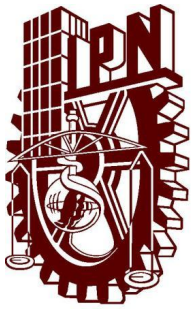


INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Laboratorio de Sistemas Inteligentes para la Automatización

Sensor virtual para predecir el área de contacto de los neumáticos vía redes neuronales recurrentes.

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO

P R E S E N T A:

Leopoldo Urbina Marquez

Director de tesis:
Dr. Marco Antonio Moreno Armendáriz

México, Ciudad de México Diciembre 2016

SIP-14

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 12:00 horas del día 11 del mes de noviembre de 2016 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Sensor virtual para predecir el área de contacto de los neumáticos vía redes neuronales recurrentes"

Presentada por el alumno(a):

Urbina	**Marquez**	**Leopoldo**
Apellido paterno	Apellido materno	Nombre(s)

Con registro:

B	1	4	0	4	4	4

aspirante de: **MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Director de Tesis

Dr. Marco Antonio Moreno Armendáriz

Dr. Sergio Suárez Guerra

Dr. Oleksiy Pogrebnyak

Dr. Luis Pastor Sánchez Fernández

Dra. Nareli Cruz Cortés

Dr. Francisco Hiram Calvo Castro

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Marco Antonio Ramírez Salinas

ESTADOS UNIDOS MEXICANOS
INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

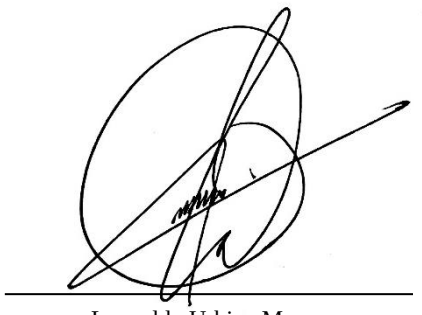
INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de _____México_____ el día _25_ del mes ___Noviembre____ del año __2016__, el que suscribe _____Leopoldo Urbina Marquez_____ alumno (a) del Programa de Maestría en Ciencias en Ingeniería de Cómputo con número de registro b140444, adscrito al ____Centro de Investigación en Computación____, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del __Dr. Marco Antonio Moreno Armendáriz__ y cede los derechos del trabajo intitulado _Sensor virtual para predecir el área de contacto de los neumáticos vía redes neuronales recurrentes_, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección: _b140444@sagitario.cic.ipn.mx_. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Leopoldo Urbina Marquez

Resumen

En este trabajo, un sensor virtual predictivo basado en una red neuronal recurrente es propuesto para predecir una de las zonas más críticas de un vehículo, el área comprendida entre los neumáticos y el terreno, la cuál es uno de los temas más importantes en la industria automotriz, de hecho, mantener una adecuada área de contacto del neumático garantiza la comodidad y la maniobrabilidad del vehículo. Este sensor virtual predictivo es útil para un sistema de suspensión activa, ya que permite al sistema estar preparado en lugar de limitarse a reaccionar ante una perturbación. El sensor suave usa una red neuronal recurrente la cual permite que el sensor virtual tenga una predicción precisa del área de contacto del neumático. El entrenamiento de una red neuronal recurrente es complicado para algoritmos basados en el cálculo del gradiente, debido a la aparición de algunos valles en la superficie de error. Como alternativa, el proceso de entrenamiento se puede modelar como un problema de optimización y este es resuelto mediante un algoritmo de evolución diferencial.

Abstract

In this work, recurrent neural network soft sensor is proposed to predict the measure of one of the most critical issues in a car which is the area where its tires are in contact with the ground.In fact, preserving a steady tire contact patch suitable ensures the comfort and maneuverability of the vehicle. This predictive soft sensor is particularly useful for an active suspension system because it allows the suspension to be prepared instead of only reacting to a disturbance. The core of the soft sensor is an recurrent neural network, the recurrent neural network allows the soft sensor to have a correct prediction of the contact area of the tire. The training of a recurrent neural network is complicated for algorithms based on the gradient, because of the apparition of some valleys on the error surface. For this reason, the training problem is considered as an optimization problem, solved by using a differential evolution algorithm on its training phase.

Agradecimientos

A mi madre, mi hermana y mi abuela, ya que me es imposible imaginar que sería de mí sin el apoyo incondicional de ellas. Gracias a ellas he alcanzado este punto en mi vida, y es mi más grande deseo el saber que están orgullosas de la persona en la que me he convertido.

Al Dr. Marco Antonio Moreno Armendáriz, por su confianza, consejos y tutela, que sirvieron tanto para mi desarrollo académico como personal.

Al Dr. Carlos Alberto Duchanoy Martínez, por su amistad y apoyo que hicieron de esta etapa de mi vida un continuo desarrollo.

A mis compañeros de la carrera, por su aliento y por la confianza depositada en mí.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), por el apoyo económico que me permitió enfocarme totalmente en mis estudios de maestría.

Al Instituto Politécnico Nacional, por la excelente formación académica que me ha brindado a lo largo de estos 9 años.

"Courage is not having the strength to go on; it is going on when you don't have the strength."

Theodore Roosevelt

Índice general

Resumen VII

Abstract IX

Agradecimientos XI

Índice XVI

Índice de figuras XVII

Índice de tablas XIX

Índice de algoritmos XXI

1. Introducción **1**
- 1.1. Definiciones . . . 1
  - 1.1.1. Área de contacto del neumático . . . 1
- 1.2. Planteamiento del problema . . . 1
- 1.3. Objetivos . . . 2
  - 1.3.1. Objetivo general . . . 2
  - 1.3.2. Objetivos específicos . . . 2
- 1.4. Justificación . . . 3
  - 1.4.1. Beneficios esperados . . . 3
  - 1.4.2. Alcances y límites . . . 3
- 1.5. Organización de la tesis . . . 4
- 1.6. Publicaciones y reconocimientos . . . 4

2. Estado del arte y fundamentos **7**
- 2.1. Antecedentes y estado del arte . . . 7

2.2. Fundamento Teórico 10
2.2.1. Optimización 10
2.2.2. Redes Neuronales 19

3. Diseño de sensor virtual **23**
3.1. Selección de variables de entrada 25
3.1.1. Modelo matemático 25
3.1.2. Lista de variables de entrada 29
3.2. Adquisición de Datos 32
3.3. Acoplamiento del modelo regresor 34
3.3.1. Modelo regresor 34
3.3.2. Entrenamiento por gradiente 38
3.3.3. Entrenamiento por optimización 39
3.3.4. Preprocesamiento de datos 46

4. Resultados experimentales **49**
4.1. Condiciones de los experimentos 49
4.2. Experimentos 51
4.3. Discusión 56

5. Conclusiones y trabajo futuro **59**
5.1. Conclusiones 59
5.2. Trabajo a Futuro 60

Referencias **61**

Índice de figuras

2.1. Puntos donde el gradiente es cero. 11
2.2. Pasos generales de una técnica de búsqueda 18
2.3. Neurona Simple 21
2.4. Red neuronal de tres capas y n neuronas. 22

3.1. Proceso de diseño de un sensor virtual [16] 24
3.2. Modelo matemático desarrollado en [13] 25
3.3. Descripción geométrica de la llanta. 26
3.4. Diferentes áreas de contacto en situaciones de reposo, aceleración, frenado y vuelta. 30
3.5. Descripción física de algunas variables de entrada. 31
3.6. Pista objetivo. 32
3.7. Arquitectura de la NARX Propuesta. 36
3.8. Tipos de conexión de la retroalimentación. 39
3.9. Conjunto de datos. 46
3.10. Tipo de conexión para los tres distintos de conjuntos de datos. 48

4.1. Arquitectura final de la NARX. 50
4.2. Predicción del área de contacto de los cuatro neumáticos. . . . 52
4.3. Acercamiento predicción neumático 2. 53
4.4. Comportamiento de los últimos errores de validación promedio de la población. 54
4.5. Comportamiento general de los errores. 55

Índice de tablas

3.1. Dimensiones de las matrices de pesos y bías 37

4.1. Parámetros del algoritmod de ED 50
4.2. Ventajas de la detección temprana de sobreajuste 56
4.3. Criterio de paro de los experimentos 56

Índice de algoritmos

1. Algoritmo de gradiente ascendente 12
2. Algoritmo basado en el método de Newthon 12
3. Algoritmo de Ascenso 13
4. Algoritmo de ascenso escalado 14
5. Algoritmo de ascenso escalado con remplazo 15
6. Algoritmo de búsqueda aleatoria 16
7. Algoritmo de ascenso escalado con remplazo aleatorio 16

8. Algoritmo de evolución diferencial para entrenamiento 45

Capítulo 1

Introducción

En éste capítulo, se expone de manera breve el estado del arte que envuelve a este trabajo de tesis, así como los antecedentes tecnológicos que inspiraron al desarrollo de este trabajo. Del mismo modo se plantean los objetivos y alcances de esta investigación, por último se detalla la organización y contenido de los capítulos de la tesis.

1.1. Definiciones

1.1.1. Área de contacto del neumático

El área de contacto del neumático es la superficie del caucho del neumático que hace contacto con el terreno. La dimensión del área de contacto dependerá de la carga que actúa sobre ella, así como la presión de inflado, entre otras variables físicas.

1.2. Planteamiento del problema

Para asegurar el confort y la seguridad de un vehículo, estos cuentan con un sistema de suspensión, el cuál tiene la tarea de absorber las irregularidades del terreno. A lo largo del tiempo se buscó que este sistema lograra adaptarse a los distintos usos según el camino por donde se conduce. La suspensión activa es una solución real a este reto. Estos sistemas son capaces de combinar grandes niveles de confort, control y maniobrabilidad, lo que se traduce en mayor seguridad. Un sistema de monitoreo es encargado de

sensar en todo momento el perfil de los distintos caminos y enviar señales de control a los subsistemas de suspensión delantera y trasera. [14] Con el fin de mejorar el desempeño de las suspensiones activas, así como de asegurar el confort y la seguridad de un vehículo, el presente trabajo pretende aportar una aplicación de una red neuronal recurrente en un sensor virtual predictivo. Este sensor será útil ya que podrá predecir irregularidades en el terreno, la ventaja de este sensor radica en que el control podrá responder antes que suceda la perturbación, en lugar de responder una vez que sucedió. Para el entrenamiento de la red neuronal con la que funcionará el sensor, se propone un algoritmo inspirado en la evolución diferencial, el cual se modificó para el evitar el sobre entrenamiento de la red neuronal recurrente.

1.3. Objetivos

1.3.1. Objetivo general

Diseñar un sensor virtual basado en una red neuronal recurrente entrenada por un algoritmo de evolución diferencial, con la finalidad de predecir el área de contacto de las llantas con el terreno.

1.3.2. Objetivos específicos

Objetivo 1. Obtener el conjunto de datos que servirá para el entrenamiento, validación y prueba de la red neuronal recurrente usada en el sensor virtual.

Objetivo 2. Elegir la arquitectura de la red neuronal recurrente con base al criterio de error mínimo.

Objetivo 3. Minimizar la función de error de la red neuronal recurrente mediante el algoritmo de evolución diferencial.

Objetivo 4. Evitar el sobreajuste de los datos que pueda presentar la red neuronal recurrente mediante la detección temprana de sobreajuste implementada en el algoritmo de evolución diferencial.

1.4. Justificación

Los neumáticos son un componente importante para garantizar la seguridad de un vehículo terrestre. La razón radica en que el área de contacto entre la llanta y el suelo es donde las fuerzas del vehículo interactúan con el entorno, mantener un área de contacto adecuada garantiza el confort y la maniobrabilidad del vehículo. Un ejemplo de la importancia de monitorear el comportamiento de la llanta es la normativa impuesta en el 2005 por la Administración Nacional de Seguridad del Tráfico (NHTSA, por sus siglas en inglés), en la cual se exige la instalación de sistemas de monitoréo de presión de neumático (TPMS, por sus siglas en inglés) [1]. Además, en el estudio presentado en [9] se demuestra que las condiciones adversas del terreno, aunado con los defectos en los neumáticos, son la mayor causa de accidentes en carretera, los sensores en los neumáticos podrían ser usados como sistemas de prevención, ser parte de un sistema de control, como pueden ser las suspensiones adaptativas, sistemas de frenado anti bloqueo, sistemas avanzados de asistencia al conductor o prevención de coliciones, así como mejorar la maniobrabilidad en condiciones de terreno desfavorables.

1.4.1. Beneficios esperados

Esta tesis busca obtener:

- Diseño de la arquitectura de una red neuronal recurrente
- Una meteheurística adecuada al problema
- Un sensor virtual capaz de predecir el área de contacto

1.4.2. Alcances y límites

Los datos requeridos para el entrenamiento de la red neuronal serán obtenidos basados en un modelo previamente validado. Dado a que se realiza una optimización para una pista, si el sensor virtual se prueba en otra pista el desempeño no será el ideal. El sensor virtual funciona para los parámetros para los que fue entrenado, velocidad y frecuencia de muestreo, si se usan otros parámetros requiere volver a ser entrenado.

1.5. Organización de la tesis

Capítulo 1 inicia con una descripción del problema, los objetivos y finalmente, con la justificación de la tesis.

Capítulo 2 inicia con una introducción general sobre los sensores virtuales predictivos, así como el estado del arte. Después se presentan las bases que rigen el contenido de la tesis, se exponen las ventajas de usar una red neuronal recurrente sobre otros métodos predictivos, además de las complicaciones que se encuentran en el proceso de entrenamiento de estas redes y como un algoritmo de evolución diferencial puede ser una solución eficiente.

Capítulo 3 inicia con la descripción del proceso de diseño de la red neuronal recurrente usada en la tesis, así como el acoplamiento del algoritmo de evolución diferencial al proceso de entrenamiento.

Capítulo 4 inicia con los resultados obtenidos a partir del proceso de entrenamiento, así como el desempeño del sensor virtual predictivo.

Capítulo 5 inicia con el análisis de los resultados obtenidos en el Capítulo 4, así como proponer el trabajo que se pueda alcanzar en un futuro gracias a este trabajo de tesis.

1.6. Publicaciones y reconocimientos

Publicaciones relacionados con la tesis

[41] Urbina, L., Duchanoy, C. A., Faustino-González, G., Moreno-Armendáriz, M. A., Cruz-Villar, C. A., & Calvo, H. (2015, October). A novel tire contact patch soft sensor via Neural Networks. In Electrical Engineering, Computing Science and Automatic Control (CCE), 2015 12th International Conference on (pp. 1-6). IEEE.

[*] Urbina, L., Duchanoy, C. A., Moreno-Armendáriz, M. A., Lara, D., & Calvo, H. (2016). Pronóstico del área de contacto de los neumáticos de un vehículo vía redes neuronales recurrentes. Research in Computing Science

[*] Duchanoy, C. A., Moreno-Armendáriz, M. A., Urbina, L., Cruz-Villar, C. A., & Calvo, H, de Jesus Rubio, J. A Novel Recurrent Neural Network Soft Sensor via a Differential Evolution Training Algorithm for the Tire Contact Patch. Neurocomputing (Impact Factor 2.392)

[*] Aceptados en espera de publicación.

Publicaciones no relacionados con la tesis

[42] Urbina, L., Duchanoy, C. A., Moreno-Armendáriz, M. A., Lara, D., & Calvo, H. (2015, Noviembre). Implementación sobre FPGA de la estrategia evolutiva CMA-ES para optimización numérica. Research in Computing Science Issue 105 (pp. 97-106).

Reconocimientos

Premio al mejor desempeño académico 2016 por el programa de maestría en Ciencias en Ingeniería de Cómputo.

Capítulo 2

Estado del arte y fundamentos

En este capítulo se presenta el estado del arte de la investigación relacionada con la tesis además de los dos temas principales de este trabajo. La sección de optimización presentará las características de la optimización por gradiente y mediante métodos evolutivos. Finalmente, la sección dedicada a redes neuronales abordará temas como arquitectura, ventajas, desventajas y entrenamiento de estas.

2.1. Antecedentes y estado del arte

El diseño de un sensor destinado para su uso en los neumáticos, presenta algunos retos. En [29] se exponen tres principales retos. El primer reto es desarrollar un sistema de medición directa de la deformacion o tension del neumático, el sistema debe contar con la propiedades mecánicas de tener baja rigidez y alta capacidad de deformación, estas propiedades permitirán al sensor deformarse junto con la llanta. El segundo reto es crear un sistema de comunicación inalámbrica entre la llanta y el vehículo que trabaje sin batería, esto es debido al posible desbalanceo del neumático. Por último, el sensor se debe caracterizar y probar en condiciones reales.

Algunas metodologías para medir parámetros en los neumáticos tales como presión o temperatura, se encuentran reportados en [25], donde se detalla un primer intento por medir el área de contacto, mediante la obtención de una imagen térmica de la llanta, se puede observar el área que está en contacto con el suelo. Esto es debido al calor generado por la fricción de la llanta con el terreno, esta metodología presenta ciertas desventajas, por ejemplo, para

lograr que el cambio de temperatura en el dibujo del neumático sea evidente, es necesario que pase cierto tiempo para que la temperatura en la llanta aumente. Algunas alternativas para medir directamente el área de contacto mediante un sensor piezoeléctrico, están reportadas en, [30], [25], [27] y [28]. Estos trabajos previamente mencionados tienen un problema en común, debido a que los sensores se encuentran dentro del neumático, si el sensor se daña, se necesitan herramientas especializadas para hacer el remplazo, además la manufactura requerida para el desarrollo del sensor es costosa. Con el fin de resolver el problema de la comunicación inalámbrica y el gasto eléctrico que esto conlleva, en [24] y [44] se proponen diseños de generadores piezoeléctricos, ambos basado en generar energía eléctrica por medio de un espectro de vibración y aceleraciones centrípetas. Al igual que los sensores anteriores, estos dispositivos son suceptibles a daños durante la operación del vehículo, su instalación requiere de herramienta especializada y el costo de la manufactura es alto. En [7] se desarrolló un banco de pruebas para obtener el área de contacto de la llanta, el proceso de medición está basado en el fenómeno de la reflexión interna total frustrada (FTIR, por sus siglas en inglés). El banco de pruebas consiste en un panel de plástico iluminado alrededor de su periferia, en el la llanta es apoyada, debido al fenómeno FTIR, la luz es reflejada del área de contacto. Mediante una videocámara, la luz reflejada es captada para después ser procesada.

Como alternativa se puede optar por una medición indirecta, por ejemplo, en [33] un sistema de posicionamiento global (GPS, por sus siglas en inglés) es usado para estimar los ángulos de deslizamiento longitudinal, del mismo modo en [5], se logra estimar la rigidez en los extremos del dibujo de una llanta mediante la información de velocidad en el GPS en conjunto con el sistema de navegación inercial (INS, por sus siglas en inglés). En [4] la estimación del movimiento de un vehículo es aproximada mediante las mediciones de una unidad de medición inercial (IMU, por sus siglas en inglés) y un módulo de GPS.

Otra alternativa es usar un sensor virtual para obtener la medida del área de contacto de la llanta, un sensor virtual es un modelo computacional el cual permite la medición de casi cualquier parámetro [16]. El sensor virtual relaciona mediciones más sencillas de obtener con la medición de interés, todo a través de un modelo. Los sensores virtuales se dividen en, manejados por datos y manejados por modelo. Los sensores manejados por modelo, requieren forzosamente de un modelo matemático que describa el sistema. En la mayoría de los casos la dinámica del sistema es muy compleja para poderla

modelar [36]. Los sensores virtuales manejados por datos no necesitan de un modelo matemático, estos requieren de un modelo de caja negra para simular el sistema, tales modelos pueden ser métodos estadísticos multivariantes, maquinas de soporte vectorial, redes neuronales, entre otros. Por ejemplo, en [43], se propone un sensor virtual basado en el filtro de Kalman y modos deslizantes para estimar el centro de gravedad y el ángulo de deslizamiento lateral del vehículo. En [32] se diseña un sensor virtual basado en una red neuronal, capaz de estimar los estados del vehículo tales como la velocidad de guiñada y el ángulo de deslizamiento lateral, estados que son importantes para el control de estabilidad del vehículo. En [10] se desarrolla un sensor virtual mediante la aplicación de un filtro de Kalman extendido, para estimar el coeficiente de fricción y el ángulo de deslizamiento lateral. En [34], se realiza la estimación de dos estados del vehículo, ángulo de deslizamiento lateral y ángulo de balanceo, estos ángulos son esenciales para aplicaciones de control de estabilidad del vehículo, tales como el control de estabilidad antivuelco. En la estimación del deslizamiento lateral del vehículo, un algoritmo recursivo de mínimos cuadrados (RLS, por sus siglas en inglés) con un factor de olvido es empleado. En la estimación del ángulo de balanceo, se usa el filtro de Kalman.

Las redes neuronales suelen ser empleadas comúnmente en los sensores virtuales debido a sus capacidades como aproximadores universales [21]. Dependiendo de la arquitectura y el tipo de conexiones que tenga una red neuronal, puede ser aplicada para resolver problemas con dinámica compleja. En una arquitectura más compleja, una red neuronal recurrente (RNN, por sus siglas en inglés) puede resolver problemas modelados como series de tiempo. Por lo tanto un sensor virtual que use una red neuronal recurrente, además de poder estimar mediciones complejas de un sistema dinámico, es capaz de predecirlas. Un ejemplo de sensor predictivo se presenta en [17], donde se usa un modelo de inferencia neuro-difuso adaptativo para predecir el comportamiento de un vehículo en la maniobra de cambio de carril. En [40] se presenta un sistema de control inteligente de neumático, donde sensores piezoeléctricos se usan como entrada a una red neuronal la cual estima y predice, las diferentes fuerzas presentadas en el neumático. En [26], se propone una red neuronal para predecir irregularidades en el terreno basado en las aceleraciones del vehículo.

Uno de los principales problemas en el uso de redes neuronales, es el entrenamiento. El proceso de entrenamiento consiste en ajustar los pesos sinápticos y bías de la red hasta que el error entre las señales de salida de la

red y la señal deseada sea mínimo. Existen distintos métodos con los cuales la red puede ser entrenada, la mayoría de estos métodos tratan de calcular el gradiente o en el mejor caso, aproximarlo, el algoritmo de Levenberg-Marquardt [39] es uno de ellos. Estos métodos son eficientes para el entrenamiento de redes neuronales sin retroalimentación, en el caso de las redes recurrentes existen diversas dificultades [2]. La principal razón es la presencia de valles espurios en la superficie del error; estos valles son mínimos locales y cambian dependiendo de la secuencia de entrada [11]. En [35] se describe una forma de lograr un entrenamiento exitoso sin caer en un valle espurio, basta con monitorear la magnitud del gradiente, si este es muy grande, es un indicio de que estamos en un valle, entonces el conjunto de entrada se cambia temporalmente para entrenar con otra secuencia. Existen otros métodos para lograr un entrenamiento exitoso, en [3] sugieren usar algoritmos de búsqueda estocástica, tales como el algoritmo de recocido simulado, evolución diferencial, entre otros. El problema de entrenamiento puede ser interpretado como un problema de optimización, por lo tanto puede ser resuelto por métodos no basados en el gradiente [20]. En [22] es usado un algoritmo de evolución diferencial como base de entrenamiento de una red neuronal, estos métodos estocásticos pueden evitar perderse en los valles espurios debido a su mecanismo de búsqueda.

2.2. Fundamento Teórico

2.2.1. Optimización

El proceso de optimización es la selección del mejor elemento de un conjunto de posibles soluciones, siempre con respecto a algún criterio. Los problemas de optimización constan de los siguientes elementos:

Variables de diseño.- Son los parámetros del sistema que se pueden modificar para alterar el valor de la función objetivo.

Restricciones.- Generalmente son un conjunto de ecuaciones o inecuaciones que limitan el valor que pueden adquirir las variables de diseño. Por ejemplo, la potencia máxima de un motor.

Espacio de búsqueda.- Son el conjunto de todas las posibles combinaciones de valores en las variables de diseño que a su vez cumplen el conjunto de restricciones.

Función objetivo.- Es una función que emula el funcionamiento del

sistema que se desea optimizar, como resultado tendremos un valor cuantitativo. En otras palabras es la función que vincula un elemento del espacio de búsqueda con el desempeño del sistema. Un ejemplo de función objetivo es el error cuadrático medio.

El proceso de optimización incluye dos ramas diferentes pero estrechamente relacionada una de la otra. La primer rama es el modelado de sistemas, esta rama se encarga de describir y expresar el sistema de estudio de forma que sea lo más cercano a la realidad. La segunda rama es la resolución de modelos, esta busca el mejor método de solución, es importante conocer el funcionamiento de cada método de solución, ya que de este modo podremos elegir cuál método de solución es más adecuado para cada problema, y modelarlo de una forma más amigable.

De acuerdo a la característica del problema, hay distintas maneras de clasificar a un problema de optimización, analizar la naturaleza del problema permitirá elegir el método de solución adecuado, ya que *"No existe un optimizador universal"*. Los métodos de resolución se pueden clasificar en los siguientes tipos tipos:

Métodos basados en gradiente

Los métodos basados en gradiente usan el fundamento matemático de la derivada para determinar que valores del dominio representan un máximo, mínimo o punto de inflexión de la función. Este tipo de métodos de optimización son muy precisos, pero requieren que tanto la función objetivo como las restricciones sean funciones continuas y definidas en cada punto.

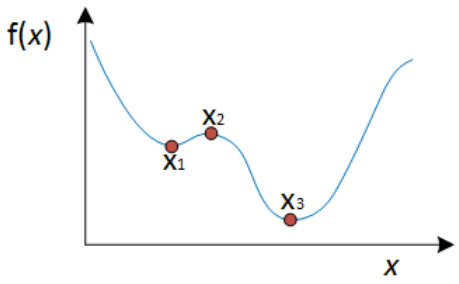


Figura 2.1: Puntos donde el gradiente es cero.

Gradiente Ascendente La mecánica de este algoritmo es simple, ver

algoritmo 1. Iniciamos en un valor arbitrario para x, después repetidamente nos movemos en una porción α del gradiente, donde α es un valor positivo normalmente elegido entre 0 y 1. El algoritmo parará hasta que el cambio en x sea mínimo o se termine el número de iteraciones. El gradiente asegurará que el elemento siempre se dirija hacia un mínimo, pero también es común que se pueda estancar en un punto de inflexión (x_2 de la Figura 2.1) o en un mínimo local (x_1 de la Figura 2.1).

Algoritmo 1: Algoritmo de gradiente ascendente

$\overrightarrow{x} \leftarrow$ Inicializar con un vector aleatorio
mientras *no se cumpla ningún criterio de paro* **hacer**
  $\overrightarrow{x} \leftarrow \overrightarrow{x} + \alpha\vec{\nabla}f(\overrightarrow{x})$
fin
devolver $\overrightarrow{x}$

Si no tuviéramos un control en el incremento del gradiente (α), el algoritmo se desplazará con un mayor paso, pudiendo omitir algunas soluciones, por el lado contrario, si el valor de α es muy pequeño, el algoritmo fácilmente podría detenerse al mínimos locales o puntos de inflexión.

Método de Newton

El problema de poder controlar el valor de α se puede lograr con una modificación al algoritmo 1. Si además de calcular el gradiente, se calcula el valor del Hessiano (segunda derivada para una dimensión), podremos aplicar el método de Newton. El algoritmo 2, contiene esta implementación, gracias a la modificación el valor de α es amortiguado en medida que nos acerquemos a un mínimo.

Algoritmo 2: Algoritmo basado en el método de Newthon

$\overrightarrow{x} \leftarrow$ Inicializar con un vector aleatorio
mientras *no se cumpla ningún criterio de paro* **hacer**
  $\overrightarrow{x} \leftarrow \overrightarrow{x} - \alpha\frac{\vec{\nabla}f(\overrightarrow{x})}{\mathbf{H}_f(\overrightarrow{x})}$
fin
devolver $\overrightarrow{x}$

Gracias al empleo de la segunda derivada, el método de newton converge más rápido que el del gradiente ascendente. Pero esto no asegura llegar a un

mínimo global, los algoritmos basados en gradiente son excelentes buscadores locales. Una forma de sortear los mínimos locales es elegir un valor de α que ayude a salir de estos valles, o contar con distintos elementos que hagan una búsqueda organizada.

Métodos basados en explotación

Los métodos basados en el gradiente asumen que siempre se puede calcular la derivada del sistema, esto es verídico si se está optimizando un sistema ya estudiado o simplificado en distintas funciones matemáticas. Pero en la mayoría de los casos esto no es posible. Lo único que se puede hacer es modificar las entradas del sistema y cuantificar el desempeño de este. A esto se le conoce como modelo de caja negra, esta caja negra describe el problema que se quiere optimizar. La caja contiene entradas con las que se alimentará con la solución candidata para el sistema. La caja negra arrojará en la salida el desempeño de la solución introducida. Los algoritmos de explotación se caracterizan por hacer una búsqueda local alrededor de una candidato a solución.

Algoritmo de ascenso Esta técnica es simple, de cierto modo guarda relación con la técnica del gradiente ascendente pero este algoritmo no necesita saber la dirección del gradiente. De manera iterativa se prueban nuevas soluciones candidatas y se elije a la mejor. Dependiendo del tipo de modificación (mod), esta mejorará el desempeño de la optimización. La modificación más común es sumarle a la solución candidata un número uniformemente aleatorio entre -1 y 1 o sumarle un número aleatorio con una distribución gaussiana $N(\mu, \sigma)$, ver algoritmo 3.

Algoritmo 3: Algoritmo de Ascenso

```
S ← solución candidata inicial
mientras no se cumpla ningún criterio de paro hacer
    R ← mod(S)
    si calidad(R) ≥ calidad(S) entonces
        S ← R
    fin
fin
devolver S
```

Algoritmo de ascenso escalado El algoritmo de ascenso es parecido al algoritmo del gradiente ascendente. La única diferencia es que la operación de modificación (*mod*) es una búsqueda estocástica con el principal objetivo de encontrar mejores soluciones. Algunas veces este encontrará soluciones peores en la cercanía, y otras veces encontrará soluciones mejores. El algoritmo de ascenso se puede modificar para obtener una mejor solución para cada solución propuesta. Haciendo n operaciones *mod* a la solución candidata, se espera que en esta búsqueda la probabilidad de encontrar una mejor solución aumente. El algoritmo de ascenso escalado hace un muestreo alrededor del candidato a solución para escoger al mejor. Ver algoritmo 4.

Algoritmo 4: Algoritmo de ascenso escalado

```
n ← número de modificaciones permitidas
S ← solución candidata inicial
mientras no se cumpla ningún criterio de paro hacer
    R ← mod(S)
    para n veces hacer
        W ← mod(S)
        si calidad(W) ≥ calidad(R) entonces
            R ← W
        fin
    fin
    si calidad(R) ≥ calidad(S) entonces
        S ← R
    fin
fin
devolver S
```

Algoritmo de ascenso escalado con remplazo Existe una variación del algoritmo de ascenso escalado, en el cual ya no se compara al elemento **R** con **S**, en su lugar, simplemente se remplaza **S** con **R**. Al hacer esta sustitución se corre el riesgo de perder a la mejor solución, por lo tanto se agrega una nueva variable ($S*$) que almacenará el valor del mejor resultado descubierto hasta el momento. El algoritmo 5 describe el comportamiento del algoritmo de ascenso escalado con remplazo.

Algoritmo 5: Algoritmo de ascenso escalado con remplazo

```
n ← número de modificaciones permitidas
S ← solución candidata inicial
S* ← S
mientras no se cumpla ningún criterio de paro hacer
    R ← mod(S)
    para n veces hacer
        W ← mod(S)
        si calidad(W) ≥ calidad(R) entonces
            R ← W
        fin
    fin
    S ← R
    si calidad(S) ≥ calidad(S*) entonces
        S* ← S
    fin
fin
devolver S*
```

Métodos basados en exploración

Este tipo de algoritmos se caracterizan por hacerle frente a los problemas de optimización global. Este tipo de algoritmos garantiza que dentro de los límites establecidos, la mayor parte del espacio de búsqueda es explorada. Esto es algo que los algoritmos de explotación difícilmente lograrían, la razón radica en que la operación de modificación hace ligeros cambios en las soluciones. La operación *mod* no es capaz de hacer cambios radicales. Si la solución candidata estuviera varada en un optimo local, la operación *mod* no sería suficiente para poder alejarnos del óptimo local.

Búsqueda aleatoria Existen varias formas de construir un algoritmo de optimización global. El más simple es la búsqueda aleatoria. Este algoritmo es el extremo de la exploración, al contrario del algoritmo de ascenso, donde la operación *mod* genera pequeños cambios que pueden ser considerados como el extremo de la explotación. El algoritmo 6 describe la búsqueda aleatoria.

Algoritmo de ascenso escalado con remplazo aleatorio El algoritmo de ascenso escalado con remplazo aleatorio es un ejemplo de una búsque-

Algoritmo 6: Algoritmo de búsqueda aleatoria

```
S* ← alguna de las soluciones candidatas aleatorias
mientras no se cumpla ningún criterio de paro hacer
    S ← una solución candidata aleatoria
    si calidad(S) ≥ calidad(S*) entonces
        S* ← S
    fin
fin
devolver S*
```

da y explotación balanceada. Se hace una explotación al estilo del algoritmo de ascenso escalado por determinado tiempo. Una vez terminado el tiempo, se inicia de nuevo el algoritmo de ascenso escalado pero en otra ubicación escogida de manera aleatoria. Para más detalle, ver el algoritmo 7.

Algoritmo 7: Algoritmo de ascenso escalado con remplazo aleatorio

```
T ← conjunto de números enteros
S ← solución candidata aleatoria inicial
S* ← S
mientras no se cumpla ningún criterio de paro hacer
    tiempo ← tiempo aleatorio escogido de T
    para tiempo hacer
        R ← mod(S)
        si calidad(R) ≥ calidad(S) entonces
            S ← R
        fin
    fin
    si calidad(S) ≥ calidad(S*) entonces
        S* ← S
    fin
    S ←alguna solución candidata aleatoria
fin
devolver S*
```

Métodos basados en poblaciones

Los métodos basados en poblaciones difieren de los métodos anteriores ya que se mantienen alrededor de una muestra de soluciones candidatas en lugar de una solución única. Cada una de las soluciones están involucradas en modificar el comportamiento de la siguiente generación. Es común que la mayoría de los métodos basados en poblaciones imitan conceptos de la biología.

La computación evolutiva contiene a un conjunto de estás técnicas que imitan a algunos modelos biológicos de poblaciones así como reglas de la genética y la evolución. Un algoritmo elegido de esta colección es conocido como un algoritmo evolutivo (AE). Entre los AE se incluyen a el Algoritmo Genético (AG) y las estrategias de evolución (ES). Debido a que están inspirados en la biología, el CE métodos tienden a usar términos de la genética y la evolución.

Evolución Diferencial

Evolución Diferencial (DE) es un algoritmo de optimización global originalmente propuesto por Price y Storn en 1995 [38]. Debido a su simplicidad y a la alta capacidad de búsqueda global, ha sido ampliamente utilizado en muchos de los problemas reales. DE es regido por tres parámetros:

F.- Factor de escala

CR.- Probabilidad de cruza

NP.- Tamaño de la población

El rendimiento de la DE depende en gran medida de los parámetros de control F y CR. En general, un valor grande de F favorece a la exploración global, mientras que valores bajos de F se inclinan a una explotación local. El factor CR determina el número de nuevos componentes que pueden ser introducidos en una solución nueva. ED es una variante de la computación evolutiva diseñada principalmente para espacios con valores reales multidimensionales, además se caracteriza por que los hijos deben competir directamente contra sus padres para su inclusión en la nueva población; ED determina el valor de la mutación basándose en la varianza actual en la población. Si la población se propaga, la mutación hará cambios importantes. Si la población se condensa en una región determinada, la mutación será pequeña. De este modo ED es un algoritmo de mutación adaptativa.

Algoritmo básico de ED

El diagrama de la Figura 2.2 contiene los pasos generales de un algoritmo de ED. Asumimos que $x = [x_1, x_2, ..., x_D]$ y $f(x)$ es la función a minimizar. Donde D es la dimensión del problema.

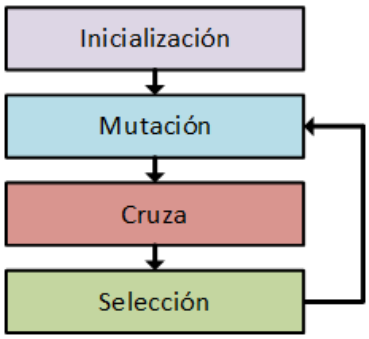


Figura 2.2: Pasos generales de una técnica de búsqueda

Inicialización.- El ED inicializa a los NP individuos aleatóriamente en el espacio de búsqueda, después, las tres operaciones (mutación, cruza y selección) se realizan secuencialmente en cada individuo hasta que un criterio de paro sea cumplido. Donde $x_{i,g}$ es el individuo i que pertenece a la NP población en la generación g, por lo tanto $x^j_{i,g}$ es el elemento j del individuo i en la generación g.

Mutación.- Para cada individuo i de la generación g, el vector mutante $v_{i,g+1}$ es generado por la ecuación 2.1.

$$v_{i,g+1} = x_{r_1,g} + F\left(x_{r_2,g} - x_{r_3,g}\right) \tag{2.1}$$

Donde r_1, r_2 y r_3 son individuos escogidos aleatóriamente de toda la población, de hecho r_1, r_2, r_3 y el índice i son diferentes entre sí. $x_{r_1,g}$ es el vector base, $(x_{r_2,g} - x_{r_3,g})$ es la diferencia vectorial. Como podemos observar de la ecuación 2.1 el factor de escala F controla la amplitud de la diferencia vectorial.

En [31] se hace una comparativa de desempeño entre los distintos tipos de funciones de mutación. La ecuación 2.2 explica la creación del vector mutante por medio de la función *DE/rand/1/bin*, este nombre nos dice las características de esta función: *DE* es de Evolución diferencial por sus siglas en inglés,

rand indica que los individuos elegidos para la mutación son escogidos de maneara aleatoria, *1* es el número de parejas de individuos elegidos y *bin* significa que una recombinación binomial es usada. La ecuación 2.3 representa la función *DE/best/1/bin*, a diferencia de la anterior, se elige al mejor elemento para la mutación. Las ecuaciones 2.4 y 2.5 representan las funciones *DE/rand/2/bin* y *DE/best/2/bin* que a diferencia de la anteriores, usan dos pares de individuos para la mutación.

$$v_{DE/rand/1/bin} = r_1 + F\left(r_2 - r_3\right) \tag{2.2}$$
$$v_{DE/best/1/bin} = best + F\left(r_1 - r_2\right) \tag{2.3}$$
$$v_{DE/rand/2/bin} = r_1 + F\left(r_2 + r_3 - r_4 - r_5\right) \tag{2.4}$$
$$v_{DE/best/2/bin} = best + F\left(r_1 + r_2 - r_3 - r_4\right) \tag{2.5}$$

Cruza.- Para cada elemento j del individuo i de la generación g, el vector de prueba $u_{i,g+1}$ es generado por la operación de cruza descrita en la ecuación 2.6. La operación de cruza se hace entre el vector objetivo $x_{i,g}$ y el vector mutante $v_{i,g+1}$ con base en la probabilidad de cruza CR

$$u^j_{i,g+1} = \begin{cases} v^j_{i,g+1} & \text{si } rand \leq CR \\ x^j_{i,g} & \text{otro caso} \end{cases} \tag{2.6}$$

Donde, *rand* es un número aleatorio uniformemente distribuido entre $[0, 1]$

Selección.- Para cada individuo i de la generación g, un mecanismo de selección voraz es usado para determinar si el vector de prueba $u_{i,g+1}$ puede o no sobrevivir a la siguiente generación, la ecuación 2.7 desarrolla el proceso de selección.

$$x_{i,g+1} = \begin{cases} u_{i,g+1} & \text{si } f(u_{i,g+1}) < f(x_{i,g}) \\ x_{i,g} & \text{otro caso} \end{cases} \tag{2.7}$$

2.2.2. Redes Neuronales

Como se explicó en la sección 2.2.1, el modelado del problema es parte medular de la optimización. Existen distintos métodos que son usados para

aproximar un modelo. Esto es dependiendo de la complejidad del sistema. Para nuestro caso de estudio se eligió un modelo de caja negra debido a que nuestro problema presenta una dinámica compleja. Existen distintos modelos de caja negra tal como: maquinas de soporte vectorial, métodos estadísticos multivariantes, métodos adaptativos, redes neuronales y métodos híbridos. Las redes neuronales presentan un gran desempeño en el modelado de sistemas dinámicos complejos, además que al añadir diversos complementos a su arquitectura, son capaces de realizar predicción en diversos tipos de sistemas [8]. Fortuna en [16] recomienda el uso de las redes neuronales para sensores virtuales.

Una red neuronal (NN, por sus siglas en inglés) es una técnica basada en datos inspirada en el cerebro humano, al igual que nuestro cerebro, una red neuronal hace conexiones entre datos de entrada con una señal de salida. En nuestro cerebro estás conexiones se logran mediante "neuronas", las cuales envían pulsos eléctricos de un punto a otro. En una red neuronal, estas conexiones están representadas por valores numéricos llamados pesos sinápticos, estos pesos deben ser ajustados por algún método de entrenamiento, con el fin de obtener la salida deseada.

Neurona simple

Para poder comprender como entrenar una red neuronal, primero debemos comprender como funciona una red desde su más simple forma. Hagan propone en [19] una nomenclatura amigable e intuitiva. En la Figura 2.3, se muestra el diagrama de una neurona simple, que es la unidad básica de una red neuronal. La neurona consta de una entrada escalar p que es multiplicada por el peso sináptico w para formar el producto wp, el producto de la entrada con el peso sináptico es enviado a un bloque de suma que también recibe el producto de la constante 1 por el bías b. La salida del bloque de suma (n) es enviada a la función de transferencia f que produce la salida de la neurona a

La salida de la neurona se calcula con la ecuación 2.8

$$a = f\left(wp + b\right) \tag{2.8}$$

Como se puede notar, los valores del peso sináptico y bías son parámetros ajustables, los cambios en los parámetros se logran mediante algoritmos de aprendizaje, mientras que el tipo de función de transferencia es elegida por el tipo de problema.

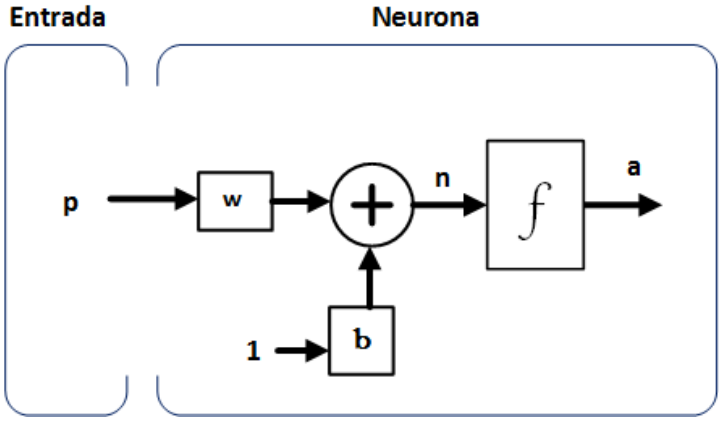


Figura 2.3: Neurona Simple

Función de transferencia Existen dos clases principales de funciones de transferencia, la lineal y las no lineales. Cada tipo de función de transferencia tiene características únicas que la hacen tener un mejor desempeño para ciertos problemas que para otros.

La ecuación 2.9 representa el comportamiento de una función de transferencia lineal, esta función es ideal para clasificaciones lineales y es la función principal de la red de elementos lineares adaptativos (ADALINE, por sus siglas en inglés)

$$a = n \tag{2.9}$$

Una de las ecuaciones más usadas del tipo no-lineal es la función *log-sigmoide*. La ecuación 2.10 describe el comportamiento de esta función, esta función permite a su entrada cualquier valor real, y lo limita a la salida a valores entre 0 y 1. Esta función es comúnmente usada en redes multicapa y gracias a que es una función diferenciable, esta puede ser usada en algoritmos de entrenamiento por gradiente.

$$a = \frac{1}{1 + e^{-n}} \tag{2.10}$$

Redes multicapa

El modelo de neurona contempla una sola entrada y una sola salida, ahora podremos considerar aumentar el número de entradas de la neurona, el número de neuronas y el número de capas neuronales. Por lo tanto los valores de pesos y bías dejarán de ser valores escalares, ahora se constituirán de matrices de datos. En la Figura 2.4 se puede apreciar las conexiones de una red neuronal de tres capas con n neuronas. En este diagrama se puede notar que existen R entradas, S^1 neuronas en la primer capa, S^2 neuronas en la segunda capa, por último S^3 neuronas en la tercer capa. El número de neuronas de cada capa puede ser diferente entre ellas. La salida de las capas uno y dos, son entradas de las capas dos y tres respectivamente.

Las redes multicapa poseen un mejor desempeño que las redes de una sola capa. Por ejemplo, una red con 2 capas, la primera log-sigmoidal y la segunda lineal, puede modelar una mayor variedad de funciones que una red de una sola capa simplemente no podría.

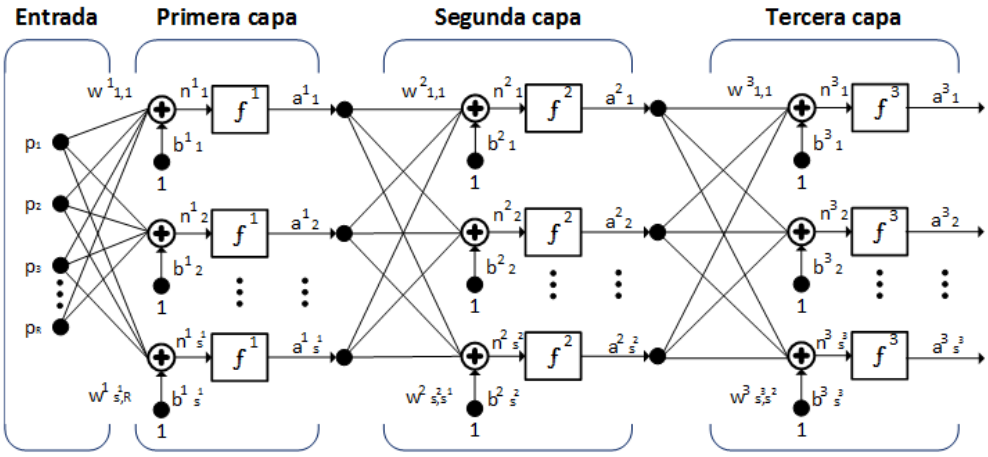


Figura 2.4: Red neuronal de tres capas y n neuronas.

Capítulo 3

Diseño de sensor virtual

La esencia de la Ingeniería es el diseño. Diseñar como acto instintivo se podría llamarse crear o innovar, si el objeto no existe o es una modificación de lo existente, se conoce como transformación. En el ámbito ingenieril, el diseño se define como el proceso previo de visualización en la búsqueda de una solución en cualquier campo. Ninguna metodología en especial puede manejar a la perfección todas las situaciones que se presentan en un momento dado, encontraremos tantas metodologías como soluciones existentes, el motivo principal es que el proceso de diseño es una experiencia humana y como tal, difiere entre cada sociedad. Para ser de utilidad, una metodología debe usarse flexiblemente como guía y no una rutina.

El profesor George Dieter propone un método de diseño [12], el cual consiste en los siguientes pasos:

1. Definición del problema
2. Recopilación de información
3. Conceptualización
4. Evaluación
5. Arquitectura del producto
6. Configuración del diseño
7. Diseño paramétrico
8. Diseño a detalle

Según Diter, la parte más importante del proceso es la definición del problema; ya que el verdadero problema no siempre es el que se ve a simple vista. Definir al problema lo más detallado posible ayudará a evitar caer en soluciones inadecuadas; además nos permitirá tener una mejor perspectiva para el desarrollo de las siguientes fases del diseño.

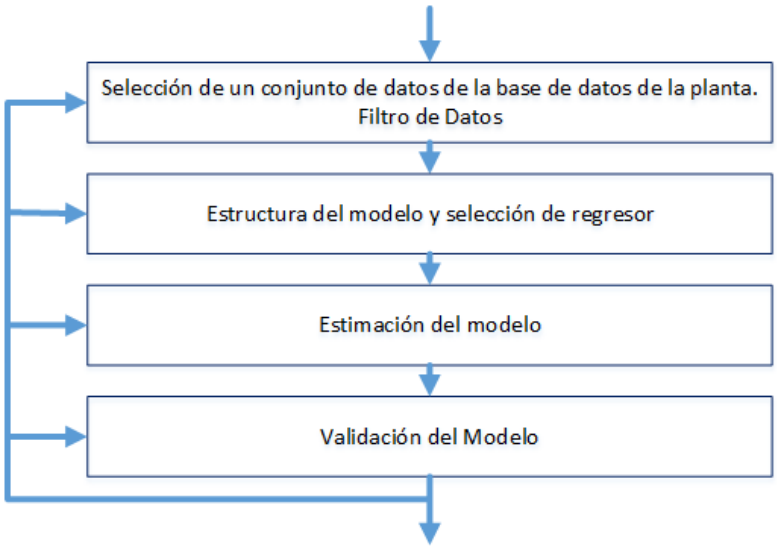


Figura 3.1: Proceso de diseño de un sensor virtual [16]

Así como existen metodologías de diseño mecánico, eléctrico, aerodinámico, por decir algunas. Luigi Fortuna en su libro *Soft Sensors for Monitoring and Control of Industrial Processes* [16] propone una metodología, esta se puede observar en la Figura 3.1. Como ya se ha mencionado antes, una metodología no es para seguirla al pie de la letra, esta puede ser flexible dependiendo de la necesidad del problema. La metodología propuesta para este trabajo de tesis es la siguiente:

1. Selección de variables de entrada

2. Adquisición de datos

3. Acoplamiento del modelo regresor

4. Validación del modelo regresor

El paso 1, 2 y 3 se verán a detalle en este capítulo, dejando para el siguiente capítulo el paso 4.

3.1. Selección de variables de entrada

El primer paso en el diseño de un sensor virtual es el análisis de los datos disponibles, el resultado de este análisis es una lista de variables que estén estrechamente relacionadas con el área de contacto del neumático. La selección de variables debe estar sustentada por un modelo matemático que describa a la planta, en caso de no existir, se pueden incluir experimentos del sistema para obtener las variables relacionadas al fenómeno de interés.

3.1.1. Modelo matemático

Este trabajo de tesis basó su selección de variables en el trabajo presentado en [13],en este trabajo se encuentra un modelo matemático desarrollado para un vehículo tipo baja de tamaño normal. En la Figura 3.2 se puede apreciar el modelo completo, compuesto por el modelo de las llantas, modelo de la suspensión, modelo del sistema de frenado, modelo del sistema de propulsión, modelo del sistema de dirección y modelo del chasis, que como podemos ver, este interactúa con todos los demás sistemas.

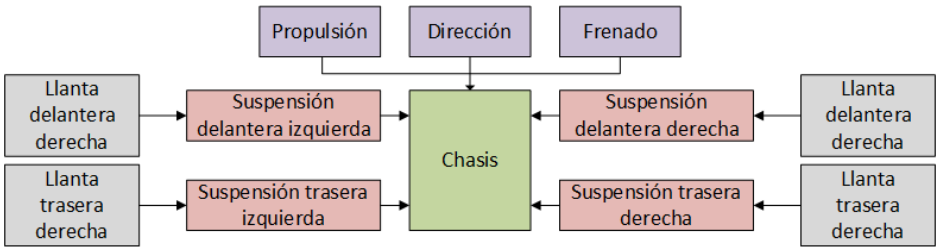


Figura 3.2: Modelo matemático desarrollado en [13]

Del diagrama de la Figura 3.2, podemos notar la relación que tienen directa o indirectamente cada sistema del automóvil, todos entrelazados en el chasis. El chasis es el sistema que conjunta a todos los demás, este sistema junto con las llantas, son los únicos sistemas que interactúan con el ambiente, ya sea con el terreno o con el aire.

Se dará una breve explicación de cada parte del modelo, profundizando más en el modelo de las llantas, el cual es de nuestro interés. Concluyendo con una lista de variables relacionadas con el área de contacto de las llantas.

Modelo de la llanta

El área de contacto de la llanta es el último e indudablemente el enlace más importante del sistema de propulsión, la razón radica en que en esta área es donde todas las fuerzas externas son transmitidas, con excepción de la fuerza de arrastre. El comportamiento de los neumáticos es parecido al de un resorte, la llanta transmite la vibración generada por la forma del terreno al sistema de suspensión que este a su vez amortiguará la vibración para transmitir lo menos posible al chasis. El modelo parte de considerar la forma de terreno como una variación en la altura (eje z), debido a esta variación se puede calcular el área de la llanta. Este modelo no considera los efectos ocasionados por la deformación de la llanta, ya que resulta complicado modelarlos, del mismo modo se considera al área de contacto de forma rectangular. En la Figura 3.3 se muestran gráficamente las variables que ayudarán a modelar el neumático.

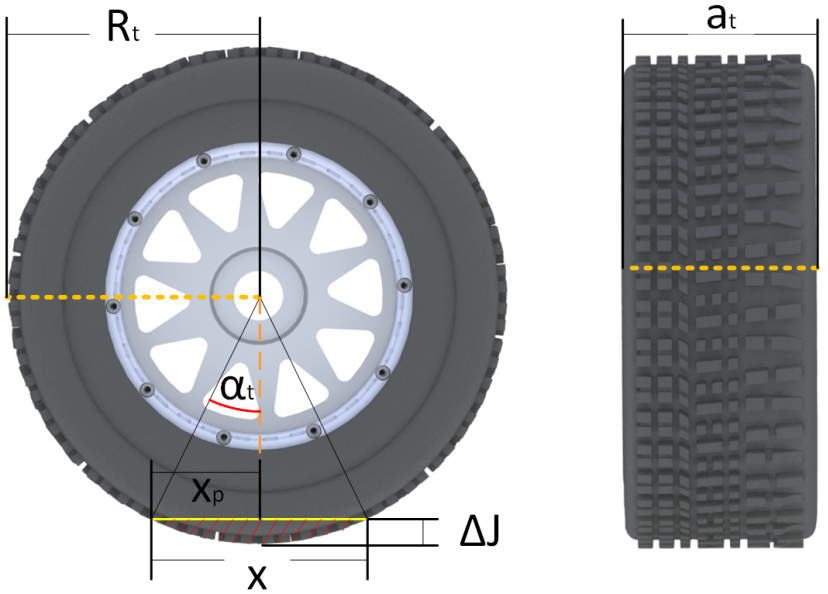


Figura 3.3: Descripción geométrica de la llanta.

El área de contacto del neumático como ya se mencionó es representado

por una figura rectangular (Ecuación 3.1).

$$A_c = a_t \cdot x \tag{3.1}$$

Donde:

A_c	Área de contacto de la llanta
a_t	Ancho del área de contacto de la llanta
x	Largo del área de contacto de la llanta

Dado que no se considera deformación en el ancho de la llanta, el largo del neumático es la única variable a calcular. Para obtener el largo del neumático es necesario el siguiente desarrollo matemático. Usaremos las identidades trigonométricas para calcular el valor del angulo de corte de la llanta, ver ecuación 3.2.

$$\alpha_t = \arccos\left(\frac{R_t - \Delta J)}{R_t}\right) \tag{3.2}$$

Donde:

$alpha_t$	Ángulo de corte de la llanta
R_t	Radio de la llanta
ΔJ	Distancia comprimida de la llanta

La distancia comprimida de la llanta está directamente relacionada con la geometría del terreno, pero así mismo está relacionada con la posición del neumático, esta puede cambiar debido a la dinámica del vehículo o alguna perturbación.

$$\Delta J = Q - \Delta z \tag{3.3}$$

Donde:

Q	Cambio de la altura del terreno
ΔZ	Cambio en la altura de la llanta

$$x_p = R_t \cdot \sin(\alpha_t) \tag{3.4}$$

x_p Mitad del largo del área de contacto

$$x = 2 \cdot x_p \tag{3.5}$$

Aplicando álgebra a las ecuaciones 3.2, 3.3, 3.4 y 3.5, dan lugar a la ecuación 3.6

$$x = 2R_t \sin\left[\arccos\left(\frac{R_t - \Delta J)}{R_t}\right)\right] \tag{3.6}$$

Por último, sustituimos el valor del largo del área de contacto en la ecuación 3.1 y obtenemos la ecuación 3.7.

$$A_c = 2a_t R_t \sin\left[\arccos\left(\frac{R_t - \Delta J)}{R_t}\right)\right] \tag{3.7}$$

Donde:

A_c	Área de contacto de la llanta
a_t	Ancho del área de contacto de la llanta
x	Largo del área de contacto de la llanta

Modelo de la suspensión

El sistema de suspensión tiene como principal tarea absorber las perturbaciones generadas por las irregularidades en el camino, en general el sistema de suspensión se compone por elementos mecánicos como son barras, resortes y amortiguadores. Este sistema se encuentra entre el chasis y las llantas, el sistema otorga control al vehículo así como confort al usuario. El modelo de la suspensión detalla el cambio en la geometría que sufre la suspensión con respecto al cambio de terreno, por lo tanto; se debe considerar los cambios que sufre la suspensión como variable a seleccionar.

Modelo del sistema de frenado

El sistema de frenado tiene el objetivo de disminuir la velocidad del vehículo a petición del conductor, transforma la energía cinética del vehículo en calor. Básicamente el sistema de frenado transmite la presión que ejerce el conductor al pedal de freno, a través de un fluido que amplifica dicha presión, con el objetivo de reducir la velocidad del vehículo con un esfuerzo mínimo.

Modelo del sistema de propulsión

El sistema de propulsión o comúnmente llamado tren de potencia, tiene el propósito de cambiar el estado de movimiento del vehículo, esto lo logra

haciendo uso del combustible almacenado en el tanque, con el fin de generar energía motriz para mover el vehículo. Este sistema por lo general contiene los siguientes elementos; motor, sistemas de transmisión, baterías y tanque de combustible.

Modelo del sistema de dirección

El sistema de dirección tiene el objetivo de orientar las ruedas delanteras para que el vehículo tome la trayectoria deseada por el conductor. Para que el conductor no tenga que realizar esfuerzo en la orientación de las ruedas, el vehículo dispone de un mecanismo de asistencia.

Modelo del chasis

El chasis es una estructura multipropósito, principalmente es la encargada de sostener y dar rigidez al vehículo, el chasis recibe todas las cargas y esfuerzos. El chasis de un vehículo consta de un armazón en el cual se sujeta e integra los componentes de todos los sistemas. El modelo del chasis está formado por; cálculo de momentos generados por el modelo de la suspensión, cálculo de momentos producidos por el modelo del sistema de propulsión y frenado, cálculo de momentos generados por el sistema de dirección y el peso del vehículo, todas estás fuerzas influyen en el chasis ya sea en la altura del mismo o en sus ángulos de cabeceo y alabeo.

3.1.2. Lista de variables de entrada

De la Figura 3.4 podemos notar que el área de contacto está relacionado con los cambios de aceleración del automóvil, así como la orientación de las llantas. El modelo matemático antes descrito fue desarrollado para un vehículo de tipo SAE baja.

El conjunto de variables obtenidas son las siguientes: la aceleración de las llantas en el eje z (a_t_1, a_t_2, a_t_3, a_t_4), el desplazamiento de la suspensión en cada amortiguador (x_s_1, x_s_2, x_s_3, x_s_4), el ángulo de cabeceo y balanceo lateral del chasis (Φ y Θ respectivamente) y por último el ángulo de la dirección (Ω). Para futuras referencias el número de cada llanta empieza de la frontal izquierda y continua en contra del sentido de las manecillas del reloj.

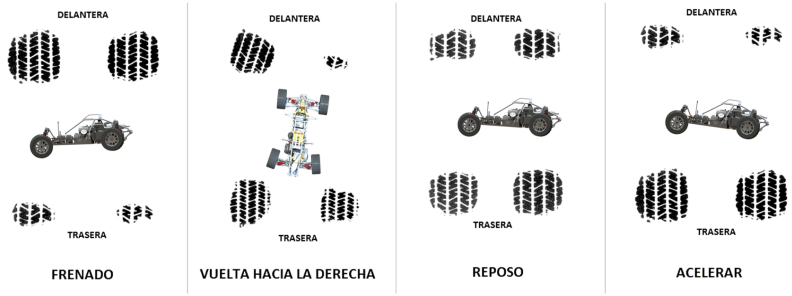


Figura 3.4: Diferentes áreas de contacto en situaciones de reposo, aceleración, frenado y vuelta.

Aceleración de las llantas en el eje *z* [a_s_{llanta}]

Esta variable no tiene relación alguna con el giro del neumático, se refiere a la aceleración que tienen las llantas debido a un cambio repentino en la distancia entre los neumáticos y el terreno. En la Figura 3.5(a) podemos notar que un cambio en el terreno representa un cambio en la aceleración del neumático.

Desplazamiento de la suspensión en cada amortiguador [x_t_{llanta}]

La geometría de la suspensión tiene una relación directa con el cambio de área de contacto de las llantas. El elemento de la suspensión elegido fueron los amortiguadores dado que son la parte de la suspensión que sufren los cambios más significativos. En la Figura 3.5(a) se nota de manera física el cambio de geometría del amortiguador.

Ángulo de cabeceo del chasis [Φ]

Es uno de los tres ángulos de navegación del auto, este ángulo representa la inclinación en el eje y del automóvil (ver Figura 3.5(b)), la información que provee esta variable es el cambio en la distribución del peso del vehículo, el cual influye en el cambio del área de contacto.

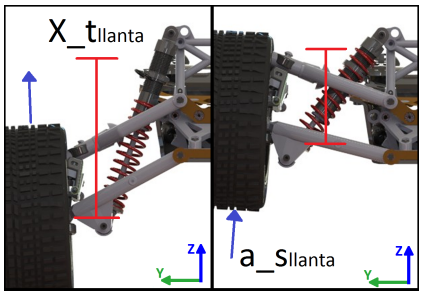
Ángulo de balanceo lateral del chasis [Θ]

Al igual que el ángulo de cabeceo, el ángulo de balanceo lateral es otro de los ángulos de navegación, este ángulo representa la inclinación en el eje x del

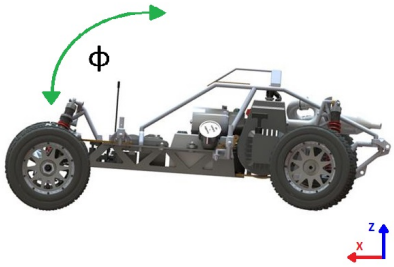
automóvil (3.5(c)), en complemento con el ángulo de cabeceo, esta variable nos proveerá información acerca del cambio de distribución de peso lateral, lo que implica un cambio en el área de contacto.

Ángulo de la dirección de los neumáticos [Ω]

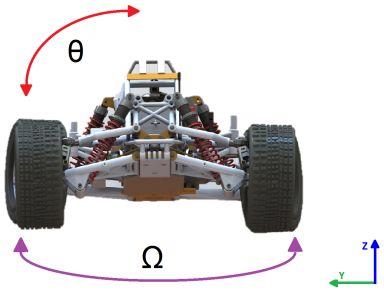
Como se puede apreciar en la Figura 3.4, el cambio en la orientación de las llantas delanteras provocan un cambio significativo en la forma en que los neumáticos entran en contacto con el terreno, tanto de los delanteros como los traseros. Por esta razón, el ángulo de la dirección (ver Figura 3.5(c)) es elegido como variable de entrada del Sensor Virtual.



(a) Zona del amortiguador



(b) Ángulo de cabeceo del chasis



(c) Vista frontal del vehículo

Figura 3.5: Descripción física de algunas variables de entrada.

3.2. Adquisición de Datos

La adquisición de datos está acotada a la frecuencia de muestreo y a la lectura sincronizada de los datos, en otras palabras, la recopilación de datos depende de cuan rápido podemos leer las variables y que sean leídas todas en el mismo instante. La arquitectura de la instrumentación puede lograr la lectura sincronizada de las once variables, además los microprocesadores actuales permiten una frecuencia de muestreo mayor a 100 KHz. El problema es la frecuencia en que el área de contacto puede ser medida, en [7] se muestra la metodología para medir el área de contacto de un neumático, cómo se puede apreciar, necesita una cámara fotográfica que en su mayoría posen frecuencias de muestreo de 30 Hz. Esta frecuencia de muestreo es muy baja. Considerando la frecuencia de muestreo de 30 Hz y una velocidad constante de 50 $\frac{Km}{hr}$, la distancia de predicción es de 46.30 cm. Esta distancia de predicción no es adecuada para el sensor ya que cualquier perturbación menor a 463 mm no será tomada en cuenta por el sensor.

Figura 3.6: Pista objetivo.

Este trabajo obtendrá las mediciones basado en el modelo desarrollado en [13], este es un modelo dinámico integral de un vehículo con los 6 subsistemas ya mencionados en la sección 3.1.1. El modelo recibe como entradas, la velocidad de desplazamiento y la forma del terreno para cada neumático. El modelo permite una frecuencia máxima de muestreo de $5Khz$, suponiendo que el vehículo mantiene una velocidad constante de 50 $\frac{Km}{hr}$, la distancia de predicción es de 2.8 mm, está distancia de predicción es mejor que la que se obtiene por la instrumentación, ya que el sensor tomaría en cuenta a todos

los eventos mayores a 2.8 mm. Las mediciones obtenidas por el modelo, son normalizadas para que la red neuronal tenga un mejor desempeño.

La pista de prueba consta de una serie de topes reductores de velocidad alternados (ver Figura 3.6). El vehículo mantiene una velocidad constante de 50 $\frac{Km}{hr}$, el primer reductor de velocidad es para el neumático delantero izquierdo, después de haber pasado el reductor, el neumático delantero derecho golpea el reductor de velocidad, y así sucesivamente. La simulación consta de $6s$, en los los neumáticos golpean al menos una vez un reductor.

Cabe mencionar que la red neuronal es entrenada para este tipo de pista de pruebas, si se quiere abarcar todos los casos posibles, es necesario diseñar una pista de prueba capaz de simular todos los estados físicos y mecánicos del vehículo.

3.3. Acoplamiento del modelo regresor

Existen distintos tipos de modelos de caja negra que sirven para aproximar una función, algunos ejemplos de ellos son; maquinas de soporte vectorial, métodos estadísticos multivariantes, métodos adaptativos, redes neuronales y métodos híbridos. Las características del problema definen el modelo a elegir. Dado que nuestro problema es un sistema dinámico complejo, que además requiere un modelo que permita predicción de datos, se eligió el modelo de redes neuronales recurrentes (RNN, por sus siglas en inglés) debido al gran desempeño al modelar sistemas dinámicos complejos, además del desempeño en la predicción de diversos sistemas [6].

En el sección 2.2.2, se dan las bases del funcionamiento de una red neuronal. Como ya se mencionó, la facultad de predicción de una RNN es atribuido a las conexiones de retroalimentación, estas conexiones permiten a la RNN emular series de tiempo.

3.3.1. Modelo regresor

La principal característica de una red neuronal recurrente es que contiene al menos una conexión de retroalimentación en su arquitectura. Esto permite a la red neuronal poder aprender secuencias de datos. Las arquitecturas más usadas de RNN consisten en un perceptrón multicapa, al cual se le añaden conexiones de retroalimentación. Estas conexiones explotan e incrementan las cualidades de modelado no-lineal del perceptrón multicapa. Un ejemplo de estas redes es la red neuronal no lineal autorregresiva con entrada exógena (NARX, por sus siglas en inglés), debido a las características de esta red, tiene la habilidad de simular lineas de tiempo. Una de las ventajas en la arquitectura de la red NARX es el acoplamiento de bloques de líneas de retardo (TDL, por sus siglas en inglés). Estos bloques de retardo se encuentran implementados tanto en la entrada de la red (TDL_{in}^{n}), como en la retroalimentación (TDL_{fb}^{n}). En la ecuación 3.8 se encuentra la arquitectura del vector de TDL, tanto para entrada de de la red como de la retroalimentación. Como se puede observar el tamaño del bloque de retroalimentación puede ser del tamaño que uno desee, aumentar el tamaño de estos bloques hará más sencilla la labor de modelar series de tiempo, pero la consecuencia es aumentar el tamaño de la matriz de pesos que conlleva a un mayor tiempo

de cómputo.

$$
\begin{aligned}
TDL_{in}^{n}(t) &= \begin{bmatrix} p(t-1) \\ p(t-2) \\ \vdots \\ p(t-n) \end{bmatrix} \\
TDL_{fb}^{n}(t) &= \begin{bmatrix} a^{M}(t-1) \\ a^{M}(t-2) \\ \vdots \\ a^{M}(t-n) \end{bmatrix}
\end{aligned}
\tag{3.8}
$$

Donde:

$p(t)$	Vector de entrada en el tiempo t
$a^M(t)$	Vector de salida de la NARX en la última capa M en t tiempo
n	Número de líneas de retardo en el bloque

La Figura 3.7 muestra la arquitectura de la NARX elegida para este trabajo, donde: R^1 es la longitud del vector de entrada; S^1 y S^2 son el número de neuronas de la primer capa y la segunda capa respectivamente; $n^1(t)$ y $n^2(t)$ son la señal de entrada de las funciones de transferencia f^1 y f^2 respectivamente.

Como se explicó en la sección 2.2.2, las dimensión de las matrices de pesos y bías están relacionadas con la cantidad de neuronas. Para calcular la dimensión de cada matriz, es necesario proponer una arquitectura, la arquitectura elegida tiene las siguientes características: la dimensión de los bloques de retardo es de tres tiempos cada uno ($TDL_{in}^{n} = TDL_{fb}^{n} = 3$); veinticinco neuronas para la primer capa ($S^1 = 25$); la dimensión del vector de entrada son las once variables que se eligieron ($R^1 = 11$); el número de neuronas de la capa de salida son las cuatro áreas de contacto, una por cada llanta ($S^2 = 4$); por último, las funciones de transferencia elegidas para la red son *logsig* para ambas capas ($f^1(n^1(t)) = logsig(n^1(t))$ y $f^2(n^2(t)) = logsig(n^2(t))$)

Con los anteriores valores propuestos podemos obtener las dimensiones de las matrices de pesos y bías. En la Tabla 3.1 se reportan las dimensiones de cada matriz. La arquitectura propuesta da como resultado un total de 1254 elementos, entre ellos pesos sinápticos y bías. Es notable como el tamaño de los bloques de retardo tienen gran influencia en la dimensión de las matrices,

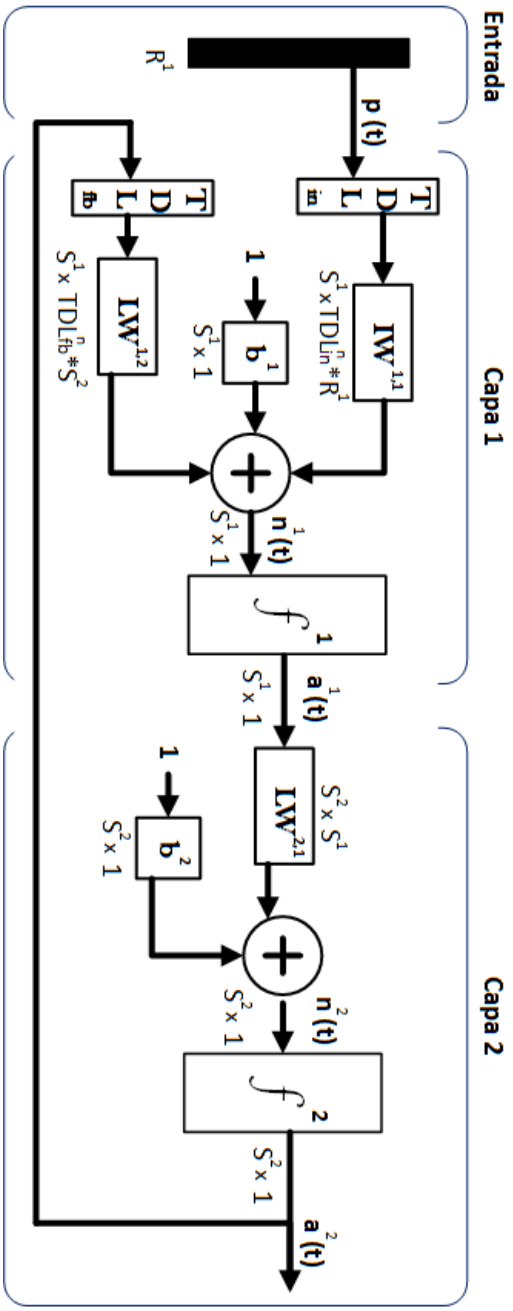


Figura 3.7: Arquitectura de la NARX Propuesta.

y como consecuencia de esta influencia, tiene una estrecha relación con la cantidad de elementos que podemos modificar en la NARX como se puede notar en la Tabla 3.1.

Tabla 3.1: Dimensiones de las matrices de pesos y bías

Matriz	**Ecuación**	**Dimensión**	**Elementos**
$\mathbf{IW^{1,1}}$	$S^1 \times TDL_{in}^n \cdot R^1$	25×33	825
$\mathbf{LW^{1,2}}$	$S^1 \times TDL_{fb}^n \cdot S^2$	25×12	300
$\mathbf{b^1}$	$S^1 \times 1$	25×1	25
$\mathbf{LW^{2,1}}$	$S^2 \times S^1$	4×25	100
$\mathbf{b^2}$	$S^2 \times 1$	4×1	4
			1254

La ecuación 3.9 describe como calcular la salida de la red neuronal diseñada ($a^2(t)$). Para más información acerca de las funciones de transferencia (f^1 o f^2) ver [19].

$$\begin{aligned} \mathbf{a}^1(t) &= \mathbf{logsig}\left(\mathbf{IW}^{1,1}\mathbf{TDL}_{in}^n(t) + \mathbf{LW}^{1,2}\mathbf{TDL}_{fb}^n(t) + \mathbf{b}^1\right) \\ \mathbf{a}^2(t) &= \mathbf{logsig}\left(\mathbf{LW}^{2,1}\mathbf{a}^1(t) + \mathbf{b}^2\right) \end{aligned} \tag{3.9}$$

Donde:

- $\mathbf{IW}^{1,1}$ Matriz de pesos del vector de entrada de la red
- $\mathbf{LW}^{1,2}$ Matriz de pesos entre la capa 1 y la capa 2
- $\mathbf{LW}^{2,1}$ Matriz de pesos entre la capa 2 y la capa 1
- $\mathbf{b}^1$ Vector de bías de la capa 1
- $\mathbf{b}^2$ Vector de bías de la capa 2

El desempeño de la NARX puede ser cuantificado como la diferencia entre el valor deseado y el valor obtenido por la red neuronal. A esta diferencia se le conoce como función de error y es descrita en la ecuación 3.10. Esta función de error calcula el promedio de la diferencia entre la salida de la red neuronal y el valor de salida deseado en un tiempo t.

$$e(t) = \frac{1}{N_T} \sum_{i \in N_T} \left| \mathbf{t}(t) - \mathbf{a}^M(t) \right| \tag{3.10}$$

Donde:

$\mathbf{a}^M(t)$	Vector de salida de la red neuronal
$\mathbf{t}(t)$	Vector de salida deseado
$N_T =$	Número de elementos en el vector de salida

La función del error $e(t)$ es una diferencia puntual. Para medir el desempeño en toda la linea de tiempo es necesario calcular la función del error a lo largo de toda la linea. La ecuación 3.11 describe como calcular el error de la linea de tiempo.

$$E = \frac{1}{N_{lt}} \sum_{t \in N_{lt}} e(t) \tag{3.11}$$

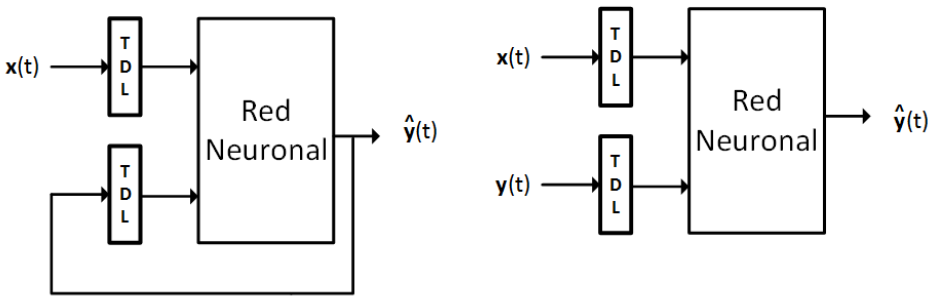
Donde:

N_{lt}	Número de datos en la línea de tiempo

3.3.2. Entrenamiento por gradiente

Conocido por retropropagación del error (*Backpropagation* en inglés), es un algoritmo de entrenamiento del tipo supervisado, es decir necesita de un conjunto de datos para ser implementado. Los datos de entrenamiento consisten de duplas de valores numéricos, una componente son las entradas y el otro la salida deseada o *"target"*. Este método es ideal para entrenar redes del tipo estático. Las redes de tipo estático son aquellas en las que las conexiones entre cada capa no forman un ciclo. El tipo de red que se eligió contiene transiciones de retroalimentación, en otras palabras no es una red estática, por lo tanto no es candidata para ser resuelta por *Backpropagation*.

En [18], se propone realizar algunas modificaciones a la red para que esta pueda ser entrenada por medio del gradiente. La arquitectura simplificada de la NARX se muestra en la Figura 3.8(a), la principal modificación es desconectar la retroalimentación de la red neuronal como se aprecia en la Figura 3.8(b). La premisa en que nos basamos poder desconectar la retroalimentación es la siguiente: **El vector de retroalimentación podrá sustituir los valores deseados de la red en lugar de los valores de salida**, esto se puede hacer, debido a que el gradiente busca igualar el valor de salida de la red ($\mathbf{a}^M(t)$) con el valor deseado($\mathbf{t}(t)$), por lo tanto, en algún momento el valor de salida será el deseado y se cumplirá la premisa.



(a) Red conectada. (b) Red desconectada.

Figura 3.8: Tipos de conexión de la retroalimentación.

El objetivo del algoritmo de entrenamiento por gradiente es encontrar el punto mínimo que represente la diferencia entre la salida de la red y el target, esto se logra por medio de ecuaciones diferenciales. La dirección del gradiente modifica los valores de pesos y bías para obtener un nuevo valor de error, la hypersuperficie de error adquiere una mayor dimensión entre más neuronas y capas se agreguen a la red neuronal. El método del gradiente presenta dificultades debido a la presencia de valles espurios en la hypersuperficie del error.

Estos valles son espurios, porque no están relacionados con el problema del entrenamiento. Dependen únicamente de las entradas de red, de los estados de red iniciales y de la arquitectura de la red. Existen dos tipos principales de valles espurios en la superficie de error. Los valles de raiz que dependen de la secuencia de entrada y los valles de arquitectura que aparecen independientemente de la secuencia de entrada y son características fundamentales de la arquitectura de red. Los valles espurios son un grave problema en el proceso del entrenamiento, debido a que estancan la búsqueda del verdadero mínimo global [23].

3.3.3. Entrenamiento por optimización

El proceso de entrenamiento puede ser considerado como un problema de optimización en un espacio de búsqueda continuo. El objetivo del aprendiza-

je, consiste en encontrar la configuración de pesos y bías que correspondan al mínimo global de la función de error. La mayor parte de los entrenamientos concluyen en un mínimo local "suficientemente bueno". Al considerar el proceso de entrenamiento como un problema de optimización, los valles espurios son interpretados como mínimos locales. Dado este planteamiento, los métodos de optimización estocástica pueden presentar una gran ventaja respecto a los métodos basados en el gradiente, esto es debido al mecanismo de exploración que los métodos estocásticos presentan, esto les permite sortear los mínimos locales y no quedar atrapados en ellos.

En este trabajo, proponemos como método de entrenamiento una versión modificada del algoritmo de entrenamiento basado en evolución diferencial que fue propuesto inicialmente para redes con propagación hacia adelante en [22], el cual se aplicó a una red neuronal recurrente.

Algoritmo de ED para entrenamiento de una RNN

El algoritmo de evolución diferencial (ED) puede ser clasificado como un algoritmo evolutivo para espacios de búsqueda continua. Por esta razón, este puede ser aplicado para optimizar los pesos y bías de una red neuronal recurrente. En un entrenamiento estándar, las entradas y el valor deseado de la red son valores conocidos, por lo tantos son valores fijos. Los valores de pesos sinápticos y bías cambian con el objetivo de hacer un relación entre los valores de entrada de la red y la salida deseada. El valor que calificará el desempeño de la red neuronal es el error de la linea de tiempo, ver ecuación 3.11.

La principal ventaja de usar un algoritmo no basado en el gradiente es que la NARX puede ser entrenada sin hacerle ninguna modificación a su arquitectura, es decir se entrena totalmente conectada (Figura 3.8(a)), el algoritmo de evolución diferencial fue modificado para incluir el proceso de entrenamiento, validación y prueba. El proceso de entrenamiento se realiza con la red neuronal conectada, contrario a los procesos de validación y prueba que se realizan de manera desconectada. La razón se discutirá en la sección 3.3.4.

El objetivo de un proceso de optimización es minimizar la función objetivo, en nuestro caso, minimizar el valor del error de la linea de tiempo (ecuación 3.11). Esto se logrará modificando los valores de los pesos sinápticos y bías. Como se detalló en la sección 2.2.1, el algoritmo de evolución diferencial opera a partir de una población de soluciones candidatas. Cada

posible solución representa a un individuo de la población. Por lo tanto cada individuo representa a un vector que contiene a todos los bías y pesos sinápticos, recordemos que para la arquitectura propuesta, cada individuo contiene 1254 elementos (ver Tabla 3.1). A la longitud del vector del individuo se le conoce como dimensión del problema D. Las cuatro fases del proceso de entrenamiento son las siguientes:

Inicialización.- La población (N_p) es inicializada con valores aleatorios entre $[-1, 1]$. Una mayor población hará que el algoritmo tenga mayores propiedades de exploración, pero esto requerirá mayor tiempo de cómputo.

Mutación.- Para una generación G, para cada individuo p (vector objetivo), se mutan todos los pesos y bías de cada neurona n, entrada i y capa L. La operación de mutación se rige por las ecuaciones 3.12 y 3.13. A el vector mutado se le conoce como vector donante.

$$v^{L,n,i}_{w(p,G)} = w^{L,n,i}_{r1,G} - F * \left(w^{L,n,i}_{r2,G} - w^{L,n,i}_{r3,G}\right) \tag{3.12}$$

$$v^{L,n}_{b(p,G)} = b^{L,n}_{r1,G} - F * \left(b^{L,n}_{r2,G} - b^{L,n}_{r3,G}\right) \tag{3.13}$$

Donde:

p, $r1$, $r2$ y $r3$	Indices aleatorios y diferentes que pertenecen a N_p
$v^{L,n,i}_{w(p,G)}$	Vector donante de pesos sinápticos
$v^{L,n}_{b(p,G)}$	Vector donante de bías
F	Factor de escala entre $[0, 2]$.

Se usará la función *DE/rand/1/bin* debido a que esta función esta orientada a buscar distintas soluciones en el espacio de búsqueda. En diferencia de alguna función que tome en cuenta al mejor individuo.

Cruza.- La etapa de recombinación o cruza, incorpora soluciones exitosas de generaciones anteriores. El vector de prueba $u^{L,n,i}_{w(p,G)}$ está compuesto por elementos del vector donante y el vector objetivo. Cada elemento de los vectores de pesos y bías pasa por la operación de cruza, las ecuaciones de cruza son las número 3.14 y 3.15.

$$u_{w(p,G)}^{L,n,i} = \begin{cases} v_{w(p,G)}^{L,n,i} & \text{si } rand \leq CR \\ w_{p,G}^{L,n,i} & \text{otro caso} \end{cases} \tag{3.14}$$

$$u_{b(p,G)}^{L,n} = \begin{cases} v_{b(p,G)}^{L,n} & \text{si } rand \leq CR \\ b_{p,G}^{L,n} & \text{otro caso} \end{cases} \tag{3.15}$$

Donde:

$rand$	Número aleatorio uniformemente distribuido entre $[0, 1]$
CR	Probabilidad de cruza elegida entre $[0, 1]$

Selección.- El vector objetivo $w_{p,G}$ es comparado con el vector de prueba $u_{w(p,G)}^{L,n,i}$, ambos vectores son evaluados mediante la función de desempeño, el que presente un mejor desempeño asegurará un lugar en la siguiente generación. Las ecuaciones de la etapa de selección son la número 3.16 y 3.17.

$$w_{p,G+1} = \begin{cases} u_{w(p,G+1)} & \text{si } E(u_{w(p,G)}, u_{b(p,G)}) < E(w_{p,G}, b_{p,G}) \\ w_{p,G} & \text{otro caso} \end{cases} \tag{3.16}$$

$$b_{p,G+1} = \begin{cases} u_{b(p,G+1)} & \text{si } E(u_{w(p,G)}, u_{b(p,G)}) < E(w_{p,G}, b_{p,G}) \\ b_{p,G} & \text{otro caso} \end{cases} \tag{3.17}$$

Donde:

E	Función de desempeño o función de error (ecuación 3.11).

El proceso de entrenamiento es cubierto casi en su totalidad por la adaptación hecha al algoritmo de evolución diferencial. Para que el algoritmo de evolución diferencial pueda detectar un sobreajuste en los datos, es necesario anexar una sección a nuestro algoritmo. El sobreajuste de los datos ocurre cuando la red neuronal aprende el conjunto de datos de entrenamiento pero falla al probar con nuevos datos. Para más información, ver [19]. Para enfrentar este problema, al algoritmo de evolución diferencial se le añadió el módulo de detección temprana, que consiste de un nuevo conjunto de datos, un nuevo criterio de paro y una generación de validación.

Validación.- Cada numero de generaciones N_{G_v}, el conjunto de datos de entrenamiento es cambiado por el conjunto de datos de validación. En la generación de validación G_v, los individuos no presentan mutación cruza o selección. La generación de evaluación consiste en obtener el desempeño de cada individuo con el conjunto de datos de validación (ecuación 3.11). Una vez obtenido el desempeño de cada individuo, se compara el promedio de esta generación con la generación pasada, si el número de incrementos consecutivos es mayor al número permitido, el algoritmo es detenido. El número de incrementos consecutivos es puesto a cero si el error en la generación es menor que en la generación anterior. La ecuación 3.18 describe el proceso de validación.

$$i_O = \begin{cases} i_O + 1 & \text{si } E_{Val}(G_v - 1) < E_{Val}(G_v) \\ 0 & \text{otro caso} \end{cases} \tag{3.18}$$

Donde:

$E_{Val}(G_v)$	Error de validación promedio de la población (ecuación 3.19)
i_O	Contador de incrementos consecutivos

$$E_{Val} = \frac{1}{N} \sum_{p \in N} E(p) \tag{3.19}$$

Donde:

N	Tamaño de la población
p	Individuo de la población
E	Función de desempeño o función de error (ecuación 3.11).

Criterios de paro.- El proceso de optimización es un proceso iterativo, por lo que se usarán las siguientes condiciones para detener el algoritmo:

1. *Máximo número de generaciones.* El algoritmo es detenido si el número de generación actual G rebasa el número máximo de generaciones G_{max}.

2. *Detección temprana de sobreajuste.* Es activado si el contador de incrementos consecutivos i_O, excede el número máximo permitido N_O.

3. *Error mínimo.* Si algún individuo alcanza un valor de error menor al valor deseado E_{min}, el algoritmo se detiene.

Prueba.- Con el objetivo de cuantificar el desempeño del entrenamiento, una vez detenido el algoritmo, se elige al mejor individuo de la población p_{best}. Una vez elegido, se usan los valores finales de pesos y bías. Estos valores son puestos a prueba con una conjunto de datos distinto al de entrenamiento y validación, se usa la ecuación 3.19 para calcular el valor del error de prueba $E_{test}(G_{last})$. El pseudocódigo del algoritmo de evolución diferencial para entrenamiento de redes neuronales recurrentes es presentado en el algoritmo 8.

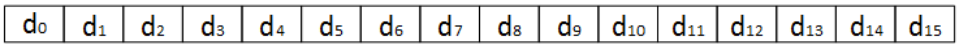
Algoritmo 8: Algoritmo de evolución diferencial para entrenamiento

mientras *no se cumpla ningún criterio de paro* **hacer**
  si *la generación es de validación* **entonces**
    para cada *p individuo de la población N* **hacer**
      Calcular la función de error $E(w_{p,G}, b_{p,G})$
    fin
    Calcular el error promedio de la población $E_{Val}(G_v)$
    si $E_{Val}(G_v)$ *es mayor que* $E_{Val}(G_v - 1)$ **entonces**
      incrementar en uno a i_O
    en otro caso
      igualar a cero el contador i_O
    fin
  en otro caso
    para cada *p individuo de la población N* **hacer**
      Asignar números enteros aleatorios para $r1$, $r2$, $r3$ acotados entre $[1, N]$ tal que el individuo $p \neq r1 \neq r2 \neq r3$.
      para cada *entrada i de la neurona n de la capa L* **hacer**
        $v^{L,n,i}_{w(p,G)} = w^{L,n,i}_{r1,G} - F * \left(w^{L,n,i}_{r2,G} - w^{L,n,i}_{r3,G}\right)$
        $v^{L,n}_{b(p,G)} = b^{L,n}_{r1,G} - F * \left(b^{L,n}_{r2,G} - b^{L,n}_{r3,G}\right)$
      fin
      para cada *entrada i de la neurona n de la capa L* **hacer**
        si *rand(0,1)* $\leq CR$ **entonces**
          $u^{L,n,i}_{w(p,G)} = v^{L,n,i}_{w(p,G)}$
        en otro caso
          $u^{L,n,i}_{w(p,G)} = w^{L,n,i}_{p,G}$
        fin
        si *rand(0,1)* $\leq CR$ **entonces**
          $u^{L,n}_{b(p,G)} = v^{L,n}_{b(p,G)}$
        en otro caso
          $u^{L,n}_{b(p,G)} = b^{L,n}_{p,G}$
        fin
      fin
    fin
    Calcular la función de error $E(w_{p,G}, b_{p,G})$ y $E(u_{w(p,G)}, u_{b(p,G)})$
    si $E(u_{w(p,G)}, u_{b(p,G)}) < E(w_{p,G}, b_{p,G})$ **entonces**
      $w_{p,G+1} = u_{w(p,G)}$
      $b_{p,G+1} = u_{b(p,G)}$
    en otro caso
      $w_{p,G+1} = w_{p,G}$
      $b_{p,G+1} = b_{p,G}$
    fin
  fin
fin
devolver $w_{p,G+1}, b_{p,G+1}$

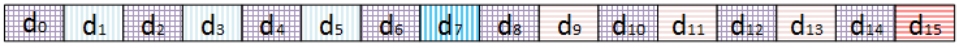
3.3.4. Preprocesamiento de datos

La importancia de la normalización de datos es casi tan relevante como el entrenamiento de la red neuronal, en [37] se demuestra la relación que tiene la normalización con el tiempo de entrenamiento de la red. Las unidades físicas de cada una de las variables es distinta, al igual que la dimensión de la magnitud, estas características hacen de la normalización una necesidad. Para el preprocesamiento se eligió una transformación lineal, la cual toma en cuenta el valor mínimo y máximo que es físicamente posible para cada entrada y salida de la red neuronal.

Como ya se mencionó, este trabajo obtendrá las mediciones basado en el modelo desarrollado en [13], este modelo permite una frecuencia máxima de muestreo de 5 Khz. Para el algoritmo de entrenamiento necesitamos tener tres conjuntos de datos, el conjunto de entrenamiento, validación y prueba. Para que el algoritmo pueda entrenar la red neuronal completamente conectada, el conjunto de entrenamiento debe ser un conjunto de datos continuo, es decir ser una linea de tiempo. Los otros dos conjuntos no necesariamente deben de ser lineas de tiempo, ya que el objetivo de estos puntos es evaluar el desempeño de la red neuronal en puntos diferentes a los valores de entrenamiento, por lo tanto la red neuronal no se probará conectada.



(a) Linea de Tiempo.



(b) Conjunto de entrenamiento, validación y prueba.

Figura 3.9: Conjunto de datos.

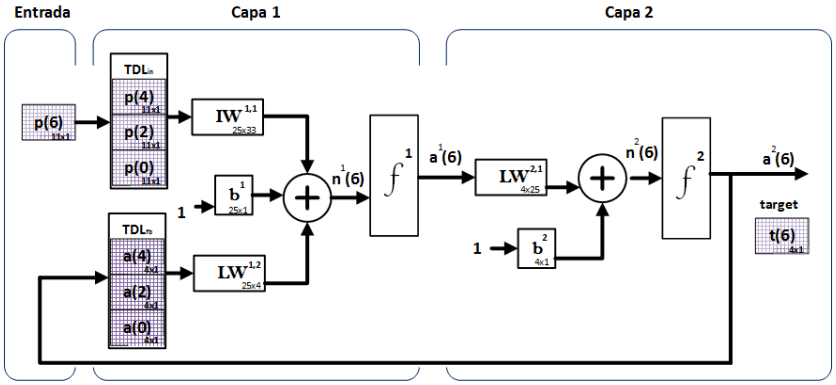
Con la idea de cumplir los requerimientos para formar los conjuntos de datos, se optó por tomar para el conjunto de entrenamiento, los datos obtenidos a una frecuencia de muestreo de 2.5 Khz. Esto es la mitad del total de los datos. Para el conjunto de validación y prueba, los datos se obtendrán de la misma frecuencia pero con un retraso de medio periodo de muestreo. Como ejemplo, en la Figura 3.9(a) se propone una linea de 16 datos con una frecuencia de muestreo de 5 Khz, el conjunto de entrenamiento debe ser el 80 % del número de datos total, dejando un 10 % para el conjunto

de validación y otro 10 % para el conjunto de prueba. Para este ejemplo, el conjunto de entrenamiento está formado por todos los datos que tengan la frecuencia de muestreo de 2.5 Khz, si estos ocho datos de entrenamiento los tomamos como el 80 % del todo, queda elegir a un dato para el conjunto de entrenamiento y uno para el conjunto de validación. En la Figura 3.9(b) los datos de entrenamiento se distinguen con un patrón de cuadrícula, el dato de validación tienen un patrón rallado vertical y finalmente el dato de prueba se distingue con un rallado horizontal.

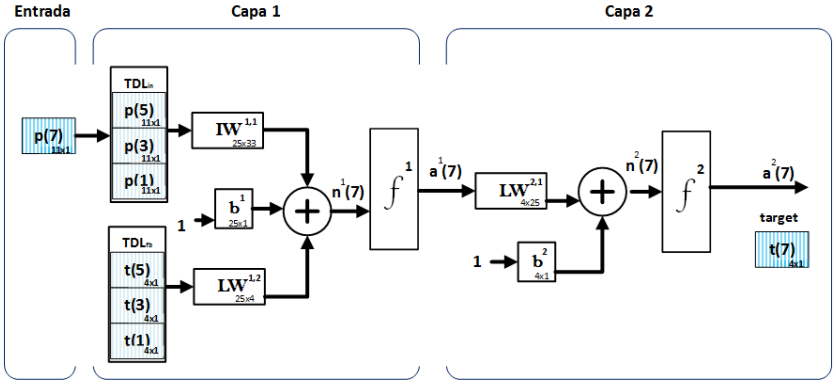
Recordemos que un bloque de dato (d_n) contiene los vectores de entrada y target de la señal. Los indicadores que están sobre el dato de entrenamiento seis, el dato de validación siete y el dato de prueba quince en la Figura 3.9(b), servirán como ejemplo para mostrar las diferencias entre la red neuronal conectada y desconectada. De acuerdo con la arquitectura elegida en la sección 3.3.1, el tamaño de ambos bloques de retardo es de 3. Por lo tanto la red neuronal necesita los datos en el tiempo $d(t)$, $d(t-1)$, $d(t-2)$ y $d(t-3)$.

El proceso de entrenamiento para el dato 6, requiere del valor de la salida calculada por la red neuronal de 3 tiempos anteriores, en la Figura 3.10(a) podemos observar la retroalimentación de los valores a la salida de la red neuronal. Por esta razón se requiere de una linea de tiempo para el entrenamiento ya que para el siguiente dato se requiere del valor de salida anterior.

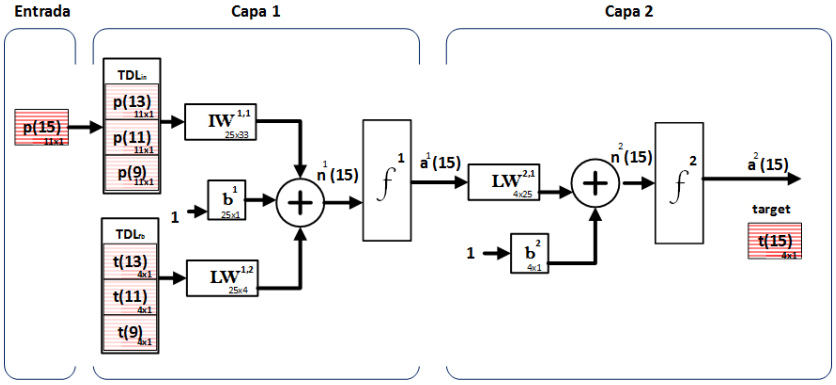
A diferencia del proceso de entrenamiento, como ya se mencionó, el proceso de validación y prueba requieren de puntos homogéneamente distribuidos en la linea de tiempo diferentes a los usados en el entrenamiento. En la Figura 3.9(b) los datos de validación siete y el dato de prueba quince serán usados como ejemplos. Sin importar si se trata de un punto de validación o de prueba, ambos necesitan de los tres datos anteriores (recordemos que el valor del bloque de retraso es de 3), estos puntos requieren de las últimas tres salidas de la red neuronal, para obtener la anterior salida es necesario haber calculado la anterior y así sucesivamente. Se vuelve un tanto complejo calcular toda la linea de tiempo para obtener el valor de un punto, por esta razón se optó por usar la red desconectada para el caso de los datos de validación y prueba. En las Figuras 3.10(b) y 3.10(c), en lugar de llenar el bloque de retraso de la retroalimentación (TDL_{fb}), el bloque se llena de los valores deseados de la señal, esto permite calcular rápidamente el desempeño de la red neuronal en cada punto de prueba o validación.



(a) Entrenamiento para el sexto dato (d_6)



(b) Validación para el séptimo dato (d_7)



(c) Prueba para el quinceavo dato (d_{15})

Figura 3.10: Tipo de conexión para los tres distintos de conjuntos de datos.

Capítulo 4

Resultados experimentales

El sensor virtual necesita de un conjunto de datos para ser entrenado, el tamaño de este conjunto dependerá del tiempo que dure la pista y el horizonte de predicción, si nuestra pista dura 1 segundo pero queremos tener un horizonte de predicción de un milímetro (velocidad constante de 50 $\frac{Km}{hr}$), tendríamos aproximadamente 14 000 datos. A mayor número de datos, mayor el tiempo de cómputo necesario para entrenar la red neuronal. Por lo tanto el horizonte de predicción elegido es la diferencia entre anticiparse a una perturbación o reaccionar ante ella. A velocidad constante de 50 $\frac{Km}{hr}$ o 13.889 $\frac{m}{s}$, usando la frecuencia de muestreo máxima de 2.5 Khz, el horizonte de predicción máximo es de 5.55 mm (13.889 $\frac{m}{s}$/ 2.5Khz). Cabe recordar que el número de datos es la frecuencia de muestreo por el tiempo que deseamos mostrar.

4.1. Condiciones de los experimentos

El proceso de entrenamiento se logró usando el algoritmo de evolución diferencial (ED) de la sección 3.3.1. El algoritmo de evolución diferencial usa los siguientes tres parámetros: el factor de escala (F), la probabilidad de cruza (CR) y el tamaño de la población (N_p). El desempeño del algoritmo depende totalmente de estos parámetros. Los valores adecuados para el factor de cruza y la constante de mutación son 0.9 y 0.5 respectivamente, estos valores se eligieron debido a que presentan un excelente desempeño en problemas de altas dimensiones [15]. Para poder elegir el tamaño de la población del algoritmo, se hizo un entrenamiento en una arquitectura de prueba. En

esta prueba, se analizó el desempeño del algoritmo con un tamaño de población pequeño (5 individuos), este bajo número de individuos aporta una baja diversidad en las soluciones, disminuyendo la capacidad de exploración del algoritmo. En valores de población más altos (45 individuos), el desempeño de los individuos en promedio, es mejor que en poblaciones pequeñas, ya que aumentamos la diversidad en la población, el costo se ve reflejado en un mayor consumo computacional. Finalmente, se eligió una población de 25 individuos, este tamaño de población se encuentra balanceado entre diversidad y costo computacional.

Los valores finales de los parámetros del algoritmo de evolución diferencial, están reportados en la Tabla 4.1.

CR	F	N_p
0,9	0,5	25

Tabla 4.1: Parámetros del algoritmod de ED

Una vez fijados los parámetros del algoritmo de ED, la siguiente tarea es elegir la mejor arquitectura de la red NARX. Debido a los requerimientos del sensor, once variables físicas que mapean a las 4 áreas de contacto de los neumáticos, las únicas variables que tenemos para modificar son: número de capas de la red neuronal, número de neuronas en cada capa y tamaño de los bloques de retardo.

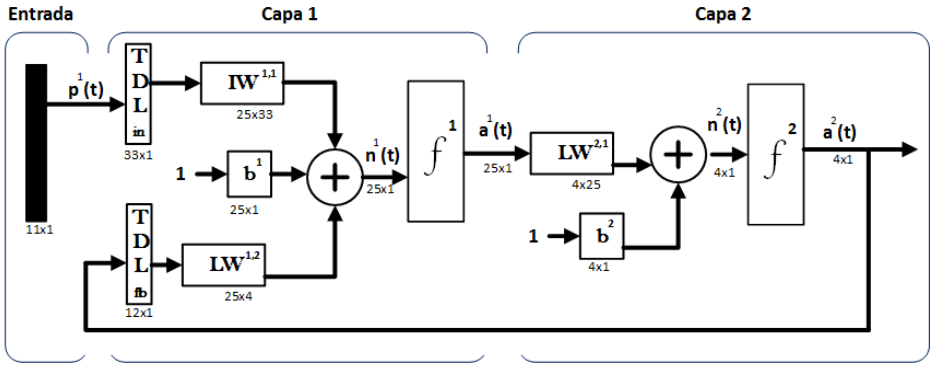


Figura 4.1: Arquitectura final de la NARX.

Para poder elegir una arquitectura de entre muchas combinaciones posibles, se realizaron distintos experimentos probando diferentes tipos de arquitecturas. En estos experimentos se fijó un valor de error y se eligió a la

arquitectura que alcance el error deseado con un costo de cómputo bajo. La arquitectura que obtuvo mejores resultados es la descrita en la sección 3.3.1. En la Figura 4.1, están detalladas las dimensiones de cada unas de las matrices así como los bloques de retardo. Las conclusiones obtenidas de estos experimentos serán discutidas en la sección 5.1.

La ecuación 4.1 desarrolla la manera de la salida de la red neuronal($a^2(t)$).

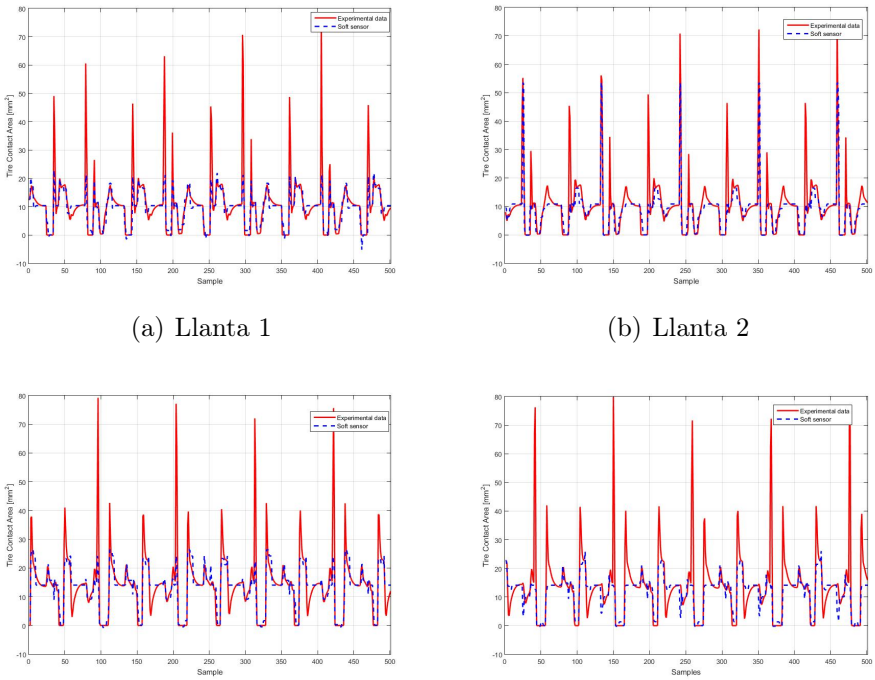
$$\begin{aligned} \mathbf{a}^1(t) &= \mathbf{logsig}\left(\mathbf{IW}^{1,1}\mathbf{TDL}^3_{in}(t) + \mathbf{LW}^{1,2}\mathbf{TDL}^3_{fb}(t) + \mathbf{b}^1\right) \\ \mathbf{a}^2(t) &= \mathbf{logsig}\left(\mathbf{LW}^{2,1}\mathbf{a}^1(t) + \mathbf{b}^2\right) \end{aligned} \tag{4.1}$$

Donde:

$\mathbf{IW}^{1,1}$	Matriz de pesos del vector de entrada de la red
$\mathbf{LW}^{1,2}$	Matriz de pesos entre la capa 1 y la capa 2
$\mathbf{LW}^{2,1}$	Matriz de pesos entre la capa 2 y la capa 1
$\mathbf{b}^1$	Vector de bías de la capa 1
$\mathbf{b}^2$	Vector de bías de la capa 2

4.2. Experimentos

A partir de las condiciones establecidas en la sección 4.1, en la Figura 4.2 se muestra la predicción hecha por un experimento de entrenamiento exitoso. Con el fin de analizar mejor la predicción hecha por la red, la Figura 4.3 muestra un acercamiento del resultado obtenido.



(a) Llanta 1

(b) Llanta 2

(c) Llanta 3

(d) Llanta 4

Figura 4.2: Predicción del área de contacto de los cuatro neumáticos.

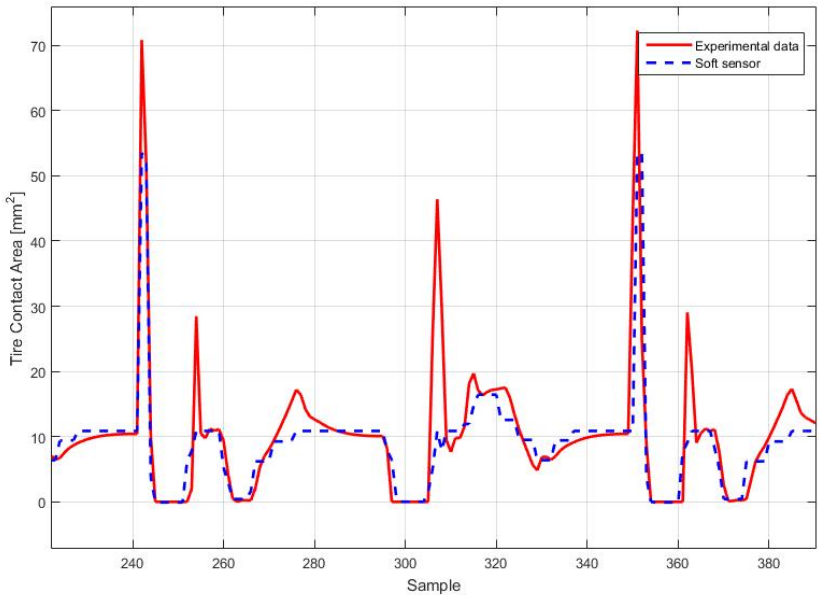


Figura 4.3: Acercamiento predicción neumático 2.

Para medir el desempeño del entrenamiento de la NARX, se miden tres tipos distintos de error:

El error de validación promedio de la población $E_{Val}(G_v)$, el error de entrenamiento del mejor individuo $E(w_{*,G}, b_{*,G})$ y el error de prueba del mejor individuo $E_{Pru}(w_{*,G}, b_{*,G})$. Estos errores tienen distintos propósitos.

En la Figura 4.4 podemos observar el comportamiento del error de validación. A pesar que este error presenta algunos aumentos, generalmente este error tiende a disminuirse. Para este experimento la generación de validación se calcula cada cincuenta generaciones de entrenamiento, y si el comportamiento del error presenta seis incrementos consecutivos, el entrenamiento es detenido. El valor del último error de validación promedio de la población fue de $E_{Val}(G_{vF}) = 3{,}196$ cm^2.

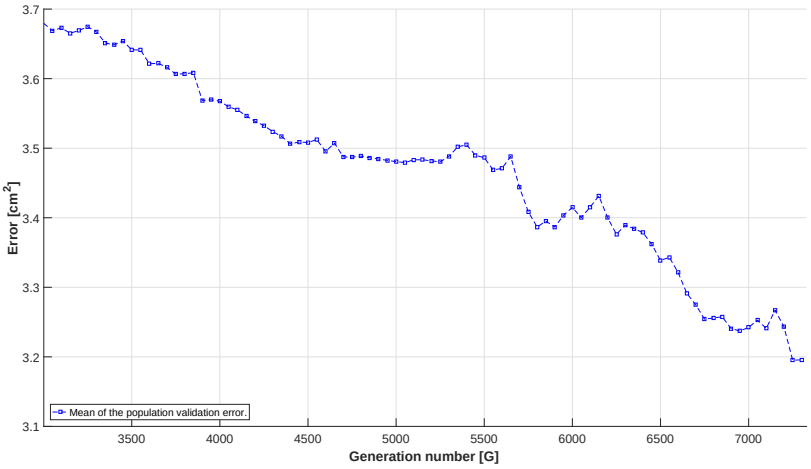


Figura 4.4: Comportamiento de los últimos errores de validación promedio de la población.

El comportamiento general de los tres errores descritos se muestra en la Figura 4.5. Dado a que este experimento fue exitoso, el motivo de paro fue haber alcanzado el error de entrenamiento establecido, el cual fue de 3 cm^2. El algoritmo se detiene una vez que un individuo alcanza este error, en cada generación se verifica cual es el mejor individuo y este se compara con el error establecido. El error de entrenamiento del mejor individuo fue de $E(w_{*,G}, b_{*,G}) = 2{,}981\ \text{cm}^2$.

Debido a que el algoritmo de optimización se enfoca en minimizar el error de entrenamiento, podemos observar como el error de validación oscila. Una vez alcanzado algún criterio de paro se calcula el error de prueba del mejor elemento. Para este experimento el valor del error fue $E_{Pru}(w_{*,G}, b_{*,G}) = 3{,}718\ \text{cm}^2$.

Además de los tres errores con los que se cuantifica el desempeño del entrenamiento, en la Figura 4.5 se grafica un cuarto error, este error nos ofrece mas información acerca del desempeño del mejor individuo de la población, el error de validación del mejor individuo $E_{Val}(w_{*,G}, b_{*,G})$. Es de esperar que este error siempre se mantenga por debajo del error de validación promedio de la población ($E_{Val}(G_v)$). Para este experimento el valor fue de $E_{Val}(w_{*,G}, b_{*,G}) = 3{,}106\ \text{cm}^2$.

El objetivo principal de la detección temprana de sobreajuste es evitar que

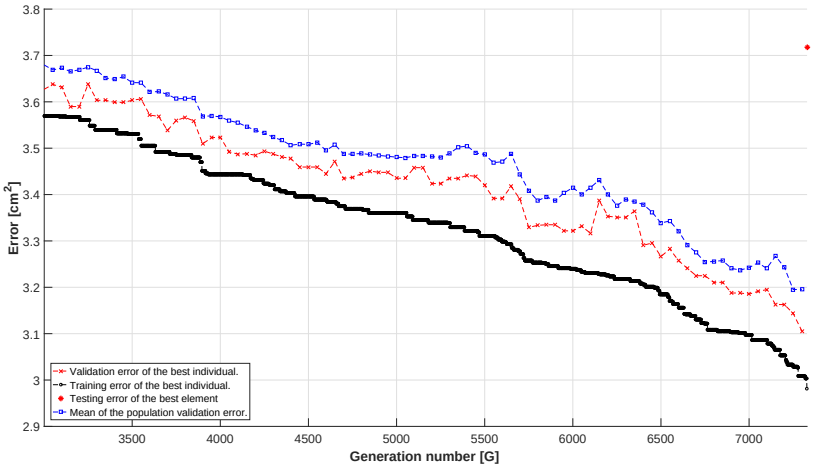


Figura 4.5: Comportamiento general de los errores.

el modelo empiece a sobre-entrenarse con los datos de entrenamiento. Con la intensión de demostrar las ventajas de contar con la detección temprana, se diseñó un conjunto de experimentos. Se ejecutaron cien experimentos con el algoritmo sin detección y otra centena de experimentos con el algoritmo de detección temprana activado. Las características para este experimento son iguales a las que se establecieron en la sección 4.1 pero con las siguientes diferencias; El error mínimo establecido es 2,5 cm^2, el máximo número de generaciones es de 25000, la generación de validación se calcula cada 250 generaciones y el algoritmo se detendrá si la población presenta cuatro incrementos consecutivos. El error mínimo establecido es casi imposible de ser alcanzado, la razón es que se busca forzar el entrenamiento de la NARX a través de más generaciones sin ser detenido.

Como resultado del experimento antes descrito, en la Tabla 4.2, se analizan los siguientes errores; el último error de entrenamiento del mejor individuo $E(w_{*,G_F}, b_{*,G_F}))$ y el error de prueba del mejor elemento $(E_{Pru}(w_{*,G}, b_{*,G}))$. Además del número de generaciones requeridas por el experimento. Todos estos valores son el promedio de los 100 experimentos. En la Tabla 4.3 se encuentra el desglose de los 200 experimentos realizados, detallando en porcentaje el criterio de paro de los experimentos, ya sea por haber alcanzado el máximo número de generaciones $(Alto_{MaxGeneraciones})$, debido a la detección temprana de sobreajuste $(Alto_{Sobreajuste})$ o por haber alcanzado el erro

mínimo establecido ($Alto_{Error}$).

Tabla 4.2: Ventajas de la detección temprana de sobreajuste

Promedio de	Sin detección temprana	Con detección temprana	Porcentaje de mejora con detección temprana
$\mathbf{E}(w_{*,G_F}, b_{*,G_F})$	3.213	3.268	-1.732 %
$\mathbf{E}_{Pru}(w_{*,G_F}, b_{*,G_F})$	4.602	4.411	4.154 %
Generaciones	25000	17593	29.630 %

Tabla 4.3: Criterio de paro de los experimentos

Porcentaje de	Sin detección temprana	Con detección temprana
$\mathbf{Alto}_{Error}$	0 %	0 %
$\mathbf{Alto}_{Sobreajuste}$	0 %	75 %
$\mathbf{Alto}_{MaxGeneraciones}$	100 %	25 %

4.3. Discusión

Los parámetros que se pueden modificar en una red neuronal son bastantes, desde el número de neuronas, número de capas, tamaño del bloque de retardo, el horizonte de predicción, todos estos parámetros influye en la precisión de la predicción.

Al aumentar el número de neuronas podremos tener una mayor precisión en la aproximación, pero esto incrementaría las dimensiones del problema, haciendo más costosa la optimización del entrenamiento. Si se elige un horizonte de predicción muy amplio, tendremos menos datos con los que entrenar y esto reduciría el costo del entrenamiento, pero esto volvería a la predicción más susceptible al fallo, debido a que en un horizonte mayor hay más perturbaciones que no serán contadas en la predicción. Por otro lado, pensando en una aplicación real, nuestro problema está acotado al tiempo de respuesta de un amortiguador magnetoreológico en una suspensión activa, no habrá mucha diferencia un sensor que pueda trabajar a altas frecuencias cuando la respuesta del amortiguador sea del orden de 15 Khz.

En la Figura 4.3 se puede apreciar que el área predicha por el sensor virtual es muy similar al área medida una vez alcanzado el horizonte de predicción. Esto demuestra que el horizonte escogido es funcional, ya que

predice de manera correcta además de permitir el correcto funcionamiento de una suspensión activa.

El resultado de la Tabla 4.2 muestra como el algoritmo de detección temprana mejora la calidad de la predicción. En el caso del último error de entrenamiento del mejor individuo, si desactivamos el criterio de paro por sobre ajuste, no habrá otro criterio que detenga al algoritmo y este continuará buscando una mejor solución hasta que se acaben las generaciones permitidas o se alcance el error mínimo establecido. Esta es la razón por la que el último error de entrenamiento del mejor individuo es ligeramente mejor si no activamos la detección temprana. El error de prueba del mejor elemento es 4,154 % mejor con la detección temprana activado, la razón es que el sobreajuste en los datos es muy bajo, esto permite a la NARX tener un buen desempeño cuando es puesta a prueba con un conjunto de datos distinto a los usados en el entrenamiento. Al estar activado el algoritmo de detección temprana de sobreajuste, este permitió un ahorro promedio de 29,630 % de generaciones, esto implica un menor gasto de recursos de cómputo y por ende, un menor tiempo de entrenamiento. De los cien experimentos ejecutados con detección temprana de sobreajuste, 75 % de los experimentos fueron detenidos por sobreajuste, dejando el resto llegar al número de generaciones máximas permitidas.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Este trabajo realizó un sensor virtual predictivo mediante un tipo de redes neuronales recurrentes, la red NARX. Este sensor virtual es capaz de predecir el área de contacto de los neumáticos a una frecuencia de 2,5 Khz que es equivalente a predecir 0,4 ms en el futuro. Este tiempo permite una predicción precisa del área, esto con el fin de anticiparse a las perturbaciones del terreno, en lugar de reaccionar ante ellas. El problema de entrenar una red neuronal recurrente por métodos basados en el cálculo del gradiente se debe a la presencia de valles espurios. Debido a la capacidad de búsqueda de los algoritmos estocásticos, estos son una excelente alternativa para el problema.

Modificar cada elemento de la arquitectura de la red neuronal se verá directamente reflejada en la precisión de la predicción, y la dificultad de su entrenamiento. Al modificar el tamaño de los bloques de retardo se notó que el desempeño de la red neuronal es proporcional al tamaño de estos bloques, a menor tamaño, menor rendimiento, pero llega un punto en el que aumentar el tamaño de los bloques de retardo no reflejan mejoría en la predicción, pero si se vuelve más tardado su entrenamiento. Por este motivo se eligió un tamaño de bloque de tres retardos.

Debido a los problemas que presenta entrenar una red neuronal recurrente por medio de un algoritmo basado en el gradiente, algoritmos que realizan búsquedas estocásticas tienen un mejor desempeño. Una de las características más importante de los algoritmos de búsqueda es que es difícil que estos se

queden varados en una zona de mínimo local, dependiendo de la función de mutación, este tipo de algoritmo puede realizar una búsqueda más amplia sobre el espacio de soluciones.

No obstante, aún cuando se lograse ajustar la salida de la red neuronal a cada uno de los datos de entrenamiento, el algoritmo de entrenamiento debe ser capaz de detectar un sobreajuste a estos datos, esto se logró mediante el seguimiento del error de validación, el cual se calcula con un conjunto distinto al de entrenamiento.

El desempeño de una suspensión semi-activa puede ser notablemente mejorado, si el sistema pudiera anticiparse a los cambios en el terreno, en lugar de realizar acciones preventivas que correctivas. Esto ha motivado a proponer un sensor virtual capaz de predecir el área de contacto de los neumáticos 0,4 ms y una precisión de 3 cm^2. Esto se logró implementando una red NARX en el sensor virtual. Las características de estas redes permiten ser el candidato ideal para predicción, pero su entrenamiento es complicado si se opta por usar un entrenamiento basado en el cálculo del gradiente del error. Como solución se propuso un método de entrenamiento usando el algoritmo de evolución diferencial. El trabajo de tesis describe la adecuación de este algoritmo al entrenamiento de redes neuronales recurrentes.

5.2. Trabajo a Futuro

Con el deseo de continuar con el desarrollo alcanzado por este trabajo, se plantea mejorar los resultados obtenidos por la arquitectura de la red NARX propuesta por medio de los parámetros usados en el algoritmo de evolución diferencial. Se planea hacer un análisis del impacto que tiene cada unas de las variables de entrada elegidas con respecto al área de contacto de los neumáticos, esto con el objetivo de reducir este número de variables y reducir el tiempos de entrenamiento. Comparar el desempeño del entrenamiento obtenido con otros algoritmos de optimización como enjambre de partículas. En un futuro, implementar el sensor virtual como parte de un sistema de suspensión activa y finalmente, llevar la metodología a un vehículo convencional.

Referencias

[1] Administration, N. H. T. S., et al. Federal motor vehicle safety standards; tire pressure monitoring systems; controls and displays. Tech. rep., Technical report, Department of Transportation, http://www. nhtsa. gov/cars/rules/rulings/tirepresfinal/index. html, 2000.

[2] Atiya, A. F., and Parlos, A. G. New results on recurrent network training: unifying the algorithms and accelerating convergence. *Neural Networks, IEEE Transactions on 11*, 3 (2000), 697–709.

[3] Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on 5*, 2 (1994), 157–166.

[4] Berntorp, K. Joint wheel-slip and vehicle-motion estimation based on inertial, gps, and wheel-speed sensors. *IEEE Transactions on Control Systems Technology 24*, 3 (May 2016), 1020–1027.

[5] Bevly, D. M., Sheridan, R., and Gerdes, J. C. Integrating ins sensors with gps velocity measurements for continuous estimation of vehicle sideslip and tire cornering stiffness. In *American Control Conference, 2001. Proceedings of the 2001* (2001), vol. 1, IEEE, pp. 25–30.

[6] Cai, X., Zhang, N., Venayagamoorthy, G. K., and Wunsch, D. C. Time series prediction with recurrent neural networks trained by a hybrid pso–ea algorithm. *Neurocomputing 70*, 13 (2007), 2342–2353.

[7] Castillo, J., Blanca, A. P. D. L., Cabrera, J., and Simón, A. An optical tire contact pressure test bench. *Vehicle System Dynamics 44*, 3 (2006), 207–221.

[8] Connor, J., and Atlas, L. Recurrent neural networks and time series prediction. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on* (1991), vol. 1, IEEE, pp. 301–306.

[9] Consortium, A., et al. Intelligent tyre systems-state of the art and potential technologies. *APOLLO Deliverable D7 for Project IST-2001-34372. Also available at http://www. vtt. fi/apollo., Technical Research Centre of Finland (VTT)* (2003).

[10] Dakhlallah, J., Glaser, S., Mammar, S., and Sebsadji, Y. Tire-road forces estimation using extended kalman filter and sideslip angle evaluation. In *American Control Conference, 2008* (2008), IEEE, pp. 4597–4602.

[11] De Jesús, O., Horn, J. M., and Hagan, M. T. Analysis of recurrent network training and suggestions for improvements. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on* (2001), vol. 4, IEEE, pp. 2632–2637.

[12] Dieter, G., and Schmidt, L. *Engineering Design, 5th ed.* McGraw-Hill Book Co, 2012.

[13] Duchanoy Martínez, C. A. Desarrollo de un modelo dinámico integral de un vehículo todo terreno con 6 subsistemas, su validación y estudio de maniobrabilidad y confort.

[14] Ezeta, J. H., Mandow, A., and Cerezo, A. G. Los sistemas de suspensión activa y semiactiva: una revisión. *Revista Iberoamericana de Automática e Informática Industrial RIAI 10*, 2 (2013), 121–132.

[15] Feng, L., Yang, Y.-F., and Wang, Y.-X. A new approach to adapting control parameters in differential evolution algorithm. In *Asia-Pacific Conference on Simulated Evolution and Learning* (2008), Springer, pp. 21–30.

[16] Fortuna, L., Graziani, S., Rizzo, A., and Xibilia, M. G. *Soft sensors for monitoring and control of industrial processes*. Springer Science & Business Media, 2007.

[17] Ghaffari, A., Khodayari, A., Arvin, S., and Alimardani, F. An anfis design for prediction of future state of a vehicle in lane change

behavior. In *Control System, Computing and Engineering (ICCSCE), 2011 IEEE International Conference on* (2011), IEEE, pp. 156–161.

[18] Hagan, M. T., Demuth, H. B., and Beale, M. H. Neural Network Design.

[19] Hagan, M. T., and Jesus, O. D. Backpropagation Algorithms for a Broad Class of Dynamic Networks. *IEEE Transactions on Neural Networks 18*, 1 (2007), 14–27.

[20] Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[21] Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks 2*, 5 (1989), 359–366.

[22] Ilonen, J., Kamarainen, J.-K., and Lampinen, J. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters 17*, 1 (2003), 93–105.

[23] Jesus, O. D., Horn, J., and Hagan, M. Analysis of recurrent network training and suggestions for\nimprovements. *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222) 4* (2001), 2632–2637.

[24] Keck, M. A new approach of a piezoelectric vibration-based power generator to supply next generation tire sensor systems. In *Sensors, 2007 IEEE* (Oct 2007), pp. 1299–1302.

[25] Li, L., and Wang, F.-Y. *Advanced motion control and sensing for intelligent vehicles*. Springer Science & Business Media, 2007.

[26] Liu, S., Pang, X., Ji, H., and Chen, H. Prediction of track irregularities using narx neural network. In *Circuits,Communications and System (PACCS), 2010 Second Pacific-Asia Conference on* (Aug 2010), vol. 1, pp. 109–112.

[27] Makki, N., and Pop-Iliev, R. Piezoelectric power generation for sensor applications: design of a battery-less wireless tire pressure sensor.

In *SPIE Microtechnologies* (2011), International Society for Optics and Photonics, pp. 806618–806618.

[28] Matsuzaki, R., and Todoroki, A. Passive wireless strain monitoring of actual tire using capacitance–resistance change and multiple spectral features. *Sensors and Actuators A: Physical 126*, 2 (2006), 277 – 286.

[29] Matsuzaki, R., and Todoroki, A. Intelligent tires based on measurement of tire deformation. *Journal of solid mechanics and Materials engineering 2*, 2 (2008), 269–280.

[30] Matsuzaki, R., and Todoroki, A. Wireless monitoring of automobile tires for intelligent tires. *Sensors 8*, 12 (2008), 8123–8138.

[31] Mezura-Montes, E., Velázquez-Reyes, J., and Coello Coello, C. A. A comparative study of differential evolution variants for global optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (2006), ACM, pp. 485–492.

[32] Miao, P., Liu, G., Zhang, D., Jiang, Y., Zhang, H., and Zhou, H. Sideslip angle soft-sensor based on neural network left inversion for multi-wheel independently driven electric vehicles. In *Neural Networks (IJCNN), 2014 International Joint Conference on* (2014), IEEE, pp. 2171–2175.

[33] Miller, S. L., Youngberg, B., Millie, A., Schweizer, P., and Gerdes, J. C. Calculating longitudinal wheel slip and tire parameters using gps velocity. In *American Control Conference, 2001. Proceedings of the 2001* (2001), vol. 3, IEEE, pp. 1800–1805.

[34] Nam, K., Oh, S., Fujimoto, H., and Hori, Y. Estimation of sideslip and roll angles of electric vehicles using lateral tire force sensors through rls and kalman filter approaches. *IEEE Transactions on Industrial Electronics 60*, 3 (March 2013), 988–1000.

[35] Phan, M. C., Beale, M. H., and Hagan, M. T. A procedure for training recurrent networks. In *The 2013 International Joint Conference on Neural Networks (IJCNN)* (2013).

[36] Slišković, D., Grbić, R., and Hocenski, Ž. Methods for plant data-based process modeling in soft-sensor development. *AUTOMATIKA: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije 52*, 4 (2012), 306–318.

[37] Sola, J., and Sevilla, J. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE transactions on nuclear science 44*, 3 (1997), 1464–1468.

[38] Storn, R., and Price, K. *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*, vol. 3. ICSI Berkeley, 1995.

[39] Suratgar, A. A., Tavakoli, M. B., and Hoseinabadi, A. Modified levenberg-marquardt method for neural networks training. *World Acad Sci Eng Technol 6* (2005), 46–48.

[40] Toplar, H. Experimental analysis of smart tires.

[41] Urbina, L., Duchanoy, C. A., Faustino-González, G., Moreno-Armendáriz, M. A., Cruz-Villar, C. A., and Calvo, H. A novel tire contact patch soft sensor via neural networks. In *Electrical Engineering, Computing Science and Automatic Control (CCE), 2015 12th International Conference on* (2015), IEEE, pp. 1–6.

[42] Urbina, L., Duchanoy, C. A., Moreno-Armendáriz, M. A., Lara, D., and Calvo, H. Implementación sobre fpga de la estrategia evolutiva cma-es para optimización numérica. In *Research in Computing Science Issue 105 (2015)* (2015), pp. 97—-106.

[43] Zhang, W., Ding, N., Yu, G., and Zhou, W. Virtual sensors design in vehicle sideslip angle and velocity of the centre of gravity estimation. In *Electronic Measurement & Instruments, 2009. ICEMI'09. 9th International Conference on* (2009), IEEE, pp. 3–652.

[44] Zhu, M., and Worthington, E. Design and testing of piezoelectric energy harvesting devices for generation of higher electric power for wireless sensor networks. In *Sensors, 2009 IEEE* (Oct 2009), pp. 699–702.