Universidad de Cienfuegos “Carlos Rafael Rodríguez”

Facultad de Informática

Carrera de Ingeniería Informática

TÍTULO:

MODELO DE RED NEURONAL ARTIFICIAL PARA PREDECIR LA CARGA TÉRMICA DE UNA EDIFICACIÓN

Trabajo de Diploma para optar por el Título de Ingeniería en Informática

Autor (a):

Sadys Monteagudo Bellas.

Tutores:

Msc. Yailem Arencibia Rodríguez del Rey

MSc. Boris G. Vega Lara

Cienfuegos, Cuba

Curso 2012 – 2013

Declaración de autoría

Yo Sadys Monteagudo Bellas, me declaro como único autor del trabajo de diploma titulado “Modelo de Red Neuronal Artificial para predecir la carga térmica de una edificación”, y autorizo al Departamento de Informática de la Facultad de Ingeniería en la Universidad de Cienfuegos “Carlos Rafael Rodríguez” para que haga el uso que estime pertinente con este trabajo.

Para que así conste firmo la presente a los 10 días del mes de Junio del año 2013.

Sadys Montegudo Bellas.

Nombre completo del autor

MSc. Yailem Arencibia Rodríguez del Rey

MSc Boris Vega Lara

Nombre completo de los tutores

Los abajo firmantes certificamos que el presente trabajo ha sido revisado según acuerdo de la dirección de nuestro centro y el mismo cumple los requisitos que debe tener un trabajo de esta envergadura referente a la temática señalada.

_____	_____
Firma Tutor	Firma Tutor
_____	_____
Firma ICT	Firma Vicedecano(a)

Pensamiento

Procuro no cargar mi memoria con datos que puedo encontrar en cualquier manual, ya que el gran valor de la educación no consiste en atiborrarse de datos, sino en preparar al cerebro a pensar por su propia cuenta y así llegar a conocer algo que no figure en los libros.

Albert Einsten

Dedicatoria

Guille:

Te dedico el fruto de mis sacrificios con la esperanza de iluminarte un camino de sabiduría.

Mami:

Por todos esos momentos en que me respaldaste.
Por toda la verdad que me hiciste ver.
Por toda la alegría que trajiste a mi vida.
Por todo lo incorrecto que tú convertiste en correcto.
Por todos los sueños que tú hiciste realidad.
Por todo el apoyo que me diste cuando me sentí insegura.
Por alzarme cuando no podía alcanzar.
Por darme alas y hacerme volar.
Perdí mi fe, me la diste de nuevo.
Dijiste que ninguna estrella estaba fuera de alcance.
Tú eres quien me sostuvo y nunca me dejó caer.
Fuiste mi fuerza cuando estaba débil.
Fuiste mi voz cuando no podía hablar.
Fuiste mis ojos cuando no podía ver.
Viste lo mejor que había en mí.
Tú has sido mi inspiración, mi héroe, y el impulso de mi vida.
Para ti va mi eterno amor.
TE ADORO

Agradecimientos

A mis abuelos, por darle la base a mi vida, por aguantarme, por hacerme una persona de bien, por la luz a lo largo del camino y por su amor incondicional estaré siempre agradecida, no existe algo que pueda hacer para entregarles siquiera la mitad de lo que me han entregado, para ustedes, mis viejitos lindos, mi corazón, mi amor y mi eterna gratitud. Los amo.

A mi mamá por todo el amor que encuentro en ella.
A mi papá porque a pesar de las distancias nos hemos mantenido unidos y siempre nos hemos amado,
A mi hermano por estar siempre a mi lado.

A mi novio. Rafa te amo: gracias por tu apoyo, por aguantarme mis malacrianzas, por amarme desde el día que nos conocimos, y sobre todo gracias por estar a mi lado.

A mi familia toda, y a la familia de mi novio que es también mi familia.

A mis tutores:
Boris: gracias por la confianza de ofrecerme esta oportunidad.
Yailem: gracias por la ayuda y la guía que me brindaste, gracias por tu tiempo y dedicación, gracias por ser un ejemplo para mí.

A Ciro por su ayuda incondicional.

A mis amigas y amigos de toda la vida, y a mis compañeros de aula, por acompañarnos unos a los otros a lo largo de este camino.
A todos mis profesores, y a todos los que me han enseñado algo.
A todos las personas que me han querido y amado.

A todos: MI CORAZÓN

Pide un deseo...

lo tienes?

bien!!!

ahora cree en ello con todo tu corazón.

Resumen

La presente investigación titulada **"Modelo de red neuronal artificial para predecir la carga térmica de una edificación"**, constituye una alternativa de eficiencia energética y económica para el hotel Jagua de Cienfuegos, que utiliza un sistema de climatización centralizado "todo agua". La operación de este sistema se realiza de forma empírica a partir de la experiencia de los operarios, por lo que el consumo energético es elevado. Con el empleo de Redes Neuronales Artificiales para predecir la carga térmica del hotel, se plantea una estrategia operacional del sistema de climatización, que permite ahorrar energía y garantiza el confort térmico de la edificación.

Los modelos de redes neuronales artificiales parcialmente recurrentes como: LRN, ELMAN, y NARX, son los más apropiados para modelar sistemas dinámicos como el de esta investigación.

La experimentación se desarrolla en Matlab, consiste en probar con diferentes variantes para cada modelo, modificando parámetros de la arquitectura y el entrenamiento.

Los criterios utilizados para medir el desempeño de los modelos fueron: las funciones (incluidas en Matlab) MSE y FIT; además las pruebas no paramétricas: Friedman, Friedman Aligned Rank, Quade y Contrast Estimation.

El modelo seleccionado como mejor en la predicción de la carga térmica del hotel Jagua resultó ser el LRN con: 4 neuronas de entrada y 1 en la capa de salida; 2 capas ocultas con 10 y 4 neuronas respectivamente; funciones de transferencia TANSIG y PURELINE.; función de entrenamiento TRAINGDX, y de aprendizaje LEARNGD; entrenamiento con 2000 iteraciones y un error mínimo de 0,001.

Contenido

Ilustraciones

Tablas

Introducción

Actualmente el uso de la electricidad es fundamental para realizar gran parte de las actividades cotidianas del hombre; gracias a este tipo de energía mejora su calidad de vida. Con tan solo oprimir botones se obtiene luz, calor, frío, imagen o sonido. Su uso es indispensable, pero lo es aún más el utilizarla eficientemente.

Ahorrar y usar eficientemente la energía eléctrica, así como cuidar el medio ambiente, no son sinónimo de sacrificar o reducir el nivel de bienestar o el grado de satisfacción de necesidades cotidianas. Por el contrario, un cambio de hábitos y actitudes pueden favorecer una mayor eficiencia en el uso de la electricidad, el empleo racional de los recursos energéticos, la protección de la economía familiar y la preservación del entorno natural.

Con la Revolución Industrial la energía se vio incrementada y junto a ella la contaminación, es por eso que procesos naturales como el efecto invernadero o lluvia ácida, hoy, son peligrosos agentes involucrados en consecuencias muy dañinas como el calentamiento global [1].

En los últimos años el consumo de energía eléctrica se ha elevado a un ritmo superior al crecimiento económico, ya que suple las necesidades del aparato productivo. Esto está relacionado con mayores niveles de vida y propósitos no materializados, mezcla esta que lleva a reflexionar, sobre todo si se tiene en cuenta que en energía se gasta una importante cantidad.

Debido a este ritmo de crecimiento se deben tomar una serie de acciones que impidan, aumente el índice físico del consumo energético. La medida de ahorro más popular es el establecimiento del horario de verano, aplicada en más de 75 países [2]. Otras variantes que gozan de aceptación son:

• Sustitución por las mezclas combustibles (alcohol o gas licuado a la gasolina, empleo de sistemas híbridos como gasolina-celda combustible, gasolina-gas licuado, gasolina-alcohol, entre otros).

• Mayor empleo de las fuentes de energía alternativa (eólica, fotovoltaica, digestores de biogás, incineración de desechos sólidos urbanos, biomasa).

• Cogeneración donde sea posible.

• Uso de nuevas tecnologías, como las celdas combustibles y las baterías de óxido sólido.

• Empleo de consumidores eficientes y ambientalmente amistosos (lámparas compactas eficientes y ahorradoras, tecnologías de punta, automatización de procesos industriales).

• Disminución del consumo energético con sistemas de educación a la población y los principales consumidores.

• La modernización de instalaciones de generación y eléctricas ineficientes.

También resulta imprescindible identificar y explotar todas las reservas de eficiencia, extendiéndose el proceso al acomodo de carga, lo que es sinónimo de eliminar todas las producciones y servicios que no están haciendo trabajo útil [2].

Actualmente la reducción del consumo energético asociado al uso y construcción de edificios y el incremento de la eficiencia energética en el acondicionamiento climático de los mismos, son cuestiones que están recibiendo gran atención en el ámbito internacional [3].

Estudios recientes indican que el consumo de energía en los edificios representa más del 35% del consumo energético mundial, siendo más de la mitad atribuible a sistemas de climatización. El 33% de la emisión global de CO2, incluyendo las emisiones debido al uso eléctrico, proviene de las edificaciones [4]. Es importante destacar que el mayor consumo de energía eléctrica en estas edificaciones se le atribuye al proceso de enfriamiento del agua circulante para alcanzar la climatización [5].

En Cuba, las grandes edificaciones asociadas a un mayor consumo energético son aquellas destinadas a la comercialización y al turismo, siendo los hoteles las más afectadas. Hoteles en Cuba reportan alrededor del 65% del total del consumo de electricidad en el área de climatización.

A nivel mundial, los sistemas de climatización utilizados en la hotelería, son generalmente del tipo centralizado, de compresión mecánica, con enfriadores de agua y distribución del agua helada a los diferentes sectores del hotel. Estos son conocidos como sistemas "todo agua". Grandes ciudades se destacan en este ámbito.

En varios edificios oficiales, oficinas y la mayoría de los hoteles en Toronto, la ciudad más grande de Canadá, se utilizan estos sistemas [6]. Sucede también en grandes edificaciones de la ciudad de Estocolmo [7] y en Guangzhou, China [8]. En Hawai [9], y en las Antillas Holandesas [7], la instalación de sistemas de climatización centralizada todo agua, es una práctica habitual.

En grandes universidades del mundo se utilizan estos sistemas como una técnica efectiva para la climatización. Tal es el caso de la Universidad de Cornell [7], la Universidad de California en Merced, USA (University of California at Merced in USA.) [10], y en grandes universidades de Hong Kong [11].

En Cuba, hoteles de Varadero, de la Cayeria Norte y de Trinidad, utilizan estos sistemas, tomando como experiencias su instalación alrededor del mundo [12]. La perspectiva nacional es extender el uso de estos en futuras construcciones hoteleras.

Los *chillers*[1], son sistemas que centralizan la generación del fluido térmico encargado de transportar la energía a los locales. Estos sistemas son flexibles desde el punto de vista del control centralizado y permiten establecer estrategias operacionales que

[1] ***Chiller:*** Término del idioma inglés usado para referirse a la unidad enfriadora de agua. Es un caso especial de máquina de refrigeración cuyo cometido es enfriar un medio líquido.

reducen considerablemente los costos de energía, manteniendo las condiciones de *confort*[2].

Hoy día, la estrategia de operación del *chiller* empleada en la mayoría de los hoteles del país es puramente empírica, y se establece a partir del conocimiento y la experiencia de los operadores. Como consecuencia, el sobreconsumo energético es elevado.

En la última década se han llevado a cabo una serie de proyectos alrededor del mundo, enfocados a reducir el consumo energético de los equipos encargados de la climatización. Tal es el caso de los investigadores Prívara, Siroký [13] y Borrelli [10].

Los estudios sobre plantas de climatización centralizadas indican que es posible un ahorro de energía considerable (hasta un 35% en algunos casos en que la operación es muy deficiente), con la implementación de controles óptimos. Algunas investigaciones exhiben ahorros significativos alrededor de un 20% a un 30%, en dependencia del aislamiento e impermeabilidad de la zona, y por supuesto, del clima específico de cada región [14]. Los investigadores Ma y Wang [15], realizan un análisis detallado de las diferentes estrategias empleadas en la literatura para reducir el consumo energético de los *chillers* y al mismo tiempo garantizar la demanda para satisfacer el confort.

En Cuba existen investigaciones que han diseñado estrategias de optimización para un mejor funcionamiento del *chiller*, elevando la eficiencia y productividad de los mismos. Tal es el caso de los investigadores Hernández [16] y Montelier [14] que demuestran el uso de técnicas de Inteligencia Artificial (IA) en aplicaciones del control del clima de edificaciones.

El hotel Jagua en Cienfuegos, constituye una de las principales instalaciones hoteleras en la provincia, con alto índice de ocupación todo el año y por tanto grandes gastos de energía, la mayoría asociados al área de climatización. Por ello se necesita mejorar la

[2] C***onfort:*** (galicismo de *confort*) Es aquello que produce bienestar y comodidad. La mejor sensación global durante la actividad es la de no sentir nada, indiferencia frente al ambiente. Para realizar una actividad el ser humano debe ignorar el ambiente, debe tener confort.

operación del sistema de climatización utilizado en dicha edificación, que es de tipo centralizado y todo agua.

Siguiendo uno de los principios más empleados en la literatura, como en las investigaciones de Ma y Wang [15] y Siroký y Oldewurtel [17], la estrategia a trazar en este proyecto es disminuir el consumo de energía del *chiller*. Para ello se necesitan realizar ajustes dinámicos en la modelación de la carga térmica.

La modelación dinámica de las cargas térmicas depende fundamentalmente de variables meteorológicas (temperatura ambiente, radiación solar y temperatura de *set-point*[3]) y en algunos casos es considerable tener en cuenta el perfil de ocupación del hotel.

Investigaciones recientes reportadas en la literatura demuestran el uso de Redes Neuronales Artificiales (RNA) en la modelación de cargas térmicas, a partir de las cuales se obtienen muy buenos resultados. En este ámbito se destacan investigadores como Ben-Nakhi y Mahmoud [18]; Hou, Lian, Yao y Yuan [19]; Kwok y Lee [20]; Abbassi y Bahar [21]; Pandey y Hindoliya [22]; Ge, Xiao y Wang [23]; Armas [24]; Valdivia [25] y Montero [26].

Las soluciones propuestas a partir de estas investigaciones no consideran la masa de la edificación como una red térmica formada por resistencias y capacitancias, donde la modelación de la carga térmica no solo depende de sus entradas (variable meteorológicas y de ocupación) para un instante de tiempo, sino también del pasado de estas. Esta es la característica que confiere dinamismo al proceso.

Las RNA cobran gran importancia en la modelación de procesos dinámicos gracias a dos propiedades fundamentales: su capacidad de encontrar patrones de forma inductiva, por medio de los algoritmos de aprendizaje basado en los datos existentes, y la no linealidad con la que pueden representarse [27].

[3] Temperatura de ***set-point:*** Temperatura que se considera indicada para lograr el confort.

Las Redes Neuronales Recurrentes (RNR), llevan estas dos propiedades de forma muy directa, por lo que son muy utilizadas en la modelación de procesos dinámicos. Investigadores como Luera Peña y Minim [28], Rabuñal [29], Vera y Bustamante [30], Piedra y Lòpez [31] y Callinan [32] así lo demuestran.

Por todas las razones anteriormente expuestas urge una solución al siguiente ***problema científico:*** ¿Cómo predecir la carga térmica de un hotel con sistema de climatización centralizado todo agua, considerando las propiedades resistivas-capacitivas de la edificación?

Se identifica como ***objeto de estudio*****:** Las RNA aplicadas a modelos térmicos de edificaciones. Se define como ***campo de acción*****:** Las RNA aplicadas al modelo térmico de climatización del hotel Jagua.

Idea a Defender**:**

Con el empleo de RNA para predecir la carga térmica de un hotel, será posible plantear una estrategia operacional del sistema de climatización del hotel Jagua, que permita ahorrar energía y garantice el confort térmico de la edificación.

La labor científico investigativa está encaminada a dar cumplimiento al siguiente ***objetivo general*****:** Proponer un modelo de RNA para predecir la carga térmica de un hotel con sistema de climatización centralizado todo agua, considerando las propiedades resistivas-capacitivas de la edificación.

Objetivos específicos

1- Estudiar el estado del arte sobre el proceso de climatización centralizado todo agua en edificaciones y las aplicaciones de RNA en procesos dinámicos.
2- Seleccionar las RNA más apropiadas para modelar el sistema de climatización del hotel Jagua.
3- Experimentar en Matlab con los modelos seleccionados.
4- Comparar los resultados obtenidos de los diferentes modelos de RNA para proponer el mejor, en la predicción de la carga térmica del hotel Jagua.

El trabajo se estructura de la siguiente forma: introducción, tres capítulos, conclusiones, referencias bibliográficas, bibliografía y anexos.

Capítulo 1 nombrado “Marco teórico de la investigación”. Se exponen los conceptos asociados al dominio del problema y el estado del arte sobre el proceso de climatización centralizado todo agua en edificaciones y la aplicación de RNA en procesos dinámicos**.**

Capítulo 2 nombrado “Diseño del experimento”: Se presentan las características (arquitectura y aprendizaje) generales de los modelos de RNA seleccionados y los detalles de su experimentación en Matlab.

Capítulo 3 nombrado “Análisis de los resultados”: Se comparan los resultados obtenidos por los diferentes modelos para llegar a conclusiones respecto al mejor en la predicción de la carga térmica del hotel Jagua.

Capítulo I: Marco teórico de la investigación.

El presente capítulo enuncia las principales características vinculadas al objeto de estudio, deja claro que son y cómo funcionan los sistemas de climatización centralizada, todo agua. Se analizan las técnicas computacionales que se aplican a estos sistemas, justificando el uso de las RNA en la solución del problema planteado. Finalmente da respuesta al porqué utilizar RNA recurrentes para modelar el sistema de climatización utilizado en el hotel Jagua como un proceso dinámico.

1.1 Características de los esquemas térmicos en hotelería.

Las redes hoteleras son unidades prestadoras de servicio para el sector del turismo. Estos deben presentar condiciones de confort óptimas acorde a su política de hospedaje y a sus características propias [33].

Cada hotel implementa determinados sistemas térmicos para crear condiciones idóneas de confort con el objetivo de prestar el mejor servicio [34].

Estos sistemas varían en dependencia de las necesidades económicas y los objetivos específicos del hotel. Se trata siempre que los índices de consumo disminuyan y que los equipos que intervienen trabajen eficientemente.

Para disminuir los costos de producción y aumentar la eficiencia energética en el acondicionamiento de aire y el calentamiento de agua, se utilizan esquemas térmicos que trabajen en conjunto. Los sistemas de clima trabajan a partir de la compresión de vapor, donde un compresor hace circular el refrigerante por el sistema, a la salida de este se instala un intercambiador para absorber el calor, disminuyendo la temperatura del refrigerante y utilizando el calor generado para el calentamiento de agua. Combinando dos circuitos con diferentes propósitos y apoyándose uno del otro, se logra mejorar las condiciones térmicas e incrementar la eficiencia energética de la instalación [34].

La instalación de los sistemas de climatización centralizada se hace cada vez más frecuente en todo el mundo. Es el caso varios edificios oficiales, oficinas y la mayoría de los hoteles en Toronto, la ciudad más grande de Canadá, donde se utilizan estos sistemas [6].

Sucede también en grandes edificaciones de la ciudad de Estocolmo [7] y en Guangzhou, China [8].

En grandes universidades del mundo se utilizan estos sistemas como una técnica efectiva para la climatización, tal es el caso de la Universidad de Cornell [7], la Universidad de California en Merced, USA (University of California at Merced in USA.) [10], y en grandes universidades de Hong Kong [11].

En Hawai [9], y en las Antillas Holandesas [7], la instalación de sistemas de climatización centralizada todo agua, es un práctica habitual.

1.2 Principales características de los sistemas de climatización en Cuba.

En Cuba, la producción de frío en función de la climatización en los hoteles generalmente se realiza por unidades centralizadas que enfrían un fluido el cual es bombeado para todas las habitaciones y locales del hotel, conocidas como sistemas centralizados todo agua.

El funcionamiento se basa en un grupo de *chillers* y un sistema de bombeo que hace circular agua por las tuberías hasta las unidades *FanCoil*[4], donde se transfiere la potencia de frío al acondicionamiento del aire de cada habitación [33].

El consumo asociado al *chiller* depende de la temperatura de agua helada (*set-point*) que se elige como consigna y de la carga térmica a vencer, que se refleja en la diferencia de temperatura del agua de retorno y a la salida del *chiller*. Mientras más helada se quiera el agua a la salida del *chiller* más esfuerzo de los compresores se

[4] ***FANCOIL:*** Es una unidad terminal provista básicamente de un ventilador y un serpentín de intercambio térmico por donde circula agua helada. Puede disponer también de filtro de aire y batería de calefacción (eléctrica o agua caliente).

requiere y por consiguiente mayor consumo energético [16]. Todo esto en función de las características térmicas y de la estrategia operacional de la instalación [33].

En Cuba hoteles de Varadero, de la Cayeria Norte, de Trinidad como por ejemplo los hoteles Iberostar y Trinidad del Mar, por solo mencionar algunos, utilizan estos sistemas, tomando experiencias a partir de la instalación de estos, alrededor del mundo. En Cienfuegos, el hotel Jagua y el hotel La Unión, tienen instalado sistemas de climatización centralizados, todo agua.

1.3 Técnicas de identificación de los modelos de carga térmica: Circuitos Resistivos-Capacitivos.

El control avanzado de los sistemas de climatización centralizada en edificaciones y los métodos de detección de fallas, requieren modelos térmicos para predecir o estimar el funcionamiento esperado de estos. La implementación de los modelos exige frecuentemente cálculos computacionales de alto rigor, razón por la cual resulta imprescindible la obtención de modelos eficaces y eficientes que sean capaces de captar la dinámica del sistema para predecir la carga térmica.

En grandes edificaciones, como son los hoteles, donde existe una variedad en las características constructivas del sistema, se torna muy difícil obtener al detalle las propiedades físicas. Para solucionar este problema, en varias investigaciones se representa toda la masa del edificio como una red térmica formada por resistencias y capacitancias, como es el caso de Wang y Xu [35], y Lee y Braun [36]. Frecuentemente en la literatura, este es un punto común de partida, donde la estructura de la red sufre algunas modificaciones en función de las características físicas y térmicas propias de la instalación. Luego se plantean diferentes vías de solución dependiendo de las salidas que se deseen obtener, Braun y Chaturvedi [37], Ghiaus y Hazyuk [38], para mejorar las medidas encaminadas a elevar la eficiencia energética de la instalación.

Se trata de corresponder el modelo de la edificación con circuitos eléctricos a partir de diferentes configuraciones resistivas-capacitivas. Este método es una de las alternativas

más viables hoy día para el diseño de modelos de carga térmica, así lo demuestran Bacher y Madsen [39], Siroký y Oldewurtel [17].

Maniobrar la carga térmica de forma dinámica en función de las variables, sin afectar el *confort* es sin duda una posibilidad inmejorable de elevar los índices de eficiencia.

1.4 Sistema de climatización del hotel Jagua.

Una de las funciones de los hoteles turísticos y en general de toda arquitectura, es la de crear espacios que brinden condiciones de confort a los usuarios. La búsqueda de la calidad de vida para todos es un pacto de responsabilidad que involucra la calidad del ambiente construido y en la conciencia sobre la eficiencia energética [40].

Arquitectura eficiente es aquella coherente con las condiciones climáticas, ambientales, económicas, culturales y tecnológicas. Si el tipo de respuesta es un modelo (edificio) totalmente desvinculado del ambiente lo más seguro es que sus niveles de habitabilidad estén alejados de los aceptables llegando incluso a condiciones interiores peores a las del exterior. Dicho de otro modo, el edificio funciona peor que el clima. Estos edificios, según su escala, requieren mayores instalaciones y mayores cantidades de energía que reparen los problemas de diseño para proporcionar condiciones de habitabilidad aceptables y por lo tanto mayores costos de instalaciones y funcionamiento.

El edificio es en sí un sistema artificial costoso, creado por el ser humano para satisfacer sus necesidades y desarrollar sus actividades en condiciones de confort [40].

¿Cuál es el mejor modo de lograr el confort?

Sin duda, que el edificio responda al **"dónde"** funcionando coherentemente con el clima, y al **"cómo"** de acuerdo al uso eficiente de recursos y tecnologías [40].

El potencial de uso eficiente de la energía en los edificios es muy grande, por tanto siempre se debe tratar de optimizar el mismo, a través de la aplicación de los principios del acondicionamiento natural y el diseño de los dispositivos de climatización artificial. Se trata de lograr condiciones de confort con eficiencia y no simplemente de consumir

menos energía. Aunque los sistemas artificiales de calefacción o refrigeración (sistemas complementarios) difícilmente puedan solucionar errores graves de proyecto porque ese no es su cometido, estos ayudan, en gran medida, a cumplir con objetivos de confort y la calidad del aire interior para los usuarios, el manejo y control de la energía en los edificios [40].

Los sistemas de climatización utilizados en la hotelería en Cuba, como es el caso del hotel Jagua en Cienfuegos, son generalmente del tipo centralizado, de compresión mecánica, con enfriadores de agua y distribución del agua helada a los diferentes sectores del hotel. Estos son conocidos como sistemas centralizados todo agua: se basan en la distribución de energía a los diversos locales, exclusivamente mediante agua. El agua fría es utilizada por unidades *FanCoil* que se instalan en cada ambiente individual [40].

Ilustración 1: Sistema de climatización del hotel Jagua.

Desarrollar un modelo general que integre todos los factores que influyen en el comportamiento energético y determinan el consumo en un hotel resulta en ocasiones una tarea engorrosa, y en cierta medida difícil de lograr [40]. En el funcionamiento de los sistemas de climatización por agua helada, los principales factores a tener en cuenta son los siguientes:

1. Las condiciones climáticas

2. Propiedades físicas

3. La estrategia y nivel ocupacional

5. Los objetivos turísticos del hotel

6. Los hábitos, tradiciones y objetivos de los clientes

Por otra parte, para resolver este problema mediante procedimientos tradicionales, se necesita la modelación detallada de un conjunto de sistemas complejos e interrelacionados. En este tipo de solución intervienen recursos, investigadores y un período de investigación que no siempre están disponibles, debido a que es un caso de optimización, con un extenso campo de búsqueda, donde la interrelación entre las variables participantes, en ocasiones no está claramente definida o existen incertidumbres e imprecisiones, que no siempre pueden ser resueltas satisfactoriamente por esta vía [41].

Es precisamente en este escenario donde se logran resultados adecuados, que pueden considerarse precisos, mediante la combinación de técnicas tradicionales y de la IA. Estas últimas tienen la propiedad de reproducir patrones, manejar la vaguedad de términos y la incertidumbre de forma útil, confiriendo una gran potencia de cálculo y la toma de decisiones oportunas en tiempo real para obtener las mejores prestaciones de los sistemas.

1.5 Introducción a la Inteligencia Artificial.

Los lenguajes de programación llamados convencionales permiten la creación de sistemas computacionales para la solución de problemas con la ayuda de la computadora. En ocasiones es necesario, sin embargo, diseñar sistemas que imiten la forma de pensar del hombre para acometer la solución de problemas de la práctica que presentan estas dificultades, por lo que ha sido necesaria la creación de técnicas de programación que garanticen la solución de los mismos con un mínimo de esfuerzo y recursos. El intento de abordar problemas poco estructurados, donde no se conoce de antemano cuál es el mejor método para resolverlo o no pueden ser solucionados a través de los lenguajes convencionales, marcó el origen de la Inteligencia Artificial.

Quien acuñó el término Inteligencia Artificial fue el matemático John McCarthy, en el año 1956, durante un congreso en Dartmouth (U.S.A.) para agrupar a todos los métodos, técnicas e intentos de simular el intelecto humano en la computadora.

Muchas son las definiciones o interpretaciones sobre este término:

Según Elaine Rich y Kevin Knight, la Inteligencia Artificial estudia cómo lograr que las máquinas realicen tareas que, por el momento, son realizadas mejor por los seres humanos [42].

Según Ray Kurzweil, la IA es el arte de crear maquinas con capacidad de realizar funciones que realizadas por personas requieren de inteligencia [43].

Según George F. Lugar y William A. Stubblefied, la IA es la rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente [44].

Según Robert J. Schalkoff, la IA es el campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales [45].

En sus inicios se levantó una barrera en contra de este nombre y de todo lo que él significa. No obstante el uso histórico acuñó el nombre de Inteligencia Artificial como disciplina dentro del campo de la Ciencia de la Computación.

Esta investigación asume que la IA es la rama de la ciencia de la computación que se encarga de que procesos computacionales simulen, en cierta manera, la inteligencia humana.

Muchos son los problemas abordados por la IA y tienen muy poco en común excepto que todos ellos son muy complicados. La IA proporciona un conjunto de técnicas, herramientas y métodos que han demostrado su aplicabilidad y que permiten resolver estos problemas. Se acude a sus técnicas cuando es necesario incorporar en un sistema informático, conocimiento o características propias del ser humano.

Una técnica de I.A. es un método para explotar el conocimiento.

- Aprendizaje Automático (Machine Learning).
- Ingeniería del conocimiento (Knowledge Engineering).
- Lógica difusa (Fuzzy Logic).
- Sistemas reactivos (Reactive Systems).
- Sistemas Multiagente (Multi-Agent Systems).
- Sistemas basados en reglas (Rule-Based Systems).
- Razonamiento basado en casos (Case-Based Reasoning).
- Sistemas expertos (Expert Systems).
- Redes Bayesianas (Bayesian Networks).
- Vida artificial (Artificial Life). La VA no es un campo de la IA, sino que la IA es un campo de la VA.

    - Computación evolutiva
    - Estrategias evolutivas
    - Algoritmos Genéticos

- Técnicas de Representación de Conocimiento.
- Visión artificial.
- Audición artificial.
- Lingüística computacional.
- Procesamiento del lenguaje natural (Natural Language Processing).
- Minería de datos (Data Mining).
- Redes neuronales artificiales (Artificial Neural Networks). [46–48]

1.5.1 Técnicas de Inteligencia Artificial aplicadas a los sistemas de climatización centralizada, todo agua.

Han sido muchos los modelos de optimización empleados en los sistemas de climatización, todo agua utilizando técnicas de la IA, con diferentes comportamientos y costos computacionales que definen ventajas y desventajas en su utilización.

Investigaciones recientes reportadas en la literatura demuestran el uso de las técnicas de IA, específicamente el uso de RNA en la modelación de cargas térmicas para

sistemas de climatización. En este ámbito se destacan investigadores como Ben-Nakhi y Mahmoud [18]; Hou, Lian, Yao y Yuan [19]; Kwok y Lee [20]; Abbassi y Bahar [21]; Pandey y Hindoliya [22]; Ge, Xiao y Wang [23]; Armas [24]; Valdivia [25] y Montero [26].

Autores como Hernández [16], Montelier y Armas [14], Sözen y Özalp [49], proponen la utilización de los Algoritmos Genéticos (AG) en este tema.

Armas [24], demuestra las potencialidades de los criterios termoeconómicos incorporando herramientas de inteligencia artificial para el diseño de sistemas centralizados, en un estudio que dirige hacia el diseño óptimo del sistema.

Las herramientas de IA no solo se emplean como método de optimización (AG) sino que se utilizan para conformar un modelo híbrido del sistema con redes neuronales artificiales para modelar las propiedades termodinámicas de los refrigerantes, tal como se muestra en las investigaciones de Sözen y Özalp [49], Armas y otros [50]

Valdivia [25] propone un procedimiento para la optimización de un sistema de climatización centralizada por agua helada en la etapa prematura del diseño comercial, para ello se crea un modelo híbrido que combina herramientas termoeconómicas con técnicas de inteligencia artificial como son las RNA y los AG para minimizar el costo de los productos finales del sistema (agua fría para climatización de locales y agua caliente para calentamiento de agua sanitaria).

Montero y otros investigadores [26], emplean un modelo de Red Neuronal Artificial con una arquitectura feed-foward para predecir el consumo energético y de gas licuado en una instalación teniendo en cuenta condiciones climatológicas y nivel de ocupación.

Las soluciones propuestas a partir de estas investigaciones mostraron muy buenos resultados en cuanto a su objetivo, pero no consideran la masa de la edificación como una red térmica formada por resistencias y capacitancias, donde la modelación de la carga térmica no solo depende de sus entradas (variable meteorológicas) para un instante de tiempo, sino también del pasado de estas. Esta es la característica que confiere dinamismo al proceso. Por tal razón, estas soluciones no se ajustan al problema planteado en esta investigación.

Maniobrar la temperatura de suministro (carga térmica) de forma dinámica, en función de la variación de la carga, sin afectar el *confort* es sin duda una posibilidad inmejorable de elevar los índices de eficiencia.

Las RNA cobran gran importancia en la modelación de procesos dinámicos gracias a dos propiedades fundamentales: su capacidad de encontrar patrones de forma inductiva, por medio de los algoritmos de aprendizaje basado en los datos existentes, y la no linealidad con la que pueden representarse [27].

A partir de las características antes señaladas, y luego de una profunda investigación, queda clara la necesidad de un nuevo criterio que integre las propiedades físicas y térmicas de una manera más sencilla y exacta, y que aporte un mejor nivel de predicción del comportamiento energético o carga térmica.

1.6 Redes Neuronales Artificiales: una técnica de la IA.

En las últimas décadas las RNA han recibido un interés particular como una de las técnicas de IA, puesto que ofrece los medios para modelar de manera efectiva y eficiente problemas grandes y complejos. Los modelos de RNA son capaces de encontrar relaciones (patrones) de forma inductiva por medio de los algoritmos de aprendizaje basado en los datos existentes, más que requerir la ayuda de un modelador para especificar la forma funcional y sus interacciones [27].

Las RNA se basan en la analogía que existe en el comportamiento y función del cerebro humano, en particular del sistema nervioso, el cual está compuesto por redes de neuronas biológicas que poseen bajas capacidades de procesamiento, sin embargo toda su capacidad cognitiva se sustenta en la conectividad de éstas [27].

Ilustración 2: Sistema nervioso del cerebro humano.

La unidad de una red neuronal artificial es un procesador elemental llamado neurona que posee la capacidad limitada de calcular, en general, una suma ponderada de sus entradas y luego le aplica una función de activación para obtener una señal que será transmitida a la próxima neurona. Estas neuronas artificiales se agrupan en capas o niveles y poseen un alto grado de conectividad entre ellas, conectividad que es ponderada por los pesos. A través de un algoritmo de aprendizaje supervisado o no supervisado, las RNA ajustan su arquitectura y parámetros de manera de poder minimizar alguna función de error que indique el grado de ajuste a los datos y la capacidad de generalización de las RNA [27].

Las redes neuronales se utilizan en la resolución de problemas prácticos concretos, que normalmente no han sido bien resueltos mediante sistemas más tradicionales. Gracias a su capacidad de aprendizaje, robustez, no linealidad y tolerancia a la imprecisión e incerteza del entorno, desde hace unos años las redes neuronales vienen alcanzando excelentes resultados en aplicaciones diversas. Así como las aplicaciones prácticas de las redes neuronales resultaban hace unos años en fase experimental, en la actualidad muchas compañías las aplican de un modo rutinario a numerosos problemas; en este sentido, la aplicación de redes neuronales puede considerarse que ha alcanzado ya su madurez. Los más habituales son los relacionados con clasificación, estimación funcional y optimización; en general, el del reconocimiento de patrones suele considerarse como un denominador común. Se pueden señalar, entre otras, las siguientes áreas de aplicación de los sistemas neuronales: reconocimiento del habla,

reconocimiento de caracteres, visión, robótica, control, procesamiento de señales, predicción, economía, defensa, bioingeniería, etc.

Las RNA al margen de "parecerse" al cerebro presentan una serie de características propias del cerebro. Por ejemplo las RNA aprenden de la experiencia, generalizan de ejemplos previos a ejemplos nuevos y abstraen las características principales de una serie de datos [51].

La topología o arquitectura de redes consiste en la organización y disposición de las neuronas formando capas o agrupaciones de neuronas más o menos alejadas de la entrada y salida de la red [52].

Entre los tipos básicos de aprendizaje se tiene:

Aprendizaje supervisado. Ocurre cuando se le proporciona a la red tanto la entrada como la salida correcta, y la red ajusta sus pesos tratando de minimizar el error de su salida calculada [30].

Aprendizaje no supervisado. Se presenta cuando a la red se le proporcionan únicamente los vectores de entrada, y ésta ajusta sus interconexiones basándose únicamente en sus estímulos y su propia salida. Las leyes de aprendizaje determinan como la red ajustará sus pesos utilizando una función de error o algún otro criterio. La ley de aprendizaje adecuada se determina en base a la naturaleza del problema que se intenta resolver [30].

1.6.1 Redes Neuronales Artificiales: una técnica efectiva para modelar procesos dinámicos.

Desde hace varios años han sido estudiados y reconocidos a nivel mundial los potenciales beneficios que ofrecen las redes neuronales en su aplicación para el control de procesos industriales, particularmente para la modelación de sistemas no lineales o dinámicos.

Su aprendizaje adaptativo, auto-organización, tolerancia a fallos, operación en tiempo real y fácil inserción dentro de la tecnología existente, ha hecho que su utilización se haya extendido en áreas como la biología, finanzas, industrial, medio ambiente, militar, salud, etc. [53]. Están funcionando en aplicaciones que incluyen identificación de procesos [54], detección de fallos en sistemas de control [55], modelación de dinámicas no lineales [56], [57], control de sistemas no lineales [58], [59], y optimización de procesos [60–62].

Luera y Minim [28] emplean las RNA para predecir el perfil de la temperatura en el centro de un producto acondicionado en latas durante el procesamiento térmico. Para la modelación, utilizando redes neuronales, se construyó una red de tipo recurrente ya que el tratamiento térmico de alimentos es un proceso de tiempo dependiente, también conocido como proceso dinámico. Rabuñal [29] en su tesis doctoral, desarrolla una herramienta informática que, mediante algunas técnicas de IA como las RNA, ayuda a ingenieros civiles a resolver problemas de diferente naturaleza relacionados con dos áreas de la Ingeniería Civil: Ingeniería de la Construcción e Hidrología.

Dentro del área de la construcción se trabajó en la búsqueda de una solución obtenida por métodos evolutivos en concreto, mediante la técnica de Programación Genética (PG), al problema del cálculo o determinación de las longitudes de anclaje de armaduras pasivas de acero en vigas de hormigón armado. Así mismo se utilizaron RNA para dar un resultado más predecible.

Dentro del área de hidrología se propone la aplicación de dos técnicas de Inteligencia Artificial: las Redes de Neuronas Artificiales y la Programación Genética. El objetivo fue mostrar cómo estas dos técnicas pueden trabajar conjuntamente para resolver el problema de la predicción del caudal de una cuenca urbana.

Vera y Bustamante [30] proponen la construcción de un modelo dinámico para pronosticar que permite alternativas de solución con niveles de incertidumbre mejores de los que suministra las técnicas convencionales. El uso de las RNA aporta soluciones más eficientes y seguras que los métodos convencionales.

Piedra y Lòpez [31] abordan una vía para solucionar la congestión que ocurre en las redes de Internet a través de la utilización de Redes Neuronales Artificiales. Por las características propias del tráfico de red: heterogeneidad y dinamismo, se utilizó un modelo de RNA recurrente, para predecir el rendimiento de una red, en términos de ancho de banda disponible y latencia- error mínimo, que comparado con otros sistemas proporciona mejores predicciones.

Callinan [32] aplica RNA al problema de control del péndulo invertido. El péndulo invertido se usa típicamente para comparar técnicas de control en los sistemas altamente inestables no lineales. Las RNA tienen características únicas que permiten controlar sistemas no lineales. RNA Feedforward y RNA recurrentes se usaron para modelar el péndulo invertido, y estas primeras mostraron mejores resultados.

1.7 Redes Neuronales Recurrentes.

Generalmente los modelos de RNA se clasifican por el tipo de aprendizaje o por topología de la red (arquitectura). Dentro de esta última existen dos clasificaciones importantes según el tipo de enlace que presentan las neuronas dentro de la red:

- Redes Feed-forward, también conocidas como estáticas o unidireccionales.
- Redes Feed-back, también conocidas como dinámicas o recurrentes.

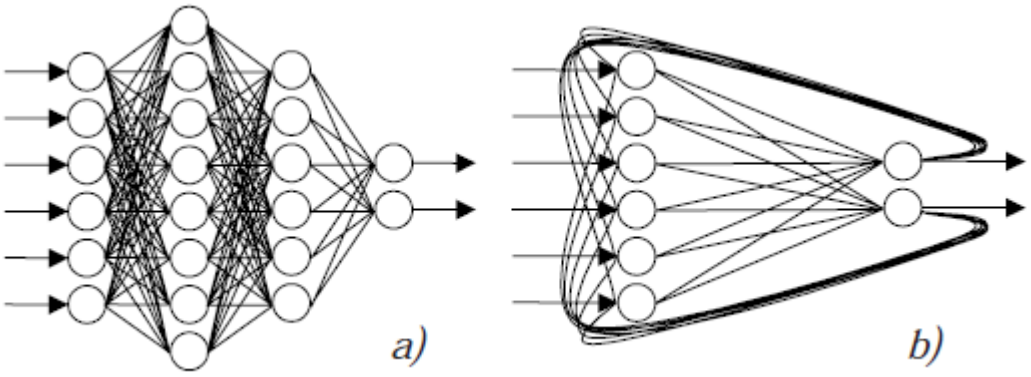


Ilustración 3: Ejemplo de RNA: Feed-forward (a) y Feed-back (b).

Las redes feed-forward son muy aplicables en la práctica, pero ¿qué pasará ante la necesidad de modelar mediante RNA un proceso que dependa de parámetros cambiantes en el tiempo?, ¿serán las redes feed-foward capaces de modelar estos procesos?

Las RNA feed-forward, en su arquitectura básica, representan un modelo estático cuyos parámetros tienen valores fijos, por tanto no son efectivas ante la necesidad de modelar un proceso que dependa de parámetros cambiantes en el tiempo. A diferencia de estas, las Rede Neuronales Recurrentes (RNR) crean un estado interno de la red que permite que exhiba comportamiento temporal dinámico.

Las RNR son sistemas dinámicos cuyo estado puede evolucionar siguiendo un comportamiento no lineal, que representan relaciones causales en el tiempo y el espacio. Se simulan como redes con conexiones de retroalimentación (feed-back). Son un intento de establecer una correspondencia entre una secuencia de entrada SE y una de salida SS, que se van a interpretar como patrones temporales SE (ti) y SS(tj).

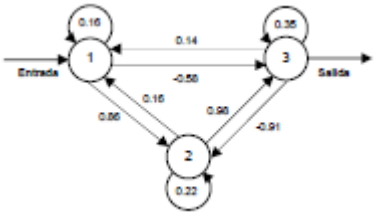


Ilustración 4: RNA con retroalimentación.

A diferencia de las redes feed-forward en las que la meta es que la red converja a un valor de estados fijo en cierto período de tiempo, las RNR tienen comportamiento variables en el tiempo [63].

Las RNR se aplican a los datos de serie de tiempo y que usan salidas de unidades de la red en ***t*** de tiempo como el aporte para otras unidades en ***t + 1*** de tiempo [64].

Esta red viene caracterizada por la existencia de lazos de realimentación. Estos lazos pueden ser entre neuronas de diferentes capas, neuronas de la misma capa o, más

sencillamente, entre una misma neurona. Esta estructura recurrente la hace especialmente adecuada para estudiar la dinámica de sistemas no lineales [30].

Las redes neuronales retroalimentadas emulan más fielmente la estructura del cerebro humano, en donde los fenómenos de retroalimentación son fundamentales.

En una RNA con retroalimentación no hay una relación biunívoca entre una entrada y una salida. Permite considerar la historia previa de información que ha recibido la RNA para evaluar la salida [29].

Para que una red sea dinámica debe poseer memoria. Las RNR pueden usar su memoria interna para tramitar secuencias arbitrarias de entradas. Esto los hace aplicables en gran cantidad de procesos, sobre todo aquellos que tienen características dinámicas o de no linealidad.

Son especialmente útiles en aplicaciones tales como el reconocimiento de patrones secuenciales, cambiantes en el tiempo, ya que las capacidades de predicción y mapeo de las RNR así lo permiten [63].

1.7.1 Modelos con arquitectura parcialmente recurrente.

Las arquitecturas parcialmente recurrentes se originan al emplear redes feed-forward para la modelación de sistemas dinámicos. Estas se construyen a partir de una topología tipo feed-forward, sobre la que se consideran realimentaciones desde las capas de salida hacia las capas de entrada o hacia capas ocultas [65–67]

En esta categoría se incluyen las redes de Elman, las redes de Jordan y variantes de estas clases.

Red de Jordan

La primera red parcialmente recurrente fue propuesta por Jordan denominándose Red Secuencial de Jordan [65]. Esta estructura se forma en dos fases: en la primera fase se añaden conexiones recurrentes desde las salidas de la red hacia las unidades

contextuales Ci que forman una capa contextual, y en la segunda fase se realizan bucles (ciclos) sobre estas unidades [68].

En la Ilustración 5(b) se muestra un ejemplo de la red de Jordan. Nótese que las salidas asociadas con cada estado, se retroalimentan sobre las unidades contextuales que representan el siguiente estado del sistema, y trabajan en paralelo con las entradas. Los bucles sobre las unidades contextuales Ci proporcionan cierta memoria.

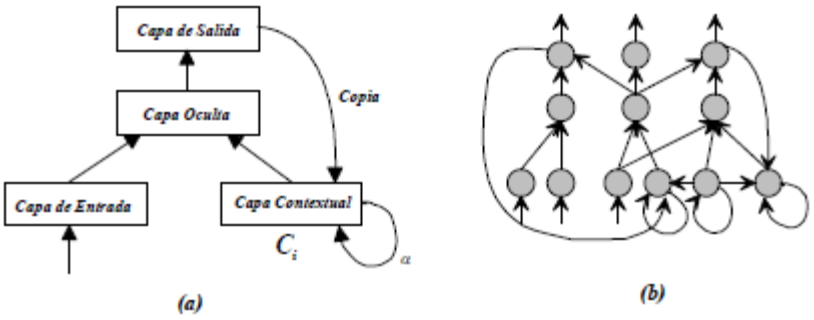


Ilustración 5: Red de Jordan: Modelo General (a) y Ejemplo (b).

En la red de Jordan, el aprendizaje tiene lugar sólo en las conexiones entre las unidades de entrada y las unidades ocultas, así como también entre las unidades ocultas y las de salida [69]. El aprendizaje se realiza de forma supervisada.

Red de Elman

Elman introdujo una arquitectura llamada Simple Recurrent Network (SRN) o Red Recurrente Simple [70]. En una red de este estilo, las conexiones de realimentación van desde la capa oculta a la capa contextual [65].

La red es muy parecida a la red de Jordán, excepto eso que, en lugar de las unidades de salidas, las unidades ocultas se retroalimentan sobre las unidades contextuales; y las unidades contextuales no tienen ninguna auto-conexión [69].

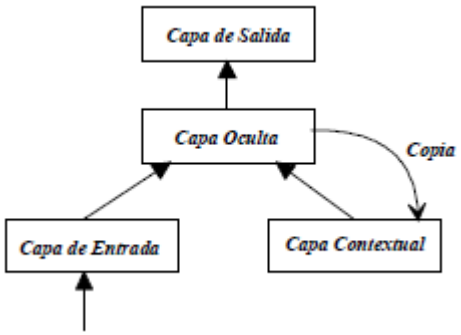


Ilustración 6: Estructura de una SRN de Elman.

La capa de entrada se puede considerar dividida en 2 partes: unidades de entrada y unidades contextuales. Las unidades contextuales simplemente toman una copia de las salidas de los nodos ocultos, lo que proporciona, en cierto modo, una memoria al sistema [70]. Generalmente se utiliza en reconocimiento y reproducción de secuencias (predicción de series temporales, reconocimiento de voz).

Red LRN

Demuth, Beale y Hagan [71], [72] proponen una versión ampliada de la red de Elman: Layer-Recurrent Network (LRN). En esta red, hay un lazo de información retroactiva con un retraso de tiempo, alrededor de cada capa de la red, excepto por la última capa. La red original Elman tuvo solo dos capas, y usaba una función de transferencia “tansig” para la capa oculta y una función de transferencia “purelin” para la capa de salida.

La LRN generaliza la red Elman para tener un número arbitrario de capas y tener funciones de transferencias arbitrarias en cada capa [71], [72]. La Ilustración 7 muestra una LRN de dos capas.

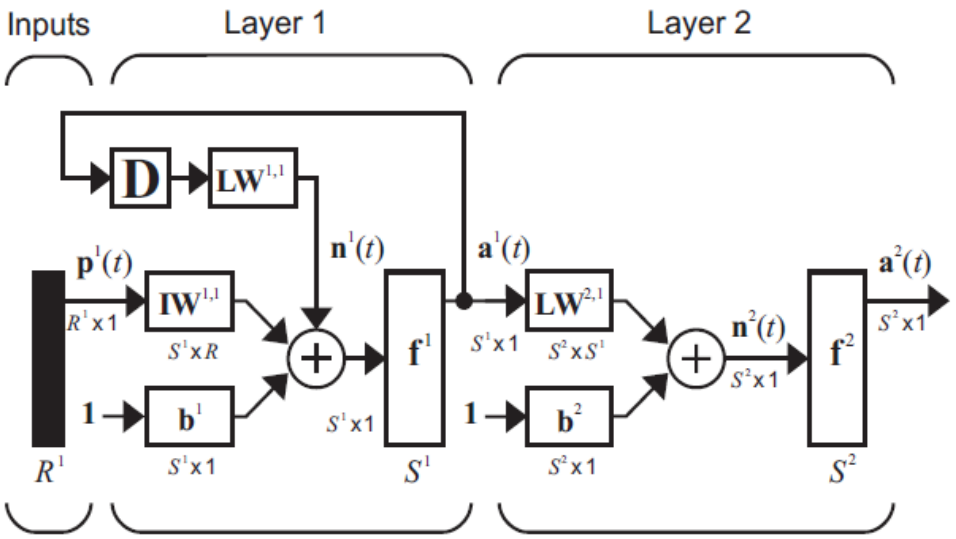


Ilustración 7: Estructura de una LRN de dos capas.

En la red Elman SRN y LRN el aprendizaje se realiza de forma supervisada.

Red NARX

Dentro de las redes parcialmente recurrentes se encuentran las redes feed-forward con realimentaciones o con sinapsis dinámica, Demuth, Beale y Hagan [71], [72] proponen la red autorregresiva no lineal con aportes exógenos (nonlinear autoregressive network with exogenous inputs - NARX) y algunas variantes de esta.

La red NARX es una red dinámica parcialmente recurrente, con conexiones de información retroactiva incluyendo varias capas de la red. El modelo NARX se basa en el modelo lineal ARX, que es comúnmente usado en modelado de series temporales [71], [72].

En la arquitectura NARX estándar, la salida de la red es realimentada a la entrada de la red, tal y como se muestra en la Ilustración 8.(a). Dado que la salida real del sistema está disponible durante el entrenamiento de la red (aprendizaje supervisado), se puede crear una arquitectura de series paralelo, tal y como se muestra en la Ilustración 8.(b), en al cual se utiliza la salida real del sistema en lugar de retroalimentar la salida estimada de la red. Esto tiene dos ventajas. Primero que la entrada a la red es más

precisa. Segundo, que la red resultante tiene una arquitectura meramente feedforward, y para su entrenamiento puede usarse un backpropagación estático [71], [72].

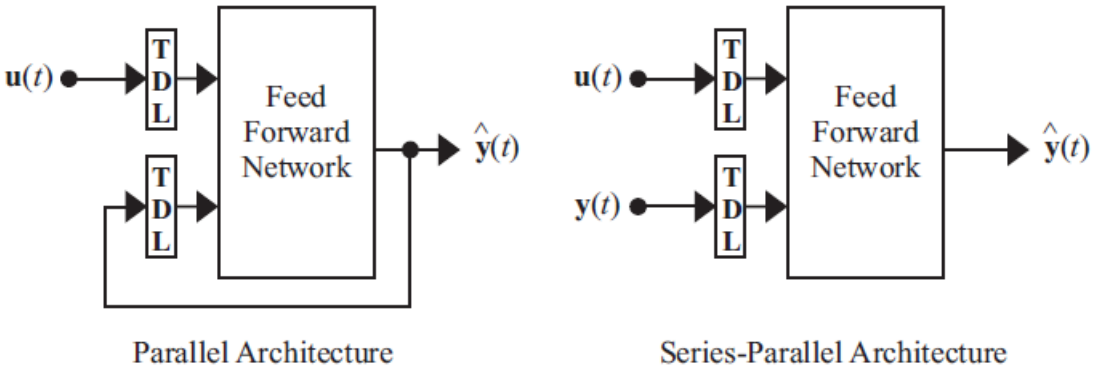


Ilustración 8: Arquitectura paralela (a) y arquitectura series paralelas (b).

Demuth, Beale y Hagan [72] proponen una vía de solución para reacomodar la red en la forma paralela original. La función closeloop convierte de la configuración de serie paralela (lazo abierto-open loop), para la configuración paralela (lazo cerrado-closed loop). La Ilustración 9 muestra la red NARX openloop y la Ilustración 10 muestra la red NARX closeloop.

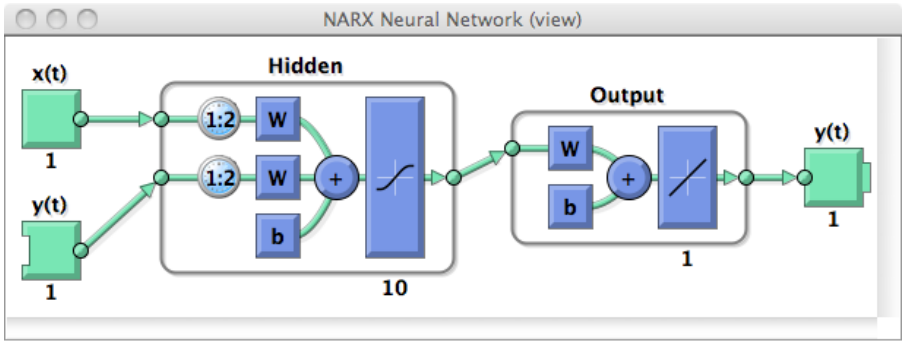


Ilustración 9: Arquitectura series paralela NARX-Open Loop.

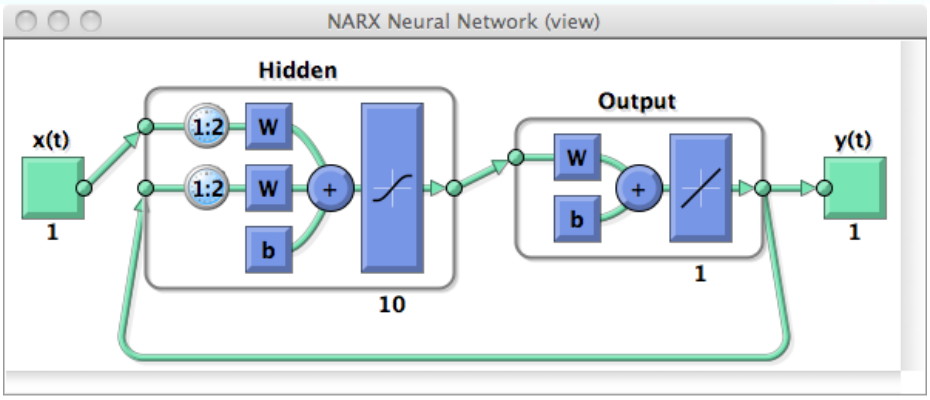


Ilustración 10: Arquitectura series paralela NARX-Closed Loop.

En una red de tipo NARX el aprendizaje se realiza de forma supervisada.

1.7.2 Modelos con arquitectura totalmente recurrente.

Las arquitecturas completamente recurrentes corresponden a estructuras neuronales, en las cuales se permite que cualquier neurona de la arquitectura presente conexiones hacia cualquier otra neurona.

La Ilustración 11 presenta un diagrama de tal arquitectura.

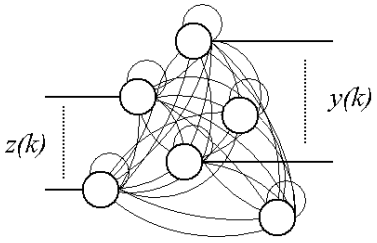


Ilustración 11: Arquitectura completamente recurrente.

En ella se distinguen neuronas de entrada y neuronas de salida. Así, si bien la arquitectura no presenta capas, es posible reordenar la estructura entre neuronas de entrada y salida y neuronas internas [73].

La arquitectura reorganizada se presenta en la Ilustración 12:

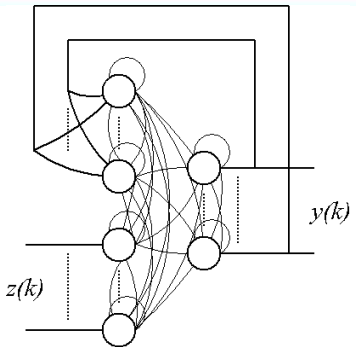


Ilustración 12: Arquitectura completamente recurrente reorganizada en capas.

Dentro de esta clasificación se encuentran las redes de Hopfield y sus variantes, las redes de tipo Teoría de la Resonancia Adaptativa (Adaptive Resonance Theory - ART), las redes de Hamming, entre otras.

Red de Hopfield

Las redes de Hopfield son un importante tipo de redes recurrentes, descritas en 1982 por J.J. Hopfield [74].

Obsérvese, en la Ilustración 13, que solo está formada de una capa: la capa Hopfield.

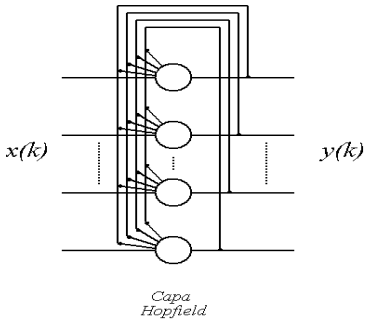


Ilustración 13: Red neuronal de Hopfield.

La red de Hopfield solo tiene una capa y las neuronas son iniciadas con el vector de entradas, entonces la red itera hasta que la salida converge. Cuando la red opera correctamente, la salida resultante debe ser uno de los vectores prototipos. En la red Hopfield, la neurona distinta de cero indica cuál patrón prototipo es seleccionado. Dicha

red produce la selección del patrón prototipo en su salida [75]. El aprendizaje en este tipo de redes se realiza de manera no supervisada.

Las redes de Hopfield han encontrado muchas aplicaciones, principalmente en memorias asociativas, problemas de optimización y búsqueda [76], [77], reconocimiento de patrones temporales [78] y otras aplicaciones de inteligencia artificial [79].

Red de Hamming

La red de Hamming fue diseñada para problemas binarios de reconocimiento de patrones (donde cada elemento del vector de entrada tiene solo dos posibles valores). Esta red en su arquitectura presenta capas feedforward y capas feedback [75].

La Ilustración 14 muestra una red Hamming estándar. Nótese que el número de neuronas de la primera capa es igual al número de neuronas de la segunda capa.

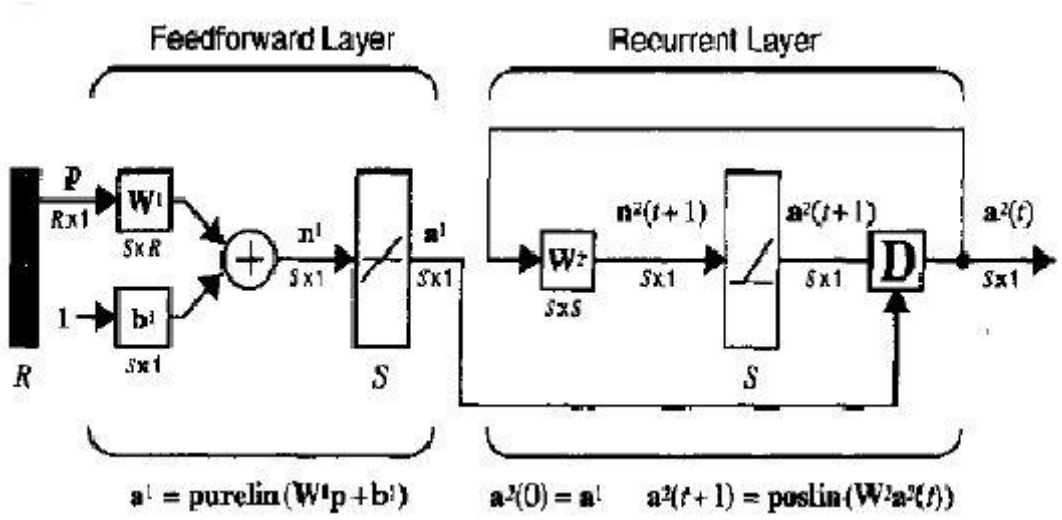


Ilustración 14: Arquitectura estándar de una red Hamming.

El objetivo de una red Hamming es decidir cuál vector prototipo está más cercano al vector de entradas. La decisión se toma a partir de la salida de la capa recurrente, la cual presenta una neurona por cada patrón prototipo. Cuando dicha capa converge, solo tendrá una neurona que su salida no es cero, esta neurona indica el patrón prototipo más cercano al vector de entradas [75].

Las redes Hamming realizan un aprendizaje supervisado.

Red ART

Carpenter y Grossberg [80] definieron la Teoría de Resonancia Adaptativa, aplicable a los sistemas competitivos, en los cuales cuando se presenta cierta información de entrada sólo una de las neuronas de salida se activa alcanzando su valor de respuesta máximo después de competir con otras**.**

ART se basa en resonar la información de entrada con los representantes o prototipos de las categorías que reconoce la red. Si entra en resonancia con alguno, es suficientemente similar, la red considera que pertenece a dicha categoría y únicamente realiza una pequeña adaptación del prototipo almacenado representante de la categoría para que incorpore características del dato presentado. Cuando no resuena con ninguno, no se parece a los recordados por la red y esta se encarga de crear una nueva categoría con el dato de entrada como prototipo de la misma [81].

Las redes de tipo ART realizan un aprendizaje no supervisado.

Otros modelos totalmente recurrentes

Otros modelos de redes totalmente recurrentes son las de tipo Activación Interactiva y Competencia (IAC por sus siglas en inglés), BSB (Brain State in a Box), BAM (Bidirectional Associative Memory) [82].

Tsoi y Back [67] señalan que las arquitecturas completamente recurrentes presentan problemas de estabilidad, convergencia lenta, y variaciones repentinas del error de estimación entre observaciones.

Muchos otros modelos de RNR fueron encontrados en la bibliografía. En la actualidad siguen apareciendo nuevos, pero los modelos tradicionales continúan siendo los más utilizados a nivel internacional en infinitas aplicaciones de la IA.

1.7.3 Algoritmos de entrenamiento.

Muchos algoritmos han sido desarrollados para el entrenamiento de las redes recurrentes, la mayoría de ellos basan su trabajo en el algoritmo Backpropagation, y este se basa en la técnica del gradiente y en la regla de aprendizaje del mínimo error cuadrado medio (LMS- *Least Mean Squared*).

- **Gradiente Descendente**

El método del gradiente descendente define una función $\boldsymbol{E(W)}$ que proporciona el error que comete la red en función del conjunto de pesos sinápticos W. El objetivo del aprendizaje será encontrar la configuración de pesos que corresponda al mínimo global de la función de error, aunque en muchos casos es suficiente encontrar un mínimo local lo suficientemente bueno [83].

El principio general del método es el siguiente: dado un conjunto de pesos $\boldsymbol{W(0)}$ para el instante de tiempo $\boldsymbol{t = 0}$, se calcula la dirección de máxima variación del error. La dirección de máximo crecimiento de la función $\boldsymbol{E(W)}$ en $\boldsymbol{W(0)}$ viene dado por el gradiente $\boldsymbol{\nabla E(W)}$. Luego, se actualizan los pesos siguiendo el sentido contrario al indicado por el gradiente $\boldsymbol{\nabla E(W)}$, dirección que indica el sentido de máximo decrecimiento [84]. De este modo se va produciendo un descenso por la superficie de error hasta alcanzar un mínimo local.

- **Error cuadrático medio**

La regla de aprendizaje LMS, trata de minimizar la diferencia entre valor deseado y obtenido en la salida de la red. Aunque la superficie de error es desconocida, el método de gradiente descendente consigue obtener información local de dicha superficie a través del gradiente [85].

Algoritmo Backpropagation

El algoritmo Backpropagation utiliza la misma técnica de aproximación en pasos descendientes que emplea el algoritmo LMS, la complicación está en el cálculo del gradiente, donde no se sabe lo lejos o lo cerca que se está del punto mínimo.

En la velocidad de convergencia del algoritmo influye el elegir un incremento adecuado a través de la tasa de aprendizaje $\boldsymbol{\alpha}$. Un valor pequeño de α significa que la red tendrá que hacer un gran número de iteraciones, si se toma un valor muy grande, los cambios en los pesos serán muy grandes, avanzando muy rápidamente por la superficie de error, con el riesgo de saltar el valor mínimo del error y estar oscilando alrededor de él, pero sin poder alcanzarlo. Es recomendable aumentar el valor de α a medida que disminuye el error de la red durante la fase de entrenamiento, para garantizar así una rápida convergencia, teniendo la precaución de no tomar valores demasiado grandes que hagan que la red oscile alejándose demasiado del valor mínimo [85].

Otra forma de incrementar la velocidad de convergencia consiste en utilizar una técnica llamada **momento**. Este término adicional tiende a mantener los cambios de peso en la misma dirección. Siendo $\boldsymbol{\beta}$ el parámetro de momento, el cual toma valores positivos menores a 1 y determina el efecto en $\boldsymbol{t+1}$del cambio de los pesos en el instante *t*. Con este momento se consigue la convergencia de la red en menor número de iteraciones, ya que si en *t* el incremento de un peso era positivo y en $\boldsymbol{t+1}$ también, entonces el descenso por la superficie de error en $\boldsymbol{t+1}$es mayor. Sin embargo, en *t* el incremento era positivo y en $\boldsymbol{t+1}$es negativo, el paso que se da en $\boldsymbol{t+1}$es más pequeño, lo cual es adecuado, ya que eso significa que se ha pasado por un mínimo y los pasos deben ser menores para poder alcanzarlo [85].

Durante la aplicación del algoritmo backpropagation, el aprendizaje se produce mediante la presentación sucesiva de un set de entrenamiento. Cada presentación completa del set de entrenamiento se denomina epoch. Así, el proceso de aprendizaje se repite epoch tras epoch hasta que los pesos sinápticos se estabilizan y la performance de la red converge a un valor aceptable [86].

Dependiendo del tipo de algoritmo de entrenamiento que utilice, es posible que las RNR requieran tiempos de procesamiento muy grandes. No obstante, a pesar de esto, la velocidad con la que convergen al resultado esperado ignorando máximos y/o mínimos locales sobrepasa con creces esta desventaja, en especial en redes con un número de neuronas reducido. En cuanto a su porcentaje de error, las RNR son más efectivas al detectar, clasificar y aprender nuevos patrones [63].

1.8 Conclusiones

- El uso e instalación de los sistemas de climatización centralizada todo agua, cada vez es más frecuente a nivel mundial, y también en Cuba, donde la perspectiva nacional es emplearlos en futuras construcciones hoteleras.
- En el hotel Jagua de la provincia de Cienfuegos se utiliza este tipo de sistema, pero la operación del mismo se realiza de forma empírica a partir del conocimiento y experiencia de los operarios, por lo que el consumo energético es elevado.
- Las RNA como técnica de la IA, resultan efectivas para modelar procesos dinámicos, específicamente las RNR en problemas de predicción, donde el factor tiempo cobra un papel fundamental.

Capítulo II: Diseño del experimento.

El presente capítulo abarca la identificación de los modelos para predecir la carga térmica, enfatizando en los circuitos Resistivos-Capacitivos a partir de dos de las estructuras presentes en la literatura, y se identifican los modelos de RNA adecuados para la modelación del sistema. Luego se precisa el comportamiento de las variables presentes en el proceso de estudio, y se realiza una breve descripción de la herramienta utilizada durante el desarrollo del mismo. Finalmente se realiza el experimento.

2.1 Identificación de los modelos para predecir cargas térmicas.

2.1.1 Circuitos Resistivos-Capacitivos.

Una de las técnicas más empleadas para estimar la carga térmica en una edificación, con sistema de climatización centralizada todo agua, y su modelo correspondiente es la equivalencia de circuitos eléctricos a partir de diferentes configuraciones resistivas-capacitivas. Este método es una de las alternativas más viables hoy día para el diseño de modelos de carga térmica [17], [39].

Cuando se modela una edificación como un circuito RC, las paredes de la misma se dividen en dos partes: pared interior y pared exterior. La pared exterior funciona como un capacitador que adquiere la temperatura del ambiente exterior a medida que pasa el tiempo. A su vez la pared interior se resiste a dejar entrar o adquirir la temperatura exterior, hasta que con el paso del tiempo sucede.

Un circuito RC trata de satisfacer determinada temperatura, teniendo en cuenta las características mencionadas anteriormente.

Existen dos propuestas de circuitos RC presentes en la literatura que son las más utilizadas a nivel internacional: Red RC de Braun y Red RC de Ghiaus.

Estructura del modelo de carga térmica a partir de la red RC de Braun.

La propuesta de Braun es posiblemente el esquema en redes RC de mayor impacto en el mundo, y ha sido validado en diferentes aplicaciones [87–89]. En la Ilustración 15 se presenta este modelo:

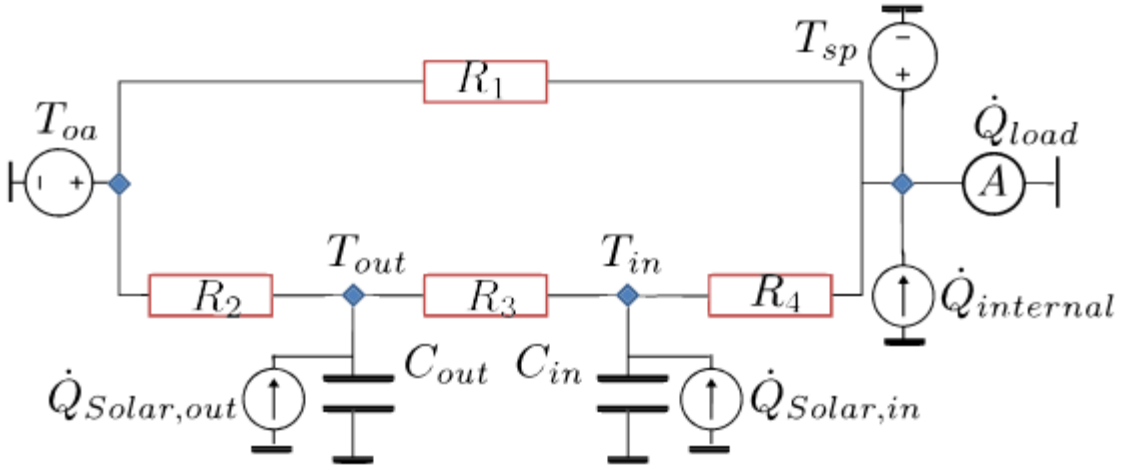


Ilustración 15: Circuito de Braun.

Las entradas del modelo son la temperatura ambiente *Toa*, la temperatura de *set-point Tsp*, y las radiaciones solares. *QSolar,out* y *QSolar,in* son la radiación solar global que incide directamente sobre el exterior de las paredes de la construcción y la radiación solar difusa que llega al interior de la habitación, respectivamente.

Estructura del modelo de carga térmica a partir de la red RC de Ghiaus.

Ghiaus y Hazyuk [38] proponen un esquema de red RC a partir por los estándares europeos. Este esquema es usado por el grupo *MIGRER* (*Intelligent Buildings and Rational Management of Renewable Energy*) en Francia. El autor determina la temperatura del edificio, basándose en el conocimiento previo de las características constructivas del edifico y la estimación de los elementos del circuito equivalente.

Para alcanzar esta meta, se deben conocer una serie de registros históricos de mediciones que permitan establecer el modelo, a partir de los datos medidos. En la Ilustración 16 se muestra la red propuesta por Ghiaus y Hazyuk [38].

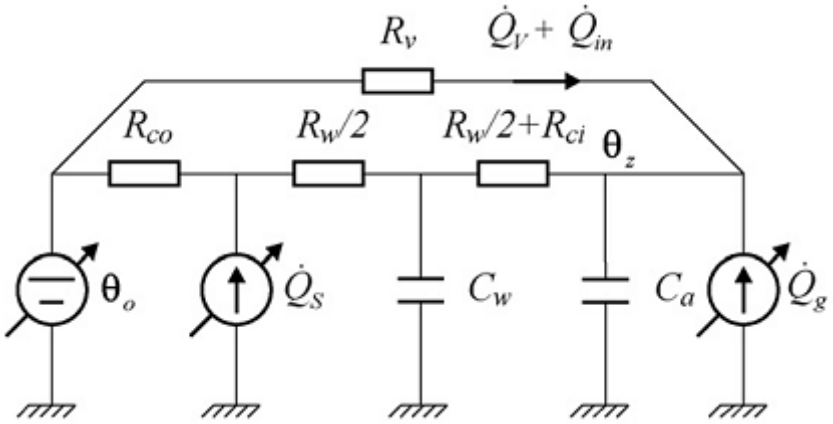


Ilustración 16: Circuito de Ghiaus.

Nótese que la temperatura ambiente y la radiación solar global son las principales perturbaciones del sistema. No se necesita conocer la radiación solar difusa, solo se requiere la radiación global [90].

El objetivo de ambos circuitos, es determinar la dinámica de la carga térmica existente en el edificio para satisfacer la exigencia de la temperatura de confort.

Por las propiedades estructurales de la edificación del hotel Jagua y por la zona geográfica en que se encuentra, es más conveniente modelar sobre la base de un circuito de Braun, donde la radiación global difusa forma parte de las variables o perturbaciones del sistema.

2.1.2 Identificación de los modelos de RNA.

En las aplicaciones del control del clima de edificaciones, la modelación dinámica de las cargas térmicas depende de: variables meteorológicas (temperatura ambiente, radiación solar), de ocupación y temperatura de set-point.

Maniobrar la carga de suministro (carga térmica) de forma dinámica, sin afectar el *confort* es sin duda una posibilidad inmejorable de elevar los índices de eficiencia.

Las soluciones estudiadas que utilizan RNA no consideran la masa de la edificación como una red térmica formada por resistencias y capacitancias, donde la modelación de

la carga térmica no solo depende de sus entradas (variable meteorológicas y de ocupación) para un instante de tiempo, sino también del pasado de estas. Esta es la característica que confiere dinamismo al proceso.

A diferencia de las redes feed-forward, las redes con arquitectura feed-back o redes neuronales recurrentes, crean un estado interno de la red que permite que exhiba comportamiento temporal dinámico.

Considerando la edificación como un circuito RC donde el factor tiempo juega un rol fundamental, y a partir de las variables presentes en el sistema, se hace necesario crear un modelo de RNR que permita predecir la carga de suministro necesaria (carga térmica). Esto permite garantizar un uso racional de la energía sin afectar el confort para los clientes.

Existen modelos totalmente recurrentes, en los cuales se permite que cualquier neurona de la arquitectura presente conexiones hacia cualquier otra neurona y todas estén interconectadas. Otros modelos son parcialmente recurrentes, estos contienen realimentaciones desde las capas ocultas o de salida hacia las capas de entrada, proporcionando cierta memoria y haciendo que su salida converja hacia un estado que depende, no solo de sus entradas para un instante de tiempo, sino también del pasado de estas. Las características proporcionadas por los modelos parcialmente recurrentes son las que se necesitan en este proyecto, por tanto son estos los que quedan propuestos para el desarrollo de los experimentos.

2.2 Comportamiento de las variables presentes en el proceso.

Los valores de las variables (temperatura ambiente, radiación solar global, radiación solar difusa y temperatura de set-point) presentes en el proceso, son datos de los meses julio y agosto del año 2011, en un período de muestreo para cada hora del día, obtenidos a partir de históricos anuales del Centro Meteorológico Provincial y del hotel Jagua. En dichos meses la carga necesaria que precisan los sistemas de climatización, es mucho mayor, y por tanto se requiere métodos alternativos para un uso eficiente de la energía eléctrica, sin dejar a un lado la satisfacción del cliente.

Los valores de la variable 'carga térmica', son datos de los meses julio y agosto del año 2011, en un período de muestreo para cada hora del día, obtenidos a partir de la simulación del proceso en TRNSYS, y que presentan una dinámica en su comportamiento para cada hora del día.

TRNSYS es uno de los programas de simulación aceptados en la lista de estándar europeo de sistemas térmicos *(ENV-12977-2)*. El alto nivel de detalle de *TRNBuild* (*TRNSYS Building*), conocido como "*Type 56*", es compatible con los requisitos exigidos por *ANSI/ASHRAE Standard 140-2001* [91]. Durante la pasada década, TRNSYS ha sido ampliamente empleado en simulaciones de sistemas energéticos. Estudios comparan el rendimiento del software frente a resultados experimentales, así como los resultados simulados con otros programas para edificaciones certificados en la industria, como *EnergyPlus*.

A partir de las variaciones del clima y la dinámica de ocupación del edificio, TRNSYS es capaz de determinar la dinámica de la afectación en la zona térmica, pero la técnica de identificación de sistemas que utiliza es llamada caja-negra (*black-box*) donde no se conoce la estructura del modelo y solo se cuenta con bases de datos de las entradas y las salidas. Es utilizada frecuentemente como última alternativa para sistemas no lineales o lineales bajo ciertas consideraciones. Por esta razón y además por ser un software privativo, se hace necesario acudir a otros alternativas.

Es necesario aclarar que uno de los factores a tener en cuenta en un proceso de simulación energética de sistemas de climatización en una edificación, es la estrategia de ocupación de dicha edificación. En el caso del hotel Jagua, hoy día, la realidad del comportamiento de la ocupación no está registrada, por tanto la simulación del proceso en TRNSYS toma la ocupación en su caso más crítico, o sea a un 100%.

Comportamiento de la temperatura y la radiación solar.

El continuo monitoreo de las variables meteorológicas es una medida que siempre debe estar presente a la hora de analizar el comportamiento de la dinámica de la carga térmica de cualquier edificación.

Los valores de la temperatura ambiente, la radiación solar global, y de la radiación solar difusa, sobre la superficie terrestre en la provincia de Cienfuegos en el mes de julio del año 2011, aparecen reflejados en los **Anexos 1, 2 y 3** respectivamente.

Los valores de la temperatura ambiente, la radiación solar global, y de la radiación solar difusa, sobre la superficie terrestre en la provincia de Cienfuegos en el mes de agosto del año 2011, aparecen reflejados en los **Anexos 4, 5 y 6** respectivamente.

Los valores de la temperatura de set-point aparecen reflejados en el **Anexo 7**.

El hecho de que los datos sean el resultado de años anteriores no representa una mala predicción meteorológica. Fuentes, tales como Gwerder y Tödtli [92], Siroký y Oldewurtel [17], y Meteotest [93], plantean que el clima para un período de 10 años en fechas simultáneas sufre variaciones inferiores al 5% de la media estimada, la temperatura media tiene un error de 1.5°C y la radiación solar de 15W/m2 [93]; por tanto, un análisis de las bases de datos recientes en concordancia con afectaciones severas del clima futuro dan una buena medida del comportamiento de variables medioambientales como la temperatura ambiente y la radiación solar.

Comportamiento de la carga térmica.

El perfil térmico presenta una dinámica en su comportamiento para cada hora del día. La carga térmica (Q) necesaria para lograr el confort, depende de las variables climatológicas, no solo para determinado momento en el tiempo sino que depende también, del pasado de estas variables y del suyo propio, a causa de las propiedades físicas que definen la edificación como un circuito RC.

Los valores de carga térmica que se utilizan en los experimentos fueron obtenidos a través de la simulación en TRNSYS, a partir de una investigación precedente desarrollada por Bombino[33]. Dichos valores aparecen reflejados en el **Anexo 8**.

2.3 Descripción de la herramienta utilizada para el desarrollo del experimento: Matlab.

Existen softwares de desarrollo que aprovechan las ventajas del cálculo computacional para aplicar las técnicas de identificación, siendo *Matlab* uno de los programas autenticados internacionalmente y con mayor utilidad en diversos campos.

Matlab (MATrix LABoratory, Laboratorio de Matrices) es un lenguaje de alto rendimiento para la informática técnica. Es un paquete de software que permite hacer matemática y cómputo, visualización, análisis de datos, desarrollo de algoritmos, simulación, modelado, y programar en un ambiente abierto y flexible. Ofrece un lenguaje intuitivo a ingenieros, científicos y matemáticos para expresar problemas y sus soluciones matemáticas y gráficas [94].

Matlab es un entorno de computación y desarrollo de aplicaciones, totalmente integrado, orientado para llevar a cabo proyectos en donde se encuentren implicados un gran número de cálculos matemáticos y se requiera la visualización gráfica de los mismos. Integra análisis numérico, cálculo matricial, proceso de señales y visualización gráfica [95].

En Matlab el elemento básico de almacenamiento de información, es la matriz que tiene una característica fundamental y es que no necesita dimensiones. Esto le permite resolver varios problemas de computación técnica (especialmente aquellos que tienen formulaciones matriciales y vectoriales) en una fracción de tiempo similar al que se gastaría, cuando se escribe un programa en un lenguaje no interactivo como C o FORTRAN. Además cuenta con un lenguaje de programación propio (lenguaje M).

Matlab se ha desarrollado sobre un período de años con entradas provenientes de muchos usuarios, en los entornos universitarios. Es la herramienta instructiva estándar para cursos avanzados e introductorios en matemáticas, ingeniería y ciencia. En la industria es la herramienta escogida para investigación de alta productividad, desarrollo y análisis.

Desde sus inicios, Matlab ha estado renovándose. Producto del interés en distintas áreas han surgido gran cantidad de funciones específicas para cada área, que vienen agrupadas en paquetes denominados toolbox. Estos "toolboxes" o "cajas de herramientas" son colecciones de funciones para Matlab desarrolladas para resolver problemas de un tipo particular [96]. Algunos de los toolboxes de las versiones más actuales son: bioinformática, identificación de sistemas, lógica difusa, optimización, procesamiento de imágenes, procesamiento de señales, adquisición de datos, estadística, redes neuronales, entre otros.

Algunas de las aplicaciones de Matlab pueden ser:

- Análisis matemáticos.
- Simulaciones y Modelado.
- Cálculo simbólico.
- Desarrollo de algoritmos.
- Modelado.
- Exploración, visualización y análisis de datos.
- Creación de gráficas.

Estas características que se han comentado hacen de Matlab una herramienta de trabajo muy extendida entre los estudiantes, técnicos e investigadores [97].

En esta investigación se hace uso del Matlab en su versión r2012b como herramienta durante las fases: desarrollo del experimento (en el Capítulo 2) y análisis de los resultados (en el Capítulo 3).

2.4 Desarrollo del experimento.

La experimentación consiste en probar con diferentes modelos de RNR, específicamente aquellos que presenten una arquitectura parcialmente recurrente, con el objetivo de evaluar su aplicabilidad en la predicción de la carga térmica en el proceso de climatización centralizada, todo agua, utilizado en el hotel Jagua.

Para cada modelo se realizan diferentes variaciones en algunos de los parámetros como: cantidad de capas ocultas, cantidad de neuronas en las capas ocultas y cantidad de iteraciones. De estas variantes, se toman las de mejores resultados y a estas se le varían los algoritmos de entrenamiento, las funciones de transferencia y aprendizaje, en busca de que mejoren aún más.

La experimentación se desarrolla con tres modelos de RNR: Red ELMAN, LRN y NARX. Se realizan variaciones a cada uno de estos modelos hasta lograr buenos resultados.

2.4.1 Generalidades sobre la experimentación en Matlab.

2.4.1.1 Etapas del experimento.

La experimentación queda dividida en dos etapas: la primera etapa es para la creación y entrenamiento de la red y en la segunda etapa (etapa de prueba) se simula la red para ver el comportamiento de esta ante nuevos valores de entradas.

Etapa 1: Creación y entrenamiento de la red

- Paso #1: se cargan los datos para la creación de la red.

En esta etapa, las variables de entrada y de salida deseada (target) presentan un comportamiento para cada hora del día, durante el mes de julio del año 2011 por ser uno de los meses más calurosos del año y por ello el consumo de energía es mayor. Se tienen 744 datos para cada variable.

Estos datos son obtenidos como vectores columna, y se le asignan a nuevas variables traspuestos como vectores filas, por cuestiones de filosofía de Matlab.

- Paso #2: pre-procesamiento de los datos.

El entrenamiento de redes neuronales puede ser más eficiente si se realiza un pre-procesamiento en los datos de entradas y en targets de la red, así lo demuestran Demuth, Beale y Hagan [71], [72] y Marín [98].

Las funciones de procesamiento transforman las entradas y las salidas dadas a la red de manera más eficiente para su entrenamiento. Las salidas obtenidas de la simulación en la etapa de entrenamiento son revertidas a las características de sus datos originales.

Se pueden pasar los datos pre-procesados como parámetros, cuando se llama a la función de creación de la red, o después de que se crea la red durante la etapa de entrenamiento [71], [72].

Antes de entrenar la red, es útil modificar las entradas y las salidas que le serán dadas, a fin de que siempre caigan dentro de un alcance especificado. La función *mapminmax* modifica las entradas y los targets en el intervalo de -1 a 1 [-1,1].

El siguiente código muestra cómo usar esta función.

```
[pn,ps] = mapminmax(p);

[tn,ts] = mapminmax(t);
```

Las entradas y las salidas originales son dadas a la función en los matrices p y t respectivamente.

Las entradas y las salidas que son devueltos, pn y tn respectivamente, caerán en el intervalo [-1,1], y las variables ps y ts contienen la estructura de estos, incluyendo los valores mínimos y máximos de las entradas y los targets originales [71], [72].

Con los modelos experimentados durante esta investigación se pre-procesan los datos de entrada: la variable "Entradas_Etapa1", contiene ahora los datos de entrada en el intervalo [-1,1], y su estructura se guarda en la variable "estructura_entrada_Etapa1". También se pre-procesan los datos de salida: la variable "Target_Etapa1" contiene ahora los datos de salida en el intervalo [-1,1], y su estructura se guarda en la variable "EstructuraTarget_Etapa1".

- Paso #3: Conversión de consecutivo a secuencial.

Cuando existe una secuencia de tiempo para los valores de entrada y salida de la red y es necesario un orden de prioridad, se convierten estos valores a una secuencia con un orden lógico. La función *con2seq* se encarga de ello.

- Paso #4: creación de la red.

Existen muchas funciones para la creación de una RNR, según el modelo que se desee. Por ejemplo: '*elmannet'* crea el modelo de tipo Elman; '*layrecnet'* crea el modelo de tipo LRN; '*narxnet*' crea el modelo de tipo NARX aunque existen otras variantes para la creación de este modelo. De manera general, cada una de estas funciones recibe como argumento los delays de la red, la cantidad de neuronas en cada capa, y la función de entrenamiento. La utilización de cada una es ampliada durante el desarrollo de cada modelo utilizado para el experimento.

La función preparets utilizada durante este paso, se encarga de redefinir las entradas y los targets de la red, teniendo en cuenta los delays definidos, para simular su comportamiento.

Los delays representan los retrasos de tiempo que toman los modelos, es decir definen los valores pasados, de las entradas y los targets. En cada modelo se definen como delays los dos primeros valores, tanto de las entradas como de los targets. Es necesario aclarar que solo se toman dos valores porque así lo recomiendan Demuth, Beale y Hagan [72], [75], quienes enfatizan en cómo a partir de cinco delays disminuye la velocidad de convergencia.

- Paso #5: entrenamiento de la red.

La función *train* entrena la red creada durante el paso#4.

Las funciones de transferencia y el algoritmo de entrenamiento que se utiliza, depende del modelo de red que se implementa y de las características del sistema que se modela. Matlab define por defecto estos parámetros de acuerdo al modelo de red que se entrena, pero antes de realizar el entrenamiento, se pueden definir los parámetros de entrenamiento deseados, así como las funciones de transferencia y de aprendizaje. En

cada modelo varían dichos parámetros y funciones, con el objetivo de mejorar la respuesta de la red.

Se conocen las entradas al modelo y la salida que se desea obtener; por tanto se realiza un entrenamiento supervisado, con el objetivo de que la red aprenda a partir de patrones de comportamiento de los datos.

El siguiente código muestra cómo utilizar la función *train*:

net = train(net,Xs,Ts,Xi,Ai);

Donde:

Xs: nuevo vector de entradas (definido durante la creación).

Xi: delays de las entradas.

Ai: delays del target.

Ts: nuevo vector target (definido durante la creación).

Luego del entrenamiento de la red, se definen los valores de los pesos y bios adecuados para la simulación y con los que la red da una mejor respuesta.

- Paso #6: simulación de la red.

La función *sim* simula la red creada y entrenada en los pasos 4 y 5 respectivamente. El siguiente código muestra cómo utilizar dicha función:

yobtenida = sim(net,Xs,Xi,Ai)

Donde:

Xs: nuevo vector de entradas (definido durante la creación).
Xi: delays de las entradas.
Ai: delays del target.

La salida resultante de la simulación se almacena en la variable “yobtenida”.

Luego se realza el post-procesamiento de los datos almacenados en “yobtenida”.

Si la función *mapminmax* se usa para ubicar los datos de las salidas dadas en el intervalo de -1 a 1, entonces la red será entrenada para producir salidas a ese mismo intervalo [71], [72]. Para convertir estas salidas a las mismas unidades que fueron destinadas en las salidas originales, uso la estructura "EstructuraTarget_Etapa1".

[y] = mapminmax('reverse',yobtenida,EstructuraTarget_Etapa1);

Finalmente se grafican todos los resultados obtenidos.

El código generalizado de la implementación, en Matlab, de la Etapa 1 aparece reflejado en el Anexo 9.

Etapa 2: Prueba

En esta etapa se simula la red creada y entrenada durante la etapa1, para medir el comportamiento de la misma ante nuevos valores de entrada y nuevos targets.

- Paso #1: se cargan los datos para las nuevas entradas y salidas de la red.

En esta etapa, las variables de entrada y de salida deseada (target), presentan un comportamiento para cada hora del día, durante el mes de agosto del año 2011 por ser uno de los meses más calurosos del año y por ello el consumo de energía es mayor. Se tienen 744 datos para cada variable.

- Paso #2: pre-procesamiento de los datos.

Se pre-procesan los datos que forman parte de las nuevas entradas y los nuevos targets. Este paso se realiza de igual forma que el paso #2 de la etapa1.

En los modelos desarrollado en esta investigación se pre-procesan los datos de entrada: la variable "Entradas_Etapa2", contiene ahora los datos de entrada en el intervalo de -1 a 1, y su estructura se guarda en la variable "estructura_entrada_Etapa2". También se pre-procesan los datos de salida: la variable "Target_Etapa2" contiene ahora los datos de salida en el intervalo de -1 a 1, y su estructura se guarda en la variable "EstructuraTarget_Etapa2".

- Paso #3: Conversión de consecutivo a secuencial.

Este paso se realiza de igual forma que el paso #3 de la etapa1.

- Paso #4: simulación

Se simula la red, para nuevos valores de entradas y nuevos valores targets. Este paso se realiza de igual forma que el paso #6 de la etapa1.

La función preparets utilizada durante este paso y durante el paso #4 de la etapa 1, se encarga de redefinir las entradas y los targets de la red, teniendo en cuenta los delays definidos, para simular su comportamiento.

Luego se realiza el post-procesamiento de los datos obtenidos durante la simulación, de la misma forma que se realiza durante la etapa1.

Finalmente se grafican todos los resultados.

El código generalizado de la implementación, en Matlab, de la Etapa 2 aparece reflejado en el Anexo 10.

2.4.1.2 Generalidades de los modelos: arquitectura y entrenamiento.

Entradas y salida de la red

Los problemas de predicción, consisten en hallar la estimación de una variable continua de salida, a partir de la presentación de un conjunto de variables predictivas de entradas (discretas y/o continuas) [98].

Las variables deben seguir una distribución normal o uniforme, y el rango de posibles valores debe ser aproximadamente el mismo y acotado dentro del intervalo de trabajo de la función de activación empleada en las capas ocultas y de salida de la red neuronal. Así, las variables de entrada y salida suelen acotarse en valores comprendidos entre 0 y 1 ó entre −1 y 1. se trata es de incluir en el modelo las variables predictivas que realmente predigan la variable dependiente o de salida [98].

La cantidad de neuronas de entradas es equivalente a la cantidad de variables independientes que existan en el problema, en este caso son cuatro neuronas de entradas.

La cantidad de neuronas en la capa de salida es equivalente a la cantidad de variables dependientes que existan en el problema, en este caso es una neurona en la capa de salida.

Las neuronas de entradas para todos los modelos representan las variables presentes en el proceso. Las variables son:

- la temperatura ambiente
- la temperatura de *set-point*
- la radiación solar global que incide directamente sobre el exterior de las paredes de la construcción
- la radiación solar difusa que llega al interior de la habitación

La neurona en la capa de salida, representa la carga necesaria para lograr el confort (carga térmica).

Capas Ocultas

Varios investigadores como: Demuth, Beale y Hagan [75], Tanco [85], Federico [86] y Marín[98], enfatizan en que una red neuronal es eficiente con hasta dos capas ocultas en su arquitectura. Raramente se han desarrollado modelos de RNA con más de dos capas ocultas. Por tal razón los modelos propuestos varían su arquitectura con 1 o 2 capas ocultas, según sus características.

La selección de la cantidad de neuronas en las capas ocultas es un área en la que aún queda mucho por indagar. Por tanto, dicha selección queda en manos de la experiencia de los investigadores y es depende en gran medida de la experimentación. Algunas investigaciones precedentes relacionadas con el tema en estudio, tales como Montelier [14] y Visanzay [99], muestran como una red de entre 1 y 4 neuronas en la capa de salida, es eficiente con entre 4 y 25 neuronas aproximadamente para las capas ocultas.

Durante el experimento se toman 15 neuronas para las arquitecturas de una capa oculta, y se toman 10 y 4 neuronas, para arquitecturas de 2 capas ocultas.

Pesos y Bios

Los pesos son la intensidad o fortaleza con que se conectan las neuronas dentro de la red. La función umbral o bios representa el valor que la neurona debe sobrepasar para activarse. Demuth, Beale y Hagan [71], [72], [75], Tanco [85], Federico [86], y Marín[98], recomiendan la inicialización de estos aleatoriamente con valores pequeños entre -1 y 1.

Es importante destacar que para cada una de las variantes de cada modelo se hacen corridas a la red generando los pesos y los bios aleatoriamente, hasta encontrar un mejor resultado. Una vez obtenido dicho resultado, se captan los valores de los pesos y de los bios y se le pasan a la red. Finalmente cada modelo tiene sus propios valores de pesos y bios.

Funciones de transferencia

Demuth, Beale y Hagan [71], [72], [75], Tanco [85], Federico [86], y Marín[98], recomiendan dos formas básicas para la función de activación o de transferencia, de las neuronas ocultas y/o de salida en una red multicapa: la función sigmoidal (tangente hiperbólica Ilustración 17 o logística Ilustración 18), y la función lineal (o identidad) Ilustración 19.

Para aprovechar la capacidad de las redes neuronales de aprender relaciones complejas o no lineales entre variables, se recomienda la utilización de funciones no lineales al menos en las neuronas de la capa oculta [98]. Por tanto, en la presente investigación se utiliza la función sigmoidal (logística o tangente hiperbólica) como función de activación en las neuronas de la capa oculta.

La elección de la función de activación en las neuronas de la capa de salida dependerá del tipo de tarea impuesto. En tareas de predicción o aproximación de una función, generalmente se emplea la función de activación lineal para la capa de salida.

TANSIG (Hyperbolic tangent sigmoid transfer function)

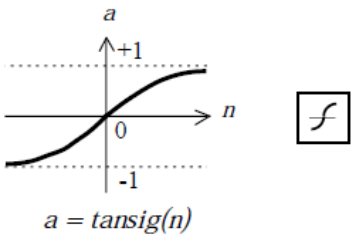


Ilustración 17: Función sigmoidal tangente hiperbólica.

Descripción: *tansig* es una función comúnmente usada en redes multicapa, como función de transferencia o activación, para las capas ocultas. Esta función toma los valores de entrada, los cuales pueden oscilar entre más y menos infinito, y restringe la salida a valores entre -1 y 1, de acuerdo a la expresión: $\boldsymbol{a} = \frac{e^{n}-e^{-n}}{e^{n}+e^{+n}}$

LOGSIG (Hyperbolic tangent sigmoid transfer function)

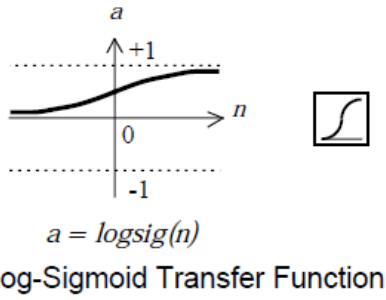


Ilustración 18: Función sigmoidal logística.

Descripción: *logsig* es una función comúnmente usada en redes multicapa, como función de transferencia o activación, para las capas ocultas. Esta función toma los valores de entrada, los cuales pueden oscilar entre más y menos infinito, y restringe la salida a valores entre 0 y 1, de acuerdo a la expresión: $\boldsymbol{a} = \frac{1}{1+e^{-n}}$

PURELIN (Linear transfer function)

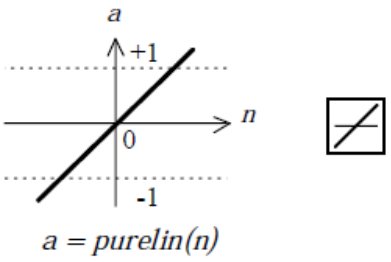


Ilustración 19: Función lineal.

Descripción: *purelin* es una función comúnmente usada en redes multicapa, como función de transferencia o activación, para la capa de salida. En esta función la salida es igual a su entrada: $\boldsymbol{a = n}$

Algoritmos de entrenamiento

El algoritmo de entrenamiento utilizado es el backpropagation, que implementa el método del gradiente descendente basado en la técnica del mínimo error cuadrático medio, tal como se explica durante el Epígrafe 1.7 del Capítulo 1 de la presente investigación.

Cuando se crea una red recurrente (ya sea una red ELMAN o LRN, o de tipo NARX), Matlab utiliza por defecto la función de entrenamiento TRAINLM. Dicha función es muy rápida pero requiere mucha memoria y grandes características de hardware para su corrida.

A pesar de que la función TRAINLM es muy eficiente para el entrenamiento de redes recurrentes, solo es recomendable su uso cuando las condiciones de hardware así lo permiten (por ejemplo como mínimo necesita 4-core 2.8 GHz Intel i7 con 12 GB RAM). La utilización de dicha función bajo condiciones extremas, causa errores inesperados en el funcionamiento de una red neuronal [71], [72].

Por tales razones, el uso de la función TRAINLM para el entrenamiento de redes recurrentes en esta investigación queda descartado.

Demuth y otros [71], [72], recomiendan cuatro funciones en Matlab para el entrenamiento de redes recurrentes, que de manera general, adaptan el algoritmo backpropagation en su implementación.

TRAINGD (Gradient descent backpropagation)

Descripción: *traingd* es una función de entrenamiento que actualiza los valores de los pesos y los bios acorde al método del gradiente descendente, y utiliza el algoritmo backpropagation.

El entrenamiento ocurre acorde a los siete parámetros principales de la función triangd, que se muestran a continuación con los valores que toman por defecto:

net.trainParam.epochs 10 (número máximo de iteraciones)

net.trainParam.goal 0 (meta del error)

net.trainParam.lr 0.01 (rango de aprendizaje)

net.trainParam.max_fail 5 (máximo de fracasos de validación)

net.trainParam.min_grad 1e-10 (mínimo de error del gradiente)

net.trainParam.show 25 (despliegues con que se muestra el comportamiento de las iteraciones)

net.trainParam.time inf (tiempo máximo de entrenamiento en segundos)

El valor de la tasa o rango de aprendizaje (lr) controla el tamaño del cambio de los pesos en cada iteración. Se deben evitar dos extremos: un ritmo de aprendizaje demasiado pequeño puede ocasionar una disminución importante en la velocidad de convergencia y la posibilidad de acabar atrapado en un mínimo local; en cambio, un ritmo de aprendizaje demasiado grande puede conducir a inestabilidades en la función de error, lo cual evitará que se produzca la convergencia debido a que se darán saltos en torno al mínimo sin alcanzarlo.

Demuth, Beale y Hagan [75], Tanco [85], Federico [86], y Martín y Sanz [82], en sus investigaciones, explican que se debe elegir un ritmo de aprendizaje lo más grande posible sin que provoque grandes oscilaciones. Dichos autores recomiendan asignar un valor inicial de la tasa de aprendizaje comprendido entre 0.05 y 0.5.

El parámetro Show muestra el comportamiento del entrenamiento para cada iteración del algoritmo. (Si el valor de la función es modificado para NaN, entonces el comportamiento del entrenamiento nunca se muestra.)

Los demás parámetros determinan cuando finaliza el entrenamiento, esto ocurre si se cumple alguna de las siguientes condiciones:

•Se alcanza el máximo número de iteraciones (epochs).

•Se excede el tiempo máximo del entrenamiento (time).

•El error alcanza su meta (goal).

•El error del gradiente cae debajo del mínimo (min_grad).

•El error de validación (max_fail) ha incrementado más desde la última vez que decreció.

TRAINGDA (Gradient descent with adaptive learning rate backpropagation)

Descripción: *traingda* es una función de entrenamiento que actualiza los valores de los pesos y los bios acorde al método del gradiente descendente con rango de aprendizaje adaptable.

La función traingda implementa el algoritmo Backpropagation con un rango de aprendizaje adaptable, el cual es utilizado como con la función traingd, excepto por los parámetros adicionales max_perf_inc, lr_dec, y lr_inc.

En cada iteración, si la función de error decrece hacia la meta, entonces el rango de aprendizaje es aumentado por el factor lr_inc. Si la función de error incrementa por el

factor max_perf_inc, el rango de aprendizaje es ajustado por el factor lr_dec y no se realizan cambios que incrementen la función de error.

La función del algoritmo es muy sensible al ajuste correcto del rango de aprendizaje. Si el rango de aprendizaje es demasiado alto, el algoritmo puede oscilar y puede volverse inestable. Si el rango de aprendizaje es demasiado pequeño, el algoritmo tarda demasiado tiempo para converger. No es práctico determinar el ajuste óptimo para el rango de aprendizaje antes de entrenarse, y, de hecho, los cambios óptimos del rango de aprendizaje se realizan durante el entrenamiento.

Un rango de aprendizaje adaptable trata de mantener el tamaño del paso de aprendizaje tan largo como sea posible mientras se mantenga estable el aprendizaje. El rango de aprendizaje es receptivo a la complejidad de la superficie local de error.

Un rango de aprendizaje adaptable requiere algunos cambios en el método de entrenamiento usado por traingd. Primero, se calcula la salida inicial de la red y el error. En cada iteración se calculan los nuevos pesos y los bios, usando el rango de aprendizaje. Luego se calculan las nuevas salidas y los errores.

Al igual que con momentum, si el nuevo error excede el viejo error por más que la proporción predeterminada, max_perf_inc (típicamente 1.04), se descartan los nuevos pesos y bios. Además, decrece el rango de aprendizaje (típicamente multiplicando por lr_dec = 0.7). De otra manera, los nuevos pesos y bios, y el rango de aprendizaje, se mantienen. Si el nuevo error es menor que el viejo error, aumenta el rango de aprendizaje (típicamente multiplicando por lr_inc = 1.05).

Este procedimiento aumenta el rango de aprendizaje, pero sólo hasta el grado que la red puede aprender sin incrementos abrumadores de error. Así se obtiene un óptimo cercano del rango aprendizaje, para el terreno local. Cuando un mayor rango de aprendizaje podría dar como resultado aprendizaje estable, este continúa aumentando. Cuando el rango de aprendizaje es demasiado alto para garantizar una disminución del error, entonces el rango de aprendizaje es disminuido hasta reanudar el aprendizaje estable.

El entrenamiento ocurre acorde a los parámetros de la función triangda, que se muestran continuación con los valores que toman por defecto:

net.trainParam.epochs 10 (máximo número de iteraciones)

net.trainParam.goal 0 (meta del error)

net.trainParam.lr 0.01 (rango de aprendizaje)

net.trainParam.lr_inc 1.05 (tamaño para aumentar el rango de aprendizaje)

net.trainParam.lr_dec 0.7 (tamaño para disminuir el rango de aprendizaje)

net.trainParam.max_fail 5 (máximo de fracasos de validación)

net.trainParam.max_perf_inc 1.04 (máximo de incremento del error)

net.trainParam.min_grad 1e-10 (mínimo de error del gradiente)

net.trainParam.show 25 (despliegues con que se muestra el comportamiento de las iteraciones)

net.trainParam.time inf (tiempo máximo de entrenamiento en segundos)

El entrenamiento termina según las mismas condiciones de la función *traingd*.

TRAINGDM (Gradient descent with momentum backpropagation)

Descripción: *traingdm* es una función de entrenamiento que actualiza los valores de los pesos y los bios acorde al método del gradiente descendente con momento, y utiliza el algoritmo backpropagation.

Gradiente descendente con momento, implementado por *traingdm*, deja a la red reaccionar no sólo ante el gradiente local, sino también a las tendencias en la superficie de error. Actuando como un filtro, momento que deja a la red ignorar pequeñas características en la superficie de error. Sin momento, una red puede quedarse

estancada en un mínimo local. Con momento, la red se puede deslizar a través del mínimo.

Gradiente descendente con momento depende de dos parámetros de entrenamiento. El parámetro 'lr' indica el rango de aprendizaje, similar al gradiente descendente simple. El parámetro 'mc' es la constante momento que toma valores entre 0 (ningún momento) y 1 (montones de momento).

El factor momento (mc) acelera la convergencia de los pesos. Demuth Beale y Hagan [75], Tanco [85], Federico [86], y Martín y Sanz [82], recomiendan asignar un valor inicial próximo a 1 (por ejemplo, 0.9).

El entrenamiento ocurre acorde a los parámetros de la función triangdm, que se muestran a continuación con los valores que toman por defecto:

net.trainParam.epochs 10 (máximo número de iteraciones)

net.trainParam.goal 0 (meta del error)

net.trainParam.lr 0.01 (rango de aprendizaje)

net.trainParam.max_fail 5 (máximo de fracasos de validación)

net.trainParam.mc 0.9 (valor de la variable momentum)

net.trainParam.min_grad 1e-10 (mínimo de error del gradiente)

net.trainParam.show 25 (despliegues con que se muestra el comportamiento de las iteraciones)

net.trainParam.time inf (tiempo máximo de entrenamiento en segundos)

El entrenamiento termina según las mismas condiciones de la función *traingd*.

TRAINGDX **(**Gradient descent with momentum and adaptive learning rate backpropagation)

Descripción: *traingdx* es una función de entrenamiento que actualiza los valores de los pesos y los bios acorde al gradiente descendente con momento, y al rango de aprendizaje adaptable. Utiliza el algoritmo backpropagation.

La función *traingdx* combina para el entrenamiento el rango de aprendizaje adaptable con momento. El entrenamiento se realiza del mismo modo que con *traingda*, excepto que tiene el coeficiente de momento *mc* como un parámetro adicional de entrenamiento.

El entrenamiento ocurre acorde a los parámetros de la función triangdx, que se muestran a continuación con los valores que toman por defecto:

net.trainParam.epochs 10 (máximo número de iteraciones)

net.trainParam.goal 0 (meta del error)

net.trainParam.lr 0.01 (rango de aprendizaje)

net.trainParam.lr_inc 1.05 (tamaño para aumentar el rango de aprendizaje)

net.trainParam.lr_dec 0.7 (tamaño para disminuir el rango de aprendizaje)

net.trainParam.max_fail 5 (máximo de fracasos de validación)

net.trainParam.max_perf_inc 1.04 (máximo de incremento del error)

net.trainParam.mc 0.9 (valor de la variable momentum)

net.trainParam.min_grad 1e-10 (mínimo de error del gradiente)

net.trainParam.show 25 (despliegues con que se muestra el comportamiento de las iteraciones)

net.trainParam.time inf (tiempo máximo de entrenamiento en segundos)

El entrenamiento termina según las mismas condiciones de la función *traingd*.

Tipo de aprendizaje

Teniendo en cuenta las características del sistema que se modela y los modelos de RNR seleccionados, el tipo de aprendizaje que se desarrolla es supervisado.

Demuth y otros [71], [72], recomiendan dos funciones en Matlab para el aprendizaje de redes recurrentes, que basan su algoritmo en la técnica del gradiente descendente.

LEARNGD (Gradient descent weight and bias learning function)

Descripción: *learngd* es la función de aprendizaje gradiente descendente para pesos y bios.

El aprendizaje ocurre acorde al parámetro de aprendizaje de la función learngd, mostrado a continuación con el valor que toma por defecto:

LP.lr - 0.01 (rango de aprendizaje)

learngd calcula los cambios en los pesos para una neurona desde la entrada a la neurona P y el error E. El rango de aprendizaje de los pesos o bios se calcula acorde al gradiente descendente.

LEARNGDM (Gradient descent with momentum weight and bias learning function)

Descripción: *learngd* es la función de aprendizaje gradiente descendente con momento para pesos y bios.

El aprendizaje ocurre acorde a los parámetros de aprendizaje de la función learngdm, mostrados a continuación con el valor que toman por defecto:

LP.lr - 0.01 (rango de aprendizaje)

LP.mc - 0.9 (constante de momentum)

learngdm calcula los cambios en los pesos para una neurona desde la entrada a la neurona P y el error E. El rango de aprendizaje LR y la constante de momento MC de los pesos o bios, se calcula acorde al gradiente descendente con momento.

2.4.2 Modelo ELMAN: Teoría y Experimentación.

Las redes Elman son redes feedforward con recurrencia y con retrasos de tiempo en las entradas, que pueden aprender arbitrariamente bien cualquier relación dinámica de entrada-salida, dando suficientes neuronas en las capas ocultas [72].

En una red ELMAN la capa de entrada se puede considerar dividida en 2 partes: unidades de entrada y unidades contextuales. Las unidades contextuales toman una copia de las salidas de los nodos de la capa oculta, lo que proporciona, en cierto modo, una memoria al sistema [70]. Las conexiones de realimentación van desde la capa oculta a la capa contextual [65].

La red original ELMAN tuvo solo dos capas (una capa oculta y una capa de salida), de hecho el incluir en una red de este tipo más de una capa oculta para la resolución de un problema solo debe ser considerado si la complejidad del mismo es alta [63].

La Ilustración 20 muestra la arquitectura de un modelo ELMAN implementado en Matlab en esta investigación.

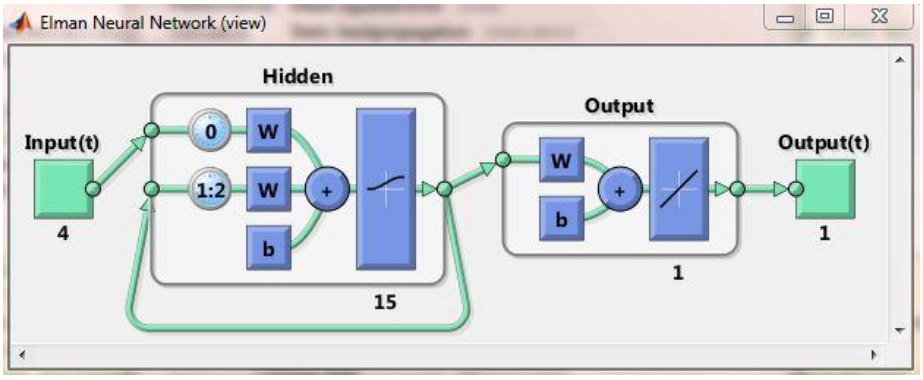


Ilustración 20: Arquitectura de un modelo ELMAN implementado en Matlab.

Matlab permite crear una red ELMAN mediante la función elmannet.

ELMANNET

Sintaxis

net = elmannet(layerDelays,hiddenSizes,trainFcn)

elmannet(delays,hiddenSizes,trainFcn) toma los siguientes argumentos,

layerdelays vector fila con valores positivos mayores que 0, que representan los delays en las capas de la red (predefinido = 1:2)

hiddenSizes vector fila con valores positivos mayores que 0, que representan la cantidad de neuronas en las capas ocultas (predefinido = 10)

trainFcn función de entrenamiento (predefinido = 'trainlm')

y devuelve una red neuronal de tipo Elman (net).

La función de entrenamiento puede ser cualquiera de las cuatro funciones mencionadas en el Epígrafe 2.4.1 (TRAINGD, TRAINGDM, TRAINGDA, TRAINGDX).

La función de aprendizaje puede ser cualquiera de las dos funciones mencionadas en el Epígrafe 2.4.1 (LEARNGD o LEARNGDM).

El error es medido acorde a la función MSE, utilizada y recomendada para este tipo de redes, y definido con un mínimo de 0,001.

La red tiene 1 capa oculta. La cantidad de neuronas en la capa oculta es de 15. La función de transferencia para la capa oculta, pueden ser cualquiera de las dos funciones sigmoideas mencionadas en el Epígrafe 2.4.1 (TANSIG o LOGSIG).

La función de transferencia para la capa de salida, es la función lineal mencionada en el Epígrafe 2.4.1 (PURELIN).

La adaptación de la red o entrenamiento se realiza mediante la función TRAIN y la simulación se realiza mediante la función SIM [71].

Para este modelo se realizan 7 variantes, con el objetivo de que la red devuelva su mejor resultado. A continuación se muestran los parámetros principales que sufren variaciones en cada una de las variantes:

- **Variante 1:** 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 2:** 500 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 3:** 2000 iteraciones, función de transferencia para la capa oculta LOGSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 4:** 500 iteraciones, función de transferencia para la capa oculta LOGSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 5:** 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDA, función de aprendizaje LEARNGD.
- **Variante 6:** 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDM, función de aprendizaje LEARNGDM.
- **Variante 7:** 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDX, función de aprendizaje LEARNGDM.

2.4.3 Modelo LRN: Teoría y Experimentación.

Demuth, Beale y Hagan [71], [72] proponen, para Matlab, una versión ampliada de la red de Elman: Layer-Recurrent Network (LRN). En esta red, hay un lazo de información retroactiva con retrasos de tiempo, alrededor de cada capa de la red, excepto por la última capa.

La LRN generaliza la red Elman para tener un número arbitrario de capas y tener funciones de transferencias arbitrarias en cada capa [71], [72].

La Ilustración 21 muestra la arquitectura de un modelo LRN implementado en Matlab en esta investigación.

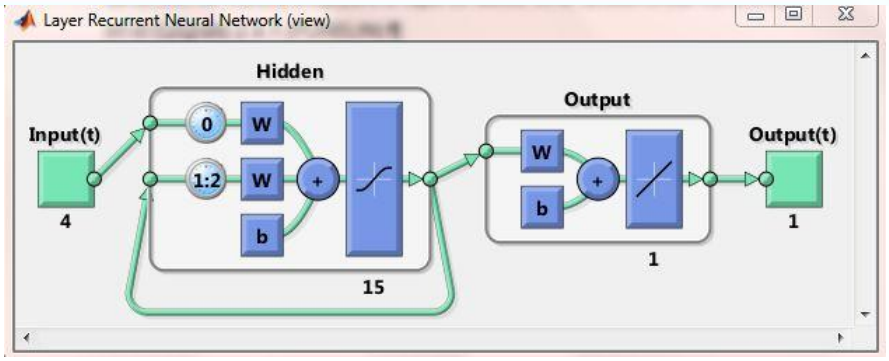


Ilustración 21: Arquitectura de un modelo LRN implementado en Matlab.

Matlab permite crear un modelo LRN mediante la función layrecnet.

LAYRECNET

Layer recurrent neural network

Sintaxis

net = layrecnet(layerDelays,hiddenSizes,trainFcn)

layrecnet(layerDelays,hiddenSizes,trainFcn) toma los siguientes argumentos,

layerdelays vector fila con valores positivos mayores que 0, que representan los delays en las capas de la red (predefinido = 1:2)

hiddenSizes vector fila con valores positivos mayores que 0, que representan la cantidad de neuronas en las capas ocultas (predefinido = 10)

trainFcn función de entrenamiento (predefinido = 'trainlm')

Y devuelve una red de tipo LRN (net).

La función de entrenamiento puede ser cualquiera de las cuatro funciones mencionadas en el Epígrafe 2.4.1 (TRAINGD, TRAINGDM, TRAINGDA, TRAINGDX).

La función de aprendizaje puede ser cualquiera de las dos funciones mencionadas en el Epígrafe 2.4.1 (LEARNGD o LEARNGDM).

El error es medido acorde a la función MSE, utilizada y recomendada para este tipo de redes, y definido con un mínimo de 0,001.

La red varía la cantidad de capas ocultas. Aquellos modelos que presentan solo una capa oculta, dicha capa tiene 15 neuronas, y aquellos modelos que tienen 2 capas ocultas, dichas capas tienen 10 y 4 neuronas respectivamente. La función de transferencia para las capas ocultas, pueden ser cualquiera de las dos funciones sigmoideas mencionadas en el Epígrafe 2.4.1 (TANSIG o LOGSIG).

La función de transferencia para la capa de salida, es la función lineal mencionada en el Epígrafe 2.4.1 (PURELIN).

La adaptación de la red o entrenamiento se realiza mediante la función TRAIN y la simulación se realiza mediante la función SIM [71].

Para este modelo se realizan 8 variantes, con el objetivo de que la red devuelva su mejor resultado. A continuación se muestran los parámetros principales que sufren variaciones en cada una de las variantes:

- **Variante 1:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 2:** 1 capa oculta con 15 neuronas, 500 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 3:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta LOGSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.

- **Variante 4:** 1 capa oculta con 15 neuronas, 500 iteraciones, función de transferencia para la capa oculta LOGSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 5:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDA, función de aprendizaje LEARNGD.
- **Variante 6:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDM, función de aprendizaje LEARNGDM.
- **Variante 7:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDX, función de aprendizaje LEARNGDM.
- **Variante 8:** 2 capas ocultas con 10 y 4 neuronas respectivamente, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDX, función de aprendizaje LEARNGDM.

2.4.4 Modelo NARX: Teoría y Experimentación.

Demuth, Beale y Hagan [71], [72] proponen la red auto regresiva no lineal con aportes exógenos (nonlinear autoregressive network with exogenous inputs - NARX) y algunas variantes de esta.

La red NARX es una red dinámica parcialmente recurrente, con conexiones de información retroactiva incluyendo varias capas de la red [71], [72].

Las Ilustraciones 22 y 23 muestran la arquitectura de los modelos NARX Closed Loop y NARX Open Loop respectivamente, implementado en Matlab en esta investigación.

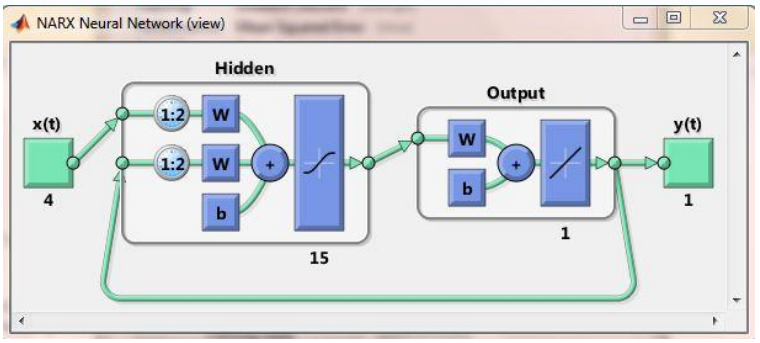


Ilustración 22: Arquitectura de un modelo NARX Closed Loop implementado en Matlab.

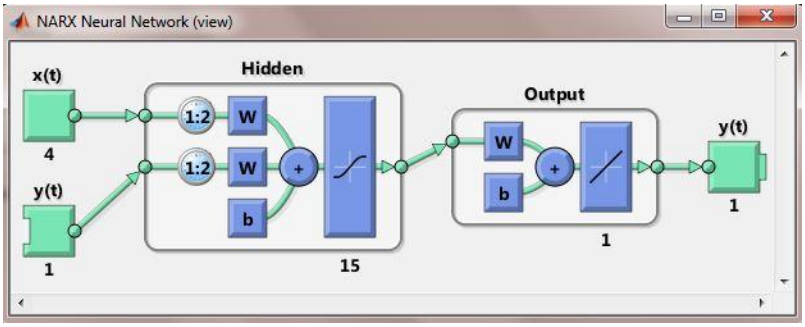


Ilustración 23: Arquitectura de un modelo NARX Open Loop implementado en Matlab.

Matlab permite crear una red NARX mediante la función narxnet

NARXNET

Sintaxis

net =narxnet(inputDelays,feedbackDelays,hiddenSizes,feedbackMode,trainFcn)

narxnet(inputDelays,feedbackDelays,hiddenSizes,feedbackMode,trainFcn) toma los siguientes argumentos,

inputDelays vector fila con valores positivos mayores que 0, que representan los delays a las entradas de la red (predefinido = 1:2)

feedbackDelays vector fila con valores positivos mayores que 0, que representan los delays en las recurrencias de la red (predefinido = 1:2)

hiddenSizes vector fila con valores positivos mayores que 0, que representan la cantidad de neuronas en las capas ocultas (predefinido = 10)

feedbackMode defiene el modo de recurrencia (predefinido lazo abierto 'open', o lazo cerrado 'closed')

trainFcn función de entrenamiento (predefinido = 'trainlm')

Y devuelve una red de tipo NARX (net).

La función de entrenamiento puede ser cualquiera de las cuatro funciones mencionadas en el Epígrafe 2.4.1 (TRAINGD, TRAINGDM, TRAINGDA, TRAINGDX).

La función de aprendizaje puede ser cualquiera de las dos funciones mencionadas en el Epígrafe 2.4.1 (LEARNGD o LEARNGDM).

El error es medido acorde a la función MSE, utilizada y recomendada para este tipo de redes, y definido con un mínimo de 0,001.

La red varía la cantidad de capas ocultas. Aquellos modelos que presentan solo una capa oculta, dicha capa tiene 15 neuronas, y aquellos modelos que tienen 2 capas ocultas, dichas capas tienen 10 y 4 neuronas respectivamente. La función de transferencia para las capas ocultas, pueden ser cualquiera de las dos funciones sigmoideas mencionadas en el Epígrafe 2.4.1 (TANSIG o LOGSIG).

La función de transferencia para la capa de salida, es la función lineal mencionada en el Epígrafe 2.4.1 (PURELIN).

La adaptación de la red o entrenamiento se realiza mediante la función TRAIN y la simulación se realiza mediante la función SIM [71].

Para este modelo se realizan 7 variantes de lazo abierto (open loop) y 7 variantes de lazo cerrado (closed loop), con el objetivo de que la red devuelva su mejor resultado. A continuación se muestran los parámetros principales que sufren variaciones en cada una de las variantes:

NARX Closed Loop

- **Variante 1:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 2:** 1 capa oculta con 15 neuronas, 500 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 3:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta LOGSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 4:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDA, función de aprendizaje LEARNGD.
- **Variante 5:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDM, función de aprendizaje LEARNGDM.
- **Variante 6:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDX, función de aprendizaje LEARNGDM.
- **Variante 7:** 2 capas ocultas con 10 y 4 neuronas respectivamente, 2000 iteraciones, función de transferencia para las capas ocultas TANSIG, función de entrenamiento TRAINGDX, función de aprendizaje LEARNGDM.

NARX Open Loop

- **Variante 1:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 2:** 1 capa oculta con 15 neuronas, 500 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.

- **Variante 3:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta LOGSIG, función de entrenamiento TRAINGD, función de aprendizaje LEARNGD.
- **Variante 4:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDA, función de aprendizaje LEARNGD.
- **Variante 5:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDM, función de aprendizaje LEARNGDM.
- **Variante 6:** 1 capa oculta con 15 neuronas, 2000 iteraciones, función de transferencia para la capa oculta TANSIG, función de entrenamiento TRAINGDX, función de aprendizaje LEARNGDM.
- **Variante 7:** 2 capas ocultas con 10 y 4 neuronas respectivamente, 2000 iteraciones, función de transferencia para las capas ocultas TANSIG, función de entrenamiento TRAINGDX, función de aprendizaje LEARNGDM.

2.5 Conclusiones.

- Se identificaron las RNR como los modelos apropiados para modelar el sistema de climatización del hotel Jagua, siendo seleccionados los modelos LRN, ELMAN, y NARX, que presentan una arquitectura parcialmente recurrente.
- Los experimentos realizados en Matlab, se organizaron en 2 etapas: la primera para la creación y entrenamiento de los modelos, y la segunda para la prueba, donde se simula su comportamiento ante nuevos valores.
- Se experimentaron con 7 variantes del ELMAN, 8 variantes del LRN, 7 variantes del NARX Closed Loop, y 7 variantes del NARX Open Loop, modificando parámetros de la arquitectura y del entrenamiento.

Capítulo III: Análisis de los resultados.

El presente capítulo aborda los distintos criterios o pruebas para medir el desempeño de los modelos. Primeramente se detallan los criterios MSE y FIT que incluye Matlab, así como los resultados obtenidos de las variantes de cada modelo. Luego se selecciona el mejor resultado, de cada variante, de acuerdo a estos criterios. En otro momento se detallan las pruebas no paramétricas (Friedman, Friedman Aligned Rank, Quade y Contrast Estimation) aplicadas a los modelos, para corroborar los resultados obtenidos, y se selecciona el mejor modelo según estas pruebas. Finalmente se define, entre todos los modelos, el mejor para la predicción de la carga térmica del hotel Jagua.

3.1 Criterios de medida del desempeño de los modelos: MSE y FIT.

Luego de realizar las Etapas 1 y 2 de la experimentación, para cada uno de los modelos seleccionados, es necesario obtener una medida del desempeño de dichos modelos. Para ello se utilizan dos criterios, incluidos en Matlab: un criterio de minimización MSE, y un criterio de maximización FIT, detallados a continuación.

MSE (Mean squared error performance function) es una función de actuación de la red o función de error. Intenta medir y minimizar el error en todo el espacio de valores de los parámetros de la red.

La función error es una función matemática definida en el espacio de pesos multidimensionales para un conjunto de patrones dados. Es una superficie que tendrá muchos mínimos (globales y locales), y la regla de aprendizaje LMS va en busca del punto en el espacio de pesos donde se encuentra el mínimo global de esta superficie. De manera general y en su forma más simple, la función trata de minimizar la suma de los cuadrados de la diferencia entre la salida deseada y la salida obtenida, para todo el conjunto de parámetros de la red [71], [72], [85].

FIT devuelve el porcentaje de la salida, medida que se obtuvo del modelo y que representa un criterio de maximización.

El fit se calcula como:

FIT = 100 * (1-norm(Y-YHAT)/norm(Y-mean(Y))) (in %)

Donde Y es la salida de validación o salida deseada y YHAT es la salida obtenida por el modelo.

La norma de un vector o matrix V viene dada por la siguiente ecuación:

SUM(ABS(V).^2)^(1/2)

Que no es más que, la raíz cuadrada de la suma de los cuadrados de los elementos de V.

Por ejemplo:

V = 1 2 3

5 8 9

3 5 7

norm(V) = sqrt(1 + 2 + 3 + 5+ 8 + 9 +3 + 5 +7)

La media de un vector o matrix V es el promedio de los valores de V.

Por ejemplo:

mean(V) = 3.0000 5.0000 6.3333

Ambos criterios se utilizan con el objetivo de analizar los resultados y poder definir el mejor modelo en la predicción de la carga térmica del hotel Jagua.

3.2 Selección de la mejor variante de cada modelo según los criterios FIT y MSE.

Luego de realizar las Etapas 1 y 2 de la experimentación en cada uno de las variantes de los modelos seleccionados, y luego de aplicar los criterios de medida de desempeño en cada variante, se obtiene los resultados finales de cada una de estas.

Resultados de las variantes de cada modelo.

- **Modelo ELMAN**

La tabla 1 muestra los resultados obtenidos de cada una de las variantes del modelo ELMAN. Nótese que la Variante7 señalada en rojo, muestra el mejor resultado para dicho modelo.

Tabla 1: Resultados de las variantes del modelo ELMAN.

Modelo ELMAN	FIT para la Etapa1: Entrenamiento	FIT para la Etapa2: Prueba	MSE
Variante 1	79.6311	71.6962	0.0054
Variante2	59.6687	51.4015	0.0187
Variante3	58.4851	51.7492	0.0231
Variante4	38.4378	31.6140	0.0436
Variante5	79.8566	76.2203	0.0038
Variante6	75.9602	69.3515	0.0055
Variante7	**87.2843**	**82.7351**	**0.0014**

- **Modelo LRN**

La tabla 2 muestra los resultados obtenidos de cada una de las variantes del modelo LRN. Nótese que la Variante8 señalada en rojo, muestra el mejor resultado para dicho modelo.

Tabla 2: Resultados de las variantes del modelo LRN.

Modelo LRN	FIT para la Etapa1: Entrenamiento	FIT para la Etapa2: Prueba	MSE
Variante 1	77.5279	71.4981	0.0063
Variante2	65.0190	63.6891	0.0149
Variante3	65.2013	55.6343	0.0194
Variante4	42.1983	36.9349	0.0412
Variante5	84.9448	77.7920	0.0028
Variante6	77.2263	74.7281	0.0062
Variante7	87.4628	83.6181	0.0022
Variante8	**89.2915**	**85.6759**	**0.0013**

- **Modelo NARX Closed Loop**

La tabla 3 muestra los resultados obtenidos de cada una de las variantes del modelo NARX Closed Loop. Nótese que la Variante7 señalada en rojo, muestra el mejor resultado para dicho modelo.

Tabla 3: Resultados de las variantes del modelo NARX Closed Loop.

Modelo NARX	FIT para la Etapa1: Entrenamiento	FIT para la Etapa2: Prueba	MSE
Variante 1	50.2267	41.7449	0.0306
Variante2	34.4318	27.7966	0.0514
Variante3	40.9169	39.1309	0.0427
Variante4	62.5702	54.3768	0.0241
Variante5	45.0598	39.3856	0.0364
Variante6	65.2460	54.1515	0.0159
Variante7	**65.1043**	**56.0688**	**0.0149**

- **Modelo NARX Open Loop**

La tabla 4 muestra los resultados obtenidos de cada una de las variantes del modelo NARX Open Loop. Nótese que la Variante7 señalada en rojo, muestra el mejor resultado para dicho modelo.

Tabla 4: Resultados de las variantes del modelo NARX Open Loop.

Modelo NARX	FIT para la Etapa1: Entrenamiento	FIT para la Etapa2: Prueba	MSE
Variante 1	61.5197	56.6060	0.0196
Variante2	48.6401	44.9973	0.0350
Variante3	54.3957	51.5605	0.0274
Variante4	75.7055	74.5106	0.0064
Variante5	66.4989	62.3226	0.0191
Variante6	75.9784	74.9410	0.0057
Variante7	**76.5140**	**75.2309**	**0.0059**

3.3 Modelo de RNA propuesto para predecir la carga térmica del hotel Jagua.

3.3.1 Mejor modelo según los criterios MSE y FIT.

Luego de aplicar los criterios, MSE y FIT de maximización y minimización respectivamente, a cada una de las variantes de los modelos, se le obtiene la medida del desempeño de cada uno de estos. Se propone entonces, el mejor para la predicción de la carga térmica del hotel Jagua según dichos criterios.

La tabla 5 muestra los resultados de la mejor variante de cada modelo.

Tabla 5: Resultados de las mejores variantes de cada modelo.

Modelo	Variante	FIT para la Etapa1: Entrenamiento	FIT para la Etapa2: Prueba	MSE
ELMAN	Variante7	87.2843	82.7351	0.0014
LRN	**Variante8**	**89.2915**	**85.6759**	**0.0013**
NARX Closed Loop	Variante7	65.1043	56.0688	0.0149
NARX Open Loop	Variante7	76.5140	75.2309	0.0059

Las respuestas obtenidas de Matlab de la variante 7 de ELMAN, variante 8 de LRN, variante 7 de NARX Closed Loop y variante 7 de NARX Open Loop, se muestran en los Anexos 11, 12, 13 y 14 respectivamente.

El mejor resultado viene dado por la Variante8 del modelo LRN, señalado en rojo en la tabla 5. Dicho resultado se selecciona como el mejor modelo según los criterios MSE y FIT.

3.3.2 Mejor modelo según las pruebas no paramétricas básicas: Friedman y Contrast Estimation, y según las pruebas no paramétricas avanzadas Friedman Aligned Rank y Quade.

Los autores García, Fernádez, Luengo y Herrera en su investigación "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power" [100], proponen una guía completa para el uso de procedimientos estadísticos no paramétricos, para comparaciones del desempeño de algoritmos o modelos en diseños de experimentos, en el área de la computación inteligente.

Dicha investigación incluye un software escrito en lenguaje JAVA, por tanto para su ejecución es necesario instalar una máquina virtual de JAVA. El software lee los resultados de todos los modelos para sus correspondientes conjuntos de datos, desde

un archivo en formato CSV almacenado previamente. Luego se ejecuta, desde la ventana comando de windows, la siguiente linea:

- java Friedman data.csv > output.tex

data.csv es el archivo donde se encuentran almacenados los resultados.

output.tex es el archivo donde se guarda la salida que genera el software.

La salida es dada en un formato LATEX, a partir de este se genera un archivo pdf con los resultados finales.

A partir de las propuestas de la investigación [100], se seleccionan 4 pruebas no paramétricas que se aplican a los modelos desarrollados durante el experimento: Prueba Friedman, Prueba Friedman Aligned Ranks, Prueba Quade y Prueba Contrast Estimation (CE), detallas en los sub-epígrafes siguientes.

3.3.2.1 Prueba Friedman

La prueba Friedman [101], [102] tiene como objetivo determinar si podemos llegar a una conclusión de que en una muestra de resultados existe diferencia entre los efectos de tratamiento. La prueba estadística convierte los resultados originales a rangos. Así, ordena por rango cada modelo, el mejor debería tener el rango de 1, el segundo rango 2, y así sucesivamente.

La Tabla 6 muestra los rangos obtenidos para cada modelo del experimento al aplicar la prueba de Friedman.

Tabla 6: Rangos de los modelos según la prueba Firedman.

Modelo	Rango
ELMAN	2.0
LRN	1.0
NARX Open Loop	3.0
NARX Closed Loop	4.0

Tal como se muestra en la tabla 6, el modelo que mejor resultado ofrece es el modelo LRN con rango 1.0, seguido por el modelo ELMAN con rango 2.0, luego el modelo NARX Open Loop con rango 3.0 y el modelo que ofrece peor resultado es el modelo NARX Closed Loop con un rango 4.0.

3.3.2.2 Prueba Friedman Aligned Ranks

Cuando el número de modelos para la comparación es pequeño, esto puede plantear una desventaja, pues las comparaciones entre ellos no son significativas. En tales casos, Hodges y Lehmann [103] recomiendan el método de rangos alineados. En esta técnica, se busca un valor de posición del desempeño promedio logrado por todos los modelos en cada conjunto de datos. Entonces, calcula la diferencia entre el desempeño obtenido por un modelo y su valor de posición. Este paso es repetido para cada modelo y sus correspondientes conjuntos de datos. Las diferencias resultantes, se denominan observaciones alineadas, las cuales mantienen sus identidades con relación al conjunto de datos y la combinación de los modelos a los que pertenecen. Finalmente son ordenadas por rango de 1 hasta n referente a cada otro. Los rangos asignados a cada observación alineada son llamados rangos alineados.

La Tabla 7 muestra los rangos obtenidos para cada modelo del experimento al aplicar la prueba Friedman Aligned Ranks.

Tabla 7: Rangos de los modelos según la prueba Friedman Aligned Ranks.

Modelo	Rango
ELMAN	3.5
LRN	1.5
NARX Open Loop	5.5
NARX Closed Loop	7.5

Tal como se muestra en la tabla 7, el modelo que mejor resultado ofrece es el modelo LRN con rango 1.5, seguido por el modelo ELMAN con rango 3.5, luego el modelo

NARX Open Loop con rango 5.5 y el modelo que ofrece peor resultado es el modelo NARX Closed Loop con un rango 7.5.

3.3.2.3 Prueba Quade

En algunos conjuntos de datos las diferencias registradas por los resultados de cada modelo, pueden ser significativamente grandes. Los rangos obtenidos para cada modelo en cada conjunto de datos, podrían ser modificados dependiendo de las diferencias observadas en el desempeño de dichos modelos. La prueba Quade, propuesta por Quade [104], ofrece un análisis, de ordenación por rango, de los resultados.

El método comienza a encontrar rangos del mismo modo que la prueba Friedman. El siguiente paso requiere los valores originales del desempeño de los modelos. Los rangos son asignados a los modelos según el alcance del tamaño del resultado en cada conjunto de datos. El alcance de los resultados se trata como la diferencia entre las observaciones más grandes y más pequeñas dentro del conjunto de datos correspondiente. Entonces, se le asigna rango 1 al modelo que menor alcance tiene, rango 2 al segundo modelo más pequeño, y así sucesivamente para cada modelo con su correspondiente conjunto de datos.

La Tabla 8 muestra los rangos obtenidos para cada modelo del experimento al aplicar la prueba Quade.

Tabla 8: Rangos de los modelos según la prueba Quade.

Modelo	Rango
ELMAN	2.0
LRN	1.0
NARX Open Loop	3.0
NARX Closed Loop	4.0

Tal como se muestra en la tabla 8, el modelo que mejor resultado ofrece es el modelo LRN con rango 1.0, seguido por el modelo ELMAN con rango 2.0, luego el modelo NARX Open Loop con rango 3.0 y el modelo que ofrece peor resultado es el modelo NARX Closed Loop con un rango 4.0.

3.3.2.4 Prueba CE (Contrast Estimation)

Usando los resultados obtenidos de cada modelo luego del experimento, se puede obtener la diferencia entre el desempeño de estos. Para este propósito Doksum [105] propone la prueba CE, que ofrece una diferencia cuantitativa entre pares de algoritmos sobre múltiples conjuntos de datos.

La Tabla 9 muestra las diferencias entre el desempeño de cada modelo.

Tabla 9: Diferencias entre el desempeño de los modelos según la prueba CE.

Modelos	ELMAN	LRN	NARX Open Loop	NARX Closed Loop
ELMAN	0,000	-2,474	9,550	24,42
LRN	2,474	0,000	12,02	26,90
NARX Open Loop	-9,550	-12,02	0,000	14,87
NARX Closed Loop	-24,42	-26,90	-14,87	0,000

Mientras más valores positivos tenga un modelo con respecto a los demás mejor es su desempeño. El modelo que mejor resultados ofrece es el modelo LRN, el cual tiene diferencias significativas positivas, respecto a los restantes modelos. Seguidamente se ubica el modelo ELMAN, luego el modelo NARX Open Loop y finalmente con el peor desempeño se ubica el modelo NARX Closed Loop.

Luego de aplicar las pruebas no paramétricas básicas (Friedman y Contrast Estimation), y las pruebas no paramétricas avanzadas (Friedman Aligned Rank y Quade), a las mejores variantes de cada modelo (Variante 7 del modelo ELMAN, Variante 8 del modelo LRN, Variante 7 del modelo NARX Open Loop y Variante 7 del modelo NARX

Closed Loop), se obtiene que: el modelo que mejor resultado ofrece viene dado por la Variante8 del modelo LRN, según dichas pruebas.

3.3.3 Mejor modelo resultante para la predicción de la carga térmica del hotel Jagua.

El resultado que se selecciona como el mejor modelo, para la predicción de la carga térmica del hotel Jagua, según los criterios MSE y FIT, según las pruebas no paramétricas básicas Friedman y Contrast Estimation, y según las pruebas no paramétricas avanzadas Friedman Aligned Rank y Quade, presenta las siguientes características:

Modelo con arquitectura parcialmente recurrente LRN, con 4 neuronas de entrada que representan las variables presentes en el proceso. Las variables son:

- la temperatura ambiente
- la temperatura de *set-point*
- la radiación solar global que incide directamente sobre el exterior de las paredes de la construcción
- la radiación solar difusa que llega al interior de la habitación

La capa de salida tiene 1 neurona que representa la carga necesaria para lograr el confort (carga térmica). Esta capa utiliza la función de transferencia PURELINE.

Presenta 2 capas ocultas con 10 y 4 neuronas respectivamente, y utilizan la función de transferencia TANSIG.

La función de entrenamiento que implementa es TRAINGDX, y utiliza la función de aprendizaje LEARNGD.

Durante el entrenamiento se realizan 2000 iteraciones (epochs), y busca minimizar el error hasta 0,001.

Los valores de los pesos y los bios son los que genera Matlab y con que la red muestra un mejor desempeño.

La Figura 24 muestra la arquitectura de este modelo.

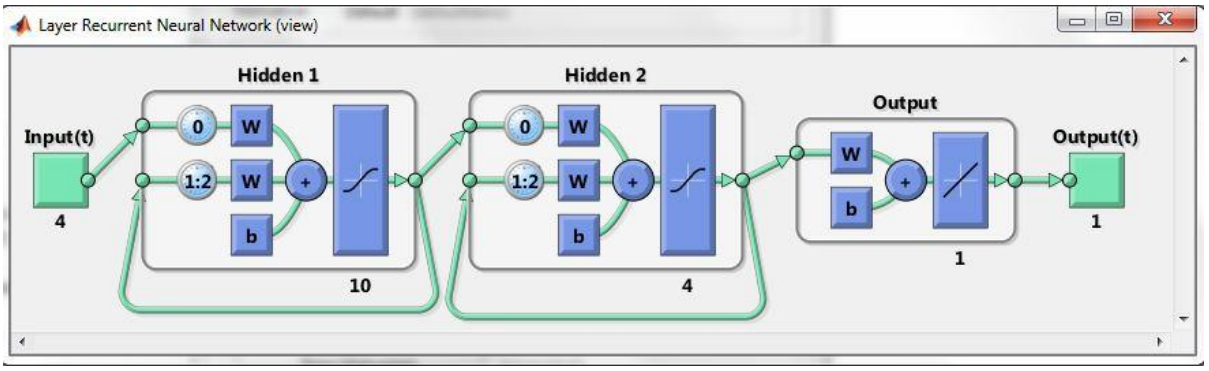


Ilustración 24: Arquitectura del modelo LRN propuesto para predecir la carga térmica necesaria para el sistema de climatización del hotel Jagua.

El código resultante de la implementación, en Matlab, del modelo seleccionado aparece reflejado en el Anexo 15.

3.4 Conclusiones

- Los criterios utilizados para medir el desempeño de los modelos fueron: las funciones (incluidas en Matlab) MSE y FIT; además las pruebas no paramétricas: Friedman, Friedman Aligned Rank, Quade y Contrast Estimation.
- El modelo seleccionado para la predicción de la carga térmica del hotel Jagua, es el LRN, con 4 neuronas de entrada y 1 en la capa de salida. Además tiene 2 capas ocultas con 10 y 4 neuronas respectivamente. Las funciones de transferencia que utiliza son TANSIG y PURELINE. La función de entrenamiento utilizada es TRAINGDX, y la de aprendizaje es LEARNGD. Durante el entrenamiento se realizan 2000 iteraciones, y se establece un error mínimo de 0,001.

Conclusiones

- El uso e instalación de los sistemas de climatización centralizado, todo agua, se ha incrementado a nivel internacional y nacional, resultando apropiado para su operación y control del consumo energético, la aplicación de Redes Neuronales Recurrentes.
- Se seleccionan como los modelos efectivos para modelar el sistema de climatización del hotel Jagua, los que presentan una arquitectura parcialmente recurrente, como: Modelo ELMAN, Modelo LRN, Modelo NARX (Open Loop y Closed Loop).
- Se experimentaron en Matlab, con 7 variantes del ELMAN, 8 variantes del LRN, 7 variantes del NARX Open Loop, y del NARX Closed Loop respectivamente, modificando parámetros en su arquitectura y entrenamiento.
- El modelo seleccionado para la predicción de la carga térmica del hotel Jagua, es el LRN, con: 4 neuronas de entrada y 1 en la capa de salida; 2 capas ocultas con 10 y 4 neuronas respectivamente; funciones de transferencia TANSIG y PURELINE.; función de entrenamiento TRAINGDX, y de aprendizaje LEARNGD; entrenamiento con 2000 iteraciones y un error mínimo de 0,001.

Recomendaciones

1. Ampliar la experimentación con Conjuntos de Datos para distintos meses del año con el objetivo de aumentar y mejorar la utilidad del modelo seleccionado.

2. Implementar un sistema hibrido donde se utilicen los algoritmos genéticos para optimizar la inicialización de los pesos y los bios de la red.

3. Analizar la ocupación del hotel como parte de las entradas del sistema, teniendo en cuenta su variación.

Referencias Bibliográficas

[1] R. Ruelas, «La importancia de ahorrar energía». 2008.
[2] A. González, «Medidas de ahorro». .
[3] M. Castilla y J. Álvarez, «Técnicas de Control de Confort en edificios.» 2010.
[4] D. Gyalistras, «Use of MPC for Building Control». 2010.
[5] J. Bermúdez, «Estrategias de control multivariable para un sistema de compresión de vapor». 2008.
[6] Anónimo, «Sistema de aire acondicionado limpio en Toronto». 2002.
[7] J. Landa y Y. Delgado, «"SOBRE EL ESTADO DEL ARTE EN SISTEMAS CENTRALIZADOS DE AIRE ACONDICIONADO CON AGUA FRÍA".Centro de Estudios de Combustión y Energía, Facultad de Ingenierías Química y Mecánica. Universidad de Matanzas "Camilo Cienfuegos"». 2007.
[8] Q. Li, Q. Meng, J. Cai, H. Yoshino, y A. Mochida, «"Predicting hourly cooling load in the building: A comparison of support vector machine and different artificial neural networks". Energy Conversion and Management», vol. 50, nº. 1, pp. 90–96, ene. 2009.
[9] J. . Van Ryzin y T. Leraand, «"Air Conditioning with Deep Seawater: A cost- Effective Alternative, Ocean Resources 2000". Technology Magazine», p. 37, sep. 1992.
[10] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, y P. Haves, «"Model Predictive Control for the Operation of Building Cooling Systems"». 2010.
[11] M. . Leung, C. . Tse, L. . Lai, y T. . Chow, «"The use of occupancy space electrical power demand in building cooling load prediction". Energy and Buildings», vol. 55, pp. 151–163, dic. 2012.
[12] J. . Moreno y J. . Sánchez, «Evaluación y análisis del sistema centralizado de aire acondicionado en el hotel Sandals Royal Hicacos.» .
[13] S. Prívara y J. Siroký, «"Model predictive control of a building heating system The first experience". Energy and Buildings», nº. 43, pp. 564–572, 2011.
[14] S. Montelier, M. de Armas, A. Borroto, J. Gómez, y C. Pérez, «"Inteligencia Artificial Aplicada a la Reducción del Consumo Energético de un Sistema de Climatización por agua Helada en un Hotel Turístico". Energética», nº. 39, pp. 13–18, jul. 2008.
[15] Z. Ma y S. Wang, «"An optimal control strategy for complex building central chilled water systems for practical and real-time applications". Building and Environment», vol. 44, pp. 1188–1198, 2009.
[16] L. Hernádez, «"Predicción del consumo energético del Hotel Jagua aplicando la simulación termodinámica y la inteligencia artificial".» 2006.
[17] J. Siroký y F. Oldewurtel, «"Experimental analysis of model predictive control for an energy efficient building heating system". Applied Energy», vol. 88, 2011.
[18] A. . Ben-Nakhi y M. . Mahmoud, «"Cooling load prediction for buildings using general regression neural networks".Energy Conversion and Management», vol. 45, pp. 2127–2141, ago. 2004.
[19] Z. Hou, Z. Lian, Y. Yao, y X. Yuan, «"Cooling-load prediction by the combination of rough set theory and an artificial neural-network based on data-fusion technique". Applied Energy», vol. 83, nº. 9, pp. 1033–1046, sep. 2006.
[20] S. Kwok y E. Lee, «"A study of the importance of occupancy to building cooling load in prediction by intelligent approach". Energy Conversion and Management», vol. 52, nº. 7, pp. 2555–2564, jul. 2011.
[21] A. Abbassi y L. Bahar, «"Application of neural network for the modeling and control of

evaporative condenser cooling load". Applied Thermal Engineering», vol. 25, nº. 17–18, pp. 3176–3186, dic. 2005.
[22] S. Pandey y D. Hindoliya, «"Artificial neural network for predation of cooling load reduction using green roof over building in Sustainable City". Sustainable Cities and Society», vol. 3, pp. 37–45, jul. 2012.
[23] G. Ge, F. Xiao, y S. Wang, «"Neural network based prediction method for preventing condensation in chilled ceiling systems". Energy and Buildings», vol. 45, pp. 290–298, feb. 2012.
[24] J. Armas, M. Lapido, J. Gómez, y Y. Valdivia, «"Evaluación termodinámica de sistemas de climatización centralizados por agua helada usando herramientas de inteligencia artificial". Revista Ingeniería e Investigación», vol. 31, nº. 2, pp. 134–142, 2011.
[25] Y. Valdivia, M. Lapido, y J. Gómez, «"Optimización de sistemas centralizados de agua helada en la etapa prematura del diseño comercial". Ingeniería Mecánica», vol. 15, nº. 1, pp. 54–56, 2012.
[26] R. Montero, C. Pérez, E. Góngora, y S. Marrero, «"Predicción del consumo de electricidad y gas LP en un hotel mediante Redes Neuronales Artificiales". Energética», nº. 42, pp. 21–28, 2009.
[27] R. Salas, «"Redes Neuronales Artificiales"». 2006.
[28] W. Luera y L. Minim, «"Aplicación de las Redes Neuronales Artificiales en la modelización del tratamiento térmico de alimentos". Ciencia y Tecnología Alimentaria», vol. 3, nº. 002, pp. 81–88, 2001.
[29] J. Rabuñal, «"Uso de Técnicas de Inteligencia Artificial en Ingeniería Civil"», Tesis Doctoral, Universidad Da Coruña, A Coruña, España, 2007.
[30] M. Vera y J. Bustamante, «"Modelo dinámico para la generación de pronóstico usando redes neuronales artificiales (RNA)"». 2007.
[31] N. Piedra y J. López, «"Estudio de la Aplicación de Redes Neuronales Artificiales en la Ingeniería de Tráfico de Internet"». 2010.
[32] T. Callinan, «"Artificial Neural Network identification and control of the inverted pendulum"». 2003.
[33] O. Bombino, «"Control Predictivo Basado en Modelos Térmicos del Hotel Jagua"», Tesis de Grado, Universidad Central «Marta Abreu» de Las Villas, Villa Clara, Cuba, 2012.
[34] V. Cuza, «"Estudio Energético del Sistema de Climatización del Hotel Jagua"». 2010.
[35] S. Wang y X. Xu, «Simplified building model for transient thermal performance estimation using GA-based parameter identification». 2006.
[36] K. Lee y J. Braun, «Development of methods for determining demand-limiting setpoint trajectories in buildings using short-term measurements». 2008.
[37] J. Braun y N. Chaturvedi, «An inverse gray-box model for transient building load prediction». 2002.
[38] C. Ghiaus y I. Hazyuk, «Calculation of optimal thermal load of intermittently heated building». 2010.
[39] P. Bacher y H. Madsen, «Identifying suitable models for the heat dynamics of buildings». 2011.
[40] A. Piccion, F. Cvetreznik, V. Chauvie, y R. Barchiesi, «"Introducción Sistemas Complementarios"». 2009.
[41] M. de Armas y J. Gómez, «"Inteligencia Artificial Aplicada al Análisis de Sistemas Energéticos con Matlab"». 2009.
[42] E. Rich y K. Knight, *«Artificial Intelligence»*, 2da ed. 1994.
[43] R. Kurzweil, «"The age of intelligent machines"». 1990.

[44] G. Lugar y W. Stubblefied, «“Artificial intelligence: Structures and strategies for complex problem solving”». 1993.
[45] R. Schalkoff, «“Artificial intelligence: An engineering approach”». 1990.
[46] N. Kemper, «INTELIGENCIA COMPUTACIONAL APLICADA EN REDES DE TELECOMUICACIONES». .
[47] J. . Salao, «Estudio de las Técnicas de Inteligencia Artificial mediante el apoyo de un Software Educativo ». .
[48] M. . Sosa, «Introducción a las técnicas de inteligencia artificial aplicadas a la gestión financiera empresarial». sep-2006.
[49] A. Sözen y M. Özalp, «“Formulation based on artificial neural network of thermodynamic properties of ozone friendly refrigerant/absorbent couples"». 2005.
[50] J. Armas, Y. Valdivia, J. Gómez, y R. Reyes, «“Empleo de la inteligencia artificial en la determinación de propiedades de refrigerantes”. Ingenierías», vol. XI, n°. 40, 2007.
[51] X. Basogain, «“Redes Neuronales Artificiales y sus aplicaciones”». 2001.
[52] D. M. . Noriega, «“Aplicación de las redes neuronales en el análisis sensorial de alimentos”. Alimentaria», vol. 9, pp. 67–72, 1999.
[53] J. Hilera y V. Martínez, «Redes neuronales artificiales. Fundamentos, modelos y aplicaciones». 1995.
[54] R. González, R. Rico, y G. Kevrekidis, «“Identification ofdistribuid parameter systems: A neural net based approach”. Comp.Chem.Eng», vol. 22, 1998.
[55] R. Andrews, J. Diederich, y A. Tickle, «“A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks”». 1995.
[56] H. Wang, Y. Oh, y E. Yoon, «“Strategies for modeling and control of nonlinear chemical process using neural networks”. Comp.Chem.Eng.», vol. 22, 1998.
[57] K. Meert y M. Rijckaert, «“Intelligent modelling in the chemical process industry with neural networks: A case study”. Comp.Chem.Eng.», vol. 22, 1998.
[58] G. Bloch, «“Neural intelligent control for a steel plant”. IEEE Trans. Neural Networks», vol. 8, 1997.
[59] A. Levin y K. Narendra, «“Control of nonlinear dynamical systems using neural networks: Controllability and Stabilization”. IEEE Trans. Neural Networks», vol. 4, 1993.
[60] C. Oller y R. Guidici, «"Neural Network based approach for optimization applied to an industrial nylon-6, 6 polymerisation process”. Comp.Chem.Eng», vol. 22, 1998.
[61] R. Altissimi, «“Optimal operation of a separation plant using artificial neural networks”, Comp.Chem.Eng», vol. 22, 1998.
[62] H. Aguiar y R. Maciel, «“Modeling and optimization of pulp and paper process using neural networks”, Comp.Chem.Eng», vol. 22, 1998.
[63] C. Oropeza, «“capítulo 3: Redes Neuronales Recurrentes”». 2001.
[64] T. . Mitchel, *Machine Learning*. USA: , 1997.
[65] C. Lin y C. Lee, «“Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems”». 1996.
[66] T. Pham y L. Xing, «“Neural networs for identification, prediction and control”». 1995.
[67] C. Tsoi y A. Back, «“Locally Recurrent Globally Feedforward networks: A Critical Review of Architectures”. IEEE Transactions on Neural Networks», vol. 5, n°. 2, pp. 229–239, 1994.
[68] A. Serrano, E. Soria, y J. Martín, «“Redes Neuronales Artificiales”». 2010.
[69] B. Kröse y P. van der Smagt, *An Introduction to Neural Networks*, 8va ed. Amsterdam: , 1996.

[70] J. Elman, «"Finding Structure in Time". Cognitive Science», vol. 14, nº. 2, pp. 179–211, 1990.
[71] H. Demuth, M. Beale, y M. Hagan, «"Neural Network Toolbox 5 User´s Guide MatLab"». 2007.
[72] H. Demuth, M. Beale, y M. Hagan, «"Neural Network Toolbox 7 User´s Guide MatLab"». 2010.
[73] O. Olurotimi, «"Recurrent Neural Network Training with Feedforward Complexity". IEEE Transactions on Neural Networks», vol. 5, nº. 2, pp. 185–197, 1994.
[74] J. Hopfield, «"Neural Networks and Physical Systems with Emergent Collective Computational Abilities". Proceedings of the National Academy of Sciences», vol. 79, pp. 2554–2558, 1982.
[75] H. Demuth, M. Beale, y M. Hagan, *Neural Network Design*. USA: PWS Publishing Company, 1996.
[76] K. Warwick, W. Irwin, y K. Hunt, «"Neural networks for control and systems". IEE Control Engineering Series», vol. 46, 1992.
[77] J. van den Berg y J. Bioch, «"Constrained Optimization with a Continuous Hopfield-Lagrange Model"». 1996.
[78] D. Lee, «"Pattern sequence recognition using a time-varying Hopfield network". IEEE Transactions on Neural Networks», vol. 13, nº. 2, pp. 330–342, 2002.
[79] S. Young, P. Scott, y C. Bandera, «"Foveal automatic target recognition using a multiresolution neural network". IEEE Transactions on Image Processing», vol. 7, nº. 8, pp. 1122–1135, 1998.
[80] G. Carpenter y S. Grossberg, «"A massively parallel architecture for a selforganizing neural pattern recognition machine". Computer Vision, Graphics, an Image Processing», vol. 37, pp. 54–115, 1987.
[81] G. Carpenter y S. Grossberg, «"The ART of Adaptive Pattern Recognition by a self-organizing neural network". IEEE Computer», pp. 77–88, mar. 1988.
[82] B. Martín del Brío y A. Sanz Molina, *Redes Neuronales y Sistemas Difusos*, 2da ed. .
[83] G. Cauwenberghs, «"A fast stochastic Error-Descent Algorithm for supervised learning and optimization". Neural Information Processing Systems.», pp. 244–251, 1993.
[84] I. De Falco, A. Della Cioppa, P. Natale, y E. Tarantino, «"Artificial Neural Networks Optimization by means of Evolutionary Algorithms". Soft Computing in Engineering Design and Manufacturing.», 1997.
[85] F. Tanco, «Introducción a las redes neuronales artificiales». .
[86] L. Federico, «Entrenamiento de Redes Neuronales Basado en Algortimos Evolutivos», Tesis de Grado, Universidad de Buenos Aires, Buenos Aires, Argentina, 2005.
[87] P. Haves y B. Hencey, «"Model Predictive Control of HVAC Systems: Implementation and Testing at the University of California, Merced"». 2010.
[88] T. McKinley y A. Alleyne, «"Identification of building model parameters and loads using on-site data logs".», presented at the Third National Conference of IBPSA-USA, USA, 2008.
[89] Z. O'Neill y S. Narayanan, «"Model-based thermal load estimation in buildings".», presented at the Fourth National Conference of IBPSA-USA, USA, 2010.
[90] Q. Zhou y S. Wang, «"A grey-box model of next-day building thermal load prediction for energy-efficient control". International Journal of Energy Research». 2008.
[91] ANSI/ASHRAE, «"Standard method of test for the evaluation of building energy analusis computer programs. Atlanta, Georgia, R. American Society of Heating, and Air-Conditioning and Engineers". Standar 140-2001.» 2001.

[92] M. Gwerder y J. Tödtli, «Predictive Control for Integrated Room Automation», presented at the 8th REHVA World Congress for Building Technologies - CLIMA, Lausanne, Switzerland, 2005.
[93] Meteotest, «Handbook part II: Theory. METEONORM v6.1.» 2010.
[94] T. A. Davis y K. Sigmon, «MATLAB Primer (7 ed.). Chapman and Hall/CRC». 2005.
[95] I. THE MATHWORKS, «Matlab, Edición del estudiante. Guía del usuario». Prentice Hall, Madrid, 2004.
[96] J. García y J. I. Rodríguez, «Aprenda Matlab 7.0 cómo si estuviera en primero». 2006.
[97] R. Campillo, «Apuntes de Sistemas de Control». 2004.
[98] J. . Marín, «Introducción a las redes neuronales aplicadas». .
[99] G. Visanzay, «Identificación de un proceso de secado utilizando Redes Neuronales Dinámicas». .
[100] S. García, A. Fernández, J. Luengo, y F. Herrera, «"Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power". Information Sciences.», pp. 2044–2064, 2010.
[101] M. Friedman, «"The use of ranks to avoid the assumption of normality implicit in the analysis of variance". Journal of the American Statistical Association», pp. 674–701, 1937.
[102] M. Friedman, «"A comparison of alternative tests of significance for the problem of m rankings". Annals of Mathematical Statistics.», pp. 86–92, 1940.
[103] J. . Hodges y E. . Lehmann, «"Ranks methods for combination of independent experiments in analysis of variance". Annals of Mathematical Statistics.», pp. 482–497, 1962.
[104] D. Quade, «"Using weighted rankings in the analysis of complete blocks with additive block effects". Journal of the American Statistical Association.», pp. 680–683, 1979.
[105] K. Doksum, «"Robust procedures for some linear models with one observation per cell". Annals of Mathematical Statistics», pp. 878–883, 1967.

Bibliografía

[1] A. Abbassi y L. Bahar, «"Application of neural network for the modeling and control of evaporative condenser cooling load". Applied Thermal Engineering», vol. 25, nº. 17–18, pp. 3176–3186, dic. 2005.

[2] H. Aguiar y R. Maciel, «"Modeling and optimization of pulp and paper process using neural networks", Comp.Chem.Eng», vol. 22, 1998.

[3] F. Alonso, «Desarrollo de un simulador de una planta de lodos activados», Tesis de Grado, Universidad Central «Marta Abreu» de Las Villas, Santa Clara, Cuba, 2009.

[4] R. Altissimi, «"Optimal operation of a separation plant using artificial neural networks", Comp.Chem.Eng», vol. 22, 1998.

[5] M. Andrade y N. Zarza, «"Cálculo de la Red Hidráulica para el Circuito de Enfriamiento de un Proceso de Inyección de Plásticos"», Tesis de Grado, Escuela Superior de Ingeniería Mecánica y Eléctrica, México D.F, 2010.

[6] R. Andrews, J. Diederich, y A. Tickle, «"A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks"». 1995.

[7] Anónimo, «Sistema de aire acondicionado limpio en Toronto». 2002.

[8] Anónimo, «Capítulo 3: Perceptron Multicapa». .

[9] ANSI/ASHRAE, «"Standard method of test for the evaluation of building energy analusis computer programs. Atlanta, Georgia, R. American Society of Heating, and Air-Conditioning and Engineers". Standar 140-2001.» 2001.

[10] A. Anta, J. L. Zamora, y R. Rodríguez, «Manual de referencia de MATLAB & SIMULINK». feb-2000.

[11] J. Armas, M. Lapido, J. Gómez, y P. Roque, «"Optimización termoeconómica por agua helada a partir de técnicas de inteligencia artificial". Ingenierías», vol. XI, nº. 40, 2008.

[12] J. Armas, M. Lapido, J. Gómez, y Y. Valdivia, «"Evaluación termodinámica de sistemas de climatización centralizados por agua helada usando herramientas de inteligencia artificial". Revista Ingeniería e Investigación», vol. 31, nº. 2, pp. 134–142, 2011.

[13] J. Armas, Y. Valdivia, J. Gómez, y R. Reyes, «"Empleo de la inteligencia artificial en la determinación de propiedades de refrigerantes". Ingenierías», vol. XI, nº. 40, 2007.

[14] P. Bacher y H. Madsen, «Identifying suitable models for the heat dynamics of buildings». 2011.

[15] X. Basogain, «"Redes Neuronales Artificiales y sus aplicaciones"». 2001.

[16] A. . Ben-Nakhi y M. . Mahmoud, «"Cooling load prediction for buildings using general regression neural networks".Energy Conversion and Management», vol. 45, pp. 2127–2141, ago. 2004.

[17] J. Bermúdez, «Estrategias de control multivariable para un sistema de compresión de vapor». 2008.

[18] G. Bloch, «"Neural intelligent control for a steel plant". IEEE Trans. Neural Networks», vol. 8, 1997.

[19] O. Bombino, «"Control Predictivo Basado en Modelos Térmicos del Hotel Jagua"», Tesis de Grado, Universidad Central «Marta Abreu» de Las Villas, Villa Clara, Cuba, 2012.

[20] J. Braun y N. Chaturvedi, «An inverse gray-box model for transient building load prediction». 2002.

[21] T. Callinan, «"Artificial Neural Network identification and control of the inverted pendulum"». 2003.

[22] R. Campillo, «Apuntes de Sistemas de Control». 2004.

[23] G. Carpenter y S. Grossberg, «"A massively parallel architecture for a selforganizing neural pattern recognition machine". Computer Vision, Graphics, an Image Processing», vol. 37, pp. 54–115, 1987.
[24] G. Carpenter y S. Grossberg, «"The ART of Adaptive Pattern Recognition by a self-organizing neural network". IEEE Computer», pp. 77–88, mar. 1988.
[25] L. Castellanos, «"Modelado y Control Predictivo de una Habitación Hotelera"», Tesis de Grado, Universidad Central «Marta Abreu» de Las Villas, Villa Clara, Cuba, 2011.
[26] M. Castilla y J. Álvarez, «Técnicas de Control de Confort en edificios.» 2010.
[27] G. Cauwenberghs, «"A fast stochastic Error-Descent Algorithm for supervised learning and optimization". Neural Information Processing Systems.», pp. 244–251, 1993.
[28] B. Collymore, «INTERPOLACIÓN DE DATOS ESPACIALES AMBIENTALES», Tesis de Grado, Universidad de Cienfuegos «Carlos rafael Rodríguez», Cienfuegos, Cuba, 2010.
[29] V. Cuza, «"Estudio Energético del Sistema de Climatización del Hotel Jagua"». 2010.
[30] T. A. Davis y K. Sigmon, «MATLAB Primer (7 ed.). Chapman and Hall/CRC». 2005.
[31] M. de Armas y J. Gómez, «"Inteligencia Artificial Aplicada al Análisis de Sistemas Energéticos con Matlab"». 2009.
[32] I. De Falco, A. Della Cioppa, P. Natale, y E. Tarantino, «"Artificial Neural Networks Optimization by means of Evolutionary Algorithms". Soft Computing in Engineering Design and Manufacturing.», 1997.
[33] H. Demuth, M. Beale, y M. Hagan, *Neural Network Design*. USA: PWS Publishing Company, 1996.
[34] H. Demuth, M. Beale, y M. Hagan, «"Neural Network Toolbox 5 User´s Guide MatLab"». 2007.
[35] H. Demuth, M. Beale, y M. Hagan, «"Neural Network Toolbox 7 User´s Guide MatLab"». 2010.
[36] K. Doksum, «"Robust procedures for some linear models with one observation per cell". Annals of Mathematical Statistics», pp. 878–883, 1967.
[37] J. Elman, «"Finding Structure in Time". Cognitive Science», vol. 14, n°. 2, pp. 179–211, 1990.
[38] L. Federico, «Entrenamiento de Redes Neuronales Basado en Algortimos Evolutivos», Tesis de Grado, Universidad de Buenos Aires, Buenos Aires, Argentina, 2005.
[39] E. Fock, F. Garde, P. Lauret, y J. Gatina, «"Artificial Neural Networks for the Prediction of Cooling Loads of HVAC Systems: A Case Study Under Tropical Climate".», presented at the World Renewable Energy Congress VI, 2000, pp. 661–664.
[40] M. Friedman, «"The use of ranks to avoid the assumption of normality implicit in the analysis of variance". Journal of the American Statistical Association», pp. 674–701, 1937.
[41] M. Friedman, «"A comparison of alternative tests of significance for the problem of m rankings". Annals of Mathematical Statistics.», pp. 86–92, 1940.
[42] J. Fritsch, «Modular Neural Networks for Speech Recognition», Tesis de Maestría, Universidad Carnegie Mellon, 1996.
[43] J. Gámez y J. Flores, «Introducción a la Inteligencia Artificial». 2008.
[44] J. García y J. I. Rodríguez, «Aprenda Matlab 7.0 cómo si estuviera en primero». 2006.
[45] S. García, A. Fernández, J. Luengo, y F. Herrera, «"Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power". Information Sciences.», pp. 2044–2064, 2010.
[46] G. Ge, F. Xiao, y S. Wang, «"Neural network based prediction method for preventing condensation in chilled ceiling systems". Energy and Buildings», vol. 45, pp. 290–298,

feb. 2012.
[47] P. Gestal, «"Computación Evolutiva para el Proceso de Selección de Variables en Espacios de Búsqueda Multimodales"», Tesis Doctoral, Universidad Da Coruña, A Coruña, España, 2009.
[48] C. Ghiaus y I. Hazyuk, «Calculation of optimal thermal load of intermittently heated building». 2010.
[49] F. Godiño, «Tutorial de MATLAB». 2000.
[50] A. González, «Medidas de ahorro». .
[51] R. González, R. Rico, y G. Kevrekidis, «"Identification ofdistribuid parameter systems: A neural net based approach". Comp.Chem.Eng», vol. 22, 1998.
[52] J. Gutiérrez, «Introducción a la Inteligencia Artificial. Aplicaciones». 2000.
[53] M. Gwerder y J. Tödtli, «Predictive Control for Integrated Room Automation», presented at the 8th REHVA World Congress for Building Technologies - CLIMA, Lausanne, Switzerland, 2005.
[54] D. Gyalistras, «Use of MPC for Building Control». 2010.
[55] P. Haves y B. Hencey, «"Model Predictive Control of HVAC Systems: Implementation and Testing at the University of California, Merced"». 2010.
[56] L. Hernádez, «"Predicción del consumo energético del Hotel Jagua aplicando la simulación termodinámica y la inteligencia artificial".» 2006.
[57] J. Hilera y V. Martínez, «Redes neuronales artificiales. Fundamentos, modelos y aplicaciones». 1995.
[58] J. . Hodges y E. . Lehmann, «"Ranks methods for combination of independent experiments in analysis of variance". Annals of Mathematical Statistics.», pp. 482–497, 1962.
[59] J. Hopfield, «"Neural Networks and Physical Systems with Emergent Collective Computational Abilities". Proceedings of the National Academy of Sciences», vol. 79, pp. 2554–2558, 1982.
[60] Z. Hou, Z. Lian, Y. Yao, y X. Yuan, «"Cooling-load prediction by the combination of rough set theory and an artificial neural-network based on data-fusion technique". Applied Energy», vol. 83, n°. 9, pp. 1033–1046, sep. 2006.
[61] N. Kemper, «INTELIGENCIA COMPUTACIONAL APLICADA EN REDES DE TELECOMUICACIONES». .
[62] T. Kohonen, «"Self-Organization and Associative Memory". Springer Series in Information Sciences», vol. 8, 1984.
[63] B. Kröse y P. van der Smagt, *An Introduction to Neural Networks*, 8va ed. Amsterdam: , 1996.
[64] R. Kurzweil, «"The age of intelligent machines"». 1990.
[65] S. Kwok y E. Lee, «"A study of the importance of occupancy to building cooling load in prediction by intelligent approach". Energy Conversion and Management», vol. 52, n°. 7, pp. 2555–2564, jul. 2011.
[66] J. Landa y Y. Delgado, «"SOBRE EL ESTADO DEL ARTE EN SISTEMAS CENTRALIZADOS DE AIRE ACONDICIONADO CON AGUA FRÍA".Centro de Estudios de Combustión y Energía, Facultad de Ingenierías Química y Mecánica. Universidad de Matanzas "Camilo Cienfuegos"». 2007.
[67] D. Lee, «"Pattern sequence recognition using a time-varying Hopfield network". IEEE Transactions on Neural Networks», vol. 13, n°. 2, pp. 330–342, 2002.
[68] K. Lee y J. Braun, «Development of methods for determining demand-limiting setpoint

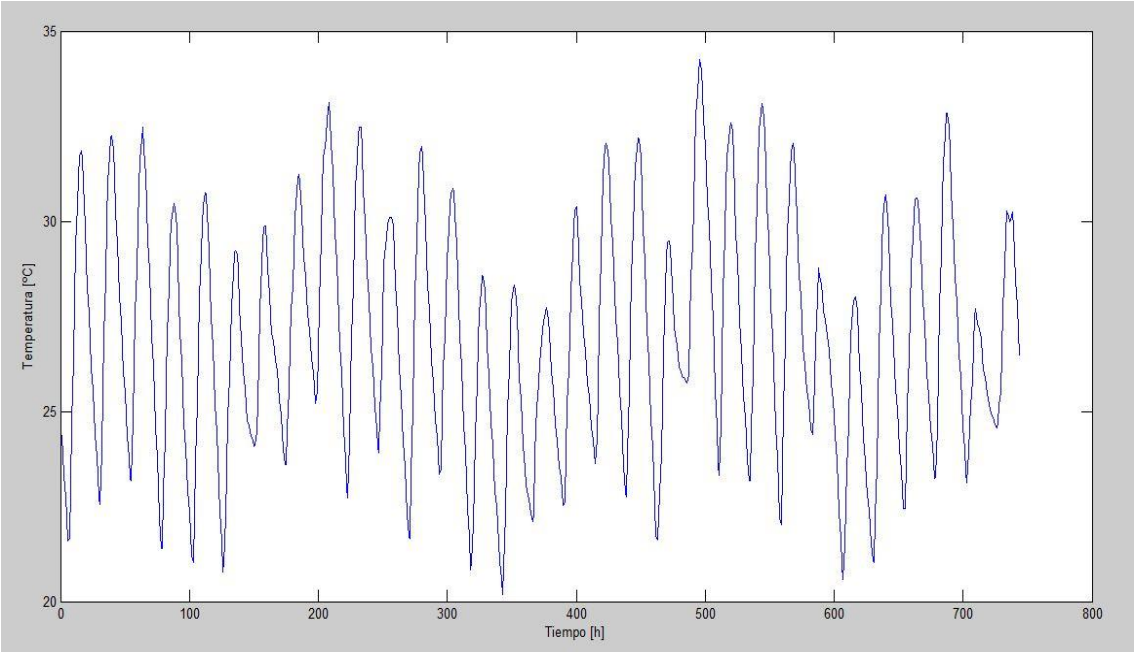
trajectories in buildings using short-term measurements». 2008.
[69] M. . Leung, C. . Tse, L. . Lai, y T. . Chow, «"The use of occupancy space electrical power demand in building cooling load prediction". Energy and Buildings», vol. 55, pp. 151–163, dic. 2012.
[70] A. Levin y K. Narendra, «"Control of nonlinear dynamical systems using neural networks: Controllability and Stabilization". IEEE Trans. Neural Networks», vol. 4, 1993.
[71] Q. Li, Q. Meng, J. Cai, H. Yoshino, y A. Mochida, «"Predicting hourly cooling load in the building: A comparison of support vector machine and different artificial neural networks". Energy Conversion and Management», vol. 50, nº. 1, pp. 90–96, ene. 2009.
[72] C. Lin y C. Lee, «"Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems"». 1996.
[73] W. Luera y L. Minim, «"Aplicación de las Redes Neuronales Artificiales en la modelización del tratamiento térmico de alimentos". Ciencia y Tecnología Alimentaria», vol. 3, nº. 002, pp. 81–88, 2001.
[74] G. Lugar y W. Stubblefied, «"Artificial intelligence: Structures and strategies for complex problem solving"». 1993.
[75] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, y P. Haves, «"Model Predictive Control for the Operation of Building Cooling Systems"». 2010.
[76] Z. Ma y S. Wang, «"An optimal control strategy for complex building central chilled water systems for practical and real-time applications". Building and Environment», vol. 44, pp. 1188–1198, 2009.
[77] C. Mallo, «"Predicción de la demanda eléctrica horaria mediante redes neuronales artificiales"». 2010.
[78] J. . Marín, «Introducción a las redes neuronales aplicadas». .
[79] B. Martín del Brío y A. Sanz Molina, *Redes Neuronales y Sistemas Difusos*, 2da ed. .
[80] T. McKinley y A. Alleyne, «"Identification of building model parameters and loads using on-site data logs".», presented at the Third National Conference of IBPSA-USA, USA, 2008.
[81] K. Meert y M. Rijckaert, «"Intelligent modelling in the chemical process industry with neural networks: A case study". Comp.Chem.Eng.», vol. 22, 1998.
[82] Meteotest, «Handbook part II: Theory. METEONORM v6.1.» 2010.
[83] J. J. Mier, «Curso de MATLAB». 2000.
[84] T. . Mitchel, *Machine Learning*. USA: , 1997.
[85] S. Montelier, M. de Armas, A. Borroto, J. Gómez, y C. Pérez, «"Inteligencia Artificial Aplicada a la Reducción del Consumo Energético de un Sistema de Climatización por agua Helada en un Hotel Turístico". Energética», nº. 39, pp. 13–18, jul. 2008.
[86] R. Montero, J. Hechavarría, A. Legrá, A. Borroto, y R. Santos, «"Análisis y síntesis de la operación de circuitos secundarios de agua fría en climatización centralizada". Ingeniería Mecánica», vol. 15, nº. 2, pp. 83–94, 2012.
[87] R. Montero, C. Pérez, E. Góngora, y S. Marrero, «"Predicción del consumo de electricidad y gas LP en un hotel mediante Redes Neuronales Artificiales". Energética», nº. 42, pp. 21–28, 2009.
[88] J. . Moreno y J. . Sánchez, «Evaluación y análisis del sistema centralizado de aire acondicionado en el hotel Sandals Royal Hicacos.» .
[89] D. M. . Noriega, «"Aplicación de las redes neuronales en el análisis sensorial de alimentos". Alimentaria», vol. 9, pp. 67–72, 1999.
[90] C. Oller y R. Guidici, «"Neural Network based approach for optimization applied to an

industrial nylon-6, 6 polymerisation process". Comp.Chem.Eng», vol. 22, 1998.
[91] O. Olurotimi, «"Recurrent Neural Network Training with Feedforward Complexity". IEEE Transactions on Neural Networks», vol. 5, nº. 2, pp. 185–197, 1994.
[92] Z. O'Neill y S. Narayanan, «"Model-based thermal load estimation in buildings".», presented at the Fourth National Conference of IBPSA-USA, USA, 2010.
[93] C. Oropeza, «"capítulo 3: Redes Neuronales Recurrentes"». 2001.
[94] S. Pandey y D. Hindoliya, «"Artificial neural network for predation of cooling load reduction using green roof over building in Sustainable City". Sustainable Cities and Society», vol. 3, pp. 37–45, jul. 2012.
[95] T. Pham y L. Xing, «"Neural networs for identification, prediction and control"». 1995.
[96] A. Piccion, F. Cvetreznik, V. Chauvie, y R. Barchiesi, «"Introducción Sistemas Complementarios"». 2009.
[97] N. Piedra y J. López, «"Estudio de la Aplicación de Redes Neuronales Artificiales en la Ingeniería de Tráfico de Internet"». 2010.
[98] L. Pinas, «TRATAMIENTO BIOLOGICO DE AGUA RESIDUALES UTILIZANDO MATLAB», Tesis de Grado, Universidad Central «Marta Abreu» de Las Villas, Santa Clara, Cuba, 2009.
[99] R. Pino, D. De la Fuente, J. Parreño, y P. Priore, «"Aplicación de Redes Neuronales Artificiales a la Previsión de Series Temporales no Estacionarias o No Invertibles". Questiio», vol. 26, nº. 3, 2002.
[100] S. Prívara y J. Siroký, «"Model predictive control of a building heating system The first experience". Energy and Buildings», nº. 43, pp. 564–572, 2011.
[101] D. Quade, «"Using weighted rankings in the analysis of complete blocks with additive block effects". Journal of the American Statistical Association.», pp. 680–683, 1979.
[102] R. Quintero, «Control de procesos industriales utilizando motores asincrónicos controlados por campo orientado», Tesis de Grado, Universidad Central «Marta Abreu» de Las Villas, Santa Clara, Cuba, 2008.
[103] J. Rabuñal, «"Uso de Técnicas de Inteligencia Artificial en Ingeniería Civil"», Tesis Doctoral, Universidad Da Coruña, A Coruña, España, 2007.
[104] E. Rich y K. Knight, *«Artificial Intelligence»*, 2da ed. 1994.
[105] J. . Rivera, «Red Neuronal Modelo de Hpfield». .
[106] J. A. Rojas, «Tutorial de MATLAB». 2000.
[107] R. Ruelas, «La importancia de ahorrar energía». 2008.
[108] S. . Russell y P. Norving, *Artificial Intelligence: A Modern Approach*. New Jersey, USA: , 1995.
[109] J. . Salao, «Estudio de las Técnicas de Inteligencia Artificial mediante el apoyo de un Software Educativo ». .
[110] R. Salas, «"Redes Neuronales Artificiales"». 2006.
[111] R. Schalkoff, «"Artificial intelligence: An engineering approach"». 1990.
[112] A. Serrano, E. Soria, y J. Martín, «"Redes Neuronales Artificiales"». 2010.
[113] L. Shi y J. Wang, «"Application of Artificial Neural Network to Predict the Hourly Cooling Load of an Office Building"». 2009.
[114] J. Siroký y F. Oldewurtel, «"Experimental analysis of model predictive control for an energy efficient building heating system". Applied Energy», vol. 88, 2011.
[115] M. . Sosa, «Introducción a las técnicas de inteligencia artificial aplicadas a la gestión financiera empresarial». sep-2006.
[116] G. Sotolongo y M. Guzmán, «"Aplicaciones de las redes neuronales. El caso de la

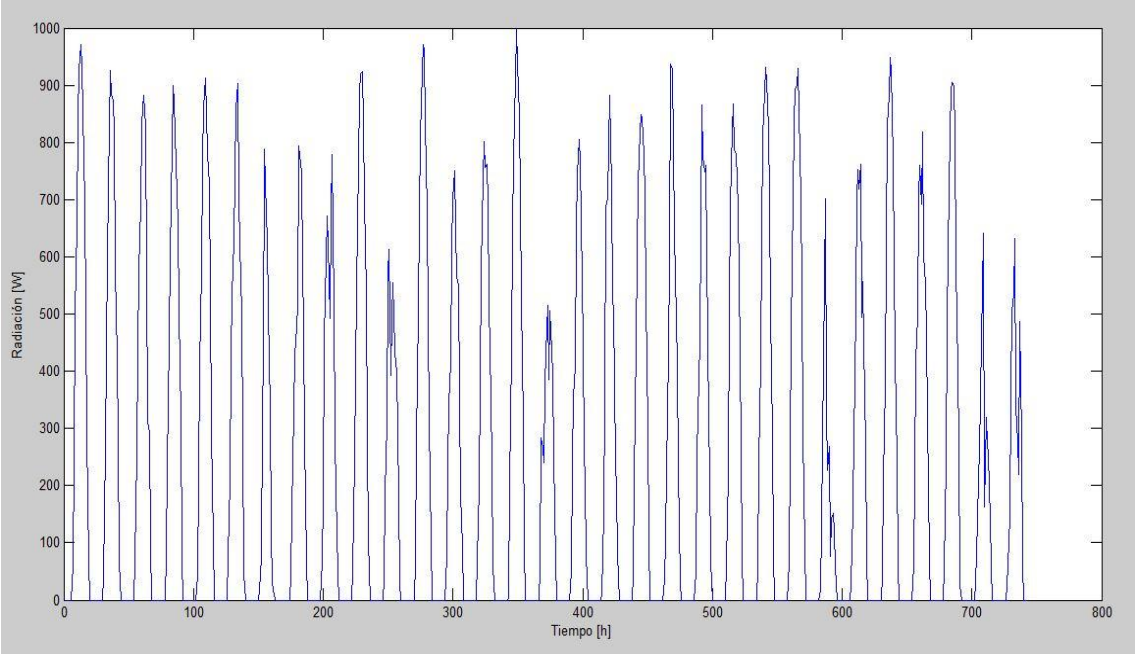
bibliometría". Ciencias de la Información», vol. 32, nº. 1, pp. 27–34, 2001.
[117] A. Sözen y M. Özalp, «"Formulation based on artificial neural network of thermodynamic properties of ozone friendly refrigerant/absorbent couples"». 2005.
[118] F. Tanco, «Introducción a las redes neuronales artificiales». .
[119] I. THE MATHWORKS, «Matlab, Edición del estudiante. Guía del usuario». Prentice Hall, Madrid, 2004.
[120] C. Tsoi y A. Back, «"Locally Recurrent Globally Feedforward networks: A Critical Review of Architectures". IEEE Transactions on Neural Networks», vol. 5, nº. 2, pp. 229–239, 1994.
[121] Y. Valdivia, M. Lapido, y J. Gómez, «"Optimización de sistemas centralizados de agua helada en la etapa prematura del diseño comercial". Ingeniería Mecánica», vol. 15, nº. 1, pp. 54–56, 2012.
[122] J. van den Berg y J. Bioch, «"Constrained Optimization with a Continuous Hopfield-Lagrange Model"». 1996.
[123] J. . Van Ryzin y T. Leraand, «"Air Conditioning with Deep Seawater: A cost- Effective Alternative, Ocean Resources 2000". Technology Magazine», p. 37, sep. 1992.
[124] M. Vera y J. Bustamante, «"Modelo dinámico para la generación de pronóstico usando redes neuronales artificiales (RNA)"». 2007.
[125] G. Visanzay, «Identificación de un proceso de secado utilizando Redes Neuronales Dinámicas». .
[126] H. Wang, Y. Oh, y E. Yoon, «"Strategies for modeling and control of nonlinear chemical process using neural networks". Comp.Chem.Eng.», vol. 22, 1998.
[127] S. Wang y X. Xu, «Simplified building model for transient thermal performance estimation using GA-based parameter identification». 2006.
[128] K. Warwick, W. Irwin, y K. Hunt, «"Neural networks for control and systems". IEE Control Engineering Series», vol. 46, 1992.
[129] D. Willett, C. Busch, y F. Siebert, «"Fast Image Analysis using Kohonen Maps"». 1994.
[130] R. Williams y D. Zipser, «"A Learning Algorithm for Continually Running Fully Recurrent Neural Networks"». 1989.
[131] S. Young, P. Scott, y C. Bandera, «"Foveal automatic target recognition using a multiresolution neural network". IEEE Transactions on Image Processing», vol. 7, nº. 8, pp. 1122–1135, 1998.
[132] Q. Zhou y S. Wang, «"A grey-box model of next-day building thermal load prediction for energy-efficient control". International Journal of Energy Research». 2008.

Anexos

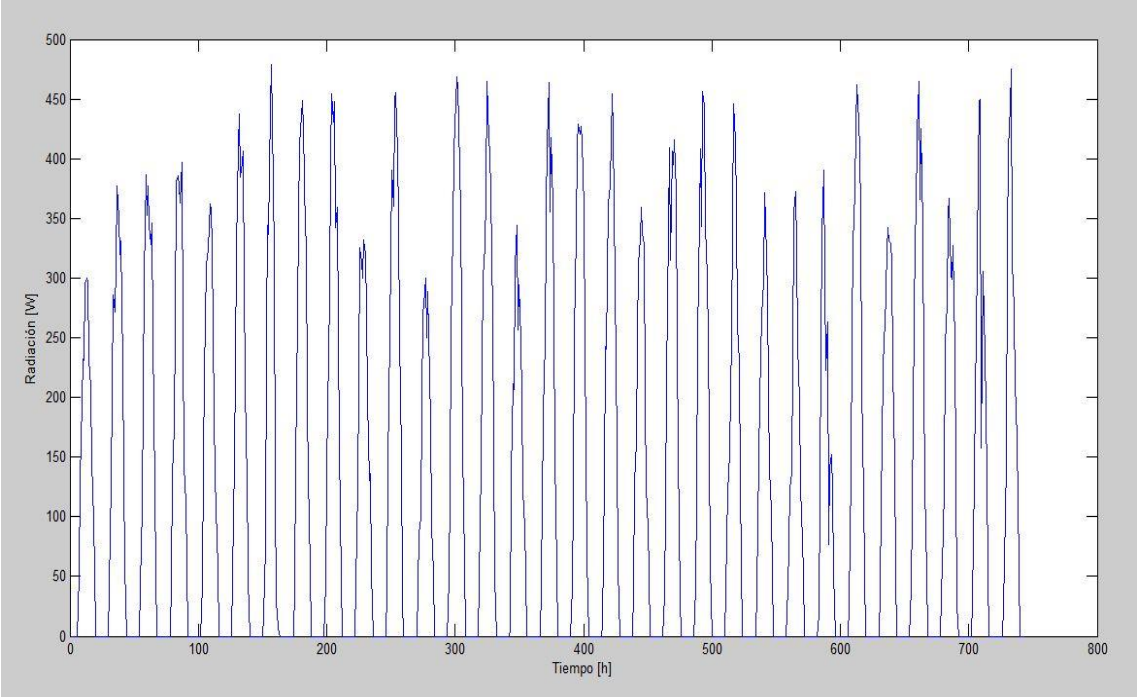
Anexo 1. Valores de la temperatura ambiente durante el mes de Julio del año 2011.



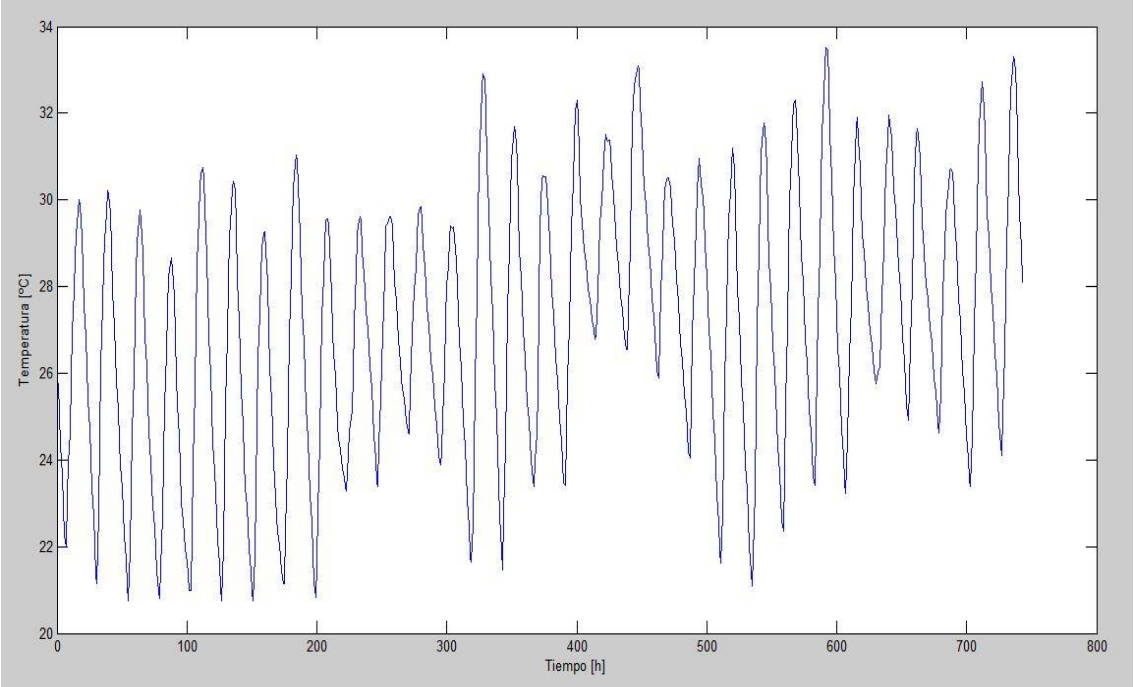
Anexo 2. Valores de la radiación solar global durante el mes de Julio del año 2011.



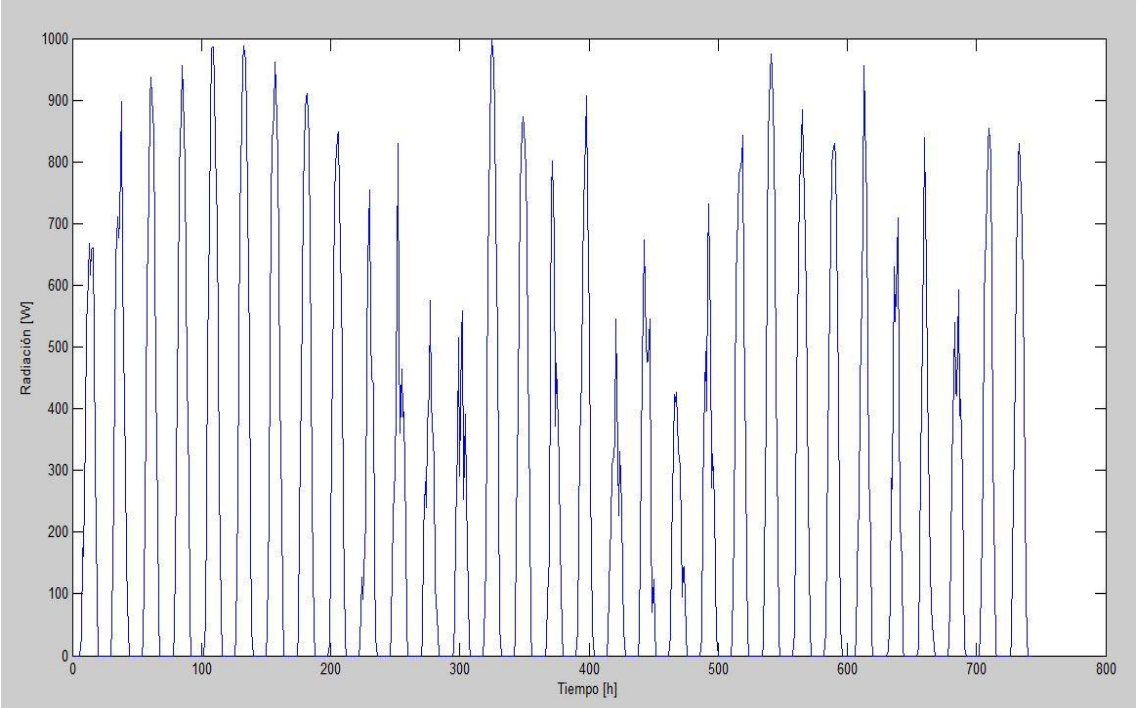
Anexo 3. Valores de la radiación solar difusa durante el mes de Julio del año 2011.



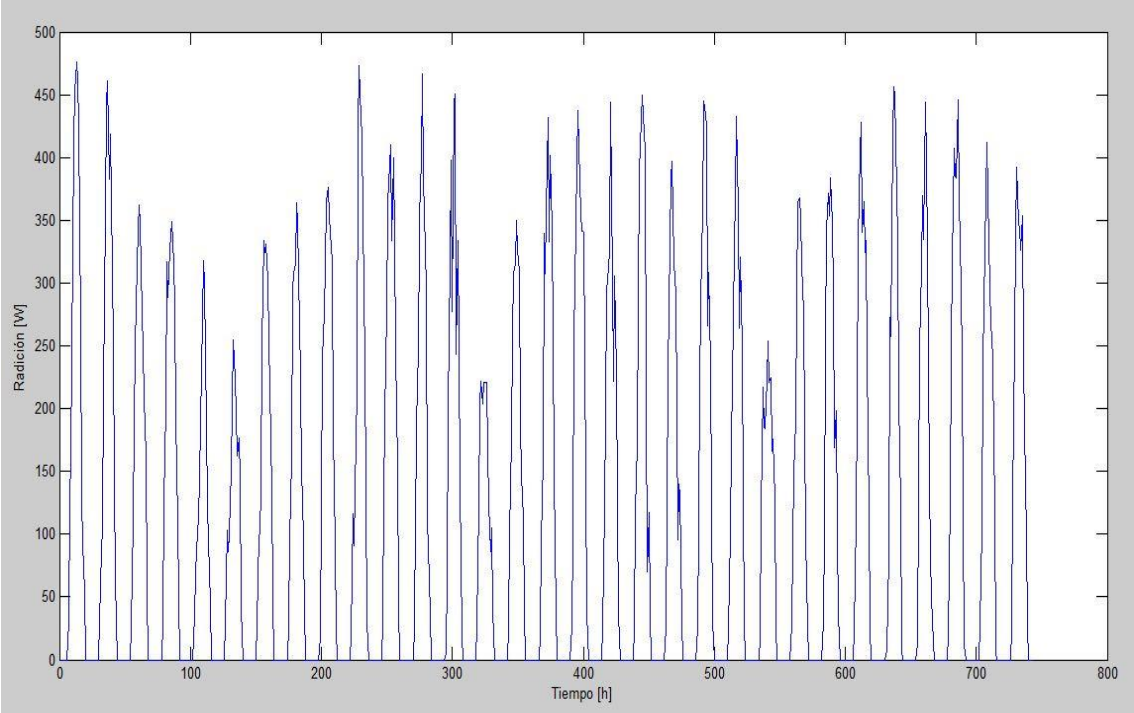
Anexo 4. Valores de la temperatura ambiente durante el mes de Agosto del año 2011.



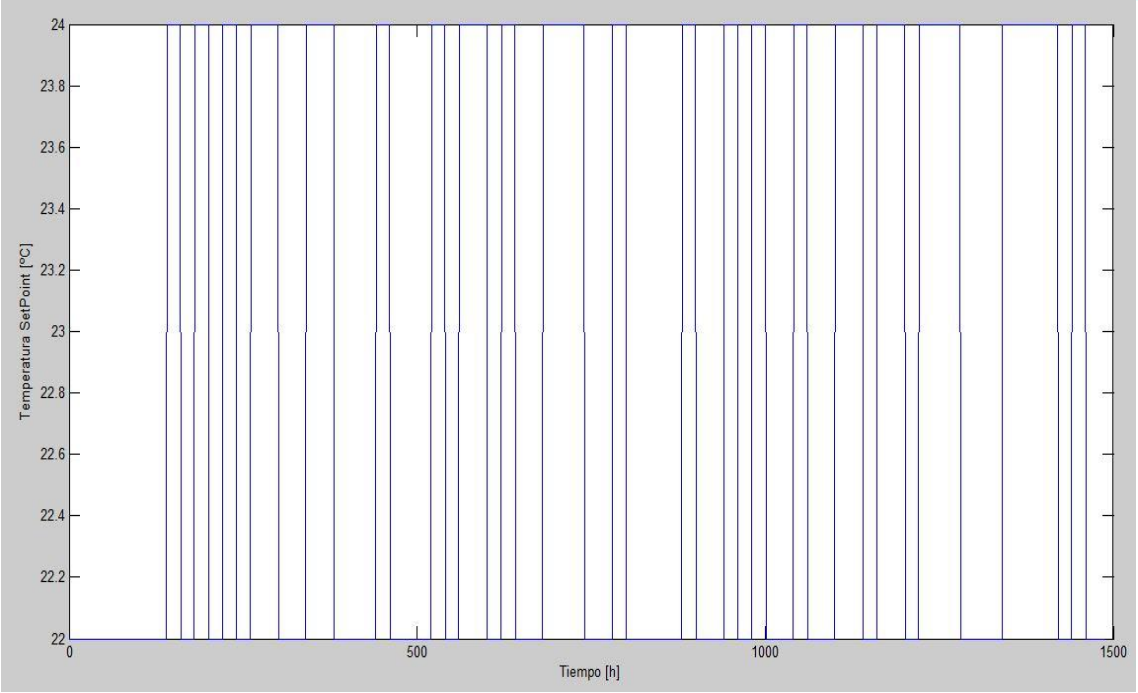
Anexo 5. Valores de la radiación solar global durante el mes de Agosto del año 2011.



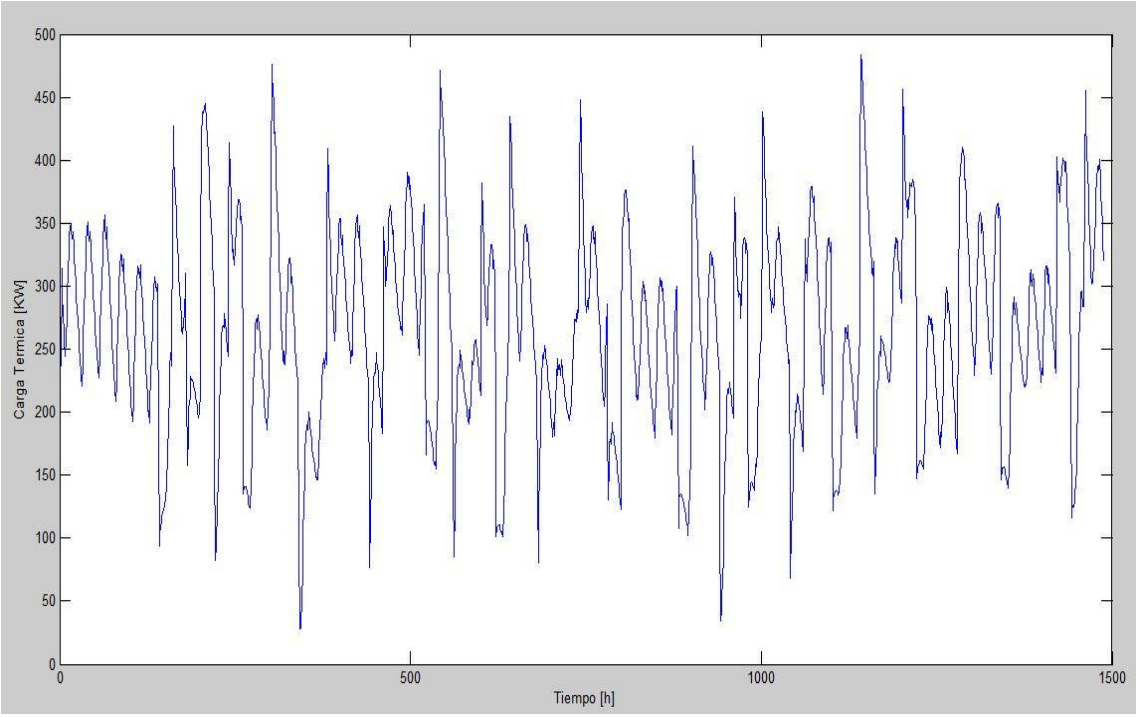
Anexo 6. Valores de la radiación solar difusa durante el mes de Agosto del año 2011.



Anexo 7. Valores de la temperatura de set point durante los meses de Julio y Agosto del año 2011.



Anexo 8. Valores de la carga térmica durante los meses de Julio y Agosto del año 2011.



Anexo 9. Código generalizado de la implementación de los modelos en Matlab: Etapa 1.

```
%% Características principales del modelo
%% Nombre del modelo Etapa1: creación y entrenamiento.

% se cargan los datos
load TambJ.mat;
load GRadJ.mat;
load DRadJ.mat;
load Tprbs_suave_05.mat;
load CT_prbs_suave_05;
load BiosC;
load PesosInputs;
load PesosC;

% se definen las variables del modelo y se realiza el
preprocesamiento de los datos

TemperaturaAmbienteJ=TambJ(1:1:744)';%%Temperatura ambiente en
Julio

RadiacionGlobalJ=GRadJ(1:1:744)';%%Radiacion global en Julio

RadiacionDifusaJ=DRadJ(1:1:744)';%%Radiacion Difusa en Julio

TemperaturaSetPointJ=Tprbs_suave_05(1:1:744)';%%Temperatura de
setpoint (la que se desea)

CargaTermicaJ=CT_prbs_suave_05(1:1:744)';%%Carga térmica obtenida
del trnsys

Entradas_Etapa1Modelo=[TemperaturaAmbienteJ;RadiacionGlobalJ;Radi
acionDifusaJ;TemperaturaSetPointJ];%%Creo el vector de entradas

[Entradas_Etapa1Modelo,estructura_entradas_Etapa1Modelo]=mapminma
x(Entradas_Etapa1Modelo);%%Prepocesamiento de los datos

Target_Etapa1Modelo=CargaTermicaJ;%% Creo el vector target

[Target_Etapa1Modelo,EstructuraTarget_Etapa1Modelo]=mapminmax(Tar
get_Etapa1Modelo);%% Prepocesamiento de los datos

%se convierte de consecutivo a sequencial
Entradas_Etapa1Modelo = con2seq(Entradas_Etapa1Modelo);
Target_Etapa1Modelo = con2seq(Target_Etapa1Modelo);
```

```
% se definen los delays
delaysModelo = (1:2);

%%Creación de la Red (se redefinen las entradas y los targets)
Red_Modelo=Funcion_Creacion(delaysModelo,cantidad_de_neuronas_por
_capas_ocultas,'función_entrenamiento');%%en esta línea se crea
el modelo mediante la función correspondiente

[VectorEntradas_Etapa1Modelo,DelaysEntradasModelo_Etapa1,DelaysTa
rgetModelo_Etapa1,VectorTarget_Etapa1Modelo]=preparets(Red_Modelo
,Entradas_Etapa1Modelo,Target_Etapa1Modelo);

%% Entrenamiento

Red_Modelo.trainParam.show = longitudes_de_muestra;

Red_Modelo.trainParam.epochs = cantidad_iteraciones;

Red_Modelo.trainParam.goal = error_minimo;

Red_Modelo.performFcn = 'mse'; % función de error

Red_Modelo.layers{1}.transferFcn='tansig';%función              de
transferencia de la capa oculta 1; esto se realiza para cada capa
oculta que tenga el modelo

Red_Modelo.layers{2}.transferFcn='purelin';%función             de
transferencia de la capa de salida

Red_Modelo.inputWeights{1,1}.learnFcn='función_aprendizaje';%func
ión de aprendizaje de las entradas

Red_Modelo.layerWeights{1,1}.learnFcn='función_aprendizaje';%func
ión de aprendizaje entre capas; esto se realiza para cada capa
que tenga el modelo

% la siguiente linea entrena el modelo creado anteriormente
Red_Modelo=train(Red_Modelo,VectorEntradas_Etapa1Modelo,VectorTar
get_Etapa1Modelo,DelaysEntradasModelo_Etapa1,DelaysTargetModelo_E
tapa1);

Red_Modelo.IW{1,1} = PesosInputs;%se asignan los valores de los
pesos de las entradas

Red_Modelo.LW{1,1} = PesosC;%se asignan los valores de los pesos
entre capas; esto se realiza para cada capa del modelo

Red_Modelo.b{1} = BiosC;%se asignan los valores de los bios de
```

```
cada capa; esto se realiza para cada capa del modelo

% Se simula el modelo con los mismos valores con que se realizo
del entrenamiento
SalidaSimulacionEntrenamientoEtapa1Modelo=sim(Red_Modelo,VectorEn
tradas_Etapa1Modelo,DelaysEntradasModelo_Etapa1,DelaysTargetModel
o_Etapa1);

Salida_Etapa1Modelo=cell2mat(SalidaSimulacionEntrenamientoEtapa1M
odelo);%se convierte de cell a matriz

%se muestra, en la figura 1, la salida de la simulación para el
entrenamiento
figure(1)
plot(Salida_Etapa1Modelo);
title('Salida obtenida de la simulación del entrenamiento');

%se postprocesan los datos
SalidaObtenidaEtapa1Modelo=mapminmax('reverse',Salida_Etapa1Model
o,EstructuraTarget_Etapa1Modelo);
SalidaDeseadaEtapa1Modelo = CT_prbs_suave_05(3:1:744)';

%se muestra, en la figura 2, la salida obtenida de la simulación
(azul) y la salida deseada (roja)
figure(2)
plot(SalidaDeseadaEtapa1Modelo,'r');
hold on;
plot(SalidaObtenidaEtapa1Modelo,'b');
title('Modelo: Etapa de Entrenamiento');
xlabel('Tiempo [h]')
ylabel('Carga Termica [KW]')
legend('Salida Deseada','Salida Obtenida')

%se calcula el % de la salida medida que se obtuvo de la
simulación
FitEntrenamiento=100*(1-norm(SalidaObtenidaEtapa1Modelo-
SalidaDeseadaEtapa1Modelo)/norm(SalidaDeseadaEtapa1Modelo-
mean(SalidaDeseadaEtapa1Modelo)))
```

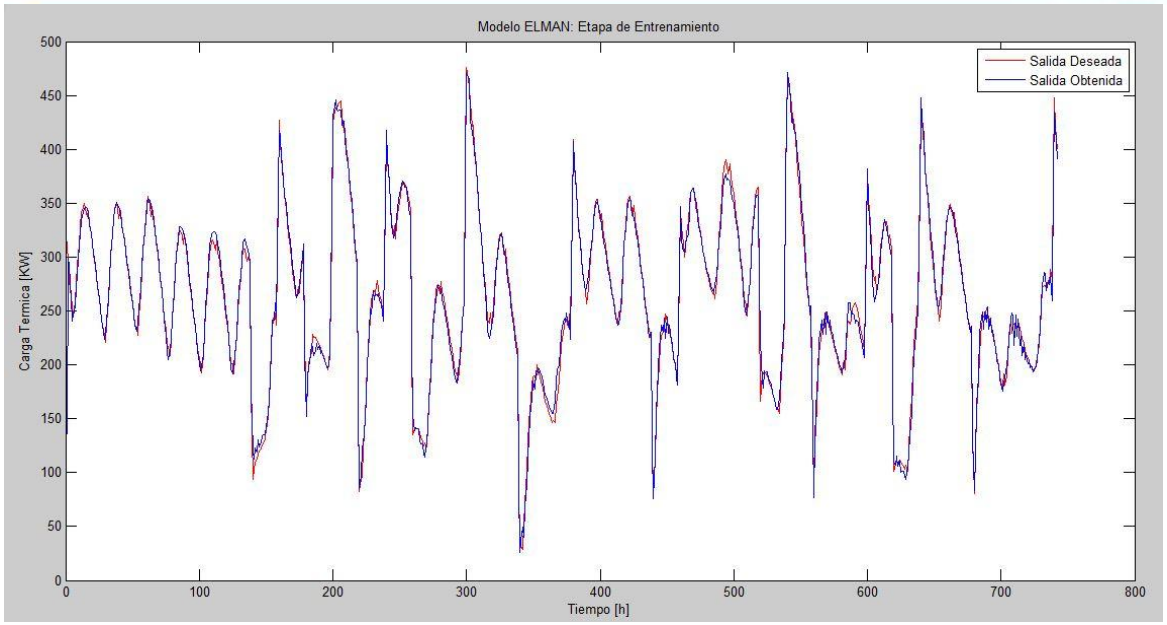
Anexo 10. Código generalizado de la implementación de los modelos en Matlab: Etapa 2.

```
%% Etapa2 Prueba
% se cargan los nuevos datos
load TambA.mat;
```

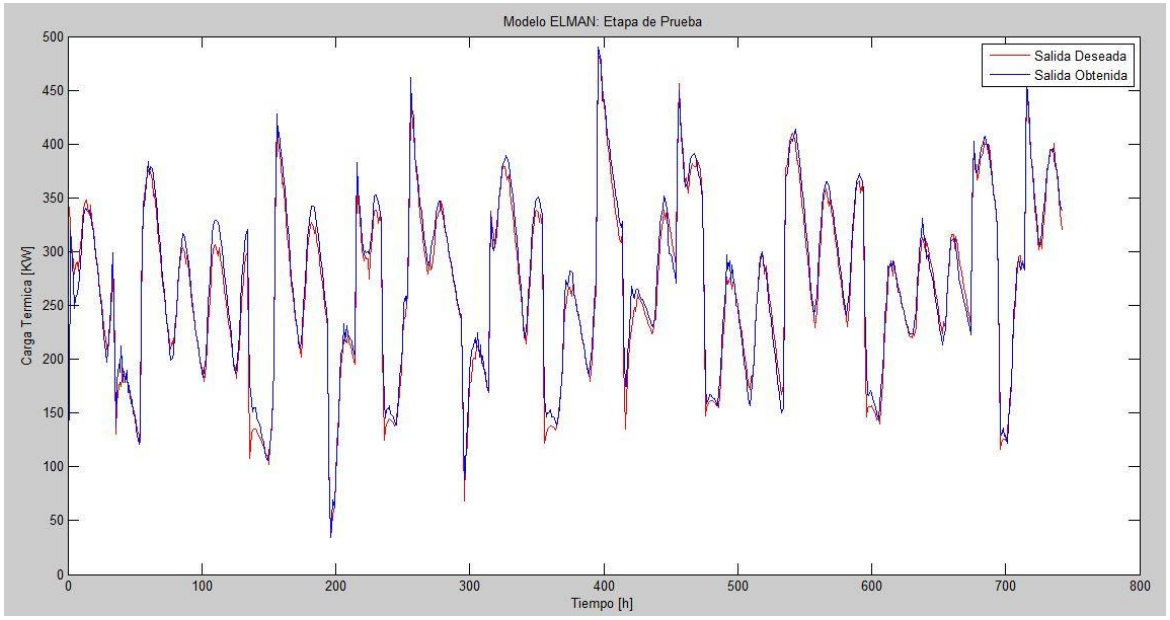
```
load GRadA.mat;
load DRadA.mat;

TemperaturaAmbienteA                =                TambA(1:1:744)';
%% Temperatura ambiente en Agosto

RadiacionGlobalA                    =                GRadA(1:1:744)';
%% Radiacion Global en Agosto

RadiacionDifusaA                    =                DRadA(1:1:744)';
%% Radiacion Difusa en Agosto

TemperaturaSetPointA          =          Tprbs_suave_05(745:1:end)';
%% Temperatura de setpoint (la que se desea)

CargaTermicaA             =            CT_prbs_suave_05(745:1:end)';
%% Carga térmica obtenida del trnsys

%   se   definen   las   nuevas   variables   y   se   realiza   el
preprocesamiento de los nuevos datos

Entradas_Etapa2Modelo=[TemperaturaAmbienteA;RadiacionGlobalA;Radi
acionDifusaA;TemperaturaSetPointA];%% se crea el nuevo vector de
entradas
[Entradas_Etapa2Modelo,estructura_entradas_Etapa2Modelo]=mapminma
x(Entradas_Etapa2Modelo);%% se preprocesan los nuevos datos

Target_Etapa2Modelo=CargaTermicaA;%%  se  crea  el  nuevo  vector
target
[Target_Etapa2Modelo,EstructuraTarget_Etapa2Modelo]=mapminmax(Tar
get_Etapa2Modelo);%% se preprocesan los nuevos datos

% se convierte de consecutivo a sequencial
Entradas_Etapa2Modelo = con2seq(Entradas_Etapa2Modelo);
Target_Etapa2Modelo = con2seq(Target_Etapa2Modelo);

% se definen los delays
delays2Modelo= (1:2);

% se simula el modelo con los nuevos datos

[VectorEntradas_Etapa2Modelo,DelaysEntradasModelo_Etapa2,DelaysTa
rgetModelo_Etapa2,VectorTarget_Etapa2Modelo]=preparets(Red_Modelo
,Entradas_Etapa2Modelo,Target_Etapa2Modelo);

SalidaSimulacionEntrenamientoEtapa2Modelo=sim(Red_Modelo,VectorEn
tradas_Etapa2Modelo,DelaysEntradasModelo_Etapa2,DelaysTargetModel
o_Etapa2);
```

```
Salida_Etapa2Modelo=cell2mat(SalidaSimulacionEntrenamientoEtapa2M
odelo);% se convierte de cell a matriz

% se muestra, en la figura 3, la salida de la simulación para la
prueba
figure(3)
plot(Salida_Etapa2Modelo);
title('Salida obtenida de la simulación de la Prueba');

% se postprocesan los datos
SalidaObtenidaEtapa2Modelo=mapminmax('reverse',Salida_Etapa2Model
o,EstructuraTarget_Etapa2Modelo);
SalidaDeseadaEtapa2Modelo = CT_prbs_suave_05(747:1:end)';

% se muestra, en la figura 4, la salida obtenida de la simulación
(azul) y la salida deseada (roja)
figure(4)
plot(SalidaDeseadaEtapa2Modelo,'r');
hold on;
plot(SalidaObtenidaEtapa2Modelo,'b');
title('Modelo: Etapa de Prueba');
xlabel('Tiempo [h]')
ylabel('Carga Termica [KW]')
legend('Salida Deseada','Salida Obtenida')

% se calcula el % de la salida medida que se obtuvo de la
simulación del modelo en la etapa de prueba
FitPrueba=100*(1-norm(SalidaObtenidaEtapa2Modelo-
SalidaDeseadaEtapa2Modelo)/norm(SalidaDeseadaEtapa2Modelo-
mean(SalidaDeseadaEtapa2Modelo)))

% se muestra la arquitectura del modelo
view(Red_Modelo)
```

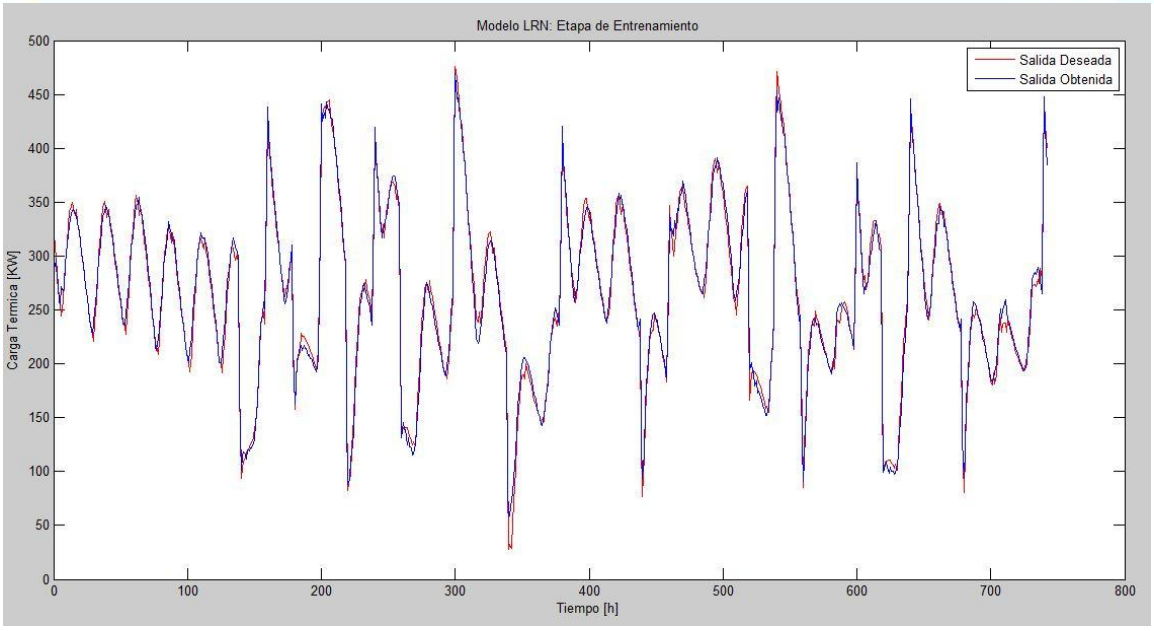
Anexo 11.1: Respuesta de variante 7 del modelo ELMAN en la Etapa 1 Entrenamiento.



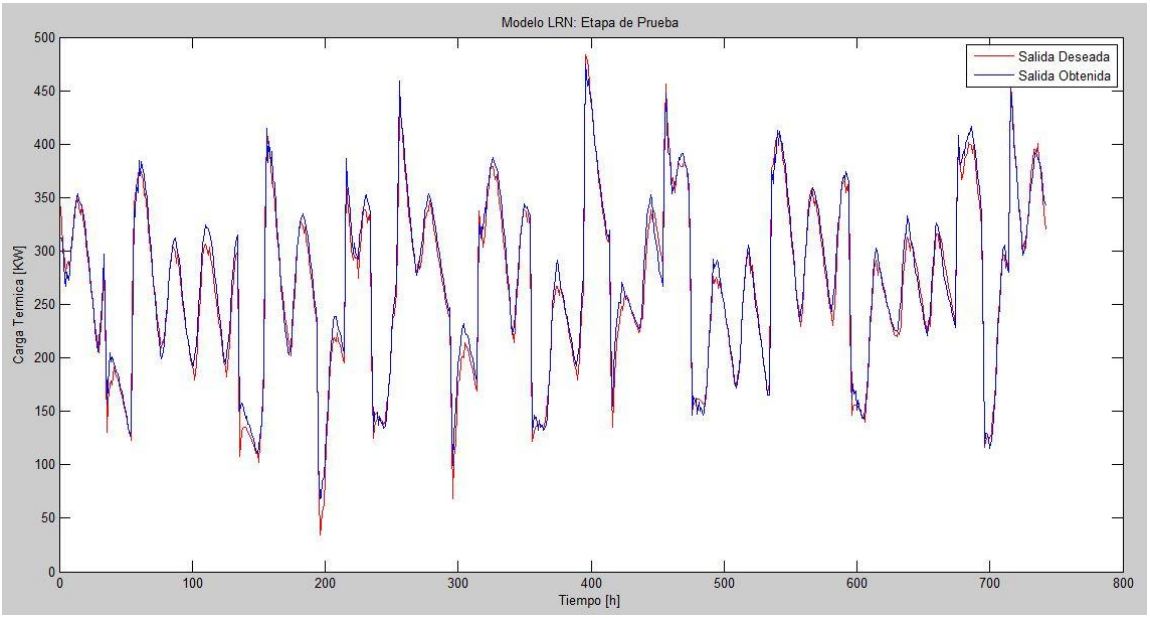
Anexo 11.2: Respuesta de variante 7 del modelo ELMAN en la Etapa 2 Prueba.



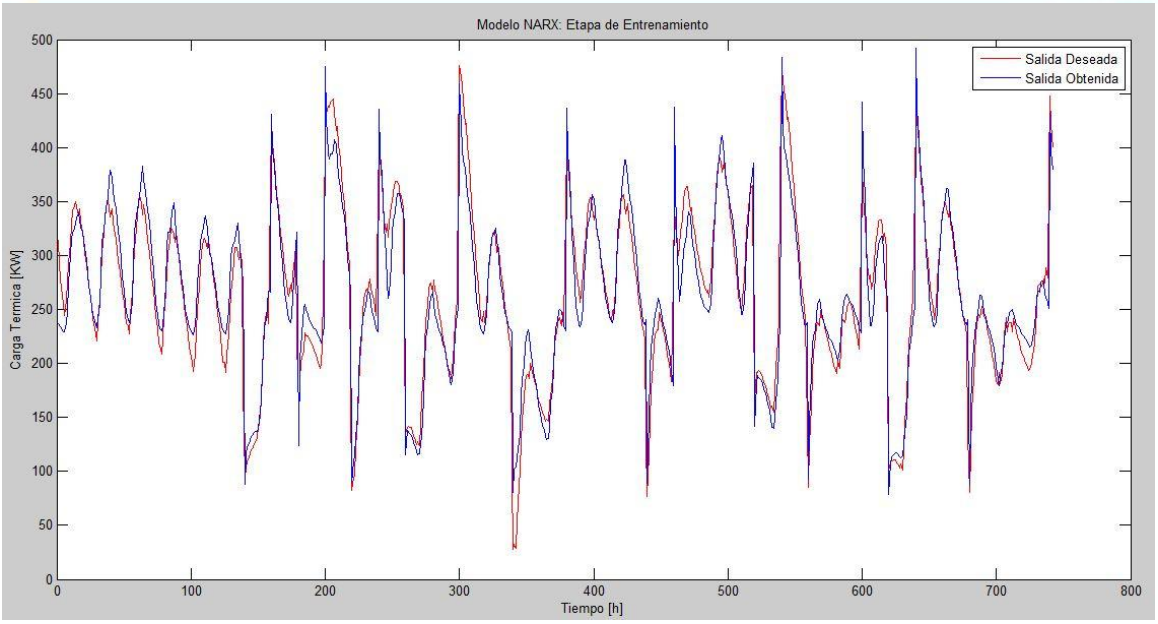
Anexo 12.1: Respuesta de variante 8 del modelo LRN en la Etapa 1 Entrenamiento.



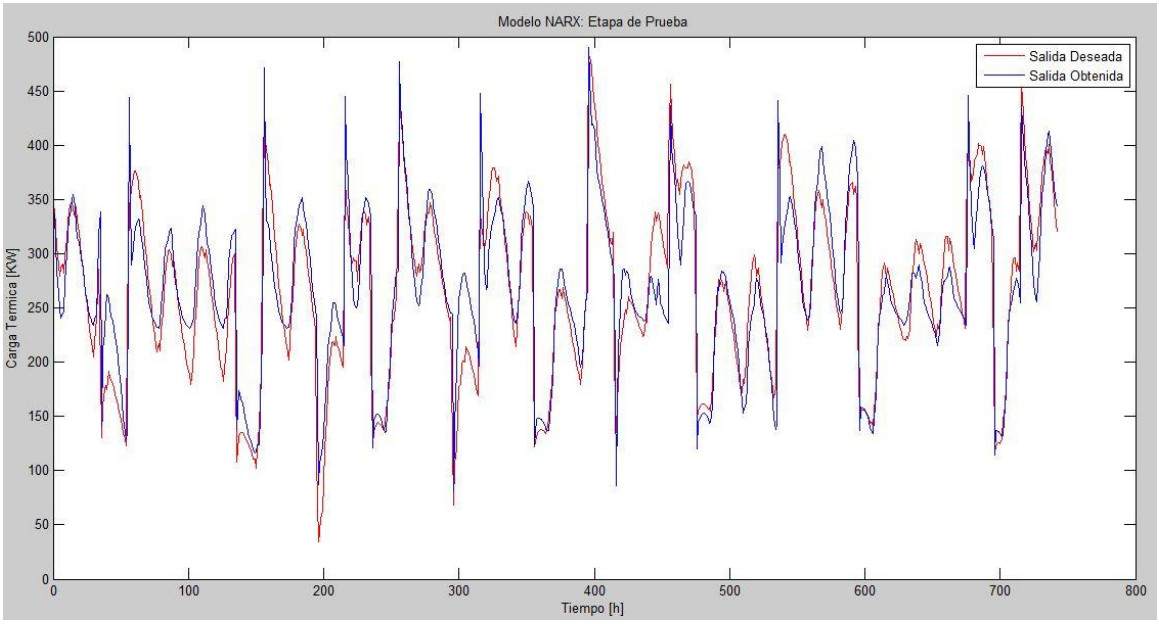
Anexo 12.2: Respuesta de variante 8 del modelo LRN en la Etapa 2 Prueba.



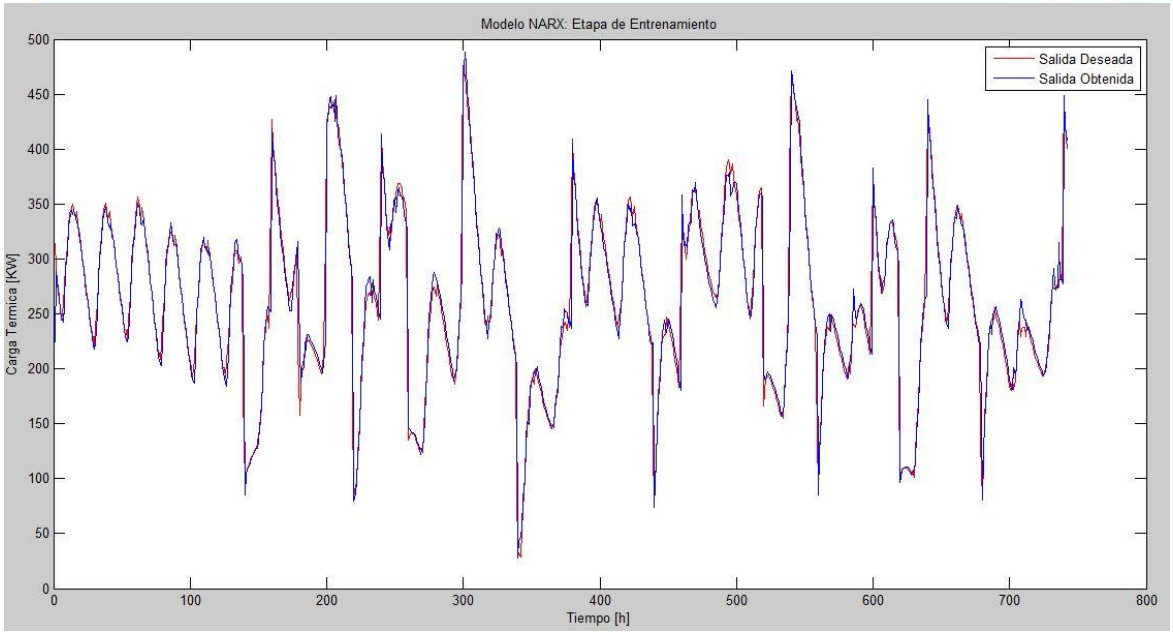
Anexo 13.1: Respuesta de variante 7 del modelo NARX Closed Loop en la Etapa 1 Entrenamiento.



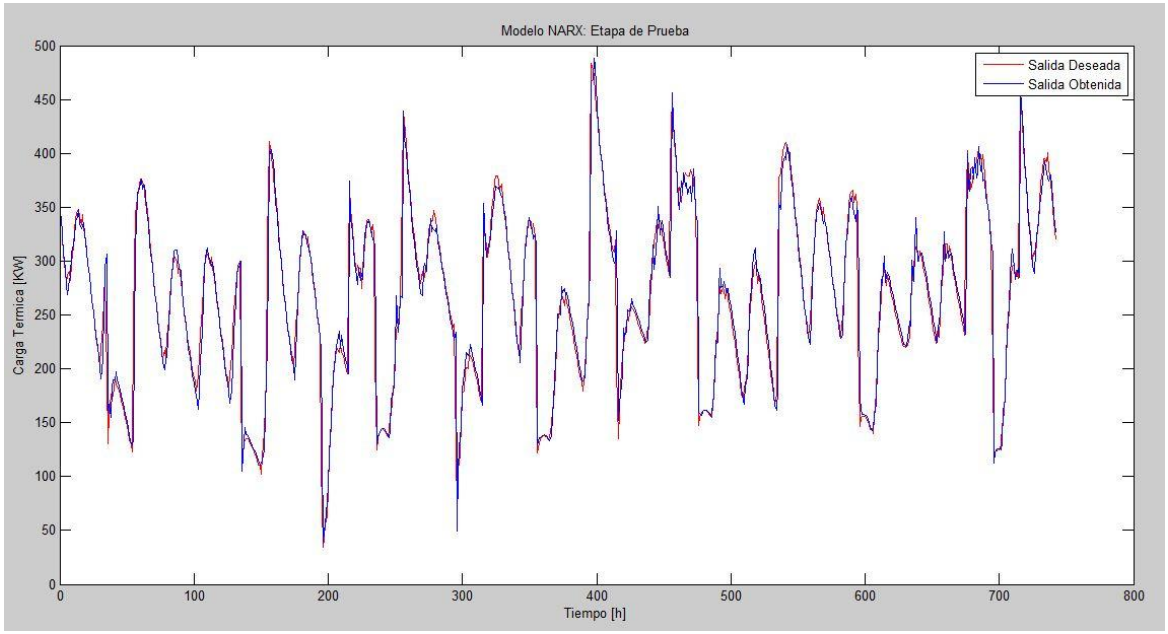
Anexo 13.2: Respuesta de variante 7 del modelo NARX Closed Loop en la Etapa 2 Prueba.



Anexo 14.1: Respuesta de variante 7 del modelo NARX Open Loop en la Etapa 1 Entrenamiento.



Anexo 14.2: Respuesta de variante 7 del modelo NARX Open Loop en la Etapa 2 Prueba.



Anexo 15. Código de la implementación, en Matlab, del modelo seleccionado como mejor en la predicción de la carga térmica del sistema de climatización del hotel Jagua.

```
clc;
clear;
```

```
tic
ticID=tic;
%% Modelo LRN con 2 capas ocultas de 10 y 4 neuronas
respectivamente, 2 delays, 0.001 error, 2000 iteraciones
%% LRN: Etapa 1: creación y entrenamiento
load TambJ.mat;
load GRadJ.mat;
load DRadJ.mat;
load Tprbs_suave_05.mat;
load CT_prbs_suave_05;
load BiosCO1;
load BiosCO2;
load BiosCS;
load PesosInputs;
load PesosCO1;
load PesosCO1_CO2;
load PesosCO2;
load PesosCO2_CS;
TemperaturaAmbienteJ = TambJ(1:1:744)';
RadiacionGlobalJ = GRadJ(1:1:744)';
RadiacionDifusaJ = DRadJ(1:1:744)';
TemperaturaSetPointJ = Tprbs_suave_05(1:1:744)';
CargaTermicaJ = CT_prbs_suave_05(1:1:744)';
Entradas_Etapa1LRN =
[TemperaturaAmbienteJ;RadiacionGlobalJ;RadiacionDifusaJ;Temperatu
raSetPointJ];
[Entradas_Etapa1LRN,estructura_entradas_Etapa1LRN] =
mapminmax(Entradas_Etapa1LRN);
Target_Etapa1LRN = CargaTermicaJ;
[Target_Etapa1LRN,EstructuraTarget_Etapa1LRN] =
mapminmax(Target_Etapa1LRN);
Entradas_Etapa1LRN = con2seq(Entradas_Etapa1LRN);
Target_Etapa1LRN = con2seq(Target_Etapa1LRN);
delaysLRN = (1:2);
%% Creación de la Red
Red_LRN = layrecnet(delaysLRN,[10,4],'traingdx');
[VectorEntradas_Etapa1LRN,DelaysEntradasLRN_Etapa1,DelaysTargetLR
N_Etapa1,VectorTarget_Etapa1LRN] =
preparets(Red_LRN,Entradas_Etapa1LRN,Target_Etapa1LRN);
%% Entrenamiento
Red_LRN.trainParam.show = 50;
Red_LRN.trainParam.epochs = 2000;
Red_LRN.trainParam.goal = 0.001;
Red_LRN.performFcn = 'mse';
Red_LRN.layers{1}.transferFcn = 'tansig';
Red_LRN.layers{2}.transferFcn = 'tansig';
Red_LRN.layers{3}.transferFcn = 'purelin';
Red_LRN.inputWeights{1,1}.learnFcn = 'learngdm';
```

```
Red_LRN.layerWeights{1,1}.learnFcn = 'learngdm';
Red_LRN.layerWeights{2,1}.learnFcn = 'learngdm';
Red_LRN.layerWeights{2,2}.learnFcn = 'learngdm';
Red_LRN.layerWeights{3,2}.learnFcn = 'learngdm';
Red_LRN =
train(Red_LRN,VectorEntradas_Etapa1LRN,VectorTarget_Etapa1LRN,Del
aysEntradasLRN_Etapa1,DelaysTargetLRN_Etapa1);
Red_LRN.IW{1,1} = PesosInputs;
Red_LRN.LW{1,1} = PesosCO1;
Red_LRN.LW{2,1} = PesosCO1_CO2;
Red_LRN.LW{2,2} = PesosCO2;
Red_LRN.LW{3,2} = PesosCO2_CS;
Red_LRN.b{1} = BiosCO1;
Red_LRN.b{2} = BiosCO2;
Red_LRN.b{3} = BiosCS;
SalidaSimulacionEntrenamientoEtapa1LRN =
sim(Red_LRN,VectorEntradas_Etapa1LRN,DelaysEntradasLRN_Etapa1,Del
aysTargetLRN_Etapa1);
Salida_Etapa1LRN=cell2mat(SalidaSimulacionEntrenamientoEtapa1LRN)
;
figure(1)
plot(Salida_Etapa1LRN);
title('Salida Obtenida durante el Entrenamiento');
SalidaObtenidaEtapa1LRN =
mapminmax('reverse',Salida_Etapa1LRN,EstructuraTarget_Etapa1LRN);
SalidaDeseadaEtapa1LRN = CT_prbs_suave_05(3:1:744)';
figure(2)
plot(SalidaDeseadaEtapa1LRN,'r');
hold on;
plot(SalidaObtenidaEtapa1LRN,'b');
title('Modelo LRN: Etapa de Entrenamiento');
xlabel('Tiempo [h]')
ylabel('Carga Termica [KW]')
legend('Salida Deseada','Salida Obtenida')
FitEntrenamiento = 100*(1 - norm(SalidaObtenidaEtapa1LRN -
SalidaDeseadaEtapa1LRN)/norm(SalidaDeseadaEtapa1LRN -
mean(SalidaDeseadaEtapa1LRN)))
%% Etapa2 Prueba
load TambA.mat;
load GRadA.mat;
load DRadA.mat;
TemperaturaAmbienteA = TambA(1:1:744)';
RadiacionGlobalA = GRadA(1:1:744)';
RadiacionDifusaA = DRadA(1:1:744)';
TemperaturaSetPointA = Tprbs_suave_05(745:1:end)';
CargaTermicaA = CT_prbs_suave_05(745:1:end)';
Entradas_Etapa2LRN =
[TemperaturaAmbienteA;RadiacionGlobalA;RadiacionDifusaA;Temperatu
```

```
raSetPointA];
[Entradas_Etapa2LRN,estructura_entradas_Etapa2LRN] =
mapminmax(Entradas_Etapa2LRN);
Target_Etapa2LRN = CargaTermicaA;
[Target_Etapa2LRN,EstructuraTarget_Etapa2LRN] =
mapminmax(Target_Etapa2LRN);
Entradas_Etapa2LRN = con2seq(Entradas_Etapa2LRN);
Target_Etapa2LRN = con2seq(Target_Etapa2LRN);
delays2LRN= (1:2);
[VectorEntradas_Etapa2LRN,DelaysEntradasLRN_Etapa2,DelaysTargetLR
N_Etapa2,VectorTarget_Etapa2LRN] =
preparets(Red_LRN,Entradas_Etapa2LRN,Target_Etapa2LRN);
SalidaSimulacionEntrenamientoEtapa2LRN =
sim(Red_LRN,VectorEntradas_Etapa2LRN,DelaysEntradasLRN_Etapa2,Del
aysTargetLRN_Etapa2);
Salida_Etapa2LRN=cell2mat(SalidaSimulacionEntrenamientoEtapa2LRN)
;
figure(3)
plot(Salida_Etapa2LRN);
title('Salida Obtenida durante la Prueba');
SalidaObtenidaEtapa2LRN =
mapminmax('reverse',Salida_Etapa2LRN,EstructuraTarget_Etapa2LRN);
SalidaDeseadaEtapa2LRN = CT_prbs_suave_05(747:1:end)';
figure(4)
plot(SalidaDeseadaEtapa2LRN,'r');
hold on;
plot(SalidaObtenidaEtapa2LRN,'b');
title('Modelo LRN: Etapa de Prueba');
xlabel('Tiempo [h]')
ylabel('Carga Termica [KW]')
legend('Salida Deseada','Salida Obtenida')
FitPrueba = 100*(1 - norm(SalidaObtenidaEtapa2LRN -
SalidaDeseadaEtapa2LRN)/norm(SalidaDeseadaEtapa2LRN -
mean(SalidaDeseadaEtapa2LRN)))
view(Red_LRN)
toc(ticID)
```