UNIVERSIDAD NACIONAL DE LOJA

ÁREA DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

INGENIERÍA ELECTROMECÁNICA

TÍTULO:

"METODOLOGÍA PARA EL DESARROLLO DE MODELOS BASADOS EN REDES NEURO-FUZZY PARA LA PREDICCIÓN DE LA POTENCIA REAL GENERADA A CORTO PLAZO POR PANELES SOLARES UBICADOS EN LA CIUDAD DE LOJA"

TESIS DE GRADO PREVIO A OPTAR POR EL TÍTULO DE INGENIERO ELECTROMECÁNICO

Autor: José Luis Paredes Luzón

Director: Ing. Juan Pablo Cabrera Samaniego

LOJA – ECUADOR

2012

CERTIFICACIÓN

Ing. Juan Pablo Cabrera Samaniego

DIRECTOR DE TESIS

CERTIFICA:

Haber dirigido, asesorado, revisado y corregido el presente trabajo de tesis de grado, en su proceso de investigación cuyo tema versa en **"Metodología para el desarrollo de modelos basados en redes neuro-fuzzy para la predicción de la potencia real generada a corto plazo por paneles solares ubicados en la ciudad de Loja",** previa a la obtención del título de Ingeniero Electromecánico, realizado por el señor egresado: **José Luis Paredes Luzón**, la misma que cumple con la reglamentación y políticas de investigación, por lo que autorizo su presentación y posterior sustentación y defensa.

Loja, 25 de septiembre de 2012

Ing. Juan Pablo Cabrera Samaniego
DIRECTOR DE TESIS

DECLARACIÓN DE AUTORÍA

La investigación, análisis y conclusiones del presente trabajo de tesis, le corresponde exclusivamente a su autor y el patrimonio intelectual a la Universidad Nacional de Loja. Autorizo al Área de la Energía, las Industrias y los Recursos Naturales No Renovables y por ende a la carrera de Ingeniería Electromecánica, hacer uso del presente documento en lo conveniente.

_____

José Luis Paredes Luzón

PENSAMIENTO

"La voz interior me dice que siga combatiendo contra el mundo entero, aunque me encuentre solo. Me dice que no tema a este mundo sino que avance llevando en mí nada más que el temor a Dios."

Mahatma Gandhi

"El honor más grande aún no se ha otorgado, la carrera más dura aún no ha comenzado. No basta con soñarlo, hay que echarle manos a la obra, luchar hasta conseguir el éxito de nuestro presente. Nunca es tarde… Enhorabuena."

G.M.Quirós

"El hombre encuentra a Dios detrás de cada puerta que la ciencia logra abrir."

Albert Einstein

DEDICATORIA

Dedico este proyecto a Dios por haberme dado a mis padres José y Eliza, quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento. Depositando su entera confianza en cada reto que se me presentaba sin dudar ni un solo momento en mi inteligencia y mi capacidad. Es por Él y ellos que soy lo que soy ahora.

También a mis hermanos Alba y Ángel por su paciencia amor y compañía en todos estos años.

Que Dios los bendiga siempre y saben que cuentan conmigo en todo…

AGRADECIMIENTO

Quisiera expresar en estas líneas mi más sincero agradecimiento a todas aquellas personas que de una u otra forma han contribuido a la realización de esta tesis.

En primer lugar quisiera dirigirme a mis padres para expresarles mi más profundo sentir de agradecimiento por todos aquellos sacrificios que mi formación ha supuesto para ellos. Asimismo quisiera agradecerles a ellos y a mis hermanos todo el cariño y apoyo moral que siempre me han prestado.

A mi director de tesis, Ing. Juan Pablo Cabrera, quisiera agradecerle la ayuda incondicional que siempre me ha prestado y el haber sabido guiar mi trabajo en los momentos más delicados. Su gran cultura científica, su disponibilidad y su simplicidad son algunos de los tantos elementos que han favorecido el desarrollo de esta tesis.

A la Universidad Nacional de Loja por donde he pasado una etapa muy importante de mi vida y he recibido una educación ejemplar.

A los docentes de la carrera de Ingeniería en Electromecánica por sus conocimientos científico, moral y humanista brindados en el proceso de enseñanza.

A mis amigos y amigas por haber compartido esta etapa de mi vida, a todos quienes colaboraron con su granito de arena para la realización de este trabajo y me han acompañado de una u otra forma durante este proceso.

TABLA DE CONTENIDOS

a.-TÍTULO 10

b. RESUMEN 11

c.- INTRODUCCIÓN 13

d. REVISIÓN DE LITERATURA 14

d.1 CAPÍTULO I: Los Paneles Solares Fotovoltaicos 14

d.1.1 Introducción 14

d.1.2 Funcionamiento de un panel solar fotovoltaico 14

d.1.3 Bases del funcionamiento de las células fotovoltaicas 14

d.1.4 Tipos de paneles en función de los materiales 15

d.1.4.1 Silicio Puro Monocristalino 15

d.1.4.2 Silicio puro policristalino 16

d.1.5 Tipos de paneles en función de la forma 18

d.1.5.1 Paneles con sistemas de concentración 18

d.1.5.2 Paneles de formato “teja o baldosa” 19

d.1.5.3 Paneles bifaciales 19

d.1.6 Elementos asociados a los paneles solares fotovoltaicos 19

d.1.6.1 Regulador 20

d.1.6.2 Batería 21

d.1.6.3 Inversores 21

d.2 CAPÍTULO II: Introducción a las Redes Neuronales Artificiales 22

d.2.1 Estructura Básica 22

d.2.1.1 Un conjunto de sinapsis o conexiones provenientes de otras neuronas o del exterior 22

d.2.1.2 Un sumador (Σ) 22
d.2.1.3 Una función de activación (F) 22
d.2.1.3.1 Funciones de activación más usadas en las redes modernas. 25
d.2.2 Clasificación de las redes neuronales artificiales. 26
d.2.2.1 Entrenamiento de las Redes Neuronales Artificiales. 27
d.2.2.1.1 Entrenamiento supervisado 28
d.2.2.1.2 Entrenamiento no supervisado. 29
d.2.3 El Perceptrón y los primeros días de las redes neuronales artificiales 29
d.3.1 Introducción 31
d.3.2 Conceptos y reglas básicas 35
d.4 CAPÍTULO: IV Sistemas Neuro-Difusos 39
d.4.1 Tipos de Neuronas Difusas 40
d.4.1.1 Neuronas exactas y neuronas difusas. 40
d.4.1.2 Neuronas difusas y redes neuronales difusas 41
d.4.1.2.1 Neurona difusa MAX (OR) 42
d.4.1.2.2 Neurona difusa MIN (AND). 42
d.4.2 Neuronas difusas generalizadas y redes. 43
d.4.2.1 Entradas sinápticas 43
d.4.2.2 Entradas dendríticas 43
d.4.2.3 Valores agregados 43
d.4.2.4 Salida de la neurona 43
d.4.3 Redes Neuronales Difusas Multicapa 44
d.4.3.1 Red Multicapa de Neuronas Difusas. 45
d.4.4 Métodos Difusos en Redes Neuronales 46

d.4.4.1 Fsom 46

d.4.4.2 Nefclass..... 46

d.4.4.3 ANFIS..... 46

d.4.4.4 FuzzyTech..... 47

d.4.4.5 Aprendizaje en un Sistema Neuro-difuso 47

d.4.5 Red ANFIS 49

d.5 Conclusiones de la Revisión Literaria..... 50

e.- MATERIALES Y MÉTODOS..... 51

e.1 Materiales..... 51

e.1.1 Panel solar fotovoltaico Exmork 100P. 51

e.1.2 Equipo de medición inalámbrica de las variables solares que cuenta la UEE (Unidad de Eficiencia Energética) del AEIRNNR. 52

e.1.3 Recursos informáticos 53

e.1.4 Herramientas e implementos extras..... 53

e.2 Métodos 54

e.2.1 Técnicas de modelado y recolección de datos 54

e.2.2.1 Procedimiento De Recolección De Datos..... 54

e.2.2.2 Técnicas De Modelado 54

f.- RESULTADOS..... 56

f.1 Procesamiento de datos..... 56

f.1.1 Importar datos a Matlab 59

f.1.1.2 Creación de archivos de extensión .m..... 61

f.2. Creación de Modelos. 63

f.2.1 Modelos que utilizan redes neuronales Feed-Forward Backpropagation 64

f.2.1.1 Estructura de los datos 64

f.2.1.1.2 Metodología para la creación de una RNA tipo BP (Back propagation) mediante la GUI nntool 65

f.2.1.1.2.1 Entrenamiento de una RNA 70

f.2.1.1.2.2 Exportando datos a la ventana de trabajo de Matlab 71

f.2.2 Resultados de los modelos generados con redes neuronales. 73

f.2.2.1 Validación Del Modelo 73

f.2.2.1.1 Modelos realizados con redes neuronales con su respectivo rendimiento: 74

f.2.3 Modelos utilizando redes Neuro-Fuzzy 93

f.2.3.1 Estructura de los datos 93

f.2.3.2 Metodología para la creación de redes neuro-fuzzy mediante la GUI anfisedit . 94

f.2.3.3 Procedimiento para el entrenamiento y validación de los datos en editor ANFIS94

f.2.3.3.1 Modelos con redes neuro-fuzzy 103

f.2.3.3.2 Resumen de evaluación y simulación del modelo Neuro-Fuzzy tipo ANFIS 113

f.3 Procedimiento para realizar aplicaciones con Matlab Builder Ex 113

f.4 Procedimiento implementar aplicaciones de Matlab Builder Ex a usuarios finales117

f.4.1 Instalando la aplicación para poder utilizar el modelo en una hoja de Excel sin tener instalado Matlab. 117

g.- DISCUSIÓN 131

h.- CONCLUSIONES 136

j.- BIBLIOGRAFÍA 140

k. ANEXOS 143

a. TÍTULO

“Metodología para el desarrollo de modelos basados en redes neuro-fuzzy para la predicción de la potencia real generada a corto plazo por paneles solares ubicados en la ciudad de Loja”

b. RESUMEN

Este trabajo investigativo se refiere a una metodología para el desarrollo de modelos basados en redes neuronales y sistemas neuro-fuzzy para la predicción la potencia real generada a corto plazo por paneles solares ubicados en la ciudad de Loja, el cual tiene el propósito de desarrollar modelos temporales basados en redes neuro-fuzzy capaces de determinar en base a la radiación y temperatura ambiente las variables de voltaje, corriente y potencia de salida en condiciones reales de operación del panel solar.

El presente trabajo investigativo está distribuido entre varios literales que se detallan a continuación:
En el literal ***d*** se presenta una revisión de la literatura necesaria para comprender conceptos y reglas básicas utilizadas para este trabajo investigativo.

El literal ***e*** revela los materiales y métodos utilizados. El panel fotovoltaico utilizado marca EXMORK es de 100W, los valores de corriente y voltaje son monitoreados a la salida del panel; y también se adquieren las variables de temperatura del panel y radiación solar con sensores adecuados para este tipo de panel solar. El tiempo de muestreo fue cada minuto.

En el literal ***f*** se exhibe los resultados del presente trabajo investigativo. En esta sección se describe el procesamiento de los datos obtenidos en Matlab, se realiza las metodologías para generar los modelos con redes neuronales y redes neuro-fuzzy tipo ANFIS; los procedimientos para realizar aplicaciones con Matlab Builder Ex y el procedimiento para implementar las aplicaciones de Matlab Builder Ex para ejecutar los modelos en una hoja de cálculo de Excel además visualizar un informe de calibración de los modelos mediante un documento con formato html.

La discusión sobre los resultados del proyecto de tesis se presenta en el literal g.

Se formuló en el literal ***h*** e ***i***, las conclusiones correspondientes a la investigación las mismas que condujeron al planteamiento de las recomendaciones respectivamente, con el fin de mejorar la metodología para el desarrollo de modelos neuro-fuzzy realizada.
Palabras Clave: *Panel solar, ANFIS, RNA'S, nntool, modelos neurofuzzy, Matlab Builder Ex.*

SUMMARY

This research work concerns a methodology for the development of models based on neural networks and neurofuzzy systems to predict the actual power generated by solar panels short term located in the city of Loja, which aims to develop models based temporary neurofuzzy networks able to determine from radiation and temperature variables of voltage, current and power output in real operating conditions of the solar panel.

This research work is distributed between several literals which are detailed below:

In the literal d is a review of the literature needed to understand basic concepts and rules used for this research work.

The literal and reveals the materials and methods used. The photovoltaic panel EXMORK mark used is 100W, the current and voltage values are monitored at the output of the panel, and also acquires the variables temperature and solar panel with suitable sensors for this type of solar panel. The sampling time was every minute.

In the literal f is exhibited the results of this research work. This section describes the processing of data from Matlab, methodologies is performed to generate models with neural networks and ANFIS type neurofuzzy networks; procedures for Ex applications with Matlab Builder and how to deploy applications Matlab Builder Ex to run models in Excel spreadsheet also view a report of calibration models using a html document.

The discussion on the results of the thesis project is presented in subsection g.

Formulated in the literal i, the research findings for the same approach that led to recommendations respectively, in order to improve the methodology for the development of neurofuzzy models performed.

Keys words: Solar panel, ANFIS, ANN'S, nntool, neurofuzzy models, Matlab Builder Ex

c.- INTRODUCCIÓN

El presente trabajo investigativo presenta una "METODOLOGÍA PARA EL DESARROLLO DE MODELOS BASADOS EN REDES NEURO-FUZZY PARA LA PREDICCIÓN DE LA POTENCIA REAL GENERADA A CORTO PLAZO POR PANELES SOLARES UBICADOS EN LA CIUDAD DE LOJA", mediante la cual se realiza la elaboración de modelos con redes neuronales y sistemas neuro-fuzzy tipo ANFIS , también se crea una aplicación para introducir un complemento en una hoja de Excel que ejecuta o simula los modelos generados para usuarios que no tengan instalado el programa Matlab y se puede visualizar también el reporte o informe de calibración de los modelos mediante un documento con formato html.

En el desarrollo de este trabajo se muestra los procedimientos realizados mediante imágenes capturadas y descripción de las mismas al momento de utilizar el programa Matlab en las metodologías empleadas tanto al generar los modelos como para realizar la aplicación que genera el complemento en Excel y en los anexos también se presentan los scripts utilizados en este trabajo.

Con los datos de potencia obtenidos en los modelos se puede analizar en qué lugar es factible y/o rentable la instalación de paneles solares policristalinos EXMORK de 100 W para así contribuir al desarrollo e implementación de energías renovables mediante paneles solares fotovoltaicos. Lo cual permitiría disminuir los impactos ambientales que producen las energías no renovables, beneficiando de esta manera a la sociedad lojana particularmente.

Con todo lo expresado se aspira también a que los estudiantes de la carrera de Ingeniería Electromecánica se interesen en aplicar métodos modernos de estimación para la solución efectiva a problemas en que se presente el manejo de la información imprecisa e incierta que existe en los problemas relacionados con el mundo real y se utilice también como referente para estudios posteriores en este campo.

d. REVISIÓN DE LITERATURA

d.1 CAPÍTULO I: Los paneles solares fotovoltaicos

d.1.1 Introducción

Los paneles solares fotovoltaicos son un elemento clave en la conversión directa de la energía solar a eléctrica, los paneles fotovoltaicos experimentan en la actualidad una demanda sin precedentes. Los problemas derivados del cambio climático y la progresiva concienciación han provocado un cambio de mentalidad hacia este producto.

d.1.2 Funcionamiento de un panel solar fotovoltaico

El funcionamiento de los paneles se basa en el efecto fotovoltaico. Este efecto se produce cuando sobre materiales semiconductores convenientemente tratados incide la radiación solar produciéndose electricidad.

d.1.3 Bases del funcionamiento de las células fotovoltaicas

Cuando el conjunto queda expuesto a la radiación solar, los fotones contenidos en la luz transmiten su energía a los electrones de los materiales semiconductores que pueden entonces romper la barrera de potencial de la unión P-N y salir del semiconductor a través de un circuito exterior, produciéndose así corriente eléctrica.

El módulo más pequeño de material semiconductor con unión P-N y por lo tanto con capacidad de producir electricidad, es denominado célula fotovoltaica. Estas células fotovoltaicas se combinan de determinadas maneras para lograr la potencia y el voltaje deseados. Este conjunto de células sobre el soporte adecuado y con los recubrimientos que le protejan convenientemente de agentes atmosféricos es lo que se denomina panel fotovoltaico. Ver figura 1.

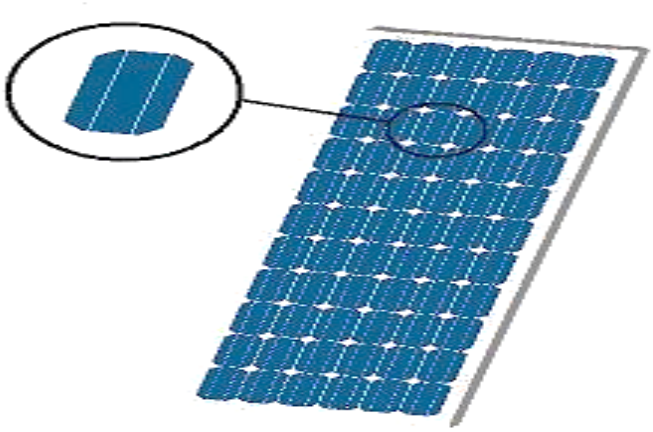


Figura 1. Célula fotovoltaica y panel fotovoltaico

d.1.4 Tipos de paneles en función de los materiales

Existen diferentes tipos de paneles solares en función de los materiales semiconductores y los métodos de fabricación que se empleen. Los tipos de paneles solares que se pueden encontrar en el mercado son:

d.1.4.1 Silicio Puro Monocristalino

Basados en secciones de una barra de silicio perfectamente cristalizado en una sola pieza. En laboratorio se han alcanzado rendimientos máximos del 24,7% para éste tipo de paneles siendo los comercializados alrededor del 16%.

Figura 2. Panel solar monocristalino

d.1.4.2 Silicio puro policristalino

Los materiales son semejantes a los del tipo anterior aunque en este caso el proceso de cristalización del silicio es diferente. Los paneles policristalinos se basan en secciones de una barra de silicio que se ha estructurado desordenadamente en forma de pequeños cristales. Son visualmente muy reconocibles por presentar su superficie un aspecto granulado. Se obtiene con ellos un rendimiento inferior que con los monocristalinos (en laboratorio del 19.8% y en los módulos comerciales del 14%) siendo su precio también más bajo.

Por las características físicas del silicio cristalizado, los paneles fabricados siguiendo esta tecnología presentan un grosor considerable. Mediante el empleo del silicio con otra estructura o de otros materiales semiconductores es posible conseguir paneles más finos y versátiles que permiten incluso en algún caso su adaptación a superficies irregulares.

Figura 3.Panel solar Policristalino

Existen también los llamados paneles Tándem que combinan dos tipos de materiales semiconductores distintos. Debido a que cada tipo de material aprovecha sólo una parte del espectro electromagnético de la radiación solar, mediante la combinación de dos o tres tipos de materiales es posible aprovechar una mayor parte del mismo. Con este tipo de paneles se ha llegado a lograr rendimientos del 35%. Teóricamente con uniones de 3 materiales podría llegarse hasta rendimientos del 50%.

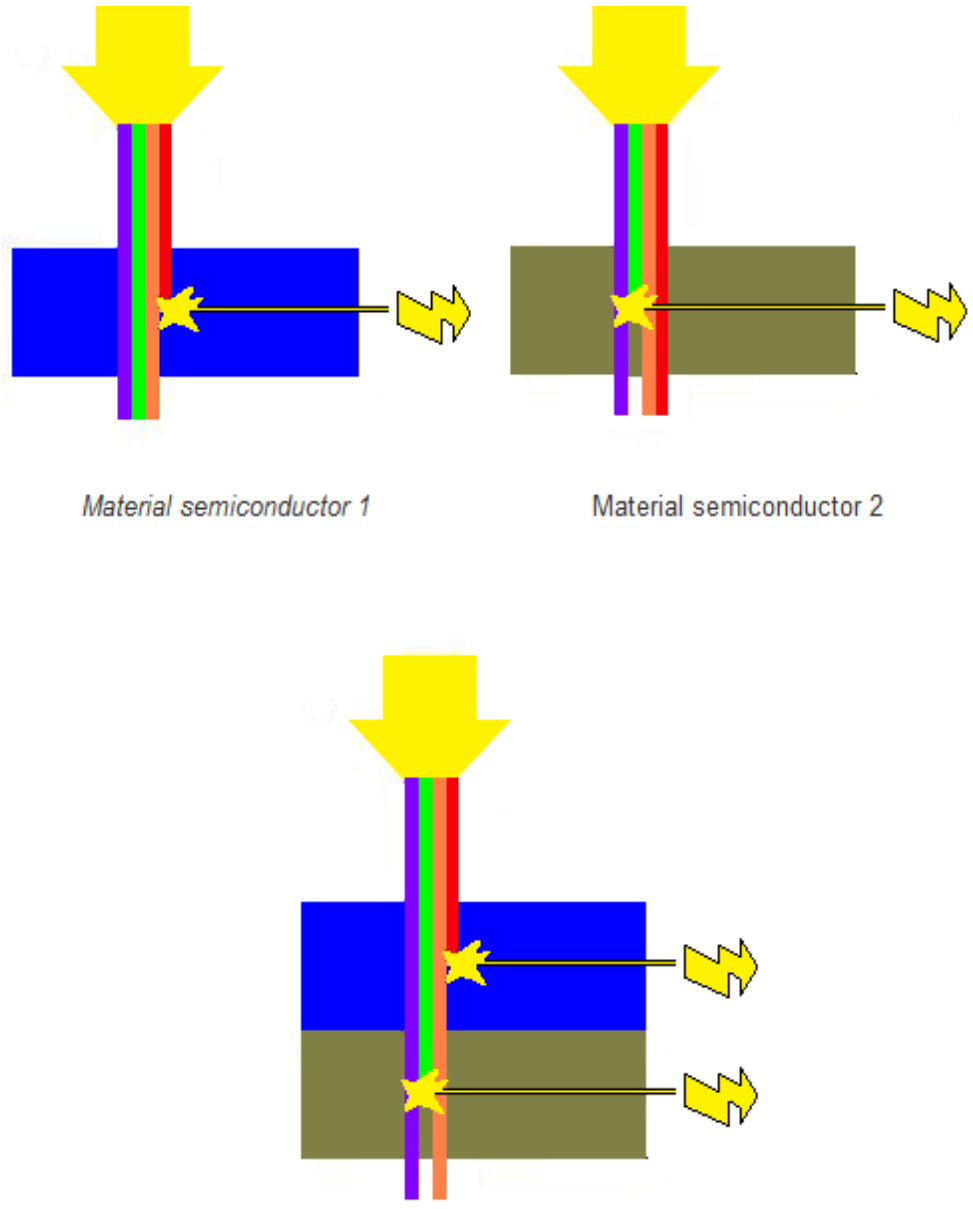


Figura 4. Célula Tandem (Material semiconductor 1 + 2)

La figura 4 muestra, (1) Célula con material semiconductor 1, solo aprovecha una parte del espectro electromagnético de que está compuesta la luz solar (2) La célula con el

material semiconductor 2 aprovecha otra parte del espectro electromagnético de la luz diferente al del material semiconductor 1 (3) en la célula Tandem se combinan ambos tipos de materiales, con lo que se aprovecha la parte del espectro electromagnético de ambos tipos de materiales son capaces de transformar en energía eléctrica. El rendimiento total será en teoría la suma de los rendimientos de ambos tipos de células por separado

La mayoría de los módulos comercializados actualmente están realizados de silicio monocristalino, policristalino y amorfo. El resto de materiales se emplean para aplicaciones más específicas y son más difíciles de encontrar en el mercado.

Mención especial merece una nueva tecnología que está llamada a revolucionar el mundo de la energía solar fotovoltaica. Se trata de un nuevo tipo de panel solar muy fino, muy barato de producir y que según dicen sus desarrolladores presenta el mayor nivel de eficiencia de todos los materiales. Este nuevo tipo de panel está basado en el Cobre Indio Galio Diselenido (CIGS) y se prevé que en un futuro no muy lejano, debido a su competitiva relación entre producción de energía/costo pueda llegar a sustituir a los combustibles fósiles en la producción de energía.

d.1.5 Tipos de paneles en función de la forma

También es posible clasificar los tipos de paneles en función de su forma. Empleándose cualquiera de los materiales antes comentados se fabrican paneles en distintos formatos para adaptarse a una aplicación en concreto o bien para lograr un mayor rendimiento. Algunos ejemplos de formas de paneles distintos del clásico plano son:

d.1.5.1 Paneles con sistemas de concentración

Un ejemplo de ellos es el modelo desarrollado por una marca española, el cual mediante una serie de superficies reflectantes concentra la luz sobre los paneles fotovoltaicos. Aunque el porcentaje de conversión no varíe, una misma superficie de panel producirá más electricidad ya que recibe una cantidad concentrada de fotones. Ver figura 5.

Actualmente se investiga en sistemas que concentran la radiación solar por medio de lentes. La concentración de la luz sobre los paneles solares es una de las vías que están desarrollando los fabricantes para lograr aumentar la efectividad de las células fotovoltaicas y bajar los costes.

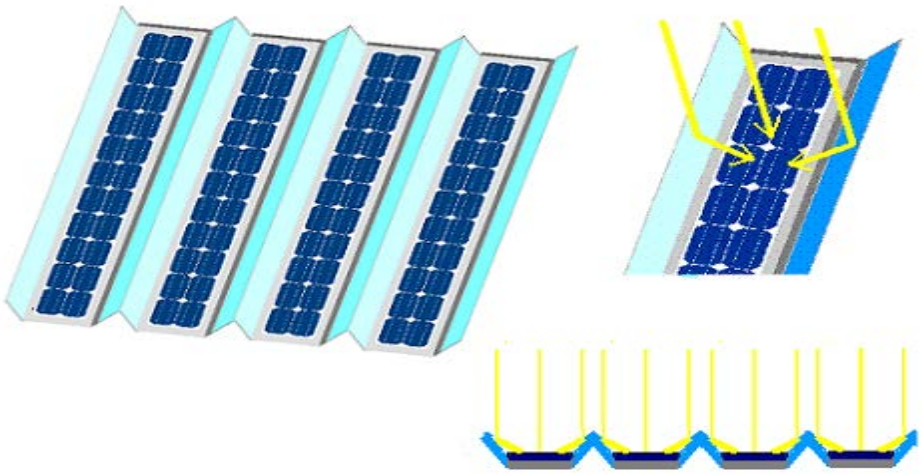


Figura 5. Paneles con sistemas de concentración

d.1.5.2 Paneles de formato “teja o baldosa”

Estos paneles son de pequeño tamaño y están pensados para combinarse en gran número para así cubrir las grandes superficies que ofrecen los tejados de las viviendas. Aptos para cubrir grandes demandas energéticas en los que se necesita una elevada superficie de captación.

d.1.5.3 Paneles bifaciales

Basados en un tipo de panel capaz de transformar en electricidad la radiación solar que le recibe por cualquiera de sus dos caras. Para aprovechar convenientemente esta cualidad se coloca sobre dos superficies blancas que reflejan la luz solar hacia el reverso del panel.

d.1.6 Elementos asociados a los paneles solares fotovoltaicos

El panel solar es el elemento encargado de captar la energía del sol y de transformarla en energía eléctrica que se pueda ser usada. Asociado los paneles existen otros componentes que se utilizan en las instalaciones como elementos de seguridad o que amplían las posibilidades del uso de la instalación. Los componentes esenciales de una instalación fotovoltaica son:

- Regulador
- Baterías
- Inversor

Figura 6. Elementos adicionales de una instalación fotovoltaica.

d.1.6.1 Regulador

Es el elemento que regula la inyección de corriente desde los paneles a la batería. El regulador interrumpe el paso de energía cuando la batería se halla totalmente cargada evitando así los negativos efectos derivados de una sobrecarga. En todo momento el regulador controla el estado de carga de la batería para permitir el paso de energía eléctrica proveniente de los paneles cuando esta empieza a bajar.

d.1.6.2 Batería

Almacena la energía de los paneles para los momentos en que no hay sol, o para los momentos en que las características de la energía proporcionada por los paneles no son suficientes o adecuadas para satisfacer la demanda (falta de potencia al atardecer, amanecer, días nublados). La naturaleza de la radiación solar es variable a lo largo del día y del año, la batería es el elemento que solventa este problema ofreciendo una disponibilidad de energía de manera uniforme durante todo el año.

d.1.6.3 Inversores

El elemento que transforma las características de la corriente de continua a alterna. La mayoría de los aparatos eléctricos funcionan con corriente alterna y tanto los paneles como las baterías suministran energía eléctrica en forma de corriente continua. Es por ello que se hace necesario este elemento que modifique la naturaleza de la corriente y la haga apta para su consumo por muchos aparatos.

d.2 CAPÍTULO II: Introducción a las Redes Neuronales Artificiales

d.2.1 Estructura Básica

La neurona artificial es la unidad fundamental de procesamiento de la información de cualquier modelo de RNA y a partir de ahora la identificaremos con la letra k. Fue diseñada para imitar las características de primer orden de la neurona humana. Su modelo se muestra en la figura 7 y en ella se identifican tres elementos básicos.

d.2.1.1 Un conjunto de sinapsis o conexiones provenientes de otras neuronas o del exterior

Cada conexión está caracterizada por un peso o fortaleza propio, cuya función es análoga a la de la unión sináptica de la neurona biológica, y que se identificará, a partir de este momento, con la letra W con dos subíndices, el primero se refiere a la neurona analizada y el segundo a la entrada de la sinapsis a la cual se refiere el peso. El peso W_{kp}, por ejemplo, corresponde a la conexión entre la neurona k y la neurona p de la capa previa, y es positivo si la sinapsis asociada es excitadora, negativo, si es inhibidora y 0 si la conexión no existe.

d.2.1.2 Un sumador (Σ)

Para sumar las señales de entrada, pesadas por sus respectivas conexiones. Específicamente, una señal X_p que se encuentra a la entrada de la sinapsis p que la conecta a la neurona k, es multiplicada por el peso sináptico W_{kp} . Estas entradas ponderadas son sumadas constituyendo esta operación una combinación lineal, que se identifica en la figura como NET_k.

d.2.1.3 Una función de activación (F)

Para limitar la amplitud de la salida (A_k) de la neurona k. Esta función limita el rango de amplitud permisible de la señal de salida a valores finitos, motivo por el cual también se conoce como función ‘squashing’. Típicamente el rango de amplitud normalizado de salida de una neurona es el intervalo cerrado [0,1] o el intervalo [-1,1].

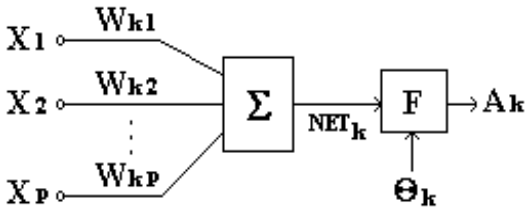


Figura 7.- Modelo no lineal de una neurona artificial.

Las señales de entrada (X) son normalmente continuas aunque también pueden ser discretas.

El modelo de la neurona puede además incluir una entrada adicional aplicada externamente a la función de activación. Si es negativa es conocida como entrada de umbral y denotada por Θ_k, como se muestra en la figura 7, y tiene el efecto de disminuir la entrada neta a la función de activación F de la neurona. Es un parámetro externo a la neurona k. En términos matemáticos, la salida (A_k) en este caso será igual a:

$$A_k = F (NET_k - \Theta_k) \quad (1)$$

F(.) es la función de activación.

Si la entrada adicional es positiva es más conocida como bias, y hay autores que acostumbran a denotarlo por B_k para diferenciarlo del umbral ya que realiza el efecto contrario, es decir, aumenta la entrada neta a la función de activación. Matemáticamente la salida correspondiente es:

$$A_k = F (NET_k + B_k) \quad (2)$$

El trabajo de la neurona artificial puede resumirse entonces de forma muy simple:

Una neurona k recibe un conjunto de entradas $X_1, X_2, \ldots X_p$ (cada una de las cuales representa la salida de otra neurona de una capa previa o, simplemente, proviene del exterior), y calcula un valor de salida A_k (que envía a otras neuronas). Para ello, cada entrada es multiplicada por un peso (análogo a la fortaleza sináptica de la conexión que

une la neurona k con la señal X), que es la potencia con que dicha entrada afecta a la neurona, y todas las "entradas pesadas" son entonces sumadas para determinar el nivel de activación o combinación lineal de la neurona (NET_k), siendo:

$$NET_k = \sum_{j=1}^{p} W_{kj} X_j \qquad (3)$$

Donde W_{k1}, W_{k2},…, W_{kp}, son los pesos sinápticos de las conexiones que llegan a la neurona k.

Finalmente, una función de activación determina el valor de salida, utilizando la expresión vista anteriormente para la salida A_k.

De manera equivalente las ecuaciones anteriores se pueden reformular adicionando una nueva sinapsis de entrada X_o = +1 ó -1 y peso igual a B_k o Θ_k, respectivamente, como:

$$V_k = \sum_{j=0}^{p} W_{kj} X_j \qquad \text{y} \qquad A_k = F(V_k) \qquad (4)$$

Y el modelo de la neurona k queda como en la figura 8, equivalente matemáticamente al anterior.

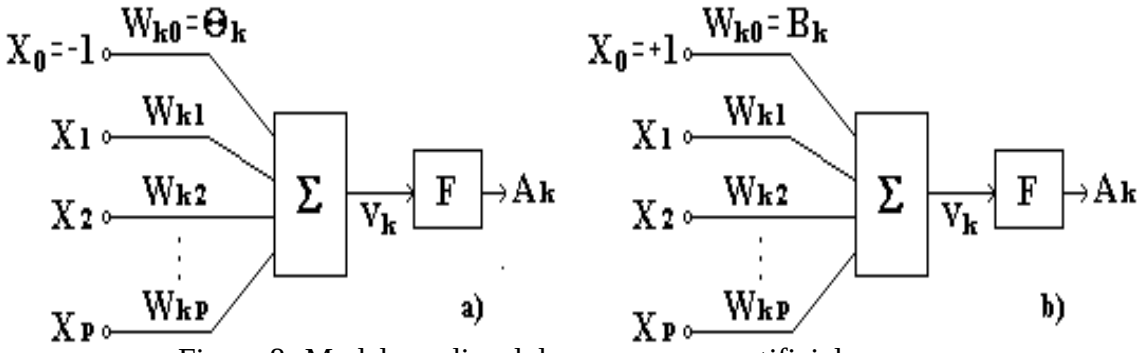


Figura 8- Modelo no lineal de una neurona artificial.

a) Con umbral adicional.

b) Con bias adicional.

Nota: El uso del umbral o el bias tiene el efecto de aplicar una transformación afín a la salida de la combinación lineal (NET_k) en el modelo, como se muestra por:

$$V_k = NET_k - \Theta_k \quad \text{o por} \quad V_k = NET_k + B_k \qquad (5)$$

En particular, dependiendo de si se trata de umbral ($\Theta_k < 0$) o de bias ($\Theta_k > 0 = B_k$), la relación entre el nivel de activación interna efectiva (V_k), también llamado potencial de activación, y la combinación lineal de la salida NET_k , es modificada como se muestra en la figura 9.

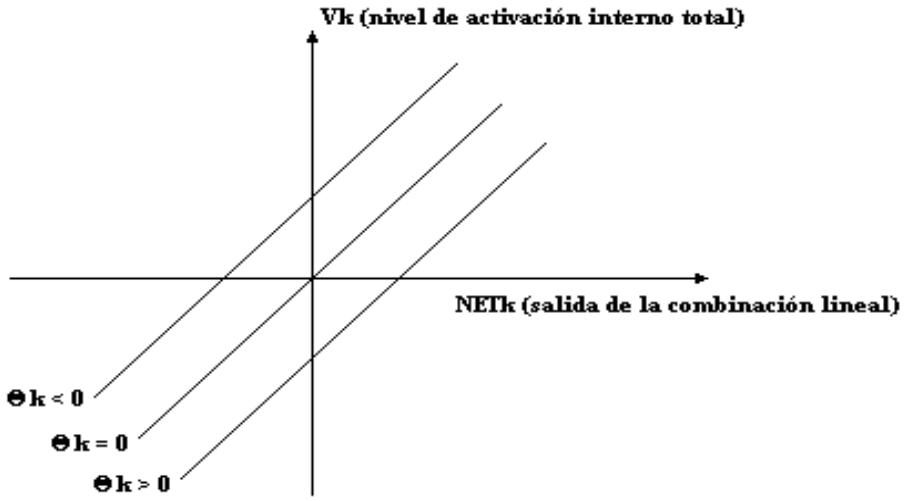


Figura 9.- Transformación afín producida por la presencia de un umbral.

Note que como resultado de la transformación afín el gráfico V_k vs NET_k no pasa por el origen. Esto, como se verá más adelante, contribuye a mejorar el entrenamiento de la red.

d.2.1.3.1 Funciones de activación más usadas en las redes modernas.

La función de activación, denotada por F en las figuras 7 y 8, define la salida de la neurona en términos del nivel de actividad de sus entradas. Inicialmente no se utilizaban estas funciones, siendo la neurona un simple elemento lineal de cálculo. En las RNA modernas la función de activación hace que la neurona artificial simule más exactamente las características transferenciales no lineales de la neurona biológica y

permite, por tanto, un funcionamiento más general de las mismas, lo cual las hace muy atractivas para el trabajo con sistemas no lineales.

Existen cuatro tipos básicos de funciones de activación que dan lugar a distintos tipos de neuronas artificiales. Ellas son:

- Función escalón o umbral.
- Función lineal.
- Función sigmoidal.
- Función gaussiana.

d.2.2 Clasificación de las redes neuronales artificiales.

Aunque una simple neurona puede realizar ciertas tareas, sobre todo de reconocimiento de patrones simples, la potencia está realmente cuando las neuronas se conectan formando redes. La manera en la cual esto se hace está íntimamente relacionada con el algoritmo de aprendizaje que será utilizado para entrenar la red. En cualquier caso se pueden encontrar tres tipos de neuronas:

- Aquellas que reciben estímulos externos y que toman la información de la entrada. Están relacionadas con el aparato sensorial en el sistema biológico, mientras que en el artificial pueden ser señales provenientes de sensores o de otros sectores del sistema. Se llaman **nodos de entrada** y no realizan cálculos como los explicados para las neuronas.
- Las neuronas de entrada trasmiten la información a ciertos elementos internos que se ocupan de su procesado. Es en las sinapsis y neuronas correspondientes a este segundo nivel donde se genera cualquier tipo de representación interna de la información. Puesto que no tienen contacto con el exterior, estos elementos se denominan **neuronas ocultas**.
- Una vez finalizado el período de procesamiento, la información llega a las unidades de salida, cuya misión es dar la respuesta del sistema, estas señales pueden controlar directamente potencias u otros sistemas. Las neuronas son conocidas como **neuronas de salida**.

En general se pueden identificar diferentes arquitecturas neuronales dependiendo de la forma en que se conectan las neuronas. Ellas son:

- Redes de alimentación hacia adelante o feedforward.
- Redes recurrentes.
- Redes con estructuras enrejadas.

Aunque ya se ha dicho, debe señalarse que alrededor del 90 % de las RNA que se utilizan en ingeniería hoy día, en nuestras aplicaciones, tienen arquitectura feedforward.

d.2.2.1 Entrenamiento de las Redes Neuronales Artificiales.

El funcionamiento de una red consta de dos etapas: la de entrenamiento o aprendizaje y la de reconocimiento. Es, precisamente, la capacidad de aprender, una de las propiedades más interesantes y de mayor significado práctico de las redes neuronales. Una red aprende de sus alrededores a través de un proceso iterativo en el cual se producen cambios en los valores de los pesos sinápticos y de los umbrales o biases de sus neuronas y que se llama algoritmo de aprendizaje. El tipo de aprendizaje está determinado por la manera en la cual tiene lugar el cambio de esos parámetros.

Inicialmente no se conoce el conjunto de pesos sinápticos que permite obtener la salida deseada para los diferentes estímulos aplicados, y se supone uno de forma aleatoria. La etapa de entrenamiento se inicia cuando se presentan a la red de forma secuencial los vectores o patrones de entrada con el objetivo de ajustar los pesos, de manera que la aplicación de un conjunto de entradas produzca el conjunto de salidas deseado con un error menor que el permisible. Durante el entrenamiento, en cada paso del proceso iterativo, se cambian los pesos los pesos de la red que gradualmente convergen a valores tales que permitan que cada vector de entrada produzca el vector de salida deseado. El aprendizaje incluye un proceso estadístico por lo que no existe una única salida para los pesos y los umbrales o biases de la red.

Matemáticamente durante el aprendizaje:

$$W_{kj}(n+1) = W_{kj}(n) + \Delta W_{kj}(n) \qquad (6)$$

Dónde:

n: es el número de la iteración.

W_{kj} (n): es el peso sináptico entre las neuronas k de la capa actual y j de la capa previa.

ΔW_{kj} (n) es el ajuste aplicado a los pesos en el paso n.

Los algoritmos de entrenamiento se clasifican en supervisados y no supervisados.

Luego de la etapa de aprendizaje la red es capaz ya de entrar a la etapa de reconocimiento en la cual puede presentarse a la red un vector nuevo que no formó parte del entrenamiento, y entonces la red deberá decidir a cuál de los vectores de entrada conocidos se parece más y dar un vector de salida en consecuencia.

d.2.2.1.1 Entrenamiento supervisado

Incluye la modificación de los pesos sinápticos de la RNA aplicando un conjunto de patrones de entrenamiento o ejemplos de tareas, cada ejemplo consiste de una señal de entrada única y su correspondiente salida deseada. Durante el entrenamiento los pesos se modifican de forma que se minimice la diferencia entre la respuesta deseada y la salida actual producida por la señal de entrada, de acuerdo con un criterio estadístico apropiado. El entrenamiento es repetido para muchos ejemplos del conjunto hasta que la red permanezca en estado estable donde no hay un cambio significativo de los pesos. Los ejemplos de entrenamiento son reaplicados durante el entrenamiento pero en orden diferente cada vez. Requiere que cada vector de entrada tenga su vector objetivo representando la salida deseada; juntos son llamados “par de entrenamiento”. Por eso se dice que es supervisado siendo la salida deseada “el maestro”.

Usualmente una red es entrenada para un número dado de pares de entrenamiento. El procedimiento es como sigue: se aplica un vector de entrada y se calcula la salida de la red, la cual se compara con el correspondiente vector objetivo. La diferencia (error) es utilizada para cambiar los pesos de acuerdo con un algoritmo que tiende a minimizar el error. Los vectores del par de entrenamiento se aplican secuencialmente, los errores se calculan y los pesos se ajustan para cada vector, hasta que el error para el conjunto de entrenamiento completo esté por debajo de un nivel aceptable y prefijado.

Cuando es procesado el último vector de entrada se pregunta si el error es menor que el permisible y en caso negativo se repite nuevamente el proceso para todos los patrones de entrada. En caso positivo termina el proceso de entrenamiento y queda fija la matriz de pesos de interconexión.

A pesar de tener éxito en muchas aplicaciones, entre ellas las de ingeniería, el algoritmo supervisado ha sido criticado por ser poco adecuado biológicamente; es difícil concebir un mecanismo de entrenamiento en el cerebro que compare las salidas actuales y la deseada.

d.2.2.1.2 Entrenamiento no supervisado.

En él no se requiere vector objetivo para la salida, y no se realiza, por consiguiente, una comparación con un ideal predeterminado. El conjunto de entrenamiento consiste sólo de un vector de entrada. El algoritmo de entrenamiento modifica los pesos de la red para producir un vector de salida. La aplicación de un vector de entrenamiento o de uno similar a él producirá el mismo vector de salida. El proceso de entrenamiento en consecuencia extrae las propiedades estadísticas de un conjunto de entrenamiento y grupos de vectores similares en clases. Aplicando un vector de una clase dada a la entrada se producirá un vector de salida específico, pero no hay forma de determinar antes del entrenamiento cual patrón de salida específico será producido por una clase de vector de entrada dado. De aquí se deduce que la salida de tales redes tienen generalmente que ser transformadas en una forma comprensible posteriormente al proceso de entrenamiento. Esto no representa un problema serio ya que es usualmente simple identificar la relación entrada - salida establecida por la red.

d.2.3 El Perceptrón y los primeros días de las redes neuronales artificiales

A fines de la década de los 50 e inicios de los 60, el sicólogo Frank Rosemblatt, desarrolló un algoritmo para el ajuste de los pesos de una red que tenía tres tipos de unidades: sensoriales, asociativas y de respuesta, existiendo entre las dos primeras pesos fijos y entre las dos segundas pesos ajustables. Las uniones entre las unidades sensoriales y asociativas se usaban simplemente para extraer los patrones de los

alrededores, mientras que el aprendizaje se producía mediante el cambio de los pesos ajustables. A esta red se le llamó perceptrón.

Una simple neurona artificial, como la descrita en este curso, con pesos ajustables, conectados a diferentes variables de un patrón de entrada X, con umbral o bias y con función de activación umbral sin signo opera como la usada por Rosemblatt para simular la neurona biológica y por ello es conocida como perceptrón artificial o simplemente perceptrón.

El perceptrón fue la forma más simple de RNA y a pesar de sus limitaciones ha sido ampliamente estudiado y constituye la base de redes más complejas. Hoy día también se conoce con este nombre a una red formada por varias de esas neuronas artificiales formando una capa simple e incluso a redes con una o más capas ocultas donde los pesos entre las neuronas de la capa de salida y los de la última capa oculta son los únicos que se cambian durante el entrenamiento manteniéndose constantes el resto de los pesos de la red, por lo cual las capas ocultas reciben el nombre de preprocesadores no lineales.

La mayor limitación del perceptrón en cualquiera de sus variantes consiste en que no es capaz de clasificar patrones que no sean linealmente separables (es decir, que no puedan ser ubicados en diferentes lados de un hiperplano), y la gran mayoría de los problemas reales no lo son. Esto ocurre, como se demostró después matemáticamente, con independencia del tipo de función de activación que se utilice y fue lo que dio inicio a la época oscura de las redes neuronales.

d.3 CAPÍTULO III: Fundamentos de Lógica Difusa

d.3.1 Introducción

Establecida desde hace varios años como una tecnología de vanguardia, la lógica difusa ó fuzzy logic está penetrando con fuerza en el mundo del control y promete convertirse en el método de mando universal de toda clase de dispositivos eléctricos y electrónicos.
La Lógica Difusa es, desde un punto de vista práctico, un método de razonamiento estadístico que permite especificar los problemas de control del mundo real en términos probabilísticos, sin necesidad de recurrir a modelos matemáticos y con un nivel de abstracción mucho más elevado (Rojas Purón, 2003). En contraste con la lógica convencional, que utiliza conceptos absolutos para referirse a una realidad, la lógica difusa la define en grados variables de pertenencias a los mismos, siguiendo patrones de razonamientos similares a los del pensamiento humano.

Así por ejemplo, mientras dentro del marco rígido de la lógica formal un recinto está solamente ¨ oscuro ¨ (0) o claro (1), para la lógica difusa son posibles también todas las condiciones relativas intermedias percibidas por la experiencia humana como ¨ muy claro ¨ , ¨ algo oscuro ¨ , ¨ ligeramente claro ¨ , ¨ extremadamente oscuro ¨, etc. Las condiciones extremas o absolutas asumidas por la lógica formal son sólo caso particular dentro del universo de la lógica difusa. Esta última nos permite ser relativamente imprecisos en la representación de un problema y aún así llegar a la solución correcta.

La lógica difusa es una ciencia relativamente reciente, aunque la idea de vaguedad que promulga ya había sido discutida desde el siglo XVIII por Berkeley, Hume, Kant, Bayes y otros pensadores. Incluso Aristóteles, creador de la lógica formal, admitía la existencia de diferentes grados de verdad y falsedad. Sin embargo, es Lofti Zadeth, profesor de computadores de la Universidad de Berkeley, quien en 1965 propone un

método de razonamiento abstracto similar a patrón de pensamiento humano para representar los problemas de control del mundo real y crea la lógica difusa. A comienzos de los setenta 70s, el ingeniero Ebrahim Mandani, basado en la teoría de Zadeth, desarrolla el primer sistema de control fuzzy práctico, aplicado a una máquina de vapor.

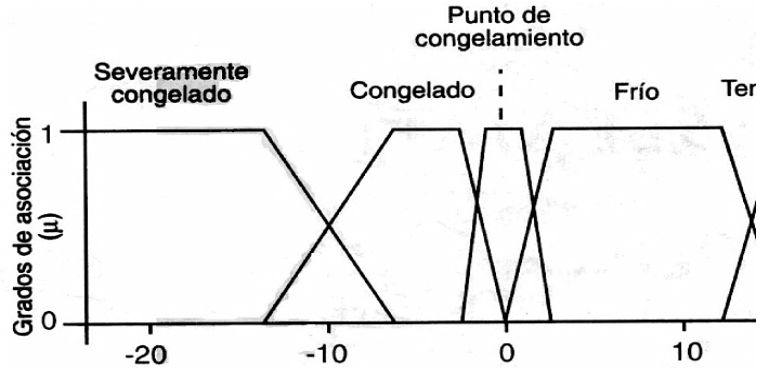


Figura 10. Valores reales y valores difusos.

El sistema de Mandani combinaba la experiencia de un operador humano con un conjunto de reglas lógicas para controlar automáticamente la cantidad de vapor o throuttle y la temperatura de la caldera de acuerdo a la presión de esta última y la velocidad de la máquina. Las dos entradas (velocidad y presión) eran procesadas de acuerdo a un algoritmo y producían dos salidas (vapor y temperatura) que actuaban sobre el proceso en la forma deseada. A finales de los 70s, los ingenieros daneses Lauritz Meter Holmblad y Jens Jurgen Ostergaard desarrollan el primer sistema de control fuzzy comercial, destinado a una planta de cemento.

A pesar de que han transcurrido muchas décadas desde su creación, hasta hace poco el mundo occidental está reconociendo el verdadero valor de la fuzzy logic. Además de factores culturales, dos razones explican esta actitud. En primer lugar la palabra fuzzy sugería algo confuso y sin forma. Esto alejaba sicológicamente a la comunidad técnica de la idea de utilizarla prácticamente. En segundo lugar, no había forma de probar analíticamente funcionaba correctamente debido a que la fuzzy logic, en contraste con la teoría de control convencional, no estaba basada en modelos matemáticos. La situación en Japón fue diferente.

Los japoneses aceptaron fonéticamente la palabra fuzzy, sin traducción y libre de las connotaciones negativas normalmente asociadas con el término, y adoptaron la teoría de Zadeth como propia. Esto le permitió evolcionar más tempranamente que los occidentales a la fase experimental. Así lograron comprobar que no eran necesarias las imposiciones para desarrollar y producir sistemas inteligentes.

Actualmente Japón es el líder mundial en la producción de aplicaciones basadas en fuzzy logic, con ventas estimadas para 1997 en 6.1 billones de dólares. En Japón funcionan también el más espectacular de todos los sistemas fuzzy creados por el hombre: el subterráneo de Sendai, inaugurado en 1987. Desde entonces, un controlador inteligente ha mantenido los trenes rodando velozmente a lo largo de la ruta, frenando y acelerando suavemente, deslizándose entre estaciones y deteniéndose exactamente, sin perder un segundo ni sacudir bruscamente un solo pasajero.

El interés actual en la lógica difusa surge también de la necesidad impuesta por nuevas tecnologías como la inteligencia artificial y las redes neuronales de disponer de sistemas expertos capaces de procesar información, tomar decisiones y responder a estímulos en forma similar al cerebro humano. Los investigadores están utilizando técnicas fuzzy para diseñar redes neuronales y estas, a su vez, para producir reglas de lógica difusa.

Una lavadora controlada por fuzzy logic, por ejemplo, puede distinguir una prenda ligeramente sucia, lavando esta última con mayor vigor que la primera. Adicionalmente, puede calcular automáticamente el volumen de la carga de ropa, la velocidad, el nivel de agua y detergente y los tiempos óptimos de lavado, centrifugado, enjuague, agitación, etc. Si se agrega una red neuronal, esta le permitirá aprender nuevos hábitos mediante el desarrollo dinámico de nuevas reglas (Rojas Purón, 2003).

Actualmente, muchos productos de uso corriente (cámaras fotográficas y de video, electrodomésticos, etc); así como una gran variedad de controladores industriales, dispositivos médicos y otros sistemas relativamente complejos, están basados en fuzzy logic. La tendencia continuará a medida que los diseñadores encuentren nuevas

aplicaciones para esta tecnología. Otros usos de la fuzzy logic incluyen (Rojas Purón, 2003):

- Modelos de control de trenes, aviones, botes y otras naves.
- Sistemas de seguridad para el hogar y la oficina.
- Sistemas de control y predicción climáticos.
- Controladores de velocidad de motores AC y DC.
- Servomecanismos y robots.
- Control de líneas de producción.

La facilidad de la fuzzy logic para adquirir y representar conocimientos ha estimulado también su aplicación en la solución de problemas sociológicos, sicológicos, políticos, administrativos, económicos, epidemiológicos y de otras disciplinas. Existen incluso paquetes de software basados en fuzzy logic, como el Fuzzy Decisión Maker de Fuzzy Logic Inc. Que ayudan a las personas a tomar decisiones de todo tipo, desde solucionar un problema familiar hasta adquirir uan casa, un carro o un VCR.

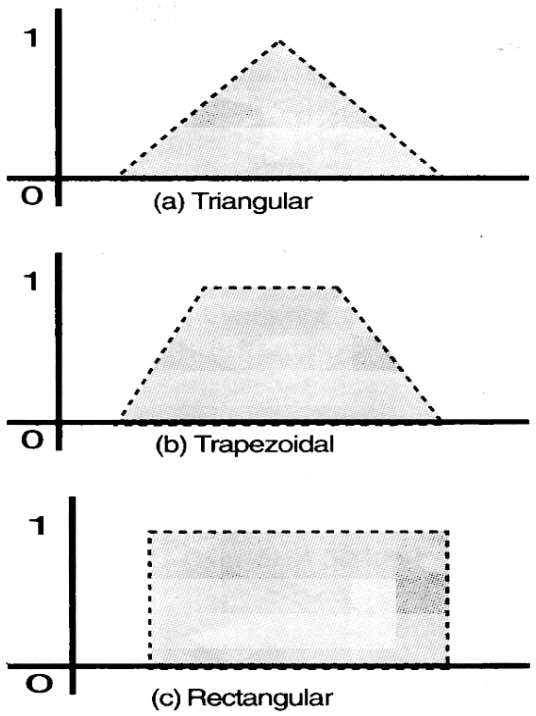


Figura 11. Funciones de pertenencias comunes.

En general existen cinco tipos de situaciones en las cuales la aplicación de técnicas de control fuzzy resulta ventajosa o necesaria:

1. Sistemas complejos que son difíciles de modelar por métodos convencionales.
2. Sistemas controlados por expertos humanos.
3. Sistemas con entradas y salidas complejas y continuas.
4. Sistemas que utilizan la observación humana como entrada o como base de las reglas.
5. Sistemas que son confusos por naturaleza, como los encontrados en las ciencias sociales y del comportamiento.

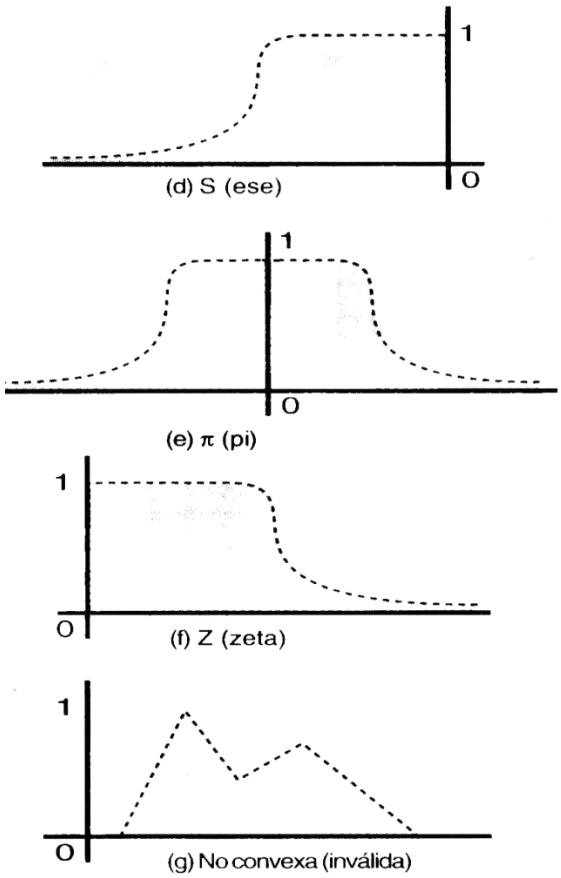


Figura 12. Otras funciones de pertenencias comunes.

La Fuzzy Logic puede abarcar muchos aspectos de la vida moderna. Sin embargo, la más grande dificultad con la que se enfrenta su universalización es la polarización cultural del mundo occidental a favor de la lógica tradicional del ¨1 ¨ y el ¨ 0 ¨ y de los modelos matemáticos lineales. Estos últimos son adecuados para describir procesos simples. No obstante, la mayoría de los procesos del mundo real son no lineales y

resultan demasiado complejos para ser modelados matemáticamente. Este es el tipo de problemas que deben resolverse por fuzzy logic.

d.3.2 Conceptos y reglas básicas

Para comprender intuitivamente el concepto de lógica difusa, consideremos como ejemplo un florero A con diez rosas rojas y un florero B con diez orquídeas. En los términos absolutos de la lógica formal, proposiciones del tipo A es un florero de orquídeas y B no es un florero de rosas pueden negarse o afirmarse categóricamente sin crear confusión. Suponga que en el florero A se cambian dos rosas por dos orquídeas. Evidentemente, ahora no podemos afirmar con la misma determinación que A es un florero de rosas porque también contiene orquídeas. En este caso, resulta más preciso decir que A es mayormente un florero de rosas o que A es particularmente un florero de orquídeas. Con este tipo de ambigüedades presentadas en la descripción de la realidad es que trabaja la lógica difusa.

En lógica difusa, los diferentes valores que pueden adoptar una variable del mundo real se subdividen o clasifican en grupos y cada valor dentro de un grupo se le asigna una cuota de pertenencia al mismo denominada grado de asociación o membresía. Esta última se designa generalmente como y puede adoptar valores entre 0 y 1. Los grupos se denominan μ conjuntos difusos (ajustes difusos ó fuzzy sets).

Las formas triangulares y trapezoidales son las más utilizadas en aplicaciones de control difuso debido a que simplifican la manipulación aritmética y representan adecuadamente la experiencia humana. Sin embargo, son también posibles otras formas (figura 11 y 12), incluyendo la función rectangular utilizada por la lógica tradicional.

Una vez representadas en forma de conjuntos o valores de lógica difusa, las variables de entrada y de salida de un proceso deben ser relacionadas mediante reglas lógicas. Estas reglas, derivadas de la experiencia práctica o por ensayo y error, son las que describen y determinan el comportamiento final del sistema.

Las reglas utilizadas en la lógica difusa son de tipo:

Si (tal cualidad ó expresión) y (tal cualidad ó expresión) … **Entonces** … (resultado)

SI una serie de circunstancias particulares de entrada ocurre, **ENTONCES** una serie de acciones particulares de salidas resultan. Por ejemplo **SI** M es grande y T es un poco alta, **ENTONCES** P es muy fuerte.

Las reglas se construyen en base a operadores lógicos similares a los utilizados en álgebra booleana, pero con una significación diferente. Cada regla específica las condiciones de entrada que deben cumplirse para que la regla sea evaluada como válida y los valores que en consecuencia serán transferidos a la salida.
Los tres operadores básicos de la lógica difusa son:

- AND.
- OR.
- NOT.

Las operaciones AND, OR y NOT definidas por estos operadores se denominan también en su orden, conjunción o intersección, disyunción o unión y negación o complemento. En la figura 13 se ilustra la forma de evaluar cada una de ellas. Aunque en este caso las reglas se aplican sobre dos entradas (S1 y S2), las mismas son extensivas a múltiples entradas.
La operación AND (Y) de dos valores difusos μx igual al menor de los valores de entrada.

Por ejemplo, si $\mu A = 0.8$ y $\mu B = 0.3$, entonces $\mu x = 0.3$.

En el caso de la figura 13.a, la regla IF S1 AND S2 THEN… se aplica para los valores de entrada con $\mu > 0$ comprendidos dentro del área sombreada. Por tanto, los valores difusos que se transfieren a la salida son los que siguen el contorno dibujado en línea oscura.La operación OR (O) de dos valores difusos μA y μB produce como resultado un valor difuso μx igual al mayor valor de entrada. Por ejemplo, si $\mu A = 0.6$ o $\mu B = 0.4$, entonces $\mu x = 0.6$. En el caso de la figura 13b, la regla IF S1 OR S2 THEN … se aplica para todos los valores de entrada con $\mu > 0$ comprendidos dentro de cualquier forma. Nuevamente, los valores difusos que se transfieren a la salida son los que siguen el contorno dibujado en línea oscura. Si la regla contiene más de un operador OR, el valor

difuso que se aplica a la salida es el máximo de cada combinación individual. La operación NOT (NO) de un valor difuso μA produce como resultado un valor difuso μx igual a 1-μx. Por ejemplo, si μA = 0.1, entonces μx = 0.1. en el caso de la figura 13c, la regla IF NOT S1 THEN … se aplica para todos los valores dentro de la forma de S1 con μ > 0. Los valores difusos que se aplican a la salida son los que siguen el contorno dibujado en línea oscura. La zona sombreada corresponde a los valores originales complementados. El efecto neto del operador NOT es, por tanto, la inversión de las formas. Las figuras 13d y 13e ilustran como traba el operador NOT en operaciones AND y OR.

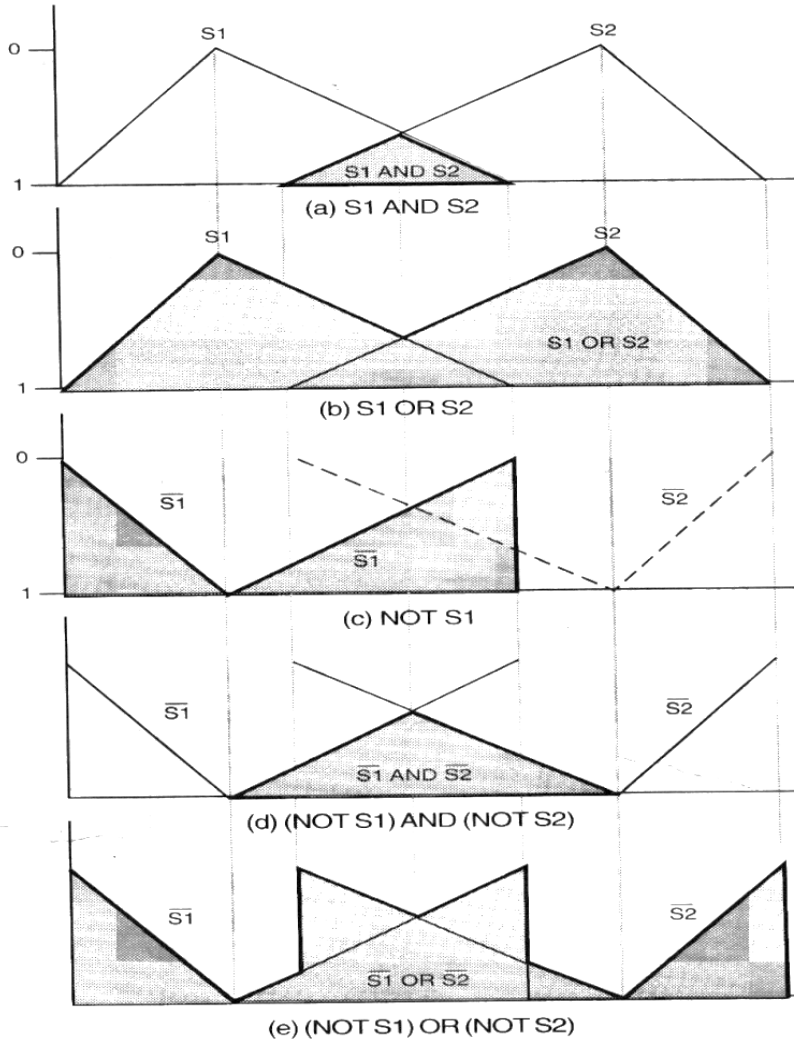


Figura 13. Operadores de lógica difusa.

Se debe tener en cuenta que el desarrollo de cualquier sistema difuso parte de la experiencia de un experto humano. (Costa Branco, 1998) (Costa Branco, et al., 1998)

(Ramos, 1994).La primera tarea del diseñador es, por tanto, comprender como el experto trabaja en el área de su dominio. Este proceso de asimilación no es fácil porque la verdadera maestría no se utiliza paso a paso sino en forma intuitiva. Una vez traducido este conocimiento a unas reglas, el sistema difuso resultante debe ser capaz de controlar la máquina casi en la misma forma como lo haría el experto.

d.4 CAPÍTULO: IV Sistemas Neuro-Difusos

Los sistemas Neuro-difusos forman parte de una nueva tecnología de computación llamada computación flexible (*soft-computing*), la cual tiene su origen en la teoría de la Inteligencia Artificial. La computación flexible engloba un conjunto de técnicas que tienen en común la robustez en el manejo de la información imprecisa e incierta que existe en los problemas relacionados con el mundo real (por ejemplo: reconocimiento de formas, clasificación, toma de decisiones, etc.). En algunos casos, las técnicas de computación flexible pueden ser combinadas para aprovechar sus ventajas individuales; algunas de estas técnicas son: la lógica difusa, las redes neuronales, los algoritmos evolutivos o algoritmos genéticos, la teoría del caos, la teoría del aprendizaje, el razonamiento aproximado. (Chahuara Quispe, 2005)

Las neuronas difusas son las que permiten relaciones difusas adaptativas y operadores en la sinapsis, esto para convertir las entradas externas en salidas sinápticas. Es importante resaltar que la fusificación puede ser realizada en todos los aspectos del trabajo de una neurona artificial (como son: las entradas los pesos, el operador de agregación, la función de transferencia, y la salida), pero los componentes principales en las redes neuronales difusas son:

1. La fusificación de las entradas dendríticas y
2. La operación de agregación de una neurona convencional.

Por ello es que se tienen como resultado una variedad de neuronas difusas ya que, si en vez de utilizar la suma utilizamos los operadores max, o min para realizar la agregación, o en lo general cualquier operador norma-T o norma-S.

d.4.1 Tipos de Neuronas Difusas

d.4.1.1 Neuronas exactas y neuronas difusas.

Una red neuronal consiste en unidades de procesamiento de información interconectadas llamadas “Neuronas Artificiales”. La estructura de una neurona artificial está constituida como se muestra en la figura 14 por entradas externas, dendritas, neurona (cuerpo) y un axón a través del cual la salida de la neurona es transmitida a otras.

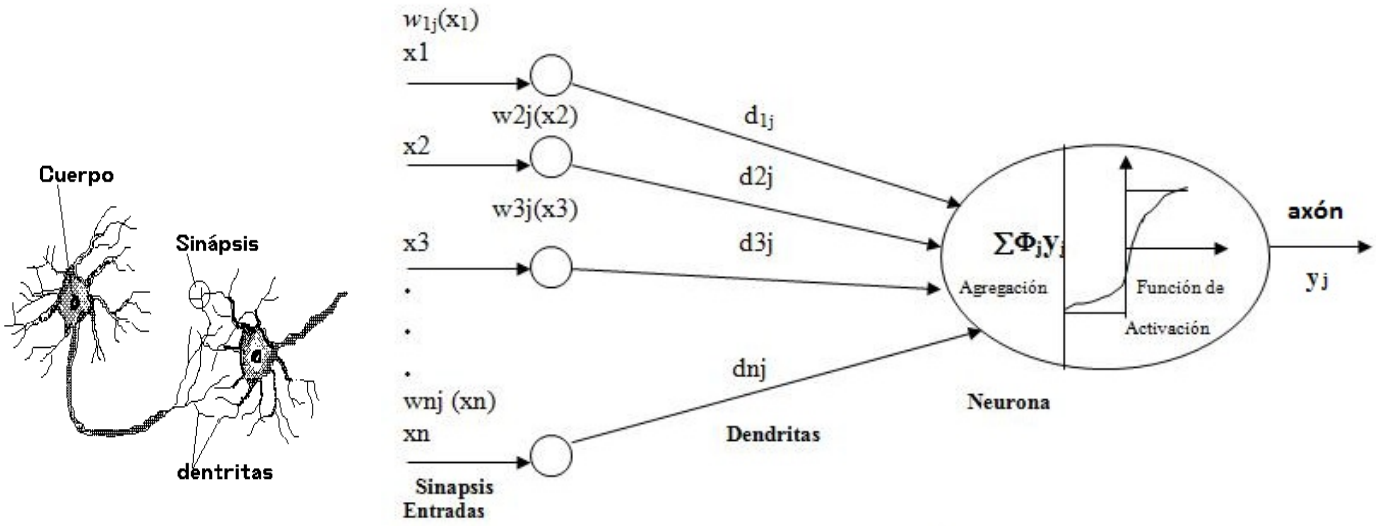


Figura 14.Modelo de una neurona artificial

Ubiquémonos en una de las neuronas de la red neuronal, digamos la j-ésima neurona. Esta neurona tiene un vector de entradas externas [x1, x2,. . ., xn] ^{T}, además es modificada con pesos w1j, w2j, w3j. . . , wnj los cuales representan un proceso sináptico. En general estos pesos están en función de las entradas externa, es decir, w1j(x1), w2j(x2), w3j(x3),. . ., wnj(xn). Cada salida sináptica (pesos) es la entrada a la

neurona, llamada entradas dendrítica o de información, de hecho la entrada a la neurona de la neurona es un vector de entradas dendríticas [d1j, d2j, d3j,. . ., dnj]^T , donde cada entrada dendrítica es una versión transformada de la entrada externa de la entrada xi esto es:

$$d_{ij} = w_{ij}(x_i) \tag{7}$$

La neurona produce una salida cuando la actividad de agregación de todas las entradas dendríticas excede un nivel de biasT_j.

Matemáticamente:

$$Ij = \sum_{i=1}^{n} dij \tag{8}$$

En esta fase de agregación se usan operadores de agregación generalmente normas T y S. Finalmente la salida de la neurona yj es realizada por otra operación esencial dentro de la misma neurona, esto es la función de activación. Matemáticamente:

$$y_j = \Phi_j \left[I_j , T_j \right] \tag{9}$$

Donde $\Phi_j$es la función de activación la cual describe el grado para la cual la neurona esactiva, $I_j$es el total de la fase de agregación sobre la neurona, y $T_j$es el bias.

$$y_j = sign\left[\sum_{i=1}^{N} w_{ij} + T_j \right] \tag{10}$$

d.4.1.2 Neuronas difusas y redes neuronales difusas

Básicamente la estructura de una neurona difusa es igual a la de una neurona artificial exceptuando que alguno o todos sus componentes y parámetros pueden ser descritos a través de matemáticas de la lógica difusa por lo tanto existen una gran variedad de posibilidades para la fusificación de las neuronas artificiales, pero todas poseen el interés de procesar información bajo el paradigma *logic-oriented.*

En la ilustración 15 se muestra una neurona difusa, donde las entradas externas vector x = [x1,x2, . . . ,xn]^T están definidas por señales difusas limitadas por funciones de membresía. Las entradas externas después de ser modificadas por los pesos sinápticos *wij*, se convierten en entradas dendríticas o entradas de información a la neurona. Las entradas dendríticas son procesadas por el operador de agregación *Ij* que selecciona el mínimo del producto de las d_{ij} ($d_{ij} = w_{ij}\ x_i$).

$$I_j = \bigwedge_{i=1}^{n} d_{ij} = \bigwedge_{i=1}^{n} x_i w_{ij} \qquad (11)$$

Por simplicidad se usa xi en vez de μi, cuando nos referimos a entradas difusas.

Este tipo de neurona difusa implementa la conjunción difusa (compuerta AND).

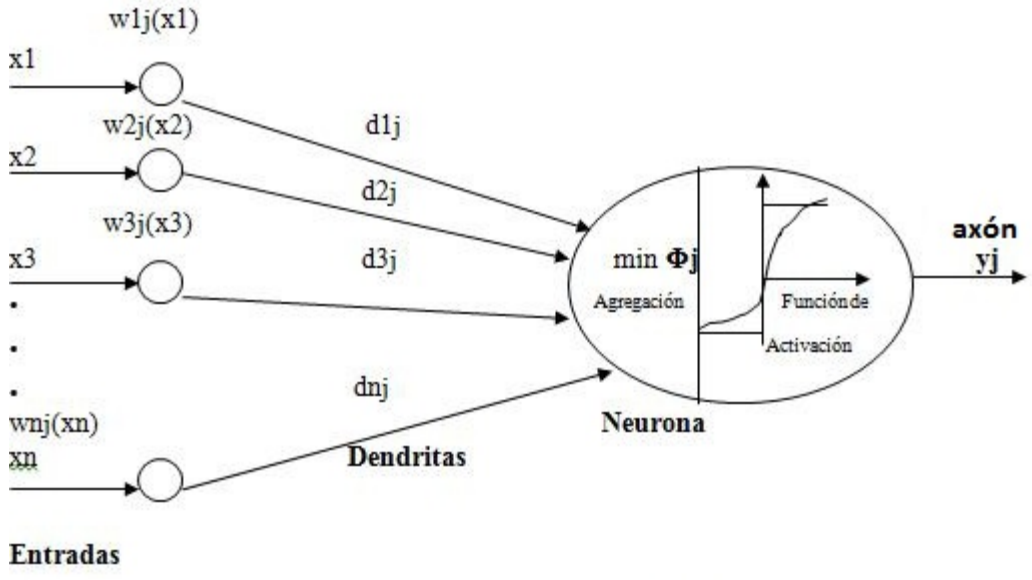


Figura 15. Neurona difusa min (AND)

La salida *yj* es un valor real dentro del intervalo [0-1] indicando el grado con el cual las entradas externas aplicadas son capaces de generar un valor lingüístico (LOW, MIDIUM, HIGH, etc.).

d.4.1.2.1 Neurona difusa MAX (OR).

Una neurona difusa MAX usa una función de agregación que selecciona el máximo de las entradas dendríticas la neurona:

$$I_j = \bigvee_{i=1}^{n} x_i w_{ij} \qquad (12)$$

Este tipo de neurona implementa la función lógica OR.

d.4.1.2.2 Neurona difusa MIN (AND).

Una neurona difusa MIN usa una función de agregación que selecciona el mínimo de las entradas dendríticas la neurona:

$$I_j = \bigwedge_{i=1}^{n} x_i w_{ij} \tag{13}$$

Llamada neurona difusa AND ya que implementa la función AND.

d.4.2 Neuronas difusas generalizadas y redes.

Consideremos una red neuronal constituida por m neuronas difusas las cuales aceptan n entradas. Como sabemos la fusificación dentro de una neurona puede ser en las entradas sinápticas (pesos), la operación de agregación y la función de transferencia. De hecho conjuntos difusos pueden ser utilizados para describir varios aspectos de procesamiento neuronal (Gupta M., and Knopf, G. K., 1992) algunas convenciones frecuentes de las redes neuronales difusas son:

d.4.2.1 Entradas sinápticas

El vector de entrada x = [x1, x2, x3, . . ., xn]^T para una neurona difusa pueden ser grados de membresía a un conjunto difuso.

d.4.2.2 Entradas dendríticas

Para cada j-ésima neurona de la red las entradas dendríticas son también limitadas por un grado de membresía. De hecho si decimos que u designa un elemento del universo del discurso genérico [0,1], usaremos las entradas dendríticas como conjuntos difusos para definir las cantidades difusas esto es:

$$dij = \sum_{j=1}^{m} \mu dij(u)/u, \mathrm{u} \in [0,1] \tag{14}$$

d.4.2.3 Valores agregados

La salida del operador de agregación en cada una de las m neuronas difusas de la red puede ser entendida como un grado de membresía en el intervalo binario de modo que tenemos:

$$Ij = \sum_{j=1}^{m} \mu Ij(u)/u, \mathrm{u} \in [0,1] \tag{15}$$

d.4.2.4 Salida de la neurona

Finalmente la salida yj de las m neuronas difusas pueden ser también grados de membresía a un conjunto difuso:

$$yj = \sum_{j=1}^{m} \mu yj(u)/u, \text{u} \in [0,1] \qquad (16)$$

La función de pesos w_{ij} que transforma una entrada externa x_i en una señal dendrítica d_{ij}para la *j-ésima* neurona difusa no tiene que ser solo una ganancia simple. Esta puede, en general, ser una relación difusa en todo el plano cartesiano $w_{ij}= x_i \times d_{ij}$. Una conjunción sináptica de la relación entre la entrada externa x_iy la entrada dendrítica d_{ij} asume muchas formas. Mas general tenemos que la entrada dendrítica d_{ij} definida por la composición $x_i o w_{ij}$de una señal de entrada difusa y la relación del peso se define por:

$$dij = xi \circ wij \qquad (17)$$

El concepto de negación difusa es usado cuando se producen entradas excitatorias e inhibitorias a una neurona difusa. Las salidas sinápticas *dij* , tal vez se modifican produciendo un efecto excitatorio o inhibitorio para la definición de una nueva variable δij o denotando también entradas excitatorias e inhibitorias recibidas por la neurona. Tenemos que:

$$\delta ij = \begin{cases} dij \text{ para entradas excitatorias} \\ \overline{dij} \text{ para entradas inhibitorias} \end{cases}$$

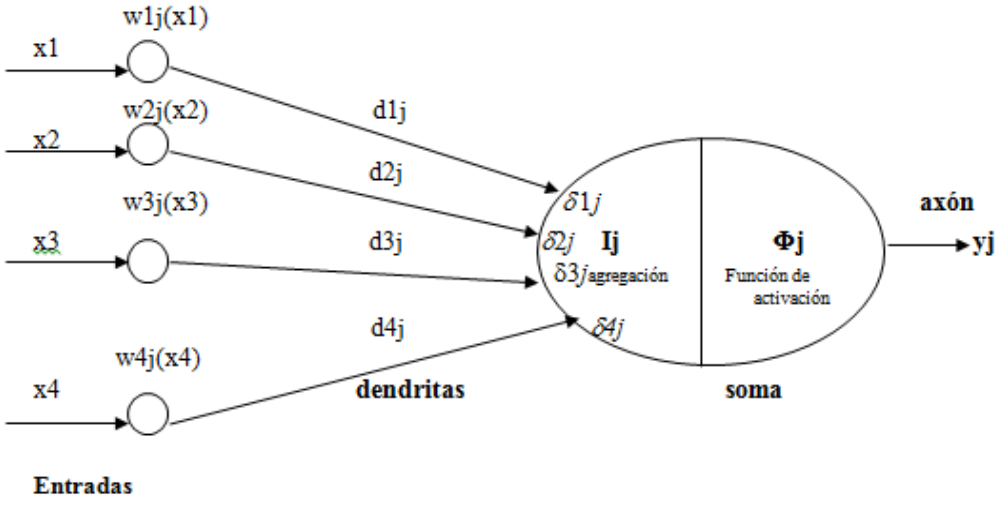


Figura 16. Entradas de exitación e Inhibición para una neurona difusa

d.4.3 Redes Neuronales Difusas Multicapa

Las neuronas discutidas en las secciones anteriores pueden ser conectadas para formar redes multicapa. Las redes multicapa que contemplaremos, tienen tres capas cada una un una implementando una función diferente. La capa de entrada simplemente envía las entradas para cada nodo oculto. La capa oculta está compuesta de neuronas tipo AND Y OR, estas neuronas performan una norma de operación dentro de la matriz de pesos y entradas. Las salidas de la capa oculta performan la operación de las salidas de la misma capa y la salida del vector de pesos. La norma de operación puede ser del tipo S o T.
En la figura 17 se muestra una red de neuronas difusas multicapa.

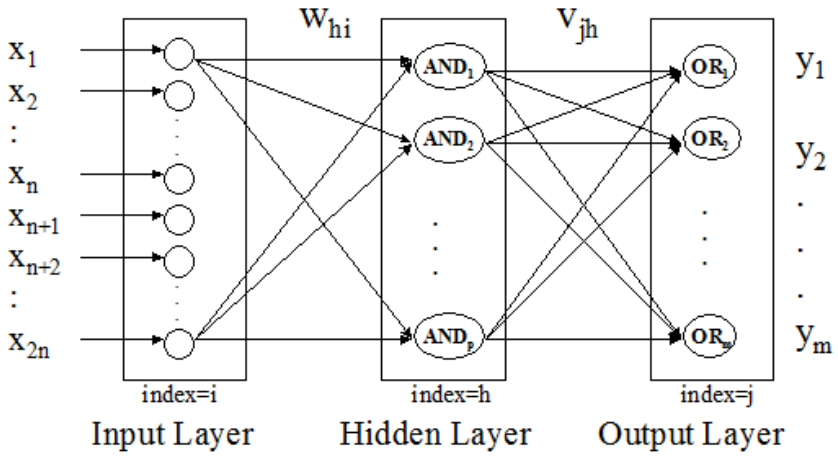


Figura 17. Red de neuronas difusas multicapa

d.4.3.1 Red Multicapa de Neuronas Difusas.

En la figura 17, la capa oculta usa neuronas AND que utilizan una norma tipo T de agregación. La salida de la capa oculta es denotada por Z^h donde h es el índice del nodo oculto.

$$z_h = \left[T_{i=1}^{n}\left(x_i S w_{hi}\right)\right] T \left[T_{i=1}^{n}\left(\overline{x}_i S w_{h(n+i)}\right)\right] \qquad \text{h} = 1,2,\ldots,\text{p} \tag{18}$$

Ésta es implementada en MATLAB con el producto de normas T y la suma probabilística de las normas S.

En la figura 17, la salida usa neuronas OR para formar una norma S de agregación. La salida de la red es denotada por y^j donde j es el índice de la neurona de salida.

$$y_j = \left[S_{h=1}^{p} \left(z_h \; T \; v_{hj} \right) \right] \qquad \text{j} = 1,2,\ldots,\text{m} \tag{19}$$

La evaluación de todas las neuronas difusas en la red multicapa puede ser combinada dentro de una sola función.

Podemos darnos cuenta que existe una relación muy estrecha de Redes Neuronales y Lógica Difusa ya que las dos técnicas pueden atacar al mismo tipo de problemas, pero a las redes neuronales se le puede dificultar la obtención de información de entrada, por lo cual se hace uso de la técnica de Lógica Difusa la cual, las entradas están en función de las reglas descritas por los diseñadores del sistema. Esto indica que puede haber una dependencia de una técnica a otra en ciertas etapas del procesamiento de información ya que pueden ir cambiando los datos en el procesamiento.

d.4.4 Métodos Difusos en Redes Neuronales

Los sistemas Neuro-Difusos combinan la capacidad de aprendizaje de las RNAs (Redes Neuronales Artificiales) con el poder de interpretación lingüística de los sistemas de inferencia difusos, obteniéndose los siguientes resultados:

- Aplicabilidad de los algoritmos de aprendizaje desarrollados para redes neuronales.
- Posibilidad de promover la integración de conocimiento (implícito que puede ser adquirido a través del aprendizaje y explícito que puede ser explicado y entendido).
- La posibilidad de extraer conocimiento para una base de reglas difusas a partir de un conjunto de datos. Existen sistemas de desarrollo que han logrado unir la Lógica Difusa con las Redes Neuronales, por ejemplo se tiene:

d.4.4.1 FSOM

FSOM (*FuzzySelf-OrganizingMaps*) consiste en un sistema difuso optimizado a partir de la técnica de los mapas auto-organizados de Kohonen.

d.4.4.2 NEFCLASS

El algoritmo NEFCLASS está basado en la estructura del perceptrón multicapa cuyos pesos son modelados por conjuntos difusos. Así, se preserva la estructura de una red neuronal, pero se permite la interpretación del sistema resultante por el sistema difuso asociado, es decir, la RNA deja de ser una "caja negra" de la neurona postsináptica.

d.4.4.3 ANFIS

El acrónimo ANFIS se deriva de sistema de inferencia difuso neuro adaptativo (adaptive neuro-fuzzyinferencesystem, en inglés). Usando un conjunto de datos de entrada y de salida, la función ANFIS construye un sistema de inferencia difuso cuyos parámetros de funciones de pertenencia son ajustados usando un algoritmo de backpropagation por si solo o en combinación con un método de mínimos cuadrados. Este ajuste permite que los sistemas difusos aprendan de los datos que están modelando (TheMathWorks, 2010).

ANFIS es el sistema de desarrollo utilizado en este proyecto de tesis.

d.4.4.4 FuzzyTech

Es un software que propone un método de desarrollo de sistemas Neuro-difuso similar a ANFIS.

d.4.4.5 Aprendizaje en un Sistema Neuro-difuso

El Sistema Neuro-difuso consiste de un sistema difuso tradicional excepto que cada etapa, puede ser representada por una capa de neuronas a las que se puede proveer capacidades de aprendizaje de Redes Neuronales para optimizar el conocimiento del sistema como muestra la ilustración 18.

Figura 18. Diagrama de bloques de un Sistema Neuro-difuso

En la capa de fusificación, cada función de pertenencia de entrada del antecedente de una regla difusa representa una neurona. Los parámetros de estas neuronas, como los vértices de las funciones de pertenencia, pueden ser entrenados para determinar la forma final y la ubicación de las funciones de pertenencia. Las salidas de estas neuronas funciones de pertenencia son conectadas a la capa de reglas difusas como lo especifiquen las reglas difusas y a través de enlaces con pesos que representan el proceso de agregación de las variable lingüísticas de entrada. La capa de reglas difusas representa la base de reglas difusas; cada neurona representa una regla difusa de tipo SI-ENTONCES. Las salidas de las neuronas están conectadas a la capa de defusificación a través de enlaces con pesos; los pesos de estos enlaces representan la significancia relativa de las reglas asociadas con las neuronas. Sus valores pueden ser asignados de acuerdo a conocimientos anteriores o inicializados como 1.0 y luego entrenados para reflejar su importancia real para las funciones de pertenencia de salida contenidas en la capa de defusificación. La función de la capa de defusificación es la evaluación de las reglas; en este cada consecuente “Entonces Y es B” como función de pertenencia de salida representa una neurona; la certeza de cada consecuente es calculada, y es considerada como lo bien que se ajustan las reglas que tienen el mismo consecuente (proceso de agregación del resultado). Los pesos de cada enlace de salida de estas neuronas representan los centros de área de cada función de pertenencia del consecuente y son entrenables, la salida final es entonces calculada usando algún método de defusificación.

Para realizar el entrenamiento de los sistemas neuro-difusos la estructura neuro-difusa puede ser configurada con valores iniciales obtenidos de los conocimientos anteriores, y luego, sintonizados utilizando un algoritmo de entrenamiento tal como Retro-propagación del Error, y se puede hacerlo con el siguiente procedimiento:

1. Presentar una muestra de entrada, y computar la salida correspondiente
2. Computar el error entre la salida y el valor objetivo
3. Se ajustan los pesos de conexión y las funciones de pertenencia
4. Si el error es mayor que la tolerancia, volver al paso 2, si no es así, el entrenamiento ha sido finalizado.

d.4.5 Red ANFIS

Una red ANFIS es una rutina de entrenamiento para sistemas de inferencia neuro-difusa de tipo Sugeno (J.-S. R. Jang, 1993). Esta red aplica una combinación del método de mínimos cuadrados y el método de gradiente descendiente para el entrenamiento de redes backpropagation (Cortés D., 2002, mayo)

El uso de la función ANFIS aplica técnicas de inferencia difusa para modelar sistemas razón por la cual se utilizará este método para realizar el modelo Neuro-difuso del panel solar. Como se observa en sistemas de inferencia difusos, la forma de las funciones de pertenencia depende de parámetros, y precisamente cambiar estos parámetros cambia la forma de las funciones. En lugar de observar a los datos y escoger los parámetros de las funciones de pertenencia, el comando ANFIS permite escoger automáticamente estos parámetros.

Si se supone que se quiere aplicar un sistema de inferencia difuso a un sistema para el cual se tiene una colección de datos de entrada y de salida que se requieren para el modelamiento del sistema, y para el cual no se tiene necesariamente una estructura predeterminada basada en las características de las variables del sistema. En este caso no es posible discernir la forma de las funciones de pertenencia simplemente al observar los datos. Por tanto en lugar de escoger los parámetros asociados con una función de pertenencia dada arbitrariamente, estos parámetros pueden ser escogidos de forma que se tomen en cuenta los tipos de variaciones en los datos. En estos casos, las técnicas de entrenamiento neuro-adaptativo incorporadas en ANFIS son de mucha ayuda.

d.5 CONCLUSIONES DE LA REVISIÓN LITERARIA

- La variable principal que influye en el funcionamiento del panel solar fotovoltaico es la radiación solar.

- La lógica difusa tiene varias ventajas notables con respecto a las técnicas de control tradicional. En primer lugar, nos permite visualizar las situaciones dinámicas y complejas del mundo real en un lenguaje más cercano a la experiencia práctica humana que los modelos matemáticos. Estos nos liberan de la tediosa tarea de implementar la solución de un problema y nos permite concentrarnos más en la solución propiamente dicha. No obstante, la lógica difusa tiene algunas desventajas. Por ejemplo, aunque los sistemas difusos son fáciles de diseñar y rápidos de traducir a prototipos, requieren también una gran labor de simulación y depuración antes de convertirse en productos finales. Además, es difícil desarrollar un modelo que permita evaluar la eficiencia de un sistema difuso debido a que este último trabaja con reglas muy simples.

- En los modelos neuro-difusos existe una relación muy estrecha de Redes Neuronales y Lógica Difusa ya que las dos técnicas pueden atacar al mismo tipo de problemas, pero a las redes neuronales se le puede dificultar la obtención de información de entrada, por lo cual se hace uso de la técnica de Lógica Difusa en la cual, las entradas están en función de las reglas descritas por el diseñador del sistema. Esto indica que puede haber una dependencia de una técnica a otra en ciertas etapas del procesamiento de información ya que pueden ir cambiando los datos en el procesamiento, por cual al realizar un modelo neuro-difuso se requiere de sistemas de desarrollo que relacionen la Lógica Difusa con las Redes Neuronales.

- La red Anfis del Toolboxes de Matlab establece las técnicas de este tipo de entrenamiento que provee un método para el procedimiento de modelamiento

difuso para el aprendizaje de información de un conjunto de datos. ANFIS de cierta forma calcula los parámetros de las funciones de pertenencia que ayudan de mejor manera a que el sistema de inferencia difusa se ajuste a los datos de entrada y de salida dados.

e.- MATERIALES Y MÉTODOS

e.1 Materiales

Los materiales utilizados para el presente proyecto investigativo se detallan a continuación:

e.1.1 Panel solar fotovoltaico Exmork 100P.

El panel solar utilizado se encuentra en la Casa Autosustentable del AEIRNNR como se muestra en la figura 19 y tiene las características mostradas en la tabla 1.

Tabla 1. Características técnicas del panel solar marca Exmork 100W.

Parámetros / Tipo		100P Silicio Poli cristalino
Potencia máxima (watt)	W	100
Tolerancia de potencia		+3% /-3%
Voltaje óptima (Vmp)	V	17.5V
Corriente óptima (Imp)	A	5.71A
Voltaje máxima (Voc)	V	22.0V
Corriente máxima (Isc)	A	6.14A
Dimensiones		1130x670x35mm
Marco (tipo, material y gruesor)		Aluminium anodizado. Alloy 35mm
Voltaje máxima externa permitida		600V
Coeficiente de temperatura de Isc		±0.05%
Coeficiente de temperatura de Voc		-0.33%
Coeficiente de temperatura de P		-0.23%
Coeficiente de temperatura de Imp		+0.08%
Coeficiente de temperatura de Vmp		-0.33%
Resistencia a cargas mecánicas		200kg/m2
Diodos Bypass		2
Eficiencia de conversión		> 15.75%

Nota: Las especificaciones eléctricas indicadas corresponden a condiciones normalizadas de pruebas: 1 KW/m2, masa de aire: 1.5 y células 25°C.

Figura 19. Panel solar utilizado

e.1.2 Equipo de medición inalámbrica de las variables solares que cuenta la UEE (Unidad de Eficiencia Energética) del AEIRNNR.

Este equipo de medición cuenta con 4 sensores como se muestra en la figura 20 que permiten adquirir las 4 variables que intervienen en la generación de la potencia del panel, estas variables a medir son:

- Radiación solar
- Temperatura
- Intensidad
- Voltaje

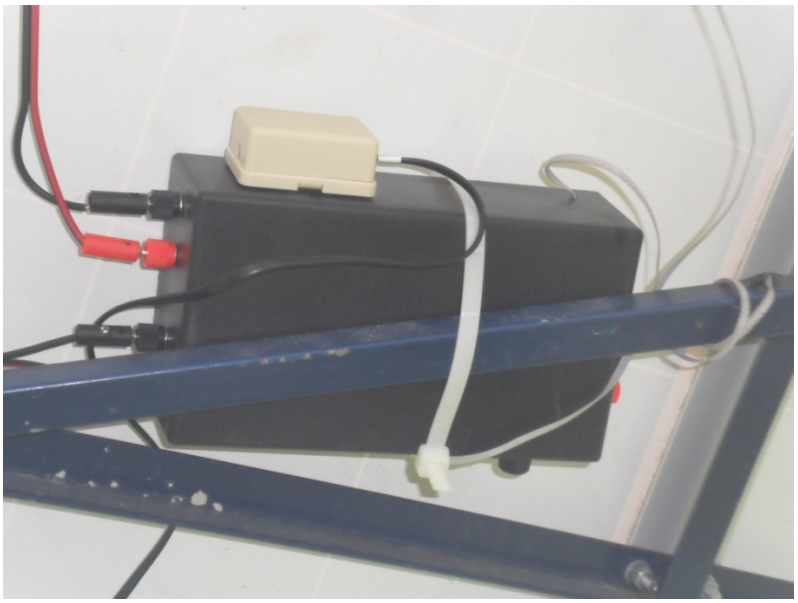


Figura 20. Equipo de medición utilizado montado en el panel solar.

e.1.3 Recursos informáticos

Computadora portátil DELL Inspiron i5, necesaria para receptar los datos que transmite inalámbricamente el equipo de medición, se muestra en la figura 21.

Figura 21. Computadora portátil adquiriendo datos

e.1.4 Herramientas e implementos extras

Herramientas eléctricas para el montaje del equipo de medición.

Extensiones para alimentación de corriente eléctrica al equipo de medición.

Multímetro para calibrar sensores de temperatura, corriente y voltaje.

e.2 Métodos

e.2.1 Técnicas de modelado y recolección de datos

Se empleará la técnica de la observación y la toma de datos utilizando un instrumento electrónico de interface inalámbrica y el software Matlab para el procesamiento, entrenamiento y validación de los modelos generados.

e.2.2.1 Procedimiento De Recolección De Datos

La adquisición de datos se la realizó en un período de una semana desde el 16 al 23 de junio del 2012 para el entrenamiento de los modelos y 5220 mediciones para su validación que se tomaron posteriormente del 23 al 27 de junio del 2012, con un intervalo de muestreo de 1 minuto.

Figura 22. Montaje del equipo de medición para la adquisición de datos

Las variables que se adquirieron fueron la radiación solar (W/m^2), temperatura del panel solar fotovoltaico (°C), intensidad (A) y voltaje (V).

e.2.2.2 Técnicas De Modelado

Se recurrió a dos tipos de técnicas de modelado que son: el modelado con redes neuronales y el modelado con redes neuro-fuzzy (neuro-difusas) empleando el programa Matlab.

Se aplicaron 10079 mediciones para el entrenamiento de los modelos y 5220 mediciones para su validación que se tomaron posteriormente del 23 al 27 de junio del 2012.

f.- RESULTADOS

f.1 Procesamiento de datos

La transferencia de datos del equipo de medición inalámbrico para adquirir las variables en tiempo real de la radiación y temperatura del panel solar; así como el voltaje e intensidad y se lo realizó mediante la interfaz del programa Matlab, los mismos que el usuario del equipo puede guardarlo en una base de datos como archivos .XLS que se muestra en la figura 23 y 24.

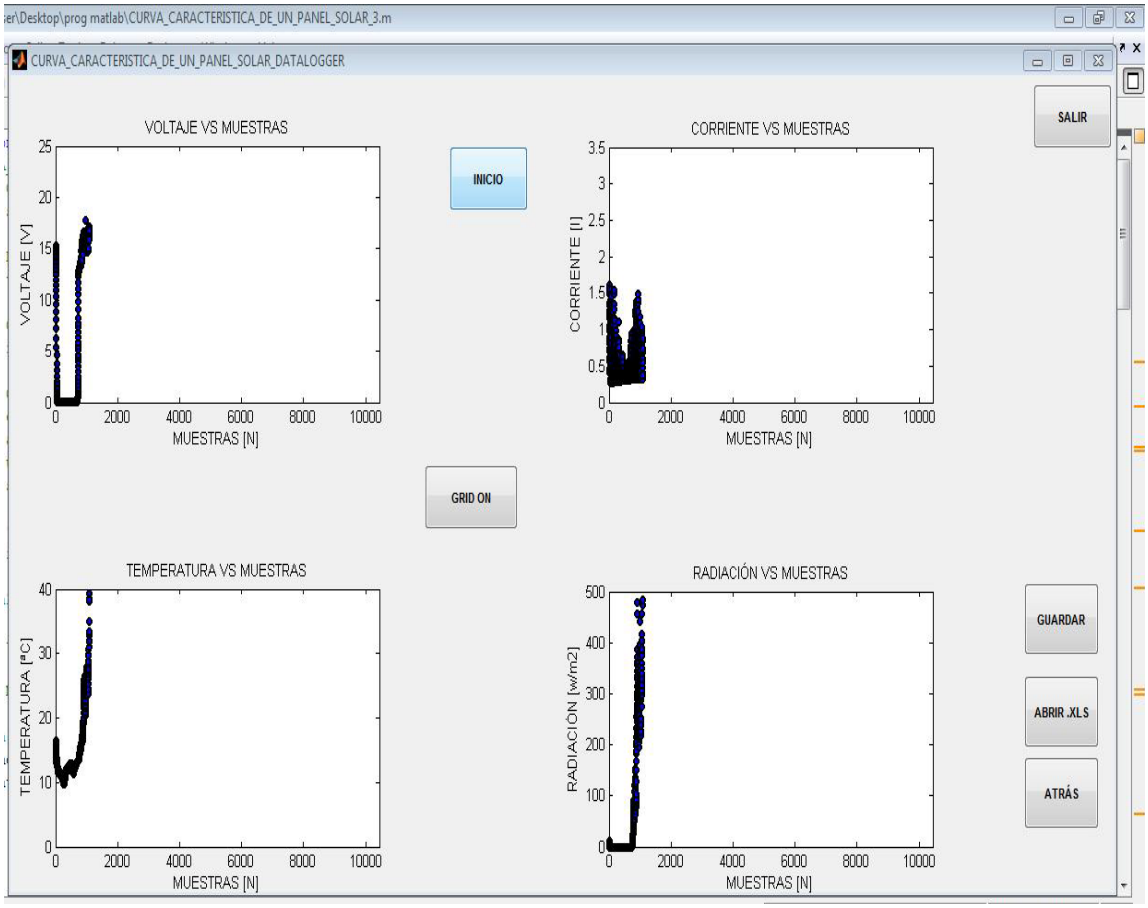
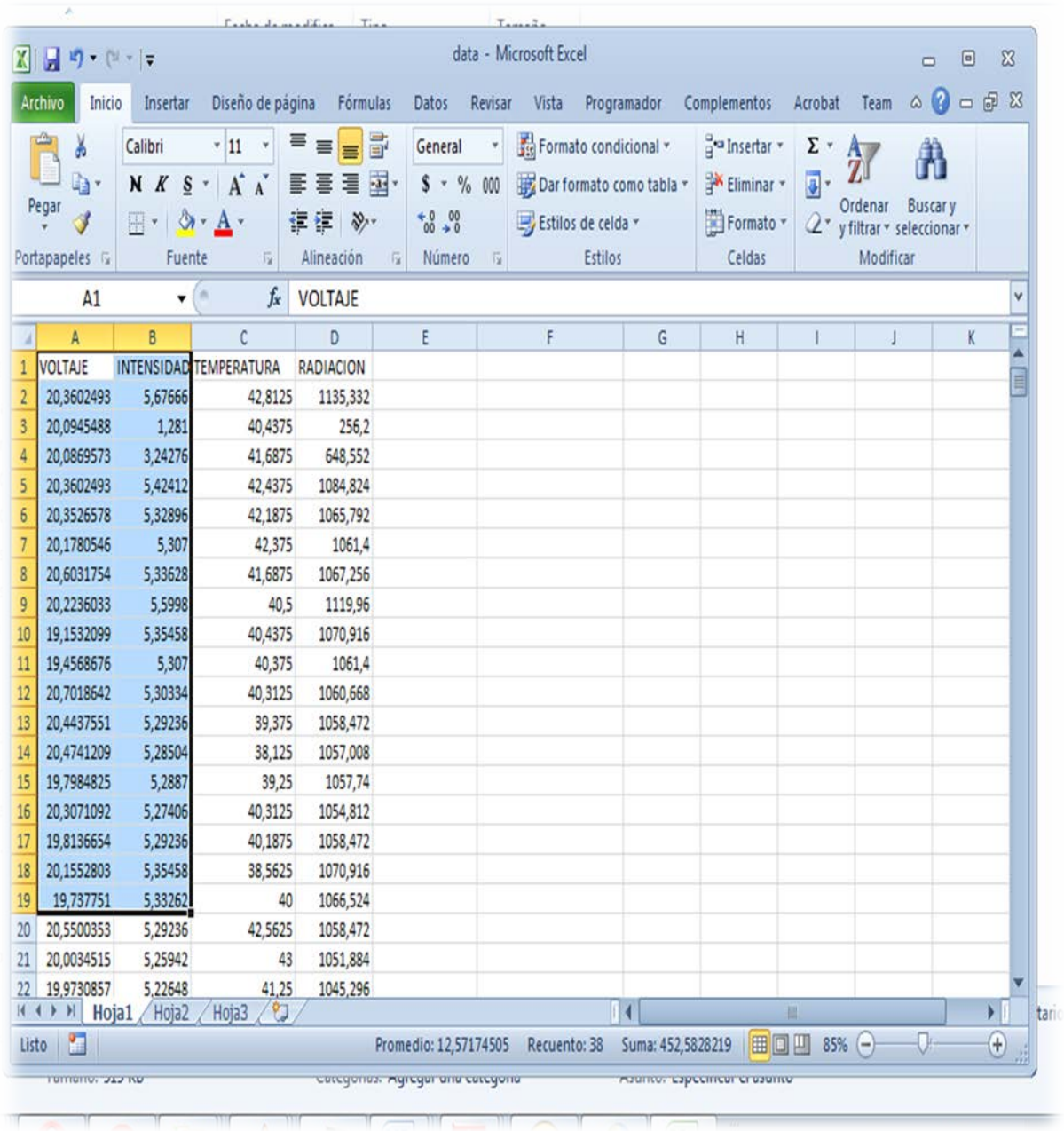


Figura 23. Programa Matlab almacenando datos en la pc



	A	B	C	D
1	VOLTAJE	INTENSIDAD	TEMPERATURA	RADIACION
2	20,3602493	5,67666	42,8125	1135,332
3	20,0945488	1,281	40,4375	256,2
4	20,0869573	3,24276	41,6875	648,552
5	20,3602493	5,42412	42,4375	1084,824
6	20,3526578	5,32896	42,1875	1065,792
7	20,1780546	5,307	42,375	1061,4
8	20,6031754	5,33628	41,6875	1067,256
9	20,2236033	5,5998	40,5	1119,96
10	19,1532099	5,35458	40,4375	1070,916
11	19,4568676	5,307	40,375	1061,4
12	20,7018642	5,30334	40,3125	1060,668
13	20,4437551	5,29236	39,375	1058,472
14	20,4741209	5,28504	38,125	1057,008
15	19,7984825	5,2887	39,25	1057,74
16	20,3071092	5,27406	40,3125	1054,812
17	19,8136654	5,29236	40,1875	1058,472
18	20,1552803	5,35458	38,5625	1070,916
19	19,737751	5,33262	40	1066,524
20	20,5500353	5,29236	42,5625	1058,472
21	20,0034515	5,25942	43	1051,884
22	19,9730857	5,22648	41,25	1045,296

Figura 24. Variables almacenadas en formato .XLS desde Matlab.

Los valores obtenidos en la semana del 16 al 23 de junio del 2012 se muestran en la figura 25.

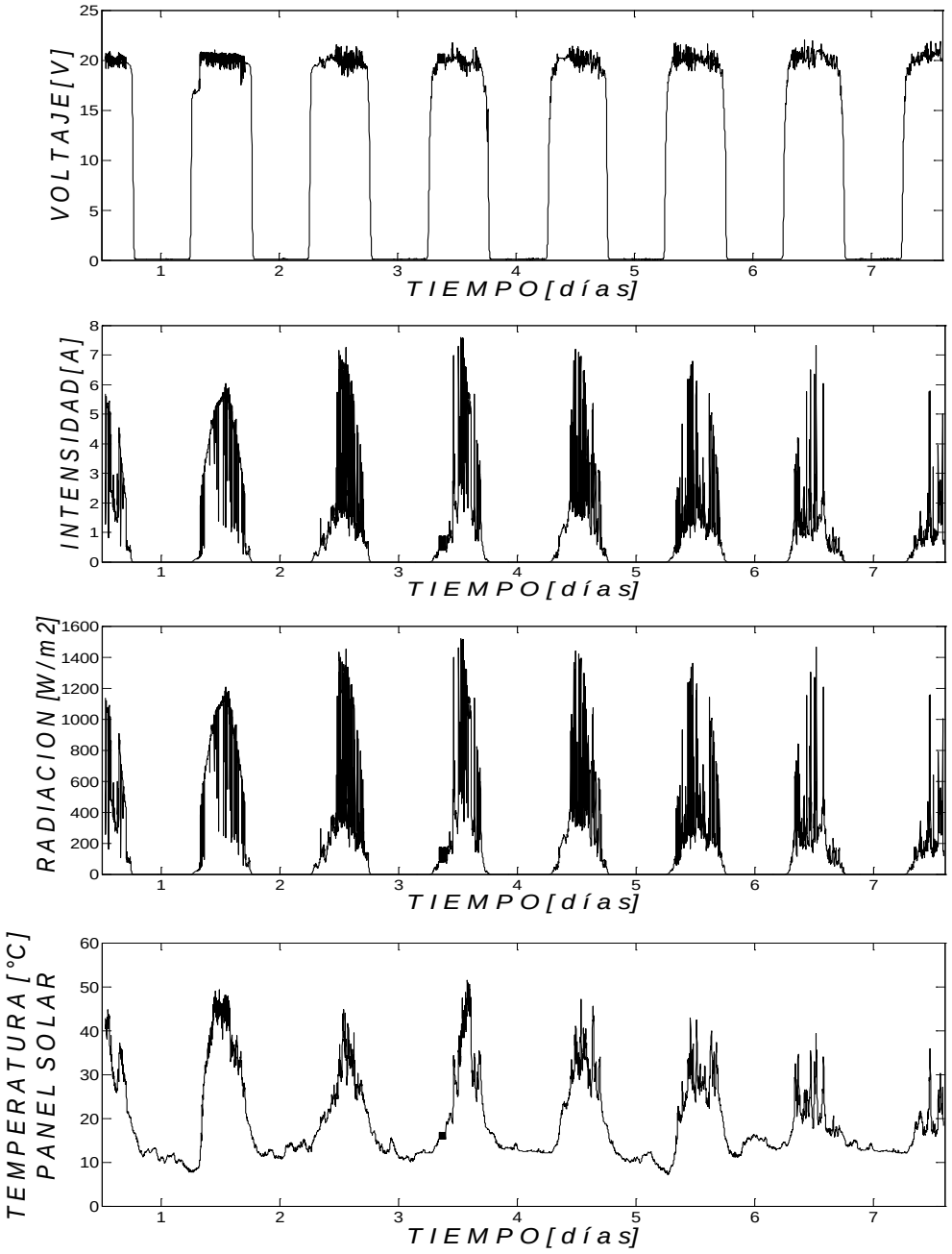


Figura 25. Variables almacenadas durante 7 días.

f.1.1 Importar datos a Matlab

Existen varias formas de importar datos para el presente proyecto investigativo se utilizó el comando **querybuilder** que es una herramienta fácil de usar, esta interfaz gráfica de usuario (GUI) permite el intercambio de datos con su base de datos, la figura 26 muestra la GUI de Visual Query Builder.

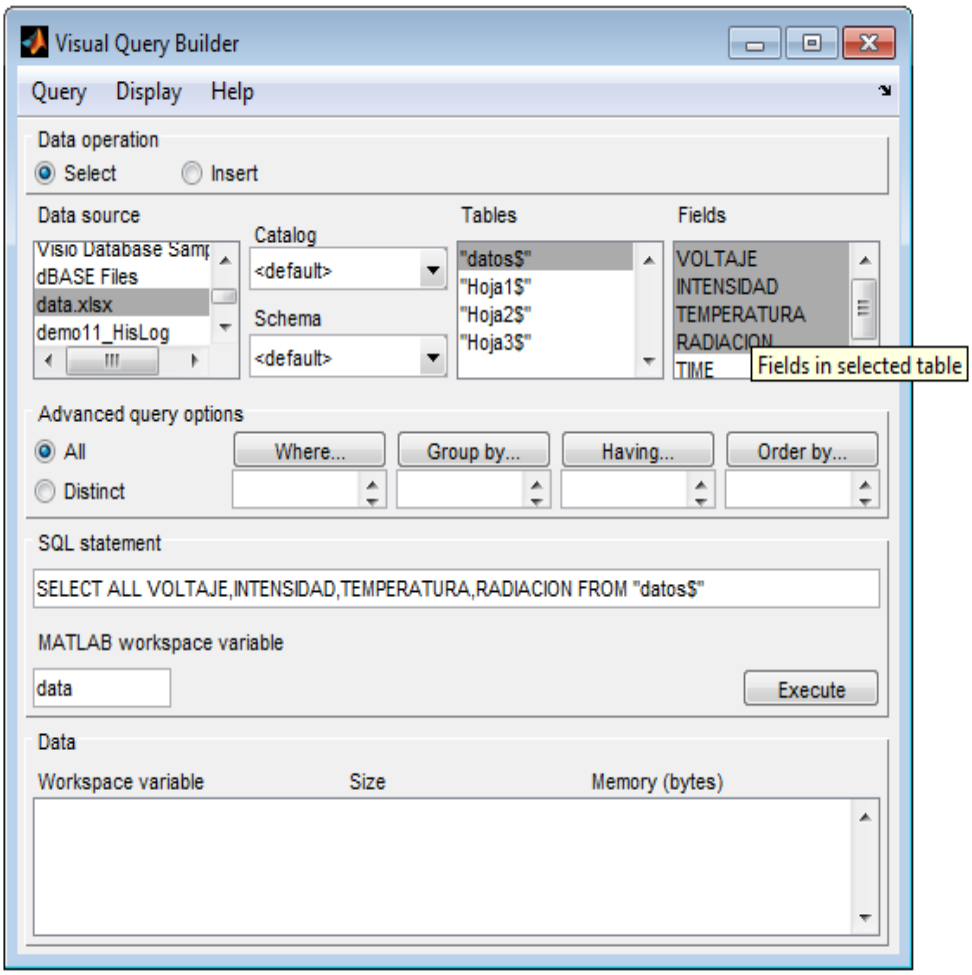


Figura 26. GUI de Visual Query Builder

Para acceder a cada variable en una matriz como estructura podemos importarlos seleccionado el menú: dataset **Query/preferences** y seleccionamos en **database toolbox/data return format/dataset/apply/ok.**

Las matrices de dataset tienen la ventaja que pueden contener diferente clases de variables, incluyendo, carácter numérico, lógico, categórico, y celda.

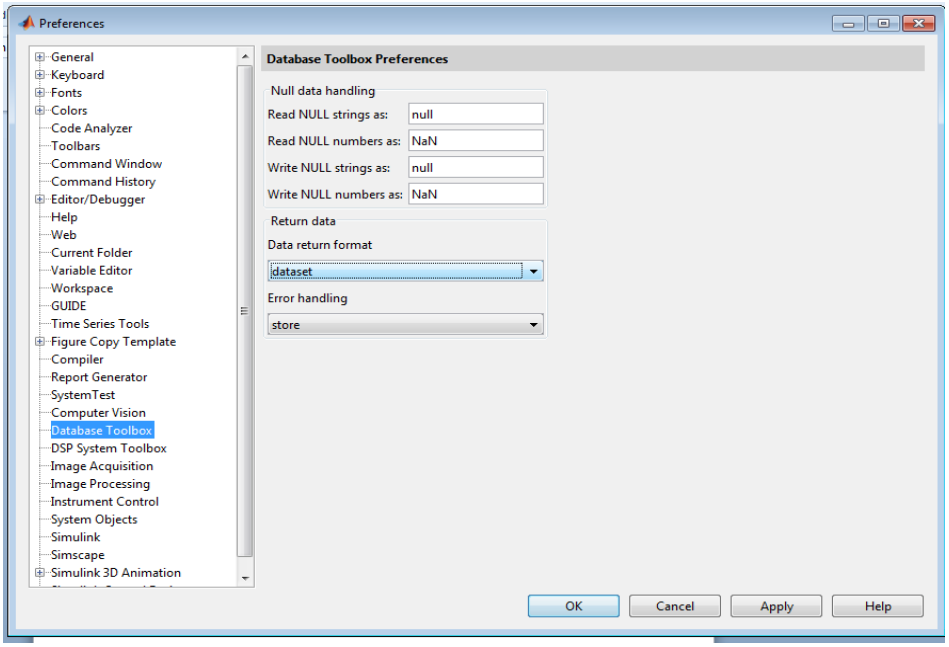


Figura 27. Preferencias de la GUI de Visual Query Builder

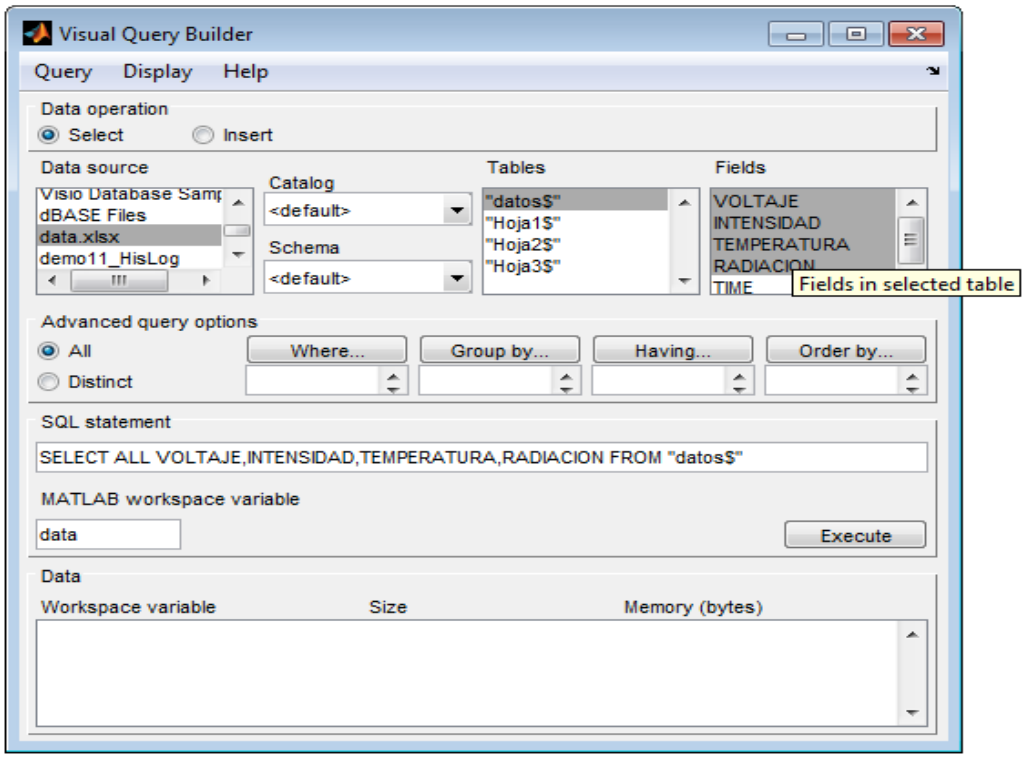


Figura 28. Selección de los parámetros para importar en la GUI Visual Query Builder

Para exportar las variables deseadas de la base de datos seleccionamos Execute y ya tenemos importada nuestra variable **data** al workspace.

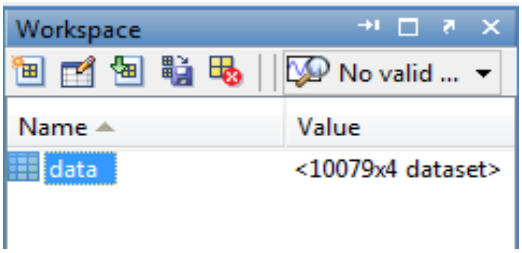


Figura 29. Datos importados al Workspace de Matlab

Posteriormente guardamos este archivo en nuestra carpeta de trabajo y tenemos importados los datos adquiridos en formato .XLS a nuestra carpeta de trabajo en el programa Matlab.

Figura 30. Datos guardados en la carpeta de trabajo en Matlab

f.1.1.2 Creación de archivos de extensión .m

Un script es una secuencia de comandos que se pueden ejecutar a menudo y que se pueden guardar en un archivo de extensión .m para no tener que escribirlos de nuevo. Las demostraciones de MATLAB son un ejemplo de estos scripts. La mayoría de las funciones de MATLAB están en realidad en archivos .m.

En este tipo de archivos se pueden crear, editar y guardar con cualquier editor de textos. Un archivo de script es simplemente una lista de comandos de MATLAB. Cuando se escribe el nombre del archivo en el prompt de MATLAB, su contenido se ejecuta.

Para elaborar nuestro script, generamos un nuevo archivo de texto con el nombre "**METODOLOGÍA.m"** que se muestra en la figura 31.

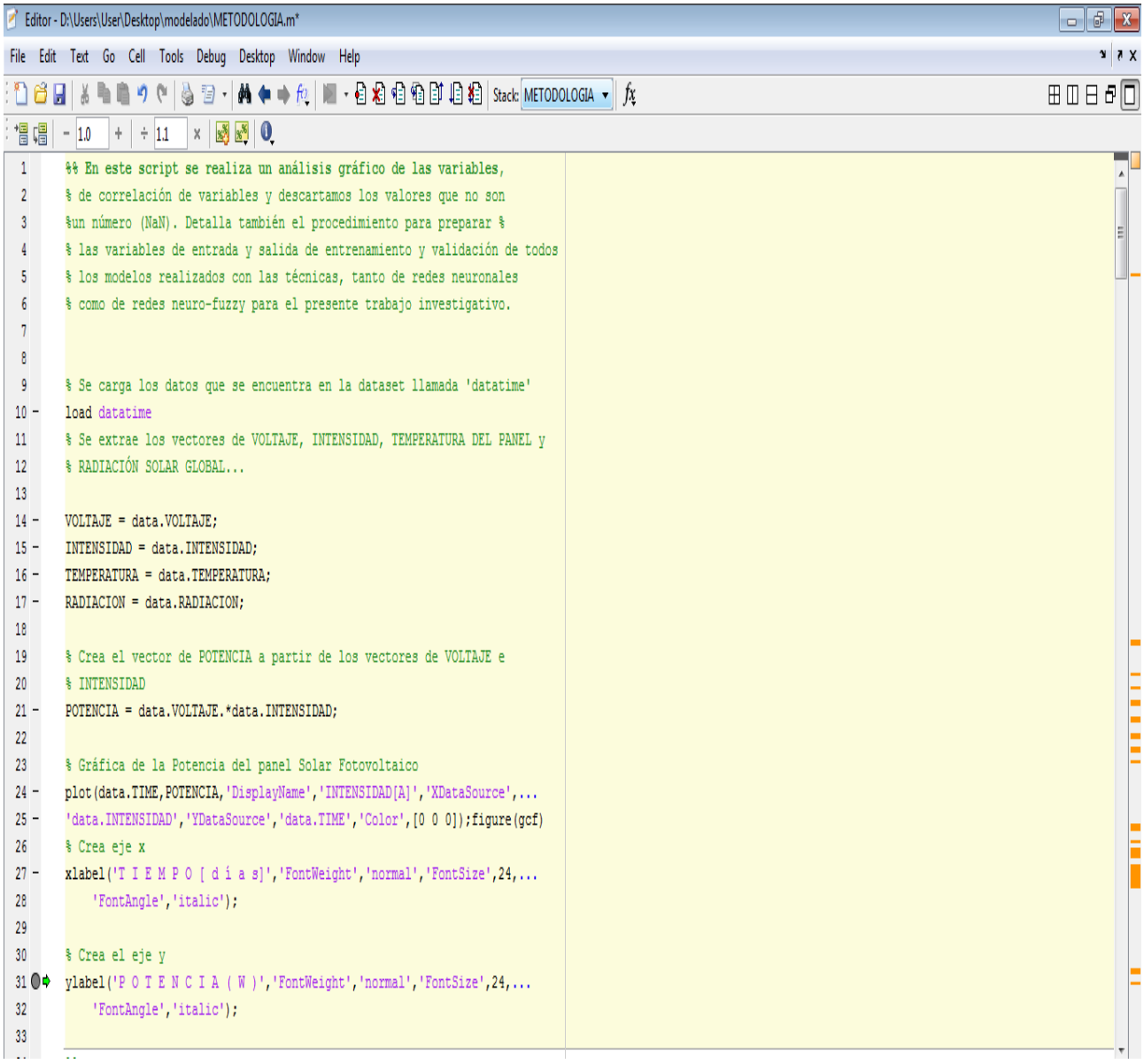


Figura 31. Script METODOLOGÍA.m creado en Matlab.

En este script se realiza un análisis gráfico de las variables, de correlación de variables y descartamos los valores que no son un número (NaN). Detalla también el procedimiento

para preparar las variables de entrada y salida de entrenamiento y validación de todos los modelos realizados con las técnicas, tanto de redes neuronales como de redes neuro-fuzzy para el presente trabajo investigativo.

También se obtiene la variable POTENCIA como se ilustra en la figura 32 a partir del producto de las variables de voltaje e intensidad adquiridos durante los 7 días.

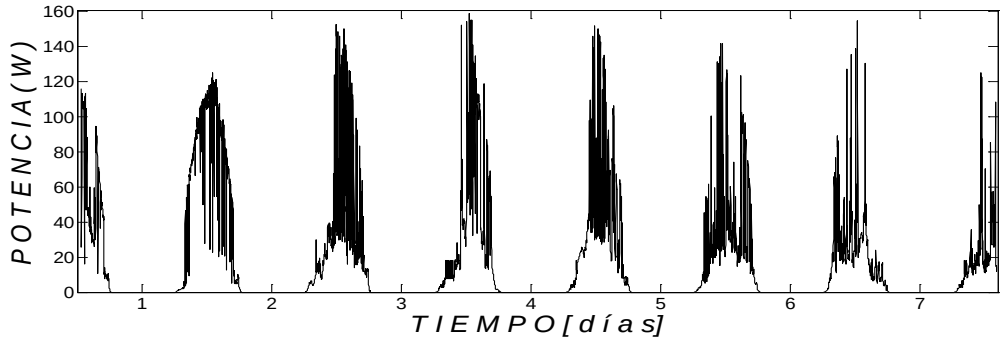


Figura 32. Potencia del panel obtenida a partir de las variables de voltaje e intensidad

f.2. Creación de Modelos.

Como se mencionó anteriormente para realizar nuestro tema de investigación, se utilizó dos técnicas avanzadas para seleccionar y comparar el mejor modelo que describa las variables de salida del panel solar fotovoltaico.

La primera técnica que se utilizó fueron las redes neuronales artificiales mediante la **GUI nntool** y la segunda técnica utilizada son las redes neuro-difusas con la utilización de la **GUI anfisedit,** como se mencionó para aplicar estas técnicas de modelado se utilizará el programa Matlab.

A continuación se detallará detenidamente en primer lugar la metodología para la generación de modelos con redes neuronales y en segundo lugar se detallará el procedimiento para realizar modelos aplicando redes neurodifusas (neurofuzzy).

f.2.1 Modelos que utilizan redes neuronales Feed-Forward Backpropagation

f.2.1.1 ESTRUCTURA DE LOS DATOS

Para el entrenamiento con redes neuronales debemos tomar en cuenta que las variables de entrada como de salida deber estar dispuestos en forma de columnas como se muestra en la figura 33.

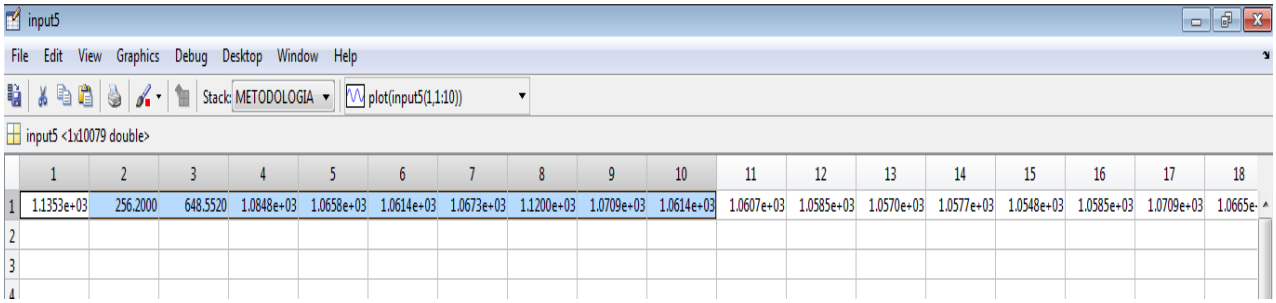


Figura 33. Pantalla que muestra que deben estar los datos en una red

Las variables de entrada (input) y las variables de salida deseadas (target) utilizados para generar varios modelos en este trabajo investigativo se muestran en la siguiente tabla:

Tabla 2. Variables de entrada y salida utilizadas en los modelos con redes neuronales

INPUT	TARGET
RADIACION TEMPERATURA	VOLTAJE INTENSIDAD
RADIACION TEMPERATURA	VOLTAJE
RADIACION TEMPERATURA	INTENSIDAD
RADIACION	INTENSIDAD
RADIACION	VOLTAJE
RADIACION	POTENCIA
RADIACION	TEMPERATURA
RADIACION TEMPERATURA	POTENCIA

f.2.1.1.2 Metodología para la creación de una RNA tipo BP (Backpropagation) mediante la GUI nntool

La GUI nntool permite crear RNA´s tipo BP con distintas arquitecturas. La arquitectura de la red BP más empleada es la RNA tipo BP de capas múltiples con prealimentación (Feed-Forward Backpropagation), este tipo de red es empleada en esta tesis para desarrollar los modelos para el panel solar fotovoltaico. Para crear una RNA tipo BP de capas múltiples con prealimentación es necesario seleccionar la opción Feed-forward backprop del menú **Network Type** en la ventana **Create New Network**. Al realizar esta acción se despliega la ventana mostrada en la figura 34 la cual muestra los parámetros a introducir con el fin de personalizar la RNA.

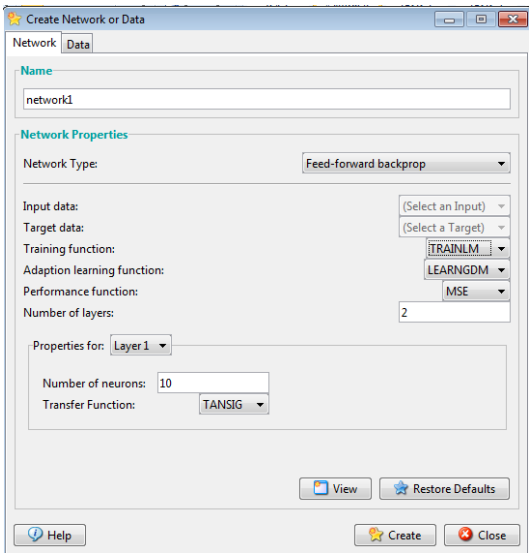


Figura 34. Ventana desplegada para crear una RNA tipo BP de prealimentación.

El primer dato a introducir con el fin de personalizar la RNA BP es el nombre que se le asignara, esto dentro del espacio Network Name:, Matlab le asigna un nombre por omisión (ejemplo: network1). El área **Network Type:** además de mostrar un menú para la selección de la arquitectura, cuenta con sub-menús que permiten personalizar la arquitectura seleccionada. El primero de ellos, **Input ranges**: , permite definir los limites de los valores en las señales de entrada. Estos valores pueden ser asignados automáticamente por Matlab, ubicando los valores máximo y mínimo de un vector previamente declarado como de entradas en la ventana Network/Data Manager. La dimensión del vector de entradas debe ser: dos columnas, una para el limite inferior y

otra para el limite superior, y la misma cantidad de renglones como de entradas que se hallan declarado en el espacio **Number of layers:**.

El número de nodos en la capa de entrada será la cantidad introducida en el espacio **Number of layers:**.

Durante el entrenamiento de una RNA BP los pesos y sesgos son ajustados de forma interactiva de tal forma que se minimice la función de ejecución de la RNA. La función de ejecución por omisión para la red con prealimentación es la del error medio al cuadrado (MSE por sus siglas en ingles), esto es: el error cuadrado promedio entre las salidas de la red y las salidas deseadas (ver figura 35 en la opción Performance function:). Otras opciones de funciones de ejecución son MSEREG y SSE. Esta opción determina la forma en la que los pesos de la RNA son ajustados.

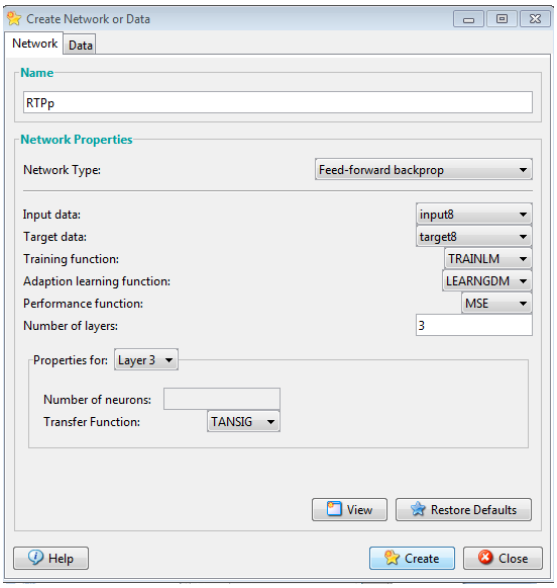


Figura 35. RNA BP personalizada

En el entrenamiento de una RNA BP el formato de las entradas puede ser como un solo vector o una matriz que agrupe un conjunto de vectores de entrada en el cual cada elemento de entrada ocupa un renglón (conocido como simulación por lotes).

Para entrenar una RNA BP se requiere de vectores que contengan las entradas a la RNA y los valores correctos de las salidas correspondientes.

Al diseñar una BP se puede seleccionar la función de transferencia a emplear tanto en la capa oculta como en la de salida.

La selección de función de transferencia a utilizar, se realiza dentro del espacio Propiedades de capas (Properties for:) en el menú **Transfer Function:** (ver figura 35). las posibles opciones son tres: Función Lineal (PURELIN), Tangente Sigmoidea (TANSIG) y Sigmoidea Logarítmica (LOGSIG).

En el menú **Properties for**: la capa **Layer 1** representa la capa oculta y **Layer 2** la capa de salida de la RNA BP creada. De la misma forma se puede definir el número de neuronas en la capa seleccionada dentro del espacio en blanco Number of neurons:.

El algoritmo a emplear para la etapa de entrenamiento se selecciona en el menú **Training function**: (ver figura 36), la RNA BP estándar es la que empléa una generalización de la Regla de Aprendizaje de Widrow-Hoff (en este caso la opción TRAINLM) la cual utiliza el ALGORITMO de Levenberg-Marquardt, que emplea modificaciones para disminuir el uso de memoria.

Existen siete parámetros de entrenamiento asociados con el entrenamiento de gradiente descendente, ver figura 36.

La opción **show** determina la cantidad de iteraciones del algoritmo antes de desplegar su estado actual (se le puede asignar el valor NaN para anular dicho despliegue).

La opción **epochs** determina la cantidad de iteraciones a realizar antes de parar el entrenamiento.

El entrenamiento podría parar antes de realizar la cantidad de iteraciones determinadas en epochs si el valor de la función de ejecución toma un valor menor al del parámetro goal, si el valor del gradiente es menor al valor introducido en **mingrad**, o si el tiempo de entrenamiento es mayor que el consignado en el espacio time (en segundos).

El parámetro **max_fail** esta relacionado con técnicas de optimización del paro de entrenamiento.

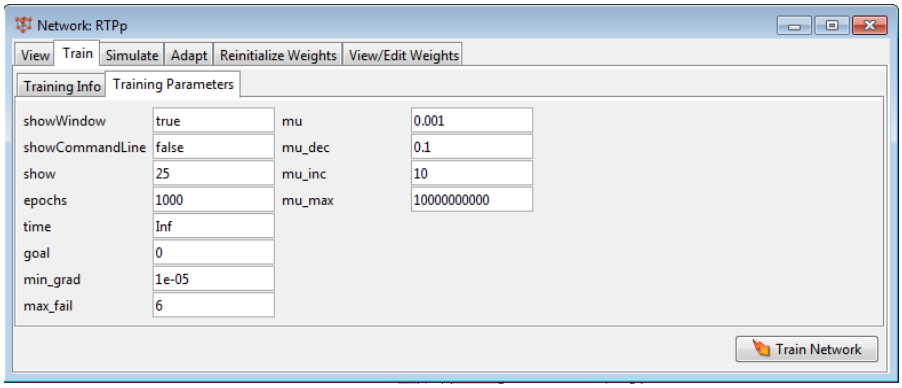


Figura 36. Ventana Training Parameters en la opción Train.

Después de haber personalizado la RNA creada se puede seleccionar la opción **Create** para generar en la ventana **Network/Data Manager** el objeto que emula la RNA BP con los parámetros introducidos. Estando lista para inicializar sus pesos (**Initialize**), entrenarse (**Train**), simularse (**Simulate**) ó actualizarse (**Adapt**). En la tabla 3 se muestran los algoritmos de entrenamiento disponible para una RNA BP.

Tabla 3. Funciones de entrenamiento disponibles en Matlab para una RNA BP

Función Matlab	**Descripción**
TRAINGD	Los pesos y sesgos de la RNA se actualizan en la dirección del gradiente negativo de la función de ejecución. Respuesta lenta.
TRAINGDM	Los pesos y sesgos de la RNA se actualizan en la dirección del gradiente negativo de la función de ejecución. Se emplea un nuevo parámetro (momento) que permite esquivar mínimos locales. De esta forma la actualización es función del último cambio realizado y del gradiente.
TRAINGDA	Se emplea un factor de entrenamiento variable. El factor de entrenamiento se establece en base a las magnitudes de error obtenidas en la iteración actual y anterior. De igual forma los cambios en pesos y sesgos son función sus valores actuales y anteriores.
TRAINGDX	Es similar al entrenamiento TRAINGDA con la diferencia que en este método se emplea además un parámetro de momento.

TRAINR	Actualización incremental de orden aleatorio.
TRAINRP	Retro propagación con Resiliencia. Emplea el signo de la derivada de la función de ejecución para determinar el cambio en los pesos y sesgos, la magnitud de la derivada no se considera.
TRAINCGF	En los algoritmos de gradiente descendente conjugado se realiza una búsqueda a lo largo de direcciones conjugadas de decremento del gradiente para determinar la magnitud del cambio en los pesos y sesgos. En este caso se emplea la actualización mediante el método Fletcher-Reeves, en el cual se determina la dirección del cambio en base a la dirección de búsqueda de descenso de gradiente actual con la anterior.
TRAINCGP	Método de gradiente descendente conjugado con el algoritmo Polak-Ribiére
TRAINCGB	Método de gradiente descendente conjugado empleando el método de restablecimiento de dirección de búsqueda Powell-Beale.
TRAINSCG	Emplea un gradiente conjugado escalado. Es el único algoritmo de gradiente conjugado que no requiere una línea de búsqueda.
TRAINBFG	Emplea métodos de Newton para el cálculo de las actualizaciones. Requiere almacenar una aproximación de la matriz Hessiana
TRAINOSS	Método de la secante de un paso. Es una combinación de los métodos de gradiente conjugado y Newton.
TRAINLM	Algoritmo de Levenberg-Marquardt. Emplea modificaciones para disminuir el uso de memoria.
TRAINBR	Regulación Bayesiana. Modificación al algoritmo Levenberg-Marquardt.

f.2.1.1.2.1 ENTRENAMIENTO DE UNA RNA

Para entrenar la RNA creada, primero se debe seleccionar dicha RNA en la ventana Network /Data Manager. Posteriormente se presiona el botón Train. Al hacer esto se activa una nueva ventana con el título Network: Nombre de la RNA, vea la figura 38.

Dicha ventana permite observar datos concernientes al estado de la RNA, tal como valores de inicialización, parámetros de simulación, y su arquitectura. La tecla **Train** contiene opciones que permiten definir los datos de entrada y salida mediante la ventana **Training Info**, figura 37. En dicha ventana se puede observar en el área de Training Results que el archivo con los datos de salida y errores tiene incluido al inicio el nombre de la RNA. Esto permite identificarlos fácilmente si se desea exportarlos al espacio de trabajo de Matlab.

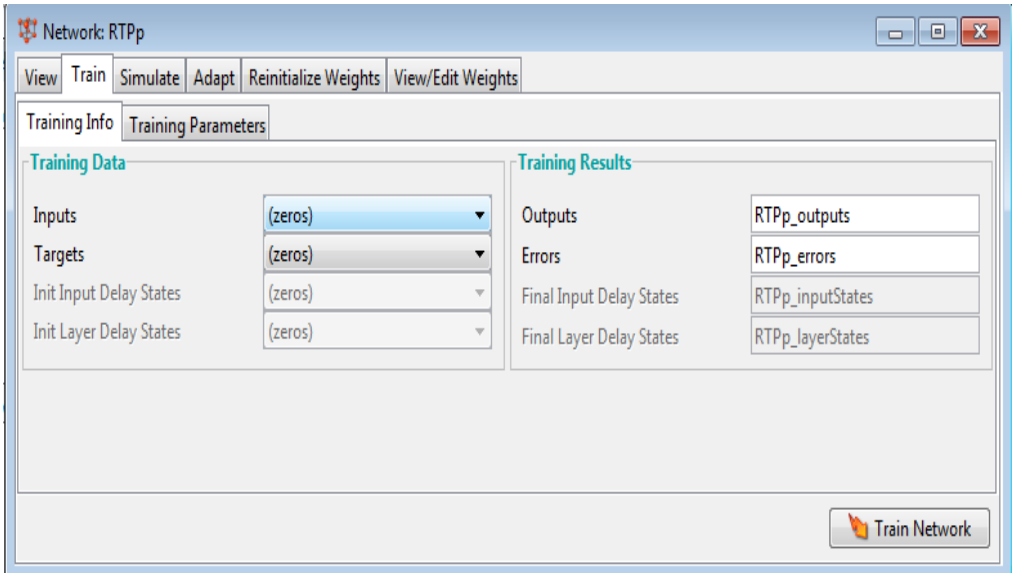


Figura 37. Ventana con información del proceso de entrenamiento en nntool.

Para definir los parámetros que regirán durante el entrenamiento, tal como número de épocas, y error final es necesario dirigirse a la ventana **Training Parameters** que cuenta con apartados que permiten definir dichos valores, figura 31. Después de haber definido los parámetros adecuados es posible activar el botón **Train Network** el cual iniciara la ejecución de dicha tarea. Para verificar que una RNA ha sido entrenada adecuadamente es necesario retornar a la ventana **Network/ Data Manager** y

seleccionar la opción **Simulate** del apartado Networks **Only**, dicha opción activara la ventana de la RNA y en ese momento se selecciona la opción **Simulate**, en la ventana desplegada por esta opción se puede modificar el nombre del archivo que almacena los datos de salida, así como los valores de entrada y su origen, figura 38.

Después de hacer las modificaciones pertinentes se presiona el botón **Simulate network** ubicado en la parte inferior derecha.

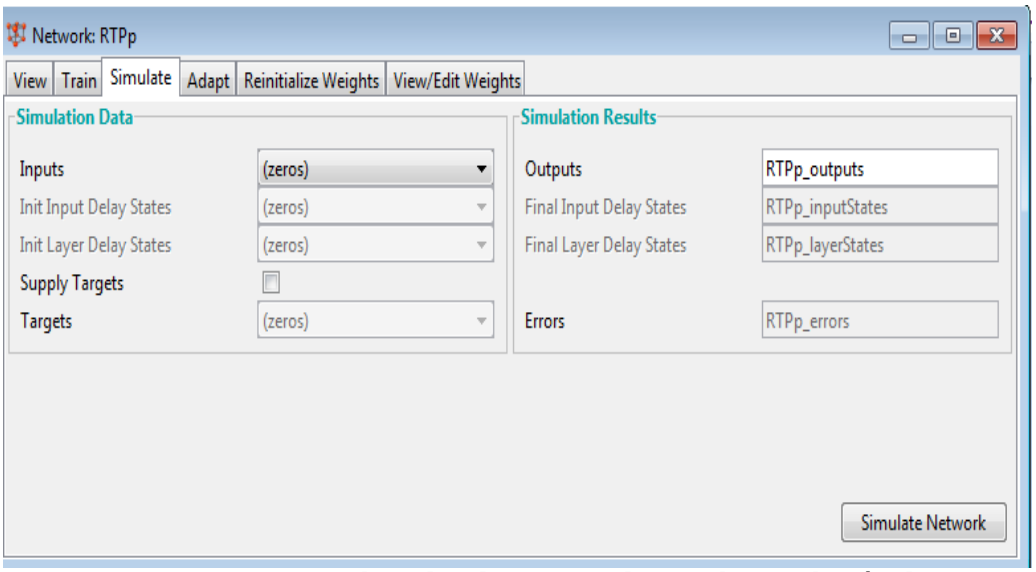


Figura 38. Ventana para introducir los datos a emplear en la simulación de una RNA

f.2.1.1.2.2 Exportando datos a la ventana de trabajo de Matlab

Para exportar los datos generados por una RNA con destino el espacio de trabajo de Matlab es necesario retornar a la pantalla Network / Data Manager, figura 39. Al estar en la ventana Network / Data Manager se puede seleccionar los datos que se desea sean exportados al espacio de trabajo de Matlab y seleccionar el botón ***Export*** lo cual dará las opciones ***Export*** o ***Save from Network / Data Manager***, figura 40. Si solo se desea exportar los datos al espacio de trabajo de Matlab se selecciona la opción ***Export***. De otra manera la segunda opción ***Save*** almacenará los datos en un archivo MAT.

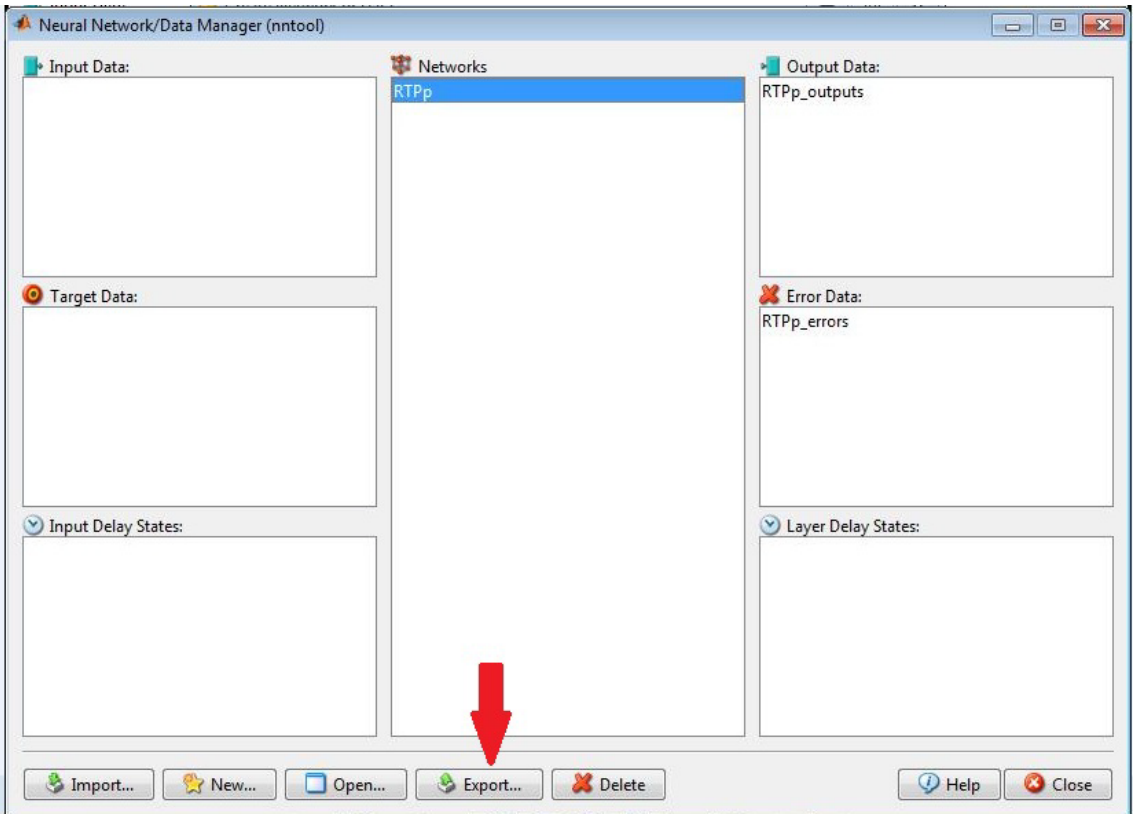


Figura 39. Pantalla Network / Data Manager

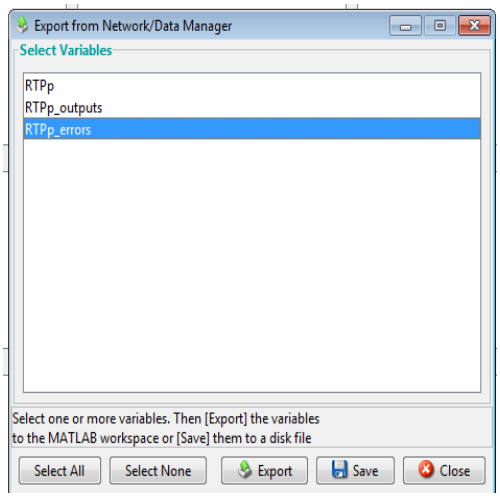


Figura 40. Ventana para exportar datos desde nntool.

f.2.2 Resultados de los modelos generados con redes neuronales.

Todos los modelos generados con la GUI nntool tienen las siguientes características:

Tipo de Red Neuronal: FEED FORWARD BACKPROPAGATION
Función de Entrenamiento: TRAINLM
Función de Aprendizaje: LEARNGDM

Función de trasferencia. Las opciones utilizadas son tres: Función Lineal (PURELIN), Tangente Sigmoidea (TANSIG) y Sigmoidea Logarítmica (LOGSIG).

Como se especificó anteriormente se utilizó un conjunto de 10079 dados para el entrenamiento y otro conjunto de datos de 5720 muestras para validar los modelos.

Solamente se mostrará los gráficos de los mejores modelos generados y a final se presentará en una tabla los resultados de todos los modelos.

f.2.2.1 Validación Del Modelo

Se utiliza la función ***sim.m*** que permite obtener los valores de salida de una red neuronal y la función ***mse.m*** para determinar el error del modelo, los comandos se muestran en la figura 41.

```
%% MODELO 4
load RPT

y4 = sim(RTP,W);
ep = Ppp'-y4;
perf4 = mse(ep);
plot (y4,'DisplayName','estimada','XDataSource','estimada','YDataSource','data.TIME','Color',[0 0 0])
% Create xlabel

xlabel('MUESTRAS','FontWeight','normal','FontSize',16,...
    'FontAngle','italic');

% Create ylabel
ylabel('POTENCIA','FontWeight','normal','FontSize',16,...
    'FontAngle','italic');
hold all
plot(Ppp')
```

Figura 41. Comandos utilizados para la validación y simulación del modelo con redes neuronales.

f.2.2.1.1 Modelos realizados con redes neuronales con su respectivo rendimiento:

1) Modelo REDVI

Variables utilizadas en el entrenamiento:

INPUT**: RADIACION, TEMPERATURA**

TARGET**: VOLTAJE, INTENSIDAD**

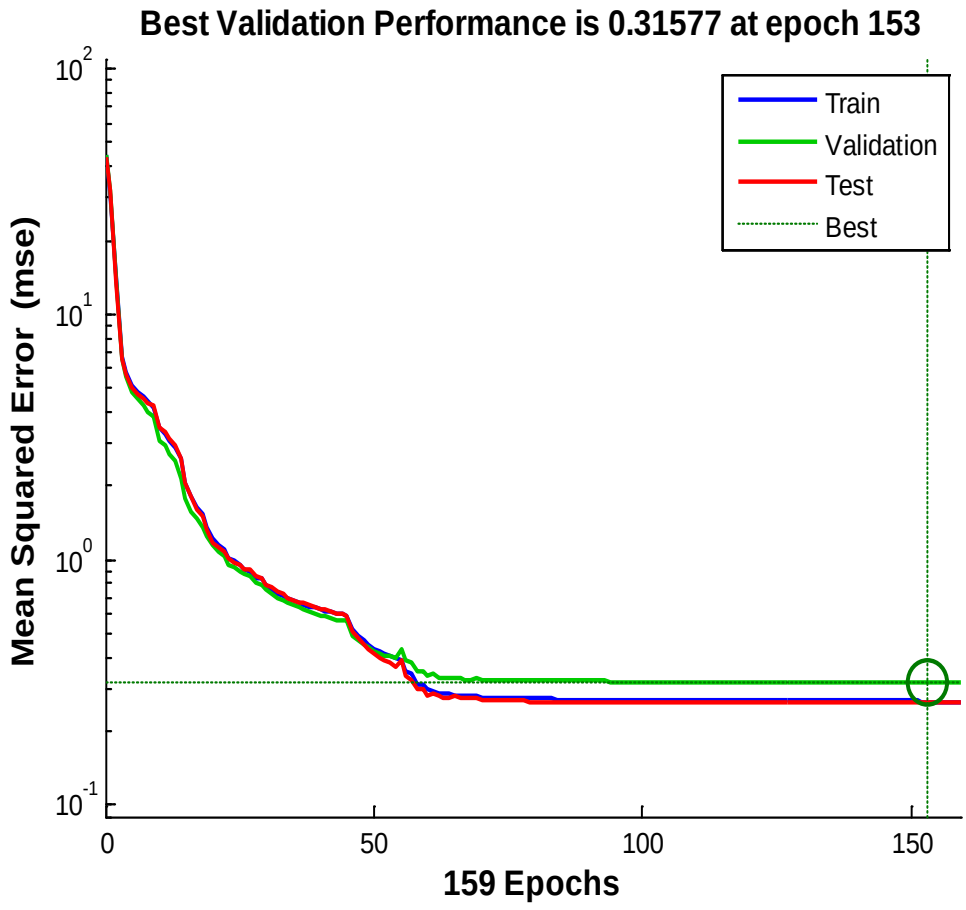


Figura 42. Rendimiento del modelo REDVI

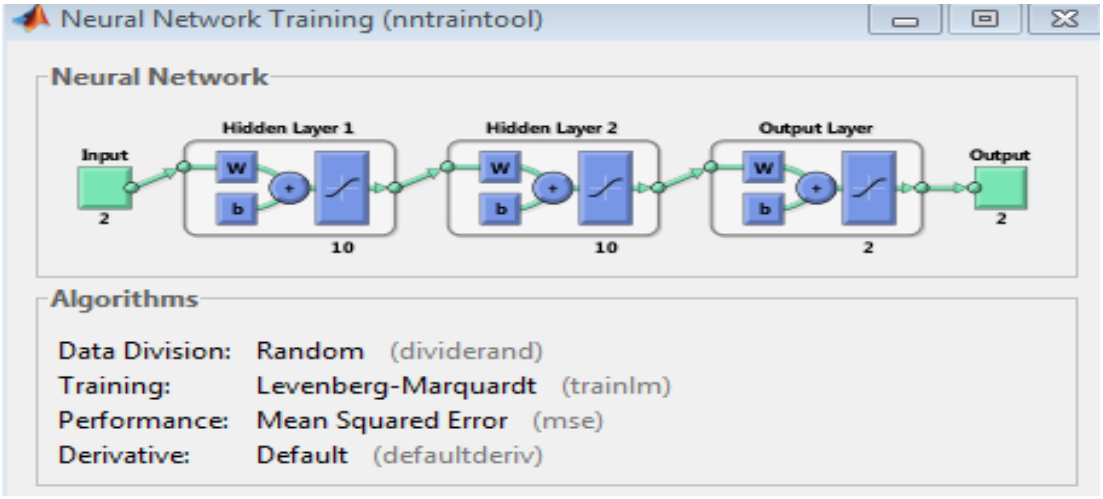


Figura 43. Estructura del modelo

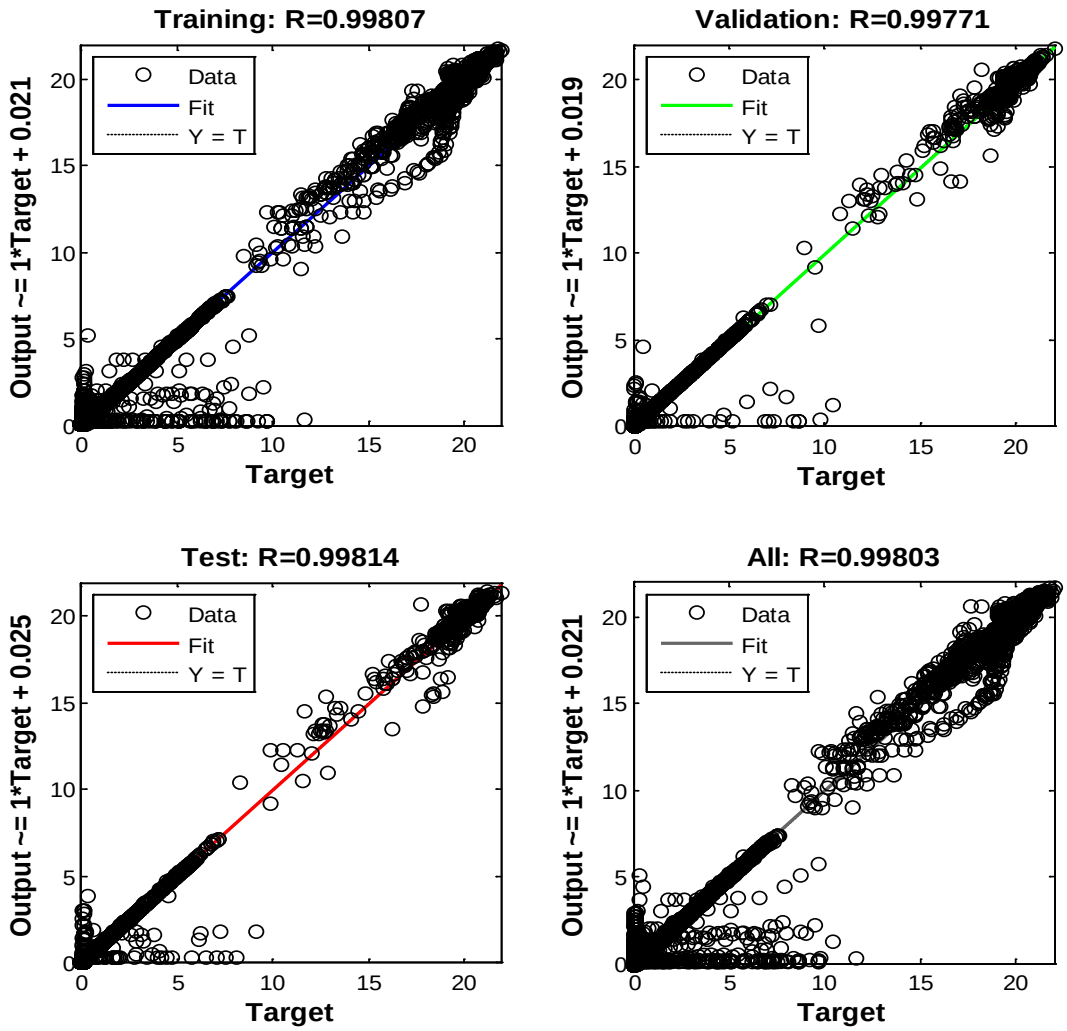


Figura 44. Gráfica de regresión de la red

Validación y simulación de los datos del modelo REDVI

Este modelo tiene dos salidas que se evalúa, una voltaje y otra de intensidad por ello se muestra en las figuras 48 y 49.

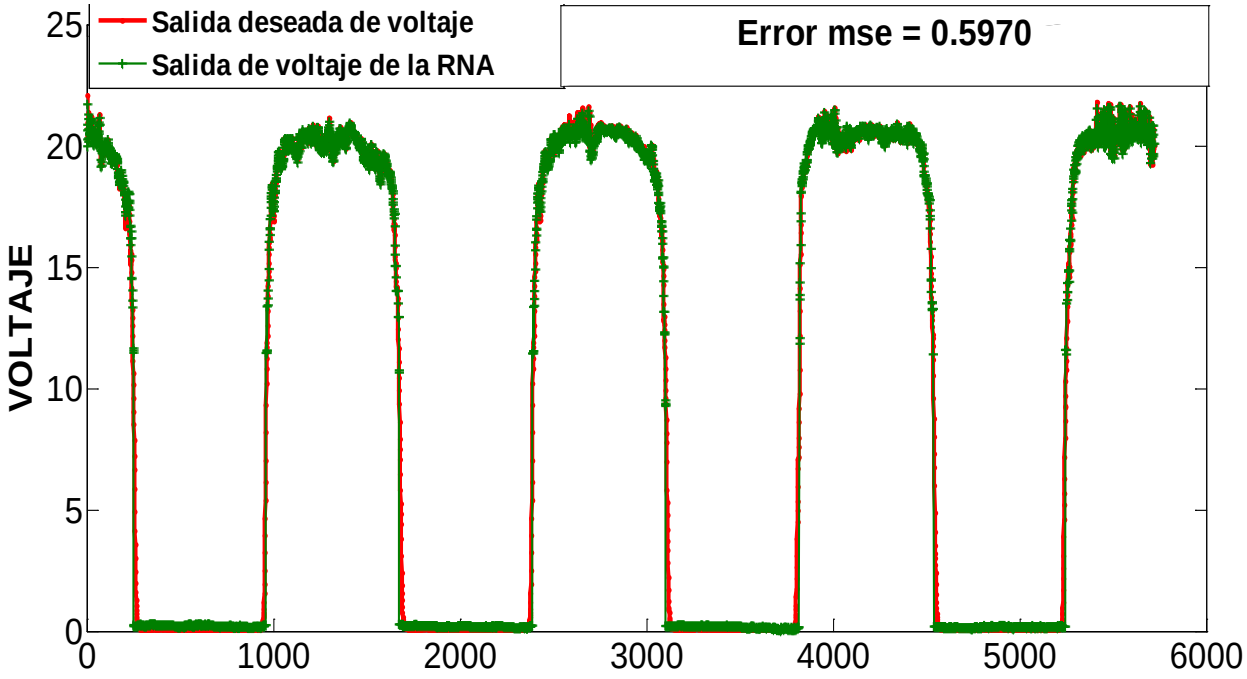


Figura 45. Variable de Voltaje proyectado por la red

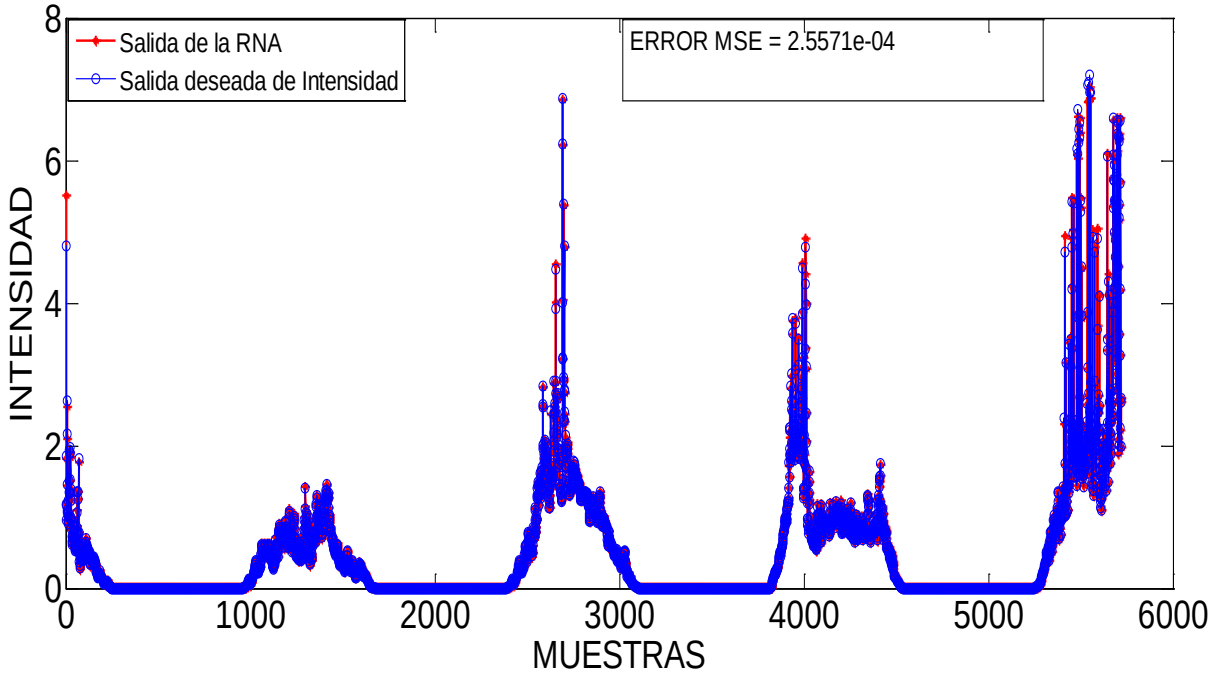


Figura 46.Variable de Intensidad proyectado por la red

2) Modelo REDV3

Variables utilizadas en el entrenamiento:

INPUT: RADIACION, TEMPERATURA

TARGET: VOLTAJE

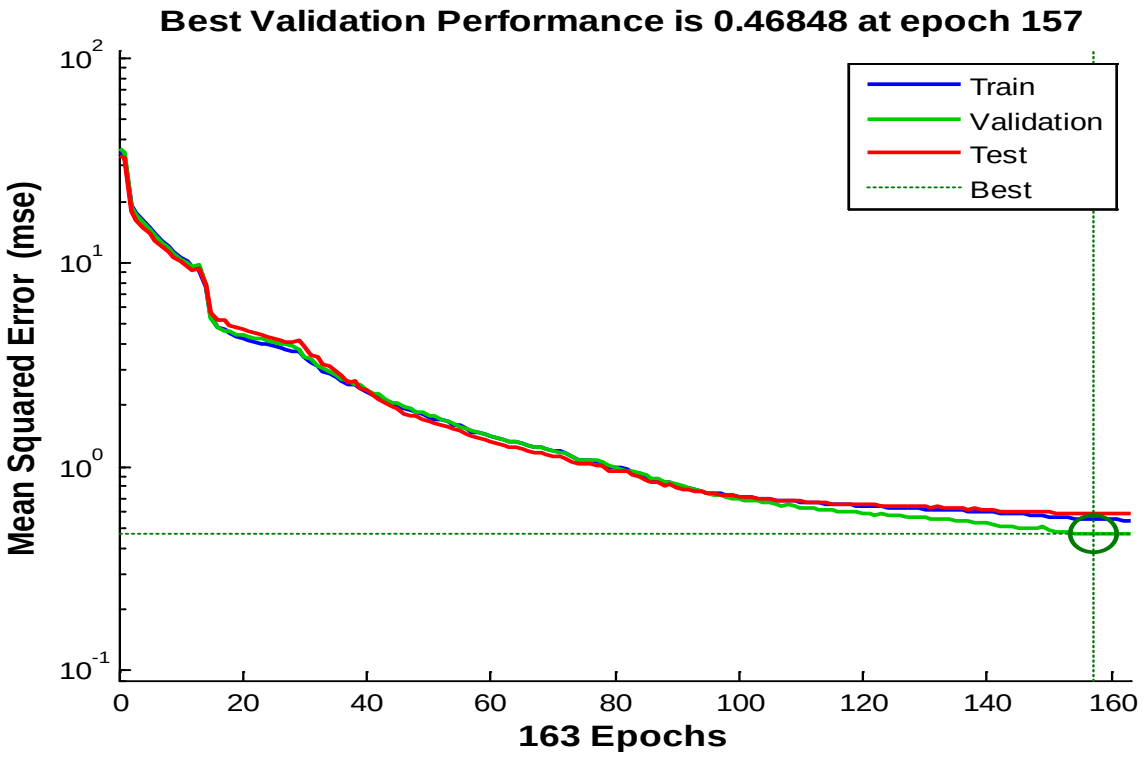


Figura 47. Rendimiento del modelo REDVI

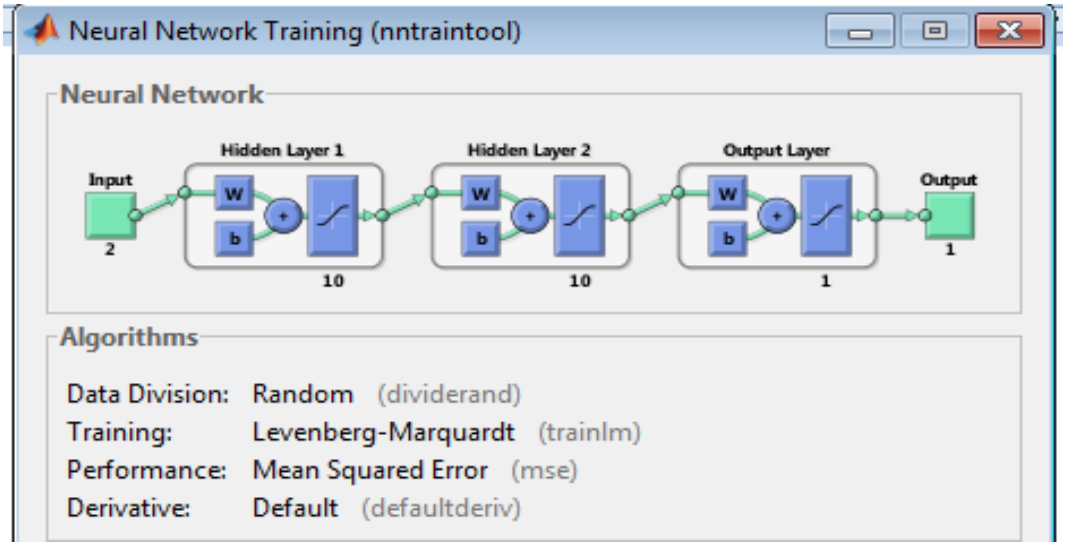


Figura 48. Estructura del modelo

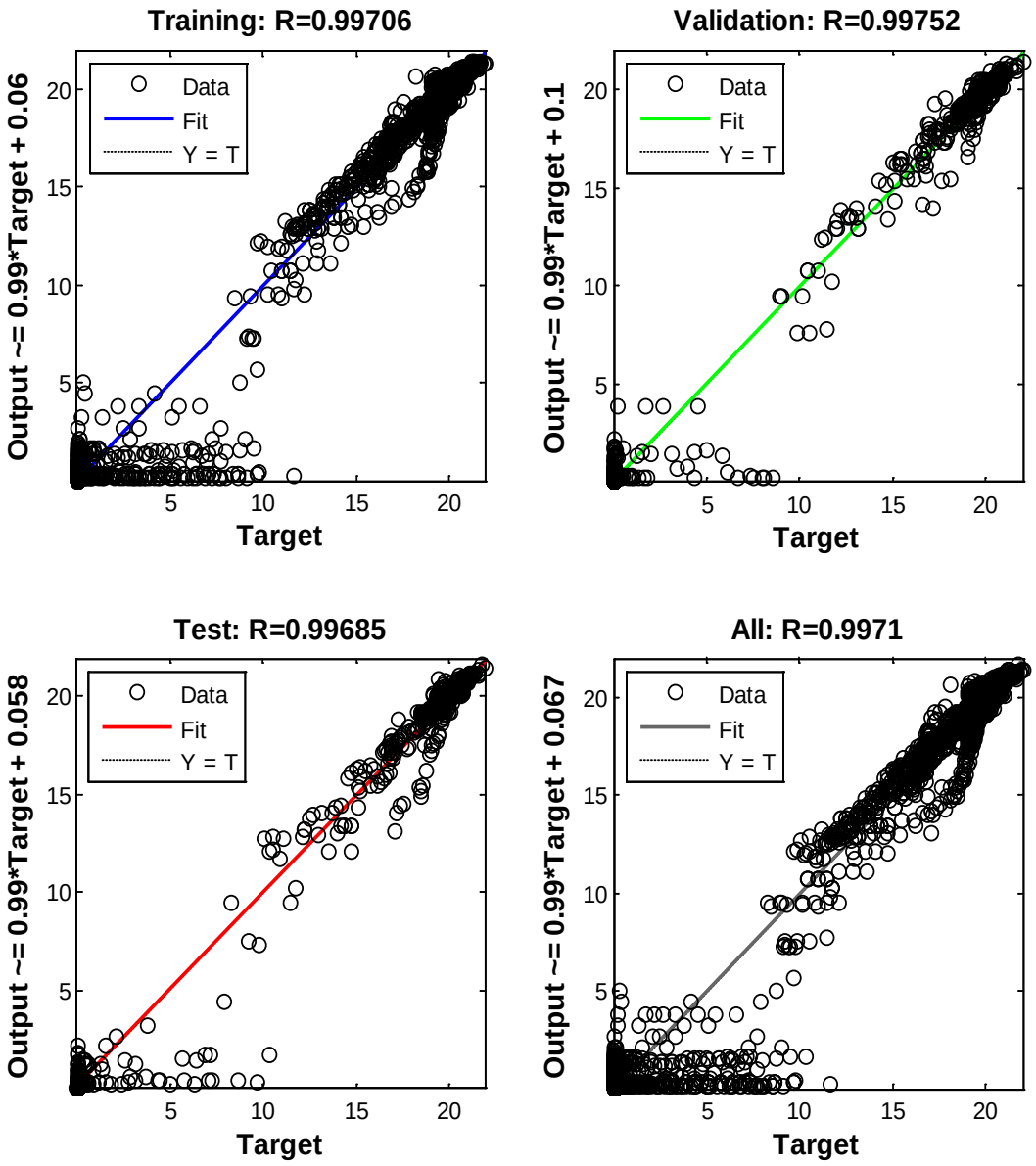


Figura 49. Gráfica de regresión de la red

Validación del modelo REDV3

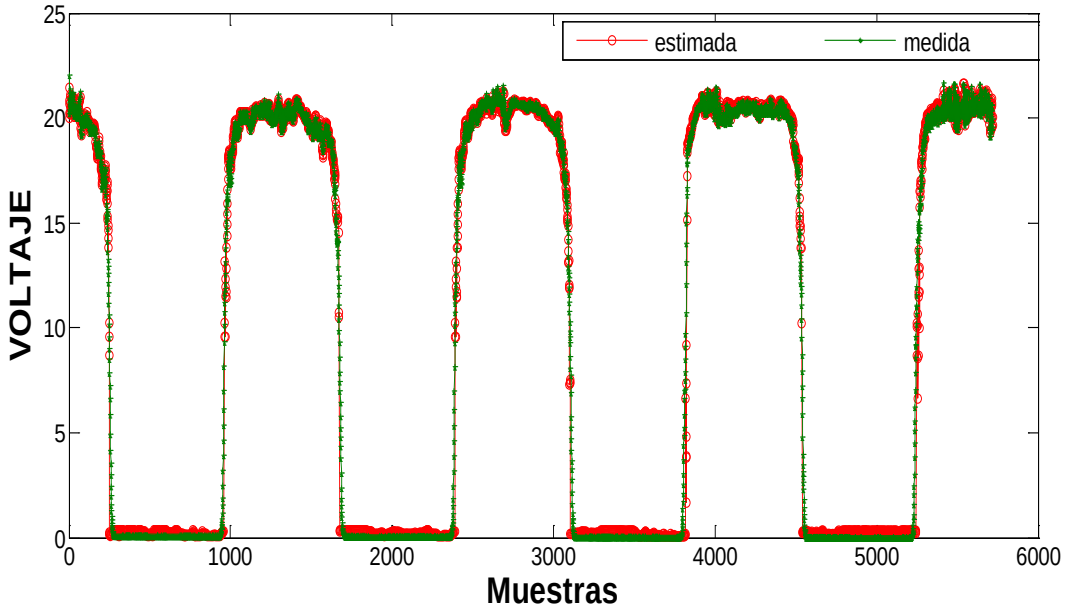


Figura 50. Variable de Voltaje proyectado por la red

MSE = 0,7612

3) MODELO REDI2

Variables utilizadas en el entrenamiento:

INPUT: RADIACION, TEMPERATURA

TARGET: INTENSIDAD

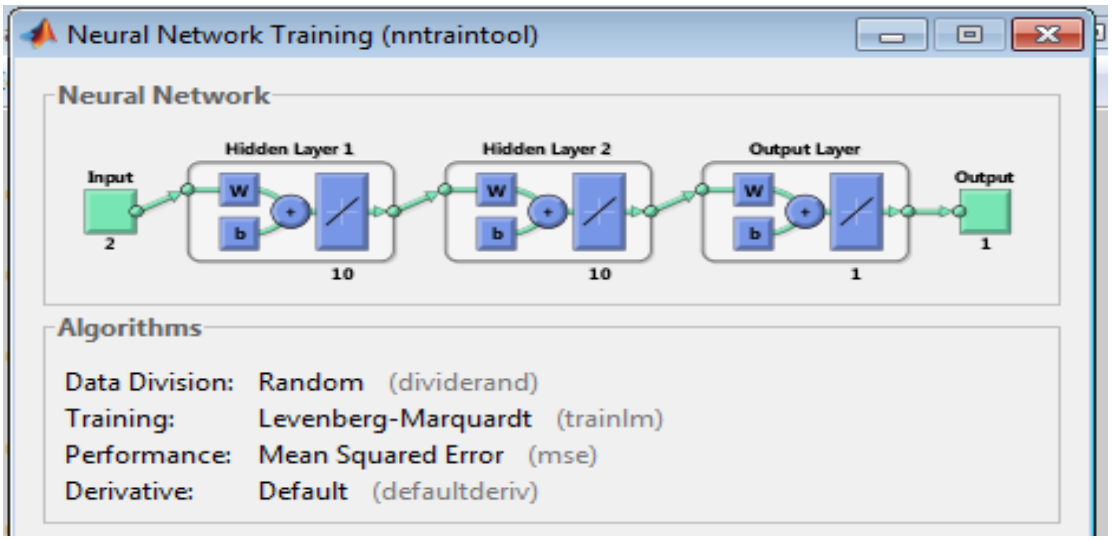


Figura 51. Estructura del modelo

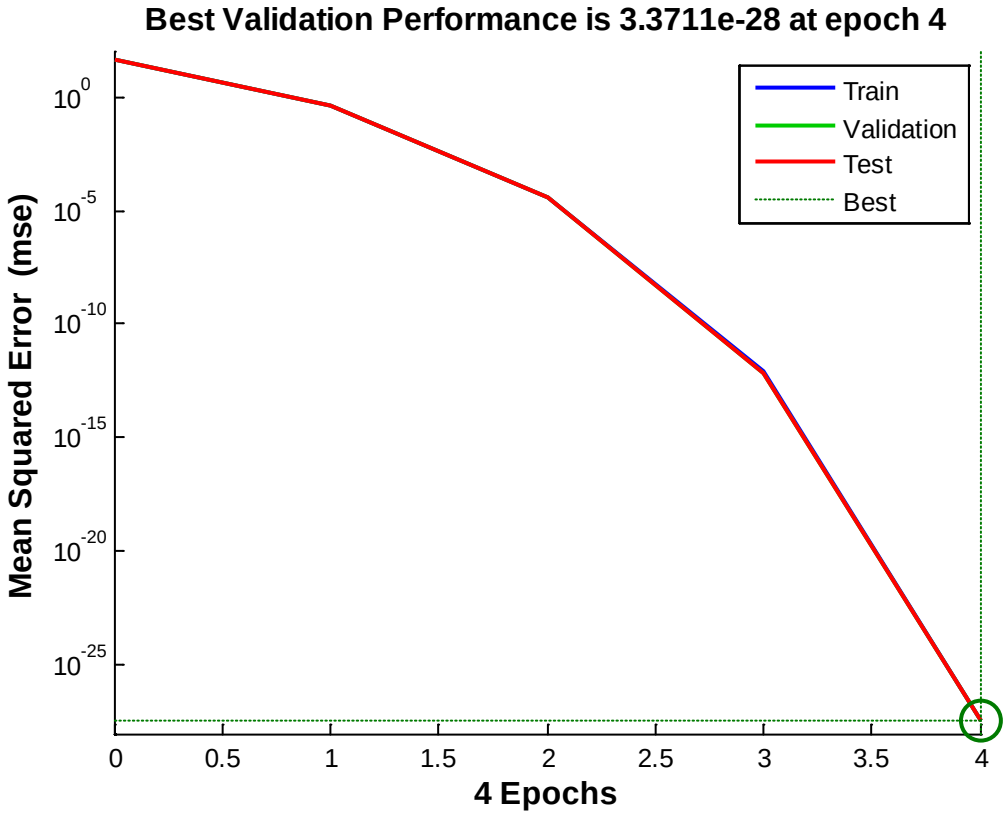


Figura 52. Rendimiento del modelo REDVI

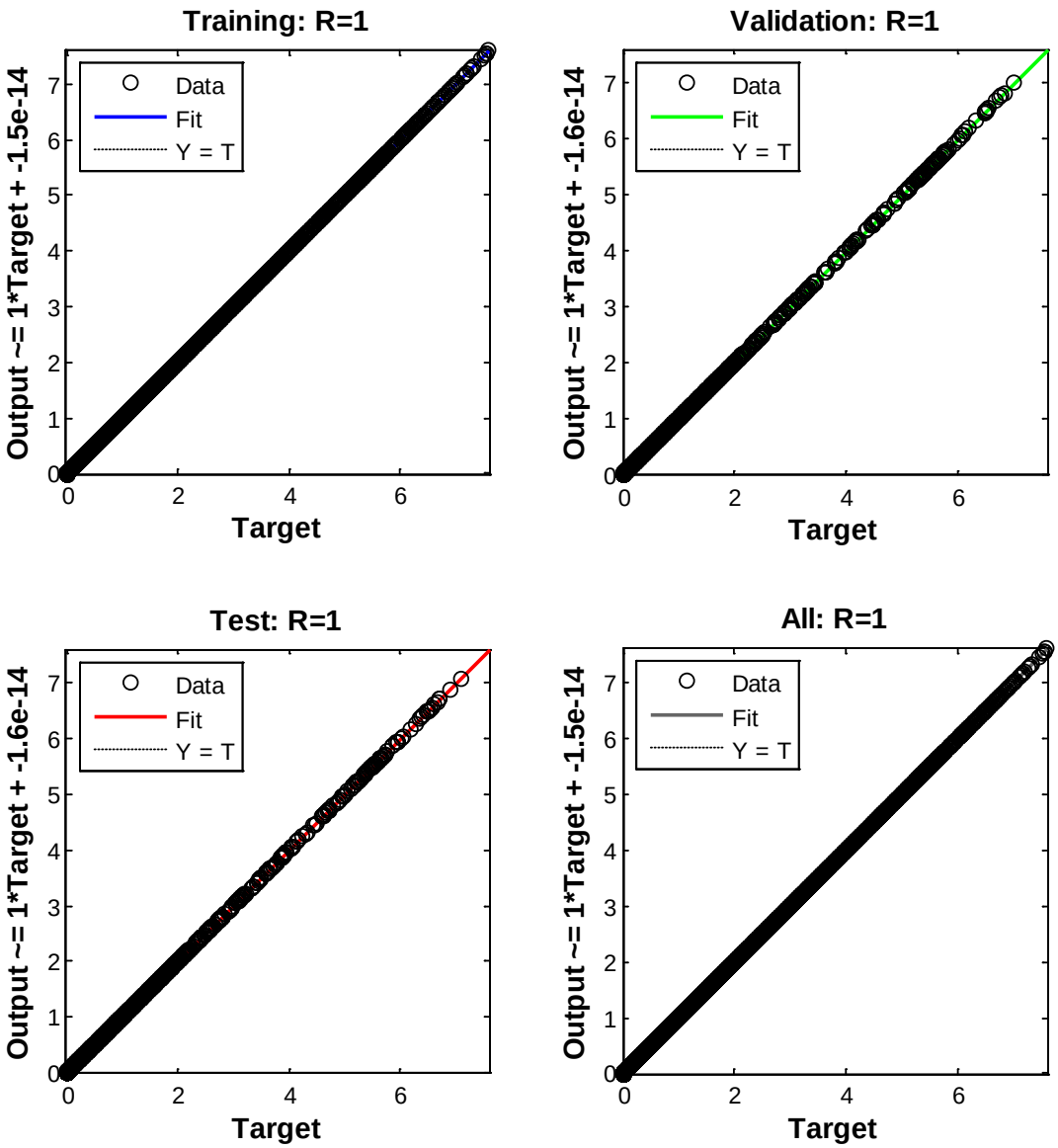


Figura 53. Gráfica de regresión de la red

Validación del modelo REDI2

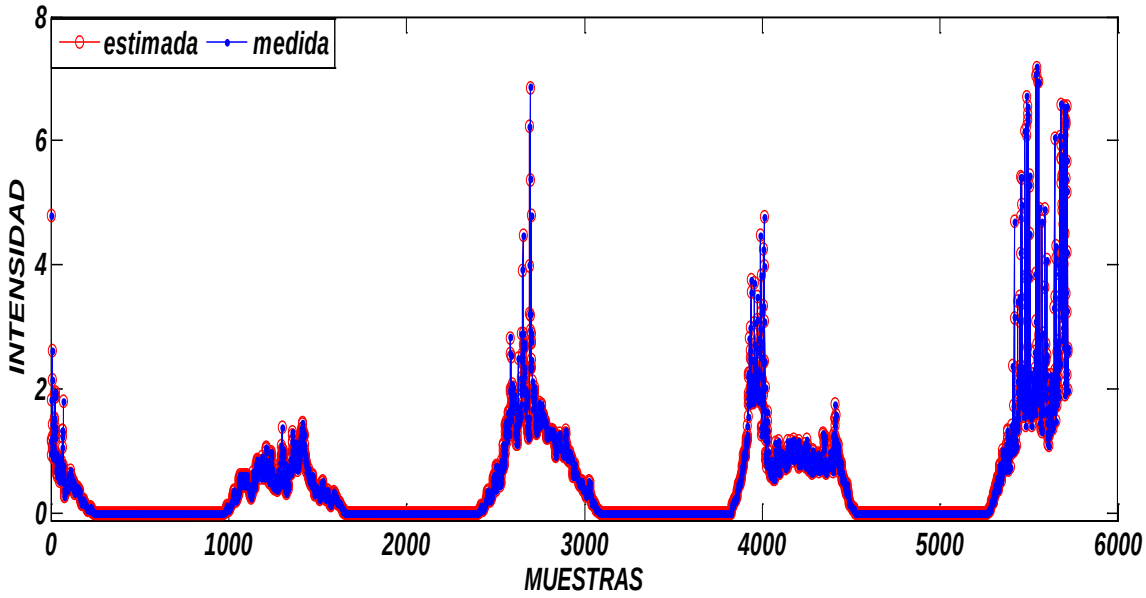


Figura 54. Variable de Intensidad proyectado por la red

MSE = 9.7782e-07

4) MODELO RTP

Variables utilizadas en el entrenamiento:
INPUT: RADIACION, TEMPERATURA
TARGET: POTENCIA

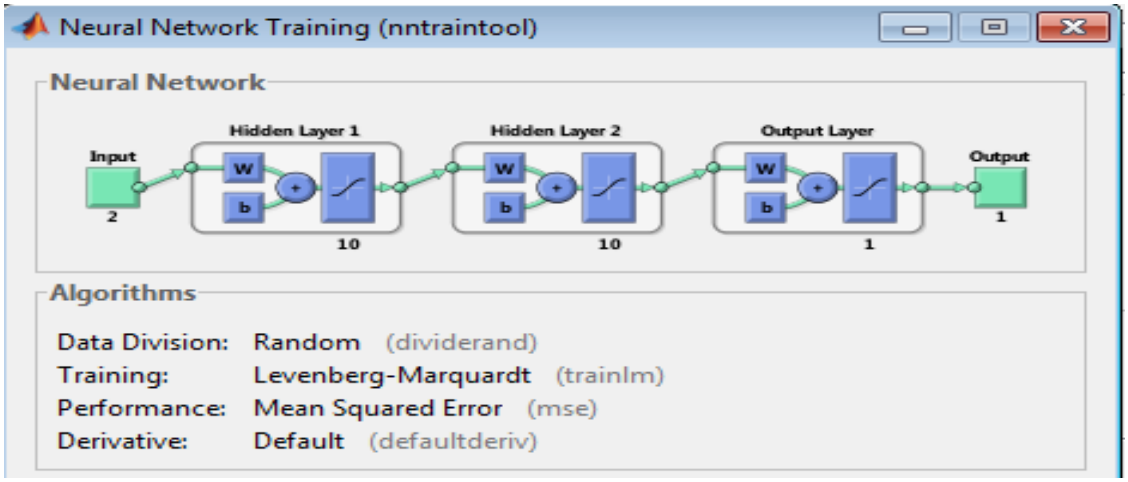


Figura 55. Estructura del modelo

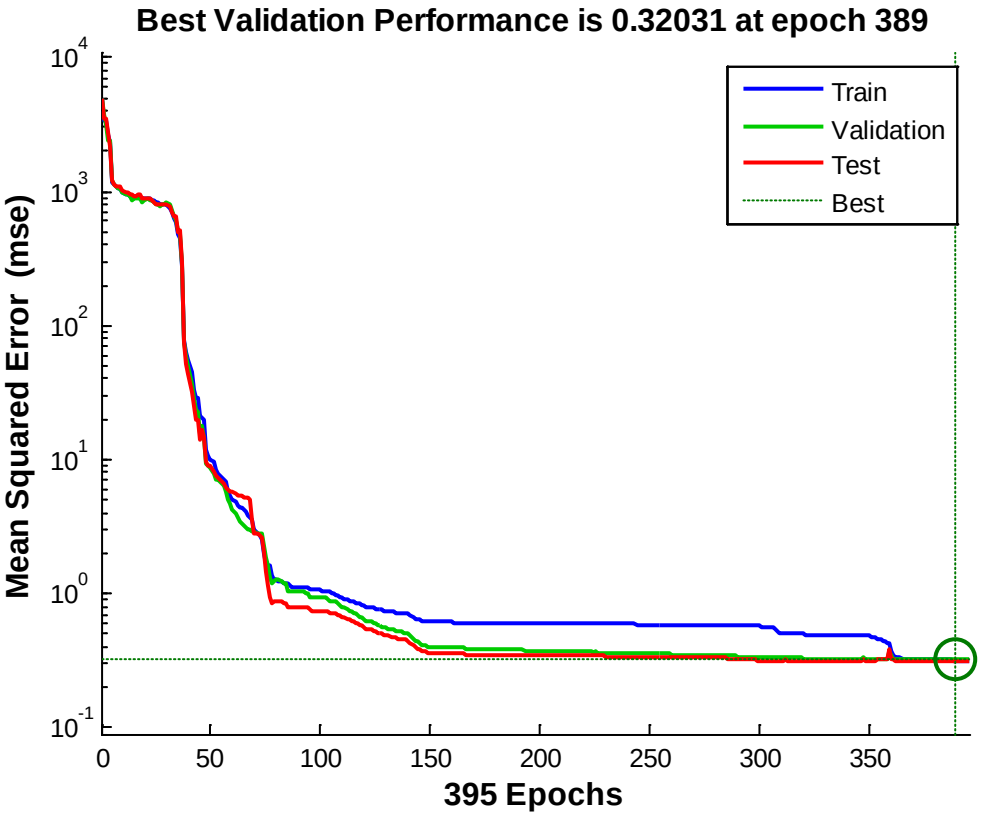


Figura 56. Rendimiento del modelo REDVI

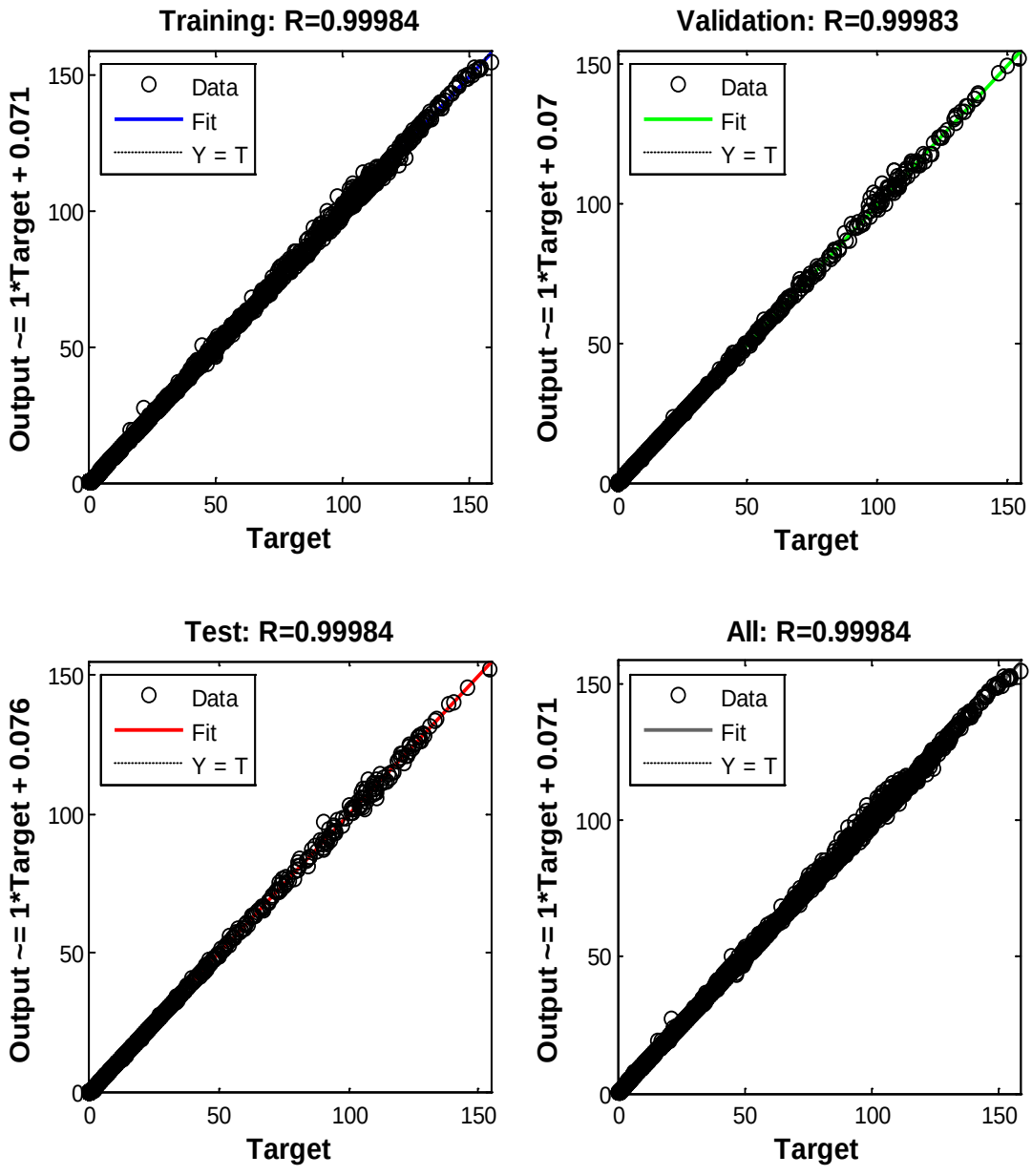


Figura 57. Gráfica de regresión de la red

Validación del modelo RTP

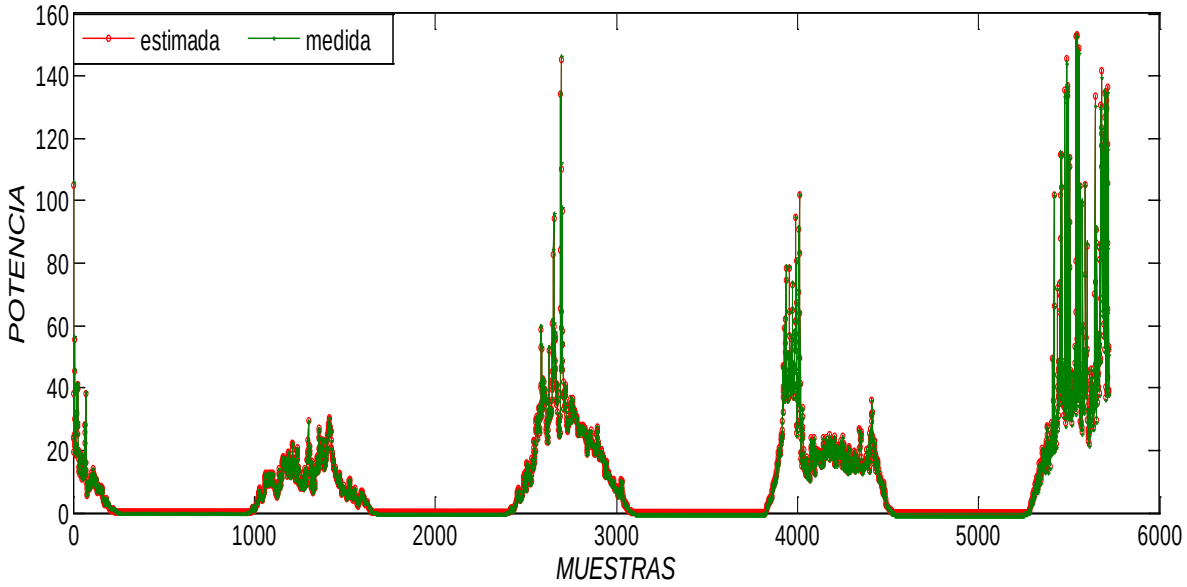


Figura 58. Variable de Potencia proyectado por la red

MSE = 0,0412

5) MODELO RT

Variables utilizadas en el entrenamiento:

INPUT: RADIACION

TARGET: TEMPERATURA

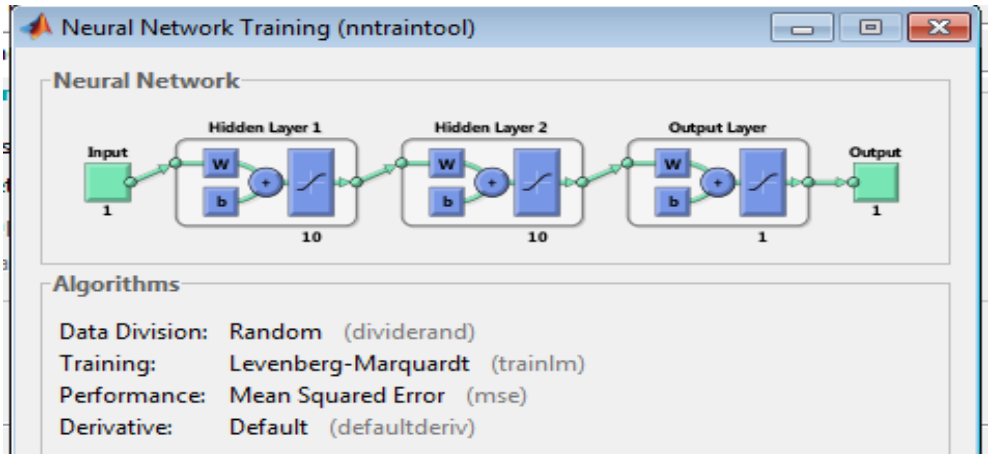


Figura 59. Estructura del modelo

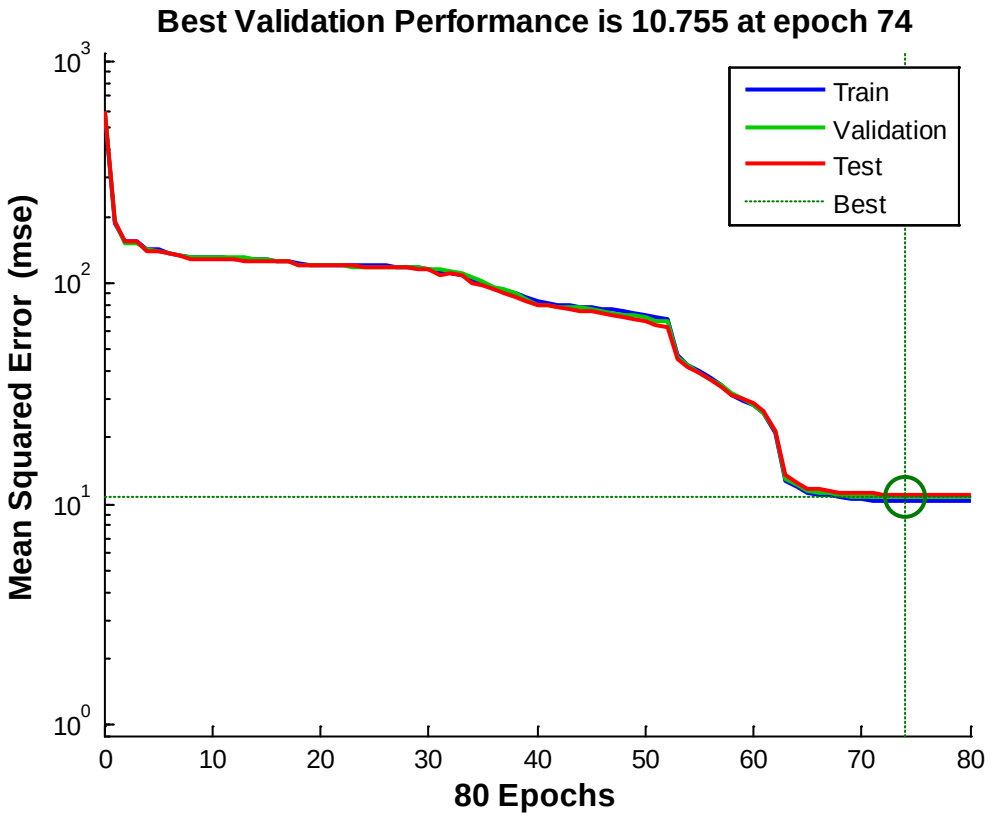


Figura 60. Gráfica de regresión de la red

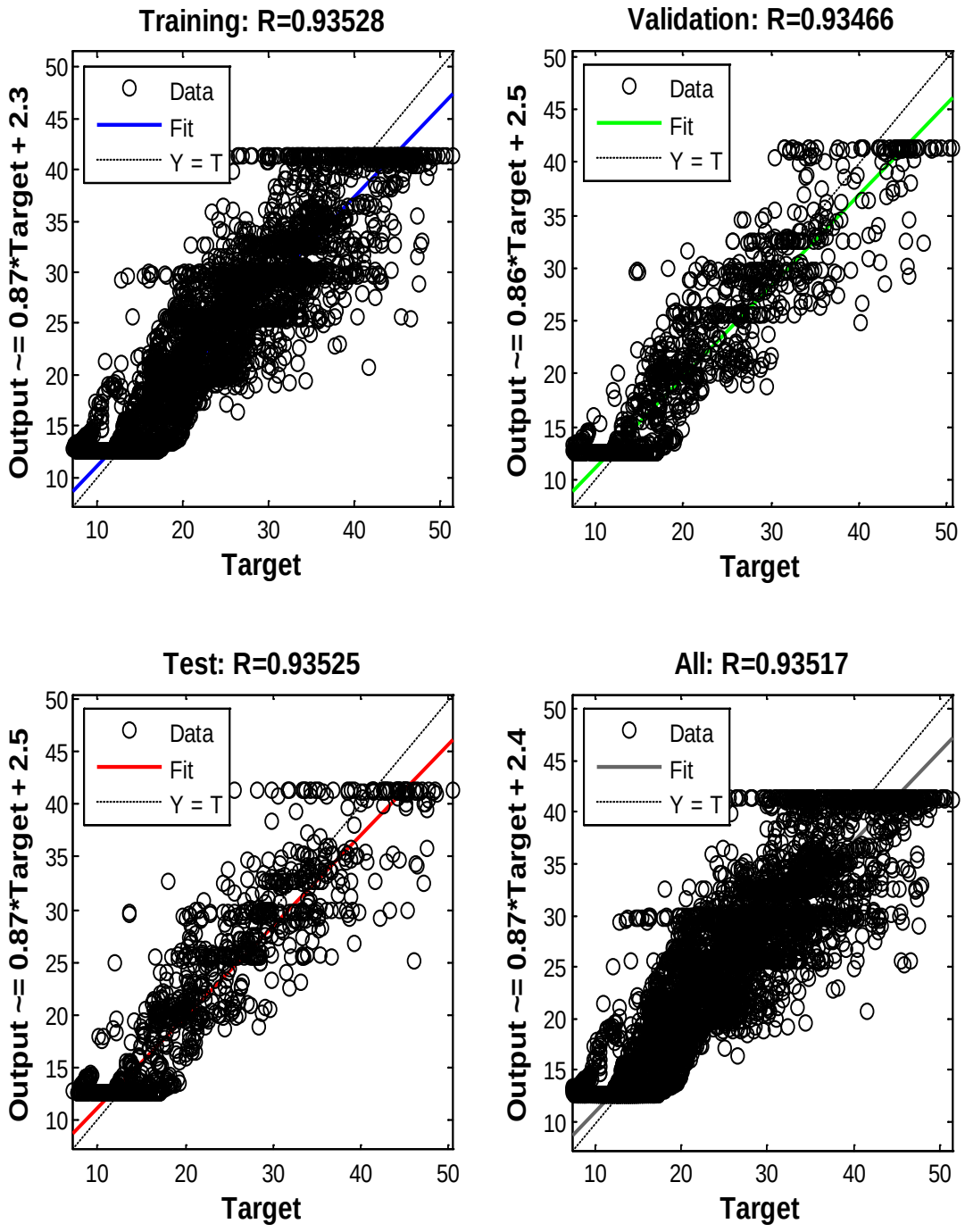


Figura 61. Gráfica de regresión de la red

Validación del modelo RT

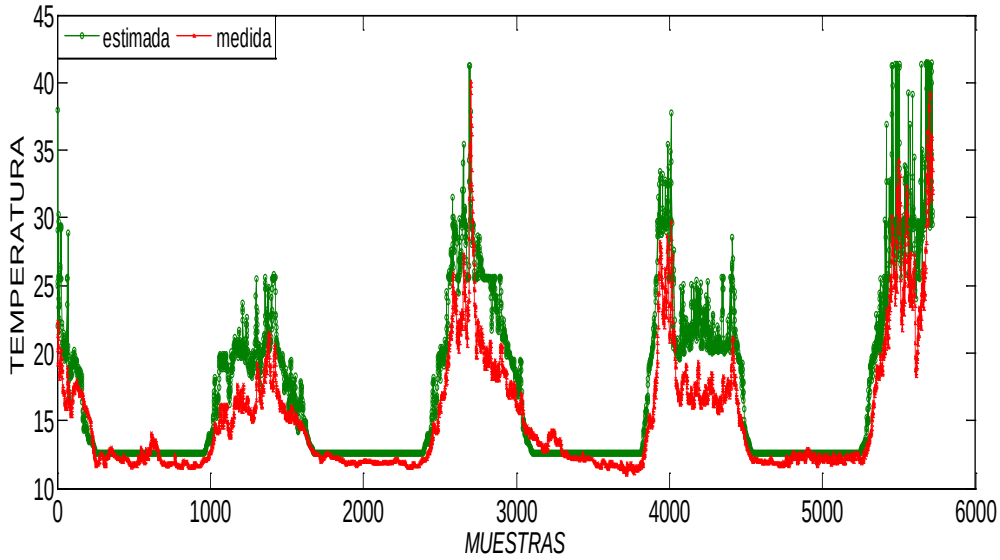


Figura 62. Variable de Temperatura proyectado por la red

MSE = 8,8190

6) **MODELO RP**

Variables utilizadas en el entrenamiento:
INPUT: RADIACION
TARGET: POTENCIA

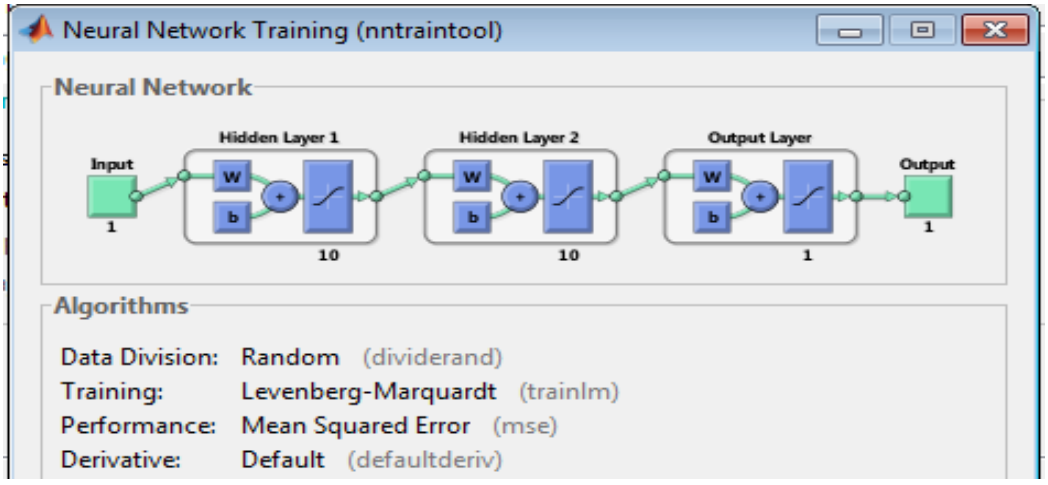


Figura 63. Variable de Potencia proyectado por la red

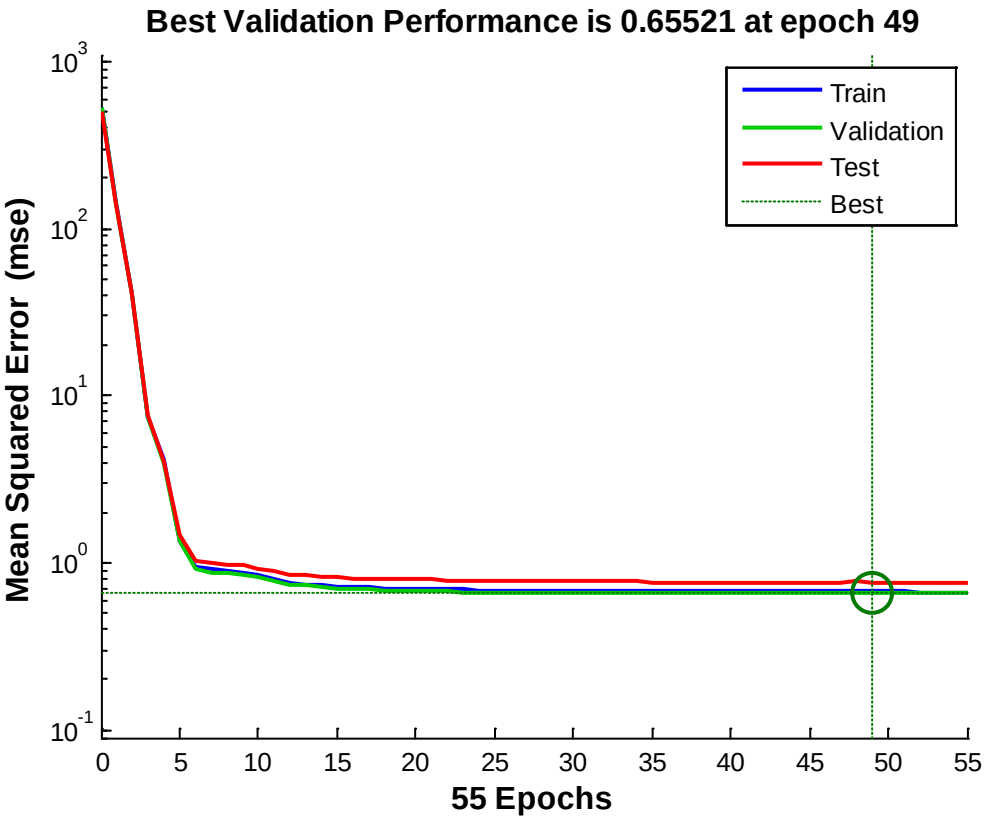


Figura 64. Gráfica de regresión de la red

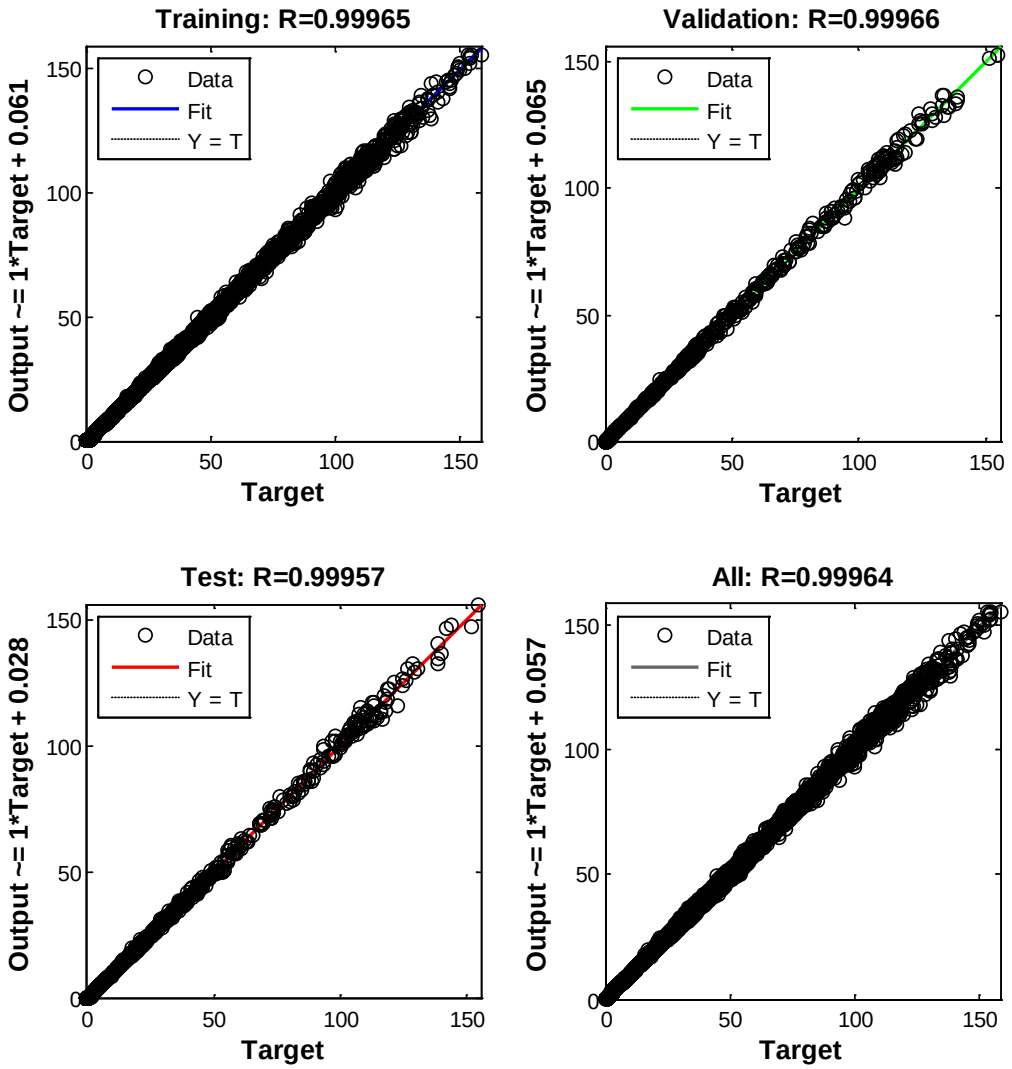


Figura 65. Gráfica de regresión de la red

Validación de los datos del modelo RP

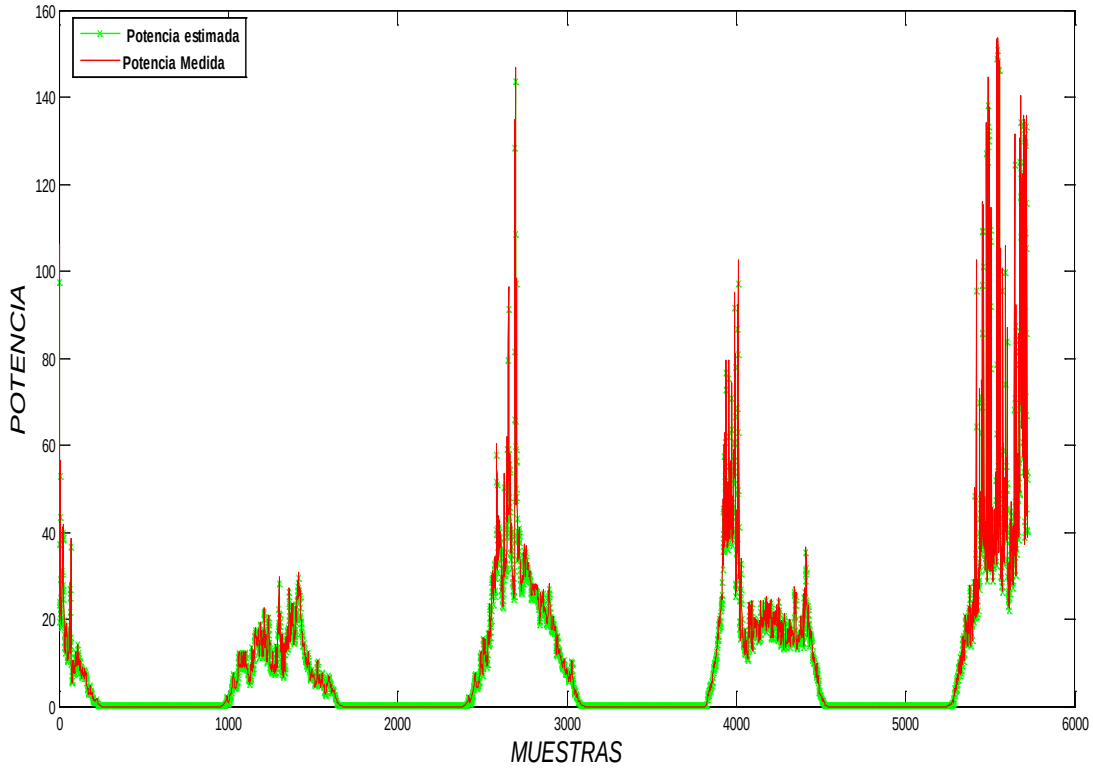


Figura 66. Gráfica de validación del modelo RP

MSE = 0.4505

En la tabla 4 se muestra un resumen de todos los modelos generados utilizando redes neuronales artificiales

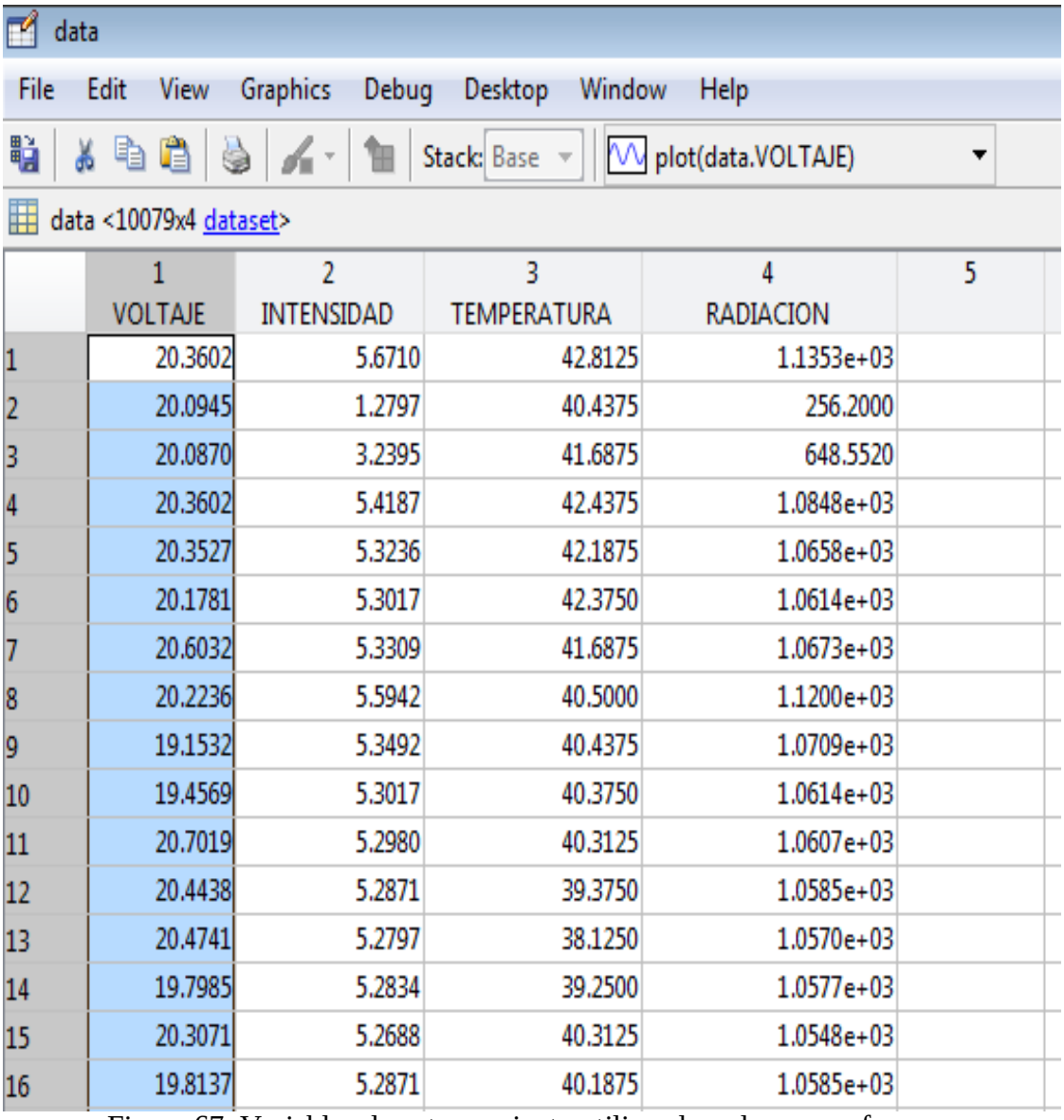
Tabla 4. Modelos con redes neuronales generados

NOMBRE DE LA RED	INPUT	TARGET	MSE…FUNCION DE TRANSFERENCIA		
			LOGSIG	**PURELIN**	**TANSIG**
REDVI	RADIACION TEMPERATURA	VOLTAJE INTENSIDAD			0.31577
REDVI2	RADIACION TEMPERATURA	VOLTAJE INTENSIDAD	33.3362		
REDVI3	RADIACION TEMPERATURA	VOLTAJE INTENSIDAD		23.1311	
REDV	RADIACION TEMPERATURA	VOLTAJE	59.5151		
REDV2	RADIACION TEMPERATURA	VOLTAJE		47.069	
REDV3	RADIACION TEMPERATURA	VOLTAJE			0.46848
REDI	RADIACION TEMPERATURA	INTENSIDAD			1.6303E-07
REDI2	RADIACION TEMPERATURA	INTENSIDAD		3.3711E-28	
REDI3	RADIACION TEMPERATURA	INTENSIDAD	10.6493		
RTP	RADIACION TEMPERATURA	POTENCIA			0.32031
RTP2	RADIACION TEMPERATURA	POTENCIA		0.5675	
RTP3	RADIACION TEMPERATURA	POTENCIA	4644.3049		
RT	RADIACION	TEMPERATURA			10.755
RT2	RADIACION	TEMPERATURA		17.18	
RT3	RADIACION	TEMPERATURA	183.9498		
RP	RADIACION	POTENCIA	0,4505		

f.2.3 MODELOS UTILIZANDO REDES neuro-fuzzy

f.2.3.1 ESTRUCTURA DE LOS DATOS

A diferencia de las redes neuronales, cada dato de cada variable debe estar en forma de filas como se ilustra en la figura 67.



	1 VOLTAJE	2 INTENSIDAD	3 TEMPERATURA	4 RADIACION	5
1	20.3602	5.6710	42.8125	1.1353e+03	
2	20.0945	1.2797	40.4375	256.2000	
3	20.0870	3.2395	41.6875	648.5520	
4	20.3602	5.4187	42.4375	1.0848e+03	
5	20.3527	5.3236	42.1875	1.0658e+03	
6	20.1781	5.3017	42.3750	1.0614e+03	
7	20.6032	5.3309	41.6875	1.0673e+03	
8	20.2236	5.5942	40.5000	1.1200e+03	
9	19.1532	5.3492	40.4375	1.0709e+03	
10	19.4569	5.3017	40.3750	1.0614e+03	
11	20.7019	5.2980	40.3125	1.0607e+03	
12	20.4438	5.2871	39.3750	1.0585e+03	
13	20.4741	5.2797	38.1250	1.0570e+03	
14	19.7985	5.2834	39.2500	1.0577e+03	
15	20.3071	5.2688	40.3125	1.0548e+03	
16	19.8137	5.2871	40.1875	1.0585e+03	

Figura 67. Variables de entrenamiento utilizando redes neuro-fuzzy

Las variables de entrenamiento (training) y las variables de prueba (checking) utilizados para generar varios modelos en este trabajo investigativo se muestran en la siguiente tabla:

Tabla 5. Variables utilizadas en los modelos neuro-fuzzy

Nombre de los modelos	Variables de entrenamiento (training) de un conjunto de 10079 datos	Variables de prueba (checking) de otro conjunto de 5720 datos
Modelo de potencia	RADIACION, TEMPERATURA, POTENCIA	RADIACION, TEMPERATURA, POTENCIA
Modelo de intensidad	RADIACION, TEMPERATURA, INTENSIDAD	RADIACION, TEMPERATURA, INTENSIDAD
Modelo de voltaje	RADIACION, TEMPERATURA, VOLTAJE	RADIACION, TEMPERATURA, VOLTAJE
Modelo de temperatura de panel	RADIACION, TEMPERATURA,	RADIACION, TEMPERATURA,

f.2.3.2 Metodología para la creación de redes neuro-fuzzy mediante la GUI anfisedit

Anfis es un sistema de inferencia donde los parámetros de membresía son ajustados usando el algoritmo de retropropagación (red neuronal artificial), esto provoca que el sistema difuso aprenda de los datos de entrada o salida proporcionados.

***Restricciones del editor Anfis:**

- Solo trabaja con sistemas difusos Sugeno.
- Tiene una única salida, pero sus funciones de membresía pueden ser todas del mismo tipo o diferentes.
- Solo puede existir una regla inferencia que haga referencia a cada una de las funciones de pertenencia de la salida.
- Se debe tener peso unitario para cada regla.

f.2.3.3 Procedimiento para el entrenamiento y validación de los datos en editor ANFIS

1. Digite en la ventana de comandos (Command Window) de Matlab ***anfisedit*** lo cual hará que aparezca la figura 68.

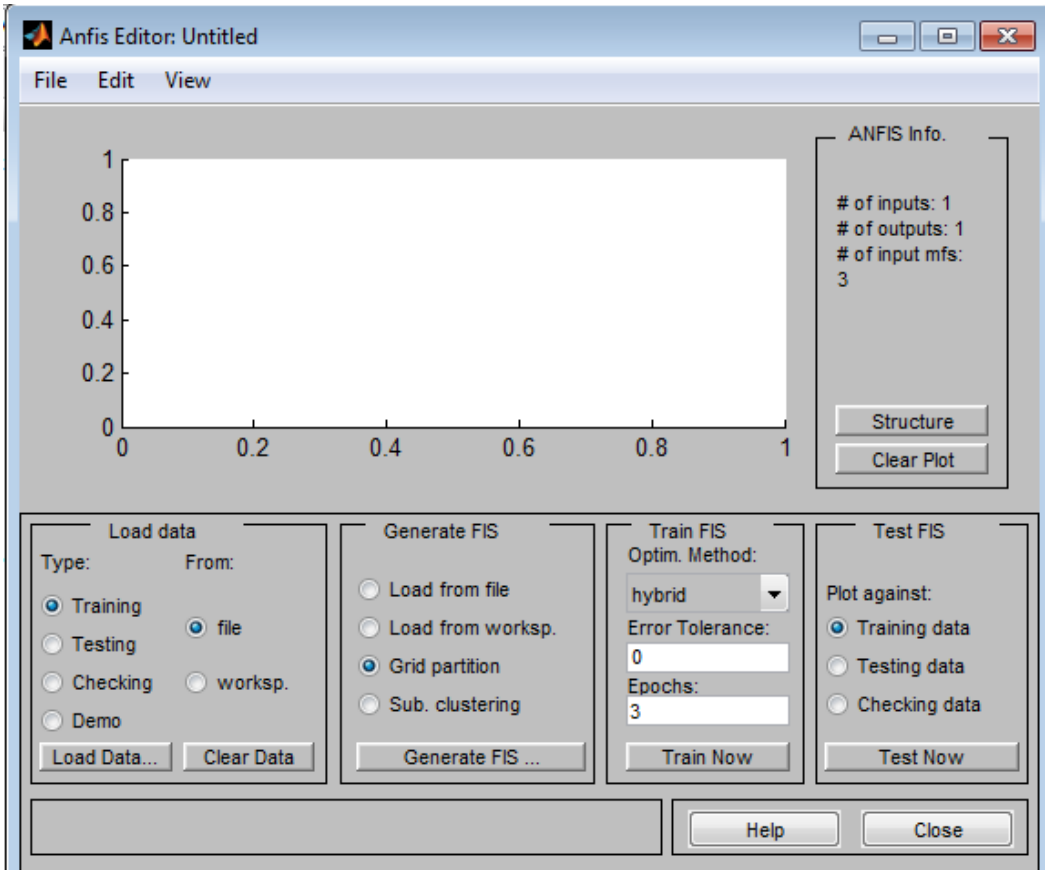


Figura 68. Ventana al editor de anfis

2. Busque el campo *Load data* y seleccione las opciones *Training y worksp.* Luego presione el botón *Load Data...* lo cual provocará se pida la entrada de datos que se usarán en el entrenamiento. Ver figura 69.

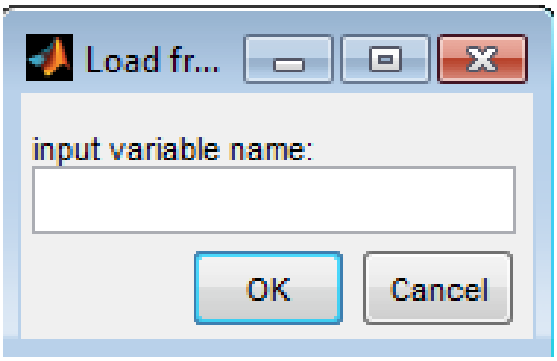


Figura 69. Pantalla para cargar datos de entrenamiento de la red.

3. Digite el nombre *ej.* ***RP*** y presione *OK*. Esto provocará que se carguen los datos y que se grafiquen como lo muestra la figura 70.

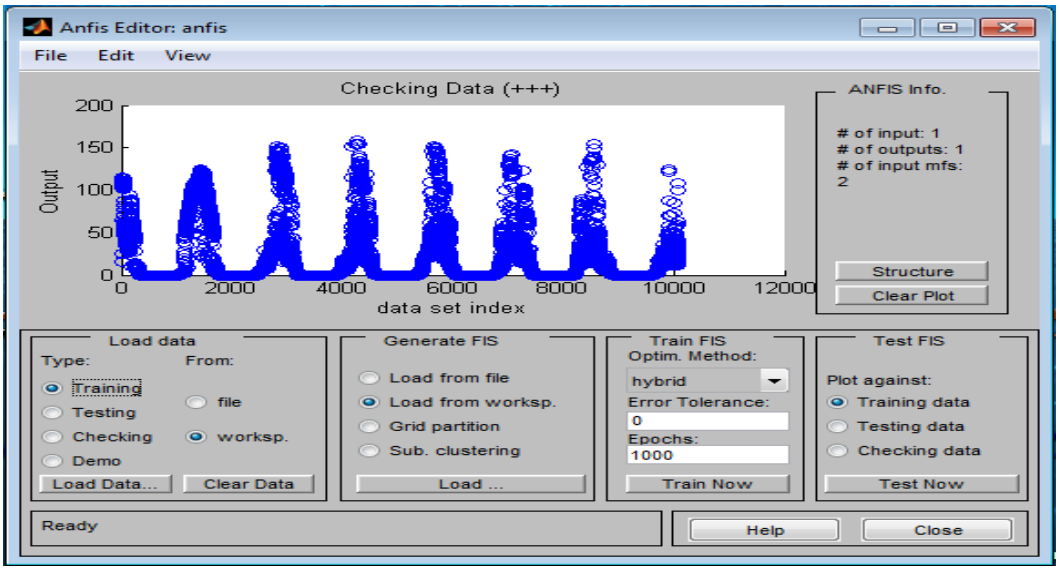


Figura 70. Datos de entrenamiento de red.

4. Busque el campo *Load data* y seleccione las opciones *Cheking y worksp.* Luego presione el botón *Load Data...* lo cual provocará se pida la entrada de datos que se usarán en el entrenamiento. Ver figura 70.

5. Digite el nombre *por ej.* ***rp*** y presione *OK*. Esto provocará que se carguen los datos y que se grafiquen como lo muestra la figura 71.

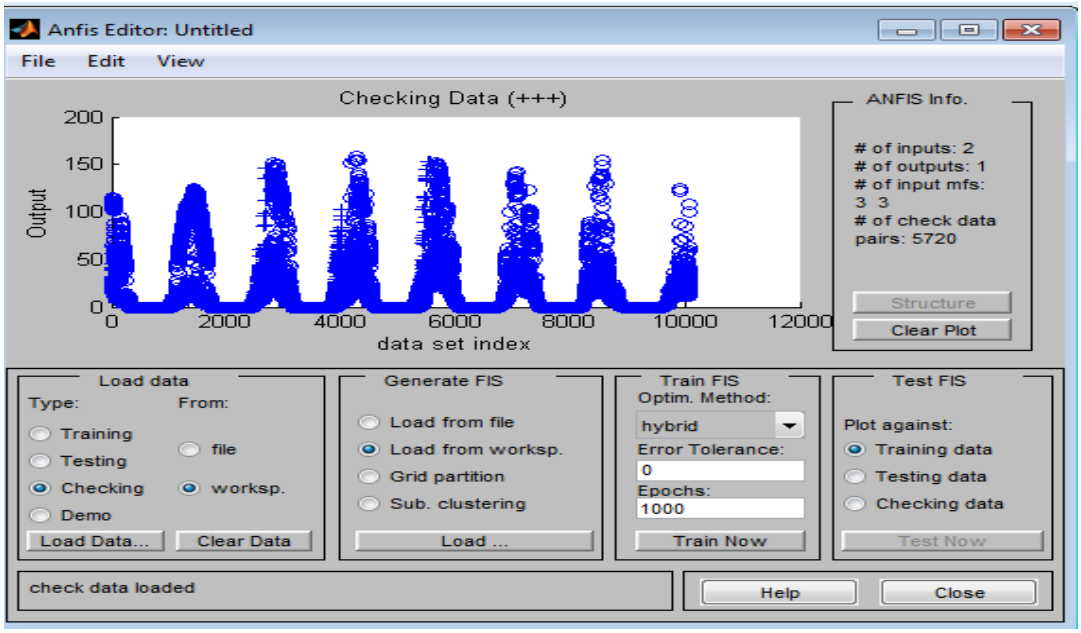


Figura 71. Datos de entrenamiento y de verificación.

Estos datos serán usados para ajustar las funciones de membresía.

6. Ahora se puede generar el sistema difuso.

En esta tesis se utiliza las funciones por defecto de genfis1, genfis2 y genfis3 para generar el sistema difuso, a continuación se detalla el procedimiento a seguir para usar estas funciones.

6. a GENFIS1

Genera la estructura difusa del sistema de inferencia a partir de datos utilizando el método de partición de rejilla (grid partition). La función requiere un conjunto de datos y reconoce las primeras columnas como entradas y solamente su última columna como salida, la estructura es como sigue:

fismat = genfis1 (data)

El número predeterminado de las funciones de afiliación, numMFs , es 2, el miembro de entrada por defecto el tipo de función es "**gbellmf**" , y el de salida por defecto pertenencia a un tipo de función es "lineal" . La siguiente tabla resume los métodos de inferencia por defecto.

Tabla 6. Métodos de Inferencia difusa de GENFIS1

Método de Inferencia	Defecto
Y	productos
O	máximo
Implicación	productos
Agregación	máximo
Defuzzificación	wtaver

6. b GENFIS2

Genera la estructura difusa del sistema de inferencia de datos mediante el agrupamiento sustractivo (subtractive clustering).

Estructura genfis2

fismat = genfis2 (Xin, xout, radii)

os argumentos para genfis2 son como sigue:

- Xin es una matriz en la que cada fila contiene los valores de entrada del conjunto de datos.
- Xout es una matriz en la que cada fila contiene los valores de salida para en conjunto de datos.
- radii es un vector que especifica un rango de centro del conglomerado de influencia en cada una de las dimensiones de datos, suponiendo que los datos cae dentro de una unidad hyperbox.

  Por ejemplo, si la dimensión de datos es 3 (por ejemplo, Xin tiene dos columnas y xout tiene una columna), radios = [0,5 0,4 0,3] especifica que los rangos de influencia en las dimensiones de datos primera, segunda, y tercera (es decir, el primera columna de Xin , la segunda columna de Xin , y la columna de xout) son 0,5, 0,4 y 0,3 veces la anchura del espacio de datos, respectivamente. Si los radios es un valor escalar, entonces este valor escalar se aplica a todas las dimensiones de datos, es decir, cada centro del cúmulo tiene una zona de influencia esférica con un radio dado.

La entrada por defecto pertenencia a un tipo de función es "gaussmf ' , y el de salida por defecto pertenencia a un tipo de función es "lineal" .

La siguiente tabla resume los métodos de inferencia por defecto.

Tabla 7. Métodos de Inferencia difusa en GENFIS2

Método de Inferencia	Defecto
Y	productos
O	PROBOR
Implicación	productos
Agregación	máximo
Defuzzificación	wtaver

6. c **GENFIS3**

Generar la estructura difusa del sistema de inferencia a partir de datos utilizando la agrupación FCM. La función requiere distintos conjuntos de datos de entrada y salida como argumentos de entrada. Cuando sólo hay una salida, se puede utilizar genfis3 para generar un FIS iniciales para ANFIS formación. El método de extracción de la primera regla utiliza la función FCM para determinar el número de reglas y funciones de pertenencia de los antecedentes y consecuentes.la estructura es:

fismat = genfis3 (Xin, xout)

De esta forma fismat genera una estructura de tipo Sugeno FIS (fismat) dada la entrada de datos Xin y salida de datos xout . Las matrices de Xin y xout deben tener una columna para cada entrada y salida de la FIS, respectivamente.

La siguiente tabla resume los métodos de inferencia por defecto.

Tabla 8. Métodos de Inferencia difusa

Método de Inferencia	Defecto
Y	productos
O	PROBOR
Implicación	productos
Agregación	suma
Defuzzificación	wtaver

Para cargar un sistema ya hecho, basta con que se seleccione *Generate FIS* y cargar un sistema ya hecho ya sea desde disco o desde el espacio de trabajo como se muestra en la figura 72.

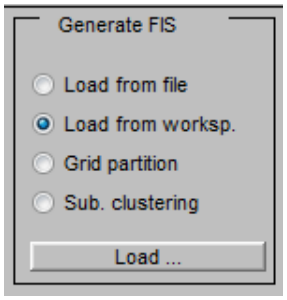


Figura 72. Recorte de pantalla de la opción para generar el fis desde el Editor de Anfis

7. También se puede modificar las propiedades del fis, en el menú ***edit / membership Function Editor*** lo cual hará que se muestre una pantalla de configuración del sistema difuso, ver figura 73.

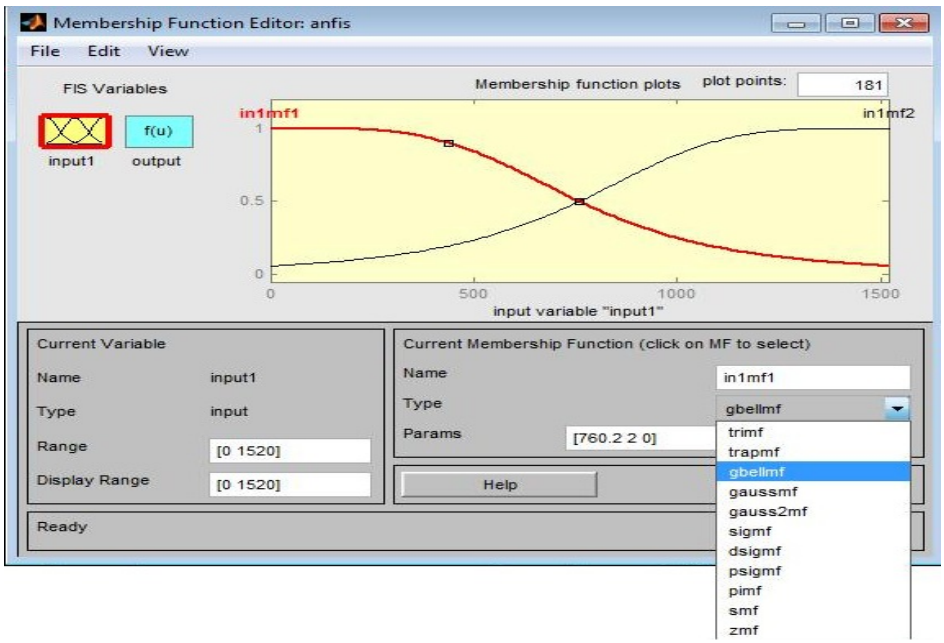


Figura 73. Parámetros para sistema difuso.

Es de hacer notar que en la figura 74 solo podemos modificar el número de funciones de membresía, el tipo de función de membresía y el tipo de salida que se requiere, la cual solo puede ser constante o lineal debido a la limitante de solo poder usar un sistema del tipo *Sugeno*.

8. Una vez generado el sistema difuso podemos ver la estructura, seleccionando el botón con el nombre *structure*, ver figura 74.

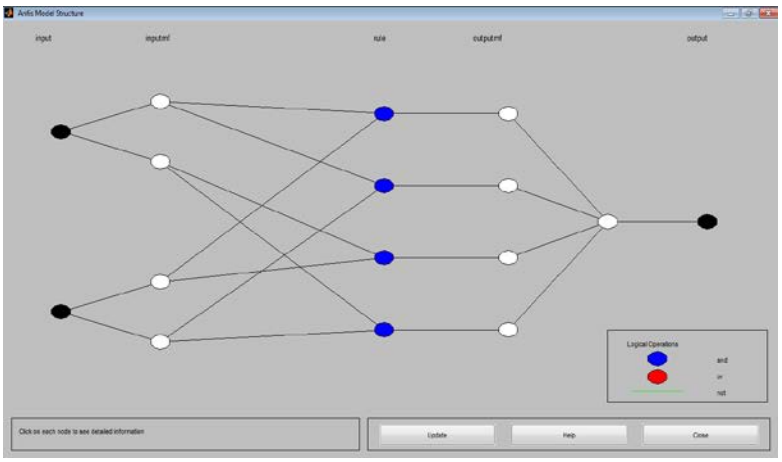


Figura 74. Estructura del sistema difuso creado con ANFIS.

9. Ahora solo nos resta entrenar el sistema, para lo cual debemos hacer algunas configuraciones primero. Esto se hace seleccionando método de optimización de la red, el error, y el número de iteraciones (epochs) que se necesitan.

10. Una vez configurados los parámetros de entrenamiento, presione el botón *Train now*, lo que provocará que el sistema sea entrenado con los datos cargados, el sistema mostrará una gráfica con el error de los datos de entrenamiento (training) y el error de los datos de verificación (checking) ver figura 75.

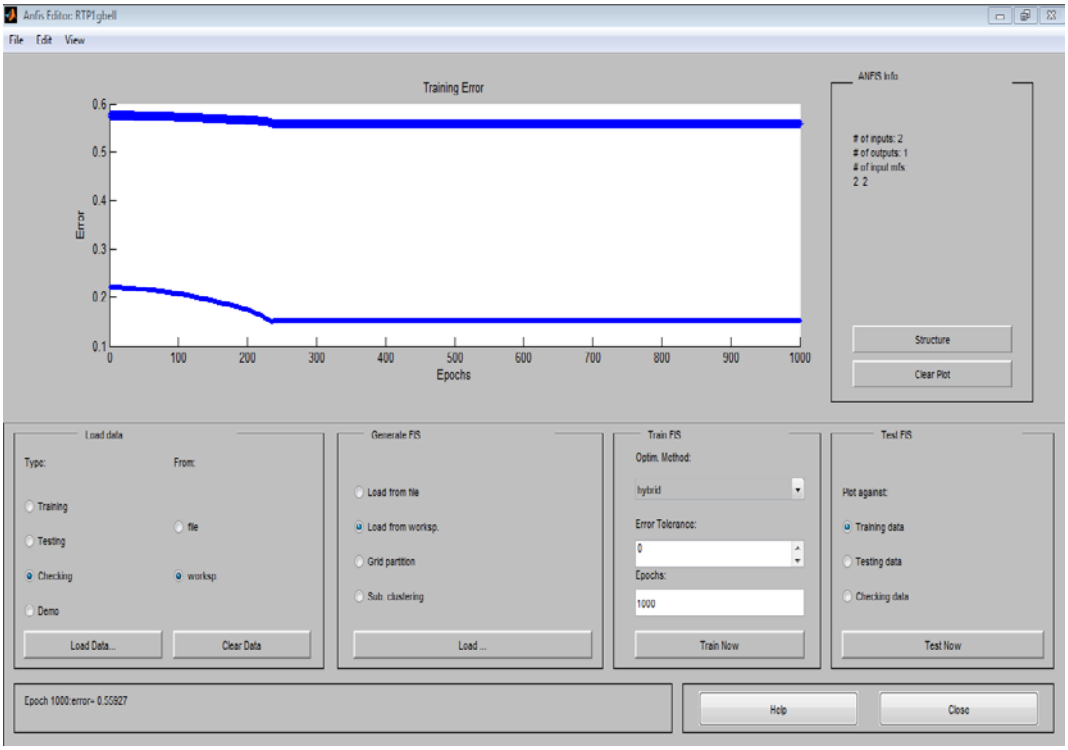


Figura 75. Gráfica de errores de entrenamiento y de verificación.

11. Una vez hecho el paso anterior ya esta creado el sistema difuso a partir de datos de entrada, para guardarlo, solo es necesario ir a la opción *File*, *export*, *to disk*, lo cual desplegará la ventana para escribir el nombre del sistema y luego presionar el botón guardar.

12. Si queremos ver en la interfaz gráfica de usuario el sistema almacenado, basta con poner en la línea de comandos de Matlab **>>*fuzzy nombre*** lo cual desplegará el sistema creado. Esto nos permite editar las funciones de membresía para adaptarlas a un caso particular.

13. Si queremos verificar algunas gráficas como por ejemplo los datos de entrenamiento y de verificación, basta seleccionar dichas opciones en la figura 75, y luego presionar el botón *Test now*

f.2.3.3.1 Modelos con redes neuro-fuzzy

a) Modelo de Potencia

Entradas: Radiación, Temperatura

Salida: Potencia

Entrenamiento con GENFIS 1

Tabla 9. Características de la red anfis de potencia

# MUESTRAS data		FM§ entrada	NE††	FM salida	Fis de entrenamiento	Método de optimización	Error	
training	checking						training	checking
10079	5720	GBELL	250	LINEAR	Genfis1	HÍBRIDO	0.55927	0.15108

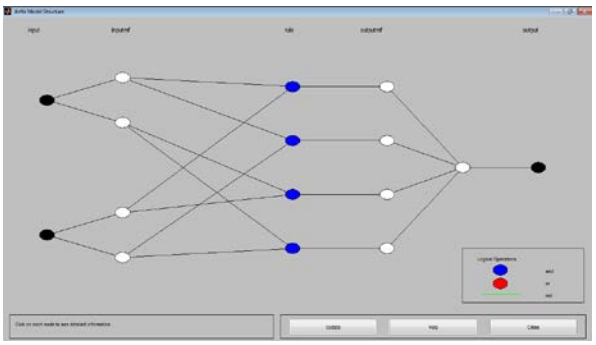


Figura 76. Estructura de la red neurofuzzy de potencia

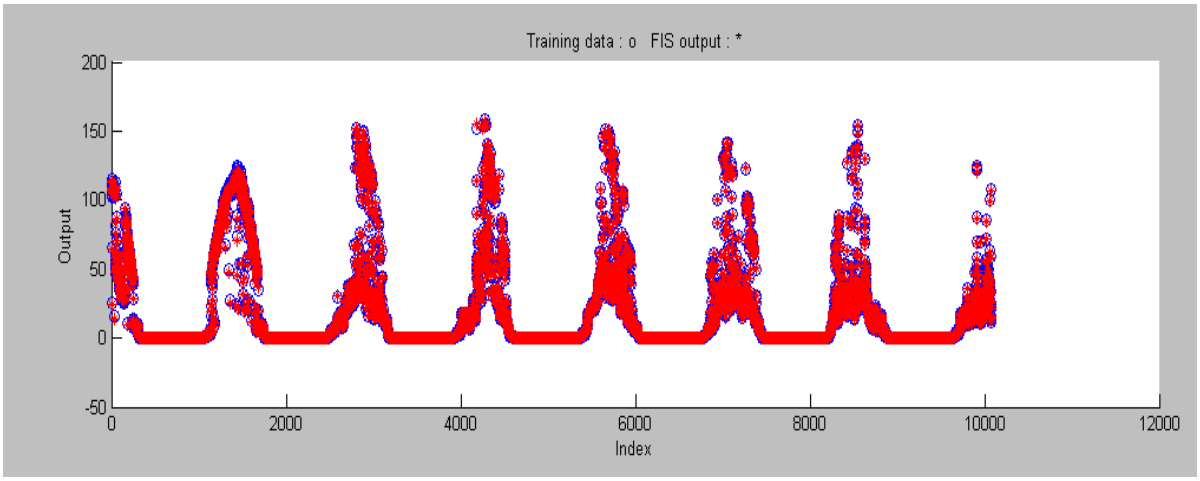


Figura 77. Training de la red neurofuzzy de potencia

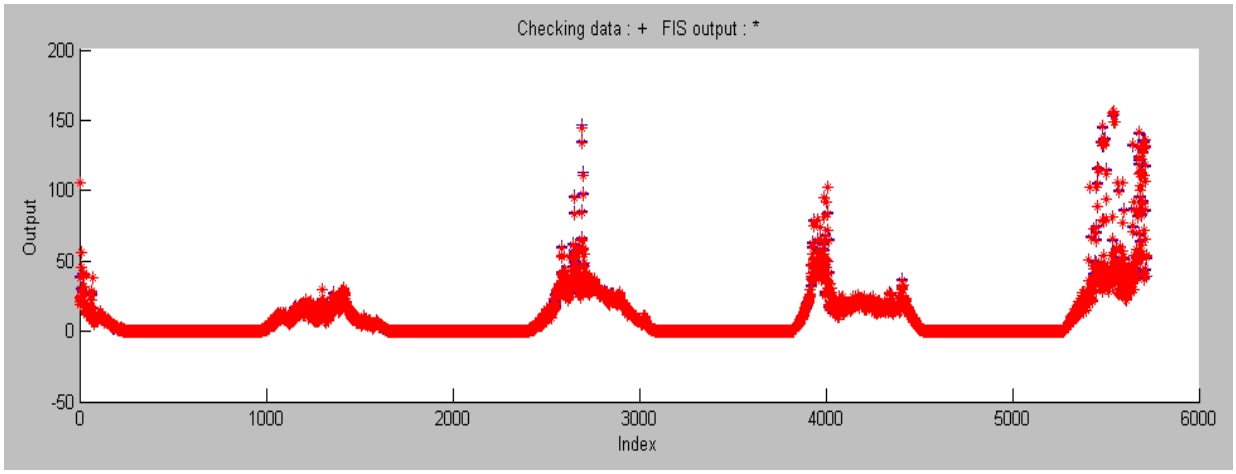


Figura 78. Checking de la red neurofuzzy de potencia

b) Modelo de intensidad

Entradas: Radiación, Temperatura

Salida: Intensidad

Tabla 10. Características de la red anfis de potencia

# MUESTRAS data		FM§ entrada	NE††	FM salida	Fis de entrenamiento	Método de optimización	Error	
training	checking						training	checking
10079	5720	GAUSS2	250	LINEAR	Genfis1	HÍBRIDO	1,1996e-8	0.0009888

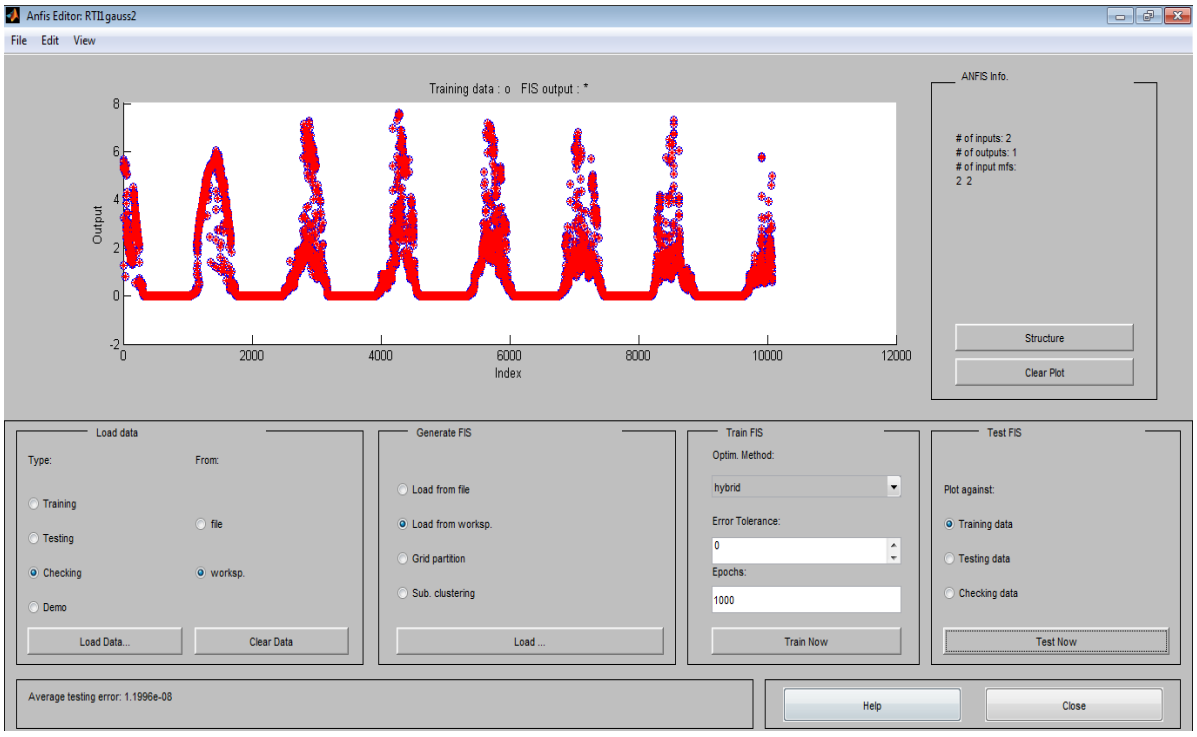


Figura 79. Estructura de la red neurofuzzy de intensidad

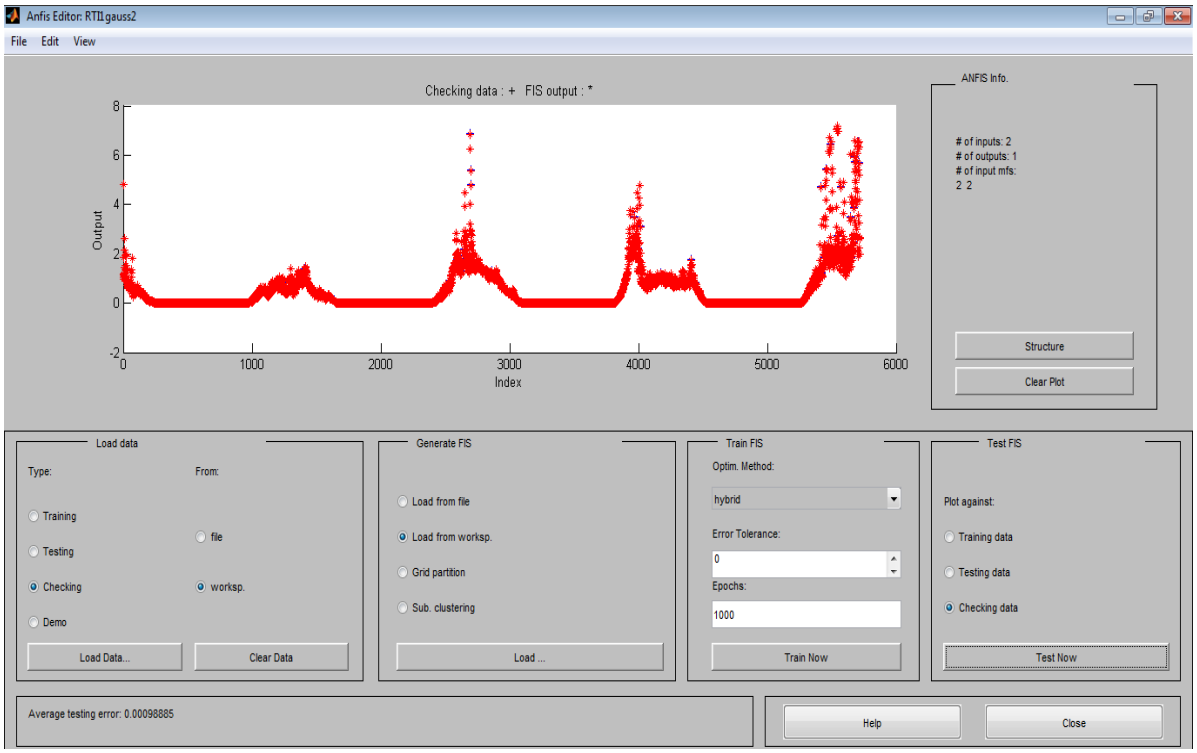


Figura 80. Checking de la red neurofuzzy de potencia

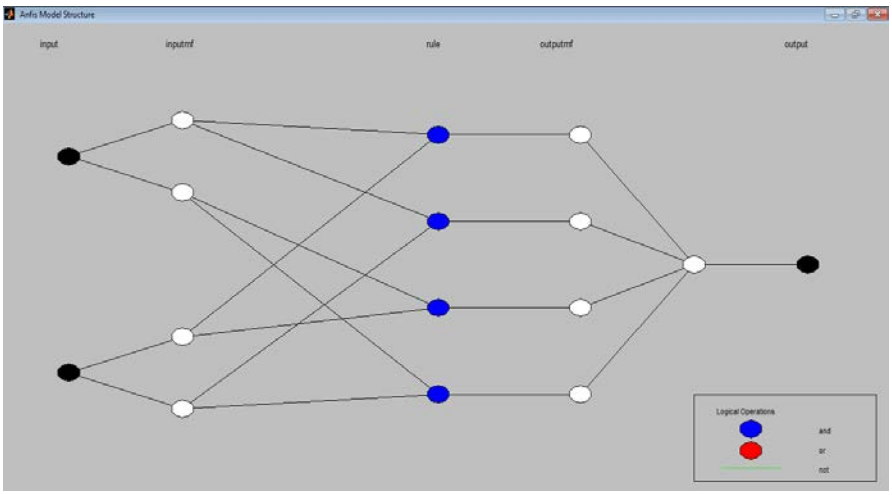


Figura 81. Estructura de la red neurofuzzy de intensidad

c) Modelo de voltaje

Entradas: Radiación, Temperatura
Salida: Voltaje

Tabla 11. Características de la red anfis de voltaje

# MUESTRAS data		FM§ entrada	NE††	FM salida	Fis de entrenamiento	Método de optimización	Error	
training	checking						training	checking
10079	5720	GBELL	500	LINEAR	Genfis1	HÍBRIDO	2,5314	2,7698

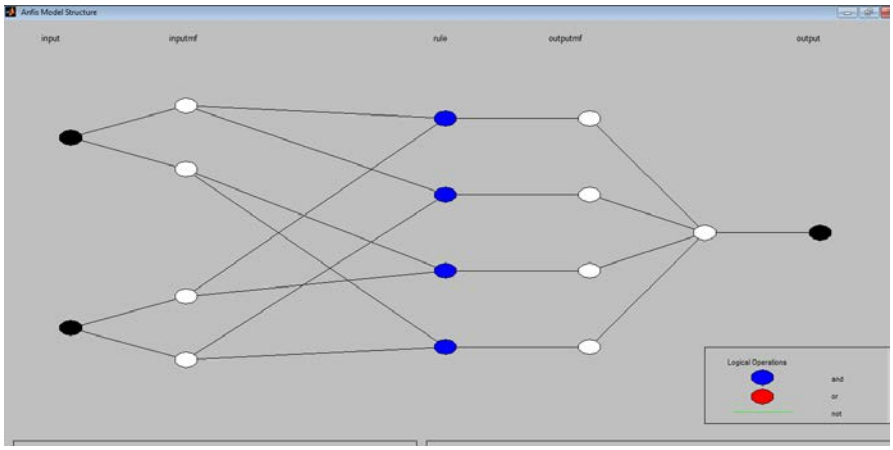


Figura 82. Estructura de la red neurofuzzy de voltaje

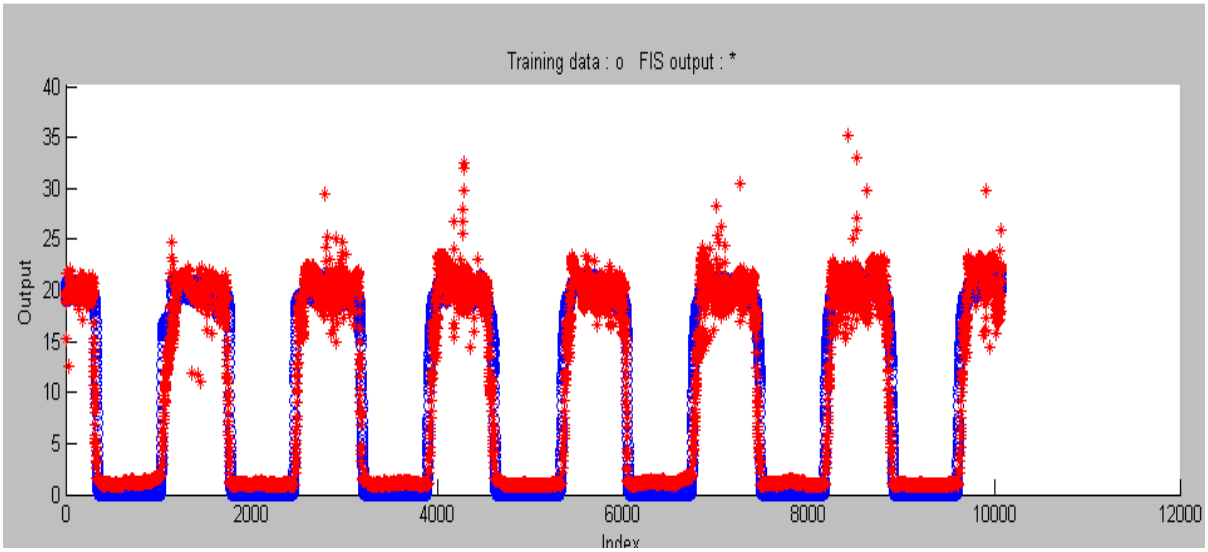


Figura 83. Estructura de la red neurofuzzy de voltaje

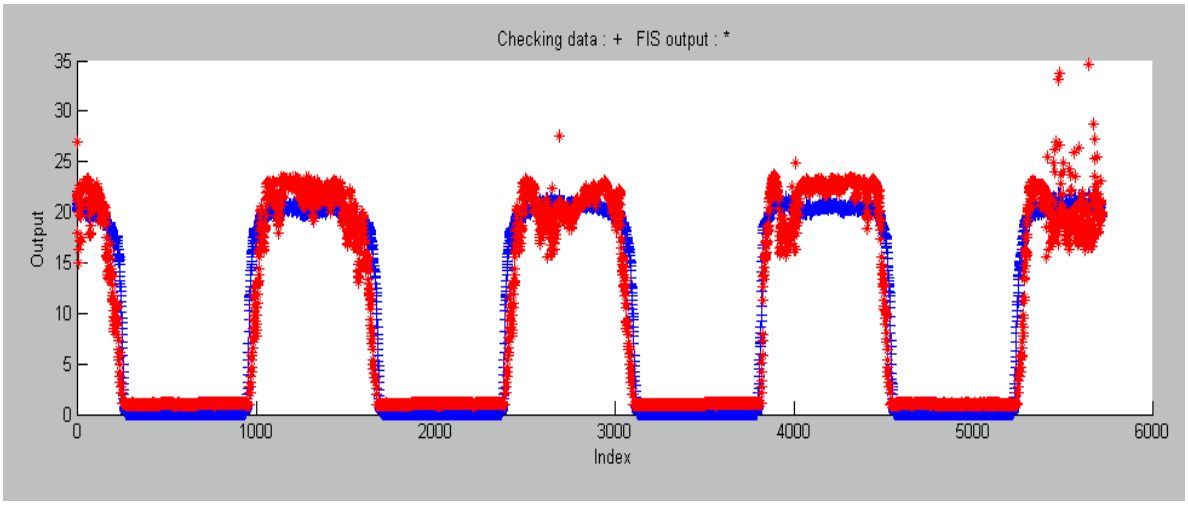


Figura 84. Checking de la red neurofuzzy de voltaje

d) Modelo de Temperatura

Entrada: Radiación
Salida: Temperatura

Tabla 11. Características de la red anfis de Temperatura

# MUESTRAS data		FM§ entrada	NE††	FM salida	Fis de entrenamiento	Método de optimización	Error	
training	**checking**						training	checking
10079	5720	GBELL	500	LINEAR	Genfis1	HÍBRIDO	3.3102	2.96

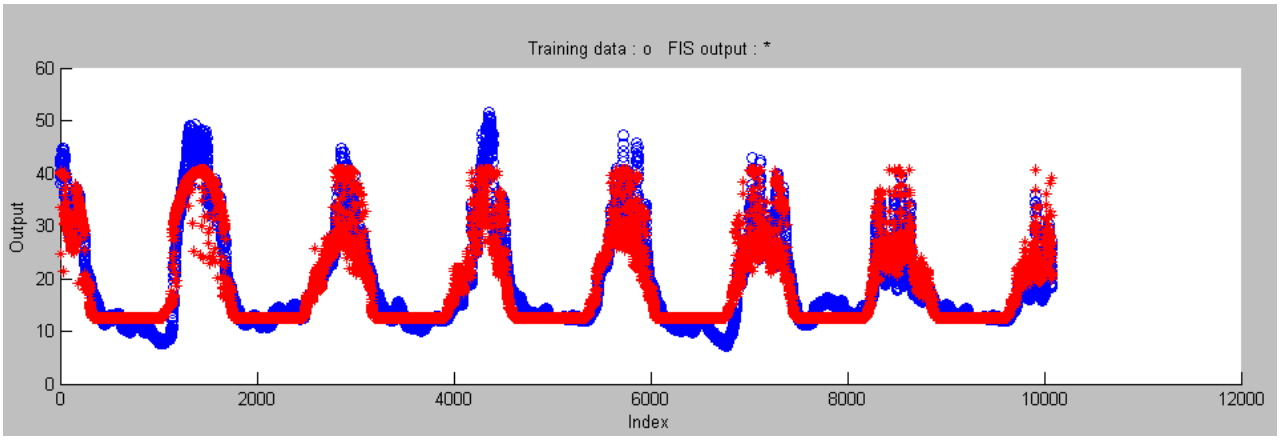


Figura 85. Estructura de la red neurofuzzy de Temperatura

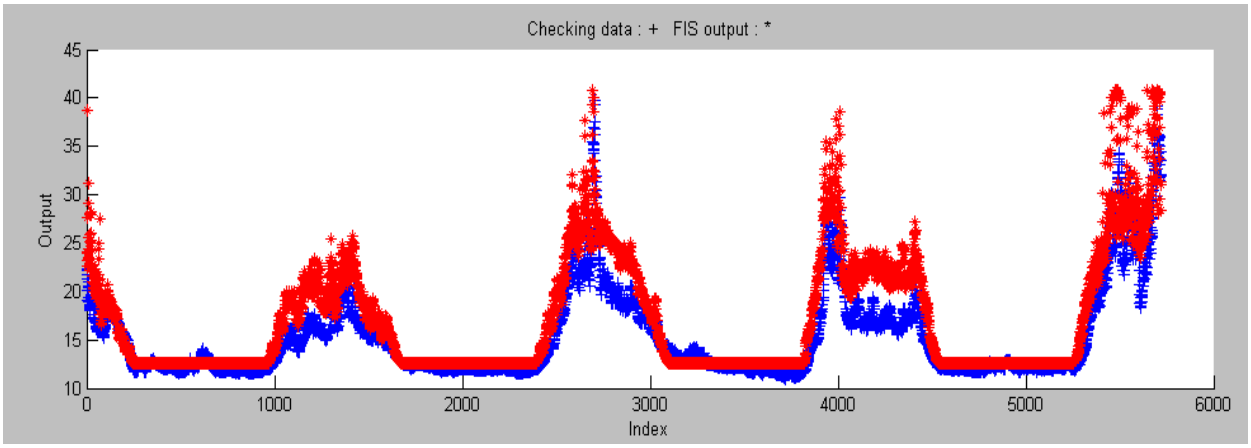


Figura 86. Checking de la red neurofuzzy de Temperatura

e) Modelo de Potencia2

Entrada: Radiación
Salida: Potencia

Tabla 11. Características de la red ANFIS de Potencia

# MUESTRAS data		FM§ entrada	NE††	FM salida	Fis de entrenamiento	Método de optimización	Error	
training	checking						training	checking
10079	5720	GBELL	500	LINEAR	Genfis1	HÍBRIDO	0,83827	0,66325

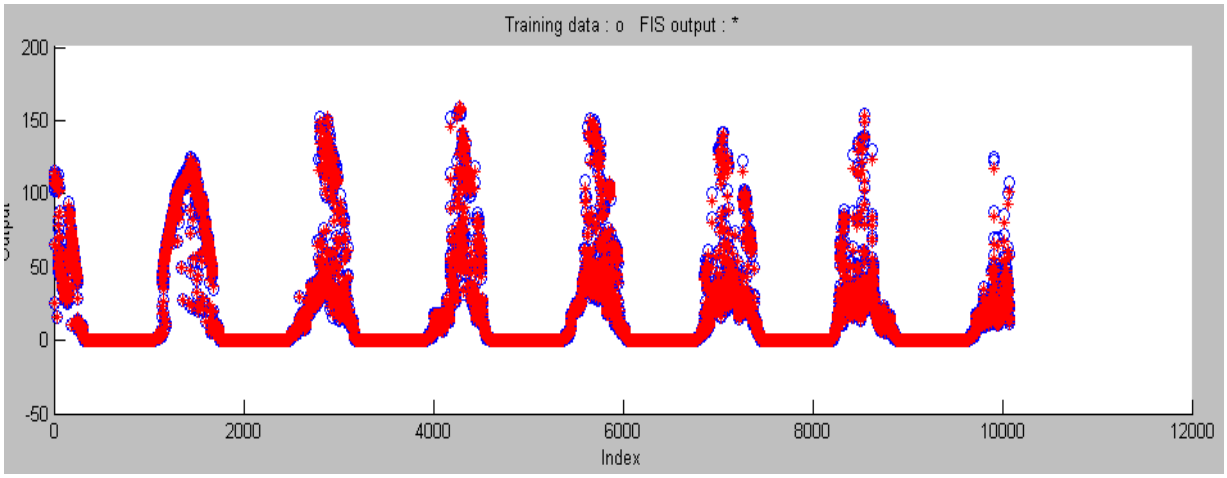


Figura 87. Estructura de la red neurofuzzy de Potencia

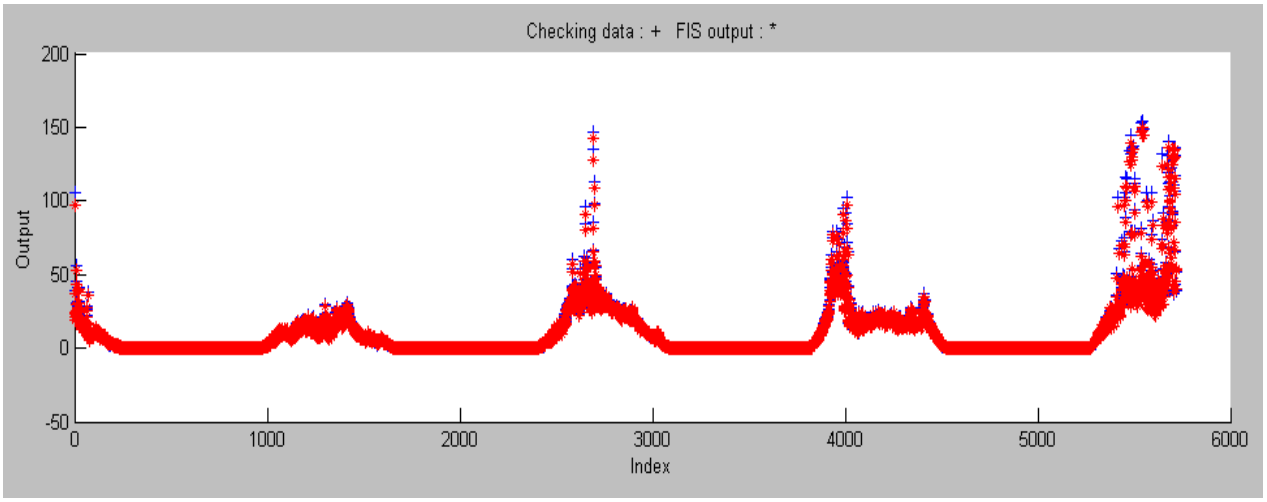


Figura 88. Estructura de la red neurofuzzy de Potencia

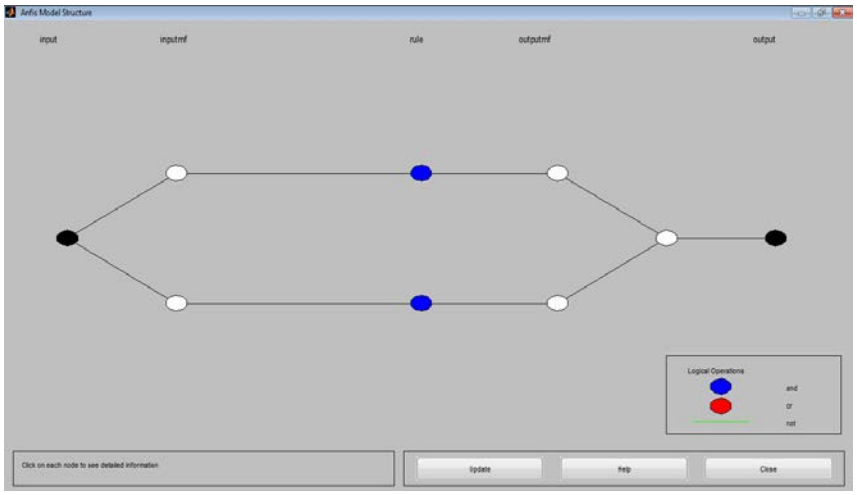


Figura 89. Estructura de la red neurofuzzy de Potencia

A continuación se muestra una tabla de todos los modelos realizados con sus respectivas características y porcentajes de error.

Tabla 12. Modelos generados con redes tipo Anfis

a) Modelo de potencia

# MUESTRAS data		FM§ entrada	NE††	FM salida	Fis de entrenamiento	Método de optimización	Error	
training	checking						training	checking
10079	5720	GBELL	250	LINEAR	Genfis1	HÍBRIDO	0.55927	0.15108
10079	5720	GAUSS	180	LINEAR	Genfis1	HÍBRIDO	0.56325	0.19839
10079	5720	GAUSS2	250	LINEAR	Genfis1	HÍBRIDO	0.57036	0.22989
10079	5720	GBELL	1000	LINEAR	Genfis2	HÍBRIDO	0.60672	0.26858
10079	5720	GAUSS		LINEAR	Genfis2	HÍBRIDO	0.60782	0.26366
10079	5720	GAUSS2	1000	LINEAR	Genfis2	HÍBRIDO	0.60571	0.25093
10079	5720	GBELL	1000	LINEAR	Genfis3	HÍBRIDO	0.59156	0.19752
10079	5720	GAUSS	1000	LINEAR	Genfis3	HÍBRIDO	0.57988	0.17074
10079	5720	GAUSS2	1000	LINEAR	Genfis3	HÍBRIDO	0.58866	0.20454

b) Modelo de Voltaje

# MUESTRAS data		FM§ entrada	NE††	FM salida	Fis de entrenamiento	Método de optimización	Error	
training	checking						training	checking
10079	5720	GBELL	500	LINEAR	Genfis1	HÍBRIDO	2,5314	2,7698
10079	5720	GAUSS	500	LINEAR	Genfis1	HÍBRIDO	2,4244	2,981
10079	5720	GAUSS2	500	LINEAR	Genfis1	HÍBRIDO	2,9306	3,3269
10079	5720	GBELL	500	LINEAR	Genfis2	HÍBRIDO	3,4161	4,1152
10079	5720	GAUSS	500	LINEAR	Genfis2	HÍBRIDO	3,5907	4,24
10079	5720	GAUSS2	500	LINEAR	Genfis2	HÍBRIDO	3,2402	3,9473
10079	5720	GBELL	500	LINEAR	Genfis3	HÍBRIDO	3,9792	4,3293
10079	5720	GAUSS	500	LINEAR	Genfis3	HÍBRIDO	3,6715	3,9701
10079	5720	GAUSS2	500	LINEAR	Genfis3	HÍBRIDO	4,2939	4,5257

c) Modelo de temperatura

# MUESTRAS data		FM§ entrada	NE††	FM salida	Fis de entrenamiento	Método de optimización	Error	
training	checking						training	checking
10079	5720	GBELL	500	LINEAR	Genfis1	HÍBRIDO	3.3102	2.96
10079	5720	GAUSS	500	LINEAR	Genfis1	HÍBRIDO	3.4055	2.7562
10079	5720	GAUSS2	500	LINEAR	Genfis1	HÍBRIDO	3.516	2.59
10079	5720	GBELL	500	LINEAR	Genfis2	HÍBRIDO	3.339	3.1103
10079	5720	GAUSS	500	LINEAR	Genfis2	HÍBRIDO	3.5403	3.198
10079	5720	GAUSS2	500	LINEAR	Genfis2	HÍBRIDO	3.5578	3.1718
10079	5720	GBELL	500	LINEAR	Genfis3	HÍBRIDO	3.3867	2.8193
10079	5720	GAUSS	500	LINEAR	Genfis3	HÍBRIDO	3.3906	2.8659
10079	5720	GAUSS2	500	LINEAR	Genfis3	HÍBRIDO	3.3704	2.8889

d) Modelo de temperatura

# MUESTRAS data		FM§ entrada	NE††	FM salida	Fis de entrenamiento	Método de optimización	Error	
training	checking						training	checking
10079	5720	GBELL	500	LINEAR	Genfis1	HÍBRIDO	0,83827	0,66325

f.2.3.3.2 Resumen de evaluación y simulación del modelo Neuro-Fuzzy tipo ANFIS

Para simular el modelo se utiliza otro conjunto de datos(5720 datos). La función ***evalfis.m*** permite simular los datos arrojados por el sistema neurofuzzy.

Para evaluar el error entre los resultados medidos y los obtenidos por el modelo de la red neurofuzzy tipo ANFIS se utiliza la función ***mse.m***

La siguiente tabla muestra los mejores modelos seleccionados, los cuales han sido simulados y verificados su error con las funciones descritas anteriormente .

Tabla 13. Resumen de los mejores modelos evaluados con la función ***mse.m***

MODELO		FM entrada	FM salida	Error		
Input	**Output**			**Training (10079 datos)**	**Checking (5720 datos)**	***evalfis.m mse.m***
RADIACION TEMPERATURA	POTENCIA	GBELL	Lineal	0.55927	0.15108	0,02283
RADIACION TEMPERATURA	VOLTAJE	GBELL	Lineal	2,5314	2,7698	7.6718
RADIACION TEMPERATURA	INTENSIDAD	GAUSS2	Lineal	1,1996e-8	0.00098885	9,7782E-07
RADIACION	TEMPERATURA	GBELL	Lineal	3.3102	2.96	7.0053
RADIACION	POTENCIA	GBELL	Lineal	0,83827	0,66325	0,4399

f.3 Procedimiento para realizar aplicaciones con Matlab Builder Ex

Para realizar aplicaciones en matlab podemos utilizar el comando ***deploytool*** para iniciar un la contrucción de un proyecto. En este caso se realizará un proyecto utilizando la opción Excel Add-In que utiliza las funciones de Matlab Builder Ex para generar aplicaciones compatibles con Microsoft Excel.Ver figuras 90, 91 y 92

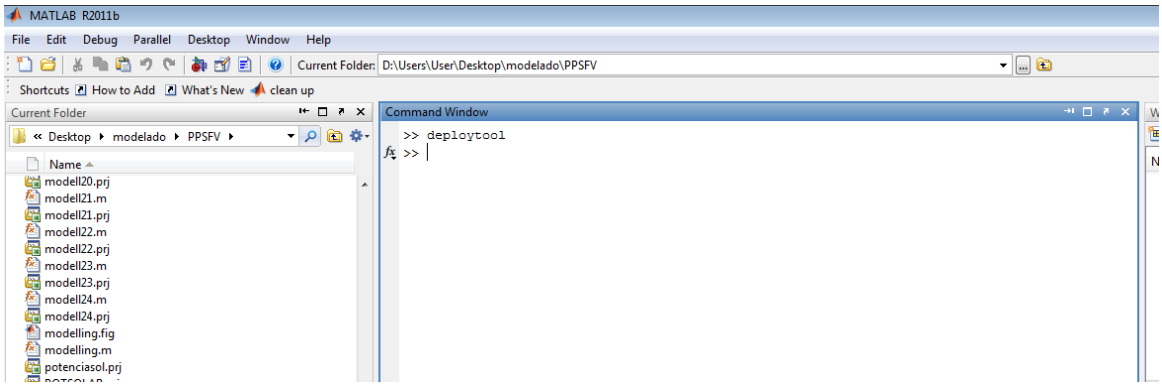


Figura 90. Ventana de command window

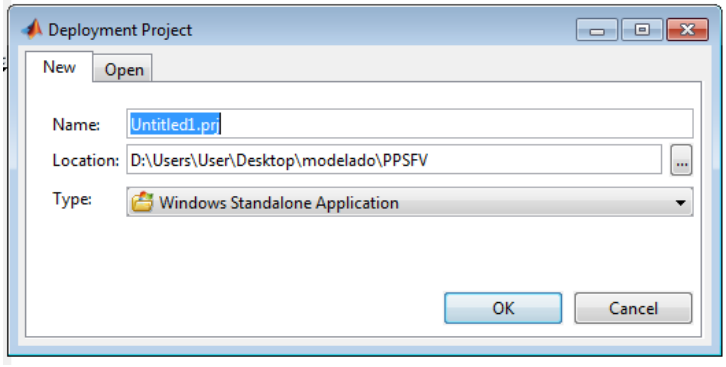


Figura 91. Ventana de Deployment Proyect

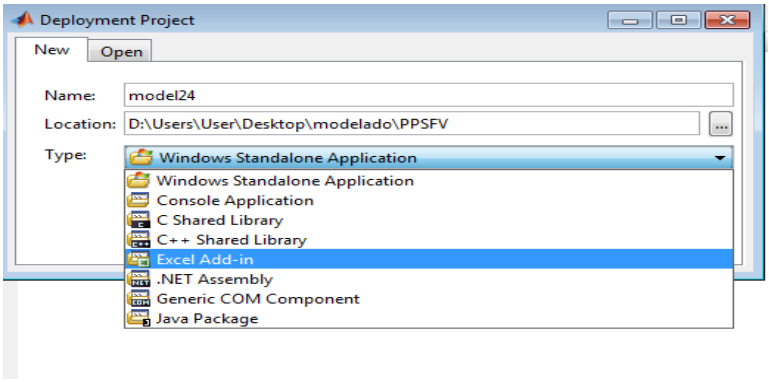


Figura 92. Ventana de Deployment Proyect\Excel Add-in

Podemos ubicar un nombre como se ilustra en la figura 93, para el proyecto que se realice en nuestro caso va a ser ***modell24.***

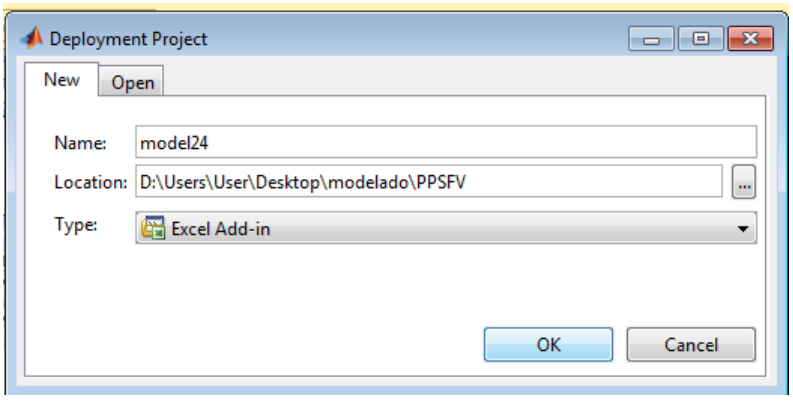


Figura 93. Ventana de Deployment Proyect\Excel Add-in

Seguidamente aparece la ventana de nuestro proyecto en Excel Add-in

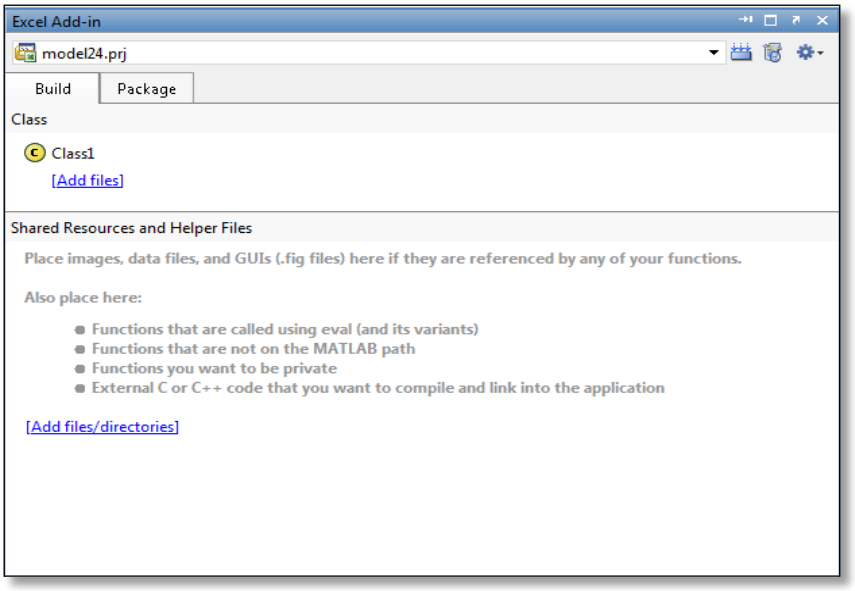


Figura 94. Ventana de Excel Add-in

En esta ventana se carga o se arrastra los archivos, generados por los modelos, en primeramente se arrastra el archivo principal en el primer bloque y en el segundo bloque los modelos y otros archivo secundarios necesarios para ejectarse el archivo principal.

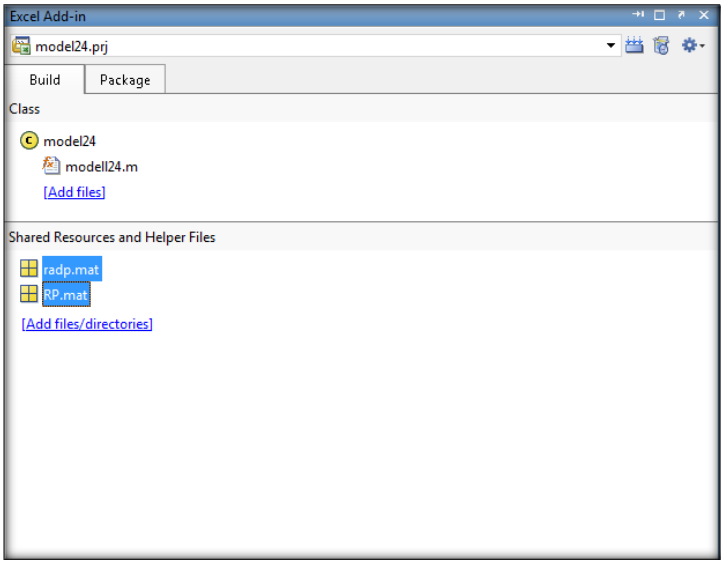


Figura 95. Ventana de Excel Add-in con sus archivos cargados

Ahora se puede dar clic en el botòn de Build  para construir la aplicación, y se empieza a contruir el proyecto como se muestra en la figura 96 y 97.

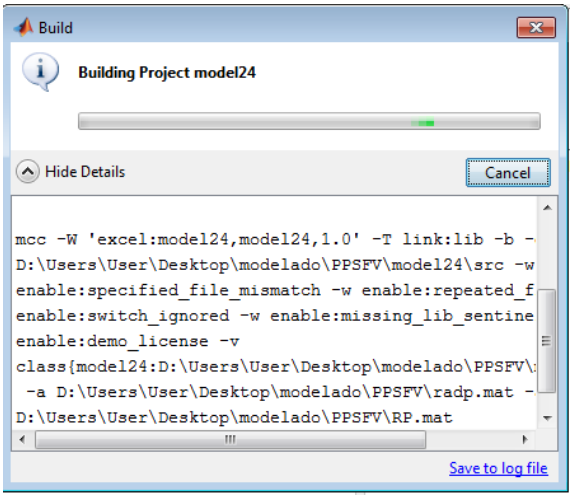


Figura 96.Ventana de Build (construyendo el proyecto)

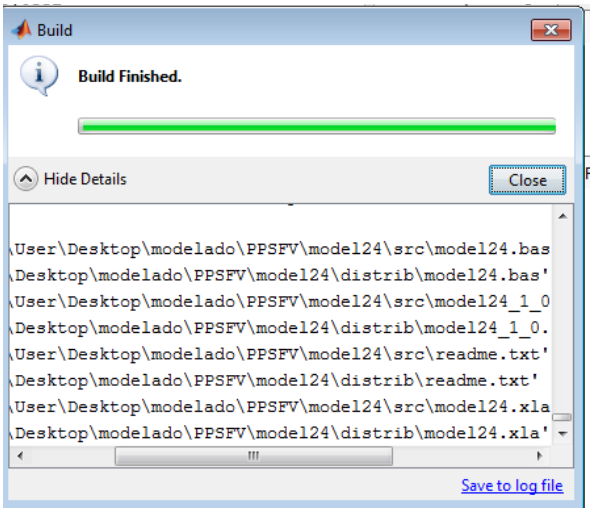


Figura 97. Ventana Build (finalizado el proceso de construcción)

En la pestaña de package se puede agregar el compilador de matlab (mcr) para finalmente presionar el botòn de package para comprimir todo el proyecto en una sola aplicaciòn.

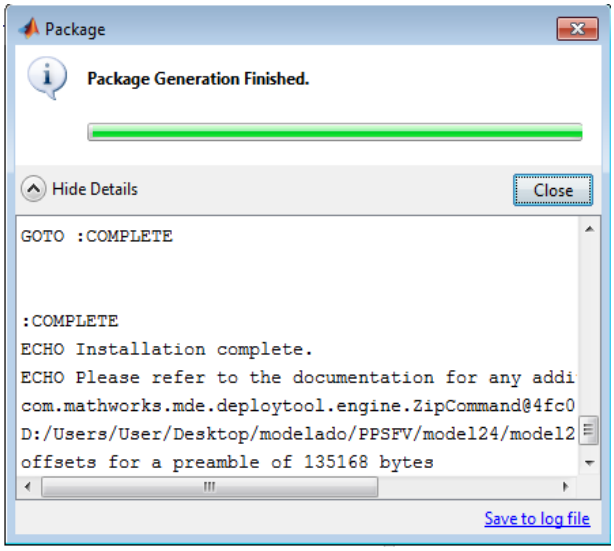


Figura 98. Ventana de Package (proceso finalizado)

Finalmente tenemos nuestra aplicación para poder instalarla como complemento de Excel.Ver figura 99.

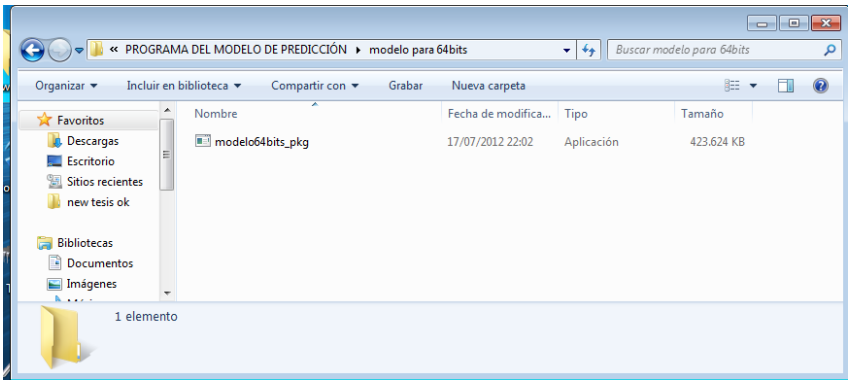


Figura 99. Ventana de la aplicaciòn creada por Matlab Builder Ex

f.4 Procedimiento implementar aplicaciones de Matlab Builder Ex a usuarios finales

Este procedimiento se lo realiza solo una vez para poder utilizar los modelos creados en Matlab y sin necesidad de tener instaladoMatlab se puede utilizarlos desde Excel.

f.4.1 Instalando la aplicación para poder utilizar el modelo en una hoja de Excel sin tener instalado Matlab.

Ejecutamos la aplicación que se encuentra en la carpeta:

METODOLOGÍA\EXCELMODELS\PROGRAMA DEL MODELO DE PREDICCIÓN\modelo para 64bits.

Aquí se selecciona la arquitectura de su pc ya sea de 32bits o 64bits. En la figura 100 se ha elegido la aplicación de 64 bits.

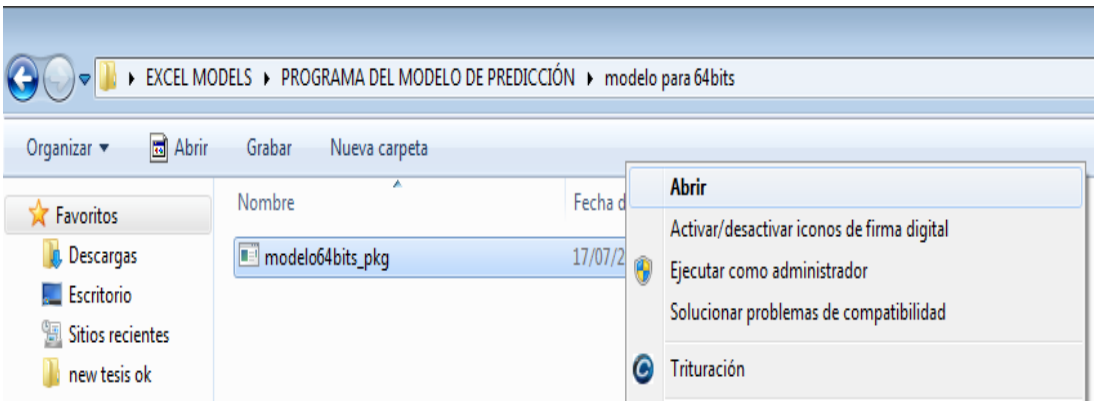


Figura 100. Aplicación de 64 bits ejecutando

Luego aparece una pantalla en la que prepara el instalador como muestra la figura 101**.**

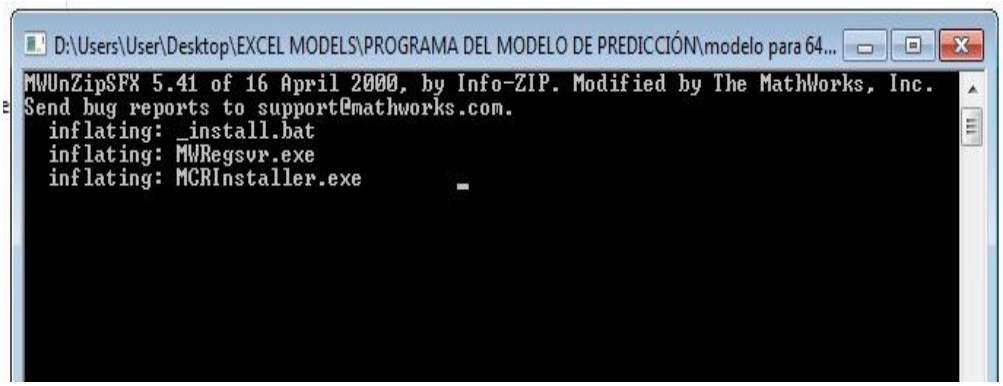


Figura 101. Pantalla de preparación de la instalación

Seguidamente el programa descomprime los archivos.

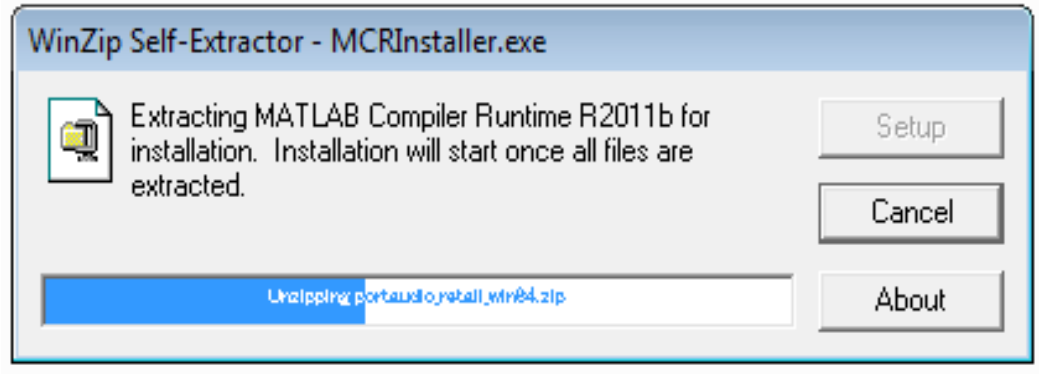


Figura 102. Descomprime el MCRInstaller.exe

A continuación aparece la pantalla de instalación del compilador de Matlab. Como muestra la figura 103, aquí se debe dar clic en *Next*.

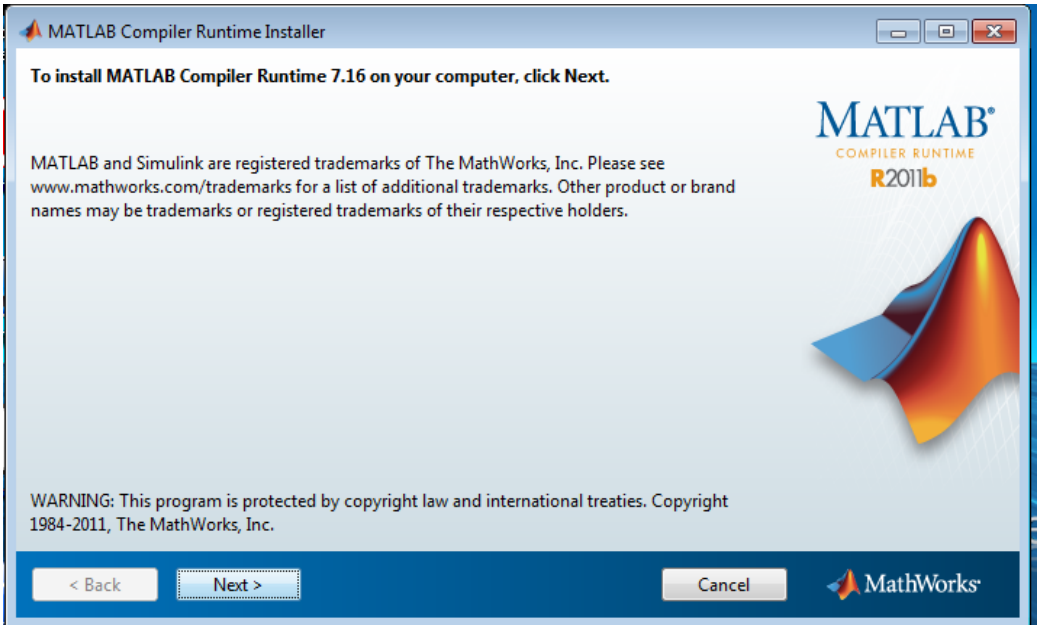


Figura 103. Instalación del compilador de Matlab

Clic en ***install.***

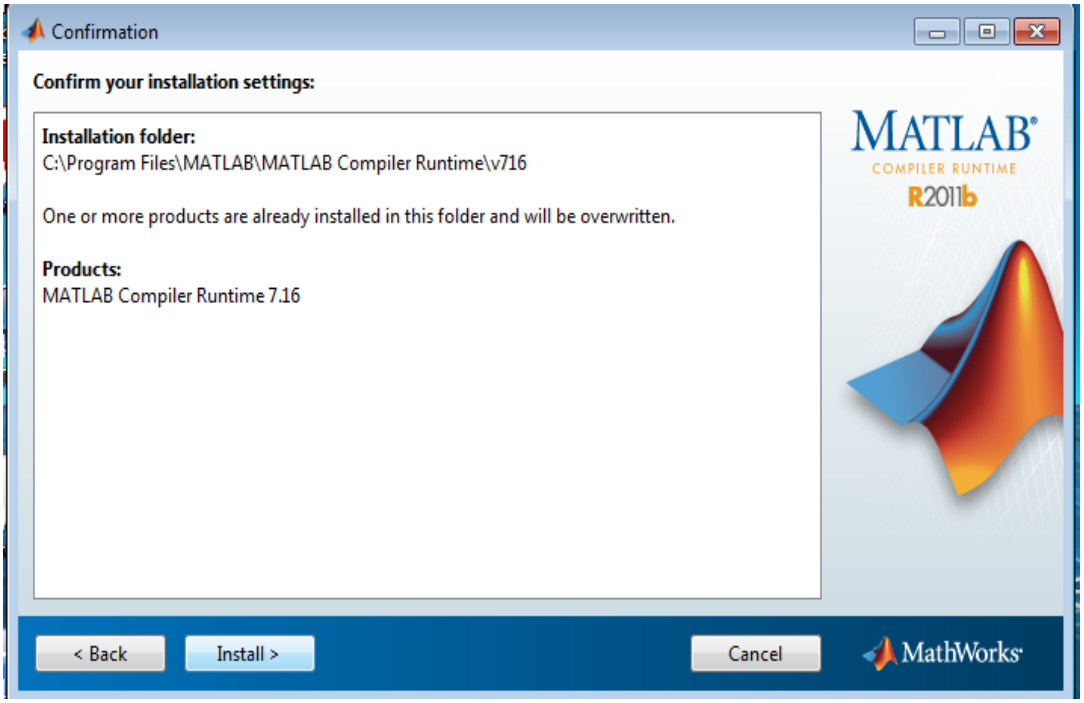


Figura 104. Instalando compilador (mcr)

Clic en ***finish***

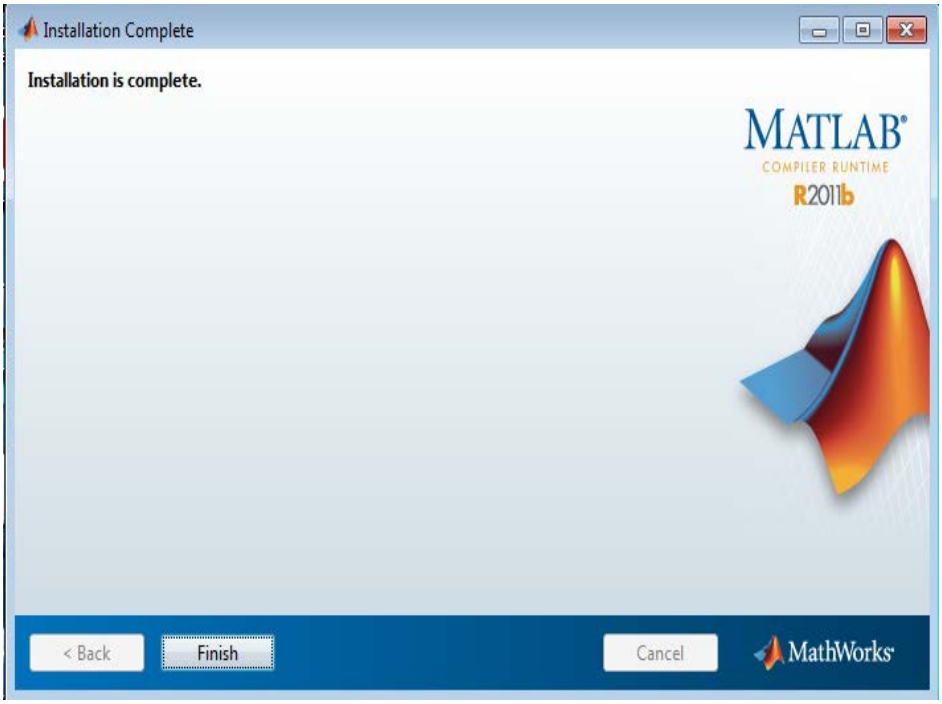


Figura 105. Finalización de la instalación de la aplicación.

Una vez instalada la aplicación se debe configurar la hoja de excel que viene personalizada en la carpeta de instalación ***EXCEL MODEL***, se puede usar el archivo personalizado de excel incluido en la carperta del programa se llama ***Potencia panel solar Exmork de 100W*** como se muestra en la figura 106.

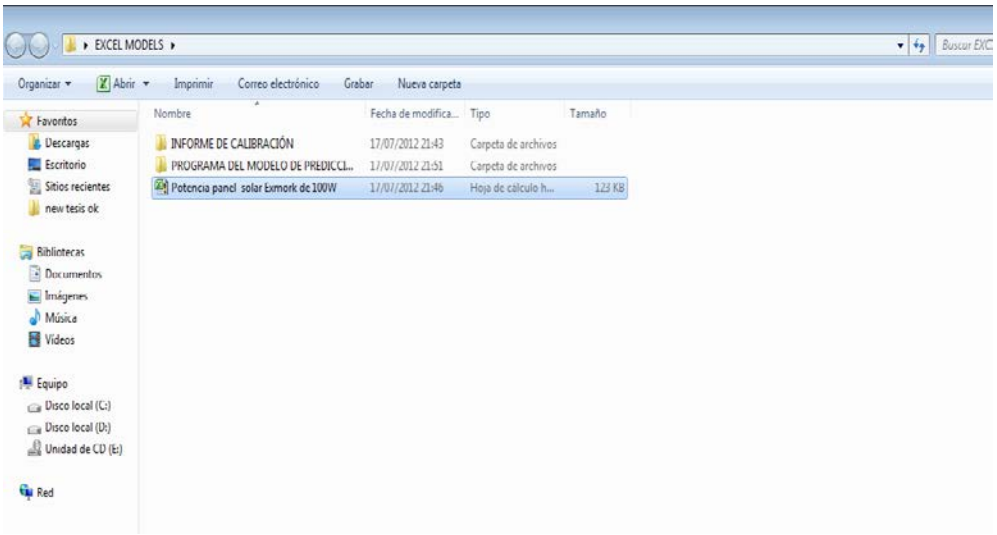


Figura 106. Ubicación del archivo de excel personalizado

A continuación se configura las ***opciones*** en el menú ***Archivo*** de la hoja de excel.

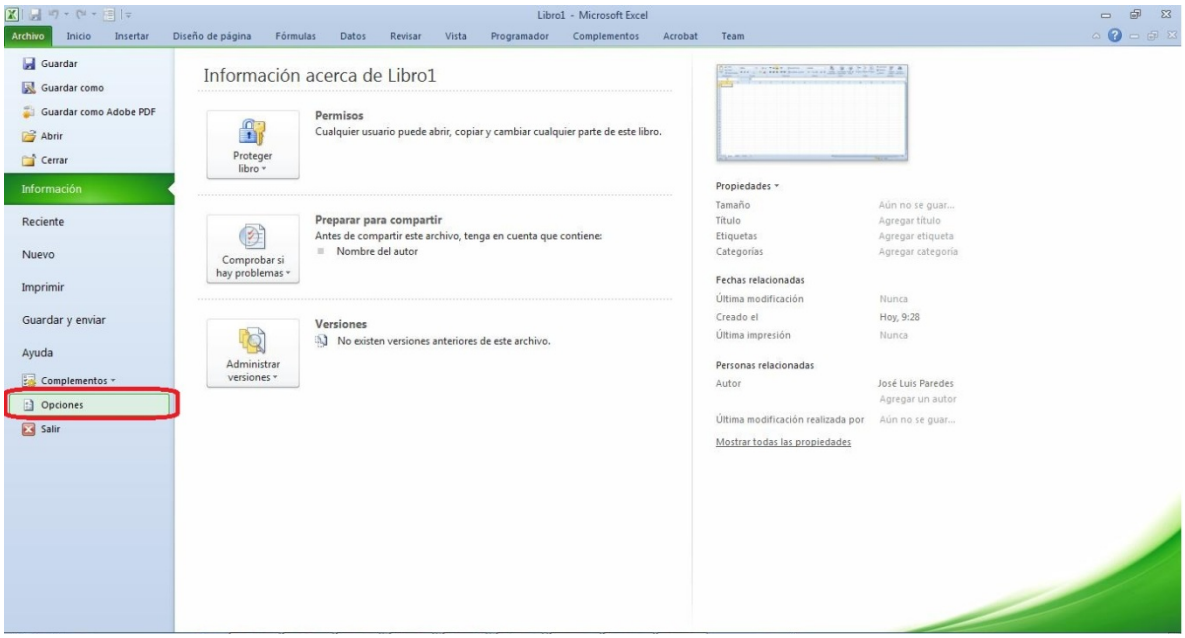


Figura 107. Configuración de las opciones del menú archivo de excel.

Clic en ***Centro de confianza.***

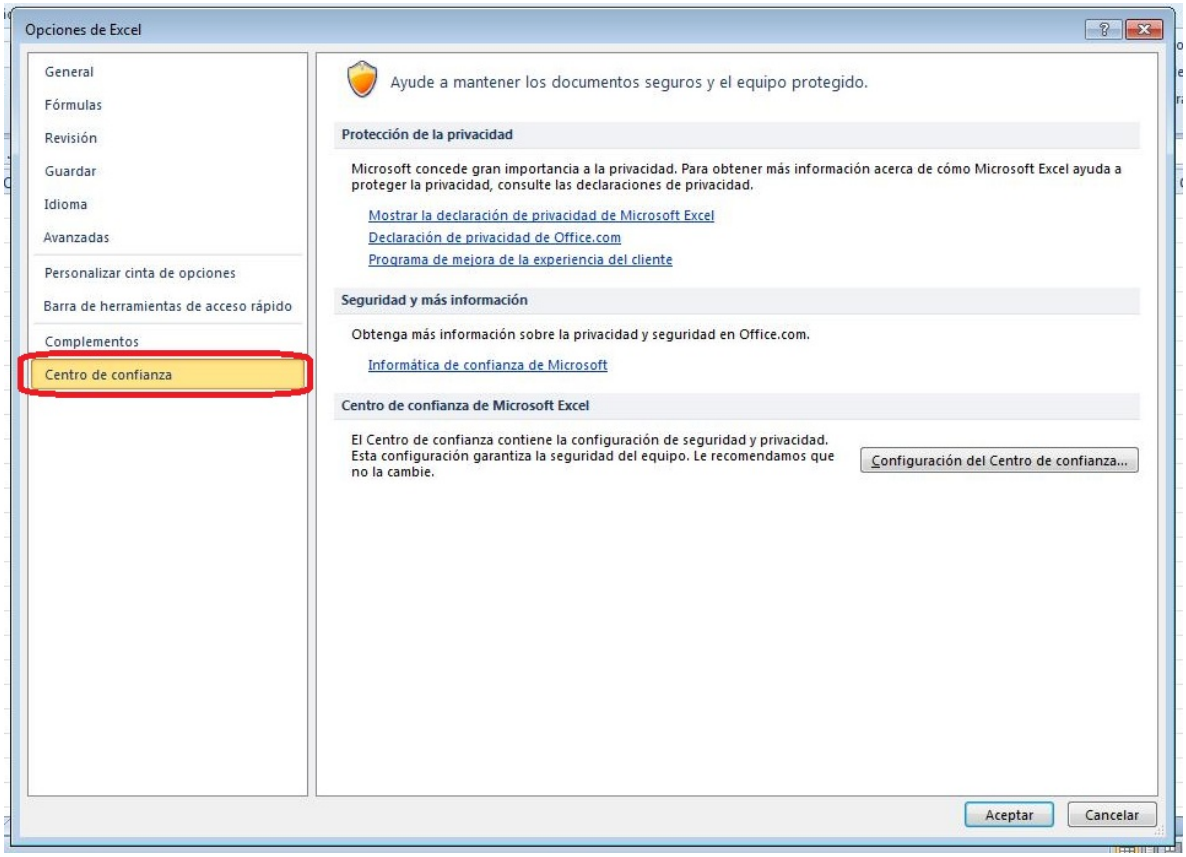


Figura 108. Configuración de las opciones de excel.

Clic en ***configuración de macros*** y activamos la opción ***Configurar en el acceso al modelo de objeto de los proyectos VBA*** como se ilustra en la figura.

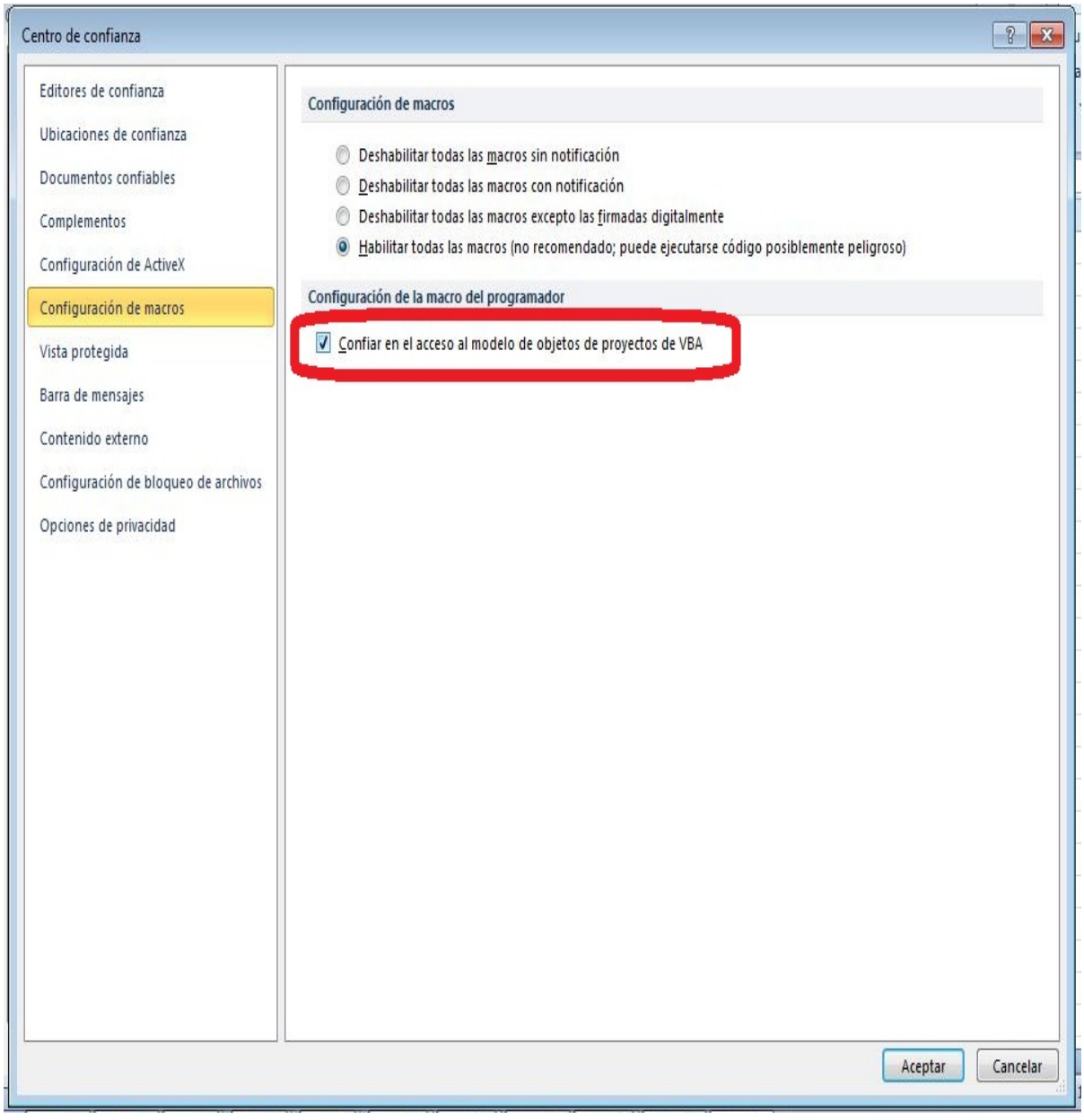


Figura 109. Configuración de Centro de confianza de Excel/ configuración de macros.

También se debe configurar los complementos para ello damos clic en ***complementos*** y luego en ***complementos de Excel*** .como se muestra en la figura 110.

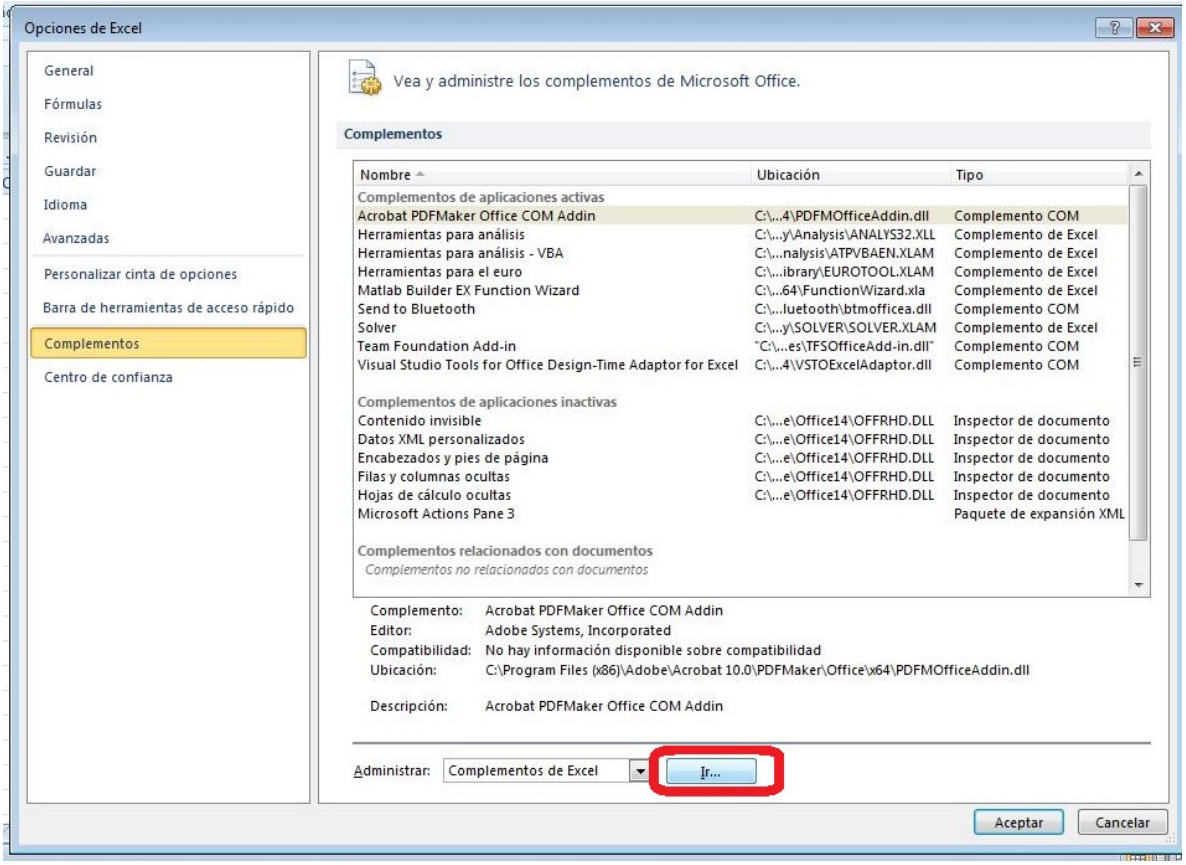


Figura 110. Configuración de Complementos de Excel

Luego aparece una ventana de complementos damos clic en examinar.

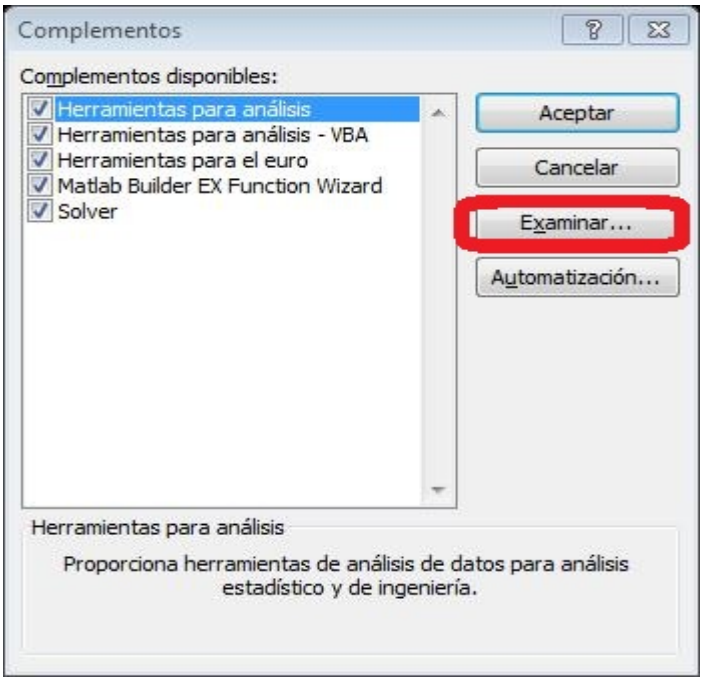


Figura 111. Configuración de complementos

Buscamos la dirección que está instalado el compilador por defecto la ubicación es:
C:\Program Files\MATLAB\MATLAB CompilerRuntime\v716\toolbox\matlabxl\matlabxl\win64

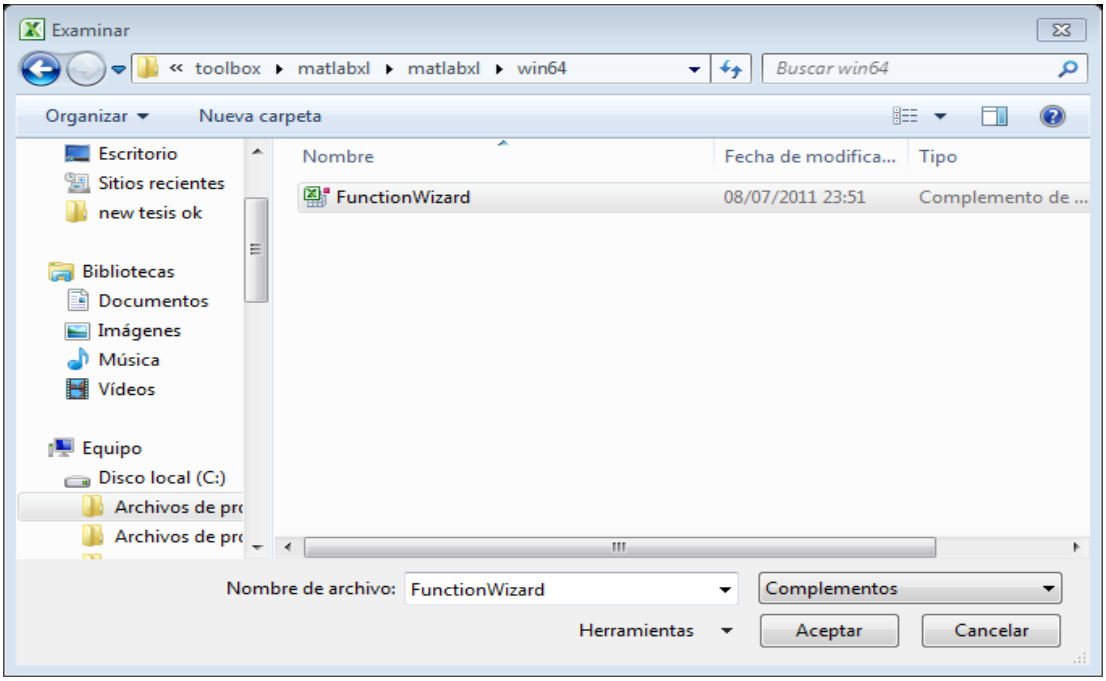


Figura 112. Configuración del directorio del compilador.

Una vez realizado estos pasos ya estamos listos para configurar las funciones que tiene la aplicación instalada, es decir la función de los modelos de redes neuronales y neuro-fuzzy.

Damos clic en la pestaña Complementos de la barra de menús de Excel, como muestra la figura 113.

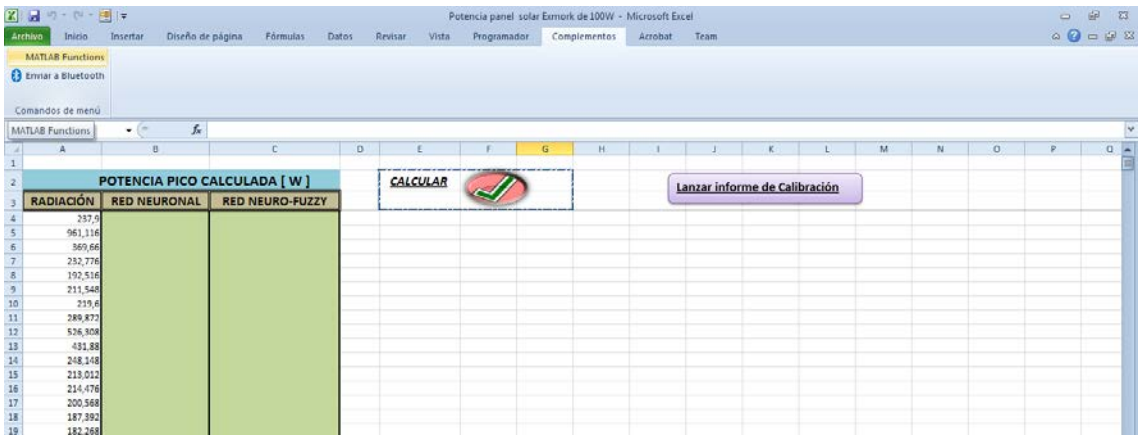


Figura 113. Configuración de la barra de menús Complementos

Aquí aparece una página de inicio para incorporar componentes de Matlab Builder Ex.

Clic en ok como muestra la figura siguiente.

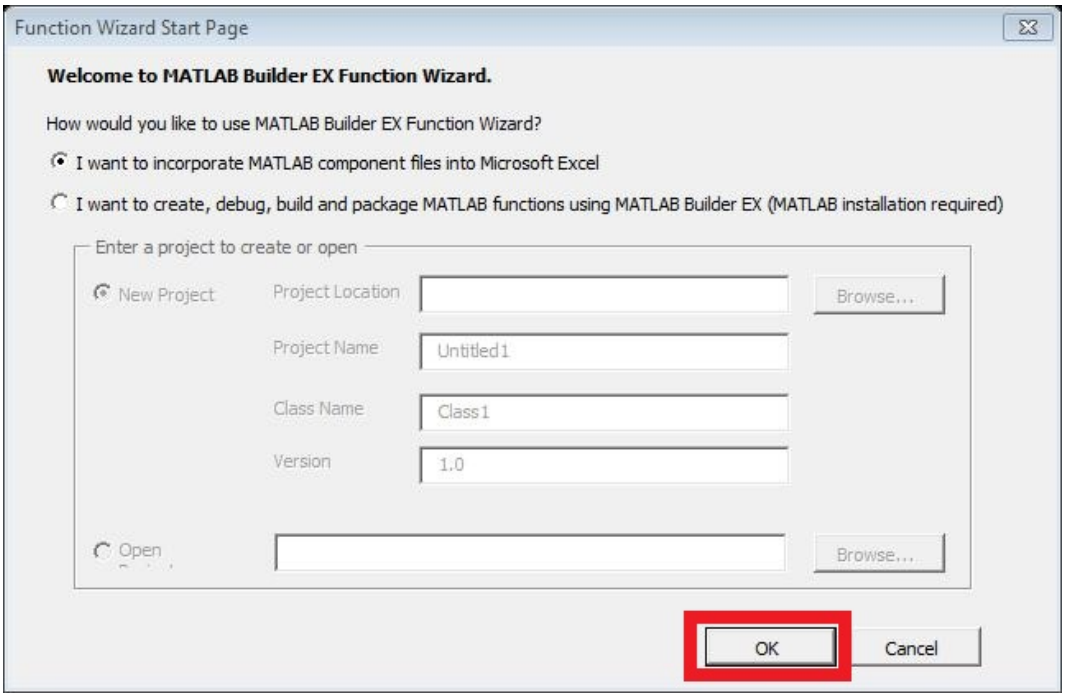


Figura 114. Configuración de la ventana de Matlab Builder Ex.

Clic en Add.

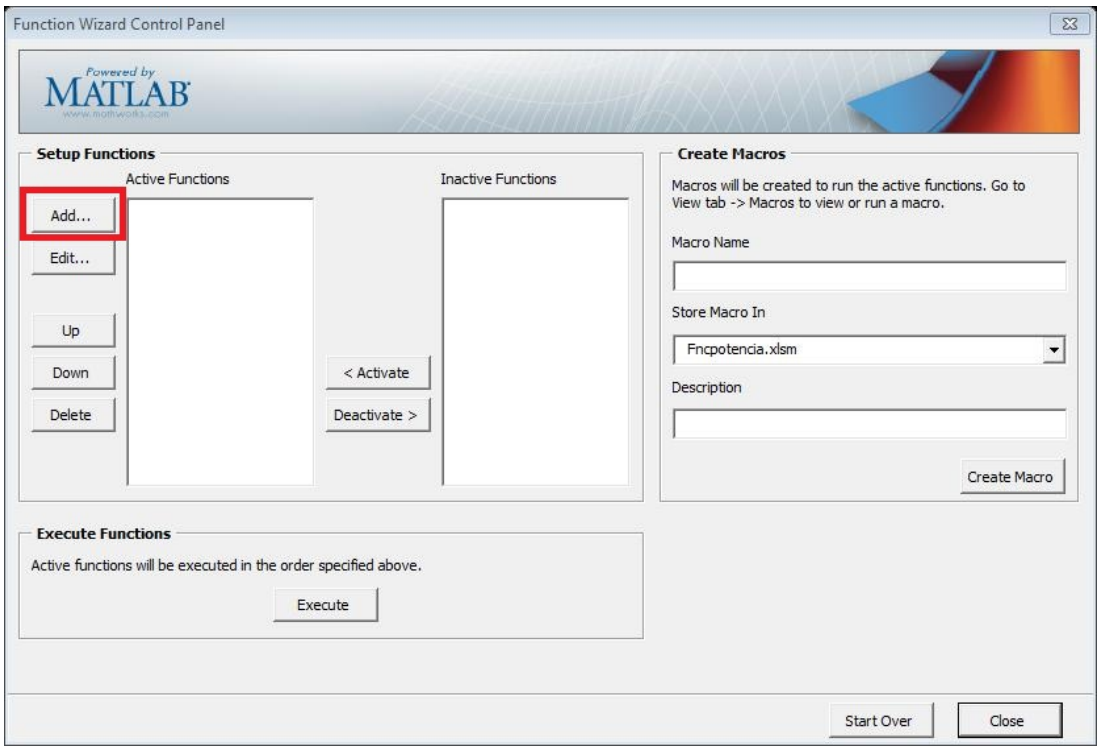


Figura 115. Control de panel de funciones de Matlab Builder Ex

-Seleccionamos el componente ***modell 24 1.0***

-Clic en ***Add.*** Como se muestra en la figura 116.

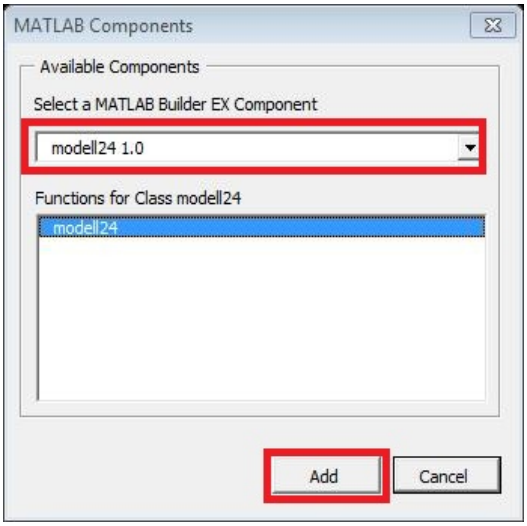


Figura 116. Selección de componentes de Matlab Builder Ex

Seguidamente se selecciona las entradas y salidas de la función y se selecciona el rango en la hoja de Excel. Para la input RAD se selecciona la columna de RADIACIÓN de nuestra hoja de cálculo y para output se selecciona las dos columnas de los 2 modelos de la hoja de cálculo, como se muestra en las figuras 117,118, 119 y 120.

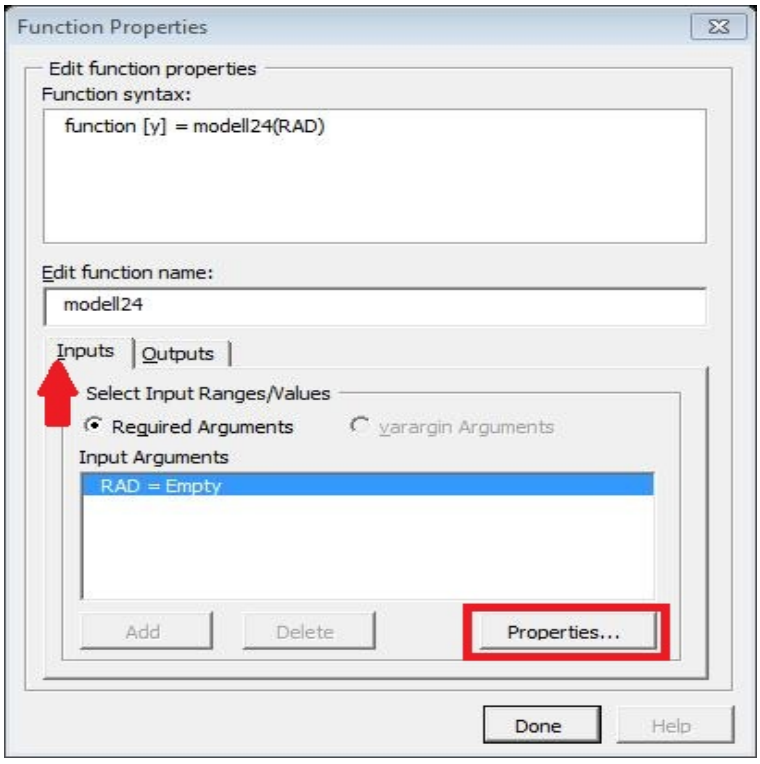


Figura 117. Configuración de las entradas de los modelos

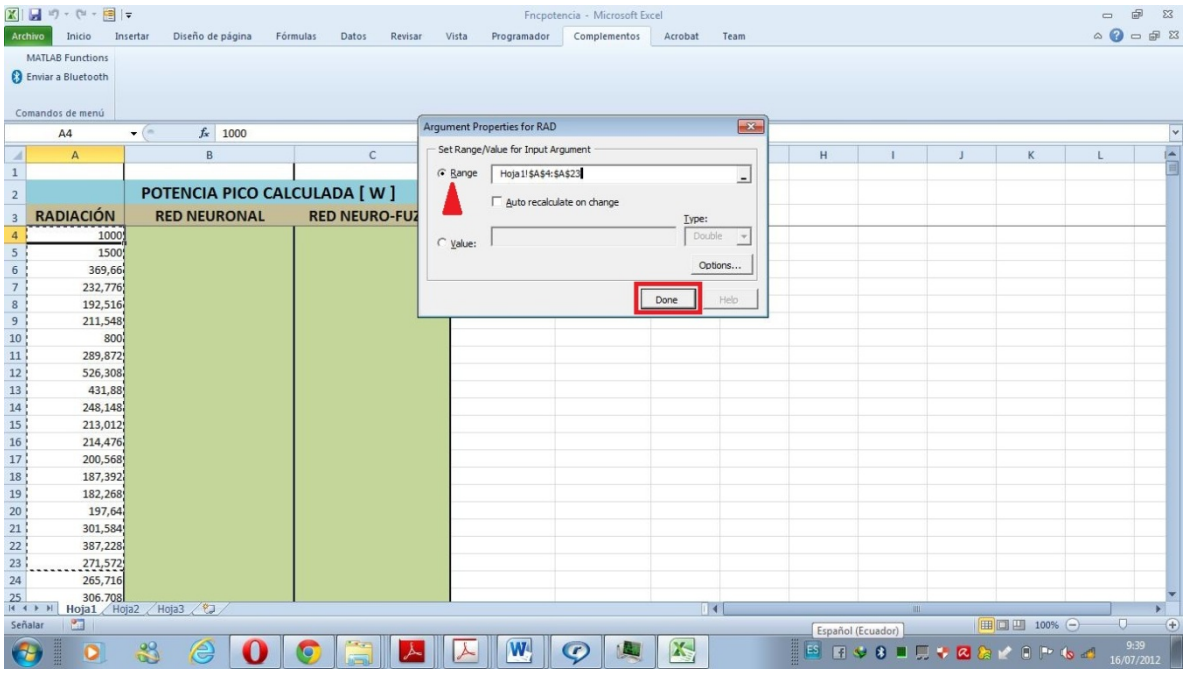


Figura 118. Selección del rango en la hoja de Excel.

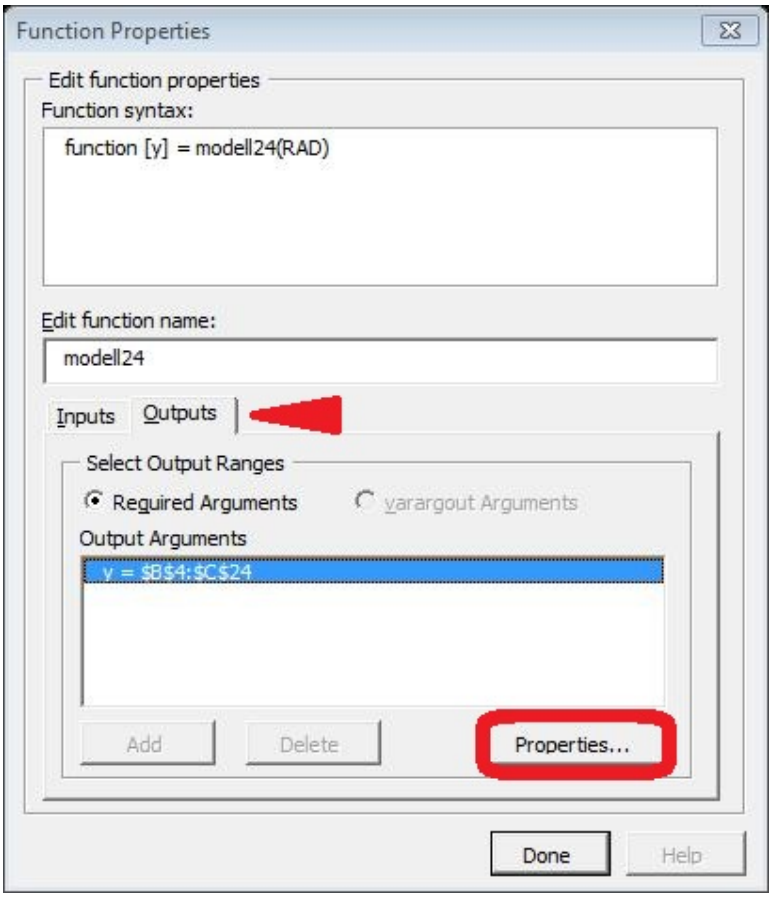


Figura 119. Configuración de las salidas de los modelos realizados

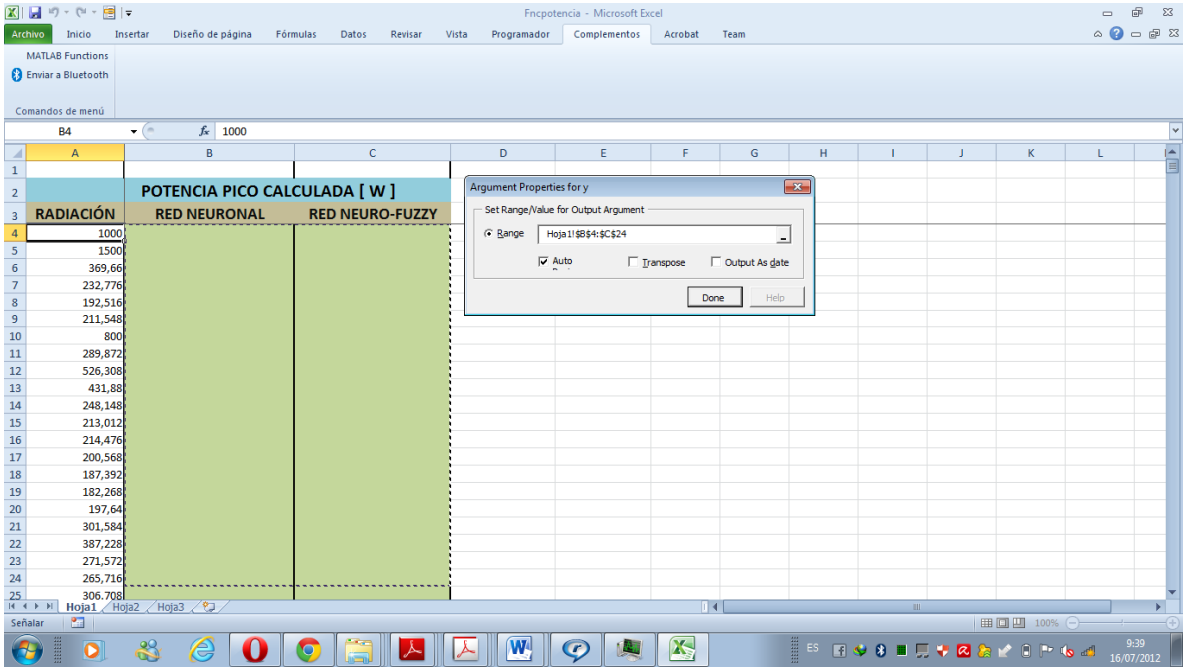


Figura 120. Selección del rango de las salidas de los modelos en la hoja de Excel.

Seguidamente ponemos un nombre ***potencia*** para crear una macro, clic en ***create macro*** como muestra la figura 121 y cerramos la ventana en ***close.***

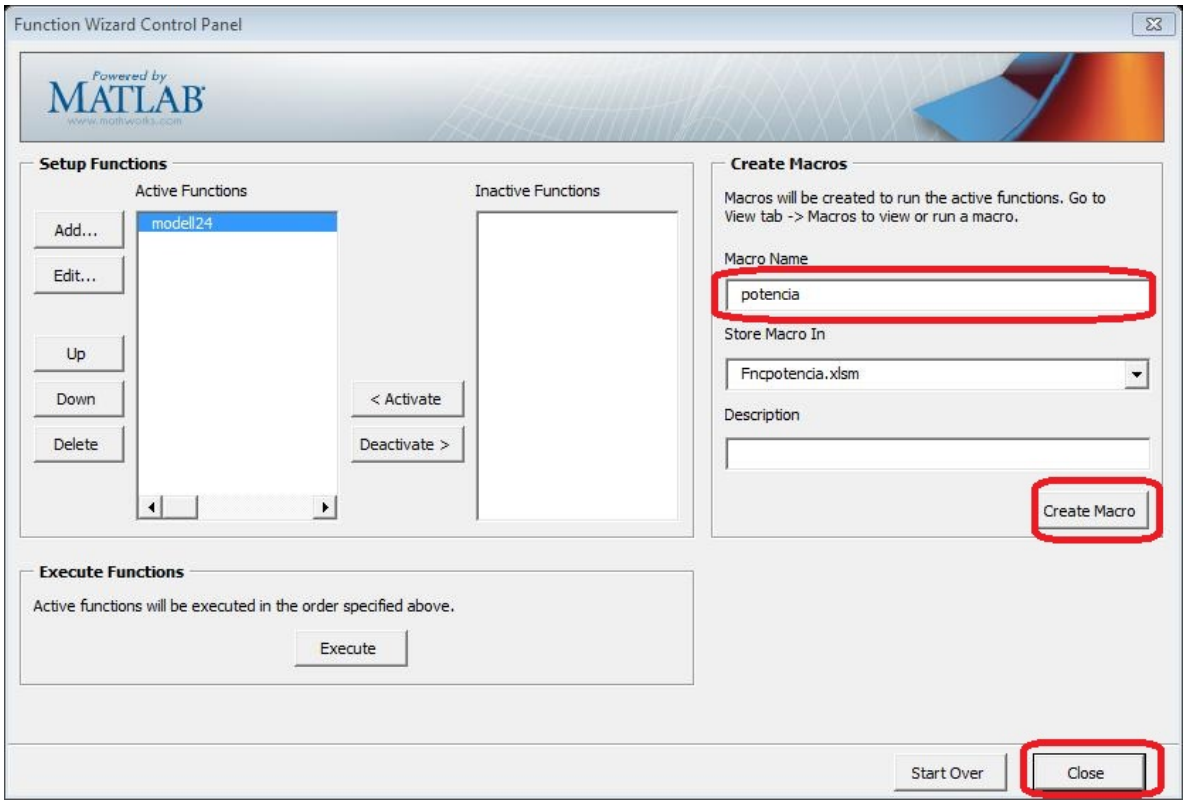


Figura 121. Crear macro para ejecutar los modelos

Finalmente asignamos la macro creada ***potencia*** **.**Véase *figura 122.*

-Clic derecho en el botón 

-Asignar macro y elegimos el nombre ***potencia***.

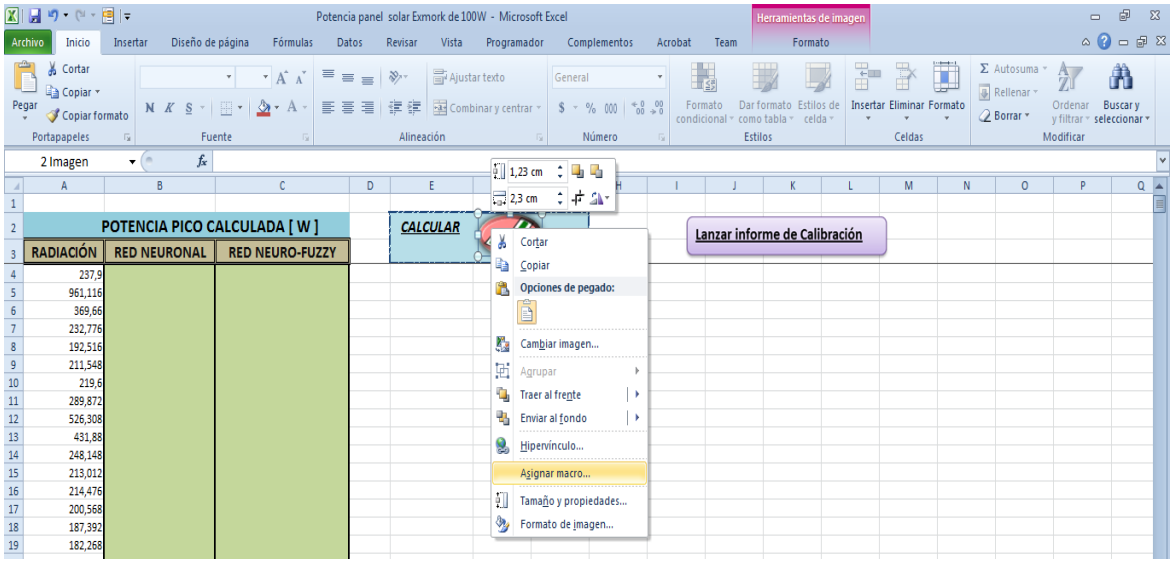


Figura 122. Pantalla para asignar macro.

Está todo listo ya tenemos configurada nuestra hoja de Excel, podemos guardar (Ctrl + G) los cambios realizados para ejecutar los modelos creados con redes neuronales y con redes neurofuzzy.

Para ello damos clic en ***calcular*** y se generan los modelos y su gráfica respectiva que se muestra en la figura 123.

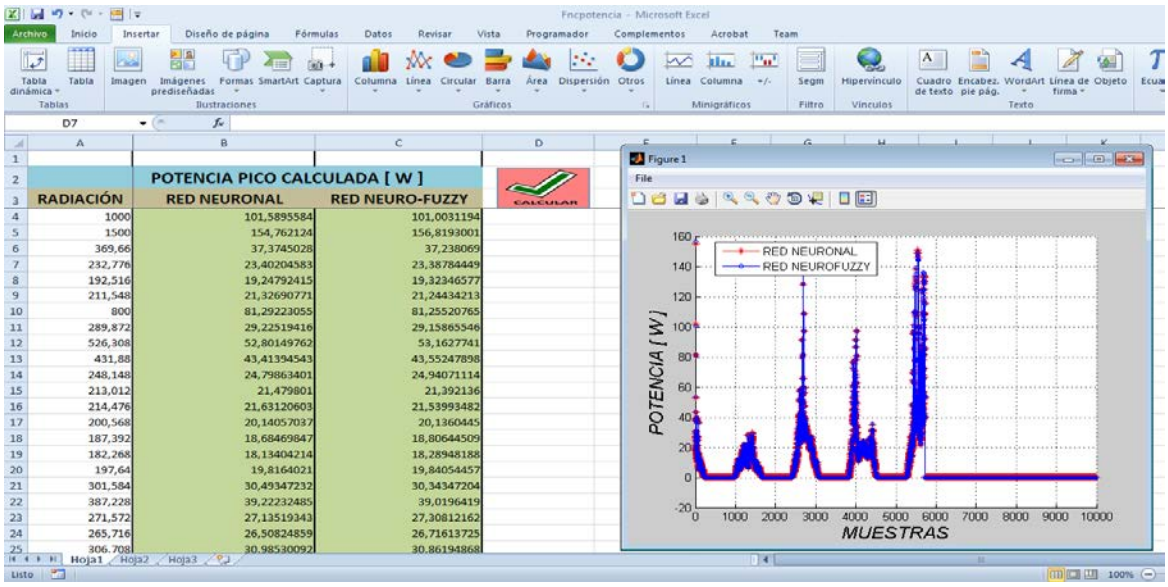


Figura 123 .Pantalla de ejecución de los modelos desarrollados.

Adicionalmente se puede dar clic en el botón Lanzar informe de Calibración Para ver un informe sobre la construcción de los modelos. Esto se ilustra en la figura 124.

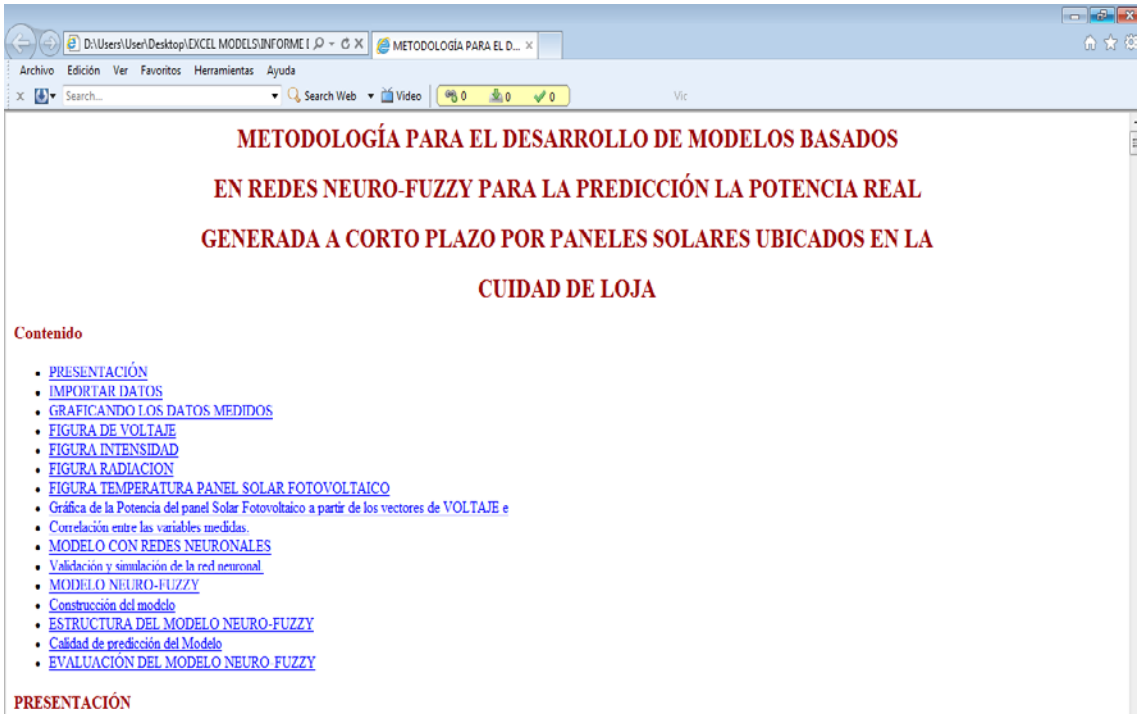


Figura 124. Pantalla del informe de calibración

g.- DISCUSIÓN

En resumen, dentro de este trabajo investigativo se realiza una metodología a seguir para elaborar modelos utilizando redes neuronales y redes neurofuzzy tipo ANFIS, los mismos que son capaces de determinar en base de la Radiación Solar, la potencia real producida por un paneles fotovoltaicos policristalinos marca Exmork de 100W, aportando con ello a una solución para realizar la estimación de la generación real de un sistema fotovoltaico, implementado en las condiciones de irradiación de la ciudad de Loja.

El proyecto de investigación se lo realizó en la ciudad de Loja bajo sus condiciones ambientales, la adquisición de datos necesarios para generar los modelos finales se realizó con el equipo que cuenta la Unidad de Eficiencia Energética del AEIRNNR, se obtuvo las variables de radiación Solar, temperatura del panel así como la intensidad y voltaje del panel solar en circuito abierto, es decir sin estar conectado a la carga o a algún equipo adicional.

Para realizar los modelos se utilizó dos conjuntos de datos con un intervalo de tiempo de muestreo de un minuto, el primer conjunto de datos realizado en una semana (10079 datos) se utiliza para el entrenamiento de los modelos y el segundo grupo de datos para la validación de los modelos (5720 datos).

De acuerdo al análisis de los datos obtenidos se puede evidenciar que existe una relación directa entre radiación e intensidad del panel. Por estas consideraciones se realizó varios modelos utilizando redes neuronales y redes neurofuzzy utilizando varias combinaciones de entrada y salida de los modelos como se muestra en resumen en la tabla 14 y 15 que se muestra a continuación:

Tabla 14. Modelos con redes neuronales que tienen mejor porcentaje de error

Tipo de Red Neuronal: **FEED FORWARD BACKPROPAGATION**
Función de Entrenamiento: **TRAINLM**
Función de Aprendizaje: **LEARNGDM**

Nombre de los modelos	**INPUT**	**TARGET**	**MSE RNA'S**	
			Mse train	**Mse validación**
REDV3	RADIACION TEMPERATURA	VOLTAJE	0.46848	0.761
REDI2	RADIACION TEMPERATURA	INTENSIDAD	3.3711E-28	9.7782E-07
RTP	RADIACION TEMPERATURA	POTENCIA	0.32031	0.0412
RT	RADIACION	TEMPERATURA	10.755	8.81
RP	RADIACION	POTENCIA	0.65521	0,4505

Tabla 15. Modelos con redes neurofuzzy tipo Anfis que tienen mejor porcentaje de error

Nombre del modelo	**Entradas y salidas utilizadas**		**FM entrada**	**FM salida**	**Error**		
	Input	**Output**			**Training (10079 datos)**	**Checking (5720 datos)**	***evalfis.m mse.m***
RTP1gbell	RADIACION TEMPERATURA	POTENCIA	GBELL	Lineal	0.55927	0.15108	0,02283
RTV1gbell	RADIACION TEMPERATURA	VOLTAJE	GBELL	Lineal	2,5314	2,7698	7.6718
RTI1gauss2	RADIACION TEMPERATURA	INTENSIDAD	GAUSS2	Lineal	1,1996e-8	0.00098885	9,7782E-07
RT1gbell	RADIACION	TEMPERATURA	GBELL	Lineal	3.3102	2.96	7.0053
radp	RADIACION	POTENCIA	GBELL	Lineal	0,83827	0,66325	0,4399

En estas tablas muestran un resumen de los modelos que tienen el menor porcentaje de error según las entradas y salidas asignadas.

Es importante señalar que para la creación de los modelos con redes neuronales se utilizó la interfaz gráfica de funciones ***nntool*** de ***Neural Network Toolbox*** implementado en Matlab, las funciones de membresía de entrada y salida para una red tipo feed forward backpropagation utilizada se especifican en la tabla 14, cuya utilización se detalla en la metodología para creación de redes neuronales artificiales realizada.

Para la validación del modelo con redes neuronales se utiliza la función ***sim.m*** que simula la red creada y para evaluar su error se utiliza la función ***mse.m*** que proporciona el rendimiento de la red de acuerdo a los mínimos cuadrados.

Por otro lado para realizar el entrenamiento de los modelos con redes neurofuzzy se utilizó la interfaz gráfica ***anfisedit*** de ***Fuzzy Logic Toolbok.*** En la generación del sistema de inferencia difusa (FIS) se utiliza tres tipos de técnicas avanzadas de inferencia difusa que son: genfis1, genfis2 y genfis3, dando como mejor resultado al error de entrenamiento y de prueba el genfis 1 en todos los modelos con este tipo de redes Anfis.

En la evaluación de los modelos de la red tipo Anfis se utilizó la función ***evalfis.m*** que permite simular los resultados que arroja la red e igualmente para determinar el error se utilizó la función ***mse.m***.

Cabe señalar que todas estas funciones descritas se tomaron basadas en la bibliografía consultada y en los webinars de Matlab, es decir los tutoriales que presenta los desarrolladores de Matlab y que utilizan este tipo de funciones para evaluar los modelos con redes neuronales y redes neurofuzzy tipo ANFIS.

Una vez realizado el análisis de los mejores modelos se elige el modelo de cada red que tenga una variable de entrada es decir la Radiación Solar con el objeto de predecir la potencia del panel solar ingresando solamente la radiación Solar, por lo tanto se elige 2 modelos, un modelo con redes neurofuzzy y el otro con redes neuronales. Las características de los modelos se presentan en la tabla 16 y 17.

Tabla 16. Resultados del modelo con redes neurofuzzy.

MODELO CON REDES NEURONALES ARTIFICIALES *RP* Tipo de Red Neuronal: **FEED FORWARD BACKPROPAGATION** Función de Entrenamiento: **TRAINLM** Función de Aprendizaje**: LEARNGDM**			
INPUT	**TARGET**	**MSE RNA'S**	
		MSE train	**MSE validación**
RADIACION	POTENCIA	0,65521	0,4505

Tabla 17. Resultados del modelo con redes neuronales artificiales.

MODELO CON REDES NEUROFUZZY TIPO ANFIS *radp*						
		FM entrada (2)	**FM salida**	**Error**		
Input	**Output**			**Training (10079 datos)**	**Checking (5720 datos)**	**EVALFIS MSE**
RADIACION	POTENCIA	GBELL	Lineal	0,83827	0,66325	0,4399

De los resultados que arrojan las tablas 16 y 17 se desprende que el modelo RP que utiliza redes neuronales tiene un porcentaje de error de 0,45% y el modelo radp que pertenece a la red tipo Anfis tiene un porcentaje de error de 0,44%, por lo tanto se puede evidenciar que ambos modelos seleccionados ofrecen un error de predicción aceptable y con alto grado de precisión.

Adicionalmente en el presente trabajo investigativo, para la ejecución de los dos modelos se crea una aplicación que genera un complemento en una hoja de Excel para ser ejecutado sin necesidad de instalar el programa Matlab, también se realiza un formato de hoja de cálculo para configurar y generar estos modelos. Esta hoja de cálculo de Excel muestra un informe de calibración, es decir un informe de cómo se realizó el análisis de los datos, la construcción, simulación y validación de estos modelos tanto para redes neuronales como para el modelo neurofuzzy.

Finalmente cabe mencionar que los modelos se pueden compartir con cualquier usuario que no tenga conocimientos de programación del programa Matlab ya que los modelos se ejecutan en una interfaz fácil de utilizar como es Excel, por lo tanto puede ser utilizado para verificar en qué lugar es factible y/o rentable la instalación de paneles solares de este tipo y así contribuye al desarrollo e implementación de energías renovables mediante paneles solares fotovoltaicos. Lo cual permitiría disminuir los impactos ambientales que producen las energías no renovables, beneficiando de esta manera a la sociedad lojana particularmente.

h.- CONCLUSIONES

- El presente trabajo investigativo ha cumplido con su objetivo principal, el cual es desarrollar modelos temporales basados en redes neurofuzzy capaces de determinar en base a la radiación y temperatura ambiente las variables de voltaje, corriente y potencia de salida en condiciones reales de operación del panel. Esto se lo demuestra en la discusión realizada que se muestra en las tablas 14 y 15, que se refiere a los modelos que permiten calcular en tiempo real la potencia generada por los paneles solares fotovoltaicos policristalinos marca Exmork de 100W, por lo tanto, se demuestra que los modelos con redes neuronales artificiales y redes neurofuzzy tipo Anfis modelan la potencia del panel obteniendo resultados precisos y se los puede utilizar en una interfaz fácil como lo es Excel.

- Se determina que la variable de radiación solar permite calcular con un alto grado de precisión y confiabilidad la potencia del panel solar por lo cual se eligen los mejores modelos que tienen como variable de entrada la radiación solar para predecir la potencia del panel solar y finalmente estos modelos se los utiliza para crear una aplicación que se pueda visualizar y ejecutar desde el programa Excel.

- Los modelos que utilizan redes neuronales artificiales y redes neurofuzzy tipo Anfis que no intervienen en la aplicación final realizada, pueden también ser utilizados por usuarios de Matlab si se requiere calcular las variables de salida individualmente como voltaje e intensidad pues estos modelos en su mayor parte han proyectado un error de validación inferior al 1%.

- Se ha logrado determinar y entrenar la estructura de una red neuronal tipo feed forward backpropagation que tiene como variable de entrada la radiación solar su estructura está compuesta por: dos capas ocultas, en la primera y en la segunda capa oculta con 10 neuronas cada una, la función de entrenamiento: Levenberg Marquard backpropagation (TRAINLM), función de aprendizaje LEARNGDM (Gradient descent with momentum

weight and bias learning function) y con una función de transferencia TANSIG (tangente sigmoidal hiperbólica).

- Se establece que la estructura de la red neuronal presentó un excelente desempeño al predecir la potencia del panel solar como variable de salida, pues obtuvo un error de ***0,45%*** con la función ***mse.m*** utilizando los datos de validación y demuestra que las redes neuronales son realmente precisas a la hora de calcular la potencia del panel solar.

- El modelo con redes neurofuzzy tipo Anfis, utiliza el método partición de rejilla utilizando ***genfis1,*** *se* generaron 2 reglas neuro-difusas, tiene como función de membresía de entrada ***gbell,*** función de membresía de salida ***lineal*** y el tipo de entrenamiento es ***híbrido.***

- Se establece el mejor método para generar el sistema de inferencia (FIS) que presenta el menor porcentaje de error para el entrenamiento de la red neurofuzzy tipo Anfis es el método de partición de rejilla que utiliza la función genfis1.

- El modelo con redes neurofuzzy tipo Anfis genera un error de predicción con los datos utilizados para la validación de ***0,43 %,*** *por* lo cual resulta ser el mejor modelo para predecir la potencia que genera el panel solar fotovoltaico.

- Matlab permite realizar aplicaciones de las funciones generadas, por lo tanto, se hace uso de Matlab Builder Ex para realizar una aplicación de los modelos seleccionados para compartir con usuarios de Microsoft Excel que no tengan instalado Matlab.

- Los modelos con redes neurofuzzy tipo Anfis tienen la ventaja frente a los modelos con redes neuronales tipo feed forward backpropagation pues son capaces de interpretar valores diferentes o superiores a los de entrenamiento para dar una respuesta de salida más confiable y precisa, esto puede hacerlo por el conjunto de reglas difusas que intervienen en el Sistema de Inferencia Difusa establecido durante el entrenamiento.

i.- RECOMENDACIONES

- Al realizar la adquisición de datos es necesario verificar que los sensores utilizados estén calibrados correctamente y revisar continuamente el equipo de medición puesto que se puede estropear por condiciones climáticas o por agentes externos de cualquier otra naturaleza, con lo cual estaremos en la capacidad de utilizar datos que nos permitan generar modelos precisos y confiables.

- Para seleccionar las variables que intervienen en este tipo de modelos, es necesario realizar un análisis gráfico o estadístico que nos permita visualizar la relación y correlación entre las variables y de ser necesario aplicar otras técnicas de minería de datos par así desechar las variables que no contribuyan o puedan causar confusión al desarrollar los modelos ya que para tener precisión en los modelos lo fundamental es determinar previamente cuáles serán las mejores entradas posibles para los modelos.

- Al construir modelos con redes neuronales artificiales se debe adquirir los datos cuando las variables de salida a predecir, puedan alcanzar sus puntos máximos y mínimos para garantizar el óptimo aprendizaje de la red neuronal.

- Cuando se realiza el sistema de inferencia difuso para modelos con redes neurofuzzy tipo Anfis en la GUI de Anfisedit, es conveniente utilizar varias funciones implementadas en Matlab como genfis1, genfis2, genfis3 y también se puede manipular las funciones de membresía de entrada y salida predeterminadas para disminuir el error de entrenamiento de los modelos a generarse y obtener una mejor respuesta de este tipo de redes.

- Realizar el análisis sobre los datos de potencia calculados con las técnicas de modelado implementadas en este trabajo investigativo para determinar la eficiencia real del panel solar fotovoltaico policristalino y así poder verificar en que lugar es factible o rentable la implementación de este tipo de paneles.

- Se puede mejorar la visualización de los resultados de los modelos utilizando gráficas tridimensionales entre las variables de entrada y las variables predichas.

- Al realizar aplicaciones con Matlab es necesario verificar que la versión del compilador utilizado sea compatible con la versión que utilice en Matlab y tener instalado un compilador externo principalmente todo el paquete de Microsoft Visual Studio 2010 Ultimate o superior.

- Recurrir al menú de ayuda de Matlab y a los webinarios grabados o seminarios virtuales de la página web de mathworks para realizar aplicaciones y entender la estructura de las funciones que se pueden aplicar al utilizar técnicas de modelado con redes neuronales y redes neurofuzzy tipo Anfis. .

- A los futuros estudiantes que tengan interés en el proyecto, se les recomienda realizar una metodología para generar aplicaciones con las funciones de Matlab Builder for Java y Matlab Builder for .NET para así permitir que otros usuarios también tengan otra interfaz de visualización para el acceso a estos modelos.

- Implementar un modelo con redes neuronales con datos históricos que permita predecir la radiación solar, para pronosticar el potencial solar a largo plazo del panel solar.

j.- BIBLIOGRAFÍA

LIBROS

[1] BABUSKA, R.; H. B. Verbruggen. 1996. An overview of fuzzy modeling for control. Control Engineering Practice, vol. 4.

[2] CAROTENUTO, R. 2001. Iterative system inversion technique. International Journal of Adaptive Control and Signal Processing, vol. 15.

[3] CIPRIANO A., Ramos M. 1994. Sistemas expertos difusos: fundamentos y aplicaciones. Actas X Seminario de Ingeniería Eléctrica.

[4] CIRRINCIONE M., M. Pucci, G. Cirrincione, M. G. Simões. 2005. A Neural Non-linear Predictive Control for PEM-FC'. J. Electrical Systems 1-2.

[5] CORTES D., Garcia J. 2002.Uso de Redes Neuronales para Analizar Modelos Mecánicos de Tejidos Biológicos. VI Congreso Colombiano de Elementos Finitos y Modelamiento Numérico. Bogotá, Colombia.

[6] COSTA Branco, P.J., Dente, J.A. 1998. An experiment in automatic modeling an electrical drive system using fuzzy logic. IEEE .Transactions on System Man and Cybernetics. - Part C: Applications and Reviews, Vol. 28, No. 1.

[7] GUPTA, M.M. and Knopf, G. K. 1992. A multitask Visual Information Processor with a Biologically Motivated Design' J. Visual Commun. Image Represent. Vol. 3. No. 3.

[8] HERNÁNDEZ, Jorge. Fuzzy Logic. 1994. Revista Electrónica & Computadores, No 1.11.

[9] J.-S. R. Jang. 1993. ANFIS: adaptive-network-based fuzzy inference system. IEEE Transactions on Systems, Man and Cybernetics, vol. 23.

[10] MORARI M., y E. Zafiriou. 1989. Robust Process Control. Englewood Cliffs, NJ: Prentice-Hall.

[11] PAULO E. M. Almeida, M. Godoy Simões. 2003. Neural Optimal Control of PEM Fuel Cells with Parametric CMAC Networks. In Industry Applications Conference, 2003. 38th Annual meeting. Vol 2, Octubre.

[12] R. DEL TORO, M. Schmittdiel, R. Haber-Guerra, y R. Haber-Haber. 2007. System identification of the high performance drilling process for network-based control. In Proc. of ASME IDETC 2007, Las Vegas, NV, USA.

[13] THE MATHWORKS Inc. 2002. Guía de usuario de toolbox de lógica difusa de Matlab ® Versión 5.3.The Mathworks Inc. Natick, MA.

Artículos de revistas

[1] QUINTERO DEL POZO Abelardo. 2002. Identificación mediante un Sistema Neuro-difuso de Plantas no Lineales. Departamento de Control Automático. Instituto de Cibernética. Matemática y Física, La Habana, CUBA.

[2] ROJAS Purón L. D, Morera M, Columbié A. 2003. Identificación de motor de inducción con bomba centrífuga usando lógica difusa. Revista Energética. Ciudad de la Habana.

[3] ROJAS Purón L. D., Izquierdo R, Turro A. 2003. Aplicación de base de conocimientos en supervisión de accionamiento eléctrico de hidrotransporte de pulpa lateritica. Revista Tecnología Química.

Tesis

[1] CHAHUARA Quispe, José Carlos, Torre. 2005. Tesis. Universidad Nacional Mayor de San Marcos. Facultad de Ingeniería Electrónica. EAP de Ingeniería Electrónica. Lima, Perú.

[2] COSTA Branco, P. J. 1998. Learning from examples using Fuzzy Logic in modelling and control of an electro-hydraulic actuator. Tesis Doctoral. Instituto Superior Técnico de Lisboa.

[3] FACENDO, A. "Caracterización del Comportamiento Eléctrico de una Planta de Celda de Combustible como Fuente de Generación Eléctrica", Trabajo especial de Grado para Optar al Título de Ingeniero Electricista, UNEFA, Maracay, Venezuela, En Desarrollo.

[4] FERNÁNDEZ Rojel, Daniel. 2005. Las Celdas de Combustible, una Alternativa Ecoeficiente en la Generación de Energía Eléctrica. Facultad de Ciencias de la Ingeniería de la Universidad Diego Portales, Chile.

Páginas web

- http://3.bp.blogspot.com/_IbpNGWMr5ng/Sv7f3lN8mFI/AAAAAAAAADY/u-2et3warU8/s400/kit+solar.jpg

- http://www.accessecosolar.com/files/articoli/Panel%20solar%20fotovoltaico%20Sharp%20monocristalino%20de%20185%20Watt.jpg

- http://www.mathworks.com

- http://www.mathworks.es/programs/trials/trial_request.html?eventid=661447004&s_cid=vid_ui_trials

- http://www.prisolar.com/477-large/230w-24v-panel-fotovoltaico-policristalino-pr-ysp-230p.jpg

- http://www.proviento.com.ec/SOLAR%20PANEL%20EXMORK%20100P.pdf

k. ANEXOS

ANEXO #1. Script de funciones utilizado para generar los modelos con redes neuronales y redes neurofuzzy

```
%% METODOLOGÍA PARA EL DESARROLLO DE MODELOS BASADOS EN REDES
NEUROFUZZY
%  PARA LA PREDICCIÓN LA POTENCIA REAL GENERADA A CORTO PLAZO POR PANELES
%  SOLARES UBICADOS EN LA CIUDAD DE LOJA
% PRESENTACIÓN
% El presente trabajo investigativo realiza la construcción y validación de los
% modelos generados utilizando redes neuronales y redes neuro-fuzzy, los
% mismos que permiten obtener la potencia real generada por paneles
% solares policristalinos marca Exmork de 100 W; al ingresar solamente una
%variable de entrada que es la radiación.

%% IMPORTAR DATOS
% El conjunto de datos utilizado contiene las variables de: radiación solar (W/m2); temperatura (°C),
Intensidad (A) y voltaje (V) del panel solar fotovoltaico.
% Se aplicaron 10079 mediciones para el entrenamiento, desde el 12 al 23 de junio del 2012 y para la
validación se tomaron 5720 datos del 23 al 27 de junio del 2012.
% El intervalo de muestreo fue de 1 minuto.
% Los datos se importan utilizando el comando querybuilder y se almacenan en la
% dataset llamada 'data'.
VOLTAJE = data.VOLTAJE;
INTENSIDAD = data.INTENSIDAD;
TEMPERATURA = data.TEMPERATURA;
RADIACION = data.RADIACION;

%% GRAFICANDO LOS DATOS MEDIDOS

%% FIGURA DE VOLTAJE

plot(VOLTAJE,'DisplayName','VOLTAJE','XDataSource','VOLTAJE','YDataSource','data.TIME','Color',
[0 0 1]);figure(gcf)
xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');
```

```
% Create ylabel
ylabel('V O L T A J E [ V ]','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');

%% FIGURA INTENSIDAD
plot(INTENSIDAD,'DisplayName','INTENSIDAD[A]','XDataSource','data.INTENSIDAD','YDataSource
','data.TIME','Color',[0 0 1]);figure(gcf)
% Create xlabel
xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');

% Create ylabel
ylabel('I N T E N S I D A D [ A ]','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');

%% FIGURA RADIACIÓN
plot(RADIACION,'DisplayName','RADIACION','XDataSource','RADIACION','YDataSource','data.TIM
E','Color',[0 0 1]);figure(gcf)
% Create xlabel
xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');

% Create ylabel
ylabel('R A D I A C I O N  [W / m 2]','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');

%% FIGURA TEMPERATURA PANEL SOLAR FOTOVOLTAICO
plot(TEMPERATURA,'DisplayName','TEMPERATURA','XDataSource','TEMPERATURA','YDataSour
ce','data.TIME','Color',[0 0 1]);figure(gcf)
% Create xlabel
xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');
 % Create ylabel
ylabel('T E M P E R A T U R A  [ °C ]','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');

%% Gráfica de la Potencia del panel Solar Fotovoltaico a partir de los vectores de VOLTAJE e
% INTENSIDAD.
```

```
plot(POTENCIA,'DisplayName','INTENSIDAD[A]','XDataSource',...
'data.INTENSIDAD','YDataSource','data.TIME','Color',[0 0 1]);figure(gcf)

% Crea eje x
xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
   'FontAngle','italic');

% Crea el eje y
ylabel('P O T E N C I A ( W )','FontWeight','normal','FontSize',12,...
   'FontAngle','italic');

%% Correlación entre las variables medidas.
% Para visualizar la relación entre las variables utilizamos el comando
% plotmatxix que genera una figura con  todas las variables
% obtenidas.

open plotmatrix.fig % grafica  todas las variables obtenidas para realizar un análisis visual de las variables

%%  MODELO CON REDES NEURONALES

% Construcción del modelo

% Para el entrenamiento con redes neuronales debemos tomar en cuenta que
% las variables de entrada como de salida deber estar dispuestos en forma de columnas.
% Las variables de entrada (input) y las variables de salida deseadas
% (target) utilizados. Para generar.. el modelo se utilizó el comando
% nntool

% Configuración de la red neuronal

% Tipo de Red Neuronal: FEED FORWARD BACKPROPAGATION
% Función de Entrenamiento:Levenberg-Marquardt (TRAINLM)
% Función de Aprendizaje: Gradient descent with momentum backpropagation
% (LEARNGDM)
% Función de trasferencia: Tangente Sigmoidea (TANSIG).
% Número de capas: 3,dos capas ocultas y una capa de salida.
% número de neuronas: 20, una por cada capa oculta.
```

```
% RP ....representa la red neuronal creada

%% Validación y simulación de la red neuronal.
load validar % carga la data de validación
% Para validar el modelo se utiliza otro conjunto de datos(5720 datos).
% La función sim permite simular los datos arrojados por la red neuronal.
% Para evaluar el error entre los resultados medidos y los obtenidos por
% el modelo de la red neuronal se utiliza la función mse.

% Examinar la distribución de los errores.

load RP                % representa la red neuronal creada
Q = RAD';              % Datos de Radiación solar para la validación.
PP = Ppp';             % Potencia generada por la red neuronal.
simnet = sim(RP,Q);    % Simula los datos de la red.
errornet = PP-simnet;  % Error entre la Potencia generada por la red neuronal y la Potencia medida.
performance = mse(ePPP)  % Error del rendimiento de la red neuronal.

% Gráfica de la red que compara los datos de potencia de la red y la potencia medida.

plot (simnet','DisplayName','medida','YDataSource','medida');figure(gcf);
legend('Potencia RED NEURONAL','Location','best');

xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
   'FontAngle','italic');

ylabel('POTENCIA [ W ]','FontWeight','normal','FontSize',12,...
   'FontAngle','italic');
 %%
plot (Q','DisplayName','medida','YDataSource','medida','Color',[0.86 0.16 0]);figure(gcf);
legend('Potencia medida','Location','best');

xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
   'FontAngle','italic');

ylabel('POTENCIA [ W ]','FontWeight','normal','FontSize',12,...
```

```
    'FontAngle','italic');

%%  MODELO NEURO-FUZZY

% Para el entrenamiento con redes neuro-fuzzy debemos tomar en cuenta que
% las variables de entrada como de salida deber estar dispuestos en forma de filas.

% Para generar.. el modelo se utilizó el comando
% anfisedit

 % VARIABLES UTILIZADAS EN EL MODELO RED NEURO-FUZZY.

% Variables para el entrenamiento(trainning) en ANFIS.

RP = [RADIACION Pp];

 % Variables para el validación(check) en ANFIS.

rt = [RAD TEMP];
Pp = [VOLT.*I]';
%% Construcción del modelo

% Generación del Sistema de Inferencia Difuso

% La RADIACION y la POTENCIA del conjunto de datos de entrenamiento son las
% variables..utilizadas para generar Sistema de Inferencia Difuso.

% El método usando la partición de rejilla que está programado en la función genfis1.m

RP = [RADIACION Pp];

fismatp = genfis1([RADIACION Pp]);
```

```
%% ESTRUCTURA DEL MODELO NEURO-FUZZY

% Tipo de inferencia difusa: Takagi-Sugeno-Kang
 % Número de reglas difusas: 2
% Número de funciones de membresía en la entrada: 2,(gbell)
% Método de defusificación: wtaver
% Número de parámetros no lineales: 2
% Número de funciones nodos:2
% Número de funciones de membresía en la salida: 2 (lineal)

%% Utilización del genfis
% genfis1
%fismat = genfis1([RADIACION TEMPERATURA])
%fismatp = genfis1([RADIACION Pp])
%fismatv = genfis1([RADIACION VOLTAJE])
%fisrtv = genfis1([RADIACION TEMPERATURA VOLTAJE])
%fisri = genfis1([RADIACION INTENSIDAD])

%fisrtp1 = genfis1([RADIACION TEMPERATURA Pp])
%fisrtp2 = genfis2([RADIACION TEMPERATURA],Pp,0.5)
%fisrtp3 = genfis3([RADIACION TEMPERATURA],Pp)
%fist1= genfis1([RADIACION TEMPERATURA]);

%fisrti1 = genfis1([RADIACION TEMPERATURA INTENSIDAD])
%fisrti2 = genfis2([RADIACION TEMPERATURA],INTENSIDAD,0.5)
%fisrti3 = genfis3([RADIACION TEMPERATURA],INTENSIDAD)
%fist2= genfis2(RADIACION, TEMPERATURA,[0.6 0.2]);

%fisrtv1 = genfis1([RADIACION TEMPERATURA VOLTAJE])
% fisrtv2 = genfis2([RADIACION TEMPERATURA], VOLTAJE,[0.5 0.4 0.3]);
% fisrtv3 = genfis3([RADIACION TEMPERATURA], VOLTAJE)
%fist3 = genfis3(RADIACION, TEMPERATURA);

%%  EVALUACIÓN DEL MODELO NEURO-FUZZY
% Para simular el modelo se utiliza otro conjunto de datos(5720 datos).
% La función evalfis permite simular los datos arrojados por el sistema neurofuzzy.
```

```
% Para evaluar el error entre los resultados medidos y los obtenidos por
% el modelo de la red neuronal se utiliza la función mse.

load radp % modelo neurodifuso generado
Ppp = VOLT.*I
anfisPP = evalfis(RAD,radp);
errorfis= Ppp-anfisPP;
performanceFis = mse(errorfis);

plot(Ppp,'DisplayName','Potencia medida','YDataSource','medida','Color',[0 0 1]);figure(gcf);

legend('Potencia medida','Location','best');

 xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');

ylabel('POTENCIA [ W ]','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');
plot(anfisPP,'DisplayName','Potencia estimada','YDataSource','proyectada','Color',[1 0 0]);figure(gcf);

legend('Potencia RED NEURO-FUZZY','Location','best');

xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');

ylabel('POTENCIA [ W ]','FontWeight','normal','FontSize',12,...
    'FontAngle','italic');

%% COMPARACIÓN DEL MODELO DE RED NEURONAL Y EL MODELO NEURO-FUZZY

X = RAD;

load RP
y1 = sim(RP, X')';

load radp
```

```
y2 = evalfis(X ,radp);


hold all
plot(y1,'Marker','*','Color',[1 0 0],'DisplayName','RED NEURONAL');
plot(y2,'MarkerSize',3,'Marker','o','LineWidth',1,'Color',[0 0 1],...
   'DisplayName','RED NEUROFUZZY');
hold all
xlabel('MUESTRAS','FontWeight','normal','FontSize',12,...
   'FontAngle','italic');
ylabel('POTENCIA [ W ]','FontWeight','normal','FontSize',12,...
   'FontAngle','italic');
print -dmeta
legend('RED NEURONAL','RED NEUROFUZZY','Location','best');
grid on
```

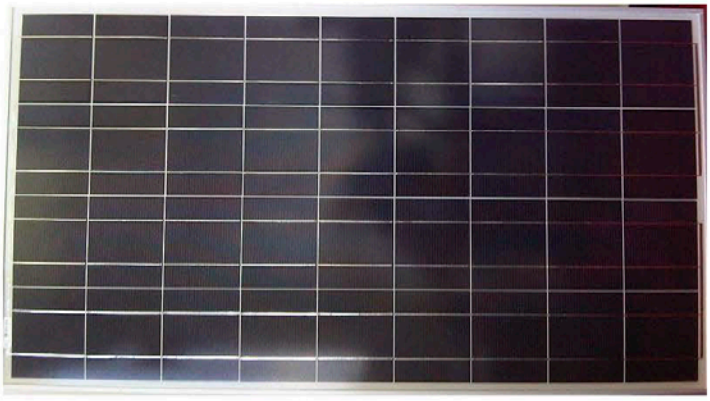
ANEXO # 2. Script de funciones utilizado para generar la aplicación en Matlab builder ex

```
function y = modell24(RAD)
%Fncpot performs using a pre-trained
% Neural-Network or ANFIS
% USAGE:
% y = Fncpot(RAD)

X = RAD;
% Load model

load RP   % MODELO DE RED NEURONAL ARTIFICIAL
y1 = sim(RP, X')';

load radp            % MODELO DE RED NEUROFUZZY
y2 = evalfis(X ,radp);

 %GRÁFICA DE LOS DOS MODELOS
 hold all
plot(y1,'Marker','*','Color',[1 0 0],'DisplayName','RED NEURONAL');
plot(y2,'MarkerSize',3,'Marker','o','LineWidth',1,'Color',[0 0 1],...
    'DisplayName','RED NEUROFUZZY');
hold all
xlabel('MUESTRAS','FontWeight','normal','FontSize',16,...
    'FontAngle','italic');
ylabel('POTENCIA [ W ]','FontWeight','normal','FontSize',16,...
    'FontAngle','italic');
print -dmeta
legend('RED NEURONAL','RED NEUROFUZZY','Location','best');
grid on
y = [y1 y2];

%#function struct
%#function struct\evalfis
%#function network
%#function network\sim
```

ANEXO # 3. Características del panel solar EXMORK 100 w

EXMORK
艾莫克新能源

SOLAR PANEL EXMORK 100 W



Specifications:

Type: Polycrystalline PV Module 12V
Model: 100P
Maximum Power: 100 Wp ±3%
Open Circuit Voltage: 22VDC
Maximum Power Voltage: 17.5 VDC
Short Circuit Current: 6.14 A
Maximum Power Current: 5.71 A
Dimensions: 1130 x 670 x 35 mm
Junction Box with 2 Bypass Diodes and cables

WARRANTY:

Technical defects: 2 years
Power degradation: 10 year 90%, 25 years 85%

EXMORK
CREATE YOUR OWN ENERGY

For over 10 years we have been specializing in wind and water power products. We service everything we sell. Scores of companies come and go, but we have consistently been in the market to take care of our customers. We use high quality cells from SunTech: Good quality, low-priced: -renewable power need not cost the earth!