UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA:

INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de: INGENIERO/A ELECTRÓNICO.

TEMA:

DESARROLLO Y EVALUACIÓN DE UN CONTROL NEURO-DIFUSO TIPO ANFIS FRENTE A UN CONTROL PID CONVENCIONAL APLICADO AL PÉNDULO INVERTIDO.

AUTORES:

MARÍA JOSÉ JARA CHICO

DIEGO RAÚL ROCHA PROAÑO

DIRECTOR:

JUNIOR RAFAEL FIGUEROA OLMEDO

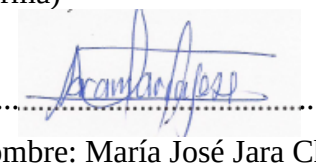
Quito, Septiembre del 2016.

CESIÓN DE DERECHOS DE AUTOR

Nosotros María José Jara Chico con documento de identificación N° 171147659-6 y Diego Raúl Rocha Proaño con documento de identificación N° 172075103-9, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de grado/titulación intitulado: "DESARROLLO Y EVALUACION DE UN CONTROL NEURO-DIFUSO TIPO ANFIS FRENTE A UN CONTROL PID CONVENCIONAL APLICADO AL PÉNDULO INVERTIDO", mismo que ha sido desarrollado para optar por el título de: Ingeniero/a Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

(Firma) (Firma)



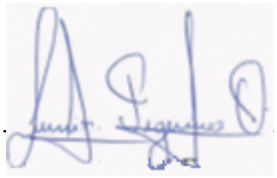
Nombre: María José Jara Chico
Cédula: 171147659-6
Fecha: 2016-09-08

Nombre: Diego Raúl Rocha Proaño
Cédula: 172075103-9
Fecha: 2016-09-08

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de titulación, DESARROLLO Y EVALUACION DE UN CONTROL NEURO-DIFUSO TIPO ANFIS FRENTE A UN CONTROL PID CONVENCIONAL APLICADO AL PÉNDULO INVERTIDO, realizado por Diego Raúl Rocha Proaño y María José Jara Chico, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerado como trabajo final de titulación.

Quito, 8 de Septiembre del 2016

……  …

Ing. Junior Rafael Figueroa Olmedo

C.I: 0802820183

ÍNDICE GENERAL

INTRODUCCION 1

CAPITULO I 1

ANTECEDENTES 1

1.1. Planteamiento del problema 1

1.2. Tema 1

1.3. Justificación 1

1.4. Objetivos 2

1.4.1. Objetivo General 2

1.4.2. Objetivos Específicos 2

1.5. Alcance 3

CAPITULO II 4

MARCO CONCEPTUAL 4

2.1. Sistema de Inferencia Difuso 4

2.2. Modelo Difuso Tipo Sugeno 5

2.3. Métodos de Partición del Espacio de Entrada 6

2.3.1. Partición Tipo Rejilla (*Grid*) 6

2.3.2. Partición Tipo Árbol (*Tree*). 7

2.3.3. Partición Tipo Disperso (*Scatter*). 8

2.4. Identificación de Sistemas 9

2.4.1. Estimador de Mínimos Cuadrados para la Identificación de Sistemas. 11

2.4.2. Estimador de Mínimos Cuadrados Recursivo 13

2.4.3. LSE Recursivo para Sistemas Variantes en el Tiempo 15

2.5. Redes Neuronales Adaptativas 17

2.5.1. Arquitectura 18

2.5.2. Algoritmo de Retro-propagación: Regla Delta Generalizada 19
2.5.3. Aprendizaje con Término de Momento 21
2.5.4. Regla de Aprendizaje de Retro-Propagación para una Red Adaptativa. 22
2.5.5. Diferentes Maneras de Combinar el Gradiente Descendente y el Estimador de Mínimos Cuadrados. 23
2.5.6. Regla de Aprendizaje Híbrida 24
2.6. Modelo ANFIS 27
2.6.1. Arquitectura ANFIS 27
2.6.2. Algoritmo de Aprendizaje Híbrido 30

CAPITULO III 32
DESCRIPCIÓN DE LA PLANTA 32
3.1. Introducción 32
3.2. Descripción de la Planta Péndulo Invertido. 32
3.3. Modelamiento Matemático del Péndulo 33
3.3.1. Análisis del Movimiento Angular del Péndulo 35
3.3.2. Análisis del Movimiento Rectilíneo del Péndulo y el Carro Móvil 36
3.3.3. Modelado del Sistema Péndulo Invertido en Simulink 39
3.4. Linealización del Modelo 40
3.5. Función de Transferencia del Sistema Péndulo Invertido 41

CAPITULO IV 44
IMPLEMENTACIÓN Y SIMULACIÓN 44
4.1. Introducción 44
4.2. Algoritmo de un Sistema de Inferencia Neuro-Difusa Adaptativa (ANFIS) 44

4.3. Identificación de la Planta Péndulo Invertido utilizando una Red ANFIS .. 49
4.4. Configuración de los Parámetros del Bloque ANFIS Para Identificación. . 52
4.5. Controlador Neuro-Difuso Tipo ANFIS 54
CAPITULO V 62
ANALISIS Y RESULTADOS 62
5.1. Introducción 62
5.2. Análisis Comparativo entre el Controlador PID y el Controlador ANFIS .. 62
5.2.1. Test de Wilcoxon 63
CONCLUSIONES 65
RECOMENDACIONES 67
REFERENCIAS 68

ÍNDICE DE FIGURAS

FIGURA 2.1: ESTRUCTURA DE UN SISTEMA DE INFERENCIA DIFUSO 4
FIGURA 2.2: MODELO TIPO SUGENO 5
FIGURA 2.3: MODELO DE PARTICIÓN TIPO REJILLA 7
FIGURA 2.4: MÉTODO DE PARTICIÓN TIPO ÁRBOL 8
FIGURA 2.5: MÉTODO DE PARTICIÓN TIPO DISPERSO 8
FIGURA 2.6: IDENTIFICACIÓN DE PARÁMETROS 10
FIGURA 2.7: RED ADAPTATIVA 18
FIGURA 2.8: RED NEURONAL 19
FIGURA 2.9: RED NEURONAL ADAPTATIVA 22
FIGURA 2.10: MODELO SUGENO 28
FIGURA 2.11: ARQUITECTURA ANFIS 28
FIGURA 2.12: ARQUITECTURA ANFIS Y MODELO DIFUSO SUGENO 30
FIGURA 3.1: PÉNDULO INVERTIDO 33
FIGURA 3.2: DIAGRAMA MECÁNICO 34
FIGURA 3.3: DIAGRAMA DE CUERPO LIBRE 36
FIGURA 3.4: PLANTA MISO 38
FIGURA 3.5: DIAGRAMA DE BLOQUES 39
FIGURA 4.1: DIAGRAMA DE FLUJO PRINCIPAL 45
FIGURA 4.2: SUBPROCESO DEL ALGORITMO ANFIS 46
FIGURA 4.3: SUBPROCESO DEL ALGORITMO ANFIS 47
FIGURA 4.4: SUBPROCESO DEL ALGORITMO ANFIS 48
FIGURA 4.5: SUBPROCESO DEL ALGORITMO ANFIS 49
FIGURA 4.6: DIAGRAMA DE LA IDENTIFICACIÓN DE LA PLANTA 50
FIGURA 4.7: BLOQUE ANFIS 51
FIGURA 4.8: BLOQUE DE LOS PARÁMETROS ANFIS 52
FIGURA 4.9: REPRESENTACIÓN DE LAS SEÑALES 54
FIGURA 4.10: FOTOGRAFÍA DEL PÉNDULO INVERTIDO DEL LABORATORIO DE TEORÍA DE CONTROL 55
FIGURA 4.11: BLOQUE DE PARÁMETROS ANFIS 55
FIGURA 4.12: DIAGRAMA DE CONTROL 56
FIGURA 4.13: REPRESENTACIÓN DE LAS SEÑALES DE SALIDA 57

FIGURA 4.14: EJEMPLOS PROPIOS DE FEEDBACK 58
FIGURA 4.15: EJEMPLO DEL FEEDBACK DEL PÉNDULO INVERTIDO 58
FIGURA 4.16: CONTROL PID 59
FIGURA 4.17: BLOQUE DE LOS PARÁMETROS DEL PID 59
FIGURA 4.18: REPRESENTACIÓN DE LAS SEÑALES DE SALIDA DEL PID 60
FIGURA 4.19: DIAGRAMA DEL CONTROLADOR NEURO-DIFUSO 61
FIGURA 4.20: REPRESENTACIÓN GRÁFICA DE LAS SEÑALES DEL CONTROLADOR ANFIS 61
FIGURA 5.1: TABLA DE WILCOXON 63
FIGURA 5.2: RESULTADOS 64

ÍNDICE DE TABLAS

TABLA 2.1: DEFINICIÓN DE PARÁMETROS. 31
TABLA 2.2: PROCEDIMIENTO DEL APRENDIZAJE HÍBRIDO. 31
TABLA 3.1: DESCRIPCIÓN DE LAS VARIABLES DE ESTADO DEL PÉNDULO INVERTIDO. . 34
TABLA 3.2: VALORES DE LAS VARIABLES INFLUYENTES EN EL MODELADO DEL PÉNDULO. 38
TABLA 4.1. PARÁMETROS DE APRENDIZAJE NEURO-DIFUSO 53

ÍNDICE DE ECUACIONES

ECUACIÓN 2.1 6
ECUACIÓN 2.2 11
ECUACIÓN 2.3 11
ECUACIÓN 2.4 11
ECUACIÓN 2.5 12
ECUACIÓN 2.6 12
ECUACIÓN 2.7 12
ECUACIÓN 2.8 12
ECUACIÓN 2.9 12
ECUACIÓN 2.10 13
ECUACIÓN 2.11 13
ECUACIÓN 2.12 13
ECUACIÓN 2.13 13
ECUACIÓN 2.14 13
ECUACIÓN 2.15 13
ECUACIÓN 2.16 13
ECUACIÓN 2.17 14
ECUACIÓN 2.18 14
ECUACIÓN 2.19 14
ECUACIÓN 2.20 14
ECUACIÓN 2.21 15
ECUACIÓN 2.22 15
ECUACIÓN 2.23 15
ECUACIÓN 2.24 15
ECUACIÓN 2.25 15
ECUACIÓN 2.26 16
ECUACIÓN 2.27 16
ECUACIÓN 2.28 16
ECUACIÓN 2.29 16
ECUACIÓN 2.30 16
ECUACIÓN 2.31 17

ECUACIÓN 2.32 17
ECUACIÓN 2.33 17
ECUACIÓN 2.34 17
ECUACIÓN 2.35 17
ECUACIÓN 2.36 20
ECUACIÓN 2.37 20
ECUACIÓN 2.38 20
ECUACIÓN 2.39 20
ECUACIÓN 2.40 20
ECUACIÓN 2.41 20
ECUACIÓN 2.42 20
ECUACIÓN 2.43 20
ECUACIÓN 2.44 21
ECUACIÓN 2.45 21
ECUACIÓN 2.46 21
ECUACIÓN 2.47 21
ECUACIÓN 2.48 23
ECUACIÓN 2.49 23
ECUACIÓN 2.50 23
ECUACIÓN 2.51 23
ECUACIÓN 2.52 23
ECUACIÓN 2.53 23
ECUACIÓN 2.54 23
ECUACIÓN 2.55 25
ECUACIÓN 2.56 25
ECUACIÓN 2.57 25
ECUACIÓN 2.58 25
ECUACIÓN 2.59 25
ECUACIÓN 2.60 26
ECUACIÓN 2.61 27
ECUACIÓN 2.62 27
ECUACIÓN 2.63 27
ECUACIÓN 2.64 28
ECUACIÓN 2.65 28

ECUACIÓN 2.66 29
ECUACIÓN 2.67 29
ECUACIÓN 2.68 29
ECUACIÓN 2.69 29
ECUACIÓN 2.70 30
ECUACIÓN 2.71 31
ECUACIÓN 3.1 36
ECUACIÓN 3.2 36
ECUACIÓN 3.3 37
ECUACIÓN 3.4 37
ECUACIÓN 3.5 37
ECUACIÓN 3.6 37
ECUACIÓN 3.7 37
ECUACIÓN 3.8 37
ECUACIÓN 3.9 37
ECUACIÓN 3.10 37
ECUACIÓN 3.11 37
ECUACIÓN 3.12 39
ECUACIÓN 3.13 39
ECUACIÓN 3.14 39
ECUACIÓN 3.15 40
ECUACIÓN 3.16 40
ECUACIÓN 3.17 40
ECUACIÓN 3.18 40
ECUACIÓN 3.19 40
ECUACIÓN 3.20 40
ECUACIÓN 3.21 40
ECUACIÓN 3.22 40
ECUACIÓN 3.23 40
ECUACIÓN 3.24 41
ECUACIÓN 3.25 41
ECUACIÓN 3.26 41
ECUACIÓN 3.27 41
ECUACIÓN 3.28 41

ECUACIÓN 3.29 42
ECUACIÓN 3.30 42
ECUACIÓN 3.31 42
ECUACIÓN 3.32 42
ECUACIÓN 3.33 42
ECUACIÓN 3.34 43

RESUMEN

El presente proyecto técnico está dirigido a la investigación y desarrollo de un controlador neuro-difuso tipo ANFIS, por lo que se abordará en un inicio conceptos generales referentes a la lógica difusa y las redes neuronales las cuales forman parte de la inteligencia artificial, tomando en consideración solo las temáticas empleadas por este tipo de controladores. Por medio del software Matlab se desarrolló la programación del controlador ANFIS y se utilizó el entorno de programación gráfica Simulink para realizar las respectivas pruebas de funcionamiento de una manera gráfica. Dicho entorno permite realizar la configuración de los parámetros y variables utilizadas por la red ANFIS. Una vez desarrollado el controlador ANFIS a nivel de software se realizó el modelamiento matemático de la planta sobre la cual se realizarán las acciones de control, además se identificaron las variables a controlar. Para la implementación del controlador ANFIS ya se cuenta de antemano con la planta del péndulo digital ubicada en el Laboratorio de Teoría de Control de la Universidad Politécnica Salesiana Campus Sur. Una vez implementado el controlador se procedió a tomar los datos necesarios para realizar un análisis comparativo entre el controlador PID (ejemplo incluido en el software del sistema) y el controlador ANFIS. Con los resultados obtenidos se estableció, cuál de los dos controladores presentó mejores prestaciones en cuanto a su desempeño en las acciones de control; tomando en consideración los siguientes parámetros de diseño: máximos picos, tiempo de estabilización, error en estado estacionario, estabilidad, etc.

ABSTRACT

This is a research and development project aimed to develop a neuro-fuzzy controller, general concepts, characteristics and its classification grouped under ANFIS type neuro-fuzzy control will be used to design an algorithm to control this project. The programming of the controller and the development of a graphical interface for displaying parameters in both steady state and transitory regime will be accomplished using MATLAB software. Once the ANFIS type neuro-fuzzy controller is developed, a software level a study will be performed on the plant on which the controller is going to be operated to validate the performance of the controller. It will cover the operating characteristics of the component present in the module names "Digital Pendulum" Furthermore, the mathematical modeling of the plant will be established and the control variables will be identified. In order to implement the ANFIS type neuro-fuzzy controller a plant that is already located at the Laboratory of Control Theory of the Polytechnic Salesian University, South Campus. When the ANFIS type neuro-fuzzy controller is implemented, data will be collected in order to make a comparative analysis between the current PID controller that is already installed at the plant against the ANFIS type neuro-fuzzy controller developed for this project and with the analysis of the results, a list of advantages and disadvantages for each controller will be established. For each of the following design parameters: maximum peaks, settling time, steady-state error, stability, etc.

INTRODUCCION

Los sistemas de control neuro-difusos engloban un conjunto de tecnologías compuestas por la lógica difusa y las redes neuronales artificiales, las cuales se complementan entre sí teniendo en común la capacidad de manejar información indeterminada y aleatoria, en el diseño de sistemas inteligentes. Estos sistemas han aumentado el interés de investigadores en diversas áreas científicas y de la ingeniería, debido a la creciente necesidad de adaptación de los sistemas inteligentes para resolver los problemas en situaciones reales.

Varios estudios sobre las investigaciones realizadas a nivel nacional sobre los controladores neuro-difusos han puesto en manifiesto que existen varias publicaciones desarrolladas en diferentes universidades; por dicha razón se ha tomado como punto de partida el desarrollo de un controlador muy conocido a nivel práctico denominado ANFIS. Tomando en consideración que aproximadamente el 95% de procesos de control a nivel industrial usan controladores PID, se realizó un análisis comparativo entre ambas tecnologías, la distribución de desarrollo de este trabajo se presenta a continuación.

En el Capítulo I se encuentra la parte contextual de este proyecto en donde se detalla el funcionamiento de un tipo de sistema de inferencia difuso conocido como el modelo difuso tipo Sugeno; también se realiza un estudio de los métodos de partición de los datos de entrada y explica la temática de dos algoritmos de aprendizaje como son el Estimador de Mínimos Cuadrados y el algoritmo de Retro Propagación del Error, cuya combinación dan como resultado la regla de aprendizaje híbrida la cual es utilizado en de la arquitectura de la red ANFIS que se encuentra detallada en este mismo capítulo.

En el Capítulo II se detalla el modelamiento de la planta del péndulo digital, así como también sus características y configuraciones.

En el Capítulo III se realiza el desarrollo del controlador neuro-difuso tipo ANFIS en el software Matlab y se explica el procedimiento para crear el bloque S-Function que permite relacionar el algoritmo elaborado en Matlab con el entorno grafico Simulink.

En el último Capítulo IV se realiza un análisis comparativo entre el controlador PID y el controlador ANFIS utilizando el test de Wilcoxon, el mismo que analizará el indicador de desempeño IAE y permitirá establecer que controlador tiene mejores prestaciones.

CAPITULO I

ANTECEDENTES

1.1. Planteamiento del problema

La investigación y aplicación de sistemas neuro-difusos a nivel mundial ha sido objeto de diversas investigaciones en la actualidad, por el contrario, a nivel nacional el campo de aplicación de estos modelos sobre los sistemas de control de procesos no ha sido muy abordado debido a la falta de información de cómo realizar un controlador neuro-difuso y de datos que corroboren su desempeño frente a otro tipo de controlador.

En la actualidad las plantas industriales poseen sistemas más sofisticados con un modelo matemático complejo e impreciso por lo tanto es necesario recurrir a un tipo de sistema de control inteligente basado en la experiencia humana, como es un controlador neuro-difuso el cual se basa en la información recopilada de un experto y por medio de reglas difusas y un bloque de inferencia llegar a la decisión que normalmente tomaría un ser humano.

Existen investigaciones con enfoques de distintos tipos de control que conlleva el modelamiento de la planta, este proyecto tiene como propósito elaborar un análisis de desempeño y comparación entre dos tipos de controladores que sirvan como guía de elaboración de proyectos futuros que tengan incluyan algún tipo de control neuro difuso.

1.2. Tema

DESARROLLO Y EVALUACION DE UN CONTROL NEURO-DIFUSO TIPO ANFIS FRENTE A UN CONTROL PID CONVENCIONAL APLICADO AL PÉNDULO INVERTIDO.

1.3. Justificación

Algunas de las actividades realizadas por el hombre tales como operaciones matemáticas, almacenamiento de datos etc. Pudieron ser simplificadas mediante el perfeccionamiento de los sistemas computacionales. Dichos sistemas actualmente

facilitan la realización de algoritmos de control con un entorno más amigable para el usuario a través de conceptos y métodos utilizados en la inteligencia artificial como son la lógica difusa y las redes neuronales a las cuales de manera colectiva se denominan metodologías de Control Inteligente, esta metodología evita complicaciones matemáticas que se presentan durante el desarrollo de controladores convencionales, más aun cuando se trata con procesos no lineales de difícil modelamiento matemático. Como es el caso del péndulo invertido donde el control realizado por un ser humano puede ser de una manera aparentemente obvia e intuitiva, a pesar de tener una difícil interpretación mediante métodos matemáticos convencionales, es por esto que el presente proyecto de titulación es de carácter investigativo, científico y práctico con el cual se aportará al desarrollo de controladores neuro-difusos, debido a que a nivel institucional no existen trabajos similares de investigación que se enfoquen en el desarrollo de este tipo de controladores.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar y evaluar un control neuro-difuso tipo ANFIS frente a un control PID convencional aplicado al péndulo invertido perteneciente a la Universidad Politécnica Salesiana campus sur sede Quito para determinar que controlador brinda los mejores parámetros de control.

1.4.2. Objetivos Específicos

- Modelar y estudiar matemáticamente la dinámica de la planta (Péndulo Invertido) que servirá como herramienta para aplicar el controlador a diseñar.
- Desarrollar el algoritmo de control sobre la función de transferencia del péndulo invertido, así como en el sistema físico real para evaluar sus características en régimen transitorio y permanente.
- Verificar el funcionamiento del controlador neuro-difuso tipo ANFIS tanto en software como hardware.
- Realizar un análisis comparativo entre el control PID convencional y el controlador neuro-difuso tipo ANFIS desarrollado, con la finalidad de

determinar qué sistema de control muestra mejores características de funcionamiento sobre la planta del péndulo invertido.

1.5. Alcance

El presente proyecto está dirigido a la investigación y desarrollo de un controlador neuro-difuso por lo que se abordará conceptos generales, características y su clasificación en la que consta el control neuro-difuso Tipo ANFIS en base al que se diseñará el algoritmo de control a utilizar en este proyecto. Por medio del software Matlab se desarrollará la programación del controlador y una interfaz gráfica para la visualización de los parámetros tanto en su régimen estacionario como en el régimen transitorio. Una vez desarrollado el controlador se comprobará el funcionamiento de dicho controlador, abarcando características de funcionamiento de los componentes con los que cuenta el modulo "Péndulo Digital". Además, se establecerá el modelamiento matemático de la planta y se identificarán las variables a controlar. Para la implementación del controlador neuro-difuso tipo ANFIS ya se cuenta con la planta que se encuentra ubicada en el Laboratorio de Teoría de Control de la Universidad Politécnica Salesiana Campus Sur. Cuando ya se tenga implementado el controlador neuro-difuso tipo ANFIS se procederá a tomar los datos necesarios para realizar un análisis comparativo entre el controlador PID con el que ya cuenta la planta y con el controlador neuro-difuso tipo ANFIS implementado y con los resultados obtenidos se establecerán las ventajas y desventajas propias de cada controlador.

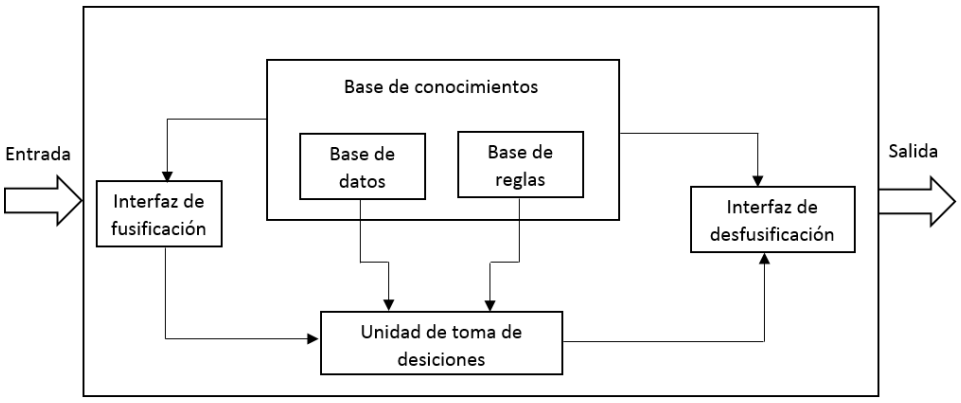
CAPITULO II

MARCO CONCEPTUAL

2.1. Sistema de Inferencia Difuso

Un sistema de inferencia difuso está basado en conceptos de conjuntos difusos, reglas de inferencia del tipo $if - then$, y razonamiento difuso. La estructura de un sistema de inferencia difuso tiene tres componentes que son: la base de reglas, en donde constan todas las reglas de inferencia; la base de datos, en la cual se definen las funciones de membresía usadas en las reglas de inferencia y el mecanismo de razonamiento, que realiza el proceso de inferencia con el conjunto de reglas de inferencia y así entregar una salida adecuada como se indica en la Figura 2.1. (Jang J.-S. R., 2002) Se puede tener entradas de tipo difuso o del tipo numéricas, por lo general las salidas son conjuntos difusos. Cuando es necesario tener un valor numérico en la salida se debe aumentar un sistema de desfusificación. (Jang, Sun, & Mizutani, 1997)

Figura 0.1: Estructura de un sistema de inferencia difuso



Estructura de un sistema de inferencia difusa de 1 entrada y 1 salida, (Ponce Cruz, 2010)

En un sistema difuso se puede determinar dos zonas denominadas antecedente y consecuente, el antecedente define una región difusa en la entrada, mientras que el consecuente determina la ubicación de la salida dentro de la región difusa. Existen tres tipos de sistemas de inferencia difusa: Tipo Mamdani, Tipo Sugeno, Tipo Tsukamoto. Para el caso específico del desarrollo del presente proyecto se utilizará el

sistema de inferencia difuso del tipo Sugeno el que se detallará en la siguiente sección de este capítulo.

2.2. Modelo Difuso Tipo Sugeno

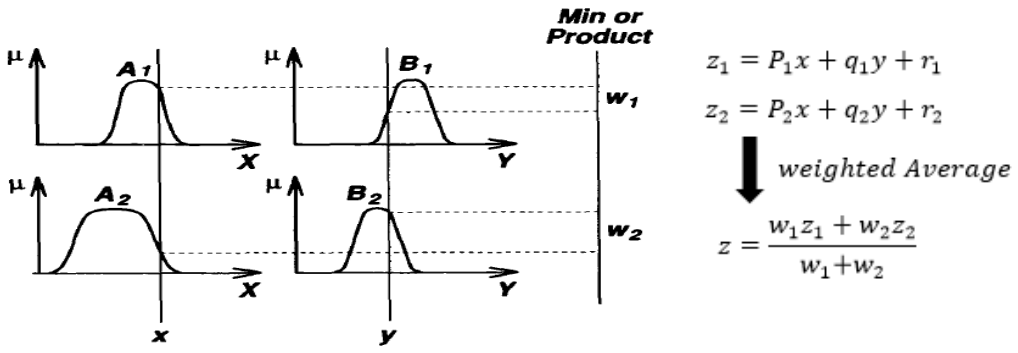
En la búsqueda por desarrollar una aproximación sistemática creando reglas difusas, partiendo de un conjunto de entradas y salidas de datos, Takagi, Sugeno y Kang propusieron un modelo difuso conocido como TSK, también llamado modelo difuso tipo Sugeno. En el modelo tipo Sugeno una regla difusa se la representa de la siguiente manera:

$$Si\ x\ es\ A\ and\ y\ es\ B\ then\ z = f(x, y)$$

Donde A y B son conjuntos difusos correspondientes al antecedente, mientras que z es una función específica de (x, y), la cual se encuentra en el consecuente. Generalmente $f(x, y)$ es una función polinómica, en las variables de entrada x y y, esta puede ser cualquier función siempre que describa correctamente la salida del modelo que este contenida en la región difusa especificada por el antecedente de la regla. Se denomina modelo difuso Sugeno de primer orden cuando $f(x, y)$ es un polinomio de primer orden. (Jang, Sun, & Mizutani, 1997)

En la Figura 2.2 se representa el modelo difuso tipo Sugeno de primer orden y su razonamiento.

Figura 0.2: Modelo Tipo Sugeno



Razonamiento del Modelo Difuso tipo Sugeno de primer orden, (Jang, Sun, & Mizutani, 1997)

Debido a que cada regla tiene una salida específica, la salida total es obtenida mediante el peso promedio, por tanto, se evita el consumo de tiempo durante el proceso de desfusificación requerido en el modelo de Mamdani. (Jang, Sun, & Mizutani, 1997). En la práctica, el peso promedio es remplazado algunas veces con la suma de los pesos, de donde:

$$z = w_1 z_1 + w_2 z_2 \tag{0.1}$$

Con la suma de los pesos se reduce el cálculo especialmente en el entrenamiento del sistema de inferencia difuso. Al reemplazar el peso promedio con la suma de los pesos esto podría llevar a la perdida de significados lingüísticos en la función de membresía, salvo que la suma de los pesos $(\sum_i w_i)$ sea aproximadamente uno. (Jang, Sun, & Mizutani, 1997)

2.3. Métodos de Partición del Espacio de Entrada

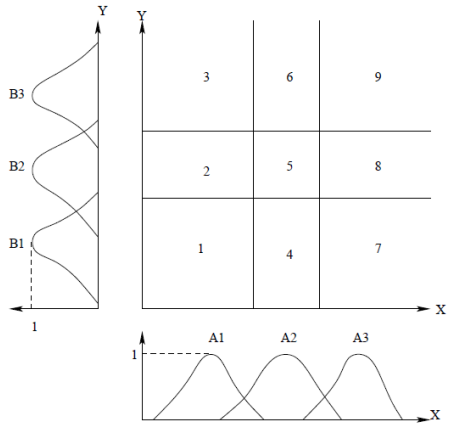
Para elegir un método de partición es importante tener en claro que el antecedente de una regla difusa define una región difusa en la entrada, mientras que el consecuente determina la ubicación de la salida en una región difusa; es en el consecuente de las reglas difusas donde se encuentra la diferencia entre los tipos de sistemas de inferencia difusa ya que el consecuente puede ser un conjunto difuso, un valor constante o una ecuación lineal de inferencia difusa, por ende cambia también su proceso de desfusificación, sin embargo sus antecedentes serán los mismos, por esta razón los métodos de partición son aplicables para todos los tipos de sistemas de inferencia difusa, pero cada método de partición tiene características que solventan problemas como el tipo de entradas y la cantidad de reglas de inferencia. A continuación, se describen los métodos de partición y sus características. (Jang, Sun, & Mizutani, 1997)

2.3.1. Partición Tipo Rejilla (*Grid*)

El método de partición tipo rejilla se lo escoge para diseños de controladores difusos, donde las entradas solamente son variables de estado. Este método de partición necesita solamente un pequeño número de funciones de membresía para cada entrada. Sin embargo, se encuentra problemas cuando se tiene un gran número de entradas, ya que al tener más entradas el número de reglas difusas también aumenta. En la Figura

2.3 se muestra un ejemplo mediante la representación gráfica de este método. (Buragohain, 2008)

Figura 0.3: Modelo de Partición tipo rejilla

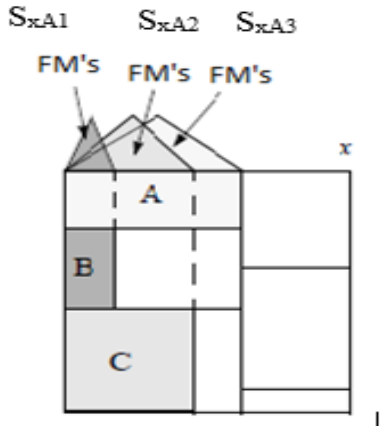


Estructura del método de partición tipo rejilla, (Buragohain, 2008)

2.3.2. Partición Tipo Árbol (*Tree*).

El método de partición tipo árbol resuelve el problema de un incremento exponencial en el número de reglas difusas y define cada región del universo de discurso de manera unívoca, pero es necesario aumentar el número de funciones de membresía para cada entrada. De esta manera se presentan reglas con síntesis más irregulares y con términos lingüísticos más difíciles de interpretar. Para identificar las funciones A, B y se necesitan tres funciones de membresía distintas a la variable de entrada x como se muestra a continuación en la Figura 2.4. (Jang, Sun, & Mizutani, 1997)

Figura 0.4: Método de Partición tipo árbol

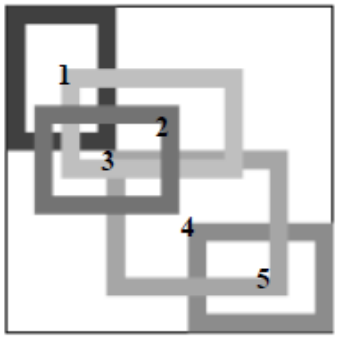


Estructura del método de Partición tipo árbol, (Jang, Sun, & Mizutani, 1997)

2.3.3. Partición Tipo Disperso (*Scatter*).

La partición de tipo dispersa depende de un conocimiento previo del sistema o de la capacidad de extracción de este tipo de reglas a partir de datos, sin permitir un tratamiento sistemático. (Jang, Sun, & Mizutani, 1997). En la Figura 2.5 es representado el método de partición disperso.

Figura 0.5: Método de Partición tipo disperso



Estructura del método de partición tipo disperso, (Jang, Sun, & Mizutani, 1997)

2.4. Identificación de Sistemas

Los métodos utilizados para la identificación de sistemas emplean conjuntos de datos de entrada y de salida para determinar un modelo matemático de un sistema de destino *(target system)* también llamado sistema desconocido.

Un sistema de identificación consta de dos pasos que son:

Identificación de la Estructura. - Para comenzar es necesario tener un previo conocimiento del sistema de destino y con esto elegir la estructura del modelo que se aproxima a dicho sistema. Por lo general esta clase de modelos se representa más con una función parametrizada. (Jang, Sun, & Mizutani, 1997)

$$y = f(\boldsymbol{u}; \boldsymbol{\theta})$$

Donde:

$y = salida\ del\ sistema$

$\boldsymbol{u} = es\ el\ vector\ de\ entrada$

$\boldsymbol{\theta} = es\ el\ vector\ de\ parámetros$

La función f, se basa en la experiencia de los diseñadores y de las variables del sistema de destino.

***Identificación de parámetros*. -** Una vez conocida la estructura del modelo se aplican técnicas de optimización para determinar el vector de parámetros $\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}}$ tal que el modelo resultante sea $\hat{y} = f(\boldsymbol{u}; \widehat{\boldsymbol{\theta}})$, y se pueda describir el sistema de manera propicia.

En la Figura 2.6 se tiene un diagrama esquemático utilizado en la identificación de parámetros, en el que se aplica una entrada $\boldsymbol{u_i}$ al sistema de destino y al modelo de destino. Para actualizar el vector de parámetros $\boldsymbol{\theta}$ se utiliza la diferencia entre la salida del sistema de destino $\boldsymbol{y_i}$ y la salida del modelo $\widehat{\boldsymbol{y}}_{\boldsymbol{i}}$, así mediante la actualización de dichos parámetros se reduce esta diferencia.

Figura 0.6: Identificación de parámetros

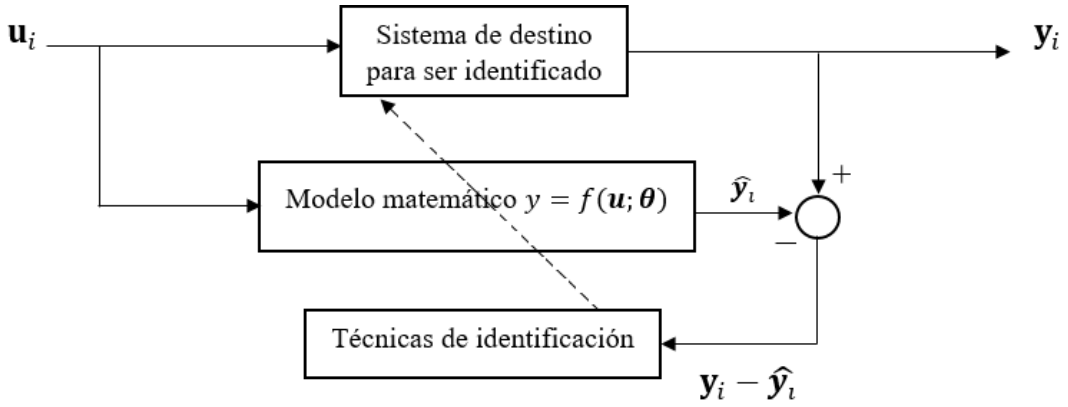


Diagrama de bloques para la identificación de parámetros, (Jang, Sun, & Mizutani, 1997)

Se debe considerar que el conjunto de datos está compuesto de m pares de entradas-salidas $(\boldsymbol{u_i}; \boldsymbol{y_i})$, $i = 1, \dots, m$; este conjunto es llamado conjunto de datos de entrenamiento o conjunto de datos de muestra. De forma general, $\boldsymbol{u}_i$ y $\boldsymbol{y}_i$ representan los vectores de entrada y salida deseada respectivamente.

Para la identificación de un sistema es necesario realizar la identificación de la estructura y la identificación de parámetros varias veces hasta obtener un modelo satisfactorio. El procedimiento para la identificación de un sistema es el siguiente: (Jang, Sun, & Mizutani, 1997)

1. Parametrizar y especificar una clase de modelo matemático que represente el sistema a ser identificado.
2. Realizar la identificación de los parámetros que mejor se ajusten al conjunto de datos de entrenamiento.
3. Hacer pruebas de validación para observar si el modelo que fue identificado responde de manera correcta a un conjunto de datos diferentes empleados en la identificación.
4. Terminar el proceso una vez que los resultados de la prueba de validación sean satisfactorios. Caso contrario se debe seleccionar otra clase de modelos y se repite el paso 2 y 4.

Antes de profundizar en las partes centrales del control neuro-difuso se hará una introducción al método de los mínimos cuadrados para modelos lineales en la identificación de sistema.

2.4.1. Estimador de Mínimos Cuadrados para la Identificación de Sistemas.

El método de los mínimos cuadrados es una herramienta matemática potente y bien desarrollada, ha sido utilizado en diversas áreas por décadas, incluyendo el control adaptativo, procesamiento de señales, estadística, etc. En la actualidad resulta ser una herramienta esencial e indispensable para construir modelos matemáticos lineales. Los sistemas lineales por mínimos cuadrados proporcionan la base matemática más básica e importante para resolver problemas de modelado neuro-difuso. (Picón Martorell, 2005)

La salida de un modelo lineal y es dada por la siguiente expresión de parametrización lineal.

$$y = \theta_1 f_1(\boldsymbol{u}) + \theta_2 f_2(\boldsymbol{u}) + \cdots + \theta_n f_n(\boldsymbol{u}) \tag{0.2}$$

Donde $\boldsymbol{u} = \left[u_1, \ldots, u_p\right]^T$ corresponde al vector de entradas del modelo, $f_1, \ldots, f_n$ están en función del vector de entrada $\boldsymbol{u}$ y se las conoce como funciones conocidas y $\theta_1,\ldots,\theta_n$ son los parámetros desconocidos a ser evaluados. La regresión lineal es utilizada para el ajuste de datos. Por lo tanto, la ecuación 2.2 representa la ecuación de regresión lineal y sus términos $\boldsymbol{\theta}_i$ son conocidos como coeficientes de regresión. Para identificar los parámetros desconocidos $\boldsymbol{\theta}_i$ se realiza un proceso experimental. (Jang, Sun, & Mizutani, 1997)

$$\begin{cases} f_1(\boldsymbol{u}_1)\theta_1 + f_2(\boldsymbol{u}_1)\theta_2 + \cdots + f_n(\boldsymbol{u}_1)\theta_n = y_1, \\ f_1(\boldsymbol{u}_2)\theta_1 + f_2(\boldsymbol{u}_2)\theta_2 + \cdots + f_n(\boldsymbol{u}_2)\theta_n = y_2, \\ \vdots \\ f_1(\boldsymbol{u}_m)\theta_1 + f_2(\boldsymbol{u}_m)\theta_2 + \cdots + f_n(\boldsymbol{u}_m)\theta_n = y_m, \end{cases} \tag{0.3}$$

Al expresar el sistema usando notación matricial se obtiene:

$$\boldsymbol{A\theta} = \boldsymbol{y} \tag{0.4}$$

Dónde ¨$\boldsymbol{A}$¨ una matriz $m \times n$, también conocida como matriz de diseño, ¨$\boldsymbol{\theta}$¨ es un vector de parámetros desconocidos de dimensiones $n \times 1$ y ¨$\boldsymbol{y}$¨ es un vector de salida con dimensiones $m \times 1$.

$$\boldsymbol{A} = \begin{bmatrix} f_1(\boldsymbol{u}_1) & \cdots & f_n(\boldsymbol{u}_1) \\ \vdots & \ddots & \vdots \\ f_1(\boldsymbol{u}_m) & \cdots & f_n(\boldsymbol{u}_m) \end{bmatrix} \quad ; \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad ; \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Del conjunto de datos de la matriz $[\boldsymbol{A} \vdots \boldsymbol{y}]$ la i-enésima fila es representada por $[\boldsymbol{a}_i^T \vdots y_i]$, esta expresión relaciona los pares de datos de entrenamiento de entrada-salida de la siguiente manera: $\boldsymbol{a}_i^T = [f_1(\boldsymbol{u}_i), \cdots, f_n(\boldsymbol{u}_i)]$.

La condición para poder encontrar de manera correcta el vector desconocido $\boldsymbol{\theta}$ es que $m \geq n$ caso contrario si $m = n$ es decir $\boldsymbol{A}$ es cuadrada y no singular (invertible) entonces se resuelve en la ecuación 2.3 como sigue:

$$\boldsymbol{\theta} = \boldsymbol{A}^{-1}\boldsymbol{y} \tag{0.5}$$

Sin embargo, cuando m es mayor que n, esto indica que existen más pares de datos que parámetros de ajuste. Cuando esto sucede no siempre es posible obtener una solución exacta que satisfaga todas las m ecuaciones, debido a que los datos pueden presentar distorsión por efectos de ruido ó el modelo elegido podría ser erróneo para describir el sistema de destino, es por esto que a la ecuación 2.5 se le deben incorporar un vector de error llamado $\boldsymbol{e}$ el cual contemple el ruido o un error de modelado como se presenta en la siguiente ecuación: (Jang, Sun, & Mizutani, 1997)

$$\boldsymbol{A\theta} + \boldsymbol{e} = \boldsymbol{y} \tag{0.6}$$

Se busca que $\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}}$ lo cual minimiza la suma de los errores al cuadrado definidos por la siguiente ecuación.

$$E(\boldsymbol{\theta}) = \sum_{i=1}^{m}(y_i - \boldsymbol{a}_i^T\boldsymbol{\theta})^2 = \boldsymbol{e}^T\boldsymbol{e} = (\boldsymbol{y} - \boldsymbol{A\theta})^T(\boldsymbol{y} - \boldsymbol{A\theta}) \tag{0.7}$$

En la ecuación 2.6, $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{A\theta}$ es el vector error producido por una selección específica de $\boldsymbol{\theta}$. $E(\boldsymbol{\theta})$ tiene una forma cuadrática y tiene un único mínimo en $\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}}$. (Jang, Sun, & Mizutani, 1997)

Teorema del Estimador de Mínimos Cuadrados:

Se minimiza el error cuadrático en la ecuación 2.6, cuando $\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}}$, llamado estimador de mínimos cuadrados o LSE por sus siglas en inglés, el cual satisface a la siguiente ecuación: (Jang, Sun, & Mizutani, 1997)

$$\boldsymbol{A}^T\boldsymbol{A}\widehat{\boldsymbol{\theta}} = \boldsymbol{A}^T\boldsymbol{y} \tag{0.8}$$

Si $\boldsymbol{A}^T\boldsymbol{A}$ es no singular (invertible), el estimador de mínimos cuadrados $\widehat{\boldsymbol{\theta}}$ es único y está dado por:

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y} \tag{0.9}$$

2.4.2. Estimador de Mínimos Cuadrados Recursivo

El estimador de mínimos cuadrados también puede ser expresado de la siguiente manera:

$$\boldsymbol{\theta}_k = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y} \tag{0.10}$$

En la ecuación 2.10 se asume que el número filas de $\boldsymbol{A}$ y $\boldsymbol{y}$ están representadas por k, por esto se añade el subíndice k en dicha ecuación; esto permite indicar el número de pares de datos que usa el estimador $\boldsymbol{\theta}$. Si un nuevo par de datos $(\boldsymbol{a}^T; y)$ está disponible, entonces en lugar de usar todos los conjuntos de datos disponibles para recalcular el estimador de mínimos cuadrados, se desea usar el estimador actual $\boldsymbol{\theta}_k$ para poder obtener el estimador siguiente $\boldsymbol{\theta}_{k+1}$ con un mínimo esfuerzo. Este problema se lo conoce como *Identificación por Mínimos Cuadrados Recursivos*. El estimador siguiente $\boldsymbol{\theta}_{k+1}$ se lo expresa como: (Picón Martorell, 2005)

$$\boldsymbol{\theta}_{k+1} = \left(\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}^T\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}\right)^{-1}\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}^T\begin{bmatrix}\boldsymbol{Y}\\ y\end{bmatrix} \tag{0.11}$$

Para simplificar la notación anterior, se utilizará las matrices $\boldsymbol{P}_k$ y $\boldsymbol{P}_{k+1}$, las cuales quedan identificadas por:

$$\boldsymbol{P}_k = (\boldsymbol{A}^T\boldsymbol{A})^{-1} \tag{0.12}$$

$$\begin{aligned}\boldsymbol{P}_{k+1} &= \left(\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}^T\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}\right)^{-1}\\ &= \left([\boldsymbol{A}^T\ \boldsymbol{a}]\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}\right)^{-1}\\ &= (\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{a}\boldsymbol{a}^T)^{-1}\end{aligned} \tag{0.13}$$

Estas dos matrices se relacionan mediante la siguiente ecuación:

$$\boldsymbol{P}_k^{-1} = \boldsymbol{P}_{k+1}^{-1} - \boldsymbol{a}\boldsymbol{a}^T \tag{0.14}$$

Empleando $\boldsymbol{P}_k$ y $\boldsymbol{P}_{k+1}$ se obtiene lo siguiente:

$$\begin{cases}\boldsymbol{\theta}_k &= \boldsymbol{P}_k\,\boldsymbol{A}^T\boldsymbol{y},\\ \boldsymbol{\theta}_{k+1} &= \boldsymbol{P}_{k+1}(\boldsymbol{A}^T\boldsymbol{y} + \boldsymbol{a}y)\end{cases} \tag{0.15}$$

Suprimiendo el término $\boldsymbol{A}^T\boldsymbol{y}$ de la ecuación 2.15 se puede formular $\boldsymbol{\theta}_{k+1}$ en términos de $\boldsymbol{\theta}_k$, desde el inicio de la ecuación y se obtiene:

$$\boldsymbol{A}^T\boldsymbol{y} = \boldsymbol{P}_k^{-1}\boldsymbol{\theta}_k \tag{0.16}$$

Sustituyendo la ecuación 2.16 en la ecuación 2.15 y aplicando la ecuación 2.14 se tiene:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{P}_{k+1}(\boldsymbol{P}_k^{-1}\boldsymbol{\theta}_k + \boldsymbol{a}y)$$

$$= \boldsymbol{P}_{k+1}[(\boldsymbol{P}_{k+1}^{-1} - \boldsymbol{a}\boldsymbol{a}^T)\boldsymbol{\theta}_k + \boldsymbol{a}y]$$

$$= \boldsymbol{\theta}_k + \boldsymbol{P}_{k+1}\,\boldsymbol{a}(y - \boldsymbol{a}^T\boldsymbol{\theta}_k) \tag{0.17}$$

Por lo tanto $\boldsymbol{\theta}_{k+1}$ puede ser expresada como una función del estimador anterior $\boldsymbol{\theta}_k$ y del nuevo par de datos $(\boldsymbol{a}^T; y)$. El estimador siguiente $\boldsymbol{\theta}_{k+1}$ es igual al estimador actual $\boldsymbol{\theta}_k$ adicionándole un término de corrección, dicho término se basa en el nuevo par de datos $(\boldsymbol{a}^T; y)$. (Jang, Sun, & Mizutani, 1997)

El cálculo de $\boldsymbol{P}_{k+1}$ a partir de la ecuación 2.13 implica el empleo de una matriz invertible de $n \times n$; lo cual ocuparía muchos recursos computacionales, por esta razón se debe encontrar una fórmula gradual para $\boldsymbol{P}_{k+1}$ y aplicarla a la ecuación 2.14, con lo que se obtiene la siguiente ecuación:

$$\boldsymbol{P}_{k+1} = (\boldsymbol{P}_k^{-1} + \boldsymbol{a}\boldsymbol{a}^T)^{-1} \tag{0.18}$$

La siguiente fórmula incremental para $\boldsymbol{P}_{k+1}$ se obtiene al aplicar la fórmula de la matriz inversa al teorema de Lemma donde: $\boldsymbol{A} = \boldsymbol{P}_k^{-1}$; $\boldsymbol{B} = \boldsymbol{a}$ y $\boldsymbol{C} = \boldsymbol{a}^T$.

$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \boldsymbol{P}_k\boldsymbol{a}\,(\boldsymbol{I} + \boldsymbol{a}^T\boldsymbol{P}_k\boldsymbol{a})^{-1}\boldsymbol{a}^T\boldsymbol{P}_k$$

$$= \boldsymbol{P}_k - \frac{\boldsymbol{P}_k\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{P}_k}{1+\boldsymbol{a}^T\boldsymbol{P}_k\boldsymbol{a}} \tag{0.19}$$

En resumen, el estimador de mínimos cuadrados recursivo para el problema de $\boldsymbol{A}\boldsymbol{\theta} = \boldsymbol{y}$, donde: la $k-$ésima fila $(1 \leq k \leq m)$ de $[\boldsymbol{A} \vdots \boldsymbol{y}]$, expresada por $[\boldsymbol{a}_k^T \vdots y_k]$, se obtiene secuencialmente y puede ser calculado de la siguiente manera:

$$\begin{cases} \boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \frac{\boldsymbol{P}_k\boldsymbol{a}_{k+1}\boldsymbol{a}_{k+1}^T\boldsymbol{P}_k}{1+\boldsymbol{a}_{k+1}^T\boldsymbol{P}_k\boldsymbol{a}_{k+1}}, \\ \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{P}_{k+1}\boldsymbol{a}_{k+1}(y_{k+1} - \boldsymbol{a}_{k+1}^T\boldsymbol{\theta}_k) \end{cases} \tag{0.20}$$

Donde k varía desde 0 hasta $m-1$. Por lo general el Estimador de Mínimos Cuadrados total $\widehat{\boldsymbol{\theta}}$ es igual a $\boldsymbol{\theta}_m$, que corresponde al estimador que usa todos los m pares de datos.

Para empezar el desarrollo de la ecuación 2.20 es necesario seleccionar los valores iniciales de $\boldsymbol{\theta}_0$ y $\boldsymbol{P}_0$. Se puede evitar determinar los valores iniciales recogiendo los n

primeros puntos de datos y resolviendo $\boldsymbol{\theta}_n$ y $\boldsymbol{P}_n$ directamente con la siguiente ecuación:

$$\begin{cases} \boldsymbol{P}_n = (\boldsymbol{A}_n^T \boldsymbol{A}_n)^{-1} \\ \boldsymbol{\theta}_n = \boldsymbol{P}_n \boldsymbol{A}_n^T \boldsymbol{y}_n \end{cases} \tag{0.21}$$

Donde $\boldsymbol{A}_n : \boldsymbol{y}_n$ es la matriz compuesta de los primeros n pares de datos. Así se puede iniciar las iteraciones del algoritmo desde el $(n+1)$ *ésimo* par de datos. Sin embargo, algunas veces es más ventajoso utilizar fórmulas recursivas en la ecuación 2.20 durante el proceso de identificación. Para su realización se debe tomar en cuenta que:

$$\boldsymbol{P}_k = (\boldsymbol{P}_0 + \boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} \tag{0.22}$$

y $\boldsymbol{\theta}_k$ es:

$$\boldsymbol{\theta}_k = \boldsymbol{P}_k(\boldsymbol{A}_k \boldsymbol{y}_k + \boldsymbol{P}_0^{-1} \boldsymbol{\theta}_0) \tag{0.23}$$

donde $[\boldsymbol{A}_k : \boldsymbol{y}_k]$ es la matriz de datos, compuesta por k el par de datos. Eligiendo $\boldsymbol{P}_0 = \propto \boldsymbol{I}$ se tiene que:

$$\lim_{\propto \to \infty} \boldsymbol{P}_0^{-1} = \lim_{\propto \to \infty} \frac{1}{\propto} \boldsymbol{I} = 0 \tag{0.24}$$

Por lo tanto, cuando $\propto$ es igual a un número muy grande, las ecuaciones 2.22 y 2.23 se las puede forzar a ser similares a la ecuación 2.20, independientemente de $\boldsymbol{\theta}_0$. En la práctica, $\boldsymbol{\theta}_0$ es usualmente una matriz nula. (Jang, Sun, & Mizutani, 1997)

2.4.3. LSE Recursivo para Sistemas Variantes en el Tiempo

A partir de la ecuación 2.12, se tiene $\boldsymbol{P}_k = (\boldsymbol{A}^T \boldsymbol{A})^{-1}$, donde k es el número de pares de datos encontrados hasta ahora, también representa el número de filas de $\boldsymbol{A}$. Si k es mayor que el número de parámetros de ajuste y los pares de datos contienen gran información, entonces el producto de $\boldsymbol{A}^T \boldsymbol{A}$ usualmente es definido positivo y como k tiende al infinito, el término $\frac{1}{k} \boldsymbol{A}^T \boldsymbol{A}$ se aproxima a una matriz constante no singular (invertible), por lo tanto, se tiene: (Jang, Sun, & Mizutani, 1997)

$$\lim_{k \to \infty} \boldsymbol{P}_k = \lim_{k \to \infty} \frac{1}{k} \left(\frac{1}{k} \boldsymbol{A}^T \boldsymbol{A} \right)^{-1} = 0 \tag{0.25}$$

Lo cual indica que la ganancia de adaptación $\boldsymbol{P}_{k+1} \boldsymbol{a}_{k+1}$ en la ecuación 2.20 disminuye en cada iteración. Esta es una consecuencia directa del error cuadrático definido en la

ecuación 2.7, la cual trata igual a cada componente del error. Para los sistemas invariantes en el tiempo, este trabajo funciona bien, ya que la disminución de la ganancia de adaptación significa que se está acercando al punto óptimo en el espacio de parámetros. Sin embargo, para sistemas variantes en el tiempo, este no es apropiado, ya que la ganancia de adaptación decreciente no puede realizar un seguimiento de los parámetros óptimos. Una alternativa simple para resolver estos problemas es restablecer la matriz $\boldsymbol{P}_k$ a $\boldsymbol{P}_0$ ocasionalmente, ya que el LSE converge rápidamente a los parámetros óptimos actuales. Por supuesto, el tiempo obvio para restablecer $\boldsymbol{P}_k$ es cuando se sospecha que se ha producido un cambio de parámetros significativo. (Jang, Sun, & Mizutani, 1997)

Otra manera para lidiar con sistemas variantes en el tiempo es introducir un factor de olvido λ el cual pone mayor énfasis a los datos que sean más recientes.

$$E(\boldsymbol{\theta}) = \sum_{i=1}^{m} \lambda^{m-i}(y_i - \boldsymbol{a}_i^T\boldsymbol{\theta})^2 \tag{0.26}$$
$$= (\boldsymbol{y} - \boldsymbol{A\theta})^T\boldsymbol{W}(\boldsymbol{y} - \boldsymbol{A\theta})$$

donde $\boldsymbol{W}$ es una matriz diagonal y $0 < \lambda \leq 1$.

$$\boldsymbol{W} = \begin{bmatrix} \lambda^{m-1} & 0 & \cdots & 0 \\ 0 & \lambda^{m-2} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & 1 \end{bmatrix} \tag{0.27}$$

donde $0 < \lambda \leq 1$. Para la ecuación $\widehat{\boldsymbol{\theta}}_w = (\boldsymbol{A}^T\boldsymbol{WA})^{-1}\boldsymbol{A}^T\boldsymbol{Wy}$ el LSE correspondiente que minimiza la medida del error ponderado se define como:

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{A}^T\boldsymbol{WA})^{-1}\boldsymbol{A}^T\boldsymbol{Wy} \tag{0.28}$$

Para obtener la fórmula para un LSE recursivo con un factor de olvido, se debe definir $\boldsymbol{\theta}_k$ nuevamente como:

$$\boldsymbol{\theta}_k = (\boldsymbol{A}^T\boldsymbol{WA})^{-1}\boldsymbol{A}^T\boldsymbol{Wy} \tag{0.29}$$

donde $[\boldsymbol{A} \vdots \boldsymbol{y}]$ contiene los k pares de datos y $\boldsymbol{\theta}_k$ es el LSE que usa estos k pares de datos. Entonces se tiene:

$$\boldsymbol{\theta}_{k+1} = \left(\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}^T \begin{bmatrix}\lambda\boldsymbol{W} & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}\right)^{-1} \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}^T \begin{bmatrix}\lambda\boldsymbol{W} & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{y}\\ y\end{bmatrix} \tag{0.30}$$
$$= (\lambda\boldsymbol{A}^T\boldsymbol{WA} + \boldsymbol{aa}^T)^{-1}(\lambda\boldsymbol{A}^T\boldsymbol{W}y + \boldsymbol{a}y)$$

donde $(\boldsymbol{a}^T; y)$ es el $(k+1)$ é$simo$ par de datos, el cual está disponible. Para simplificar la notación, nuevamente se introducen las variables $\boldsymbol{P}_k$ y $\boldsymbol{P}_{k+1}$, las cuales ahora están ligeramente diferente que antes:

$$\boldsymbol{P}_k = (\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A})^{-1}$$

$$\boldsymbol{P}_{k+1} = \left(\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}^T \begin{bmatrix}\lambda\boldsymbol{W} & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix}\right)^{-1} = (\lambda\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A} + \boldsymbol{a}\boldsymbol{a}^T)^{-1} \quad (0.31)$$

donde $\boldsymbol{P}_k$ y $\boldsymbol{P}_{k+1}$ están relacionados a través de:

$$\lambda\boldsymbol{P}_k^{-1} = \boldsymbol{P}_{k+1}^{-1} - \boldsymbol{a}\boldsymbol{a}^T \quad (0.32)$$

Usando $\boldsymbol{P}_k$ y $\boldsymbol{P}_{k+1}$, se puede reescribir $\boldsymbol{\theta}_k$ y $\boldsymbol{\theta}_{k+1}$ de la siguiente manera:

$$\boldsymbol{\theta}_k = \boldsymbol{P}_k\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{y}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{P}_{k+1}(\lambda\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{y} + \boldsymbol{a}y) \quad (0.33)$$

Si se elimina $\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{y}$ y $\boldsymbol{P}_k$ de las tres ecuaciones anteriores, se obtiene la fórmula recursiva para $\boldsymbol{\theta}_{k+1}$:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{P}_{k+1}\boldsymbol{a}(y - \boldsymbol{a}^T\boldsymbol{\theta}_k) \quad (0.34)$$

Donde $\boldsymbol{P}_{k+1}$ puede ampliarse a partir de la ecuación 2.32 usando la matriz inversa.

$$\boldsymbol{P}_{k+1} = \frac{1}{\lambda}\left(\boldsymbol{P}_k - \frac{\boldsymbol{P}_k\boldsymbol{a}\boldsymbol{a}^T\boldsymbol{P}_k}{\lambda + \boldsymbol{a}^T\boldsymbol{P}_k\boldsymbol{a}}\right) \quad (0.35)$$

Es obvio que la ecuación 2.35 se reduce la ecuación 2.19 si $\lambda = 1$. Si λ es pequeño, los datos recientes son más relevantes y el algoritmo tiene mayor capacidad de rastrear los parámetros variables en el tiempo. Sin embargo, al mismo tiempo los estimadores pueden variar para mostrar efectos de ruidos y perturbaciones. Por lo que el valor de λ depende de la tarea a realizar y es establecido de una manera experimental. (Picón Martorell, 2005)

2.5. Redes Neuronales Adaptativas

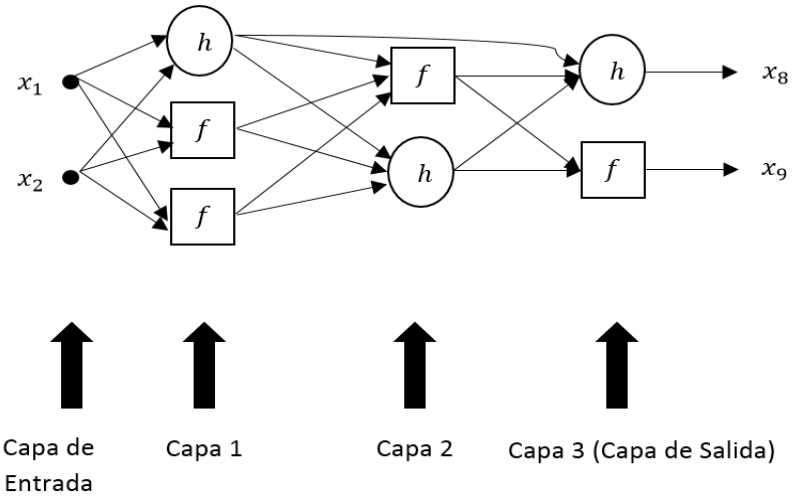
Una red neuronal adaptativa es una estructura de red, la cual consta de nodos interconectados por medio de enlaces unidireccionales. Cada nodo representa una unidad de proceso, el enlace entre nodos especifica la relación causal. Las salidas de los nodos dependen de parámetros modificables por lo que se dice que son nodos adaptativos. Tanto los parámetros modificables como la regla de aprendizaje que se

haya elegido deben ser actualizados para minimizar la medida del error. (Jang, Sun, & Mizutani, 1997)

2.5.1. Arquitectura

Una red adaptativa es una estructura de red heterogénea, en general, su comportamiento de entradas y salidas se determina por la recopilación de parámetros modificables. Una red adaptativa está compuesta por un conjunto de nodos conectados por enlaces unidireccionales, los enlaces indican la dirección del flujo de la señal de un nodo a otro para generar la salida. Cada nodo cumple la función de nodo estático en sus señales de entrada, es decir, la salida de un nodo depende de las entradas actuales y no existe una dinámica o estados internos en cada nodo. Una función de nodo es habitualmente una función parametrizada, que tiene parámetros modificables, cambiando estos parámetros se consiguen diferentes comportamientos, dentro de la red de adaptación. Cada nodo puede tener una función de nodo específico y diferente de los demás. Los enlaces en una red adaptativa simplemente se utilizan para especificar la dirección de propagación de la salida del nodo, por lo general no existen parámetros relacionados con los enlaces. La Figura 2.7 representa una típica red adaptativa con dos entradas y dos salidas. (Jang, Sun, & Mizutani, 1997)

Figura 0.7: Red Adaptativa



Representación de una red adaptativa con dos entradas y dos salidas, (Jang, Sun, & Mizutani, 1997)

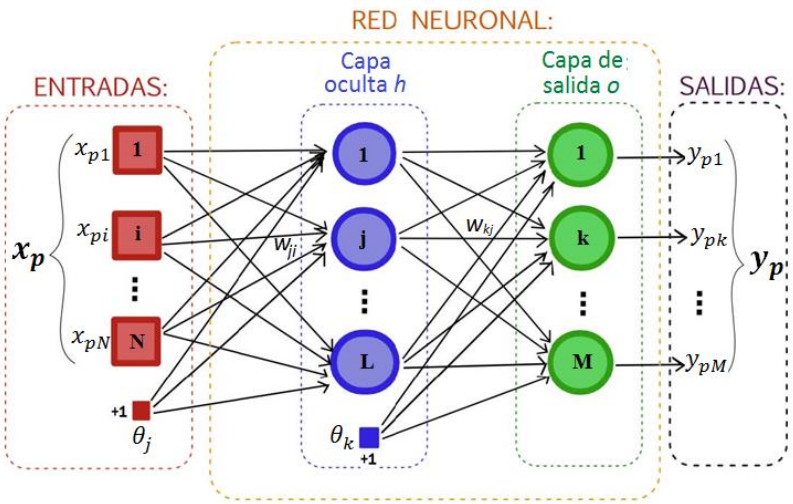
La unión de estos conjuntos de parámetros locales da como resultado un conjunto de parámetros generales de la red. Si el conjunto de parámetros de un nodo no está vacío. (Jang J.-S. R., 2002)

El objetivo de la arquitectura mostrada en la Figura 2.7 es construir una red para lograr un mapeo no lineal deseado, que está regulado por un conjunto de datos que consta de pares de entrada-salida deseadas de un sistema objetivo a ser modelado. Los datos de entrenamiento y los procedimientos utilizados en el ajuste de los parámetros para mejorar el rendimiento de la red se refieren a menudo como las reglas de aprendizaje o algoritmos de adaptación. (Jang J.-S. R., 2002)

2.5.2. Algoritmo de Retro-propagación: Regla Delta Generalizada

Debido a que el error de la capa de salida es el único que puede calcular de manera exacta, el algoritmo de retro-propagación plantea propagar hacia atrás este error para apreciar el error en las salidas de las neuronas de las capas ocultas, con la intención de modificar los pesos sinápticos de estas neuronas. La precisión de la estimación dependerá del número de conjuntos de entrada y de que el tamaño de la muestra sea representativo. Se describirá a continuación las etapas del algoritmo. (Caicedo & Lopez, 2009)

Figura 0.8: Red Neuronal



Representación gráfica de una Red Neuronal, (Nazmul & Hojjat, 2013)

1. Se selecciona un conjunto de datos representativos del sistema para entrenar la red agrupados en un vector de entrada del tipo $\boldsymbol{x}_p = \left[x_{p1}, x_{p2}, \ldots x_{pi}, \ldots, x_{pN}\right]^T$ del cual dependerá la calidad de aprendizaje.
2. Se calcula la entrada neta de la $j - ésima$ neurona de la capa oculta.

$$Neta_{pj}^{h} = \sum_{i}^{N} w_{ji}^{h}\, x_{pi} + \theta_{j}^{h} \tag{0.36}$$

3. Se calcula la salida de la capa oculta.

$$i_{pj}^{h} = f_{j}^{h}\left(Neta_{pj}^{h}\right) \tag{0.37}$$

4. Cuando se han calculado las salidas de las neuronas de la capa oculta, las salidas se convierten en las señales de excitación de las neuronas de la capa de salida con esto se puede calcular la entrada neta de la $k - ésima$ neurona que está en la capa de salida como:

$$Neta_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o}\, i_{pj}^{o} + \theta_{k}^{o} \tag{0.38}$$

5. Basado en la función de activación de la $k - ésima$ neurona de la capa de salida se puede calcular la salida estimada de la red neuronal frente al estímulo de entrada:

$$y_{pk} = f_{k}^{o}(Neta_{pk}^{o}) \tag{0.39}$$

6. Se calcula el error entre el valor de la salida de la red y el valor deseado en cada neurona de la capa de salida.

$$\delta_{pk}^{o} = \left(d_{pk} - y_{pk}^{o}\right) f_{k}^{o'}\left(Neta_{pk}^{o}\right) \tag{0.40}$$

7. Se actualizan los pesos en la capa de salida mediante la siguiente ecuación:

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \eta \delta_{pk}^{o} i_{pj}^{h} \tag{0.41}$$

8. Se actualizan los pesos en la capa oculta mediante la siguiente ecuación:

$$w_{ji}^{h}(t+1) = w_{ji}^{h}(t) + \eta \delta_{pj}^{h} x_{pi} \tag{0.42}$$

9. Se calcula el error global del sistema y se verifica que cumpla con la condición de finalización.

$$E_p = \frac{1}{2} \sum_{p=1}^{P} \sum_{k=1}^{M} (d_{pk} - y_{pk})^2 \tag{0.43}$$

Si no se cumple con las condiciones de finalización del algoritmo ya sea porque el error calculado a la salida no satisface al valor deseado o por que el número de

iteraciones no fue suficiente por lo que la solución no converge por lo tanto se considera hacer ajustes al diseño de la red y presentar un nuevo patrón de entrenamiento. (Caicedo & Lopez, 2009)

2.5.3. Aprendizaje con Término de Momento

Una manera sencilla de incrementar el parámetro de velocidad de aprendizaje, de modo de evitar la inestabilidad, es modificar la regla delta generalizada incluyendo un término de momento. Si el parámetro de aprendizaje η es pequeño el ajuste de pesos que se da de iteración a iteración es pequeño dando como resultado un aprendizaje lento. Por el contrario, si el parámetro de aprendizaje η es grande el aprendizaje es rápido con la desventaja de que la red presente oscilaciones y se vuelva inestable.

Para evitar la posibilidad de inestabilidad se modifica la regla delta mostrada en las ecuaciones 2.41 y 2.42, incluyendo el termino momento en la ecuación de la variación del peso. (Flores López & Fernandéz Fernández, 2008)

$$w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \Delta w_{kj}^{o}(t) \tag{0.44}$$

$$\Delta w_{kj}^{o}(t) = \eta . \delta_{pk}^{o} . y_{pj}^{h} + \alpha \Delta w_{kj}^{o}(t-1) \tag{0.45}$$

Donde α es la constante de momento, generalmente un número positivo. Si en la ecuación 2.44, α=0 entonces se tiene la regla delta generalizada.

El aprendizaje en término de momento variante en el tiempo.

$$\Delta w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \eta . \delta_{pk}^{o} . y_{pj}^{h} + \alpha [w_{kj}^{o}(t) - w_{kj}^{o}(t-1) \tag{0.46}$$

En la ecuación 2.46 además de utilizar el valor de los pesos en el instante t, se utiliza una fracción del valor de los pesos del instante anterior $t-1$, que se controla con el valor de α. (Caicedo & Lopez, 2009)

El mismo procedimiento y tomando en consideración la ecuación 2.42 para la capa oculta se obtienen la siguiente ecuación:

$$\Delta w_{kj}^{o}(t+1) = w_{kj}^{o}(t) + \eta . \delta_{pk}^{o} . x_{pj}^{h} + \alpha [w_{kj}^{o}(t) - w_{kj}^{o}(t-1) \tag{0.47}$$

Sin embargo, se puede realizar diferentes modificaciones como:

- El parámetro de velocidad de aprendizaje η sea dependiente de la conexión y de la iteración $\eta_{ji}(k)$.

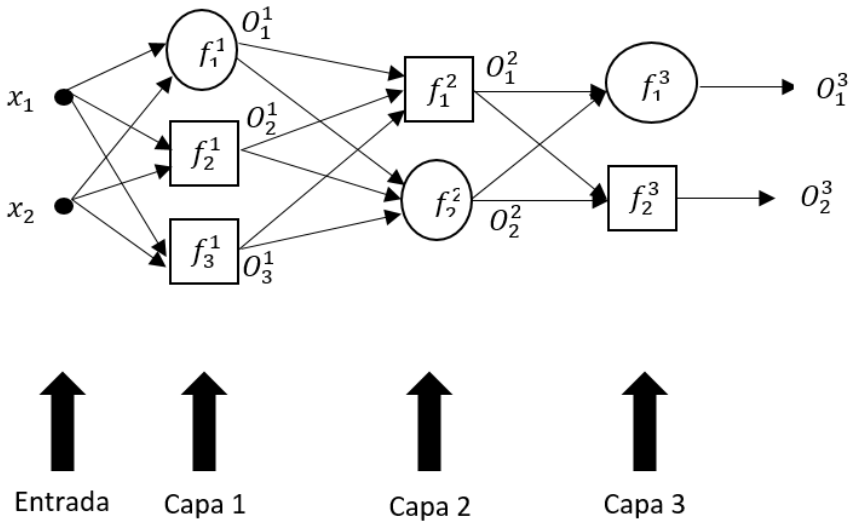
- Para el caso de que se tenga algunos pesos fijos, entonces el parámetro de velocidad de aprendizaje para dichos pesos se puede tomar como $\eta_{ji}(k)$= 0.

2.5.4. Regla de Aprendizaje de Retro-Propagación para una Red Adaptativa.

Se presenta una regla de aprendizaje básica para redes adaptativas anticipadas, su funcionamiento se basa en el método del gradiente descendente explicado en el apartado 2.5.2, la parte central consiste en obtener de forma recursiva el vector gradiente de cada elemento definido como la derivada de la medida del error con respecto a un parámetro mediante la regla de la cadena y la regla de aprendizaje de retro-propagación, ya que el vector gradiente se calcula en dirección opuesta a la salida de cada nodo y se utilizan técnicas de optimización y regresión en base a derivadas para el ajuste de parámetros. (Jang, Sun, & Mizutani, 1997)

Una red adaptativa está compuesta de L capas y la $k-ésima$ capa posee N nodos, entonces la salida y la función del $i-ésimo$ nodo $(i = 1,2,\dots,N)$ en la $k-ésima$ capa se puede representar como o_i^k y f_i^k respectivamente, como se indica en la Figura 2.9.(Jang, Sun, & Mizutani, 1997)

Figura 0.9: Red Neuronal Adaptativa



Representación de las capas de una red neuronal adaptativa, (Jang, Sun, & Mizutani, 1997)

Pasos del algoritmo de retro propagación para una red adaptativa

1. Inicializar α, β, γ que son los parámetros de los nodos.
2. Si la condición de parada es falsa se ejecutan los pasos 3 y 10.
3. Se coloca un vector entrada $\boldsymbol{x}_p = \left[x_{p1}, x_{p2}, \dots x_{pi}, \dots, x_{pN}\right]^T$
4. Se calculan las salidas de la red neuronal

$$O_i^L = f_1^{L-1}, O_2^{L-1}, \dots, O_N^{L-1}, \alpha, \beta, \gamma, \dots) \tag{0.48}$$

$$donde\ N\ es\ el\ número\ de\ nodos\ de\ la\ antepenultima\ capa$$

5. Se calcula el error entre la salida de la red y la salida deseada

$$\epsilon_i^L = \frac{\partial E_p}{\partial O_{i,p}^L} = \frac{\partial}{\partial O_{i,p}^L}\left[\sum_{i=1}^{M}(d_{i,p} - O_{i,p}^L)^2\right] = -(d_{i,p} - O_{i,p}^L) \tag{0.49}$$

6. Se evalúa el error para las capas ocultas

$$\epsilon_i^k = \frac{\partial E_p}{\partial O_{i,p}^k} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \cdot \frac{\partial O_{m,p}^{k+1}}{\partial O_{i,p}^k} \tag{0.50}$$

$$donde\ \#(k+1) representa\ el\ número\ de\ nodos\ de\ la\ capa\ k+1$$

7. Se calcula el vector gradiente de las capas ocultas

$$\frac{\partial E_p}{\partial \alpha} = \frac{\partial E_p}{\partial O_{i,p}^k} \cdot \frac{\partial O_{i,p}^k}{\partial \alpha} \cdot \epsilon_i^k \cdot \frac{\partial O_{i,p}^k}{\partial \alpha} \tag{0.51}$$

8. Se almacena el valor del vector gradiente en el error total

$$\frac{\partial E_p}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha} = \sum_{p=1}^{P} \sum_{O^* \epsilon S} \frac{\partial E_p}{\partial O^*} \cdot \frac{\partial O^*}{\partial \alpha} \tag{0.52}$$

9. Se actualiza el parámetro α

$$\alpha(t+1) = \alpha(t) - \eta \frac{\partial E}{\partial \alpha} \qquad con \quad \eta = \frac{\rho}{\sqrt{\sum_\alpha (\frac{\partial E}{\partial \alpha})^2}} \tag{0.53}$$

10. Se verifica si el error global cumple con las condiciones para finalizar.

$$E = \sum_{p=1}^{P} E_p = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M}(d_{m,p} - O_{m,p}^L)^2 \tag{0.54}$$

2.5.5. Diferentes Maneras de Combinar el Gradiente Descendente y el Estimador de Mínimos Cuadrados.

Los recursos computacionales usados por el estimador de mínimos cuadrados (LSE) son generalmente más altos que la del gradiente descendente (SD) en el proceso de

adaptación. Sin embargo, para lograr un nivel de rendimiento prescrito, el LSE es generalmente mucho más rápido. Por lo que, en función de la disposición los recursos informáticos y el nivel requerido de rendimiento, se puede elegir entre al menos cinco tipos de reglas de aprendizaje híbridas que combinan SD y LSE en diferentes grados, de la siguiente manera: (Jang, Sun, & Mizutani, 1997)

- Una pasada de LSE solamente: los parámetros no lineales son fijos, mientras que los parámetros lineales son identificados por la aplicación de una sola vez de LSE.
- SD solamente: todos los parámetros son tratados como no lineales y actualizados al aplicar una SD iterativa.
- SD y LSE: los parámetros lineales y no lineales se identifican en primer lugar. Cada iteración (época) del SD utilizado para actualizar los parámetros no lineales es seguido por el LSE para identificar los parámetros lineales.
- LSE solamente: las salidas de una red de adaptación se linealizan con respecto a sus parámetros y entonces se emplea el algoritmo de *Kalman*, el método de *Newton Gauss* o el método de *Levenberg-Marquardt* para actualizar todos los parámetros.

La elección de uno de los métodos antes mencionados se basa en un compromiso entre la complejidad y el rendimiento computacional.

2.5.6. Regla de Aprendizaje Híbrida

A pesar de que se puede utilizar por separado el algoritmo de retro-propagación o gradiente descendente para identificar los parámetros en una red adaptativa, un método de optimización simple usualmente toma un largo tiempo antes de converger. Sin embargo, se puede observar que la salida de una red adaptativa (asumiendo que solo hay una) es lineal en algunos de los parámetros de la red, por lo tanto, es posible identificar estos parámetros lineales a través del método de mínimos cuadrados, este enfoque conduce a una regla de aprendizaje híbrida que combina el gradiente descendente y el estimador de mínimos cuadrados para una identificación rápida de los parámetros. (Jang J.-S. R., 2002)

2.5.6.1. Aprendizaje Off- Line (Fuera de Línea o Aprendizaje por Lotes)

Una regla de aprendizaje híbrida la cual combine el método del gradiente y el estimador de mínimo cuadrados para la identificación de parámetros evita inconvenientes propios de solamente usar un método, estos inconvenientes son que el sistema es lento y tiene la probabilidad de quedar atrapado en los mínimos locales. (Jang J.-S. R., 2002)

Por simplicidad, se asume que la red adaptativa bajo consideración tiene sólo una salida representada por

$$out = F(\boldsymbol{i}, S) \tag{0.55}$$

Si existe una función H de tal manera que la función compuesta $H \circ F$ sea lineal en algunos de los elementos de S, entonces estos elementos pueden ser identificados por el método de mínimos cuadrados. Más formalmente, el conjunto de parámetros S se puede dividir en dos subconjuntos.

$$S = S_1 \oplus S_2 \tag{0.56}$$

Donde al reemplazar H a la ecuación 2.56

$$H(out) = H \circ F(\boldsymbol{i}, S) \tag{0.57}$$

que es lineal en S_2. Ahora dando valores a los elementos de S_1, se puede conectar p datos de entrenamiento en la ecuación anterior y obtener una ecuación matricial de la forma:

$$\boldsymbol{A\theta} = \boldsymbol{y} \tag{0.58}$$

por lo tanto, se trata de un problema de mínimos cuadrados lineales estándar y la mejor solución para $\boldsymbol{\theta}$ que minimiza $\|\boldsymbol{A\theta} - \boldsymbol{y}\|^2$ es el estimador de mínimos cuadrados (LSE) $\widehat{\boldsymbol{\theta}}$, definido por:

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y} \tag{0.59}$$

donde $\boldsymbol{A}^T$ es la transpuesta de $\boldsymbol{A}$ y $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$ es la pseudo inversa de $\boldsymbol{A}$ si $\boldsymbol{A}^T\boldsymbol{A}$ es no singular. Por supuesto también se puede emplear la fórmula del LSE recursivo para calcular $\widehat{\boldsymbol{\theta}}$. En concreto si $\boldsymbol{a}_i^T$ *es el* $i-$ *ésimo* vector fila de la matriz $\boldsymbol{A}$ definido en la ecuación $\boldsymbol{A\theta} = \boldsymbol{y}$ y el $i-$ *ésimo* elemento de $\boldsymbol{y}$ sea y_i, entonces $\boldsymbol{\theta}$ se puede calcular de manera interactiva de la siguiente manera:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \boldsymbol{P}_{i+1}a_{i+1}(y_{i+1} - \boldsymbol{a}_{i+1}^T\boldsymbol{\theta}_i)$$

$$\boldsymbol{P}_{i+1} = \boldsymbol{P}_i - \frac{\boldsymbol{P}_i a_{i+1} \boldsymbol{a}_{i+1}^T \boldsymbol{P}_i}{1+\boldsymbol{a}_{i+1}^T \boldsymbol{P}_i a_{i+1}}, \quad i = 0,1, \ldots., p-1 \tag{0.60}$$

donde el estimador de mínimos cuadrados $\widehat{\boldsymbol{\theta}}$ es igual a $\boldsymbol{\theta}_p$.

En el algoritmo de aprendizaje híbrido se ocupan los siguientes pasos: en el paso hacia adelante, después de que se presenta un vector de entrada, se calcula la salida de los nodos en la capa de red, capa por capa, hasta obtener la fila correspondiente a la matriz **A** e $\boldsymbol{y}$ en la ecuación $\boldsymbol{A\theta} = \boldsymbol{y}$. Este proceso se repite para todos los pares de datos de entrenamiento para formar la matriz completa $\boldsymbol{A}$ e $\boldsymbol{y}$; entonces los parámetros en S_2 son identificados por la fórmula de la pseudo inversa en la ecuación 2.59 ó las fórmulas de mínimos cuadrados recursivo en la ecuación 2.60. Después de que los parámetros en S_2 son identificados, el error es medido para cada conjunto de datos de entrenamiento. (Jang, Sun, & Mizutani, 1997)

En el retroceso la señal del error puede ser representado mediante las siguientes ecuaciones 2.49 y 2.50 se propagan desde el extremo de la salida hacia el extremo de la entrada; el vector gradiente es acumulado para cada entrada de datos de entrenamiento. Para el final de paso en retroceso para todos los datos de entrenamiento, los parámetros en S_1 son actualizados por el método de la gradiente descendente en la ecuación $\Delta\alpha = -\eta \frac{\partial^+ E}{\partial \alpha}$.

Ademán pueden ser explorados por el método original del gradiente descendente, sino que también reducirá sustancialmente el tiempo necesario para alcanzar la convergencia. Sin embargo, algunas veces los parámetros en S_2 se ocultan y se requiere de un método de transformación para recuperarlos.

2.5.6.2. Aprendizaje On line (En línea o Aprendizaje Patrón por Patrón)

El aprendizaje en *Online* es una identificación de parámetros para sistemas con características variantes. Para modificar la regla de aprendizaje fuera de línea para obtener una versión en línea, es obvio que el gradiente descendente debería basarse en la ecuación 2.52 en lugar de E. Estrictamente hablando, esto no es realmente un procedimiento de búsqueda de gradiente para minimizar E, sin embargo, se aproximará a uno si la tasa de aprendizaje es pequeña.

Adicionalmente se coloca un factor de olvido λ a la fórmula recursiva original:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \boldsymbol{P}_{i+1}\boldsymbol{a}_{i+1}(\boldsymbol{y}_{i+1}^T - \boldsymbol{a}_{i+1}^T\boldsymbol{\theta}_i$$

$$\boldsymbol{P}_{i+1} = \frac{1}{\lambda}\left[\boldsymbol{P}_i - \frac{\boldsymbol{P}_i\boldsymbol{a}_{i+1}\boldsymbol{a}_{i+1}^T\boldsymbol{P}_i}{\lambda+\boldsymbol{a}_{i+1}^T\boldsymbol{P}_i\boldsymbol{a}_{i+1}}\right] \quad (0.61)$$

donde el valor típico de λ en la práctica esta entre 0.9 y 1. Entre más pequeño sea λ, más rápido será el decaimiento de datos pasados. Sin embargo, un λ pequeño a veces causa inestabilidad numérica, por lo tanto, debe ser evitado. (Jang, Sun, & Mizutani, 1997)

2.6. Modelo ANFIS

Aplicando los conceptos de redes neuronales y lógica difusa se desarrolla el controlador tipo ANFIS por sus siglas en inglés (Adaptive Neuro Fuzzy Inference System) que significa Sistema de Inferencia Neuro Difuso Adaptativo. Es así que un controlador neuro-difuso está estructurado como una red neuronal la cual posee una capacidad de aprendizaje a nivel numérico elevada y la lógica difusa que tiene una gran capacidad de interpretación; al combinar las capacidades propias de las redes neuronales y de la lógica difusa se obtiene gran capacidad de aprendizaje, una excelente interpretación y la incorporación de conocimientos previos. Los valores de las funciones de membresía de los sistemas difusos son aprendidos por las redes neuronales esto permite la construcción de las reglas de inferencia IF-THEN o construir lógica de decisión. (Jang J.-S. R., 2002)

2.6.1. Arquitectura ANFIS

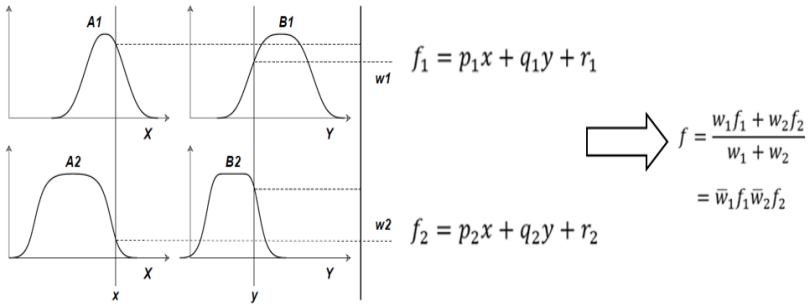
Por simplicidad, se asume que el sistema de inferencia difusa bajo ciertas consideraciones tiene dos entradas x e y, así como una salida f. Para un modelo difuso tipo Sugeno de Primer Orden como el mostrado en la Figura 2.10 se tiene el conjunto de dos reglas difusas $if - then$ como se muestra a continuación.

$$Regla\ 1: If\ x\ is\ A_1\ and\ y\ is\ B_1, then\ f_1 = p_1x + q_1y + r_1 \quad (0.62)$$

$$Regla\ 2: If\ x\ is\ A_2\ and\ y\ is\ B_2, then\ f_2 = p_2x + q_2y + r_2 \quad (0.63)$$

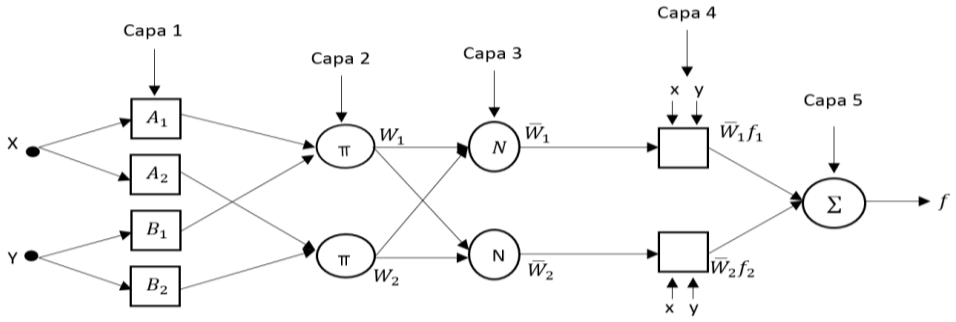
En la Figura 2.10 y 2.11 se representa el mecanismo de razonamiento difuso del modelo tipo Takagi-Sugeno para una red ANFIS. (Jang J.-S. R., 2002)

Figura 0.10: Modelo Sugeno



Mecanismo de razonamiento para este modelo Sugeno, (Jang J.-S. R., 2002)

Figura 0.11: Arquitectura ANFIS



Representación de la arquitectura ANFIS, (Jang, Sun, & Mizutani, 1997)

Donde los nodos de la misma capa tienen funciones similares, como se describe a continuación. Aquí se representa la salida de la $i-ésima$ capa de nodos l como $O_{l,i}$

Capa 1.- Cada nodo es un nodo adaptativo con una función de nodo igual a:

$$O_{l,i} = u_{Ai}(x), \qquad for\ i = 1,2\ or \tag{0.64}$$

$$O_{l,i} = u_{Bi-2}(y), \qquad for\ i = 3,4 \tag{0.65}$$

Donde, $O_{l,i}$ es el grado de pertenencia de un conjunto difuso ($A = A_1, A_2, B_1\ o\ B_2$) y especifica el grado en que la entrada dada ($x\ \ o\ y$) satisface el cuantificador A. Aquí

la función de membresía o pertenencia utilizada para A puede ser cualquier función con parámetros apropiados, tales como la función campana generalizada dada por:

$$u_A(x) = \frac{1}{1+\left|\frac{x-c_i}{a_i}\right|^{2b_i}} \tag{0.66}$$

donde $\{a_i, b_i, c_i\}$ es un conjunto de parámetros. Como cambian los valores de estos parámetros, la forma de la función campana varía en consecuencia, exhibiendo así varias configuraciones de las funciones de membresía para el conjunto difuso A. Los parámetros en esta capa se conocen como *parámetros de premisa.* (Jang, Sun, & Mizutani, 1997)

Capa 2.- Cada nodo en esta capa es un nodo fijo etiquetado con π, cuya salida es el producto de todas las señales entrantes:

$$O_{2,i} = w_i = u_{Ai}(x)u_{Bi}(y), \qquad i = 1,2 \tag{0.67}$$

La salida de cada nodo representa la intensidad de disparo de una regla. En general, cualquier otro operador norma-T que realice una intersección difusa AND puede ser utilizado como la función del nodo en esta capa. (Jang, Sun, & Mizutani, 1997)

Capa 3.- En esta cada los nodos fijos se etiquetan como N. El nodo $i-ésimo$ se encarga de calcular la relación entre la intensidad de disparo de la $i-ésima$ y la regla difusa con respecto a la suma de sus intensidades de disparo.

$$O_{3,i} = \overline{w}_i = \frac{w_i}{w_1+w_2}, \quad i = 1,2 \tag{0.68}$$

Por conveniencia, cada salida de esta capa se conoce como la intensidad de disparo normalizada. (Jang, Sun, & Mizutani, 1997)

Capa 4.- Cada nodo i perteneciente a esta capa es denominado nodo adaptativo y se determinada de la siguiente manera.

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i(p_i x + q_i y + r_i) \tag{0.69}$$

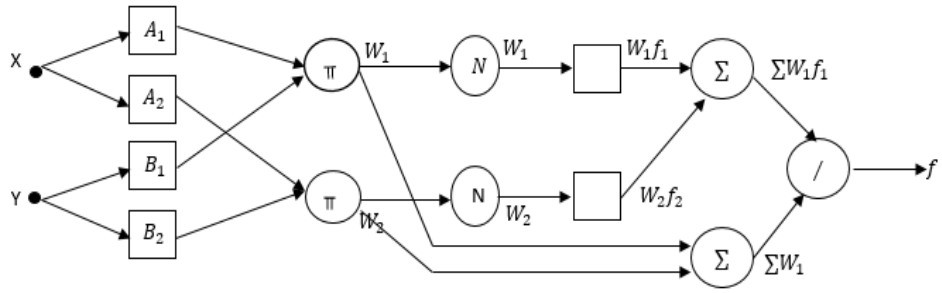
donde $\overline{w}_i$ es la intensidad de disparo normalizada de la Capa 3 y $\{p_i, q_i, r_i\}$ son el conjunto de parámetros conocidos como *parámetros del consecuente*. (Jang, Sun, & Mizutani, 1997)

Capa 5.-El nodo en esta capa es denominado como $\sum$, debido a que realiza el sumatorio de todas las señales entrantes:

$$f = O_{5,1} = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{0.70}$$

De tal manera lo que se obtiene es una red adaptativa que es funcionalmente similar a un modelo difuso Sugeno. Teniendo en cuenta que la estructura de esta red adaptativa no es única; se puede unir las capas 3 y 4 para obtener una red equivalente con sólo cuatro capas. De la misma manera, se puede realizar la normalización del peso en la última capa; en la Figura 2.12 se muestra una red ANFIS de este tipo. En el caso extremo, incluso se puede reducir toda la red en un solo nodo adaptativo con el mismo conjunto de parámetros. Obviamente, la asignación de funciones de nodo y la configuración de red son arbitrarias, como cada nodo y cada capa realiza funciones significativas y modulares. (Jang J.-S. R., 2002)

Figura 0.12: Arquitectura ANFIS y Modelo Difuso Sugeno



Representación de la arquitectura ANFIS con el modelo difuso Sugeno, donde se normalizan los pesos en la última capa, (Jang, Sun, & Mizutani, 1997)

2.6.2. Algoritmo de Aprendizaje Híbrido

Dentro de la arquitectura ANFIS, cuando los valores de los parámetros de premisa son fijos, el total se puede expresar como una combinación lineal de los parámetros del consecuente, se representa de la ecuación 2.55 la salida de la red adaptiva empleando un algoritmo de aprendizaje eficiente para que la representación sea útil en la implementación del control y realizar el ajuste del conjunto de parámetros anterior a partir de una muestra de datos. (Jang, Sun, & Mizutani, 1997)

Su representación general se puede observar en la ecuación 2.71

$$f = \frac{w_1}{w_1 w_2} f_1 + \frac{w_2}{w_1 w_2} f_2$$

$$= \overline{w}_1(p_1x + q_1y + r_1) + \overline{w}_2(p_2x + q_2y + r_2)$$

$$= (\overline{w}_1x)p_1 + (\overline{w}_1y)q_1 + (\overline{w}_1)r_1 + (\overline{w}_2x)p_2 + (\overline{w}_2y)q_2 + (\overline{w}_2)r_2 \quad (0.71)$$

A partir de lo explicado en el apartado 2.5.6.1 se deduce que:

Tabla 0.1: Definición de parámetros.

$S = S_1 \oplus S_2$	Conjunto de parámetros
S_1	Parámetros de premisa
S_2	Parámetros del consecuente

Referencia de los parámetros que se explicaron en el apartado 2.5.6.1

En el paso hacia adelante, las salidas se dirigen con destino a la Capa 4, mientras que a través del algoritmo de mínimos cuadrados se definen los parámetros consecuentes. En el paso hacia atrás, es necesario actualizar los parámetros de premisa usando los datos del error esto es posible mediante el algoritmo de gradiente descendente. (Jang, Sun, & Mizutani, 1997)

Es por esto que existe una convergencia más rápida para el aprendizaje hibrido debido a que los campos de búsqueda se reducen.

En la Tabla 2.2 se realiza una representación de la identificación de los parámetros ANFIS en una red neuronal a través del algoritmo de aprendizaje híbrido. (Jang J.-S. R., 2002)

Tabla 0.2: Procedimiento del aprendizaje híbrido.

	Paso adelantado	Paso retrasado
Parámetros de premisa	Fijo	Gradiente descendente
Parámetros del consecuente	Estimador de mínimos cuadrados	Fijo
Señales	Salidas de los nodos	Señales de error

Representación de dos pasos para el desarrollo del aprendizaje híbrido para ANFIS, Nicolás.

CAPITULO III

DESCRIPCIÓN DE LA PLANTA

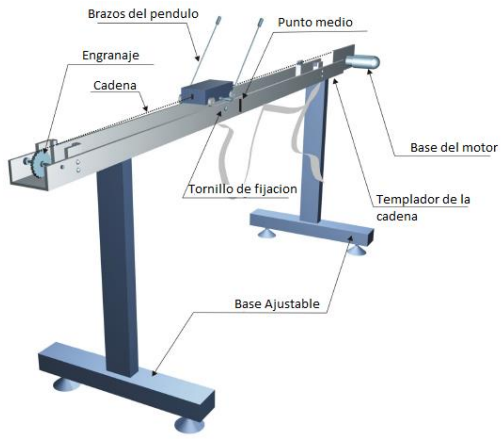
3.1. Introducción

En este capítulo se presenta la caracterización de la planta péndulo invertido y de sus componentes, además se realiza el modelado matemático de la planta necesario para implementarlo en el entorno de Simulink y encontrar su función de transferencia para implementar el algoritmo de control.

3.2. Descripción de la Planta Péndulo Invertido.

El péndulo invertido está constituido por una pieza móvil conocida también como carro móvil, la cual se desplaza a lo largo de una pista de 1 metro de longitud. En el eje del carro móvil están sujetos dos péndulos que son capaces de moverse libremente. El carro se mueve de una manera horizontal hacia atrás y hacia adelante, este movimiento genera una oscilación en el péndulo. El carro móvil se mueve por la tracción de la cadena que pasa por el centro del mismo y esto permite el movimiento en ambas direcciones, la cadena esta acoplada a un motor DC que se encuentra ubicado al final del riel, tal como se muestra en la Figura 3.1. (FeedBack Insruments Ltd., 2006)

Figura 0.1: Péndulo Invertido



Estructura mecánica del péndulo invertido, (FeedBack Insruments Ltd., 2006)

El voltaje aplicado al motor será considerado como la señal de control; se obtendrá otras dos señales del péndulo usando encoders ópticos, estas dos señales son la posición angular del péndulo (θ) y la posición del carro móvil en el riel (x), de estas dos variables depende la salida de control.

3.3. Modelamiento Matemático del Péndulo

Para la realización de un controlador es primordial realizar el modelamiento de la planta a controlar, con esto se busca obtener toda la información que sea posible acerca del proceso.

Se empezará por presentar el modelo mecánico del péndulo en la Figura 3.2 y las variables de estado en la Tabla 3.1.

Figura 0.2: Diagrama Mecánico

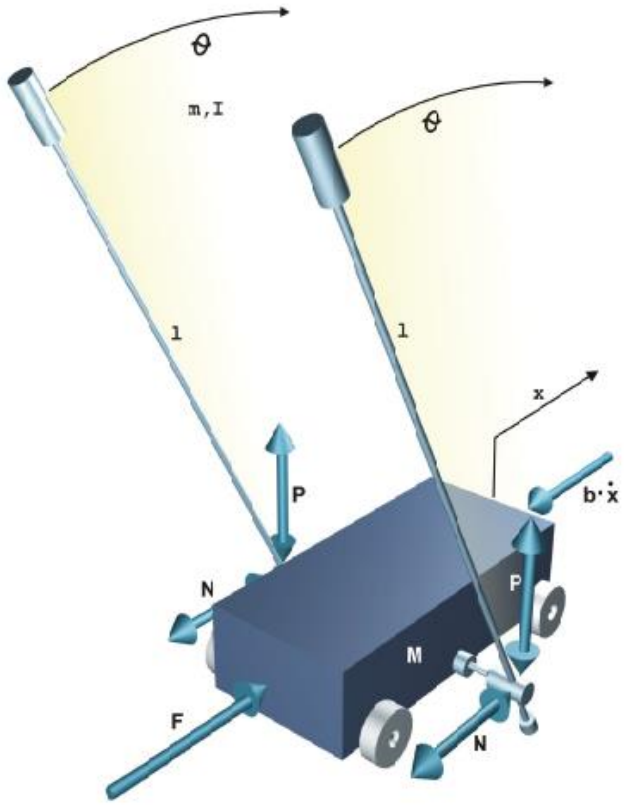


Diagrama mecánico del péndulo invertido, (FeedBack Insruments Ltd., 2006)

Tabla 0.1: Descripción de las variables de estado del péndulo invertido.

Símbolo	Parámetro
θ	Ángulo del brazo del péndulo respecto a la línea vertical P.
F	Fuerza horizontal aplicada al carro móvil para posicionar el péndulo.
N	Componente horizontal de la fuerza de reacción en el punto de pivote.
P	Componente vertical de la fuerza de reacción en el punto de pivote.
x	Coordenada de posición del carro móvil.
I	Momento de inercia del péndulo.
b	Coeficiente de fricción del carro móvil.
$b.\dot{x}$	Fuerza de fricción ejercida sobre el carro móvil.

m	Masa de la barra.
M	Masa del carro.
l	Longitud del brazo del péndulo.
g	Gravedad.

Variables del péndulo invertido, (FeedBack Insruments Ltd., 2006)

El sistema del péndulo invertido es no lineal, lo que significa que por lo menos uno de sus estados (x y su derivada $\dot{x}$) o (θ y su derivada $\dot{\theta}$) es parte de una función no lineal. Para representar un sistema no lineal mediante una función de transferencia dicho sistema debe ser linealizado. Para linealizar el sistema del péndulo invertido este proyecto tomará como referencia la segunda ley de Newton. (FeedBack Insruments Ltd., 2006)

3.3.1. Análisis del Movimiento Angular del Péndulo

Para empezar el análisis del movimiento angular se definen las coordenadas en (x, y) del centro de gravedad de la barra del péndulo como (x_G, y_G) tal como se muestra en la Figura 3.3 que corresponde al diagrama de cuerpo libre del péndulo invertido. (FeedBack Insruments Ltd., 2006)

Figura 0.3: Diagrama de cuerpo libre

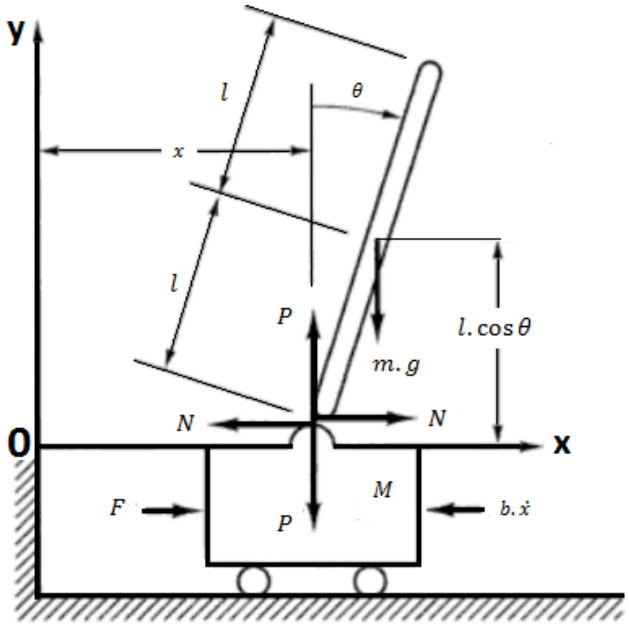


Diagrama de cuerpo libre del péndulo invertido, (Ogata, 2003)

$$x_G = x + l.\sin\theta$$

$$y_G = l.\cos\theta \tag{0.1}$$

Considerando el diagrama el cuerpo libre de la Figura 3.3 y el movimiento rotacional que tiene la barra del péndulo alrededor de su centro de gravedad es posible obtener la ecuación de movimiento del sistema de la siguiente manera:

$$I.\ddot{\theta} + d.\dot{\theta} = P.l.sen\theta - N.l.cos\theta \tag{0.2}$$

La ecuación 3.2 se planteó asumiendo que el sentido de giro del ángulo θ gira en sentido horario. El término $d.\dot{\theta}$ es el momento de fricción el cual se genera en el movimiento de rotación y d es el coeficiente de amortiguamiento. Aunque para este sistema d tiene un valor insignificante es necesario para el modelamiento. (FeedBack Insruments Ltd., 2006)

3.3.2. Análisis del Movimiento Rectilíneo del Péndulo y el Carro Móvil

El movimiento horizontal del centro de gravedad del péndulo está dado por la siguiente ecuación:

$$m.\ddot{x} + m.l.\frac{d^2}{dt^2}\ (sin\,\theta) = N \tag{0.3}$$

El movimiento vertical del centro de gravedad del péndulo esta expresado mediante la siguiente ecuación:

$$m.l.\frac{d^2}{dt^2}\ (cos\theta) = P - (m.g) \tag{0.4}$$

Al aplicar la regla de la cadena en los términos de $\frac{d^2}{dt^2}\ (sin\,\theta)$ y $\frac{d^2}{dt^2}\ (cos\theta)$ en las ecuaciones 3.3 y 3.4, se tiene:

Para la ecuación 3.3

$$\frac{d^2}{dt^2}\ (sin\,\theta) = -\dot{\theta}^2.sin\,\theta + \ddot{\theta}.cos\,\theta \tag{0.5}$$

Para la ecuación 3.4

$$\frac{d^2}{dt^{\,2}}\ (\cos\theta) = -\left[\dot{\theta}^2.\cos\theta + \ddot{\theta}.\sin\theta\right] \tag{0.6}$$

Reemplazando la ecuación 3.5 en la ecuación 3.3 se obtiene:

$$N = m.\dot{x} + m.l.\ddot{\theta}.cos\,\theta - m.l.\dot{\theta}^2.sin\,\theta \tag{0.7}$$

Reemplazando la ecuación 3.6 en la ecuación 3.5 se obtiene:

$$P = m.g - m.l.\dot{\theta}^2.cos\,\theta - m.l.\ddot{\theta}.sin\,\theta \tag{0.8}$$

Para describir el movimiento horizontal del carro móvil se tiene:

$$F = M.\ddot{x} + b.\dot{x} + N \tag{0.9}$$

Tomando en consideración la combinación del movimiento angular y rectilíneo, además considerando las fuerzas sobre el péndulo y el carro móvil se obtienen las ecuaciones no lineales de movimiento como se muestra a continuación:

Reemplazando la ecuación 3.7 en la ecuación 3.9, se obtiene:

$$F = (M+m)\ddot{x} + b.\dot{x} + m.l.\ddot{\theta}.\cos\theta - m.l.\dot{\theta}^2.\sin\theta \tag{0.10}$$

Reemplazando las ecuaciones 3.7 y 3.8 en la ecuación 3.3, se tiene:

$$0 = (I + m.l^2)\ddot{\theta} - m.g.l.\sin\theta + m.l.\ddot{x}.\cos\theta + d.\dot{\theta} \tag{0.11}$$

El modelo establecido por las ecuaciones de movimiento 3.10 y 3.11 tienen los siguientes valores de parámetros como se indican en la Tabla 3.2.

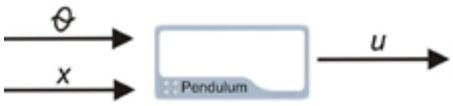
Tabla 0.2: Valores de las variables influyentes en el modelado del péndulo.

Símbolo	Parámetro	Valor
I	Momento de inercia del péndulo.	Aproximadamente 0.099 $Kg.m^2$ Depende de su configuración.
b	Coeficiente de fricción del carro móvil.	0.05 Ns/m
d	Coeficiente de amortiguación.	0.005 Nms/rad
m	Masa del péndulo.	0.23 Kg
M	Masa del carro.	2.4 Kg
l	Longitud del brazo del péndulo.	0.36-0.40 m Depende de su configuración
g	Gravedad.	9.8 m/s

Nota: Parámetros del Péndulo, (FeedBack Insruments Ltd., 2006)

El péndulo invertido es una planta MISO, lo que significa que la planta tiene múltiples entradas y una salida como se lo representa en la Figura 3.4.

Figura 0.4: Planta MISO



Planta SIMO una entrada varias salidas, (FeedBack Insruments Ltd., 2006)

El modelo que describe las ecuaciones 3.10 y 3.11, no determina la relación entre la fuerza F y la señal de control real. La señal de control real corresponde al voltaje u suministrado por la tarjeta de control (PCI-1711). Si se asume que la relación entre la señal de control u y la velocidad generada del carro móvil es lineal, se podría añadir el vector velocidad generado por el motor y obviar el vector F o interpretar la señal

de voltaje u como la fuerza F que se necesita generar, con la consideración que se tiene un voltaje constante que moverá el carro con una velocidad constante.

$$F = k_{Fu}.\frac{du}{dt} \tag{0.12}$$

La ecuación 3.12 describe la relación entre la señal de voltaje de control u y la fuerza del carro móvil. Donde k_{Fu} es la ganancia entre la derivada del voltaje u y la fuerza F.

3.3.3. Modelado del Sistema Péndulo Invertido en Simulink

Partiendo de las ecuaciones 3.10 y 3.11 se asumen los valores de los parámetros expuestos en la Tabla 3.2 y se despejan los términos $\ddot{x}$ y $\ddot{\theta}$, con dichos términos se obtiene dos nuevas ecuaciones con las que se determinará el diagrama de bloques que representará el modelado de la planta el cual se muestra en la Figura 3.5.

$$\ddot{x} = \frac{F+m.l.\dot{\theta}^2.\sin\theta - m.l.\ddot{\theta}.\cos\theta - b\ .\dot{x}}{m+M} \tag{0.13}$$

$$\ddot{\theta} = \frac{m.g.l.\sin\theta - m.l.\ddot{x}.\cos\theta - d.\dot{\theta}}{m.l^2+I} \tag{0.14}$$

Figura 0.5: Diagrama de bloques

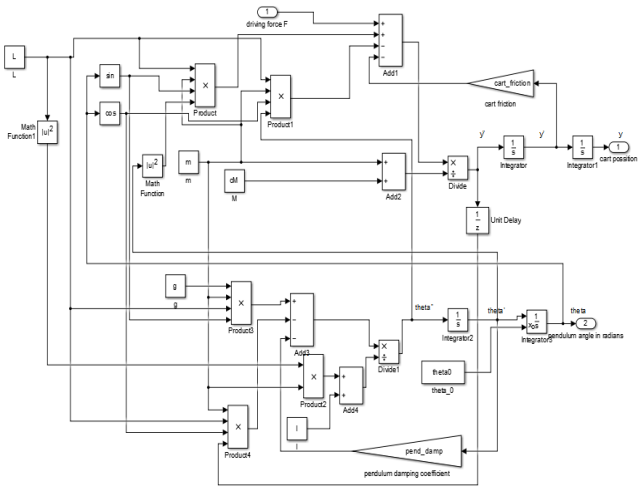


Diagrama de bloques del sistema péndulo invertido, (FeedBack Insruments Ltd., 2006)

3.4. Linealización del Modelo

Los modelos no lineales necesitan linealización y obtener su función de transferencia con esto se puede realizar un controlador para este tipo de sistemas. Esta linealización es válida solo para pequeñas desviaciones de los valores de estado a partir de su valor nominal. El valor nominal es llamado punto de equilibrio. En el específico caso del péndulo invertido se tiene dos puntos de equilibrio que son cuando $\theta = 0$ (péndulo invertido) y $\theta = \pi$ (control de grua).

La linealización del modelo fenomenológico del péndulo invertido se puede empezar al sustituir las funciones no lineales en las ecuaciones 3.10 y 3.11 por su equivalente lineal. Dicha linealización de un punto de trabajo es realizada mediante el método de aproximación de Taylor de las funciones no lineales.

Cuando el ángulo se encuentra en un punto de equilibrio es decir $\theta_0 = 0$, y existen pequeñas desviaciones del ángulo θ con respecto al punto de equilibrio, se supondrá que las funciones $\sin\theta$ y $\cos\theta$ pueden ser linealizadas de la siguiente manera:

$$\sin\theta = \sin\theta|_{\theta=\theta_0=0} + \frac{d\sin\theta}{d\theta}\Big|_{\theta=\theta_0=0} .(\theta - \theta_0) \cong \theta - \theta_0 \tag{0.15}$$

$$\cos\theta = \cos\theta|_{\theta=\theta_0=0} + \frac{d\cos\theta}{d\theta}\Big|_{\theta=\theta_0=0} .(\theta - \theta_0) \cong 1 \tag{0.16}$$

$$\dot{\theta}^2 = 0 \tag{0.17}$$

Al reemplazar los resultados de las ecuaciones 3.15, 3.16 y 3.17, en las ecuaciones 3.10 y 3.11 se obtiene:

$$F = (M + m)\ddot{x} + b.\dot{x} + m.l.\ddot{\theta} \tag{0.18}$$

$$0 = (I + m.l^2)\ddot{\theta} - m.g.l.(\theta - \theta_0) + m.l.\ddot{x} + d.\dot{\theta} \tag{0.19}$$

Considerando que las variables de desviación están dadas por:

$$\theta = \theta - \theta_0 \tag{0.20}$$

$$X = x - x_0 \tag{0.21}$$

Reemplazando las ecuaciones 3.20 y 3.21 en las ecuaciones 3.18 y 3.19 respectivamente y considerando que x_0 es constante se tiene:

$$F = (M + m)\ddot{X} + b.\dot{X} + m.l.\ddot{\theta} \tag{0.22}$$

$$0 = (I + m.l^2)\ddot{\theta} - m.g.l.\theta + m.l.\ddot{X} + d.\dot{\theta} \tag{0.23}$$

Las ecuaciones 3.22 y 3.23 únicamente son válidas para variaciones pequeñas respecto a la posición donde $\theta_0 = 0$. Para la posición de $\theta_0 = \pi$, es necesario realizar las siguientes sustituciones:

$$\sin\theta = \sin\theta|_{\theta=\theta_0=\pi} + \frac{d\sin\theta}{d\theta}\Big|_{\theta=\theta_0=\pi} .(\theta - \theta_0) \cong -(\theta - \theta_0) \quad (0.24)$$

$$\cos\theta = \cos\theta|_{\theta=\theta_0=\pi} + \frac{d\cos\theta}{d\theta}\Big|_{\theta=\theta_0=\pi} .(\theta - \theta_0) \cong -1 \quad (0.25)$$

$$\dot{\theta}^2 = 0 \quad (0.26)$$

Al realizar el procedimiento anterior y usando las variables de deviación, las ecuaciones de movimiento 3.10 y 3.11 tendrán la siguiente forma:

$$F = (M + m)\ddot{X} + b.\dot{X} - m.l.\ddot{\theta} \quad (0.27)$$

$$0 = (I + m.l^2)\ddot{\theta} + m.g.l.\theta - m.l.\ddot{X} + d.\dot{\theta} \quad (0.28)$$

El modelo lineal del péndulo, así como el modelo no lineal, constan de una entrada que es la fuerza F y dos salidas que son el ángulo del péndulo θ y la posición del carro móvil x. Sin embargo, la tarea del péndulo invertido es la estabilización del ángulo razón por la cual es posible considerar la posición del carro móvil x como una salida incontrolada.

Con una entrada F y la salida θ, se pueden obtener dos modelos lineales para cada una de las desviaciones del ángulo θ que son sus puntos de equilibrio $\theta_0[0, \pi]$ en forma de funciones de transferencia. La relación entre el voltaje de control y la fuerza esta descrita en la ecuación 3.12.

Cuando se diseñan controladores se debe tener en cuenta dos sucesos cuando es una aplicación de tiempo real se encuentra delimitada la posición del carro y la señal de control, la posición del carro es delimitada físicamente por la longitud del carril igual de -0.5m a 0.5m y la señal de control de -2.5V a +2.5V generando una magnitud de fuerza de aproximadamente de -20.0N a 20.0N.

3.5. Función de Transferencia del Sistema Péndulo Invertido

La función de transferencia en teoría de control se usa para relacionar las entradas y salidas de un componente o un sistema los cuales son descritos por medio de ecuaciones diferenciales lineales invariantes en el tiempo.

Se puede definir la función de transferencia de un sistema descrito mediante una ecuación diferencial lineal que no varía con el tiempo como el cociente entre la transformada de Laplace de la función de salida y la transformada de la place de la función de entrada suponiendo que todas las condiciones iniciales son cero. (Ogata, 2010)

$$a_n\frac{d^n y(t)}{dt^n}+a_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}}+\cdots+a_1\frac{dy(t)}{dt}+a_0 y(t) = b_m\frac{d^m y(t)}{dt^m}+b_{m-1}\frac{d^{m-1}y(t)}{dt^{m-1}}+\cdots+b_1\frac{dy(t)}{dt}+b_0 y(t)\; n \geq m$$

Función de transferencia = $G(s)=\left.\frac{\mathcal{L}[salida]}{\mathcal{L}[entrada]}\right|_{condicion\ inicial\ nula}$

Partiendo del anterior concepto de función de transferencia se puede representar la dinámica de un sistema mediante ecuaciones algebraicas en función de s. Aplicando la transformada de Laplace a la ecuación 3.27 de movimiento y despejando $X(s)$ se obtiene:

$$F(s)=(m+M).s^2.X(s)+b.s.X(s)+m.l.s^2.\theta(s) \quad (0.29)$$

$$X(s)=\frac{F(s)-m.l.s^2.\theta(s)}{(m+M)s^2+b.s} \quad (0.30)$$

Ahora al aplicar la transformada de Laplace a la ecuación 3.23 de movimiento se tiene:

$$0=(I+m.l^2).s^2.\theta(s)-m.g.l.\theta(s)+m.l.s^2.X(s)+d.s.\theta(s)$$

$$0=(I+m.l^2).s^2.\theta(s)-m.g.l.\theta(s)+m.l.s^2.\frac{F(s)-m.l.s^2.\theta(s)}{(m+M)s^2+b.s}+d.s.\theta(s)$$

$$-m.l.s^2.F(s)=\left[[(m+M)(l+m.l^2)-m^2l^2]s^4+[b(l+m.l^2)+d(m+M)]s^3+[b.d-m.g.l(m+M)]s^2-m.g.l.b.s\right]\theta(s)$$

$$m.l.s^2F(s)=\left[[m^2l^2-(m+M)(l+m.l^2)]s^4-[b(l+m.l^2)+d(m+M)]s^3+[m.g.l(m+M)-b.d]s^2+m.g.l.b.s\right]\theta(s) \quad (0.31)$$

$$\frac{\theta(s)}{F(s)}=\frac{m.l.s^2}{[m^2l^2-(m+M)(l+m.l^2)]s^4-[b(l+m.l^2)+d(m+M)]s^3+[m.g.l(m+M)-b.d]s^2+m.g.l.b.s}$$

$$\frac{\theta(s)}{F(s)}=\frac{s}{\left[m.l-(m+M)\left(\frac{I}{m.l}+1\right)\right]s^3-\left[\frac{d}{m.l}(m+M)+b\left(\frac{I}{m.l}+1\right)\right]s^2+\left[g(m+M)-\frac{b.d}{m.l}\right]s+b.g} \quad (0.32)$$

Se despeja $\theta(s)$ de la ecuación 3.33 y se obtiene:

$$\theta(s)=\frac{m.l.s^2.X(s)}{(I+m.l^2).s^2+d.s-m.g.l} \quad (0.33)$$

Reemplazando $\theta(s)$ en ecuación 3.36 se obtiene la función de transferencia del sistema.

$$\frac{X(s)}{F(s)} = \frac{s}{\left[m.l-(m+M)\left(\frac{l}{m.l}+l\right)\right]s^3-\left[\frac{d}{m.l}(m+M)+b\left(\frac{l}{m.l}+l\right)\right]s^2+\left[g(m+M)-\frac{b.d}{m.l}\right]s+b.g}.\frac{\left(l+m.l^2\right)s^2+d.s-m.l.g}{-m.l.s^3}$$

$$\frac{X(s)}{F(s)} = \frac{m.l.s^2}{[m^2l^2-(m+M)(I+m.l^2)]s^4-[b(l+m.l^2)+d(m+M)]s^3+[m.g.l(m+M)-b.d]s^2+m.g.l.b.s}.\frac{\left(I+m.l^2\right)+d.s-m.l}{-m.l.s^2} \quad (0.34)$$

CAPITULO IV

IMPLEMENTACIÓN Y SIMULACIÓN

4.1. Introducción

En este capítulo se presenta la implementación del sistema neuro-difuso tipo ANFIS por medio del software Matlab, desarrollando un código de programación y un programa en Simulink para realizar las pruebas de simulación. Estas dos plataformas se relacionan mediante un bloque especial llamado S-function (la información detallada de su creación y configuración se encuentran en el Anexo 1); este bloque es parte principal de la red neuro-difusa tipo ANFIS que se aplicará sobre la planta del péndulo invertido, para obtener datos de la planta de acuerdo al tipo de control al que sea sometido, ya sea un PID convencional o un control neuro-difuso tipo ANFIS.

4.2. Algoritmo de un Sistema de Inferencia Neuro-Difusa Adaptativa (ANFIS)

Para la implementación del sistema de inferencia neuro-difuso adaptativo ANFIS para un sistema MISO (Sistema de Múltiples entradas y Una Salida), se utilizó el método de partición de entradas tipo rejilla, método que se encuentra detallado en el apartado 2.3, así como también se utiliza el algoritmo de aprendizaje híbrido mediante el cual se obtienen los parámetros no lineales a través del algoritmo del gradiente descendente utilizando el error de retro-Propagación, mientras que los parámetros lineales son estimados utilizando el algoritmo de Mínimos Cuadrados Recursivos, el detalle del algoritmo de aprendizaje híbrido se encuentra en el apartado 2.6.2.

La lógica de programación del algoritmo ANFIS se ha desarrollado mediante un código de programación en el software Matlab, mientras que para su implementación se utilizó el entorno de programación visual Simulink que se encuentra incluido dentro del mismo software. Para la comprobación del algoritmo ANFIS es necesario comunicar los dos entornos de programación a través de un bloque S-function.

En la Figura 4.1, se presenta un organizador gráfico el cual indica la forma secuencial del algoritmo del controlador Neuro-Difuso ANFIS.

Figura 0.1: Diagrama de flujo principal

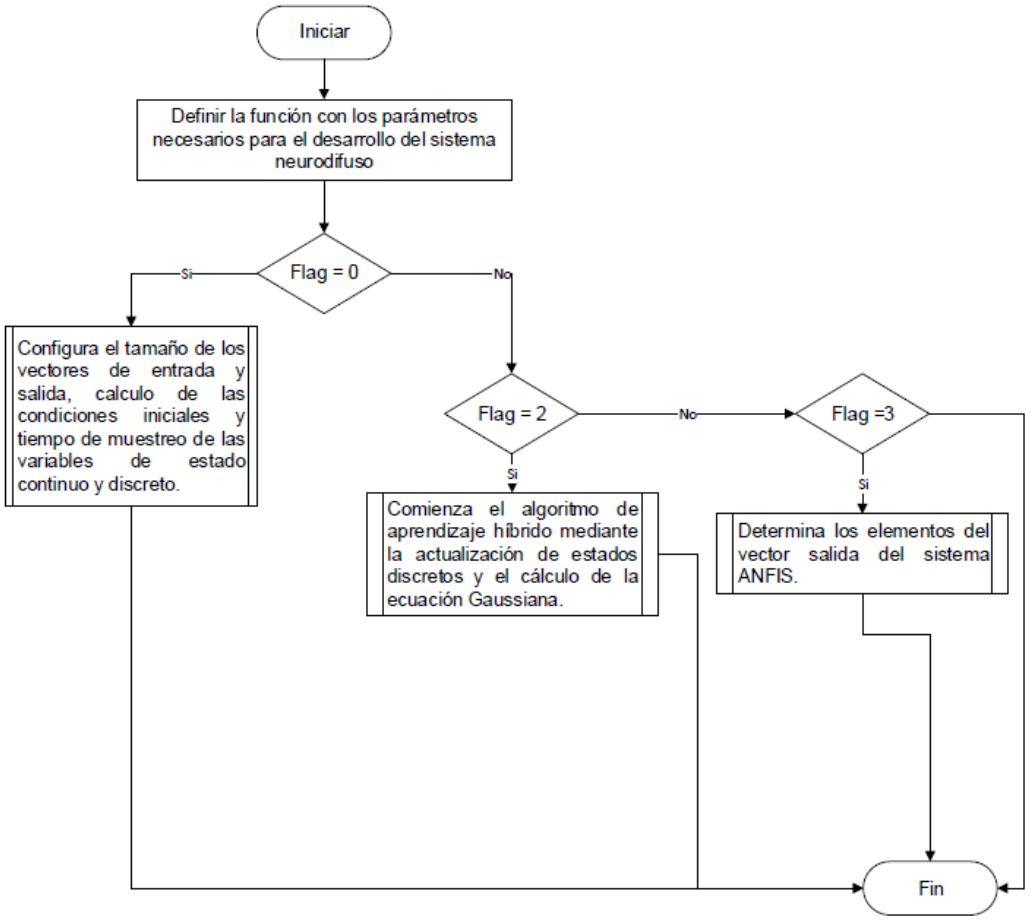


Diagrama general del algoritmo del controlador neuro-difuso ANFIS, Diego Rocha, María José Jara

Una vez expuesto el diagrama de flujo general del algoritmo del controlador ANFIS se procede a realizar organizadores gráficos específicos de cada subproceso basados en el valor que adquiera la variable *flag*, estos valores deben ser números enteros que van desde el 0 hasta el 6 y dependiendo de su valor le permite realizar una tarea específica a la S-function.

Cuando la variable *flag* toma un valor de cero *(flag=0)* se determina el tamaño de los vectores entrada y salida, las condiciones iniciales y las características básicas de la S-Function, así como se representa el subproceso de la Figura 4.2.

Figura 0.2: Subproceso del algoritmo ANFIS

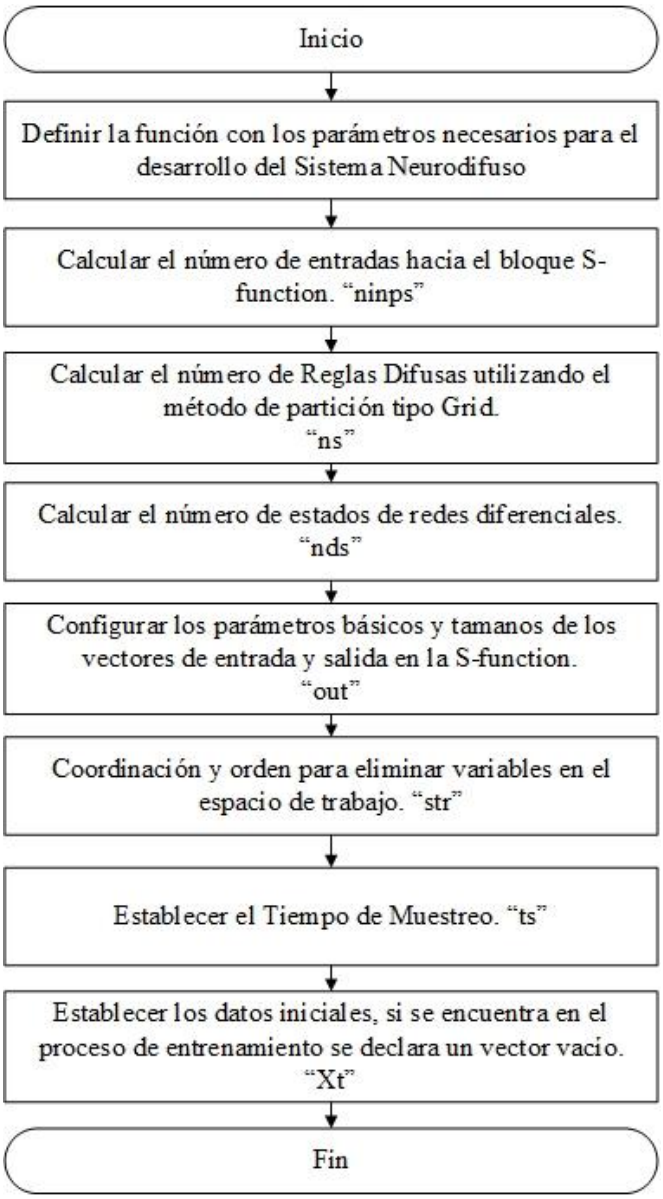
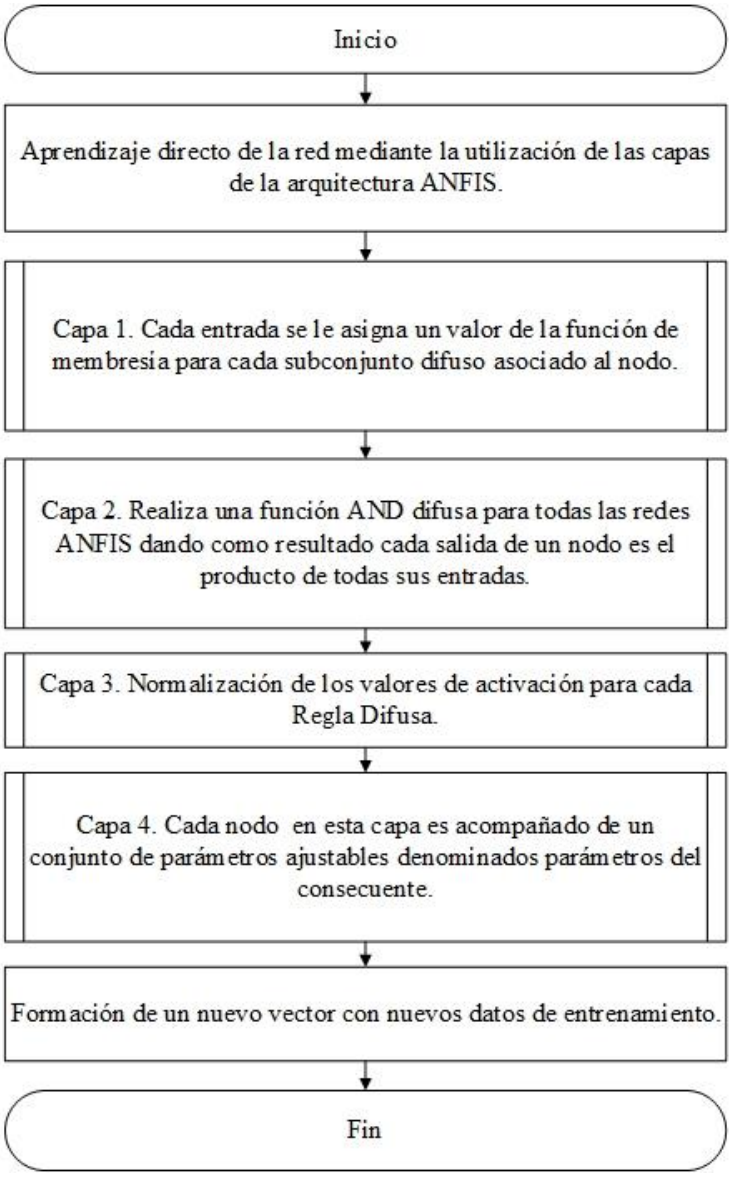


Diagrama de flujo de la definición y parametrización de variables iniciales, Diego Rocha. María Jose Jara

La implementación de la red neuronal se realiza mediante el algoritmo de aprendizaje Híbrido On-line (detallado en el apartado 2.5.6.2) y la arquitectura ANFIS la cual

consta de cinco capas que desarrollan un procedimiento específico para el controlador neuro-difuso. En la Figura 4.3 se indica un organizador gráfico con todo el procedimiento para la actualización de los parámetros de la red ANFIS obteniendo los datos de entrenamiento.

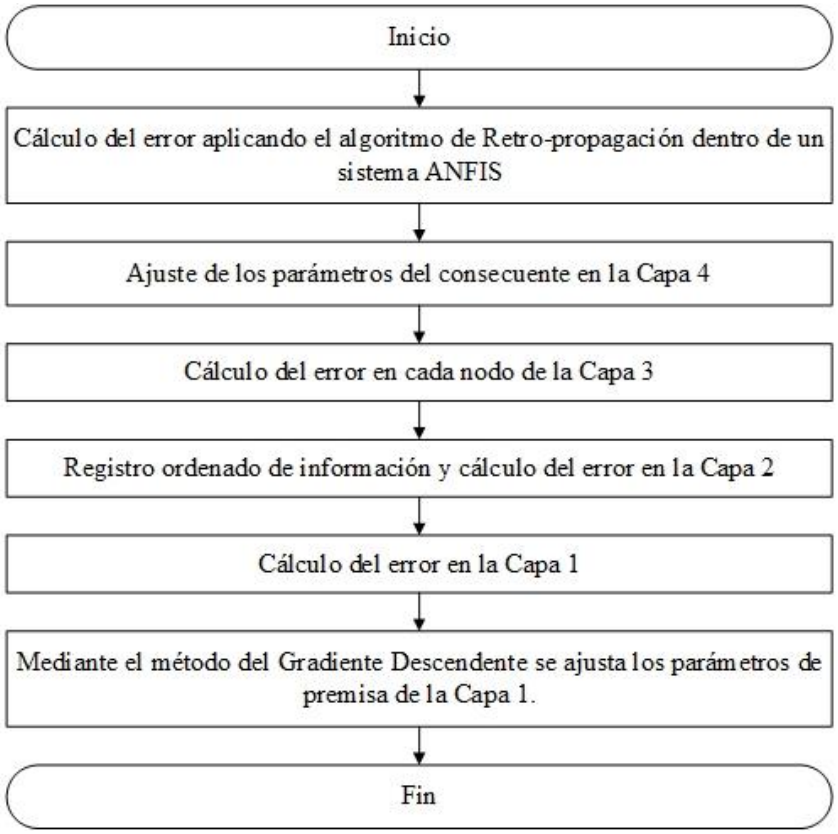
Figura 0.3: Subproceso del algoritmo ANFIS



Implementacion de la red ANFIS, Diego Rocha, María Jose Jara

En la Figura 4.4 se representa el algoritmo de error de Retro-Propagación, el mismo que se detalló en el apartado 2.5.2 para obtener el cálculo del error en cada capa.

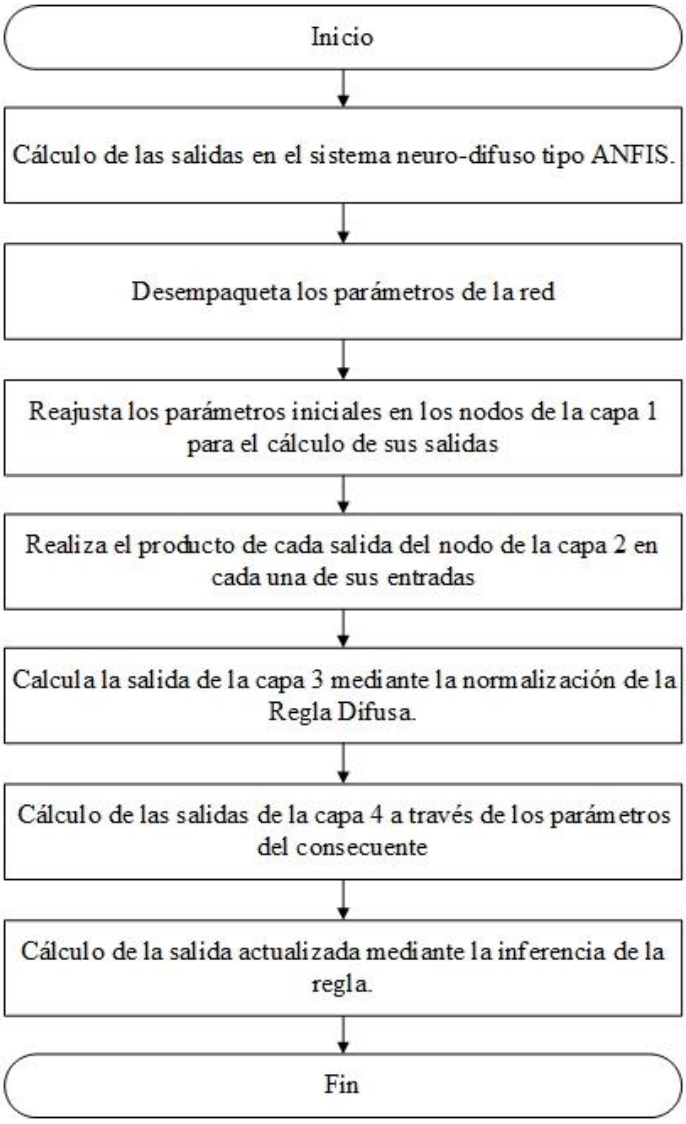
Figura 0.4: Subproceso del algoritmo ANFIS



Algoritmo de Aprendizaje Hibrido Online, Diego Rocha, María José Jara

A continuación, se representa en la Figura 4.5 la evaluación y cálculo de las variables de salida en función de los elementos del vector estado, mediante el reemplazo de los parámetros de premisa, la asignación de un valor de la función de membresía, el producto para cada entrada, la normalización de la regla, los parámetros del consecuente y la inferencia de las Reglas Difusas.

Figura 0.5: Subproceso del algoritmo ANFIS



Cálculo de las salidas de un sistema neuro-difuso tipo ANFIS, Diego Rocha, María José Jara

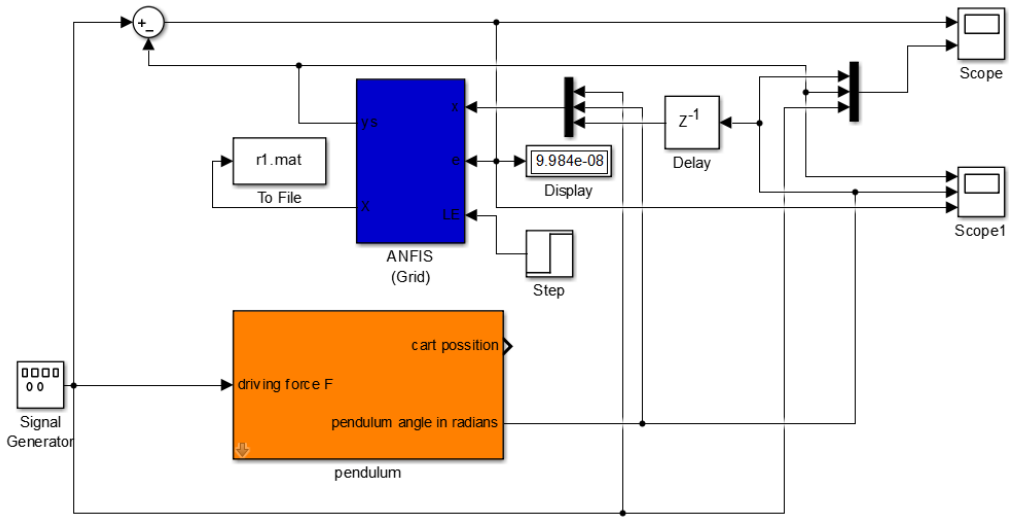
Estos diagramas de flujo están desarrollados en base al algoritmo del controlador neuro-difuso adaptativo ANFIS; el código de programación se encuentra detallado en el Anexo 2.

4.3. Identificación de la Planta Péndulo Invertido utilizando una Red ANFIS

Para la identificación de la planta se utilizó el método de identificación directo inverso en donde la red neuronal determina cual es la entrada que la produjo a través de la salida de la planta, una vez que la red neuronal haya aprendido el comportamiento

inverso de la planta es posible usarla para controlar la planta directamente, para más detalles acerca de los sistemas de identificación referirse al Anexo III. En la Figura 4.3 se muestra la configuración para la identificación del sistema del péndulo invertido utilizando la red ANFIS, entonces como ya se cuenta con el modelado matemático de la planta obtenido en el apartado 3.3.3 se utiliza una red neuronal ANFIS para que esta aprenda y simule el comportamiento del sistema del péndulo invertido.

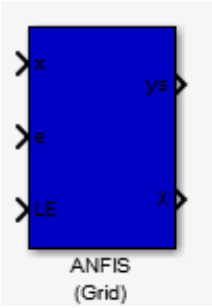
Figura 0.6: Diagrama de la Identificación de la planta



Sistema de Identificación ANFIS, Diego Rocha, María José Jara

En la Figura 4.4 se presenta el bloque S-function sobre el cual fue desarrollado el algoritmo de la red neuronal ANFIS y que se emplea para el entrenamiento de la planta péndulo digital.

Figura 0.7: Bloque ANFIS



Bloque S-function de la red ANFIS, Diego Rocha, María José Jara

La red neuronal ANFIS utilizada en el proceso de entrenamiento posee varios parámetros los cuales se describen a continuación:

El parámetro LE tiene dos estados discretos 0 y 1. Un estado de "1" en su entrada LE, realiza el aprendizaje y los valores que se obtienen se almacenan en una matriz, esto quiere decir que los parámetros de la capa 1 y la capa 4 se encuentran actualizados, cuando el estado es de "0" en su entrada LE, deja de aprender y entra en un estado de simulación, esto quiere decir, que los parámetros de la capa 1 y la capa 4 se encuentran fijos.

En el siguiente parámetro error *(e)*, se ingresa la señal del error de entrenamiento, la cual fue obtenida de la diferencia entre las señales de entrada hacia la planta y la señal de salida de la red neuronal ANFIS, cabe señalar que esta señal es un escalar y se lo visualiza a través del bloque pantalla (*display*).

Los parámetros x del controlador ANFIS son las entradas actuales del sistema ANFIS, entre ellas se tiene la entrada de la planta, la salida de la planta y una señal retrasada. Cuando se suministra a la red más de una entrada se debe utilizar un multiplexor formando así un vector de retardos.

El parámetro y_s es un escalar que representa la salida de la red neuronal ANFIS.

El parámetro X proporciona los estados de la red ANFIS durante el entrenamiento, en formato vector, con el fin de ser guardado y almacenado para utilizarlo en el controlador como datos iniciales evitando que en cada entrenamiento de la red ANFIS se inicie desde cero. Cada vez que se inicia una sesión en Simulink, mediante el almacenamiento de los valores en una matriz, estos datos se los incluye en los

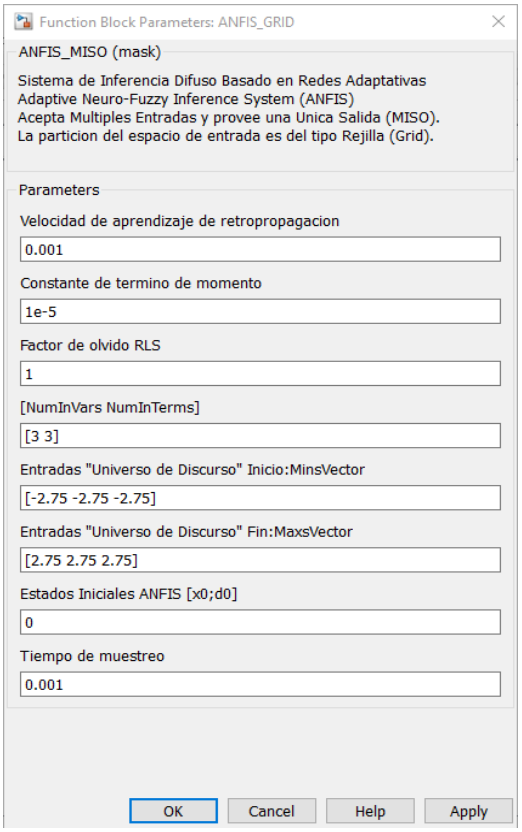
parámetros del bloque "*To file*" donde se selecciona el nombre del archivo *.mat* y se almacena la actualización del vector *X*.

El proceso de identificación también está compuesto por un generador de señales, un sumador, un osciloscopio que indica las formas de onda de las señales y la planta del péndulo invertido en la cual consta su modelo matemático verificar el correcto funcionamiento.

4.4. Configuración de los Parámetros del Bloque ANFIS Para Identificación.

Para realizar el aprendizaje y el control a través de una red neuro-difusa tipo ANFIS se crea un bloque de funciones, que permite variar los parámetros de cada instancia del bloque, proporcionando una sencilla interfaz. En la Figura 4.8 se representa el bloque del sistema ANFIS con sus respectivos parámetros de configuración.

Figura 0.8: Bloque de los parámetros ANFIS



Parámetros para el aprendizaje neuro difuso, Diego Rocha, María José Jara

A continuación, se presenta una breve descripción de los parámetros involucrados en el bloque ANFIS.

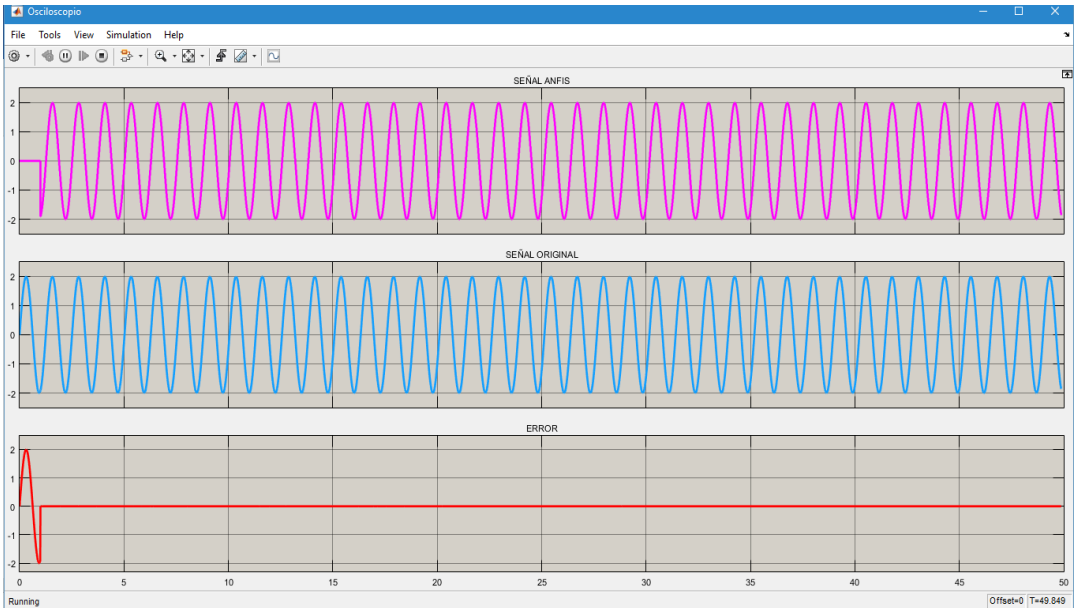
Tabla 0.1. Parámetros de aprendizaje neuro-difuso

Símbolo	Parámetro
Ita (η)	Tasa de aprendizaje
Alpha (α)	El término de momento
Lambda (λ)	El factor de olvido
NumInVars	El número de las variables de entrada en Simulink es el número de las señales de entrada.
NumInTerms	Selecciona el número de subconjunto difuso que divide el dominio de cada entrada.
Entradas "Universo de Discurso" Inicio: MinsVector	Determina el universo de discurso normal de las entradas.
Entradas "Universo de Discurso" Fin: MaxsVector	Este vector contiene el límite mínimo superior para cada entrada, y los valores mínimos junto con el vector de parámetros anterior asumen que el universo de discurso para cada entrada se conoce con anticipación.
Estado Inicial ANFIS [x0; d0]	Inicializa el sistema ANFIS a un estado que no posee ningún conocimiento de la tarea principal para la que es usado y se encuentra listo para el entrenamiento inicial.
Tiempo de Muestreo	Ajusta los parámetros del tiempo de muestreo del bloque.

Descripción de los parámetros del sistema de aprendizaje neuro-difuso, Diego Rocha, María José Jara

Las señales generadas se representan en la Figura 4.9 donde se encuentra la señal de identificación ANFIS que en términos de reconocimiento será. La salida deseada y la señal de origen de la planta, las cuales se encuentran en el rango de 2V y -2V aproximadamente dentro del rango de la señal del control que se indicó en el apartado 3.4. En la Figura 4.9 también se muestra la señal del error cuyo valor puede ser visualizado en un bloque y se encuentra alrededor de 9.94e-8, siendo el valor mínimo del error una vez que la planta ha realizado el proceso de aprendizaje.

Figura 0.9: Representación de las señales

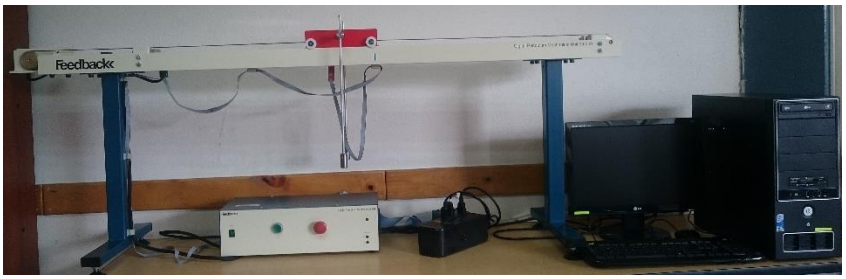


Representacion de las salidas del sistema ANFIS, Diego Rocha, María José Jara

4.5. Controlador Neuro-Difuso Tipo ANFIS

Culminado el proceso de identificación de la planta se realiza la implementación sobre la planta física del péndulo digital como se indica en la Figura 4.10, en la que se observa el péndulo digital conectado a la interfaz que permite comunicar la planta con el computador para los diferentes procesos de control.

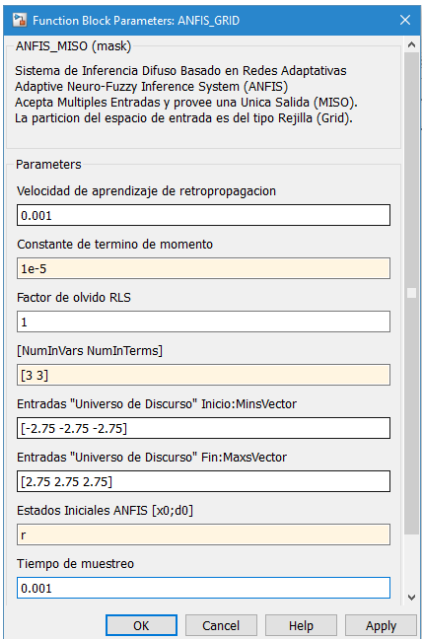
Figura 0.10: Fotografía del Péndulo Invertido del laboratorio de Teoría de Control



Planta física péndulo digital, Diego Rocha, María José Jara

Se procede a ubicar la matriz con los datos obtenidos en el proceso de entrenamiento (*r*) en el campo Estados iniciales ANFIS [x0;d0] como se indica en la Figura 4.12, para implementar el controlador neuro-difuso, se configura LE=0, es decir, la red ANFIS ya no se encuentra en modo aprendizaje ya que sus parámetros de premisa de la capa 1 y sus parámetros del consecuente de la capa 4 están fijos, y entra en un estado de prueba, después de haber aprendido el comportamiento de las señales de salida y entrada, para ser utilizados en el sistema de control y de esa manera obviar el proceso de identificación de la planta estableciendo los parámetros de control ya obtenidos en el proceso de identificación detallado en el apartado 4.3.

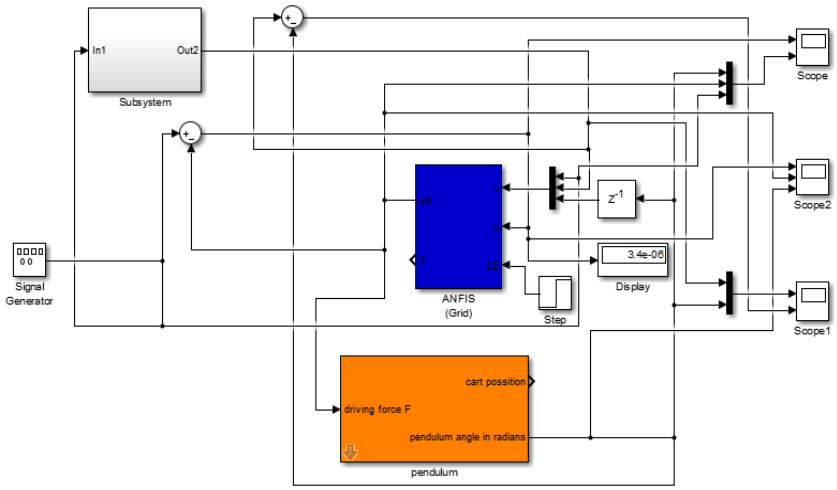
Figura 0.11: Bloque de parámetros ANFIS



Configuración de los parámetros del controlador ANFIS, Diego Rocha, María José Jara

En la Figura 4.12 se observa la configuración del bloque de control ANFIS como controlador hacia la planta del péndulo invertido, así como también los bloques de osciloscopio para la visualización de la salida real, la salida deseada y el error.

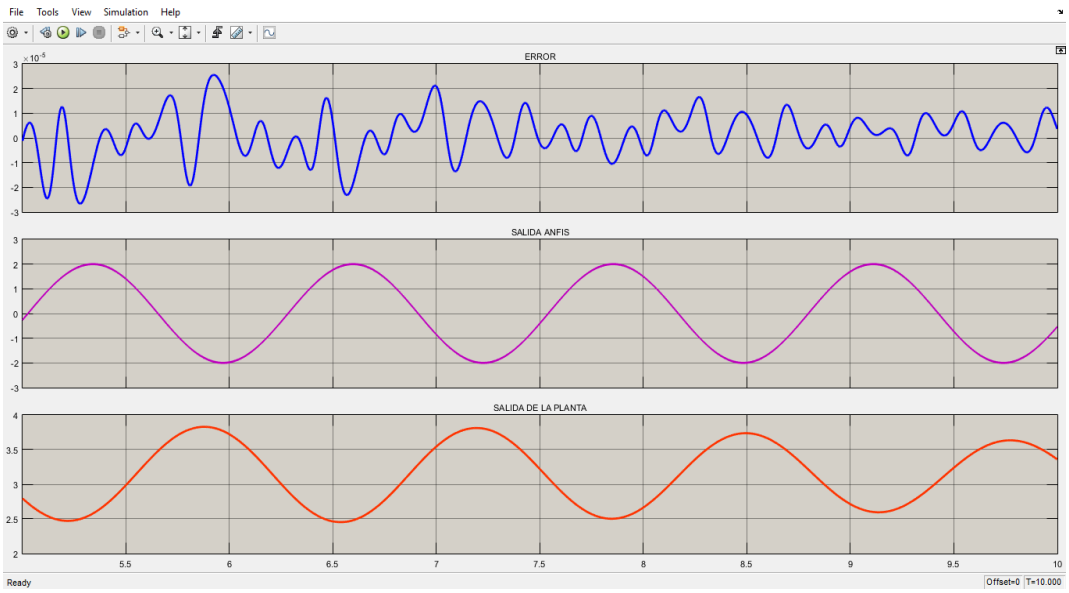
Figura 0.12: Diagrama de control



Controlador neuro-difuso, Diego Rocha, María José Jara

Realizado el controlador se procede a verificar que las salidas tanto del controlador como la salida de la planta son las deseadas y que el error entre las salidas es mínimo cercano a cero es decir aproximadamente 3.4e-6, así se puede realizar diversas pruebas exponiendo a una señal de entrada hacia la planta y visualizando el comportamiento de la misma. En la Figura 4.13 se realiza la representación gráfica de la salida del controlador ANFIS.

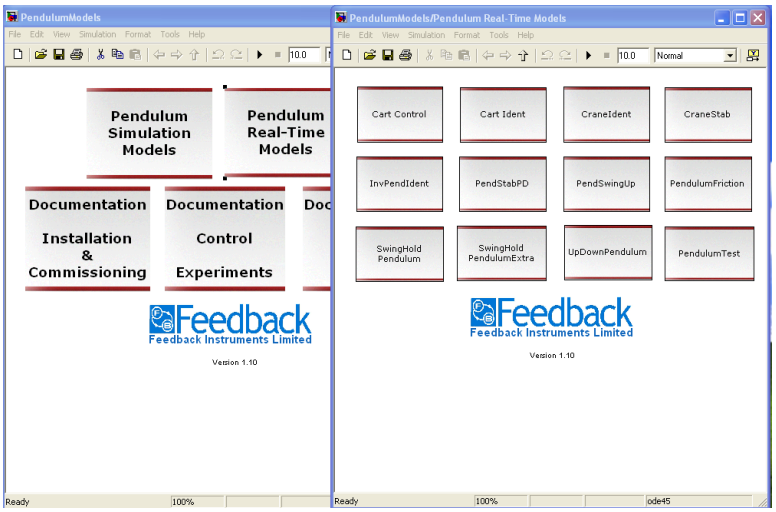
Figura 0.13: Representación de las señales de salida



Representación del controlador ANFIS, Diego Rocha, María José Jara

Para el proceso de comprobación del controlador ANFIS sobre la planta física es necesario usar los ejemplos suministrados por *Pendulum Simulink Models Feedback* el cual tiene una interfaz gráfica a través de Matlab que permite la comunicación entre la planta y el software, para el caso de este trabajo se usa el ejemplo llamado *SwingHold Pendulum*, ubicado en la carpeta de *Real Time Models*, tal y como se muestra en la Figura 4.14.

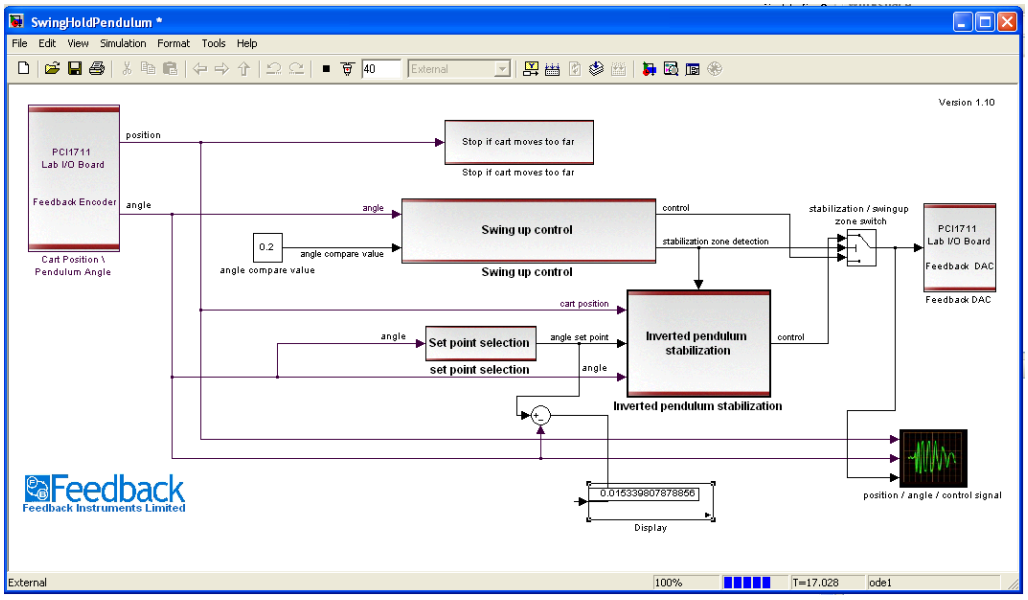
Figura 0.14: Ejemplos propios de Feedback



Ventana Principal de Pendulum Simulink Models Feedback, (FeedBack Insruments Ltd., 2006)

En la Figura 4.15 se observa el ejemplo *SwingHold Pendulum* que cuenta con un sistema de control PID, localizado en el bloque etiquetado como *Inverted pendulum stabilization.*

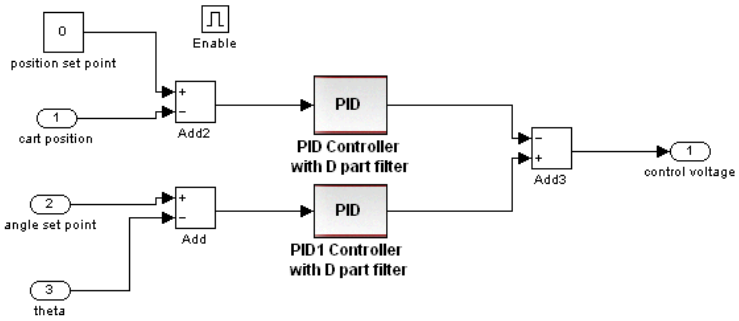
Figura 0.15: Ejemplo del Feedback del péndulo invertido



Ejemplo SwingHold Pendulum, (FeedBack Insruments Ltd., 2006)

En la Figura 4.16 se muestra la estructura interna del subsistema *Inverted Pendulum Stabilization* donde se encuentra localizado el controlador PID conformado por dos bloques que generan la señal de control total.

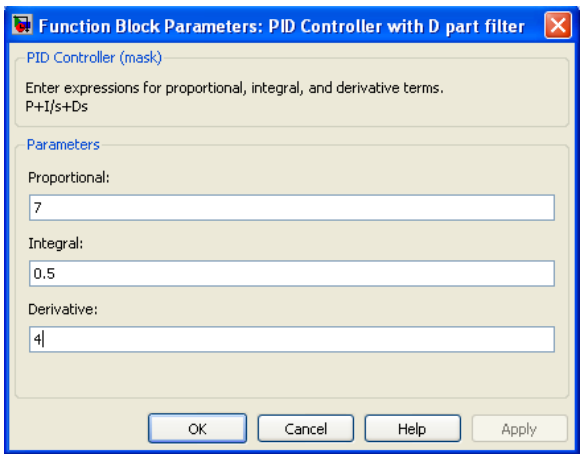
Figura 0.16: Control PID



Control PID dentro del bloque Inverted pendulum stabilization, (FeedBack Insruments Ltd., 2006)

Los controladores PID de la Figura 4.16 se encuentran sintonizados con los valores de las constantes integral, proporcional y derivativa que se muestra en la Figura 4.17; estas constantes pueden ser modificadas según la necesidad del usuario o según el método de sintonización que se desee utilizar como por ejemplo el método de Ziegler-Nichols.

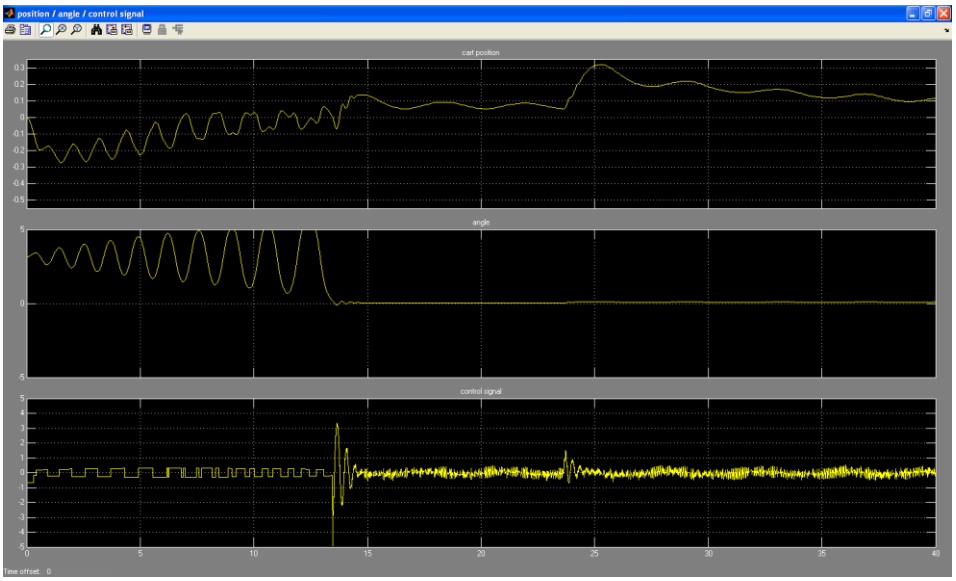
Figura 0.17: Bloque de los parámetros del PID



Valores de las variables del controlador PID, (FeedBack Insruments Ltd., 2006)

El controlador PID tiene la función de controlar el ángulo del péndulo, mientras que la posición del carro se regula mediante un control ON/OFF utilizando en el proceso los finales de carrera localizados en los extremos del riel. Las señales obtenidas con el controlador PID se muestra en la Figura 4.18 en la cual se representa tres señales que son; la posición del carro móvil, la posición del ángulo θ en su régimen estacionario y transitorio, y la señal del error que tiene un valor de 0.015336.

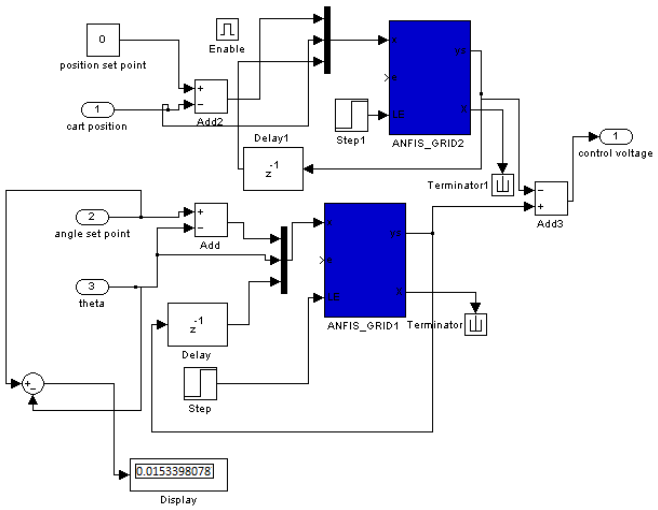
Figura 0.18: Representación de las señales de salida del PID



Señales del controlador PID, (FeedBack Insruments Ltd., 2006)

Entonces tomando el ejemplo de la Figura 4.16 se reemplazan los controladores PID que regulan el ángulo θ y la posición X por los controladores ANFIS como se indica en la Figura 4.19, de manera que sea posible verificar el comportamiento del controlador.

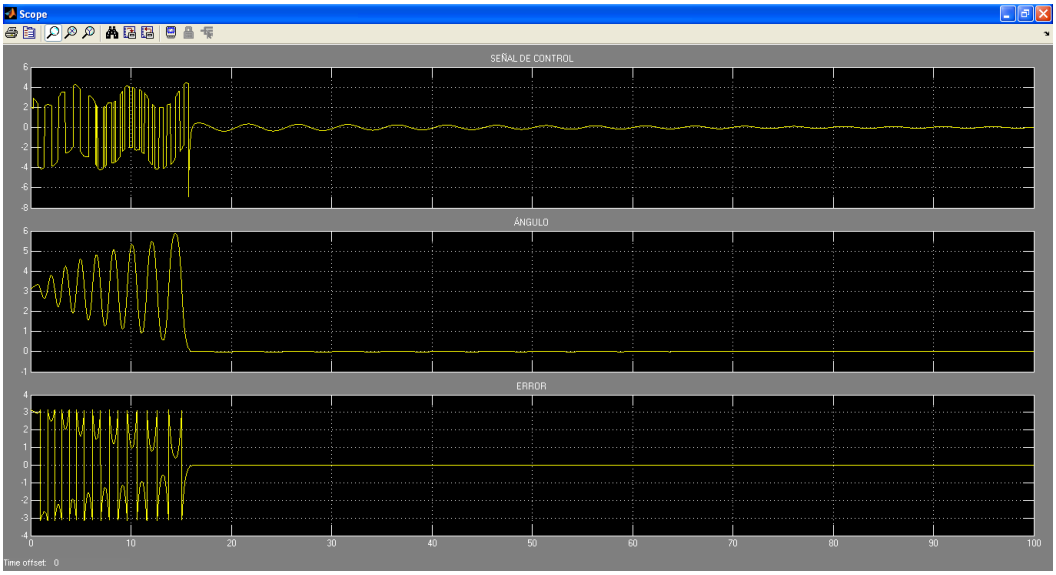
Figura 0.19: Diagrama del controlador Neuro-Difuso



Sistema de control ANFIS aplicado a la planta física del péndulo invertido, Diego Rocha, María José Jara

Las señales obtenidas se representan en la Figura 4.20, donde se encuentran la señal de control, la señal del ángulo y la señal del error. Como se puede observar en la Figura 4.19 el valor del error una vez estabilizado el sistema es de 0.01533, es decir se aproxima a cero.

Figura 0.20: Representación gráfica de las señales del Controlador ANFIS



Representación de las señales en el controlador en tiempo real, Diego Rocha, María José Jara

CAPITULO V

ANALISIS Y RESULTADOS

5.1. Introducción

En este capítulo se realiza un análisis del desempeño del péndulo invertido controlado con el PID dado por el fabricante y el controlador Neuro-Difuso tipo ANFIS desarrollado en el capítulo anterior para realizar un test estadístico utilizando un de rendimiento (IAE) y así se podrá indicar cuál de los dos controladores tiene un mejor desempeño en su entorno de trabajo.

5.2. Análisis Comparativo entre el Controlador PID y el Controlador ANFIS

Para el análisis se ha realizado una tabla basada en el Test de Wilcoxon el cual trabaja sobre rangos de orden y con valores no paramétricos para muestras pequeñas mayores a 25. En la Figura 5.1 se realiza una representación del análisis estadístico utilizando el método ya mencionado, en el cual se toma en consideración el coeficiente del error absoluto denominado IAE, en base a este coeficiente se plantean tres hipótesis que son:

a. Si el coeficiente IAE_{PID} es igual al IAE_{ANFIS} no existe ninguna mejora entre controladores.
b. Si el coeficiente IAE_{PID} es menor al IAE_{ANFIS} estaría demostrado que el controlador neuro-difuso muestra mejores prestaciones.
c. Si el coeficiente IAE_{PID} es mayor al IAE_{ANFIS} no se lograría mejorar el tipo de control que tiene la planta del péndulo invertido.

5.2.1. Test de Wilcoxon

Es necesario tabular los datos de los coeficientes IAE_{PID} e IAE_{ANFIS} para la elaboración del test como se muestra en la Figura 5.1.

Figura 0.1: Tabla de Wilcoxon

TEST DE WILCOXON PARA DATOS PAREADOS, EN EL PÉNDULO INVERTIDO									
HIPOTESIS	Principal (Ho): PID=ANFIS						Media PID = Mediana ANFIS		
	Alternativa (Ha): PID>ANFIS						Media PID > Mediana ANFIS		
No	P. PID	P. ANFIS	Diferencias	Rango Asignados	T+	T-	Ligaduras	Diferencias Ordenadas	Rangos de Orden
1	13,411	11,401	2,010	11	11			1,554	1
2	13,401	11,339	2,062	22	22			1,920	2
3	13,385	11,300	2,085	27	27			1,926	3
4	13,408	11,400	2,008	10	10			1,986	4
5	14,275	12,255	2,020	15	15			1,994	5
6	14,439	12,422	2,017	13	13			2,000	6
7	14,355	12,320	2,035	18	18			2,003	7
8	14,275	12,211	2,064	23	23			2,204	8
9	14,314	12,214	2,100	31	31			2,007	9
10	14,253	12,253	2,000	6	6			2,008	10
11	14,481	12,375	2,106	32	32			2,010	11
12	14,486	12,435	2,051	21	21			2,014	12
13	13,405	11,328	2,077	26	26			2,017	13
14	13,412	11,338	2,074	25	25			2,019	14
15	14,193	12,186	2,007	9	9			2,020	15
16	14,475	12,456	2,019	14	14			2,022	16
17	14,514	12,369	2,145	35	35			2,030	17
18	13,442	11,238	2,204	8	8			2,035	18
19	14,212	12,198	2,014	12	12			2,041	19
20	14,372	12,386	1,986	4	4			2,045	20
21	13,438	11,397	2,041	19	19			2,051	21
22	14,256	12,253	2,003	7	7			2,062	22
23	14,265	12,345	1,920	2	2			2,064	23
24	13,423	11,401	2,022	16	16			2,065	24
25	14,626	12,596	2,030	17	17			2,074	25
26	13,549	11,405	2,144	34	34			2,077	26
27	13,392	11,297	2,095	29	29			2,085	27
28	14,281	12,216	2,065	24	24			2,091	28
29	14,427	12,328	2,099	30	30			2,095	29
30	13,471	11,426	2,045	20	20			2,099	30
31	14,295	12,180	2,115	33	33			2,100	31
32	14,336	12,342	1,994	5	5			2,106	32
33	14,286	12,732	1,554	1	1			2,115	33
34	14,311	12,220	2,091	28	28			2,144	34
35	14,493	12,567	1,926	3	3			2,145	35

Test de Wilcoxon para PID vs. ANFIS, Diego Rocha, María José Jara

Con los datos obtenidos del test de Wilcoxon se determinaron los resultados mostrados en la Figura 5.2, en la cual se indica la suma total de las diferencias positivas y negativas, la aproximación por la normal (Z), el nivel de confianza al 95% teniendo una constante de 1,96. Mediante estos valores se interpreta la decisión de la aproximación con respecto al nivel de confianza.

Figura 0.2: Resultados

Total T+			630
Total T-			0
T=min (T+, T-)			630
Z			5,159
Za			1,96
Decisión	Z≤Za	Se acepta la Ho	
	Z>Za	Se acepta la Ha	Ok

Resultado del test de Wilcoxon, Diego Rocha, María José Jara

Teniendo en cuenta los resultados anteriores se acepta la hipótesis alternativa ya que el IAE_{PID} es mayor que el IAE_{ANFIS} por lo tanto se observa que el controlador neuro-difuso tiene un mejor tiempo estabilización y el error disminuye considerablemente.

CONCLUSIONES

Una vez concluido este trabajo, se logró implementar un algoritmo de programación mediante el software Matlab cumpliendo con todas las características funcionales de una red ANFIS, comprobando que para el entrenamiento de la red ANFIS se debieron combinar dos algoritmos de aprendizaje uno el Algoritmo de Retro-Propagación del Error que permitió actualizar los parámetros de premisa que se ubican en la Capa 1 y el otro el algoritmo de Mínimos Cuadrados Recursivos utilizado para modificar los parámetros del consecuente en la Capa 4, obteniendo así un algoritmo de aprendizaje híbrido de gran potencia que combina todas las ventajas de las redes adaptativas y los sistemas de inferencia difuso.

El proceso de aprendizaje y por tanto la velocidad con las que se ejecutan las acciones de control de una red ANFIS son más rápidas debido a la combinación de dos algoritmos de aprendizaje que se basan en cuestiones matemáticas y estadísticas que toman en consideración todas las características dinámicas propias de la planta. Gracias a ello no fue necesaria la linealización de la planta, más bien se utilizó el modelo no lineal completo para realizar el proceso de aprendizaje que toma en cuenta todos los parámetros de funcionamiento de la planta obteniendo mejores resultados que el controlador PID, el cual utiliza técnicas básicas de control y muchas veces la configuración de sus parámetros se basan en métodos de sintonización empíricos, por dicha razón es necesario realizar procesos de linealización que terminan restándoles propiedades de la planta.

Una vez obtenido el modelo matemático del péndulo digital fue posible realizar un bloque con las características dinámicas de la planta en el entorno de simulación gráfico Simulink, el cual permitió realizar las pruebas de simulación de la red ANFIS tanto como un identificador del sistema, así como un controlador antes de aplicarlo a la planta física, con lo cual se pudo evitar que el sistema en su conjunto sufriera algún tipo de daño debido a la inestabilidad propia de su construcción.

Gracias a las funciones propias del software Matlab fue posible realizar la programación del algoritmo de la red ANFIS el cual pudo ser enlazado con el entorno de programación grafico Simulink a través de un bloque S-function que permitió realizar varias pruebas de simulación con diferentes configuraciones de control basadas en las redes neuronales adaptativas y variando los parámetros de la red ANFIS.

Utilizando el sistema de control directo-inverso se relacionaron varias simulaciones hasta obtener los mejores resultados en cuanto a la medida del error que se encuentra en un valor de 9.98e-08.

El controlador neuro-difuso tipo ANFIS desarrollado e implementado en el presente trabajo de titulación muestra un desempeño superior frente al controlador PID para el proceso de estabilización del péndulo invertido, esto fue evidenciado al realizar varias pruebas de funcionamiento en donde se muestrearon tiempos de estabilización del péndulo y a partir de estos datos fue posible la realización de una tabla estadística la cual se basó en el coeficiente del error absoluto IAE de ambos controladores y con estos datos se pudo realizar el test de Wilcoxon, obteniendo como resultado que el valor experimental corregido (Z) tiene un valor de 5,159 que comparado con el nivel de confianza (Za) estándar de 1,96 pone en evidencia que la hipótesis aceptada es la que plantea un rendimiento superior para el controlador neuro-difuso tipo ANFIS.

RECOMENDACIONES

Para implementar el controlador ANFIS en una planta diferente a la que se encuentra en el laboratorio de Teoría de Control de la Universidad Politécnica Salesiana Campus Sur es necesario referirse al manual del fabricante para verificar las especificaciones de la planta en cuanto a la variación de las señales de control y los rangos que deben ser modificados en la red ANFIS, los datos susceptibles a modificación pueden ser los valores máximos y mínimos del universo de discurso y podrían conservarse fijas las constantes de la velocidad de aprendizaje, el factor de olvido y el término de momento.

Se recomienda verificar el funcionamiento de la red ANFIS como controlador utilizando otras configuraciones adaptativas como las mostradas en el Anexo III debido a que este trabajo solo se utilizó la configuración del controlador directo-inverso.

Para realizar futuros trabajos empleando la red ANFIS como controlador se recomienda realizar un análisis de desempeño basado en el mismo test de Wilcoxon, pero con otros controladores que tengan características similares al neuro-difuso como puede ser un controlador difuso o un controlador neuronal que se encuentran incluidos en varias librerías propias del software Matlab.

REFERENCIAS

Buragohain, M. (Julio de 2008). *Indian Institute of Technology Guwahati.* Obtenido de http://www.iitg.ernet.in/engfac/chitra/notes/Thesis_Mrinal.pdf

Caicedo, & Lopez. (2009). *Una aproximación práctica a las Redes Neuronales Artificiales.* Colombia.

FeedBack Insruments Ltd. (01 de 12 de 2006). Digital Pendulum Control Experiments. *Manual: 33-936S*. Crowborough, Sussex, UK.

Flores López, R., & Fernandéz Fernández, J. M. (2008). *Google Books.* Obtenido de Las redes neuronales artificiales fundamentos teóricos y aplicaciones prácticas: https://books.google.com.ec/books?id=X0uLwi1Ap4QC&pg=PA75&dq=termino+momento&hl=es&sa=X&redir_esc=y#v=onepage&q=termino%20momento&f=false

Jang, J.-S. R. (06 de 08 de 2002). *Adaptive-Network-Based-Fuzzy-Inference-System.* Obtenido de IEEEXplore Digital Library: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=256541&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel1%2F21%2F6499%2F00256541

Jang, J.-S. R., Sun, C.-T., & Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing: A computational Approach to learning and Machine Intelligence.* New Jersey 07458: Prentice Hall, Inc. Upper Saddle River.

Nazmul, S., & Hojjat, A. (2013). *Synergies of Fuzzy Logic, Neural Networks and Evolutionary Computing.* Unitecd Kinddon: John Wiley & Sons. Ltd.

Ogata, K. (2003). *Ingenieria de Control Moderna.* Madrid: Pearson Educacion. S.A.

Picón Martorell, M. C. (2005). HERRAMIENTA COMPUTACIONAL PARA DESARROLLAR SISTEMAS DIFUSOS CON ENTRENAMIENTO NEURODIFUSO. *HERRAMIENTA COMPUTACIONAL PARA DESARROLLAR SISTEMAS DIFUSOS CON ENTRENAMIENTO NEURODIFUSO*. Barquisimeto: UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO".

Ponce Cruz, D. (2010). *Inteligencia Artificial con aplicación a la ingeniería.* Alfaomega Grupo Editor S.A.

ANEXOS

ANEXO 1

FUNCIONES S-FUNCTION[1]

Una S-function (System-function) es un lenguaje de descripción de un bloque de Simulink escrito en código de MATLAB, C, C ++ o Fortran. Las S-functions en C, C ++ y Fortran son compiladas como archivos MEX mediante la utilidad *mex* antes de ser utilizadas en el entorno de Simulink.

Las S-functions siguen una forma general y pueden alojar sistemas continuos, discretos e híbridos. Al seguir una serie de reglas simples, se puede implementar un algoritmo de programación y utilizar el bloque S-Function de Simulink para añadir este algoritmo a un modelo Simulink. Dicho en otras palabras, una S-function es una función compuesta de dos elementos, un bloque de Simulink definible por el usuario y un programa escrito en cualquiera de los lenguajes de programación antes mencionados, el cual es llamado o cargado en el bloque.

Las S-functions son utilizadas para una variedad de aplicaciones, incluyendo: creación de nuevos bloques de propósito general; adición de bloques que representan controladores de dispositivos de hardware; incorporación de código C existente en una simulación; descripción de un sistema como un conjunto de ecuaciones matemáticas; descripción de modelos dinámicos complicados (con parámetros variantes con el tiempo, sistemas no lineales, con diferentes periodos de muestreo, etc.) y para animaciones gráficas.

El uso más común de las S-functions es para crear bloques personalizados en Simulink. Cuando se utiliza una S-function para crear un bloque de propósito general, se puede utilizar muchas veces este bloque en un modelo, variando los parámetros con cada instancia del bloque.

Las S-functions utilizan una sintaxis de llamado especial conocida como API S-function, la cual permite a este tipo de funciones interactuar con el motor de Simulink. Esta interacción es muy similar a la interacción que tiene lugar entre el motor y los bloques incorporados en Simulink.

[1] Ing. Junior Rafael Figueroa Olmedo

CONCEPTOS PRELIMINARES

Para crear las S-functions, es necesario entender su funcionamiento. Tal conocimiento requiere una comprensión de cómo el motor Simulink simula un modelo, incluyendo las matemáticas de bloques. Esta sección comienza explicando las relaciones matemáticas entre las entradas, los estados y salidas de un bloque.

Matemáticas de bloques de Simulink

Un bloque de Simulink se compone de un conjunto de entradas, un conjunto de estados y un conjunto de salidas (ver Figura A1), donde las salidas están en función del tiempo de simulación, las entradas y los estados.

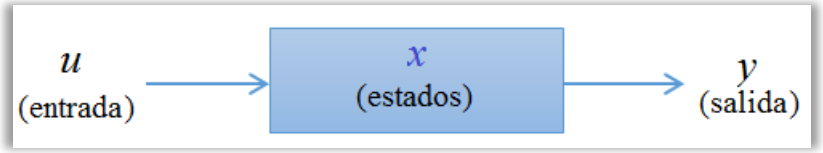


Figura A1 Representación general de un bloque en Simulink.

Las siguientes ecuaciones expresan las relaciones matemáticas entre las entradas, las salidas, los estados y el tiempo de simulación.

$$y = f_0(t, x.u) \qquad \text{(Salidas)}$$

$$\dot{x} = f_d(t, x.u) \qquad \text{(Derivadas)}$$

$$x_{d_{k+1}} = f_u\left(t, x_c, x_{d_k}, u\right) \qquad \text{(Actualización)}$$

donde $x = [x_c; x_d]$

Etapas de Simulación

La ejecución de un modelo Simulink se desarrolla en etapas. Primero viene la fase de inicialización, en donde el motor de Simulink incorpora bloques de librería en el modelo, propaga anchos de señales, tipos de datos y tiempos de muestreo, evalúa los parámetros del bloque, determina el orden de ejecución del bloque y asigna memoria. El motor entonces entra en un bucle de simulación, en donde cada pasada a través del bucle se conoce como un paso de simulación. Durante cada paso de simulación, el motor ejecuta cada bloque en el modelo en el orden determinado durante la inicialización. Para cada bloque, el motor invoca funciones que calculan los estados del bloque, las derivadas y las salidas para el tiempo de muestreo actual.

La Figura A2 ilustra las etapas de una simulación. El bucle de integración se lleva a cabo sólo si el modelo contiene estados continuos. El motor ejecuta este bucle hasta que el solucionador alcanza la precisión deseada para los cálculos de estado. El bucle de simulación entonces continúa hasta que la simulación se ha completado.

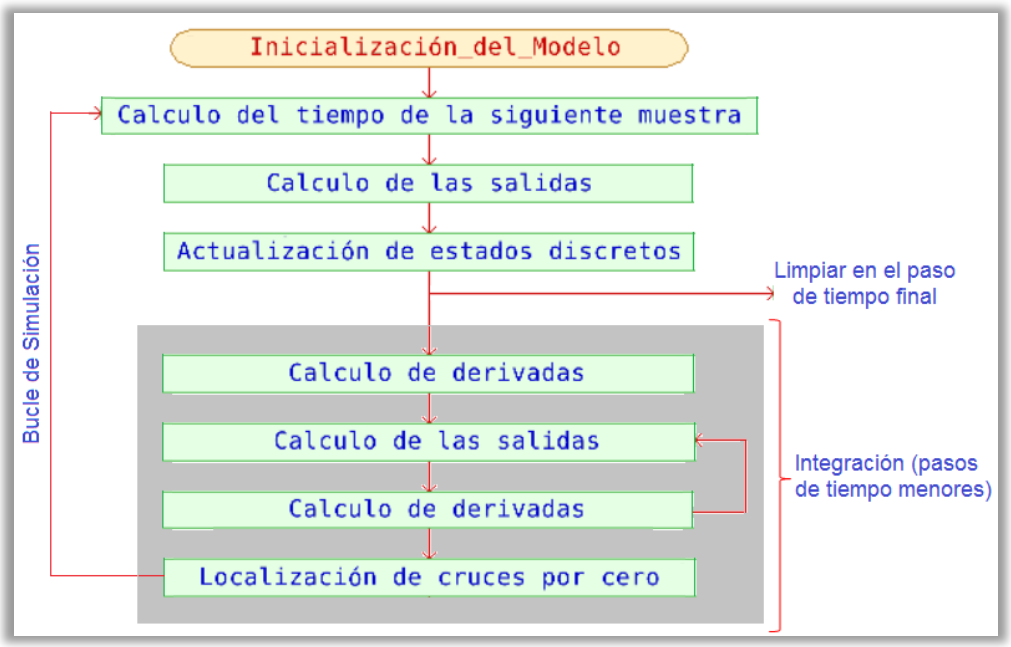


Figura A2 Fases de simulación de un modelo Simulink.

IMPLEMENTACIONES DISPONIBLES PARA S-FUNCTIONS

Se pueden implementar las funciones S-functions utilizando alguno de los siguientes caminos:

a. **Level-1 MATLAB S-function:** proporciona una sencilla interfaz de MATLAB para interactuar con una pequeña porción del API S-function. Contiene el protocolo con el cual Simulink puede acceder a información escrita en código de Matlab.
b. **Level-2 MATLAB S-function:** proporciona acceso a un conjunto más amplio del API S-function y soporta generación de código desarrollado por el usuario. En la mayoría de los casos, se utiliza este tipo de funciones cuando se quiere implementar la S-function en el entorno de MATLAB.
c. **C MEX S-function**: proporciona la mayor flexibilidad de programación en este tipo de funciones. Permite implementar el algoritmo como una *C MEX S-function* o escribir un contenedor *S-function* para llamar a un código existente

en C, C++ o Fortran. Para escribir este tipo de funciones se requiere un amplio conocimiento del API S-function.

d. **S-Function Builder:** es una interfaz gráfica de usuario para la programación de un subconjunto de funcionalidades de la S-function. Si el usuario es nuevo en la escritura de funciones C MEX S-functions, puede utilizar el S-Function Builder para generar nuevas S-functions o incorporar código existente en C o C++ sin interactuar con el API S-function.

FUNCIONAMIENTO Y CONFIGURACIÓN DE UNA S-FUNCTION

En este trabajo se va a implementar un bloque de función mediante el método Level-1 MATLAB S-function, por dicha razón solo se explicará el funcionamiento y creación para este tipo de S-function. Para este método de implementación la función de MATLAB debe tener la siguiente forma

$$[sys, x0, str, ts] = f(t, x, u, flag, p_1, p_2, \dots)$$

donde f es el nombre de la S-function. Durante la simulación de un modelo, el motor de Simulink invoca repetidamente a f, usando el argumento $flag$ para indicar la tarea (o tareas) a realizar para una invocación en particular. La S-function realiza la tarea y retorna los resultados en un vector de salida denotado por $[sys, x0, str, ts]$.

Una plantilla para un Level-1 MATLAB S-function etiquetada como *sfuntmpl.m*, se encuentra localizada en *matlabroot/toolbox/simulink/blocks*. La plantilla consiste de una función de nivel superior y un conjunto de funciones locales, conocidas como métodos de devolución de llamada (*S-function callback methods*), donde cada una es identificada por un valor en particular del $flag$. La función de nivel superior invoca a la función local indicada por el $flag$. Las funciones locales realizan las tareas reales requeridas por la S-function durante el proceso de simulación. El motor de Simulink pasa los siguientes argumentos a un Level-1 MATLAB S-function:

Tabla A1 Argumentos de entrada para una S-function.

$\boldsymbol{t}$	Tiempo de simulación actual
$\boldsymbol{x}$	Vector columna correspondiente a las variables de estados
$\boldsymbol{u}$	Vector columna correspondiente a las variables de entradas
$\boldsymbol{flag}$	Valor entero que indica la tarea a ser realizada por la S-function

$p_1, p_2, \ldots$	Parámetros adicionales

Existen seis tipos de tareas que realiza la S-function y cada una de las cuales se designa por un número entero asignado a la variable $flag$. En la Tabla A2 se describen los valores que puede asumir la variable $flag$.

Tabla A2 Valores y funciones de la variable $flag$

Valor del $flag$	Descripción
0	**Inicialización y Configuración** Define las características básicas del bloque S-Function, incluyendo los tiempos de muestreo, especificación y/o cálculo de las condiciones iniciales de las variables de estados continuos y discretos, configuración de los tamaños de los vectores de entrada y salida.
1	**Cálculo de Derivadas** Calcula las derivadas de las variables de estados continuos. También realiza cómputos que involucran a los vectores de entrada.
2	**Actualización de Estados Discretos** Actualiza los estados discretos, tiempos de muestreo y los principales requisitos del paso de tiempo (*time step*).
3	**Cálculo de las Salidas** Evalúa y calcula las variables de salida en función de los elementos del vector de estado (y en algunos casos, también los elementos de vector de entrada).
4	**Cálculo del Tiempo de la siguiente Petición** Calcula el tiempo de la siguiente petición (*hit*) en tiempo absoluto. Esta rutina se usa sólo cuando se especifica un tiempo

	de muestreo para una variable discreta en el método de configuración.
9	**Finalización** Rutinas y cálculos adicionales al final de la ejecución de la simulación.

Como ya se ha mencionado un tipo de implementación Level-1 MATLAB S-function retorna un vector de salida que contiene los siguientes elementos:

sys: vector principal de los resultados solicitados por Simulink. Dependiendo del valor de $flag$ enviado por Simulink, este vector contendrá diferente información, según se muestra en la Tabla A3.

El siguiente conjunto de tres argumentos de salida son utilizados por Simulink sólo cuando $flag = 0$, de lo contrario, se ignoran:

$x0$: vector columna con los valores de los estados iniciales (se coloca un vector vacío si no hay estados en el sistema).

str: originalmente destinado para uso futuro. Debe establecerse como una matriz vacía, [ ].

ts: matriz de dos columnas que contiene los tiempos de muestreo y los tiempos *offsets* del bloque.

Por ejemplo, si desea que la S-function se ejecute en cada paso de tiempo (tiempo de muestreo continuo), se debe establecer ts a [0 0]. Si se requiere que la S-function se ejecute a la misma velocidad como la del bloque al que está conectado (tiempo de muestreo heredado), se establece ts a [-1 0]. Si se desea que esta se ejecute cada 0.25 segundos (tiempo de muestreo discreto) empezando en 0.1 segundos después del tiempo de inicio de la simulación, se ajusta ts a [0.25 0.1].

Tabla A3 Componentes del vector sys dependiendo del valor del $\boldsymbol{flag}$.

Si $\boldsymbol{flag = 0}$	$\boldsymbol{sys} = [\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{e}, \boldsymbol{f}, \boldsymbol{g}]$ Donde $\boldsymbol{a} =$ número de estados de tiempo continuo.

	$\boldsymbol{b}$ = número de estados de tiempo discreto. $\boldsymbol{c}$ = número de salidas (Nota: esto no es necesariamente el número de estados). $\boldsymbol{d}$ = número de entradas. $\boldsymbol{e} = \mathbf{0}$ (es requerido que sea 0, no se utiliza actualmente) . $\boldsymbol{f}$ = 0 (no) o 1 (sí) para alimentación algebraica directa a través de la entrada a la salida. Esto sólo es relevante si durante $\boldsymbol{flag} = \mathbf{3}$, las variables de salida dependen algebraicamente de las variables de entrada). $\boldsymbol{g}$ = número de tiempos de muestreo. Para procesos continuos, colocar un valor de 1.
Si $\boldsymbol{flag} = \mathbf{1}$	sys = vector columna de las derivadas de las variables de estado.
Si $\boldsymbol{flag} = \mathbf{3}$	sys = vector columna de las variables de salida.
Si $\boldsymbol{flag} = \mathbf{2, 4, 9}$	ya que estos indicadores no se utilizan en este trabajo, sólo pueden enviar un vector nulo: sys = [ ].

CREACIÓN DE UNA S-FUNCTION EN SIMULINK

Para incorporar una S-function del tipo Level-1 MATLAB S-function en un modelo Simulink, arrastre un bloque *S-Function* desde la biblioteca de bloques etiquetada como *User-Defined Functions* en el modelo. A continuación dar doble clic con el botón izquierdo del mouse y en la ventana que se despliega coloque el nombre de la función (que debe tener el formato solicitado para una S-function) en el campo *S-function name* del cuadro de diálogo del bloque, tal y como se ilustra en la Figura A3. En este ejemplo en particular la función desarrollada en MATLAB fue guardada con el nombre de *anfisim_grid.m* y corresponde al algoritmo de programación del sistema ANFIS utilizado en los procesos de control de este trabajo (para más detalle de la función ver el Anexo A).

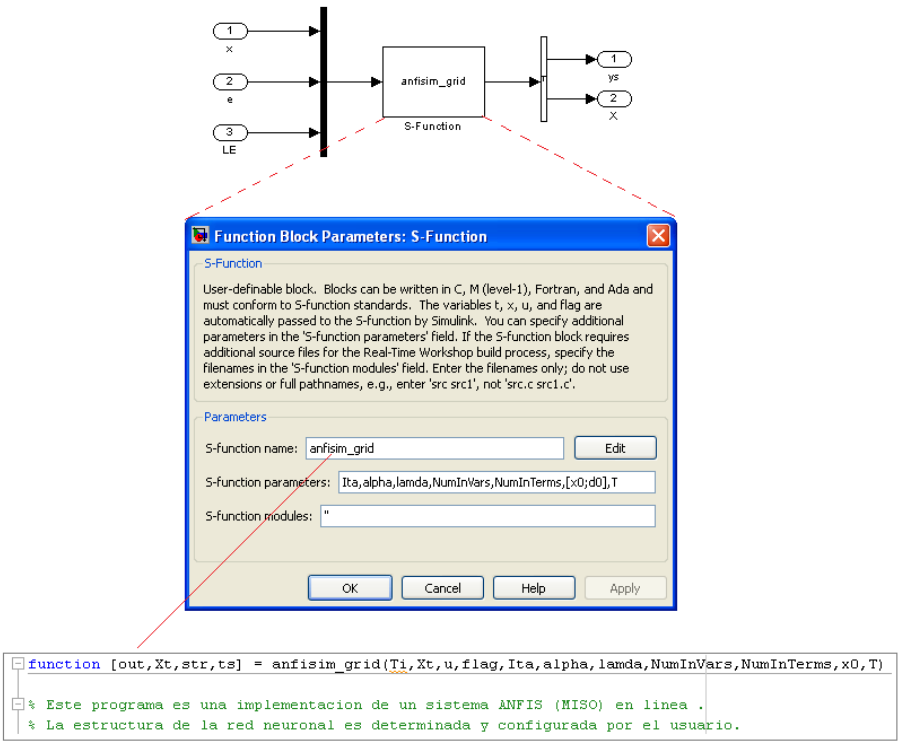


Figura A3 Creación de una S-function en Simulink.

Cuando se invoca al bloque S-Function, el motor de Simulink siempre pasa los parámetros del bloque t, x, u y $flag$, como argumentos de la función *anfisim_grid.m*. El motor también puede pasar a la función *anfisim_grid.m* parámetros adicionales del bloque especificados por el usuario. El usuario puede especificar estos parámetros en el campo *S-function parameters* del cuadro de diálogo *S-Function Block Parameters*. Para utilizar este campo, se debe conocer de antemano los parámetros que la S-function requiere. Los parámetros se introducen uno tras otro separados por una coma y en el orden requerido por la función desarrollada en MATLAB. Los valores de los parámetros pueden ser constantes, nombres de variables definidas en MATLAB o el área de trabajo (*workspace*), o expresiones de MATLAB.

En el ejemplo mostrado en la Figura A3 los parámetros fueron definidos mediante los nombres de las siguientes variables: Ita, alpha, lamda, NumInVars, NumInTerms, [x0,d0], T. Cada una de estas variables tiene un significado y propósito dentro del funcionamiento del algoritmo ANFIS. Una vez que se definen los parámetros adicionales, el motor de Simulink pasa estos parámetros a la función *anfisim_grid.m* y los coloca en la lista de argumentos de esta función en el orden en el cual estos fueron definidos en el cuadro de diálogo de bloque S-Function. Se puede utilizar esta

capacidad para permitir que la misma S-function implemente varias opciones de procesamiento.

En este punto cabe mencionar que también es factible utilizar la opción de enmascaramiento para crear cuadros de diálogo e iconos personalizados para el bloque S-Function de Simulink. El cuadro de diálogo enmascarado permite que sea más fácil especificar los parámetros adicionales para la S-function. En primer lugar se va a crear un subsistema seleccionando todos los elementos relacionados con el bloque S-Function, luego se da clic derecho con el mouse y se elige la opción *Create Subsystem*, tal como se muestra en la Figura A4.

Tras realizar los pasos indicados, se genera un bloque como el mostrado en la Figura A5. Para asignarle el color al interior del bloque se debe dar clic derecho sobre el mismo y seguir la ruta *Background Color >Light Blue*.

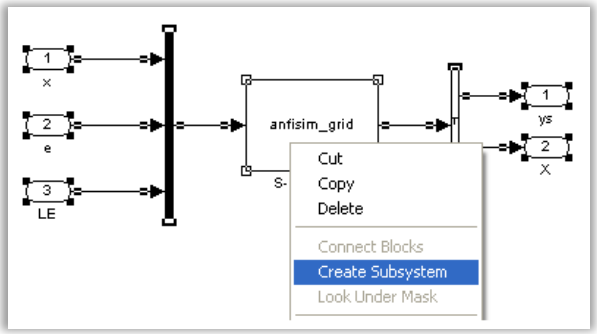


Figura A4 Creación de un Subsistema.

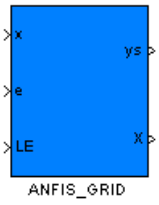


Figura A5 Presentación final de bloque del Subsistema.

Una vez creado el subsistema ANFIS_GRID es posible personalizarlo. El editor de máscaras se abre al hacer clic derecho en el menú contextual del bloque y seleccionando *Mask Subsystem*, tras realizar esta acción se despliega una ventana como la mostrada en la Figura A6. El Editor de Máscara (Mask Editor) contiene un conjunto de pestañas de paneles, cada una de las cuales permite definir una característica de la máscara: el panel *Icon & Ports* permite definir el icono del bloque;

el panel *Parameters* permite definir e introducir los parámetros del subsistema, que en nuestro caso corresponden a los parámetros adicionales del bloque de la S-Function; el panel *Initialization* permite especificar los comandos de inicialización y el panel *Documentation* permite definir el tipo, la descripción y la ayuda de la máscara.

Los parámetros del bloque se definen en el panel mostrado en la Figura A7. En las celdas de la columna etiquetada como *Variable* se escriben los nombres de los parámetros que deben coincidir con los nombres de los parámetros adicionales definidos en el bloque de la S-Function. En las celdas de la columna etiquetada como *Prompt* se puede redactar una pequeña descripción del significado de las variables creadas. Para ir añadiendo variables se debe dar clic en el botón *Add*.

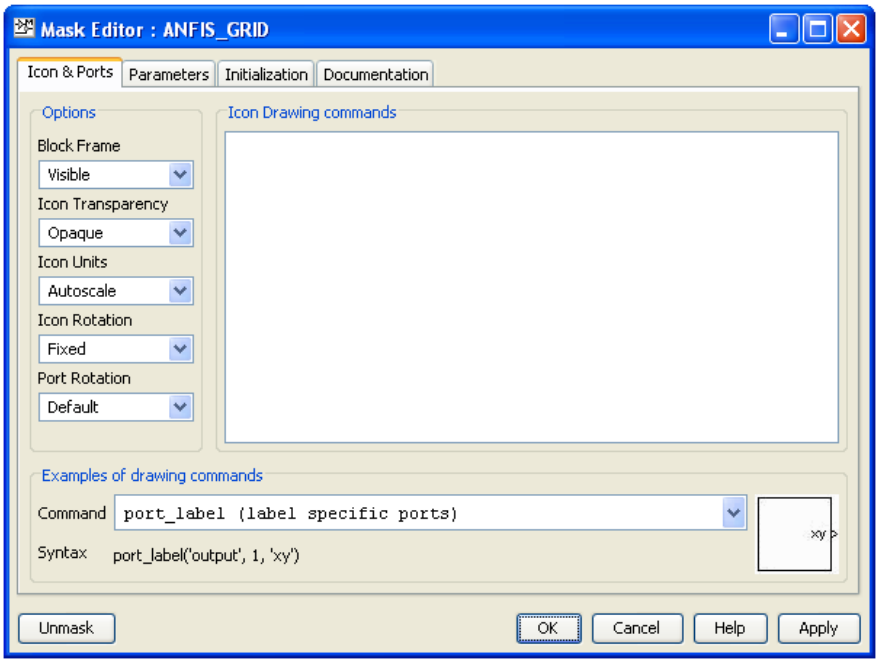


Figura A6 Ventana del Editor de Máscara del bloque ANFIS_GRID

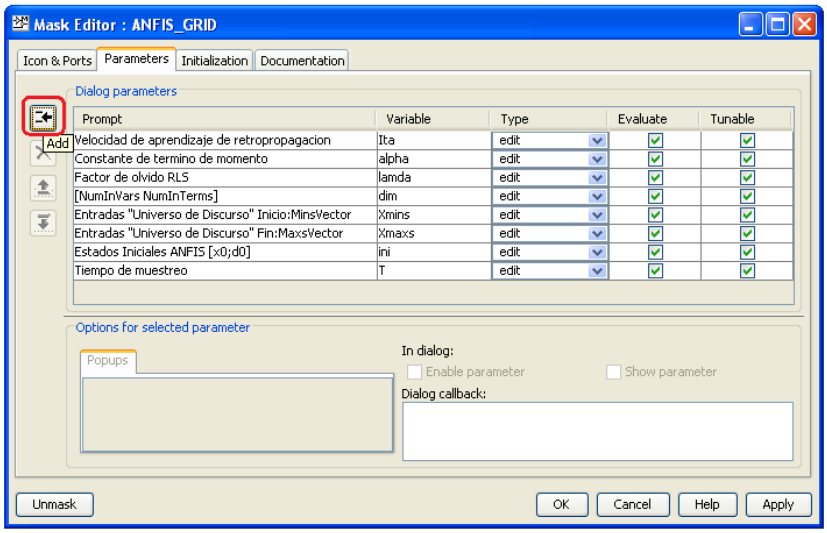


Figura X7 Ventana del Editor de Máscara – Creación de los parámetros.

En el panel *Initialization* se ha desarrollado un código de programación (ver Figura A8) que es capaz de determinar los parámetros iniciales del sistema ANFIS, tanto los parámetros no lineales de la Capa 1 como los parámetros lineales de la Capa 4. El objetivo de este código inicial es que la red neuronal empiece su proceso de aprendizaje partiendo de una información previa, con lo cual el tiempo de aprendizaje se acorta. Para ver el código completo de la inicialización revisar el Anexo A.

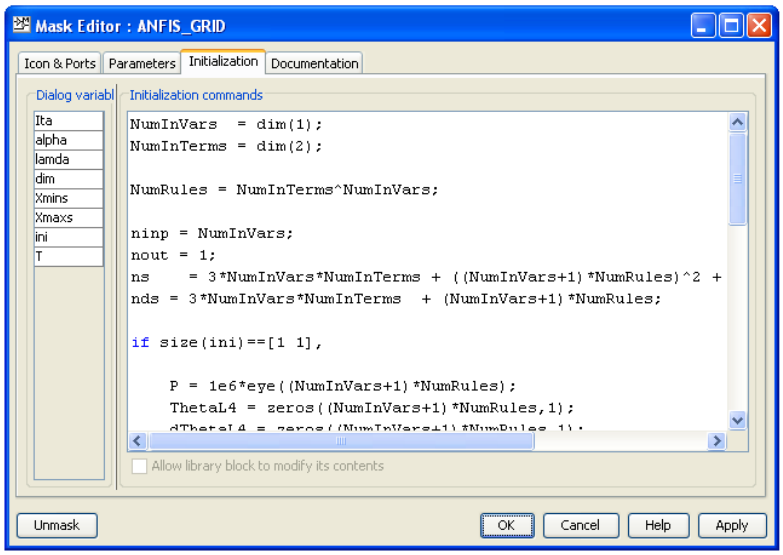


Figura A8 Ventana del Editor de Máscara – Definición del código de inicialización.

En el panel *Documentation* se ha realizado una pequeña descripción del subsistema, haciendo énfasis de que se trata de un bloque que representa el algoritmo de un sistema ANFIS (ver ventana de la Figura A9).

Una vez que se hayan realizado todas las configuraciones antes indicadas, se da clic en el botón OK. Para verificar que todos los cambios se han aplicado se da doble clic en el bloque ANFIS_GRID tras lo cual se despliega la ventana mostrada en la Figura X10. Según se observa se han creado satisfactoriamente todos los elementos y comentarios configurados en la ventana del editor de máscaras.

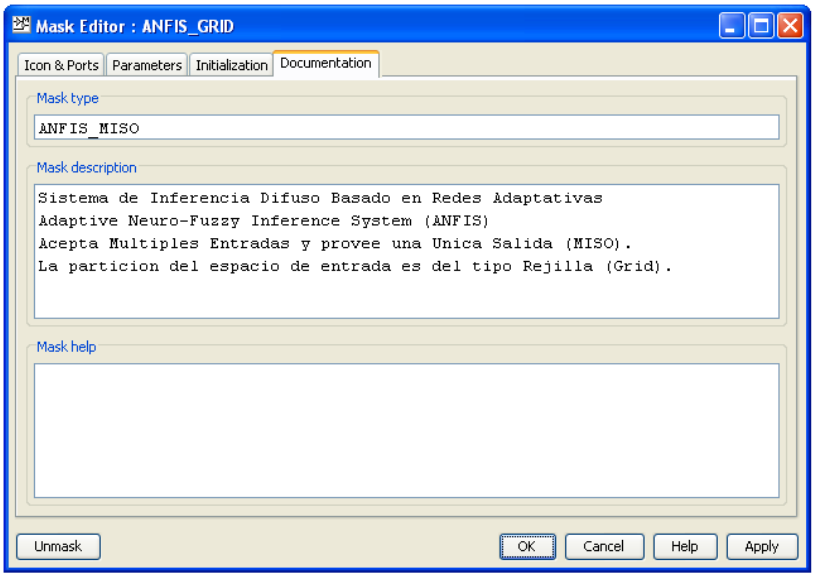


Figura A9 Ventana del Editor de Máscara – Descripción de la máscara.

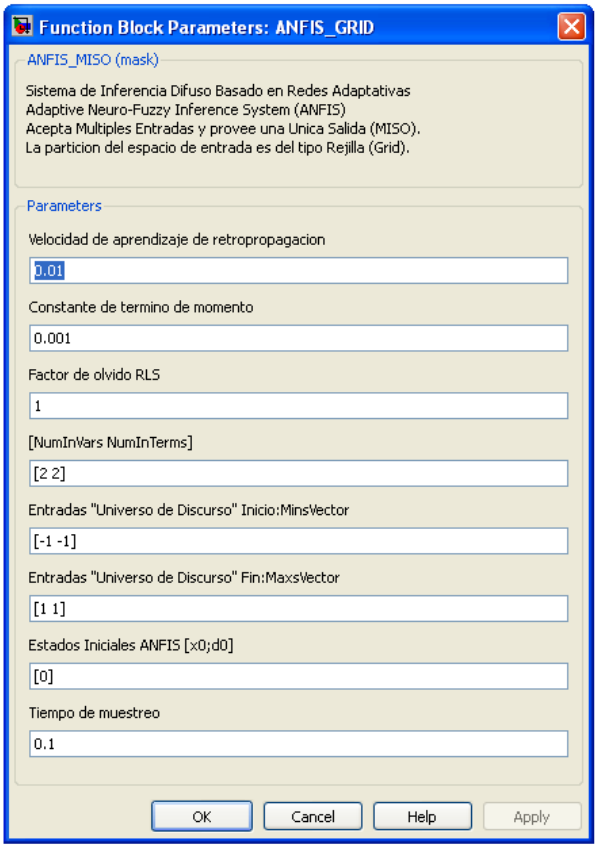


Figura A10 Ventana que se crea después de aplicar el enmascaramiento.

En los recuadros en blanco de la Figura X.10 es donde se configuran los valores de los parámetros adicionales del bloque S-Function que posteriormente serán pasados a la función *anfisim_grid.m* y que son fáciles de identificar gracias a la descripción provista en la parte superior de cada recuadro.

La máscara creada es factible de ser modificada o borrada, simplemente haciendo clic derecho en el menú contextual del bloque ANFIS_GRID y seleccionando *Edit Mask*.

ANEXO II

CÓDIGO EN MATLAB ANFIS_GRID[2]

```
function [out,Xt,str,ts] =
anfisim_grid(Ti,Xt,u,flag,Ita,alpha,lamda,NumInVars,NumInTerms,x0,T)
 % Este programa es una implementación de un sistema ANFIS (MISO) en línea.
% La estructura de la red neuronal es determinada y configurada por el usuario.
% El espacio de entrada es particionado usando el método tipo rejilla (grid).
% Los parámetros de premisa (no-lineales) en la Capa 1 son estimados por el
% Gradiente Descendente a través de retro propagación del error.
% Los parámetros del consecuente (lineales)en la Capa 4 son estimados por
% el algoritmo del Estimador de Mínimos Cuadrados Recursivo (RLSE)
 NumRules = NumInTerms^NumInVars; % Numero de reglas difusas para una
partición tipo Rejilla (Grid).
 %% ---------------------- % Información Inicial ----------
if flag == 0
      ninps = NumInVars+2; % número de entradas para el bloque S-Function: [ x y
LE]
      ns = 3*NumInVars*NumInTerms + ((NumInVars+1)*NumRules)^2 +
(NumInVars+1)*NumRules;
      nds = 3*NumInVars*NumInTerms + (NumInVars+1)*NumRules;
      out = [0,ns+nds,1+ns+nds,ninps,0,1,1];  % #estados continuos, #estados
discretos, #salidas, #entradas, sin uso, f, #ts
      str = [];                  % Cuando flag=0 esta variable siempre se establece
como una matriz vacia
      ts = T;                    % tiempo de muestreo
      Xt = x0;                   % vector columna con los valores de los estados
iniciales
 %%  ----------Actualizacion de Estados Discretos----------
elseif flag == 2
    x = u(1:NumInVars);
    e = u(NumInVars+1);
    learning = u(NumInVars+ 2);
 if learning == 1
   off=1;
   off_end=NumInVars*NumInTerms;
```

[2] Ing. Junior Rafael Figueroa Olmedo

```
   mean1=reshape(Xt(off:off_end),NumInVars,NumInTerms);
   off=off_end+1;
   off_end=off + NumInVars*NumInTerms-1;
   sigma1=reshape(Xt(off:off_end),NumInVars,NumInTerms);
   off=off_end+1;
   off_end=off+NumInVars*NumInTerms-1;
   b1=reshape(Xt(off:off_end),NumInVars,NumInTerms);
   off=off_end+1;
   off_end=off + ((NumInVars+1)*NumRules)^2-1;
P=reshape(Xt(off:off_end),(NumInVars+1)*NumRules,(NumInVars+1)*NumRules)
;
   off=off_end+1;
   off_end=off + (NumInVars+1)*NumRules-1;
   ThetaL4 = Xt(off:off_end);
   off=off_end+1;
   off_end=off + NumInVars*NumInTerms-1;
   dmean1=reshape(Xt(off:off_end),NumInVars,NumInTerms);
   off=off_end+1;
   off_end=off + NumInVars*NumInTerms-1;
   dsigma1=reshape(Xt(off:off_end),NumInVars,NumInTerms);
   off=off_end+1;
   off_end=off + NumInVars*NumInTerms-1;
   db1=reshape(Xt(off:off_end),NumInVars,NumInTerms);
   off=off_end+1;
   off_end=off + (NumInVars+1)*NumRules-1;
   dThetaL4 = Xt(off:off_end);  % Presente con fines de crecimiento futuro. No juega
ningún papel en esta versión.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
  %              OPERACION HACIA ADELANTE            %
  %                                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
  % CAPA 1: Capa de Fusificacion
    In1 = x*ones(1,NumInTerms);
    Out1 = 1./(1 + (abs((In1-mean1)./sigma1)).^(2*b1));
```

```
% CAPA 2: Capa del Antecedente de la Regla
 precond = combinem(Out1);
 Out2 = prod(precond,2)';
 S_2 = sum(Out2);
% CAPA 3: Capa de Normalización de la Regla
 if S_2~=0
    Out3 = Out2/S_2;
 else
    Out3 = zeros(1,NumRules);
 end
% CAPA 4: Capa del Consecuente de la Regla
 Aux1 = [x; 1]*Out3;
% Nuevos datos de entrenamiento en forma de un vector columna.
 a = reshape(Aux1,(NumInVars+1)*NumRules,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%          SECCION DE APRENDIZAJE DE PARAMETROS         %
%              RETROPROPAGACION DEL ERROR               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CAPA 4.
 ThetaL4_mat = reshape(ThetaL4,NumInVars+1,NumRules);
% Algortimo de Retropropagacion del Error
% CAPA 3
 e3 = [x' 1]*ThetaL4_mat*e;
% CAPA 2
 denom = S_2*S_2;
 ThetaE32 = zeros(NumRules,NumRules);
 if denom~=0
   for k1=1:NumRules
      for k2=1:NumRules
         if k1==k2
           ThetaE32(k1,k2) = ((S_2-Out2(k2))/denom)*e3(k2);
         else
           ThetaE32(k1,k2) = -(Out2(k2)./denom)*e3(k2);
         end
```

```
        end
      end
    end
    % Suma ThetaE32 a lo largo de las filas para encontrar la contribución de cada
    % nodo de la capa 3 (indexado por k2) a un solo nodo de la capa 2(indexado por
k1).
      e2 = sum(ThetaE32,2);
    % CAPA 1
    Q = zeros(NumInVars,NumInTerms,NumRules);
    for i=1:NumInVars
       for j=1:NumInTerms
          for k=1:NumRules
             if Out1(i,j)== precond(k,i) && Out1(i,j)~=0
                Q(i,j,k) = (Out2(k)/Out1(i,j))*e2(k);
             end
          end
       end
    end
    ThetaE21 = sum(Q,3);
    % AJUSTE DE LOS PARAMETROS DE PREMISA DE LA CAPA1 POR EL
GRADIENTE DESCENDENTE
    if isempty(find(In1==mean1, 1))
       deltamean1 = ThetaE21.*(2*b1./(In1-mean1)).*Out1.*(1-Out1);
       deltab1 = ThetaE21.*(-2).*log(abs((In1-mean1)./sigma1)).*Out1.*(1-Out1);
       deltasigma1 = ThetaE21.*(2*b1./sigma1).*Out1.*(1-Out1);
       dmean1 = Ita*deltamean1 + alpha*dmean1;
       mean1 = mean1 + dmean1;
       dsigma1 = Ita*deltasigma1 + alpha*dsigma1;
       sigma1 = sigma1 + dsigma1;
       db1 = Ita*deltab1 + alpha*db1;
       b1 = b1 + db1;
      % Ordena los términos en la Capa 1.
         for j=1:NumInTerms-1
            if any(mean1(:,j)>mean1(:,j+1))
              for i=1:NumInVars
                 [mean1(i,:) index1] = sort(mean1(i,:));
```

```
                sigma1(i,:) = sigma1(i,index1);
                b1(i,:) = b1(i,index1);
            end
        end
    end
  end
  % Fijacion de los parametros del consecuente por el RLSE
   P = (1./lamda).*(P - P*(a*a')*P./(lamda+a'*P*a));
   ThetaL4 = ThetaL4 + P*a.*e;
  %%%%%%%%% FINAL DEL PROCESO DE APRENDIZAJE DE LOS
PARAMETROS %%%%%%%%%
  % Almacenamiento del vector de estado
  % Xt = [mean1 sigma1 b1 P ThetaL4 dmean1 dsigma1 db1 dThetaL4]
   Xt = [ reshape(mean1,NumInVars*NumInTerms,1);
          reshape(sigma1,NumInVars*NumInTerms,1);
          reshape(b1,NumInVars*NumInTerms,1);
          reshape(P,((NumInVars+1)*NumRules)^2,1);
          ThetaL4;
          reshape(dmean1,NumInVars*NumInTerms,1);
          reshape(dsigma1,NumInVars*NumInTerms,1);
          reshape(db1,NumInVars*NumInTerms,1);
          dThetaL4;];
end   % fin del bucle "if learning==1"
out=Xt
%%  ----------------------  Salidas ------------------------
elseif flag == 3 % Descomprimir los primeros parámetros de la red...
   off=1;
   off_end=NumInVars*NumInTerms;
   mean1=reshape(Xt(off:off_end),NumInVars,NumInTerms);
   off=off_end+1;
   off_end=off + NumInVars*NumInTerms-1;
   sigma1=reshape(Xt(off:off_end),NumInVars,NumInTerms);
   off=off_end+1;
   off_end=off+NumInVars*NumInTerms-1;
   b1 =reshape(Xt(off:off_end),NumInVars,NumInTerms);
```

```
   off=off_end+1;
   off_end=off + ((NumInVars+1)*NumRules)^2 - 1;
   % P =
reshape(Xt(off:off_end),(NumInVars+1)*NumRules,(NumInVars+1)*NumRules);of
f=off_end+1;
   off_end=off + (NumInVars+1)*NumRules - 1;
   ThetaL4 = Xt(off:off_end);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %           OPERACION HACIA ADELANTE                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % CAPA 1: Capa de Fusificacion
    x = u(1:NumInVars);
    In1 = x*ones(1,NumInTerms);
    Out1 = 1./(1 + (abs((In1-mean1)./sigma1)).^(2*b1));
  % CAPA 2: Capa del Antecedente de la Regla
    precond = combinem(Out1);
    Out2 = prod(precond,2)';
    S_2  = sum(Out2)
  % CAPA 3: Capa de Normalización de la Regla
    if S_2~=0
       Out3 = Out2/S_2;
    else
       Out3 = zeros(1,NumRules);
    end
  % CAPA 4: Capa del Consecuente de la Regla
    Aux1 = [x; 1]*Out3;
    a = reshape(Aux1,(NumInVars+1)*NumRules,1);  % Nuevos datos de
entrenamiento en forma de un vector columna.
  % Capa 5: Capa de Inferencia de la Regla (Suma de nodos)
    outact = a'*ThetaL4;
  % Formacion del vector de salidas del bloque S-Function
    out=[outact;Xt];
else
   out=[];
end
```

ANEXO III

REDES NEURONALES PARA CONTROL DE PROCESOS[3]

Las metodologías de control tradicionales se basan principalmente en la teoría de sistemas lineales, mientras que los sistemas reales presentan características no lineales y tienen dinámicas que no son difíciles o imposibles de modelar, ya sea por la presencia de ruido, incertidumbre, múltiples lazos, etc., que crean problemas a los ingenieros para tratar de diseñar algoritmos de control. Es, por tanto, un gran desafío para los ingenieros diseñar un algoritmo de control eficiente. Desde el punto de vista del diseño, las especificaciones para los algoritmos de control deben ser lo suficientemente simples para ser implementadas y comprendidas con propiedades tales como la capacidad de aprendizaje, la robustez y la no linealidad. Una de las razones por las que las redes neuronales se han vuelto muy populares en aplicaciones de control es que cumplen algunos de estos criterios para el diseño e implementación. Desde un punto de vista práctico, el paralelismo inherente y la capacidad de adaptación rápida de las redes neuronales son ventajas adicionales.

La potencia de los algoritmos de aprendizaje, la variedad de arquitecturas y la capacidad de entrenamiento a partir de las funciones de entrada/salida y/o datos experimentales hacen de las redes neuronales la tecnología preferida para muchas aplicaciones. Las redes neuronales ofrecen soluciones simples a problemas complejos de control. El éxito del algoritmo de retropropagación para entrenar redes con múltiples capas ha contribuido en la aplicación de las redes neuronales para fines de control. El uso de las redes neuronales en aplicaciones de control, incluyendo el control de procesos, robótica, aplicaciones de fabricación, entre otros, ha experimentado recientemente un rápido crecimiento.

El objetivo básico del control neuronal es proporcionar la señal de entrada apropiada a un sistema físico dado (proceso o planta) para producir una respuesta deseada. Normalmente hay dos pasos a seguir cuando se utilizan redes neuronales para el control: ***la identificación del sistema*** (planta o proceso) y ***el diseño de control***.

[3] Ing. Junior Rafael Figueroa Olmedo

1.4.5.1 Identificación del Sistema

El concepto clave de la identificación es el proceso de determinar un modelo dinámico para un sistema desconocido. El modelo identificado se puede utilizar posteriormente para fines de control. La identificación del sistema consta de dos pasos principales: el primer paso es elegir un modelo paramétrico apropiado y el segundo paso es ajustar los parámetros del modelo de acuerdo con algunas leyes de adaptación de modo que la respuesta del modelo ante una señal de entrada pueda aproximar la respuesta del sistema real.

El problema de la identificación de la estructura del modelo y la estimación de sus parámetros puede formularse como un problema de aprendizaje y un mapeo entre los espacios de entrada/salida conocidos. Las redes neuronales multicapas tienen buenas capacidades de aproximación, las cuales proporcionan una herramienta poderosa para la identificación de sistemas desconocidos con no linealidades. La red neuronal se compone de señales retardadas de las entradas y salidas, con un suficiente número de capas y neuronas, que son capaces de igualar el comportamiento de entrada/salida del correspondiente mapeo no lineal de la planta. Esto implica que la función no lineal de la planta se sustituye por una red neuronal con matrices de peso fijo, pero desconocidos, la cual aprende utilizando un algoritmo de aprendizaje adecuado y un conjunto de datos disponibles.

En la fase de identificación del sistema, se desarrolla un modelo de red neural de la planta a ser controlada. El identificador se compone de una red neural multicapa incorporada en paralelo con un sistema dinámico, donde la estructura proviene de la función de error estándar en la literatura de identificación y control del sistema. La información estructural está en realidad contenida en la conectividad y los pesos de la red neuronal. La identificación del sistema puede llevarse a cabo de dos maneras: ***identificación del modelo hacia adelante*** e ***identificación del modelo directo inverso***.

La primera etapa de la identificación del modelo hacia delante (o simplemente identificación directa) es entrenar una red neuronal para representar la dinámica hacia adelante de la planta. El error entre la salida de la planta y_p y la salida del modelo de red neuronal y_m se utiliza como la señal de entrenamiento de la red neural. La configuración básica para la identificación del modelo hacia delante se muestra en la Figura 1.15 (a). El modelo estimado mediante la red neuronal utiliza las entradas y

salidas actuales y anteriores de la planta real (que se muestran como señales retardadas en la figura) para predecir los valores futuros de la salida de la planta. La red puede ser entrenada fuera de línea (*offline)* en modo por lotes, utilizando datos recogidos de la operación de la planta. Cualquiera de los algoritmos de entrenamiento de retropropagación discutidos en este capítulo puede ser utilizado para el entrenamiento de la red.

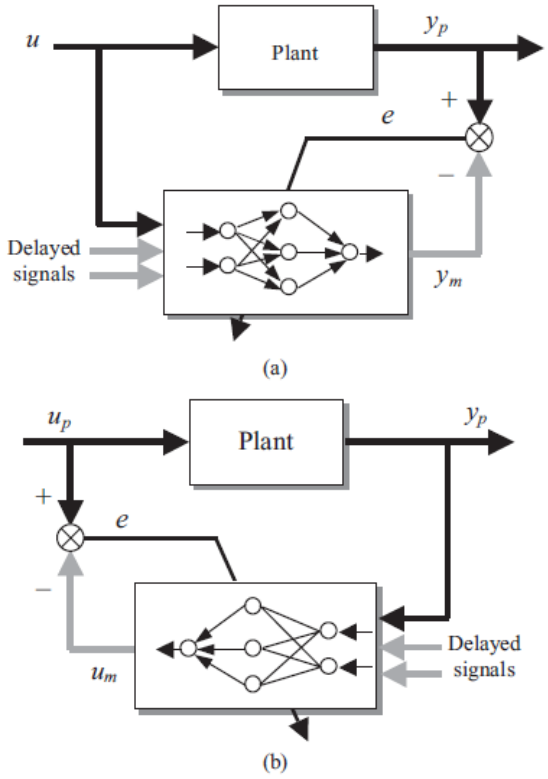


Figura 1.15 Identificación de la planta utilizando redes neuronales. (a) Identificación hacia adelante; (b) Identificación directo inverso. (Siddique & Adeli, 2013)

Tal vez uno de los esquemas de modelos neuronales más ampliamente aplicados es el enfoque del modelo directo inverso, en donde, mediante la salida de la planta real y_p la red neuronal determina cuál debió ser la entrada u_p que la produjo. Una vez que una red neuronal ha sido entrenada para aprender el comportamiento inverso de la planta, entonces esta puede ser configurada para controlar directamente la planta. La configuración básica del modelo directo inverso se muestra en la Figura 1.15 (b). En la arquitectura del modelo directo inverso, la red es entrenada fuera de línea (*offline)* utilizando patrones obtenidos a partir de las características de lazo abierto de la planta (o en lazo cerrado).

Existen otros modelos para la identificación de sistemas no lineales mediante redes neuronales, pero en este documento solo se van a tratar los dos antes mencionados.

1.4.5.2 Diseño de Control

La investigación sobre los sistemas de control basados en redes neuronales ha recibido considerable atención en los últimos años. Las estructuras de control neural más ampliamente utilizadas son similares a las empleadas en los sistemas de control adaptativos. Una red neuronal se utiliza para estimar el sistema no lineal desconocido y la formulación de control es entonces diseñada usando la red neuronal estimada. El proceso de estimación utiliza las entradas y salidas medidas del sistema real y se logra a través del uso de varios tipos de arquitecturas de redes neuronales. El objetivo de este apartado es presentar algunas de las estructuras típicas de control mediante redes neuronales. Los esquemas considerados son: Control Directo, Control Indirecto, Control por Retropropagación a Través del Tiempo, Control Directo Inverso y Control Adaptativo

Control Directo o Aprendizaje Especializado

En la arquitectura de control directo (Figura 1.16), el controlador neuronal es entrenado en línea (*online*) para minimizar alguna norma del error entre la señal de salida y_p y la de referencia r. El error se propaga hacia atrás a través de la planta en cada muestra, para ajustar los parámetros de la red. Esta arquitectura fue nombrada como control de aprendizaje especializado por Psaltis et al. (1999). Una ventaja de este tipo de control es que la identificación de la planta no está implicada en este método.

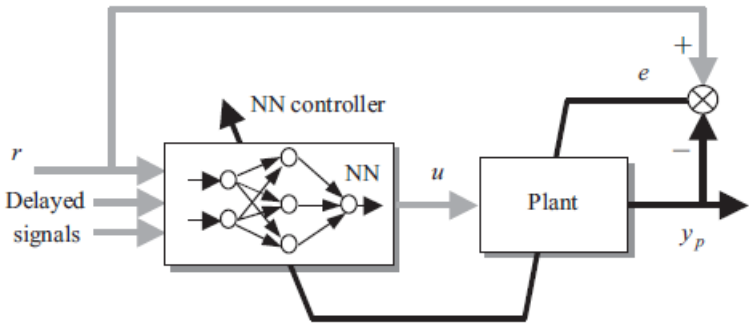


Figura 1.16 Arquitectura de control directo o aprendizaje especializado (Siddique & Adeli, 2013)

Sin embargo, debido a la ubicación de la planta, el Jacobiano de la planta (es decir,$\partial y/\partial u$) es requerido (Saerens y Soquet, 1989), el cual es difícil de obtener si la dinámica de la planta no se conoce a priori. Con el fin de evitar esto, los elementos del Jacobiano pueden ser aproximados por sus signos, que son las orientaciones de los parámetros de control que influyen en las salidas de la planta.

El entrenamiento involucra a la señal de referencia r como señal de entrada al controlador neuronal. El error $e = r - y_p$ es propagado hacia atrás a través de la planta y utilizado para ajustar los parámetros del controlador. El objetivo del entrenamiento del controlador neuronal es generar la señal de control requerida u para conducir la planta a la salida deseada, tal que la salida de la planta y_p coincida con la señal de referencia r sobre el entrenamiento por épocas k, es decir, $\lim_{k\to\infty} |r - y_p| \leq \varepsilon$ donde $\varepsilon \geq 0$.

La desventaja del control directo es que la estabilidad inicial de la planta no está garantizada para este método de control. A pesar de las desventajas, muchos investigadores han aplicado el esquema de control adaptativo directo a una variedad de sistemas no lineales desconocidos con gran éxito y un mejor rendimiento (Noriega y Wang, 1998; Park et al., 2005).

Control Indirecto o Aprendizaje Inverso Especializado

Las principales desventajas en el esquema de control directo eran la determinación del Jacobiano y la estabilidad inicial de la planta. Cuando el inverso de la planta (o el Jacobiano de la planta) no está bien definido o la estabilidad inicial de la planta es fundamental, un esquema de control indirecto es considerablemente más éxito que el control directo.

En esta arquitectura, un emulador neuronal de la planta es entrenado para representar la respuesta de la planta. El esquema del entrenamiento del emulador neuronal se muestra en la Figura 1.17 (a), que es el mismo descrito en la identificación hacia delante de la planta. La diferencia entre la salida de la planta y_p y la respuesta del emulador $\hat{y}$ se utiliza para ajustar los parámetros del emulador neuronal. Una vez que el emulador neuronal está lo suficiente entrenado, este es utilizado para entrenar un controlador neuronal con la misma señal de referencia r y del error $e_c = \hat{y} - r$. El error e_c se propaga hacia atrás a través del emulador neuronal para ajustar los

parámetros del controlador neuronal. El entrenamiento del controlador neuronal es mostrado en Figura 1.17 (b).

Este esquema de control puede ser encontrado con diferentes nombres en la literatura, como el control inverso anticipativo (Narendra, 1995) y aprendizaje inverso especializado (Hunt et al., 1992). La ventaja añadida del control indirecto es que los parámetros del emulador neuronal pueden ser reajustados en línea durante el funcionamiento del controlador si es que el emulador neuronal parece no ser exacto. Una variante cercana del control indirecto es el control por modelo interno, en el cual un modelo interno se coloca en paralelo con la planta y un controlador es utilizado en serie con la planta. El modelo interno es eventualmente un modelo de la planta hacia adelante como se muestra en la Figura 1.15 (a). Detalles de la arquitectura de control por modelo interno se pueden encontrar en Sen et al. (1998).

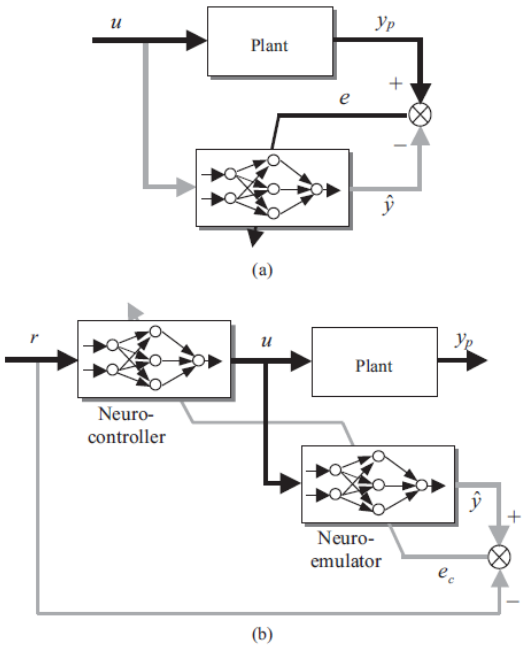


Figura 1.17 Arquitectura del control indirecto. (a) Entrenamiento del emulador neuronal; (b) Entrenamiento del controlador neuronal. (Siddique & Adeli, 2013)

Una de las desventajas del esquema de control indirecto parece ser la robustez del controlador, puesto que no hay un lazo de retroalimentación utilizado en la estrategia de control. Mientras que el entrenamiento del controlador neuronal emplea la misma señal de referencia r y la señal de error $e_c = \hat{y} - r$ (que se propaga hacia atrás a través del controlador neuronal), una mala convergencia y un control inestable

pueden ser un problema en la etapa inicial. A pesar de estas desventajas, el control indirecto tiene más éxito que el control directo y ha encontrado una amplia gama de aplicaciones (Nguyen & Widrow., 1990; Wu et al., 1992; Khalid et al, 1993; Yang & Linkens, 1994).

Control por Retropropagación a Través del Tiempo

Hay otro modelo de arquitectura de control de red neural que utiliza un algoritmo de aprendizaje de retropropagación que fue propuesto por Jordan y Rumelhart (1990), Narendra & Parthasarathy (1990) y Nguyen & Widrow (1990). Werbos (1990b) clasificó este método como la arquitectura de retropropagación a través del tiempo. La arquitectura se asemeja a muchas estructuras tradicionales de control adaptativo, conocidas como controles indirectos adaptativos.

En este esquema de control se utilizan dos redes neuronales para controlar la planta, como se muestra en la Figura 1.18. La primera red neuronal es un emulador neuronal, el cual puede ser entrenado fuera de línea (offline) usando una arquitectura de aprendizaje generalizada o incluso en línea (online) mediante la inyección de entradas aleatorias (randómicas) para aprender la dinámica hacia delante de la planta. La segunda red neuronal es un controlador. Esta arquitectura permite el entrenamiento en línea (online) del controlador neuronal así como el error $e = |r - y_p|$ puede ser propagado hacia atrás a través del emulador en cada muestra. Algunas aplicaciones de esta arquitectura se pueden encontrar en Omatu et al. (1995).

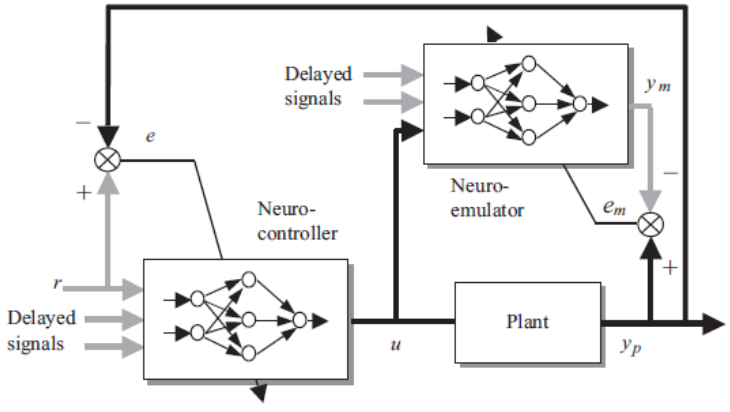


Figura 1.18 Arquitectura de retropropagación a través del tiempo (Siddique & Adeli, 2013)

Control Directo Inverso

Este es el esquema de control neuronal que más se aplica. Como el término lo sugiere, un control directo inverso utiliza un modelo inverso de la planta que está simplemente en cascada con la misma, con la finalidad de que la respuesta deseada (entrada del controlador neuronal) sea igual a la salida real de la planta, a la cual previamente se le aplicó un proceso de identificación inversa. El término "control directo inverso" es adoptado desde Werbos (1990a).

El proceso de modelado se muestra en la Figura 1.19 (a), primero construye un sistema de identificación inversa de la planta para estimar la salida del modelo inverso $\hat{u}$. La salida estimada $\hat{u}$ es comparada con la señal entrenada u y el error $e = u - \hat{u}$ se utiliza para entrenar el modelo inverso. Una vez obtenido el modelo inverso, este se conecta en cascada con la planta para funcionar como si se tratase de un controlador de lazo abierto. Los parámetros del controlador neuronal se ajustan directamente. La arquitectura de control directo inverso se muestra en la Figura 1.19 (b). La señal de referencia r debe cubrir un espacio de entrada/salida lo suficientemente grande, mientras se realiza la construcción del modelo inverso antes indicado.

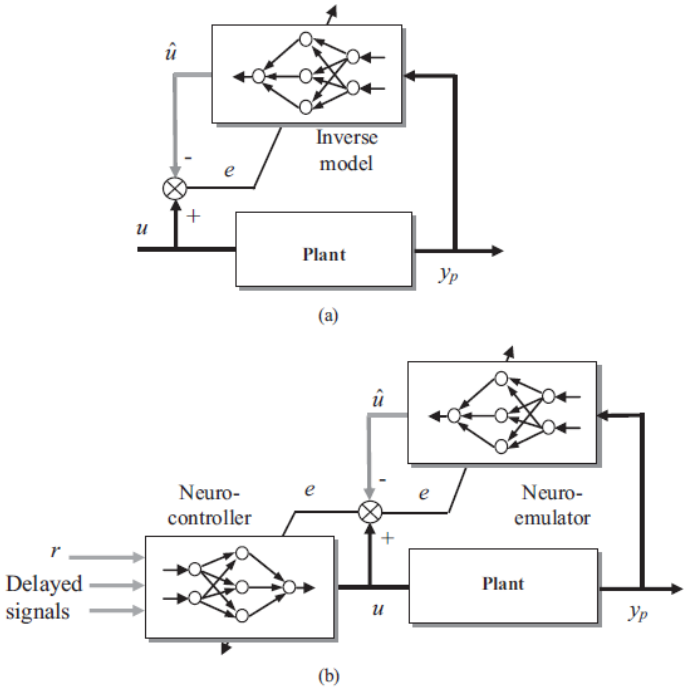


Figura 1.19 Arquitectura del control directo inverso. (a) Modelamiento inverso; (b) Control en lazo abierto.