TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

Localización de fugas hidráulicas en ductos aplicando redes neuronales artificiales

Tesis que presenta

Esvan de Jesús Pérez Pérez

Como requisito para obtener el grado de

Maestro en Ciencias en Ingeniería Mecatrónica

Director de tesis: **Dr. Francisco Ronay López Estrada (ITTG)**
Codirector de tesis: **Dr. Jesús Darío Mina Antonio (CENIDET)**
Asesores: **Dr. Elías Neftalí Escobar, M.C. Rafael Mota Grajales**

Tuxtla Gutiérrez, Chiapas, México

Febrero de 2019

SEP
SUBSECRETARÍA DE EDUCACIÓN SUPERIOR

DIRECCIÓN GENERAL DE EDUCACIÓN SUPERIOR TECNOLÓGICA
Instituto Tecnológico de Tuxtla Gutiérrez

Tuxtla Gutiérrez, Chiapas, [redacted]

OFICIO No.No. DEPI-xxxx/2024.

C. SUSHEYTO DE LOS SANTOS RUIZ
PASANTE DE LA MAESTRÍA EN CIENCIAS EN INGENIERÍA MECATRÓNICA
EGRESADO DEL INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ.
P R E S E N T E.

Habiendo recibido la comunicación, por parte de los CC., DR. EDIG NEFTALÍ ESCOBAR GÓMEZ, DR. HÉCTOR RICARDO HERNÁNDEZ DE LEÓN Y DR. ALEJANDRO MEDINA SANTIAGO, en el sentido, de que se encuentra satisfactorio el contenido de la tesis denominada "CONTROLADOR DIFUSO PARA UN ROBOT MÓVIL CON TRACCIÓN DIFERENCIAL", elaborada por usted, como prueba escrita para obtener el Grado de Maestro en Ciencias en Ingeniería Mecatrónica, AUTORIZAMOS a que se proceda a la impresión de la misma.

Sin otro particular, quedamos de usted muy.

ATENTAMENTE
"Por mi patria y por mi bien"
Tux., Xx.

M. C. LUCÍA MARÍA CRISTINA VENTURA CARRIZOS
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN.

ING. JOSÉ LUIS MÉNDEZ [illegible]
DIRECTOR

C.p. Coordinación correspondiente.
C.p. Archivo.

[illegible]

DIRECCIÓN DE EDUCACIÓN SUPERIOR TECNOLÓGICA
INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

Soli Deo Gloria

Resumen

Las redes neuronales se han aplicado actualmente para trabajar con datos de sistemas complejos, obteniendo resultado favorables con respecto a métodos convencionales, particularmente en diagnósticos de sistemas dinámicos para detección de fallas, basado en datos. Alentados por los resultados, en el presente trabajo se proponen y evalúan dos Arquitecturas de redes neuronales para realizar la localización de fugas hidráulicas en ductos. Empleando la planta piloto que está localizada en el laboratorio de hidráulica del TecNM-ITTG, se realizan experimentos para una base de datos del comportamiento de la planta piloto. Se realiza la metodología para cumplir con los objetivos planteados. Inicialmente, se emplea una red neuronal para obtener el factor de fricción que es un parámetro muy importante en hidráulica de fluidos. Secuencialmente, se emplea datos experimentales obtenidos de la planta piloto del TecNM-ITTG, para enriquecer la base de datos se utiliza un simulador validado. Con los datos experimentales y los datos de simulación, se crea una Base de datos completa, que servirá para entrenamiento y validación de una segunda y principal red neuronal para la localización de fugas hidráulicas. Como resultado se obtuvo una método robusto de localización de fugas, utilizando un conjunto de datos híbridos para el entrenamiento, validación y pruebas. La estimación de la posición de la fugas con la red fueron validadas con datos experimentales, dando un margen de error promedio del 0.629 % respecto a las posiciones reales en la tubería.

Abstract

Neural networks have now been applied to work with data from complex systems, obtaining favorable results with respect to conventional methods, particularly in diagnostics of dynamic systems for fault detection, based on data. Encouraged by the results, this paper proposes and evaluates two architectures of neural networks to perform the location of hydraulic leaks in ducts. Using the pilot plant that is located in the TecNM-ITTG hydraulics laboratory, experiments are carried out for a database of the behavior of the pilot plant. The methodology is carried out to comply with the objectives set. Initially, a neural network is used to obtain the friction factor that is a very important parameter in fluid hydraulics. Sequentially, experimental data obtained from the pilot plant of the TecNM-ITTG is used, to enrich the database a validated simulator is used. With the experimental data and the simulation data, a complete database is created, which will serve for training and validation of a second and main neural network for the location of hydraulic leaks. As a result, a robust method for locating leaks was obtained, using a set of hybrid data for training, validation and testing. The estimation of the position of the leaks with the network were validated with experimental data, giving an average margin of error of 0.629 % with respect to the actual positions in the pipeline.

AGRADECIMIENTOS

Mi más grande agradecimiento y dedicación para mi Padres por su amor y gran apoyo que me han brindado, junto a mis hermanos que son mis grandes amigos para toda la vida. Todo lo que hago es por ellos, por su cariño y motivación. Gracias por enseñarme amar a Dios y seguir el camino de la fe.

Gracias a mis amigos y camaradas de la vida, por su apoyo sin condiciones ni pretextos. Que estuvieron en los momentos de alegrías y dificultades. A Carlos por su amistad por años y compañero de inicio a fin de esta maestría. A mis grandes amigos del grupo Turix, Peter, Julio, Peñate, gracias por los consejos. A mi camarada Robert, por toda su ayuda en el desarrollo del proyecto. Con especial mención a mi amigo y mentor Ildeberto, gran ejemplo a seguir, preciso en sus consejos y aportaciones.

A mi director de tesis Dr. Ronay López-Estrada, gracias por su motivación, apoyo, asesorías y tiempo dedicado a mi persona para el desarrollo del proyecto. Al Dr. Jesús Mina, por sus valiosas aportaciones y asesorías. Al TecNM-ITTG, por permitirme realizar mis estudios en sus instalaciones. Se extiende este agradecimiento, al Dr. Carlos Ríos por todo el apoyo y soporte brindado, al Dr. Elías Escobar y M.C. Rafael Mota por las asesorías y recomendaciones para realizar la tesis.

A la Dra. Lizeth Torres, por su gran apoyo, aportaciones y valiosa experiencia brindada para realizar la tesis y recibirme durante la estancia en el Instituto de Ingeniería de la UNAM. Al Dr. Vicenç Puig, por su genialidad y darme la oportunidad de trabajar en el Institut de Robòtica i Informàtica en la UPC.

Finalmente, al Consejo Nacional de Ciencia y Tecnología por el soporte económico y recursos necesarios para completar el proyecto.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

> Las dificultades preparan a personas comunes para destinos extraordinarios.
>
> C.S. LEWIS

La distribución eficiente de fluidos en redes de ductos es un tema de gran importancia ya sea de plantas industriales, sistemas de distribución de agua y combustibles, entre otros (Ocampo-Martinez et al., 2013). Sin embargo, como todo sistema de distribución de líquidos, los ductos están sujetos a factores que ocasionan fugas como el envejecimiento, fallas en la instalación, eventos naturales, corrosión, entre otros. Las fugas causan pérdidas económicas, dañan la infraestructura del sistema de distribución, causan daños al medio ambiente y en ciertos casos, como en las fugas de hidrocarburos, representan riesgos para la salud (Olivera-Villaseñor and Rodríguez-Castellanos, 2012). Por lo cual, los estudios dedicados a la detección de fugas justifican su pertinencia desde una perspectiva científica, tecnológica y social.

De acuerdo con datos de Organización para la Cooperación y el Desarrollo Económicos (OCDE), en la Ciudad de México se reportó, a inicios del 2016, una pérdida mayor al 40 % en los sistemas de distribución de agua debido a fugas (OECD, 2016); aun considerando que a determinadas horas del día se cierran algunos tanques y bajan la presión en ciertos sectores de la red, ya que manteniendo un servicio continuo las 24 horas las fugas alcanzarían el 60 %. Entre las diferentes ciudades que contempla el estudio de la OCDE, Tuxtla Gutiérrez, Chiapas, se corona como la ciudad con más perdidas de agua, por encima de ciudades como la Ciudad de México, Nueva York, Pekin, entre otras; esto debido a que las pérdidas por fugas

en los sistemas de agua potable alcanza en promedio el 70 % (Figura 1.1). En este sentido, la detección y localización de fugas en redes de tuberías es un problema de interés internacional y nacional, ya que las fugas restringen el derecho al agua, tal como se menciona en resolución 64/292 de la Asamblea General de las Naciones Unidas. Dicha resolución menciona además la importancia de que los gobiernos redoblen esfuerzos para satisfacer la necesidad de derecho al agua (ONU-Hábitat, 2010). Por lo cual, la supervisión y el monitoreo automático de los sistemas de distribución es un reto para la ingeniería y su objetivo consiste en localizar tan rápido como sea posible, la presencia de fugas con un mínimo de instrumentación y costo.

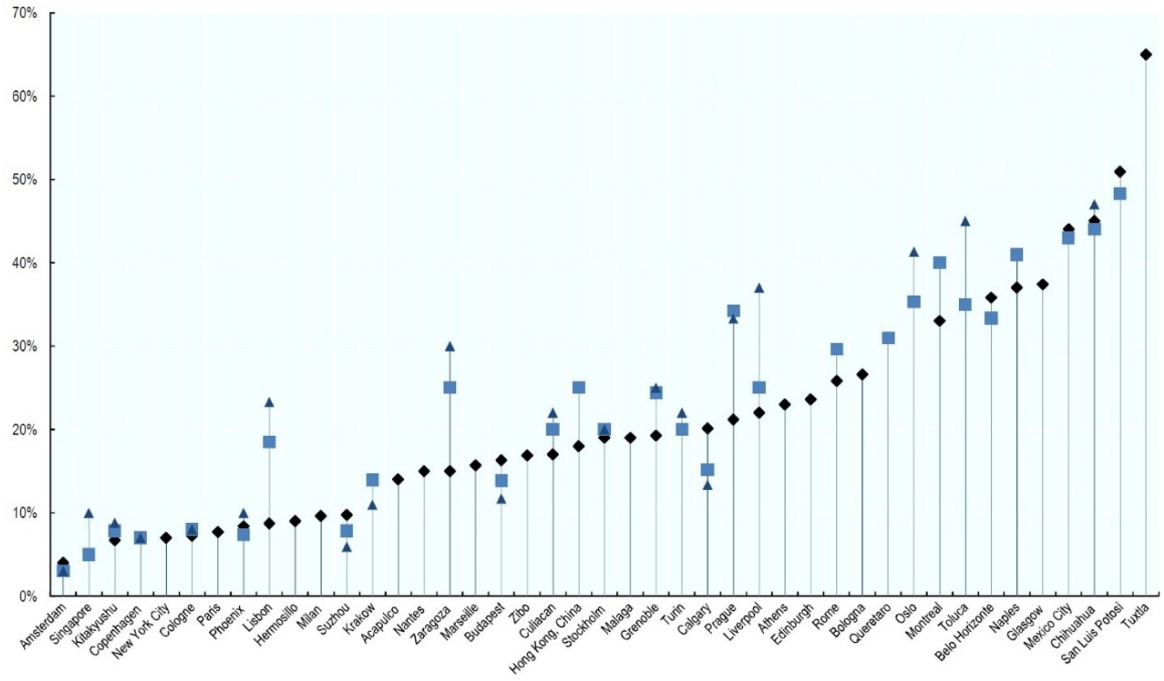


Figura 1.1: Proporción de perdidas de agua en ciudades encuestadas (OECD, 2016)

Una fuga en una tubería presurizada provoca una discontinuidad en la dinámica del fluido, es decir, el caudal antes de la fuga no es el mismo que el caudal después de la fuga, esto significa que el número de Reynolds aguas arriba de la fuga es diferente al número de Reynolds aguas abajo, y como consecuencia, la fricción cuasi-estática también es diferente antes y después de la fuga (Verde et al., 2014). Muchos de los algoritmos de diagnóstico de fugas que existen, se han diseñado asumiendo que la fricción es constante. Esta hipótesis es válida para tuberías muy rugosas y con un caudal turbulento. Pero en situaciones diversas esta hipótesis se vuelve invalida (Rojas et al., 2018).

La manera consensuada por la comunidad hidráulica para calcular la fricción cuasi-estática en régimen turbulento es utilizando la ecuación de Colebrook, que es una ecuación implícita que requiere resolverse con métodos iterativos, lo que se traduce como un costo computacional que se suma al costo computacional del algoritmo de diagnóstico de fugas. Con el fin de evitar esta situación, se han propuesto diferentes aproximaciones explícitas de la ecuación de Colebrook, las cuales calculan la fricción con cierto error de aproximación. Es importante notar que al estar trabajando con modelos dinámicos que representan un sistema hidráulico, existen parámetros que son muy sensibles ya que su comportamiento depende fuertemente de otros parámetros, principalmente con la fricción, la cual depende del caudal y presión. La ecuación de Colebrook esta definida como:

$$\frac{1}{\sqrt{f}} = -2 \cdot log_{10}\left(\frac{2.51}{Re \cdot \sqrt{f}} + \frac{\varepsilon}{3.7 \cdot D}\right) \tag{1.1}$$

Donde Re es el número de Reynolds, ε/D es la rugosidad relativa y f es el factor de fricción. Nótese que para la solución a la ecuación (1.1) se encuentra de forma implícita, por lo que para calcular f es necesario utilizar métodos iterativos o considerar aproximaciones.

En la literatura existen diversas técnicas para el cálculo del factor de fricción, un método es la solución por medio del algoritmo de Newton-Raphson (Yıldırım, 2009), la ecuación de Moody (Haaland, 1983), Churchill (Churchill, 1973), Swamee-Jain (Swanee and Jain, 1976), Serghides (Serghides, 1984), Brkic (Brkić, 2011), por mencionar las más importantes. Sin embargo, al ser aproximaciones existe un error derivado de los cálculos numéricos y del método utilizado. En el trabajo de Brkić (2011) se realizó una comparación de los errores de diversos métodos de aproximación, dando como resultado que las aproximaciones con menor error máximo se encontraban por debajo del 0.14 % y por otro lado las aproximaciones con mayor error relativo se encontraban cercanos al 8.0 %. Derivado de ello, recientemente se ha propuesto el uso de redes neuronales artificiales (RNA) como una alternativa para calcular la fricción y reducir el error de aproximación típicamente presentado por las expresiones implícitas. Particularmente, los autores en Brkić and Ćojbašić (2016) propusieron la arquitectura de una RNA que calcula la fricción con un error de 0.07 %, lo cual es menor al error que se obtendría si se calculara la fricción empleando algunas de las aproximaciones explícitas más usadas por

la comunidad hidráulica, sin embargo, no se consideraron datos experimentales para entrenamiento y validación de la RNA. La importancia de trabajar con datos experimentales radica en verificar que los algoritmos y métodos de detección de fugas son factibles y funcionales en los sistemas reales de ductos donde diversos factores afectan la dinámica del sistema, y el efecto de las variaciones en los parámetros de la planta debido a factores poco predecibles o controlables (Gao et al., 2015), tales como fricción, rugosidad relativa, flujo másico, densidad, entre otros. Este hecho motivó al desarrollo de este trabajo en la que se considera la estimación de un parámetro tan importante como es la fricción con una RNA, considerando datos experimentales de la planta piloto del TecNM-ITTG, con el fin de incorporar la estimación de la fricción.

Este trabajo propone una metodología con un enfoque basado en Inteligencia Artificial para localizar fugas hidráulicas. Para lograr este objetivo, se propone utilizar una RNA para calcular los valores de la fricción cuasi-estática en una tubería después de la aparición de una fuga a partir de la medición de los caudales en los extremos y considerando mediciones de presión en la entrada y salida de la tubería (aguas arriba y aguas abajo). En conjunto las fricciones y los datos de presión conforman la base de datos para entrenamiento de una segunda RNA que estima la posición de la fuga. Como resultado, se obtuvo un método de diagnóstico robusto capaz de predecir las ubicaciones de las fugas con un error porcentual promedio de 0.629 %.

1.1. Estado del arte

No hay ningún sistema en el mundo que esté exento de perturbaciones, interferencias y fallas. Las redes de distribución de agua no son la excepción. Por esta razón, necesitan estar equipados con sistemas de detección y aislamiento de fallas que permitan el diagnóstico oportuno de anomalías comunes, como fugas, bloqueos o sensores no calibrados. El diagnóstico y ubicación de fugas ha sido ampliamente estudiado desde el punto de vista del método de balance de masas, el cual es idóneo para los sensores que se tienen instalados en la mayoría de los ductos, sensores de flujo y presión en los extremos, ya que es fácil calcular el gasto másico

que circula por el ducto y reconoce el momento en que se presenta una fuga debido a que existe un desbalance entre el gasto másico de entrada y de salida (Verde et al., 2016).

Es innegable que la supervisión y seguridad de procesos industriales como el transporte de fluidos es de suma importancia para garantizar el consumo y costos. Isermann (1984) menciona que la supervisión de los procesos técnicos es un tema de mayor desarrollo debido a las crecientes demandas de confiabilidad y seguridad, además que, es posible monitorear variables no mensurables como estados de proceso, parámetros de proceso y cantidades características. La rama de control automático encargado del desarrollo de tales sistemas es la detección y diagnóstico de fallas. Nótese que una fuga representa una falla en el sistema, debido a que se ve como una desviación anómala de los parámetros considerados normales. Existen métodos de diagnostico de fallas basados en modelos, e.g. en Billmann and Isermann (1987) propusieron un método para la detección temprana y la localización de pequeñas fugas en tuberías, desarrolló un observador de estado adaptativo no lineal y una técnica de correlación especial, basada en las mediciones de presión y flujo en la entrada y salida de la tubería. En Afzal and Udpa (2002) desarrollaron una técnica para detección de fugas de gas natural, el cual consiste de procesamiento de señales y un filtro adaptativo para eliminación de ruido en los datos obtenidos de la inspección de fugas de flujo magnético. En el estudio realizado por Besançon et al. (2007) presentaron en simulación un enfoque en línea basado en observadores para la detección de fugas en tuberías. En efecto, dado que se pueden detectar varias fugas, se proporciona un observador que apunta a la estimación simultánea de las magnitudes y las posiciones de las fugas sobre la base de algún modelo apropiado, de tal manera, en Reddy et al. (2011) desarrollaron una metodología para detectar y localizar las fugas que se producen en los gasoductos, para tuberías simples y redes de ductos, empleando una técnica de estimación de estado eficiente basada en un modelo de función de transferencia. También, se desarrolló un método de localización de fugas basado en las mediciones de presión y el análisis de sensibilidad de presión de los nodos en una red, donde se propone una metodología de colocación de sensores óptima, basada en algoritmos genéticos (Pérez et al., 2011), estas metodologías se han aplicado a la Red de Barcelona utilizando el simulador PICCOLO. La ubicación del sensor y las metodologías de detección y localización de fugas se aplican a varias áreas de administración del distrito (DMA) en la simulación y en la realidad.

Se han reportado métodos alternativos para detección y localización de fugas. Por ejemplo, en Meng et al. (2012) realizó un estudio experimental con señales acústicas, se caracteriza por ser no invasivos en el sistema, el método consiste en el análisis de vibraciones que se produce al presentarse una fuga. Esta técnica también fue presentada en Mostafapour and Davoudi (2013) donde hicieron el análisis de señales acústicas obtenidas de mediciones de sensores de una tubería de gas, obteniendo una estimación aproximada de la fuga en 6 %, a diferencia con el trabajo de Sun and Chang (2014) donde utilizó una onda de presión negativa (NPW) para localizar fugas. Sin embargo, el uso de estas técnicas requieren de muchos puntos de mediciones, además, el problema de este tipo de técnicas es la alta sensibilidad a ruido externo, el cual dificulta el análisis de la señal producida por la fuga.

Recientemente se ha propuesto algunos enfoques para detección y aislamiento de fugas basados en modelos mediante la aplicación de diferentes filtros para estimación de estados. En Arifin et al. (2015) se utilizó un filtro de partículas como sensor virtual para estimar los estados en ubicaciones intermedias; los residuos entre los estados estimados y las mediciones reales en las ubicaciones intermedias se utilizan para la detección y localización de fugas. Posteriormente, Delgado-Aguiñaga et al. (2016) propuso un enfoque de que se basa en un modelo no lineal, derivado de las ecuaciones de Water-Hammer y en el Filtro de Kalman Extendido (EKF), que se utilizan para estimar los coeficientes de fuga y monitorear cada nueva fuga. Igualmente, en Verde and Rojas (2017) usaron el EFK para desarrollar un esquema recursivo simple, para detectar y aislar un conjunto de fugas secuenciales en una tubería. De la misma manera, los autores en Santos-Ruiz et al. (2018) presentaron una metodología para la detección y aislamiento de fugas (LDI) en tuberías basadas en la fusión de datos desde dos enfoques: una estimación de estado estable y EKF. El método propuesto considera solo las medidas de presión y caudal en los extremos de la tubería, que contienen sensores intrínsecos y ruido de proceso. El promedio de error de la estimación de fugas es inferior al 3 % en la posición de fuga. Además, la incorporación de una estimación de estado estable muestra que la solución del problema de LDI ha mejorado significativamente con respecto a la que solo considera la estimación de EKF. También se realizó un análisis experimental sobre la efectividad del enfoque propuesto para diferentes tasas de muestreo y para diferentes posiciones de fuga. No obstante la efectividad que han demostrado los métodos basados en

modelos, en diferentes ocasiones resultan muy complejos de aplicar debido sobre todo a la dificultad de caracterizar dichos modelos.

En múltiples trabajos se han propuestos técnicas de detección de fallas usando métodos basados datos tales como Machine Learnig o aprendizaje automático, Maquina de soporte de vectores (SVM), Redes Neuronales Artificiales (RNA), Análisis de Componentes Principales (PCA), entre otros (Murvay and Silea, 2012). En Goldberg (1987) desarrolaron técnicas relacionadas algoritmos genéticos aplicados al problema del control de un gasoducto, se aplicaron algoritmos genéticos a dos problemas de optimización de tuberías, los algoritmos genéticos se utilizan como un mecanismo de aprendizaje básico en un sistema de aprendizaje de reglas más grande denominado sistema clasificador de aprendizaje (LCS). Chen et al. (2004) propuso considerar técnicas de MVS para detectar ondas de presión negativas (NPW). En el enfoque, la detección de NPW se formula como un problema de aprendizaje supervisado y el método de SVM se emplea para desarrollar el algoritmo de detección. Del mismo modo, en Asefa et al. (2005) se presentó una aplicación hidrológica de una nueva metodología de aprendizaje estadístico con SVM, para detección de fugas y para reproducir el comportamiento de los modelos de flujo y transporte. Los autores en Khodayari-Rostamabad et al. (2009) aplicaron los métodos de SVM, técnicas de kernelización, PCA, mínimos cuadrados parciales y métodos para reducir la dimensionalidad del espacio de características; demostraron el rendimiento óptimo de estos métodos utilizando datos reales de MFL recopilados de tuberías. Además, se demostró que las estructuras de variables latentes de baja dimensión pueden ser efectivas para visualizar el comportamiento de agrupamiento en clúster del clasificador. De mismo modo, en Mashford et al. (2009) se estudió un método para predecir el tamaño y la ubicación de la fuga usando SVM con datos obtenidos de sensores de presión que monitorean la red de agua, el SVM fue entrenada y probada con datos de simulación de EPANET. También en Mandal et al. (2012) se propuso un nuevo esquema de detección de fugas basado en la teoría de conjuntos aproximados y SVM para superar el problema de la detección de fugas falsas. En este enfoque, se explora la "teoría de conjuntos aproximados"para reducir la longitud de los datos experimentales y generar reglas. Está incrustado para mejorar el proceso de toma de decisiones. Por otra parte, los autores en Liang and Zhang (2012) propusieron un nuevo método de detección de fugas en tuberías basado en datos con PCA, durante los períodos tran-

sitorios sin historial de fugas, capturan las características dinámicas de los datos de presión no estacionarios. Para validar el algoritmo se consideró un estudio de caso experimental.

Diversos autores han trabajado con RNA para monitoreo y diagnóstico en tuberías e instalaciones hidráulicas, trabajando con modelos probabilísticos y datos obtenidos de diversas plantas hidráulicas. Algunas de las fugas son ocasionadas por fallas en los actuadores o válvulas de control, por lo que, en Linaric and Koroman (2003) se realizó el análisis y simulación de fallas en válvulas de control electro-hidráulicas mediante modelos de RNA. Un método para detección de fugas en procesos industriales con materiales químicos, fue realizado por Maki and Loparo (1997) en procesos que requieren la observación simultánea de datos múltiples. La característica principal del enfoque es que la detección de las fallas se produce durante los períodos transitorios de operación del proceso. La red detecta la tendencia dinámica de cada medición, y la segunda etapa de la red detecta y diagnostica las fallas. Una red neuronal de series de tiempo fue presentada por Mounce et al. (2002) para detección de fugas usando datos de mediciones de sensores para construir directamente un modelo empírico para predicción y clasificación de fugas. Con este tipo de enfoque, Caputo and Pelagagge (2002) diseñaron una RNA multicapa con el fin de localizar fugas basadas en la información de presión y caudal, con estrategias para generar datos de entrada y para correlacionar con RNA dichos datos con el estado del sistema de distribución de fluidos. Un sistema de detección de fugas para tuberías fue desarrollado por Salvatore et al. (2004) utilizando RNA para determinar el tamaño y la ubicación de las fugas. Este sistema puede detectar y localizar fugas hasta el 1 % de los caudales en tuberías que transportan materiales peligrosos en aproximadamente 100 [s]. Se realizó un preprocesamiento adecuado de los datos para compensar las variaciones operativas y para evitar las alarmas falsas. El paquete detecta fugas tan pequeñas como el 1 % del caudal de entrada y predice correctamente el segmento de fugas de la tubería con una probabilidad de éxito superior al 50 % para la fuga más pequeña. Una combinación de herramientas de agrupamiento y clasificación para la detección de fugas se propuso en da Silva et al. (2005), donde se usaron para clasificar el modo de ejecución e identificar los transitorios operativos y de proceso. El sistema Neuro-difuso se entrena inicialmente fuera de línea, la metodología se aplica a un caso de monitoreo de gasoducto. En Santos et al. (2013) realizaron una comparación entre Red Perceptrón Multicapa (MLP) y una Red de Función de Base Radial (RBF).

En el cual, usaron utilizando datos experimentales de un micrófono instalado dentro de una tubería de hierro galvanizado de 60 m de longitud, en diversas condiciones de operación. Las dinámicas de estos ruidos en el tiempo se utilizaron como entradas a los modelos neuronales para localizar y determinar la magnitud de las fugas. Los resultados obtenidos de Los conjuntos de prueba, con fugas causadas intencionalmente, mostraron que las dos estructuras neuronales fueron capaces de detectar y localizar fugas en las tuberías. Sin embargo, la red Perceptrón multicapa mostró un rendimiento ligeramente mejor. En Abdulla and Herzallah (2015) presentaron un sistema de toma de decisiones para la detección de fugas basado en técnicas clasificación con RNA, usaron datos de un sistema de distribución experimental. Donde las variables de presión y flujo se pueden usar como firma de fuga en las condiciones multi-operacionales diseñadas. Un método para la detección temprana y la predicción de fugas en calderas, fue desarrollado en Rostek et al. (2015) utilizando RNA recurrentes. Los resultados obtenidos se utilizaron en la implementación piloto de un sistema de diagnóstico y predicción que cubre seis bloques de una central eléctrica profesional. A su vez, una RNA con aprendizaje bayesiano para maximizar la precisión de la predicción de fugas, fue planteada por Leu and Bui (2016), el método propuesto también indica los factores más importantes que afectan la fuga de agua. Algunos métodos más novedosos se han publicado, por ejemplo, en Zadkarami et al. (2016) presentaron un método que no solo detecta la ocurrencia de una fuga, sino que también sugiere su ubicación y tamaño. Está compuesto por un clasificador de red neuronal de percepción de múltiples capas (MLPNN) junto a métodos de extracción de características que incluyen las técnicas estadísticas, la transformada ondícula y una fusión de ambos métodos. Posteriormente, en Zadkarami et al. (2017) presentaron de igual forma un método de diagnóstico de fugas con MLPNN fusionado con la técnica Dempster-Shafer, Ambos métodos se aplicaron a los primeros 20 km del oleoducto Golkhari a Binak ubicado en el sur de Irán. Un estudio experimental sobre la detección de fugas de un sistema de distribución de agua fue presentado en Li et al. (2018), en el cual, investigaron las características de las señales acústicas de fuga en los segmentos de tubo de conexión. Después, se establece un clasificador basado en RNA. El método tuvo una precisión de estimación de 97.2 % y 96.9 %. Por ultimo, en Kang et al. (2018) propuso un novedoso, rápido y preciso sistema de detección de fugas de agua con un diseño que fusiona una red neuronal convolucional unidimensional

y una máquina de vectores de soporte. Una red de tuberías de agua real está representada por una red gráfica y se supone que los eventos de fuga ocurren en puntos virtuales en la gráfica. El rendimiento se validó en un banco de pruebas basado en una red de sensores inalámbricos, implementado en una red de distribución real.

Con base en el estudio del estado del arte y en los trabajos previos, considerando las ventajas que ofrecen los métodos de detección y localización de fugas con RNAs, el cual permite ubicar las fugas con mayor precisión y reducir la carga computacional en el sistema de diagnostico. En este sentido, se plantea la propuesta de desarrollar un método de localización de fugas hidráulicas aplicando RNAs.

1.2. Planteamiento del problema

Los derrames y fugas de fluidos peligrosos en tuberías pueden representar un riesgo de seguridad significativo para la población, las plantas industriales y el medio ambiente. Por lo tanto, en la distribución de fluidos, se encuentra el desafió de monitorear el estado de las tuberías con el fin de identificar condiciones anormales y localizar fugas cuando aparecen. En la literatura, a pesar de los significativos esfuerzos realizados, una solución efectiva para el problema de monitoreo de ductos no se ha encontrado y todavía representa un desafío significativo.

En este trabajo se propone implementar un método para localizar de fugas usando Redes Neuronales Artificiales (RNA), las cuales han demostrado su utilidad y robustez con la capacidad de aprendizaje adaptativo. El reto principal consiste en implementar una metodología basado en RNAs, para estimación del factor de fricción y localización de fugas usando las presiones de entrada y salida junto con las fricciones aguas arriba y aguas abajo. Las RNAs son debidamente entrenadas con un conjunto de datos enriquecidos con datos del simulador de la tubería y datos experimentales obtenidos mediante sensores de presión y caudal. Dando como resultado un método confiable, robusto y eficiente para localizar fugas.

1.3. Hipótesis

La consideración de técnicas de RNAs estimarán las posiciones de las fugas con error porcentual promedio menor al 1 %.

Hipótesis nula

El error porcentual promedio de localización de fugas con la aplicación de RNAs es igual o mayor al 1 %.

Hipótesis nula: $H_0 : \mu \geq 1\,\%$

Hipótesis alternativa

El error porcentual promedio de localización de fugas con la aplicación de RNAs es menor al 1 %.

Hipótesis alternativa: $H_1 : \mu < 1\,\%$

1.4. Objetivos

1.4.1. General

Desarrollar una metodología para localizar fugas utilizando una RNA entrenada y validada utilizando datos de un simulador de la planta piloto del ITTG y datos experimentales (presiones y caudales).

1.4.2. Específicos

- Obtener un conjunto de datos de un simulador validado de la planta piloto (para entrenamiento, validación y pruebas de la RNA).
- Obtener un conjunto de datos experimentales para pruebas de la RNA.

- Implementar una RNA para la estimación de la fricción.
- Ordenar el conjunto de datos obtenidos del simulador y el conjunto de datos experimentales para el entrenamiento de la RNA.
- Diseñar una RNA para la localización de fugas utilizando la fricción y el cabezal de presión aguas arriba y aguas abajo.
- Implementar una RNA para calcular la posición de fuga con las fricciones aguas arriba y aguas abajo una vez que una fuga sucedió.

1.5. Justificación

Este trabajo tiene como objetivo principal desarrollar una metodología para la localización de fugas hidráulicas en ductos mediante la aplicación de RNAs. Esto puede ser útil para diversos sectores públicos y privados encargados del monitoreo y diagnostico de ductos de distribución, dando soporte para resolver algunos de los problemas de origen económico y ambientales que se presenten al ocurrir fugas hidráulicas. La importancia de este trabajo de tesis radica, entonces, en el impacto social que pudiera tener en un futuro ya que la detección oportuna de fugas se va a canalizar a las personas encargadas de tomar decisiones para darle solución al problema.

1.6. Organización del documento

El contenido de este documento está dividido en siete capítulos que se describen a continuación.

- En el Capítulo 2 se presentan los conceptos básicos de la teoría de detección y diagnóstico de fallas, para posteriormente analizar lo existente en el campo de detección de fallas haciendo uso de la teoría de control. Se describen los fundamentos para las Redes Neuronales. Se describen las redes perceptrón multicapa y las redes neuronales de retropropagación.

- En el Capítulo 3 se presenta el sistema experimental del cual se obtuvieron los datos experimentales con mediciones de presión y caudal. Se explica cómo y con base a que se creó el simulador que permitió la generación de los datos para el entrenamiento de la RNA-2.

- En el Capítulo 4 se expone la metodología y el diseño de experimentos que se llevaron a cabo para probar y validar el desempeño de las redes implementadas. En esta sección se explica como fue entrenada cada red, que tipo de datos para entrenamiento, validación y pruebas se utilizaron.

- En el Capítulo 5 se muestran los resultados del entrenamiento de la RNA-2 para localización de fugas y la validación del rendimiento de la RNA-2 con datos experimentales. Además, se realiza la prueba de hipótesis.

- En el Capítulo 6 se presentan las conclusiones obtenidas, de igual manera, se propone un panorama para trabajos futuros basados en los resultados obtenidos en esta tesis.

Capítulo 2

Fundamentos de detección y diagnóstico de fallas

Una falla es todo cambio en el comportamiento de alguno de los componentes del sistema, lo cual se aprecia como una desviación no permitida de alguna de sus propiedades o parámetros característicos, de manera que éste ya no puede satisfacer la función para la cual ha sido diseñado Blanke et al. (2006). De esta manera, una fuga en una red hidráulica se clasifica como falla, pues al ocurrir el sistema ya no opera dentro de su estado nominal. Todos los sistemas son susceptibles a las fallas. En algunos casos, los lazos de control ocultan las fallas evitando ser observadas hasta alcanzar un grado tal que produzcan una avería irreparable que obligue a detener del sistema o proceso. Por ello existe una creciente necesidad e interés en desarrollar sistemas de diagnóstico y control que coadyuvan a operar de forma aceptable incluso después de la aparición de una falla y que sean capaces de parar el proceso antes de que se originen daños irreparables en el mismo.

En este capítulo se presentara información relevante con respecto al marco teórico de la teoría de detección y diagnóstico de fallas. Se inicia explicando la metodología de diseño de estos sistemas y las características deseables de dichos sistemas. Se define el concepto de falla así como su clasificación.

2.1. Metodología de diseño de algoritmos de detección y diagnóstico de fallas

Existe una metodología de diseño de sistemas de diagnóstico de fallas propuesta por Blanke et al. (2006), la cual está dividida en cinco etapas. Las etapas de esta metodología se enumeran a continuación:

1. Análisis del sistema a dos niveles: a nivel de componentes mediante un análisis de propagación de fallas a través de todos los subsistemas más relevantes, así como una evaluación de la severidad de los mismos y a nivel de estructura de cara a analizar la redundancia presente en el sistema que ayudará en el diseño del sistema de diagnóstico y reposición.

2. Diseño del sistema de diagnóstico a partir del análisis estructural y teniendo en cuenta las medidas disponibles y las fallas que se desean diagnosticar. En el caso de que no se puedan diagnosticar todas las fallas que se deseen, se deberá modificar la instrumentación disponible hasta conseguirlo. El sistema de diagnóstico de fallas deberá no sólo detectar y aislar las fallas sino también estimar su tamaño.

3. Diseño de los mecanismos de tolerancia para cada uno de las fallas consideradas según se trate de fallas en sensores, actuadores y/o planta.

4. Diseño del supervisor a partir de la información acerca de las fallas proporcionada por el sistema de diagnóstico, el supervisor deberá activar los mecanismos de tolerancia que se han diseñado para cada uno de ellos.

5. Aplicación y pruebas en simulación y sobre el sistema real.

La primera tarea a realizar en un sistema de control tolerante activo consiste en el diagnóstico de la falla en tiempo real, llegando no sólo a su detección y aislamiento sino también a la estimación de su magnitud. Por lo tanto, el diagnóstico de falla se puede a su vez dividir en tres etapas según su profundidad:

- Detección de la falla: decisión de si existe o no una falla así como la determinación de su instante de aparición.
- Aislamiento de la falla: localización del componente en el cual se ha producido la falla.
- Identificación y estimación de la falla: identificación del modo de falla y estimación de su magnitud.

Como todo proceso, existen diversas características o elementos considerar en el diseño de algún algoritmo de diagnóstico. A continuación se presentan las características deseables en un sistema de diagnóstico:

- Rápida detección y diagnóstico: El sistema debe responder rápidamente en la detección y diagnóstico de fallas. Sin embargo esto es difícil de lograr debido a que una respuesta rápida implica que haya alta sensibilidad a las influencias de altas frecuencia.
- Aislabilidad: el sistema debe ser capaz de distinguir entre dos o más fallas. Es decir identificar la falla ocurrida de forma precisa.
- Robustez: el sistema debe ser robusto ante ruido e incertidumbre del modelo matemático, ruido de medición y perturbaciones externas. De forma que no se vea afectado o atenué el efecto de dichas señales espurias.
- Identificación de nuevas fallas: el sistema de diagnóstico debe ser capaz de decidir, si el proceso está trabajando de forma normal o no. En caso de que funcione de manera anormal, debe ser capaz de reconocer si se debe a una falla conocida o bien si se trata de un nuevo tipo de falla.
- Estimación del error de clasificación. Esta característica es para generar confianza en el usuario, con base a la estimación de posibles errores que se puedan presentar al realizar la clasificación.
- Adaptabilidad: Que el algoritmo sea capaz de adaptarse a cambios inesperados tanto de la estructura como del ambiente que rodea al sistema.

- Facilidad de explicación del origen de la falla: Además de la habilidad de identificar la fuente de la falla, un sistema debe ser capaz de proveer explicaciones de cómo se originaron y propagaron las fallas hasta las condiciones actuales del proceso.
- Identificación de múltiples fallas. Es un requerimiento difícil de lograr, pero es importante que el sistema pueda identificar varias fallas a la vez.
- Almacenamiento y requerimientos computacionales. Generalmente las soluciones rápidas en tiempo real requieren de algoritmos de baja complejidad computacional, sin embargo, requieren de mucho espacio de almacenamiento de datos. En la actualidad se prefiere buscar un balance entre estos requerimientos.

2.2. Clasificación de fallas

Una falla es definida como una desviación no permitida de al menos de una propiedad característica de una variable con respecto a un comportamiento estable o adecuado. Por lo tanto, una falla es un estado que puede llevar al sistema a funcionar incorrectamente o bien generar una avería operacional en el sistema (Glen et al., 2006).

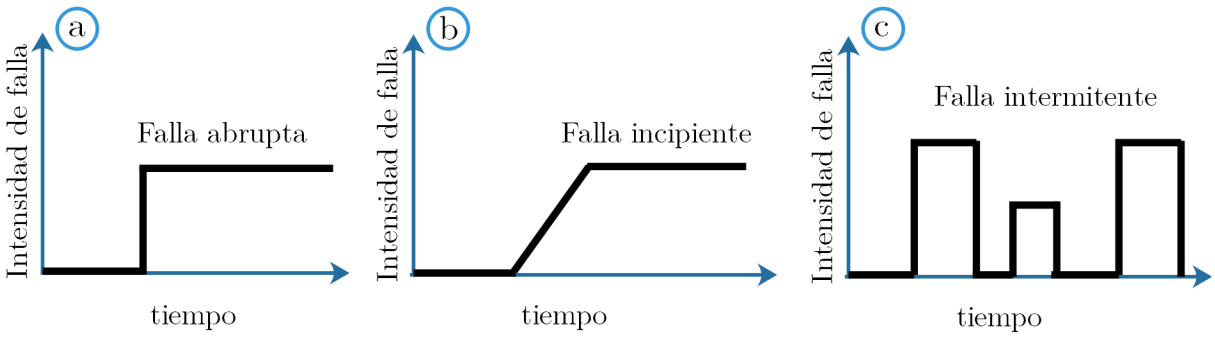


Figura 2.1: Clasificación de fallas respecto al tiempo

Según (Hamayun et al., 2016), las fallas pueden ser clasificadas en diversas categorías según el enfoque que se aplique. Tomando como base el tiempo de aparición de las fallas, estas pueden ser clasificadas en:

- Fallas abruptas: son aquellas cuyo efecto aparece repentinamente (por ejemplo, modelada mediante un escalón, Figura 2.1a).

- Fallas Incipientes: Son las fallas cuyo efecto aparece de forma progresiva. (Por ejemplo, modelada mediante una rampa, Figura 2.1b).
- Fallas intermitentes: son aquellas que si efecto aparece repentinamente, pero de igual forma desaparecen (por ejemplo, modelada mediante un tren de pulsos, Figura 2.1c).

Con respecto al modelo del proceso, las fallas pueden agruparse en las siguientes categorías:

- Fallas aditivas. Aparecen como desviaciones o sesgos del valor de la variable medida en los sensores.
- Fallas multiplicativas. Afectan directamente en cambios en los parámetros del proceso.

Ahora bien, según el alcance de las fallas dentro del sistema, estas puedes ser clasificadas como locales o globales. Y con base a su forma en sistemáticas o aleatorias. En la Tabla 2.1 se muestra la variedad de de los tipos de fallas y su relación con los componentes del sistema.

Tabla 2.1: Tipos de Fallas y su relación con los componentes

Tipo de Falla		Componentes			
		Mecánicos	Eléctricos	Electrónicos	Software
Forma	Sistemáticas	✓		✓	✓
	Aleatorias		✓	✓	
Proceder ante el tiempo	Abrupta			✓	✓
	Incipiente	✓	✓	✓	
	Intermitente		✓	✓	✓
	Ruido		✓	✓	
Alcance	Local	✓	✓	✓	✓
	Global				

2.3. Métodos de detección y diagnóstico de fallas

Durante las últimas décadas las investigaciones teóricas y experimentales han demostrado nuevas maneras de detectar y diagnosticar fallas. Se distingue dos tareas primordiales, la

detección de la falla, que es tan sólo el reconocer que una falla ha ocurrido; y la segunda tarea es la de diagnosticar la falla es decir, encontrar la causa y la ubicación de la falla. Muchos métodos avanzados para la detección de fallas están basados en modelos matemáticos de procesos y en la teoría de sistemas con el objetivo de generar los síntomas de las fallas. Por su parte, los métodos de diagnóstico, son generalmente relaciones causales del tipo falla-síntoma y están basados en estadísticos, inteligencia artificial y "soft computing". Esto genera dos campos o áreas de estudio: Detección y Aislamiento de Fallas (FDI por sus siglas en inglés) y Detección y Diagnóstico de Fallas (FDD).

Clasificación de los Métodos de Detección y Diagnóstico de Fallas

En un modelo o algoritmo de diagnóstico de fallas existen dos componentes principales: a) el tipo de conocimiento que se posee y b) el tipo de estrategia de búsqueda del diagnóstico. Esta estrategia es una función del esquema de representación de conocimiento, el cual es altamente influenciado por el tipo de conocimiento a-priori que se tenga del sistema. Por lo tanto, la información a-priori es el elemento más importante en los sistemas de diagnóstico, y es lo que permite que se hayan desarrollado diversas técnicas, según la información que se tenga en el momento de diseño de algoritmo.

El conocimiento a-priori que se necesita para el diagnóstico de fallas es el conjunto de fallas y su relación con los síntomas observados al momento de la aparición de estas. Un sistema de diagnóstico debe de tener estas relaciones de forma explícita (como en una tabla de búsqueda) o que puedan ser inferidas de alguna fuente de dominio del conocimiento. Este conocimiento a-priori del dominio puede ser desarrollado de un entendimiento fundamental del proceso usando principios básicos del conocimiento humano, traducidos a un modelo causal del proceso en cuestión. Por otro lado, el conocimiento puede ser obtenido de experiencias pasadas; este conocimiento simple y evidente es conocido como conocimiento basado en historial. Además. la información obtenida atribuye al uso de métodos basados en los datos de los sistemas analizados.

Mientras que, el conocimiento basado en modelos puede ser clasificado en cualitativo o cuantitativo. Este modelo es desarrollado con base al entendimiento fundamental de las

propiedades físicas del proceso. En los modelos cuantitativos este entendimiento es expresado en términos de relaciones matemáticas funcionales entre las entradas y salidas del sistema. En contraste, en los modelos cualitativos estas relaciones son expresadas en términos de funciones cualitativas centradas alrededor de diferentes unidades en un proceso.

A diferencia con los enfoques basados en modelos (cualitativos o cuantitativos) en donde un modelo del sistema es conocido o aproximado, en los enfoques basados en el historial del proceso es de suma importancia contar con un gran número de datos históricos del proceso. Por consiguiente, en los métodos basados en datos, es necesario contar con la suficiente cantidad de información relevante acerca de la dinámica del sistema, para aplicar los algoritmos de diagnóstico.

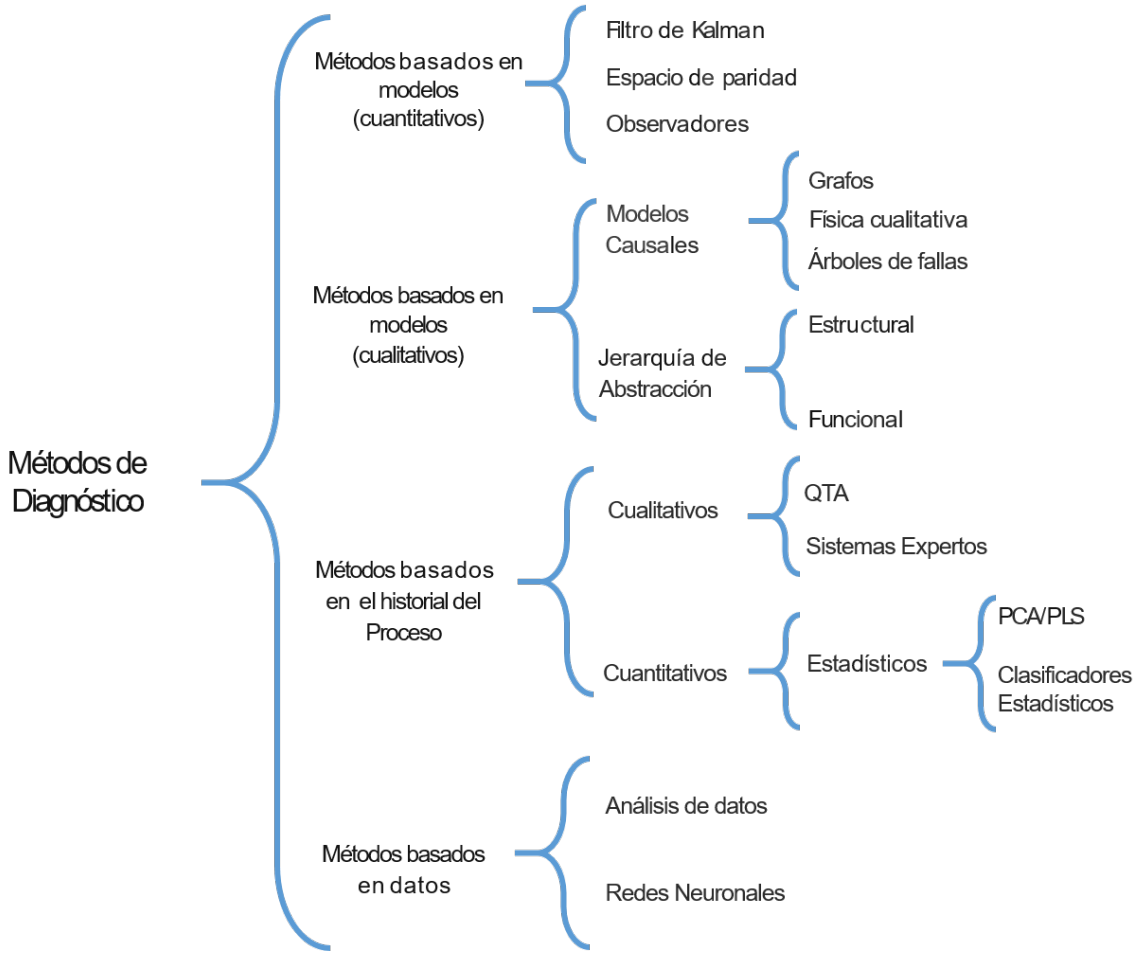


Figura 2.2: Clasificación de los métodos de detección y diagnóstico de fallas

Existen diversas formas en la que esta información puede ser transformada y presentada

como conocimiento a-priori al sistema de diagnóstico. Este proceso es conocido como extracción de rasgos y es realizado para facilitar la tarea de diagnóstico. Este proceso puede ser cualitativo o cuantitativo.En la Figura 2.2, se muestra una clasificación de diversas técnicas o algoritmos de detección y diagnóstico de fallas, en la literatura existen cuatro grandes grupos de técnicas de diagnóstico de fallas (Venkatasubramanian et al., 2003):

2.3.1. Métodos Cuantitativos basados en modelos

Como su nombre lo dice son métodos basados en modelos determinísticos del comportamiento del sistema. Es decir, con base a un modelo matemático que modele la relaciones de las variables del proceso. Con base a este modelo conocido, es posible generar relaciones de redundancia que permitirán generar residuos y de esta forma poder aislar o identificar una falla. Algunos métodos que caen en esta categoría son: Observadores para sistemas dinámicos (Garcia and Frank, 1997), ecuaciones de paridad (Gertler, 1998), filtro de Kalman (Huang et al., 2012), métodos de estimación paramétrica (Lopez-Estrada et al., 2014), entre otros.

2.3.2. Métodos Cualitativos basados en modelos y estrategias de búsqueda

Estos métodos están basados en la estrategia de razonamiento de causa-efecto sobre el comportamiento del sistema. Los métodos más populares en esta categoría son los arboles de fallas y la teoría de grafos. Sin embargo, una limitante de estos métodos es la generación de un gran número de hipótesis de posibles fallas, haciendo más difícil la toma de decisiones (Puig et al., 2010).

2.3.3. Métodos basados en el historial del proceso

Estos métodos requieren de una gran cantidad de datos del sistema a revisar. De igual forma se pueden subdividir en cualitativos y cuantitativos.

De los métodos cualitativos de esta categoría destacan: Sistemas expertos y Análisis de tendencias (Qualitative Trend Analysis). Los primeros consisten en una serie de antecedentes (una serie de eventos) y una parte de consecuencias, la cual mapea dichos eventos a una falla conocida (Maurya et al., 2005). La información del proceso entra al sistema en forma de estos antecedentes y consecuencias. Esto implica un mapeo explicito de síntomas conocidos a causas originarias de la falla. El QTA utiliza la información presente en las mediciones de los sensores. Existen dos pasos básicos: identificación de las tendencias en las mediciones e interpretación de tendencias en términos de los escenarios de fallas (Maurya et al., 2007).

Por otro lado, los métodos cuantitativos de esta categoría, son métodos estadísticos (PCA/PLS). El Análisis de las Componentes Principales (PCA), es una herramienta estadística aplicable a sistemas multivariantes, que permite la transformación de los datos multivariantes a un espacio de menor dimensión el cual retiene la información más relevante acerca del proceso (Ding et al., 2010). Esta compresión de información favorece el uso de esquemas de monitoreo de procesos multivariantes mediante técnicas aplicadas a procesos univariados como gráficas de Control Estadístico de Procesos (SPC). Así, una muestra actual del proceso se compara con las condiciones de operación normal resumidas en la gráfica de SPC, para detectar fallas en sensores y actuadores así como del proceso.

2.3.4. Métodos basados en datos

Existen varios métodos que utilizan datos de mediciones de un sistema de tuberías, estos parámetros son implementados en técnicas que están basadas en el comportamiento del sistema de acuerdo al tiempo de análisis. Se ha empleado la estadística como metodología de estudio, ya que se ha experimentado con los parámetros de algunas redes que ya llevan varios años de servicio, por lo que existe un comportamiento promedio de esta red hidráulica. Por algún lapso de tiempo se ha comportado de una forma estable y en otro momento ha sido lo contrario. Estos datos son importantes cuando se plantea estudiar un método mediante redes neuronales y análisis de datos, obteniendo una aproximación en la localización de una fuga en un ducto Schmidhuber (2015).

Análisis de datos

Se basa en el procesamiento de grandes volúmenes de datos históricos, pero desde otro punto de vista en la que no se busca explorar un esquema del proceso, sino descifrar un problema de clasificación. Para este propósito se ha propuesto el uso de máquinas con vectores de soporte, Modelo probabilístico, entre otras (Russell et al., 2012). El principal inconveniente de estas técnicas radica en el costo computacional y en que generalmente operan como un sistema de çaja negra", incapaz de brindar información adicional sobre la falla.

Redes neuronales artificiales

Una Red Neuronal Artificial (RNA) es un procesador distribuido y con un gran número de elementos simples interconectados masivamente en una estructura paralela, con una tendencia natural a almacenar y procesar conocimiento experimental, haciéndolo apto para su uso y de esta forma emular ciertas características propias de los humanos. Las redes neuronales tienen similitud al cerebro en dos cosas: el conocimiento de la red se adquiere a través de un proceso de aprendizaje y este conocimiento se almacena en los pesos sinápticos o conexiones entre neuronas (Yao, 1999). Las RNA tienen la capacidad de procesar información debido a su estructura paralela y distribuida, y su capacidad de entrenamiento, por tanto, de generalización. Estas dos propiedades hacen que las RNA, tengan la capacidad de resolver cierto tipo de problemas muy complejos que hasta el momento no habían podido resolverse de forma satisfactoria (Hassoun, 1995). En Algunas Características principales de una RNA:

1. Aprendizaje adaptativo: Las RNA tienen la cualidad de aprender a realizar tareas a partir de un entrenamiento, ya que cuenta con la capacidad de auto-ajuste de los elementos de procesado (neuronas) que integran el sistema, ajustándose de manera que la red reproduzca los resultados buscados. Para resolver el problema no hace falta que se disponga de información que describa los procesos internos que se dan para llegar a la respuesta del sistema, pues la red auto-organiza su distribución interna de pesos en las conexiones mediante la información recibida del entrenamiento.

    - Neuronas ocultas: son las encargadas del procesamiento de la información, ya que

estas reciben estímulos y emiten salidas dentro del sistema o proceso; no tienen contacto con el exterior

2. Tolerancia a fallos: Las RNA tienen la capacidad de aprender a identificar patrones de señales con ruido, distorsionados o incompletos siempre que haya suficiente información correcta para ser capaz de distinguir errores.

3. Fácil inserción dentro de la tecnología actual: El alto grado de conectividad de las RNA y su rapidez de ejecución las hace ideales para su implementación en distintos sistemas sin mayor dificultad e incluso la implementación en hardware es sencilla.

4. Operación en tiempo real: El entrenamiento de las RNA pueda que sea muy costoso en tiempo, sin embargo, una vez entrenadas, su respuesta ante las nuevas variables de entrada es muy rápida.

Este trabajo está dedicado al desarrollo de una metodología basada RNA para la detección y localización de fugas en redes de distribución hidráulica. La técnica se detalla en la siguiente sección.

2.4. Redes neuronales artificiales para detección de fugas

Las Redes Neuronales Artificiales son sistemas de procesamiento de información cuya estructura y funcionamiento están inspirados en las redes neuronales biológicas. Funcionando como un modelo computacional que se compone de un gran número de unidades básicas de cálculo llamadas neuronas artificiales que están conectadas entre sí en una red de comunicación que permite realizar cálculos de gran complejidad, la arquitectura típica de una red neuronal se presenta en la Figura 2.3, donde contiene una capa de entrada, con un numero de entradas n, siguiendo con una o más capas ocultas y una capa de salida, tanto el numero de neuronas, como en numero de capas son determinadas de acuerdo al problema a resolver. Las RNAs han sido ampliamente utilizados en actividades como clasificación (Darken and Moody, 1992), reconocimiento de patrones, reconocimiento de imágenes (Lowe and Webb, 1990), entre otros (Isasi Viñuela and Galván León, 2004).

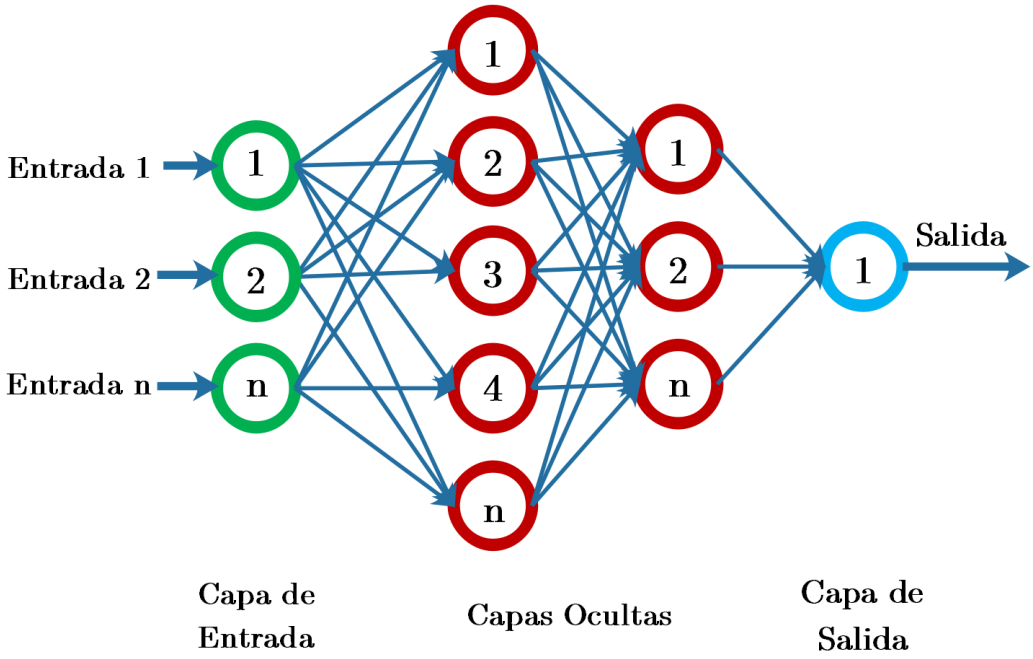


Figura 2.3: Arquitectura típica de una red Neuronal

Recientemente han tenido un gran auge en la detección y localización de fallas en diversos sistemas, basados en el procesamiento de grandes cantidades de datos que por métodos convencionales no se podrían procesar (Hirose, 2012), (Samarasinghe, 2016). En este capítulo se aborda el modelo más básico de una red neuronal, la neurona artificial. Posteriormente, se describe el perceptrón multicapa y su algoritmo entrenamiento. Finalmente, se describen los modelos de redes neuronales que son empleados en este proyecto.

2.4.1. Neurona artificial

Una neurona artificial es una unidad de computo que toma un vector de entradas $\mathbf{x} \in \mathbb{R}^n$ asociado a un vector de pesos $\mathbf{w} \in \mathbb{R}^n$ y un pesos extra $b \in \mathbb{R}$ conocido como sesgo. La salida o activación h de la neurona está definida por la Ecuación 2.1:

$$h(\mathbf{x}; \mathbf{w}, b) = f(\sum_{i=1}^{n} x_i w_i + b) \tag{2.1}$$

donde $f : \mathbb{R} \to \mathbb{R}$ es la función de activación.

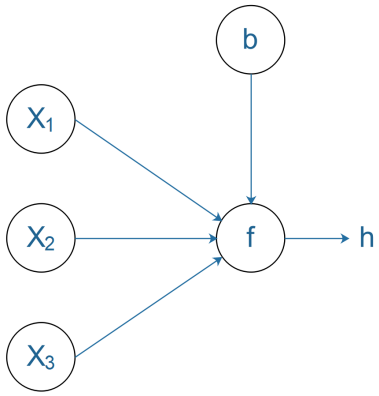


Figura 2.4: Neurona artificial de tres entradas x_1, x_2, x_3 ilustradas al inicio. El sesgo b se considera el peso de una entrada adicional con valor constante 1. Los pesos $w1, w2, w3$ son las conexiones de las entradas y la función de activación f.

El primer arquetipo de una neurona artificial fue presentado por McCulloch and Pitts (1943). La Figura 2.4 muestra el diagrama de una neurona artificial. Es preciso definir la salida de la neurona en términos de vectores como en la Ecuación 2.2:

$$h(\mathbf{x};\mathbf{w},b) = f(\mathbf{w}^T\mathbf{x} + b) \tag{2.2}$$

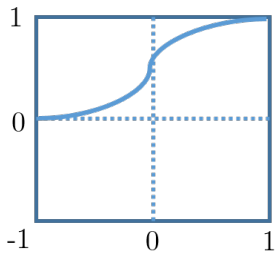
2.4.2. Funciones de activación

La función de activación es la que calcula la suma ponderada de la entrada, agrega un sesgo y luego decide si debe dispararse o no (Basheer and Hajmeer, 2000). El propósito principal de la función de activación es introducir la no linealidad en la salida de una neurona, las más comunes son las siguientes:

- Función sigmoide.

  Esta es una función suave y es continuamente diferenciable. Esta es una característica de gran funcionalidad de la función sigmoide. Básicamente, esto significa que cuando existen varias neuronas que tienen una función sigmoide como su función de activación, la salida tampoco es lineal. La función varía de 0 a 1 con forma de S.

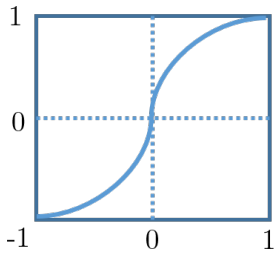
$$f(x) = \frac{1}{e^x + e^x}$$



Función sigmoide

- Función tangente hiperbólica.

$$f(x) = \frac{e^x - e^x}{e^x + e^x}$$

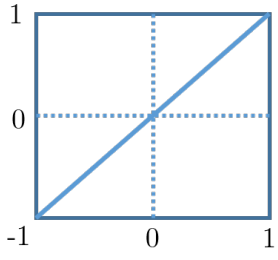


Función tangente hiperbólica

Esta función de activación puede verse como la función sigmoide escalada en el intervalo (-1; 1). Básicamente resuelve el problema de que los valores son todos del mismo signo. Todas las demás propiedades son las mismas que las de la función sigmoide. Es continuo y diferenciable en todos los puntos. La función no es lineal, por lo que se replantea fácilmente los errores.

- Función Lineal.

$$f(x) = x$$

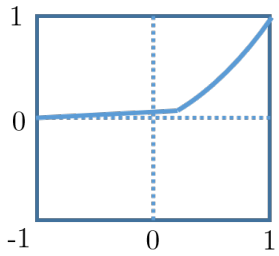


Función lineal

Una función de línea recta donde la activación es proporcional a la entrada (que es la suma ponderada de la neurona).

- Función exponencial normalizada o softmax.

$$f_j(\mathbf{x}) = \frac{e^{x_j}}{\sum_{i=1}^{n} e^{x_i}}, \text{ para } j = 1 \ldots n$$

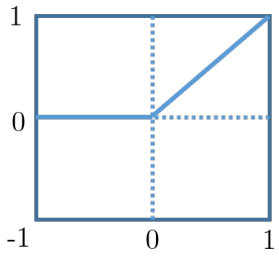


Función softmax

La función softmax también es un tipo de función sigmoidea, pero es útil cuando se esta tratando de manejar problemas de clasificación. La función sigmoide como fue capaz de manejar solo dos clases. Cuando existen más de dos clases, la simple clasificación de sí o no para una sola clase no ayudaría entonces. La función softmax comprimiría las salidas para cada clase entre 0 y 1 y también se dividiría por la suma de las salidas. Esto esencialmente da la probabilidad de que la entrada esté en una clase particular. La función softmax se usa idealmente en la capa de salida del clasificador, donde se el objetivo es tratar de alcanzar las probabilidades para definir la clase de cada entrada.

- Función rectificada lineal (ReLU)

$$f(z) = ln(1 + e^x)$$



Función ReLU

ReLU es la función de activación más utilizada al diseñar redes en la actualidad. La función ReLU no es lineal, lo que significa que podemos fácilmente propagar los errores

y tener múltiples capas de neuronas activadas por la función ReLU. La principal ventaja de usar la función ReLU sobre otras funciones de activación es que no activa todas las neuronas al mismo tiempo, esto significa que si la entrada es negativa, la convertirá a cero y la neurona no se activará. Por conlleva a que, en un momento dado, solo unas pocas neuronas se activan, lo que hace que la red sea escasa, haciéndola eficiente y fácil de calcular.

2.4.3. Perceptrón multicapa

El Perceptrón multicapa es una generalización del perceptrón simple y surgió como consecuencia de las limitaciones de dicha arquitectura en lo referente al problema de la separabilidad no lineal. Minsky and Papert (1969) mostraron que que la combinación de varios Perceptrónes simples inclusión de neuronas ocultas podía resultar una solución adecuada para tratar ciertos problemas no lineales. Sin embargo, los autores no presentaron una solución al problema de como adaptar los pesos de la capa de entrada a la capa oculta, pues la regla de aprendizaje del Perceptrón simple no puede aplicarse en este escenario. Sin embargo, la idea de combinar varios Perceptrónes sirvió de base para estudios posteriores realizados por Rumelhart et al. (1986), donde presentaron una manera de retropropagación de los errores medidos en la salida de la red hacia las neuronas ocultas, dando lugar a la llamada regla delta generalizada.

Como se muestra en la Figura 2.5 las conexiones entre neuronas se dan entre capas de izquierda a derecha. Por este motivo estas redes también toman el nombre de redes alimentadas hacia adelante. La capa más a la izquierda es la capa de entrada, la capa más a la derecha es la capa de salida y las capas intermedias son las capas ocultas. El numero de capas en una red esta dado por n_l y la capa numero l se denota por L_l.

Los parámetros de la red están dados por $(\mathbf{W},\mathbf{b}) = ((\mathbf{W}^{(1)},\mathbf{b}^{(1)}),(\mathbf{W}^{(2)},\mathbf{b}^{(2)}))$, donde $w^l_{ij} \in \mathbf{W^{(l)}}$ es el peso asociado a la conexión entre la neurona j en la capa l y la neurona i en la capa $l+1$; y b^l_i es el sesgo asociado a la neurona i en la capa $l+1$.Además,s_l denota el numero de neuronas en la capa l.

Para explicar el proceso de computo del perceptrón multicapa consideremos el ejemplo

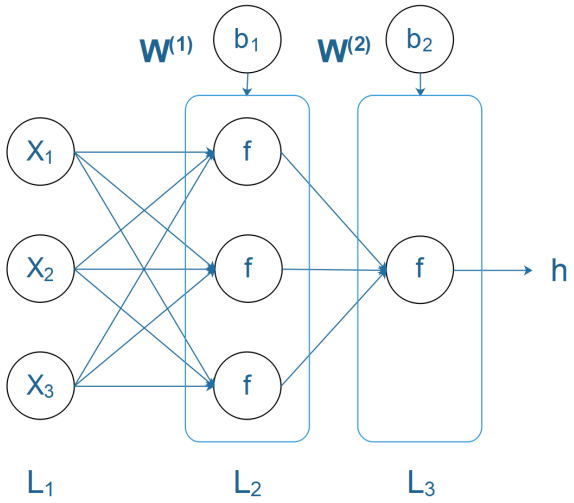


Figura 2.5: Perceptrón multicapa con una capa de entrada L_1, una capa oculta L_2 y una capa de salida L_3 (con una sola neurona). Los pesos $\mathbf{W}^{(1)}$ conectan las capasL_1 y L_2; mientras que los pesos $\mathbf{W}^{(2)}$ conectan las capas L_2 y L_3.

presentado en la Figura 2.5. La activación o salida de la neurona i en la capa l se denota a_i^l. Para la capa de entrada $l = 1$ se tiene que $a_i^l = x_i$. El calculo $h(x)$ para la red esta dado por las siguientes expresiones:

$$a_1^{(2)} = f(w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3 + b_1^{(1)}), \tag{2.3}$$

$$a_2^{(2)} = f(w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3 + b_2^{(1)}), \tag{2.4}$$

$$a_3^{(2)} = f(w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3 + b_3^{(1)}), \tag{2.5}$$

$$h(\mathbf{x}) = a_1^{(2)} = f(w_{11}^{(2)} a_1^{(2)} + w_{12}^{(2)} a_2^{(2)} + w_{13}^{(2)} a_3^{(2)} + b_1^{(1)}). \tag{2.6}$$

En lo siguiente se denotará z_i^l como la suma pesada de las entradas a la neurona i de la capa l incluyendo el sesgo, de modo que $a_i^l = f(z_i^l)$. Esto permite utilizar una notación más compacta si se extiende la definición de la función $f(\cdot)$ para aplicarla a vectores por elemento. Entonces

se pueden reescribir las ecuaciones anteriores como:

$$\mathbf{z}^{(2)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}, \tag{2.7}$$

$$\mathbf{a}^{(2)} = f(\mathbf{z}^{(2)}), \tag{2.8}$$

$$\mathbf{z}^{(3)} = \mathbf{W}^{(2)}\mathbf{x}^{(2)} + \mathbf{b}^{(2)}, \tag{2.9}$$

$$h(\mathbf{x}) = a^{(3)} = f(\mathbf{a}^{(3)}), \tag{2.10}$$

$$\tag{2.11}$$

En general, si se tiene en cuenta que $\mathbf{a}^{l} = \mathbf{x}$, la activación $\mathbf{a}^{(l+1)}$ para la capa $l+1$ se computa como:

$$\mathbf{z}^{(l+1)} = \mathbf{W}^{(1)}\mathbf{a}^{(1)} + \mathbf{b}^{(1)}, \tag{2.12}$$

$$\mathbf{a}^{(l+1)} = f(\mathbf{z}^{(l+1)}). \tag{2.13}$$

La Figura 2.5 muestra un ejemplo de red neuronal muy básico. No obstante, se pueden construir redes neuronales con diferentes patrones de conectividad entre neuronas, es decir, diferentes arquitecturas. Por ejemplo, una arquitectura más compleja puede ser formada con múltiples capas ocultas donde se tiene una capa de entrada L_1, una capa de salida L_{nl} y cada capa oculta l está completamente conectada a la capa $l+1$. En esta configuración, el computo de la salida de la red se realiza calculando todas las activaciones de la capa L_2, después para la capa L_3 y Así sucesivamente hasta la capa L_{nl} usando las Ecuaciones 2.12 y 2.13. Este es un ejemplo de una red neurona alimentada hacia adelante, ya que el gráfico de conectividad no tiene ciclos o bucles dirigidos. Además, las redes neuronales pueden tener múltiples neuronas de salida. La Figura 2.6 muestra una red neuronal con dos neuronas de salida y dos capas ocultas.

En este caso, se sobrecarga la definición de la función de salida de forma que $h(\mathbf{x}) \in \mathbb{R}^2$. En general, se considerara que la función de salida está sobrecargada, por lo que produce un vector correspondiente al numero de salidas de la red neuronal.

Dentro del marco de las redes de neuronas, el Perceptrón multicapa es en la actuali-

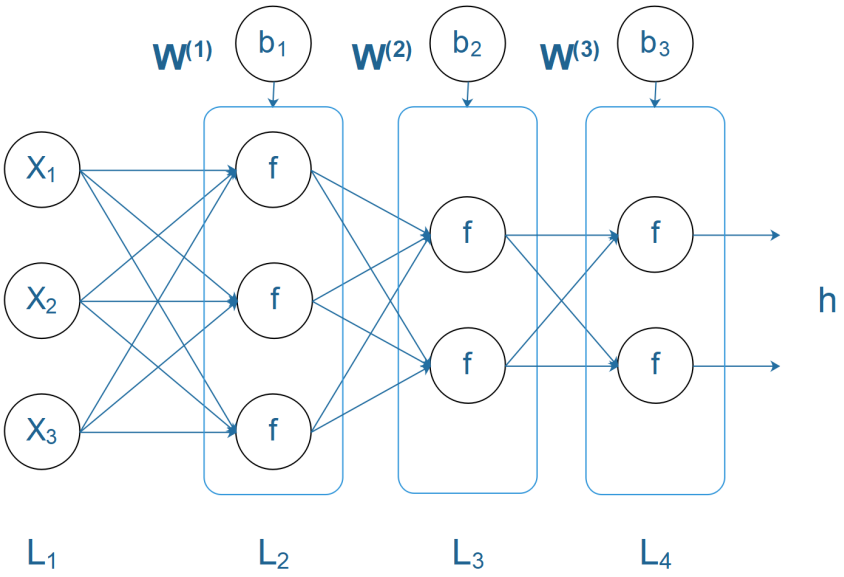


Figura 2.6: Perceptrón multicapa con una capa de entrada L_1; dos capas ocultas L_2 y L_3; y una capa de salida L_4 con dos neuronas.

dad una de las arquitecturas más utilizadas en la resolución de problemas. Esto es debido, fundamentalmente, a su capacidad como aproximador universal, así como a su fácil uso y aplicabilidad.

Por otra parte, esto no implica que sea una de las redes más potentes y con mejores resultados en sus diferentes áreas de aplicación. De hecho, el Perceptrón multicapa posee una serie de limitaciones, como el largo proceso de aprendizaje para problemas complejos dependientes de un gran número de variables; la dificultad para realizar un análisis teórico de la red debido a la presencia de componentes no lineales y a la alta conectividad.

2.4.4. Algoritmo de retropropagación

La idea general del algoritmo de retropropagación es minimizar una función de distancia o error E. La regla o algoritmo de aprendizaje es el mecanismo mediante el cual se van adaptando y modificando todos los parámetros de la red. Entre las etiquetas reales del conjunto de datos de entrenamiento $\mathbf{y}$ las etiquetas predichas $h(\mathbf{x})$ por la red para este conjunto. Consideremos un conjunto de entrenamiento $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}$ con m ejemplos. Una de las funciones de error más comunes es el error cuadrático medio mostrado en la Ecuación 2.14:

$$E(\mathbf{W},\mathbf{b}) = \frac{1}{2m}\sum_{i=1}^{m} \| \, h(\mathbf{x}^i;\mathbf{W},\mathbf{b}) - \mathbf{y}^{(i)} \, \|^2 . \tag{2.14}$$

Otra función ampliamente utilizada es la entropía cruzada mostrada en la Ecuación 2.15:

$$E(\mathbf{W},\mathbf{b}) = -\frac{1}{m}\sum_{i=1}^{m} \mathbf{y}^{(i)} \log(h(\mathbf{x}^{(i)};\mathbf{W},\mathbf{b})). \tag{2.15}$$

La explicación del algoritmo retropropagacioon se realizara utilizando el error cuadrático medio, no obstante, no es difícil extenderla a entropía cruzada. El objetivo es minimizar E como una función de $\mathbf{W}$ y $\mathbf{b}$. El proceso de entrenamiento comienza inicializando cada peso $w_{ij}^{(l)}$ y $b_i^{(l)}$ con valores pequeños tomados de una distribución normal $mathcalN(0,\epsilon)$. Para minimizar la función de error se emplea el método del gradiente descendiente, por lo que un paso de actualización de los pesos $\mathbf{W}$ y $\mathbf{b}$ se da de la forma siguiente:

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} E(\mathbf{W},\mathbf{b}), \tag{2.16}$$

$$b_i^{(l)} := b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} E(\mathbf{W},\mathbf{b}). \tag{2.17}$$

Donde α es la tasa de aprendizaje y := es asignación de variables. El paso clave es calcular estas derivadas parciales. Ahora se describirá la forma en que el algoritmo de retropropagacion permite calcular estas derivadas de forma eficiente.

La intuición detrás del algoritmo es la siguiente. Dado un ejemplo de entrenamiento $(\mathbf{x};\mathbf{y})$, durante la primera etapa (hacia adelante) se calculan todas las activaciones de la red, incluyendo las salidas $h(\mathbf{x};\mathbf{W};\mathbf{b})$. La segunda etapa (hacia atrás) consiste en calcular para cada nodo i en la capa l, la fracción del error $\delta_i^{(l)}$ que determina la contribución del nodo al error total de la red. Para un nodo de salida, podemos calcular directamente la diferencia entre la activación de la red y el valor objetivo, el cual se denota por $\delta_i^{(n_l)}$. Más detalladamente el algoritmo puede ser descrito de la siguiente manera:

1. Realizar la propagación hacia adelante de toda la red. Se calculan las activaciones de cada capa con las Ecuaciones 2.12 y 2.13.

2. Por cada nodo i en la capa de salida se calcula:

$$\delta_i^{(n_l)} = \frac{\partial}{\partial_{z_i}^{(n_l)}} \frac{1}{2} \parallel \mathbf{y} - h(\mathbf{x}) \parallel^2 = -(y_i - a_i^{(nl)}) f'(z_i^{(nl)}). \tag{2.18}$$

3. Por cada capa oculta se calcula:

$$\delta_i^l = (\sum_{j=1}^{sl+1} w_{ji}^{(l)} \delta_j^{(l+1)}) f'(z_i^{(l)}). \tag{2.19}$$

4. Se actualizan los pesos de la red de acuerdo con las Ecuaciones 2.16 y 2.17.

En el caso del perceptrón multicapa se trata de un algoritmo de aprendizaje supervisado, es decir, la modificación de los parámetros se realiza para que la salida de la red sea lo más próxima posible a la salida proporcionada por el supervisor o salida deseada. En la Figura se observa una red neuronal con el algoritmo de retropropagación, que se presentó en esta sección.

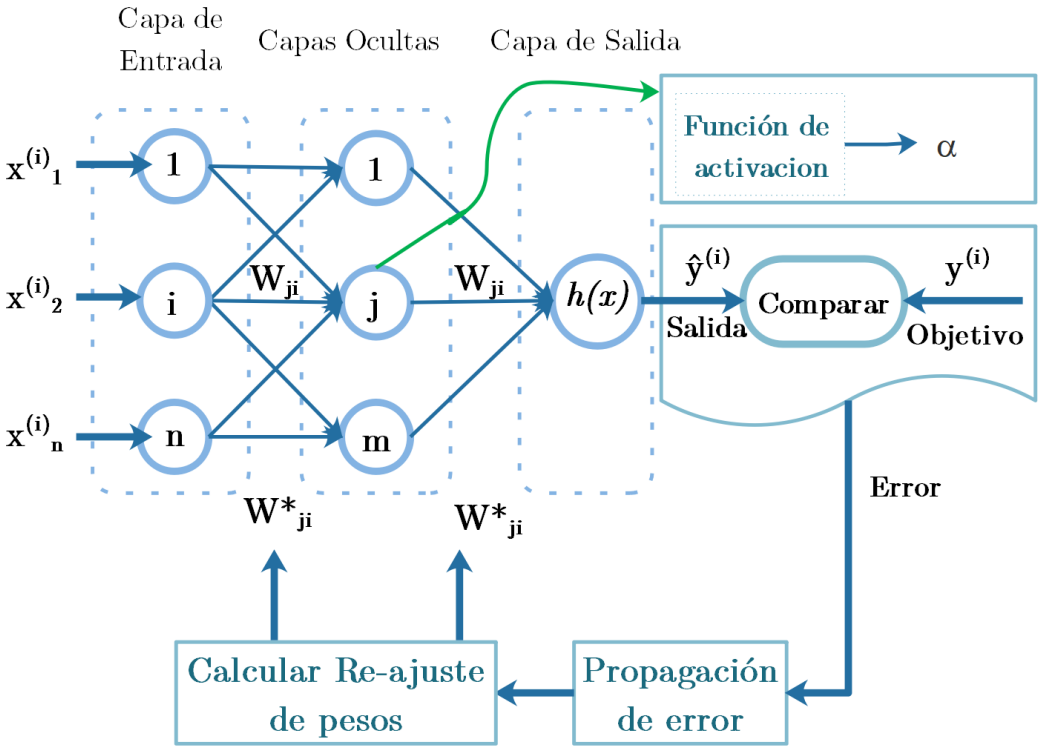


Figura 2.7: Algoritmo de retropropagación aplicado a una red neuronal.

Por tanto, para cada patrón de entrada a la red es necesario disponer de un patrón de salida deseada. Puesto que el objetivo es que la salida de la red sea lo más próxima posible a la salida deseada, el aprendizaje de la red se formula como un problema de minimización.

2.4.5. Entrenamiento y generalización

La red neuronal multicapa hacia adelante (FFN por sus siglas en ingles) funciona en dos modos: entrenamiento y modo de predicción. Para el entrenamiento de la red neuronal de FFN y para la predicción utilizando la red neuronal de FFN se necesitan dos conjuntos de datos, el conjunto de entrenamiento y el conjunto que queremos predecir (salidas deseadas).

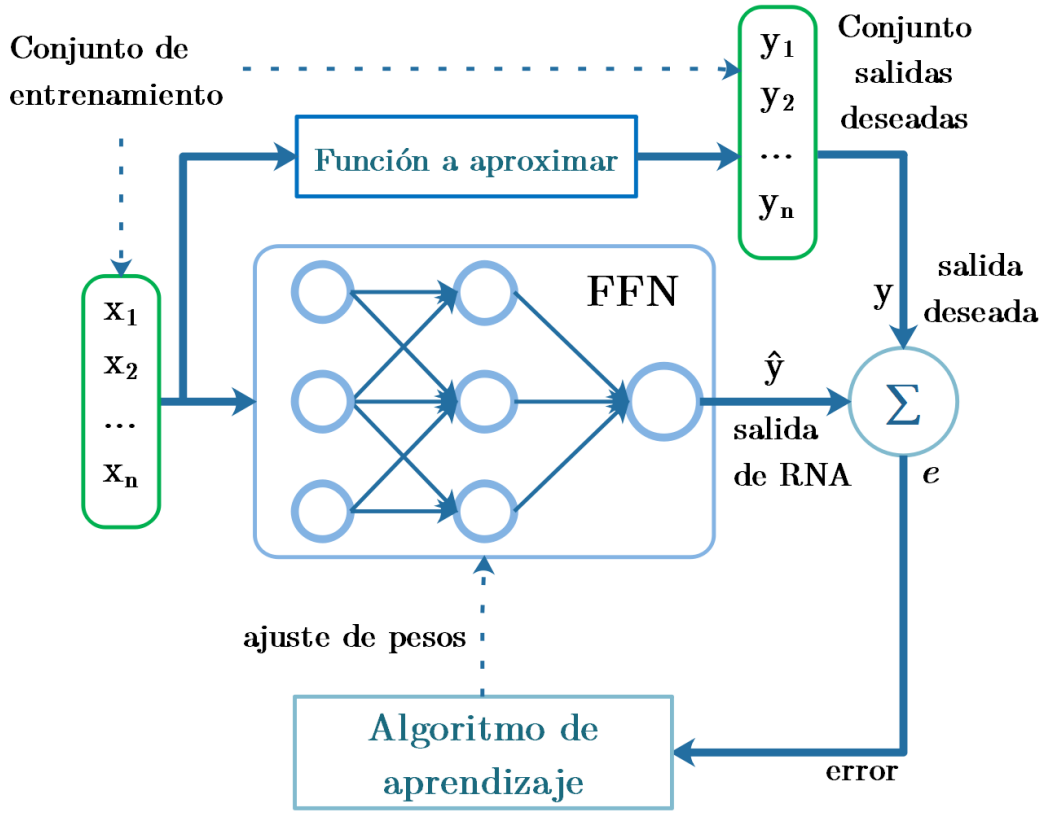


Figura 2.8: Diagrama representativo del entrenamiento supervisado de una red neuronal

En la Figura 2.8 se observa que el entrenamiento comienza con valores arbitrarios de los pesos que pueden ser números aleatorios, y continúa de forma iterativa. Cada iteración del conjunto de entrenamiento completo se denomina época. En cada época, la red ajusta

los pesos en la dirección que reduce el error (consulte el algoritmo de retropropagación). A medida que continúa el proceso iterativo de ajuste incremental, las ponderaciones convergen gradualmente al conjunto de valores óptimo localmente (Figura 2.9). Muchas épocas son generalmente requeridas antes de que se complete el entrenamiento.

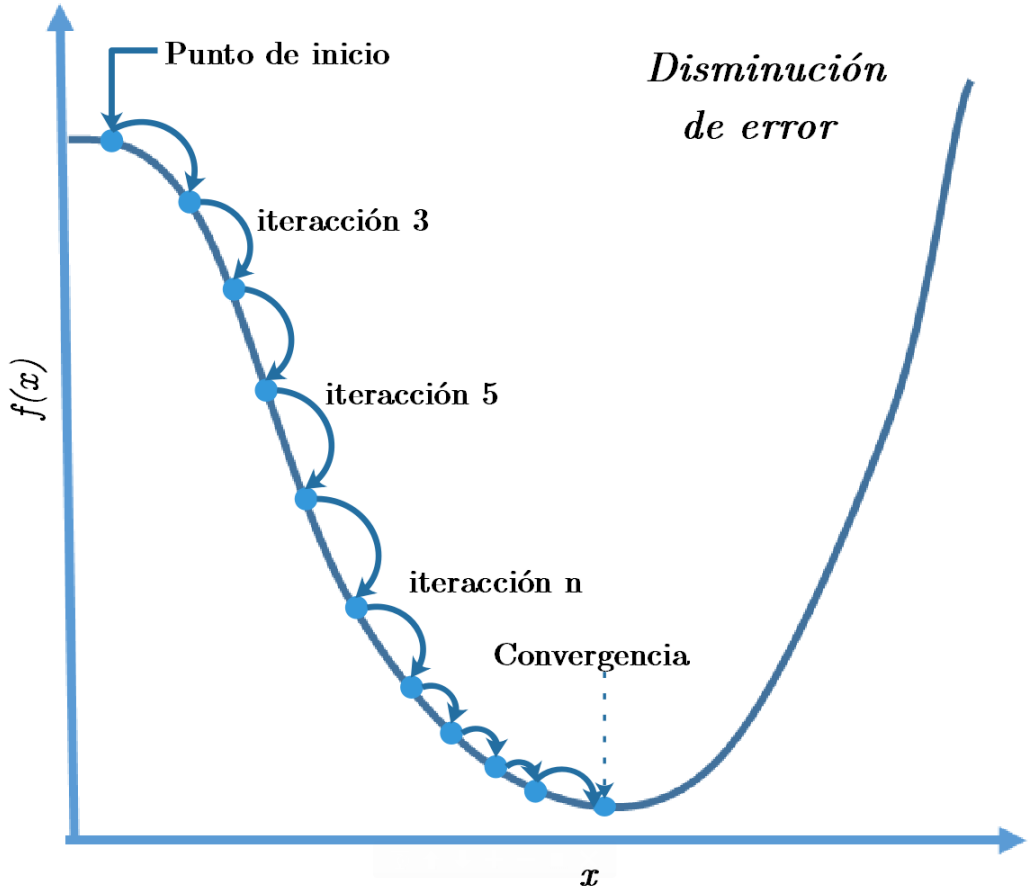


Figura 2.9: Ajuste del error por iteraciones hasta la convergencia a un valor final

Para un conjunto de entrenamiento dado, el aprendizaje de propagación hacia atrás puede proceder de una de dos formas básicas: modo de patrón y modo de proceso por lotes. En el modo de patrón de aprendizaje de propagación hacia atrás, la actualización de peso se realiza después de la presentación de cada patrón de entrenamiento. En el modo por lotes de aprendizaje de propagación hacia atrás, la actualización del peso se realiza después de la presentación de todos los ejemplos de entrenamiento (es decir, después de toda la época). Desde un punto de vista en linea , el modo de patrón se prefiere al modo por lotes, ya que requiere menos almacenamiento local para cada conexión sináptica (Hertz et al., 1991).

Además, dado que los patrones se presentan a la red de manera aleatoria, el uso de la actualización de ponderaciones patrón por patrón hace que la búsqueda en el espacio de peso sea estocástica, lo que hace que sea menos probable que el algoritmo de propagación inversa quede atrapado en un mínimo local. Por otro lado, el uso del modo de entrenamiento por lotes proporciona una estimación más precisa del vector gradiente. El modo de patrón es necesario para usar, por ejemplo, en el control de procesos en línea, porque no hay todos los patrones de entrenamiento disponibles en el tiempo dado. En el análisis final, la efectividad relativa de los dos modos de entrenamiento depende del problema resuelto (Bertsekas and Tsitsiklis, 1995).

En el modo de predicción, la información fluye hacia adelante a través de la red, desde las entradas hasta las salidas. La red procesa un ejemplo a la vez, produciendo una estimación de los valores de salida basados en los valores de entrada. El error resultante se utiliza como una estimación de la calidad de la predicción de la red entrenada.

En el aprendizaje de la propagación hacia atrás, generalmente comienza con un conjunto de entrenamiento y se usa el algoritmo de propagación hacia atrás para calcular los pesos sinápticos de la red. La esperanza es que la red neuronal así diseñada se generalice. Se dice que una red generaliza bien cuando la relación de entrada-salida calculada por la red es correcta (o casi correcta) para los patrones de entrada/salida que no se utilizaron en el entrenamiento de la red. La generalización no es una propiedad mística de las redes neuronales, pero puede compararse con el efecto de una buena interpolación no lineal de los datos de entrada (Wieland and Leighton, 1987).

El principio de generalización se muestra en la Figura 2.10a. Cuando el proceso de aprendizaje se repite demasiadas iteraciones (es decir, la red neuronal está sobreentrenada o sobreajustada, entre sobreentrenada y sobreajustada no hay diferencia), la red puede memorizar los datos de entrenamiento y, por lo tanto, ser menos capaz de generalizar entre patrones similares de entrada-salida. La red proporciona resultados casi perfectos para ejemplos del conjunto de entrenamiento, pero falla para ejemplos del conjunto de pruebas. El sobreajuste se puede comparar con la elección incorrecta del grado de polinomio de regresión (Figura 2.10b). Se pueden producir sobrecargas severas con datos ruidosos, incluso cuando hay muchos más

casos de entrenamiento que pesos.

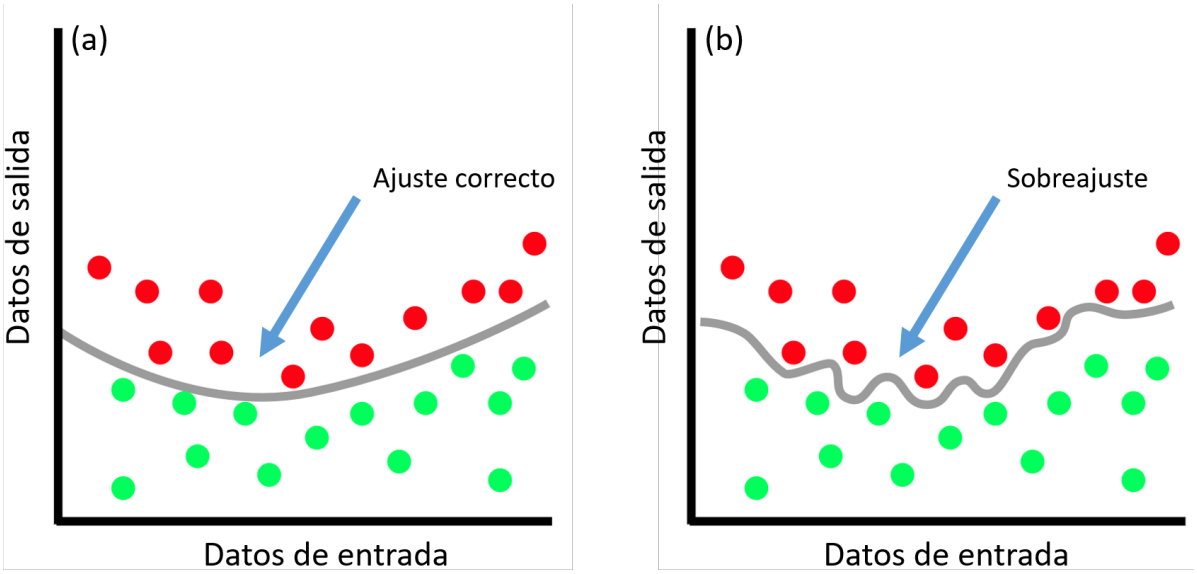


Figura 2.10: Principio de generalización y sobreajuste. (a) Buen ajuste Datos (buena generalización). (b) Datos sobreajustados (generalización deficiente).

La condición básica para una buena generalización es un conjunto suficientemente grande de los casos de capacitación. Este conjunto de entrenamiento debe estar en el mismo subconjunto representativo del conjunto de todos los casos a los que desea generalizar. La importancia de esta condición está relacionada con el hecho de que hay dos tipos diferentes de generalización: interpolación y extrapolación. La interpolación se aplica a los casos que están más o menos rodeados de casos de capacitación cercanos; Todo lo demás es extrapolación. En particular, los casos que están fuera del rango de los datos de entrenamiento requieren extrapolación. La interpolación a menudo se puede hacer de manera confiable, pero la extrapolación es notoriamente poco confiable. Por lo tanto, es importante tener suficientes datos de entrenamiento para evitar la necesidad de extrapolación. Los métodos para seleccionar buenos conjuntos de entrenamiento surgen del diseño experimental (Wu et al., 1996). Para una discusión elemental de sobrealimentación, ver (Smith, 1993).Además, En el articulo de Geman et al. (1992), muestra un enfoque detallado. Dada una cantidad fija de datos de entrenamiento, existen algunos enfoques efectivos para evitar el sobreajuste y, por lo tanto, obtener una buena generalización.

2.4.6. Elección del modelo de RNA

La pregunta crucial en la selección del modelo es "¿Cuántas unidades ocultas debo usar?". Algunos libros y artículos ofrecen reglas de oro"para elegir una topología, por ejemplo, el tamaño de la capa oculta debe estar en algún lugar entre el tamaño de la capa de entrada y el tamaño de la capa de salida (Blum, 1992), o algunas otras reglas, pero esas reglas son totales disparates. No hay forma de determinar una buena topología de red solo a partir del número de entradas y salidas. Depende críticamente de la cantidad de casos de entrenamiento, la cantidad de ruido y la complejidad de la función o clasificación que está tratando de aprender (Hornik, 1991). Una elección inteligente de la cantidad de unidades ocultas depende de si está utilizando una parada temprana o alguna otra forma de regularización. De lo contrario, simplemente debe probar muchas redes con diferentes números de unidades ocultas, estimar el error de generalización para cada una y elegir la red con el error de generalización mínimo estimado.

Otro problema en la selección del modelo es la cantidad de capas ocultas que usa. En la red neuronal de alimentación directa de múltiples capas con cualquiera de las funciones de activación de la capa oculta no lineal continua, una capa oculta con un número arbitrariamente grande de unidades es suficiente para la propiedad de .ªproximación universal"(Hornik, 1993),(Bishop et al., 1995). De todos modos, no hay ninguna razón teórica para usar más de dos capas ocultas. En (Kurková, 1992) se le dio una prueba constructiva sobre los límites (grandes, pero los límites, no obstante) en el número de neuronas ocultas en redes neuronales de dos ocultos.

En la práctica, se requiere dos capas ocultas para el aprendizaje de la función, que en su mayoría es continua, pero tiene algunas discontinuidades (Masters, 1993). Desafortunadamente, el uso de dos capas ocultas exacerba el problema de los mínimos locales, y es importante usar muchas inicializaciones aleatorias u otros métodos para la optimización global. Otro problema es que la capa oculta adicional hace que el gradiente sea más inestable, es decir, que el proceso de entrenamiento se ralentiza drásticamente. Se recomienda encarecidamente usar una capa oculta y luego, si el uso de una gran cantidad de neuronas ocultas no resuelve el problema, puede valer la pena probar la segunda capa oculta.

2.4.7. Caída de peso

La caída de peso agrega un término de penalización a la función de error. La penalización habitual es la suma de pesos al cuadrado por una constante de desintegración. En un modelo lineal, esta forma de caída de peso es equivalente a la regresión de cresta. La caída de peso es un subconjunto de los métodos de regularización. El término de penalización en la reducción de peso, por definición, penaliza a los grandes pesos. Otros métodos de regularización pueden involucrar no solo los pesos, sino varios derivados de la función de salida (Bishop et al., 1995). El término de penalización de caída de peso hace que los pesos converjan a valores absolutos más pequeños de lo que lo harían de otra manera. Grandes pesos pueden dañar la generalización de dos maneras diferentes. Los pesos excesivamente grandes que conducen a unidades ocultas pueden hacer que la función de salida sea demasiado aproximada, posiblemente con casi discontinuidades. Excesivamente los grandes pesos que conducen a las unidades de salida pueden provocar salidas salvajes más allá del rango de los datos si la función de activación de salida no está limitada al mismo rango que los datos.

El principal riesgo con grandes ponderaciones es que las salidas de nodos no lineales podrían estar en una de las partes planas de la función de transferencia, donde la derivada es cero. En tal caso el aprendizaje es irreversible. El término de compensación permite la continuación del aprendizaje incluso con grandes pesos. Para decirlo de otra manera, los pesos grandes pueden causar una variación excesiva de la salida (Geman et al., 1992). Para una discusión de la caída de peso, ver por ejemplo (Ripley, 1996).

2.4.8. Parada temprana

La parada temprana es el método más utilizado para evitar el sobreajuste. El principio de la detención temprana es dividir los datos en dos conjuntos, entrenamiento y validación, y calcular el error de validación periódicamente durante el entrenamiento. El entrenamiento se detiene cuando la tasa de error de validación comienza a subir. Es importante darse cuenta que el error de validación no es una buena estimación del error de generalización. Un método para obtener una estimación del error de generalización es ejecutar la red en un tercer conjunto de

datos, el conjunto de prueba, que no se utiliza en absoluto durante el proceso de capacitación (Weiss and Kulikowski, 1991). La desventaja de la validación de muestras divididas es que reduce La cantidad de datos disponibles tanto para la formación como para la validación.

Otra posibilidad de obtener una estimación de la generalización es utilizar la llamada validación cruzada (Stone, 1974). La validación cruzada es una mejora en la validación de muestras divididas que le permite utilizar todos los datos para la capacitación. En la validación cruzada k-fold, divide los datos en k subconjuntos de igual tamaño. Entrenas las veces en red k, cada vez que dejas fuera a uno de los subconjuntos del entrenamiento, pero utilizando solo el subconjunto omitido para calcular cualquier criterio de error que te interese. Si k es igual al tamaño de la muestra, esto se denomina validación cruzada (Figura 2.11).

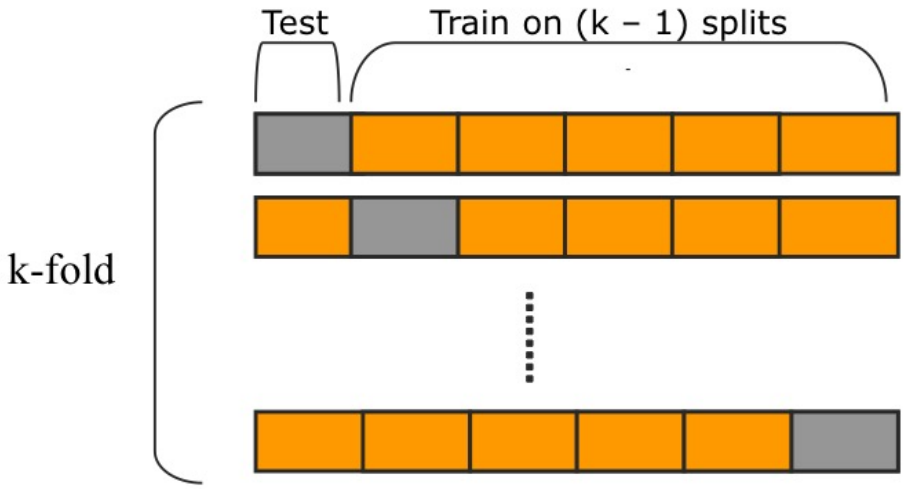


Figura 2.11: Validación cruzada k-fold.

Si bien varias autores han sugerido que la validación cruzada se aplique a la detención temprana, la forma correcta de hacerlo no es obvia. La desventaja de validación cruzada es que tienes que volver a entrenar la red muchas veces. Pero en el caso de las redes neuronales de FML, la variabilidad entre los resultados obtenidos en diferentes ensayos a menudo se debe al hecho de que el aprendizaje terminó en muchos mínimos locales diferentes. Por lo tanto, el método de validación cruzada es más adecuado para redes neuronales sin el peligro de caer en mínimos locales (por ejemplo, función de base radial, RBF (Lohninger, 1993)). Existe un método similar a la validación cruzada, el llamado bootstrapping (Hjorth, 1994), (Efron and Tibshirani, 1994). El bootstrapping parece funcionar mejor que la validación cruzada en

muchos casos.

La parada temprana tiene sus ventajas (es rápida, requiere solo una decisión importante por parte del usuario: qué proporción de casos de validación usar) pero también algunas desventajas (cuántos patrones se utilizan para la capacitación y para el conjunto de validación (Amari et al., 1997), cómo dividir los datos en entrenamiento y conjunto de pruebas, cómo saber ese error de validación realmente sube).

2.4.9. Red neuronal de retropropagación hacia adelante (FFBPN)

El perceptrón multicapa da indicios para realizar trabajos con una red (FFBPN), La red neuronal prealimentada fue previamente la más sencilla de tipo de red neuronal artificial ideada (Schmidhuber, 2015). En esta red, la información se mueve hacia adelante. De los nodos de entrada, a través de los nodos ocultos, hacia los nodos de salida. No hay ningún ciclo o bucle en estas redes. Utiliza el algoritmo de aprendizaje de retropropagación para aprender. Durante el entrenamiento de esta red, los cálculos se llevan a cabo desde la capa de entrada de la red hacia la capa de salida, y los valores de error se propagaron a las capas anteriores. Las redes de avance a menudo tienen una o más capas ocultas de neuronas sigmoideas seguidas de una capa de salida de neuronas lineales (Zell et al., 1994).

Este tipo de arquitectura de RNA se utilizó para realizar la estimación del factor de fricción, las múltiples capas de neuronas con funciones de transferencia no lineal permiten a la red aprender relaciones no lineales y lineales entre los vectores de entrada y salida. La capa de salida lineal permite que la red produzca valores fuera del rango –1 a +1. Por otro lado, las salidas de una red, como por ejemplo entre 0 y 1, se producen, entonces la capa de salida debe usar una función de activación sigmoidea (logsig).

El algoritmo siguiente es para obtener una red entrenada, el cual consiste en un método de cálculo del gradiente utilizado en algoritmos de aprendizaje supervisado, necesarios en el cálculo de los pesos que se utilizarán en la red.

Algoritmo 1 Entrenamiento de una RNA

Entrada: Conjunto de datos (CD), Algoritmo de entrenamiento, tipo de RNA.
Salida: RNA entrenada.

Inicialice todos los pesos y sesgos en la red;
while mientras que la condición de terminación no se cumple **hacer**
for para cada tupla de entrenamiento X en CD **hacer**
// Propagar la entrada hacia adelante:
for para cada unidad de capa de entrada j **hacer**
$O_j = I_j$; // la salida de una unidad de entrada es su valor de entrada actual
fin for
for para cada unidad de capa oculta o de salida j **hacer**
$I_j = \sum_i W_{ij} O_i + \theta_j$; // calcular la entrada de red de la capa j a la capa anterior
$O_j = \frac{1}{1+e^{-Ij}}$; // calcular la salida de cada unidad j
fin for
// Retropropagación de los errores:
for para cada unidad j en la capa de salida **hacer**
$Err_j = O_j(1-O_j)(T_j - O_j)$; // calcular el error
fin for
for para cada unidad j en la capa oculta, desde la última hasta la primera capa oculta **hacer**
$Err_j = O_j(1-O_j)\sum_k(Err_k)(W_{ij})$; // calcular el error en la siguiente capa, k
fin for
for para cada peso en nework **hacer**
$\Delta W_{ij} = (Err_j)(O_i)$; // incremento de peso
$W_{ij} = W_{ij} + \Delta W_{ij}$; // actualización de peso
fin for
for para cada sesgo θ_j en la red **hacer**
$\Delta\theta_j = Err_j$; // incremento de sesgo
$\theta_{j(new)} = \theta_{j(old)} + \Delta\theta_j$; // actualización de sesgo
fin for
fin for
fin while

2.4.10. Red neuronal de retropropagación hacia adelante en cascada (CFBPN)

La arquitectura propuesta de la RNA para estimar la posición de fuga es una Cascade-Forward Backpropagation Network (CFBPN, por sus siglas en ingles) (Lashkarbolooki et al., 2013a), esta red es similar a una red Feed-Forward (FFBPN) que consiste en capas de entrada, ocultas y una de salida. En una red de retroalimentación, la información siempre se mueve en una dirección; hacia adelante, nunca va hacia atrás. El algoritmo de aprendizaje de Backpropagation (retropropagación) se usó para entrenar estas redes (Pwasong and Sathasivam, 2016). Durante el entrenamiento, los cálculos se llevan a cabo desde la capa de entrada hacia la capa de salida, y los valores de error vuelven a la capa anterior. La arquitectura de red neuronal artificial más popular para la predicción de datos es la Backpropagation; esta red utiliza funciones continuamente valoradas y aprendizaje supervisado Jain et al. (1996), sin embargo, la diferencia principal entre estos modelos es que el modelo CFBNP incluye una conexión de peso desde la entrada a la siguiente capa y desde la siguiente capa a las capas sucesivas (Figura 2.12).

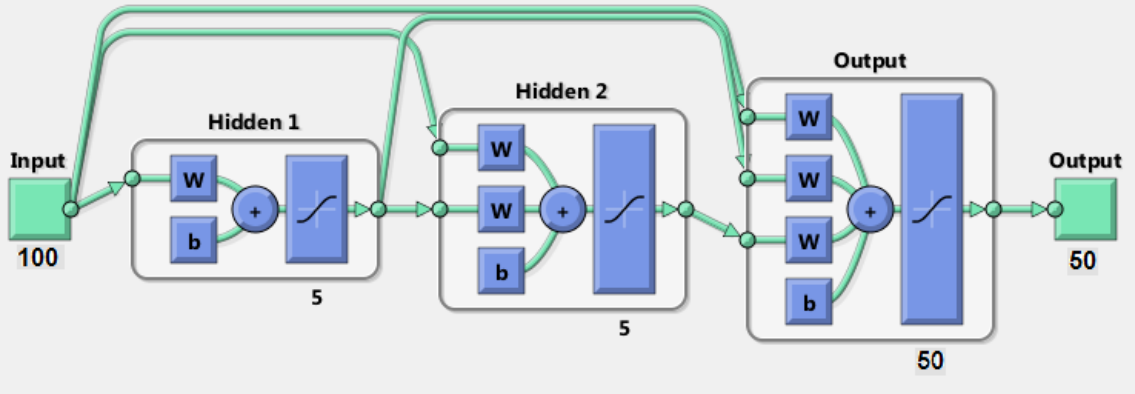


Figura 2.12: Arquitectura de la CFBPN.

Si bien las redes de avance de dos capas pueden aprender virtualmente cualquier relación de entrada-salida, las redes de feedforward con más capas pueden aprender relaciones complejas más rápidamente Lashkarbolooki et al. (2013b). Por ejemplo, una red de tres capas tiene conexiones desde la capa 1 a la capa 2, capa 2 a la capa 3 y capa 1 a la capa 3. La red de tres capas también tiene conexiones desde la entrada a las tres capas. Las conexiones adicionales pueden mejorar la velocidad a la que la red aprende la relación deseada. El modelo de inteligencia artificial de CFBPN es similar a la red neuronal de retropropagación hacia

adelante en el uso del algoritmo de retropropagación para la actualización de pesos, pero la diferencia principal de esta red es que cada capa de neuronas está relacionada con todas las capas de neuronas anteriores.

Las conexiones adicionales pueden mejorar la velocidad a la que la red aprende la correlación deseada. El FFBPN y el CFBPN son similares porque usan el algoritmo Backpropagation para actualizar los pesos, pero el indicativo principal de esta red es que cada capa de neuronas está relacionada con todas las capas anteriores de neuronas Hedayat et al. (2009). Una conexión multiplica su valor por un peso de conexión, como lo hace una conexión sináptica.

2.4.11. Ventajas y desventajas de las redes neuronales FFN

La aplicación de redes neuronales de las Redes Neuronales Prealimentadas Hacia adelante (FFN) ofrece las siguientes propiedades y capacidades útiles:

- Aprendizaje. Las FFN pueden adaptarse sin la ayuda del usuario.
- No linealidad. Una neurona es un dispositivo no lineal. En consecuencia, una red neuronal es en sí misma no lineal. La no linealidad es una propiedad muy importante, particularmente si la relación entre entrada y salida es inherentemente no lineal.
- Mapeo de entrada-salida. En la capacitación supervisada, cada ejemplo consta de una señal de entrada única y la respuesta deseada correspondiente. Un ejemplo recogido del conjunto de entrenamiento se presenta a la red, y los coeficientes de peso se modifican para minimizar la diferencia entre la salida deseada y la respuesta real de la red. La capacitación de la red se repite para muchos ejemplos en el conjunto de capacitación hasta que la red alcanza el estado estable. Por lo tanto, la red aprende de los ejemplos al construir un mapeo de entrada-salida para el problema.
- Robustez. Las redes neuronales de FFN son muy robustas, es decir, su rendimiento se degrada con gracia en presencia de cantidades crecientes de ruido (al contrario, por ejemplo, a PLS).

Sin embargo, también existen algunos problemas y desventajas de las RNA. Para algunos problemas de aproximación a través de funciones sigmoidales, los RNA están convergiendo lentamente, lo que refleja el hecho de que no se utiliza una visión física en la construcción del mapeo aproximado de parámetros en el resultado. El gran problema es el hecho de que las RNA no pueden explicar su predicción, los procesos que tienen lugar durante la capacitación de una red no se pueden interpretar bien y esta área aún está en desarrollo (Andrews et al., 1995), (Sharma et al., 2013). El número de pesos en una RNA es generalmente bastante grande y el tiempo para entrenar la RNA es demasiado alto. Por lo anterior, la redes neuronales presentan una ventaja respecto a otros métodos de clasificación e identificación de variables por su naturaleza adaptativa y dan solución a los requerimientos de estimación de las posiciones de fuga.

Capítulo 3

Sistema experimental: Planta Piloto del TecNM-ITTG

Este capítulo comprende el planteamiento del modelo dinámico del simulador del ducto y los conceptos empleados en el desarrollo de la metodología para la localización de fugas con RNAs. Al momento de diseñar la metodología basada en datos, es necesario tomar en cuenta diferentes factores que influyen en el desarrollo y en el buen funcionamiento de este. Inicialmente se describe el sistema físico de la planta piloto y como se realiza la obtención de datos experimentales, el sistema fue construida previamente en Bermúdez et al. (2017). Se desarrolla un simulador de la planta piloto para obtener el conjunto de datos para el entrenamiento de la RNA para localizar fugas, es necesario contar con el simulador validado para enriquecer la base de datos de fugas, por lo que, con el fin de profundizar en el tema, para desarrollar un método que permita localizar fugas hidráulicas en ductos se ha realizado una recopilación de información en la cual se detalla y se define los aspectos fundamentales para realizar el trabajo.

3.1. Sistema físico de la planta piloto

El sistema físico de la planta piloto en el Laboratorio de Hidráulica del TecNM-ITTG, fue diseñado y construido con la finalidad de desarrollar métodos de detección y localización de

fugas, la planta piloto está construida en con tubería PVC de 2 pulgadas en forma de serpentín, cuenta con sensores de presión y caudal aguas arriba y aguas abajo. En la Figura 3.1 se muestra el diagrama P&I de la tubería experimental, cuenta con una longitud total equivalente en una línea recta de 64.48 [m]. Tiene cuatro válvulas distribuidas arbitrariamente para simular las fugas. El flujo presurizado es impulsado por una bomba centrífuga de 5 [HP] controlada por un variador de frecuencia Micromaster 420. En los extremos de la tubería, se han montado transmisores de presión y flujo, que proporcionan las mediciones en línea requeridas para el diagnóstico de fugas. Los detalles completos sobre el diseño de la tubería y sus parámetros físicos se pueden consultar en Bermúdez et al. (2017).

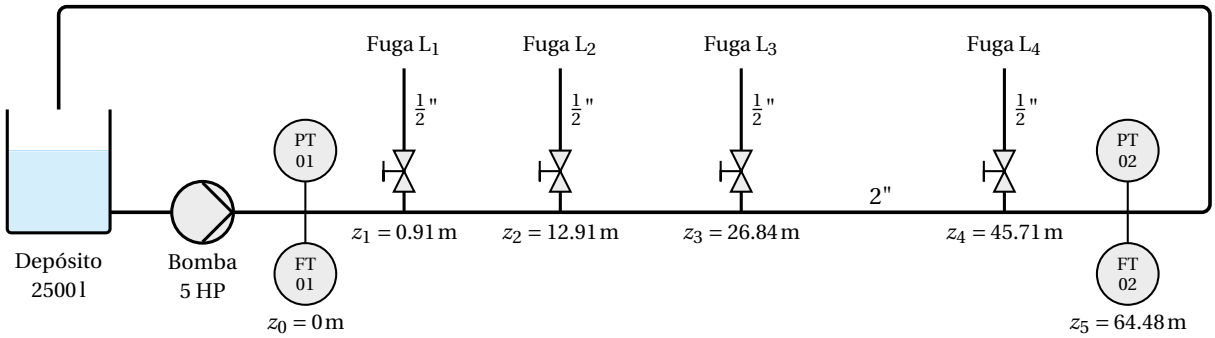


Figura 3.1: Diagrama P&I de la planta piloto TecNM-ITTG

Las posiciones de las válvulas para simular las fugas se distribuyen de la siguiente manera:

- Posición de la primera fuga $z_1 = 0.91[m]$.
- Posición de la segunda fuga $z_2 = 12.91[m]$.
- Posición de la tercera fuga $z_3 = 26.84[m]$.
- Posición de la cuarta fuga $z_4 = 45.71[m]$.

En la Figura 3.2 se muestra el diagrama esquemático de la planta piloto y los sensores industriales de presión Yokowaga modelo EJA530E y sensores de caudal magnético Signet modelo 2551, con las monturas adecuadas para cada sensor. Se cambiaron las mangueras de re-circulación del agua de las fugas y se agregaron recipientes para obtener fugas atmosféricas, se hizo este cambio con el fin de evitar la resistencia hidrálica que provocan las mangueras.

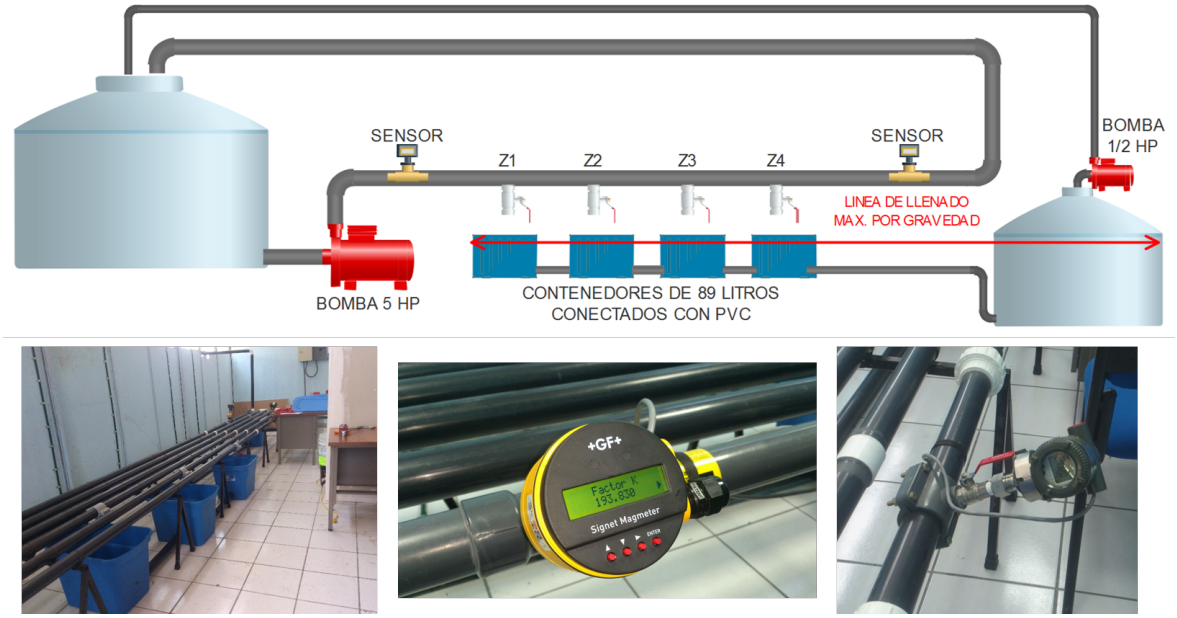


Figura 3.2: Diagrama esquemático de la planta piloto TecNM-ITTG

Finalmente, se reconfiguró el sistema de adquisición de datos, donde se obtienen lecturas de corriente directamente de los sensores, en la Figura 3.3 se observa el diagrama del sistema de adquisición de datos, utilizando chasis de alto rendimiento de National Instrument modelo cDAQ-9174 que tiene la capacidad de incorporar módulos de entradas y salidas analógicas y digitales, por lo cual, se utilizó el modulo para entradas de corriente NI-9203.

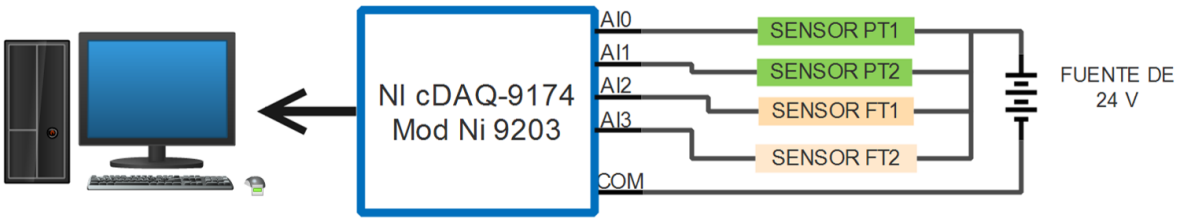


Figura 3.3: Diagrama del sistema de adquisición de datos

Las conexiones del sistema de adquisición de datos y control de instrumentación, se muestra en la Figura 3.4 donde se utiliza una DAQ NI-6002 con la conexión a un modulo de relevadores para activar la apertura y cierre de las electroválvulas que simulan de fugas. Además, se observan con las conexiones de los sensores de presión y caudal al modulo NI-9203. Para finalmente, conectar las dos tarjetas a la PC, y hacer la programación del sistema.

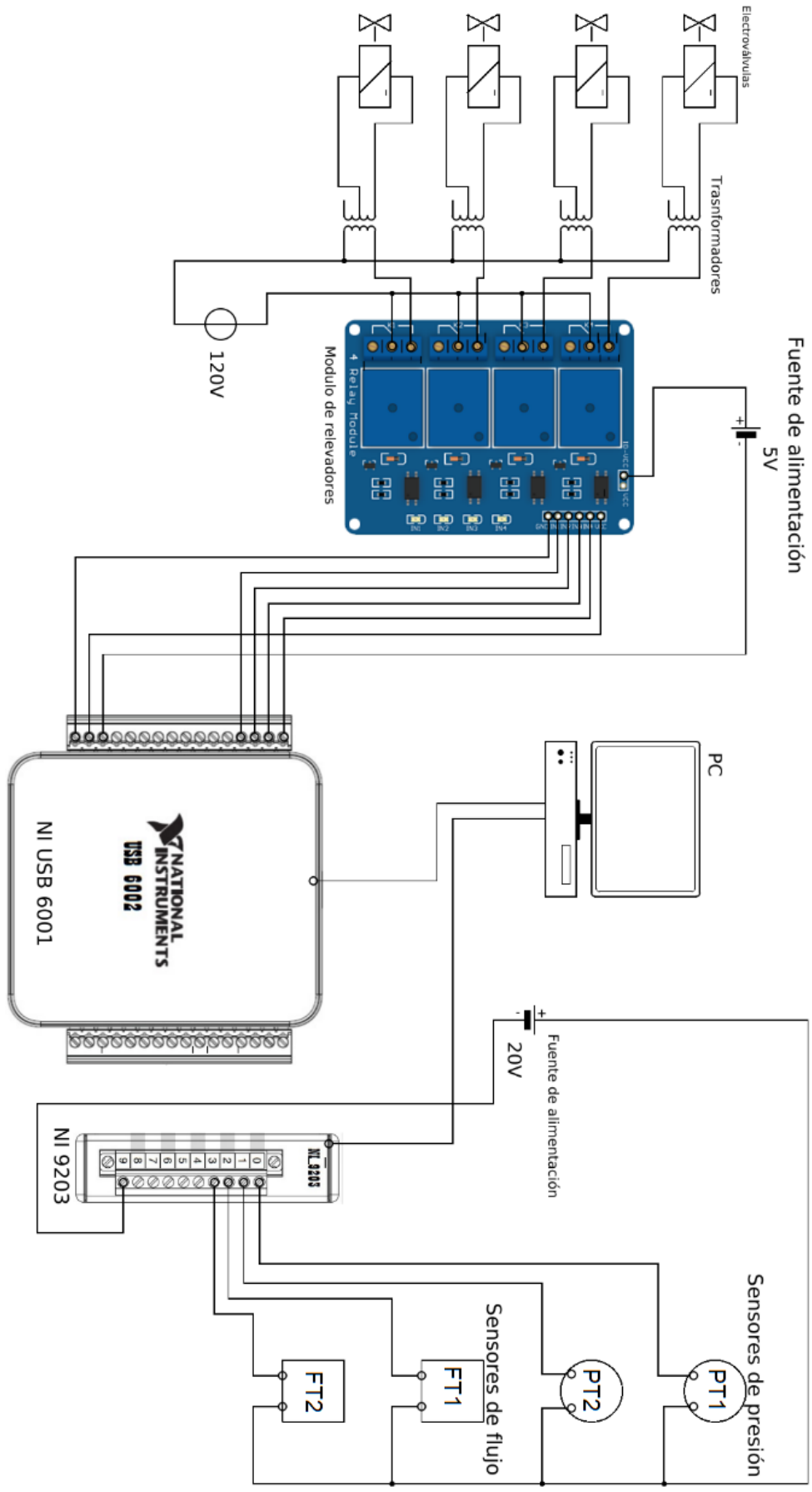


Figura 3.4: Diagrama completo de conexiones de la planta piloto.

Con esta instrumentación, junto con el software para el control de los actuadores y adquisición se complementa para obtener un sistema SCADA. Se desarrolla una interfaz de usuario para el sistema de adquisición de datos y los sensores instalados en la tubería.

3.1.1. Sistema SCADA de la planta piloto

Contando con la instrumentación y sistema de adquisición de datos conectados correctamente, se procede a crear la interfaz de usuario, en la cuál, se tendrá el monitoreo y el control del sistema de la planta piloto. en la Figura 3.5, se observa de manera general el diagrama de bloques del funcionamiento y flujo de información del sistema SCADA.

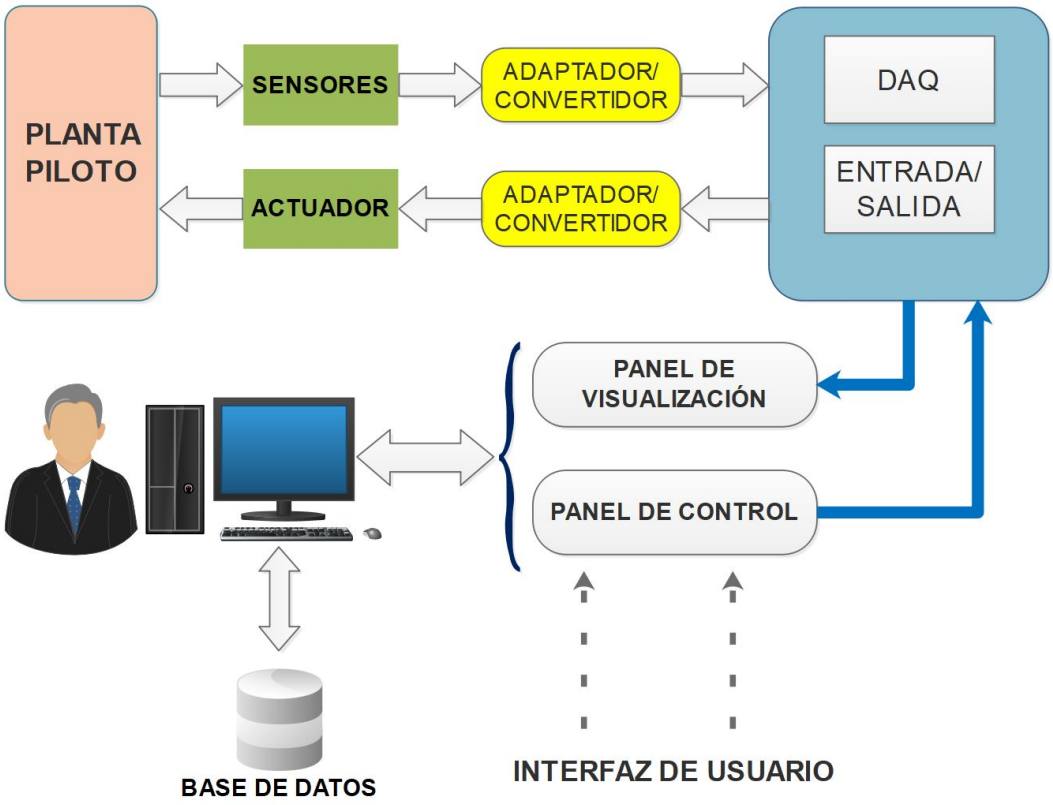


Figura 3.5: Diagrama del sistema SCADA de la planta piloto

En el proceso de utilización del planta piloto, se capturan datos de presión y caudal, estos parámetros son captados por un transductor, el cual alimenta una señal eléctrica a un transmisor, el cual entrega una señal análoga eléctrica en forma de corriente normalizada de 4

a 20 [mA]. Estas señales eléctricas son procesadas para ser transmitidas mediante técnicas digitales y eventualmente registradas por una computadora, por lo que se necesita hacer una conversión de datos análogo/digital o viceversa. Esta conversión la realiza la DAQ de National instruments.

Posteriormente, todas las señales digitales se envían hacia una computadora donde se reúne la información de toda la planta piloto. Simultáneamente se muestra la información en una interfaz de usuario en la computadora para que el operador pueda tomar decisiones; estos datos digitalizados son almacenados para su análisis, proporcionando así históricos para la toma de decisiones.

3.1.2. Interfaz de usuario del sistema SCADA

La interfaz de usuario se desarrolló con el ToolBox App Designer de MATLAB, en el cual se programaron las secuencias de control de las válvulas y la captura de datos de los sensores de presión y caudal. La magnitud medida por los indicadores se observa por medio de gráficas y manómetros virtuales para los sensores en tiempo real (Los códigos para desarrollar la aplicación se encuentran en el Apéndice B).

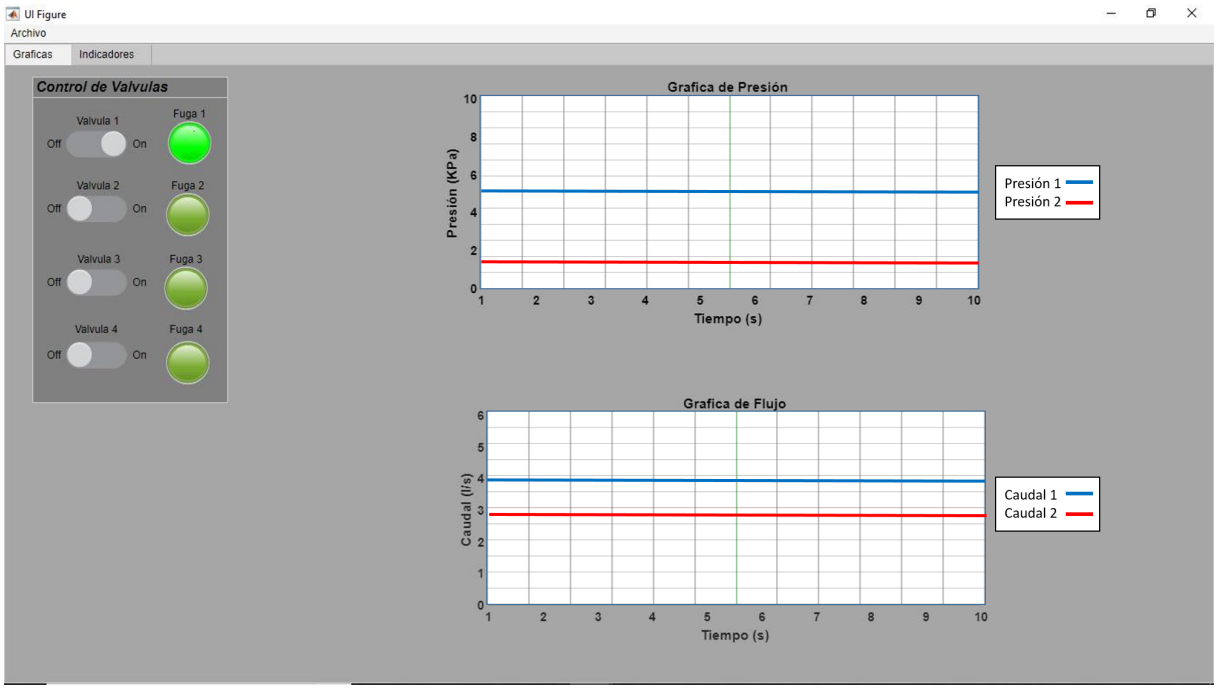


Figura 3.6: Interfaz de usuario: control de válvulas y gráfica de presión y caudal

La interfaz contiene un "switch"de encendido para cada válvula y un indicador cuando se activa, junto con las gráficas para observar el comportamiento de las presiones y caudales cuando la planta está en marcha. Por ejemplo, en la Figura 3.6 se observa que cuando la Fuga 1 se activa, los datos correspondientes a dicho evento son visualizados de manera gráfica en monitor de control. Mientas que en la Figura 3.7 muestran las presiones y caudales en tiempo real en manómetros virtuales.

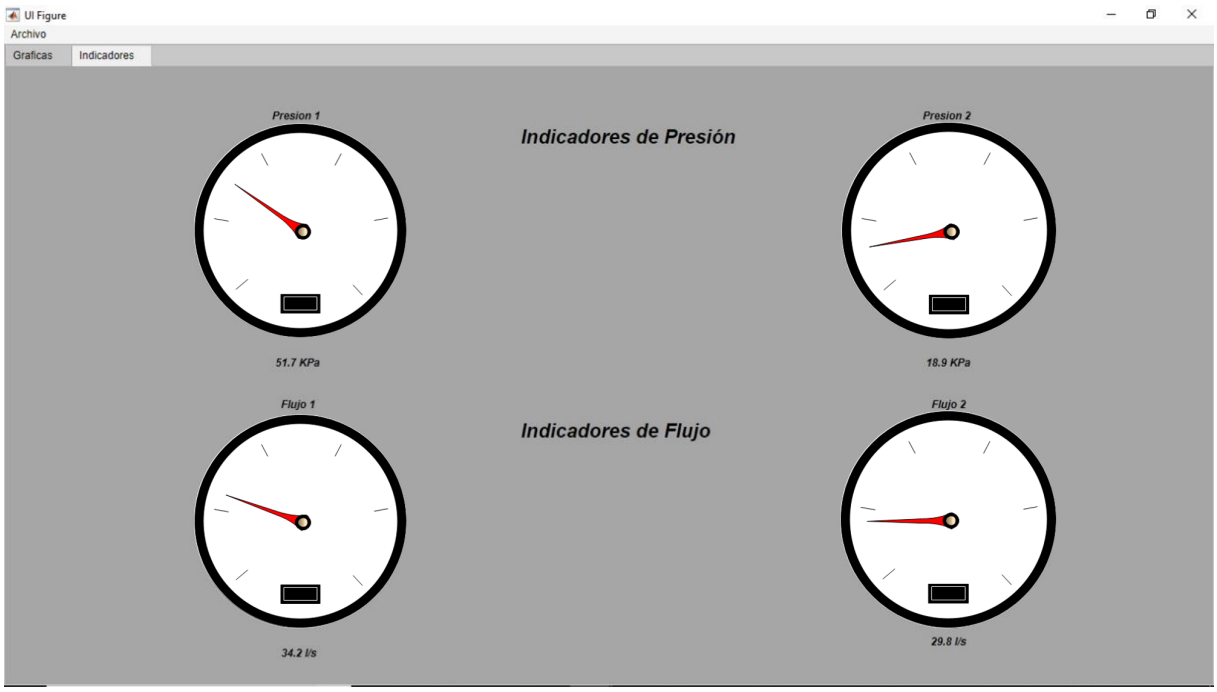


Figura 3.7: Interfaz de usuario: Visualización de los sensores en tiempo real

Para cada experimento se asigna el tiempo de duración de la prueba y el nombre del archivo donde se guardarán los datos obtenidos durante la duración de la corrida. La sistema permite realizar muchos procesos automáticamente, desde la interfaz, lo que permite que se optimicen procesos cuando se realizan las pruebas. Con esto, se optimizan varios procesos de se realizaron los experimentos para simulaciones de fuga con las cuatro válvulas disponibles.

3.2. Diseño del simulador de la planta piloto del TecNM-ITTG

En este punto es importante mencionar que a pesar de que se cuenta con una planta piloto para simular fugas, está no genera toda la información necesaria para entrenar la RNA. Esto debido a que por limitaciones físicas no es posible obtener datos de fugas en posiciones diferentes a la ubicación de las válvulas de fugas. Para solventar este problema se trabajó con un simulador de la planta piloto el cual contiene un modelo matemático validado por Santos-Ruiz et al. (2018) el cual permite simular fugas en diferentes puntos de la planta. El simulador que se utiliza fue diseñado en OPENMODELICA por el grupo de TURIX-Dynamics y se encuentra disponible en [https://github.com/esvandejesus/PipelineFMUsimulator.git], la Tabla 3.1 contiene los siguientes parámetros de la planta piloto:

Tabla 3.1: Parámetros de la planta piloto

Parámetro	Valor
Longitud equivalente, L	64.48 $[m]$
Diametro, D	0.0486 $[m]$
Rugosidad relativa, ε	4.84 x 10^{-4}
Viscosidad cinemática, ν	8.03 x $[10^{-7} m^2/s]$
Velocidad de la onda, b	422.754 $[m/s]$
Aceleración de la gravedad, g	9.782757 $[m/s^2]$

Internamente el simulador contiene el modelo dinámico de la planta piloto, el cual está contemplado como un fluido en un ducto horizontal cerrado, sin tomas laterales, se obtiene a partir de los principios físicos de conservación del impulso (momentum) y de conservación de la masa (ecuación de continuidad), en términos del tiempo t y de una variable espacial z definida convenientemente en la dirección axial de la tubería.

La dinámica espacio-temporal del flujo queda modelada por un conjunto de ecuaciones diferenciales parciales hiperbólicas (Chaudhry, 1979), comúnmente referidas como ecuaciones del golpe de ariete:

$$\frac{\partial H(z,t)}{\partial t} + \frac{b^2}{gA_r}\frac{\partial Q(z,t)}{\partial z} = 0 \tag{3.1}$$

$$\frac{1}{A_r}\frac{\partial Q(z,t)}{\partial t} + g\frac{\partial H(z,t)}{\partial z} - \frac{f(Q(z,t))|Q(z,t)|}{2DA_r^2} = 0 \tag{3.2}$$

Donde H indica la carga de presión en $m.c.a$, la Q indica el caudal en m^3/s, la b indica la velocidad de la onda de presión en m/s, la g indica la gravedad en m/s^2, la A_r indica el área transversal del ducto en m^2 y la D el diámetro del ducto en la tubería en m.
No se conoce una solución analítica explicita para las ecuaciones diferenciales parciales (3.1)-(3.2). Frecuentemente, la solución se obtiene numéricamente usando diferencias finitas o con el método de características (Conejero et al., 2016), y se requieren dos condiciones de frontera; estas pueden ser presiones o caudales conocidos en los extremos del tubo. Respecto a las condiciones del flujo turbulento, se considera el factor de fricción f como una función no lineal implícita del número de Reynolds (Re) y de la rugosidad interior ε de la tubería. Esta dependencia está dada por la fórmula de Colebrook-White (Colebrook and White, 1937), aunque en los cálculos frecuentemente se utiliza la aproximación explícita de Swamee-Jain (Swanee and Jain, 1976):

$$f = 0.25\left[log_{10}\left(\frac{\varepsilon}{3.7} + \frac{5.74}{Re^{0.9}}\right)\right]^{-2} \tag{3.3}$$

Para una tubería de longitud L, al considerar una partición espacial con N secciones y $N+1$ secciones,$\{z_i\}\{z_i = 0, z_2, ..., z_N, z_{N+1} = L\}$, y aproximando las derivadas parciales por diferencias finitas de primer orden, se obtiene una versión de (3.1)-(3.2) discretizada en la variable espacial z:

$$\frac{\partial H(z,t)}{\partial z_i} = \frac{\triangle H_i}{\triangle z_i} = \frac{H_i(t) - H_{i+1}(t)}{\triangle z_i}; \tag{3.4}$$

$$\frac{\partial Q(z,t)}{\partial z_i} = \frac{\triangle Q_i(t)}{\triangle z_i} = \frac{Q_i(t) - Q_{i+1}(t)}{\triangle z_i} \tag{3.5}$$

$$\frac{\partial H(t)}{\partial t} = \dot{H}_i \tag{3.6}$$

$$\frac{\partial Q(t)}{\partial t} = \dot{Q}_i \tag{3.7}$$

En cada sección i del ducto, el conjunto de ecuaciones del fluido se transforma en:

$$\dot{H}_{i+1}(t) = a_2 \frac{Q_i(t) - Q_{i+1}(t)}{\triangle z_i} \tag{3.8}$$

$$\dot{Q}_i(t) = a_1 \frac{H_i(t) - H_{i+1}(t)}{\triangle z_i} + \mu_i Q_i(t)|Q_i(t)| \tag{3.9}$$

donde:

$$a_1 = gA_r; a_2 = \frac{b^2}{gA_r}; \mu_i = -\frac{f}{2DA_r}$$

Para considerar en las Ecuaciones (3.8) y (3.9) de forma general, con el efecto del caudal de la k-ésima fuga en la posición $z_{f,k}$ del ducto, se debe incluir el modelo de Torricelli para flujo por orificio (Torres et al., 2014).

$$Q(z,0) = Q^0(z); H(z,0) = H^0(z) \tag{3.10}$$

donde:

$H(t)$ = Representa la función de Heaviside, la ocurrencia

de una fuga al instante t

$Q_{f,k}$ = Caudal k-ésimo de fugas en el ducto [m^3/s],

$z_{f,k}$ = Posición k-ésima de fuga [m].

Por lo tanto, para el efecto de una sola fuga se tiene:

$$\lambda_f = \sqrt{2g} A_f C_f \geq 0 \tag{3.11}$$

λ_f = Coeficiente de fuga [$m^{5/2}$]

C_f = Coeficiente de descarga.

A_f = Área del orificio de fuga [m^2].

$$\frac{\triangle Q_i(t)}{\triangle z_i} = \frac{Q_i(t) - Q_{i+1}(t) - Q_f(t)}{\triangle z_i} \tag{3.12}$$

Considerando las extracciones, las ecuaciones para una sección en el ducto se muestran de la siguiente manera. Estas ecuaciones ya representan el efecto de una fuga en una sola sección de tubería.

$$\dot{Q}_i(t) = a_1 \frac{H_i(t) - H_{i+1}(t)}{\triangle z_i} + \mu_i Q_i(t)|Q_i(t)|; \forall_i = 1...n \tag{3.13}$$

$$\dot{H}_{i+1}(t) = a_2 \frac{Q_i(t) - Q_{i+1}(t) - Q_{f,}(t)}{\triangle z_i}; \forall_i = 2...n-1 \tag{3.14}$$

Las dos últimas ecuaciones representan el caudal y la carga de presión, en una sección i del ducto, considerando el efecto de una fuga. el modelo en un formato esquemático se observa en la Figura 3.8.

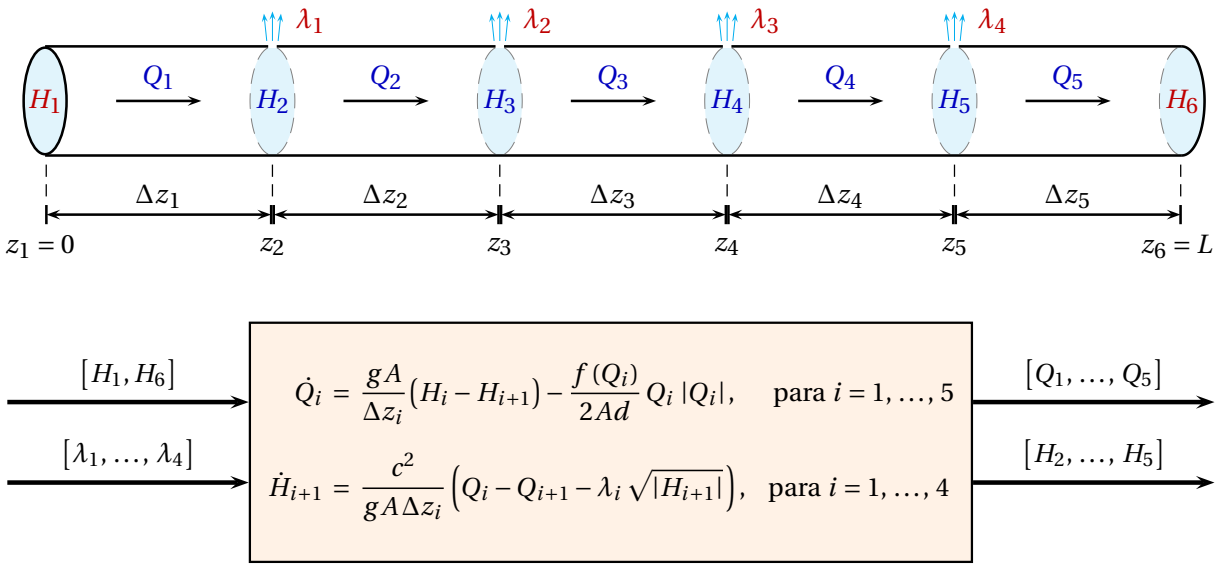


Figura 3.8: Modelo esquemático del simulador de la planta piloto

3.2.1. Unidad de Maquetas Funcionales (FMU) del Simulador de la planta piloto

El simulador tiene la capacidad de ser exportado de un modelo precompilado a un modelo de aplicación que integra el modelo matemático del sistema de tubería, para ser utilizado mediante herramientas como Simulink o el lenguaje Python. Esta modelo de exportación es realizado por una interfaz de maquetas funcionales (FMI), que define una interfaz estandariza-

da que se utilizará en simulaciones por computadoras. En la practica, la implementación del FMI mediante una herramienta de modelado de software permite la creación de un modelo de simulación que puede interconectarse o la creación de una biblioteca de software llamada FMU (Unidad de maquetas funcionales) (Pazold et al., 2012).

En la Figura 3.9 se observa el diagrama de bloque para la obtención de la FMU, donde se describe el modelo del sistema para el análisis de variables y parámetros que contiene el sistema de la planta piloto, para después generar la interfaz y encapsularlo por medio del Wrapper del FMI y finalmente generar la FMU que servirá para hacer las simulaciones.

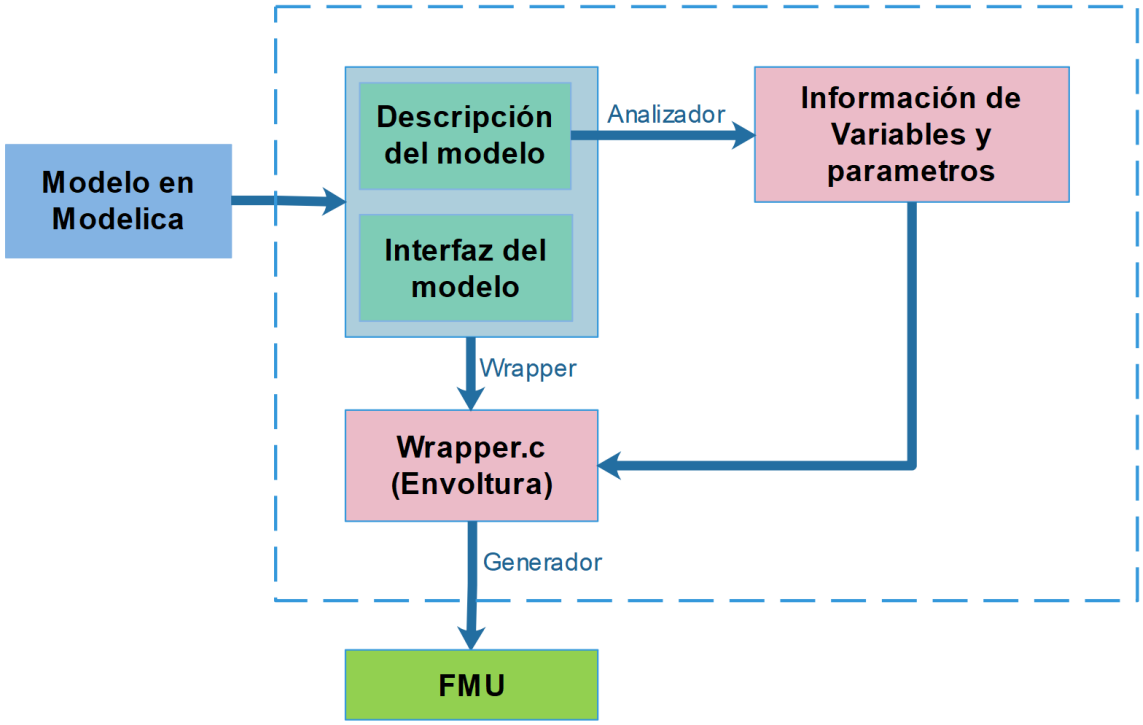


Figura 3.9: Diagrama de Bloques para obtencion de la FMU

Como resultado se obtuvo un archivo pipeline.FMU el cual contiene el modelo dinámico del ducto, los resultados de este simulador fueron validados con datos experimentales del sistema físico del TecNM-ITTG. El simulador es fundamental para obtener experimentos para simular fugas en diferentes posiciones, que en el sistema físico no se pueden obtener.

3.2.2. Utilización del simulador en Python

En este trabajo se utilizó el lenguaje Python para hacer las simulaciones necesarias para obtener los datos de caudal de entrada y salida, en el diagrama de bloques mostrado en la Figura 3.10, en el cual, se deben ingresar los parámetros del ducto, tiempo de simulación, una posición en metros de la fuga (z) determinada por el usuario y la Función escalón, se deben cargan las librerías del modelo, finalmente se obtendrán los caudales; estos resultados se guardan en un archivo con formato .CSV para el uso posterior de los datos de las simulaciones.

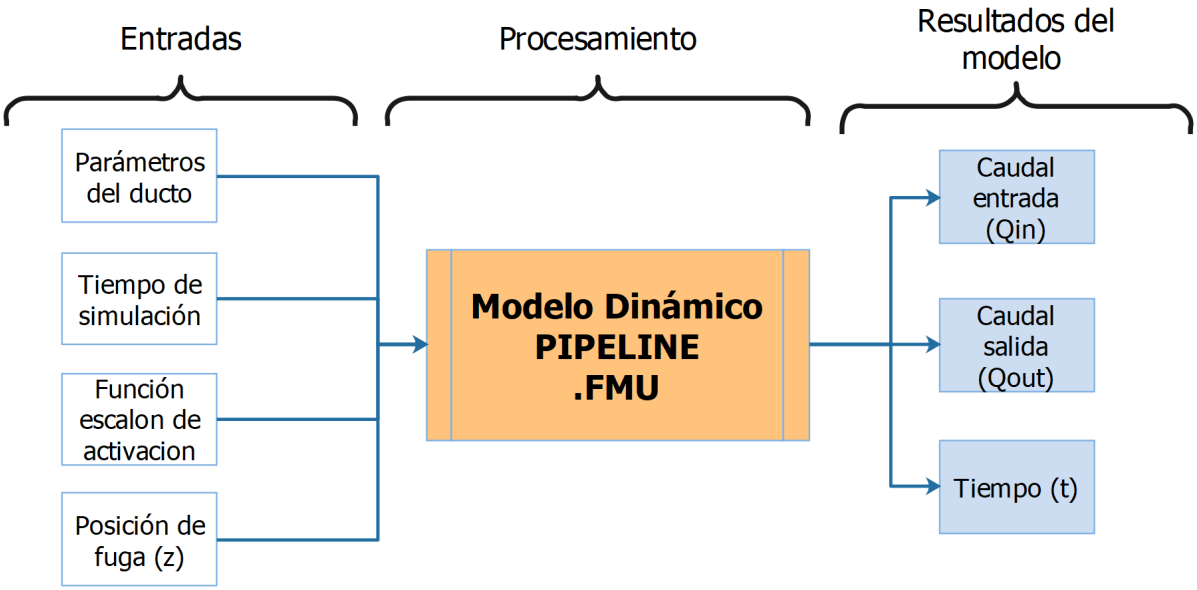


Figura 3.10: Diagrama de flujo para simulación y obtención de caudales

Capítulo 4

Localización de fugas con RNAs

La metodología empleada para la localización de fugas hidráulicas se considera la aplicación de Redes Neuronales Artificiales(RNA), para cumplir con el objetivo general de la tesis. Para realizar la metodología, se considera el uso de dos RNAs, la primera red (RNA-1) se utilizará para calcular el factor de fricción, considerado entre los más importantes en hidráulica. La segunda red (RNA-2) tiene el propósito de estimar la posición de una fuga (z). Inicialmente el uso de una RNA-1, diseñado por Brkić and Ćojbašić (2016), fue entrenada usando como entradas el número de Reynolds (Re) y la rugosidad relativa (ε/D). Las fricciones (entrada y salida) estimadas por la RNA-1, serán utilizadas junto con las presiones como entradas para la RNA-2.

Dada la necesidad de contar con una base de datos para entrenamiento y validación de la RNA-2, se obtiene un conjunto de datos experimentales de la planta piloto del ITTG, por consiguiente, para potenciar y enriquecer la base de datos, se utiliza un simulador del modelo dinámico del ducto, ya que, es imposible provocar fugas en cualquier sección del ducto; resultando fundamental para la generación de datos para entrenamiento de la RNA-2 para localización de fugas.

Planteando que la solución para la localización de fugas es un método híbrido, inicialmente se requiere el modelo para generar datos en los distintos escenarios de fugas a través del ducto, y es pieza clave para el entrenamiento de la RNA-2, lo interesante es que dicho modelo debe de estar bien diseñado y validado con datos experimentales. Por otra parte, el entrenamiento

y validación de la RNA-2 se hace uso de datos de simulación y datos experimentales, lo cual robustece el aprendizaje de la RNA-2. Lo más interesante es, que en la etapa de pruebas, basado exclusivamente en datos experimentales, los resultados son muy buenos.

4.1. Diseño de la RNA-1: Cálculo de factor de fricción

En hidráulica, la ecuación de Colebrook se usa actualmente para el calculo del factor de fricción. Al clasificar los datos disponibles y los del experimento realizado por Colebrook and White (1937), desarrolló un ajuste de curva que describía la rugosidad de transición, entre la zona turbulenta lisa y la rugosa (Colebrook et al., 1939). La ecuación de Colebrook también se considera como una base adecuada para el diagrama de Moody, siendo ampliamente utilizado exceptuando su zona laminar (Moody, 1944). En otras palabras, al dibujar su diagrama, Moody usó la ecuación de Colebrook para toda la zona turbulenta y para la zona laminar. El diagrama de Moody es un gráfico en forma no dimensional que relaciona el factor de fricción de (f) el número de Reynolds (Re) y la rugosidad relativa (ε/D) para un flujo desarrollado en una tubería circular (Figura 4.1).

Sin embargo, la ecuación de Colebrook es implícita con respecto al factor de fricción (f). No se puede reorganizar para obtener el factor de fricción directamente sin un cálculo aproximado. Por lo que, se tienen que usar métodos iterativos para aproximar a la solución de la ecuación de Colebrook. La RNA-1 tiene un enfoque no-iterativo, se usará para calcular el factor de fricción.

Para configurar el modelo RNA-1, los parámetros de entrada del Número de Reynolds (Re) y la rugosidad relativa de la tubería (ε/D) se transformaron a escalas logarítmicas. La RNA propuesta demuestra el error relativo de hasta el 0.07 % que tuvo la alta precisión en comparación con la gran mayoría de las aproximaciones explícitas precisas de la ecuación de Colebrook.

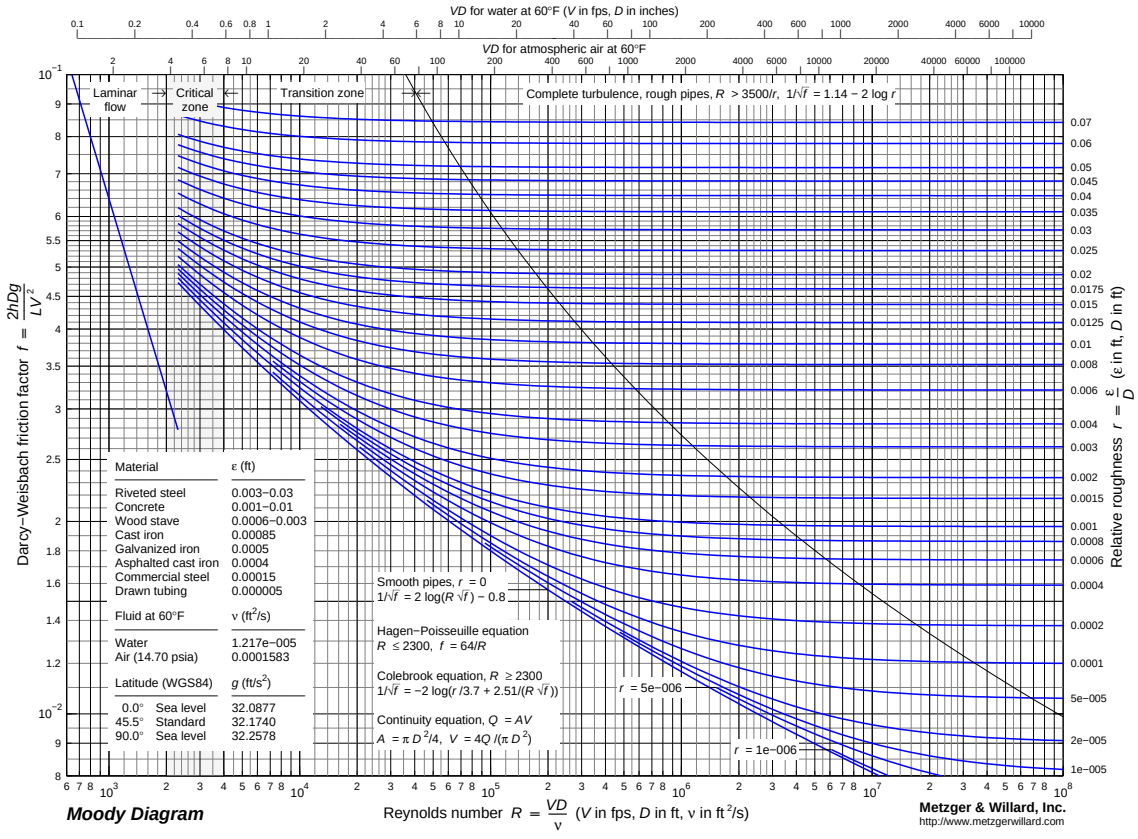
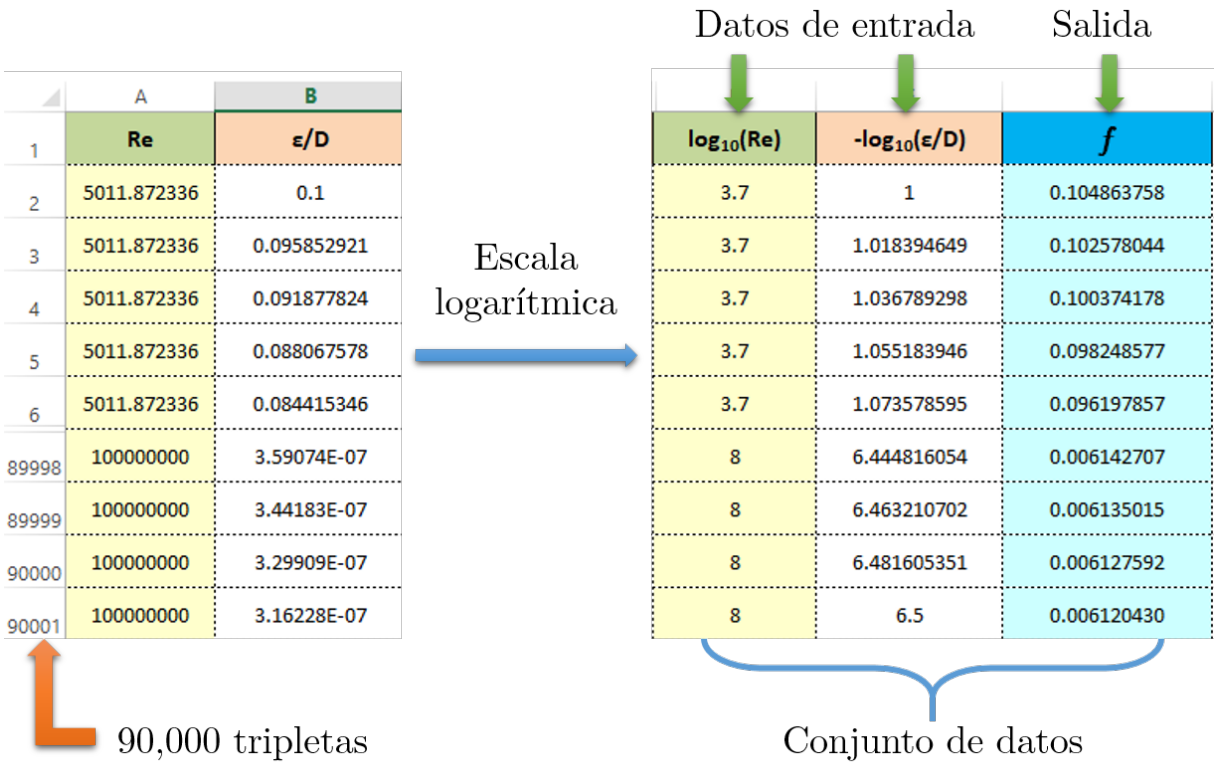


Figura 4.1: Diagrama de Moody para el factor de fricción en ductos de paredes lisas y rugosas

4.1.1. Preparación del conjunto de datos para la RNA-1

Para generar el conjunto de entrenamiento para el modelo ANN, la ecuación de Colebrook se resolvió de manera iterativa. La solución iterativa se usa porque se requirió una solución altamente precisa del factor de fricción (f), mientras que mientras tanto la carga computacional fue irrelevante, ya que fue un esfuerzo único para preparar los datos de entrenamiento. El conjunto de datos de entrenamiento se puede preparar de manera eficiente utilizando solucionadores de hojas de cálculo, como MS-Excel. Para obtener la mayor precisión en el cálculo utilizando MS-Excel, se debe habilitar el cálculo iterativo y el número máximo de iteraciones (se establece en 32.767 iteraciones, que fue el número máximo de ciclos permitido por el software con la mayor precisión) tiene que ser configurado. Se usó un conjunto de datos de 90,000 tripletas en los que los valores del factor de fricción (f) se generaron utilizando los

valores del numero de Reynols de $5000 - 10^8$ y para la rugosidad relativa se usaron del rango de $10^{-7} - 0.1$.



	A	B
1	Re	ε/D
2	5011.872336	0.1
3	5011.872336	0.095852921
4	5011.872336	0.091877824
5	5011.872336	0.088067578
6	5011.872336	0.084415346
89998	100000000	3.59074E-07
89999	100000000	3.44183E-07
90000	100000000	3.29909E-07
90001	100000000	3.16228E-07

$\log_{10}(Re)$	$-\log_{10}(\varepsilon/D)$	f
3.7	1	0.104863758
3.7	1.018394649	0.102578044
3.7	1.036789298	0.100374178
3.7	1.055183946	0.098248577
3.7	1.073578595	0.096197857
8	6.444816054	0.006142707
8	6.463210702	0.006135015
8	6.481605351	0.006127592
8	6.5	0.006120430

Figura 4.2: Generación de conjunto de datos para entrenamiento de la RNA-1

La conjetura principal de la red está relacionada con los rangos de los parámetros de entrada en que la rugosidad relativa es muy pequeña ya que varía de 10^{-7} a 0.1, mientras que el número de Reynolds es considerablemente grande con el rango de 5000 a 10^8, este problema puede evitar que la RNA se capacite adecuadamente y conducir a resultados menos precisos en la fase de aplicación, por lo tanto, el conjunto de datos de entrada sin procesar es normalizado por una escala logarítmica para proporcionar los datos de entrada de la RNA con el mismo orden de magnitud. Como se puede observar en la Figura 4.2, la transformación logarítmica se aplica para (Re) y (ε/D) quedando los valores de entrada en el nuevo dominio donde el $log(Re)$ está en el rango entre 3.7 y 8, mientras que el $-log(\varepsilon/D)$ está en el rango entre 1 y 6.5. A continuación describe la arquitectura y entrenamiento de la RNA-1.

4.1.2. Arquitectura de la RNA-1

Se utiliza una estructura de red neuronal prealimentada hacia adelante (FFBPN) que consta de tres capas (Figura 4.3). La primera capa de entrada tiene dos neuronas, la segunda capa oculta tiene cincuenta neuronas y la tercera capa de salida tiene una neurona, con una función de transferencia sigmoidea en la capa oculta y una función de transferencia lineal en la capa de salida.

En general, una RNA debe ser entrenada o adaptada, ya sea antes o durante su uso. La RNA-1 utilizada fue debidamente entrenada y validada por un entrenamiento supervisado fuera de línea antes de la aplicación de la red en la que se utilizaron los datos obtenidos por la solución iterativa de la ecuación de Colebrook.

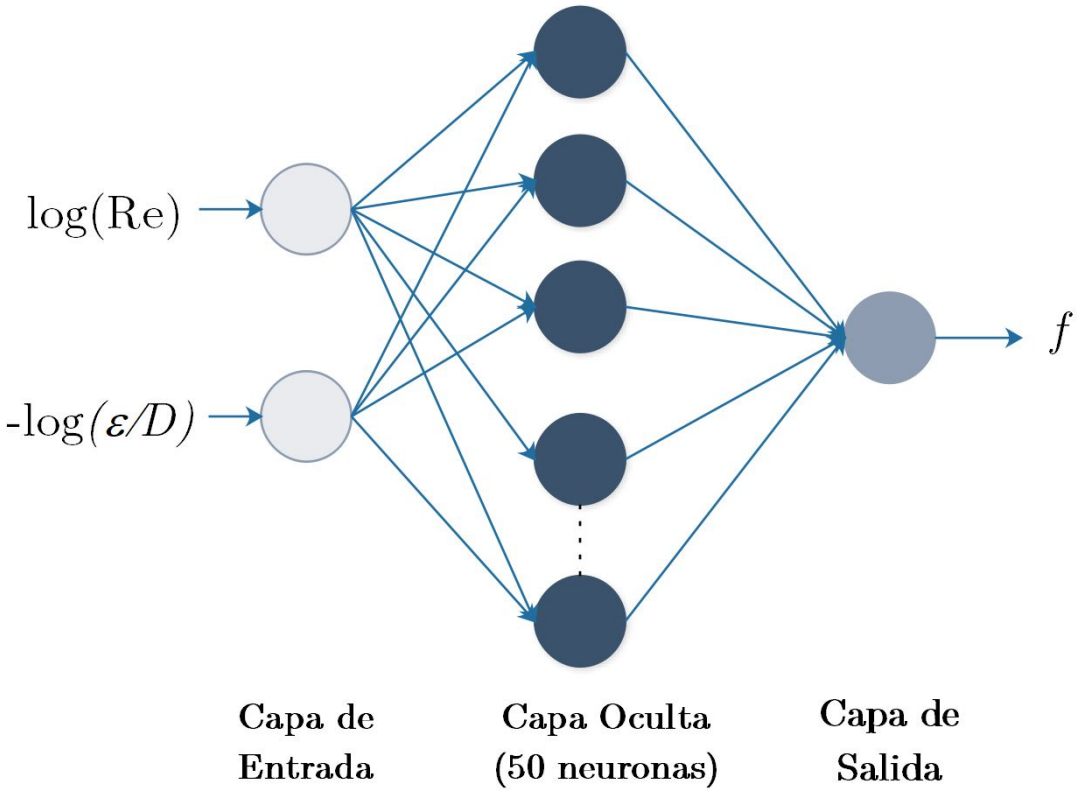


Figura 4.3: Arquitectura propuesta para la RNA-1

La RNAs puede imitar funciones de manera eficiente y reconocer patrones. Pueden ser entrenados para resolver un problema (habilidad para aprender). La calidad de esta solución depende en gran medida de la cantidad de datos disponibles para la capacitación y la

estructura de una red. Debe subrayarse que la RNA-1 desarrollada no utiliza la ecuación de Colebrook para el cálculo. Utiliza solo los resultados producidos por la ecuación de Colebrook para establecer sus patrones internos. Cada red neuronal se considera un sistema de caja negra; por lo tanto, puede verse en términos de sus entradas y salidas sin ningún conocimiento sobre su funcionamiento interno y sus componentes internos. Por lo tanto, el conjunto de datos de entrada sin procesar debe normalizarse para proporcionar los datos de entrada para la RNA-1 con aproximadamente el mismo orden de magnitud. El conjunto de datos preparado completo se dividió en subgrupos de entrenamiento, validación y prueba:

- Conjunto de entrenamiento (70 %, 63,000 tripletas)
- Conjunto de Validación (15 %, 10,500 tripletas)
- Conjunto para pruebas (15 %, 10,500 tripletas)

Se procede a realizar el entrenamiento que será descrito en siguiente sección.

4.1.3. Entrenamiento de la RNA-1

En la Figura 4.4 se observa el esquema completo del proceso de entrenamiento de la RNA. Para el entrenamiento de la RNA-1, se utilizó el algoritmo Levenberg-Marquardt de retropropagación, mientras que el error cuadrático medio (MSE) se usó como medida de rendimiento durante la fase de entrenamiento. Los valores de MSE para esta estructura de RNA-1 se calcularon entre 10^{-12} después de 5,000 épocas de entrenamiento.

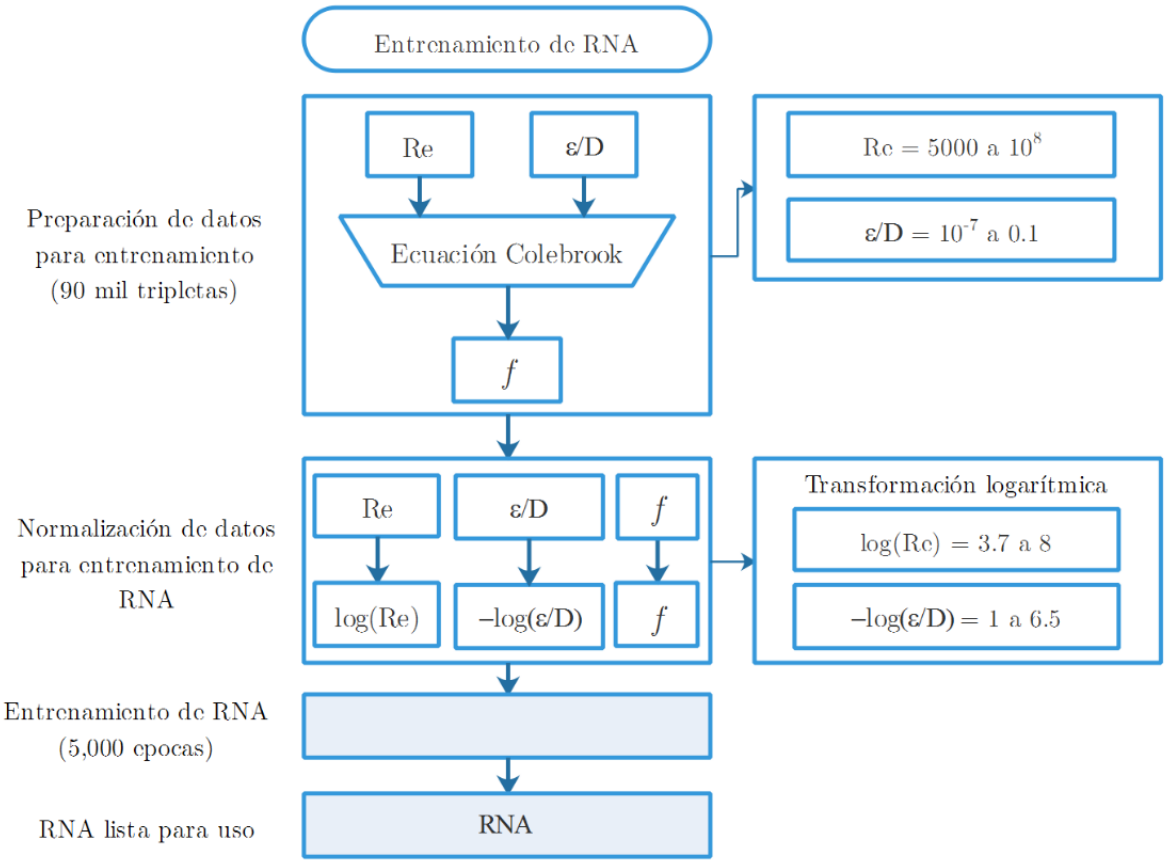


Figura 4.4: Esquema de proceso de entrenamiento de la RNA

El objetivo principal era minimizar la función de rendimiento, en este caso la función MSE (Ecuación 4.1), que se define como.

$$MSE = \frac{1}{Y}\sum_{k=1}^{Y} e_k^2 = \frac{1}{Y}\sum_{k=1}^{Y} (t_k - y_k)^2 \tag{4.1}$$

donde Y indica el número de muestras, e_k es el error de la red neuronal, y t_k son los valores objetivo, mientras que y_k son valores de salida de red.

El aprendizaje de la RNA-1 se detiene cuando el índice de error resulta aceptablemente pequeño para cada uno de los patrones aprendidos o cuando el número máximo de iteraciones del proceso ha sido alcanzado.

4.1.4. Exactitud de los resultados estimados por la RNA-1

El rendimiento de los conjuntos de datos de entrenamiento, pruebas y validación en comparación con la salida deseada es mostrado en la Figura 4.5.

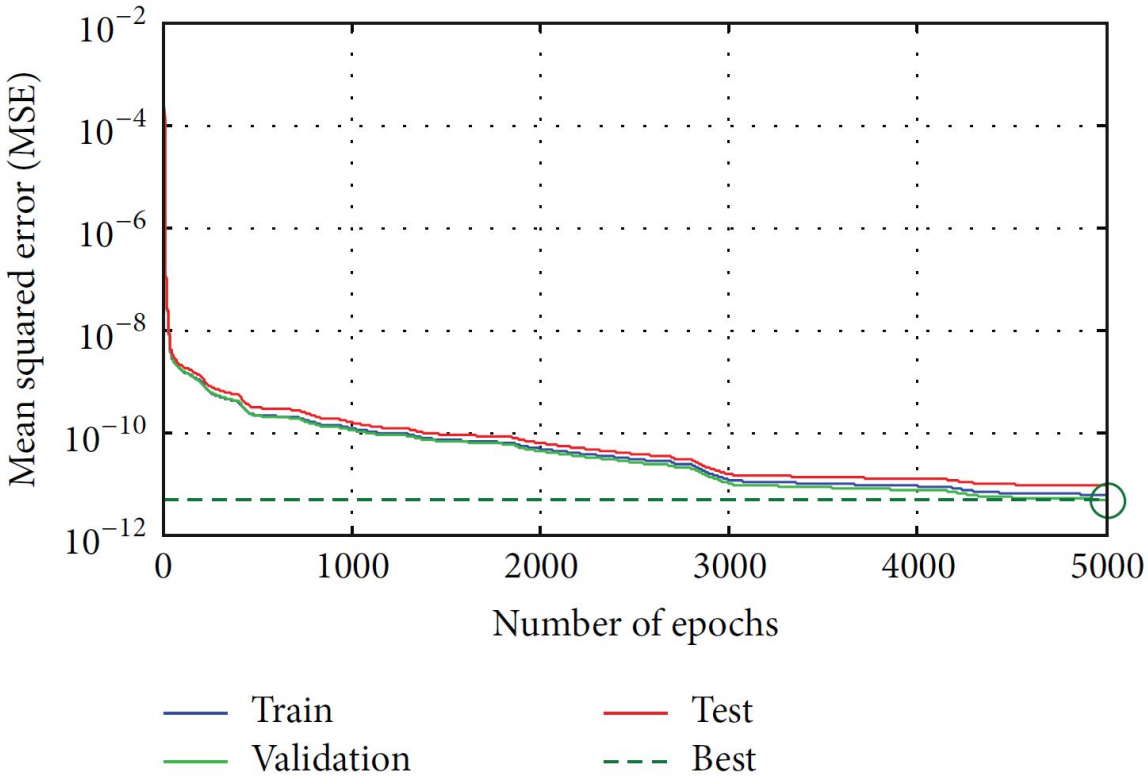


Figura 4.5: El error cuadrático medio (MSE) durante el proceso del entrenamiento de la propuesta de la RNA-1

Hay que tener en cuenta que hay tres niveles en la exactitud (Brkić, 2014):

1. El primer nivel está relacionado con la naturaleza de la ecuación de Colebrook, que es una relación empírica (de hecho, existe la posibilidad de usar otras ecuaciones con mayor precisión y, por lo tanto, la metodología mostrada se puede usar para desarrollar las RNAs para tal caso).

2. El segundo nivel explica la precisión relacionada con la solución de la ecuación de Colebrook; la ecuación de Colebrook se puede resolver de manera precisa mediante el procedimiento iterativo.

3. El tercero está relacionado con las estructuras de RNA-1 propuestas y las aproximaciones relevantes que pueden usarse para evitar el procedimiento iterativo; sus errores pueden estimarse y compararse con el error de la solución iterativa (el error obtenido de la estructura RNA-1 sugerida pertenece a la tercera categoría).

El error relativo del factor de fricción estimado a través de la estructura RNA-1 propuesta en esto es de hasta 0.07% (Figura 4.6). Esto significa que el enfoque RNA-1 propuesto puede usarse no solo como un enfoque extremadamente preciso, sino también como uno computacionalmente efectivo.

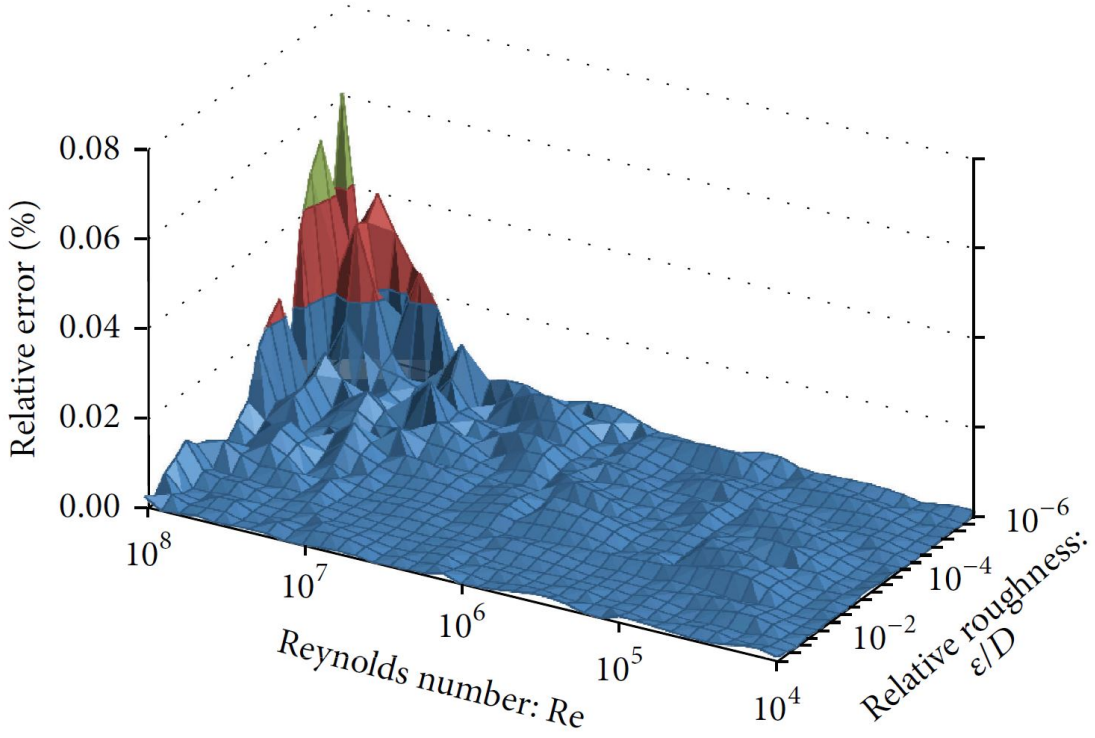


Figura 4.6: Distribución del error estimado producido por la RNA-1 en comparación con la ecuación de Colebrook

Además, hasta cierto punto, un aumento en la complejidad de la estructura de RNA-1 aumentaría su potencial para producir resultados aún más precisos. Por lo tanto, el equilibrio correcto de precisión y complejidad es necesario durante la fase de diseño de la red. Además, la precisión depende de la cantidad de términos en el conjunto de entrenamiento. La complejidad de la red en la fase de explotación es relativamente poco importante ya que la RNA-1 es

una especie de caja negra. Puede producir salidas para entradas y su complejidad interna no es crucial (Olden and Jackson, 2002).

La RNA-1 se puede usar fácilmente y se aplicaría sin ninguna dificultad debido a la complejidad de su estructura, en contraste con el uso de las fórmulas aproximadas (Giustolisi et al., 2011). Las mismas circunstancias de comodidad pueden ser experimentadas, como al aplicar los códigos de computadora preparados para las fórmulas aproximadas. Se podrán ingresar datos de entrada en un programa y una computadora debe poder producir salidas sin ningún inconveniente. Según las Figuras 4.6, el error relativo no se distribuye equitativamente en todo el rango práctico del número de Reynolds (Re) y la rugosidad relativa (ε/D). La misma situación con esta distribución del error se produciría para las aproximaciones explícitas que muestra Winning and Coole (2013). El error relativo producido por la RNA-1 se acumula en la zona con valor es pequeños de la rugosidad relativa (ε/D) los valores altos del número de Reynolds (Re).

4.2. Diseño de la RNA-2: Estimación de la posición de fuga

Por lo anterior, para realizar un procedimiento eficiente y preciso para la estimación del factor de fricción del flujo (f), se utilizó un enfoque basado en el sistema computacionalmente inteligente. Se desarrolla la RNA-1 para la solución del problema. Por consiguiente, en esta sección se presenta el diseño de una RNA-2 para estimar la posición de la fuga usando datos de presión de entrada y salida junto con las fricciones de entrada y salida (Calculadas por la RNA-1).

Se considera el uso de datos de los sensores de Presión (H) y Caudal (Q)(Aguas arriba y aguas abajo), ampliamente utilizados en plantas hidráulicas, para mediar parámetros propios de los sistemas hidráulicos. Sin embargo, como se ha mencionado existen factores muy importantes que se deben calcular al hacer diagnósticos de fugas, utilizando métodos de detección como lo es el factor de fricción. La RNA-1 estima el factor de fricción con gran precisión, considerando además, que ya no es necesario recalcular iterativamente el factor de fricción, reduciendo la carga computacional. La RNA-1 requiere del numero de Reynolds

(Re) y la rugosidad relativa (ε/D) como entrada para obtener el factor de fricción. Es posible obtener el numero de Reynolds (Re)con los datos de Caudales (Q), que esta directamente relacionado con la resistencia hidráulica y la rugosidad relativa (ε/D) se mantiene de acuerdo a las características físicas de la tubería.

La resistencia hidráulica depende del caudal que se considera como el principal problema en la determinación del factor de fricción del flujo hidráulico (f). Para una tubería, la resistencia hidráulica generalmente se expresa a través del factor de fricción de (f) que no es una cantidad constante. El factor de fricción (f) está relacionado con el caudal (Q), más precisamente, con el número de Reynolds (Re) y la rugosidad relativa (ε/D). Además, ambos, el número de Reynolds (Re) y la rugosidad relativa(ε/D) dependen del caudal. Esta relación se ve reflejada en la obtención del numero de Reynolds (Re) mediante la Ecuación (4.2), se trata de un parámetro adimensional cuyo valor indica si el flujo sigue un modelo laminar o turbulento; depende de la velocidad del fluido, el diámetro de la tubería y la viscosidad cinemática Streeter et al. (1988).

$$Re = \frac{D}{A_r \nu} Q \tag{4.2}$$

Como se puede observar, el número de Reynolds (Re) se ve afectado por la velocidad del flujo, mientras que la rugosidad relativa (ε/D) depende del grosor de una región de flujo dentro de las tuberías, denominada capa límite, que se encuentra cerca de la superficie interna de la pared de la tubería Afzal et al. (2013). Por el contrario, en este documento la rugosidad relativa (ε/D) conserva su definición clásica, lo que implica que no debe variar con el caudal (se tratará de manera efectiva como una cantidad geométrica y, por lo tanto, debe ser constante independientemente de la velocidad de flujo con la advertencia de que el flujo es turbulento).

Cuando una fuga aparece, existe una discontinuidad en los parámetros respecto a las condiciones nominales del sistema. En la Figura 4.7 se observa que la fuga ocurre en el segundo 90, el gasto volumétrico aumenta; cambiando los parámetros de los caudales. De manera que, al cambiar el caudal, afectará directamente al número de Reynolds, por consiguiente el factor de fricción aguas arriba (f_{in}) y aguas abajo (f_{out}) cambiará cuando ocurra una fuga.

Por esta razón, se considera el uso de las fricciones (entrada (f_{in}) y salida (f_{out})) después de que ocurre una fuga como datos de entradas para la RNA-2 junto con las presiones (entrada

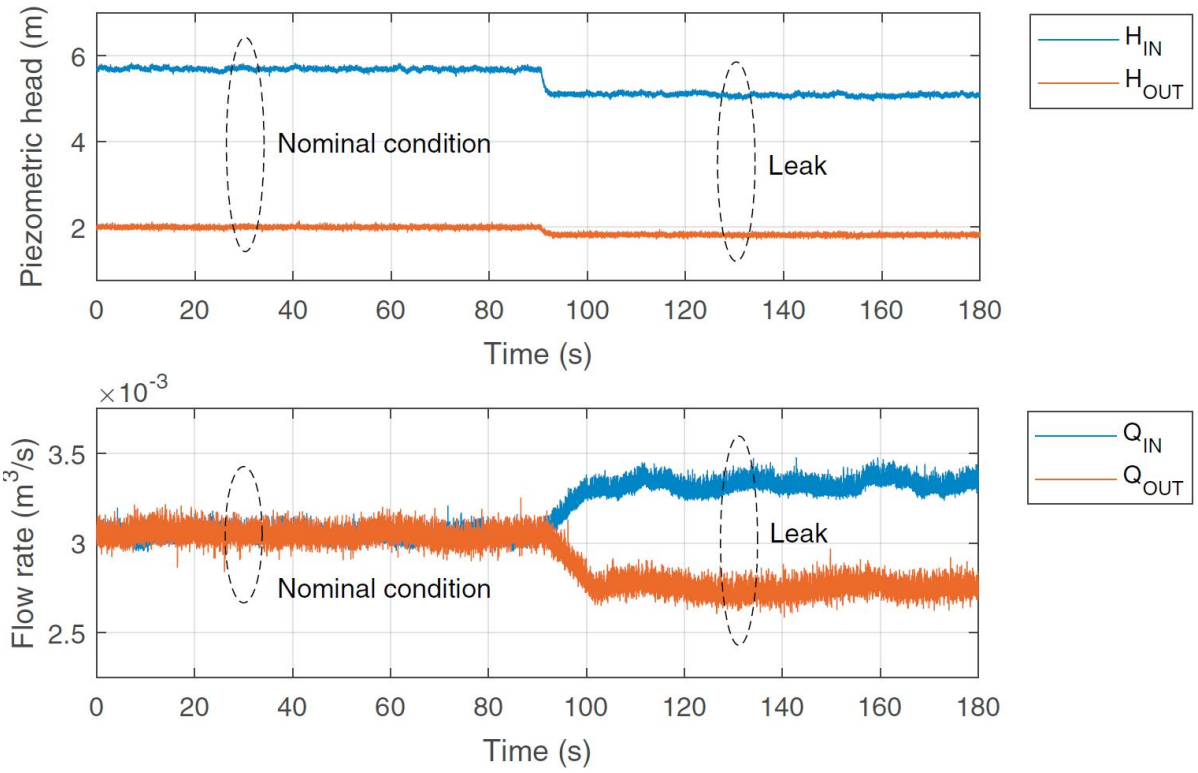


Figura 4.7: Gráfica de una fuga con presiones y caudales de entrada (azul) y salida (rojo)

(H_{in}) y salida (H_{out})), teniendo como salida la posición de la fuga (z). Los modelos de RNAs como el que se desarrollan aquí se pueden generar fácilmente en el software MATLAB.

4.2.1. Obtención de conjunto de datos para la RNA-2

La base de datos para entrenamiento y validación se obtuvieron de dos fuentes: El primer conjunto fue tomado de la planta piloto del TecNM-ITTG en Tuxtla Gutiérrez, Chiapas. La función principal de la planta es simular fugas hidráulicas por medio de cuatro válvulas ubicadas en diferentes posiciones a lo largo del ducto. El segundo conjunto de datos se obtuvo por medio de un simulador que contiene internamente el modelo dinámico de la planta piloto, en el cual, se contempla un fluido presurizado en un ducto horizontal cerrado, sin tomas laterales, dentro del simulador es posible hacer pruebas con fugas en diferentes posiciones a lo largo de la tubería. Nótese que es necesario contar con un simulador validado para enriquecer la base de datos de fugas, ya que es imposible físicamente provocar fugas en cualquier sección de la planta piloto.

Conjunto de datos experimentales

Se realizaron pruebas en el planta piloto, siguiendo el procedimiento que a continuación se presenta:

1. Se configura el programa de planta para hacer los experimentos, donde se asigna el tiempo por cada corrida, quedando de la siguiente forma:
   - 180 segundos antes de la fuga.
   - 180 segundos después de la fuga
2. Se asigna un nombre al archivo donde se guardarán los datos capturados
3. La frecuencia de muestreo para la captura de datos es de 1000 muestras por segundo

Se asigna arbitrariamente una frecuencia de 45 hz al variador de velocidad de la bomba de 5 H.P., equivalente al 75 % de su capacidad. Como resultado las condiciones nominales de operación de planta piloto, se obtuvo un Presión de entrada (H_{in}) de 5.7087 [mca] y Presión de salida (H_{out}) de 1.998 [mca].

Se realizaron corridas para cada fuga $z1...z4$ una por una, repitiendo los experimentos 4 veces. Posteriormente, se etiqueta cada experimento de acuerdo al numero de fuga y numero de experimento. La función principales de este conjunto de datos es validar y evaluar el rendimiento de estimación de la fuga por la RNA-2.

Conjunto de datos simulados

Se procede a obtener datos simulados de flujo obtenidos del simulador del planta piloto del TecNM-ITTG. Primero, se obtuvieron los datos de flujo con el simulador, seguidamente se calculó el número de Reynolds con los caudales, se agrega la rugosidad relativa (ε/D) (propiedad del material del ducto), se hace la escala logarítmica de los datos, se obtiene las fricciones de entrada y salida calculadas por la RNA-1, los datos son ordenados y las presiones (entradas y salidas) son agregados, en los siguientes párrafos este procedimiento es explicado en detalle:

Primero. El simulador utilizado fue diseñado en OPENMODELICA y exportado en una librería .FMU para su uso en un Script de Python, donde se cargaron los parámetros de la tubería y las condiciones de operación de la presión de entrada y salida, junto con la posición de fuga.

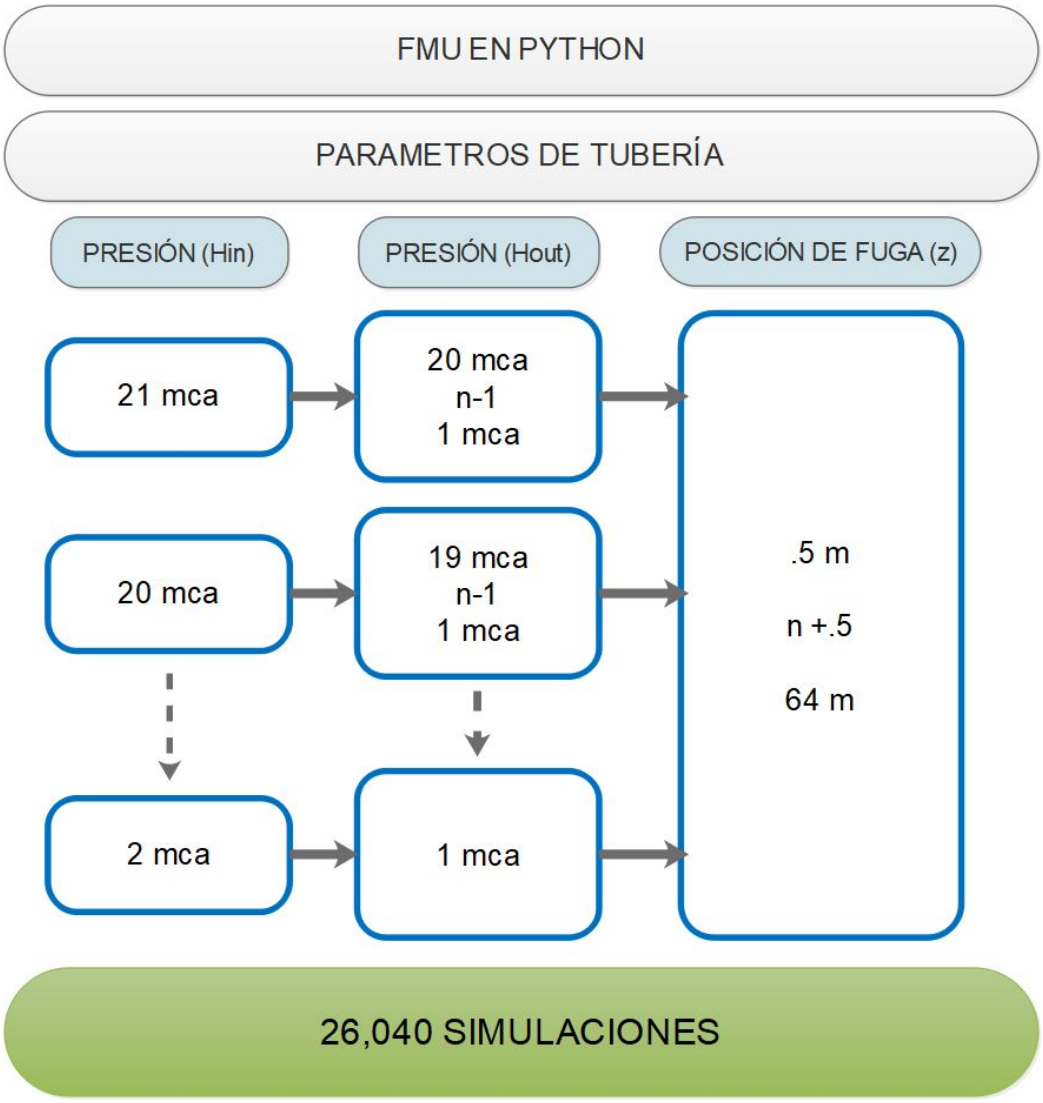


Figura 4.8: Diagrama del procedimiento para realizar las simulaciones

En el Diagrama de la Figura 4.8 muestra el procedimiento para obtener un amplio rango de datos con diferentes las condiciones de operación para el entrenamiento de la RNA-2, donde se propuso realizar pruebas con diferentes diferenciales de presión, variando (H_{in}) de 21 [mca] a 2 [mca] y (H_{out}) de 20 [mca] a 1 [mca], a su vez para cada simulación en diferentes

posiciones de fuga variando 0.5m a lo largo de la tubería de 64 m, por lo tanto, se obtuvieron datos de flujo en las posiciones que no tienen el laboratorio de tuberías experimental, como resultado se generaron 26,040 simulaciones de fugas. Los datos de flujo de entrada y salida (Q_{in} y Q_{out}) fueron guardados en un archivo .CVS y posteriormente con los flujos obtenidos, se calculó el número de Reynolds.

Segundo. Al obtener el conjunto de datos Caudales en las simulaciones se procede a calcular el número de Reynolds. En la Figura 4.9 se observa el resultado de una simulación donde el caudal es afectado al efectuar una fuga en el segundo 150, cambiando las mediciones de los caudales por el aumento del gasto volumétrico.

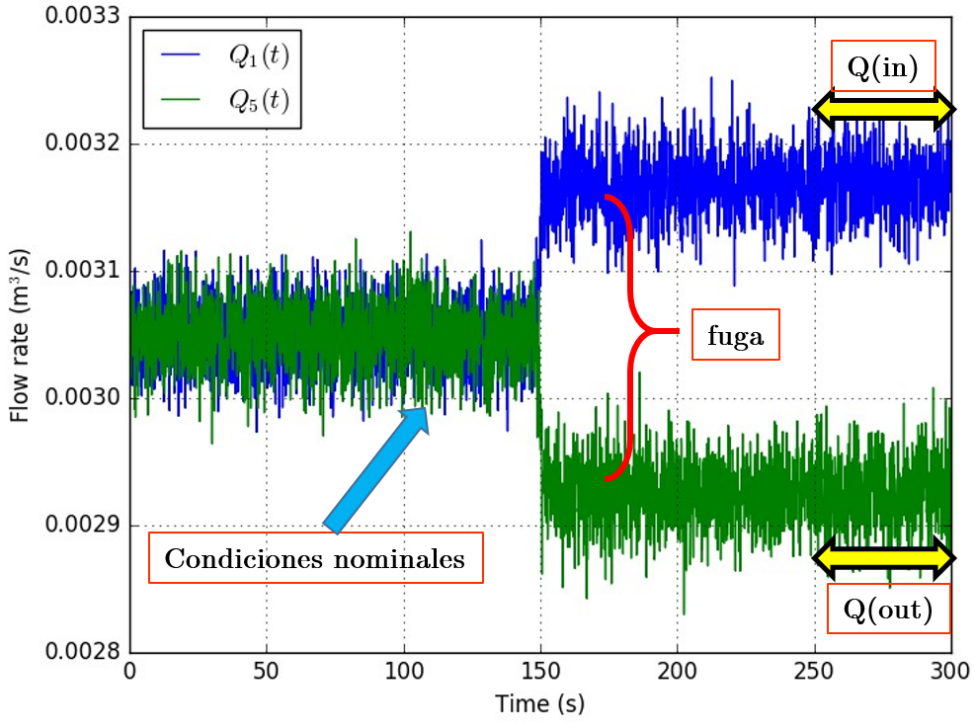


Figura 4.9: Gráfica de simulación de una fuga con caudales de entrada (azul) y salida (verde)

En cada una de las simulaciones (26,040), se extrae los Caudales Q_{in} y Q_{out} después de que ocurre la fuga, se toman un promedio en los últimos 50 segundos de la simulación, por lo cual, se obtiene un valor para Q_{in} y Q_{out} por cada simulación. Se obtiene un Archivo .CSV con los caudales Q_{in} y Q_{out} de las 26,040 simulaciones. En seguida, se calcula el número de Reynolds con el arreglo de 26,040 datos (Q_{in} y Q_{out}): Donde la viscosidad cinemática se

considera a 25 grados centígrados. Finalmente, se prepara un archivo .CSV, que contendrá los números de Reynolds (Re) calculados y se agregan los datos de Rugosidad relativa (ε/D). Los datos de (Re) y (ε/D) se procede hacer la escala logarítmica, quedando $log(Re)$ y $-log(\varepsilon/D)$ respectivamente. Estos dos valores, servirán como entradas para la RNA-1 que calcula las fricciones. En la Figura se aprecia el diagrama del procedimiento.

Tercero. Se usan los datos del archivo .CSV como entradas para la RNA que calculará las fricciones (f_{in} y f_{out}). Se agregan los datos de presiones a la entrada H_{in} y salida H_{out}, junto con los datos de posición de la pérdida (z), para tener el conjunto de 4 valores de datos de entrada y como salida serán posiciones de las fugas, que sera estimada por la RNA-2.

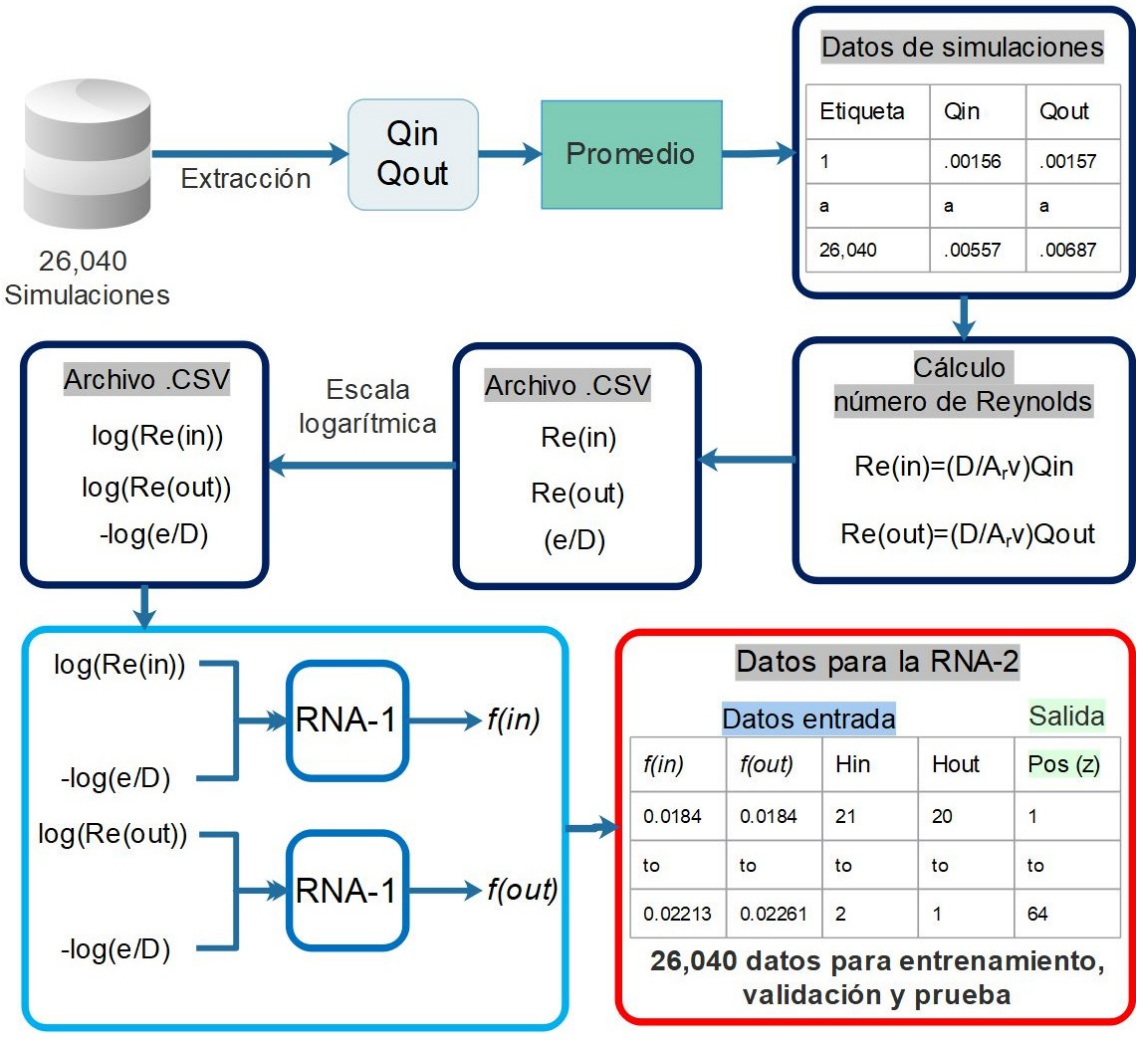


Figura 4.10: Diagrama del proceso para obtener el conjunto de datos

4.2.2. Arquitectura de la RNA-2

La estructura topológica de la RNA-2 es prealimentada hacia adelante en cascada (CFBPN), se ha seleccionado la arquitectura más adecuada, después de ensayos con diferentes configuraciones. No se dispone de reglas concretas para determinar el número de capas ocultas y el número de neuronas ocultas que debe tener una red para resolver un problema específico; el tamaño de las capas, tanto de entrada como de salida, suele estar determinado por la naturaleza de la aplicación (Hunter et al., 2012). Así, la solución para localizar fugas con la RNA-2 se emplea que las fricciones (f_{in} y f_{out}) junto con las presiones (H_{in} y H_{out}) sean el cuarteto de entradas aplicadas en la primera capa y la posición de fuga (z), que es la salida, sea considera en la última capa de la red.

El número de neuronas ocultas interviene en la eficiencia de aprendizaje y de generalización de la RNA-2; además, en general una sola capa oculta suele ser suficiente para la convergencia de la solución, sin embargo, existen ocasiones en que un problema es más sencillo de resolver con más de una capa oculta (Islam and Murase, 2001). Por lo tanto, el número óptimo de capas y neuronas ocultas se determina a través de la experimentación, quedando la siguiente configuración.

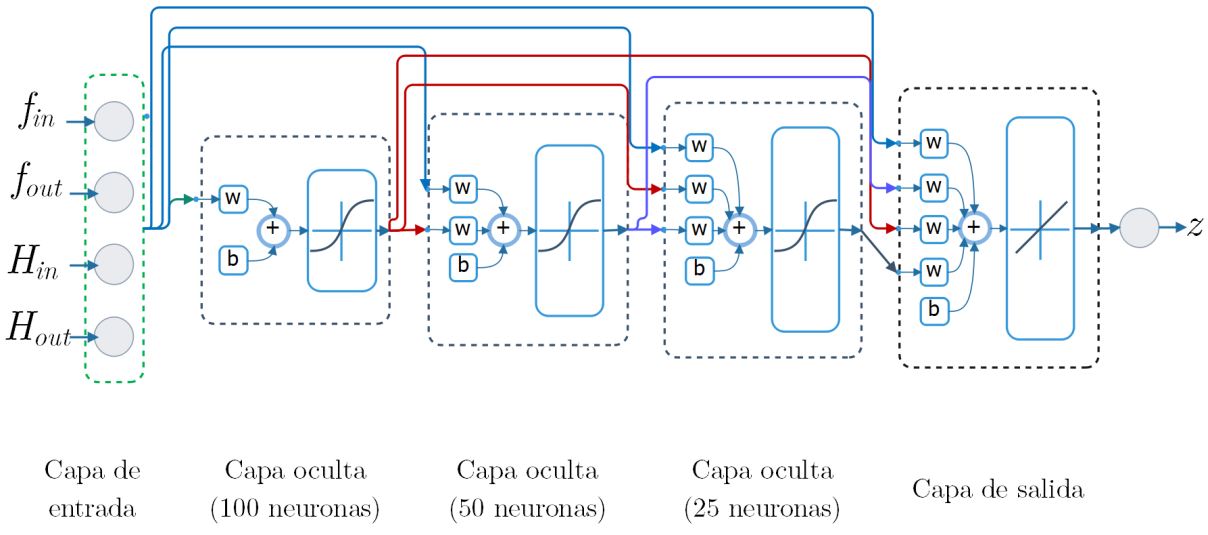


Figura 4.11: Arquitectura de la red de retro propagación en cascada hacia adelante.

La arquitectura de la red consta de cuatro capas en su estructura. La primera, la capa de

entrada tiene cuatro neuronas, las siguientes tres capas ocultas contienen, cien, cincuenta y veinticinco neuronas, respectivamente y una neurona en la capa de salida. La RNA-2 utilizada fue debidamente entrenada y validada de manera supervisada fuera de línea antes de la aplicación de la red. Como se puede ver en la Figura 4.11, la función tangente sigmoide hiperbólica (tansig) se utilizó como una función de activación en las capas ocultas de la RNA-2, posteriormente en la capa de salida, se utilizó la función lineal (purelin).

4.2.3. Entrenamiento de la RNA-2

La RNA-2 es de aprendizaje supervisado, lo que implica la uso de un entrenamiento moderado por un agente externo para que las entradas produzcan las salidas deseadas mediante el fortalecimiento de las conexiones. Una manera de llevar esto a cabo es a partir de la instauración de pesos sinápticos (Cruz and Herrera, 2011). Por tal razón, el conjunto de pares entradas-salidas se aplica a la RNA, es decir, ejemplos de entradas y sus salidas correspondientes (Ozuna and Rosemberg, 2002).

La red es entrenada con el algoritmo de retropropagación Gradiente Conjugado Escalado (Møller, 1993) que actualiza los valores de peso y sesgo, siendo fiable y eficaz en el entrenamiento de conjuntos de datos complejos (Offor and Alabi, 2016), (Meireles et al., 2003). Del conjunto de datos que se obtuvo en las simulaciones, se preparó en un archivo .CSV como ya se describió anteriormente. El conjunto de datos completamente preparado se dividió en subconjuntos de entrenamiento, validación y pruebas:

- Un conjunto de 70 %, 18,228 cuartetos de datos diseñados para entrenar la RNA a través de un procedimiento iterativo que actualiza el valor de pesos sinápticos y minimiza una función de error Charalambous (1992); Barnard (1992).

- Un conjunto de 15 %, 3,906 cuartetos de datos utilizados en un procedimiento llamado validación cruzada Zhang et al. (1999), que sirve para prevenir la aparición de sobreajuste Lawrence and Giles (2000). El entrenamiento se detiene si el error relativo a este conjunto de validación comienza a aumentar ("parada anticipada"). El procedimiento

se ejecuta simultáneamente con el entrenamiento, aunque no modifica el valor de los pesos sinápticos Prechelt (1998).

- La muestra de prueba 15 %, 3,906 cuartetos no tuvo ningún efecto en el entrenamiento y por lo tanto proporcionó una medida independiente del rendimiento de la RNA durante y después del entrenamiento.

El proceso de entrenamiento, durante cada iteración, al usar nuevos datos del conjunto de entrenamiento, el algoritmo de retropropagación permite que la salida generada de la red se compare con la salida deseada y se obtenga un error para cada una de las salidas. Al propagarse el error hacia atrás, desde la capa de salida hasta la capa de entrada, los pesos sinápticos de cada neurona se modifican para cada ejemplo, con el objeto de que la red converja hacia un estado que permita clasificar exitosamente todos los patrones de entrenamiento. Por esta razón, el aprendizaje de la RNA-2 se realiza por corrección de error. Al transcurrir el entrenamiento de la red, esta aprende a correlacionar distintas características del conjunto de entradas, luego del entrenamiento, un conjunto de datos de pruebas y validación den la capacidad de generalización, entendida como la facilidad de dar salidas satisfactorias a entradas no presentadas en la fase de entrenamiento (González and Hernando, 1995).

Se aplica la función de activación tansig en las capas ocultas y la función de activación purelin en la capa de salida. Estas funciones son comúnmente usadas al trabajar con el algoritmo de retropropagación. El aprendizaje de la RNA se detiene cuando el índice de error resulta aceptablemente pequeño para cada uno de los patrones aprendidos o cuando el número máximo de iteraciones del proceso ha sido alcanzado (Manning et al., 2014), (Yousefian and Kamalasadan, 2017). La función de rendimiento utilizada para entrenar la RNA es el error cuadrático medio (MSE), denotado por la Ecuación 4.3.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2 = \frac{1}{n}\sum_{i=1}^{n} (BD_i - RNA2_i)^2 \tag{4.3}$$

donde n indica el número de muestras del conjunto de datos de entrenamiento, e_i es el error de la red neuronal, y BD_i son los valores objetivo, mientras que $RNA2_i$ son valores de salida de red.

Capítulo 5

Resultados y discusión

La RNA-2 fue desarrollada en Matlab 2017b, a su vez dada la complejidad del modelo RNA y la cantidad del conjunto de datos, se decidió utilizar GPU con la plataforma CUDA desarrollada por NVIDIA, que permite el procesamiento paralelo de los cálculos numéricos, el Hardware que se utilizó fue una tarjeta NVIDIA GeForce GTX-750Ti de 2048 MB GDDR5 con 640 núcleos CUDA. En la Tabla 5.1 se describe los atributos con los que la RNA-2 fue entrenada.

Tabla 5.1: Atributos de la RNA-2

Atributo	Descripcion
Num. de Datos	26,040
Tipo de red	CFBPN
Variables de entrada	$fin; fout; Hin; Hout$
Variable de salida	posicion de fuga (z)
Algoritmo de entrenamiento	Gradiente Conjugado Escalado
Función de activación (capas ocultas)	tansig
Función de activación (capa de salida)	purelin
Función de rendimiento	MSE (predeterminado)
Iteraciones	13,340

El Error cuadrático medio (MSE) se usó como una medida de rendimiento para la fase de entrenamiento (Beale et al., 1992). El objetivo principal era minimizar la función de rendi-

miento. Los resultados del entrenamiento y validación con datos experimentales de la planta piloto se muestran a continuación.

5.1. Resultado del entrenamiento

El rendimiento de los conjuntos de datos de entrenamiento, pruebas y validación en comparación con la salida deseada es mostrado en la Figura 5.1. La muestra destinada a la validación es usada para medir el grado de generalización de la red, deteniendo el entrenamiento cuando este ya no mejora, esto evita el sobreajuste, entendido como un pobre rendimiento del modelo para predecir nuevos valores.

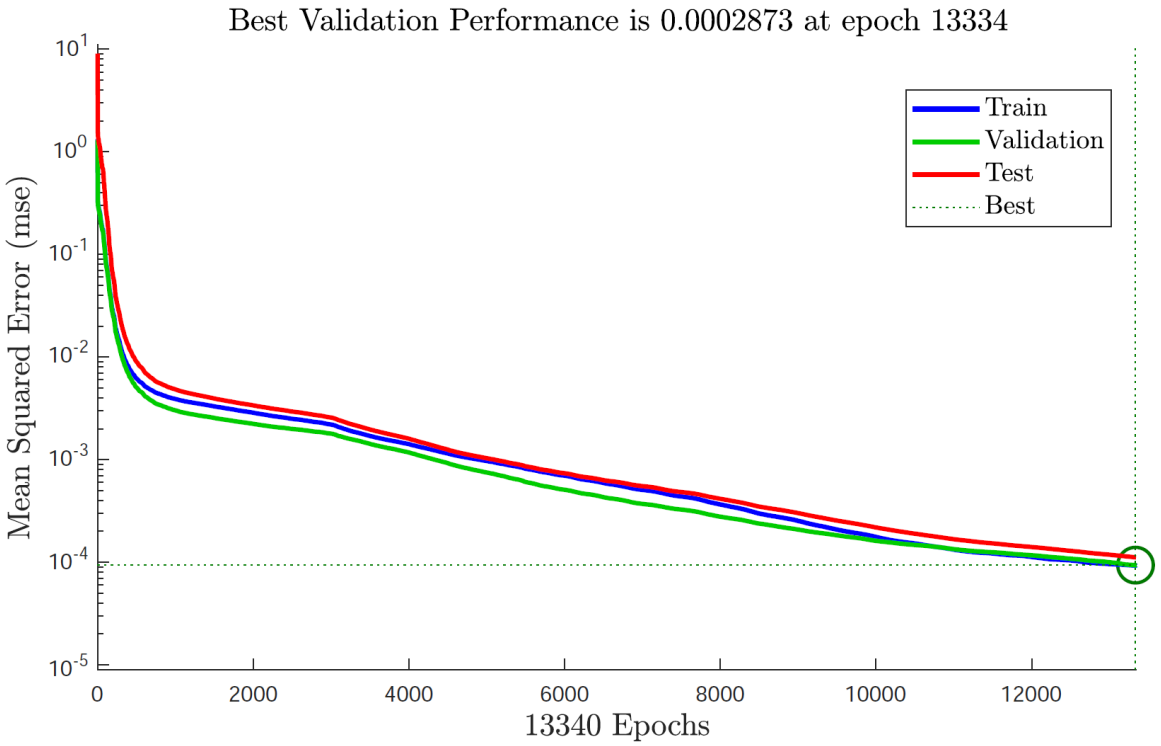


Figura 5.1: El error medio cuadrático (MSE) durante el proceso de capacitación de la RNA propuesta.

El entrenamiento de la estructura RNA propuesta se hizo hasta 13,340 épocas. El error medio cuadrado (MSE) de esta estructura RNA-2 se calculó en 10^{-4}, después de lo cual no hubo más tendencia a disminuir (Figura 5.1). Debido a que el valor MSE es muy pequeño, lo más cercano a cero, el modelo de RNA-2 es capaz de generalizar con gran precisión. A

su vez, se obtuvieron los mismos resultados con la prueba. Además, se probaron diferentes arquitecturas de RNAs para este trabajo, sin embargo, la configuración actual de RNA-2 resultó en el mejor rendimiento.

En la Figura 5.2 se observan los resultados del coeficiente de correlación de para la estructura de RNA-2 diseñada. La línea indica los valores esperados y los círculos negros representan los valores pronosticados. La predicción es eficiente y se constata un buen desempeño de la red, pues se obtiene un índice global de 0,99908 que indica una relación lineal fuerte entre los datos de posición de la fuga (z) y los otorgados por la RNA-2.

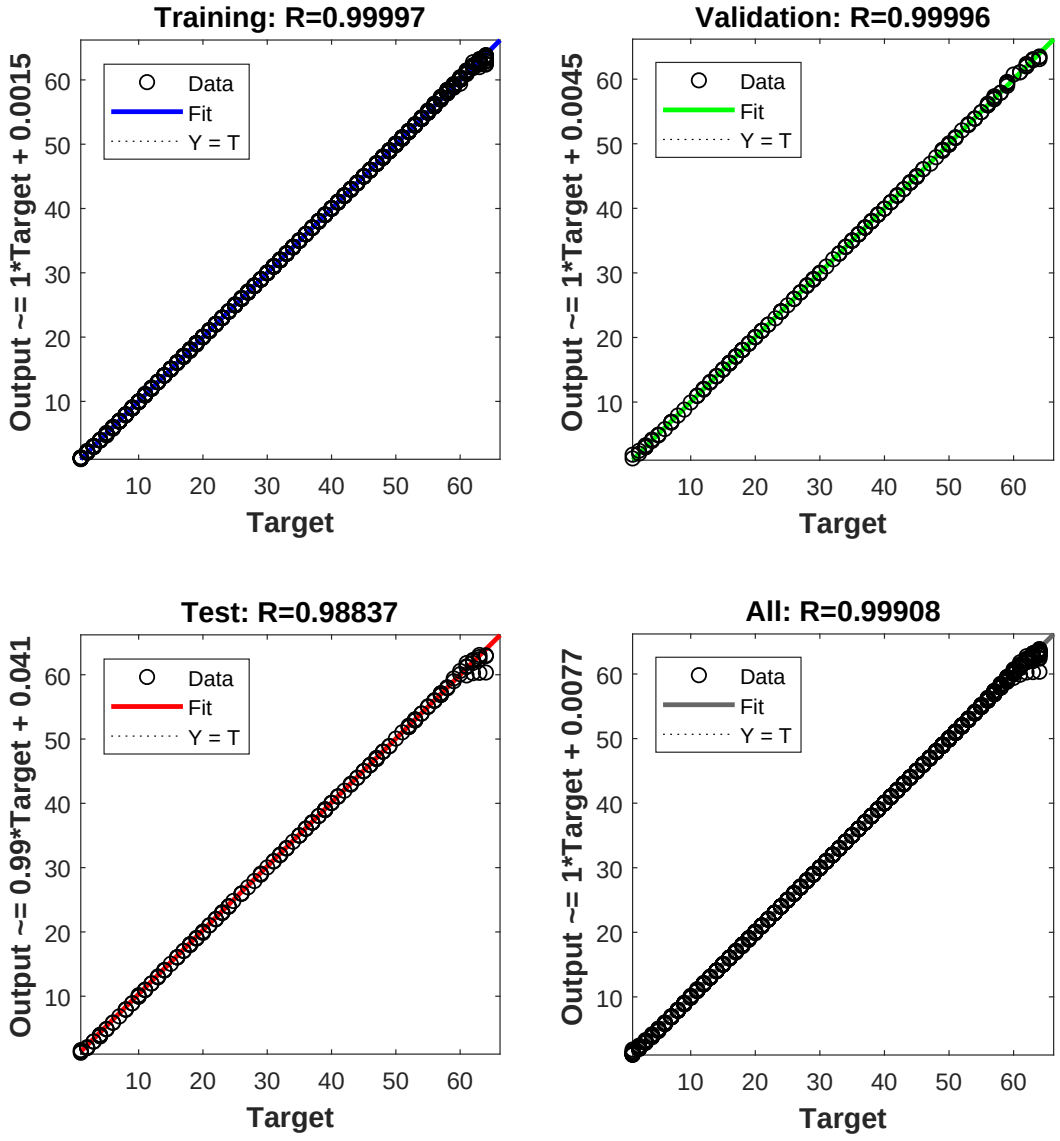


Figura 5.2: Correlación entre los valores esperados y pronosticados.

5.2. Validación de RNA-2 con datos experimentales

Una vez que la RNA-2 se ha entrenado y validado correctamente con el conjunto de datos de 26,040 cuartetos obtenidos con el simulador, procedemos a verificar cuál es la estimación de la posición de la fuga con los datos experimentales obtenidos de la planta piloto. Vale la pena mencionar que las posiciones de la fuga son conocidas anteriormente, las fugas están ubicadas en las siguientes posiciones: la primera fuga es a .91 [m], la segunda a 12.91 [m], la tercera a 26.84 [m] y la cuarta fuga está ubicada a 45.71 [m]. Entonces el desafío principal de la RNA-2 es localizar la fuga con el menor error posible.

Como se vio en el capitulo anterior en la seccion de diseños de experimentos se preparó un conjunto de datos experimentales con las cuatro fugas disponibles, repitiendo el experimento 4 veces por fuga. Las condiciones nominales de operación con las cuales se realizaron los experiementos fueron: presión de entrada (Hin) a 5.7087 mca y Presión de salida (Hout) en 1.9998 mca. Por consiguiente se procede hacer la evaluación de la RNA-2 con datos experimentales, las siguientes Tablas contienen los resultados obtenidos.

Tabla 5.2: Validación de la RNA con datos experimentales con la fuga (z_1)

Prueba	Fricción entrada (f_{in})	Fricción salida (f_{out})	Posición Real	Posición estimada por la RNA-2	Error porcentual
1	0.0200052	0.0202999	.91 [m]	0.9310 [m]	2.308 %
2	0.0200053	0.0203000	.91 [m]	0.9304 [m]	2.242 %
3	0.0200067	0.0202997	.91 [m]	0.9289 [m]	2.077 %
4	0.0200048	0.0202996	.91 [m]	0.9297 [m]	2.165 %
Promedio	0.0200055	0.0202995	.91 [m]	0.93 [m]	2.198 %

En la Tabla 5.2 se observa la estimación de la ubicación de la fuga (z_1) con datos experimentales, como resultado se obtuvo un error porcentual promedio de 2.198 %. La estimación fue buena, pues se debe considerar que la fuga estaba muy cerca de la bomba, por lo que

genera una gran cantidad de turbulencia. La localización de la fuga (z_1) con la RNA-2, presenta el mayor error de aproximación respecto a la siguientes fugas.

Tabla 5.3: Validación de la RNA con datos experimentales con la fuga (z_2)

Prueba	Fricción entrada (f_{in})	Fricción salida (f_{out})	Posición Real	Posición estimada por la RNA-2	Error porcentual
1	0.0200755	0.0203425	12.91 [m]	12.8952 [m]	0.115 %
2	0.0200753	0.0203486	12.91 [m]	12.8935 [m]	0.128 %
3	0.0200739	0.0203501	12.91 [m]	12.9011 [m]	0.069 %
4	0.0200689	0.0203644	12.91 [m]	12.8702 [m]	0.308 %
Promedio	0.0200734	0.0203514	12.91 [m]	12.89 [m]	0.155 %

En la evaluación de la RNA-2 con la fuga (z_2), se puede apreciar en la Tabla 5.3 una mejora en la localización, se observa la estimación de la ubicación de la fuga (z_2) con datos experimentales, con un rendimiento reflejado en el error porcentual promedio de 0.155 %. La estimación fue muy buena, el calculo se mantuvo por debajo del .5 % de error en las 4 pruebas realizadas. La localización de la fuga (z_2) con la RNA-2, mejoró respecto a la fuga (z_1).

Tabla 5.4: Validación de la RNA con datos experimentales con la fuga (z_3)

Prueba	Fricción entrada (f_{in})	Fricción salida (f_{out})	Posición Real	Posición estimada por la RNA-2	Error porcentual
1	0.0201468	0.0204002	26.84 [m]	26.8198 [m]	0.075 %
2	0.0201451	0.0204118	26.84 [m]	26.8206 [m]	0.072 %
3	0.0201478	0.0204017	26.84 [m]	26.8213 [m]	0.070 %
4	0.0201311	0.0204122	26.84 [m]	26.8183 [m]	0.081 %
Promedio	0.0201427	0.0204065	26.84 [m]	26.82 [m]	0.075 %

La evaluación de la RNA-2 la fuga (z_3), tuvo el mejor desempeño para estimar la ubicación

de la fuga. En la Tabla 5.4 se observa la estimación de la ubicación de la fuga (z_3), donde se obtuvo un error porcentual promedio de 0.075 %. La estimación fue la mejor en rendimiento, considerando que la fuga se encuentra donde la turbulencia es menor. La localización de la fuga (z_1) con la RNA-2, presenta el menor error de aproximación respecto a las demás fugas.

Tabla 5.5: Validación de la RNA con datos experimentales con la fuga (z_4)

Prueba	Fricción entrada (f_{in})	Fricción salida (f_{out})	Posición Real	Posición estimada por la RNA-2	Error porcentual
1	0.0202211	0.0205099	45.71 [m]	45.7461 [m]	0.079 %
2	0.0202165	0.0205161	45.71 [m]	45.7578 [m]	0.105 %
3	0.0202155	0.0204899	45.71 [m]	45.7615 [m]	0.113 %
4	0.0202309	0.0205121	45.71 [m]	45.7347 [m]	0.054 %
Promedio	0.0202210	0.0205070	45.71 [m]	45.75 [m]	0.088 %

Por ultimo, en la Tabla 5.3 se observa la evaluación de la RNA-2 con la fuga (z_4), se puede apreciar una merma respecto a la fuga (z_3), la estimación de la ubicación de la fuga (z_4) , obtuvo un error porcentual promedio de 0.155 %. La estimación fue muy buena, considerando que la fuga se encuentra en parte final de la tubería y el caudal de entrada es menor, sin embargo, la RNA-2 estimó la ubicación de la fuga en las cuatro pruebas con un error muy pequeño.

Estas evaluaciones fueron realizadas para validar el rendimiento de la RNA-2, recordando que la red fue entrenada previamente con datos del simulador de la planta piloto, la RNA-2 tuvo un desempeño óptimo; a pesar de que las mediciones contienen ruido de los sensores. Durante la validación de la RNA-2, se verifico que de manera general los resultados fueron muy prometedores, donde el mejor rendimiento se obtuvo con las pruebas de la fuga (z_3), mientras que las pruebas de la fuga (z_1) el rendimiento resulto con el mayor error en la estimación.

En la Tabla 5.6 se presente una comparación del con el método presentado por Santos-Ruiz et al. (2018), donde un método de detección y localización de fugas basado en modelos fue

desarrollado. Ambos métodos son evaluados con datos experimentales de la planta piloto del TecNM-ITTG, pero mostrando enfoques de análisis diferentes, demostrando que es factible realizar diferentes técnicas para resolver la problemática de detección y localización de fugas.

Tabla 5.6: Comparación de diferentes métodos para localización de fugas hidráulicas

Posición Real	Número de pruebas	Posición estimada por la RNA-2	Error porcentual	Error porcentual en (Santos-Ruiz et al., 2018)
(z_1)=.91 [m]	4	0.93 [m]	2.198 %	8.5 %
(z_2)=12.91 [m]	4	12.89 [m]	0.155 %	1.3 %
(z_3)=26.84 [m]	4	26.82 [m]	0.075 %	1.8 %
(z_4)=45.75 [m]	4	45.75 [m]	0.88 %	0.079 %

Los resultados muestran la efectividad de la RNA-2, donde el error porcentual promedio es de .629 %, se observa que la validación con mayor error fue para l fuga (z_1), debido a que se encuentra muy cerca de la bomba, el cual genera mucha turbulencia y altera fuertemente en la dinámica del sistema, sin embargo, la RNA-2 tuvo mejor rendimiento de estimación respecto al método con Filtro de Kalman extendido. Finalmente se procede a realizar la prueba de hipótesis.

5.3. Prueba de hipótesis

Se utiliza la prueba t de student, para comprobar si la hipótesis nula se rechaza o se acepta. Se aplica el método del valor crítico, se utiliza como punto de referencia para determinar si el valor del estadístico de prueba es lo suficientemente pequeño para rechazar la hipótesis nula (Anderson et al., 2001). El conjunto de datos esta compuesto de 16 muestras de error porcentual, obtenidos de los resultado de las evaluaciones de la RNA-2 para localizar cada una de las fugas. Con base en las hipótesis planteadas al inicio del trabajo, se realiza el siguiente análisis:

- Hipótesis nula: $H_0 : \mu \geq 1\,\%$
- Hipótesis alternativa: $H_1 : \mu < 1\,\%$

Tabla 5.7: Datos de prueba de hipótesis

Dato	Resultado
Error porcentual promedio $\bar{x}$	0.629 %
Número de muestras n	16
Nivel de significación α	5 %
Grados de libertad $G_l = n - 1$	15
Valor critico t	-1.7531
Desviación estandar de los porcentajes de error σ	0.9385
Estadístico de prueba $t_p = \frac{\bar{x}-\mu}{\frac{\sigma}{\sqrt{n}}}$	-1.5812

En la tabla 5.7 están los datos para realizar la prueba de hipótesis, donde el estadístico de prueba es de $t_p = -1.581$, se encuentra dentro de la región de aceptación de H_0 respecto al valor critico de $t = -1.7531$. La Figura 5.3 se observa el diagrama de distribución normal de las

muestras, con la región de aceptación de H_0 y la prueba a una cola (izquierda) en color verde que indica la zona de rechazo de H_0.

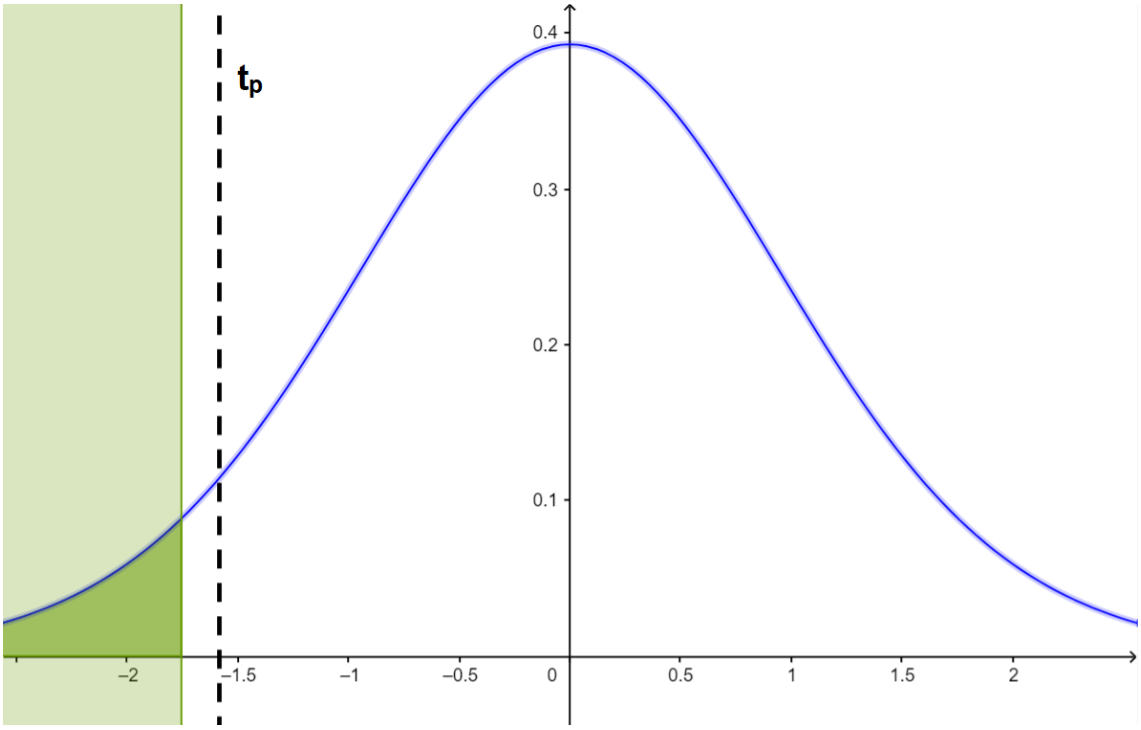


Figura 5.3: Gráfica de aceptación de H_o.

Por lo tanto, el resultado de la prueba de hipótesis establece que el valor de confianza se encuentra dentro de la región de aceptación de H_0. Se demuestra estadísticamente que no es admisible rechazar H_0 y se acepta la hipótesis nula H_0. No se cumple que el error porcentual promedio sea menor al 1 %, en la localización de fuga para todas las fugas, principalmente en fugas muy cercanas a la bomba.

Capítulo 6

Conclusiones

En este trabajo se obtuvieron resultados favorables, dando una nueva perspectiva de ubicación de fugas en tuberías al utilizar RNAs, con una importante contribución en la localización de fugas en condiciones no tan favorables, ya que los parámetros en los modelos matemáticos determinísticos se consideran ideales, sin embargo, cuando se consideran datos experimentales estos contienen ruido y otros factores con incertidumbre.

Una aportación importante a este trabajo fue el trabajar con un simulador de la planta piloto del TecNM-ITTG, que fue desarrollada de manera conjunta gracias al trabajo de compañeros del grupo Turix-Dynamics, donde participamos alumnos de Licenciatura, Maestria y Doctorado, de manera tal que se obtuvo un simulador validado para hacer simulaciones y generar los datos para entrenamiento de la RNA-2 para la localización de fugas.

De forma que al trabajar con datos experienciales y datos de simulación, se obtiene una gran cantidad de datos para poder entrenar y validar la RNA-2, capaz de adaptarse y ser robusta a las incertidumbres paramétricas del sistema. La RNA-2 con una arquitectura CFBPN, se obtuvo un predictor robusto para localización de fugas. Validando la RNA-2 con datos experimentales dio como resultado la estimación de las fugas con un error porcentual promedio de 0.629 %.

De acuerdo al diseño de experimentos propuestos para obtener la base de datos, se obtuvo un conjunto de datos híbrido, con datos obtenidos de mediciones de la tuberia y datos de simulación. Como resultado después de realizar el entrenamiento, validación y pruebas, se

obtuvo una RNA-2 eficiente, ya que obtuvo un rendimiento en la localización de la fuga menor al 1 %, en la mayoría de las pruebas. En la fuga 1, se obtuvo un error de porcentual del 2 %, debido a que la turbulencia generada por la bomba es mayor, afectando directamente a las mediciones de caudal. Por lo tanto, la hipótesis nula H_0 es aceptada.

La metodología tiene potencial para ser usado como una alternativa de localización de fugas hidráulicas dentro de un sistema de diagnostico y supervisión de tuberías. Se observó una mejora en la localización de fugas respecto al Filtro de Kalman Extendido (EKF) realizado anteriormente (Santos-Ruiz et al., 2018).

Para trabajos futuros se plantea la utilización de RNAs para detección de múltiples fugas simultaneas, donde se deben considerar el comportamiento del sistema cuando se presentas fugas simultaneas y adquirir un conjunto de datos suficientemente representativos de escenarios que pudieran presentarse al tener fallas simultaneas. Además, es pertinente mencionar que es viable usar las RNAs para diagnóstico y supervisión en redes de ductos hidráulicos, considerando que tiene potencial adaptable, pero es necesario que primero se tenga un modelo donde se puedan adquirir datos confiables de manera que se obtenga suficiente conocimiento sobre el comportamiento de los fluidos en Redes de ductos.

Durante el desarrollo de este trabajo de investigación, se obtuvo un método de localización de fugas, cumpliendo con los objetivos planteados.

Bibliografía

Abdulla, M. B. and Herzallah, R. (2015). Probabilistic multiple model neural network based leak detection system: Experimental study. *Journal of Loss Prevention in the Process Industries*, 36:30–38.

Afzal, M. and Udpa, S. (2002). Advanced signal processing of magnetic flux leakage data obtained from seamless gas pipeline. *Ndt & E International*, 35(7):449–457.

Afzal, N., Seena, A., and Bushra, A. (2013). Turbulent flow in a machine honed rough pipe for large reynolds numbers: general roughness scaling laws. *Journal of hydro-environment research*, 7(1):81–90.

Amari, S.-i., Murata, N., Muller, K.-R., Finke, M., and Yang, H. H. (1997). Asymptotic statistical theory of overtraining and cross-validation. *IEEE Transactions on Neural Networks*, 8(5):985–996.

Anderson, D. R., Sweeney, D. J., Williams, T. A., Roa, M. d. C. H., and Álvarez, T. L. (2001). *Estadística para administración y economía*. Number 311 in 1. International Thomson.

Andrews, R., Diederich, J., and Tickle, A. B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, 8(6):373–389.

Arifin, B., Li, Z., and Shah, S. L. (2015). Pipeline leak detection using particle filters. *IFAC-PapersOnLine*, 48(8):76–81.

Asefa, T., Kemblowski, M., Urroz, G., and McKee, M. (2005). Support vector machines (svms) for monitoring network design. *Groundwater*, 43(3):413–422.

Barnard, E. (1992). Optimization for training neural nets. *IEEE Transactions on Neural Networks*, 3(2):232–240.

Basheer, I. A. and Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43(1):3–31.

Beale, M. H., Hagan, M. T., and Demuth, H. B. (1992). Neural network toolbox™ user's guide. *The Mathworks Inc.*

Bermúdez, J., Santos-Ruiz, I., López-Estrada, F., Torres, L., and Puig, V. (2017). Diseño y modelado dinámico de una planta piloto para detección de fugas hidráulicas. *Asociación de México de Control Automático.*

Bertsekas, D. P. and Tsitsiklis, J. N. (1995). Neuro-dynamic programming: an overview. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 1, pages 560–564. IEEE Publ. Piscataway, NJ.

Besançon, G., Georges, D., Begovich, O., Verde, C., and Aldana, C. (2007). Direct observer design for leak detection and estimation in pipelines. In *Control Conference (ECC), 2007 European*, pages 5666–5670. IEEE.

Billmann, L. and Isermann, R. (1987). Leak detection methods for pipelines. *Automatica*, 23(3):381–385.

Bishop, C. M. et al. (1995). *Neural networks for pattern recognition*. Oxford university press.

Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., and Schröder, J. (2006). *Diagnosis and fault-tolerant control*, volume 2. Springer.

Blum, A. (1992). Neural networks in c++. *NY: Wiley*, 697.

Brkić, D. (2011). Review of explicit approximations to the colebrook relation for flow friction. *Journal of Petroleum Science and Engineering*, 77(1):34–48.

Brkić, D. and Ćojbašić, Ž. (2016). Intelligent flow friction estimation. *Computational intelligence and neuroscience*, 2016.

Brkić, D. (2014). Discussion of "gene expression programming analysis of implicit colebrook–white equation in turbulent flow friction factor calculation" by saeed samadianfard [j. pet. sci. eng. 92–93 (2012) 48–55]. *Journal of Petroleum Science and Engineering*, 124:399 – 401.

Caputo, A. C. and Pelagagge, P. M. (2002). An inverse approach for piping networks monitoring. *Journal of Loss Prevention in the Process Industries*, 15(6):497 – 505.

Charalambous, C. (1992). Conjugate gradient algorithm for efficient training of artificial neural networks. *IEE Proceedings G (Circuits, Devices and Systems)*, 139(3):301–310.

Chaudhry, M. H. (1979). Applied hydraulic transients. Technical report, Springer.

Chen, H., Ye, H., Chen, L., and Su, H. (2004). Application of support vector machine learning to leak detection and location in pipelines. In *Instrumentation and Measurement Technology Conference, 2004. IMTC 04. Proceedings of the 21st IEEE*, volume 3, pages 2273–2277. IEEE.

Churchill, S. W. (1973). Empirical expressions for the shear stress in turbulent flow in commercial pipe. *AIChE Journal*, 19(2):375–376.

Colebrook, C. and White, C. (1937). Experiments with fluid friction in roughened pipes. *Proc. R. Soc. Lond. A*, 161(906):367–381.

Colebrook, C. F., Blench, T., Chatley, H., Essex, E., Finniecome, J., Lacey, G., Williamson, J., and Macdonald, G. (1939). Correspondence. turbulent flow in pipes, with particular reference to the transition region between the smooth and rough pipe laws.(includes plates). *Journal of the Institution of Civil engineers*, 12(8):393–422.

Conejero, J. A., Lizama, C., and Rodenas, F. (2016). Dynamics of the solutions of the water hammer equations. *Topology and its Applications*, 203:67 – 83. The Iberoamerican Conference on Topology and its Applications 2014.

Cruz, P. P. and Herrera, A. (2011). *Inteligencia artificial con aplicaciones a la ingeniería*, volume 1. Marcombo.

da Silva, H. V., Morooka, C. K., Guilherme, I. R., da Fonseca, T. C., and Mendes, J. R. (2005). Leak detection in petroleum pipelines using a fuzzy system. *Journal of Petroleum Science and Engineering*, 49(3):223 – 238. An Introduction to Artificial Intelligence Applications in Petroleum Exploration and Production.

Darken, C. and Moody, J. (1992). Towards faster stochastic gradient search. In *Advances in neural information processing systems*, pages 1009–1016.

Delgado-Aguiñaga, J., Besancon, G., Begovich, O., and Carvajal, J. (2016). Multi-leak diagnosis in pipelines based on extended kalman filter. *Control Engineering Practice*, 49:139–148.

Ding, S., Zhang, P., Ding, E., Naik, A., Deng, P., and Gui, W. (2010). On the application of pca technique to fault diagnosis. *Tsinghua Science and Technology*, 15(2):138–144.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. CRC press.

Gao, Z., Cecati, C., and Ding, S. X. (2015). A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6):3757–3767.

Garcia, E. A. and Frank, P. (1997). Deterministic nonlinear observer-based approaches to fault diagnosis: a survey. *Control Engineering Practice*, 5(5):663–670.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58.

Gertler, J. (1998). Fault detection and diagnosis in engineering systems. *CRC, Virginia, USA*.

Giustolisi, O., Berardi, L., and Walski, T. (2011). Some explicit formulations of colebrook–white friction factor considering accuracy vs. computational speed. *Journal of Hydroinformatics*, 13(3):401–418.

Glen, N., Ryan, J., and Findlay, T. (2006). A review of pipeline integrity systems. In *Proceedings of 24th international north sea flow measurement workshop paper*, volume 3, pages 24–27.

Goldberg, D. E. (1987). Computer-aided pipeline operation using genetic algorithms and rule learning. part ii: Rule learning control of a pipeline under normal and abnormal conditions. *Engineering with Computers*, 3(1):47–58.

González, J. and Hernando, V. (1995). *Redes neuronales artificiales: fundamentos, modelos y aplicaciones*. Paradigma : fundamentos teóricos. Informática. RA-MA.

Haaland, S. E. (1983). Simple and explicit formulas for the friction factor in turbulent pipe flow. *Journal of Fluids Engineering*, 105(1):89–90.

Hamayun, M. T., Edwards, C., Alwi, H., et al. (2016). *Fault tolerant control schemes using integral sliding modes*. Springer.

Hassoun, M. H. (1995). *Fundamentals of artificial neural networks*. MIT press.

Hedayat, A., Davilu, H., Barfrosh, A. A., and Sepanloo, K. (2009). Estimation of research reactor core parameters using cascade feed forward artificial neural networks. *Progress in Nuclear Energy*, 51(6):709 – 718.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Addison-Wesley/Addison Wesley Longman.

Hirose, A. (2012). *Complex-valued neural networks*, volume 400. Springer Science & Business Media.

Hjorth, J. (1994). Computer intensive statistical methods chapman and hall.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.

Hornik, K. (1993). Some new results on neural network approximation. *Neural networks*, 6(8):1069–1072.

Huang, S., Tan, K. K., and Lee, T. H. (2012). Fault diagnosis and fault-tolerant control in linear drives using the kalman filter. *IEEE Transactions on Industrial Electronics*, 59(11):4285–4292.

Hunter, D., Yu, H., Pukish III, M. S., Kolbusz, J., and Wilamowski, B. M. (2012). Selection of proper neural network sizes and architectures—a comparative study. *IEEE Transactions on Industrial Informatics*, 8(2):228–240.

Isasi Viñuela, P. and Galván León, I. (2004). Redes de neuronas artificiales. *Un Enfoque Práctico, Editorial Pearson Educación SA Madrid España*.

Isermann, R. (1984). Process fault detection based on modeling and estimation methods—a survey. *automatica*, 20(4):387–404.

Islam, M. M. and Murase, K. (2001). A new algorithm to design compact two-hidden-layer artificial neural networks. *Neural Networks*, 14(9):1265–1278.

Jain, A. K., Mao, J., and Mohiuddin, K. M. (1996). Artificial neural networks: a tutorial. *Computer*, 29(3):31–44.

Kang, J., Park, Y.-J., Lee, J., Wang, S.-H., and Eom, D.-S. (2018). Novel leakage detection by ensemble cnn-svm and graph-based localization in water distribution systems. *IEEE Transactions on Industrial Electronics*, 65(5):4279–4289.

Khodayari-Rostamabad, A., Reilly, J. P., Nikolova, N. K., Hare, J. R., and Pasha, S. (2009). Machine learning techniques for the analysis of magnetic flux leakage images in pipeline inspection. *IEEE Transactions on magnetics*, 45(8):3073–3084.

Kurková, V. (1992). Kolmogorovs theorem and multilayer neural networks. *Neural networks*, 5(3):501–506.

Lashkarbolooki, M., Shafipour, Z. S., and Hezave, A. Z. (2013a). Trainable cascade-forward back-propagation network modeling of spearmint oil extraction in a packed bed using sc-co2. *The Journal of Supercritical Fluids*, 73:108 – 115.

Lashkarbolooki, M., Vaferi, B., Shariati, A., and Hezave, A. Z. (2013b). Investigating vapor–liquid equilibria of binary mixtures containing supercritical or near-critical carbon dioxide and a cyclic compound using cascade neural network. *Fluid Phase Equilibria*, 343:24 – 29.

Lawrence, S. and Giles, C. L. (2000). Overfitting and neural networks: conjugate gradient and backpropagation. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 1, pages 114–119 vol.1.

Leu, S.-S. and Bui, Q.-N. (2016). Leak prediction model for water distribution networks created using a bayesian network learning approach. *Water resources management*, 30(8):2719–2733.

Li, S., Song, Y., and Zhou, G. (2018). Leak detection of water distribution pipeline subject to failure of socket joint based on acoustic emission and pattern recognition. *Measurement*, 115:39–44.

Liang, W. and Zhang, L. (2012). A wave change analysis (wca) method for pipeline leak detection using gaussian mixture model. *Journal of Loss Prevention in the Process Industries*, 25(1):60–69.

Linaric, D. and Koroman, V. (2003). Fault diagnosis of a hydraulic actuator using neural network. In *IEEE International Conference on Industrial Technology, 2003*, volume 1, pages 108–111 Vol.1.

Lohninger, H. (1993). Evaluation of neural networks based on radial basis functions and their application to the prediction of boiling points from structural parameters. *Journal of Chemical Information and Computer Sciences*, 33(5):736–744.

Lopez-Estrada, F.-R., Ponsart, J.-C., Theilliol, D., Astorga-Zaragoza, C., and Aberkane, S. (2014). Fault diagnosis based on robust observer for descriptor-lpv systems with unmeasurable scheduling functions. *IFAC Proceedings Volumes*, 47(3):1079–1084.

Lowe, D. and Webb, A. R. (1990). Exploiting prior knowledge in network optimization: an illustration from medical prognosis. *Network: Computation in Neural Systems*, 1(3):299–323.

Maki, Y. and Loparo, K. A. (1997). A neural-network approach to fault detection and diagnosis in industrial processes. *IEEE Transactions on Control Systems Technology*, 5(6):529–541.

Mandal, S. K., Chan, F. T., and Tiwari, M. (2012). Leak detection of pipeline: An integrated approach of rough set theory and artificial bee colony trained svm. *Expert Systems with Applications*, 39(3):3071–3080.

Manning, T., Sleator, R. D., and Walsh, P. (2014). Biologically inspired intelligent decision making: a commentary on the use of artificial neural networks in bioinformatics. *Bioengineered*, 5(2):80–95.

Mashford, J., Silva, D. D., Marney, D., and Burn, S. (2009). An approach to leak detection in pipe networks using analysis of monitored pressure values by support vector machine. In *2009 Third International Conference on Network and System Security*, pages 534–539.

Masters, T. (1993). *Practical neural network recipes in C*. Morgan Kaufmann.

Maurya, M. R., Rengaswamy, R., and Venkatasubramanian, V. (2005). Fault diagnosis by qualitative trend analysis of the principal components. *Chemical Engineering Research and Design*, 83(9):1122–1132.

Maurya, M. R., Rengaswamy, R., and Venkatasubramanian, V. (2007). Fault diagnosis using dynamic trend analysis: A review and recent developments. *Engineering Applications of artificial intelligence*, 20(2):133–146.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Meireles, M. R., Almeida, P. E., and Simões, M. G. (2003). A comprehensive review for industrial applicability of artificial neural networks. *IEEE transactions on industrial electronics*, 50(3):585–601.

Meng, L., Yuxing, L., Wuchang, W., and Juntao, F. (2012). Experimental study on leak detection and location for gas pipeline based on acoustic method. *Journal of Loss Prevention in the Process Industries*, 25(1):90–102.

Minsky, M. and Papert, S. (1969). The perceptron: Principles of computational geometry. *MIT press. McCulloch, WS and Pitts, W.,(1943) A Logical Calculus of the Ideas Imminent in Nervous Activity, Bulletin of Mathematical Biophysics*, 5:115–133.

Moody, L. F. (1944). Friction factors for pipe flow. *Trans. Asme*, 66:671–684.

Mostafapour, A. and Davoudi, S. (2013). Analysis of leakage in high pressure pipe using acoustic emission method. *Applied Acoustics*, 74(3):335–342.

Mounce, S., Day, A., Wood, A., Khan, A., Widdop, P., and Machell, J. (2002). A neural network approach to burst detection. *Water Science and Technology*, 45(4-5):237–246.

Murvay, P.-S. and Silea, I. (2012). A survey on gas leak detection and localization techniques. *Journal of Loss Prevention in the Process Industries*, 25(6):966–973.

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525 – 533.

Ocampo-Martinez, C., Puig, V., Cembrano, G., and Quevedo, J. (2013). Application of predictive control strategies to the management of complex networks in the urban water cycle [applications of control]. *IEEE Control Systems*, 33(1):15–41.

OECD (2016). Water Governance in Cities. *OECD Studies on Water*.

Offor, U. H. and Alabi, S. B. (2016). Artificial neural network model for friction factor prediction. *Journal of Materials Science and Chemical Engineering*, 4(07):77.

Olden, J. D. and Jackson, D. A. (2002). Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological modelling*, 154(1-2):135–150.

Olivera-Villaseñor, R. E. and Rodríguez-Castellanos, A. (2012). Estudio del riesgo en ductos de transporte de gasolinas y diesel en méxico. *Científica*, 16(4).

ONU-Hábitat, U. N. H. S. P. (2010). The right to water, fact sheet no. 35. . United Nations, Office of the High Commissioner for Human Rights (OHCHR), World Health Organization (WHO).

Ozuna, C. and Rosemberg, J. (2002). *Aplicación de redes neuronales en la discriminación entre fallas y oscilaciones de potencia.* PhD thesis, Universidad Autónoma de Nuevo León.

Pazold, M., Burhenne, S., Radon, J., Herkel, S., and Antretter, F. (2012). Integration of modelica models into an existing simulation software using fmi for co-simulation. In *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany*, number 76 in Linköping Electronic Conference Proceedings, pages 949–954. Linköping University Electronic Press; Linköpings universitet.

Pérez, R., Puig, V., Pascual, J., Quevedo, J., Landeros, E., and Peralta, A. (2011). Methodology for leakage isolation using pressure sensitivity analysis in water distribution networks. *Control Engineering Practice*, 19(10):1157–1167.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761 – 767.

Puig, V., Quevedo, J., Escobet, T., Morcego, B., and Ocampo, C. (2010). Control tolerante a fallos (parte i): Fundamentos y diagnóstico de fallos. *Revista Iberoamericana de automática e informática industrial*, 1(1):15–31.

Pwasong, A. and Sathasivam, S. (2016). A new hybrid quadratic regression and cascade forward backpropagation neural network. *Neurocomputing*, 182:197 – 209.

Reddy, H. P., Narasimhan, S., Bhallamudi, S. M., and Bairagi, S. (2011). Leak detection in gas pipeline networks using an efficient state estimator. part-i: Theory and simulations. *Computers & chemical engineering*, 35(4):651–661.

Ripley, B. D. (1996). Pattern recognition via neural networks. *a volume of Oxford Graduate Lectures on Neural Networks, title to be decided. Oxford University Press.[See http://www. stats. ox. ac. uk/ripley/papers. html.]*.

Rojas, J., Verde, C., Torres, L., and Perez, E. (2018). On-line head loss identication for monitoring of pipelines. *10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*.

Rostek, K., Morytko, Ł., and Jankowska, A. (2015). Early detection and prediction of leaks in fluidized-bed boilers using artificial neural networks. *Energy*, 89:914–923.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Russell, E. L., Chiang, L. H., and Braatz, R. D. (2012). *Data-driven methods for fault detection and diagnosis in chemical processes*. Springer Science & Business Media.

Salvatore, B., Paolo, L., Paolo, A., and Sanjoy, B. (2004). Leak detection in liquefied gas pipelines by artificial neural networks. *AIChE Journal*, 44(12):2675–2688.

Samarasinghe, S. (2016). *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. Auerbach publications.

Santos, R. B., Rupp, M., Bonzi, S., Fileti, A., et al. (2013). Comparison between multilayer feedforward neural networks and a radial basis function network to detect and locate leaks in pipelines transporting gas. *Chem. Eng. Trans*, 32(1375):e1380.

Santos-Ruiz, I., Bermudez, J., Lopez-Estrada, F., Puig, V., Torres, L., and Delgado-Aguinaga, J. (2018). Online leak diagnosis in pipelines using an ekf-based and steady-state mixed approach. *Control Engineering Practice*, 81:55–64.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Serghides, T. (1984). Estimate friction factor accurately. *Chemical Engineering*, 91(5):63–64.

Sharma, A., Sandooja, B., and Yadav, D. (2013). Extreme machine learning: Feed forward networks. *International Journal*, 3(8).

Smith, M. (1993). *Neural networks for statistical modeling*. Thomson Learning.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, pages 111–147.

Streeter, V. L., Wylie, E. B., Bedford, K. W., Saldarriaga, J. G., et al. (1988). *Mecánica de los fluidos*, volume 9. McGraw-Hill.

Sun, L. and Chang, N. (2014). Integrated-signal-based leak location method for liquid pipelines. *Journal of Loss Prevention in the Process Industries*, 32:311–318.

Swanee, P. and Jain, A. K. (1976). Explicit equations for pipeflow problems. *Journal of the hydraulics division*, 102(5).

Torres, L., Verde, C., Carrera, R., and Cayetano, R. (2014). Algoritmos de diagnóstico para fallas en ductos. *Tecnología y ciencias del agua*, 5(4):57–78.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., and Yin, K. (2003). A review of process fault detection and diagnosis: Part iii: Process history based methods. *Computers & chemical engineering*, 27(3):327–346.

Verde, C., Molina, L., and Torres, L. (2014). Parameterized transient model of a pipeline for multiple leaks location. *Journal of Loss Prevention in the Process Industries*, 29:177 – 185.

Verde, C. and Rojas, J. (2017). Recursive scheme for sequential leaks' identification. In *Modeling and Monitoring of Pipelines and Networks*, pages 125–145. Springer.

Verde, C., Torres, L., and González, O. (2016). Decentralized scheme for leaks' location in a branched pipeline. *Journal of Loss Prevention in the Process Industries*, 43:18 – 28.

Weiss, S. M. and Kulikowski, C. A. (1991). *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers Inc.

Wieland, A. and Leighton, R. (1987). Geometric analysis of neural network capabilities. In *Proceedings of the Second IEEE International Conference on Nerual Networks*, volume 3, pages 385–392.

Winning, H. K. and Coole, T. (2013). Explicit friction factor accuracy and computational efficiency for turbulent flow in pipes. *Flow, turbulence and combustion*, 90(1):1–27.

Wu, W., Walczak, B., Massart, D., Heuerding, S., Erni, F., Last, I., and Prebble, K. (1996). Artificial neural networks in classification of nir spectral data: design of the training set. *Chemometrics and intelligent laboratory systems*, 33(1):35–46.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447.

Yousefian, R. and Kamalasadan, S. (2017). A review of neural network based machine learning approaches for rotor angle stability control. *arXiv preprint arXiv:1701.01214*.

Yıldırım, G. (2009). Computer-based analysis of explicit approximations to the implicit colebrook–white equation in turbulent flow friction factor calculation. *Advances in Engineering Software*, 40(11):1183 – 1190.

Zadkarami, M., Shahbazian, M., and Salahshoor, K. (2016). Pipeline leakage detection and isolation: An integrated approach of statistical and wavelet feature extraction with multilayer perceptron neural network (mlpnn). *Journal of Loss Prevention in the Process Industries*, 43:479–487.

Zadkarami, M., Shahbazian, M., and Salahshoor, K. (2017). Pipeline leak diagnosis based on wavelet and statistical features using dempster–shafer classifier fusion technique. *Process safety and environmental protection*, 105:156–163.

Zell, A., Mache, N., Huebner, R., Mamier, G., Vogt, M., Schmalzl, M., and Herrmann, K.-U. (1994). Snns (stuttgart neural network simulator). In *Neural Network Simulation Environments*, pages 165–186. Springer.

Zhang, G., Hu, M. Y., Patuwo, B. E., and Indro, D. C. (1999). Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis. *European Journal of Operational Research*, 116(1):16 – 32.

Apéndice A

Productos logrados en la investigación

Articulos publicados en Congreso internacional "10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes"

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-24 (2018) 373–380

Diagnosis of Fluid Leaks in Pipelines Using Dynamic PCA ⋆

I. Santos-Ruiz [*,1] **F. R. López-Estrada** [*] **V. Puig** [**]
E. J. Pérez-Pérez [*] **J. D. Mina-Antonio** [***]
G. Valencia-Palomo [****]

[*] *Tecnológico Nacional de México, Instituto Tecnológico de Tuxtla Gutiérrez, TURIX Dynamics – Diagnosis and Control Group. Carr. Panamericana km 1080, 29050, Tuxtla Gutiérrez, Chiapas (Mexico)*

[**] *Department of Automatic Control (ESAII), Universitat Politècnica de Catalunya (UPC), Rambla de Sant Nebridi 10, 08222 - Terrassa (Spain)*

[***] *Tecnológico Nacional de México, CENIDET. Interior Internado Palmira S/N, Col. Palmira, 62490, Cuernavaca, Morelos (Mexico)*

[****] *Tecnológico Nacional de México, Instituto Tecnológico de Hermosillo. Av. Tecnológico y Periférico Poniente S/N, 83170, Hermosillo, Sonora (Mexico)*

Abstract: In this paper, a data-driven system based on PCA is described to detect and quantify fluid leaks in an experimental pipeline. A dynamic PCA implementation (DPCA) was used to capture the process dynamics because the system variables are time-correlated. To detect leaks online, the Hotelling's T^2 statistic and the squared prediction error (SPE) were used as residuals, which are compared against statistically defined thresholds from a set of training data. To determine the number of delays to be included in the DPCA model as well as the number of principal components to be used, a tuning process was executed to find the residual with the optimal number of delays and components that showed the best correlation between the residuals and the leakage size. This allowed the construction of a regression model to estimate the flow rate of the leaks directly from the residual.

Figura A.1: Fuente: https://doi.org/10.1016/j.ifacol.2018.09.604

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-24 (2018) 748–754

IFAC Papers Online CONFERENCE PAPER ARCHIVE

On-line head loss identification for monitoring of pipelines ⋆

J. Rojas * **C. Verde** * **L. Torres** *,** **E. Pérez** ***

* *Universidad Nacional Autónoma de México, Instituto de Ingeniería, México*
(jrojasa@iingen.unam.mx,verde@unam.mx, ftorreso@iingen.unam.mx)
** *Cátedras CONACyT*
*** *Departamento de Ingeniería Electrónica, Instituto Tecnológico de Tuxtla Gutiérrez, México*

Abstract: This paper presents an approach based on state observers to identify on-line the parameters associated with frictional head loss in pressurized pipelines. A characteristic of the approach is that it allows the parameters' identification for any line or branch of a network. In particular, two smooth approximations for the friction are considered: a linear and a nonlinear function. These functions depend on the flow rate and two coefficients, which are identified without assuming that either the pipeline roughness nor the Reynolds number are known. Hence, the approach can be used on-line to supervise the status of a pipeline section throughout its useful life and not only during the design and maintenance stages. The simplicity and performance of the approach is shown by considering real data of two water pipelines with different physical characteristics.

Figura A.2: Fuente: https://doi.org/10.1016/j.ifacol.2018.09.659

Articulo para revista Engineering Applications of Artificial Intelligence (En revisión)

Localization of leaks in a pipeline using artificial neural networks☆

E. J. Peréz-Peréz[a], L. Torres[b,d], F. R. López-Estrada[a,*], G. Valencia-Palomo[a], J. D. Mina-Antonio[c]

[a] *Tecnológico Nacional de México / Instituto Tecnológico de Tuxtla Gutiérrez, TURIX-Dynamics Diagnosis and Control Group, Carretera Panam. km 1080,CP 29050, Tuxtla Gutiérrez, Chiapas, México.*
[b] *Universidad Autonoma de México (UNAM) / Instituto de Ingeniería, Circuito Exterior, Cd. Universitaria, 04510 Coyoacán, Ciudad de México, México.*
[c] *Tecnológico Nacional de México / CENIDET, Interior Internado Palmira S/N, Col. Palmira, 62490, Cuernavaca, Morelos, México.*
[d] *Cátedras CONACyT*

Abstract

The main contribution of this work is a new approach based on an Artificial Neural Network (ANN) to locate leaks in a water pipeline. The main novelty of this ANN, with respect to others previously designed for the same purpose, is that it requires the friction and the pressure differential as inputs to compute the output: the leak position. The ANN was trained by using the simulator of a lab pipeline, which was validated prior to the conception of the ANN. For the training of the ANN, different leaks were simulated in different positions and taking into account different pressure differentials. In order to evaluate the performance of the ANN, a set of experimental data from the lab pipeline was used.

Figura A.3: Articulo: Localization of leaks in a pipeline using artificial neural networks

Apéndice B

Codigo de Matlab

B.1. Codigo para interfaz de usuario

```
methods (Access = private)
%Code that executes after component creation
function startupFcn(app)
app.Valvula1Switch.Value = 'Off';
app.Valvula2Switch.Value = 'Off';
app.Valvula3Switch.Value = 'Off';
app.Valvula4Switch.Value = 'Off';
app.Fuga1Lamp.Color = [.47 .67 .19];
app.Fuga2Lamp.Color = [.47 .67 .19];
app.Fuga3Lamp.Color = [.47 .67 .19];
app.Fuga4Lamp.Color = [.47 .67 .19];
delete(instrfind)
app.s = daq.createSession('ni');
addDigitalChannel(app.s,'Dev1','Port0/Line0:3','OutputOnly');
outputSingleScan(app.s,[1,1,1,1]);
delete(instrfind)
app.se = daq.createSession('ni');
app.ch = addAnalogInputChannel(app.se,'cDAQ1Mod1', 0:3, 'Current');
app.se.Rate = 10;
app.se.DurationInSeconds = 600;
app.lh = addlistener(app.se,'DataAvailable', @app.Grafique);
app.se.NotifyWhenDataAvailableExceeds = 10;
app.se.startBackground();
plot(app.UIAxes,0,[0,0]);
app.UIAxes.YLim = [0,10];
title(app.UIAxes,'Grafica de Presion');
xlabel(app.UIAxes,'Tiempo (s)')
ylabel(app.UIAxes,'Presion (KPa)')
```

```
grid(app.UIAxes,'on')
plot(app.UIAxes2,0,[0,0]);
app.UIAxes2.YLim = [0,6];
title(app.UIAxes2,'Grafica de Flujo');
xlabel(app.UIAxes2,'Tiempo (s)')
ylabel(app.UIAxes2,'Caudal (l/s)')
grid(app.UIAxes2,'on')
end
%Value changed function: Valvula1Switch
function Valvula1SwitchValueChanged(app, event)
value = app.Valvula1Switch.Value;
app.estado = xor(app.estado,[0,0,0,1]);
outputSingleScan(app.s,app.estado);
actual = app.estado;
if actual(4) == 1
app.Fuga1Lamp.Color = [0.47,0.67,0.19];
else
app.Fuga1Lamp.Color = [0,1,0];
end
end
%Value changed function: Valvula2Switch
function Valvula2SwitchValueChanged(app, event)
value = app.Valvula2Switch.Value;
app.estado = xor(app.estado,[0,0,1,0]);
outputSingleScan(app.s,app.estado);
actual = app.estado;
if actual(3) == 1
app.Fuga2Lamp.Color = [0.47,0.67,0.19];
else
app.Fuga2Lamp.Color = [0,1,0];
end
end
%Value changed function: Valvula3Switch
function Valvula3SwitchValueChanged(app, event)
value = app.Valvula3Switch.Value;
app.estado = xor(app.estado,[0,1,0,0]);
outputSingleScan(app.s,app.estado);
actual = app.estado;
if actual(2) == 1
app.Fuga3Lamp.Color = [0.47,0.67,0.19];
else
app.Fuga3Lamp.Color = [0,1,0];
end
end
%Value changed function: Valvula4Switch
```

```
74 function Valvula4SwitchValueChanged(app, event)
75 value = app.Valvula4Switch.Value;
76 app.estado = xor(app.estado,[1,0,0,0]);
77 outputSingleScan(app.s,app.estado);
78 actual = app.estado;
79 if actual(1) == 1
80 app.Fuga4Lamp.Color = [0.47,0.67,0.19];
81 else
82 app.Fuga4Lamp.Color = [0,1,0];
83 end
84 end
```