

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Instrumentación y Control Automático

Detección de fallas en sensores y actuadores para un diagnóstico de un control
climático de invernadero

TESIS

Que como parte de los requisitos para obtener el grado de
Maestro en Ciencias en Instrumentación y Control Automático

Presenta:

Ing. Edgar Ulloa Hernández

Dirigido por:

M. en C. José Rivera Mejía

M. en C. Juan José García Escalante

SINODALES

M. en C. Juan José García Escalante

Presidente

Dr. Rodrigo Castañeda Miranda

Secretario

M. en C. José Rivera Mejía

Vocal

Dr. Pedro Daniel Alaniz Lumbreras

Suplente

M. en C. Edgar Rivas Araiza

Suplente

Dr. Gilberto Herrera Ruíz

Director de la Facultad



Firma



Firma



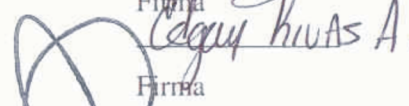
Firma



Firma



Firma



Firma



Dr. Luis Gerardo Hernández Sandoval

Director de Investigación y posgrado

Centro Universitario
Querétaro, Qro.
Abril de 2008
México

Resumen

La infraestructura de invernaderos en México, ha tenido un crecimiento espectacular, y en su implementación se necesita la participación de agricultores y empresarios. De igual manera, la industria de los invernaderos se ha enfrentado a problemas como administración y financiamiento inadecuado, procedurías deficientes y poco profesionales, y tecnología mal enfocada. Para hacer frente a dichos problemas, distintas áreas han unificado esfuerzos para erradicar los obstáculos mencionados. Actualmente, dentro del área de ingeniería se han realizado diferentes estudios para garantizar el funcionamiento correcto de los invernaderos. Donde los sistemas de detección de fallas y diagnóstico (SDDF), han sido áreas activas de la investigación. Estos, asumen un rol importante dentro de los sistemas de control. El objetivo principal del proceso de detección de fallas, es monitorear las condiciones de operación de los elementos involucrados de los sistemas continuos, para asegurar y validar la operación, bajo cualquier condición. Evitando así, la pérdida del producto e inversión. Y favoreciendo al óptimo desempeño del sistema. El siguiente trabajo, está enfocado al diseño e implementación de un sistema de detección de fallas usando inteligencia artificial para implementarse en sistemas de control de invernadero. El sistema de detección de fallas realizado, tiene la capacidad de supervisar las lecturas de los sensores y acciones de los actuadores involucrados. Además, de ser sencillo y de fácil implementación, ya sea por software o hardware.

Palabras clave: Detección de fallas, invernadero, red neuronal artificial, sensores y actuadores.

Summary

The increase of greenhouses in Mexico has an incredible growth, and for its implementation, it needs the cooperation of farmers and employers. Likewise, the greenhouses have different problems like inadequate administration and finance, poor services and bad technology applied. For solving these problems, some areas had worked together for resolve these obstacles. Now a days, the engineering areas, have developed many researches about fault detection and diagnosis systems (FDDS), for guarantee the optimal work of continuous systems. These systems, monitors the operation conditions of the involucrate elements, for assure and validate its operation. Avoiding, the lost of the whole product and hence the investment and gaining the optimal performance. This work is focused at the design and implementation of a fault detection system using artificial intelligence, for be implemented in greenhouse control systems. The FDDS implemented, has the ability to supervise the sensors readings and actions of the actuators, detecting any malfunction of both elements. The FDDS developed, results to be simple and easy for be implemented via software or hardware.

Keywords: Failure detection, greenhouse, artificial neural networks, sensors and actuators.

Dedicatorias

A mis padres Magdiel Ulloa Espinosa y Ma. Guadalupe Hernández Martínez, por enseñarme el camino del estudio, dedicación y responsabilidad.

A mis hermanos Adrián y Ma. De los Ángeles, por alegrar mi vida.

A mi novia Nadia Shalyn, por demostrarme cariño, ánimo, amor, comprensión y lo hermoso de la vida.

A mis asesores y amigos Juan José García Escalante y José Rivera Mejía.

A mis amigos Raúl Omar, José Ernesto, Juan Luís y Arturo.

Agradecimientos

A los maestros de la Facultad de Ingeniería de Licenciatura y la División de Estudios de Posgrado, por crearme las bases de un conocimiento sólido y por la ayuda brindada para la realización del presente trabajo.

A mis asesores y amigos Juan José García Escalante y José Rivera Mejía, por sus consejos y orientaciones que apoyaron al desarrollo de esta investigación.

A los integrantes del cubil de biotróica, por su apoyo, armonía y sinergia que privó en todo momento en el trabajo.

Finalmente, a CONACYT por el apoyo brindado durante mi estancia en la Facultad de Ingeniería, la cual permitió dedicarme de tiempo completo a la realización de mis estudios de posgrado y de esta manera lograr el cumplimiento del objetivo principal del convenio establecido (No. 291282).

Índice general

1. Introducción	1
1.1. Antecedentes y justificación	3
1.2. Fundamentación Teórica	6
1.3. Objetivos e hipótesis del trabajo	8
1.4. Planteamiento general	9
2. Estado del arte	11
2.1. Métodos cuantitativos	11
2.1.1. Cambio de parámetros en un modelo	14
2.1.2. Cambios estructurales	16
2.1.3. Falla de sensores y actuadores	16
2.1.4. Clasificación de algoritmos de diagnóstico	17
2.1.5. Enfoque de los modelos cuantitativos	19
2.2. Modelos cualitativos	21
2.3. Métodos basados en historiales de datos	23
2.3.1. Extracción de características cualitativas	23
2.4. Estado del arte de las redes neuronales	28
3. Metodología	37
3.1. Características de una red neuronal artificial	37

3.1.1.	Notación	39
3.1.2.	Funciones de transferencia	40
3.1.3.	Topología de una red neuronal	42
3.2.	Red neuronal backpropagation	47
3.2.1.	Antecedentes	47
3.2.2.	Estructura de la red	52
3.2.3.	Regla de aprendizaje	52
3.3.	Algoritmo de Levenberg - Marquardt	62
3.4.	Algoritmo Levenberg-Marquardt resumido	65
3.5.	Diseño de la red neuronal artificial	66
3.5.1.	Parámetros de falla y no falla	66
3.5.2.	Diseño del sistema	68
3.6.	Sensor transductor de corriente	76
3.6.1.	Acondicionamiento de señal y montaje del sensor	76
3.6.2.	Caracterización	77
3.7.	Diseño y construcción de tarjeta de adquisición de datos con MCU (<i>Micro Controller Unit</i>)	80
3.8.	Implementación e interfaz del sistema de detección de fallas y diagnóstico en el MCU 18F4620	82
4.	Análisis de Resultados	87
4.1.	Ecuaciones y validación por software del sistema de detección de fallas y diagnóstico	87
4.1.1.	RNA Temperatura interna	88
4.1.2.	RNA Temperatura externa	90
4.1.3.	RNA Humedad relativa interna	91
4.1.4.	RNA Humedad relativa externa	92
4.1.5.	RNA líneas de alimentación del motor trifásico	94

4.2. Implementación del sistema de detección de fallas y diagnóstico con el sistema TUNA SCCII	97
4.3. Prueba de respuesta de RNA en MCU	99
5. Conclusiones	104
A. Descripción de programa en MATLAB para RNA de sensores	112
B. Descripción de en programa MATLAB para RNA de líneas	121
C. Caracterización de transductor de corriente en MATLAB	124
D. Programa en BUILDER con APRO	129
E. Programa de RNA en MCU 18F4620	138

Índice de figuras

1.1. Neurona	7
1.2. Red neuronal	7
1.3. Sistema general	10
2.1. Diagrama general de diagnóstico	15
2.2. Clasificación de algoritmos de diagnóstico	18
2.3. Esquema general para el uso de la redundancia analítica	20
2.4. Formas del conocimiento cualitativo	22
2.5. Clasificación de los métodos basados en historiales de datos	24
3.1. De la neurona biológica a la neurona artificial	37
3.2. Proceso de una red neuronal	38
3.3. Neurona de una sola entrada	40
3.4. Función de transferencia hardlim	41
3.5. Función de transferencia hardlims	41
3.6. Función de transferencia lineal	42
3.7. Función de transferencia sigmoideal	44
3.8. Neurona con multiples entradas	44
3.9. Neurona con multiples entradas, notación abreviada	45
3.10. Capa de S neuronas	46
3.11. Capa de S neuronas con notación abreviada	48

3.12. Red neuronal de 3 capas	48
3.13. Red de tres capas con notación abreviada	48
3.14. Clasificación de las RNA	54
3.15. Disposición de una red sencilla de 3 capas	54
3.16. Entrenamiento Levenberg-Marquardt	66
3.17. RNA propuesta para sensores	73
3.18. RNA propuesta para líneas de alimentación	75
3.19. Diseño del circuito rectificador de precisión	77
3.20. Circuito rectificador de precisión	78
3.21. Montaje de los sensores transductores de corriente	78
3.22. Caracterización del sensor transductor de corriente	79
3.23. Curva de caracterización por ajuste polinomial	79
3.24. Tarjeta de adquisición con MCU 18F4620	82
3.25. Diseño de comunicación RS-232 con MCU	83
3.26. Diagrama flujo del código generado	84
3.27. Interfaz entre microprocesador y computadora	86
3.28. Diagrama general de implementación del SDFD	86
4.1. Respuesta de RNA para la variable temperatura interna	89
4.2. Respuesta de RNA para la variable temperatura externa	91
4.3. Respuesta de RNA para la variable humedad relativa interna	93
4.4. Respuesta de RNA para la variable humedad relativa externa	95
4.5. Respuesta de RNA para una línea de alimentación del motor trifásico	97
4.6. Acoplamiento del sistema de detección de fallas al sistema TUNA SCCII	99
4.7. Detectando fallas y diagnosticando	101
4.8. Prueba de RNA en MCU de la temperatura interna	102
4.9. Prueba de RNA en MCU de la temperatura externa	102
4.10. Prueba de RNA en MCU de las líneas de alimentación	103

Índice de cuadros

3.1. Funciones de transferencia	43
3.2. Vector de entrenamiento de Temperatura Interna	70
3.3. Vector de entrenamiento de Temperatura Externa	71
3.4. Vector de entrenamiento de Humedad Relativa Interna	72
3.5. Vector de entrenamiento de Humedad Relativa Externa	73
3.6. Vector de entrenamiento de las líneas de alimentación	74
3.7. Funciones de activación	74
3.8. Características de MCU 18F4620	81
4.1. Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de temperatura interna	90
4.2. Salidas en código binario de RNA de temperatura interna	90
4.3. Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de temperatura externa	92
4.4. Salidas en código binario de RNA de temperatura externa	92
4.5. Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de humedad relativa interna	94
4.6. Salidas en código binario de RNA de humedad relativa interna	94
4.7. Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de humedad relativa externa	96

4.8. Salidas en código binario de RNA de humedad relativa externa	96
4.9. Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de líneas de alimentación	98
4.10. Salidas en código binario de RNA de la línea de alimentación del motor trifásico .	98

Capítulo 1

Introducción

La infraestructura de invernaderos en México ha tenido un crecimiento espectacular y en su implementación, se necesita la participación de agricultores y empresarios, los cuales están convencidos de las ventajas de este tipo de producción como una alternativa de inversión.

En 1990, había aproximadamente 50 hectáreas con algún tipo de producción de vegetales bajo invernadero; para 1999, la cifra era de 600 hectáreas; en el 2001, se elevó a 950 y ahora, se suman alrededor de 2,200 hectáreas -con una diversificación de cultivos-.

La industria de los invernaderos, ha enfrentado problemas como: tecnología mal enfocada, proveeduría deficiente y poco profesional, administración y financiamiento inadecuado y poco competitivo, además de falta de personal capacitado.[Ciencia, 00]

La meta principal de todo invernadero, es optimizar la cantidad así como la calidad de la producción. La optimización y la calidad, son alcanzadas utilizando sistemas automatizados para el control del ambiente interno llamado control climático; para dar así a la planta una condición óptima de crecimiento. El óptimo desempeño del control climático de invernadero, se refleja en la producción y en la obtención de un cultivo libre de enfermedades, para garantizar así una calidad

extraordinaria en el producto, cumpliendo tiempos de producción y en el abaratamiento de los costos.

A través de los años, el control climático de invernaderos se ha automatizado, y para su realización se deben de incluir varios sensores dentro y fuera del invernadero. Y entonces, a partir de las lecturas de los sensores (temperatura, humedad relativa, radiación, velocidad y dirección de viento, etc.) es como se realizan las acciones de control, pero dichas señales no son analizadas previamente. Es por eso que las desviaciones y/o fallas en sensores y actuadores (motores trifásicos con acoplamiento mecánico que realizan la función de ventanas) afectan la operación del invernadero y al producto. Un sistema de detección de falla e identificación incluido en el control climático minimiza el daño y realiza una acción de control preventiva [Linker et al., 00].

Actualmente, el proceso de detección de fallas y diagnóstico (*FDD, Fault Detection and Diagnosis*) ha sido un área activa de la investigación por parte de la ingeniería, y asume un rol importante dentro de los sistemas de control. El objetivo principal del proceso de detección de fallas y diagnóstico, el cual puede estar inmerso como parte de las actividades de mantenimiento en fabricas o procesos, es monitorear las condiciones de operación de máquinas y/o sistemas continuos para asegurar que se está operando en condiciones óptimas [Tan et al., 07].

Por consiguiente, si se implementa un sistema detector de fallas para realizar una identificación de la misma, se tendrá una producción constante aunado con un incremento de productividad y ahorro de costo de operación.

Por otra parte, la Universidad Autónoma de Querétaro, cuenta con invernaderos para la generación de cultivo de alta calidad, basándose en un sistema de control climático inteligente [Castañeda et al., 05]. Donde las lecturas de los diferentes sensores, en conjunto con el sistema de control automático TUNA SCCII 5.0 (Tecnología Universitaria en Automatización de Sistemas

de Control Climático Inteligentes para Invernaderos), generan la acción de control sobre los actuadores, para la formación del ambiente deseado.

Sin embargo, si se registran lecturas anómalas en los sensores o la operación de control de los actuadores no son las adecuadas, el ambiente interno en el invernadero se tornará negativo e indeseable. Es decir, se tendrán daños sobre la producción (como ejemplo se tiene un crecimiento inesperado y trunco del producto o una pérdida total debido a una alteración drástica del clima interno), afectando directamente a la inversión. En la mayoría de los casos, todo efecto negativo se debe a una decisión de control errónea por adquirir lecturas anómalas de los sensores, o debido a una operación errónea de los actuadores.

El presente trabajo, propone un método de detección de fallas en sensores y actuadores para generar un diagnóstico de un control climático de invernadero, basandose en la herramienta matemática de las redes neuronales artificiales. El sistema de detección de fallas y diagnóstico, primeramente será validado por software. Posteriormente, se realizarán pruebas para su validación. Finalmente, se realizará la implementación en hardware a partir de los resultados obtenidos por la validación de software y se implementará en el sistema de control.

1.1. Antecedentes y justificación

Hoy en día es ampliamente reconocido que las propiedades de los sensores y actuadores son una pieza clave en el funcionamiento de los lazos de control. Un funcionamiento seguro y fiable es demandado en la mayor parte de los procesos y sistemas, mas allá de aquellos que normalmente son aceptados como sistemas críticos en cuanto a seguridad (como lo pueden ser plantas químicas, plantas nucleares, etc.)[Pinon et al., 95]. La detección incipiente de fallas en procesos y sistemas industriales, puede ayudar a reducir los paros y la incidencia de efectos negativos. Una parte muy

importante de cualquier sistema o proceso automático, son los sensores, ya que mediante ellos se puede conocer el estado del sistema; por lo tanto, en un sistema o proceso en el cual se pretendan implantar estrategias para la detección y diagnóstico de fallas, habrá que evaluarse el estado de funcionalidad.

Tradicionalmente los mecanismos para la detección de fallas son basados en el concepto de redundancia y más específicamente en el uso de redundancia física, es decir, en el uso de elementos repetidos en el sistema. Estos elementos nos permiten, por medio de comparaciones del funcionamiento, tomar decisiones sobre la presencia de fallas y sobre posibles acciones correctivas. Los métodos basados en redundancia física son muy confiables y permiten una rápida corrección de posibles fallas. Sin embargo, debe tenerse claro que la utilización de elementos repetidos en un sistema no puede ser llevada a la práctica siempre. Bajo ciertas condiciones, como por ejemplo, el tamaño o el peso de los dispositivos, el uso de este tipo de redundancia está limitado. Su implementación ocasiona la elevación del costo de la instrumentación total, debido al incremento en el número de elementos utilizados. Lo cual conlleva a incrementos en la implementación física y el mantenimiento de los sistemas de control.

Técnicas alternativas tuvieron que ser desarrolladas para enfrentar los problemas antes descritos. A principios de la década de los 70 fue introducido el concepto de redundancia analítica, el cual complementa los resultados disponibles de los métodos basados en redundancia física. La redundancia analítica está basada en el conocimiento del modelo matemático del sistema en cuestión, así como de las señales de entrada y de salida del sistema. Diferentes métodos basados en la redundancia analítica, fueron propuestos y sujetos a fuerte investigación en los pasados 25 años, como se puede corroborar en publicaciones [Alcorta et al., 97], [Frank et al., 90], [Isermann et al., 84], [Chow et al., 84]. El producto de este trabajo de investigación trajo como resultado el aumento de confianza en estos métodos y, consecuentemente, el que hoy en día se apliquen a sistemas reales.

Recientemente, se ha planteado la idea de diseño de sistemas de diagnóstico de fallas jerárquicas con respecto a los componentes en grandes sistemas, y dentro de ese objetivo, se plantean los métodos de diagnóstico local para sensores y actuadores.

El método propuesto está basado en algoritmos de identificación implementados en software y/o hardware. Su principal característica es la facilidad de desarrollo e implementación, utilizando diversas técnicas como lo son: la estimación paramétrica, observadores y espacio de paridad, modelos cuantitativos y cualitativos, modelos borrosos, inteligencia artificial, entre otros.

Se plantea el uso de las redes neuronales artificiales (RNA) para la realización de dichas tareas de detección y diagnóstico de fallas, en base a la capacidad de proceso de los sensores y a sus capacidades de comunicación.

Las redes neuronales artificiales, han sido ampliamente utilizadas en distintas aplicaciones para análisis de datos y extracción de características o patrones, las cuales han demostrado capacidad y versatilidad para el reconocimiento de patrones, a partir de un entrenamiento previo. En este trabajo, se propone el uso de las redes neuronales artificiales para la detección y el diagnóstico de fallas. Una vez que la red es entrenada para una tarea particular, el funcionamiento consiste en la propagación de datos a través del mapa de la red, haciendo posible así un autodiagnóstico y monitorización en tiempo real.

El criterio planteado anteriormente, es de suma importancia, ya que se desea proteger el producto o cultivo y una inversión inicial, partiendo del análisis de las lecturas de los sensores y de las acciones de control realizadas por los actuadores. Es por eso, que el presente trabajo consiste en analizar cada lectura de los sensores así como del funcionamiento adecuado de cada actuador, para entonces aplicar el sistema capaz de detectar fallas en el funcionamiento de cada elemento y así finalmente, realizar el diagnóstico de la posible falla.

El trabajo propone un método de la redundancia analítica, la cual a través de la herramienta de la inteligencia artificial, se puede diseñar e implementar una RNA que detecte las posibles fallas en sensores y actuadores para garantizar el buen desempeño del sistema del control climático. Y entonces así, evitar acciones defectuosas en el control y por consecuencia tener la pérdida del producto y en algunos casos la pérdida de los instrumentos del control y del sistema.

1.2. Fundamentación Teórica

El concepto de detección y diagnóstico de fallas, se refiere tanto a la detección como a la localización de ésta, es decir, además de poder determinar si una falla está presente, se requiere saber qué componente es el que la está ocasionando. Esto último, es indispensable en la mayor parte de los casos para determinar una acción correctiva. El presente trabajo estudia el comportamiento de los elementos (sensores y actuadores) que determinan el control climático del invernadero. Los elementos involucrados en el control son básicamente los sensores y actuadores.

Las fallas de sensores, se clasifican en diferentes tipos, como lo son: desconexión del sensor, corto circuito entre terminales del sensor, perturbación electromagnética hacia la señal, desconexión de alimentación hacia el sensor, etc. Por otro lado, el invernadero para realizar la interacción entre el microclima interno con el externo debe de contar con ventanas. Éstas, son accionadas por motores trifásicos con un acoplamiento mecánico y sus fallas posibles para este tipo de sistema son: desacoplamiento mecánico, exceso de carga, caída de fase en cualquiera de las tres líneas del motor trifásico, falta de alimentación hacia los motores trifásicos, armónicos, entre otras.

La idea propuesta basada en un método de redes neuronales, consiste en encontrar una arquitectura de RNA para la identificación de fallas los sensores y los actuadores del invernadero. Las

RNA, son modelos matemáticos simplificados de las neuronas del cerebro humano y contienen generalmente tres capas de elementos, llamadas neuronas, ver figuras 1.1 y 1.2 los cuales están altamente conectados.

De manera simplificada, cada neurona podría definirse como una suma ponderada de entradas que son pasadas a través de una función no lineal, llamada función de activación.

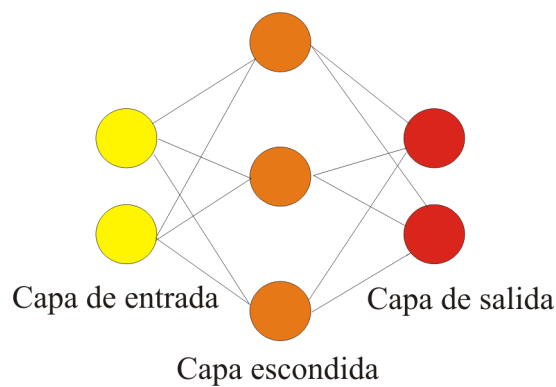


Figura 1.1: Neurona

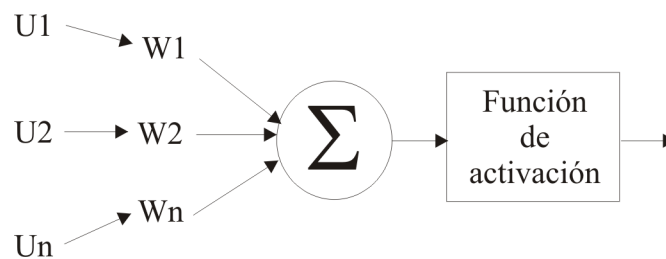


Figura 1.2: Red neuronal

La arquitectura o el modelo del sistema, se obtiene cambiando la ponderación de las entradas de cada red, hasta que su comportamiento de sea semejante al deseado, es decir, entrenar el sistema para un comportamiento deseado. El ajuste de parámetros se denomina aprendizaje de la red. Una vez con el modelo la aplicación a detección de fallas es semejante a cualquier otro tipo método

para la detección de fallas y diagnóstico.

1.3. Objetivos e hipótesis del trabajo

El objetivo general del trabajo, es la detección de fallas en los sensores, así como de los actuadores del sistema de control climático de invernadero, para generar un diagnóstico en la existencia de una falla.

La hipótesis general del trabajo para la obtención de un control climático de invernadero eficiente y evitar las oscilaciones anómalas, es implementar un sistema detector de fallas en sensores y actuadores para la generación de un diagnóstico.

Así mismo, se tienen los siguientes objetivos e hipótesis específicas para la realización dicho sistema.

- Objetivo: Desarrollar una arquitectura de red neuronal artificial genérica de detección de fallas y diagnóstico para los sensores y actuadores.
- Hipótesis: La red neuronal tiene la capacidad de detectar una fluctuación anómala en las mediciones de variables físicas del sistema.
- Objetivo: Validar la arquitecturas propuestas por medio de software para su futura implementación en hardware.
- Hipótesis: La simulación del sistema de detección de fallas es un método eficaz para validar el algoritmo.
- Objetivo: Implementar las redes neuronales validadas en hardware.

- Hipótesis: El desempeño del algoritmo implementado en hardware es mucho más eficiente que la implementación en software del mismo algoritmo.

Finalmente, el sistema tendrá las siguientes características:

- Detectar de manera eficiente las fallas en los sensores y actuadores.
- Preciso, de bajo consumo computacional y de bajo costo.
- Diseño genérico, de fácil implementación y escalable para otros sistemas.

1.4. Planteamiento general

La figura 1.3 muestra el control integral general para el óptimo desempeño del invernadero. Dentro de las subdivisiones mostradas, se encuentra el sistema de ventilación natural. Los sensores involucrados dentro del sistema de ventilación son: temperaturas (interna y externa), humedades relativas -H.R.- (interna y externa), radiación. Y los actuadores involucrados son: motores trifásicos con un acoplamiento mecánico para el movimiento lateral de las ventanas. El planteamiento general es la detección de fallas del sistema de ventilación natural tanto de los sensores como de los actuadores involucrados para su óptimo desempeño.

Finalmente, se presenta de manera general el planteamiento para la realización de este trabajo. Primeramente, se generarán las lecturas anómalas (fallas inducidas) para generar un banco de datos, los cuales se le presentarán para el aprendizaje de una arquitectura de red neuronal artificial. Seguido del primer paso, se diseñará y validará una arquitectura de red neuronal, así como el algoritmo de aprendizaje para detección y diagnóstico y como último paso, desarrollar e implementar la red neuronal de detección para poder ser implementada al sistema de control climático de invernadero.



Figura 1.3: Sistema general

Capítulo 2

Estado del arte

2.1. Métodos cuantitativos

La exactitud en los procesos de control, ha avanzado rápidamente en los últimas tres décadas por el arribo del control computarizado en procesos complejos. Acciones de control de bajo nivel como lo puede ser un control de apertura y cerradura de una válvula, llamado también control de regulación, puede ser desempeñado por el ser humano, pero ahora estas acciones son realizadas automáticamente con la ayuda de las computadoras teniendo gran éxito. Con el progreso del control distribuido y de los modelos predictivos en los sistemas de control, los beneficios a varios segmentos industriales como lo son el químico, petroquímico, cementeras, fundidoras de acero, industrias generadoras de energía han sido enormes. Sin embargo, algunas tareas importantes de control todavía son operadas manualmente por humanos, de donde se generan comúnmente los eventos anormales. Esto implica tiempo para la detección de un evento anormal, su diagnóstico para evaluar el origen del comportamiento y entonces tener así una apropiada decisión de control y/o acciones para obtener nuevamente el proceso normal, seguro y con una operación óptima.

Sin embargo, la confianza total brindada a los operadores para hacer frente a los eventos anormales o de emergencia, resulta cada vez más difícil debido a diferentes factores. Es difícil debido

a la gran variedad de posibles diagnósticos, los cuales son rodeados por una diversidad enorme de anomalías. Como lo pueden ser: fallas en unidades del proceso, degradación de la unidad del proceso, desviaciones en los parámetros y viceversa. Y es mucho mayor el grado de complejidad si a esto se le suma que algunos procesos y plantas son modernas y complejas. Citando un ejemplo, en un proceso de grandes proporciones o volúmenes de producción, se pueden encontrar mediciones de hasta 1500 variables, las cuales son observadas a cada segundo [Bailey et al., 84]. Igualmente, la tarea del diagnóstico de fallas es delicado por el hecho de que las mediciones en los procesos son insuficientes, incompletos y/o no confiables, debido nuevamente a las diferentes causas como lo son: fallas en los sensores o desviaciones.

Dadas las condiciones anteriormente mencionadas, no debe de sorprendernos que algunos de los operadores humanos, tengan la costumbre de realizar decisiones incorrectas y tomar acciones correctivas, las cuales resultan ser peores. Las estadísticas muestran que alrededor del 70 % de los accidentes dentro de la industria, son causados por los errores humanos. Estos eventos anormales repercuten en la cuestión económica y de seguridad, y a pesar del avance del control computarizado en plantas químicas, no se puede garantizar un buen desempeño. Recientemente se pueden citar, dos de los peores accidentes de plantas químicas que ocurrieron a no más de 10 años, el accidente de la planta de la Union Carbide Bhopal en la India y el accidente del Piper Alpha de la planta Occidental Petroleum Corporation [Lees et al., 96], los cuales resultan ser un problema creciente. Otro incidente lamentable es el de la explosión de una refinería en Kuwait llamado Kuwait Petrochemical's Mina Al-Ahmedi en junio del 2000, con daños materiales de aproximadamente 100 millones de dólares.

No obstante, el análisis estadístico demuestra que son mucho menos frecuentes las catástrofes para una planta química, en cambio los accidentes menores son más comunes, y ocurren día con día como lo son las lesiones, y las enfermedades, las cuales demandan un costo anual de billones de dólares ([Bureau et al., 98], [National, 90]). Es estimado que solamente en las industrias

petroquímicas de los Estados Unidos, pierden anualmente 20 billones de dólares debido a que no cuentan con un sistema de detección y diagnóstico. Este costo se puede aumentar, si se compara con la industrias farmacéuticas, industrias de elaboración de químicos especiales, de energía y otras.

Es por ello que éste es el siguiente gran reto para los ingenieros de control de procesos. En el pasado, los ingenieros mostraban cómo se podía automatizar una válvula reguladora utilizando computadoras para ser sustituido por las manos de los operadores humanos. Esto ha progresado en el aspecto de tener producción de calidad y consistencia, procesos seguros y procesos eficientes. Ahora, el reto es automatizar la detección de fallas y diagnóstico usando sistemas inteligentes de control, para proveer asistencia a los operadores humanos en las áreas donde se demande mayor atención.

La automatización de detección de fallas y diagnóstico es el primer gran paso de un mantenimiento preventivo. Debido a la gran variedad de fallas de los procesos, así como la dificultad de tener una solución en tiempo real, varias técnicas computacionales han sido desarrolladas al pasar de los años. Estas técnicas cubren una gran variedad así como de tener una diversidad como son las fallas de árboles (*Fault Trees*) y dígrafos (*Digraphs*), aproximaciones analíticas, y sistemas basados en conocimiento y redes neuronales. Desde una perspectiva de modelos matemáticos, existen métodos que requieren modelos precisos del proceso, modelos semi-cuantitativos o modelos cualitativos. Por otra parte, existen métodos que no necesariamente se debe de asumir un modelo, sino solamente información histórica. Obviamente, también existen diferentes técnicas para realizar diagnósticos. Como lo pueden ser ciertas metodologías y alternativas que por lo regular resultan difíciles de entender para quien no es especialista en los temas.

La idea básica de este capítulo es proveer un estudio comparativo y sistemático de varios métodos para el diagnóstico visto de diferentes perspectivas. Se realiza la clasificación de los métodos

en dos categorías. Los métodos basados en modelos cuantitativos y los métodos basados de datos históricos.

A manera de introducción, se dará una serie de definiciones y nomenclatura la cual es usada comúnmente en el área de fallas en procesos y diagnóstico. El término falla es definido generalmente como una desviación de un rango de operación de una medición de una variable o de un parámetro calculado de un proceso [Himmelblau et al., 78]. Esto define a la falla como un proceso anormal o síntoma, como lo puede ser una alta temperatura en un reactor o una baja productividad en un proceso, entre otros ejemplos. La causa de esta anomalía, como lo puede ser la falla de una bomba de un sistema de refrigeración o de un controlador, es llamada evento básico o raíz de la causa. El evento básico es también conocido con el nombre de mal funcionamiento o falla. Desde la visualización de la tarea de diagnóstico como un problema de clasificación, el sistema de diagnóstico es también referido a un clasificador de diagnóstico. La figura 2.1 representa los componentes de una manera en general. La figura muestra un sistema de control en donde se indican las diferentes fuentes de donde se genera una posible falla. De manera general, se deben de considerar tres clases de fallas o malfuncionamiento definidas a continuación.

2.1.1. Cambio de parámetros en un modelo

En cualquier modelado, existen procesos por debajo del nivel superior de un modelo. Estos procesos, los cuales no son modelados, son típicamente acomodados como parámetros y en éstos se incluyen iteraciones que están alrededor del límite del sistema. Las fallas en estos parámetros aparecen cuando existen perturbaciones externas. Un ejemplo de falla sería el cambio de concentración de un reactante de valor constante en un reactor de alimentación. Aquí, la concentración es una variable exógena donde su comportamiento no va con la dinámica del proceso.

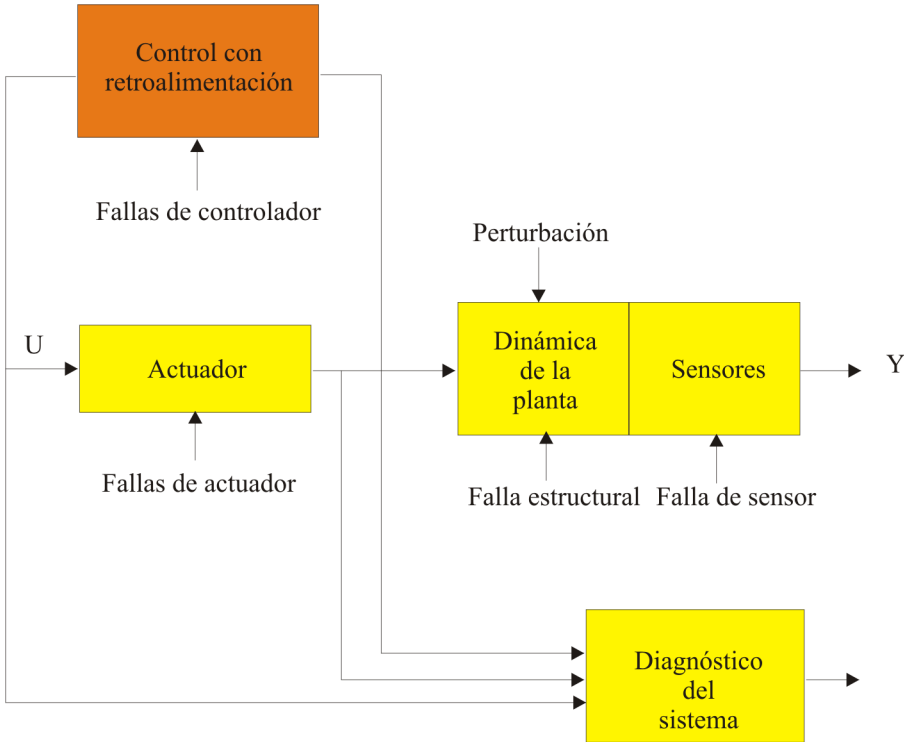


Figura 2.1: Diagrama general de diagnóstico

2.1.2. Cambios estructurales

Los cambios estructurales se refieren a los cambios del proceso. Estos ocurren debido a fuertes fallas en el equipo. Las fallas estructurales resultan ser un cambio en la información de las variables. Para manejar esta falla en el sistema de diagnóstico, es necesario remover y reestructurar de manera adecuada las ecuaciones del modelo matemático para la descripción de la situación real de proceso. Un ejemplo de una falla estructural sería una válvula enclavada, una tubería rota o una tubería con fugas, etc.

2.1.3. Falla de sensores y actuadores

La mayoría de los errores comúnmente ocurren con los actuadores y sensores. Esto puede ser debido a una falla establecida, desviaciones constantes tanto positivas como negativas, o una falla fuera del rango preestablecido como óptimo. Algunos instrumentos proveen una retroalimentación de señales las cuales son esenciales para el control de la planta. El propósito del diagnóstico, es detectar oportunamente la falla del instrumento el cual puede degenerar el rendimiento del sistema de control.

En este capítulo, se presentarán las diferentes técnicas que han sido propuestas para la solución del problema de detección de falla y diagnóstico. Se clasifican las técnicas en aproximaciones en los modelos cuantitativos, modelos cualitativos y procesos basados en historiales o datos históricos. Dentro del método cuantitativo basado en aproximaciones, se presentarán las técnicas que usa la redundancia analítica para la generación de residuos y que pueden ser usadas para la selección de las fallas durante el proceso. Se revisarán las técnicas para la generación de residuos como lo son los observadores, relaciones de paridad, filtros Kalman, etc. Bajo la técnica del modelo cualitativo, se analizarán las gráficas dirigidas signadas (*SDG signed directed graph*), falla de árboles, simulación cualitativa (*QSIM Qualitative Simulation*), y teoría del proceso cualitativo (*QPT Qual-*

itative Process Theory) para el diagnóstico de fallas. Y dentro de la técnica de aproximaciones basadas en datos históricos, se revisarán los sistemas expertos, análisis de la tendencia cualitativa (*QTA, Qualitative Trend Analysis*), redes neuronales artificiales, análisis de componentes principales (*PCA, Principal Component Analysis*) y clasificadores estadísticos.

Existen en su haber una gran cantidad de artículos que realizan revisiones dentro del campo del diagnóstico de fallas, considerándose todos los tipos diferentes de técnicas. La mayoría de los artículos como los de [Frank et al., 00] se hace un enfoque predominantemente en los modelos basados en aproximaciones. Por ejemplo, en el resumen de [Frank et al., 00], se presenta una descripción detallada de los diferentes tipos de modelos analíticos basados en aproximaciones. Los tópicos de robustez en detección de falla, generación optimizada de residuos y la generación de residuos en sistemas no lineales, son algunos de los temas que han sido explicados de manera comprensiva. Al igual, existen más artículos relacionados que caen en la misma categoría. Un resumen magnífico donde se exponen todas las técnicas disponibles para el diagnóstico de fallas, es presentado por [Kramer et al., 93]. Al transcurso de este capítulo se tratarán los temas de validación de datos, rectificación y diagnóstico de fallas. El problema de diagnóstico de fallas es visto como una extracción de características específicas así como una clasificación de etapas. La clasificación de etapas genera tres categorías: (i) reconocimiento de patrones, (ii) modelos basados en razonamiento y (iii) emparejamiento al modelo (*model-matching*).

Estrechamente asociado con el área de detección de falla y diagnóstico, está el área donde se investiga el error de detección de los datos del sensor y su subsiguiente validación. La detección del error o validación, se refiere a la identificación de la falla o los sensores que fallan en el proceso.

2.1.4. Clasificación de algoritmos de diagnóstico

Los modelos basados en un conocimiento previo pueden ser clasificados en modelos cualitativos y cuantitativos. El modelo es usualmente desarrollado de acuerdo al entendimiento fun-

damental del proceso físico. En los modelos cuantitativos, este entendimiento es expresado en términos de funciones matemáticas entre entradas y salidas del sistema. Por otro lado, los modelos de ecuaciones cualitativas son expresados en términos de funciones cualitativas.

En contraste con los modelos basados en aproximaciones, donde un conocimiento previo del modelo del proceso (tanto cuantitativo como cualitativo) es asumido, en los métodos basados en historiales, solamente se debe de tener una amplia gama de datos. Existen diferentes maneras donde estos datos pueden ser transformados y presentados como un conocimiento previo para el sistema de diagnóstico. Esto es conocido como un proceso de extracción de características de los datos históricos del proceso, lo cual facilita el diagnóstico. Este proceso de extracción puede ser realizado en los métodos cualitativos y cuantitativos. La extracción de los modelos cuantitativos se puede realizar con la ayuda de la estadística o métodos no estadísticos. Esta clasificación se presenta de manera general en la figura 2.2.

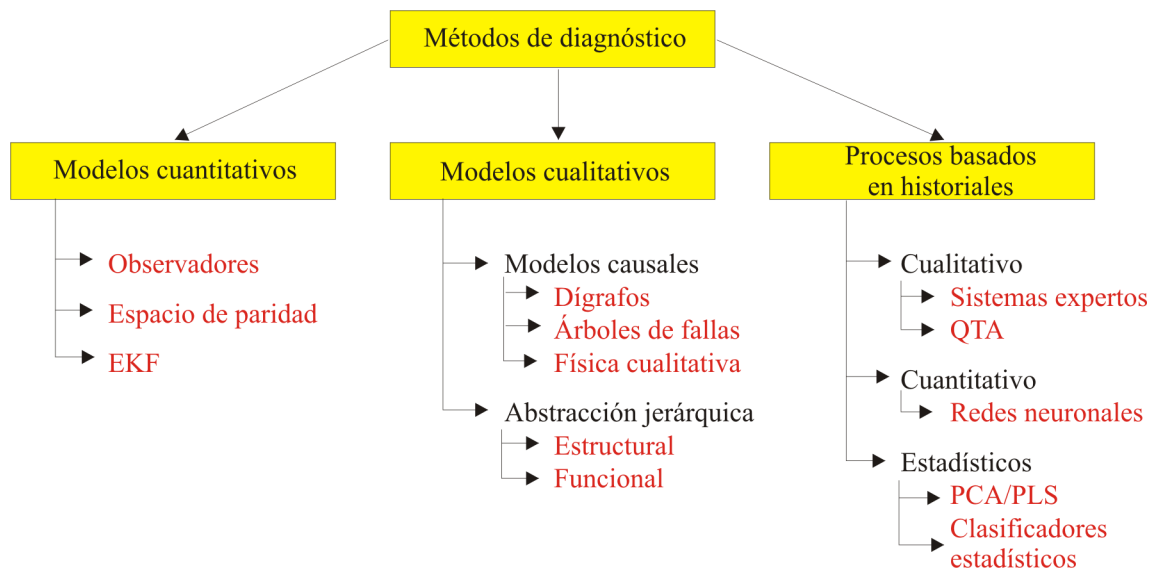


Figura 2.2: Clasificación de algoritmos de diagnóstico

2.1.5. Enfoque de los modelos cuantitativos

En esta sección, se realiza el resumen de los métodos de diagnóstico basados en modelos cuantitativos. Primeramente, se dará el concepto de redundancia analítica. Varios métodos serán descritos a manera de bosquejos incluyendo los observadores, relaciones de paridad, filtros Kalman y estimación de parámetros. Incluso también es revisado un nuevo enfoque llamado generación de residuos.

Redundancia analítica

En el área del control automático, la detección de fallas o cambios, es conocida como detección de falla y diagnóstico. Para ello, con ayuda de un modelo explícito de la planta a analizar, todos los métodos de detección de falla y diagnóstico (y cualquier método de diagnóstico por el método estadístico) requiere de dos pasos. El primer paso, es generar inconsistencias entre el comportamiento actual y el esperado. Tales inconsistencias, también llamados residuos, son señales artificiales reflejando las fallas potenciales del sistema. El segundo paso, refiere a una regla de decisión para el diagnóstico.

Existen dos tipos de redundancia: física y analítica. El primero requiere sensores redundantes. Esta redundancia es utilizada para el control de sistemas de seguridad críticos como plantas nucleares y cohetes espaciales. Sin embargo, su aplicación es limitada debido al costo extra y espacio requeridos. Por otro lado, la redundancia analítica (también llamado funcional, inherente o artificial), es desarrollada a partir de una dependencia funcional de las variables del proceso y usualmente es a partir de ecuaciones algebraicas o una serie temporal de relaciones, entradas y salidas del sistema. Este tipo de redundancia analítica puede ser clasificada en dos categorías, directa y temporal ([Chow et al., 94], [Frank et al., 90b]).

Una redundancia directa, es acompañada de relaciones algebraicas entre mediciones de difer-

entes sensores. Tales relaciones, son de gran ayuda para el cómputo del valor de medición con las mediciones de otros sensores. El valor calculado, es entonces comparado con las mediciones del valor del sensor. La discrepancia, indica que una falla en el sensor ha ocurrido. Una redundancia temporal, es obtenida a partir de relaciones diferenciales entre a diferentes las salidas de los sensores y las entradas a los actuadores. Con los datos de un proceso de entradas y salidas, la redundancia temporal es de gran uso para la detección de fallas en sensores y actuadores.

El esquema general para el uso de la redundancia analítica en un sistema de diagnóstico, es mostrado en la figura 2.3. La esencia de la redundancia analítica en el diagnóstico de fallas, es comparar el comportamiento actual del sistema contra el modelo del sistema para la consistencia. Cualquier inconsistencia expresada como residuos, puede ser usada para propósitos de detección y diagnóstico. Los residuos deben ser casi cero cuando no existe una falla, pero por el contrario, cuando exista alguna, deben ser valores significativos.

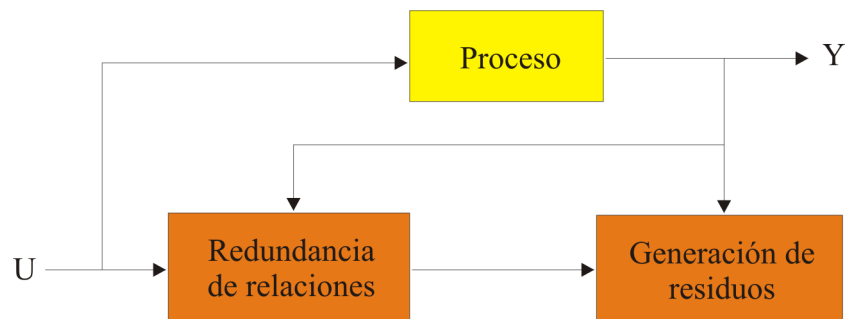


Figura 2.3: Esquema general para el uso de la redundancia analítica

Debe de notarse que la mayor ventaja de usar modelos cuantitativos, es el control sobre el comportamiento de los residuos. Sin embargo, varios factores como la complejidad del sistema, sus grandes dimensiones, la no linealidad de los procesos, lo hace más impráctico para el desarrollo de un modelo matemático del sistema.

La mayoría de los artículos con diferentes énfasis, han sido publicados a través de las dos décadas pasadas. El más antiguo es el [Willsky et al., 76], que cubre los métodos para el diseño específico de filtros de falla para el uso de pruebas estadísticas. [Isermann et al., 84], revisó los métodos de detección de falla basado en la estimación de parámetros no medibles y variables de estado.

[Isermann et al., 84] cita el problema de detección, estimación y diagnóstico de cambios, en señales de un proceso dinámico con el énfasis de los modelos estadísticos. [Frank et al., 90b], recalca los principios y las técnicas más importantes, usando identificación de parámetros y la identificación de estados con el énfasis de la robustez con respecto a los errores de los modelos. [Gertler et al., 91], presentó varios métodos para la generación de residuos incluyendo ecuaciones de paridad, observadores de diagnóstico y filtros Kalman. La teoría y la aplicación para la detección de cambios abruptos pueden ser encontrado en [Basseville et al., 86] y [Basseville et al., 93].

2.2. Modelos cualitativos

Los modelos basados en un conocimiento previo, pueden ser clasificados de manera general como cualitativos o cuantitativos. El modelo es usualmente desarrollado para el entendimiento físico del proceso. Los modelos cuantitativos, son expresados en términos de funciones matemáticas y relaciones entre entradas y salidas del sistema. En contraste, los modelos cualitativos son expresiones de funciones cualitativas. Los modelos cualitativos, pueden ser desarrollados como modelos causales cualitativos o como abstracción de jerarquías. La figura 2.4, muestra la diversidad de categorías previamente descritas.

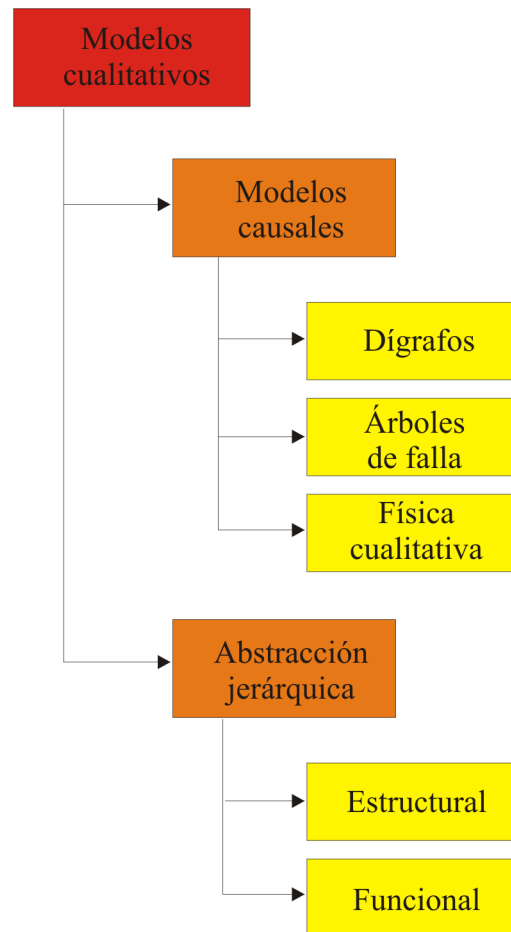


Figura 2.4: Formas del conocimiento cualitativo

2.3. Métodos basados en historiales de datos

En contraste con los enfoques donde se necesita un conocimiento previo (tanto cuantitativo como cualitativo) del proceso, en los métodos basados en historiales de datos, solamente como su nombre lo indica, se necesita una larga cantidad de datos del proceso. Existen también diferentes rutas en el cual los datos pueden ser transformados y presentados como un conocimiento previo para el sistema del diagnóstico. Esto es conocido como extracción de características. Este proceso de extracción puede ser, cualitativo como cuantitativo. Dos de los métodos más utilizados en la extracción de información por medio del método cualitativo, son los sistemas expertos y los métodos de tendencia del modelado. Los métodos que extraen información cuantitativa, pueden ser clasificados como estadísticos y no estadísticos. Las redes neuronales, son la clase más importante dentro de esta clasificación no estadística, mientras que por parte de los métodos estadísticos los de mayor uso son los PCA (*Análisis del componente principal*) y los PLS (*Partial Least Squares*). Los diferentes caminos para la extracción de este conocimiento son presentados esquemáticamente en la figura 2.5.

Dentro de este capítulo se darán las comparativas de los diferentes métodos, así como de sus fortalezas y debilidades por cada uno de ellos. Llegando a la conclusión de que cada uno es complemento de otro, es decir, no hay método solo al que se le pueda implementar el diagnóstico del sistema. Mientras se tenga un complemento entre métodos se puede realizar un mejor diagnóstico, llamados comúnmente métodos híbridos.

2.3.1. Extracción de características cualitativas

Como se mencionó anteriormente, dos de los métodos más importantes empleados para la extracción de características cualitativas, son los sistemas expertos y el enfoque de la tendencia de los modelos.

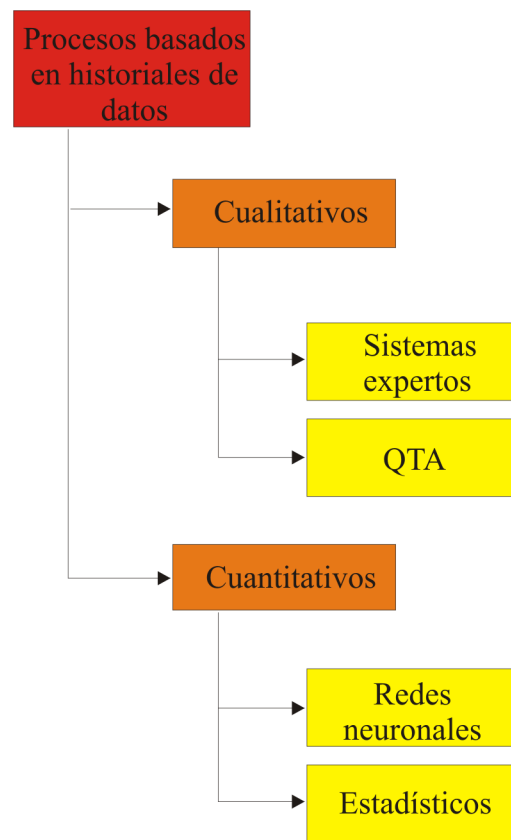


Figura 2.5: Clasificación de los métodos basados en historiales de datos

Sistemas expertos

La regla para la extracción de características, ha sido ampliamente usada en los sistemas expertos para varias aplicaciones. Un sistema experto, es generalmente un sistema muy especializado que resuelve problemas teniendo como base un dominio muy pequeño del sistema. Los principales componentes de un sistema experto son: adquisición del conocimiento, selección de la representación del conocimiento, codificación del conocimiento en un conocimiento base, el desarrollo de procedimientos para el razonamiento de diagnóstico y el desarrollo de interfaces de entrada y salida. Las principales ventajas en el desarrollo de un sistema experto para el diagnóstico son, fácil desarrollo, razonamiento transparente, la habilidad de razonar bajo ciertas circunstancias y la posibilidad de realizar explicaciones para las soluciones.

Existe en su haber un gran número de artículos donde se discuten los sistemas expertos y la aplicación es para el diagnóstico de falla de sistemas específicos. [Henley et al., 84], [Chester et al., 84] y [Niida et al., 85], [Venkatasubramanian et al., 89] son los artículos donde se discuten de manera amplia los sistemas expertos.

También hay un gran número de investigadores quienes han trabajado en la aplicación de los sistemas expertos para el diagnóstico de problemas. [Basila et al., 90], desarrollaron un sistema experto de supervisión, el cual se basa en la representación del conocimiento para representar un conocimiento heurístico. Siendo el sistema implementado en una operación para determinar la oxidación y el comportamiento de un reactor de oxidación. [Zhang et al., 91], presentaron una metodología para la formación de reglas extraídas a partir de las estructuras del sistema y de las funciones de los componentes. [Chen et al., 02], desarrollaron un sistema experto llamado **FAX**, para determinar la causa raíz de las malas operaciones y así realizar acciones correctivas para prevenir situaciones anormales. [Becraft et al., 93], han propuesto un marco integral de una red

neuronal con un sistema experto. La red neuronal, es usada como un filtro de primer nivel para diagnosticar las fallas más comunes en un proceso químico. Una vez localizadas, el sistema experto analiza el resultado y también confirma el diagnóstico y ofrece al sistema una posible solución. [Tarifa et al., 97], han propuesto un sistema híbrido que usa SDG (*Signed Directed Graphs*) y lógica difusa. El modelo SDG del proceso, es usado para la simulación cualitativa y predecir posibles fallas. Estas predicciones son para la generación de reglas if-then las cuales serán evaluadas por el sistema experto en base a la lógica difusa. [Zhao et al., 97] han presentado una red neuronal basado en la transformada Wavelet-Sigmoid y un sistema experto para el diagnóstico de fallas en un proceso de hidrocráqueo. [Wo et al., 00] presentaron un sistema experto para el diagnóstico de fallas, basado en la probabilidad. Realizaron gráficos de causa efecto y un modelo cualitativo. El sistema de diagnóstico fue aplicado a una columna de destilación.

Análisis de la tendencia cualitativa QTA

El segundo enfoque de la extracción de características cualitativas, es la abstracción de la tendencia de la información. El análisis de la tendencia así como la predicción, son componentes importantes para el monitoreo y supervisión del proceso. El modelado de la tendencia puede ser usado para explicar los eventos importantes que suceden durante el proceso, diagnosticar malas acciones y predecir futuros estados. Desde el punto de vista de un procedimiento, para la obtención de una señal de tendencia, es necesario implementar filtros para evitar así la adquisición de ruido. Para tareas como la de los diagnósticos, las representaciones de la tendencia cualitativa regularmente proveen información valiosa, que facilita el razonamiento acerca del comportamiento del proceso. En la mayoría de los casos, las fallas del proceso dejan una tendencia distinta comparada con los sensores monitoreados. Estas tendencias pueden ser utilizadas para identificar la anomalía del proceso. Entonces, una clasificación y un análisis de una tendencia del proceso pueden detectar fallas incipientes.

[Cheung et al., 90], crearon de manera formal un formato para la representación de la tendencia del proceso. Ellos introdujeron el concepto de la triangulación, la cual representa las tendencias. La triangulación, es un método donde cada segmento de una tendencia es representada por una inclinación inicial y una final y una línea que conecta estos puntos. La serie de triángulos constituyen la tendencia del proceso. [Janusz et al., 91], identifican un conjunto de datos primitivos los cuales representan la tendencia. Se utiliza el método diferencial finito, para calcular la primera y segunda derivada de los cambios de la tendencia del proceso y en base a estos valores, los datos primitivos son identificados.

Redes neuronales

Un gran interés ha sido mostrado en diferentes artículos y literatura en la aplicación de las redes neuronales para el problema de diagnóstico de fallas. Las redes neuronales han sido propuestas para la clasificación de problemas. En general, las redes neuronales utilizadas para el diagnóstico pueden ser representadas de dos maneras: i) La arquitectura de la red como una sigmoideal, de base radial, entre otras, y ii) la estrategia para el entrenamiento que puede ser la supervisada o no supervisada. Diferentes arquitecturas de red han sido usadas para el problema de diagnóstico de fallas.

En las estrategias de aprendizaje supervisado y por escoger una topología de red neuronal, ésta será parametrizada en el sentido de que el problema sea reducido para la estimación de las conexiones de los pesos. Esto hace que una red neuronal supervisada sea una buena opción para la clasificación de fallas con el detalle de que la red tiene la característica de generar y clasificar las fallas. Por otro lado, también las redes pueden ser entrenadas utilizando técnicas no supervisadas. Este tipo de redes son conocidas como auto-organizativas.

La red neuronal más popular que utiliza la estrategia de entrenamiento supervisado, ha sido la de retro-propagación. Existen artículos relacionados con la detección de fallas y diagnóstico,

usando la red de retro-propagación. En la ingeniería química [Venkatasubramanian et al., 85] y [Chan et al., 89], [Ungar et al., 90] y [Hoskins et al., 91] fueron los primeros en demostrar la gran utilidad de las redes neuronales para el problema de diagnóstico de fallas.

2.4. Estado del arte de las redes neuronales

La meta principal de un invernadero, es optimizar la cantidad así como la calidad de la producción. La optimización es alcanzada utilizando sistemas automatizados para el control del ambiente interno del invernadero, para entonces, dar así a la planta una condición óptima de crecimiento. Un cultivo hidropónico proporciona la oportunidad de controlar de manera precisa el ambiente donde se desarrolla la planta y, por lo tanto, tener más producción de cultivo.

Perder el control climático en un invernadero, es un fenómeno que usualmente tiene efectos negativos sobre la producción, inversión inicial y, consecuentemente, en los beneficios del invernadero. La pérdida del control climático del invernadero suele ser por diversas razones. Una de ellas, es la operación anómala de los sensores por diferentes causas o por una operación anómala de los actuadores (e.g. Motores trifásicos del sistema de ventilación, bombas para el riego del cultivo, etc.). Ciertas fallas son fácilmente detectables, mientras otras pueden ser difíciles de encontrar. Obviamente, un controlador de lazo cerrado del invernadero, puede ser capaz de mantener las condiciones deseadas, incluso cuando algunas partes del mecanismo de control están fuera de orden. Sin embargo, especialmente cuando las fallas afectan directamente a la planta, los efectos suelen ser desastrosos para toda la producción. Es por eso, que la detección y diagnóstico de fallas resulta ser un área muy importante. En particular, cuando se encuentra la operación anómala en una etapa temprana donde se empieza a desarrollar la falla, para realizar un diagnóstico del problema o causa. Esta detección de fallas, puede ser encontrada a partir de las mediciones de los sensores del ambiente donde se desarrolla la planta.

Existen diferentes métodos dentro del método de redundancia analítica para la detección de fallas en sensores, como los son: los métodos analíticos de donde se derivan las técnicas de estimación paramétrica y los observadores y espacios de paridad. Existe otro tipo de método de redundancia analítica el cual está basado en un conocimiento del proceso o sistema, como lo son: los modelos borrosos (*fuzzy*) y los cualitativos.

Dentro del campo de la redundancia analítica, se cuenta con los métodos basados en datos, dentro de esta rama, se encuentra la inteligencia artificial; esta técnica ha tenido una aceptación dentro del campo de la ingeniería, debido a sus grandes cualidades para la resolución de problemas y su versatilidad para la implementación en software y hardware. Varios investigadores se han declinado por este tipo de método, debido a que las redes neuronales resuelven problemas que con otros métodos tradicionales serían imposibles de resolver. Otra ventaja de aplicar una red neuronal, es la capacidad de adaptación a cualquier tipo de sistema. A continuación se presentan investigaciones recientes donde se aplica este tipo de técnica. Las redes neuronales han demostrado su efectividad en la identificación de fallas en varios procesos biológicos. [Matsura et al., 89], utilizaron redes neuronales multicapa, para detectar cinco tipos diferentes de fallas en un reactor químico. Los resultados obtenidos utilizando una red neuronal, sirvieron para diagnosticar fallas incipientes a partir de solamente tres mediciones, lo cual es muy interesante considerando la complejidad del proceso examinado.

Por su parte [Chan et al., 89], examinaron la detección de fallas en una unidad de craqueo catalítico, (proceso químico por el cual se quiebran moléculas de un compuesto produciendo así compuestos más simples) utilizando las redes neuronales. En particular, su metodología llevó a cabo un diagnóstico de fallas múltiples mientras es entrenada la red a través de las fallas individuales. Concluyeron que las redes neuronales tienen la habilidad de aprender a partir de ejemplos y de extraer características principales de los datos. Sorsa et al. (1991), reportaron tres tipos de

arquitectura de redes neuronales usados para la detección de fallas en un proceso simulado. El proceso constaba de un intercambiador de calor y un tanque reactor. Sus conclusiones fueron que una red multicapa perceptron (*MLP Multy-Layer Perceptron*), y el uso de la tangente hiperbólica como función de activación no-lineal, fue la de mejor resultado. Por supuesto, debe de entenderse que todo el trabajo fue basado exclusivamente en simulación y el entrenamiento se realizó con datos adquiridos a partir de un modelo matemático.

[Parlos et al., 94], [Hoskins et al., 94], [Chow et al., 94] y Xiaoming et al. (1997), mostraron resultados similares en la aplicación de las redes neuronales para la detección de falla y diagnóstico en plantas químicas. Sin embargo, el entrenamiento de todos estos trabajos fue a partir de datos adquiridos por simulaciones o por modelos matemáticos. Esto es riesgoso, ya que si el diagnóstico de fallas es implementado físicamente sería incorrecto, es por eso que para la estabilización de estos sistemas, la red neuronal se debe entrenar a partir de datos reales.

Algunos procesos biotecnológicos también han sido parte del estudio de la detección de fallas, [Fuente et al., 99], implementaron un nuevo método de diagnóstico y detección de anomalías, desarrollándose en diferentes partes de un proceso de tratamiento de agua residual situada en Manresa, España. Los investigadores demostraron que la aplicación de dicho método, combina algoritmos basados en técnicas de estimación de parámetros usada para generar señales de mala operación, y una red neuronal de retropropagación para analizar la frecuencia contenida de dichas señales. La forma de entrenamiento, es mediante el método de regresión de mínimos cuadrados (*RLS recursive least-square*).

La tecnología tolerante a fallas ha crecido de manera significativa en los sistemas de aviación militar, la cual una pronta detección de falla puede evitar catástrofes al sistema, así como a la vida del piloto. [Chen et al., 02], presentaron un esquema de red neuronal para la detección de la falla de un sistema así como su diagnóstico. La red neuronal, fue utilizada para detectar la falla de un

actuador, sensor y falla en su dinámica de una aeronave modelo F-16. El modelo de monitoreo de fallas fue subdividido en tres etapas: detección de falla, diagnóstico y adaptación. La investigación, únicamente realizó la detección y diagnóstico de falla. La implementación de una red neuronal facilitó el diagnóstico y la detección comparada con un modelo matemático que represente el sistema, además de las dificultades que el modelo conlleva. La red neuronal de topología Radial Basis Function, sustituyó al modelo de generación de residuos (mediciones que reflejan una operación normal) y su entrenamiento consistió en datos del sistema. Una vez terminado el entrenamiento, la red fue implementada en línea para la generación de residuos. Para llevar a cabo el diagnóstico de falla, fue necesaria una nueva arquitectura (red neuronal multicapas) que fue alimentada y entrenada por los residuos generados por la primera topología. Una vez realizadas estas acciones, se procedió a la implementación en línea y así determinar si una falla había ocurrido en el sistema e indicar su posible causa. La principal ventaja de utilizar la red neuronal, es su facilidad de implementación en el sistema, su proximidad de modelar sistemas no-lineales y su tiempo de entrenamiento.

La forma de implementar la detección, puede crecer a partir de un modelo matemático para su simulación y en un futuro para su implementación física, es el caso del estudio realizado por [Thomas et al., 02], quienes realizaron un método de detección de fallas, así como su aislamiento en sistemas no-lineales (modelo matemático) utilizando las redes neuronales, este modelo supervisado utiliza una arquitectura muy particular donde la capa escondida presenta neuronas de más, que conectadas a las neuronas de salida y en algunas de entrada. Estas neuronas suplementarias, permiten llevar a cabo una estimación en cada entrada de la red y la comparación de esta estimación con las entradas actuales, usando técnicas estadísticas, para detectar y localizar la presencia de una falla a partir de su entrada. La simulación del sistema fue a partir de un modelo matemático en tiempo discreto y consta de tres entradas y una salida. Como primer paso, se buscó una estructura de red deseada, esta búsqueda fue llevada a cabo usando una estructura, donde los pesos de la red son innecesarios. Este estudio demostró que el sistema puede ser modelado con una red de ocho

neuronas de capa escondida. Es así que la arquitectura de la red para el diagnóstico, fue de 14 neuronas de capa escondida y solamente seis fueron conectadas a la neurona de salida.

Cabe destacar que la detección de fallas, ha sido implementada en diferentes campos como es en la agricultura. [Ferentinos et al., 03a], desarrollaron una red neuronal para la detección de fallas biológicas, mecánicas y de sensores en un sistema hidropónico (Lechuga *Lactuca sativa*, var. *Vivaldi*). Esta red neuronal fue separada en dos modelos para detección de fallas: una, utilizada para detectar una mala operación del sistema hidropónico la cual es causada por anomalías mecánicas, de actuadores y fallas de sensores; la otra, para la detección de fallas biológicas según su categoría (i.e. situaciones específicas de estrés en las plantas) llamadas "fallas de transpiración". La metodología fue validada y resultó ser exitosa en la tarea de detección, en ambas aplicaciones. El modelo general de detección fue capaz de detectar una situación anómala en un tiempo relativamente corto, en la mayoría de los casos en menos de 20 a 40 minutos. En el caso del modelo de "falla de transpiración" fue detectada en un tiempo de 2 a 3 horas, que es un tiempo razonable por la interacción natural que existe entre la planta y el ambiente, es por ello, que el detectar el estrés de una planta, es de vital importancia para la producción final. La información combinada de diferentes sensores a través de las metodologías de la inteligencia artificial como lo fueron las redes neuronales, llevó a clasificar información derivada de sensores específicos y de actuadores. Es por eso que, la detección de falla y diagnóstico de un sistema, pudo ser desarrollado y fue capaz de detectar e identificar fallas específicas en partes de un sistema hidropónico, por simples lecturas adquiridas de mediciones de los sensores así como de los actuadores del sistema. La metodología de las redes neuronales así como su validación, fueron exitosas en ambas aplicaciones. [Ferentinos et al., 03b], desarrollaron un detector de fallas y diagnóstico en un sistema hidropónico, utilizando herramientas computacionales como parte de una extensión del trabajo realizado en [Ferentinos et al., 03a]. Las herramientas computacionales utilizadas, fueron las redes neuronales y los algoritmos genéticos para detectar fallas mecánicas, de sensores y biológicas; los experimentos se realizaron en invernaderos de la Universidad de Cornell, Ithaca, NY, Estados

Unidos. El algoritmo genético, fue una nueva técnica para el diseño y entrenamiento de una red neuronal, basado en métodos heurísticos de optimización. Las fallas de sensores así como de actuadores, fueron detectadas y diagnosticadas en un tiempo suficiente, de tal forma que el modelo pudo ser aplicado en línea y utilizado como supervisor de operación de un sistema. Las fallas biológicas no fueron detectadas en general y como parte principal es que el sistema desarrollado por el método de los algoritmos genéticos, puede ser aplicado a una serie de problemas como lo puede ser la decisión de la mejor arquitectura de red neuronal, funciones de activación y algoritmos de entrenamiento para un modelo en específico. La mayor ventaja de la aplicación de una red neuronal de arquitectura feedforward, para ser utilizado como un modelo de detección de falla, es que no es necesario un modelo matemático. Es por eso que la detección básicamente, es en base a mediciones reales del proceso. Los resultados de encontrar una anomalía tanto en actuadores como en sensores, fueron en tiempo real, así como en una etapa inicial; la red implementada fue capaz de aprender de interacciones tanto físicas como químicas entre las plantas y las variables medidas en la zona de la raíz. El sistema desarrollado, mostró gran capacidad de generalización en la tarea de detección y diagnóstico con nuevos datos. La arquitectura de red neuronal con una capa escondida de 28 neuronas, con algoritmo de entrenamiento de máxima pendiente (*steepest-descent*) y con función de activación de tangente hiperbólica, mostró mejores resultados comparándolo con una arquitectura de dos capas escondidas. El algoritmo genético, fue comparado con tareas de "prueba y error" teniendo resultados similares, en la mayoría de los casos, el modelo del algoritmo genético es preferible, porque es una metodología automática basada en optimización para ser aplicado en modelos específicos.

Para conseguir la máxima producción, los sistemas modernos de adquisición de datos son ahora más sofisticados, y son caracterizados por su compleja relación los subsistemas. Es por eso que la aplicación de detección de fallas, se puede aplicar a los grandes volúmenes de adquisición de datos de sistemas para detectar anomalías en las lecturas. [Jakubek et al., 04], desarrolló una red neuronal para la detección de fallas en un sistema de adquisición, donde los datos obtenidos

son de largas adquisiciones de datos. El esquema general de detección, procesa cientos de datos, producto de diferentes mediciones en tiempos determinados para examinar sus consistencias. La detección consta de tres pasos: el primero, el análisis de componentes principales de los datos de entrenamiento, son usados para determinar las áreas abundantes del espacio de medición. La detección de fallas, es realizada checando si un nuevo dato adquirido es erróneo, comparándolo con una serie de datos del mismo tipo. El segundo, la función de distribución del dato disponible, es estimado usando técnicas de regresión kernel, y en un tercer paso, la función de distribución es aproximada por una red neuronal. La red neuronal propuesta, es una función de base elipsoidal (*ellipsoidal basis function*) y fue comparada con una función de base radial (*radial basis function*), siendo la primera red de mejor rendimiento. Una particularidad importante del esquema detector de fallas, es la habilidad de auto-adaptarse a los nuevos datos. El diseño propuesto, fue realizado bajo ejemplos numéricos y físicos, con datos reales mostrando que el método sugerido produce excelentes resultados. Se realizó una simulación con mediciones reales de un motor de gas y para la validación del método, fue necesario que las mediciones adquiridas tuvieran perturbaciones, estas perturbaciones fueron desviaciones durante la medición, errores de calibración, desconexión de sensor. Los resultados de detección estuvieron alrededor de un 90%. La implementación de este esquema, se realizó en diferentes motores de gas y diesel, así como en un proceso de reactor/separador/recirculador (*Tennessee Eastman process*), demostrando que el método tiene óptimos resultados en detección de fallas.

[Mehranbod et al., 05], realizaron un método para la detección de falla de sensor así como su identificación. El método que se propuso fue de una red de confianza bayesiana (*Bayesian Belief Networks BBN*) y ésta tuvo la capacidad de identificar una anomalía en el sensor analizado. Este método, puede ser aplicado tanto a procesos transitorios como a procesos de estado estable. Durante los procesos transitorios, se puede implementar un modelo simple de red bayesiana con nodos adaptables para un sensor. El modelo simple, es usado como un bloque constructor para desarrollar un modelo de red multi-etapas para todos los sensores en el proceso bajo consideración. Dentro del

entorno de la red bayesiana, los datos condicionales probabilísticos representan relación entre las mediciones de variables del proceso. Para un modelo de red multi-etapas, los datos probabilísticos, deben estar disponibles en cada instante de tiempo durante el periodo transitorio; este concepto fue usado para evitar ciclos directos entre la red BBN que representa a los sensores de un sistema transitorio. Esto requiere generar y procesar un banco, de datos que reduce por consecuencia la eficiencia computacional. El principal aporte de la investigación, es la reducción del tamaño de los datos probabilísticos requeridos a un solo grupo. Este método, además, proporciona eficiencia computacional sin sacrificar la efectividad de detección y de identificación.

Los métodos y las investigaciones realizadas para la detección de falla han evolucionado de manera significativa, [Tan et al., 07] proponen un modelo de red neuronal híbrida para la generación de reglas, así como su aplicación hacia un proceso para la detección de falla y diagnóstico. La red híbrida, es la integración de fuzzy ARTMAP y una función de base rectangular (*RecBFN*) la cual tiene capacidad de aprendizaje y generar reglas de evaluación para lógica difusa (*fuzzy*). La red neuronal tiene la habilidad de clasificar incrementalmente muestras de datos, al mismo tiempo de generar reglas directamente del ajuste de pesos de la red para justificar sus predicciones. Es así como el método propuesto de detección y diagnóstico de fallas, fue aplicado a una planta generadora de energía. Específicamente, la eficiencia de la red, en monitorear las condiciones de operación de un sistema recirculador de agua fue evaluada, usando un conjunto de mediciones reales adquiridas por la estación de energía. Los datos extraídos fueron analizados, discutidos y comparados con los métodos para generar reglas, FAM. La comparación resultó de gran, utilidad ya que la red propuesta, es capaz de extraer mejores reglas con un menor grado de redundancia y una mejor interpretación. Las reglas extraídas fueron analizadas por expertos en el mantenimiento del sistema de recirculación de agua. El objetivo principal de este trabajo, es presentar cómo la red RecBFM puede ser utilizada para la extracción y perfeccionamiento del conocimiento a partir de datos, y cómo el conocimiento extraído puede ser presentado como reglas y así ser presentadas a los usuarios.

Las investigaciones previamente descritas, amplían el panorama para la búsqueda de la realización e implementación de un sistema detector de fallas, así como para su diagnóstico en ambientes controlados como lo puede ser el invernadero. Los nuevos proyectos y publicaciones servirán para conocer las tendencias e innovaciones de las diferentes aplicaciones. Fundamentalmente, las investigaciones tratan el comportamiento de los sensores y actuadores, para permitir el buen funcionamiento del sistema y evitar en el caso de un invernadero, la pérdida de la inversión de capital así como del producto o cultivo. El trabajo consistirá en implementar dentro del sistema de control climático de invernadero, un detector de fallas tanto de sensores como de actuadores y así realizar un diagnóstico de la posible falla. Referencias relacionadas con detección de falla y diagnóstico en un sistema hidropónico como lo son [Ferentinos et al., 03a] y [Ferentinos et al., 03b], darán de manera general la problemática existente y su metodología llevada a cabo. Teniendo así una referencia para comparación.

Capítulo 3

Metodología

3.1. Características de una red neuronal artificial

Existen varias formas de nombrar una neurona artificial, es conocida como nodo, neuronodo, celda, unidad o elemento de procesamiento (PE); en la figura 3.1 se observa un PE en forma general y su similitud con una neurona biológica.

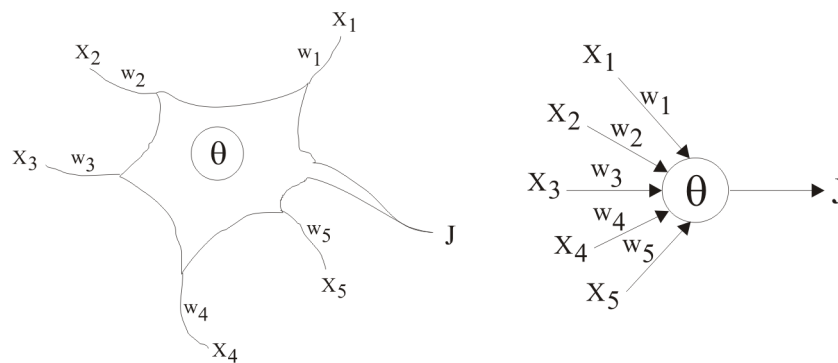


Figura 3.1: De la neurona biológica a la neurona artificial

De la observación detallada del proceso biológico, se han hallado los siguientes análogos con el sistema artificial:

- Las entradas X_i representan las señales que provienen de otras neuronas y que son capturadas

por las dendritas.

- Los pesos W_i son la intensidad de la sinápsis que conecta dos neuronas; tanto X_i como W_i son valores reales.
- θ es la función umbral que la neurona debe superar para activarse; este proceso ocurre biológicamente en el cuerpo de la célula.

Las señales de entrada a una neurona artificial X_1, X_2, \dots, X_n son variables continuas en lugar de pulsos discretos, como se presentan en una neurona biológica. Cada señal de entrada pasa a través de una ganancia o peso, llamado peso sináptico o fortaleza de la conexión, cuya función es análoga a la de la función sináptica de la neurona biológica. Los pesos pueden ser positivos (excitatorios), o negativos (inhibitorios), el nodo sumatorio acumula todas las señales de entradas multiplicadas por los pesos o ponderadas y las pasa a la salida a través de una función umbral o función de transferencia. La entrada neta a cada unidad puede escribirse de la siguiente manera:

$$neta_i = \sum_{i=1}^n W_i X_i \quad (3.1)$$

Una idea clara de este proceso se muestra en la figura 3.2, en donde puede observarse el recorrido de un conjunto de señales que entran a la red.

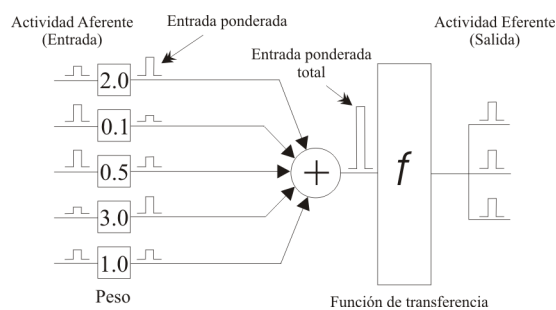


Figura 3.2: Proceso de una red neuronal

Una vez que se ha calculado la activación del nodo, el valor de salida equivale a:

$$x_i = f_i(neta_i) \quad (3.2)$$

Donde f_i representa la función de activación para esa unidad, que corresponde a la función escogida para transformar la entrada $neta_i$ en el valor de salida x_i y que depende de las características específicas de cada red.

3.1.1. Notación

Una notación matemática estándar no ha sido aún establecida para las redes neuronales, ya que sus aplicaciones son útiles en muchos campos, ingeniería, física, psicología y matemáticas. En este trabajo, se adoptó la siguiente convención para identificar las variables, de manera que fueran compatibles con las diferentes áreas, siendo lo más sencilla posible:

- Valores escalares: se representarán por medio de letra minúscula itálica.
- Vectores: se representarán con letra itálica minúscula en negrilla.
- Matrices: se representarán con letra mayúscula itálica en negrilla.

Para redes multicapa, los parámetros adoptaran la siguiente forma:

$$W_{S^c, S^c}^c \quad (3.3)$$

Donde c , es el número de la capa a la que corresponde dicho peso, y s representa las neuronas que participan en proceso.

Así $W_{1,1}^2$ representa el peso de la segunda capa que comunica la primera neurona de dicha capa con la primera neurona de la primera capa. De igual manera, el peso que representa la conexión desde la última neurona de la capa dos a la última neurona de la capa uno estará representado por:

$$W_{S^2, S^1}^2 \quad (3.4)$$

Esta convención es adoptada para todos los parámetros de la red.

$$a = \begin{cases} 1 & \text{sin} \geq 0 \\ 0 & \text{sin} < 0 \end{cases} \quad (3.5)$$

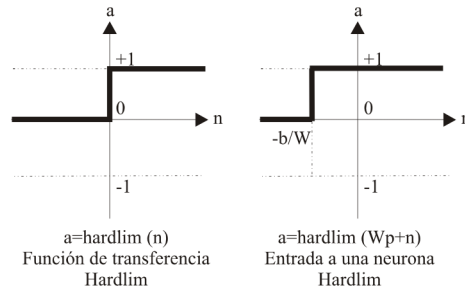


Figura 3.4: Función de transferencia hardlim

El icono para la función Hardlim reemplazara a la letra f en la expresión general, cuando se utilice la función Hardlim.

Una modificación de esta función puede verse en la figura 3.5, la que representa la función de transferencia Hardlims que restringe el espacio de salida a valores entre 1 y -1 .

$$a = \begin{cases} 1 & \text{sin} \geq 0 \\ -1 & \text{sin} < 0 \end{cases} \quad (3.6)$$

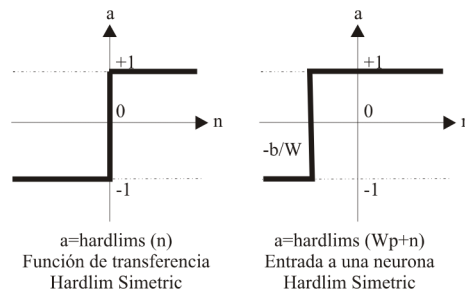


Figura 3.5: Función de transferencia hardlims

Función de transferencia Purelin (Lineal)

La salida de una función de transferencia lineal es igual a su entrada $a = n$. En la gráfica del lado derecho de la figura 3.6, puede verse la característica de la salida a de la red, comparada con la entrada p , más un valor de ganancia b , neuronas que emplean esta función de transferencia son utilizadas en la red tipo Adaline.

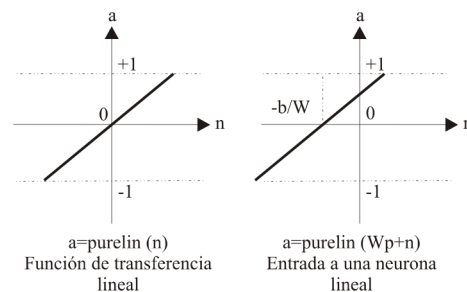


Figura 3.6: Función de transferencia lineal

Función de transferencia Logsig (Sigmoidal)

Esta función toma los valores de entrada, los cuales pueden oscilar entre más y menos infinito, y restringe la salida a valores entre cero y uno, de acuerdo a la expresión:

$$a = \frac{1}{1 + e^{-n}} \quad (3.7)$$

Esta función es comúnmente usada en redes multicapa, como la Backpropagation, en parte porque la función logsig es diferenciable.

El cuadro 3.1 hace una relación de las principales funciones de transferencia empleadas en el entrenamiento de redes neuronales.

3.1.3. Topología de una red neuronal

Típicamente una neurona tiene más de una entrada; en la figura 3.8 se observa una neurona con R entradas; las entradas individuales p_1, p_2, \dots, p_R son multiplicadas por los pesos correspon-

Nombre	Relación entrada-salida	Función
Limitador fuerte	$a = 0$ si $n < 0$ $a = 1$ si $n \geq 0$	Hardlim
Limitador fuerte simétrico	$a = -1$ si $n < 0$ $a = +1$ si $n \geq 0$	Hardlims
Lineal positiva	$a = 0$ si $n < 0$ $a = n$ si $0 \leq n$	Poslin
Lineal	$a = n$	Purelin
Lineal saturado	$a = 0$ si $n < 0$ $a = n$ si $0 \leq n \leq 1$ $a = 1$ si $n > 1$	Satlin
Lineal saturado simétrico	$a = -1$ si $n < -1$ $a = n$ si $-1 \leq n \leq 1$ $a = 1$ si $n > 1$	Satlins
Sigmoidal logaritmico	$a = \frac{1}{1+e^{-n}}$	Logsig
Tangente sigmoidal hiperbólica	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$	Tansig

Cuadro 3.1: Funciones de transferencia

dientes $w_{1,1}, w_{1,2}, \dots, w_{1,R}$ pertenecientes a la matriz de pesos \mathbf{W} .

La neurona tiene una ganancia b , la cual llega al mismo sumador al que llegan las entradas multiplicadas por los pesos, para formar la salida n ,

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \dots + w_{1,R}p_R + b \quad (3.8)$$

Esta expresión puede ser escrita en forma matricial

$$n = \mathbf{W}\mathbf{p} + b \quad (3.9)$$

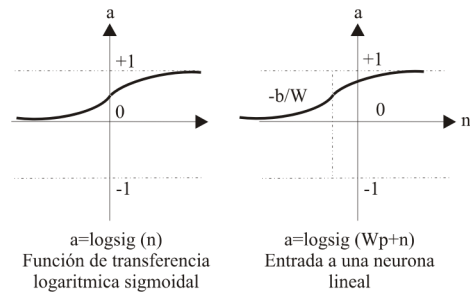


Figura 3.7: Función de transferencia sigmoideal

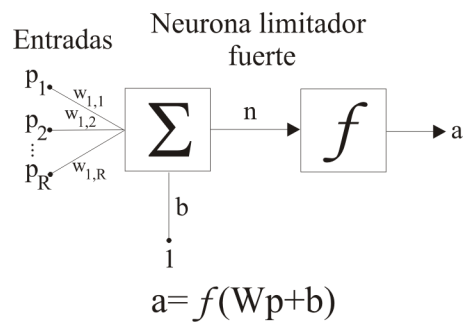


Figura 3.8: Neurona con multiples entradas

Los subíndices de la matriz de pesos, representan los términos involucrados en la conexión, el primer subíndice, representa la neurona destino y el segundo, representa la fuente de la señal que alimenta a la neurona. Por ejemplo, los índices de $w_{1,2}$ indican que este peso es la conexión desde la segunda entrada a la primera neurona. Esta convención se hace más útil cuando hay más de una neurona, o cuando se tiene una neurona con demasiados parámetros; en este caso, la notación de la figura 3.8, puede resultar inapropiada y se prefiere emplear la notación abreviada representada en la figura 3.9.

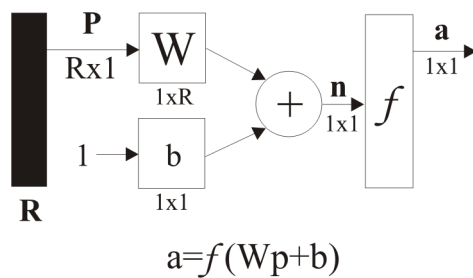


Figura 3.9: Neurona con múltiples entradas, notación abreviada

El vector de entrada \mathbf{p} es representado por la barra sólida vertical a la izquierda. Las dimensiones de p son mostradas en la parte inferior de la variable como $R \times 1$, indicando que el vector de entrada es un vector fila de \mathbf{R} elementos. Las entradas van a la matriz de pesos \mathbf{W} , la cual tiene \mathbf{R} columnas y solo una fila para el caso de una sola neurona. Una constante 1 entra a la neurona multiplicada por la ganancia escalar b . La salida de la red a , es en este caso un escalar, si la red tuviera más de una neurona a sería un vector.

Dentro de una red neuronal, los elementos de procesamiento se encuentran agrupados por capas, una capa es una colección de neuronas; de acuerdo a la ubicación de la capa en la RNA, esta recibe diferentes nombres:

- Capa de entrada: recibe las señales de la entrada de la red, algunos autores no consideran el vector de entrada como una capa, pues allí no se lleva a cabo ningún proceso.

- Capas ocultas: estas capas son aquellas que no tienen contacto con el medio exterior, sus elementos pueden tener diferentes conexiones y son estas las que determinan las diferentes topologías de la red.
- Capa de salida: recibe la información de la capa oculta y transmite la respuesta al medio externo.

Una red de una sola capa con un número S de neuronas, se observa en la figura 3.10 en la cual, cada una de las R entradas es conectada a cada una de las neuronas, la matriz de pesos tiene ahora S filas.

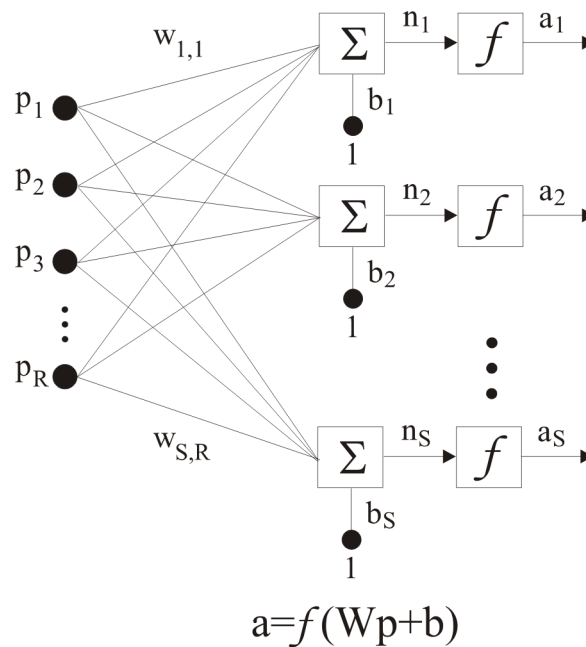


Figura 3.10: Capa de S neuronas

La capa incluye la matriz de pesos, los sumadores, el vector de ganancias, la función de transferencia y el vector de salida. Esta misma capa se observa en notación abreviada en la figura 3.11.

En la figura 3.11 se han dispuesto los símbolos de las variables, de tal manera que describan las características de cada una de ellas, por ejemplo, la entrada a la red es el vector \mathbf{p} cuya longitud R aparece en su parte inferior, \mathbf{W} es la matriz de pesos con dimensiones $S \times R$ expresadas debajo del símbolo que la representa dentro de la red, \mathbf{a} y \mathbf{b} son vectores de longitud S el cual, como se ha dicho anteriormente, representa el número de neuronas de la red.

Ahora, si se considera una red con varias capas, o red multicapa, cada capa tendrá su propia matriz de peso \mathbf{W} , su propio vector de ganancias \mathbf{b} , un vector de entradas netas \mathbf{n} , y un vector de salida \mathbf{a} . La versión completa y la versión en notación abreviada de una red de tres capas, pueden ser visualizadas en las figuras 3.12 y 3.13, respectivamente.

Para esta red se tienen R entradas, S^1 neuronas en la primera capa, S^2 neuronas en la segunda capa, las cuales pueden ser diferentes; las salidas de las capas 1 y 2 son las entradas a las capas 2 y 3 respectivamente, así la capa 2 puede ser vista como una red de una capa con $R = S^1$ entradas, $S^1 = S^2$ neuronas y una matriz de pesos W^2 de dimensiones $S^1 \times S^2$.

Las redes multicapa son más poderosas que las redes de una sola capa, por ejemplo, una red de dos capas que tenga una función sigmoideal en la primera capa y una función lineal en la segunda, puede ser entrenada para aproximar muchas funciones de forma aceptable.

En general, las redes neuronales se pueden clasificar de diversas maneras, según su topología, forma de aprendizaje (supervisado o no supervisado), tipos de funciones de activación, valores de entrada (binarios o continuos); un resumen de esta clasificación se observa en la figura 3.14.

3.2. Red neuronal backpropagation

3.2.1. Antecedentes

La regla de aprendizaje del Perceptrón de Rosenblatt y el algoritmo LMS de Widrow y Hoff fueron diseñados para entrenar redes de una sola capa. Estas redes, tienen la desventaja que solo

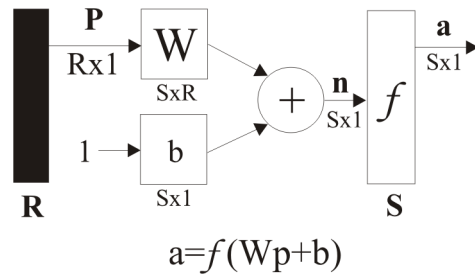


Figura 3.11: Capa de S neuronas con notación abreviada

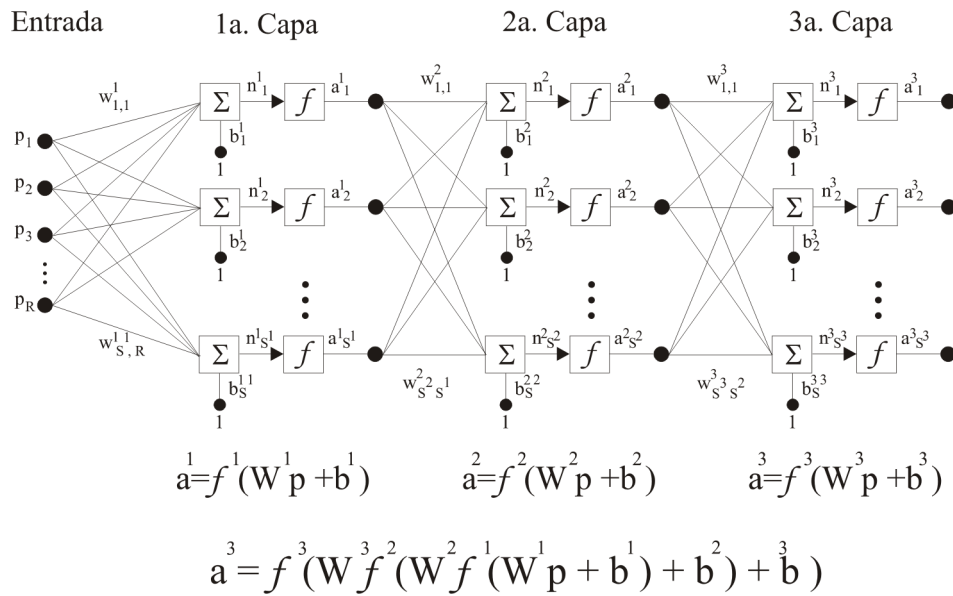


Figura 3.12: Red neuronal de 3 capas

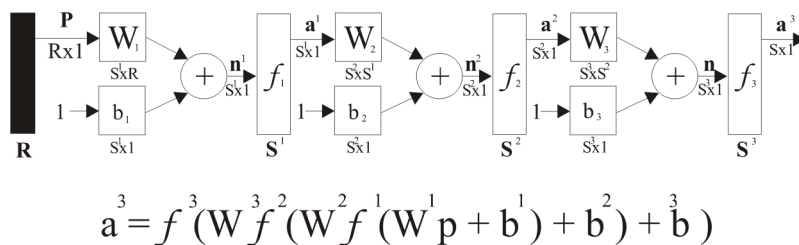


Figura 3.13: Red de tres capas con notación abreviada

pueden resolver problemas linealmente separables, fue esto lo que llevo al surgimiento de las redes multicapa para sobrepasar esta dificultad en las redes hasta entonces conocidas.

El primer algoritmo de entrenamiento para redes multicapa, fue desarrollado por Paul Werbos en 1974, éste se desarrolló en un contexto general, para cualquier tipo de redes, siendo las redes neuronales una aplicación especial, razón por la cual el algoritmo no fue aceptado dentro de la comunidad de desarrolladores de redes neuronales. Fue sólo hasta mediados de los años 80, cuando el algoritmo Backpropagation o algoritmo de propagación inversa, fue redescubierto al mismo tiempo por varios investigadores, David Rumelhart, Geoffrey Hinton y Ronal Williams, David Parker y Yann Le Cun. El algoritmo se popularizó, cuando fue incluido en el libro *Parallel Distributed Processing Group* por los psicólogos David Rumelhart y James McClelland. La publicación de este libro trajo consigo un auge en las investigaciones con redes neuronales, siendo la Backpropagation una de las redes más ampliamente empleadas, aún en nuestros días.

Uno de los grandes avances logrados con la Backpropagation, es que esta red aprovecha la naturaleza paralela de las redes neuronales, para reducir el tiempo requerido por un procesador secuencial para determinar la correspondencia entre unos patrones dados. Además, el tiempo de desarrollo de cualquier sistema que se esté tratando de analizar, se puede reducir como consecuencia de que la red puede aprender el algoritmo correcto sin que alguien tenga que deducir por anticipado el algoritmo en cuestión.

La mayoría de los sistemas actuales de cómputo, se han diseñado para llevar a cabo funciones matemáticas y lógicas a una velocidad que resulta asombrosamente alta para el ser humano. Sin embargo, la destreza matemática no es lo que se necesita para solucionar problemas de reconocimiento de patrones en entornos ruidosos, característica que incluso dentro de un espacio de entrada relativamente pequeño, puede llegar a consumir mucho tiempo. El problema es la naturaleza secuencial del propio computador; el ciclo tomar (ejecutar de la naturaleza Von Neumann

sólo, permite que la máquina realice una operación a la vez). En la mayoría de los casos, el tiempo que necesita la máquina para llevar a cabo cada instrucción es tan breve (típicamente una millonésima de segundo), que el tiempo necesario para un programa, así sea muy grande, es insignificante para los usuarios. Sin embargo, para aquellas aplicaciones que deban explorar un gran espacio de entrada o que intentan correlacionar todas las permutaciones posibles de un conjunto de patrones muy complejo, el tiempo de computación necesario se hace bastante grande.

Lo que se necesita es un nuevo sistema de procesamiento, que sea capaz de examinar todos los patrones en paralelo. Idealmente ese sistema no tendría que ser programado explícitamente, lo que haría es adaptarse así mismo, para aprender la relación entre un conjunto de patrones dado como ejemplo y ser capaz de aplicar la misma relación a nuevos patrones de entrada. Este sistema, debe estar en capacidad de concentrarse en las características de una entrada arbitraria que se asemeje a otros patrones vistos previamente, sin que ninguna señal de ruido lo afecte. Este sistema fue el gran aporte de la red de propagación inversa, Backpropagation.

La Backpropagation es un tipo de red de aprendizaje supervisado, que emplea un ciclo de propagación (adaptación de dos fases). Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, éste se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida, se compara con la salida deseada y se calcula una señal de error para cada una de ellas.

Las salidas de error, se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente hacia ésta. Sin embargo, las neuronas de la capa oculta, sólo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total. Basándose en la señal de error percibida, se actualizan los

pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento.

La importancia de este proceso, consiste en que a medida que se entrena la red, las neuronas de las capas intermedias se organizan así mismas, de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada. Después del entrenamiento, cuando se les presente un patrón arbitrario de entrada que contenga ruido o que esté incompleto, las neuronas de la capa oculta de la red responderán con una salida activa, si la nueva entrada contiene un patrón que se asemeje a aquella característica que las neuronas individuales hayan aprendido a reconocer durante su entrenamiento. Y a la inversa, las unidades de las capas ocultas tienen una tendencia a inhibir su salida, si el patrón de entrada no contiene la característica para reconocer, para la cual han sido entrenadas.

Varias investigaciones, han demostrado que durante el proceso de entrenamiento, la red Backpropagation tiende a desarrollar relaciones internas entre neuronas con el fin de organizar los datos de entrenamiento en clases. Esta tendencia se puede extrapolar, para llegar a la hipótesis consistente en que todas las unidades de la capa oculta de una Backpropagation, son asociadas de alguna manera a características específicas del patrón de entrada como consecuencia del entrenamiento. Lo que sea o no exactamente, la asociación puede no resultar evidente para el observador humano, lo importante es que la red ha encontrado una representación interna que le permite generar las salidas deseadas cuando se le dan las entradas, en el proceso de entrenamiento. Esta misma representación interna, se puede aplicar a entradas que la red no haya visto antes, y la red clasificará estas entradas según las características que compartan con los ejemplos de entrenamiento.

3.2.2. Estructura de la red

La estructura típica de una red multicapa, se observa en la figura 3.12. Puede notarse que esta red de tres capas, equivale a tener tres redes tipo Perceptrón en cascada; la salida de la primera red, es la entrada a la segunda y la salida de la segunda, es la entrada a la tercera. Cada capa puede tener diferente número de neuronas, e incluso distinta función de transferencia.

En la figura 3.12, W^1 representa la matriz de pesos para la primera capa, W^2 los pesos de la segunda y así similarmente para todas las capas que incluya una red. Para identificar la estructura de una red multicapa, se empleará una notación abreviada, donde el número de entradas va seguido del número de neuronas en cada capa:

$$R : S^1 : S^2 : S^3 \quad (3.10)$$

Donde S representa el número de neuronas y el exponente representa la capa a la cual la neurona corresponde.

La notación de la figura 3.12, es bastante clara cuando se desea conocer la estructura detallada de la red, e identificar cada una de las conexiones, pero cuando la red es muy grande, el proceso de conexión se torna muy complejo y es bastante útil utilizar el esquema de la figura 3.13.

3.2.3. Regla de aprendizaje

El algoritmo Backpropagation para redes multicapa, es una generalización del algoritmo LMS, ambos algoritmos realizan su labor de actualización de pesos y ganancias con base en el error medio cuadrático. La red Backpropagation trabaja bajo aprendizaje supervisado y por tanto, necesita un set de entrenamiento que le describa cada salida y su valor de salida esperado de la siguiente forma:

$$\{p_1, t_1\}, \{p_2, t_2\}, \dots, \{p_Q, t_Q\} \quad (3.11)$$

Donde p_Q es una entrada a la red y t_Q es la correspondiente salida deseada para el patrón q -ésimo. El algoritmo debe ajustar los parámetros de la red para minimizar el error medio cuadrático.

El entrenamiento de una red neuronal multicapa, se realiza mediante un proceso de aprendizaje, para realizar este proceso, se debe inicialmente tener definida la topología de la red esto es: número de neuronas en la capa de entrada lo cual depende del número de componentes del vector de entrada; cantidad de capas ocultas y número de neuronas de cada una de ellas; número de neuronas en la capa de la salida el cual depende del número de componentes del vector de salida o patrones objetivo y funciones de transferencia requeridas en cada capa, con base en la topología escogida, se asignan valores iniciales a cada uno de los parámetros que conforma la red.

Es importante recalcar que no existe una técnica para determinar el número de capas ocultas, ni el número de neuronas que debe contener cada una de ellas para un problema específico, esta elección es determinada por la experiencia del diseñador, que debe cumplir con las limitaciones de tipo computacional.

Cada patrón de entrenamiento, se propaga a través de la red y sus parámetros para producir una respuesta en la capa de salida, se compara con los patrones objetivo o salidas deseadas para calcular el error en el aprendizaje, este error, marca el camino más adecuado para la actualización de los pesos y ganancias que al final del entrenamiento producirán una respuesta satisfactoria a todos los patrones de entrenamiento, esto se logra minimizando el error medio cuadrático en cada iteración del proceso de aprendizaje.

La deducción matemática de este procedimiento, se realizará para una red con una capa de entrada, una capa oculta y una capa de salida y luego se generalizará para redes que tengan más de una capa oculta.

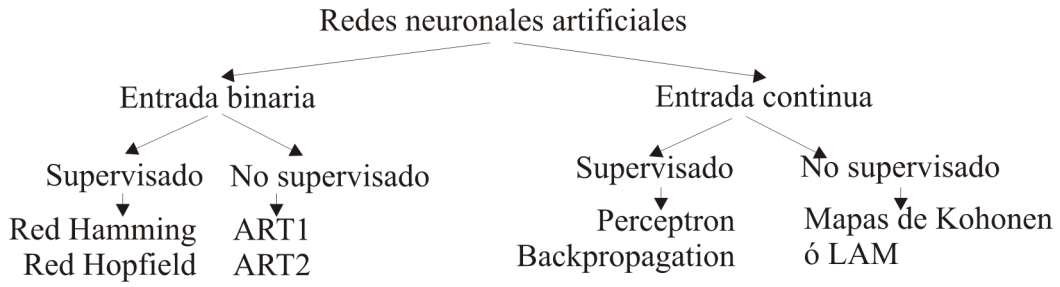


Figura 3.14: Clasificación de las RNA

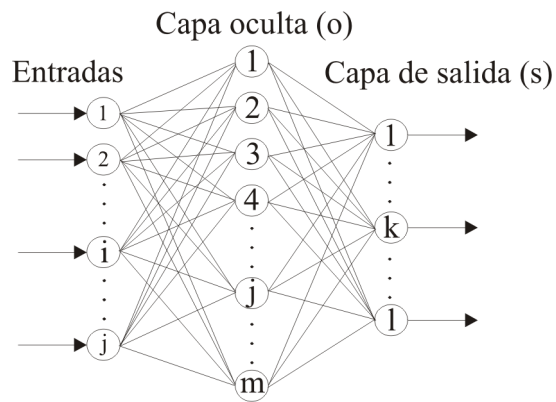


Figura 3.15: Disposición de una red sencilla de 3 capas

Es importante aclarar que la figura 3.15.

- q : Equivale al número de componentes al vector de entrada.
- m : Número de neuronas de capa oculta.
- l : Número de neuronas de la capa de salida.

Para iniciar el entrenamiento se le presenta a la red un patrón de entrenamiento, el cual tiene q componentes como se describe en la ecuación 3.12.

$$P = \left\{ \begin{array}{c} p_1 \\ p_2 \\ \vdots \\ p_i \\ \vdots \\ p_q \end{array} \right\} \quad (3.12)$$

Cuando se le presenta a la red una patrón de entrenamiento, éste se propaga a través de las conexiones existentes, produciendo una entrada neta n en cada una las neuronas de la siguiente capa, la entrada neta a la neurona j de la siguiente capa debido a la presencia de un patrón de entrenamiento en la entrada está dada por la ecuación 3.13, nótese que la entrada neta es el valor justo antes de pasar por la función de transferencia.

$$n_j^o = \sum_{i=1}^q W_{ji}^o p_i + b_j^o \quad (3.13)$$

W_{ji}^o : Peso que une la componente i de la entrada con la neurona j de la capa oculta.

p_i : Componente i del vector p que contiene el patrón de entrenamiento de q componentes.

b_j^o : Ganancia de la neurona j de la capa oculta.

Donde el superíndice (o) representa la capa a la que pertenece cada parámetro, es este caso la capa oculta.

Cada una de las neuronas de la capa oculta tiene como salida a_j^o que esta dada por la ecuación 3.14.

$$a_j^o = f^o \left(\sum_{i=1}^q W_{ji}^o p_i + b_j^o \right) \quad (3.14)$$

Donde f^o es la función de transferencia de las neuronas de la capa oculta. Las salidas a_j^o de las neuronas de la capa oculta (de l componentes), son las entradas a los pesos de conexión de la capa de salida, este comportamiento esta descrito por la ecuación 3.15.

$$n_k^s = \sum_{j=1}^m W_{kj}^s a_j^o + b_k^s \quad (3.15)$$

W_{kj}^s es el peso que une la neurona j de la capa oculta con la neurona k de la capa de salida, la cual cuenta con s neuronas, a_j^o es la salida de la neurona j de la capa oculta, la cual cuenta con m neuronas, b_k^s es la ganancia de la neurona k de la capa de salida y n_k^s es la entrada neta a la neurona k de la capa de salida.

La red produce una salida final descrita por la ecuación 3.16

$$a_k^s = f^s(n_k^s) \quad (3.16)$$

Donde f^s es la función de transferencia de las neuronas de la capa de salida.

Reemplazando 3.15 en 3.16 se obtiene la salida de la red en función de la entrada neta y de los pesos de conexión con la ultima capa oculta.

$$a_k^s = f^s \left(\sum_{j=1}^m W_{kj}^s a_j^o + b_k^s \right) \quad (3.17)$$

La salida de la red de cada neurona a_k^s se compara con la salida deseada t_k , para calcular el error en cada unidad de salida 3.18.

$$\delta = (t_k - a_k^s) \quad (3.18)$$

El error debido a cada patrón p propagado esta dado por 3.19.

$$ep^2 = \frac{1}{2} \sum_{k=1}^s (\delta_k)^2 \quad (3.19)$$

ep^2 es el error medio cuadrático para cada patrón de entrada p , δ_k es el error en la neurona k de la capa de salida con l neuronas.

Este proceso se repite para el número total de patrones de entrenamiento (r), para un proceso de aprendizaje exitoso el objetivo del algoritmo, es actualizar todos los pesos y ganancias de la red minimizando el error medio cuadrático total descrito en 3.20.

$$e^2 = \sum_{p=1}^r ep^2 \quad (3.20)$$

Donde nuevamente, e^2 es el error total en el proceso de aprendizaje en una iteración luego de haber presentado a la red los r patrones de entrenamiento.

El error que genera una red neuronal en función de sus pesos, genera un espacio de n dimensiones, donde n es el número de pesos de conexión de la red, al evaluar el gradiente del error en un punto de esta superficie, se obtendrá la dirección en la cual la función del error, tendrá un mayor crecimiento, como el objetivo del proceso de aprendizaje es minimizar el error, debe tomarse la dirección negativa del gradiente para obtener el mayor decremento del error y de esta forma su minimización, condición requerida para realizar la actualización de la matriz de pesos en el algoritmo Backpropagation:

$$W_{k+1} = W_k - \alpha \nabla ep^2 \quad (3.21)$$

El gradiente negativo de ep^2 se denotara como $-\nabla ep^2$ y se calcula como la derivada del error respecto a todos los pesos de la red.

En la capa de salida el gradiente negativo del error con respecto a los pesos es:

$$-\frac{\partial ep^2}{\partial W_{kj}^s} = -\frac{\partial}{\partial W_{kj}^s} \left(\frac{1}{2} \sum_{k=1}^l (t_k - a_k^s)^2 \right) = (t_k - a_k^s) \times \frac{\partial a_k^s}{\partial W_{kj}^s} \quad (3.22)$$

$-\frac{\partial ep^2}{\partial W_{kj}^s}$, es llamado componente del gradiente $-\nabla ep^2$ respecto al peso de la conexión de la neurona de la capa de salida y la neurona j de la capa oculta W_{kj}^s . $\frac{\partial a_k^s}{\partial W_{kj}^s}$ es la derivada de la salida de la neurona k de la capa de salida respecto, al peso W_{kj}^s .

Para calcular $\frac{\partial a_k^s}{\partial W_{kj}^s}$ se debe utilizar la regla de la cadena, pues el error no es una función explícita de los pesos de la red, de la ecuación 3.16 puede verse que la salida de la red a_k^s esta explícitamente en función de n_k^s y de la ecuación 3.15 puede verse que n_k^s esta explícitamente en función de W_{kj}^s , considerando esto se genera la ecuación 3.22.

$$\frac{\partial a_k^s}{\partial W_{kj}^s} = \frac{\partial a_k^s}{\partial n_k^s} \times \frac{\partial n_k^s}{\partial W_{kj}^s} \quad (3.23)$$

Tomando la ecuación 3.23 y reemplazándola en la ecuación 3.22 se obtiene:

$$-\frac{\partial ep^2}{\partial W_{kj}^s} = (t_k - a_k^s) \times \frac{\partial a_k^s}{\partial n_k^s} \times \frac{\partial n_k^s}{\partial W_{kj}^s} \quad (3.24)$$

$\frac{\partial n_k^s}{\partial W_{kj}^s}$ es la derivada de la entrada neta a la neurona k de la capa de salida respecto a los pesos de la conexión entre las neuronas de la capa oculta y la capa de salida, $\frac{\partial a_k^s}{\partial n_k^s}$ es la derivada de la salida de la neurona k de la capa de salida respecto a su entrada neta.

Reemplazando en la ecuación 3.24 las derivadas de las ecuaciones 3.15 y 3.16 se obtiene:

$$-\frac{\partial ep^2}{\partial W_{kj}^s} = (t_k - a_k^s) \times f'^s(n_k^s) \times a_j^o \quad (3.25)$$

Como se observa en la ecuación 3.25, las funciones de transferencia utilizadas en este tipo de red deben ser continuas para que su derivada exista en todo el intervalo, ya que el término $f^{ls}(n_k^s)$ es requerido para el cálculo del error.

Las funciones de transferencia f más utilizadas y sus respectivas derivadas son las siguientes:

$$\text{Logsig} : f(n) = \frac{1}{1 + e^{-n}}, f'(n) = f(n)(1 - f(n)), f'(n) = a(1 - a) \quad (3.26)$$

$$\text{Tansig} : f(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}}, f'(n) = 1 - (f(n))^2, f'(n) = (1 - a^2) \quad (3.27)$$

$$\text{Purelin} : f(n) = n, f'(n) = 1 \quad (3.28)$$

De la ecuación 3.25, los términos del error para las neuronas de la capa de salida están dados por la ecuación 3.29, que se le denomina comúnmente, sensibilidad de la capa de salida.

$$\delta_k = (t_k - a_k^s) f^{ls}(n_k^s) \quad (3.29)$$

Este algoritmo se denomina Backpropagation o de propagación inversa, debido a que el error se propaga de manera inversa al funcionamiento normal de la red, de esta forma, el algoritmo encuentra el error en el proceso de aprendizaje desde las capas más internas hasta llegar a la entrada; con base en el cálculo de este error, se actualizan los pesos y ganancias de cada capa.

Después de conocer 3.29, se procede a encontrar el error en la capa oculta el cual está dado por:

$$-\frac{\partial ep^2}{\partial W_{ji}^o} = -\frac{\partial}{\partial W_{ji}^o} \left(\frac{1}{2} \sum_{k=1}^l (t_k - a_k^s)^2 \right) = \sum_{k=1}^l (t_k - a_k^s) \times \frac{\partial a_k^s}{\partial W_{ji}^o} \quad (3.30)$$

Para calcular el último término de la ecuación 3.30, se debe aplicar la regla de la cadena en varias ocasiones como se observa en la ecuación 3.31, puesto que la salida de la red, no es una

función explícita de los pesos de la conexión entre la capa de entrada y la capa oculta

$$\frac{\partial a_k^s}{\partial W_{ji}^o} = \frac{\partial a_k^s}{\partial n_k^s} \times \frac{\partial n_k^s}{\partial a_k^o} \times \frac{\partial a_k^o}{\partial n_j^o} \times \frac{\partial n_j^o}{\partial W_{ji}^o} \quad (3.31)$$

Todos los términos de la ecuación 3.32 son derivados respecto a variables de las que dependen explícitamente, reemplazando 3.31 en 3.30 tenemos:

$$-\frac{\partial ep^2}{\partial W_{ji}^o} = \sum_{k=1}^l (t_k - a_k^s) \times \frac{\partial a_k^s}{\partial n_k^s} \times \frac{\partial n_k^s}{\partial a_k^o} \times \frac{\partial a_k^o}{\partial n_j^o} \times \frac{\partial n_j^o}{\partial W_{ji}^o} \quad (3.32)$$

Tomando las derivas de las ecuaciones 3.13, 3.14, 3.15, 3.16 y reemplazándolas en la ecuación 3.32, se obtiene la expresión del gradiente del error en la capa oculta

$$-\frac{\partial ep^2}{\partial W_{ji}^o} = \sum_{k=1}^l (t_k - a_k^s) \times f'^s(n_k^s) \times \mathbf{W}_{kj}^s \times f'^o(n_j^o) \times \mathbf{p}_i \quad (3.33)$$

Reemplazando la ecuación 3.29, en la ecuación 3.33 se tiene:

$$-\frac{\partial ep^2}{\partial W_{ji}^o} = \sum_{k=1}^l \delta_k \times \mathbf{W}_{kj}^s \quad (3.34)$$

Los términos del error para cada neurona de la capa oculta esta dado por la ecuación 3.34, este término también se denomina sensibilidad de la capa oculta

$$\delta_j^o = f'^o(o) \times \sum_{k=1}^l \delta_k^s \mathbf{W}_{kj}^s \quad (3.35)$$

Luego de encontrar el valor del gradiente del error, se procede a actualizar los pesos de todas las capas empezando por la de salida, para ésta la actualización de pesos y ganancias está dada por 3.36 y 3.37.

$$\mathbf{W}_{kj}(t+1) = \mathbf{W}_{kj}(t) - 2\alpha\delta_k^s \quad (3.36)$$

$$\mathbf{b}_k(t+1) = \mathbf{b}_k(t) - 2\alpha\delta_k^s \quad (3.37)$$

Donde α , es la tasa de aprendizaje que varía entre 0 y 1 dependiendo de las características del problema a solucionar.

Luego de actualizar los pesos y ganancias de la capa de salida, se procede a actualizar los pesos y ganancias de la capa oculta mediante las ecuaciones 3.38 y 3.39.

$$W_{ji}(t + \mathbf{1}) = W_{ji}(t) - 2\alpha\delta_j^o p_i \quad (3.38)$$

$$b_j(t + \mathbf{1}) = b_j(t) - 2\alpha\delta_j^o \quad (3.39)$$

Esta deducción, fue realizada para una red de tres capas, si se requiere realizar el análisis para una red con dos o más capas ocultas, las expresiones pueden derivarse de la ecuación 3.35, donde los términos que se encuentran dentro de la sumatoria, pertenecen a la capa inmediatamente superior; este algoritmo es conocido como la regla Delta Generalizada, desarrollada por Rumelhart D., ya que es una extensión de la regla delta desarrollada por Widrow en 1930.

Para algunos autores, las sensibilidades de las capas están denotadas por la letra S , reescribiendo las ecuaciones 3.29 y 3.35 con esta notación, se obtienen las ecuaciones 3.40 y 3.41.

$$\mathbf{S}^M = -2f^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) \quad (3.40)$$

$$\mathbf{s}^m \equiv f^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}, param = M-1, \dots, 2, 1 \quad (3.41)$$

En la ecuación 3.40, M representa la última capa y S^M , la sensibilidad, para esta capa la ecuación 3.41, expresa el cálculo de la sensibilidad capa por capa comenzando desde la última capa oculta. Cada uno de estos términos, involucra que el término para la sensibilidad de la capa siguiente ya esté calculado.

Como se ve el algoritmo Backpropagation, utiliza la misma técnica de aproximación en pasos descendientes que emplea el algoritmo LMS, la única complicación está en el cálculo del gradiente, pues es un término indispensable para realizar la propagación de la sensibilidad.

En las técnicas de gradiente descendiente, es conveniente avanzar por la superficie de error con incrementos pequeños de los pesos; esto se debe a que tenemos una información local de la superficie y no se sabe lo lejos o lo cerca que se está del punto mínimo, con incrementos grandes, se corre el riesgo de pasar por encima del punto mínimo, con incrementos pequeños, aunque se tarde más en llegar, se evita que esto ocurra. El elegir un incremento adecuado influye en la velocidad de convergencia del algoritmo, esta velocidad se controla a través de la tasa de aprendizaje α , la que por lo general se escoge como un número pequeño, para asegurar que la red encuentre una solución. 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.

3.3. Algoritmo de Levenberg - Marquardt

Este algoritmo, es una modificación del método de Newton, que fue diseñado para minimizar funciones que sean la suma de los cuadrados de otras funciones no lineales; es por ello, que el algoritmo de Levenberg - Marquardt, tiene un excelente desempeño en el entrenamiento de redes neuronales, donde el rendimiento de la red, esté determinado por el error medio cuadrático.

El método de Newton para optimizar el rendimiento $e(x)$ es:

$$\mathbf{X}_{k+1} = \mathbf{X}_k - \mathbf{A}_k^{-1} \mathbf{g}_k \quad (3.42)$$

$$\mathbf{A}_k = \nabla^2 \mathbf{e}(\mathbf{x})|_{x=x_k} \quad (3.43)$$

$$\mathbf{g}_k \equiv \nabla \mathbf{e}(\mathbf{x})|_{x=x_k} \quad (3.44)$$

Si asumimos que $e(x)$ es una suma de funciones cuadráticas:

$$\mathbf{e}(\mathbf{x}) = \sum_{i=1}^n v_i^2(\mathbf{x}) = \mathbf{v}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) \quad (3.45)$$

El gradiente puede ser escrito entonces en forma matricial:

$$\nabla \mathbf{e}(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) \quad (3.46)$$

Donde $J(x)$ es la matriz Jacobiana.

Ajustando el método de Newton, obtenemos el algoritmo de Levenberg - Marquardt

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k \mathbf{I}]^{-1} \mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) \quad (3.47)$$

ó

$$\Delta \mathbf{x}_k = - [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k \mathbf{I}]^{-1} \mathbf{J}^T(\mathbf{x}_k)\mathbf{v}(\mathbf{x}_k) \quad (3.48)$$

La nueva constante μ determina la tendencia el algoritmo, cuando μ_k se incrementa, este algoritmo se aproxima al algoritmo de pasos descendientes para tazas de aprendizaje muy pequeñas; cuando μ_k se decrementa este algoritmo se convierte en el método de Gauss - Newton.

El algoritmo comienza con un valor pequeño para μ_k , por lo general 0.01, si en ese paso no se alcanza el valor para $e(x)$, entonces el paso es repetido con μ_k multiplicado por un factor $v > 1$.

Si se ha escogido un valor pequeño de paso en la dirección de paso descendiente, $e(x)$ debería decrecer. Si un paso produce un pequeño valor para $e(x)$, entonces el algoritmo tiende al método de Gauss - Newton, el que se supone garantiza una rápida convergencia. Este algoritmo genera un compromiso entre la velocidad del método de Gauss-Newton y la garantía de convergencia del método de paso descendiente.

Los elementos de la matriz Jacobiana necesarios en el algoritmo de Levenberg-Marquardt son de este estilo:

$$[\mathbf{J}]_{h,l} = \frac{\partial e_{k,q}}{\partial x_l} \quad (3.49)$$

Donde x es el vector de parámetros de la red, que tiene la siguiente forma:

$$\mathbf{x}^T = [x_1, x_2, \dots, x_n] = [w_{1,1}^1, w_{1,2}^1, \dots, w_{S^1,R}^1, b_1^1, \dots, b_{S^1}^1] \quad (3.50)$$

Para utilizar este algoritmo en las aplicaciones para redes multicapa, se redefinirá el término sensibilidad de forma que sea más simple hallarlo en cada iteración.

$$s_{i,h}^m \equiv \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \quad (3.51)$$

Donde $h = (q - 1) S^M + k$.

Con la sensibilidad definida de esta manera, los términos de la matriz Jacobiana pueden ser calculados más fácilmente:

$$[\mathbf{J}]_{h,l} = \frac{\partial e_{k,q}}{\partial w_{i,j}^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \times \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = s_{i,h}^m \times \frac{\partial n_{i,q}^m}{\partial w_{i,j}^m} = s_{i,h}^m \times a_{j,q}^{m-1} \quad (3.52)$$

y para las ganancias:

$$[\mathbf{J}]_{h,l} = \frac{\partial e_{k,q}}{\partial b_i^m} = \frac{\partial e_{k,q}}{\partial n_{i,q}^m} \times \frac{\partial n_{i,q}^m}{\partial b_i^m} = s_{i,h}^m \times \frac{\partial n_{i,q}^m}{\partial b_i^m} = s_{i,h}^m \quad (3.53)$$

De esta forma, cuando la entrada p_Q ha sido aplicada a la red y su correspondiente salida a_Q^M ha sido computada, el algoritmo Backpropagation de Levenberg-Marquardt es inicializado con:

$$\mathbf{S}_q^M = -f^M(\mathbf{n}_q^M) \quad (3.54)$$

Cada columna de la matriz S_Q^M debe ser propagada inversamente a través de la red para producir una fila de la matriz Jacobiana. Las columnas pueden también ser propagadas conjuntamente de la siguiente manera:

$$\mathbf{S}_q^M = f^M(\mathbf{n}_q^M)(\mathbf{W}^{m+1})^T \mathbf{S}_q^{m+1} \quad (3.55)$$

Las matrices de sensibilidad total para cada capa en el algoritmo de Levenberg-Marquardt, son formadas por la extensión de las matrices computadas para cada entrada:

$$S^m = [S_1^m][S_2^m] \dots [S_Q^m] \quad (3.56)$$

Para cada nueva entrada que es presentada a la red, los vectores de sensibilidad son propagados hacia atrás, esto se debe a que se ha calculado cada error en forma individual, en lugar de derivar la suma al cuadrado de los errores. Para cada entrada aplicada a la red habrá S^M errores, uno por cada elemento de salida de la red y por cada error se generara una fila de la matriz Jacobiana.

3.4. Algoritmo Levenberg-Marquardt resumido

Este algoritmo puede resumirse de la siguiente manera:

1. Se presentan todas las entradas a la red, se calculan las correspondientes salidas y cada uno de los errores según:

$$\mathbf{e}_q = \mathbf{t}_q - \mathbf{a}_q^M \quad (3.57)$$

2. Se calculan las sensibilidades individuales y la matriz de sensibilidad total y con éstas, se calculan los elementos de la matriz Jacobiana.

3. Se obtiene Δx_k .
4. Se recalcula la suma de los errores cuadrados usando $x_k + \Delta x_k$. Si esta nueva suma es más pequeña que el valor calculado en el paso 1, entonces se divide μ por ν , se calcula $x_{k+1} = x_k + \Delta x_k$ y se regresa al paso 1. Si la suma no se reduce, entonces se multiplica μ por ν y se regresa al paso 3.

El algoritmo debe alcanzar convergencia cuando la norma del gradiente de

$$\nabla \mathbf{e}(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{v}(\mathbf{x}) \tag{3.58}$$

sea menor que algún valor predeterminado, o cuando la suma de los errores cuadrados ha sido reducida a un error que se haya trazado como meta.

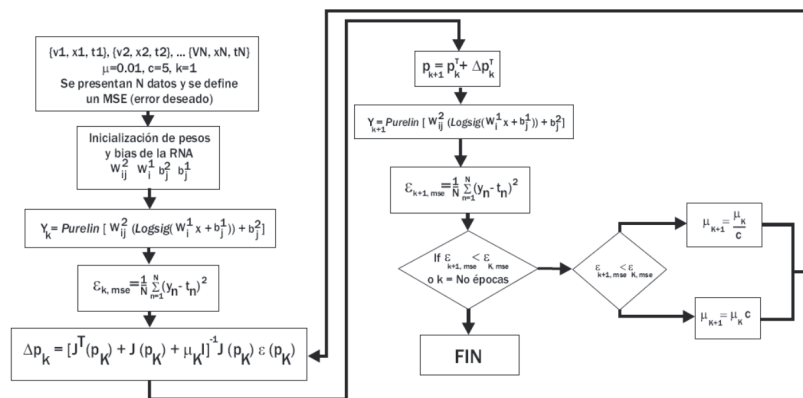


Figura 3.16: Entrenamiento Levenberg-Marquardt

3.5. Diseño de la red neuronal artificial

3.5.1. Parámetros de falla y no falla

La metodología para el entrenamiento de la RNA requiere de dos parámetros importantes. El primero de ellos, definir el rango de operación de los sensores cuando funcionan adecuadamente

según los parámetros de operación normal, para así obtener datos reales que correspondan a la operación normal. El segundo parámetro, se refiere a la definición de los sensores cuando funcionan con falla o de manera incorrecta. Para su obtención de este tipo de parámetro, fue necesaria la inducción de fallas para el conocimiento y funcionamiento durante ese instante de tiempo y para su futura adquisición de las lecturas correspondientes.

Las fallas inducidas a cada uno de los sensores fueron: 1) Desconexión general de la línea de datos de los sensores, 2) Perturbación periódica a cada lectura de sensor a diferentes instantes de tiempo. Mientras que a los actuadores, se desarrolló la instrumentación necesaria para la captura de datos de consumo de corriente en cada una de las tres líneas de los motores, para determinar si existe un desacoplamiento mecánico, una carga acoplada o un funcionamiento correcto a partir de los datos de las lecturas de consumo de corriente en cada una de las líneas de alimentación.

Las fallas inducidas para el actuador, fueron finalmente el desacoplamiento mecánico de la ventana enrollable y la inducción de una caída de fase. Dichas fallas se adquirieron y se normalizaron para el futuro aprendizaje de la RNA.

Por otro lado, el entrenamiento de la red neuronal artificial (RNA), se realizó fuera de línea (*off-line*) es decir, se generó el entrenamiento a partir de los datos capturados por el sistema de control TUNA SCCII 5.0 (Temperaturas, Humedades relativas, etc.). Una vez inducidas las fallas correspondientes y contando con un banco de datos, se genera un vector de entrenamiento así como los targets o salidas deseadas correspondientes para el entrenamiento de la RNA propuesta. Finalizado el entrenamiento, se procedera la validación de la RNA con un vector de prueba vía software.

3.5.2. Diseño del sistema

Las redes neuronales, requieren que sus datos de entradas y salidas estén normalizadas para tener el mismo orden de magnitud. La normalización suele ser crítica para la mayoría de las aplicaciones. Si las variables de entrada y salida no son del mismo orden o magnitud, algunas variables pueden aparecer teniendo más prioridad que otras. Es por ello, que el algoritmo de entrenamiento tiende a compensar diferencias entre orden y magnitud, por el ajuste de pesos.

La generación del diseño de RNA, requiere de la definición de algunas consideraciones. Primeramente, se debe realizar la normalización de datos, debido a que algunas variables contienen diferentes escalas y el uso de la normalización, conlleva al menor uso de recursos computacionales y por consiguiente, a un entrenamiento efectivo y desempeño óptimo. La normalización tiene un rango definido que para nuestra aplicación, ésta oscila entre los valores $[0, 1]$. La siguiente ecuación define la normalización de datos para cualquier tipo de sensor al que se le quiera aplicar dicha fórmula:

$$x = \frac{x' - x'_{min}}{x'_{max} - x'_{min}} \quad (3.59)$$

Donde la variable x resulta ser el valor normalizado o salida normalizada, x' es el dato a normalizar o entrada del dato y, x'_{min} y x'_{max} son el rango mínimo y máximo correspondiente del sensor a normalizar.

Vector de entrenamiento

Existen aproximaciones para el entrenamiento de una red neuronal. Pero, éstas, simplemente recaen en dos grandes grupos: aprendizaje supervisado y no supervisado. Se definirá únicamente el aprendizaje supervisado debido a que es el entrenamiento utilizado para el desarrollo de la RNA.

El aprendizaje supervisado, requiere de un maestro externo (también llamado target) para determinar el control de aprendizaje y para la incorporación de información global. El target o maestro,

puede ser un conjunto de datos (vector de entrenamiento) o un observador, el cual define el desempeño de aprendizaje.

El propósito de este tipo de aprendizaje, es el cambio de parámetros de la RNA como lo son los pesos y bias. Después de que una RNA tenga una estabilización, es decir, una relación mínima entre entradas y salidas, la tarea de aprendizaje se ha completado.

A continuación, se presenta la metodología para la obtención del vector de entrenamiento de cada una de las variables.

Primeramente, se desarrollara la normalizacion de la variable interna de temperatura. Para este tipo de variable, nos remontaremos al sensor utilizado de las siguientes características: modulo de temperatura/humedad relativa, modelo 7859 de la marca *DAVIS Instruments* con rango de operacion total de -45° a 60° C (-50° a 140° F).

Entonces de la ec. 3.59, se realiza la sustitución, donde $x'_{max} = 60^{\circ}$ C y $x'_{min} = -45^{\circ}$ C, resultan ser los valores máximos y mínimos de los datos del sensor a normalizar. Una vez normalizados los datos, se procede a definir las salidas deseadas (targets) de la RNA. El siguiente cuadro 3.2 muestra el vector de entrenamiento de la RNA de la variable temperatura interna.

El diseño de los vectores de entrenamiento, entonces resulta ser un trabajo sencillo para cada una de las variables involucradas en el control climático del invernadero. A continuación se presentan los vectores de entrenamiento de cada una de las variables faltantes correspondientes, con su sensor involucrado y características.

Temperatura externa. Para este tipo de variable, nos remontaremos nuevamente al sensor utilizado en la temperatura interna de las siguientes características: modulo de temperatura/humedad

N	Dato en °C	Dato normalizado	T1	T2	T3
n = 1	21.24	0.6309	0	0	0
n = 2	20.90	0.6274	0	0	0
n = 3	20.79	0.6266	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
n = 76	-251.3900	-1.9656	1	0	1
n = 77	-251.3300	-1.9650	1	0	1
⋮	⋮	⋮	⋮	⋮	⋮
n = 284	16.52	0.5859	0	0	0
n = 285	16.84	0.5890	0	0	0
n = 286	16.84	0.5890	0	0	0

Cuadro 3.2: Vector de entrenamiento de Temperatura Interna

relativa, modelo 7859 de la marca *DAVIS Instruments* con rango de operación total de -45° a 60°C (-50° a 140°F). El siguiente cuadro 3.3, muestra el vector de entrenamiento utilizado para el aprendizaje de la RNA.

Para la variable de humedad relativa interna y externa, se utilizó el sensor de las siguientes características: módulo de temperatura/humedad relativa, modelo 7859 de la marca *DAVIS Instruments* con rango de operación completa de 0 a 100 %. Los vectores de entrenamiento, se muestran en el cuadro 3.4 para humedad relativa interna y el cuadro 3.5, para la humedad relativa externa.

Finalmente, se presenta el cuadro 3.6, como el vector de entrenamiento utilizado en la RNA de las líneas de los motores trifásicos, y por consiguiente encontrar la falla. El sensor utilizado para este tipo de adquisición, fueron los transductores de corriente de tipo toroidal con rango de operación máximo que oscila entre los 0 a 10 A de AC, de la marca *TALEMA*. La aplicación final del sensor, es adquirir datos correspondientes al consumo de corriente por cada una de las líneas de alimentación.

N	Dato en °C	Dato normalizado	T1	T2	T3
n = 1	21.04	0.6290	0	0	0
n = 2	20.89	0.6275	0	0	0
n = 3	20.79	0.6266	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
n = 166	-251.1100	-1.9630	1	0	1
n = 167	-250.1800	-1.9541	1	0	1
⋮	⋮	⋮	⋮	⋮	⋮
n = 284	10.47	0.5283	0	0	0
n = 285	10.51	0.5287	0	0	0
n = 286	16.51	0.5287	0	0	0

Cuadro 3.3: Vector de entrenamiento de Temperatura Externa

Diseño de la red neuronal artificial

El número de neuronas de entradas y salidas, corresponden al número de entradas y salidas deseadas de la RNA respectivamente. Escoger el número de neuronas y capas para una RNA, depende directamente de la aplicación. La selección de un número de capas escondidas, es una parte crítica del diseño de una red. No existe una aproximación matemática que defina el número óptimo de capas escondidas a utilizar. Sin embargo, el número de capas escondidas puede ser escogida en base al entrenamiento presentado a la RNA usando varias configuraciones, así su selección se determinará de acuerdo a la configuración que tenga el menor número de capas escondidas y neuronas. En varias aplicaciones, el uso de una capa escondida es suficiente para la solución de problemas, pero también existen problemas donde inexorablemente el uso de RNA de dos capas o más es necesario.

La RNA utilizada en el presente trabajo para la detección de fallas y diagnóstico fue de topología feedforward (FF), la cual resulta ser de pocos elementos (en total 7 neuronas; 1 de capa de entrada, 3 de capa escondida u oculta y 3 de capa de salida) para los sensores, mientras que para los actu-

N	Dato en %	Dato normalizado	T1	T2	T3
n = 1	84.58	0.8458	0	0	0
n = 2	83.52	0.8352	0	0	0
n = 3	85.48	0.8548	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
n = 227	1065.5	10.655	1	0	1
n = 228	552.39	5.5239	1	0	1
⋮	⋮	⋮	⋮	⋮	⋮
n = 284	93.27	0.9327	0	0	0
n = 285	93.44	0.9344	0	0	0
n = 286	95.28	0.9528	0	0	0

Cuadro 3.4: Vector de entrenamiento de Humedad Relativa Interna

adores, se utilizó otra RNA FF de 6 neuronas (1 entrada, 3 en capa intermedia y 2 de salida). Las funciones de activación para ambas topologías, son las más comunes para este tipo de arquitectura (sigmoides en la capa intermedia y lineales en la capa de salida).

Comparando la topología feedforward con otras topologías como las arquitecturas de funciones de base radial (*RBF Radial Basis Function*), la topología FF es una que demanda menos recursos computacionales al momento de ser implementado, debido a su previa normalización de datos. Es así, como la topología FF trabaja con la misma efectividad como si fuera una RNA de topología RBF [Depari et al., 07].

Finalmente, se propone una RNA de topología feedforward con las siguientes características: una capa de entrada (señal del sensor normalizada), 3 neuronas de capa escondida con función de activación sigmoide y 3 neuronas de capa de salida con función de activación lineal. Durante el diseño de la RNA, varias funciones de activación se probaron como: la tangente hiperbólica en

N	Dato en %	Dato normalizado	T1	T2	T3
n = 1	83.82	0.8382	0	0	0
n = 2	78.33	0.7833	0	0	0
n = 3	78.79	0.7879	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
n = 166	1932.7	19.32	1	1	0
n = 167	1931.3	19.31	1	1	0
⋮	⋮	⋮	⋮	⋮	⋮
n = 284	92.27	0.9227	0	0	0
n = 285	90.44	0.9044	0	0	0
n = 286	89.28	0.8928	0	0	0

Cuadro 3.5: Vector de entrenamiento de Humedad Relativa Externa

la capa escondida y en la capa de salida una función sigmoide, realizando varias configuraciones de funciones de activación. Los mejores resultados en la tarea de detección y diagnóstico fueron obtenidos con las primeras funciones de activación sigmoide y lineal. La figura 3.17 presenta la RNA propuesta para la detección y diagnóstico de fallas para los sensores.

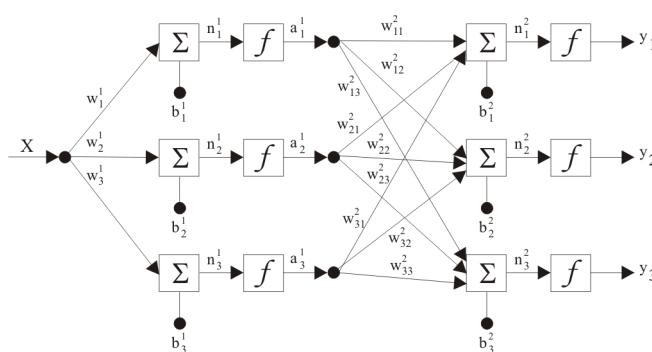


Figura 3.17: RNA propuesta para sensores

El cuadro 3.7 muestra las funciones de activación correspondientes a cada capa de la RNA.

N	Dato en A	Dato normalizado	T1	T2
n = 1	0	0	0	0
n = 2	0	0	0	0
n = 3	0	0	0	0
⋮	⋮	⋮	⋮	⋮
n = 30	1.84	0.1840	0	0
n = 31	1.84	0.1840	0	0
⋮	⋮	⋮	⋮	⋮
n = 2070	3.2373	0.3237	0	1
n = 2071	3.2373	0.3237	0	1
n = 2072	3.2373	0.3237	0	1

Cuadro 3.6: Vector de entrenamiento de las líneas de alimentación

Capa escondida	Capa de salida
$a = \frac{1}{1+e^{-n}}$	$a = n$
Sigmoide o Logsig	Lineal o Purelin

Cuadro 3.7: Funciones de activación

La figura 3.17 presenta la RNA propuesta, teniendo ésta la siguiente nomenclatura: x es la entrada del dato normalizado del sensor a analizar, $[w_1^1, w_2^1, w_3^1]$ son los pesos correspondientes de la capa escondida, $[b_1^1, b_2^1, b_3^1]$ son las salidas del primer punto de suma las cuales van dirigidas a las funciones de activación llamadas f , $[a_1^1, a_2^1, a_3^1]$ son las salidas de la capa escondidas. Los superíndices determinan la capa de la RNA, siendo ¹ la capa intermedia.

De igual manera, $[w_{11}^2, w_{12}^2, \dots, w_{22}^2, \dots, w_{32}^2, w_{33}^2]$ representan los pesos de la capa de salida de la RNA. Igualmente, $[b_1^1, b_2^1, b_3^1]$ son las ganancias del punto de suma de la capa de salida.

$[n_1^1, n_2^1, n_3^1]$ representan la salida del punto de suma y el producto de esta suma pasará por una función de activación f para su futura salida. Finalmente, $[y_1, y_2, y_3]$ resultarn ser las salidas binarias de la RNA. Las salidas binarias, representan un código binario para determinar la situación actual de la lectura analizada. Este tema será explicado más adelante.

La figura 3.18, muestra la RNA propuesta para las líneas de alimentación. Teniendo como principal característica las dos salidas, que representan un código binario para el diagnóstico actual del actuador. x nuevamente, es la entrada normalizada del sensor, $[w_1^1, w_2^1, w_3^1]$ son los pesos correspondientes de la capa escondida, $[b_1^1, b_2^1, b_3^1]$ son los bias respectivos de la capa escondida. $[n_1^1, n_2^1, n_3^1]$ son las salidas del primer punto de suma que van dirigidas a las funciones de activación sigmoidales, $[a_1^1, a_2^1, a_3^1]$ son las salidas de la capa escondida.

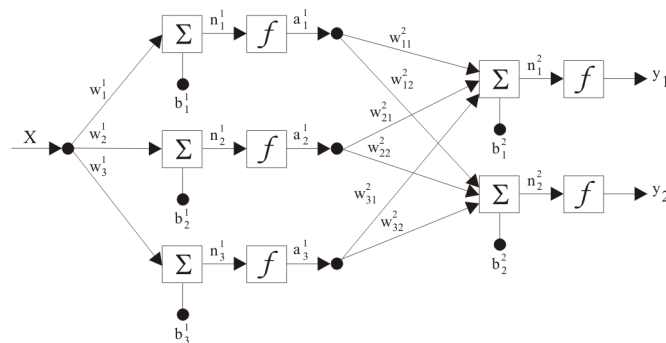


Figura 3.18: RNA propuesta para líneas de alimentación

Por otro lado, $[w_{11}^2, w_{12}^2, w_{21}^2, w_{22}^2, w_{31}^2, w_{32}^2]$ representan los pesos de la capa de salida de la RNA. Igualmente, $[b_1^1, b_2^1]$ son las ganancias del punto de suma de la capa de salida. $[n_1^1, n_2^1]$ representan la salida del punto de suma. Finalmente, $[y_1^1, y_2^1]$ resultarn ser las salidas binarias de la RNA.

Entrenamiento de RNA

De acuerdo al tipo de entrenamiento supervisado y al vector de entrenamiento generados, se dispone entonces al aprendizaje de características orientadas a la RNA. Como se mencionó en apartados anteriores, el aprendizaje de la RNA es por el método de Levenberg-Marquardt. Los cuales, se ampliará la explicación de dichos resultados en el siguiente capítulo.

3.6. Sensor transductor de corriente

3.6.1. Acondicionamiento de señal y montaje del sensor

Partiendo de la hipótesis generada, de que el consumo de corriente por cada una de las líneas de alimentación del actuador (motor trifásico), proporciona información acerca del estado del mismo. Es decir, si existe una medición de corriente, se conoce si el motor está operando de manera correcta o incorrecta.

Para la obtención de las lecturas de corriente por cada línea de alimentación, fue necesario el diseño y desarrollo de una tarjeta de acondicionamiento de señal. El sensor transductor de corriente utilizado es de la marca TALEMA, modelo AC1010, con un rango de operación de 0-10 A y un montaje que debe realizarse entre las líneas.

Para ello, se requirió de un arreglo rectificador de precisión. El cual, es un convertidor de CA a CD. La principal característica de este arreglo, es la capacidad de rectificar pequeños voltajes (mVAC) para evitar así, la caída de voltaje producida por los diodos.

La figura 3.20 muestra el diseño y construcción final de dicho circuito rectificador. Mientras que 3.19, define el diseño del circuito electrónico.

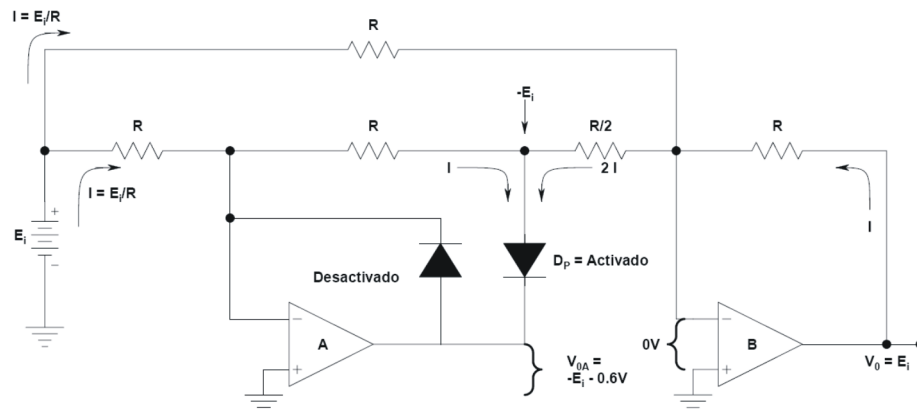


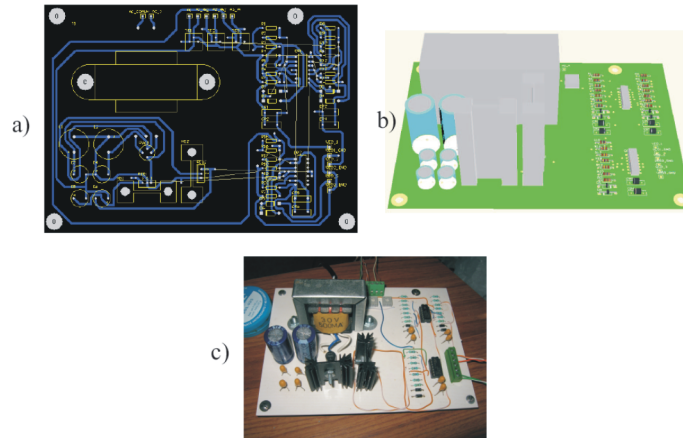
Figura 3.19: Diseño del circuito rectificador de precisión

El montaje de cada uno de los sensores transductores de corriente por cada motor fue entre las líneas de alimentación (figura 3.21). Para entonces, así realizar una adquisición de datos. Se realizó, además una caracterización por ajuste polinomial, para encontrar la relación entre el voltaje proporcionado por el acondicionamiento de señal (VDC) y la corriente consumida (Ampere) por cada una de las líneas de alimentación al motor trifásico.

3.6.2. Caracterización

El método mencionado anteriormente, para la caracterización del sensor transductor de corriente, fue el llamado ajuste polinomial con truncamiento. Dicha metodología para su caracterización, se realizó de la siguiente manera:

1. Instalación del sensor transductor de corriente en una línea con voltaje AC conocido y una resistencia de potencia conocida. Así como, de la tarjeta de acondicionamiento de señal con salida en VCD.
2. Medición de la corriente de la línea que circula por el circuito y el voltaje proporcionado por la tarjeta de acondicionamiento de señal.



a) Diseño de tarjeta de acondicionamiento de señal PROTEL, b) tarjeta de acondicionamiento 3D, c) Tarjeta física de acondicionamiento de señal

Figura 3.20: Circuito rectificador de precisión

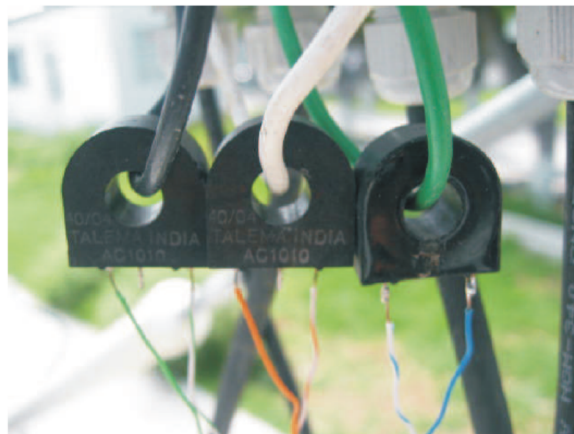


Figura 3.21: Montaje de los sensores transductores de corriente

3. Realizar diferentes lecturas y generar un tabulador, para realizar el ajuste polinomial.

La metodología antes mencionada, se puede resumir como lo muestra la siguiente figura 3.22.

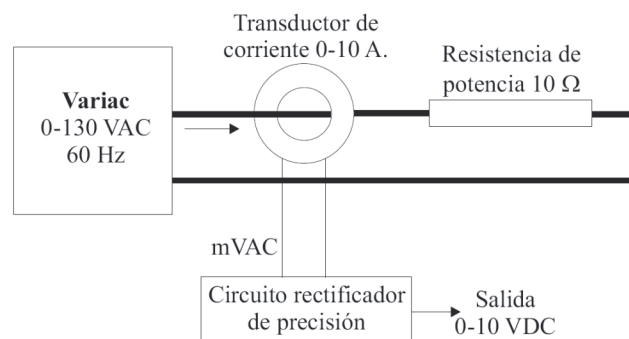


Figura 3.22: Caracterización del sensor transductor de corriente

Finalmente, realizado el tabulador con las diferentes lecturas de corriente y voltaje, se procede a la ejecución del método del ajuste polinomial. Dicha tarea, se realizó con la ayuda de una rutina programada en MATLAB, para generar la curva característica (figura 3.23) y así tener la ecuación de la curva 3.60.

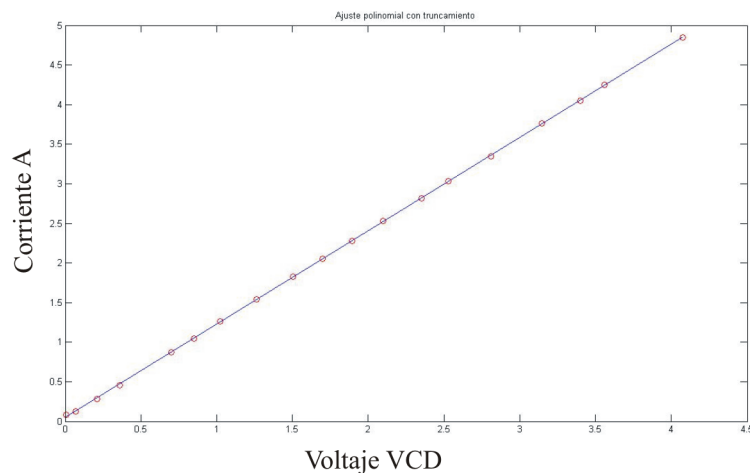


Figura 3.23: Curva de caracterización por ajuste polinomial

$$y = 0,00007629 \cdot x^2 + 1,17828369 \cdot x + 0,04866028 \quad (3.60)$$

3.7. Diseño y construcción de tarjeta de adquisición de datos con MCU (*Micro Controler Unit*)

Una vez, que se tiene conocimiento de cada elemento involucrado en el sistema de control climático, se procede al diseño de una tarjeta para la adquisición de datos.

El diseño y construcción de la tarjeta, se realizó con el MCU 18F4620 de la familia Microchip. Este dispositivo es capaz de realizar diferentes actividades que requieren del procesamiento de datos digitales, datos analógicos, control y de comunicación digital con diferentes dispositivos.

Este microcontrolador es fabricado por Microchip, familia a la cual se le denomina PIC. El modelo 18F4620, posee varias características que hacen a este microcontrolador un dispositivo muy versátil, eficiente y práctico para ser empleado en la aplicación de comunicación serial y la validación de las lecturas, que posteorimente será detallada.

Algunas de estas características se muestran a continuación:

- Soporta modo de comunicación serial, posee dos pines para ello.
- Amplia memoria de almacenamiento de datos y rutinas.
- Memoria reprogramable. La memoria en este PIC es la que se denomina FLASH; este tipo de memoria se puede borrar electrónicamente (esto corresponde a la “F” en el modelo).
- Ambiente de programación en C.

En el siguiente cuadro 3.8 se pueden observar las características más relevantes del dispositivo:

Características	18F4620
Memoria FLASH	64 K Bytes
Memoria EEPROM	1024 Bytes
Memoria RAM	3986 Bytes
Entradas/salidas	36
Velocidad máxima de operación	40 MHz
Convertidor A/D	8 canales
Resolución A/D	10 bits
Comunicación serial	UART (Universal Asynchronous Receiver-Transmitter)
Comunicación en paralelo	PSP (Paralell Slave Port)

Cuadro 3.8: Características de MCU 18F4620

Los puertos utilizados para la aplicación fueron el puerto A, utilizado para la adquisición de datos de la lectura de los sensores de temperatura y transductores de corriente (señales analógicas). Mientras que, para la lectura de los sensores de humedad relativa el puerto utilizado fué el B (señales digitales).

El diseño de la adquisición se realizó de acuerdo a la figura 3.25, la cual es una aplicación típica del microcontrolador para una comunicación serial (RS-232). De acuerdo al tipo de aplicación, se procedió al maquinado de la tarjeta y el desarrollo de ésta ayudará a la comunicación de los dispositivos involucrados. Como lo son las lecturas de los sensores, el consumo de corriente de los actuadores, la tarjeta TUNA SCCII y la PC.

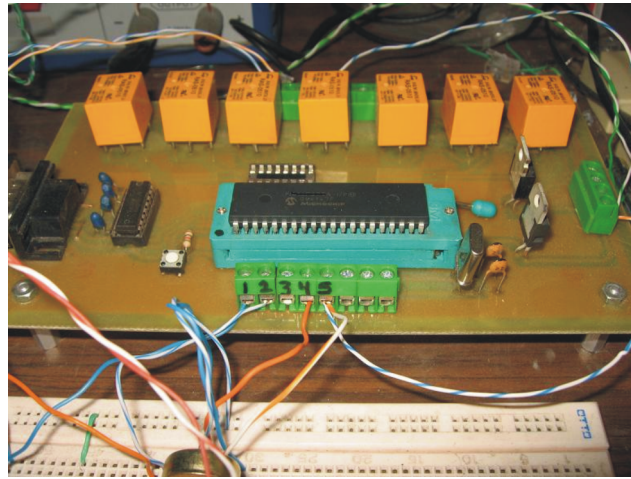


Figura 3.24: Tarjeta de adquisición con MCU 18F4620

3.8. Implementación e interfaz del sistema de detección de fallas y diagnóstico en el MCU 18F4620

Hasta ahora se ha presentado el proceso para la generación de vectores de entrenamiento requeridos así como de un entrenamiento de red neuronal, para la creación de una ecuación capaz de detectar fallas y generar salidas de diagnóstico. Además, de la generación de señales eléctricas que nos proporcionan información acerca del estado de los sensores y actuadores.

Como parte medular del trabajo, se realiza el proceso final, que es la implementación del sistema de detección de fallas en hardware. La aplicación de la red neuronal previamente entrenada off-line, es procesar la señal de entrada y obtener una respuesta correcta de acuerdo al tipo de situación que se presente.

Para corroborar el comportamiento de los algoritmos alcanzados, es necesario comparar de manera práctica los resultados reales implementados en hardware con los generados por previamente por software.

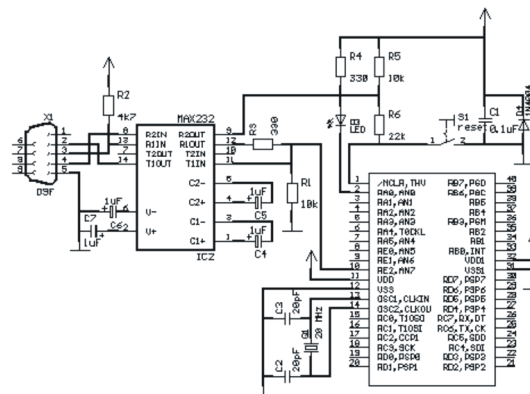


Figura 3.25: Diseño de comunicación RS-232 con MCU

El código del programa para realizar la detección de fallas bajo cualquier condición de operación, utiliza un 2 % de memoria ROM, y un 14 % de memoria RAM. El código se generó utilizando el lenguaje C bajo el compilador CCS PCWH. La idea principal del programa, consiste en la captura constante de datos a través de los puertos analógicos y digitales del MCU, para ser procesados por la red neuronal y así generar una trama extra de bits, para la validación de la lectura correspondiente. El siguiente diagrama de flujo (figura 3.26) presenta el código generado.

Donde como primer paso, después de encender el dispositivo, se comienza con la inicialización de variables (valores de pesos y bias que conforman la estructura de la red neuronal, generadas previamente off-line), inicialización de parámetros de comunicación serial y parámetros para el funcionamiento de los puertos analógicos (puerto A) y puertos digitales (puerto B).

En seguida, se genera una interrupción externa, la cual es enviada por la PC. Esta interrupción es para obtener una comunicación constante de las lecturas de los sensores en el puerto A (sensores de temperatura y transductores de corriente) y del puerto B (sensores de humedad relativa).

De acuerdo al tipo de interrupción externa, se procede a la lectura correspondiente y a su val-

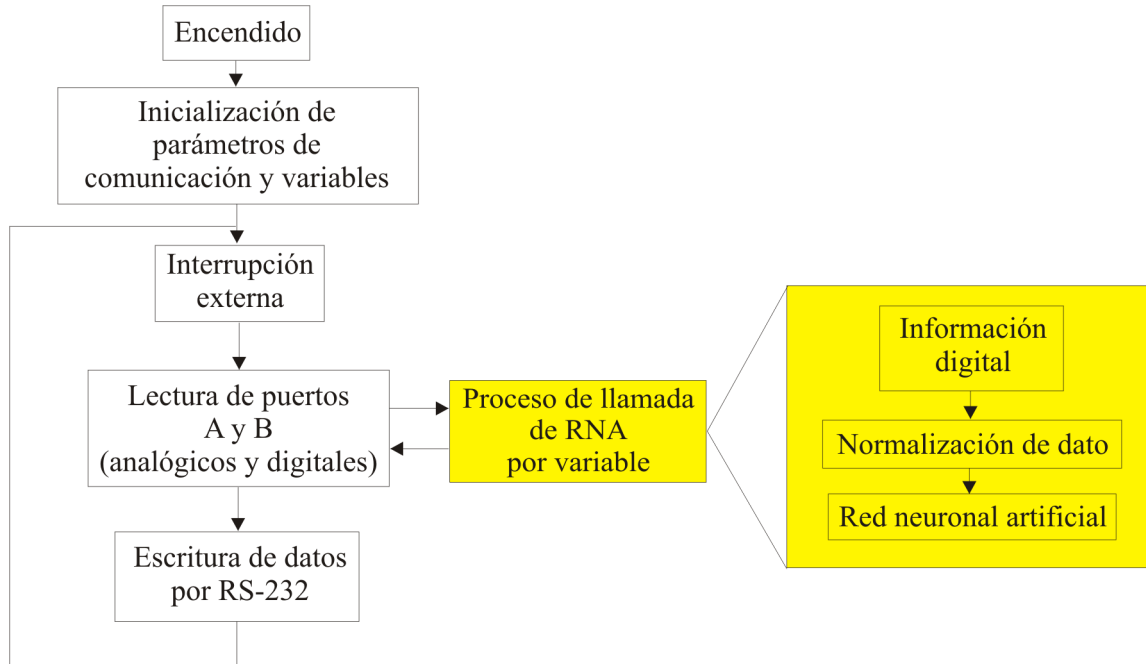


Figura 3.26: Diagrama flujo del código generado

idación con el proceso de llamada de la RNA. Internamente en el MCU, este proceso tiene 3 subprocesos, el primero de ellos es guardar la información digital en un registro, el segundo de ellos es la normalización del dato (convertir el número binario a un número decimal) y el tercero de ellos, es la aplicación de la RNA previamente establecida.

Una vez, que se termina con la aplicación de la RNA al dato leído, se procede al envío de información. Primeramente, se envía el dato leído, en tramas de 2 bytes, y después la trama correspondiente a su validación de 1 byte (salidas de la RNA correspondiente).

Debido a la comodidad de comunicación del MCU 18F4620, las lecturas de los componentes involucrados fueron transmitidas a través de la comunicación serial. Para ello, se realizó una interfaz entre la computadora y el microcontrolador, para validar y verificar el trabajo que realiza el MCU. La interfaz realizada en **BUILDER C++ 6.0** junto con la librería **APRO** (Async Profes-

sional, componente de Builder que provee facilidades de comunicación serial), únicamente realiza el monitoreo de la rutina programada por el MCU y de la adquisición de lectura de los sensores, además del verificar el estatus de los actuadores.

La principal tarea del MCU es la de validación de cada una de las lecturas adquiridas, con la ayuda de la RNA programada internamente. Por lo tanto, el sistema ahora se le denomina sistema de detección de fallas y diagnóstico de sensores y actuadores, por su tarea de validación de en las lecturas.

Durante el proceso de validación, el MCU 18F4620 añadirá a la trama de comunicación, bits extras. Estos bits, son las salidas en código binario de cada red neuronal correspondiente a su variable. Por lo cual, hace que el sistema de detección de fallas y diagnóstico sea de fácil implementación, de acuerdo a su versatilidad de implementación y validación de las señales.

A continuación, se presenta la interfaz realizada en BUILDER C++ 6.0 (figura 3.27) y el diagrama a bloques de la implementación en hardware (figura 3.28), del sistema de detección de fallas y diagnóstico. Los resultados de de las pruebas realizadas en software y hardware, se presentan en el siguiente capítulo.

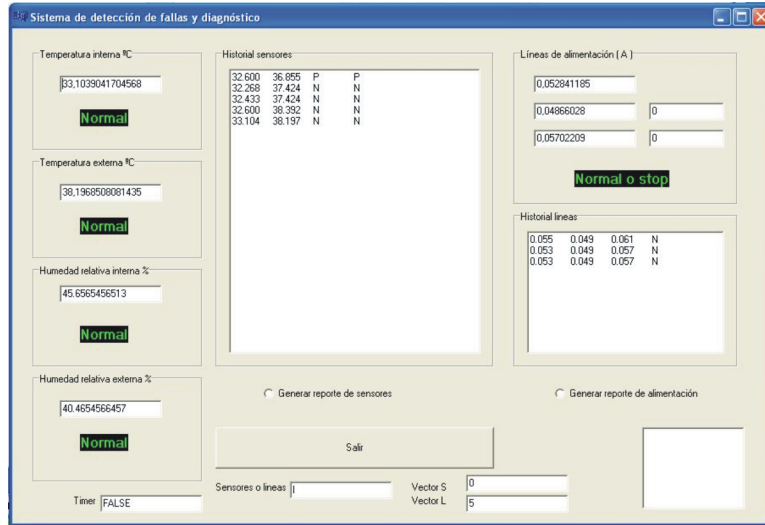


Figura 3.27: Interfaz entre microprocesador y computadora

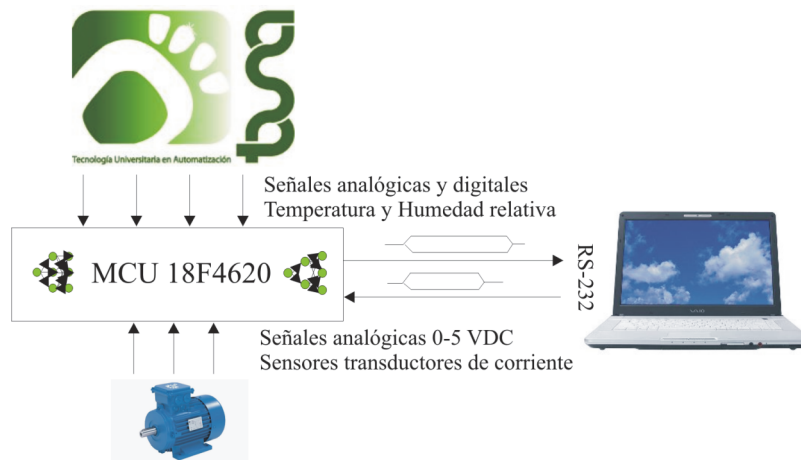


Figura 3.28: Diagrama general de implementación del SDFD

Capítulo 4

Análisis de Resultados

4.1. Ecuaciones y validación por software del sistema de detección de fallas y diagnóstico

Las redes neuronales, pueden ser representadas de forma matemática por los pesos y bias correspondientes de cada una de las capas, cuando el entrenamiento ha concluido. Esta representación matemática, resuelve el problema de detectar anomalías en las lecturas de los sensores, así como de supuestos mal funcionamientos por parte de los actuadores. La ecuación 4.1, es la forma matemática general de la expresar la RNA utilizada para la detección y diagnóstico de fallas de los sensores. Durante el desarrollo de este capítulo, se mostrará la ecuación matemática de RNA utilizada para las líneas de alimentación.

$$\mathbf{Y} = P_{urelin} [\mathbf{W}_{ij}^2 (Logsig (\mathbf{W}_i^1 \cdot x + \mathbf{b}_i^1)) + \mathbf{b}_j^2] \quad (4.1)$$

Donde de la ecuación 4.1, \mathbf{Y} representa las salidas de la red neuronal general en forma matricial con dimensión $[3 \times 1]$. De igual manera, W_{ij}^2 representan los pesos entre neuronas de capa de salida en forma matricial, con una dimensión de $[3 \times 3]$. b_j^2 es la forma de representar matricialmente los bias de la capa de salida con dimensión de $[3 \times 1]$. Por último, se tienen que W_i^1 y b_i^1

representan las matrices de pesos y bias, ambas con dimensión de $[3 \times 1]$. Finalmente, *Purelin* y *Logsig* representan las funciones de activación de capas de salida e intermedia respectivamente. x es la variable a analizar o entrada de la RNA.

La ecuación 4.2, representa la ecuación 4.1 de forma expandida, para demostrar la interconectividad entre neuronas y bias, y posteriormente introducir los valores para cada variable o sensor a analizar.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{Purelin} \left[\begin{bmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \\ w_{31}^2 & w_{32}^2 & w_{33}^2 \end{bmatrix} \left[\text{Logsig} \begin{bmatrix} w_1^1 \\ w_2^1 \\ w_3^1 \end{bmatrix} x + \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \end{bmatrix} \right] + \begin{bmatrix} b_1^2 \\ b_2^2 \\ b_3^2 \end{bmatrix} \right] \quad (4.2)$$

Durante el aprendizaje de la RNA, las variables de la ecuación 4.2 (pesos y bias) empiezan a variar de hasta que se converge a un error deseado o hasta que exista una limitante durante las iteraciones. Teniendo como resultado las siguientes ecuaciones de red neuronal listas para ser validadas por software que a continuación se presentan.

4.1.1. RNA Temperatura interna

De acuerdo al entrenamiento off-line previamente realizado, la ecuación parametrizada 4.3 de la variable de temperatura interna se presenta.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{Purelin} \left[\begin{bmatrix} 3,0940 & -0,9996 & 0,0196 \\ 0,0003 & 0,0000 & 1,0093 \\ 3,0937 & -0,9996 & -0,9897 \end{bmatrix} \left[\text{Logsig} \begin{bmatrix} 0,7989 \\ 64,8877 \\ -4,3474 \end{bmatrix} x + \begin{bmatrix} -4,4492 \\ -11,6445 \\ -13,5224 \end{bmatrix} \right] + \begin{bmatrix} 0,9794 \\ -0,0000 \\ 0,9794 \end{bmatrix} \right] \quad (4.3)$$

Al presentarle a la red neuronal, nuevamente una serie de datos reales de la variable temperatura interna a la ecuación 4.3, resulta ser que tiene un excelente desempeño para la detección (desconexión y lectura anómala) y diagnóstico. Además de diferenciar entre las lecturas erróneas

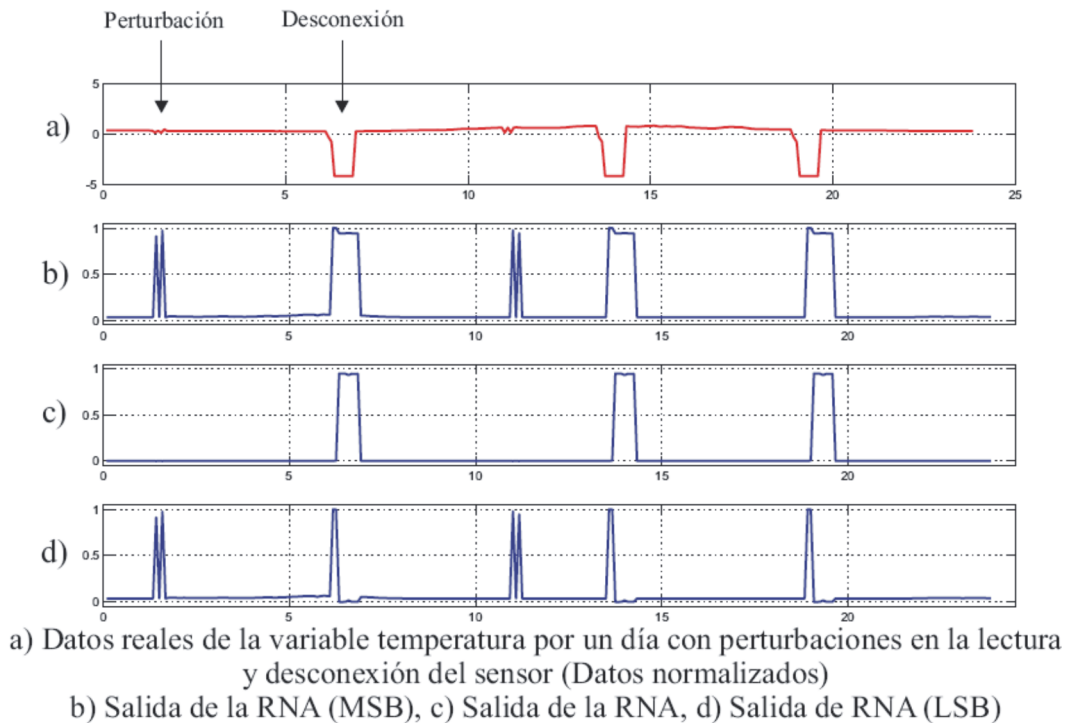


Figura 4.1: Respuesta de RNA para la variable temperatura interna

de las lecturas óptimas. La figura 4.1, demuestra el desempeño por software de la RNA propuesta para la detección de fallas para la variable de temperatura interna.

La figura 4.1, demuestra la efectividad de la RNA para la detección de fallas. La salidas de la RNA resultan ser un código binario; de acuerdo a la salida de la RNA, ésta se podrá codificar para diagnosticar la lectura del sensor. Además, de ser una información extra a la trama de datos correspondiente al sensor de temperatura interna, para determinar el estado del sensor. Por ejemplo, de la figura 4.1a, se tienen dos posibles condiciones: desconexión y perturbación.

Así entonces, se demuestra la efectividad que se tiene al detectar y diagnosticar las posibles fallas de la variable temperatura interna. El cuadro 4.2, muestran las posibles salidas de la red neuronal que posteriormente mediante la implementación en hardware se validarán las lecturas correspondientes. El cuadro 4.1, muestra la convergencia del error así como las épocas necesarias para el entrenamiento de esta RNA.

Entrenamiento	Épocas	Error (MSE)
TRAINLM	0/50	0.545254/0
TRAINLM	25/50	0.0319069/0
TRAINLM	50/50	0.0180461/0

Cuadro 4.1: Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de temperatura interna

Condiciones	Salida 1	Salida 2	Salida 3	Salida HEX
Normal	0	0	0	0
Desconexión	1	1	0	6
Perturbación	1	0	1	5

Cuadro 4.2: Salidas en código binario de RNA de temperatura interna

4.1.2. RNA Temperatura externa

De acuerdo a la ecuación general para los sensores 4.2 y al entrenamiento previo, se tiene una nueva ecuación 4.4 de red neuronal para variable de temperatura externa.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{Purelin} \left[\begin{bmatrix} 2,7835 & -0,9719 & -0,0042 \\ -0,0001 & 0,0017 & 1,0764 \\ 2,7833 & -0,9736 & -1,0806 \end{bmatrix} \left[\text{Logsig} \begin{bmatrix} 25,6155 \\ 116,4129 \\ -4,3805 \end{bmatrix} x + \begin{bmatrix} -15,7515 \\ -14,9830 \\ -15,8646 \end{bmatrix} \right] + \begin{bmatrix} 1,0038 \\ -0,0017 \\ 1,0055 \end{bmatrix} \right] \tag{4.4}$$

La ecuación 4.4 es la representación matemática de la RNA para la detección y diagnóstico de fallas para la variable de temperatura externa. La figura 4.2, demuestra de igual manera la capacidad de detectar los errores.

De manera similar, se presenta el cuadro 4.4 de código binario, para validación de las salidas de la RNA con las salidas deseadas a partir del vector de entrenamiento. Además el cuadro 4.3, muestra de nueva cuenta el entrenamiento llevado a cabo.

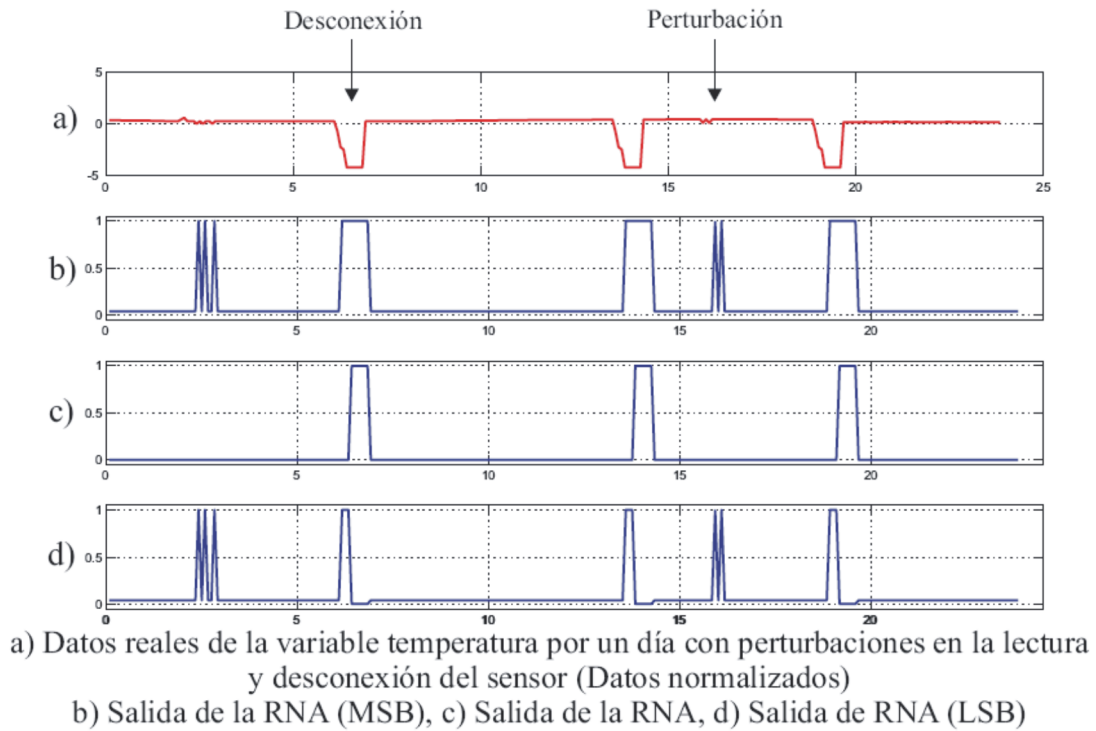


Figura 4.2: Respuesta de RNA para la variable temperatura externa

4.1.3. RNA Humedad relativa interna

La ecuación matemática de RNA para la humedad relativa interna se presenta a continuación. Esta nueva ecuación es representada por la ecuación 4.5.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{Purelin} \left[\begin{bmatrix} -0,9743 & 0,0053 & -0,9813 \\ 0,0048 & -2,1621 & -0,0000 \\ -0,9791 & 2,1673 & -0,9812 \end{bmatrix} \left[\text{Logsig} \begin{bmatrix} -3,9206 \\ -0,6107 \\ 27,9233 \end{bmatrix} x + \begin{bmatrix} 16,3674 \\ 11,9421 \\ -6,0897 \end{bmatrix} \right] + \begin{bmatrix} 1,9784 \\ 2,1573 \\ -0,1788 \end{bmatrix} \right] \quad (4.5)$$

De igual forma, se presenta la validación por simulación de esta RNA de la variable humedad relativa interna. Como dato importante, al momento de la realizar la desconexión del sensor, el dato adquirido en ese instante de tiempo, presenta una desviación violenta al momento de la lectura (por arriba de los 1000 % de humedad relativa).

Entrenamiento	Épocas	Error (MSE)
TRAINLM	0/20	1.55822/0
TRAINLM	20/20	0.0223629/0

Cuadro 4.3: Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de temperatura externa

Condiciones	Salida 1	Salida 2	Salida 3	Salida HEX
Normal	0	0	0	0
Desconexión	1	1	0	6
Perturbación	1	0	1	5

Cuadro 4.4: Salidas en código binario de RNA de temperatura externa

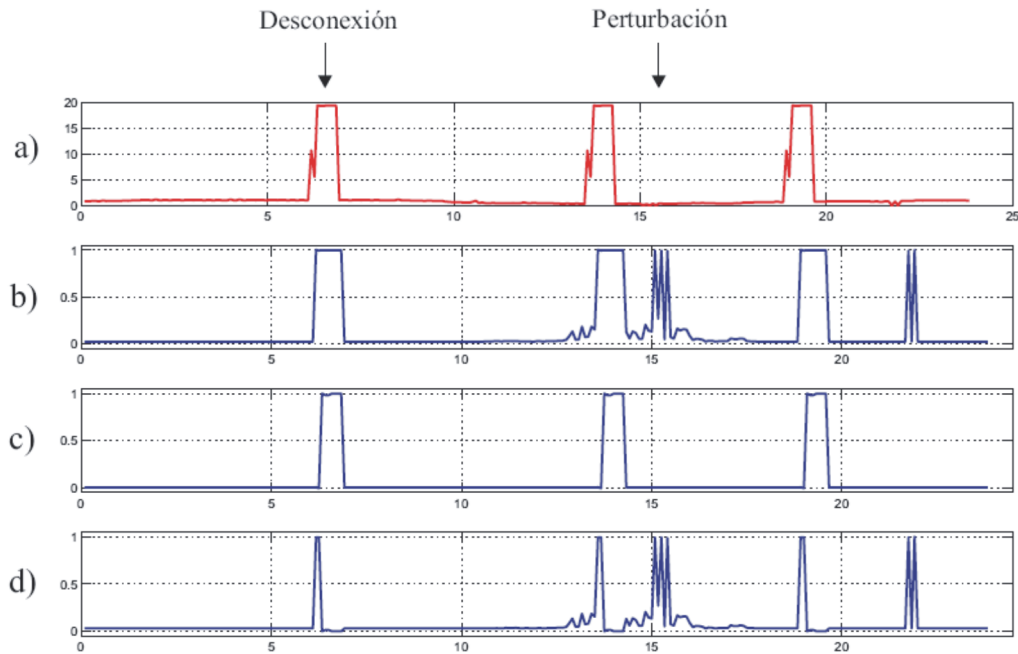
La figura 4.3 a), muestra el cambio abrupto antes mencionado, al instante de la desconexión. La RNA es capaz de encontrar esa falla gracias al buen diseño del vector de entrenamiento. El cuadro 4.6, es la representación de código binario que se desea de la RNA en el sensor de humedad relativa interna. El cuadro 4.5, muestra la convergencia del error y número de épocas para llevar a cabo el entrenamiento.

4.1.4. RNA Humedad relativa externa

Finalmente, dentro de las ecuaciones de redes neuronales para los sensores, se presenta la última ecuación representado por la ecuación 4.6. Nuevamente, se presenta la misma situación de cambio abrupto y se presenta la ecuación, validación y código binario.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{Purelin} \left[\begin{bmatrix} -0,4683 & -1,4297 & 26,4761 \\ 1,0384 & 0,0384 & 0,0000 \\ -1,5067 & -1,4681 & 26,4761 \end{bmatrix} \left[\text{Logsig} \begin{bmatrix} 2,4764 \\ -4,1939 \\ -86,5292 \end{bmatrix} x + \begin{bmatrix} -16,3346 \\ 20,7348 \\ 29,8273 \end{bmatrix} \right] + \begin{bmatrix} 1,4683 \\ -0,0384 \\ 1,5067 \end{bmatrix} \right] \tag{4.6}$$

Se presenta a continuación la validación por software de las posibles fallas del sensor de humedad relativa externa mostrada por la figura 4.4. Debe de notarse, que las arquitecturas de



a) Datos reales de la variable humedad relativa por un día con perturbaciones en la lectura y desconexión del sensor (Datos normalizados)
 b) Salida de la RNA (MSB), c) Salida de la RNA, d) Salida de RNA (LSB)

Figura 4.3: Respuesta de RNA para la variable humedad relativa interna

redes neuronales propuestas fueron efectivas en el proceso de detección de fallas y diagnóstico. Cabe mencionar, que las redes propuestas son de fácil implementación y de bajo recurso computacional al momento de ser implementadas en software. Es por ello, que el siguiente paso es la implementación en hardware.

El cuadro 4.8, muestra las salidas de las ecuaciones en código binario de la variable de humedad relativa externa. Mientras que el cuadro 4.7, muestra el entrenamiento y convergencia del error.

Posteriormente, las ecuaciones validadas de las redes neuronales de los sensores, se implementarán en hardware (microcontrolador 18F4620) para la validación de cada lectura. Es decir, a partir de las salidas de las redes, se realizará una operación de corrección en el control climático del in-

Entrenamiento	Épocas	Error (MSE)
TRAINLM	0/50	0.867626/0
TRAINLM	25/50	0.0326529/0
TRAINLM	50/50	0.0319683/0

Cuadro 4.5: Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de humedad relativa interna

Condiciones	Salida 1	Salida 2	Salida 3	Salida HEX
Normal	0	0	0	0
Desconexión	1	1	0	6
Perturbación	1	0	1	5

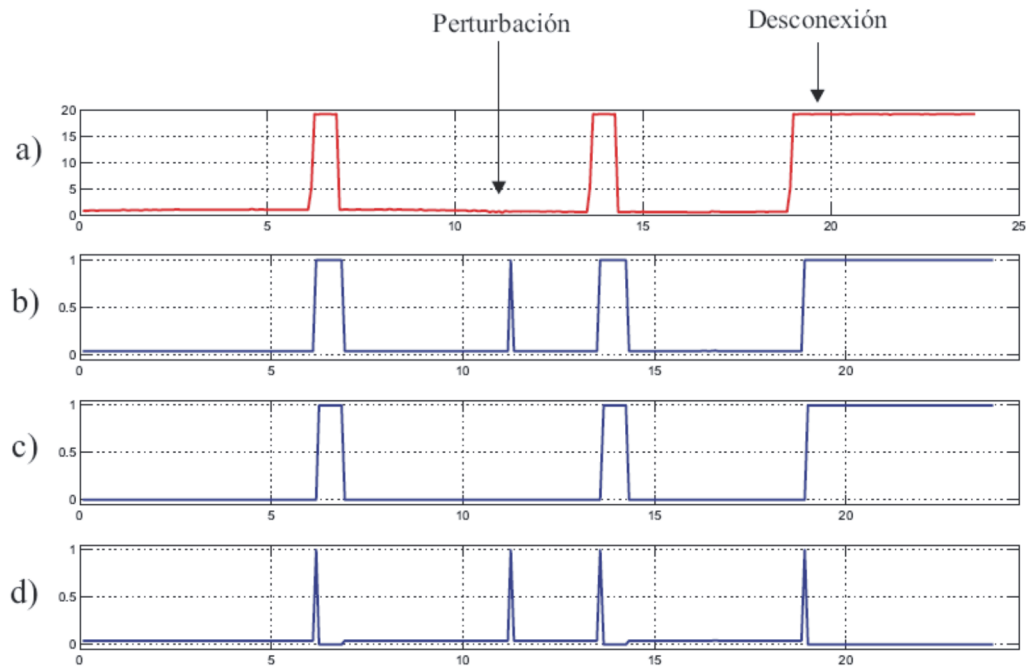
Cuadro 4.6: Salidas en código binario de RNA de humedad relativa interna

vernadero.

4.1.5. RNA líneas de alimentación del motor trifásico

Hasta ahora, se ha propuesto una topología única de red neuronal para la detección de fallas y diagnóstico por cada uno de los sensores involucrados (temperatura interna y externa, y humedad relativa interna y externa). Ahora, como se mencionó en el capítulo anterior, se propone una topología de RNA para la detección de fallas de los actuadores a partir de la lectura de corriente de cada línea de alimentación.

A continuación, se presenta la ecuación matemática general de la detección de fallas expresada por 4.7, y la ecuación desglosada representada por 4.8 de la RNA de las líneas de alimentación de los motores trifásicos.



a) Datos reales de la variable humedad relativa externa por un día con perturbaciones en la lectura y desconexión del sensor (Datos normalizados)
 b) Salida de la RNA (MSB), c) Salida de la RNA, d) Salida de RNA (LSB)

Figura 4.4: Respuesta de RNA para la variable humedad relativa externa

$$\mathbf{Y} = Purelin \left[\mathbf{W}_{ij}^2 \left(Logsig \left(\mathbf{W}_i^1 \cdot x + \mathbf{b}_i^1 \right) \right) + \mathbf{b}_j^2 \right] \quad (4.7)$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = Purelin \left[\begin{bmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \end{bmatrix} \left[Logsig \begin{bmatrix} w_1^1 \\ w_2^1 \\ w_3^1 \end{bmatrix} x + \begin{bmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \end{bmatrix} \right] + \begin{bmatrix} b_1^2 \\ b_2^2 \end{bmatrix} \right] \quad (4.8)$$

La validación del comportamiento de la red neuronal se probará con datos reales, para asegurar la detección.

Previo entrenamiento y con una convergencia de error cercano al deseado, se tienen pues las siguientes variables de la RNA. La realización de este tipo de entrenamiento fué a partir de las lec-

Entrenamiento	Épocas	Error (MSE)
TRAINLM	0/50	0.849604/0
TRAINLM	25/50	0.017972/0
TRAINLM	50/50	0.0179274/0

Cuadro 4.7: Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de humedad relativa externa

Condiciones	Salida 1	Salida 2	Salida 3	Salida HEX
Normal	0	0	0	0
Desconexión	1	1	0	6
Perturbación	1	0	1	5

Cuadro 4.8: Salidas en código binario de RNA de humedad relativa externa

turas capturadas por el movimiento de ventanas laterales (subida, bajada, desacoplamiento mecánico y caída de fase). Obteniendo la siguiente ecuación 4.9.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \text{Purelin} \left[\begin{bmatrix} -1,0029 & -0,0006 & -0,0005 \\ -0,0010 & -18,9929 & -18,9919 \end{bmatrix} \left[\text{Logsig} \left[\begin{bmatrix} -38,6225 \\ 265,0987 \\ -257,3323 \end{bmatrix} x + \begin{bmatrix} 17,0592 \\ -39,5316 \\ 38,1639 \end{bmatrix} \right] \right] + \begin{bmatrix} 1,0034 \\ 18,9929 \end{bmatrix} \right] \tag{4.9}$$

Por último, se procede a la validación de la respuesta de la red neuronal con respecto a datos obtenidos a partir de una adquisición de datos con el microcontrolador 18F4620. La validación se presenta en la figura 4.5, además de presentar el cuadro 4.10 relacionado con las salidas binarias de la RNA para las líneas de alimentación.

Las ecuaciones mostradas, fueron implementadas dentro del MCU 18F4620. La ventaja de tener un sistema de detección de fallas embebido en un microprocesador, es que se puede acoplar al sistema de manera paralela o serie. Es decir, por cada lectura que exista en cualquier sistema donde se trabaje con comunicación serial y con lecturas analógicas y digitales, la misma lectura será leída

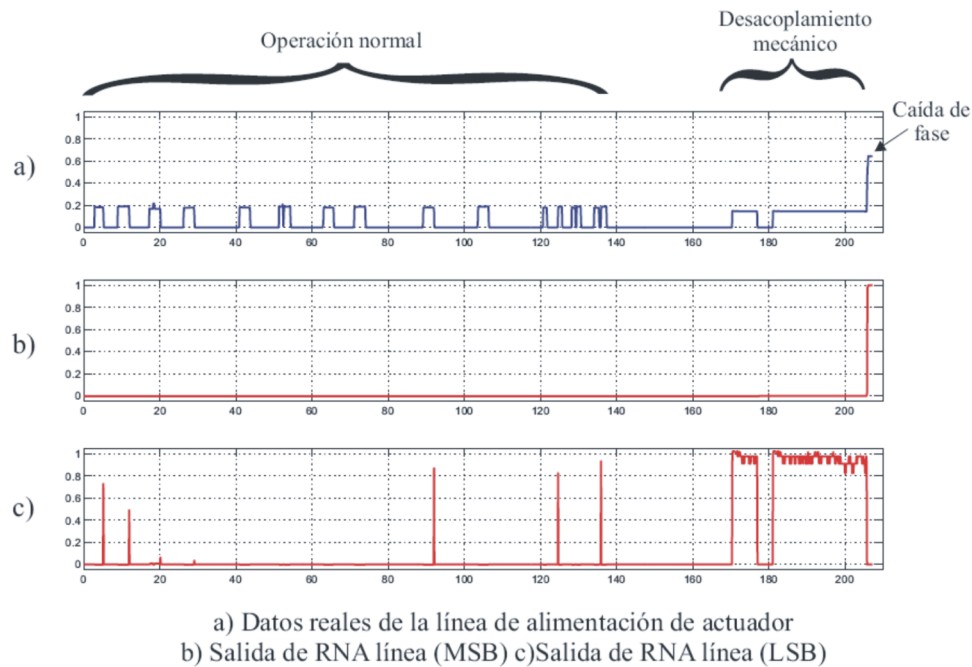


Figura 4.5: Respuesta de RNA para una línea de alimentación del motor trifásico

por el MCU para realizar el proceso de la validación. Si existe entonces, una falla, el microprocesador tomará acciones correctivas de acuerdo al tipo de falla. En el siguiente apartado, se presentan los resultados de la implementación del sistema de detección de fallas al sistema TUNA SCCII.

4.2. Implementación del sistema de detección de fallas y diagnóstico con el sistema TUNA SCCII

Una vez que se cuentan con las ecuaciones de RNA de cada una de las variables, así como las de cada actuador, la implementación resulta ser sencilla. El sistema de detección fué acoplado al sistema TUNA SCCII como lo muestra la figura 4.6.

Una vez que se tiene el acoplamiento físico del sistema de detección de fallas, se procede a

Entrenamiento	Épocas	Error (MSE)
TRAINLM	0/90	0.120917/0
TRAINLM	25/90	0.00468195/0
TRAINLM	50/90	0.00230009/0
TRAINLM	75/90	0.000178948/0
TRAINLM	90/90	0.000178938/0

Cuadro 4.9: Entrenamiento Levenberg-Marquardt (TRAINLM) y épocas de entrenamiento de líneas de alimentación

Condiciones	Salida 1	Salida 2	Salida HEX
Normal	0	0	0
Desacoplamiento mecánico	0	1	1
Caída de fase	1	0	2

Cuadro 4.10: Salidas en código binario de RNA de la línea de alimentación del motor trifásico

la adquisición de datos de cada sensor y actuador. Por cada lectura de sensor o actuador, ésta se normalizará por medio de una rutina previamente programada dentro del microprocesador. Y a su vez, la lectura será validada por la ecuación de su respectiva red neuronal. Para finalmente, enviar de manera serial la información correspondiente de su lectura.

En caso de la existencia de una falla, el MCU añadirá a la transmisión serial, las salidas binarias por cada red neuronal. De acuerdo a la trama añadida, se realizará una acción de control predefinida.

Cada vez, que existe una falla, la interfaz proporcionará un posible diagnóstico de la falla. La siguiente figura 4.7 muestra finalmente, el desempeño del sistema de detección de fallas acoplado.

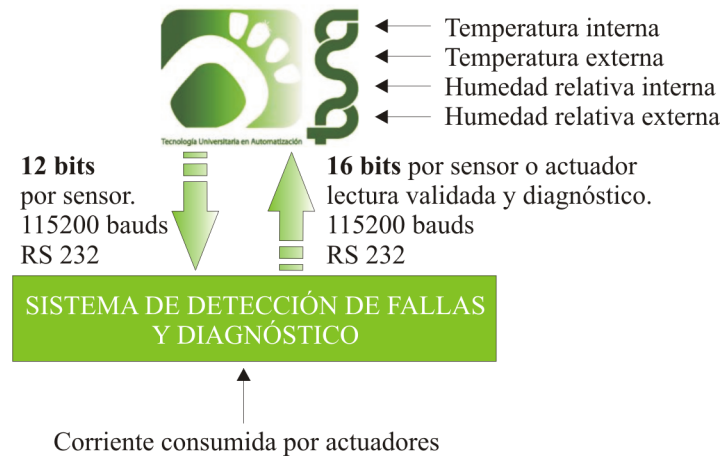


Figura 4.6: Acoplamiento del sistema de detección de fallas al sistema TUNA SCCII

4.3. Prueba de respuesta de RNA en MCU

La interfaz realizada en BUILDER 6.0, permite guardar los datos obtenidos por el MCU a archivos. Por lo que, se realizaron diferentes pruebas para verificar la efectividad que tiene la RNA implementada en el MCU.

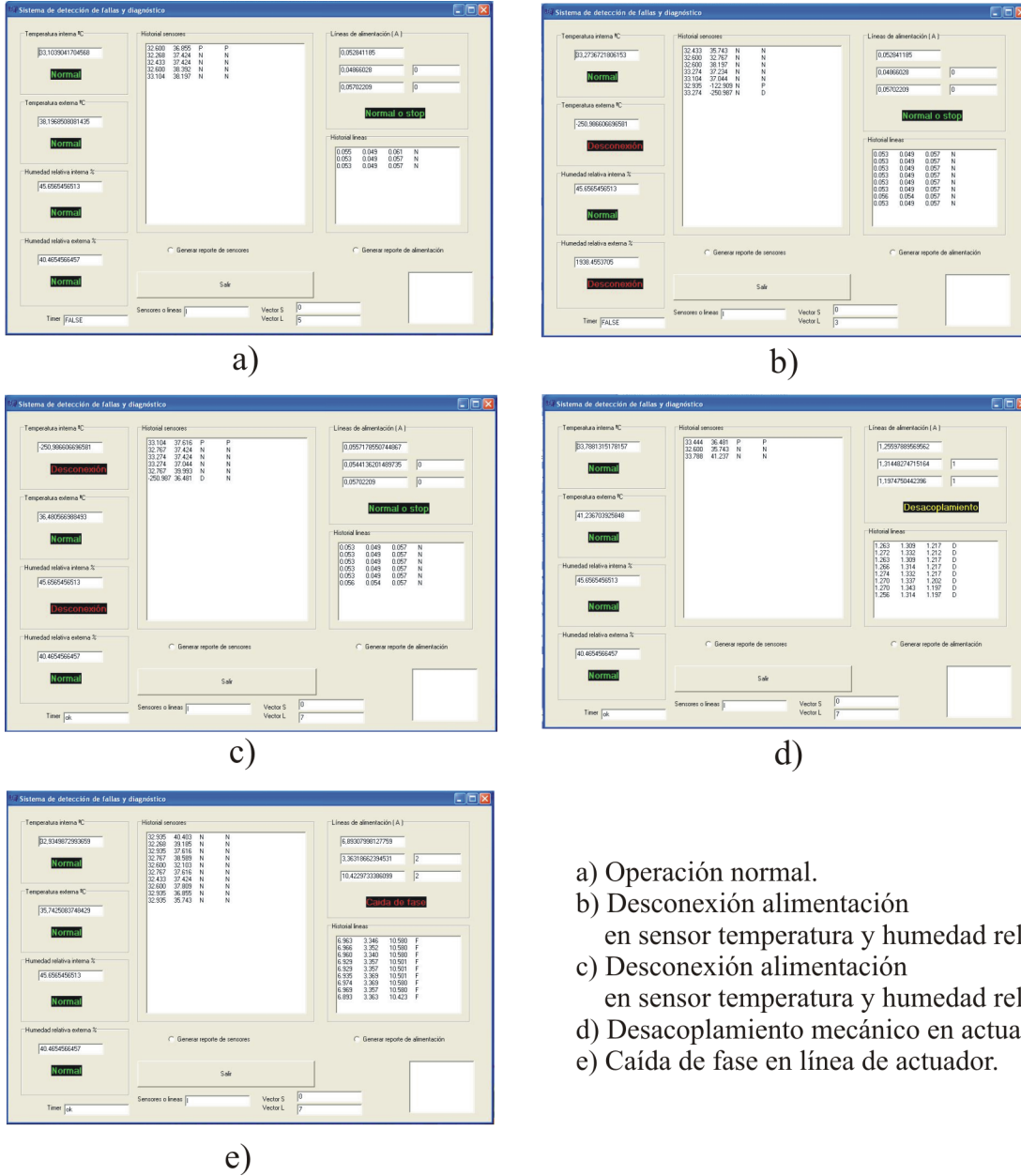
Se presentan entonces, las pruebas realizadas a los sensores de temperatura, tanto interna como externa. Se indujeron fallas como la desconexión del sensor así como de perturbación. Los resultados de inducir dichas fallas, se presentan a continuación (figura 4.8 y 4.9), junto con su diagnóstico respectivo.

La figura 4.8, muestra el monitoreo realizado desde las 10 AM hasta las 16 PM del sensor de temperatura interna. Durante este periodo, se realizaron diferentes pruebas y se indujeron perturbaciones a la lectura y desconexión del sensor. En el momento de realizar una perturbación, la lectura es adquirida por el MCU y a su vez es validada por la RNA. Una vez realizada dicha validación, la salida de la RNA resulta ser un 6H, indicando que existe dicha falla. En cambio, en el momento de realizar una desconexión, la RNA envía como información un 5H indicando efectivamente la falla.

De igual manera, analizando el sensor de temperatura externa, se llega a las mismas conclusiones que el sensor de temperatura interna. La RNA resulta ser efectiva en la tarea de detección de fallas.

A continuación se presenta la prueba de RNA en el consumo de corriente de las líneas de alimentación. Las pruebas realizadas fueron: desacoplamiento mecánico y caída de fase. La corriente respectiva por cada una de las pruebas inducidas, resulta ser diferente. Al momento de realizar dicha validación, la RNA demuestra nuevamente su efectividad. Durante el desacoplamiento mecánico, la RNA envía como información un 1H indicando dicha falla. Mientras que en una caída de fase, envía la RNA un 2H.

Es de destacar, que la red neuronal implementada en el MCU, tiene una respuesta rápida y su diagnóstico resulta ser efectivo, como se muestra a continuación el figura 4.10.



- a) Operación normal.
- b) Desconexión alimentación en sensor temperatura y humedad relativa.
- c) Desconexión alimentación en sensor temperatura y humedad relativa.
- d) Desacoplamiento mecánico en actuador.
- e) Caída de fase en línea de actuador.

Figura 4.7: Detectando fallas y diagnosticando

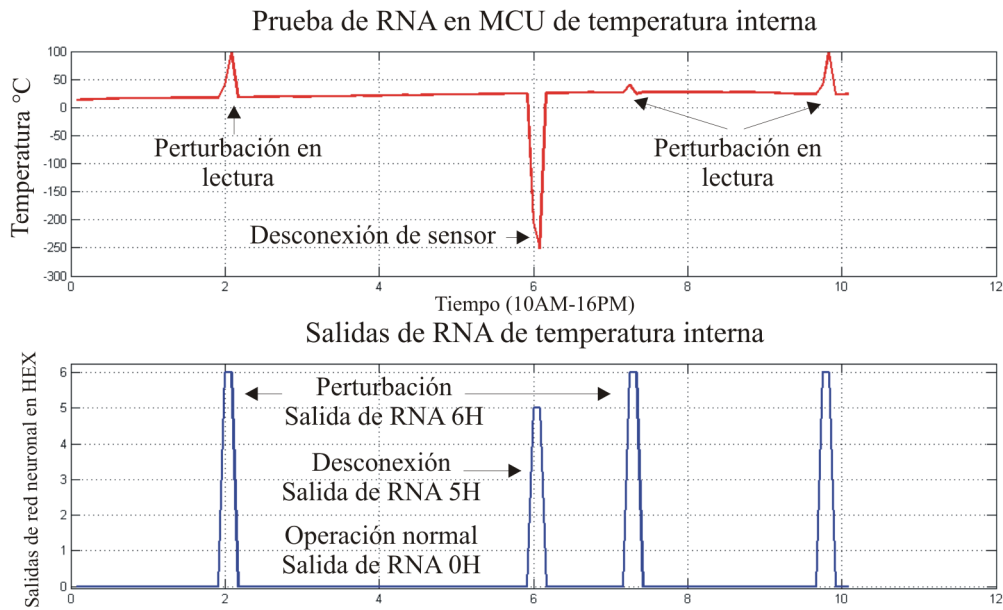


Figura 4.8: Prueba de RNA en MCU de la temperatura interna

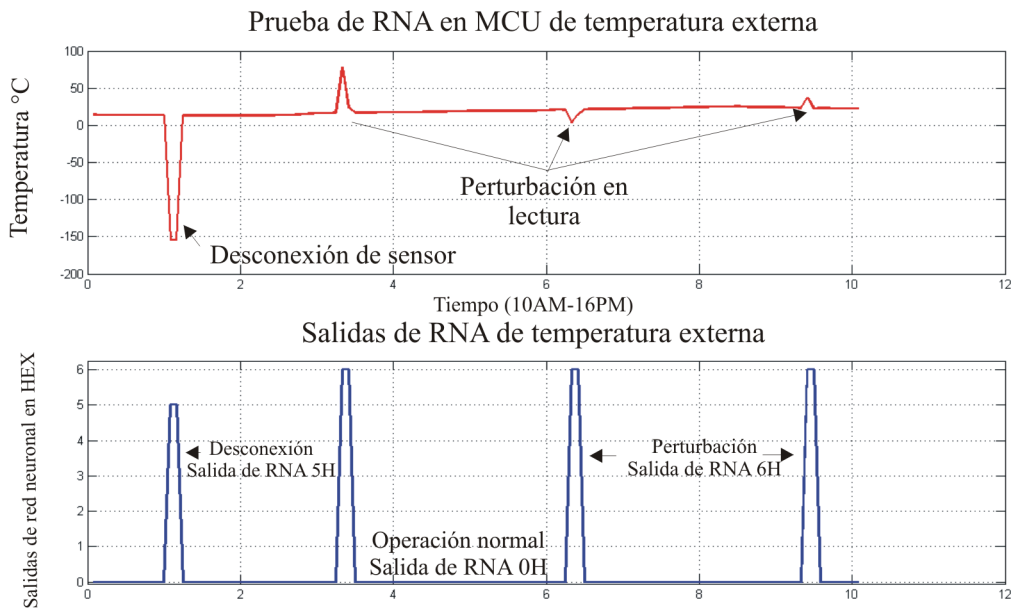


Figura 4.9: Prueba de RNA en MCU de la temperatura externa

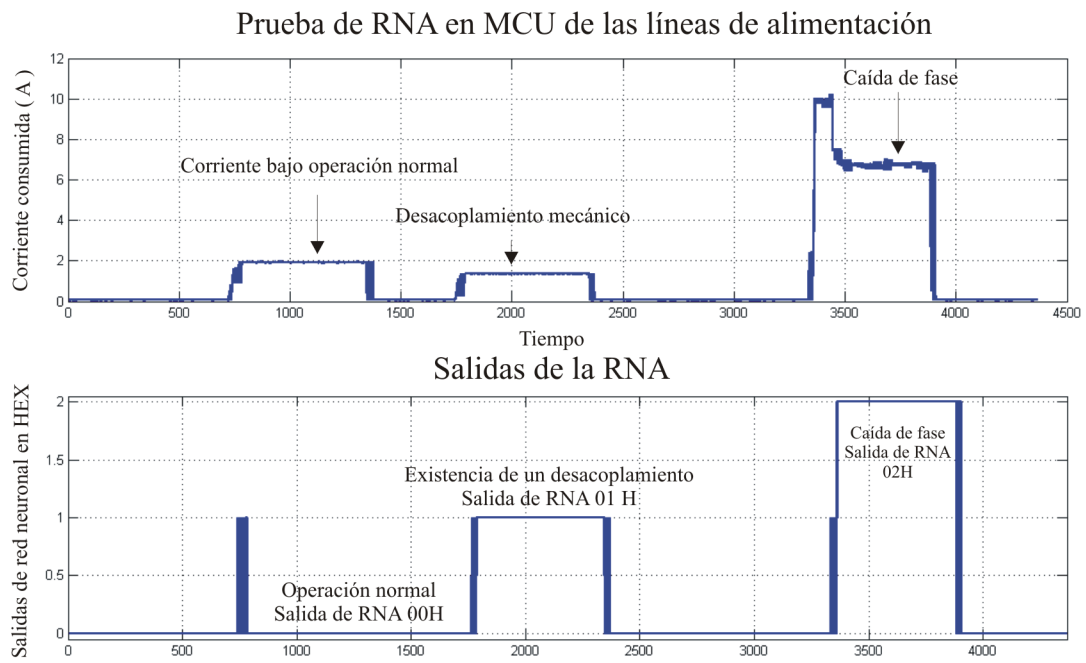


Figura 4.10: Prueba de RNA en MCU de las líneas de alimentación

Capítulo 5

Conclusiones

Con los resultados obtenidos, se demuestra la efectividad que tienen las redes neuronales para la tarea de detección de fallas, además de la generación de un posible diagnóstico de la falla. A partir, de las salidas de la RNA. Las RNA son herramientas poderosas que son robustas para el tipo de aplicación.

El sistema de detección de fallas y diagnóstico, detecta problemas de los sensores como la desconexión, el corto circuito y las lecturas anómalas. Mientras que para los actuadores, el sistema detecta caída de fase y desacoplamiento mecánico. Además de diferenciar entre lecturas óptimas y lecturas erróneas.

El desarrollo de la RNA genérica al igual que la normalización de datos, realizado al momento de generar el vector de entrenamiento, y/o al presentar un dato al MCU, hace que la arquitectura de RNA propuesta de feed-forward trabaje de manera similar que una de arquitectura de función de base radial (RBF). La ventaja entre la arquitectura de RBF con la feed-forward sin normalizar, es la propiedad de mejor aproximación al target o tarea a realizar.

Cabe destacar, que las arquitecturas de redes neuronales de sensores y actuadores propuestas,

resultan ser sencillas y de fácil implementación (software o hardware). Las redes neuronales en sí, no demandan gran capacidad de procesamiento de cómputo, si el sistema se desea implementar en software. Mas sin embargo, por su bajo grado de complejidad, también las redes neuronales pueden estar embebidas dentro de un MCU u otro tipo de hardware como lo pueden ser dsPIC, DSP ó FPGA.

Por su entrenamiento off-line realizado anteriormente, varios parámetros fueron entonces conocidos cuando se termina el entrenamiento. Lo cual, hace que las ecuaciones de RNA correspondientes, puedan ser programadas de manera más fácil.

El método provee eficiencia computacional sin sacrificar la efectividad en la tarea de detección de fallas y diagnóstico.

La aplicación en hardware del sistema de detección de fallas y diagnóstico, resulta ser de mucha utilidad, ya que si se desea implementar en otro sistema, simplemente hay que verificar el tipo de comunicación y el tipo de señal que se maneja (digital o analógica) para acoplar el sistema de manera paralela y realizar las validaciones correspondientes de las lecturas.

También se concluye, que las salidas de las diferentes RNA del sistema, pueden utilizarse para realizar acciones de control correctivas o preventivas. Dependiendo del tipo de falla.

El sistema finalmente, tiene la posibilidad de ser implementado en otros sistemas, donde se desee aplicar dicha tarea de detección de fallas o para la ejecución de acciones de control.

Bibliografía

- [Ciencia, 00] 2000 AGRO. Espectacular crecimiento de invernaderos en México, 2000 AGRO. Ciencia y tecnología, Vol. 46, junio 2004.
- [Pinon et al., 95] Alag S., Goebel K., Agonino A. A methodology for intelligent sensor validation and fusion used in tracking and avoidance of objects for automated vehicles, 1995.
- [Alcorta et al., 97] Alcorta García E., Frank P. M., Deterministic non-linear observer-based fault diagnosis: a survey. Control Eng. Practice, Vol. 5, Nr. 5, pp. 663-670, 1997.
- [Bailey et al., 84] Bailey, S.J., From desktop to plant floor, a CRT is the control operators window on the process, Control Engineering 31 (6), 1984, 86-90.
- [Basila et al., 90] Basila, M., Jr., Stefanek, G., and Cinar, A., A model-object based supervisory expert system for fault tolerant chemical reactor control. Computers and Chemical Engineering 14 (4-5), 551-560, 1990.
- [Basseville et al., 86] Basseville, M., and Benveniste, A., Detection of abrupt changes in signals and dynamic systems (Lecture Notes in Control and Information Sciences: 77). Berlin: Springer-Verlag, 1986.
- [Basseville et al., 88] Basseville, M., Detecting changes in signals and systems a survey. Automatica 24 (3), 309-326, 1988.

- [Basseville et al., 93] Basseville, M., and Nikiforov, I. V., Detection of abrupt changes -theory and application (Information and System Sciences Series). Prentice Hall, 1993.
- [Becraft et al., 93] Becraft, W., and Lee, P., An integrated neural network/expert system approach for fault diagnosis. Computers and Chemical Engineering 17 (10), 1001-1014, 1993.
- [Bureau et al., 98] Bureau of Labor Statistics., Occupational injuries and illnesses in the united states by industry. Washington, DC: Government Printing Office, 1998.
- [Castañeda et al., 05] Castañeda Miranda Rodrigo. SISTEMA DE CONTROL CLIMÁTICO INTELIGENTE PARA INVERNADEROS, Tesis doctoral, Universidad Autónoma de Querétaro UAQ, Querétaro, 2005.
- [Chester et al., 84] Chester, D., Lamb, D., Dhurjati, P., Rule-based computer alarm analysis in chemical process plants. In Proceedings of 7th Micro-Delcon. 22-29, 1984.
- [Chen et al., 92] Chen, L., and Modarres, M., Hierarchical decision process for fault administration. Computers and Chemical Engineering 16 (5), 425-448, 1992.
- [Chen et al., 02] Chen Y. M., Lee M. L., Neural networks-based scheme for system failure detection and diagnosis, Mathematics and Computers in Simulation, Volume 58, Pages: 101-109, Issue 2, 2002.
- [Cheung et al., 90] Cheung, J. T., and Stephanopoulos, G., Representation of process trends part I. A formal representation framework. Computers and Chemical Engineering 14 (4-5), 495-510, 1990.
- [Chow et al., 84] Chow E. Y., Willsky A. S., Analytical redundancy and the design of robust failure detection systems, IEEE Trans. on Automatic. Control, AC-29(7), pp. 603-614, July 1984.
- [Chow et al., 94] Chow, E. Y., and Willsky, A. S., Analytical redundancy and the design of robust failure detection systems. IEEE Transactions on Automatic Control 29 (7), 603-614, 1984.

- [Depari et al., 07] Depari A., Ferrari P., Ferrari V., Flammini A., Ghisla A., Marioli D., Taroni A., Digital signal processing for Biaxial Position Measurement With a Pyroelectric Sensor Array, *IEEE Transactions on instrumentation and measurement*, 2007, 56, pp. 75-79.
- [Ferentinos et al., 03a] Ferentinos K. P., Albright L. D., Selman B., Neural networks-based detection of mechanical sensor and biological faults in deep-trough hydroponics, *Computers and Electronics in Agriculture* 40, 65-85, 2003.
- [Ferentinos et al., 03b] Ferentinos K. P., Albright L. D., Fault detection and diagnosis in Deep-trough hydroponics using intelligent computational tools, *Biosystems Engineering*, 84(1), 13-30, 2003.
- [Frank et al., 90] Frank P. M., Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - a survey. *Automatica* 26, pp. 459-474, 1990.
- [Frank et al., 90b] Frank, P. M., Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy -a survey and some new results. *Automatica* 26, 459-474, 1990.
- [Frank et al., 00] Frank, P. M., Ding, S. X., and Marcu, T., Model-based fault diagnosis in technical processes. *Transactions of the Institution of Measurement and Control* 22 (1), 57-101, 2000.
- [Fuente et al., 99] Fuente M.J., Vega P., Neural networks applied to fault detections of a biotechnological process, *Engineering Applications of Artificial Intelligence* 12 (1999) 569-584, 1999.
- [Gertler et al., 91] Gertler, J., Analytical redundancy methods in fault detection and isolation. In *Proceedings of IFAC/IAMCS symposium on safe process* (p. 91), Baden-Baden, 1991.
- [Henley et al., 84] Henley, E.J., Application of expert systems to fault diagnosis. In *AICHE annual meeting* , San Francisco, CA, 1984.
- [Himmelblau et al., 78] Himmelblau, D. M., *Fault detection and diagnosis in chemical and petrochemical processes* . Amsterdam: Elsevier press, 1978.

- [Hoskins et al., 91] Hoskins, J. C., Kaliyur, K. M., and Himmelblau, D. M., Fault diagnosis in complex chemical plants using artificial neural networks. *American Institute of Chemical Engineers Journal* 37 (1), 137-141, 1991.
- [Hoskins et al., 94] Hoskins J. C. , Kaliyur K. M. and Himmelblau David M., Fault Diagnosis in Complex Chemical Plants Using Artificial Neural Networks, *AIChE journal*, Vol. 37, N11, January 1991.
- [Isermann et al., 84] Isermann R., Process fault detection based on modeling and estimation methods- A survey. *Automatica* 20, pp. 387-404, 1984.
- [Jakubek et al., 04] Jakubek S. M., Strasser T. I., Artificial neural networks for fault detection in large-scale data acquisition systems, *Engineering Applications of Artificial Intelligence* 17, 233-248, 2004.
- [Janusz et al., 91] Janusz, M., and Venkatasubramanian, V., Automatic generation of qualitative description of process trends for fault detection and diagnosis. *Engineering Applications of Artificial Intelligence* 4 (5), 329-339, 1991.
- [Kramer et al., 93] Kramer, M. A., and Mah, R. S. H., Model-based monitoring. In: Rippin, D., Hale, J., and Davis, J., (Eds.), *Proceedings of the second international conference on 'foundations of computer-aided process operations' (FOCAPO)* (pp. 45-68), 1993.
- [Lees et al., 96] Lees, F. P. *Loss prevention in process industries: Hazard identification, assessment and control*. London: Butterworth-Heinemann, 1996.
- [Linker et al., 00] Linker R., Gutman P.O., Seginer I., Robust model-based failure detection and identification in greenhouses, *Computers and Electronics in Agriculture* 26, pp. 255-270, 2000.

- [Matsura et al., 89] Watanabe, K., Matsura, I., Abe, M., Kubota, M., and Himmelblau, D. M., Incipient fault diagnosis of chemical processes via artificial neural networks. *American Institute of Chemical Engineers Journal* 35 (11), 1803-1812, 1989.
- [Mehranbod et al., 05] Mehranbod Nasir, Masoud Soroush, Chanin Panjapornpon, A method of sensor fault detection and identification, *Journal of process control* 15, 321-339, 2005.
- [National, 90] National Safety Council. *Injury facts 1999 Edition* , Chicago: National Safety Council, 1999.
- [Niida et al., 85] Niida, K., Expert system experiments in processing engineering. In *Institution of chemical engineering symposium series* (pp. 529 - 583), 1985.
- [Parlos et al., 94] Parlos A. G., Jayakumar Muthusami, Atiya A. F., Incipient fault detection and identification in process systems using accelerated neural network learning, *Nuclear technology* ISSN 0029-5450 CODEN NUTYBB, vol. 105, no. 2, pp. 145-161 (33 ref.), 1994.
- [Tan et al., 07] Tan S.C., Lim C.P., Rao M.V.C., A hybrid neural network model for rule generation and its application to process fault detection and diagnosis, *Engineering Applications of Artificial Intelligence* 20, pp. 203-213, 2007.
- [Tarifa et al., 97] Tarifa, E., and Scenna, N., Fault diagnosis, directed graphs, and fuzzy logic. *Computers and Chemical Engineering* 21, S649-654, 1997.
- [Thomas et al., 02] Thomas Philippe, Lefebvre Dimitri, Fault detection and isolation in non-linear systems by using oversized neural networks, *Mathematics and Computers in Simulation* 60, 181-192, 2002.
- [Ungar et al., 90] Ungar, L. H., Powell, B. A., and Kamens, S. N., Adaptive networks for fault diagnosis and process control. *Computers and Chemical Engineering* 14 (4-5), 561-572, 1990.

- [Venkatasubramanian et al., 85] Venkatasubramanian, V., Inexact reasoning in expert systems: a stochastic parallel network approach. In Proceedings of the second conference on artificial intelligence applications (pp. 191-195). Miami, FL., 1985.
- [Venkatasubramanian et al., 89] Venkatasubramanian, V., CATDEX, knowledge-based systems in process engineering: case studies in heuristic classification . Austin, TX: The CACHE Corporation, 1989.
- [Chan et al., 89] Venkatasubramanian, V., and Chan, K., A neural network methodology for process fault diagnosis. American Institute of Chemical Engineers Journal 35 (12), 1993-2002, 1989.
- [Willsky et al., 76] Willsky, A. S., A survey of design methods for failure detection in dynamic systems. Automatica 12, 601-611, 1976.
- [Wo et al., 00] Wo, M., Gui, W, Shen, D, Wang, Y., Export fault diagnosis using role models with certainty factors for the leaching process. In Proceedings of the third world congress on intelligent control and automation, volume 1 (pp. 238-241). Hefei, China, 28 June-2 July, 2000.
- [Zhang et al., 91] Zhang, J., and Roberts, P., Process fault diagnosis with diagnostic rules based on structural decomposition. Journal of Process Control 1 (5), 259-269, 1991.
- [Zhao et al., 97] Zhao, J., Chen, B., and Shen, J., A hybrid ANN-ES system for dynamic fault diagnosis of hydrocracking process. Computers and Chemical Engineering 21, S929-S933, 1997.

Apéndice A

Descripción de programa en MATLAB para RNA de sensores

```
clc;
clear all;
close all;
ar=fopen('Datos_286_PYF.FIL');
datos=fscanf(ar,'%f',[9,286]);
%Adquisicion de datos
tiempo=datos(1,:);
temp_int=datos(2,:); %Max=60.C, Min=0.C
temp_ext=datos(3,:); %Max=60.C, Min=0.C
hr_int=datos(4,:); %Max=100%, Min=0%
hr_ext=datos(5,:); %Max=100%, Min=0%
rad=datos(6,:); %Max=1800 w/m2, Min=0 w/m2
vel=datos(8,:); %Max=67 m/s, Min=1 m/s
dir=datos(9,:); %Max=360., Min=0.
fclose(ar);
%Tiempo de 12:00 am - 11:55 pm
t=length(tiempo); %283 datos
for i=1:t
    tiempo(i)=i*5/60;
end
%NORMALIZACION DE LOS DATOS _n ""x=(x'-x'min)/(x'max-x'min)""
%Escala de los sensores
temp_min=0; temp_max=60;
```


APÉNDICE A. DESCRIPCIÓN DE PROGRAMA EN MATLAB PARA RNA DE SENSORES 113

```
hr_min=0; hr_max=100;
rad_min=0; rad_max=1800;
vel_min=0; vel_max=67;
dir_min=0; dir_max=360;
for i=1:t
%Normalizacion
temp_int_n(i)=(temp_int(i)-temp_min)/(temp_max-temp_min);
hr_int_n(i)=(hr_int(i)-hr_min)/(hr_max-hr_min);
temp_ext_n(i)=(temp_ext(i)-temp_min)/(temp_max-temp_min);
hr_ext_n(i)=(hr_ext(i)-hr_min)/(hr_max-hr_min);
rad_n(i)=(rad(i)-rad_min)/(rad_max-rad_min);
vel_n(i)=(vel(i)-vel_min)/(vel_max-vel_min);
dir_n(i)=(dir(i)-dir_min)/(dir_max-dir_min);
end
%Perturbacion hacia sensores
temperatura=5; %5 .C de perturbacion
hrel=15; %10 % de perturbacion
radia=200; %100 w/m2 de perturbacion
%Normalizacion de perturbaciones
temperatura=(temperatura-temp_min)/(temp_max-temp_min);
hrel=(hrel-hr_min)/(hr_max-hr_min);
radia=(radia-rad_min)/(rad_max-rad_min);
tempo_temp_int=temp_int_n(1);
tempo_temp_ext=temp_ext_n(1);
tempo_hr_int=hr_int_n(1);
tempo_hr_ext=hr_ext_n(1);
tempo_rad=rad_n(1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%VECTOR DE ENTRENAMIENTO%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:t
%TEMPERATURA INTERNA NORMALIZADA
if (temp_int_n(i) < -4)
t1(i)=1; %Deteccion Falla
t2(i)=1; %Clasificacion Desconexion
t3(i)=0; %Clasificacion Desconexion
elseif ((tempo_temp_int>temp_int_n(i)+temperatura)|| (tempo_temp_int<temp_int_n(i)-temperatura))
t1(i)=1;
t2(i)=0;
t3(i)=1;
%%Por hacer el corto circuito
else
```

APÉNDICE A. DESCRIPCIÓN DE PROGRAMA EN MATLAB PARA RNA DE SENSORES114

```
t1(i)=0; %Detección no Falla
t2(i)=0; %Clasificación Normal
t3(i)=0; %Clasificación Normal
end
%TEMPERATURA EXTERNA NORMALIZADA
if (temp_ext_n(i) < -4)
t4(i)=1; %Detección Falla
t5(i)=1; %Clasificación Desconexion
t6(i)=0; %Clasificación Desconexion
elseif ((tempo_temp_ext>temp_ext_n(i)+temperatura) || (tempo_temp_ext<temp_ext_
t4(i)=1;
t5(i)=0;
t6(i)=1;
%%Por hacer el corto circuito
else
t4(i)=0; %Detección no Falla
t5(i)=0; %Clasificación Normal
t6(i)=0; %Clasificación Normal
end
%HUMEDAD RELATIVA INTERNA NORMALIZADA
if (hr_int_n(i) > 19 )
t7(i)=1;
t8(i)=1;
t9(i)=0;
%%Por hacer el corto circuito
elseif ((tempo_hr_int>hr_int_n(i)+hrel) || (tempo_hr_int<hr_int_n(i)-hrel))
t7(i)=1;
t8(i)=0;
t9(i)=1;
else
t7(i)=0;
t8(i)=0;
t9(i)=0;
end
%HUMEDAD RELATIVA EXTERNA NORMALIZADA
if (hr_ext_n(i) > 19 )
t10(i)=1;
t11(i)=1;
t12(i)=0;
%%Por hacer el corto circuito
elseif ((tempo_hr_ext>hr_ext_n(i)+hrel) || (tempo_hr_ext<hr_ext_n(i)-hrel))
```

APÉNDICE A. DESCRIPCIÓN DE PROGRAMA EN MATLAB PARA RNA DE SENSORES115

```
t10(i)=1;
t11(i)=0;
t12(i)=1;
else
t10(i)=0;
t11(i)=0;
t12(i)=0;
end
%RADIACIÓN GLOBAL NORMALIZADA
if (rad_n(i) > 0.7)
t13(i)=1;
t14(i)=1;
t15(i)=0;
%%Por hacer el corto circuito
elseif((tempo_rad>rad_n(i)+radia)|| (tempo_rad<rad_n(i)-radia))
t13(i)=1;
t14(i)=0;
t15(i)=1;
else
t13(i)=0;
t14(i)=0;
t15(i)=0;
end
tempo_temp_int=temp_int_n(i);
tempo_temp_ext=temp_ext_n(i);
tempo_hr_int=hr_int_n(i);
tempo_hr_ext=hr_ext_n(i);
tempo_rad=rad_n(i);
end
%-----Target de temperatura interna
t_temp_int(1,:)=t1;
t_temp_int(2,:)=t2;
t_temp_int(3,:)=t3;
%Creando la red neuronal
net_temp_int=newff([minmax(temp_int_n)], [3,3], {'logsig', 'purelin'}, 'trainlm');
%Se propone un minimo y un máximo para garantizar
%toda la gamma de valores para el entrenamiento. Todo con su
%target correspondiente para cada valor.
%3 salidas relacionadas con los targets correspondientes.
%condiciones de entrenamiento
net_temp_int.trainParam.epochs=20;
```

APÉNDICE A. DESCRIPCIÓN DE PROGRAMA EN MATLAB PARA RNA DE SENSORES 116

```
net_temp_int.trainParam.goal=0;
%inicializando pesos
net_temp_int=init(net_temp_int);
net_temp_int.iw(1,1);
net_temp_int.b{1};
%entrenamiento
[net_temp_int,tr]=train(net_temp_int,temp_int_n,t_temp_int);
%Entrenamiento de la red NET, con respecto a las entradas
%de temperatura internas normalizadas y con los
%targets correspondientes a cada
%punto de las temperaturas.
%-----Target de temperatura externa
t_temp_ext(1,:)=t4;
t_temp_ext(2,:)=t5;
t_temp_ext(3,:)=t6;
net_temp_ext=newff([minmax(temp_ext_n)], [3,3], {'logsig','purelin'}, 'trainlm');
net_temp_ext.trainParam.epochs=20;
net_temp_ext.trainParam.goal=0;
net_temp_ext=init(net_temp_ext);
net_temp_ext.iw(1,1);
net_temp_ext.b{1};
[net_temp_ext,tr]=train(net_temp_ext,temp_ext_n,t_temp_ext);
%-----Targets y entrenamiento de humedad relativa interna
t_hr_int(1,:)=t7;
t_hr_int(2,:)=t8;
t_hr_int(3,:)=t9;
net_hr_int=newff([minmax(hr_int_n)], [3,3], {'logsig','purelin'}, 'trainlm');
net_hr_int.trainParam.epochs=20;
net_hr_int.trainParam.goal=0;
net_hr_int=init(net_hr_int);
net_hr_int.iw(1,1);
[net_hr_int,tr]=train(net_hr_int,hr_int_n,t_hr_int);
%-----Target y entrenamiento de humedad relativa externa
t_hr_ext(1,:)=t10;
t_hr_ext(2,:)=t11;
t_hr_ext(3,:)=t12;
net_hr_ext=newff([minmax(hr_ext_n)], [3,3], {'logsig','purelin'}, 'trainlm');
net_hr_ext.trainParam.epochs=20;
net_hr_ext.trainParam.goal=0;
net_hr_ext=init(net_hr_ext);
net_hr_ext.iw(1,1);
```

APÉNDICE A. DESCRIPCIÓN DE PROGRAMA EN MATLAB PARA RNA DE SENSORES117

```
[net_hr_ext,tr]=train(net_hr_ext,hr_ext_n,t_hr_ext);
%-----Target y entrenamiento de radiacion
t_rad(1,:)=t13;
t_rad(2,:)=t14;
t_rad(3,:)=t15;
net_rad=newff([minmax(rad_n)],[3,3],{'logsig','purelin'},'trainlm');
net_rad.trainParam.epoch=50;
net_rad.trainParam.goal=0;
net_rad=init(net_rad);
net_rad.iw{1,1};
[net_rad,tr]=train(net_rad,rad_n,t_rad);
%
%-----DEMOSTRACION-----
%
prueba_datos=fopen('Datos_286_PYF.FIL');
%Nuevo archivo para demostración de la red neural. ARCHIVOS: Datos_284_2F, Datos_286, Datos_286_F,
%Datos_286_P, Datos_286_PYF.FIL
prueba=fscanf(prueba_datos,'%f',[9,286]);
fclose(prueba_datos);
%Datos para validación
datos_temp_int=prueba(2,:);
datos_temp_ext=prueba(3,:);
datos_hr_int=prueba(4,:);
datos_hr_ext=prueba(5,:);
datos_rad=prueba(6,:);
%Normalizacion de los datos de validación
temp_min=0; temp_max=60;
hr_min=0; hr_max=100;
rad_min=0; rad_max=1800;
vel_min=0; vel_max=67;
dir_min=0; dir_max=360;
for i=1:t
%Normalización Temp_int_nn = Temperatura interna de validación para RED
%NEURONAL
temp_int_nn(i)=(datos_temp_int(i)-temp_min)/(temp_max-temp_min);
temp_ext_nn(i)=(datos_temp_ext(i)-temp_min)/(temp_max-temp_min);
hr_int_nn(i)=(datos_hr_int(i)-hr_min)/(hr_max-hr_min);
hr_ext_nn(i)=(datos_hr_ext(i)-hr_min)/(hr_max-hr_min);
rad_nn(i)=(datos_rad(i)-rad_min)/(rad_max-rad_min);
end
%SALIDAS DE LA RED NEURONAL
```

APÉNDICE A. DESCRIPCIÓN DE PROGRAMA EN MATLAB PARA RNA DE SENSORES 118

```
%Out NN = sim (NN,Datos de entrenamiento NORMALIZADOS);
out_temp_int=sim(net_temp_int,temp_int_nn);
%Demostración de la red entrenada con otros datos para su validación
out_temp_ext=sim(net_temp_ext,temp_ext_nn);
out_hr_int=sim(net_hr_int,hr_int_nn);
out_hr_ext=sim(net_hr_ext,hr_ext_nn);
out_rad=sim(net_rad,rad_nn);
%%%%%%%%TEMPERATURA INTERNA
figure
subplot(4,1,1)
plot(tiempo,temp_int_nn,'LineWidth',2,'Color',[1 0 0]);
grid
subplot(4,1,2)
plot(tiempo,out_temp_int(1,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,3)
plot(tiempo,out_temp_int(2,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,4)
plot(tiempo,out_temp_int(3,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
%%%%%%%%TEMPERATURA EXTERNA
figure
subplot(4,1,1)
plot(tiempo,temp_ext_nn,'LineWidth',2,'Color',[1 0 0]);
grid
subplot(4,1,2)
plot(tiempo,out_temp_ext(1,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,3)
plot(tiempo,out_temp_ext(2,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,4)
plot(tiempo,out_temp_ext(3,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
```

APÉNDICE A. DESCRIPCIÓN DE PROGRAMA EN MATLAB PARA RNA DE SENSORES119

```
%%%%%%%%%HUMEDAD RELATIVA INTERNA
figure
subplot(4,1,1)
plot(tiempo,hr_int_nn,'LineWidth',2,'Color',[1 0 0]);
grid
subplot(4,1,2)
plot(tiempo,out_hr_int(1,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,3)
plot(tiempo,out_hr_int(2,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,4)
plot(tiempo,out_hr_int(3,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
%%%%%%%%%HUMEDAD RELATIVA EXTERNA
figure
subplot(4,1,1)
plot(tiempo,hr_ext_nn,'LineWidth',2,'Color',[1 0 0]);
grid
subplot(4,1,2)
plot(tiempo,out_hr_ext(1,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,3)
plot(tiempo,out_hr_ext(2,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,4)
plot(tiempo,out_hr_ext(3,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
%%%%%%%%%RADIACION
figure
subplot(4,1,1)
plot(tiempo,rad_nn,'LineWidth',2,'Color',[1 0 0]);
grid
subplot(4,1,2)
plot(tiempo,out_rad(1,:),'LineWidth',2,'Color',[0 0 1]);
```

APÉNDICE A. DESCRIPCIÓN DE PROGRAMA EN MATLAB PARA RNA DE SENSORES120

```
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,3)
plot(tiempo,out_rad(2,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
subplot(4,1,4)
plot(tiempo,out_rad(3,:),'LineWidth',2,'Color',[0 0 1]);
grid
AXIS([0 24.5 -0.05 1.05])
```


Apéndice B

Descripción de en programa MATLAB para RNA de líneas

```
clc;
clear all;
close all;
ar=fopen('normal2.txt');
datos=fscanf(ar,'%f',[3,inf]);
%Adquisición de datos
lineal=datos(1,:);
fclose(ar);
%limpiar datos
t=length(lineal);
for (i=1:t)
if (lineal(i)<0)
lineal(i)=0.0;
end
end
for k=1:t
tiempo(k)=(k/10);
end
%%%%%%%%%%%%%%Normaliación de datos
trans_min=0; trans_max=10;
for i=1:t
lineal_n(i)=(lineal(i)-trans_min)/(trans_max-trans_min);
end
```

APÉNDICE B. DESCRIPCIÓN DE EN PROGRAMA MATLAB PARA RNA DE LÍNEAS 122

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Vector de entrenamiento
for i=1:t %%Fallas de caida de fase(l10), sin carga(101), operación normal (00)
if (lineal_n(i)<0.15 && lineal_n(i)>0.144) %Sin CARGA (01)
t1(i)=0;
t2(i)=1;
elseif (lineal_n(i)>.5) %Caida de fase (10)
t1(i)=1;
t2(i)=0;
else
t1(i)=0;
t2(i)=0;
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Target motores
t_motores(1,:)=t1;
t_motores(2,:)=t2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%1 entrada a la RNA
input=[minmax(lineal_n)];
net_motores=newff(input,[3,2],{'logsig','purelin'},'trainlm');
%condiciones de entrenamiento
net_motores.trainParam.epochs=50;
net_motores.trainParam.goal=0;
%inicializando pesos
net_motores=init(net_motores);
net_motores.iw(1,1);
net_motores.b{1};
%entrenamiento
lineas(1,:)=lineal_n;
[net_motores,tr]=train(net_motores,lineas,t_motores);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%DEMOSTRACION%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%DEMOSTRACION%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
prueba_motores=fopen('normal2.txt');
prueba=fscanf(prueba_motores,'%f',[3,inf]);
fclose(prueba_motores);
%ADQ DATOS DE PRUEBA
lineal_prueba=prueba(1,:);
new_tiempo=length(lineal_prueba);
for (i=1:new_tiempo)
if (lineal_prueba(i)<0)
lineal_prueba(i)=0.0;
end
```

APÉNDICE B. DESCRIPCIÓN DE EN PROGRAMA MATLAB PARA RNA DE LÍNEAS 123

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Normaliación de datos
trans_min=0; trans_max=10; %transductor min, transductor máximo
for i=1:new_tiempo %Normalización
lineal_n_prueba(i)=(lineal_prueba(i)-trans_min)/(trans_max-trans_min);
end
lineas_prueba(1,:)=lineal_n_prueba;
%%Salidas de la RNA
out_lineas=sim(net_motores,lineas_prueba);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Lineas del motor GRAFICAS
figure
subplot (3,1,1)
plot(tiempo,lineal_n_prueba,'Linewidth',2,'color',[0,0,1])
grid
AXIS([0 210 -0.05 1.05])
subplot(3,1,2)
plot(tiempo,out_lineas(1,:),'LineWidth',2,'Color',[1 0 0]);
grid
AXIS([0 210 -0.05 1.05])
subplot(3,1,3)
plot(tiempo,out_lineas(2,:),'LineWidth',2,'Color',[1 0 0]);
grid
AXIS([0 210 -0.05 1.05])
```

Apéndice C

Caracterización de transductor de corriente en MATLAB

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ajuste polinomial de curvas con cuantizacion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parte 1
% Inicializar procesos
clear;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parte 2
% Parametros del proceso %
% Grado del polinomio deseado para ajuste
N = 5;
% Puntos a ajustar
%X= voltaje Y=corriente
X(1) = 0; Y(1) = 0;
X(2) = 0.4357; Y(2) = 0.4920;
X(3) = 0.8744; Y(3) = 0.9668;
X(4) = 1.3076; Y(4) = 1.4518;
X(5) = 1.6340; Y(5) = 1.9776;
X(6) = 1.6603; Y(6) = 2.4784;
X(7) = 1.6847; Y(7) = 2.9852;
X(8) = 1.9501; Y(8) = 3.5082;
```

APÉNDICE C. CARACTERIZACIÓN DE TRANSDUCTOR DE CORRIENTE EN MATLAB125

```
X(9) = 2.0362; Y(9) = 4.0026;
X(10) = 2.0924; Y(10) = 4.3592;
X(11) = 2.1982; Y(11) = 5.1410;
X(12) = 2.2690; Y(12) = 5.532;
X(13) = 2.3539; Y(13) = 6.177;
X(14) = 2.3985; Y(14) = 6.487;
% Bits de cuantizacion
Q = 18;
% Resolucion del ADC
ADC = 12;
% Numero de bits del contador programable del MAC
K = 8;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parte 3
% Ajuste polinomial por minimos cuadrados
P = polyfit(X,Y,N);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parte 4
% Cuantizacion de coeficientes del polinomio
% Determinacion del valor maximo absoluto de los coeficientes
cM = max(abs(P));
% Valor normalizado minimo absoluto representable en Q bits de punto fijo
L = 1.0 - 2.0^(1-Q);
% Numero de bits enteros
if (cM < L)
e = 1;
else
e = 1 + ceil(log(cM)/log(2.0*L));
end;
% Numero de bits fraccionarios
f = Q - e;
% Factor de cuantizacion
Fq = 2^f;
% Cuantizacion de los coeficientes del polinomio
PQ = floor(P*Fq + 0.5);
% Escalamiento de los coeficientes del filtro por redondeo
PS = PQ/Fq;
% Binarizacion del indice por el teorema del residuo
for i=1:N+1
Auxiliar = i-1;
for j=K:-1:1
```

APÉNDICE C. CARACTERIZACIÓN DE TRANSDUCTOR DE CORRIENTE EN MATLAB126

```
IDX(i,j) = rem(Auxiliar,2);
Auxiliar = floor(Auxiliar/2);
end;
end;
% Binarizacion de los coeficientes del polinomio en complemento 2
for i=1:N+1
% Ajuste a entero positivo equivalente en complemento 2
Auxiliar = PQ(i);
if (Auxiliar < 0)
Auxiliar = 2^Q + Auxiliar;
end;
% Binarizacion
for j=Q:-1:1
ABI(i,j) = rem(Auxiliar,2);
Auxiliar = floor(Auxiliar/2);
end;
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parte 5
% Generacion VHDL de la ROM de coeficientes
% Apertura de archivo VHDL
fid = fopen('ROM_polinomio.vhd','wt');
% Encabezado de la descripcion
fprintf(fid,'-- Coeficientes del polinomio\n');
fprintf(fid,'--\n');
fprintf(fid,'-- Universidad Autonoma de Queretaro\n');
fprintf(fid,'--\n');
fprintf(fid,'--\n');
fprintf(fid,'--\n');
fprintf(fid,'--\n');
fprintf(fid,'\n');
fprintf(fid,'library IEEE;\n');
fprintf(fid,'use IEEE.std_logic_1164.all;\n');
fprintf(fid,'\n');
fprintf(fid,'entity ROM_polinomio is\n');
fprintf(fid,' port(\n');
% Definicion del indice del coeficiente
fprintf(fid,' I : in std_logic_vector(%d downto 0);\n',K-1);
% Definicion del coeficiente del filtro
fprintf(fid,' A : out std_logic_vector(%d downto 0)\n',Q-1);
fprintf(fid,' );\n');
fprintf(fid,' end ROM_polinomio;\n');
```

APÉNDICE C. CARACTERIZACIÓN DE TRANSDUCTOR DE CORRIENTE EN MATLAB127

```
fprintf(fid,'\n');
fprintf(fid,'architecture Tabla of ROM_polinomio is\n');
fprintf(fid,'begin\n');
fprintf(fid,' process(I)\n');
fprintf(fid,' begin\n');
fprintf(fid,' case I is\n');
fprintf(fid,' -- Formato de salida %d.%d\n',e,f);
% Vaciado automatico de la tabla de equivalencias
for i=1:N+1
fprintf(fid,' when "');
% Binarizacion del argumento
for j=1:K
fprintf(fid,'%d',IDX(i,j));
end;
fprintf(fid,'" => A <= "');
% Binarizacion de la funcion
for j=1:Q
fprintf(fid,'%d',ABI(i,j));
end;
% Argumento y funcion como comentarios
fprintf(fid,'" -- Indice %d Coeficiente %1.8f\n',N-i+1,PS(i));
end;
% Fin de archivo VHDL
fprintf(fid,' when others => A <= "');
for i=1:Q
fprintf(fid,'0');
end;
fprintf(fid,'" -- Indices irrelevantes\n');
fprintf(fid,' end case;\n');
fprintf(fid,' end process;\n');
fprintf(fid,'end Tabla;\n');
% Cerrar archivo
fclose(fid);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parte 6
% Graficas comparativas
% Resolucion normalizada del ADC
M = 2^ADC;
% Normalizacion del eje horizontal
MinH = min(X);
MaxH = max(X);
```

APÉNDICE C. CARACTERIZACIÓN DE TRANSDUCTOR DE CORRIENTE EN MATLAB128

```
delta = (MaxH - MinH)/M;
H(1) = MinH;
for i=2:M
H(i) = H(i-1) + delta;
end;
% Evaluacion del eje vertical
for i=1:M
V(i) = polyval(PS,H(i));
end;
figure(1);
plot(H,V);
hold on;
title('Ajuste polinomial con truncamiento');
plot(X,Y,'ro');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parte 7
% Analisis del error
% Puntos de ajuste
A = size(X);
% Evaluacion del error en puntos de ajuste
for i=1:A(2)
Err(i) = polyval(PS,X(i)) - Y(i);
end;
% Estimacion del error
Err
```


Apéndice D

Programa en BUILDER con APRO

```
//-----  
  
#include <vcl.h>  
#pragma hdrstop  
  
#include "Unit1.h"  
#include <math.h>  
#include <stdio.h>  
//-----  
#pragma package(smart_init)  
#pragma link "AdPort"  
#pragma link "OoMisc"  
#pragma resource "*.dfm"  
TForm1 *Form1;  
int index_s=0, index_l=0;  
int flag=0;  
char opc = 't';  
unsigned char sensores[6]/*[6]*/, linea[8];  
double ti,te,hi,he,l1,l2,l3;  
double ti_aux=0.0, te_aux=0.0;  
int indexS = 0, indexL = 0;  
int count_lineas=0, count_sensores=0;  
  
//-----  
__fastcall TForm1::TForm1(TComponent* Owner)  
    : TForm(Owner)
```

```
{
}
//-----
void __fastcall TForm1::Button3Click(TObject *Sender)
{
    Form1->Close();
}
//-----
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
    if(opc == 't')
    {
        ApdComPort1->PutChar('s');
        opc='s';
        Edit8->Text=opc;
        indexS=0;
    }
}
//-----
void __fastcall TForm1::Timer2Timer(TObject *Sender)
{
    if(opc == 't')
    {
        ApdComPort1->PutChar('l');
        opc='l';
        Edit8->Text=opc;
        indexL=0;
    }
}
//-----
void __fastcall TForm1::ApdComPort1TriggerAvail(TObject *CP, WORD Count)
{
    unsigned int data;
    unsigned char t;
    char status_s1,status_s2, status_l1,status_l2;
    AnsiString S,L;
    FILE *stream;
    FILE *stream1;

    Edit9->Text=indexS;
```

```
Edit10->Text=indexL;

ApdComPort1->FlushInBuffer();
ApdComPort1->FlushOutBuffer();

switch(opc)
{
    case 's':
        if (indexS == 0)
        {
            t_espera->Enabled=true;
            Edit11->Text="FALSE";
        }
        if (indexS >=5)//5
        {
            t_espera->Enabled=false;
            Edit11->Text="ok";
        }
        while(Count-->0)
        {
            t = ApdComPort1->GetChar();
            sensores[indexS]=t;
            /*if (indexS==5 || indexS==7)
            {
                opc='t';
                TimerHR->Enabled=true;
            } */
            indexS++;
        }
        break;
    case 'l':
        if(indexL == 0)
        {
            t_espera->Enabled = true;
            Edit11->Text="FALSE";
        }
        if(indexL >=7)
        {
            t_espera->Enabled = false;
            Edit11->Text="ok";
        }
}
```

```

    }

    while(Count--)
    {
        t = ApdComPort1->GetChar();
        linea[indexL]=t;
        indexL++;
    }
    break;
}
if(indexS > 5)//5
{
    sensores[0] &= 0xFF;
    sensores[1] &= 0xFF;
    sensores[2] &= 0xFF;//salida de RNA

    sensores[3] &= 0xFF;
    sensores[4] &= 0xFF;
    sensores[5] &= 0xFF;//salida de RNA

    /*sensores[6] &= 0xFF; //H.R.
    sensores[7] &= 0xFF;

    sensores[8] &= 0xFF; //H.R.
    sensores[9] &= 0xFF; */

    data = (sensores[0]<<8 | sensores[1]);
    ti = (double) data;
    ti=ti*4;
    ti=58.143265-0.0306697*ti+0.0000163*pow(ti-1135.94,2.0)-1.2623E-8*pow(ti-1135.94,3.0);

    data = (sensores[3]<<8 | sensores[4]);//[2] y [3]
    te = (double) data;
    te=te*4;
    te=58.143265-0.0306697*te+0.0000163*pow(te-1135.94,2.0)-1.2623E-8*pow(te-1135.94,3.0);

    Edit1->Text = ti;
    Edit2->Text = te;
    if (sensores[2]==0)//Temp. Int.
    {
        Label5->Caption="Normal";
    }
}

```

```
Label5->Color=clBlack;
Label5->Font->Color=clLime;
status_s1='N';
Label7->Color=clBlack;
Label7->Font->Color=clLime;
Label7->Caption="Normal";
}
else if (sensores[2]==1)
{
Label5->Caption="Desconexión";
Label5->Font->Color=clRed;
status_s1='D';
Label7->Color=clBlack;
Label7->Font->Color=clRed;
Label7->Caption="Desconexión";
Edit4->Text="1930.24537868";
}
else if (sensores[2]==2)
{
Label5->Caption="Perturbación";
Label5->Font->Color=clYellow;
status_s1='P';
}
ti_aux=ti;
////////////////////////////////////
if (sensores[5]==0)//Temp. Int.
{
Label6->Caption="Normal";
Label6->Color=clBlack;
Label6->Font->Color=clLime;
status_s2='N';
Label8->Color=clBlack;
Label8->Font->Color=clLime;
Label8->Caption="Normal";
Edit4->Text="40.4654566457";
}
else if (sensores[5]==1)
{
Label6->Caption="Desconexión";
Label6->Font->Color=clRed;
status_s2='D';
```

```

        Label18->Color=clBlack;
        Label18->Font->Color=clRed;
        Label18->Caption="Desconexión";
        Edit4->Text="1938.4553705";
    }
    else if (sensores[5]==2)
    {
        Label16->Caption="Perturbación";
        Label16->Font->Color=clYellow;
        status_s2='P';
    }
    te_aux=te;

    //////////////////////////////////////

    sensores[6] &= 0xFF;
    sensores[7] &= 0xFF;
    data = sensores[6]<<8 | sensores[7];
    hi = (double) data;
    hi=hi*10;
    hi=721.82413-0.0747358*hi+0.0000152*pow(hi-8925.45,2.0);

    sensores[8] &= 0xFF;
    sensores[9] &= 0xFF;
    data = sensores[8]<<8 | sensores[9];
    he = (double) data;
    he=he*10;
    he=721.82413-0.0747358*he+0.0000152*pow(he-8925.45,2.0);

    Edit3->Text=hi;
    Edit4->Text=he;

    if(count_sensores<10)
    {

        //S.printf("%5.3lf\t%c\t%5.3lf\t%c",ti,status_s1,te,status_s2); //,hi,he);
        S.printf("%5.3lf\t%5.3lf\t%c\t%c",ti,te,status_s1,status_s2); //,hi,he);
        Memol->Lines->Add(S);
    }

```

```

        stream1 = fopen("C:\\Sensores.txt","a+");
        for(register int i = 0; i < Mem01->Lines->Count; i++)
        {
            fprintf(stream1, "%s \n",Mem01->Lines->Strings[i].c_str());
        }
        fclose(stream1);
        count_sensores++;
    }
    else
    {
        S.sprintf("%5.3lf\t%5.3lf\t%c\t%c",ti,te,status_s1,status_s2); //,hi,he);
        //S.sprintf("%5.3lf\t%c\t%5.3lf\t%c\t%5.3lf\t%5.3lf",ti,status_s1,te,status_s2,hi,he);
        Mem01->Lines->Clear();
        Mem01->Lines->Add(S);
        count_sensores=0;
    }
    indexS = 0;
    opc = 't';
    ApdComPort1->FlushInBuffer();
}
if(indexL > 7)
{
    linea[0] &= 0xFF;
    linea[1] &= 0xFF;
    linea[2] &= 0xFF;
    linea[3] &= 0xFF;
    linea[4] &= 0xFF;//RNA L2
    linea[5] &= 0xFF;
    linea[6] &= 0xFF;
    linea[7] &= 0xFF;//RNA L3

    data = linea[0]<<8 | linea[1];
    l1 = (double)data;
    l1 = l1*5/1024;
    //Caracterizacion
    l1=0.00007629*pow(l1,2.0)+1.17828369*l1+0.04866028;

    data = linea[2]<<8 | linea[3];
    l2 = (double)data;
    l2 = l2*5/1024;

```

```

//Caracterizacion
l2=0.00007629*pow(12,2.0)+1.17828369*12+0.04866028;

data = linea[5]<<8 | linea[6];
l3 = (double)data;
l3 = l3*5/1024;
//Caracterizacion
l3=0.41624451*pow(13,4.0)-1.12115479*pow(13,3.0)+0.88249207*pow(13,2.0)+0.89587402*13+0.05702209;

Edit5->Text=l1;
Edit6->Text=l2;
Edit7->Text=l3;
Edit12->Text=linea[4]; //salida de RNA
Edit13->Text=linea[7]; //salida de RNA
if(linea[4]==0)
{
    Label9->Caption="Normal o stop";
    Label9->Color=clBlack;
    Label9->Font->Color=clLime;
    status_l1='N';
}
else if (linea[4]==1)
{
    Label9->Caption="Desacoplamiento";
    Label9->Color=clBlack;
    Label9->Font->Color=clYellow;
    status_l1='D';
}
else if (linea[4]==2)
{
    Label9->Caption="Caida de fase";
    Label9->Color=clBlack;
    Label9->Font->Color=clRed;
    status_l1='F';
}

if(count_lineas<8)
{
    L.sprintf("%5.3lf\t%5.3lf\t%5.3lf\t%c", l1, l2, l3, status_l1);
    Memo2->Lines->Add(L);
}

```



```

        stream = fopen("C:\\Alimentacion.txt","a+");
        for(register int i = 0; i < Memo2->Lines->Count; i++)
        {
            fprintf(stream, "%s \n",Memo2->Lines->Strings[i].c_str());
        }
        fclose(stream);
        count_lineas++;
    }
    else
    {
        L.sprintf("%5.3lf\t%5.3lf\t%5.3lf\t%c",l1,l2,l3,status_l1);
        Memo2->Lines->Clear();
        Memo2->Lines->Add(L);
        count_lineas=0;
    }
    opc = 't';
    indexL = 0;
    ApdComPort1->FlushInBuffer();
}

}

//-----
void __fastcall TForm1::t_esperaTimer(TObject *Sender)
{
    opc = 't';
    Edit11->Text="Habilitado";
    t_espera->Enabled = false;
}

//-----
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
    Form1->Close();
    ApdComPort1->SWFlowOptions=swfNone;
}

//-----
void __fastcall TForm1::FormCreate(TObject *Sender)
{
    ApdComPort1->BufferFull=(0.75*ApdComPort1->InSize);
    ApdComPort1->BufferResume=(0.25*ApdComPort1->InSize);
    ApdComPort1->SWFlowOptions=swfBoth;
}

//-----

```

Apéndice E

Programa de RNA en MCU 18F4620

```
#include <16f877a.h>
//#include <18F4620.h>
#include <ctype.h>
#device *=16 ADC=10
#fuses HS, NOBROWNOUT, NOLVP, NOWDT
#use delay (clock=12000000)
#use rs232 (baud=9600, PARITY=N, STOP=1, BITS=8, xmit=PIN_C6, rcv=PIN_C7)
#include <math.h>

//////////////////////////////////////VARIABLES
int16 ti=0,te=0,hi=0,he=0,count1=0,count2=0,l1=0,l2=0,l3=0;
int high,low;
int opc,opc2='a',opc3='a';

//Variable temperatura interna y externa
int Tmin=0, Tmax=60, S_Ti=0, S_Te=0;

int16 Ti_pic=0, Te_pic=0;
int16 aux_Ti=0, aux_te=0;

float S1_Ti=0.0, S2_Ti=0.0, S3_Ti=0.0;
float S1_Te=0.0, S2_Te=0.0, S3_Te=0.0;

void temp_int (void)
{
```

```

float Ti_w1[3][1] = {0.7889, 64.8877, -4.3474};
float Ti_b1[3][1] = {-4.4492, -11.6445, -13.5224};
float Ti_w2[3][3] = {3.0940, -0.9996, 0.0196, 0.0003, 0.0000, 1.0093, 3.0937, -0.9996, -0.9897};
float Ti_b2[3][1] = {0.9794, -0.0000, 0.9794};
int i,j,k;
float aux[3][1] = {0,0,0};
opc='a';
Ti_pic=Ti_pic*4;
Ti_pic=58.143265-0.0306697*Ti_pic+0.0000163*pow(Ti_pic-1135.94,2.0)-1.2623E-8*pow(Ti_pic-1135.94,3.0);

if (Ti_pic<0)//normalizacion de datos
    S_Ti=(0x0F);
//if (Ti_pic>-250 || Ti_pic<-10)
//    S_Ti=(0xF0);
if (Ti_pic < 60 || Ti_pic > 5)
    S_Ti=(0xFF);

Ti_pic=(Ti_pic-Tmin)/(Tmax-Tmin);

for (i=0;i<3;i++)
{
    for (j=0;j<1;j++)
    {
        Ti_w1[i][j]=Ti_w1[i][j]*Ti; //Pesos capal por entrada
        Ti_b1[i][j]=Ti_b1[i][j]+Ti_w1[i][j]; //Pesos capal + bias capal
        Ti_b1[i][j]=1/(1+exp(-1*Ti_b1[i][j])); //sigmoide a cada elemento
        //OK
    }
}
//Multiplicando los pesos de la capa2 por el resultado de sigmoide
for (i=0;i<3;i++)
{
    for (j=0;j<1;j++)
    {
        for (k=0;k<3;k++)
        {
            aux[i][j]=Ti_w2[i][k]*Ti_b1[k][j]+aux[i][j];
        }
    }
}
S1_Ti=Ti_b2[0][0];

```

```

    S2_Ti=Ti_b2[1][0];
    S3_Ti=Ti_b2[2][0];
}

void temp_ext (void)
{
    float Te_w1[3][1] = {25.6155, 116.4129, -4.3805};
    float Te_b1[3][1] = {-15.7515, -14.9830, -15.8646};
    float Te_w2[3][3] = {2.7835, -0.9719, -0.0042, -0.0001, 0.0017, 1.0764, 2.7833, -0.9736, -1.0806};
    float Te_b2[3][1] = {1.0038, -0.0017, 1.0055};
    int i,j,k;
    float aux[3][1] = {0,0,0};
    opc='a';
    Te_pic=Te_pic*4;
    Te_pic=58.143265-0.0306697*Te_pic+0.0000163*pow(Te_pic-1135.94,2.0)-1.2623E-8*pow(Te_pic-1135.94,3.0);

    if (te_pic<0)//normalizacion de datos
        S_Te=(0x0F);//desc to
    //if (Te_pic>-250 || Te_pic<-10)
    //    S_Te=(0xF0);//des sen
    if (Te_pic < 60 || Te_pic > 5)
        S_Te=(0xFF);//ok

    for (i=0;i<3;i++)
    {
        for (j=0;j<1;j++)
        {
            Te_w1[i][j]=Te_w1[i][j]*Te; //Pesos capal por entrada
            Te_b1[i][j]=Te_b1[i][j]+Te_w1[i][j]; //Pesos capal + bias capal
            Te_b1[i][j]=1/(1+exp(-1*Te_b1[i][j])); //sigmoide a cada elemento
            //OK
        }
    }

    for (i=0;i<3;i++)
    {
        for (j=0;j<1;j++)
        {
            for (k=0;k<3;k++)
            {
                aux[i][j]=Te_w2[i][k]*Te_b1[k][j]+aux[i][j];
            }
        }
    }
}

```

```
                //OK
            }
        }
    for (i=0;i<3;i++)
    {
        for (j=0;j<1;j++)
        {
            Te_b2[i][j]=aux[i][j]+Te_b2[i][j]; //Salidas de la RNA
        }
    }

    S1_Te=Te_b2[0][0];
    S2_Te=Te_b2[1][0];
    S3_Te=Te_b2[2][0];
}

void main()
{
    //configuracion del ADC
    setup_port_a( ALL_ANALOG );
    setup_adc( ADC_CLOCK_INTERNAL );

    while(TRUE)
    {
        if (kbhit())
        {
            opc='a';
            opc=getc();
            switch(opc)
            {
                case 's':
                {

                    //Temperatura interna
                    opc='a';
                    set_adc_channel(0);
                    delay_ms(1);
                    ti=read_adc();

                    high=(ti >> 8) & 0x0FF;
                }
            }
        }
    }
}
```

```
low=ti & 0x0FF;
//red neuronal
putc(high);
delay_ms(10);
putc(low);
delay_ms(10);

if (ti>0x0226)//Desc.
    S_Ti=3;
else if (ti>aux_Ti+0x55 || ti<aux_Ti-0x55)
    S_Ti=2;
else //normal
    S_Ti=1;

//temp_int();

putc(S_Ti);
delay_ms(100);
aux_Ti=ti;

//Temperatura externa
opc='a';
set_adc_channel(1);
delay_ms(1);
te=read_adc();

high=(te >> 8) & 0x0FF;
low=te & 0x0FF;
putc(high);
delay_ms(10);
putc(low);
delay_ms(10);
//red neuronal

if (te>0x0226)//Desc.
    S_Te=3;
else if (te>aux_Te+0x55 || te<aux_Te-0x55)
    S_Te=2;
else //normal
    S_Te=1;
```

```
//temp_ext();

putc(S_Te);
delay_ms(100);
aux_Te=te;

//Humedades
/*delay_ms(1);
count1 = 0;
opc='m';
opc2='a';
while(opc2!='x')
{
    while(input(PIN_B7)); //Mientras alto
    while(!input(PIN_B7)); //Mientras bajo //1 Hz
    count1++;
    if (kbhit())
    {
        opc2=getc();
    }
}
hi = count1;
high =(hi >> 8) & 0x0FF;
low = hi & 0x0FF;
putc(high);
delay_ms(1);
putc(low);

delay_ms(1);
count2 = 0;
opc='m';
opc3='a';
while(opc3!='x')// 'z'
{
    while(input(PIN_B6)); //Mientras alto
    while(!input(PIN_B6)); //Mientras bajo //1 Hz
    count2++;
    if (kbhit())
    {
        opc3=getc();
    }
}
```

```
    }
    he = count2;
    high =(he >> 8) & 0x0FF;
    low = he & 0x0FF;
    putc(high);
    delay_ms(1);
    putc(low);*/
}
break;
case '1':
{
    //Linea 1
    delay_ms(1);
    set_adc_channel(2);
    delay_ms(1);
    l1=Read_ADC();

    high=(l1 >> 8) & 0x0FF;
    low=l1 & 0x0FF;
    putc(high);
    delay_ms(10);
    putc(low);

    //Linea 2
    delay_ms(10);
    set_adc_channel(3);
    delay_ms(1);
    l2=Read_ADC();

    high=(l2 >> 8) & 0x0FF;
    low=l2 & 0x0FF;
    putc(high);
    delay_ms(10);
    putc(low);

    //Linea 3
    delay_ms(10);
    set_adc_channel(4);
    delay_ms(1);
    l3=Read_ADC();
```



```
        high=(13 >> 8) & 0x0FF;
        low=13 & 0x0FF;
        putc(high);
        delay_ms(10);
        putc(low);
    }
    break;
}
} //if kbhit
} //while
}
```