegue a asentarse en una solución.

Lo habitual es aumentar el valor de α a medida que disminuye el error de la red durante la fase de aprendizaje. Así aceleramos la convergencia aunque sin llegar nunca a valores de α demasiado grandes, que hicieran que la red oscilase alejándose del mínimo.

En la práctica, si una red deja de aprender antes de llegar a una solución aceptable, se realiza un cambio en el número de neuronas ocultas o en los parámetros de aprendizaje, o simplemente, se vuelve a empezar con un conjunto distinto de pesos originales y se suele resolver el problema.

Dimensionamiento de la red. Número de neuronas ocultas

No se pueden dar reglas concretas para determinar el número de neuronas o número de capas de una red para resolver un problema concreto.

Respecto al número de capas de la red, en general tres capas son suficientes (entrada - oculta-salida). Sin embargo, hay veces que un problema es más fácil de resolver con más de una capa oculta. El tamaño de las capas, tanto de entrada como de salida, suelen venir determinado por la naturaleza de la aplicación. En cambio, decidir cuántas neuronas debe tener una capa oculta no suele ser tan evidente.

El número de neuronas ocultas interviene en la eficiencia de aprendizaje y de generalización de la red. No hay ninguna regla que indique el número óptimo, en cada problema se debe ensayar.

VIII. DESARROLLO DEL PROYECTO

ACTIVIDADES DESARROLLADAS

1. **Investigar sobre funcionamiento de los motores de combustión interna diesel y el Estado del Arte en la materia**. En esta etapa se pretende complementar el conocimiento que se tiene acerca del funcionamiento de los motores diesel, además de conocer el estado del Arte acerca del control de estos dispositivos, que servirá de apoyo para la realización de este experimento.

2. **Estudio de las RNA Feedforward y el algoritmo Backpropagation**. Se realiza una revisión de este tema, incluyendo todas las redes existentes y las aplicaciones más comunes de las RNA, para posteriormente enfocarse en las redes neuronales Feedforward y el algoritmo de entrenamiento backpropagation.

3. **Conocer Matlab y el toolbox para RNA**. Ya que se utilizará este software como herramienta para la construcción y simulación de la RNA, en esta etapa se busca aprender a utilizar este importante software matemático, además de las herramientas que este maneja para la creación y utilización de las RNA.

4. **Obtener datos de la combustión del motor para entrenar la RNA**. Para la creación y entrenamiento de una red Feedforward mediante Backpropagation, se requerirá de un conjunto de datos representativos de entradas y sus salidas objetivo, los cuales, para este experimento, serán tomados de la base de datos de Matlab.

5. **Crear la RNA, configurarla y entrenarla con los datos de combustión del motor**. Se crea la red, se configuran parámetros y se entrena la red por backpropagation, con el uso del toolbox de Matlab.

6. **Validar y probar la RNA creada**. Se valida que el error entre la salida de la red y los datos objetivo sea el requerido, y se prueba la red entrenada con nuevos datos para comprobar su ajuste a datos diferentes.

PROCEDIMIENTO

Para diseñar una RNA se deben seguir los siguientes pasos:

1. Colectar los datos
2. Crear la red
3. Configurar la red
4. Inicializar los pesos y los bias
5. Entrenar la red
6. Validar la red
7. Utilizar la red

Colectar los datos

Los datos para el entrenamiento de la red deben ser representativos y deben estar en el rango de valores para el cual se usará la red, ya que las RNA no sirven para extrapolar valores, pero son buenas para predecir valores dentro del rango en el cual fueron entrenadas. Hay un preprocesamiento de los datos, con el fin de que estos puedan ser utilizados en la red, además de agilizar la optimización.

En dicho preprocesamiento se llevan a cabo los siguientes pasos

- Normalización: ya que el rango de valores del conjunto de datos puede ser muy amplio, con esta operación se escalan los datos de manera que todos estén dentro de un rango de valores, generalmente [-1 1]. Esto facilita el cálculo para el entrenamiento de la red.
- Remover filas constantes: se eliminan filas que tienen valores constantes y que por tanto no otorgan información para el entrenamiento de la red.

Los datos se dividen aleatoriamente en tres bloques:

- Entrenamiento
- Validación
- Pruebas

Al momento de entrenarse la red, también se presentan aleatoriamente, con el objeto de encontrar más rápidamente el valor mínimo del error para la función de aproximación.

Para este proyecto, se utilizaron las funciones de Matlab:

- Mapminmax: para normalización de los datos de entrada y datos objetivo.

- Removeconstantrows: elimina filas constantes de los datos que no añaden información a la red.
- Dividerand: divide los datos aleatoriamente, 70 % para entrenamiento, 15% para validación y 15% para test, en la siguiente forma:

  - Training Ratio trainRatio: 0.7
  - Validation Ratio valRatio: 0.15
  - Test Ratio testRatio: 0.15

Crear, configurar, entrenar y validar la red

Para la creación de la red en Matlab se pueden seguir varios métodos, uno de los cuales es seguir la herramienta nftool (que es de la forma en que se creara la red para este caso) se abre un tutorial se cargan los datos, se configura la red, se entrena, y cuando termina el entrenamiento se puede verificar el ajuste de la red con observar las gráficas de desempeño de la red. Al final Matlab da la opción de crear código fuente de la red que ya se creó, con el objetivo de modificar la configuración en caso de buscar un mejor ajuste o de buscar agilizar el entrenamiento.

Se teclea nftool en Matlab

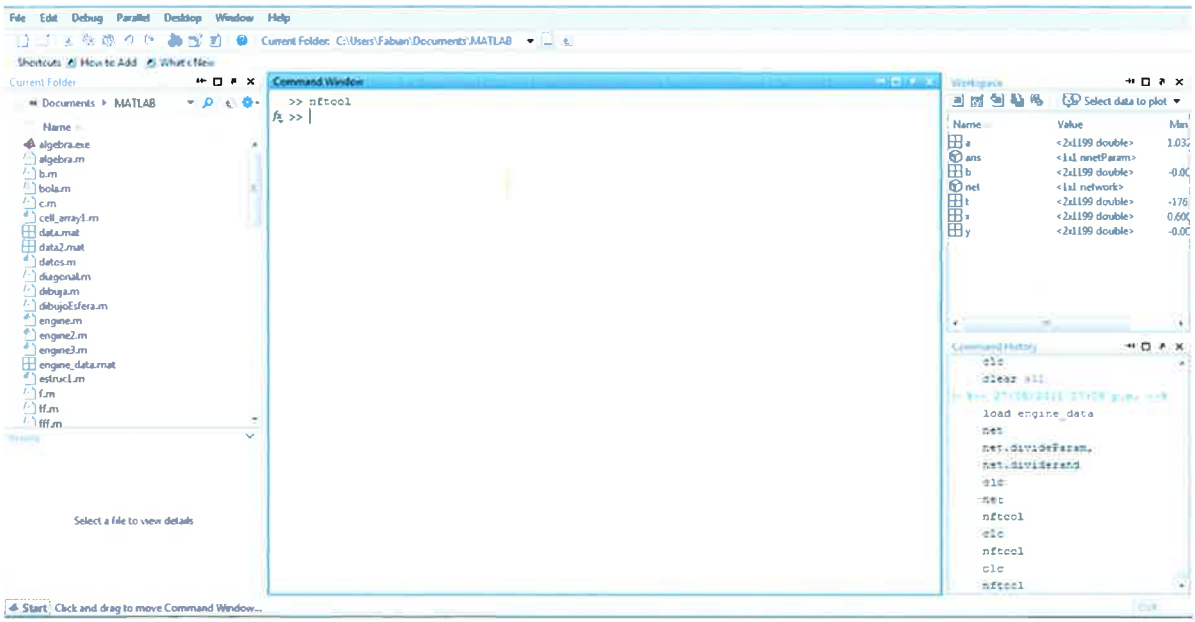

Fig. 8.1 *Pantalla de Matlab.*

Se abre el cuadro de dialogo de fitting tool neural network. Se da clic en siguiente

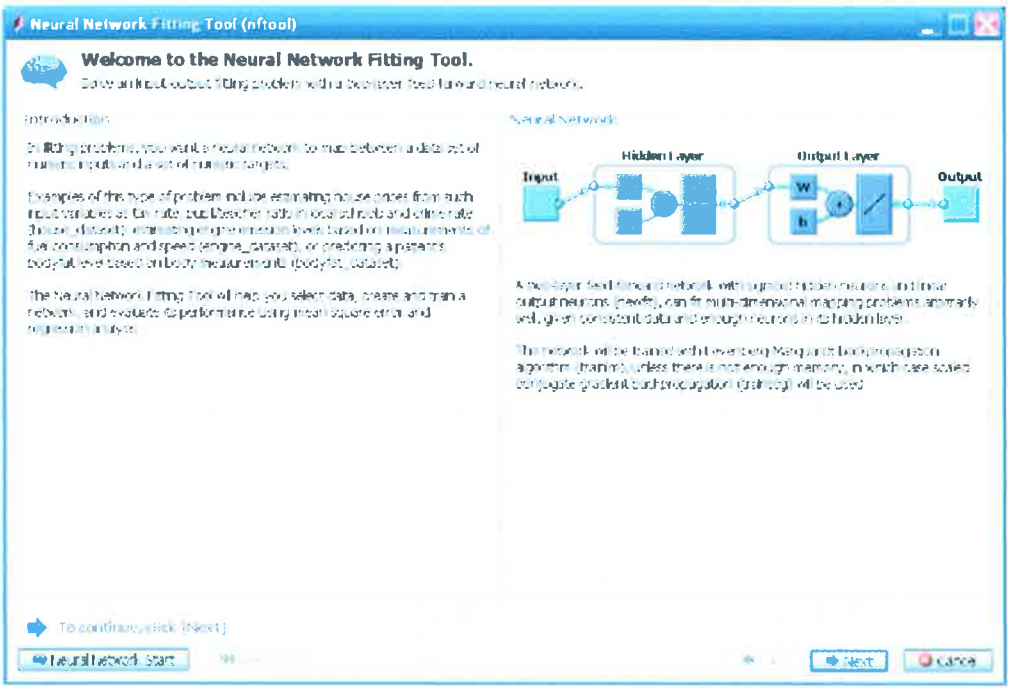

Fig. 8.2 *Herramienta de Matlab para aproximación de funciones con RNA.*

Se cargan los datos de entrenamiento para el problema del motor diesel. El conjunto de entrenamiento consiste en dos matrices de datos, una de ellas son los datos de entrada, una matriz de 2X1199 valores. El renglón uno de esta matriz es el consumo de combustible en g/kwh, y el segundo renglón es la velocidad del motor en RPM.

La matriz de datos de salida (2X1199) se utiliza como datos meta o datos deseados que se compararán con los datos arrojados por la red, para así realizar el entrenamiento. El primer renglón de esta matriz de datos es el torque en Nm y el segundo son las emisiones de NO_X en mg/m^3.

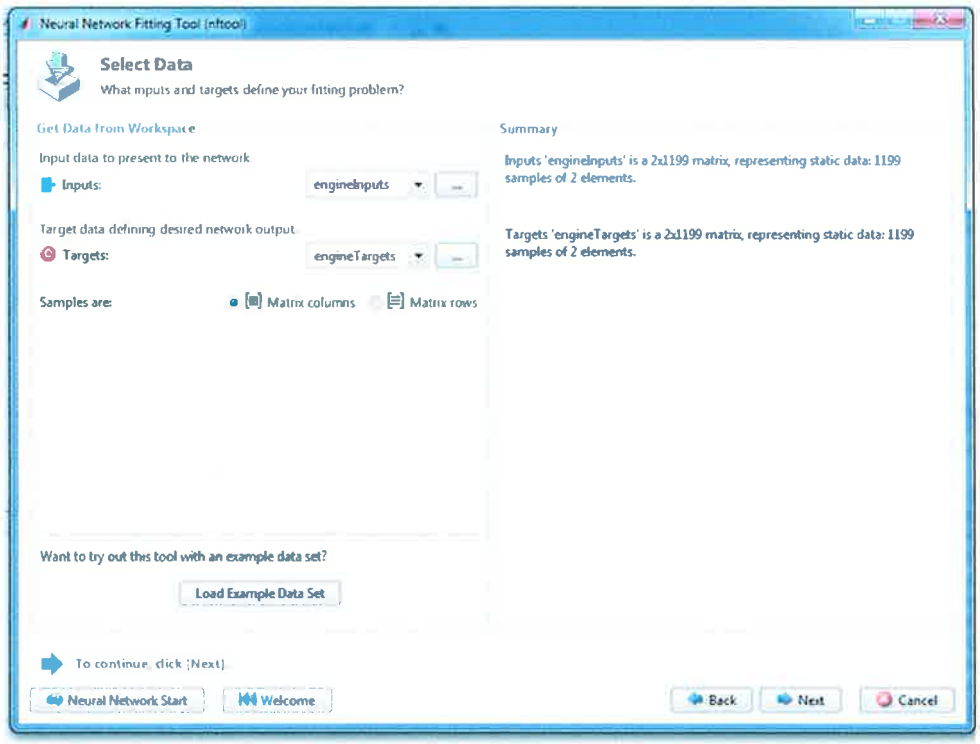

Fig. 8.3 *Cargar datos del motor.*

Se configuran los porcentajes para entrenamiento, validación y prueba de la red.

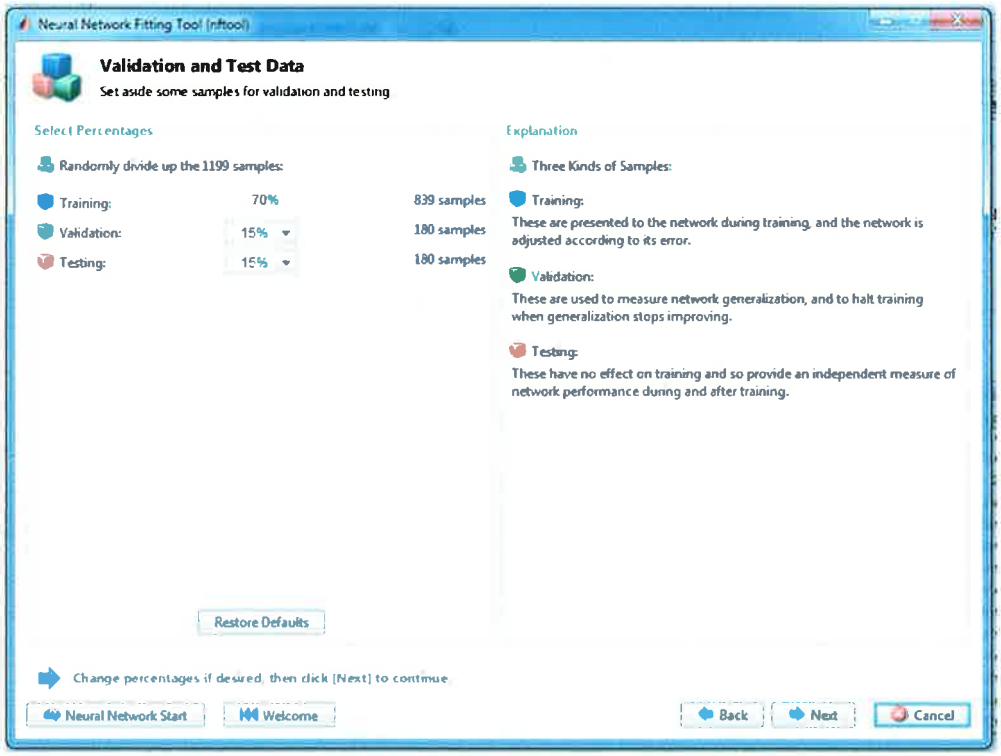

Fig. 8.4 *Configuración de datos.*

Después de realizar varias pruebas con distinta cantidad de neuronas en la capa oculta, se concluyó que el mejor ajuste se llevaba a cabo con 15 neuronas en su capa oculta. La función de transferencia de la capa oculta es la función sigmoidea. Por default Matlab utiliza el algoritmo entrenamiento Backrpopagation **Levenberg-Marquardt**, (que es el más indicado para este tipo de aplicación de redes neuronales) y la función de aproximación del cuadrado medio del error (**LMS**).

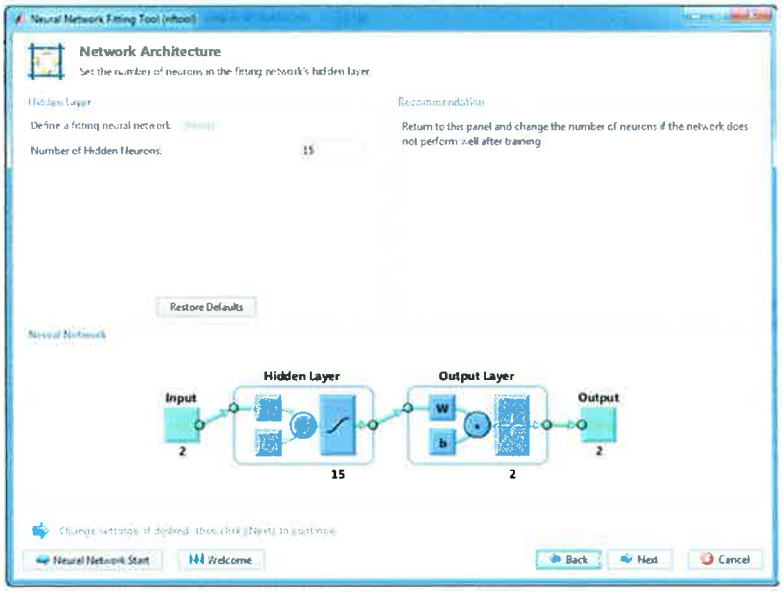

Fig. 8.5 *Configuración de 15 neuronas en capa oculta.*

Como la red recibirá dos variables de entrada, se configura automáticamente con dos neuronas en la capa de entrada. Al requerirse dos variables de salida, se configura la red con dos neuronas en la capa de salida.

Antes de entrenar la red, se deben de inicializar los pesos y los bias para que esta se pueda entrenar. Matlab incluye por default la función *initlay* Los valores que se asignan por default a los pesos y los bias son valores aleatorios entre en el rango [-1 1].

Se entrena la red.

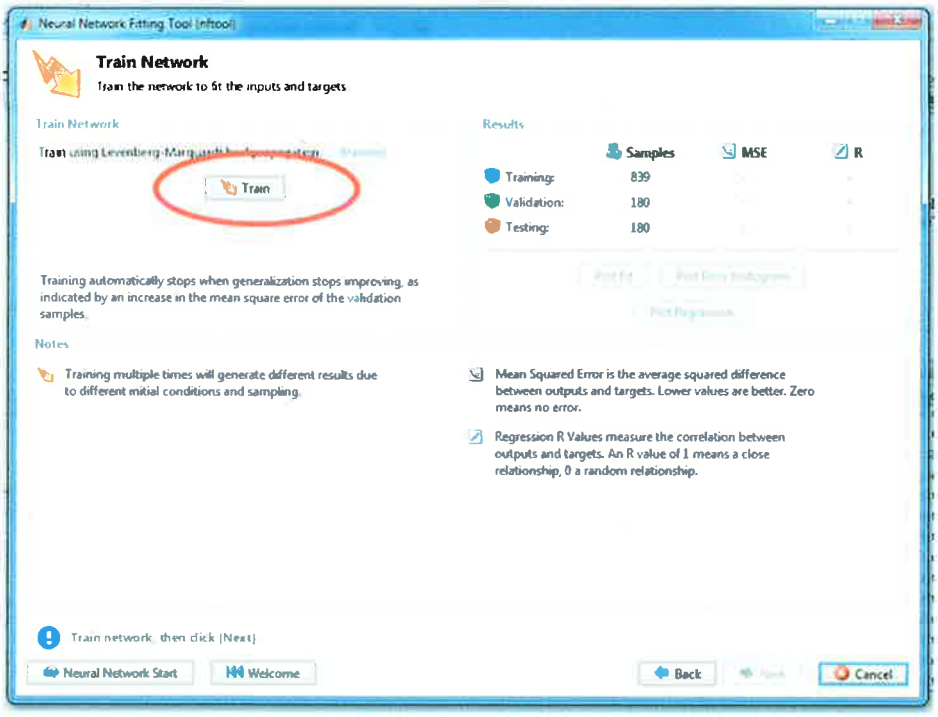

Fig. 8.6 *Entrenar la red.*

Se muestran los resultados del entrenamiento (*nntraintool*)

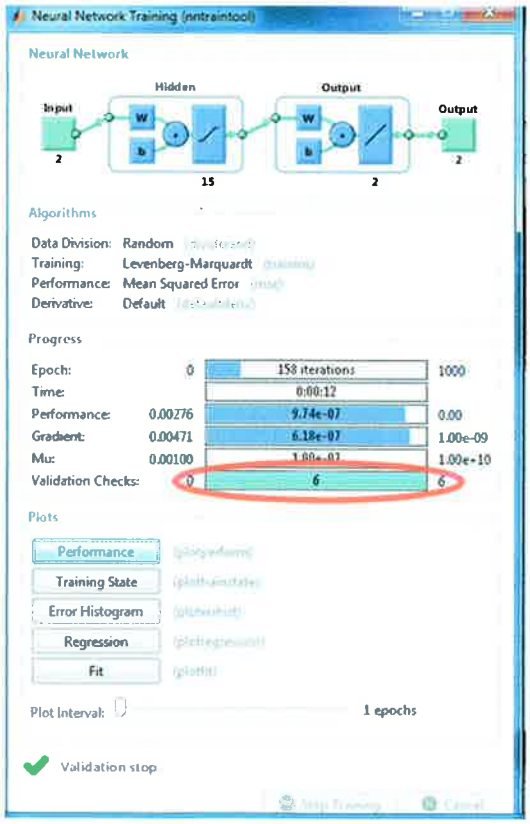

Fig. 8.7 *Resultados de entrenamiento.*

Dentro de la configuración, se definieron los llamados *criterios de paro* para el entrenamiento de la red. Matlab por default define dos criterios de paro y son

- El error cuadrado medio que se definió en 1 X 10^{-9}.
- El número de épocas o iteraciones en las que el error o no descienda o comience a ascender. Matlab define un número de 6 épocas o iteraciones de entrenamiento para ejecutar el paro del mismo.

Cualquiera de los dos criterios de paro (el que suceda primero) harán que el entrenamiento cese. En este caso el criterio de para que se cumplió fue el de 6 épocas o iteraciones sin descenso del LMS.

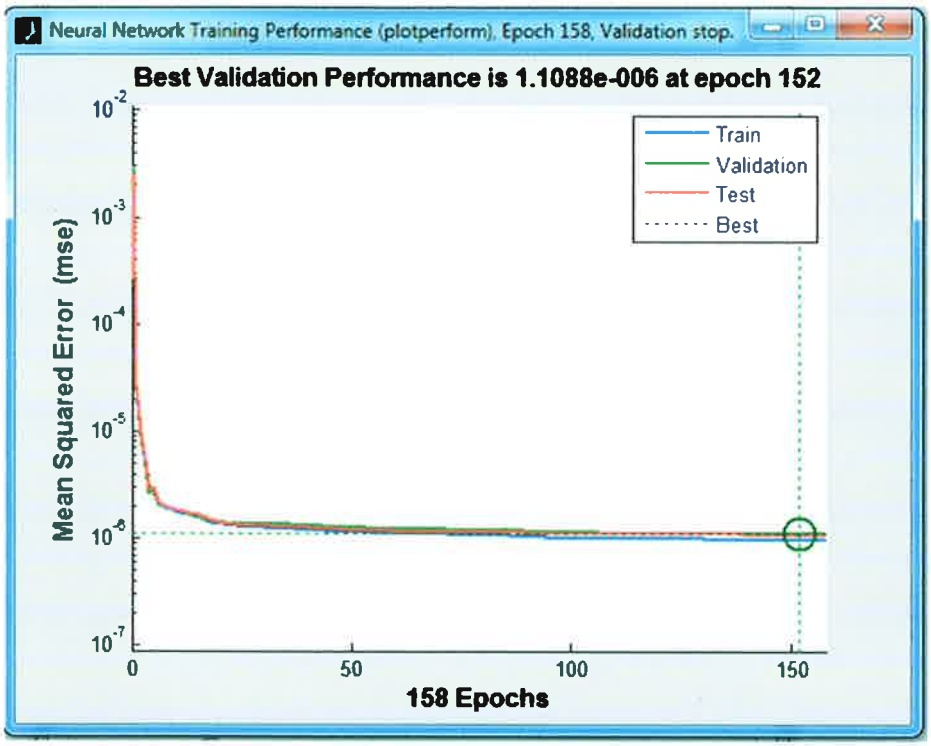

Fig. 8.8 *Gráfica de la función de desempeño.*

Después del entrenamiento, Matlab por default realiza la validación de la red, y también ejecuta una prueba con los datos. Como se vio anteriormente, se toma un 15 % de los datos para validación y otro 15 % para testear la red.

Utilizar la red

Para utilizar la red en Matlab se utiliza la función *sim,* para simular la red con cualquier conjunto de valores. El parámetro *net* es el nombre de la red, *x* es el conjunto de datos de entrada que se alimentaran a la red. La variable *y1* es la respuesta de la red a los datos presentados.

```
y1 = sim(net,x)
```

La variable *y1* almacena la respuesta de la red, que si esta fue entrenada correctamente, entonces los datos predichos serán acordes a la realidad.

IX. RESULTADOS

Para comprobar la exactitud de la red para predecir las salidas se requiere utilizar datos de entrada con las correspondientes salidas deseadas. Aquí se muestran los resultados del entrenamiento de la red por Matlab, poniendo énfasis en la parte de prueba que se realizó con el 15% de los datos.

La grafica siguiente presenta el desempeño de la función de aproximación durante todo el proceso de entrenamiento, validación y prueba de la red.

La etapa de prueba se realiza después de que la red ya ha sido entrenada, es decir que los valores de los pesos y los bias de las conexiones de la red para los que esta entregará la salida más cercana a la salida deseada ya están determinados, solamente queda verificar que así es. Es decir, cuando la red ya fue 'educada' para responder eso significa que sus pesos y bias ya están determinados, y puede usarse para predecir salidas con solo ingresarle valores de entrada.

El error que arrojaron los datos destinados para prueba indican una buena predicción de la red de las salidas, como se puede ver en la gráfica de abajo, la línea color rojo es la gráfica de la función de aproximación para este conjunto de datos.

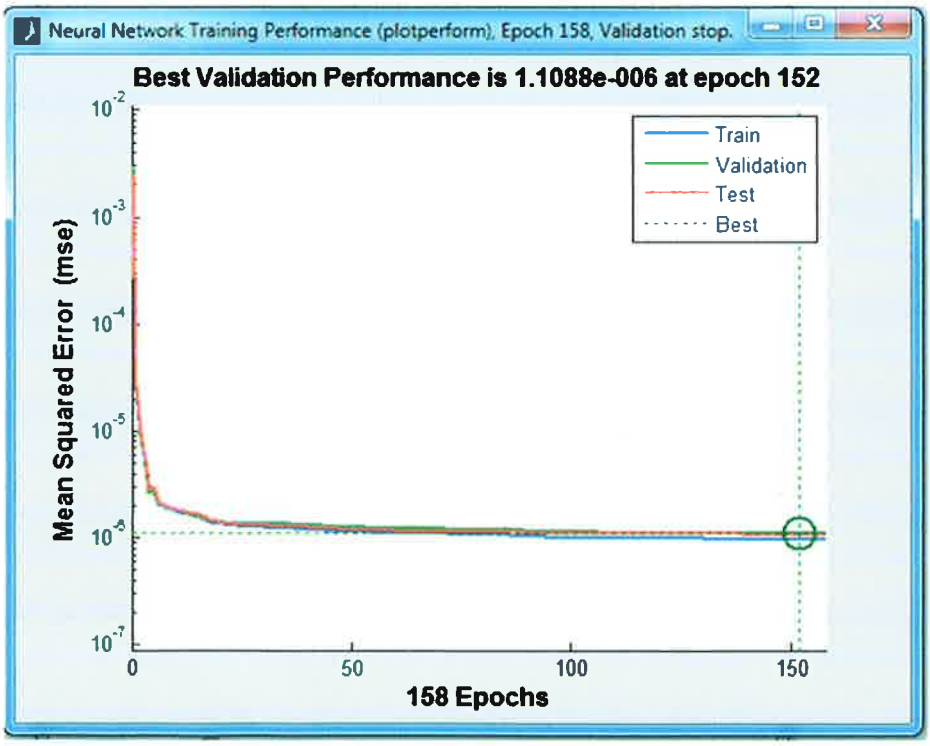

Fig. 9.1 *Gráfica de la función de desempeño.*

Por otro lado se tiene la gráfica de regresión de los datos de prueba, así como de entrenamiento y validación.

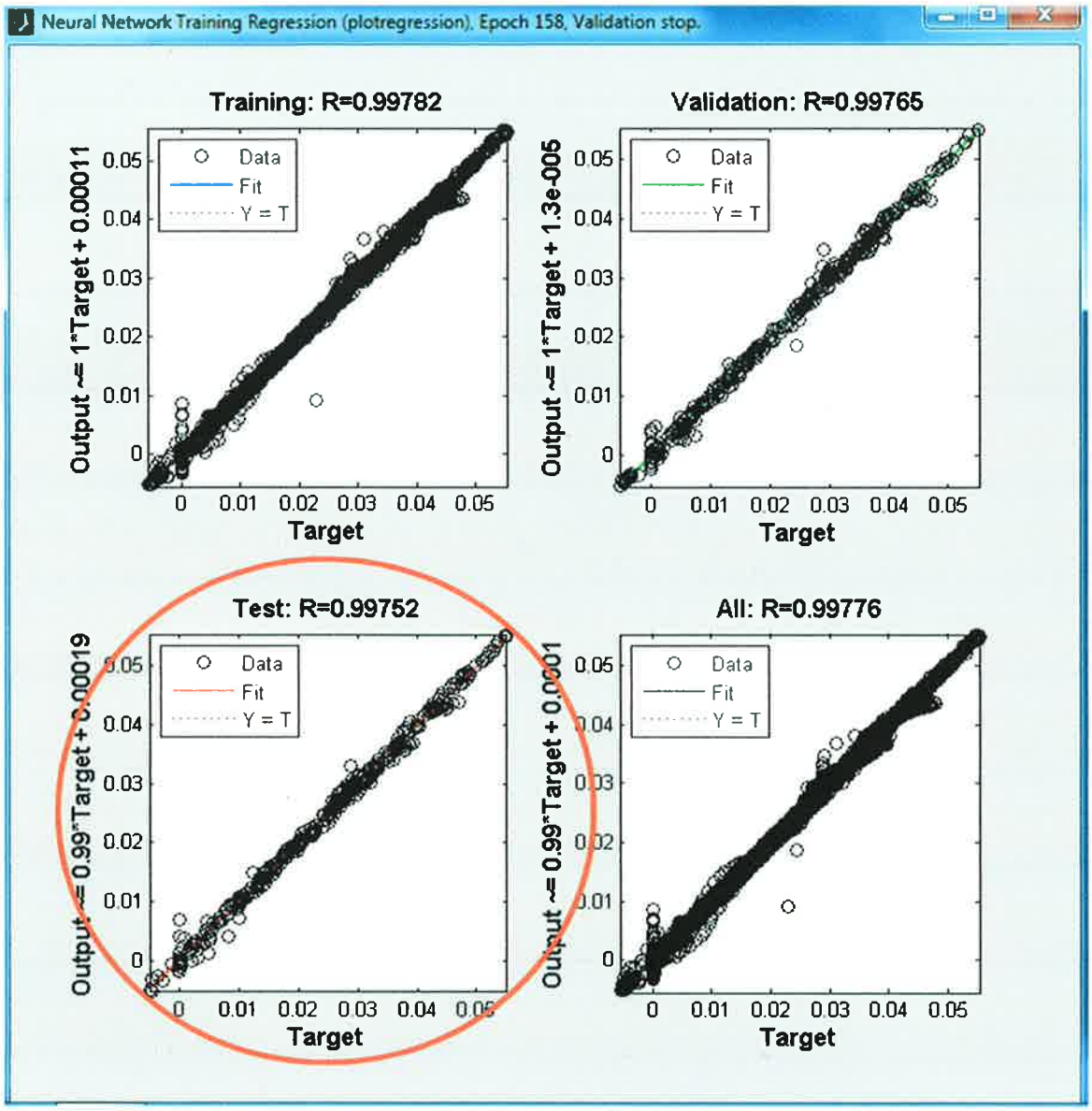

Fig. 9.2 *Grafica de regresión simple con datos reales vs calculados por la red*

Esta grafica de regresión grafica los valores de salida deseados contra los valores reales. La línea roja debajo de la nube de puntos es la predicción perfecta de la red, es decir, los datos deseados son iguales a los datos de salida de la red, mientras que la nube de puntos son los datos de la salida deseada t_i contra la salida virtual otorgada por la red a_i, los cuales como se puede ver están altamente relacionados, con un coeficiente de correlación R=0.99752, casi de la unidad, lo cual sería un ajuste casi perfecto.

El valor de la función del error (el cuadrado medio del error) de la red después del entrenamiento fue de $9.74X10^{-7}$, que es considerablemente bajo.

Por último se muestra el comportamiento de las variables como se hizo en la justificación,

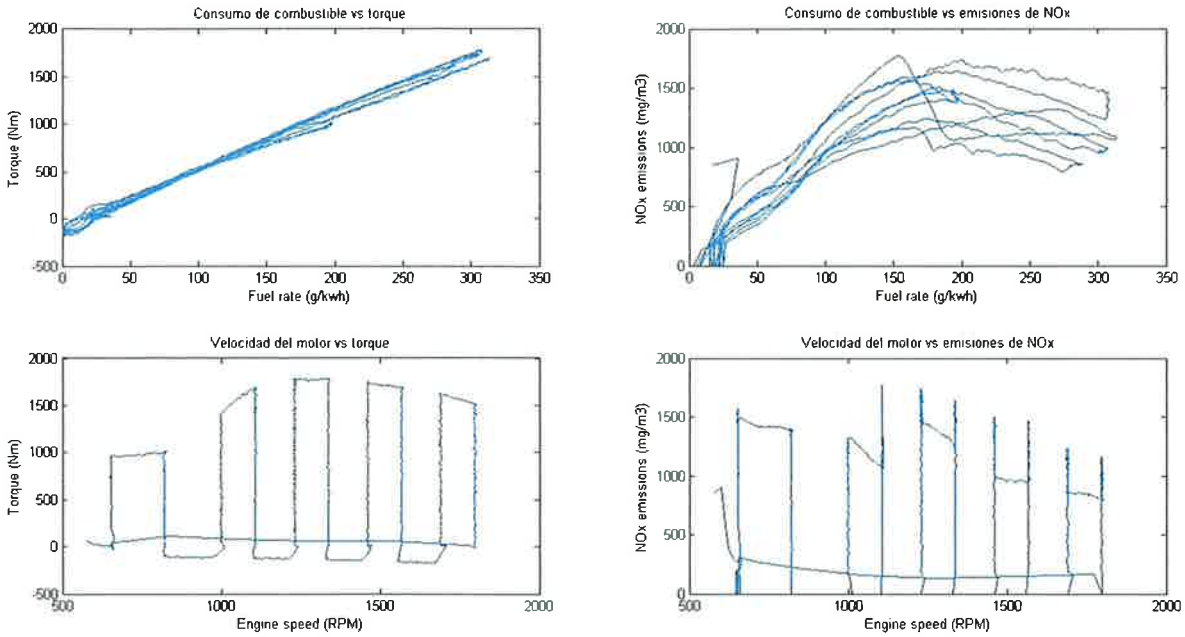

Fig. 9.3 *Comportamiento de las variables a caracterizar del motor diesel*

En la siguiente grafica se muestra el comportamiento de dichas variables. Pero ahora se grafican los valores predichos por la red neuronal, igualmente cada variable de entrada con cada variable de salida.

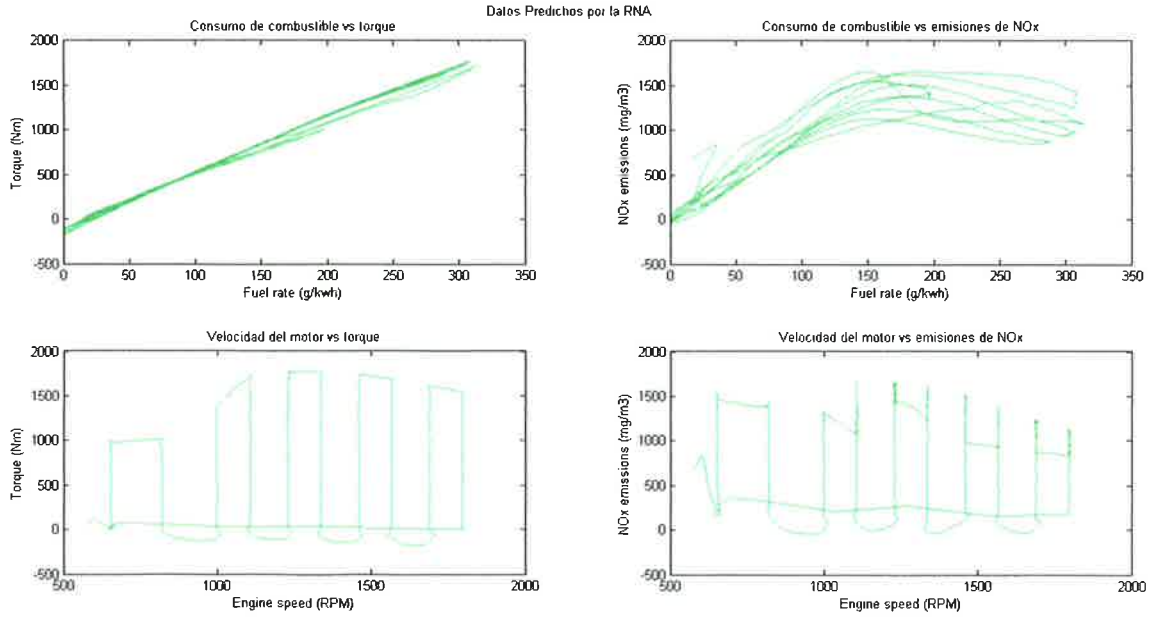

Fig. 9.4 *Valores de las variables calculados por la RNA*

Como se puede observar, el comportamiento de los datos reales y predichos es casi el mismo, solo hay algunas variaciones en la predicción de las emisiones, sin embargo se trata de valores de emisiones cero o cercanos s cero, por tanto no de mucha relevancia.

En la gráfica siguiente se incluyen ambos datos, tanto los reales (color azul) como los calculados por la red neuronal creada (color verde).

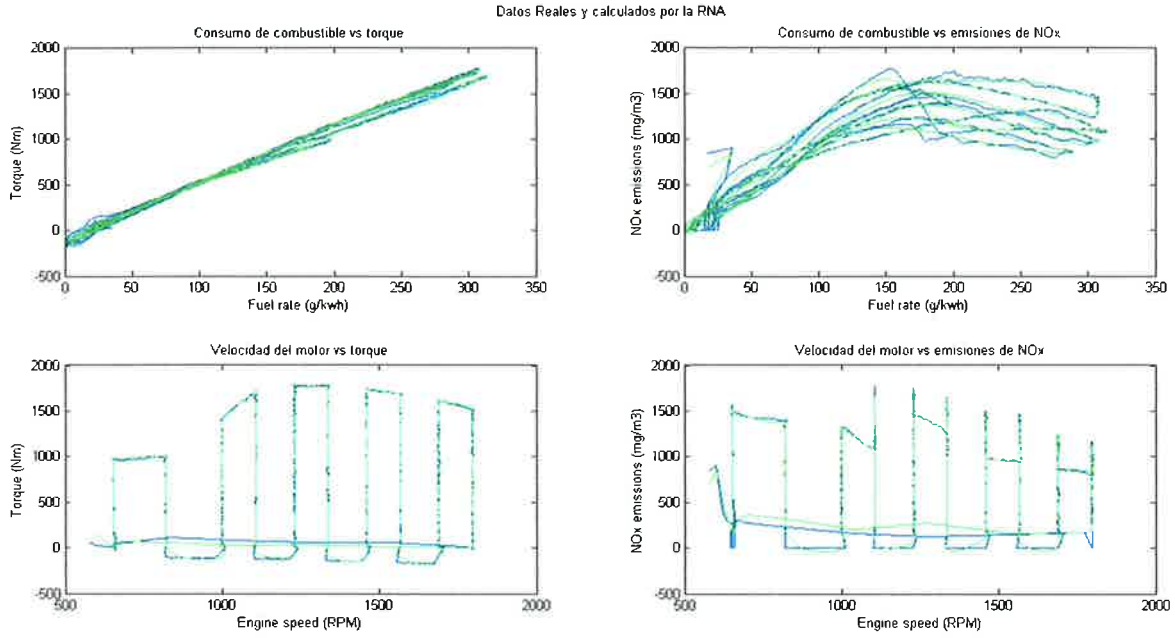

Fig. 9.5 *Datos reales vs calculados por la red.*

Como se puede observar tambien aquí en estas gráficas (aun mas claramente) el calculo de la red es muy cercano a los datos reales, mostrando que la red fue habil para acercarse al comportamiento real de los datos, inclusive cuando este es altamente no lineal, como se puede ver en las graficas de la velocidad del motor contra el torque y contra las emisiones de NO_X.

X. CONCLUSIONES

Las conclusiones que se desprenden de este trabajo son las siguientes:

- las RNA pueden ser utilizadas para el control de un MCI diesel.
- Se consiguió tener un dominio suficiente del tema de RNA, del motor diesel, así como de las herramientas aquí utilizadas para la creación de la red.
- se logró crear la RNA para el cálculo de las variables del motor, y se entrenó hasta que pudo calcular las salidas con el error adecuado.
- Las regulaciones ambientales son de 3.5 g/kwh para México y 2.6 g/kwh para E.E.U.U., y el nivel de error permite el control dentro de estos parámetros.
- al aplicar el control por RNA en los MCI mejorará substancialmente su eficiencia y el impacto ambiental.

XI. BIBLIOGRAFÍA Y REFERENCIAS

[1] Neural Network Design. Martin T. Hagan, Howard B. Demuth. Mark Beale. Edit. Thomson.

[2] Neural Network Toolbox™ 7 User's Guide.

[3] Manual Básico de Matlab 2006. Mª Cristina Casado Fernández. Servicios Informáticos U.C.M. Apoyo a Investigación y Docencia.

[4] Redes neuronales y sistemas borrosos 3ra. Edición. Bonifacio Martin del Brío. Alfredo Sanz Molina. Edit. Alfaomega.

[5] Automotive emissions control. Michael Stubblefield & John H. Haynes. Haynes Techbook.

[6] El poder del cerebro. Susan Greenfield. Edit. Crítica. 2007.

[7] *Control de la velocidad de motores y maquinas*. Marco Antonio Pérez Cisneros, Mark Readman y Peter Wellstad. División de electrónica y computación. CUCEI Universidad de Guadalajara, México. Control System Principles.

[8] International Journal of Digital Content Technology and its Applications. Volume 4, Number 6, September 2010. *Multi-factor predication of diesel engine by using artificial neural networks*. Wenyong Xiao. Shanghai Jiao Tong University, School of Mechanical Engineering. sjtu_xwy@163.com. doi: 10.4156/jdcta.vol4.issue6.19.

[9] Research Journal of Applied Sciences, Engineering and Technology 1(3): 125-131, 2009.Submitted Date: June 25, 2009 Accepted Date: July 18, 2009 Published Date: October 20, 2009. Evaluation of Artificial Neural Network Performance in Predicting Diesel Engine NO_x Emissions. O. Obodeh and C. I. Ajuwa. Mechanical Engineering Department, Ambrose Alli University, Ekpoma, Edo State, Nigeria.

[10] Neural Network-Based Intelligent Engine Control for reducing NO_X and PM from CIDI engines. C. Atkinson. West Virginia University.

[11] http://es.wikipedia.org/wiki/Unidad_de_control_de_motor

[12] http://www.monografias.com/trabajos12/redneuro/redneuro.shtml

ANEXOS

A. Plan de actividades

CRONOGRAMA DE ACTIVIDADES PARA EL TIEMPO DE PRACTICAS O ENTRENAMIENTO DE LA ESPECIALIDAD

ACTIVIDADES	TIEMPO DE ENTRENAMIENTO											
	3 - 9 de jul	10 - 16 de jul	17 - 23 de jul	24 - 30 de jul	31 jul - 6 ago	7 - 13 de ago	14 - 20 de ago	21 - 27 de ago	28 ago - 3 sep	4 - 10 de sep
Investigar sobre funcionamiento de los motores de combustión interna diesel y el Estado del Arte en la materia										
Estudio de RNA Feedforward y el algoritmo Backpropagation										
Conocer Matlab y el toolbox para RNA										
Obtener datos de la combustión del motor para entrenar la RNA										
Crear y entrenar la RNA con los datos de combustión del motor										
Validar y probar la RNA creada										
Realización, revisión y corrección del reporte										
Presentación del proyecto										

B. Código fuente de Matlab para la creación de la RNA

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by NFTOOL
% Created Mon Aug 22 10:36:43 CDT 2011
%
% This script assumes these variables are defined:
%
%   engineInputs - input data.
%   engineTargets - target data.

inputs = engineInputs;
targets = engineTargets;

% Create a Fitting Network
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% 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 = 'sample';  % Divide up every sample
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','ploterrhist', ...
  'plotregression', 'plotfit'};


% Train the Network
net.trainParam.min_grad = 1.0000e-7
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)
```

```
% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotfit(net,inputs,targets)
%figure, plotregression(targets,outputs)
%figure, ploterrhist(errors)
```

C. Objeto de la red

```
net =

    Neural Network

              name: 'Function Fitting Neural Network'
        efficiency: .cacheDelayedInputs, .flattenTime,
                    .memoryReduction
          userdata: (your custom info)

    dimensions:

         numInputs: 1
         numLayers: 2
        numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
  numFeedbackDelays: 0
  numWeightElements: 52
        sampleTime: 1

    connections:

       biasConnect: [1; 1]
      inputConnect: [1; 0]
      layerConnect: [0 0; 1 0]
     outputConnect: [0 1]

    subobjects:

            inputs: {1x1 cell array of 1 input}
            layers: {2x1 cell array of 2 layers}
           outputs: {1x2 cell array of 1 output}
```

```
            biases: {2x1 cell array of 2 biases}
      inputWeights: {2x1 cell array of 1 weight}
      layerWeights: {2x2 cell array of 1 weight}

    functions:

          adaptFcn: 'adaptwb'
        adaptParam: (none)
          derivFcn: 'defaultderiv'
         divideFcn: 'dividerand'
       divideParam: .trainRatio, .valRatio, .testRatio
        divideMode: 'sample'
           initFcn: 'initlay'
         performFcn: 'mse'
       performParam: .regularization, .normalization,
.squaredWeighting
           plotFcns: {'plotperform', plottrainstate, ploterrhist,
                     plotregression, plotfit}
         plotParams: {1x5 cell array of 2 params}
           trainFcn: 'trainlm'
         trainParam: .showWindow, .showCommandLine, .show, .epochs,
                     .time, .goal, .min_grad, .max_fail, .mu, .mu_dec,
                     .mu_inc, .mu_max

    weight and bias values:

                IW: {2x1 cell} containing 1 input weight matrix
                LW: {2x2 cell} containing 1 layer weight matrix
                 b: {2x1 cell} containing 2 bias vectors

    methods:

             adapt: Learn while in continuous use
         configure: Configure inputs & outputs
            gensim: Generate Simulink model
              init: Initialize weights & biases
           perform: Calculate performance
               sim: Evaluate network outputs given inputs
             train: Train network with examples
              view: View diagram
       unconfigure: Unconfigure inputs & outputs

    evaluate:       outputs = net(inputs)
```