ESCUELA POLITECNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA,

AUTOMATIZACIÓN Y CONTROL

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO

EN INGENIERÍA ELECTRÓNICA

# “DISEÑO E IMPLEMENTACIÓN DE CONTROLADORES INTELIGENTES PARA EL SISTEMA DE LEVITACION MAGNETICA MLS”

CRISTIAN NOÉ ZÚÑIGA YUNDA

SANGOLQUI – ECUADOR

2011

# CERTIFICACION

Por medio de la presente certificamos que el proyecto de grado para la obtención del título en ingeniería electrónica **“DISEÑO E IMPLEMENTACIÓN DE CONTROLADORES INTELIGENTES PARA EL SISTEMA DE LEVITACIÓN MAGNÉTICA MLS”** fue desarrollado en su totalidad por el Señor Cristian Noé Zúñiga Yunda

Atentamente,

Ing. Víctor Proaño — **Director**

Ing. Hugo Ortiz — **Codirector**

## AGRADECIMIENTO

A Dios por ofrecerme la dicha de vivir y de esta manera poder luchar diariamente por un futuro más próspero, a mis padres, hermano y familiares por todo el apoyo de confianza que me brindaron durante éste duro camino. Agradezco profundamente a mi novia Patricia quien fue mi apoyo durante el transcurso de mi vida universitaria y quien estuvo siempre a mi lado, a los Ingenieros Víctor Proaño y Hugo Ortiz quienes fueron los que me permitieron desarrollar el estudio y ejecución del presente proyecto y me brindaron todo su apoyo, compartieron sus conocimientos y me ayudaron a ser mejor profesional mediante la predica del ejemplo y responsabilidad.

Finalmente a todos mis amigos y compañeros con los cuales compartí muchos momentos buenos y malos logrando así llegar a la unidad y como meta final culminar con éxito mi carrera.

En fin, gracias a todas las personas que estuvieron a mi lado de una u otra forma e hicieron que haya llegado a ser la persona que soy en este momento

A todos ustedes quedo eternamente agradecido.

# DEDICATORIA

El presente trabajo dedico con todo amor a las personas que más amo en este mundo , Normita mi madre de quien adquirí la tenacidad de seguir adelante ante cualquier dificultad y quien me demostró su apoyo, amor y comprensión quien ha sido mi guía constante a lo largo de toda mi vida dentro del buen camino, a mi novia Patricia quien siempre me animó a ser el mejor en todos los aspectos relacionados con mi formación profesional y un pilar fundamental en mi camino de superación personal, a mi hermano David quien siempre supo darme una mano en momentos angustiosos, y en especial dedico la culminación de este proyecto a mi Dios quien fue mi torre fuerte y el que nunca me abandona en todos mis pasos a seguir.

Todos mis éxitos de aquí en adelante se los dedico a ustedes.

# PRÓLOGO

El proyecto “Diseño e implementación de controladores inteligentes para el sistema de levitación magnética MLS”, consiste en el diseño de controladores inteligentes para un sistema de levitación magnética que controla la posición de una esfera metálica, las cuales se desarrollarán mediante la aplicación de técnicas de control con lógica difusa y redes neuronales, a través del programa de cálculo MATLAB.

Este proyecto está abierto en el sentido de aceptar mejoras y servir como base para la realización de un trabajo similar, además de permitir el análisis de diferentes sistemas de Control Automático.

El problema tratado en este proyecto surgió como el resultado de la teoría de control inteligente dictada en el área de Control y Automatización. Bajo esta condición, el propósito del proyecto es aplicar los conocimientos adquiridos a un sistema dinámico en tiempo real; específicamente la idea es desarrollar un control con lógica difusa y redes neuronales para el sistema de levitación magnética.

De esta manera la idea es investigar el comportamiento del sistema de levitación magnética con técnicas de control inteligente.

Para lograr el objetivo, se considera tres estrategias alternativas de control, como son la lógica difusa PD, lógica difusa PID y finalmente redes neuronales. Los controladores con lógica difusa serán diseñados con la herramienta “Fuzzy Editor” del programa de cálculo MATLAB, mientras que para el diseño con redes neuronales se creará un código de MATLAB.

Para realizar la simulación de los controladores inteligentes conjuntamente con el sistema de levitación magnética se ve la necesidad de utilizar SIMULINK que es una herramienta de MATLAB.

El control que se aplicará en el sistema de levitación magnética está enfocado al control de posición de una esfera metálica y de esta manera presentar los resultados obtenidos tanto en simulación como en el sistema experimental.

El sistema de levitación magnética hoy en día constituye una de las herramientas del futuro ya que con esta tecnología se espera tener un futuro prometedor dentro del transporte masivo, en la actualidad ya se tiene trenes de levitación magnética implementados en las grandes potencias quienes son los que buscan nuevas maneras de sacar provecho a estos sistemas.

## INDICE DE CONTENIDOS

# CAPÍTULO I

# INTRODUCCIÓN

# CAPÍTULO I

# INTRODUCCIÓN

El trabajo consiste básicamente en desarrollar controladores inteligentes y determinar el comportamiento en el sistema de levitación magnética, los cuales se desarrollan a través del programa de cálculo MATLAB.

Se presenta una alternativa al diseño de controladores clásicos, como lo es el control con lógica difusa y redes neuronales, que a pesar de sus diferentes procesos de diseño, toma como punto de partida las experiencias adquiridas en los controladores clásicos.

La Escuela Politécnica del Ejército tiene varios sistemas de control automático con interfaz a computadora en los que se puede aplicar la teoría de control inteligente. Uno de estos sistemas es el levitador magnético MLS que tiene sistemas de control implementados como PID y un LQ, los cuales ya se encuentran documentados en trabajos pasados. El sistema de levitación magnética que se dispone, permite la ejecución de controladores en el entorno Simulink de MATLAB y es allí donde se realizará el trabajo para obtener controladores inteligentes funcionales que tendrán por objetivo investigar su comportamiento.

A continuación se plantea la justificación al problema y también se presenta el alcance del mismo.

## 1.1 JUSTIFICACIÓN E IMPORTANCIA.

El Departamento de Eléctrica y Electrónica tiene como una de sus tareas el desarrollo de investigación. El proyecto que se pretende realizar se enfoca dentro de la línea de investigación de Automatización y Control, específicamente en el campo de Control Inteligente. El proyecto, permitirá la mejor difusión y uso de las herramientas informáticas para el diseño de controladores inteligentes.

Se podrá tener un documento de fácil acceso para el estudio de controladores en el campo de control inteligente. El documento generado como resultado del proyecto se constituirá en una valiosa herramienta para el estudio de controladores inteligentes aplicados a un sistema real como es el levitador magnético, el texto podrá ser utilizado como un texto guía para la realización de prácticas en el MLS.

El documento se detallará de manera clara y concreta en el diseño, implementación y resultados de la aplicación de estos controladores al levitador magnético.

### 1.2 ALCANCE DEL PROYECTO

Se estudiará el sistema de levitación magnética para determinar las posibilidades y limitaciones del sistema que serán descritas en el documento del proyecto esto es necesario realizar para poder tener un conocimiento de cómo

funciona el sistema de levitación magnética para poder diseñar un controlador inteligente.

Tomando en cuenta lo antes mencionado se realizará el diseño de controladores inteligentes: controlador difuso PD, controlador difuso PID, controlador con redes neuronales mediante modelo de referencia. Realizando la simulación de los sistemas de control Inteligente en MATLAB y se implementará los controladores para el sistema levitador magnético, registrando los resultados que se obtienen.

## 1.3 FUNCIONAMIENTO DEL SISTEMA MLS

El sistema de levitación magnética “MLS” es un completo laboratorio de control listo para realizar experimentos.

El “MLS” siglas en inglés “Magnetic Levitation System” que traducido al español significa; Sistema de Levitación Magnética, es una herramienta ideal para la demostración del fenómeno de levitación magnética que es un clásico problema de control usado en muchas aplicaciones prácticas.

El levitador magnético consta de un electro-magneto, una esfera hueca metálica, un sensor de posición y una tarjeta de interfaz. Figura 1.1.

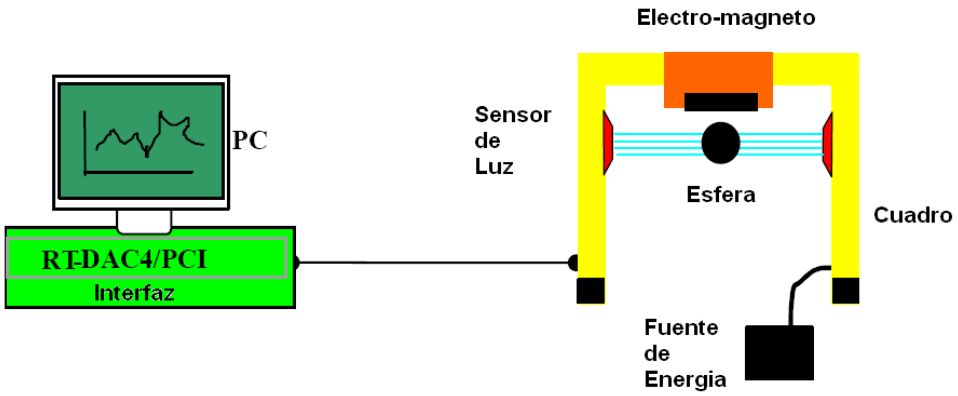



**Figura. 1.1. Laboratorio de Levitación Magnética**

El laboratorio de levitación magnética es un sistema que consta con un electro magneto que suspende a una esfera de metal dentro de un campo magnético generado por el electro magneto, la posición de la esfera es controlada por medio de un sensor y esta información es enviada mediante la tarjeta de interfaz al ordenador para realizar su debido control de posición.

El levitador magnético es una aplicación de control en la que el usuario desarrolla en Simulink el bloque de control y lo conecta al modelo de experimentación. Con ello se tiene acceso al control y la medición de las variables de posición, velocidad y corriente en el levitador. El sistema de levitación realiza la transformación de las señales de control a un valor de corriente correspondiente y para ello utiliza la tarjeta de interfaz que aplica la técnica PWM. La posición, velocidad y corriente son medidos e ingresados al entorno de trabajo de Matlab mediante la tarjeta de adquisición RT-DAC4/PCI I/O configurado en tecnología Xilinx®. Las tareas de generación de PWM y adquisición de señales que realiza el sistema son transparentes para el usuario. Figura 1.2.

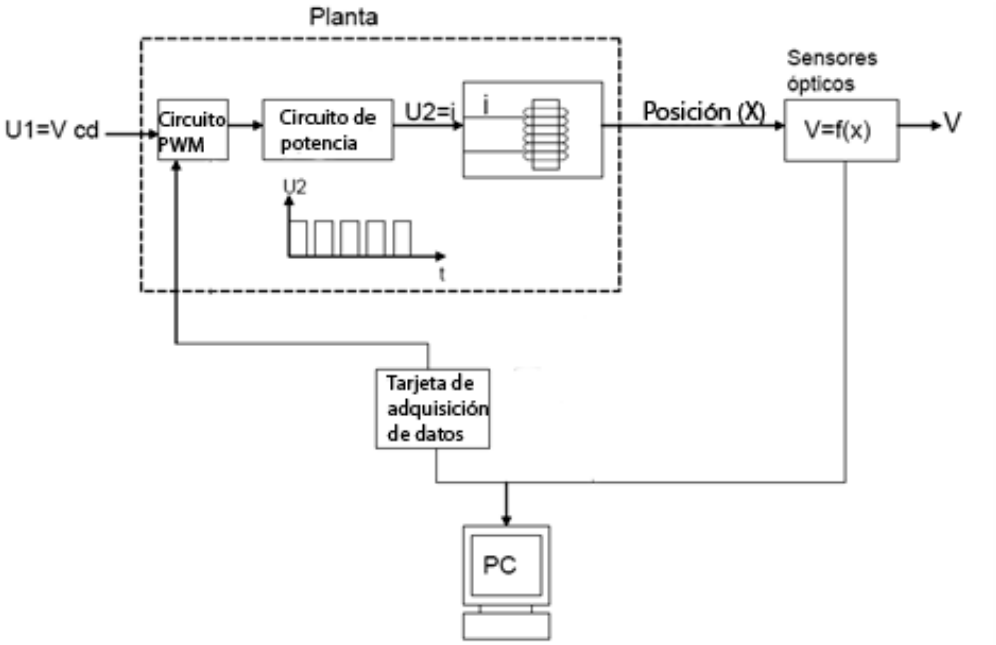



**Figura.1.2. Esquema del laboratorio de Levitación Magnética**

Para realizar el control de posición de la esfera la tarjeta RT-DAC4 se utiliza la técnica PWM, que modifica el ciclo de trabajo de una señal periódica (una sinusoidal o una cuadrada) ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

El levitador magnético utiliza una tabla en la que se detalla el respectivo valor de corriente y voltaje al ciclo de trabajo PWM que se aplica. Tabla. 1.1.

| Ciclo de trabajo PWM | Amperaje [A] | Voltaje [V] |
|---|---|---|
| 0 | 0 | 0,374811 |
| 0,1 | 0,25 | 0,262899 |
| 0,2 | 0,51 | 0,510896 |

| | | |
|---|---|---|
| 0,3 | 0,77 | 0,752465 |
| 0,4 | 1,02 | 0,993620 |
| 0,5 | 1,28 | 1,229133 |
| 0,6 | 1,52 | 1,459294 |
| 0,7 | 1,74 | 1,651424 |
| 0,8 | 1,99 | 1,875539 |
| 0,9 | 2,21 | 2,076814 |
| 1 | 2,43 | 2,269865 |

**Tabla. 1.1. Datos del levitador**

La herramienta del sistema MLS tiene una colección de funciones-M, modelos-MDL y código-C, archivos DLL que producen un ambiente de trabajo para desarrollar diseños y problemas de control.

El software integrado soporta todas las fases del desarrollo de un sistema de control:

- Proceso de identificación en línea,
- Modelamiento de sistema de control, diseño y simulación,
- Implementación de algoritmos en tiempo real.

Las herramientas del sistema MLS proveen al usuario una variedad de capacidad de software:

- Información que puede fluir entre el proceso y el ambiente de Matlab en línea,

- Experimentar el control en tiempo real de los algoritmos demostrativos,
- Desarrollo, simulación y aplicación de algoritmos de control definidos por el usuario.

Las herramientas del sistema MLS vienen incluidas en el software y manuales del levitador.

### 1.3.1 Descripción de hardware

Los componentes con mayor importancia que contiene el levitador se detalla a continuación:

- Tarjeta de interface RTDAC4/PCI adquisición y control I/O,
- Circuito de potencia
- Sensor de posición,

#### 1.3.1.1 Tarjeta de interfaz RTDAC4/PCI

La tarjeta de adquisición de datos es la tarjeta RT-DAC4/PCI que es una tarjeta multifuncional analógica y digital de I/O dedicada a la adquisición de datos en tiempo real y control bajo un ambiente Windows 95/98/NT/2000. La tarjeta utiliza un bus PCI y soporta operaciones en tiempo real aún con los retardos causados por Windows, algunas de sus características para entradas analógicas son:

- 16 canales sencillos multiplexados, resolución de 12 bits, rango de entrada ±10V, ganancias programables de (x1, x2, x4, x8, x16)

- Tiempo de conversión 1,6 es. El voltaje de referencia se encuentra en la misma tarjeta,

- Para las salidas analógicas: 4 canales, resolución de 12 bits, rango de salida +10V , −10V (polares), ±10V (bipolar), voltaje de referencia en la tarjeta,

- Entradas y salidas digitales: 32 Canales bidireccionales, se direccionan individualmente vía software, Voltaje de entrada para niveles altos VIH = 2,0V a 3,6V, para niveles bajos VIL = −0,5V a 0,8V, voltaje de salida: VOH = 2,4V (min), VOL = 0,4V (máx.), corriente de salida en el rango de 2mA a 24mA por canal.

El esquema interior de la tarjeta RTDAC4/PCI se presenta en la Figura. 1.3.

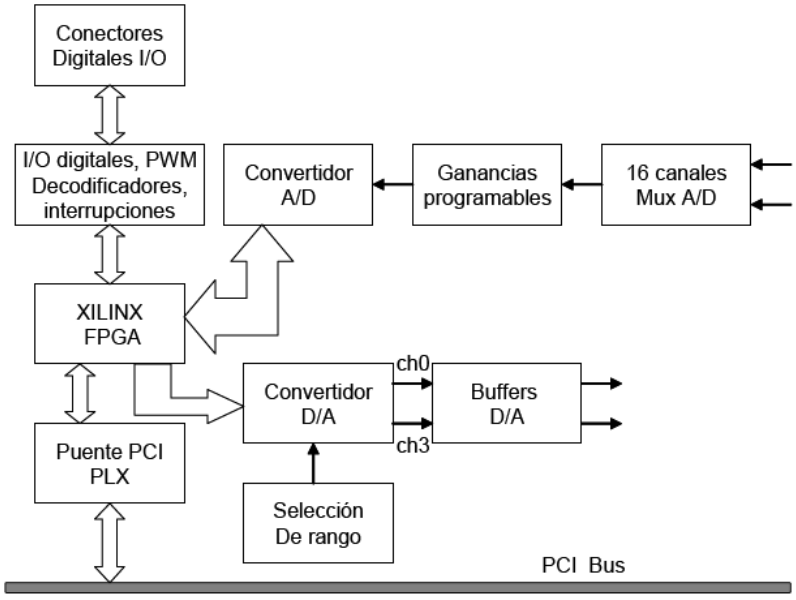



**Figura. 1.3. Esquema interior de la tarjeta RT-DAC4/PCI.**

La tarjeta contiene multiplexores de entrada analógicos conectados a 16 canales de entrada analógicos, el rango de voltaje es bipolar ±10V.

La tarjeta está equipada con un convertidor A/D de 12 bits de aproximaciones sucesivas que entrega una resolución de 5mV con un rango de entrada de ±10V. El tiempo de conversión del convertidor A/D de la tarjeta RT-DAC4/PCI es igual a 1,6µs.

La tarjeta contiene cuatro convertidores D/A de 12 bits, conectado a cuatro canales de salida analógicos, los canales pueden ser configurados por hardware, cada canal de salida analógica puede entregar hasta 10mA, existen 32 líneas digitales de I/O en la tarjeta RTDAC4/ PCI, las direcciones pueden ser configuradas independientemente. La tarjeta RT-DAC4/PCI está equipada con convertidores A/D paralelos de 12 bits.

#### 1.3.1.2 Circuito de Potencia

El circuito de potencia o driver es el encargado de elevar el nivel de corriente para hacer trabajar a la bobina o electro magneto del sistema, la bobina consume hasta 2.43 A como máximo. La etapa de potencia es controlada por el LMD18200 que es un Puente-H 3A diseñado para aplicaciones de control de movimiento. El dispositivo está construido utilizando tecnología multiproceso que combina circuitería bipolar y control CMOS en su estructura monolítica. Ideal para la conducción de los motores DC y motores paso a paso.

El fabricante presenta la curva entre la señal de voltaje medido y la corriente de la bobina. Figura 1.4.

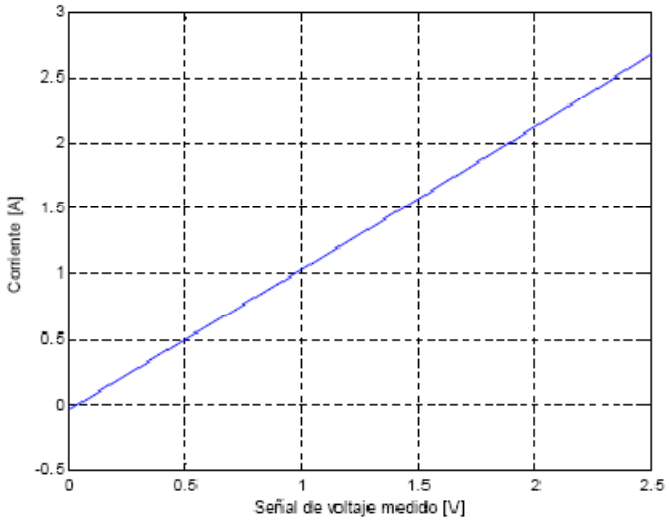



**Figura. 1.4. Característica aproximada de Corriente vs Voltaje de la bobina del electroimán.**

Se presenta el diagrama interno del dispositivo LMD18200. Figura 1.5.

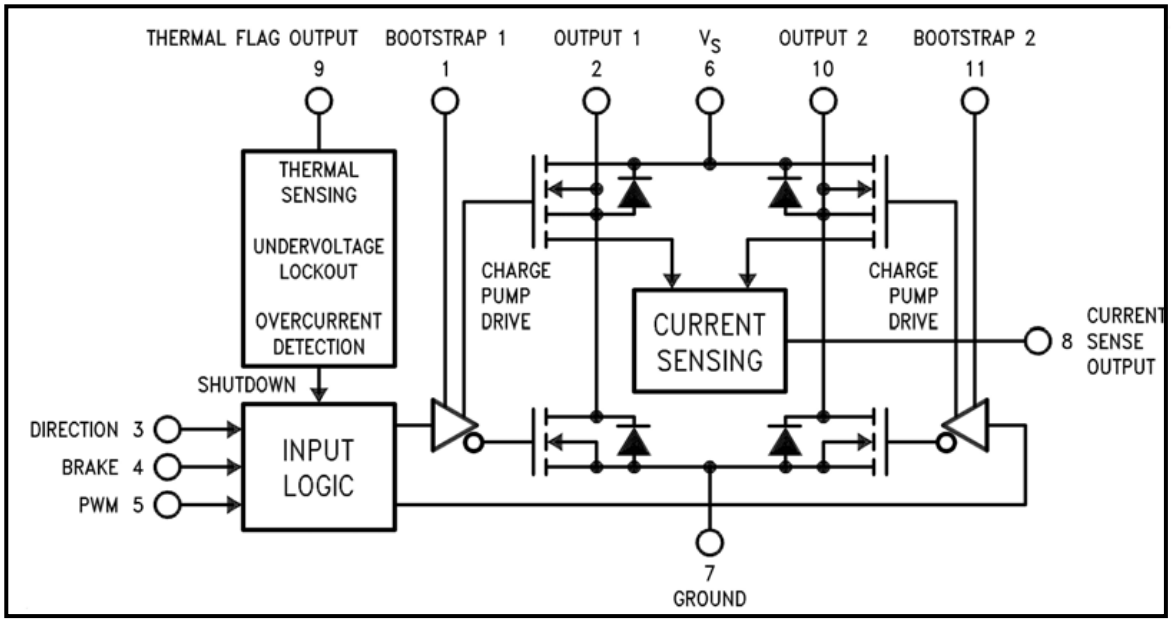



**Figura. 1.5. Diagrama de bloques funcional del LMD18200**

Un esquema general de la etapa de potencia que se utiliza en el sistema de levitación magnética se puede observar en la figura 1.6.

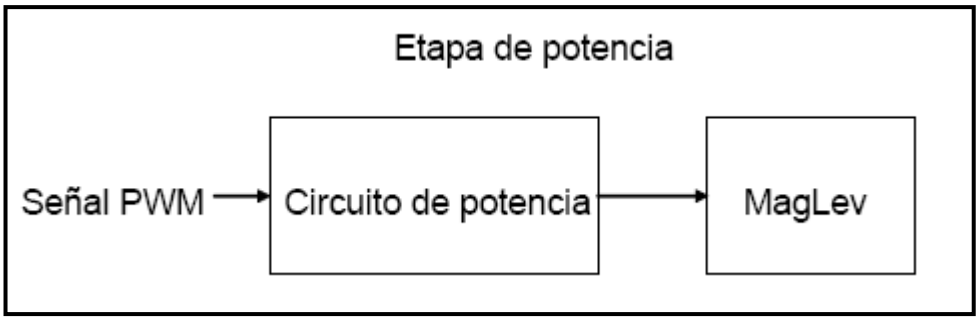



**Figura. 1.6. Diagrama de bloques de la etapa de potencia**

### 1.3.1.3 Sensor de posición

El sensor de posición es el dispositivo que se encarga en enviar los datos de posición de la esfera a través de la tarjeta de interfaz RT-DAC4. El sensor de posición genera UN haz de Luz que es receptado en el otro extremo y si la cantidad de Luz es 100% quiere decir que la esfera no se encuentra levitando. Tomando en cuenta esto, se debe realizar la calibración del sensor con la esfera en diferentes posiciones. Para tener registrado una tabla con los valores de voltaje que deben ser aplicados a cada posición.

A través de la calibración que se debe realizar como un paso previo antes de realizar la simulación, se puede mejorar la calibración del sensor. La curva muestra una relación no lineal entre la salida del sensor (voltaje) y la posición del objeto a levitar. Figura 1.7.

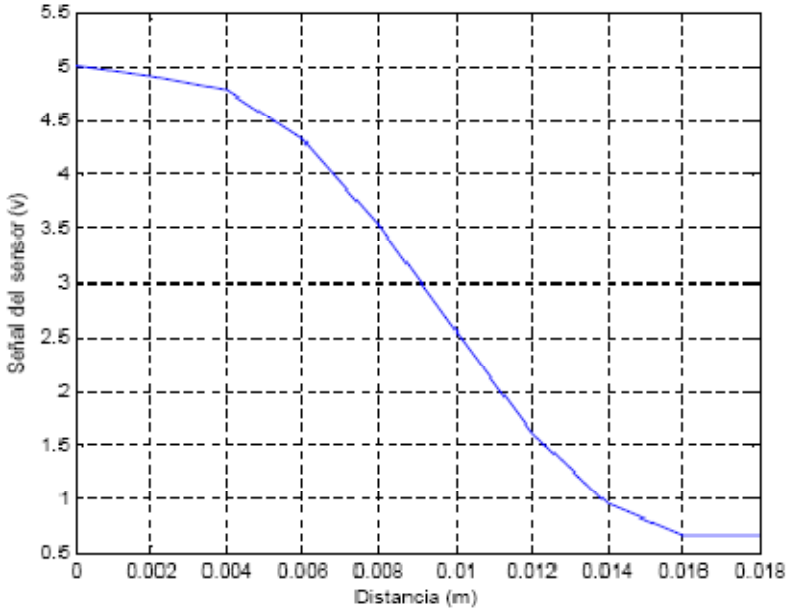



**Figura. 1.7. Relación no lineal entre Voltaje vs Posición**

Para realizar la simulación del sistema se debe empezar graficando u obteniendo los datos de voltaje vs posición del sistema. Figura 1.7.

Esta gráfica puede ser aproximada mediante un polinomio de un orden dado. Para este caso usamos un polinomio de quinto orden. El cual es lo que obtenemos a la salida del sistema total.

$$\mathbf{V = f(posición) = P(x)}$$

$$\mathbf{V = -25697073504,59^5 + 1245050011,25x^4 - 18773635,92x^3 + 79330,24x^2 - 150,21x + 5,015}$$

### 1.3.2 Descripción del software

El software del sistema MLS es versátil y permite diseñar nuevos sistemas de control con Simulink y también poder ser aplicados en tiempo real.

El software contiene herramientas de identificación, simulación en tiempo real que pueden ser utilizados para realizar un nuevo sistema de control. Figura 1.8.

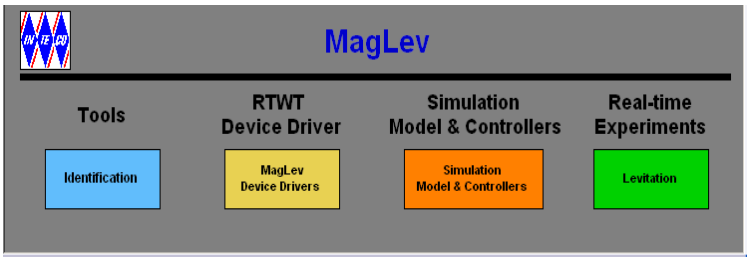



**Figura. 1.8. Descripción de software**

Las herramientas que posee el software del sistema de levitación son muy útiles en el diseño y simulación de sistemas de control, algunas de las características se explican a continuación.

#### 1.3.2.1 Identificación

La identificación es una herramienta que nos permite cambiar valores que por defecto vienen definidos en el sistema de levitación magnética. Con la identificación de sistemas es posible el procesamiento de datos, y consiste en variar cada entrada del sistema y registrar las variaciones producidas en las salidas asociadas. Los modelos obtenidos resultan satisfactorios para la mayoría

de las aplicaciones de control y proporcionan las relaciones dinámicas necesarias entre las entradas y salidas requeridas en las pruebas experimentales. En este caso la identificación puede mejorar la exactitud de las características estáticas y dinámicas del sistema MLS en el modelo matemático.

### 1.3.2.2 Simulación y modelos de controladores

El software posee controladores diseñados por el fabricante, los cuales se presentan a continuación y se los puede observar en la figura 1.9.:

- Lazo abierto (Open Loop)
- Proporcional Integral derivativo (PID)
- Controlador Optimo lineal cuadrático (LQ)


**Figura. 1.9. Controladores diseñados para sistema de levitación magnética**

Los controladores utilizan el modelo matemático de la planta que se encuentra diseñado previamente en el software por el fabricante. Figura. 1.10.

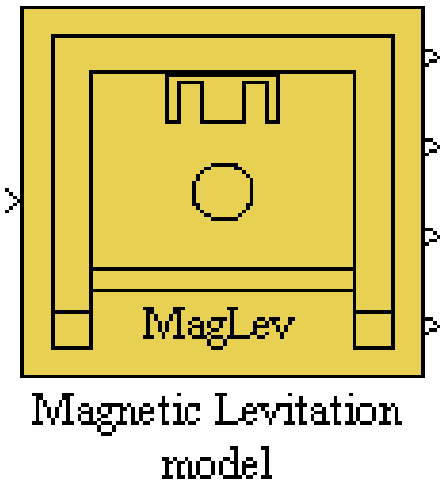



**Figura. 1.10. Modelo Matemático del sistema de levitación magnética**

El modelo matemático es necesario en este proyecto para realizar las simulaciones mas no para el diseño de los controladores inteligentes. Para realizar la simulación, el modelo de levitación magnética necesita parámetros o condiciones iniciales que son necesarios. Tabla. 1.2.

| PARAMETROS | VALOR | UNIDAD |
|---|---|---|
| $m$ | 0.04481 | $[kg]$ |
| $g$ | 9.81 | $\left[{}^{m}/_{s^2}\right]$ |
| $F_{em}$ | 0.9501 | $[N]$ |
| $F_{em}P1$ | 0.02697 | [H] |
| $\mathrm{F_{em}P2}$ | 0.004587 | [m] |

| $fi(x_1)$ | Función de posición | $[1/s]$ |
|---|---|---|
| $f_{iP1}$ | 0.0032 | $[m * s]$ |
| $f_{iP2}$ | 0.1146 | $[m]$ |
| $C_i$ | -0.043 | $[A]$ |
| $K_i$ | 2.6 | $[A]$ |
| $x_{3MIN}$ | 0.0050 | $[A]$ |
| $u_{MIN}$ | 0.00498 | |

**Tabla. 1.2. Datos medidos en la planta**

Los datos de la tabla 1.2., han sido tomados de trabajos previos realizados en el sistema de levitación magnética y son de ayuda para realizar la simulación de los controladores inteligentes.

Realizando una pequeña investigación sobre el modelo matemático se tiene dentro del bloque de simulación de SIMULINK, bloques que representan el modelo matemático de la planta. Figura 1.11.

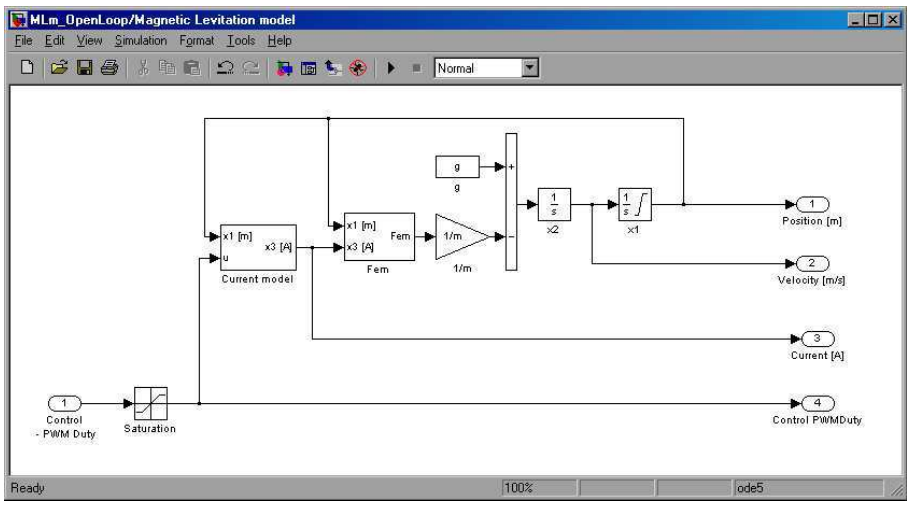



**Figura. 1.11. Interior del bloque de modelamiento del sistema de levitación magnética**

Para realizar la implementación de nuevos sistemas de control se debe tener en cuenta las tres señales, posición, velocidad y corriente.

El modelo de Simulink es coherente con el modelo matemático no lineal siguiente:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{F_{em}}{m} + g$$

$$\dot{x}_3 = \frac{1}{f_i(x_1)}\left(k_i u + c_i - x_3\right)$$

$$F_{em} = x_3^2 \frac{F_{emP1}}{F_{emP2}} \exp\left(-\frac{x_1}{F_{emP2}}\right)$$

$$f_i(x_1) = \frac{f_{iP1}}{f_{iP2}} \exp\left(-\frac{x_1}{f_{iP2}}\right)$$

Donde **Fem**. Es la fuerza electromagnética que se genera cuando se inyecta corriente a la bobina.

Donde:

$$x_1 \in [0, 0.016], \qquad x_2 \in \mathcal{R}, \qquad x_3 \in [x_{3MIN}, 2.38]$$

$$u \in [u_{MIN}, 1]$$

También hay que tomar en cuenta las curvas de la fuerza electromagnética con respecto a la corriente y la posición. Figura 1.12.

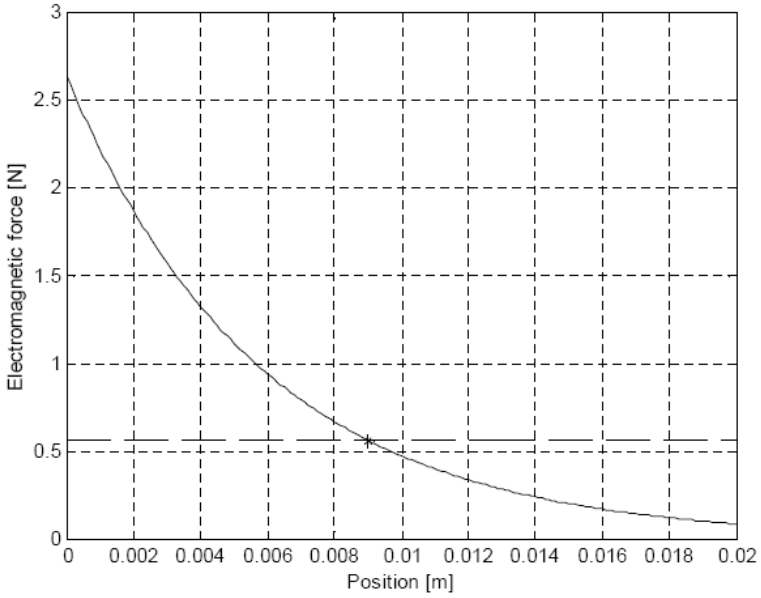



**Figura. 1.12. Fuerza Electromagnética en función de la posición**

La grafica anterior se presenta la fuerza electromagnética en función de la posición, y se puede apreciar que para una fuerza electromagnética constante de 0.6 [N] la posición de la esfera se ubica en 0.009 metros desde la bobina electromagnética.

Se muestra la variación de la fuerza electromagnética con la corriente para un valor de posición constante. Figura 1.13.

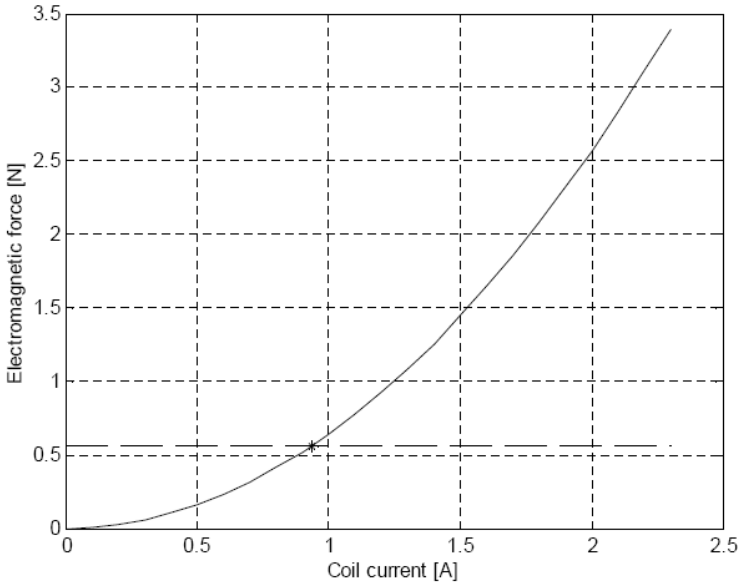



**Figura. 1.13. Fuerza Electromagnética en función de la corriente**

Se puede apreciar que para la fuerza electromagnética de 0.6N se tiene una corriente de 0.9 [A] aproximadamente. La fuerza electromagnética depende de dos variables que son la distancia desde la bobina y la corriente que se aplica a la bobina, como se aprecia claramente en las dos graficas anteriores.

La levitación de la esfera requiere el equilibrio de la fuerza de gravedad con la fuerza electromagnética que depende de dos variables, la posición y la corriente.

Esta idea muestra la complejidad del control puesto que en cada posición debe proveer la corriente exacta para mantener el equilibrio de las fuerzas.

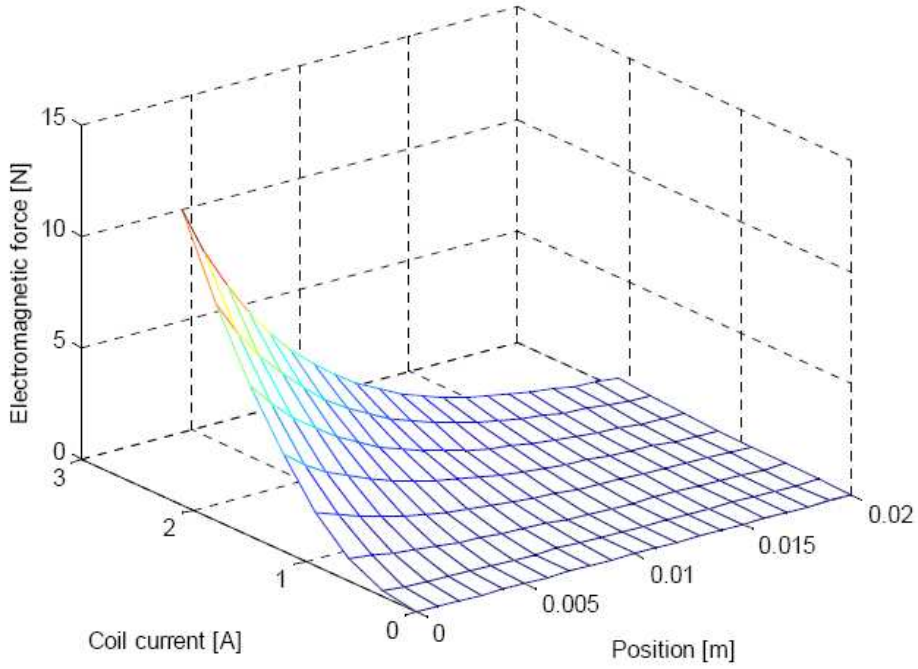



**Figura. 1.14. Posición, corriente y fuerza electromagnética**

Se puede observar en una sola gráfica la fuerza electromagnética, posición y corriente. Figura 1.14.

En las ecuaciones se observa también que la posición produce una variación en la corriente lo cual hace aún más complejo el objetivo de equipibrar las fuerzas. La relación entre la variación de corriente y la posición está definido por una función fí que se presenta en la figura 1.15.

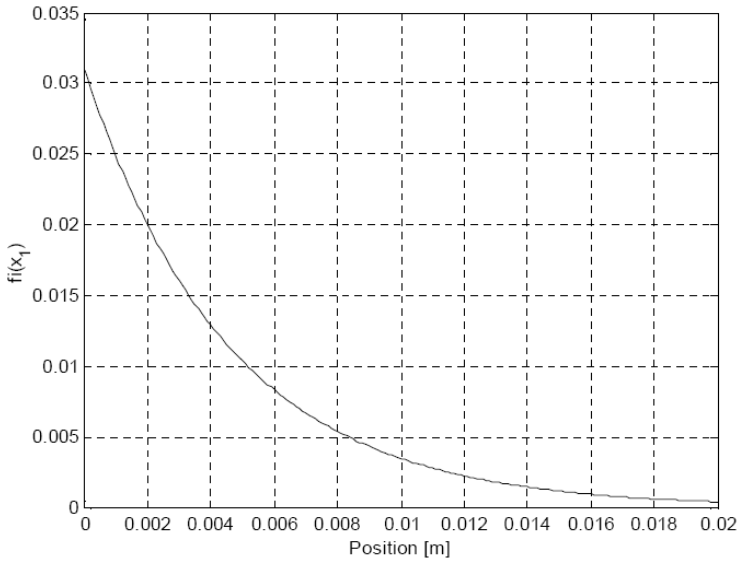



**Figura. 1.15. Función de la posición**

De esta manera se tiene definido el funcionamiento del sistema de levitación magnética con el que se procede a realizar el estudio de la teoría necesaria en cuanto se refiere a teoría de control inteligente de lógica difusa y redes neuronales.

# CAPÍTULO II

# TEORIA DE CONTROL INTELIGENTE

# CAPÍTULO II

# TEORIA DE CONTROL INTELIGENTE

El control inteligente representa actualmente una novedosa e importante rama de control automático. Los procedimientos convencionales no se sustituyen, sino que se complementan de forma considerable en función del campo de aplicación.

En los últimos años los sistemas inteligentes se han consolidando como una herramienta útil para tratar y modelar sistemas complejos y no lineales, especialmente en áreas como el control.

La teoría de control inteligente se ha convertido en una herramienta fundamental para abordar problemas complejos incluyendo el área de control automático. El control automático ha desempeñado una función vital en el avance de la Ingeniería y de la Ciencia. Además de su extrema importancia en los sistemas de vehículos espaciales, guiado de misiles, robótica etc., el control automático se ha vuelto una parte importante e integral de los procesos modernos industriales y de manufactura.

La teoría de control inteligente tiene varias ramas, entre ellas se encuentran la Lógica Difusa detallado en el punto 2.1., y Redes Neuronales detallado en el punto 2.2.

## 2.1. LOGICA DIFUSA

La Lógica Difusa es una metodología que proporciona una manera simple y elegante de obtener una conclusión a partir de información de entrada, ambigua, imprecisa, o incompleta.

En general la lógica difusa modela la forma como una persona toma decisiones basada en información visual o experimental. La lógica difusa se basa en lo relativo de lo observado o lo experimentado.

La lógica difusa utiliza expresiones que no son ni totalmente ciertas ni completamente falsas, es decir lógica difusa aplica conceptos que pueden tomar un valor cualesquiera de veracidad dentro de un conjunto de valores que oscilan entre dos extremos, la verdad absoluta y la falsedad total. Recalcando esta idea se puede tener cosas que no son blancas o negras, sino que existen infinitos matices de grises.

Conviene recalcar que lo que es difuso, impreciso o vago, no es la lógica en sí, sino el objeto que estudia. La lógica difusa se adapta mejor al mundo real en el que vivimos, e incluso puede comprender y funcionar con nuestras expresiones, del tipo "hace mucho calor", "no es muy alto", "el ritmo del corazón está un poco acelerado", etc. La clave de esta adaptación al lenguaje difuso, se basa en comprender los cuantificadores de nuestro lenguaje ("mucho", "muy" y "un poco").

La lógica difusa permite tratar información imprecisa, como estatura alta, media o baja de una persona, tal como se observa en la figura. 2., así, por ejemplo, se puede considerar a una persona que mida 2 metros, claramente como una persona alta, si previamente se ha tomado el valor de una persona de estatura baja y se ha establecido en 1 metro.

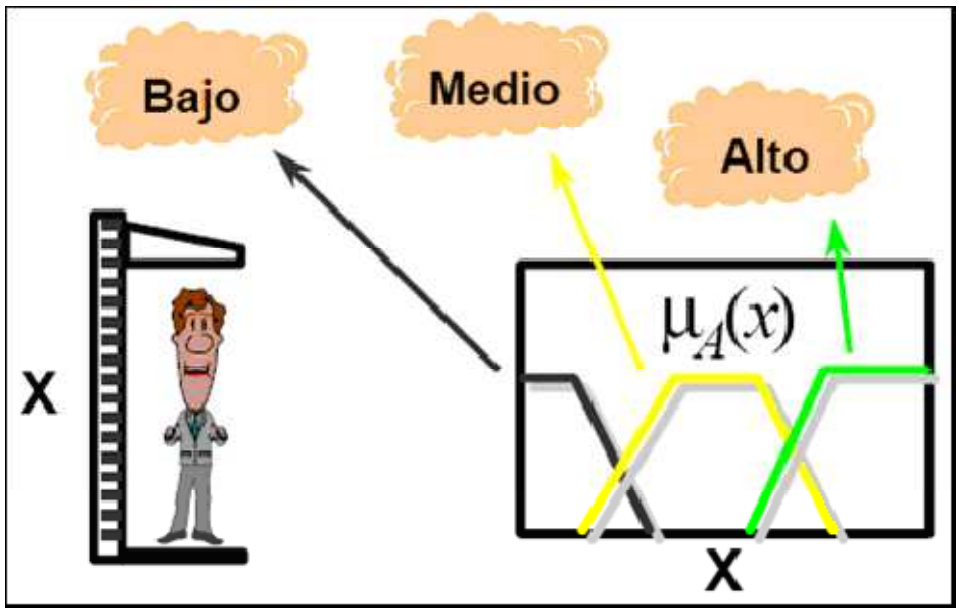



**Figura. 2. Visión de la Lógica difusa**

### 2. 1.1 Conjuntos difusos

Una buena estrategia para presentar la teoría de Conjuntos Difusos, consiste en recordar algunos aspectos de la teoría de conjuntos clásicos, y a partir de allí hacer una extensión a los conjuntos difusos.

Un conjunto clásico se define como una colección de elementos que existen dentro de un Universo. Así se establece que cada uno de los elementos del Universo pertenecen o no a un determinado conjunto. Por lo tanto, cada conjunto

puede definirse completamente por una función de pertenencia, que opera sobre los elementos del Universo, y que le asigna un valor de 1 si el elemento pertenece al conjunto, y de 0 si no pertenece.

Ahora bien, un Conjunto Difuso se define de forma similar, con una diferencia conceptual importante: un elemento puede pertenecer parcialmente a un conjunto y parcialmente a otro. Así mismo el Universo de los conjuntos clásicos, toma el nombre de Universo de Discurso, ya que los conjuntos difusos involucran una serie de variables lingüísticas, como se verán más adelante.

De esta forma, un conjunto difuso definido sobre un universo, puede definirse matemáticamente al asignar a cada posible individuo que existe en el universo un valor que representa su grado de pertenencia o membrecía en el conjunto difuso.

Cada elemento de un conjunto difuso presenta un grado de pertenencia a un conjunto difuso que puede tomar cualquier valor entre 0 y 1.

Las primeras diferencias que se hacen evidentes entre los Conjuntos Clásicos y los Conjuntos Difusos son las siguientes:

- La función de pertenencia asociada a los conjuntos clásicos sólo puede tener dos valores: 1 ó 0, mientras que en los conjuntos difusos puede tener cualquier valor entre 0 y 1.

- Un elemento puede pertenecer (parcialmente) a un conjunto difuso y simultáneamente pertenecer (parcialmente) al complemento de dicho conjunto.

❖ Las fronteras de un conjunto clásico son exactas, en tanto que las de un conjunto difuso son, precisamente, difusas, ya que existen elementos en las fronteras mismas, y estos elementos están a la vez dentro y fuera del conjunto.

Se puede considerar que la lógica clásica es un caso límite de la lógica difusa. Así pues los conjuntos difusos pueden ser considerados como una generalización de los conjuntos clásicos, como se representa conceptualmente en la figura. 2.1.

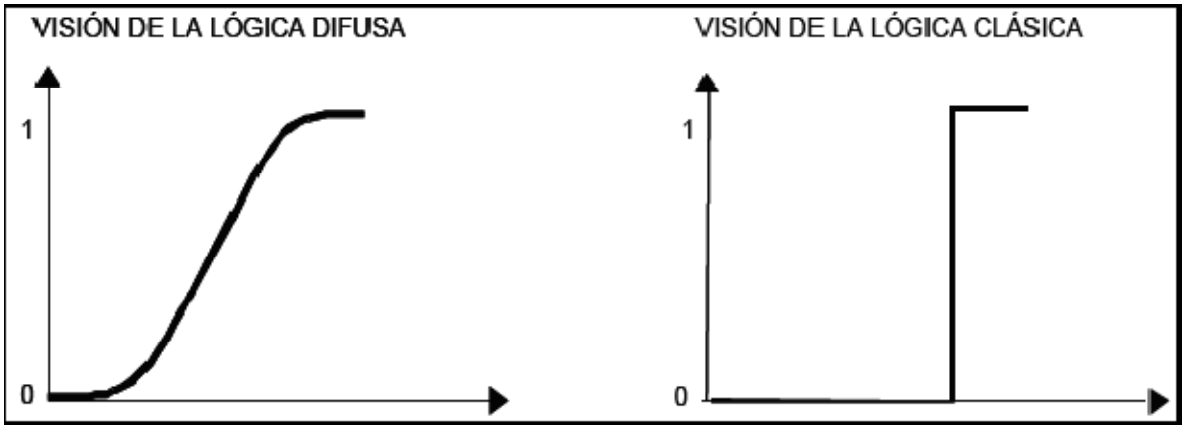



**Figura. 2.1. Funciones de pertenencia**

Ahora bien, ¿qué sentido puede tener el pertenecer parcialmente a un conjunto?

En muchos casos puede tener más sentido que pertenecer totalmente a un conjunto; así, se tienen algunos ejemplos:

*Ejemplo 1*:

Supóngase que se desea definir el conjunto de los estudiantes de la carrera de Ingeniería Electrónica de la Escuela Politécnica del Ejército que están cursando el quinto semestre. ¿Cómo clasificar a un estudiante que cursa dos materias de cuarto semestre, tres de quinto y una de sexto? y a ¿otro que toma una materia de quinto semestre, y cinco de sexto?

Evidentemente ambos son en parte miembros del conjunto Estudiantes de quinto semestre, pero sólo lo son parcialmente.

*Ejemplo 2:*

Supóngase que se desea clasificar a los miembros de un salón de clase según su estatura en tres conjuntos, Bajos, Medianos y Altos. Podría plantearse que un individuo es Bajo, si se tiene una estatura inferior a, por ejemplo, 160 cm, que un individuo es Mediano, si tiene una estatura superior o igual a 160 cm e inferior a 180 cm, y que un individuo es Alto, si tiene una estatura superior o igual a 180 cm, con lo que se lograría una clasificación en conjuntos clásicos. Sin embargo, ¿qué tan grande es la diferencia que existe entre dos estudiantes de la clase, uno con estatura de 179 cm y otro de 180 cm? Ese centímetro de diferencia quizás no represente en la práctica algo significativo, y sin embargo los dos estudiantes han quedado rotulados con etiquetas distintas: uno es Mediano y el otro es Alto. Si se optase por efectuar la misma clasificación con conjuntos difusos estos cambios abruptos se evitarían, debidos a que las fronteras entre los conjuntos permitirían cambios graduales en la clasificación. Figura. 2.2.

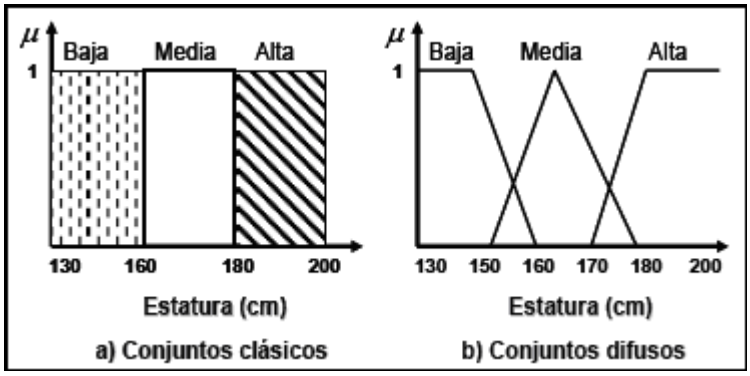



**Figura 2.2. Funciones de pertenencia para los conjuntos del ejemplo 2.**

En la Figura. 2.2. a) se tiene la representación del ejemplo 2 utilizando conjuntos clásicos. En esta representación una persona que mida 179 cm es considerada de estatura media y en cambio una persona con 180 ya no, esto no corresponde con la realidad.

En la Figura 2.2. b) se muestra cómo podría hacerse tal clasificación con conjuntos difusos. Se observa que existe un grado de pertenencia a los conjuntos difusos. Así una persona que tenga una estatura de 172 cm, pertenecería en un 40% al conjunto de personas con estatura media y en un 10% al de estatura alta.

### 2.1.2 Funciones de membrecía

La teoría de conjuntos difusos contempla la pertenencia parcial de un elemento a un conjunto.

Este grado de pertenencia se define mediante una función característica asociada al conjunto difuso, llamada Función de Membrecía. Así, si se define un conjunto difuso A con x elementos: la función de membrecía quedaría definida por $U_A(x)$, donde para cada valor que pueda tomar un elemento o variable de entrada

x, la función de membrecía $U_A(x)$ proporciona el grado de pertenencia de este valor de x al conjunto difuso A.

Muchos conceptos de teoría clásica de conjuntos se pueden hacer extensivos a los conjuntos difusos, otros son exclusivos e inherentes a la teoría de conjuntos difusos. Algunos de los más utilizados son los siguientes:

- El soporte de un conjunto difuso A en el universo de discurso U es un conjunto “crisp” (numérico) que contiene todos los elementos de U que tienen un valor de pertenencia distinto de cero en A, esto es:

$$sop(x) = \{x \ \in U / \mu_A(x) > 0\}$$

Si el soporte de un conjunto difuso no contiene ningún elemento, se tiene un conjunto difuso vacío. Si el soporte de un conjunto difuso es un solo punto, se tiene lo que se conoce como “singleton” difuso.

- El punto de cruce de un conjunto difuso es el punto de U cuyo valor de pertenencia al conjunto es igual a 0.5.

- Dos conjuntos difusos A y B son iguales si y sólo si sus funciones de membrecía $\mu_A(x) y\ \mu_B(x)$ son iguales.

- El conjunto difuso B contiene al conjunto difuso A, esto es $A \subset B$, si y sólo si $\mu_A(x) \leq \ \mu_B(x)$ para todo $x \in U$.

La función de membrecía proporciona una medida del grado de pertenencia de un elemento de U con el conjunto difuso. La forma de la función de membrecía utilizada, depende del criterio aplicado en la resolución de cada problema y

variará en función del punto de vista del usuario. La única condición que debe cumplir una función de membrecía es que tome valores entre 0 y 1, con continuidad.

Las funciones características más comúnmente utilizadas por su simplicidad matemática y su manejabilidad son: triangular, trapezoidal, gaussiana, sigmoidal, gama, pi, campana, etc., como se visualiza en la figura. 2.3; siendo la triangular la forma más utilizada para una función de membrecía.

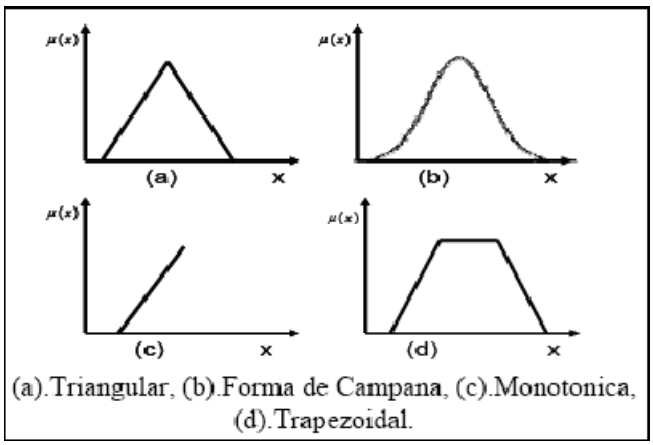



**Figura. 2.3. Funciones de membrecía más utilizadas.**

Conceptualmente existen dos aproximaciones para determinar la función característica asociada a un conjunto: la primera aproximación está basada en el conocimiento humano de los expertos y la segunda aproximación es utilizar la colección de datos para diseñar la función.

El número de funciones características asociadas a una misma variable es elegido por el experto: a mayor número de funciones características tendremos mayor resolución, pero también mayor complejidad computacional.

Además, estas funciones pueden estar solapadas o no, el hecho de estar solapadas pone de manifiesto un aspecto clave de la lógica difusa: una variable puede pertenecer con diferentes grados a varios conjuntos difusos a la vez, es decir, “el vaso puede estar medio lleno y medio vacío a la vez”.

De esta manera para el ejemplo de las estaturas, se puede escoger dos alternativas adicionales en la forma de las funciones de membrecía, tal como se puede observar en la figura. 2.4.

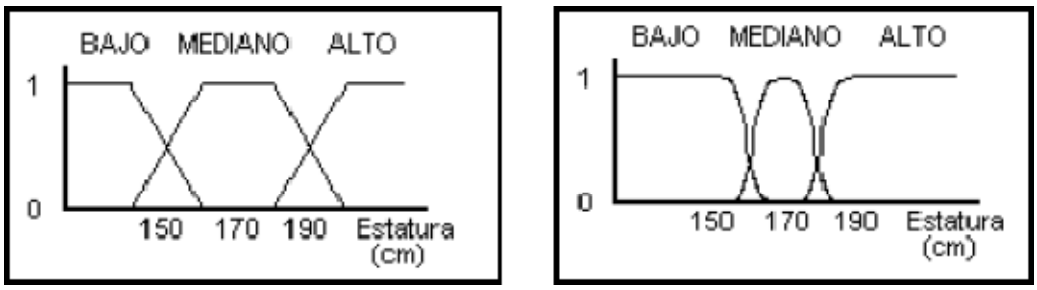



**Figura. 2.4. Representaciones alternativas para el ejemplo de las estaturas.**

### 2.1.3 Operaciones entre conjuntos difusos

Las tres operaciones básicas entre conjuntos clásicos: Unión, Intersección y Complemento, se definen también para los conjuntos difusos, intentando mantener el significado de tales operaciones. La definición de estas operaciones se hace empleando el concepto de función de pertenencia de los conjuntos.

*Complemento:* Para un conjunto difuso *A* definido sobre un Universo de discurso *U*, y cuya función de pertenencia es $\mu_A(x)$; el resultado de efectuar la operación de Complemento (que en lógica binaria es el equivalente de la operación NOT), es un nuevo conjunto difuso *A’* definido sobre el mismo Universo, y con función de pertenencia $\mu_{A'}(x)$, dada por:

$$\mu_{A'}(x) = 1 - \mu_A(x)$$

*Unión:* Para dos conjuntos difusos *A* y *B* definidos sobre el mismo Universo de discurso, y con funciones de pertenencia $\mu_A(x)\ y\ \mu_B(x)$; el resultado de efectuar la operación de unión entre estos dos conjuntos (que en lógica binaria es el equivalente de una operación OR), es un nuevo conjunto difuso *AUB* definido sobre el mismo universo, y con función de pertenencia $\mu_{A\cup B}(x)$, dada por:

$$\mu_{A\cup B}(x) = max\{\mu_A(x), \mu_B(x)\}$$

*Intersección:* Para dos conjuntos difusos *A* y *B* definidos sobre el mismo Universo de discurso, y con funciones de pertenencia $\mu_A(x)\ y\ \mu_B(x)$; el resultado de efectuar la operación de intersección entre estos dos conjuntos (que en lógica binaria es el equivalente de una operación AND), es un nuevo conjunto difuso *A* ∩ *B* definido sobre el mismo universo, y con función de pertenencia $\mu_{A\cap B}(x)$, dada por:

$$\mu_{A\cap B}(x) = min\{\mu_A(x), \mu_B(x)\}$$

Estas tres operaciones definidas para conjuntos difusos cumplen, al igual que en la teoría clásica de conjuntos, asociatividad, conmutatividad y distributividad así como las leyes de Morgan.

Sin embargo hay que destacar que existen dos leyes fundamentales de la teoría clásica de conjuntos como son el principio de contradicción: $A \cup \bar{A} = U$ y el

principio de exclusión $A \cap \bar{A} = \emptyset$ que no se cumplen en la teoría de conjuntos difusos.

La representación que se tiene entre la lógica clásica y la lógica difusa se presenta en la figura 2.5.

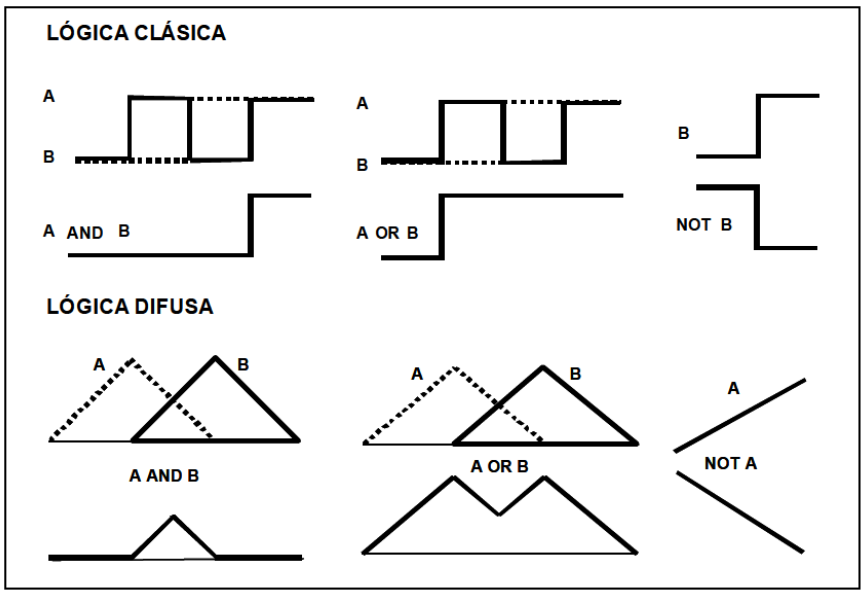



**Figura. 2.5. Actuación de los operadores OR; AND y NOT según la lógica clásica y la lógica difusa**

### 2.1.4 Tipos de operadores

Como es bien sabido se puede establecer un isomorfismo entre la teoría de conjuntos, la lógica proposicional y el álgebra booleana que garantiza que cada teorema enunciado en una de ellas tiene un homólogo en las otras dos. La existencia de estos isomorfismos permitirá traducir las reglas difusas a relaciones entre conjuntos difusos y éstas a términos de operadores algebraicos con los que se puede trabajar. En la Tabla 2.1 se muestra la correspondencia de algunos operadores.

| Teoría de Conjuntos | Algebra Booleana | Lógica Tradicional |
|---|---|---|
| Intersección | Conjunción | AND |
| Unión | Disyunción | OR |
| Complemento | Negación | NOT |

**Tabla. 2.1. Correspondencia entre operadores**

Ahora bien, el razonamiento lógico consiste en la combinación de proposiciones para producir nuevas proposiciones; así, la combinación de las proposiciones *"X es A"* y *"Y es B"* mediante el operador AND da como resultado la proposición *"X es A AND Y es B"*.

En lógica difusa una proposición puede representarse por un conjunto difuso: *"X es A" donde X* corresponde a un conjunto *A* con función de pertenencia $\mu_A(x)$, mientras que *"Y es B" donde Y* corresponde a un conjunto *B* con función de pertenencia $\mu_B(x)$, y la combinación de estas dos proposiciones con el operador AND, es decir la proposición *"X es A AND Y es B"* corresponde a un nuevo conjunto difuso con función de pertenencia $\mu_{A\ AND\ B}(x,y) = min\{\mu_A(x), \mu_B(y)\}$.

### 2.1.5 Implicación difusa

Un análisis especial debe hacerse con el operador lógico de implicación, que combina dos proposiciones con la expresión *SI... Entonces... (If... Then...)*, y que es el fundamento de las inferencias realizadas en sistemas de lógica difusa.

El operador lógico de implicación, permite encontrar un camino matemático para evaluar proposiciones como las siguientes: *"Si las vibraciones son altas Entonces el rodamiento está desgastado"*, o *"Si los ingresos del cliente son bajos*

*Entonces su capacidad de endeudamiento es poca"*. A la relación entre dos proposiciones a través del operador lógico de implicación, se las llama también reglas, las mismas que servirán en lo posterior para definir la base de reglas de los sistemas de control difuso.

### 2.1.6 Inferencia difusa

La inferencia difusa es el conjunto de proposiciones IF-THEN que modelan el problema que se quiere resolver. Una regla difusa simple tiene la forma:

"Si u es A entonces v es B"

Dónde A y B son conjuntos difusos definidos en los rangos de "ü" y "v" respectivamente. Una regla expresa un tipo de relación entre conjuntos A y B cuya función característica seria $U_{A\rightarrow B}(x, y)$ y representa lo que conocemos como implicación lógica. La elección apropiada de esta función está sujeta a las reglas de la lógica proposicional.

### 2.1.7 Variables y valores lingüísticos

Se denomina variable lingüística a aquella que puede tomar por valor términos del lenguaje natural, como: temperatura ambiente, estatura, velocidad, etc., lo que se había definido dentro de conjuntos difusos como universo de discurso. Es decir, una variable lingüística nos permite especificar la incertidumbre o subjetividad de un determinado concepto.

Una variable lingüística tiene, entre otras cosas, una colección de atributos que puede adquirir la variable, y cada atributo está representado por un conjunto difuso. Así, retomando el ejemplo de las estaturas, la variable *Estatura* tendría tres atributos, *Bajo, Mediano y Alto*, y cada uno de estos atributos estaría representado por un conjunto difuso respectivo. Estos atributos reciben el nombre de *Valores Lingüísticos*. Estos valores lingüísticos vienen a constituirse en las funciones de membrecía de una variable lingüística.

El objeto principal de estos conceptos es expresar de manera formal el hecho de que pueden asignarse como valores de una variable, palabras tomadas del lenguaje natural.

Debido a que un Sistema de Lógica Difusa puede, en general, tener varias entradas y varias salidas, la forma genérica de las reglas presentes en la Base de Reglas es la siguiente:

*IF X1 es A1 AND X2 es A2 AND ... AND Xm es Am THEN Y1 es B1 AND Y2 es B2 AND... AND Yn es Bn*

En estas reglas, *A1, A2,..., Am, B1, B2,..., Bn* son los valores Lingüísticos de las Variables Lingüísticas respectivas.

### 2.1.8 Aplicación de la lógica difusa en sistemas de control

En las teorías tradicionales se obliga a que las representaciones del mundo real que se realizan encajen dentro de modelos de simulación y sean precisos, tomando la imprecisión como un factor de distorsión.

Tomando en cuenta lo mencionado se define el sistema de lógica difuso en el siguiente esquema.

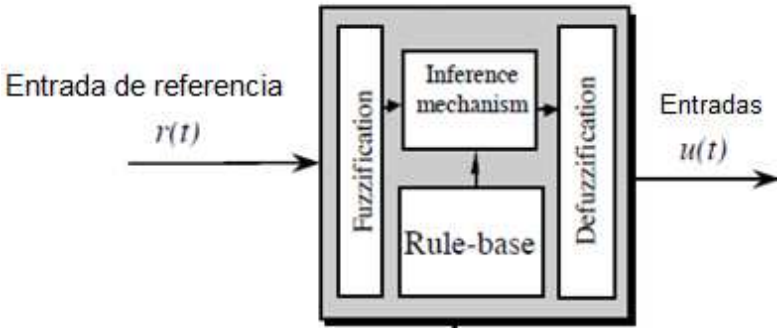



**Figura. 2.6. Arquitectura del Sistemas Difuso**

El sistema difuso provee una metodología formal para la representación, la manipulación, y aplicación de conocimiento heurístico de un ser humano acerca de cómo controlar un sistema. En esta sección se trata de proporcionar una filosofía de la forma de abordar el diseño de controladores difusos.

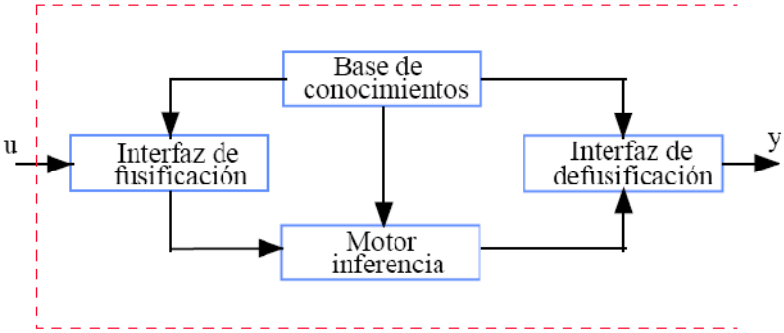



**Figura. 2.7. Arquitectura Interna del Sistema Difuso**

Como se puede observar en la figura. 2.7., el sistema difuso cuenta con cuatro bloques:

- Etapa de Fusificación
- Base de Reglas
- Maquina de Inferencia
- Etapa de Defusificación

Se detallan cada uno de ellos a continuación:

### 2.1.9 Etapa de Fusificación

En la etapa de Fusificación se transforma las variables de entrada del modelo en variables difusas, donde a cada variable de entrada se le asigna un grado de pertenencia a cada uno de los conjuntos difusos que se han considerado, mediante las funciones de membrecía asociadas a estos conjuntos difusos.

Para esta interfaz se deben tener definidos los rangos de variación de las variables de entrada y los conjuntos difusos asociados con sus respectivas funciones de pertenencia.

### 2.1.10 Base de reglas

Contiene las reglas lingüísticas del control y la información referente a las funciones de pertenencia de los conjuntos difusos. Como ya se ha visto en la inferencia difusa, las reglas tienen la forma

*Si u1 es A y u2 es B entonces y es C*

Donde A, B son los conjuntos difusos de las variables de entrada u1 y u2, mientras C es el de la variable de salida *y*.

Existen varias formas de expresar las reglas, entre las que se destacan son:

- La experiencia de expertos y el conocimiento de ingeniería de control. La base de reglas se determina a partir de entrevistas con el operador o a través del conocimiento de la dinámica del proceso.

- La modelación del proceso. Los parámetros de la base de conocimiento se obtienen a partir de datos de entrada y salida del proceso.

### 2.1.11 Máquina de Inferencia o motor de inferencia

El motor de inferencia usa los principios de la lógica difusa acerca de la inferencia difusa, para realizar un mapeo de los conjuntos difusos de entrada a los conjuntos difusos de salida.

Cada regla es interpretada como una implicación difusa. Es decir el bloque de inferencia es aquel en el que se realiza la “traducción matemática” de las reglas difusas. Las reglas difusas modelan el sistema para poder trabajar con ellas y extraer un resultado.

La máquina de inferencia realiza la tarea de calcular las variables de salida a partir de las variables de entrada, mediante las reglas del controlador y la inferencia difusa, entregando conjuntos difusos de salida.

La secuencia de cálculos que realiza el motor de inferencia incluye:

- Determinar el grado de cumplimiento de cada regla a partir de los grados de pertenencia de las variables de entrada obtenidos en la etapa de Fusificación. Debido a que las premisas de las reglas están unidas por operadores, definidas como la intersección de conjuntos difusos.

- Como se sabe, para cada regla se tiene una consecuencia, que tiene asociada una función de pertenencia. Por lo tanto a la salida se tendrá un conjunto difuso de salida representado por su respectiva función de pertenencia.

- Para evaluar el conjunto total de reglas, se unen los conjuntos difusos resultantes de cada regla, generándose un conjunto de salida.

De esta forma, se obtiene una salida difusa del controlador, con una función de pertenencia.

### 2.1.12 Etapa de Defusificación

En este bloque a partir del conjunto difuso obtenido en el bloque de inferencia y mediante métodos matemáticos de defusificación, se obtiene el valor concreto de la variable de salida. Este elemento provee salidas de valor numérico

y determinanticas a partir de los conjuntos difusos obtenidos como resultado de la inferencia.

Existen diferentes métodos de defusificación, algunos de los cuales se describen a continuación:

- ❖ Método del máximo. La salida corresponde al valor para el cual la función de pertenencia alcanza su máximo.

- ❖ Media del máximo. La salida es el promedio entre los elementos del conjunto que tienen un grado de pertenencia máximo.

- ❖ Centro de área. Genera como salida el valor correspondiente al centro de gravedad de la función de pertenencia del conjunto de salida.

### 2.1.13 Ejemplo de Péndulo invertido

Como ejemplo se procede a realizar el diseño del controlador difuso para el péndulo invertido, el ejemplo trata de controlar la posición del péndulo como la muestra la figura. 2.8.

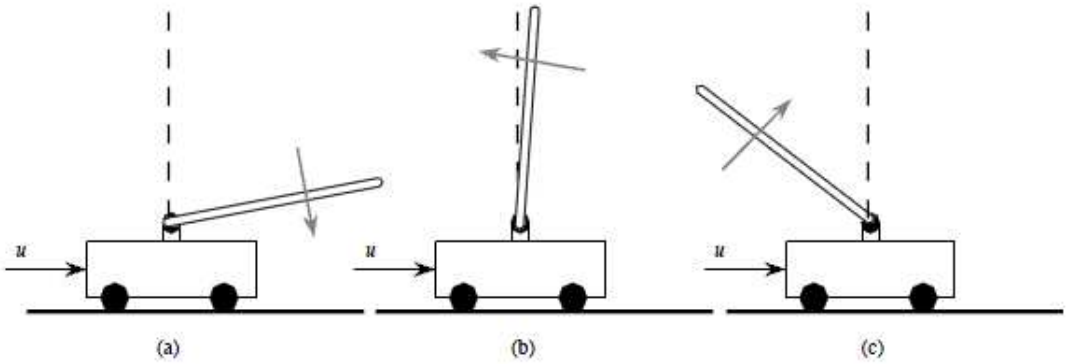



**Figura. 2.8. Sistema del péndulo invertido**

El controlador difuso es diseñado para automatizar como un humano experto debería controlar satisfactoriamente al sistema. Primero el experto o el diseñador debe tomar en cuenta que información va a ser usada como entradas para hacer el proceso de decisión, entonces el experto dice que se va a usar el error y la derivada del error;

$$e(t) = r(t) - y(t)$$

Y

$$\frac{d}{dt}e(t)$$

Estas variables serán las entradas para el controlador. Existen muchas otras opciones como por ejemplo la integral del error que también puede ser utilizada.

Luego se debe escoger la variable controlada sobre el péndulo invertido la cual es la fuerza que mueve al carro por lo que la elección es sencilla.

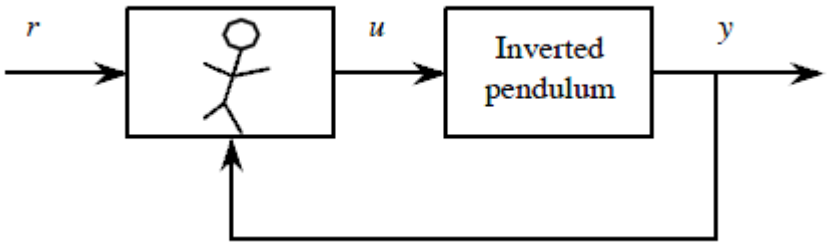



**Figura. 2.9. Humano controlando el péndulo invertido sobre el carro**

Se debe tomar en cuenta e identificar las variables que serán utilizadas como entradas o salidas para de esta manera definirlas como variables lingüísticas del sistema lógico difuso.

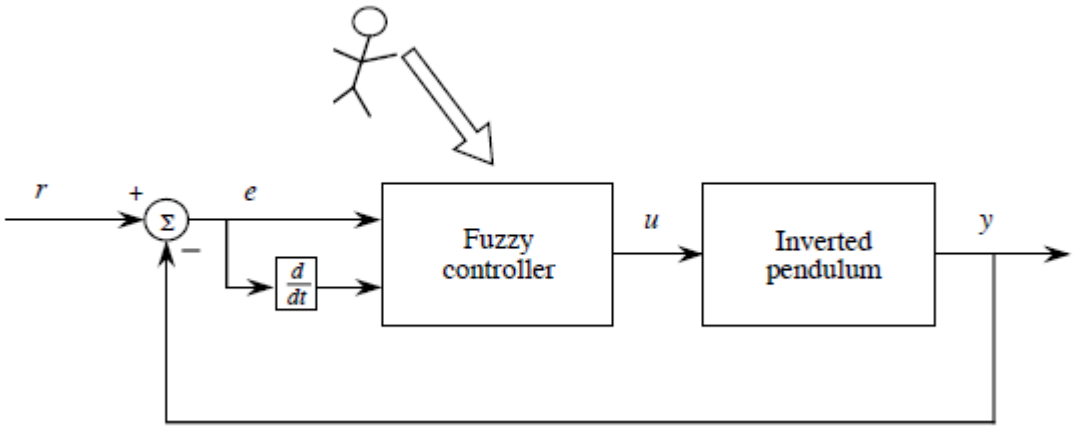



**Figura. 2.10. Controlador difuso para el péndulo invertido**

Como se puede apreciar en la figura 2.10, el controlador difuso para el péndulo invertido se lo realiza conjuntamente con el conocimiento del experto además se utiliza como entradas las variables error y derivada del error las cuales se presentaron anteriormente.

Ahora se procede a dar una descripción lingüística de cómo debe funcionar el péndulo con lógica difusa y cargar esta descripción dentro del controlador difuso.

### 2.1.13.1 Desarrollo de péndulo invertido

Continuando con el ejemplo del péndulo invertido ahora se procede a realizar la descripción lingüística del sistema. Se puede presentar en partes, las

cuales se las puede definir como variables lingüísticas que describen a las entradas y salidas del controlador. Para el péndulo invertido son las siguientes:

"error" describe *e(t)*

"cambio de error" describe $\frac{d}{dt}e(t)$

"fuerza" describe u (t)

De esta manera se va construyendo las variables lingüísticas para el controlador difuso.

Por ejemplo e(t) puede tomar valores lingüísticos durante el tiempo es decir: 0.1 en t=2 seg (e(2)=0.1), las variables lingüísticas toman valores lingüísticos sobre el tiempo que cambian dinámicamente.

Supóngase que el ejemplo del péndulo y sus variables "error", "cambio de error" y "fuerza" toman los siguientes valores:

"grande negativo"

"pequeño negativo"

"cero"

"pequeño positivo"

"grande positivo"

Estas descripciones se las define como los valores lingüísticos que dan un valor a las variables lingüísticas, y se las puede representar numéricamente de la siguiente manera;

"-2" que representa "grande negativo"

"-1" que representa "pequeño negativo"

"0" que representa "cero"

"1" que representa "pequeño positivo"

"2" que representa "grande positivo"

Las descripciones son cortas y muy bien declaras para los valores lingüísticos. De esta manera es que se puede representar valores numéricos a valores lingüísticos.

Las variables lingüísticas y valores lingüísticos proveen un lenguaje para expresar las ideas del experto de cómo debe realizar el control y las decisiones sobre el proceso para estabilizarlo.

Para el péndulo invertido cada una de las siguientes sentencias lingüísticas califica una diferente configuración del péndulo, como referencia se toma la figura. 2.11.

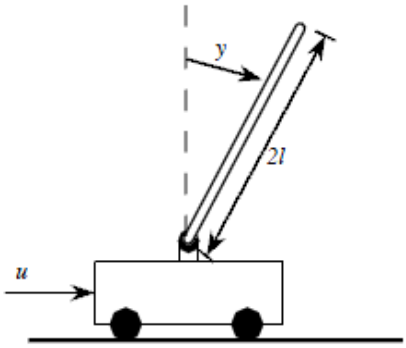



**Figura. 2.11. Péndulo invertido**

- La sentencia “error es grande positivo” puede representar la situación donde el péndulo tiene un ángulo a la izquierda con respecto a la línea vertical de referencia.

- La sentencia “error es pequeño negativo” puede representar la situación donde el péndulo tiene un ángulo pequeño a la derecha con respecto a la vertical como referencia pero no es lo suficientemente cerrado y cerca del cero para determinar que esta sentencia es cero respecto a la vertical, no es lo suficiente para justificar la cuantificación que es “grande negativo”.

- La sentencia “error es cero” puede representar la situación donde el péndulo se encuentra muy cerca de la vertical (una cuantificación lingüística no es precisa, esto quiere decir que se acepta valores que estén alrededor de cero e(t)=0 es decir se puede asumir que la cuantificación de la sentencia es mejor definida que tener “grande positivo” o “pequeño negativo”).

- La sentencia “error es grande positivo y cambio de error es pequeño positivo” puede representar la situación donde el péndulo esta en el lado izquierdo respecto a la vertical de referencia y desde $\frac{d}{dt}y < 0$, el péndulo está moviéndose desde la izquierda a la derecha en función de las manillas del reloj.

La importancia de identificar y cuantificar las sentencias sobre cómo funciona el sistema para el diseñador debe ser muy importante, que pueda

entenderlas ayudara a establecer y plasmar el conocimiento del experto dentro del controlador.

Lo que sigue a continuación es expresar e ir definiendo la base de reglas del controlador una vez que se tiene bien en claro el funcionamiento del péndulo. Para el péndulo se presenta la situación de tres posiciones con las siguientes reglas. Figura. 2.12.

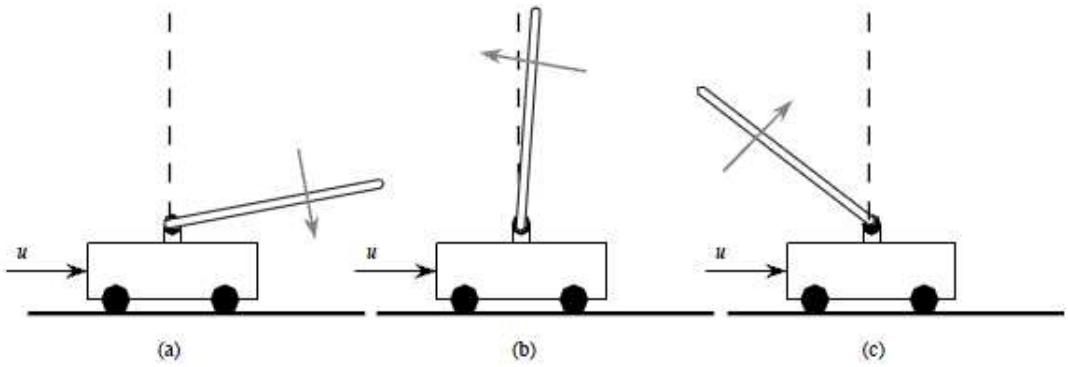



**Figura. 2.12. Posiciones del péndulo**

1. **Si** el error es grande negativo **y** el cambio de error es grande negativo **entonces** la fuerza es grande positiva.

   Esta regla cuantifica la situación de la Figura. 2.12. (a) donde el péndulo tiene un ángulo grande positivo respecto a la vertical de referencia y el carro se está moviendo en sentido de las manecillas del reloj: está claramente que se puede aplicar una fuerza grande positiva (hacia a la derecha por “u”) para que el péndulo se pueda ubicar en la posición correcta.

2. **Si** el error es cero **y** el cambio de error es pequeño positivo **entonces** la fuerza es pequeña negativa.

Esta regla cuantifica la situación de la figura 2.12. (b) donde el ángulo del péndulo se encuentra cerca de cero respecto de la vertical de referencia (una cuantificación lingüística de cero no implica que e (t)=0 sea exactamente cero).

3. **Si** el error es grande positivo y el cambio de error es pequeño negativo **entonces** la fuerza es pequeña negativa.

   Esta regla cuantifica la situación de la figura 2.12. (c) donde el péndulo está lejos de la vertical a la izquierda y se mueve en función a las manecillas del reloj: está claramente que se debe aplicar una pequeña fuerza negativa (a la izquierda por “u”) para que el péndulo se ubique en la posición deseada.

Las tres reglas descritas anteriormente son “reglas lingüísticas” que son aplicadas en la base de reglas del controlador difuso, y están formadas por las variables lingüísticas y valores lingüísticos. Ya que los valores lingüísticos no representan valores precisos de una linealidad cuantificada que representan, al igual que las reglas lingüísticas tampoco son precisas. Estas representan simples ideas abstractas acerca de cómo se debe tener un buen control y esto puede variar respecto a cada persona que realice un control difuso.

La forma general de las reglas lingüísticas listadas con anterioridad es:

**Si** premisa **entonces** consecuencia

Como se puede apreciar las tres reglas listadas con anterioridad para el péndulo invertido, la premisa (las cuales son a veces llamadas “antecedentes”) están asociadas con las entradas del controlador difuso y están en el lado izquierdo de las reglas. La consecuencia (a menudo llamadas “acciones”) está

asociada con las salidas del controlador difuso y se encuentran en el lado derecho de las reglas.

Note que cada premisa (o consecuencia) puede ser compuesto por varios “términos” (ejemplo en la regla 3 de las presentadas “error es grande positivo **y** cambio de error es pequeño negativo” es una premisa que es la unión de dos términos).

Hay que tomar en cuenta que las tres reglas no son todas las reglas sobre el control del péndulo y se debe realizar todas las combinaciones posibles, para el péndulo con dos entradas y cinco valores lingüísticos por cada una de ellas se tienen una combinación de $5^2 = 25$ posibles reglas.

Una representación tabular de un conjunto de reglas para el péndulo invertido se muestra en la Tabla 2.2. Se debe tomar en cuenta que el cuerpo de la tabla contiene la consecuencia numérica de las reglas lingüísticas mientras tanto que en la columna izquierda y la fila superior contiene los términos de la premisa lingüística numérica, donde estos valores lingüísticos representan por ejemplo para el caso de “-2” igual que “grande negativo” la tabla representa por ejemplo una regla:

**Si** error es grande positivo **y** cambio de error es pequeño negativo **entonces** fuerza es pequeña negativa

Esta es la tercera regla anteriormente descrita. La Tabla 2.2 representa el conocimiento abstracto que el experto tiene de cómo controlar el péndulo dando como entradas al error y el cambio de error.

| "Fuerza" u | | "Cambio de Error" | | | | |
|---|---|---|---|---|---|---|
| | | -2 | -1 | 0 | 1 | 2 |
| "Error" e | -2 | 2 | 2 | 2 | 1 | 0 |
| | -1 | 2 | 2 | 1 | 0 | -1 |
| | 0 | 2 | 1 | 0 | -1 | -2 |
| | 1 | 1 | 0 | -1 | -2 | -2 |
| | 2 | 0 | -1 | -2 | -2 | -2 |

**Tabla. 2.2. Tabla de reglas para el péndulo invertido**

Si se realiza un estudio a la tabla propuesta anteriormente se puede verificar que existe una diagonal de ceros y viendo el cuerpo de la tabla se puede ver que tiene una simetría. Esta simetría que surge cuando las reglas son tabuladas no es un accidente y es en realidad una representación abstracta de conocimiento acerca de cómo controlar el péndulo, y esto surge debido a la simetría dinámica del sistema.

La cuantificación difusa del conocimiento es solo cuantificada en un camino abstracto y esto es el conocimiento de cómo el experto debe controlar la planta. A continuación, vamos a mostrar cómo utilizar la lógica difusa para cuantificar plenamente el significado de las descripciones lingüísticas.

Para poder cuantificar el significado de los valores lingüísticos se empieza realizando funciones de membrecía.

Un ejemplo de una función de membrecía es la figura 2.13. En la que se presenta una función “u” versus “e(t)” que toma un especial significado, esta función “u” cuantifica la certeza que e(t) puede ser clasificada lingüísticamente como “pequeño positivo”. Para entender el trabajo de las funciones de membrecía se realiza un análisis donde se presenta como interpretar esto por varios valores de e (t):

- Si $e(t) = -\frac{\pi}{2}\ entonces\ u\left(-\frac{\pi}{2}\right) = 0$, donde indica que la certeza de $e(t) = -\frac{\pi}{2}$ no es “pequeño positivo”.

- Si $e(t) = \frac{\pi}{8}\ entonces\ u\left(\frac{\pi}{8}\right) = 0.5$, donde indica que la certeza esta a la mitad que $e(t) = \frac{\pi}{8}$ sea “pequeño positivo” donde solo la mitad de la certeza podría ser cero.

- Si $e(t) = \frac{\pi}{4}\ entonces\ u\left(\frac{\pi}{4}\right) = 1.0$, donde indica que se encuentra absolutamente en la certeza de que $e(t) = \frac{\pi}{4}$ es “pequeño positivo”.

- Si $e(t) = \pi\ entonces\ u(\pi) = 0$, donde indica la certeza de que $e(t) = \pi$ no es “pequeño positivo” (para este caso será de “grande positivo”).

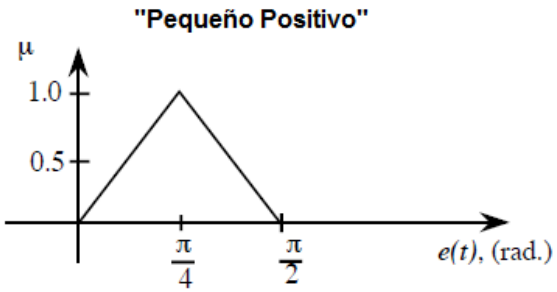



**Figura. 2.13. Función de Membrecía para el valor lingüístico “Pequeño Positivo”**

La función de pertenencia cuantifica el conocimiento, de manera continua, si los valores de e (t) pertenecen a (son miembros de) el conjunto de valores que son "pequeño positivo", y por lo tanto cuantifica el significado de la afirmación lingüística "error es pequeño positivo."

Existen funciones como en la figura 2.13., que se las puede representar de diferente manera como puede ser trapezoidal o de otras formas como se presenta en la siguiente figura. 2.14.

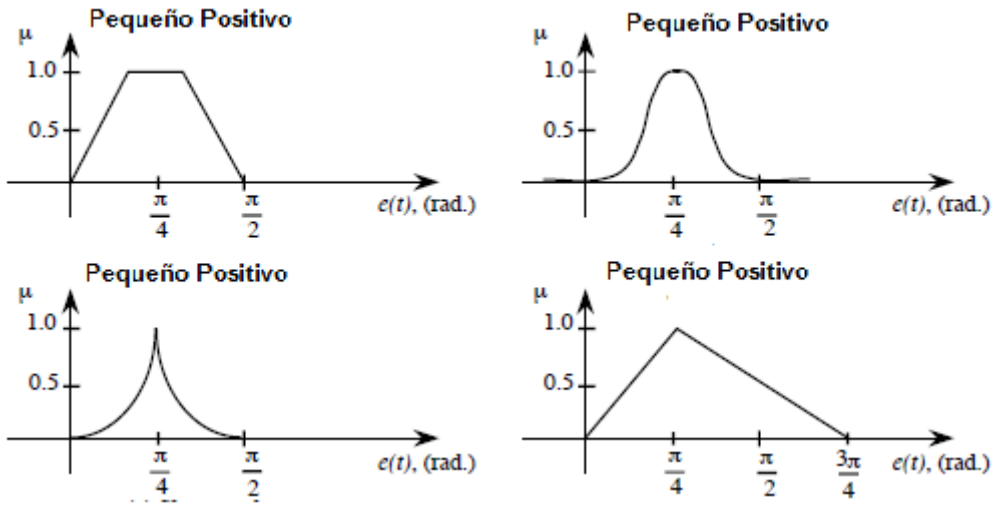



**Figura. 2.14. Funciones de Membrecía para representar el valor lingüístico "Pequeño Positivo"**

Tomando en cuenta la importancia de las funciones de membrecía, se definen funciones triangulares para el ejemplo del péndulo invertido, por ser más simples matemáticamente. Figura. 2.15.

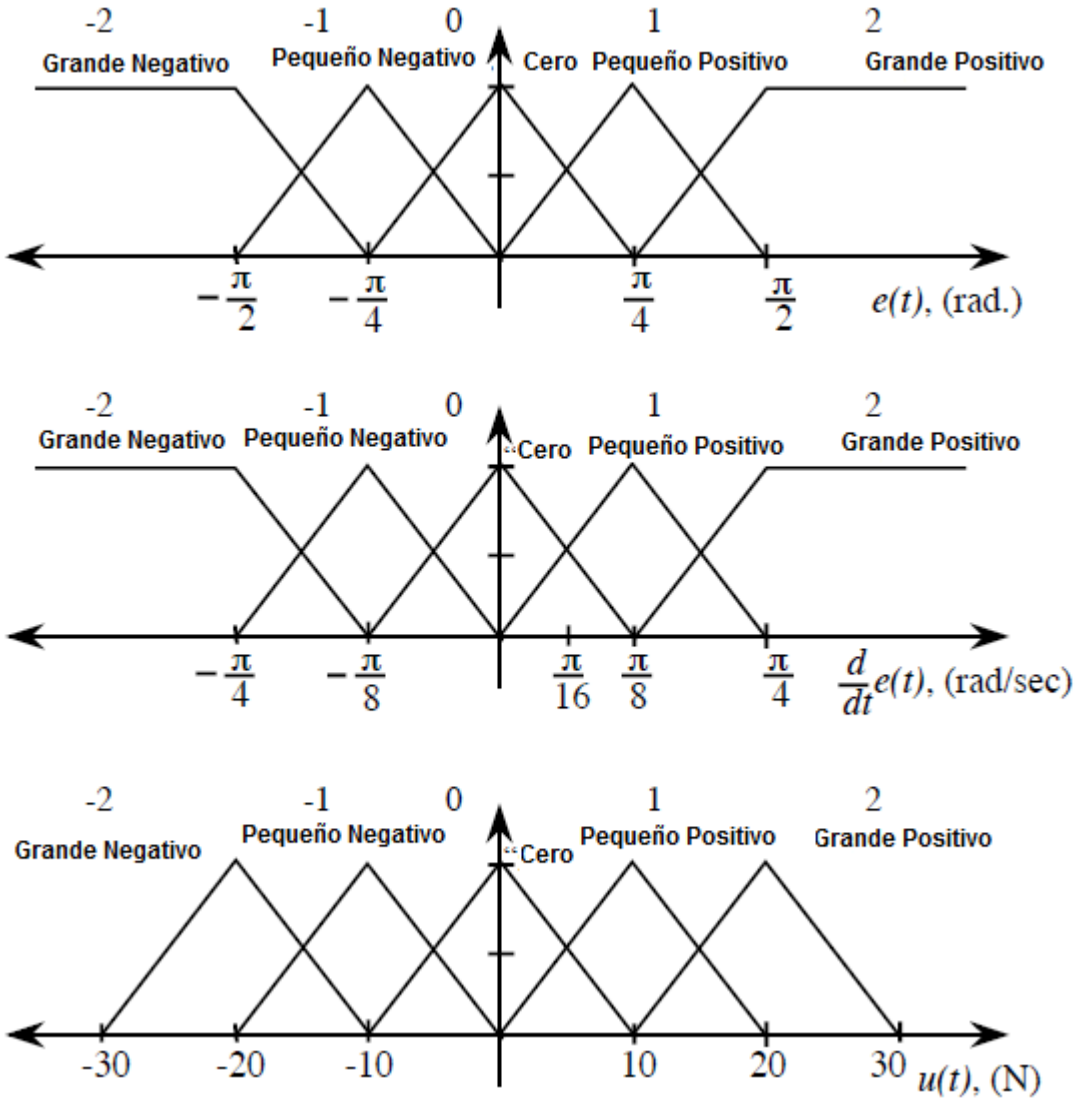



**Figura. 2.15. Funciones de membrecía para el péndulo invertido sobre el carro**

De esta manera una función de membrecía indica el grado de pertenencia de un valor lingüístico a un conjunto difuso, también se puede obtener el grado de certeza respecto a una función de membrecía; por ejemplo para el valor de $e(t) = -\pi/2$ donde la certeza del error es “grande negativo”, y como el valor de la derivada se mueve $-\pi/4$ llega a ser menos la certeza de que sea “grande negativo” y esta certeza llega a ser mas “pequeño negativo”.

Como se puede apreciar las funciones de membrecía cuantifican el significado de los estados lingüísticos que describen señales variables en el tiempo.

Se va a utilizar la notación $u_{zero}$ para representar a las funciones de membrecía asociada con el valor lingüístico “cero” esto se tendrá para las otras notaciones de igual manera.

Para entender de mejor manera el mecanismo de inferencia que se tiene en la estructura del controlador difuso se debe realizar una cuantificación de cada regla con una lógica difusa, para hacer esto cuantificamos el significado de las premisas de las reglas como se presenta en la figura 2.16., donde se tiene dos términos en la premisa de la regla:

**Si** el error es cero **y** cambio de error es pequeño positivo **entonces** la fuerza es pequeña negativa.

En la figura 2.16, se presenta los valores lingüísticos y las funciones de membrecía.

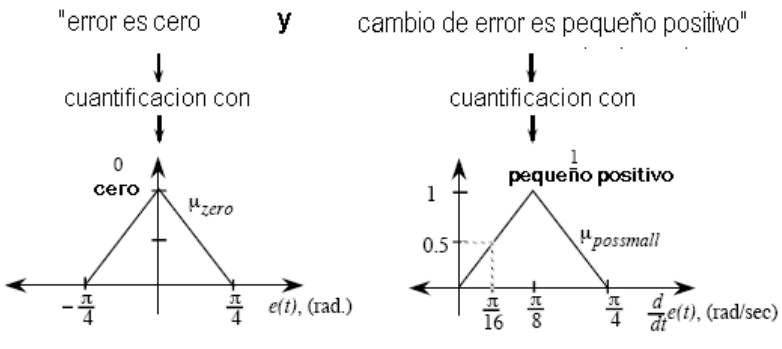



**Figura. 2.16. Funciones de membrecía de los términos de la premisa**

Para cuantificar el conocimiento**,** se empieza por suponer que error y la derivada del error toman los siguientes valores $e(t) = \pi/8$ y $\frac{d}{dt}e(t) = \pi/32$

Ubicándose en la figura 2.15., o también en la figura 2.16. Se tiene la certeza de las premisas;

$$u_{zero}\big(e(t)\big) = 0{,}5$$

Y

$$u_{pequeño\ positivo}\left(\frac{d}{dt}e(t)\right) = 0{,}25$$

Los valores $e(t)$ y $\frac{d}{dt}e(t)$, son la certeza de la siguiente sentencia:

"error es cero **y** cambio de error es pequeño positivo" si se asume que $e(t) = \pi/8$ y $\frac{d}{dt}e(t) = \pi/32$ toman estos valores

Una vez que se tiene definido la certeza de la sentencia, se procede a determinar la premisa, esto se denotará por $u_{premisa}$ existen algunos caminos que definen la premisa de los términos lingüísticos como se presenta a continuación los principales y más utilizados:

- *Mínimo:* Se define como el mínimo entre dos valores de membrecía $u_{premisa} = \min\{0.5\ , 0.25\} = 0.25.$

- *Producto:* se define entre la multiplicación de los dos valores de membrecía $u_{premisa} = (0.5)(0.25) = 0.125$.

Si se considera todas las posibilidades que pueden tomar los valores $e(t)$ y $\frac{d}{dt}e(t)$ se obtendrá una premisa multidimensional de $u_{premisa} = (e(t), \frac{d}{dt}e(t))$ que está en función de $e(t)$ y $\frac{d}{dt}e(t)$ para cada regla que se tenga en la base de reglas. Figura 2.17.

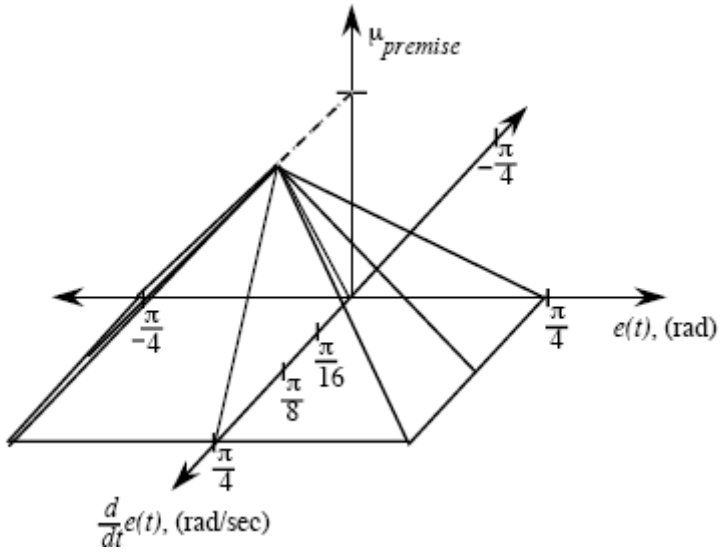



**Figura. 2.17. Función de membrecía de la premisa para una sola regla**

En general se tendrá diferentes funciones de membrecía para cada regla y estas estarán en función de $e(t)$ y $\frac{d}{dt}e(t)$. Se obtendrá un valor específico para obtener la cuantificación de la certeza de cada regla en la base de reglas, y esto conlleva a una gran capacidad de procesamiento.

Cuando esto ocurra el valor de la premisa para cada regla cambia y será aplicada para especificar la fuerza necesaria para el péndulo o en otras palabras se obtendrá el valor en decimal de la fuerza que será aplicada a la planta llamada

también este valor $U_{crisp}$ a este paso se lo conoce también como el bloque de defusificación o transformación de valores lingüísticos en valores reales numéricos.

Se debe determinar las reglas que se encuentran activas a la salida de u (t) en un tiempo “t” y para esto se utiliza las funciones de membrecía, en este caso si la premisa $u_{premisa} = (e(t), \frac{d}{dt}e(t))$ es mayor que cero se dice que la premisa esta activa. El mecanismo de inferencia busca determinar cuáles son las normas para averiguar que las reglas sean relevantes para la situación actual. En el siguiente paso, el mecanismo de inferencia tratará de combinar las recomendaciones de todas las reglas para llegar a una sola conclusión, siguiendo con el ejemplo del péndulo invertido de cómo saber que reglas se encuentran activas se supone que:

$$e(t) = 0$$

Y

$$\frac{d}{dt}e(t) = 0.294$$

En este caso se tiene diferentes valores que toman las entradas y por lo tanto se obtendrá diferentes certezas, para facilitar el entendimiento se presenta la siguiente Figura. 2.18.

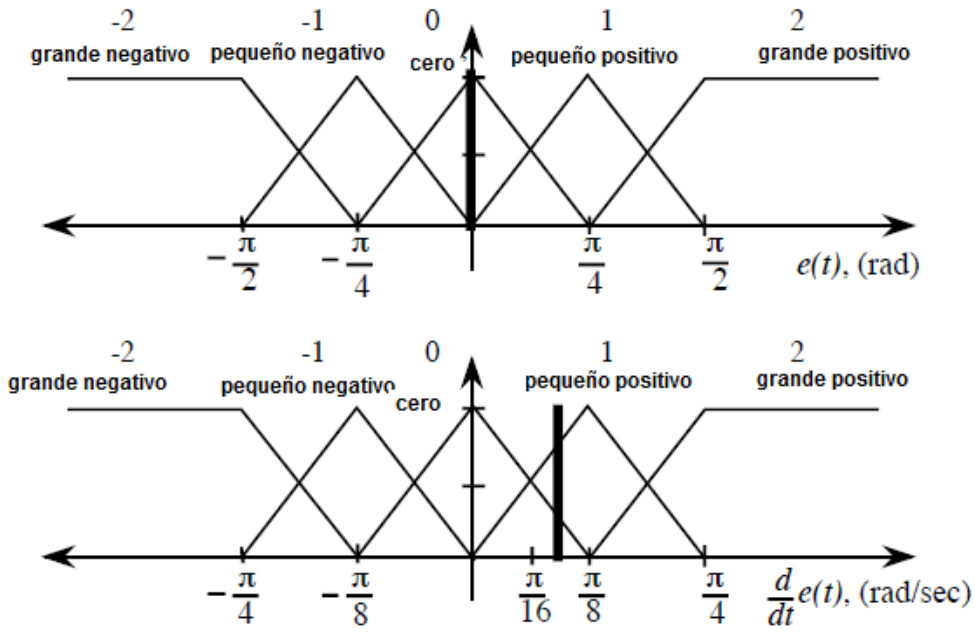



**Figura. 2.18. Funciones de pertenencia con valores de entrada**

Como se puede observar en el grafico con los valores que toman las entradas, la premisa $u_{cero}(e(t)) = 1$ quiere decir que no tiene reglas activas para este caso, pero para $\frac{d}{dt}e(t)$ tiene dos reglas activas, la premisa $u_{cero}\left(\frac{d}{dt}e(t)\right) = 0.25$ y $u_{pequeño\ positivo}\left(\frac{d}{dt}e(t)\right) = 0.75$. Por la línea negra que atraviesa la figura 2.18 de $\frac{d}{dt}e(t)$ , se puede observar que se tiene dos valores que toma el cambio de error que se denota por $\frac{d}{dt}e(t)$.

Una vez que se tiene identificado las reglas que están activas para valores de ejemplo que se tomaron $e(t) = 0$ y $\frac{d}{dt}e(t)$ =0.294, se tiene una base de reglas donde se encuentran todas las posibles combinaciones del péndulo invertido. Con la tabla se puede diferenciar las reglas que están activas para estos valores de ejemplo. Tabla 2.3.

| "Fuerza" u | | "Cambio de Error" | | | | |
|---|---|---|---|---|---|---|
| | | -2 | -1 | 0 | 1 | 2 |
| "Error" e | -2 | 2 | 2 | 2 | 1 | 0 |
| | -1 | 2 | 2 | 1 | 0 | -1 |
| | 0 | 2 | 1 | 0 | -1 | -2 |
| | 1 | 1 | 0 | -1 | -2 | -2 |
| | 2 | 0 | -1 | -2 | -2 | -2 |

**Tabla. 2.3. Tabla de reglas para el péndulo invertido**

Para los valores que toman las entradas del péndulo invertido se nota en la tabla 2.3., que las reglas en contengan las siguientes sentencias estarán activas:

"error es cero"

"cambio de error es cero"

"cambio de error es pequeño positivo"

Por lo tanto las reglas que están activas son las siguientes, tomando en cuenta la Tabla. 2.3.

1. **Si** error es cero **y** cambio de error cero **entonces** la fuerza es cero.
2. **Si** error es cero **y** cambio de error es pequeño positivo **entonces** la fuerza es pequeña negativa.

Hay que tomar en cuenta que el ejemplo del péndulo invertido solo tiene dos entradas y por lo tanto solo se tendrán como máximo 4 reglas activas, por ejemplo para el caso de $e(t) = 0$ y $\frac{d}{dt}e(t) = \frac{\pi}{8}$ solo se tendrán 2 reglas activas.

1. Para la primera regla que se tiene activa se obtiene la certeza de la premisa:

**Si** error es cero **y** cambio de error cero **entonces** la fuerza es cero

Se obtiene el *min* de la premisa:

$$u_{premisa} = \min\{0.25\,, 1\} = 0.25$$

Se representa en la siguiente figura. 2.19., la representación de la premisa:

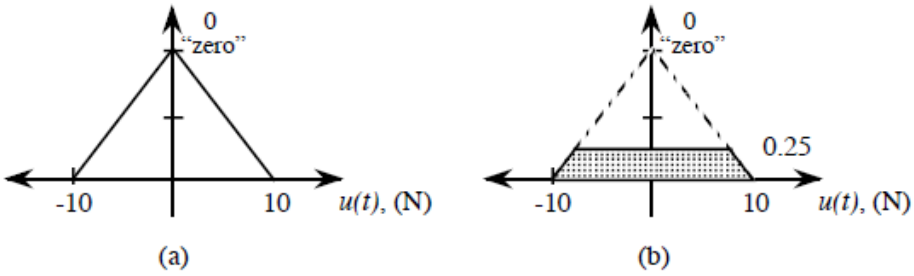



**Figura. 2.19. (a) Función de membrecía de la consecuencia y (b) conjunto implicado con la regla activa.**

2. Para la segunda regla que se tiene activa se obtiene la certeza de la premisa:

**Si** error es cero **y** cambio de error es pequeño positivo **entonces** la fuerza es pequeña negativa

Se obtiene el *min* de la premisa:

$$u_{premisa} = \min\{0.75\,, 1\} = 0.75$$

Se representa en la siguiente figura. 2.20., la representación de la premisa:

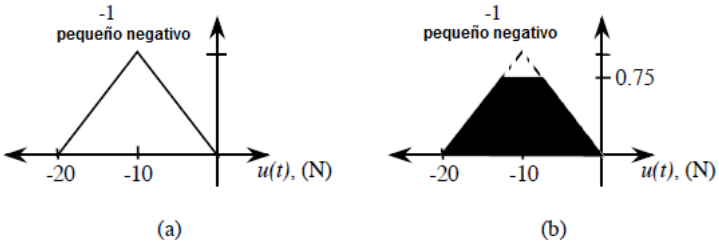



**Figura. 2.20. (a) Función de membrecía de la consecuencia y (b) conjunto implicado con la regla activa.**

Una vez que se tiene identificado la certeza de cada regla, lo siguiente que se realiza es la defusificación o la toma de decisiones para el sistema del péndulo y solo se determina las conclusiones de las reglas que se encuentran activas.

Para empezar con los procesos de defusificación se debe dibujar los conjuntos implicados en un solo eje como en la figura. 2.21.

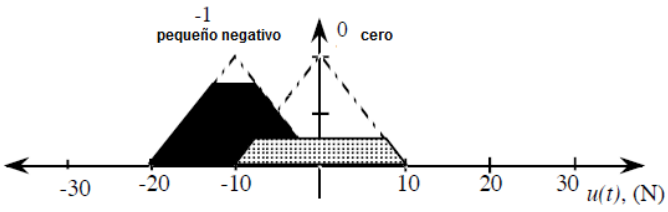



**Figura. 2.21. Conjunto difuso implicado**

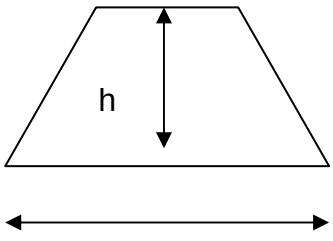
Una vez que se tiene dibujado los conjuntos difusos se deben aplicar la ecuación de Ucrisp. Y así obtener el resultado de la fuerza en valor numérico decimal, tomando el método del centroide se puede realizar este pasó con la siguiente ecuación:

$$u^{crisp} = \frac{\Sigma_i b_i \int u_i}{\Sigma_i \int u_i}$$

Donde $b_i$ es el centro de gravedad de la función de membrecía y para obtener la integral de la función de membrecía implicada se tiene que:

$$\int u_i = w\left(h - \frac{h^2}{2}\right)$$

Donde:




w

Aplicando la ecuación al ejemplo del péndulo y retomando los valores que se establecieron, el resultado es:

$$u^{crisp} = \frac{(0)(4.375) + (-10)(9.375)}{4.375 + 9.375} = -6.81\ [N]$$

Y este es el resultado de haber dado valores a las dos entradas que son el error y el cambio de error. Todo este proceso se lo realizó detallando de cómo funciona la teoría difusa y de cómo son aplicados los procesos de Fusificación, mecanismo de inferencia y defusificación para obtener el resultado de un valor numérico decimal que es aplicado al péndulo.

Realizando un resumen de lo que se tiene se presenta la Figura. 2.22.:

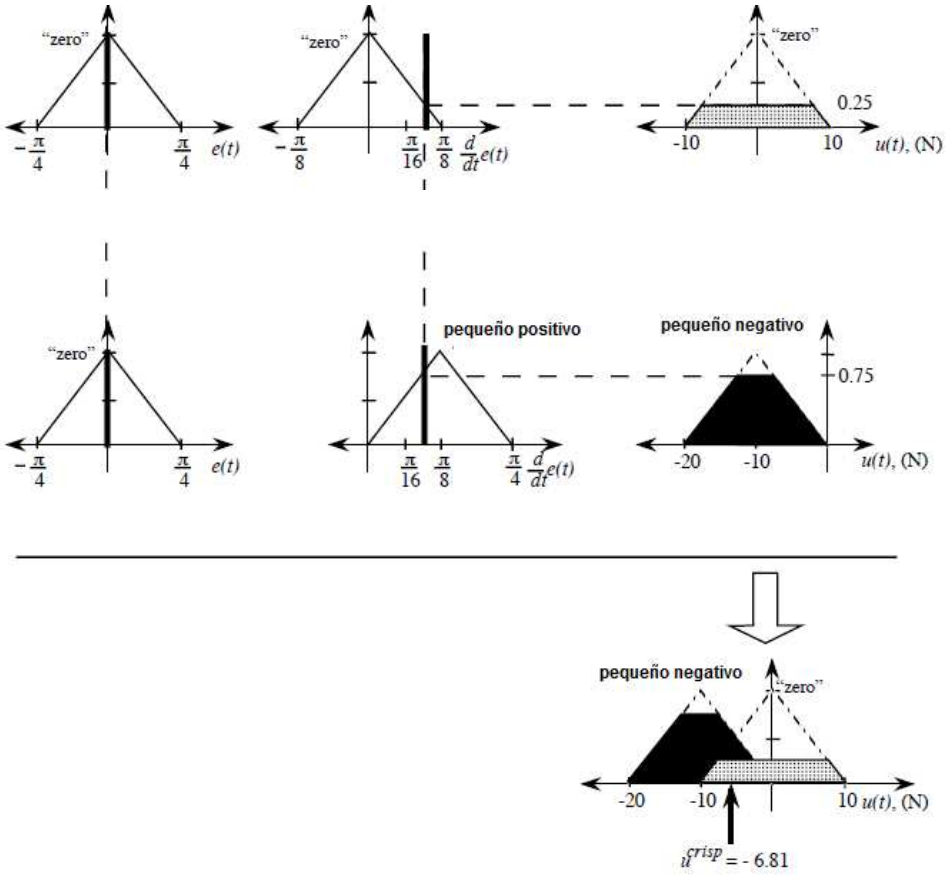



**Figura. 2.22. Representación grafica de operación del controlador difuso**

Esto complementa la descripción de la operación del controlador difuso y como se llega a obtener el resultado de la fuerza necesaria para aplicar al péndulo.

De esta manera se concluye con la teoría necesaria para empezar con el diseño del controlador difuso para el sistema de levitación magnética aplicando los mismos conceptos en un sistema diferente como lo es el sistema de levitación magnética.

## 2. 2 REDES NEURONALES

En nuestro sistema nervioso existen células llamadas neuronas que son una unidad de procesamiento. Reciben un estimulo eléctrico de otras neuronas principalmente a través de su árbol dendrítico. El estimulo eléctrico recibido al pasar de un cierto umbral causa que la neurona a su vez envié una señal eléctrica a través de su axón a otras sucesivas neuronas basado en la fuerza de interconexión entre las neuronas. La red puede ser capaz de cómputos y detección de patrones complejos.

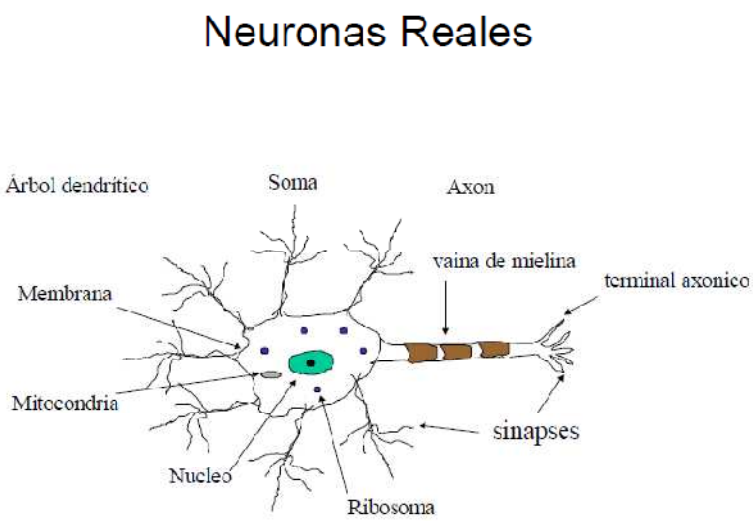



**Figura. 2.23. Neurona Real**

### 2.2.1. Sinapsis

Cuando el árbol dendrítico recibe impulsos eléctricos sobre un umbral voltaje especifico la neurona manda una señal eléctrica a través del axón a los terminales Axónicos. En esos terminales se conectan los terminales axónicos (a través de sinapsis) a otras neuronas, los terminales axónicos se pueden conectar a través de sinapsis a diferentes partes de la neurona pero típicamente se considera solo la conexión a otras dendritas.

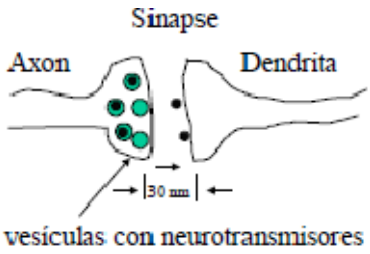



**Figura. 2.24. Conexión de una Sinapsis**

Dependiendo del neurotransmisor (ion) el potencial inducido en terminal post-sináptico (dendrita) puede ser positivo (excitado) o negativo (inhibidor).

Las redes neuronales dentro del campo electrónico son una herramienta que brinda la capacidad de realizar un control a través de aprendizaje. Es así que esta técnica es aplicada al sistema de levitación magnética.

### 2.2.2 Modelo de una Neurona

McCulloch and Pitts (1943) desarrollaron el modelo original de una neurona, esta incluyo diferentes entradas (x), pesos para esos entradas (w), una función no lineal (f) de transferencia y la salida (O).

En el modelo original no se incluyo un profesor (Teacher) para entrenar a la red. La función no lineal de transferencia usada era el escalón (hard limit o step function), Se incluyo una entrada de bias, para evitar salidas no deseadas cuando las entradas son cero.

### 2.2.3 Modelo MP de una Neurona Perceptron

Se modela la neurona como una red con pesos (conexiones sinápticas) desde los nodos j al nodo i, *wij.* Como se presenta a continuación la siguiente figura. 2.25.

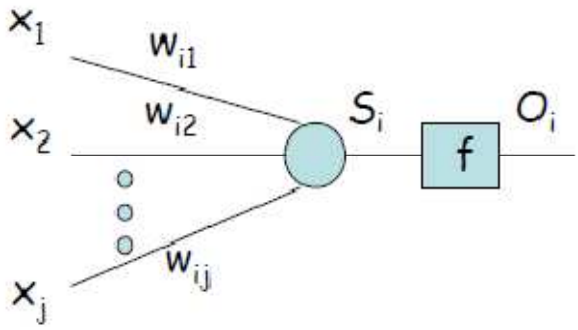



**Figura. 2.25. Modelo de Red Neuronal Perceptron**

La suma a la neurona $s_i$ se establece con la siguiente fórmula:

$$S_i = \sum (w_{ji} x_j + bias)$$

El bias es un peso adicional que fue propuesto como un valor adicional para la neurona, en caso que se tenga entradas de valores cero y evitar posibles errores a la salida.

La salida de la función de transferencia depende del tipo de función que se tiene:

$$O_i = \begin{matrix} -1\ Si\ S_i < \theta_i \\ 1\ Si\ S_i \geq \theta_i \end{matrix}$$

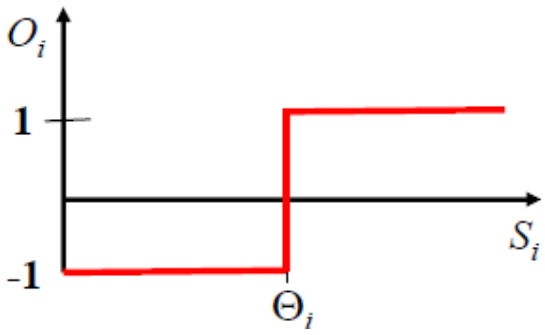



**Figura. 2.26. Función de transferencia SGN.**

Entonces la salida de la red neuronal queda expresada de la siguiente manera:

$$O_i = sgn\left(\sum (w_{ji}x_j + bias)\right)$$

McCullough y Pitts (1943) mostraron que este modelo podría calcular funciones lógicas para poder construir maquinas de estados finitos. Pueden construir compuertas lógicas (AND, OR, NOT), con estas compuertas pueden calcular cualquier función lógica usando una red de dos niveles AND-OR. También se denomina perceptor de una capa.

### 2.2.4 Redes Neuronales Artificiales (ANNs)

Las ANNs son un paradigma para hacer cómputo y para la detección de patrones basado en la interconexión paralela de neuronas artificiales. Las ANNs son un modelo basado en los complejos sistemas nerviosos de los animales y seres humanos con su gran cantidad de interconexiones y paralelismo. Los

comportamientos inteligentes de estos son una propiedad emergente del gran número de unidades no un resultado de reglas simbólicas o algoritmos.

Las características previamente expuestas de las neuronas se duplican para crear redes de neuronas artificiales que tienen capacidades similares de procesar información en forma paralela. Algunas capacidades de las redes neuronales son:

1. Aprendizaje
2. Clasificación
3. Almacenaje de información
4. Interpolación
5. Adaptación

Las redes neuronales artificiales recrean o modelan las características de las neuronas ya mencionadas en sistemas computacionales de software y hardware. Usan grupos de muchas neuronas artificiales (como la neurona McCulloch-Pitts o MP) usan principios de computación colectiva y descentralizada (paralela) son capaces de aprender, operaciones robustas y resistentes a fallas de neuronas individuales, la red hace asociaciones automáticamente.

La red crea el “programa” al ajustar los pesos durante su aprendizaje al operar requieren de sincronizada o que las señales entren al mismo tiempo a todas las neuronas en el mismo nivel (no así las biológicas que usan la suma espacio-temporal).

### 2.2.5 Redes Neuronales Multicapa

El modelo de una red neuronal con múltiples capas difiere de las que hemos visto hasta ahora. Estas redes neuronales multicapa incluyen neuronas en capas escondidas, una capa escondida significa que no es visible ni a la entrada ni a la salida.

Un perceptron de dos capas puede discernir regiones poligonales, uno de tres o más capas puede discernir regiones arbitrarias (Multi Layer Perceptron o MLP).

Estas redes neuronales tienen la capacidad de separar entradas en múltiples funciones lineales y de esta manera pueden detectar patrones más complejos que redes de una función lineal como las vistas anteriormente.

Como ejemplo de estas redes neuronales multicapa se presenta la solución de la compuerta XOR. Figura 2.27.

Sobre las líneas se indican los pesos y en los círculos indican umbrales (thresholds), la función no lineal es una “función paso” con valores de 1 si el umbral es excedido y de 0 si el umbral no es excedido.

Ejemplo XOR (hay otras soluciones posibles):

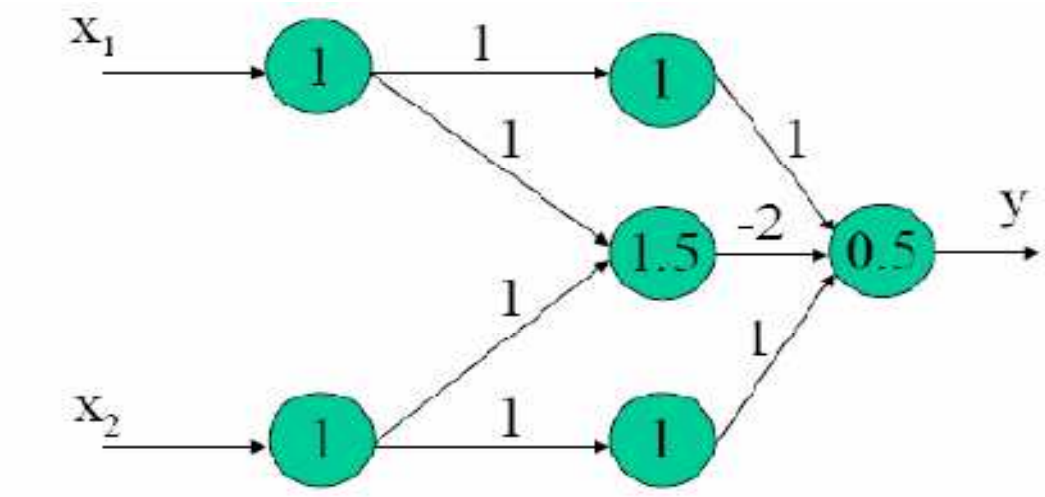



**Figura. 2.27. Representación de una red multicapa Compuerta XOR**

Si se realiza la operación de la función XOR en dicha red neuronal que se encuentra con los pesos actuales se puede obtener el resultado deseado:

| X1 | X2 | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Tabla. 2.4. Función XOR**

### 2.2.6 Funciones no Lineales

Las funciones más usadas no lineales son:

- Función Sigmoide
- Función Paso
- Función Rampa

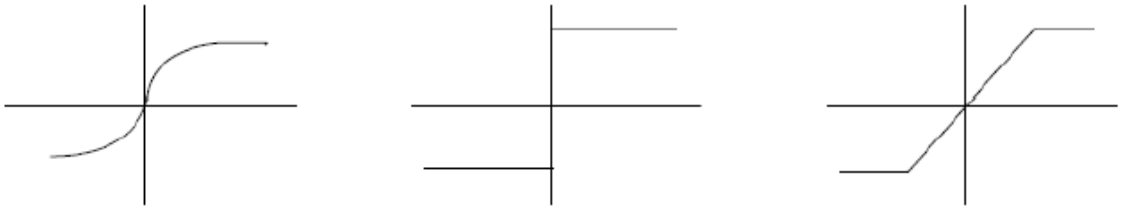


**Figura. 2.28. Funciones no lineales**

Las funciones sigmoides son:

- Mono tónicos (sin discontinuidades)
- Bounded (Limitados)
- Tiene derivados simples $f'(s) = kf(s)[1 - f(s)]$
- No lineal

Las funciones paso y rampa son:

- No mono tónicas (tiene discontinuidades)
- No tienen derivados simples
- Lineal (dentro de las áreas con limites )

Las funciones no lineales evalúan el resultado que se tiene a la salida de la neurona y se utiliza como por ejemplo para la neurona perceptron como un elemento de clasificación.

### 2.2.7 Algoritmo de aprendizaje

El proceso de aprendizaje es el proceso por el cual la red neuronal se adapta al estímulo modificando sus pesos y eventualmente produce un resultado esperado.

Hay dos tipos de aprendizaje: supervisado y sin supervisión:

Aprendizaje supervisado incluye un supervisor (o teacher) que le indica a la red cuan cerca esta de aprender y va modificando los pesos neuronales.

Sin supervisión solamente se forman grupos de clasificación de acuerdo a indicaciones o propiedades, no se calcula un error (E).

Se puede usar el error o el gradiente de error para que la red aprenda, la dirección del gradiente indica un incremento del Error que por ende cambia los pesos hacia la dirección contraria.

### 2.2.8 Aprendizaje supervisado

El modelo de aprendizaje con profesor (teacher) significa que durante el aprendizaje hay circuitos (o programas) externos a la neurona que comparan la salida deseado (t) con el actual (p) y calculan el error entre ellas (E= t - p).

Usando el error (E) un algoritmo de aprendizaje modifica los valores de los pesos uno por uno tratando de minimizar el error, este proceso requiere de muchas iteraciones.

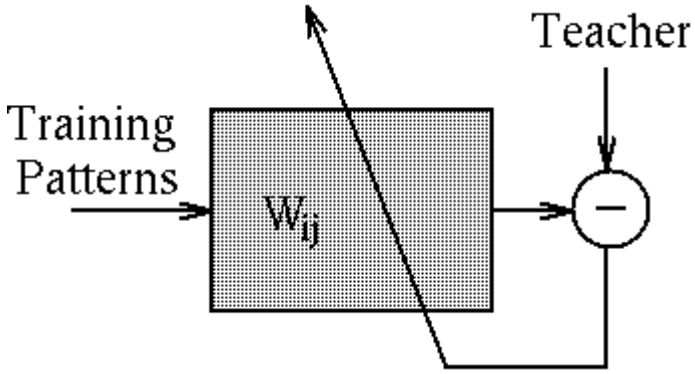



**Figura. 2.29. Neurona con Aprendizaje Supervisado**

Donde Wij., son los pesos que se van ajustando hasta obtener el valor deseado (t).

### 2.2.9 Algoritmo de aprendizaje de retro-propagación “Backpropagation”

El algoritmo “Backpropagation” para redes multicapa realiza su labor de actualización de pesos y ganancias con base en el error cuadrático medio.

El algoritmo “Backpropagation” es un método de aprendizaje supervisado y por lo tanto necesita de un set de entrenamiento o patrones de entrenamiento de la siguiente forma:

$$\{p_1,t_1\}, \{p_2,t_2\}, \ldots, \{p_Q, t_Q\}$$

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

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

Es importante recalcar que no existe una técnica para determinar el número de capas ocultas, ni el número de neuronas que debe contener cada una de ellas

para un problema especifico, esta elección es determinada por la experiencia del diseñador, el cual debe cumplir con las limitaciones de tipo computacional.

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

El algoritmo “Backpropagation” se presenta en el siguientes punto en el que se explica de mejor manera como realiza la actualización de los pesos.

### 2.2.10 El principio básico de entrenamiento de una red neuronal multicapa usando “Backpropagation”

El algoritmo de “Backpropagation” se presenta gráficamente utilizando la siguiente figura. 2.30., que representa una red neuronal de 3 capas con dos entradas y una salida.

Donde:

$f_1$: Representa la función no lineal,

$e$: Representa la suma de las entradas multiplicados por su peso correspondiente

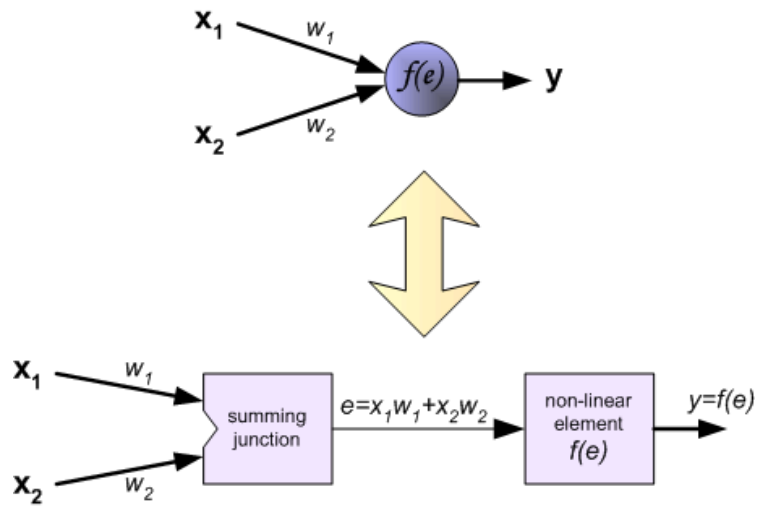



**Figura 2.30. Representación de la red neuronal**

Adicional a esto se debe tener en cuenta el “bias” (thresholds) que bebe ser sumado como un peso adicional. Y también tendrá que ser actualizado por el algoritmo.

Para representar el algoritmo de “Backpropagation” se ha realizado gráficamente el proceso.

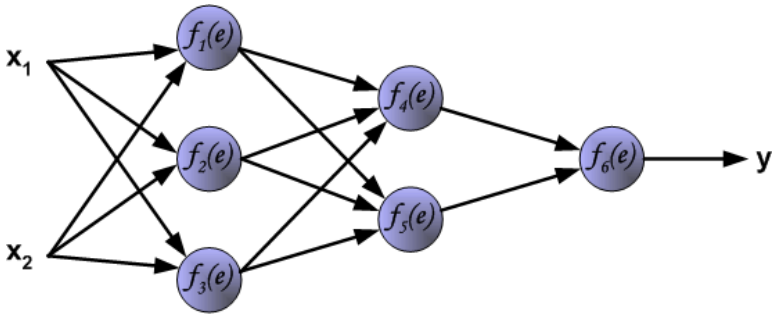



**Figura 2.31. Red Neuronal de 3 capas**

Cada neurona está compuesta de dos unidades. La primera unidad suma los productos de los coeficientes entre los pesos y la señal de entrada. La segunda unidad realiza la función no lineal, llamada función de activación.

Para poder entrenar a la red neuronal se necesita una tabla de datos o los llamados objetivos a los cuales tiene que llegar la red neuronal en su entrenamiento, el entrenamiento neuronal es un proceso interactivo. En cada interacción los coeficientes de los pesos son modificados.

Se puede determinar el valor de la señal de salida de cada neurona en cada capa de la red. Las figuras que se presentan describen como la señal es propagada a través de la red, el símbolo $W_{(x_m)}n$ representa los pesos de conexión, el símbolo "$x_m$" representa la entrada y el símbolo "*n*" representa la neurona en la capa de entrada. El símbolo "$y_n$" representa la señal de salida de la neurona *n.*

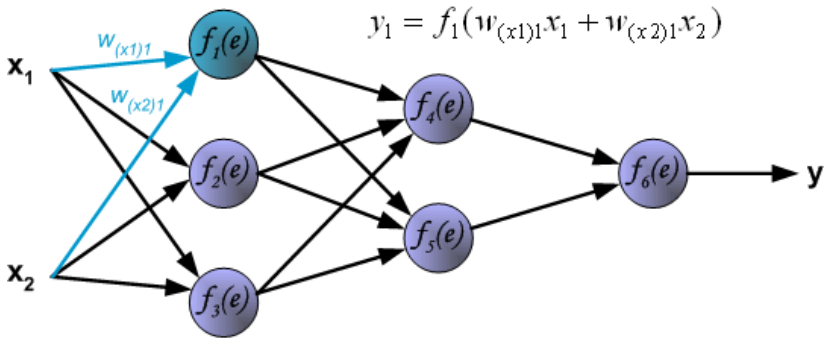



**Figura 2.32. Salida de la neurona numero uno.**

Se obtiene las ecuaciones de salida de cada neurona en la capa respectiva. Figura 2.33.

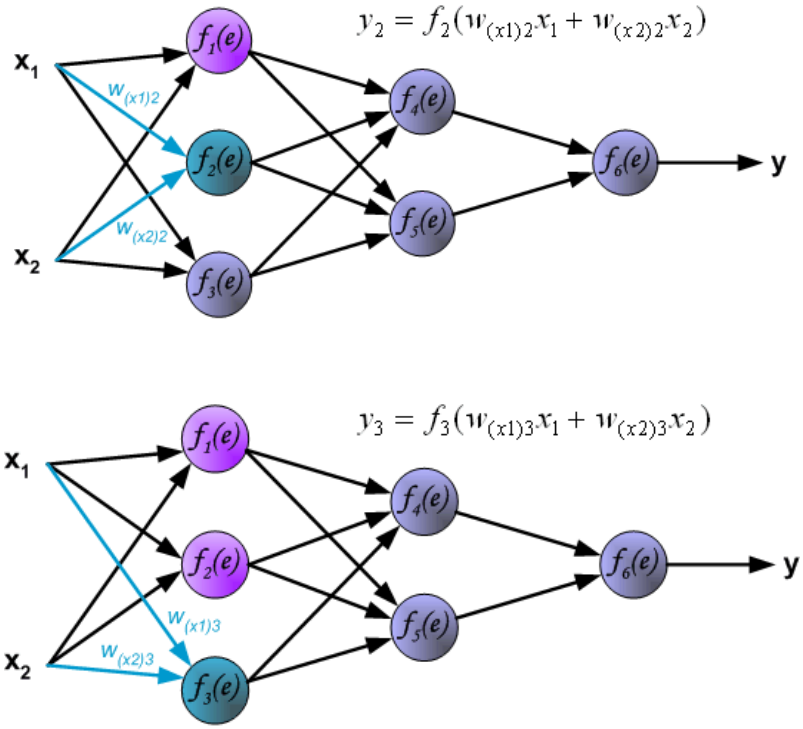
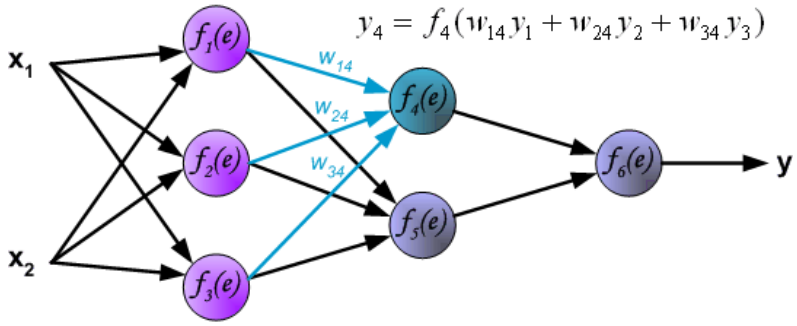



**Figura 2.33. Salida de neuronas capa 1**

Una vez que se termina con la primera capa de la red neuronal con tres neuronas, la propagación de la señal continúa con la siguiente capa oculta. Figura. 2.34.



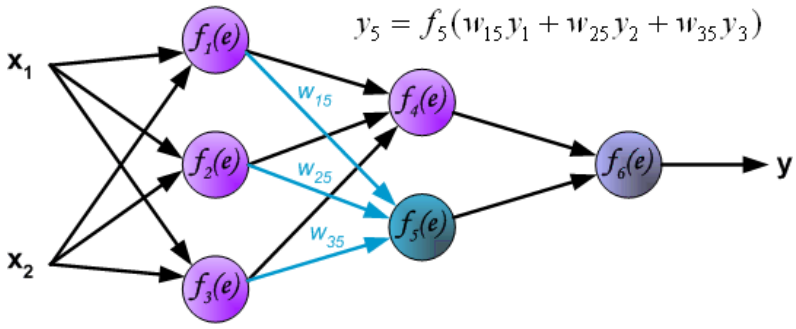



**Figura 2.34. Representación de la capa oculta**

Propagación de la señal a través de la capa de salida.

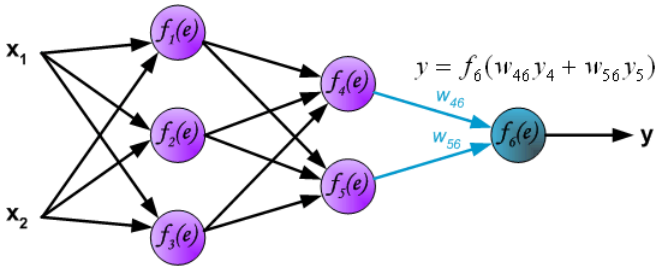



**Figura 2.35. Salida de la neurona final.**

El siguiente paso del algoritmo una vez que se llega a la señal de salida de la red, es obtener la diferencia entre el valor deseado y el obtenido, la diferencia es llamada señal de error "$\delta$" en la neurona de salida. Figura. 2.36.

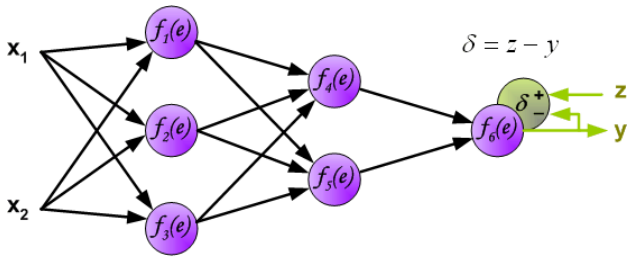



**Figura. 2.36. Error en la neurona final.**

Es imposible realizar el cálculo de error de la señal para las neuronas internas directamente, porque el valor de salida de estas neuronas es desconocido. Por muchos años la efectividad de un método para entrenar una rede neuronal multicapa fue desconocido. Solo a mediados de los ochenta el algoritmo de **Backpropagation** uno de los mejores métodos. La idea es propagar la señal de error $\delta$ hacia atrás a todas las neuronas, con la cual la señal de salida es la entrada para la neurona en discusión. Figura 2.37.

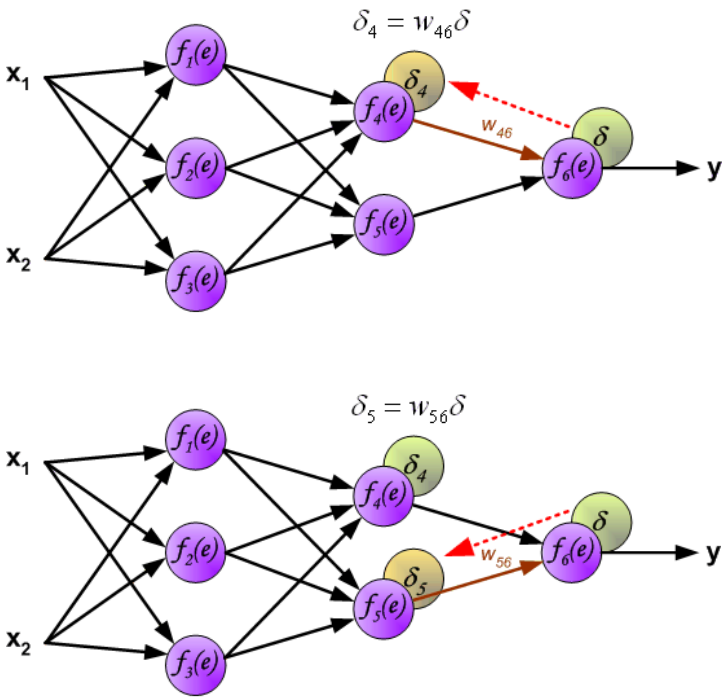



**Figura. 2.37. Propagación del error en la capa oculta.**

Los coeficientes de los pesos $w_{mn}$ usados para propagar el error hacia atrás son los mismos en cada capa. La señales son propagadas desde la salida a la entrada una tras otra, esta técnica es usada para todas las capas de la red neuronal. Si el error propagado llega desde otras neuronas estos son sumados. Figura. 2.38.

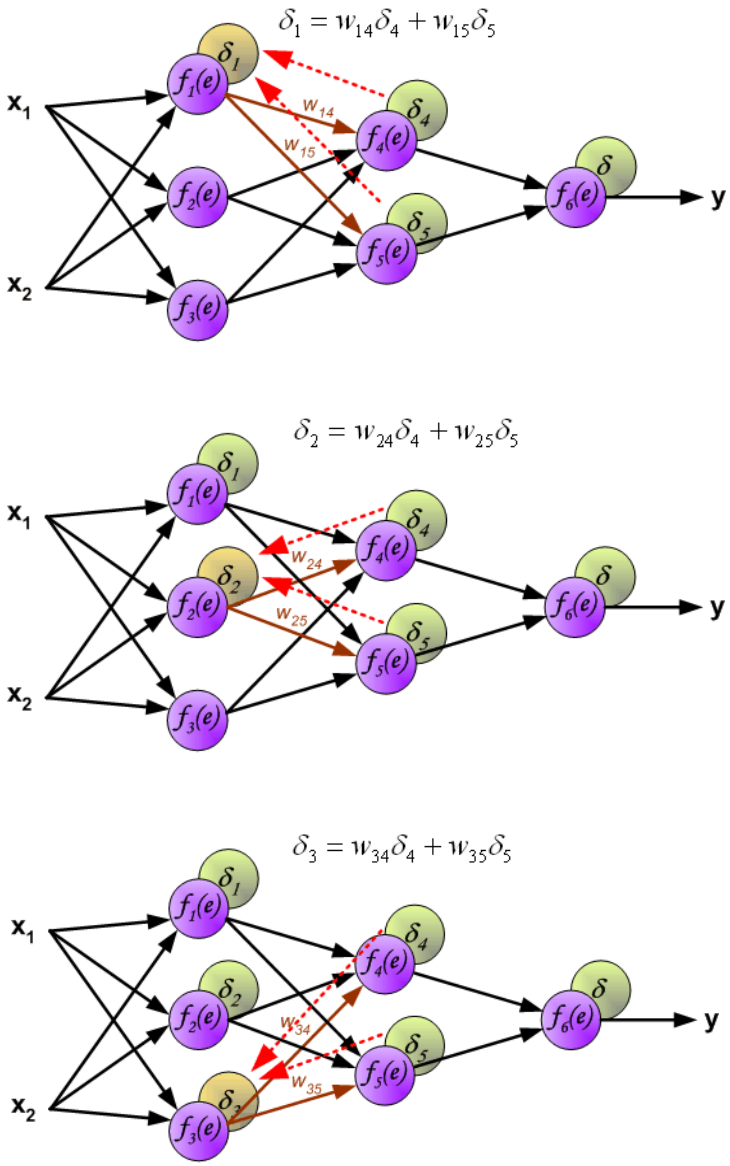



**Figura. 2.38. Propagación del error en la primera capa.**

Cuando la señal de error para cada neurona es calculada, los coeficientes de los pesos de cada nodo de neurona de entrada pueden ser modificados. Esto se lo hace con la formula de la derivada de la función de cada neurona. Figura 2.39.

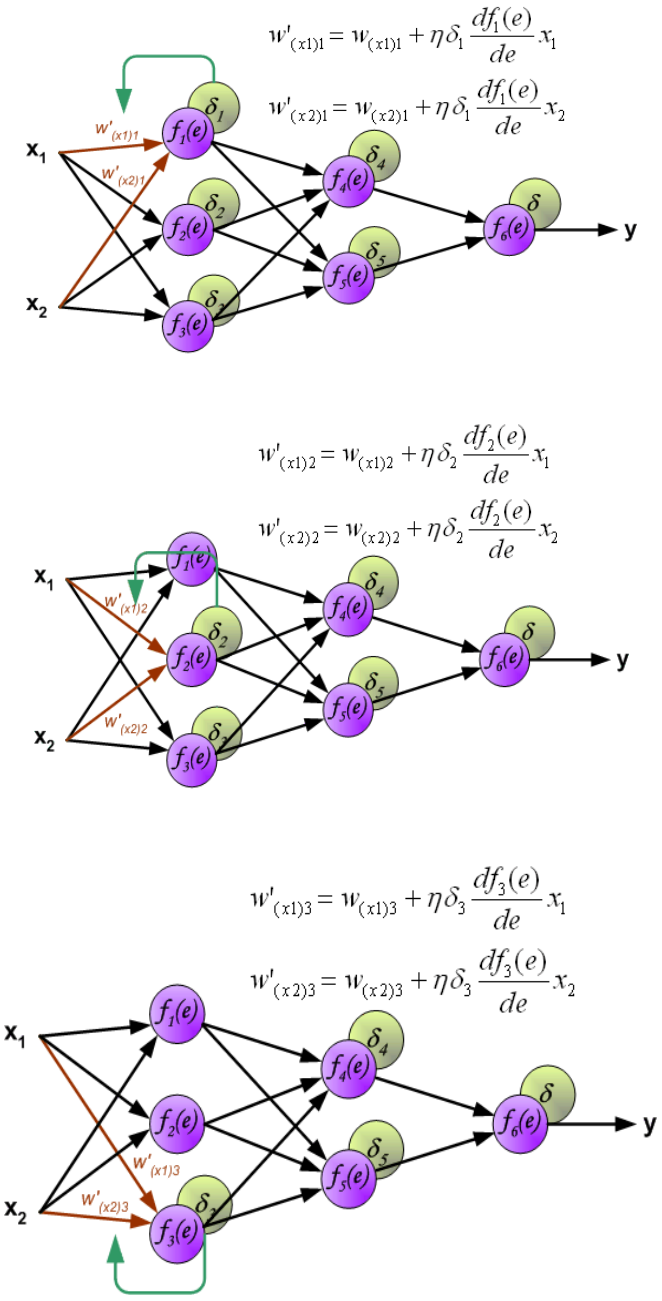



**Figura. 2.39. Derivada de la función de activación en cada neurona capa de entrada**

Y se continúa con el mismo proceso hacia la capa oculta. Figura 2.40.

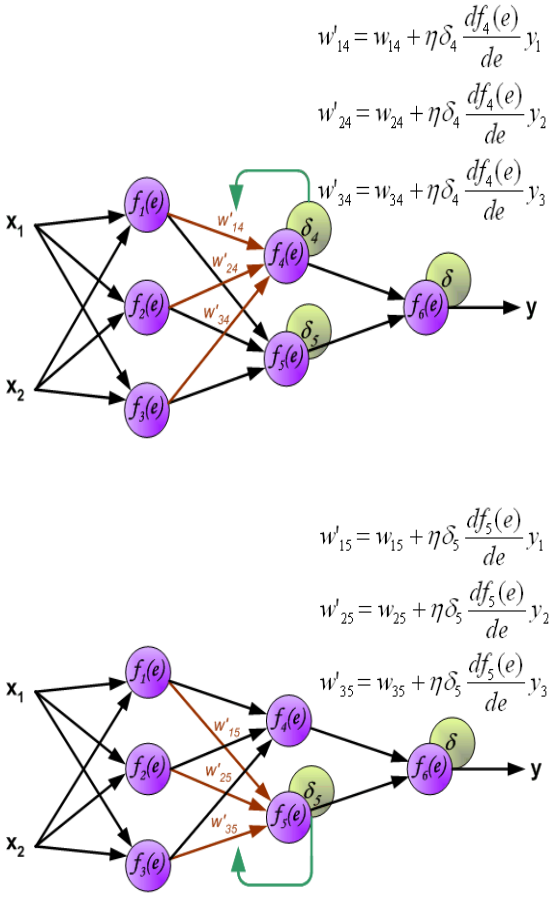



**Figura. 2.40. Derivada de la función de activación en cada neurona capa oculta**

Se continúa con la siguiente capa de salida, en la que se actualiza los pesos de salida terminando con el proceso del algoritmo.

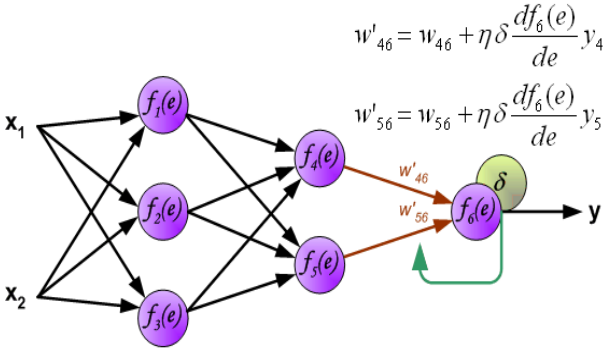



**Figura. 2.41. Derivada de la función de activación en la capa de salida**

El coeficiente $\eta$ afecta a la velocidad de aprendizaje de la red neuronal, hay pocas técnicas para seleccionar este parámetro. El primer método es empezar el proceso de entrenamiento con un valor grande, mientras los coeficientes de los pesos están siendo calculados el parámetro está siendo gradualmente más pequeño.

El segundo y más complicado método es empezar el entrenamiento con un valor pequeño, Durante el proceso de enseñanza es el parámetro el que se incrementa cuando la enseñanza avanza y luego vuelve a disminuir en la etapa final. A partir de este proceso de enseñanza con el valor del parámetro bajo permite determinar los signos de los coeficientes de los pesos.

Y de esta manera es como actúa el algoritmo de “Backpropagation”, y como es un proceso interactivo terminara hasta cuando el error sea cero y se cumpla el objetivo.

En 1989 Funahashi demostró matemáticamente que una red neuronal multicapa puede aproximar cualquier función no lineal o mapa lineal multivariable, $f(x) = R^n \rightarrow R$

Este teorema es de existencia, pues prueba que la red existe pero no indica cómo construirla y tampoco garantiza que la red aprenderá la función. Existen muchos estudios sobre las redes neuronales pero aun no se ha podido definir una técnica sobre cómo empezar a realizarlas.

# CAPITULO III

# DISEÑO Y SIMULACIÓN DE CONTROLADORES.

# CAPITULO III

# DISEÑO Y SIMULACIÓN DE CONTROLADORES

Dentro de este capítulo se explica el diseño de controladores inteligentes para el sistema de levitación magnética con la herramienta Simulink. Simulink es una herramienta muy utilizada para diseñar y simular sistemas típicos de control.

De esta manera se utiliza la herramienta para realizar la simulación de los controladores inteligentes y posteriormente pasar al modelo de experimentación; también se presenta el diseño de controladores difusos y controladores con redes neuronales, los cuales fueron investigados e implementados sobre el modelo experimental.

## 3.1 MODELO MATEMÁTICO DEL SISTEMA DE LEVITACIÓN MAGNÉTICA

El modelo matemático del sistema de levitación magnética está considerado para realizar las respectivas simulaciones, mas no es utilizado dentro del diseño de los controladores inteligentes.

En el modelo de levitación magnética se definen un conjunto de parámetros que han sido obtenidos de trabajos previos sobre el sistema. Figura 3.1.

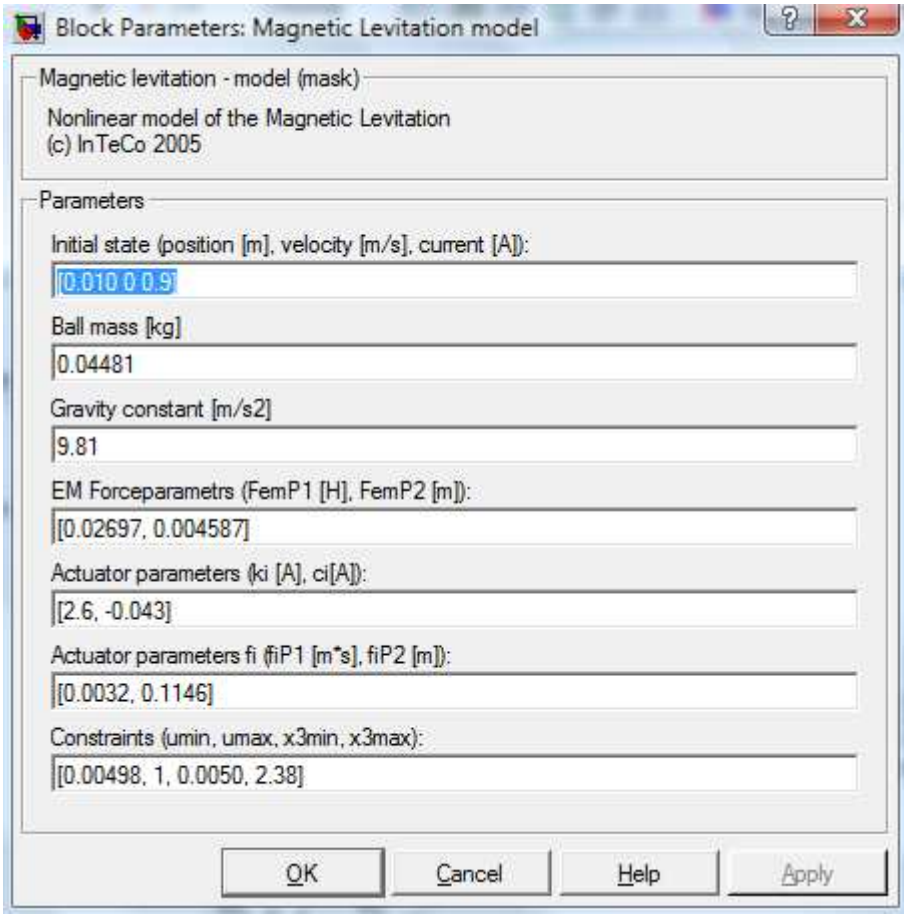



**Figura. 3.1. Parámetros del modelo de simulación**

Las ecuaciones que rigen al modelo de levitación magnético no lineal son las siguientes:

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -\frac{F_{em}}{m} + g$$

$$\dot{x}_3 = \frac{1}{f_i(x_1)}\left(k_i u + c_i - x_3\right)$$

$$F_{em} = x_3^2 \frac{F_{emP1}}{F_{emP2}} \exp\left(-\frac{x_1}{F_{emP2}}\right)$$

$$f_i(x_1) = \frac{f_{iP1}}{f_{iP2}} \exp\left(-\frac{x_1}{f_{iP2}}\right)$$

Donde **Fem**. Es la fuerza electromagnética que se genera cuando se brinda de corriente a la bobina electromagnética.

Donde:

$$x_1 \in [0, 0.016], \qquad x_2 \in \mathcal{R}, \qquad x_3 \in [x_{3MIN}, 2.38]$$

$$u \in [u_{MIN}, 1]$$

En las ecuaciones se observa que la fuerza electromagnética sigue una relación exponencial con la posición. Esta relación se muestra en la figura 3.2., y debe notarse que esta característica corresponde a un valor de corriente constante.

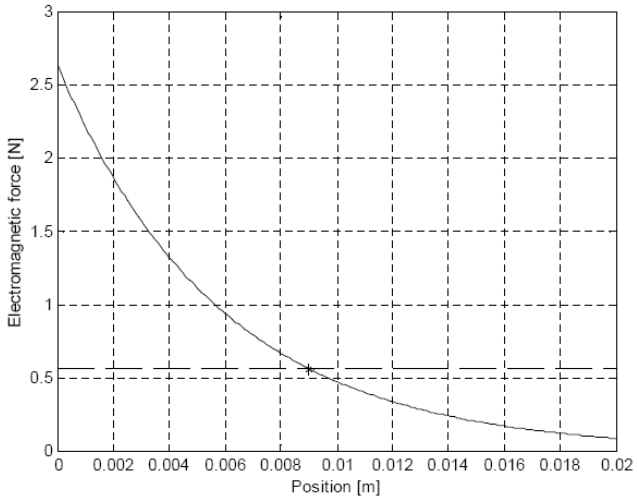



**Figura. 3.2. Fuerza Electromagnética en función de la posición**

En la figura 3.2., se puede apreciar que para una fuerza electromagnética constante de 0.6 [N] la posición de la esfera se ubica en 0.009 metros desde la bobina electromagnética.

En la figura 3.3, se muestra la variación de la fuerza electromagnética con la corriente para un valor de posición constante.

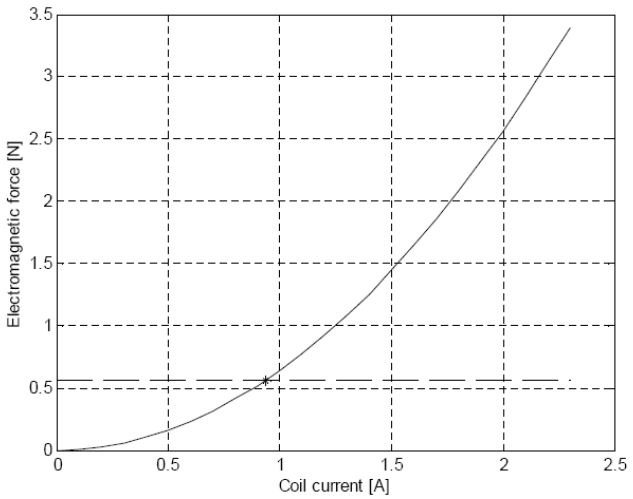



**Figura. 3.3. Fuerza Electromagnética en función de la corriente**

Se puede apreciar que para la fuerza electromagnética de 0.6N se tiene una corriente de 0.9 [A] aproximadamente. La fuerza electromagnética depende de dos variables que son la distancia desde la bobina y la corriente que se aplica a la bobina, como se aprecia claramente en las dos graficas anteriores.

La levitación de la esfera requiere el equilibrio de la fuerza de gravedad con la fuerza electromagnética que como se ha dicho depende de dos variables, la posición y la corriente. Esta idea muestra la complejidad del control puesto que en cada posición debe proveer la corriente exacta para mantener el equilibrio de las fuerzas.

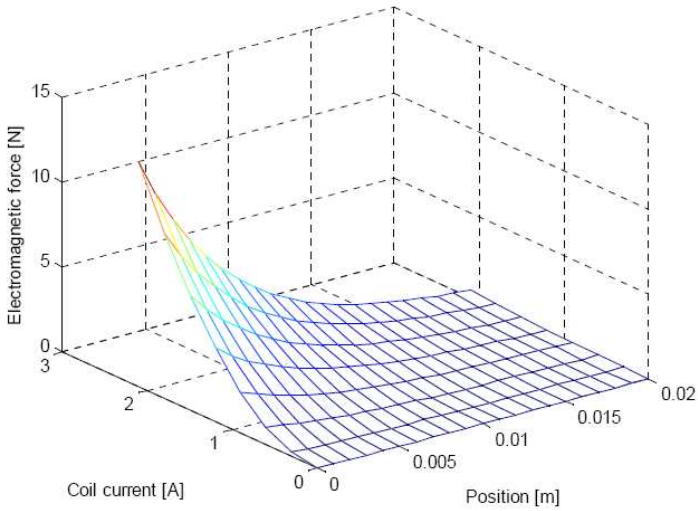



**Figura. 3.4. Posición, corriente y fuerza electromagnética**

En la Figura. 3.4**.**, se puede observar en una sola gráfica la fuerza electromagnética, posición y corriente del sistema de levitacion magnética.

En las ecuaciones se observa también que la posición produce una variación en la corriente lo cual hace aún más complejo el objetivo de equipibrar las fuerzas. La relación entre la variación de corriente y la posición está definido por una función fí que se presenta en la figura 3.5.

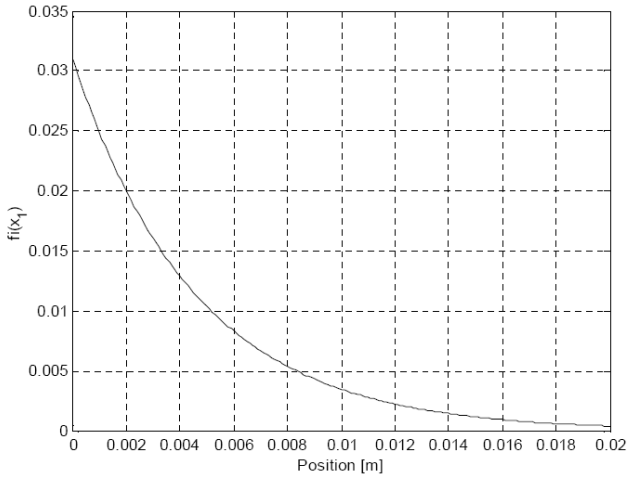



**Figura. 3.5. Función de la posición**

## 3.2 DISEÑO DEL CONTROLADOR DIFUSO PROPORCIONAL DERIVATIVO PD

La lógica difusa es básicamente una lógica multievaluada que a diferencia de la lógica binaria permite valores intermedios para definir evaluaciones convencionales como sí/no, verdadero/falso, negro/blanco, etc. De esta forma se realiza un intento de aplicar una forma más humana de pensar en la programación de computadoras.

Parte de este controlador se lo realiza tomando en cuenta la teoría en sistemas difusos y con el apoyo de la herramienta de MATLAB llamada **FIS EDITOR.**

Es así como se empieza tomando datos sobre el sistema, es decir tener un conocimiento de cómo está constituido el proceso a controlar, para de esta manera poder expresarlo en un lenguaje lingüístico entendible por el controlador difuso. Y a partir de conocimientos básicos de control se aplica una base de reglas que tendrá por objetivo simular el conocimiento humano en forma lingüística y entendible por el controlador.

Para el caso del levitador difuso primero se escoge dos entradas y una salida, también se define el rango de funcionamiento de las entradas y de su salida. Con los datos de entradas y salidas que se necesitan para el control difuso se procede analizar el funcionamiento del mismo. Se aprecia que, para que la esfera se mantenga levitando en el campo magnético es necesario aplicar tensión a la bobina, si la tensión es grande la esfera estará más cerca de la bobina, mientras que si la tensión es pequeña la esfera estará más lejos de la bobina. Con este análisis del funcionamiento del sistema se procede con las reglas que tendrán la siguiente forma:

- **SI** premisa **entonces** consecuencia

Ejemplo:

**Si** la corriente es grande **entonces** la posición de la esfera es pequeña.

Usualmente las entradas del sistema difuso son asociadas con la premisa y las salidas son asociadas con la consecuencia. Esta es la idea de cómo entender el proceso y posterior a esto se podrá ir definiendo cada una de las reglas.

Una vez detallada la forma en que se deben crear cada una de las reglas se procede a definir las **variables lingüísticas** como también los **valores lingüísticos** que servirán para formar una base de reglas difusas entendibles por el controlador.

### 3.2.1 Definición de variables lingüísticas y valores lingüísticos

Para el caso del levitador magnético se tiene dos entradas y una salida que son las llamadas **variables lingüísticas**:

- ERROR (ENTRADA)
- CAMBIO DE ERROR (ENTRADA)
- CONTROL (SALIDA)

Las dos primeras son las entradas del controlador y la última es la salida del controlador difuso.

Los **valores lingüísticos** escogidos para cada variable lingüística son:

- PEQUEÑO NEGATIVO (PN)
- CERO (CERO)
- PEQUEÑO POSITIVO (PP)

Los valores lingüísticos son los escogidos para cada variable lingüística y es así como también se debe definir el rango de cada variable lingüística o en otras palabras escoger el conjunto difuso de cada variable.

### 3.2.2 Rangos de variables lingüísticas

Si bien es cierto, el diseño del controlador difuso no requiere el modelo matemático del sistema a controlar, el conocimiento detallado del proceso facilita la definición de los universos de discurso y los rangos de los valores lingüísticos.

Para cada variable lingüística se debe definir los rangos en los que trabajarán para el sistema de levitación.

Para poder determinar el rango para las variables de **error, cambio de error y control** se empieza por realizar una simulación en el controlador PID que se encuentra en el software **ML_MAIN** del levitador magnético.

A continuación se explica cómo se empieza realizando el proceso:

Se inicia el programa MagLev, escribiendo en la ventana de trabajo de Matlab “ML_MAIN” y a continuación se presenta la ventana del software MagLev. Figura 3.6.

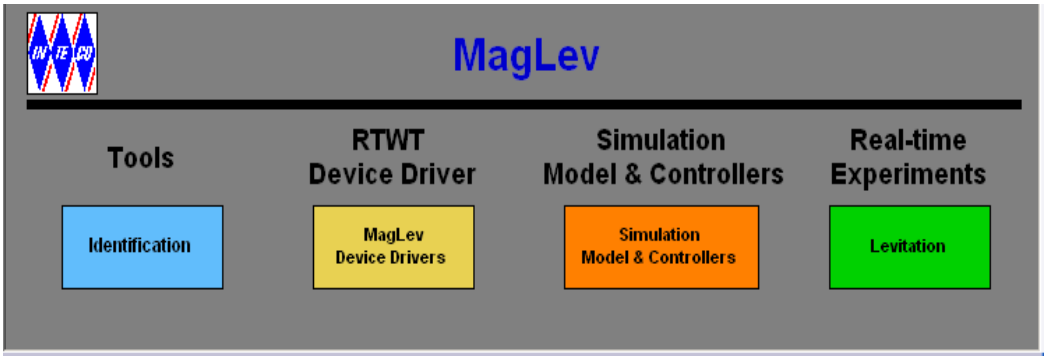



**Figura. 3.6. Ventana de Inicio MagLev**

La ventana de inicio del levitador magnético posee herramientas con las que se puede realizar un nuevo sistema de control y también simulación en tiempo real.

Ingresamos a la herramienta “Simulation Model & Controllers” donde se puede realizar la simulación del sistema con su modelamiento matemático y poder determinar el rango de las variables lingüísticas.

La ventana que se muestra en la figura 3.7, contiene controladores diseñados por el fabricante para el sistema de levitación magnética. Para encontrar el rango de trabajo de las variables lingüísticas ingresaremos al controlador PID.

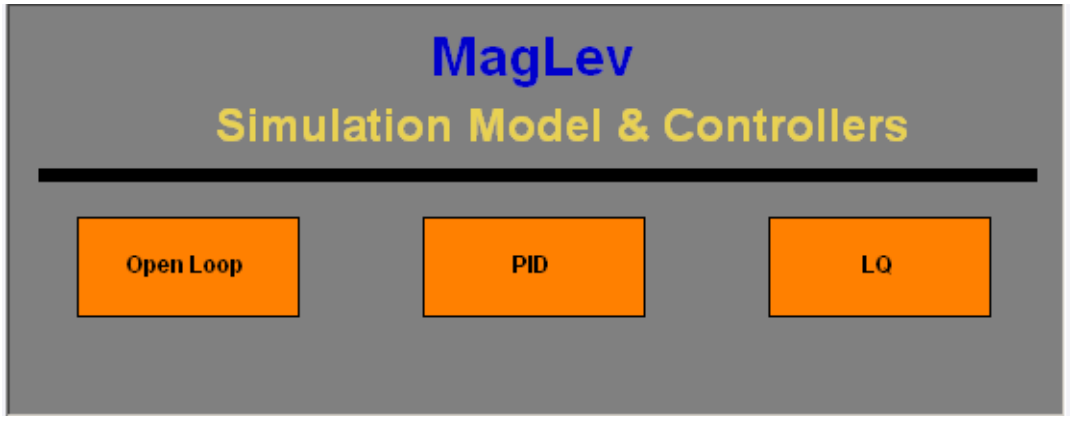



**Figura. 3.7. Tipos de controladores**

Dentro del controlador PID se puede determinar los rangos de las variables lingüísticas colocando un visualizador llamado SCOPE de las librerías de SIMULINK/MATLAB. Figura. 3.8.

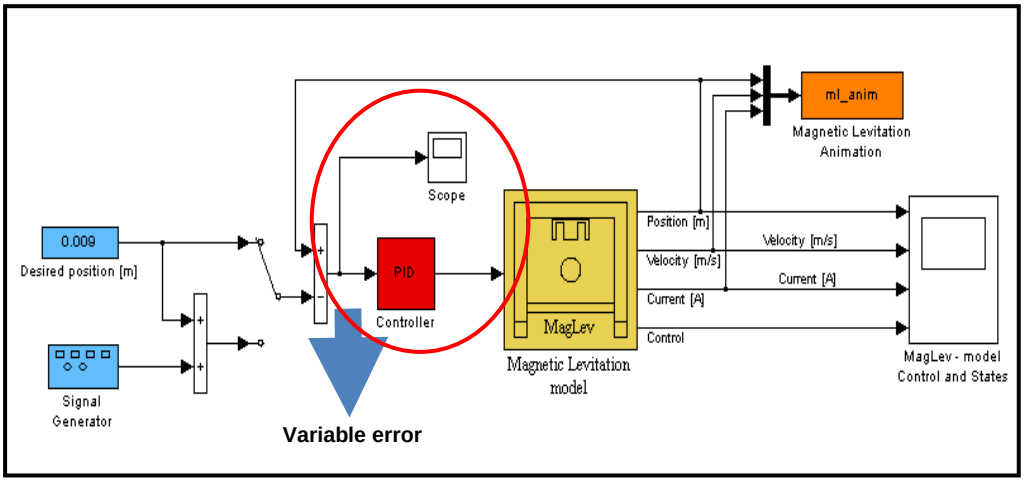



**Figura. 3.8. Visualización de variable error**

Se coloca un **Scope** con el que se puede visualizar el rango que tendrá la variable **ERROR** una vez que se realice la simulación. De igual forma se coloca visualizadores tanto para la derivada del error y la salida de control. Como ejemplo a continuación se presenta la figura. 3.9., de la variable Error.

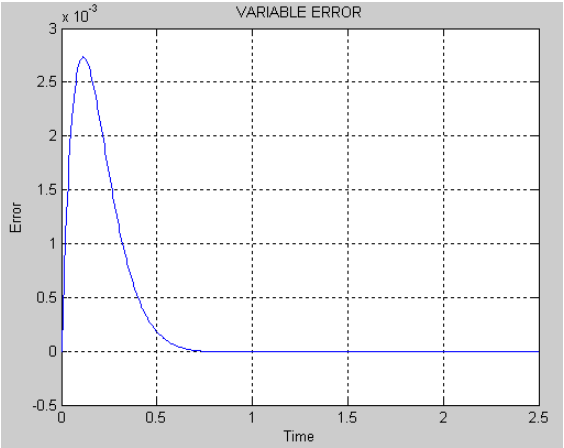



**Figura. 3.9. Rango de variación variable lingüística ERROR**

Como se puede apreciar en la Figura. 3.9., la variación que se tiene al simular el controlador PID que contiene el software, es aproximadamente de 0.025 m., con lo que se define el rango de 0.02 para el caso de la variable ERROR.

De esta forma se puede determinar el rango de cada variable lingüística y poder empezar a formar el controlador. Los rangos de las variables que se utilizaron para el diseño del controlador difuso PD son:

El rango para la variable lingüística **ERROR** se define entre -0.02 a 0.02 metros.

El rango para la variable lingüística **CAMBIO DE ERROR** se define entre -0.01 a 0.01 metros.

El rango para la variable lingüística **CONTROL** se define entre -1 a 1.

### 3.2.3 Funciones de membrecía

Una vez definido el rango de cada variable lingüística se debe escoger el tipo de función de membrecía que se asignará a cada valor lingüístico, para el proyecto se eligió funciones de membrecía triangulares para los valores lingüísticos medios, mientras que para las fronteras se escogió funciones de membrecía exponenciales.

Se puede observar las variables lingüísticas con los valores lingüísticos y su rango ya previamente obtenido para la variable lingüística error. Figura 3.10.

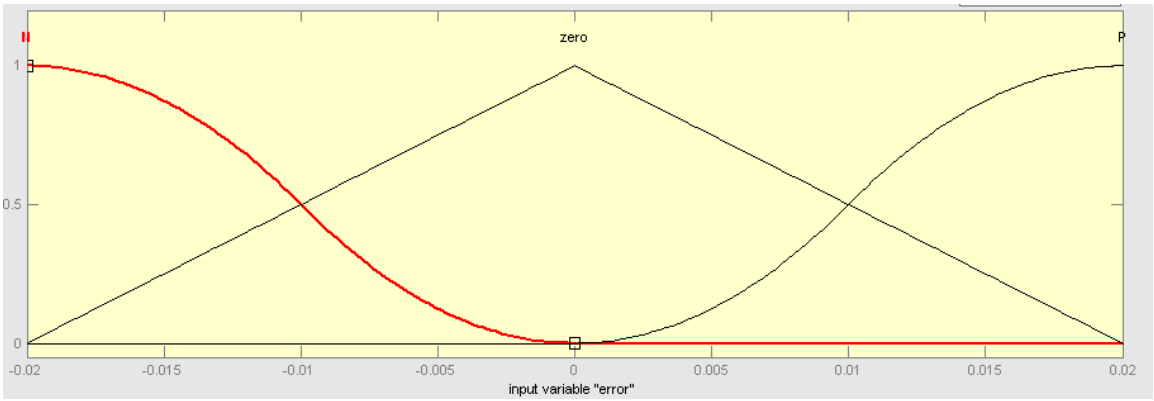



**Figura. 3.10. Funciones de Membrecía**

Es así que se define para el **cambio de error**, **error y variable de control** el rango de las variables y valores con funciones de membrecía.

Lo siguiente que se realizó fue armar la base de reglas para el sistema de levitación magnética.

### 3.2.4 Base de Reglas

La base de reglas se encuentra definido por las variables lingüísticas y los valores lingüísticos tratando de emular el conocimiento humano de cómo debe ser controlado el sistema de levitación magnética.

Se define nueve reglas que son las necesarias para realizar el control del levitador magnético. Se presentan a continuación la siguiente base de reglas.

- **Base de reglas controlador difuso PD.**

Se presenta nueve reglas que son necesarias para realizar el control del sistema de levitación magnética.

| | | | | | | |
|---|---|---|---|---|---|---|
| **1** | Si | Error es Pequeño negativo | Y | Cambio de error es Pequeño negativo | Entonces | Control es Pequeño negativo |
| **2** | Si | Error es Pequeño negativo | Y | Cambio de error es Cero | Entonces | Control es Pequeño negativo |
| **3** | Si | Error es Cero | Y | Cambio de error es Pequeño negativo | Entonces | Control Pequeño negativo |
| **4** | Si | Error es Cero | Y | Cambio de Cero | Entonces | Control es Cero |
| **5** | Si | Error es Cero | Y | Cambio de error es Pequeño positivo | Entonces | Control es Pequeño positivo |
| **6** | Si | Error es Pequeño positivo | Y | Cambio de error es Cero | Entonces | Control es Pequeño positivo |
| **7** | Si | Error es Pequeño positivo | Y | Cambio de error es Pequeño positivo | Entonces | Control es Pequeño positivo |
| **8** | Si | Error es Pequeño positivo | Y | Cambio de error es Pequeño negativo | Entonces | Control es Cero |
| **9** | Si | Error es Pequeño | Y | Cambio de error es | Entonces | Control es Cero |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | negativo | | Pequeño positivo | | |

**Tabla. 3.2. Base de nueve reglas para controlador difuso PD**

Como se puede apreciar en la base de reglas, se encuentran las variables lingüísticas y valores lingüísticos que fueron definidos. También se utiliza el conector “y”, para unir las premisas.

### 3.2.5 Sistema de Inferencia difusa o herramienta de lógica difusa

Es una herramienta con la que se puede crear el controlador de lógica difusa para el levitador. Dentro de esta herramienta se encuentran bloques que deben ser llenados por el usuario. Además de esto la herramienta permite crear archivos que contienen todo el diseño del controlador difuso. A esto se refiere con:

- Funciones de Membrecía.
- Editor de reglas.
- Visor de reglas.
- Visor de Superficie.

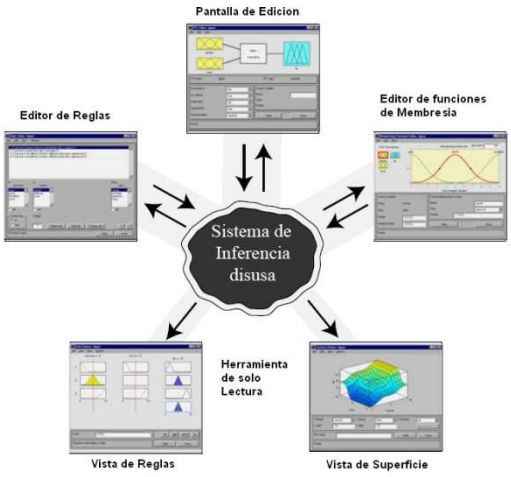



**Figura. 3.11. Sistema de Inferencia Difusa**

Para iniciar con el diseño del controlador difuso con la herramienta “fis editor”, se empieza ingresando el número de entradas y también el número de salidas.

Para el controlador difuso PD se tiene dos entradas y una salida, error, cambio de error y control. Figura 3.12.

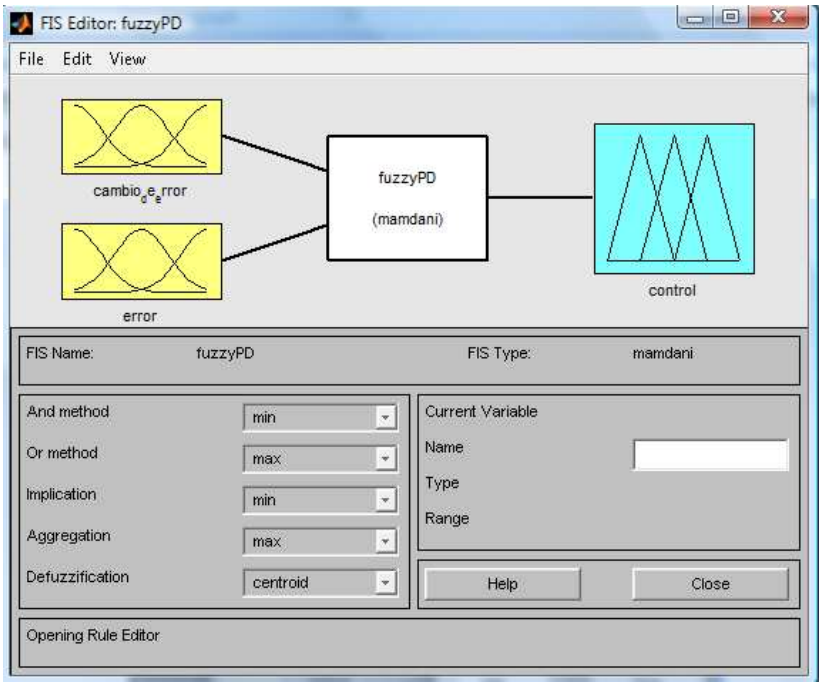



**Figura. 3.12. Herramienta FIS EDITOR**

La ventana de la herramienta se utiliza para crear las entradas y salidas necesarias para el sistema a controlar, cabe recalcar que si se utiliza muchas entradas dependerá de la capacidad de procesamiento computacional.

Una vez ingresado las entradas y salidas a la herramienta se procede a formar la base de reglas con los valores y variables lingüísticas. Figura. 3.13.

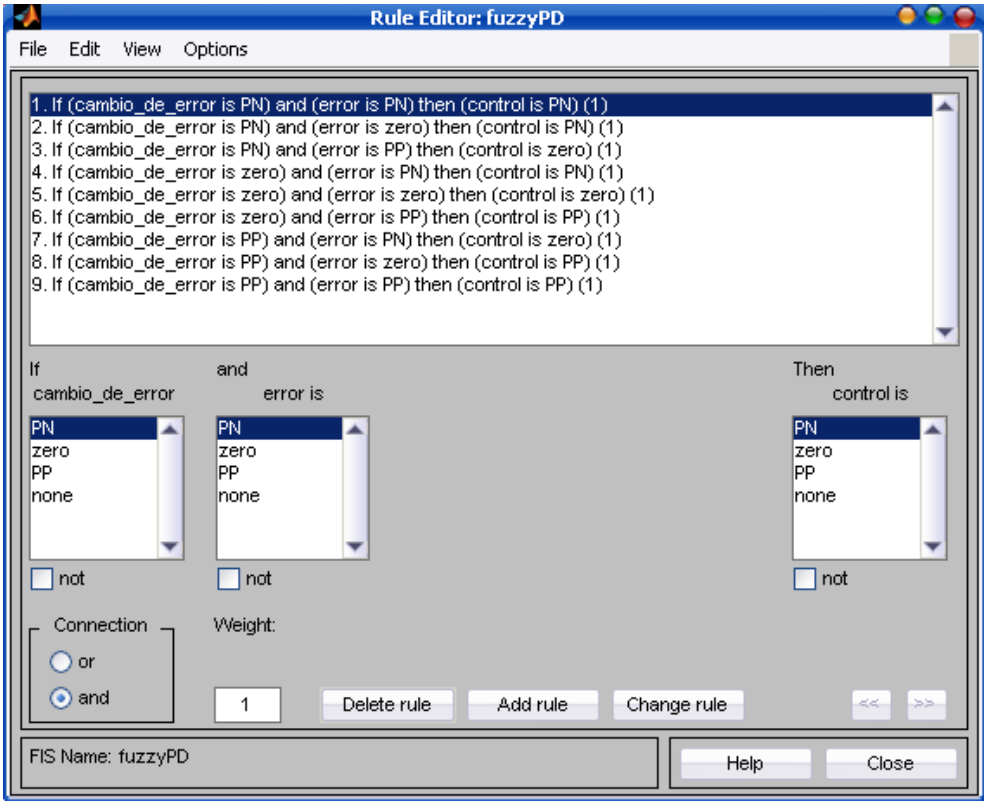



**Figura. 3.13. Editor de Reglas**

Como se puede observar en la figura 3.13, las premisas están conectadas con el operador “Y”, además de esto se encuentran los valores lingüísticos; “pequeño negativo”, “zero” y “pequeño positivo”

Una vez concluido con todo el diseño en la herramienta “Fis Editor”, se debe guardar el controlador difuso. Para esto se debe dar un nombre al archivo de extensión .FIS.

### 3.2.6 Implementación de controlador proporcional derivativo difuso PD en SIMULINK

La implementación del controlador difuso en la herramienta Simulink es necesaria para realizar la simulación de los controladores difusos creados con la herramienta “Fis editor”.

Se utiliza el modelo matemático de la planta para proceder a implementar el bloque del controlador difuso proporcional derivativo de la siguiente forma:

- **Primer Paso**

Se crea una nueva ventana con el modelo matemático y se coloca el bloque del controlador difuso. Figura. 3.14.

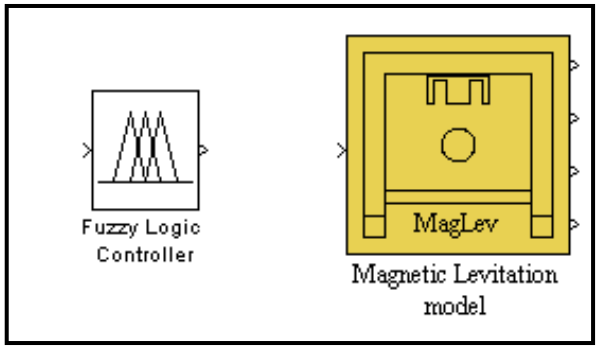



**Figura. 3.14. Bloque Difuso y Modelo Matemático**

- **Segundo Paso**

Se procede a colocar un nombre al bloque del controlador difuso para poder cargar el archivo de extensión .FIS que ya se tiene creado previamente.

Con el siguiente comando se carga el archivo de extensión **.fis** al bloque "Fuzzy Logic Controller" de Simulink.

En el entorno de trabajo de Matlab se debe escribir:

**Control_PD=readfis ('fuzzyPD')**

Una vez que se ejecute el comando se visualizará la confirmación que el archivo de extensión ".fis" ha sido cargado con éxito. Figura 3.15.

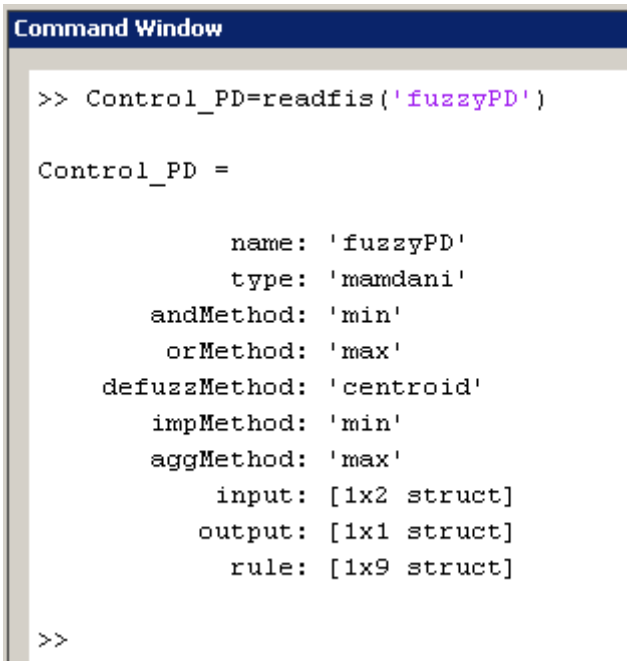



**Figura. 3.15. Ventana de trabajo Matlab y resultados**

- **Tercer Paso**

Se debe incluir los bloques de ganancias que modificarán los universos de discursos de las variables lingüísticas y también incluir dentro del diseño la derivada del error para tener las dos entradas necesarias para el bloque difuso en Simulink.

Tomando en cuenta lo mencionado se presenta el diseño del sistema de levitación magnética con el controlador difuso y los bloques respectivos. Figura. 3.16.

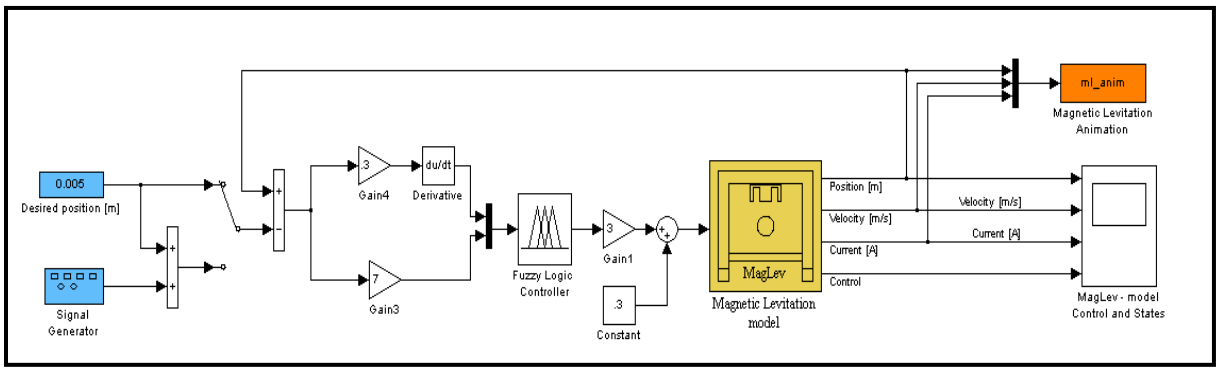



**Figura. 3.16. Controlador difuso y sus bloques respectivos.**

### 3.2.7 Explicación de diseño

A continuación se explica cada uno de los bloques de simulación que han sido agregados al diseño.

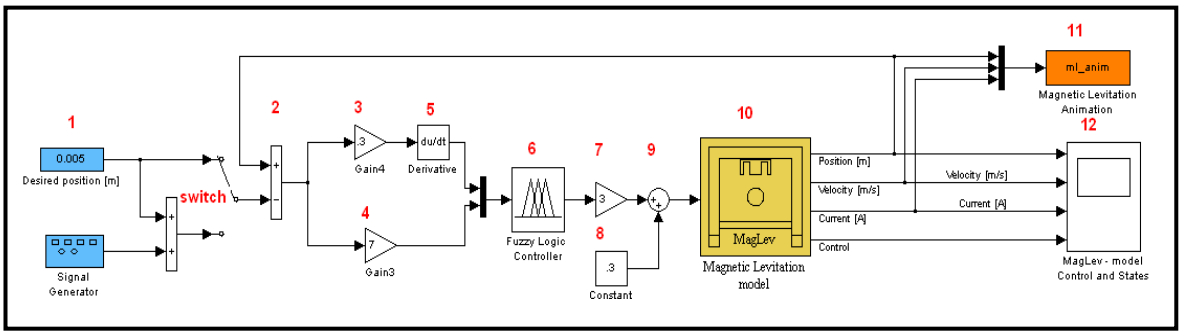



**Figura. 3.17. Controlador difuso PD y sus bloques respectivos**

Lo bloques han sido enumerados en la Figura. 3.17., para realizar la explicación de cada uno de ellos:

**1.** Bloques de SetPoint, en metros para obtener la posición de la salida, es decir en los bloques se coloca un valor deseado en el que se quiere estabilizar la posición de la esfera.

**2.** Bloque que realiza la resta entre el valor de posición obtenido a la salida con el valor deseado, es decir que a la salida de este bloque de substracción se va a obtener el error que se obtuvo al poner un valor deseado con el valor real que se obtiene en la simulación.

**3.** Bloque de ganancia con el que se puede agrandar o hacer pequeño el universo de discurso de la variable cambio de error de posición, el valor que se escoge en este caso es de: **0.3**, realizando pruebas se llego a obtener el mejor desempeño con ese valor, en este caso el valor de 0.3 influye en que el universo de discurso se hace más grande, es decir $\frac{1}{g1} \times universo\ de\ discurso$, y de esta manera el universo que se encuentra entre -0.01 a 0.01 se hace más grande entre los valores de -0.033 a 0.033. Este bloque de ganancia si se lo asemeja al controlador PID clásico es el llamado ganancia diferencial.

**4.** Bloque de ganancia proporcional, cumple el mismo propósito que la ganancia del bloque número tres, y el valor de ganancia proporcional es de **7.**

**5.** Bloque diferenciador de la variable error, es decir la función del bloque es derivar la señal error y así obtener la variable cambio de error para obtener la entrada del controlador difuso PD.

**6.** Bloque controlador difuso, en este bloque se encuentra cargado el archivo de extensión **.fis** el cual contiene toda la base de reglas, conjuntos difusos, variables, valores difusos.

**7.** Bloque de ganancia, actúa sobre la salida del controlador y para este caso la salida del controlador se hace más rápida y más fuerte, y el valor del bloque es de **3.**

**8.** Valor de offset que se agrega para darle a la simulación el efecto de la esfera que se mantenga en valores positivos

**9.** Bloque que realiza la adición del offset a la salida del controlador.

**10.** Bloque de modelo matemático en el cual esta especificado todos los parámetros matemáticamente que se asemejan al dispositivo en el campo real es decir nos brinda la capacidad de simular en software lo que se tiene en la realidad del levitador magnético.

**11.** Bloque de animación, permite establecer la simulación de la esfera levitando dentro del levitador gráficamente, es decir nos permite ver la animación de la esfera levitando.

**12.** Bloque de visualización de señales, permite visualizar el estado de las señales y presentarlas en un grafico, es decir aquí se presentan las señales de la posición, velocidad, corriente y señal de control para poder visualizar sus estados.

Los bloques que han sido agregados, como el bloque de conmutación al inicio del grafico 3.17. Es utilizado para conmutar del bloque de SetPoint constante a un bloque de SetPoint variable.

### 3.2.8 Simulación del controlador difuso proporcional derivativo PD.

En el tema anterior se realiza todo el proceso de diseño del controlador difuso PD con la herramienta **Fis Editor** y **Simulink.** A continuación se presenta la simulación del controlador en el entorno Simulink.

Lo primero que se debe realizar para la simulación del sistema, es configurar correctamente los parámetros de simulación.

Esto se realiza en la ventana “SIMULATION” figura. 3.18. En esta ventana se debe colocar el tiempo de simulación. La ventana debe quedar configurada de la siguiente manera.

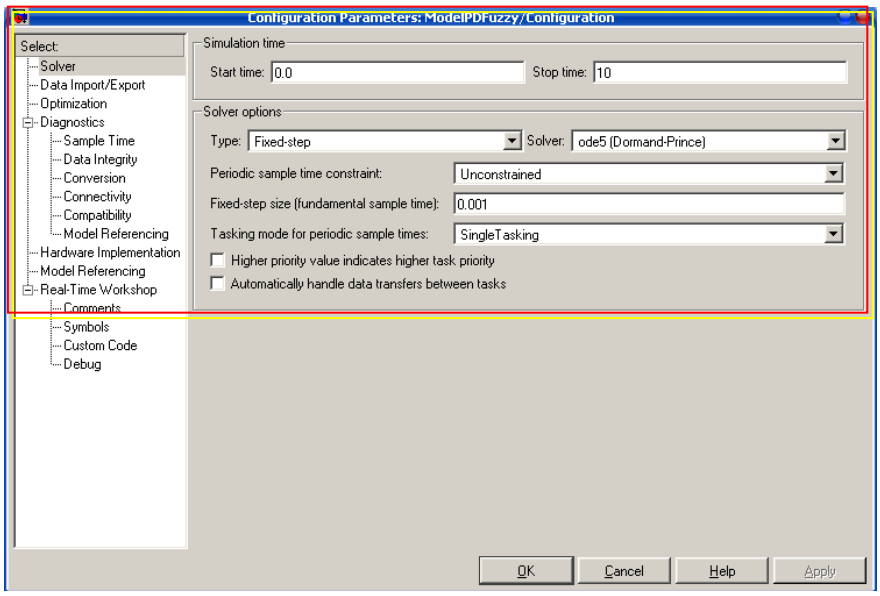



**Figura. 3.18. Ventana de Configuración parámetros de simulación**

Para empezar con la simulación se debe cargar los datos del archivo de extensión **.fis** al bloque que se encuentra en el entorno de SIMULINK. Se debe aplicar el comando en la ventana de trabajo de Matlab como se detalla a continuación:

**Control_PD = redfis ('fuzzyPD')**

Una vez que se carga el archivo de extensión **fis**, se procede a realizar la simulación en el botón "**START SIMULATION**". Figura. 3.19.

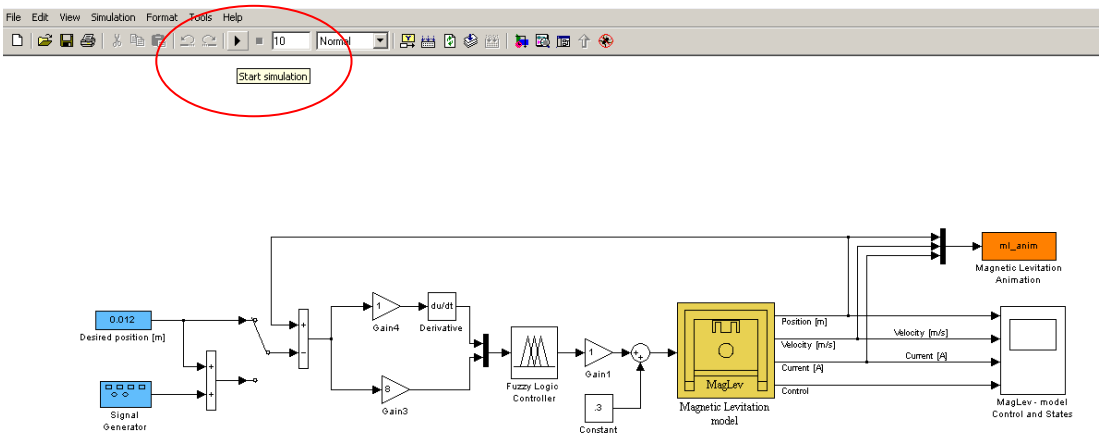



**Figura. 3.19. Inicio de simulación**

A continuación Simulink empezará a presentar los resultados de simulación como se observa en la siguiente figura 3.20.

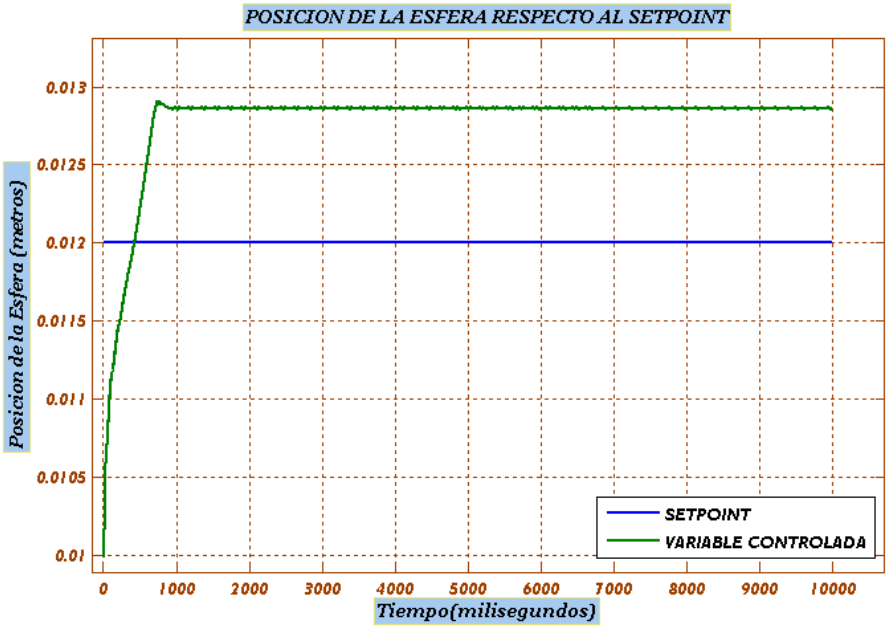



**Figura. 3.20. Posición Esfera controlador difuso PD**

**Gd=1, Gp=8, Gs=1, Constante=0.3**

Como se puede observar en la figura 3.20, la posición de la esfera se ubica en 0.0128 m., mientras que el valor deseado se encuentra en 0.012 m., teniendo un error de 0.0008 m., el cual se puede concluir como un error despreciable.

La simulación presentada en la Figura. 3.20., sobre la posición de la esfera es realizada con 9 reglas y fueron realizados con los siguientes valores de ganancias:

- Ganancia derivativa (Gd.) = 1
- Ganancia proporcional (Gp.) = 8
- Ganancia de salida (Gs.) = 1
- Constante = 0.3

La variable controlada o señal de salida que se obtiene respecto al SetPoint deseado tiene un error despreciable llamado error en estado estacionario. Figura 3.21.

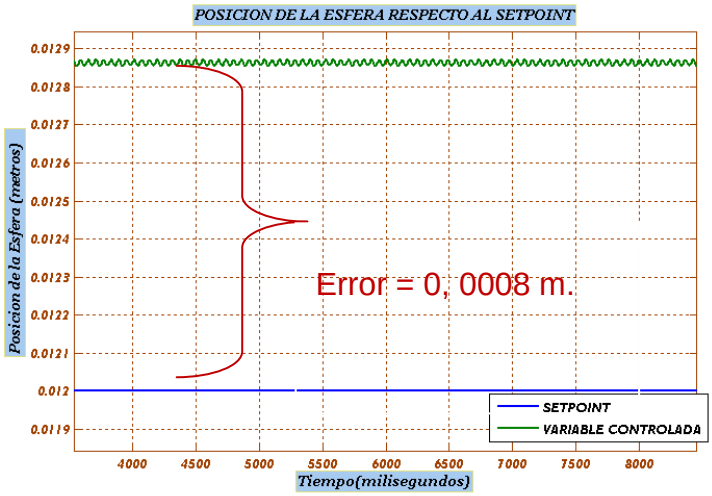



**Figura. 3.21. Error en estado estacionario**

Como se puede observar en la figura. 3.21, la variable controlada se encuentra por encima del valor deseado SetPoint y se tiene un error de 0.0008 metros que es igual a 0.8 milímetros. Este error se puede atribuir debido a que el levitador tiene características de ser inestable y por lo que su control se hace complejo. Se espera que al realizar un controlador PID difuso se obtenga mejores resultados con respecto al error en estado estacionario.

El tiempo de establecimiento se refiere al tiempo en que demora el sistema en mantener un estado constante, en este caso el tiempo que se obtiene es de 800 milisegundos y se puede observar que no posee un sobre impulso que afecte al sistema sino que se ubica rápidamente en su valor deseado sin menospreciar el error en estado estable. Figura 3.22.

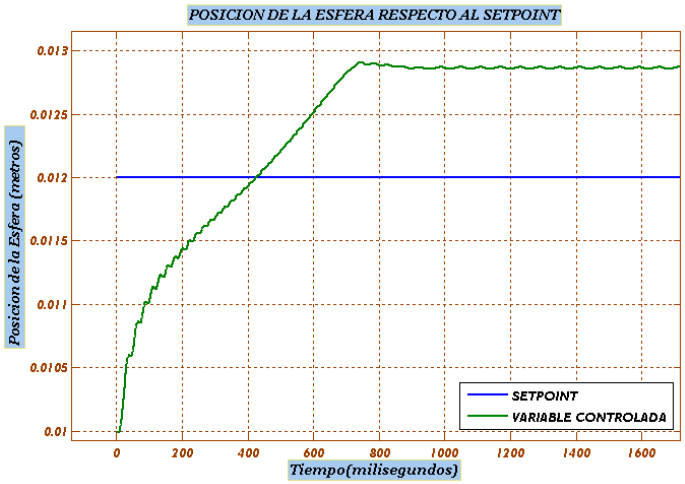



**Figura. 3.22. Tiempo de Establecimiento**

Las graficas obtenidas sobre la velocidad, corriente y control se muestran a continuación:

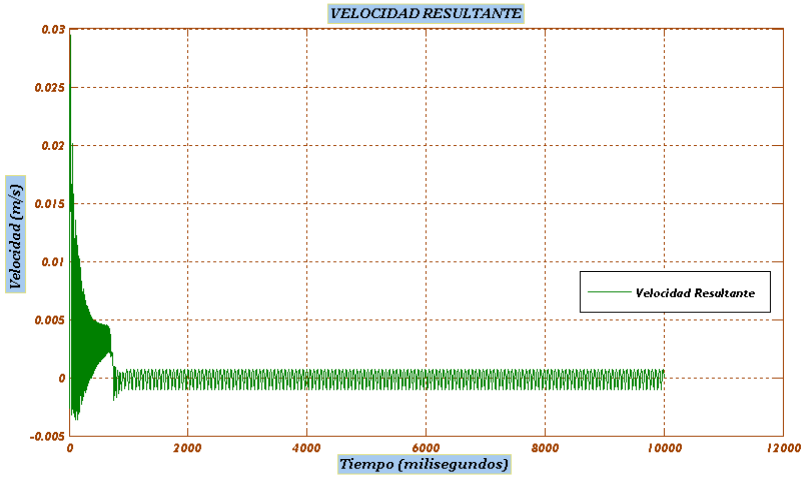



**Figura. 3.23. Velocidad de la esfera levitando por 10seg. Controlador difuso PD**

Se puede apreciar en la figura de la velocidad, que se tiene una velocidad con oscilaciones de alta frecuencia al inicio de la simulación para después ubicarse en un valor constante igual a cero una vez que la esfera se ubica en el valor deseado.

A continuación se presenta la grafica de respuesta de la corriente:

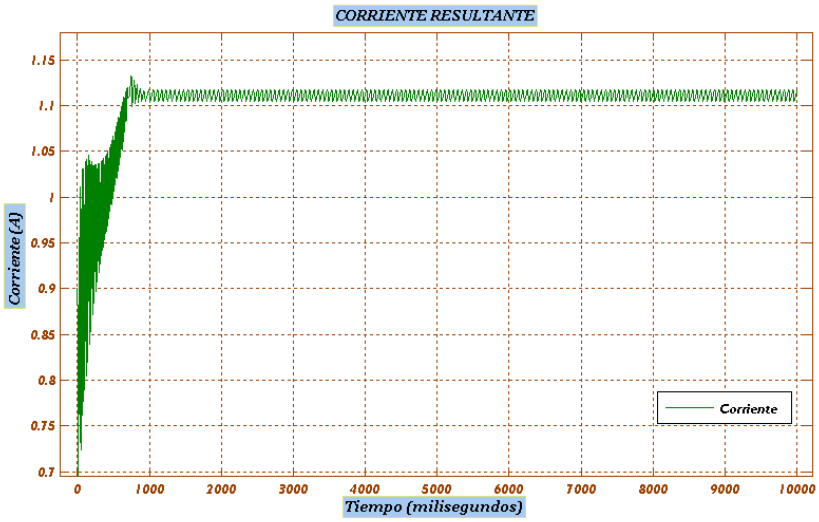



**Figura. 3.24. Corriente de la esfera levitando por 10seg. Controlador difuso PD**

Se puede observar de la figura 3.24, que se necesita un valor de corriente de 1.2 [A] para que la esfera se ubique en la posición deseada.

De esta manera es como fue realizado la simulación del controlador difuso proporcional derivativo PD para el sistema de levitación magnética.

Se espera que el diseño con controladores difusos PID sea más exacto y los resultados que se obtengan sean mejores.

## 3.3 DISEÑO DEL CONTROLADOR DIFUSO PROPORCIONAL INTEGRAL DERIVATIVO PID

El diseño del controlador difuso PID, fue realizado tomando en cuenta el diseño principal que fue el controlador difuso PD y se agregó el bloque del integrador que hace falta para crear el controlador difuso PID. Figura. 3.25.

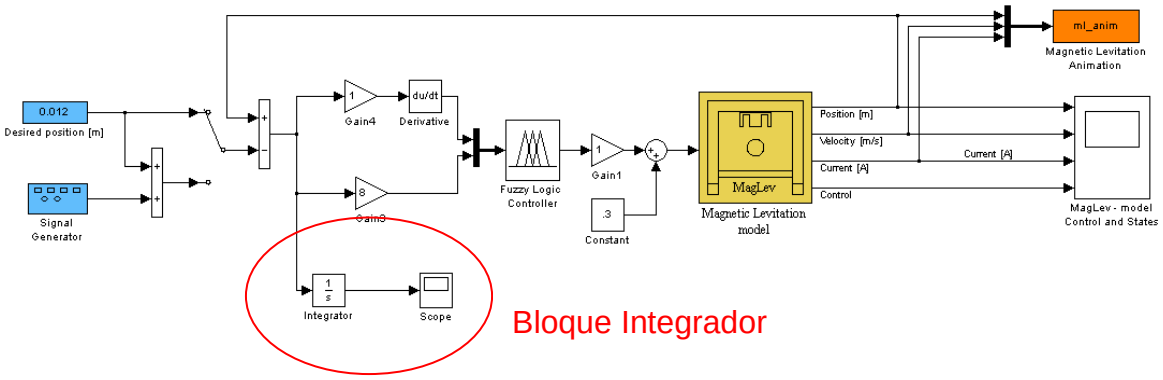



**Figura. 3.25. Bloque integrador en el sistema de levitación magnética**

En la figura. 3.25., se observa que dentro del controlador difuso PD se colocó, un bloque integrador que integrará a la variable error para formar el controlador difuso PID.

De igual manera que se encontró los rangos difusos de las variables error y cambio de error, se ubica un visualizador a la salida del bloque integrador para obtener el rango difuso de esta variable. Teniendo en cuenta esto se define a la variable **Integral** entre los rangos de **-0.06 a 0.06 [m.s]** para poder trabajar con la parte integral o en otras palabras la acumulación del error. A continuación se continúa con el diseño del controlador difuso PID en la herramienta Simulink.

### 3.3.1 Diseño del controlador PID difuso con 18 reglas

Dentro del diseño del controlador difuso PID, se encuentran los bloques de la integral del error y la derivada del error. Las ganancias son diferentes en cuanto al controlador difuso PD. Figura 3.26.

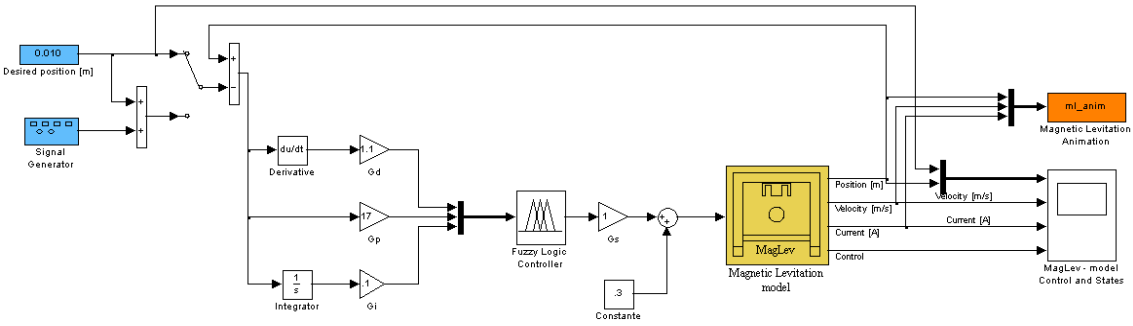



**Figura. 3.26. Controlador difuso PID 18 reglas y sus respectivos bloques**

#### 3.3.1.1 Rangos de variables lingüísticas

El rango de las variables lingüísticas para el controlador difuso PID son:

- El rango para la variable lingüística **ERROR** se encuentra entre -0.03 a 0.03 metros.
- El rango para la variable lingüística **CAMBIO DE ERROR** se encuentra entre -0.01 a 0.01 metros.
- El rango para la variable lingüística **INTEGRAL** se encuentra entre -0.04 a 0.04 m.s.
- El rango para la variable lingüística **CONTROL** se encuentra entre -1 a 1.

### 3.3.1.2 Funciones de membrecía

Las funciones de membrecía que fueron utilizadas para el diseño del controlador difuso PID se presentan a continuación:

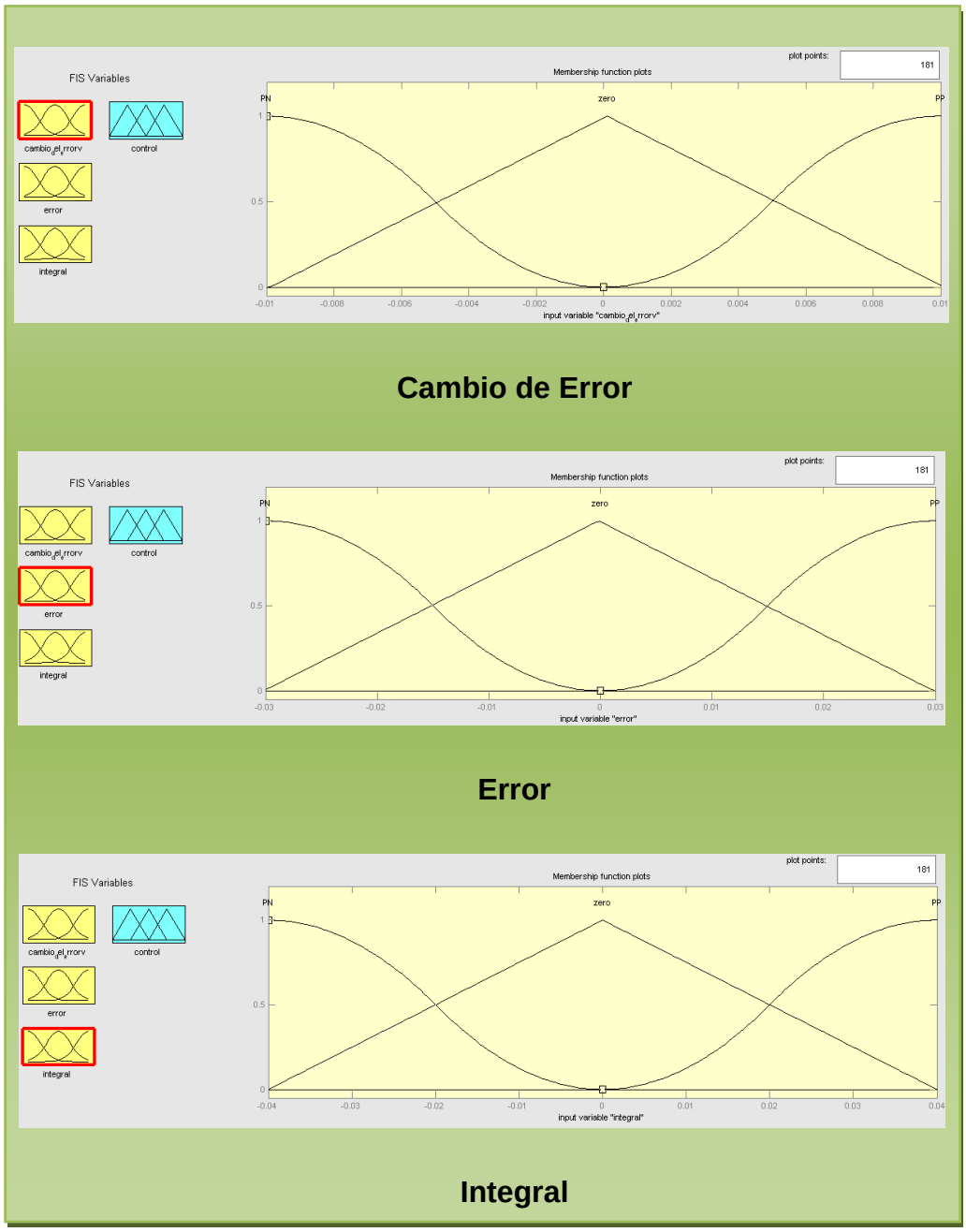



**Figura. 3.27. Funciones de membrecía variables de entradas para controlador 18 reglas**

En la figura. 3.27., se puede apreciar al editor de funciones de membrecía de las variables “error”, “cambio de error”, “integral”, que se encuentran definidas con los rangos que se mencionaron en el punto 3.3.1.1., también se puede apreciar que las funciones de membrecía son triangulares en el centro, mientras que para las fronteras son exponenciales.

La salida del controlador difuso PID se encuentra definido con funciones de membrecía triangulares. Figura 3.28.

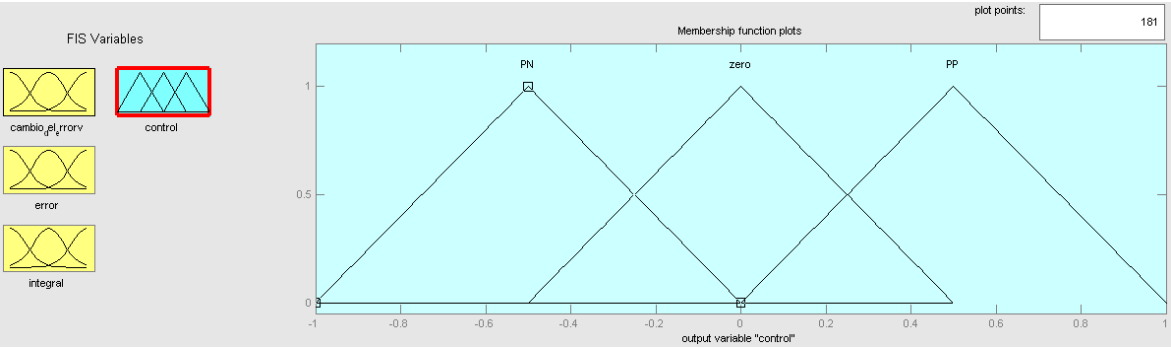



**Figura. 3.28. Funciones de membrecía variable de salida para controlador de 18 Reglas.**

En la figura. 3.28., se puede apreciar que las variables lingüísticas, pequeño negativo (PN) y Pequeño Positivo (PP) se encuentra dentro del rango de -1 a 0 y 1 a 0 respectivamente para mantener el rango de salida dentro de lo que se había definido con anterioridad que fue desde -1 a 1 la variable CONTROL en el diseño del controlador difuso PD.

Hay que notar que las entradas tienen en las fronteras funciones exponenciales abiertas, mientras que para la salida todas son cerradas. Esto es muy importante para el diseño del controlador difuso.

### 3.3.1.3 Base de reglas

La base de reglas que se utilizó es la siguiente:

| | | | | | | |
|---|---|---|---|---|---|---|
| **1** | Si | Error es Pequeño negativo | Y | Cambio de error es Pequeño negativo | Entonces | Control es Pequeño negativo |
| **2** | Si | Error es Pequeño negativo | Y | Cambio de error es Cero | Entonces | Control es Pequeño negativo |
| **3** | Si | Error es Cero | Y | Cambio de error es Pequeño negativo | Entonces | Control Pequeño negativo |
| **4** | Si | Error es Cero | Y | Cambio de Cero | Entonces | Control es Cero |
| **5** | Si | Error es Cero | Y | Cambio de error es Pequeño positivo | Entonces | Control es Pequeño positivo |
| **6** | Si | Error es Pequeño positivo | Y | Cambio de error es Cero | Entonces | Control es Pequeño positivo |
| **7** | Si | Error es Pequeño positivo | Y | Cambio de error es Pequeño positivo | Entonces | Control es Pequeño positivo |
| **8** | Si | Error es Pequeño positivo | Y | Cambio de error es Pequeño negativo | Entonces | Control es Cero |
| **9** | Si | Error es Pequeño negativo | Y | Cambio de error es Pequeño positivo | Entonces | Control es Cero |
| **10** | Si | Error es Pequeño negativo | Y | Integral es Pequeño negativo | Entonces | Control es pequeño negativo |
| **11** | Si | Error es Cero | Y | Integral es Pequeño negativo | Entonces | Control es pequeño negativo |
| **12** | Si | Error es pequeño negativo | Y | Integral es Cero | Entonces | Control es Pequeño negativo |
| **13** | Si | Error es cero | Y | Integral es Cero | Entonces | Control es Cero |

| | | | | | | |
|---|---|---|---|---|---|---|
| **14** | Si | Error es pequeño positivo | Y | Integral es Cero | Entonces | Control es pequeño positivo |
| **15** | Si | Error es cero | Y | Integral es Pequeño positivo | Entonces | Control es pequeño positivo |
| **16** | Si | Error es pequeño positivo | Y | Integral es pequeño positivo | Entonces | Control es pequeño positivo |
| **17** | Si | Error es pequeño negativo | Y | Integral es Pequeño positivo | Entonces | Control es cero |
| **18** | Si | Error es Pequeño Positivo | Y | Integral es pequeño negativo | Entonces | Control es cero |

**Tabla. 3.3. Base de 18 reglas**

### 3.3.1.4 Implementación de controlador proporcional integral derivativo PID de 18 reglas en SIMULINK

Dentro de la herramienta Simulink de Matlab se procede a realizar la implementación de todo el sistema con sus bloques respectivos como se presenta en la siguiente figura 3.29.

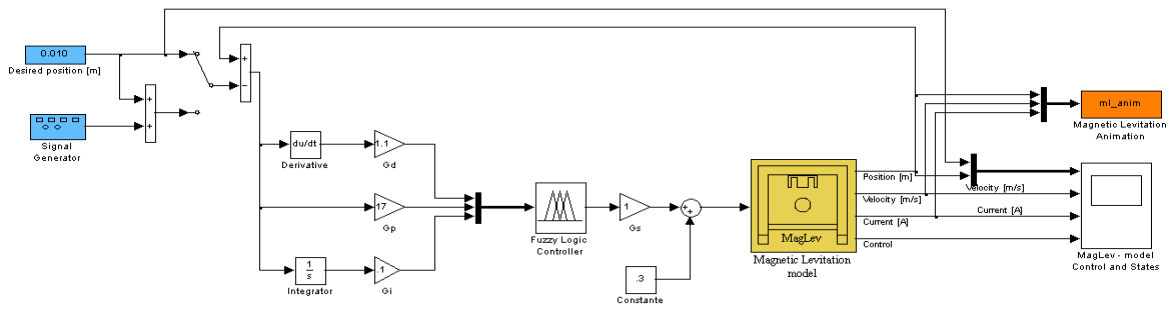



**Figura. 3.29. Controlador PID de 18 reglas**

Hay que tomar en cuenta que los valores de ganancias que se establecen en este diseño son diferentes en cuanto a las demás ganancias de los anteriores controladores difusos, por razones de diseño.

### 3.3.1.5 Simulación del controlador difuso proporcional integral derivativo PID de 18 reglas

Realizando la simulación del controlador difuso PID de 18 reglas se tiene la siguiente gráfica de respuesta.

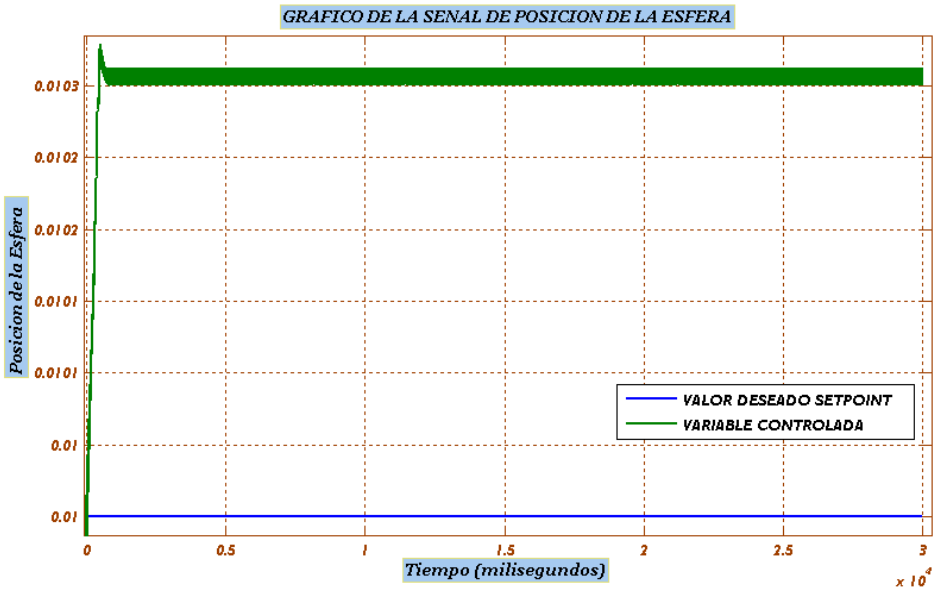



**Figura. 3.30. Posición de la esfera con 18 reglas**

**Gd=1.1, Gp=17, Gi=0.1, Gs=1, Constante=0.3**

Como se puede observar en la figura 3.30, la salida de posición se ubica con un error de 0.3 milímetros con respecto a la variable deseada. También se observa que se tiene oscilaciones de alta frecuencia cuando la esfera llega a la posición de 0.0103 m.

La simulación presentada en la figura. 3.30., sobre la posición de la esfera es realizada con 18 reglas. Para los valores de ganancias utilizados son los siguientes:

- Ganancia derivativa (Gd.) = 1.1
- Ganancia proporcional (Gp.) = 17
- Ganancia Integral (Gi.) = .1
- Ganancia de salida (Gs.) = 1
- Constante = 0.3

Una vez simulado el controlador con 18 reglas se tiene la opción de mejorar el diseño en cuanto a la variable de salida.

Se realizará un nuevo diseño con más valores lingüísticos en la “variable control”.

### 3.3.2 Diseño del controlador PID difuso con 27 reglas

Se diseña un nuevo controlador difuso PID buscando mejorar el error en estado estacionario que se tiene en el anterior controlador difuso PID.

El diseño del controlador difuso PID de 18 reglas es tomado como base para el nuevo diseño del controlador, tomando en cuenta las variables lingüísticas de las entradas del controlador difuso PID de 18 reglas que serán las mismas para el nuevo controlador difuso PID, el nuevo controlador difuso PID tendrá su salida

con mas valores lingüísticos tratando de tener una salida más exacta, lo cual no será igual que el anterior controlador difuso de 18 reglas.

### 3.3.2.1 Definición de variables lingüísticas y valores lingüísticos

Se definen las siguientes **variables lingüísticas** para el nuevo controlador difuso PID:

- ERROR (ENTRADA)
- CAMBIO DE ERROR (ENTRADA)
- INTEGRAL (ENTRADA)
- CONTROL (SALIDA)

Como se puede apreciar en comparación con el controlador PID difuso de 18 reglas, las variables lingüísticas son las mismas.

Los **valores lingüísticos** escogidos para las entradas **Error, Cambio de error e Integral** son las siguientes:

- NEGATIVO (N)
- CERO (ZERO)
- POSITIVO (P)

La razón por la que no se modifica los valores lingüísticos en las entradas es porque se busca que la salida sea más exacta en sus conjuntos difusos. Es por esta razón que la salida ha sido definida de la siguiente manera:

A continuación se presenta los **valores lingüísticos** escogidos para la salida **Control:**

- GRANDE NEGATIVO (GN)
- NEGATIVO (N)
- PEQUEÑO NEGATIVO (PN)
- CERO (ZERO)
- PEQUEÑO POSITIVO (PP)
- POSITIVO (P)
- GRANDE POSITIVO (GP)

Los valores se escogieron para poder tener una salida más exacta, y de esta manera hacer actuar a la integral para corregir el error en estado estable que se tiene en el controlador difuso de 18 reglas,

Se han definido los rangos de las variables lingüísticas de la misma forma que se obtuvo en el diseño del controlador difuso PD.

En este caso el rango para la variable lingüística **Integral** se obtuvo con el mismo proceso de colocar un visualizador a la salida del bloque de la integral. Los rangos de las variables lingüísticas que fueron utilizadas para el nuevo controlador difuso se presentan a continuación.

### 3.3.2.2 Rangos de variables lingüísticas

Los rangos de las variables lingüísticas difusos para este diseño fueron los siguientes:

- El rango para la variable lingüística **ERROR** se encuentra entre -0.02 a 0.02 metros.

- El rango para la variable lingüística **CAMBIO DE ERROR** se encuentra entre -0.01 a 0.01 metros.

- El rango para la variable lingüística **INTEGRAL** se encuentra entre -0.06 a 0.06 m.s.

- El rango para la variable lingüística **CONTROL** se encuentra entre -1 a 1.

### 3.3.2.3 Funciones de membrecía

Las funciones de membrecía que son las entradas, “error”, “cambio de error” y “integral”, han sido propuestas en la figura 3.3.1., tomando en cuenta que los rangos difusos de las variables han sido cambiados.

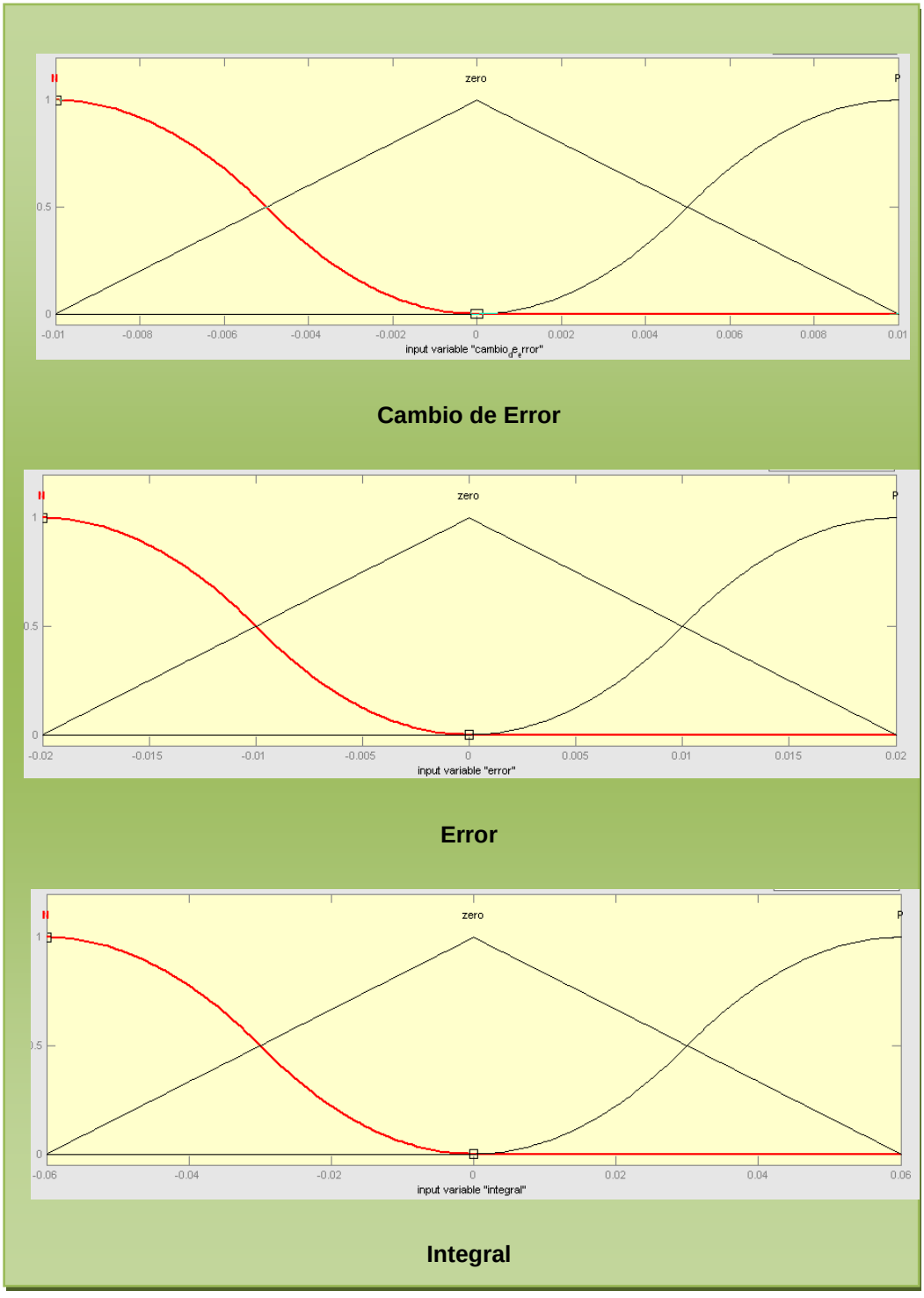



**Figura. 3.31. Funciones de membrecía Entrada para controlador difuso PID.**

Como se puede observar en la figura 3.3.1, se tiene definido las variables lingüísticas para el nuevo controlador difuso PID. Cabe notar que los rangos de las variables son diferentes a los rangos de las variables en el controlador difuso con 18 reglas.

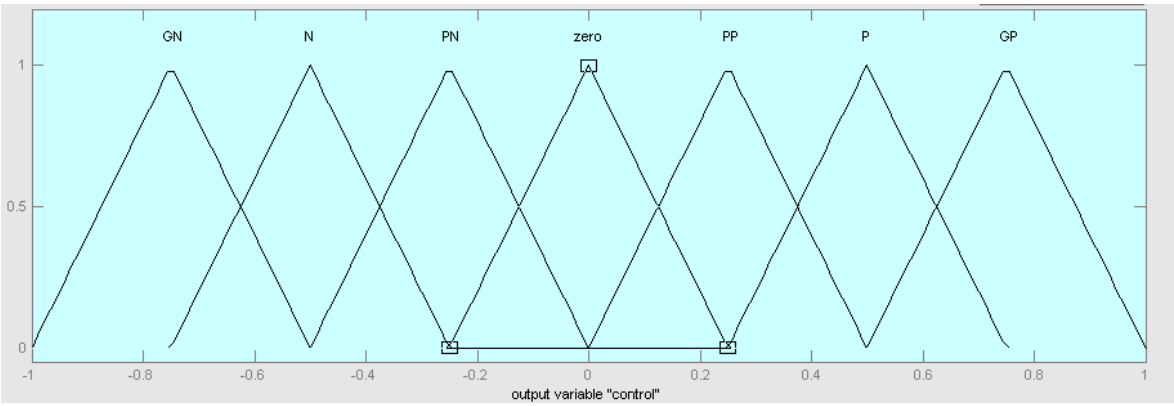



**Figura. 3.32. Funciones de membrecía para la salida del controlador difuso PID.**

Como se puede apreciar en la figura. 3.32., se tiene el universo de discurso del mismo tamaño que el de la salida del controlador difuso PID que es de -1 a 1 unidades, pero en este caso el universo se lo dividió en porciones o valores lingüísticos más pequeños para poder tener un universo de discurso lo mejor posible definido. Y para que la parte integral pueda actuar sobre el sistema y de esta manera formar la base de reglas con la parte integral que será en lo posible el que corregirá el error en estado estable que se tiene en el controlador difuso PID con 18 reglas.

### 3.3.2.4 Base de reglas

La base de reglas se encuentra definido por las variables lingüísticas y los valores lingüísticos tratando de emular el conocimiento humano en lo que concierne al control del levitador magnético, en otras palabras es el conocimiento del experto de cómo debe funcionar y que debemos controlar, para empezar tenemos 27 reglas las cuales son suficientes para poder controlar el levitador magnético y se presentan a continuación con la misma lógica expuesta en la inferencia difusa.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **1** | Si | Cambio de Error es Positivo | Y | Error es negativo | Y | La integral es negativa | Entonces | Control es negativo |
| **2** | Si | Cambio de Error es cero | Y | Error es negativo | Y | La integral es negativa | Entonces | Control es negativo |
| **3** | Si | Cambio de Error es negativo | Y | Error es negativo | Y | La integral es negativa | Entonces | Control es Grande negativo |
| **4** | Si | Cambio de Error es Positivo | Y | Error es Cero | Y | La integral es negativa | Entonces | Control es cero |
| **5** | Si | Cambio de Error es Cero | Y | Error es cero | Y | La integral es negativa | Entonces | Control es Pequeño negativo |
| **6** | Si | Cambio de Error es negativo | Y | Error es cero | Y | La integral es negativa | Entonces | Control es negativo |
| **7** | Si | Cambio de Error es Positivo | Y | Error es positivo | Y | La integral es negativa | Entonces | Control es Positivo |
| **8** | Si | Cambio de Error es Cero | Y | Error es positivo | Y | La integral es negativa | Entonces | Control es Cero |
| **9** | Si | Cambio de Error es negativo | Y | Error es positivo | Y | La integral es negativa | Entonces | Control es negativo |
| **10** | Si | Cambio de Error es positivo | Y | Error es negativo | Y | La integral es cero | Entonces | Control es cero |
| **11** | Si | Cambio de Error es Cero | Y | Error es negativo | Y | La integral es cero | Entonces | Control es Pequeño negativo |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **12** | Si | Cambio de Error es negativo | Y | Error es negativo | Y | La integral es cero | Entonces | Control es negativo |
| **13** | Si | Cambio de Error es Positivo | Y | Error es cero | Y | La integral es cero | Entonces | Control es Pequeño Positivo |
| **14** | Si | Cambio de Error es Cero | Y | Error es cero | Y | La integral es cero | Entonces | Control es Cero |
| **15** | Si | Cambio de Error es negativo | Y | Error es cero | Y | La integral es cero | Entonces | Control es Pequeño negativo |
| **16** | Si | Cambio de Error es Positivo | Y | Error es positivo | Y | La integral es cero | Entonces | Control es Positivo |
| **17** | Si | Cambio de Error es Cero | Y | Error es positivo | Y | La integral es cero | Entonces | Control es Pequeño Positivo |
| **18** | Si | Cambio de Error es negativo | Y | Error es positivo | Y | La integral es cero | Entonces | Control es cero |
| **19** | Si | Cambio de Error es Positivo | Y | Error es negativo | Y | La integral es positivo | Entonces | Control es Positivo |
| **20** | Si | Cambio de Error es Cero | Y | Error es negativo | Y | La integral es positivo | Entonces | Control es Cero |
| **21** | Si | Cambio de Error es Negativo | Y | Error es negativo | Y | La integral es positivo | Entonces | Control es negativo |
| **22** | Si | Cambio de Error es Positivo | Y | Error es cero | Y | La integral es positivo | Entonces | Control es Positivo |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **23** | Si | Cambio de Error es Cero | Y | Error es cero | Y | La integral es positivo | Entonces | Control es Pequeño positiva |
| **24** | Si | Cambio de Error es negativo | Y | Error es cero | Y | La integral es positivo | Entonces | Control es cero |
| **25** | Si | Cambio de Error es Positivo | Y | Error es positivo | Y | La integral es positivo | Entonces | Control es Grande positivo |
| **26** | Si | Cambio de Error es cero | Y | Error es positivo | Y | La integral es positivo | Entonces | Control es Positivo |
| **27** | Si | Cambio de Error es negativo | Y | Error es positivo | Y | La integral es positivo | Entonces | Control es positivo |

**3.4. Base de 27 reglas**

Si se desea analizar la base de reglas de una manera sencilla y entendible se puede decir que cuando se tiene dos variables de la misma magnitud, la salida va a tener la misma salida de las dos magnitudes. Un ejemplo para explicar esto es:

Si el **cambio de error es positivo** y el **error es positivo** y la **integral es cero** la **salida tiene que ser positivo**.

Hay que seguir la misma analogía para todas la reglas que se van a ingresar y de esta manera tener todo bien definido, es así que al terminar de definir toda la base de reglas se procede a implementar el controlador difuso PID en Simulink.

### 3.3.2.5 Implementación del controlador proporcional integral derivativo PID difuso con 27 reglas en SIMULINK.

Para realizar la implementación del controlador difuso PID de 27 reglas en Simulink se debe crear el archivo de extensión ".fis" con la herramienta "fuzzy editor". La creación del archivo permite cargar el controlador al bloque del controlador difuso en Simulink. Figura. 3.33.

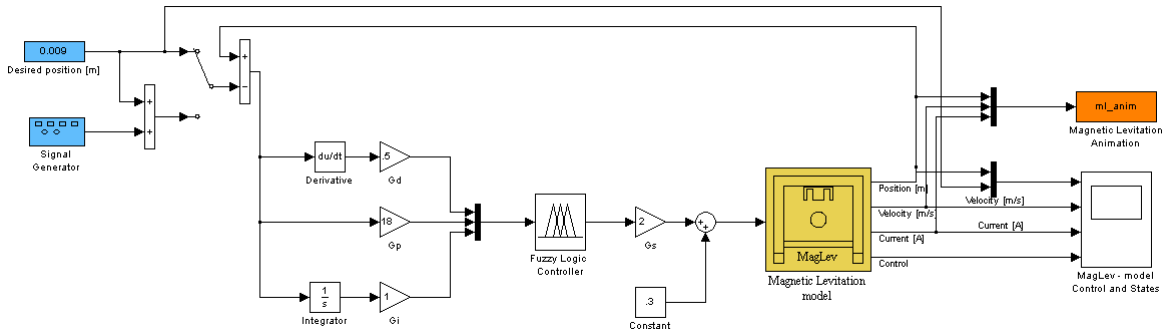



**Figura. 3.33. Diseño del controlador difuso PID de 27 reglas**

Como se puede observar en la figura. 3.33, las ganancias del sistema son diferentes con las ganancias del controlador difuso de 18 reglas, esto se lo realizó ya que el rango de las variables lingüísticas para el nuevo diseño han sido modificados.

### 3.3.2.6 Simulación del controlador difuso proporcional integral derivativo PID con 27 reglas

La simulación del controlador difuso PID de 27 reglas se presenta a continuación:

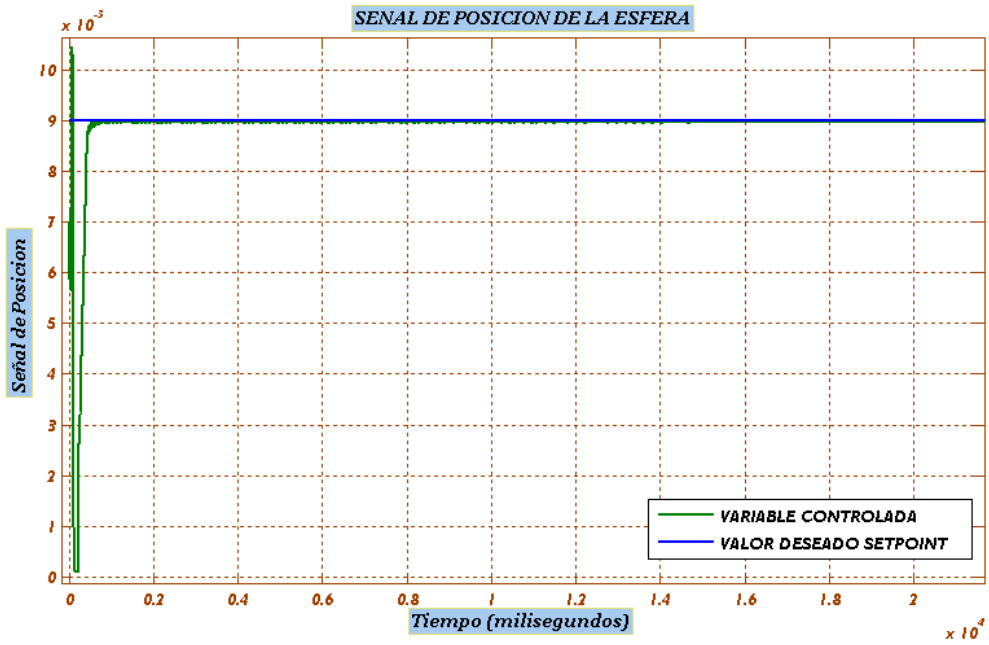



**Figura. 3.34. Simulación de Controlador difuso con 27 reglas.**

**Gd=0.5, Gp=18, Gi=1, Gs=2, Constante=0.3**

La simulación fue realizada con valores de ganancias que se exponen en la figura. 3.34., que fueron necesarios modificar ya que el rango de las variables lingüísticas fue cambiado. Se puede observar que la posición de la esfera se ubica rápidamente en el valor deseado dando a entender que el control es aceptable.

Al realizar un acercamiento a la figura 3.34., se puede observar que al realizar la simulación se tiene pequeñas oscilaciones cuando llega a estabilizarse que son insignificantes. Figura 3.35.

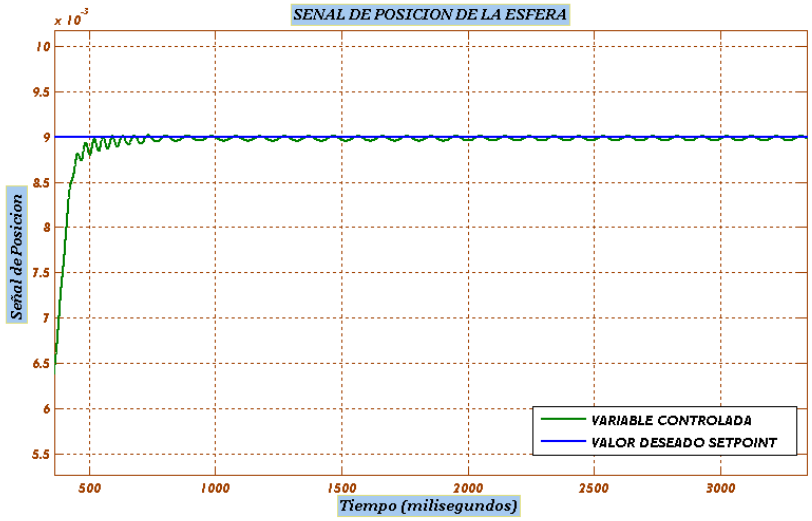



**Figura. 3.35. Pequeñas oscilaciones con ganancias de valor de Gd=0.5, Gp=18, Gi=1, Gs=2, Constante=0.3**

Como se puede observar en la figura 3.3.5, existen oscilaciones una vez que se llega al valor deseado, para corregir las oscilaciones se puede hacer un cambio en la ganancia derivativa Gd. = 0.5, y también un ajuste en la ganancia proporcional Gp = 18 y conseguir oscilaciones más suaves para llegar a eliminarlas. Figura. 3.36.

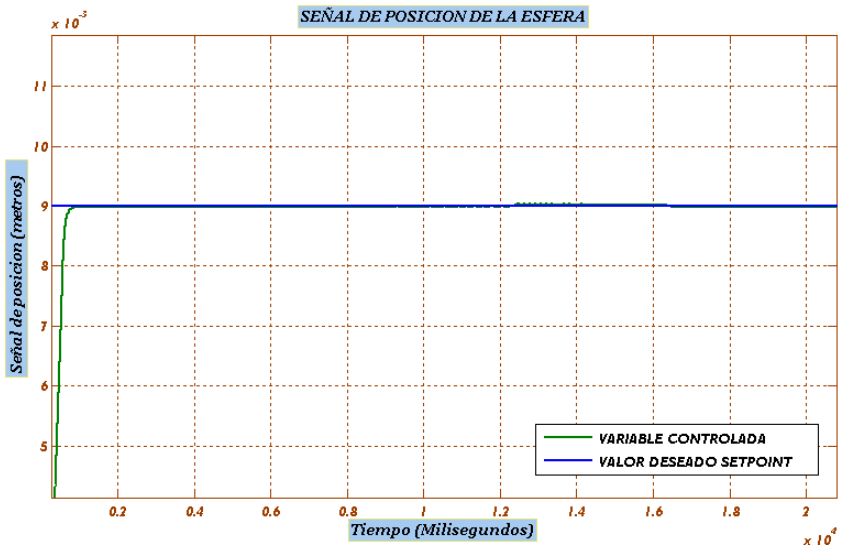



**Figura. 3.36. Pequeñas oscilaciones con ganancias de valor de Gd=0.5, Gp=18, Gi=1, Gs=2, Constante=0.3**

Como se puede ver en la figura. 3.36., las oscilaciones son más pequeñas y al final desaparecen, esto se lo puede realizar ajustando las ganancias proporcional y derivativas para conseguir un control apropiado del sistema de levitación magnética. Figura. 3.37.

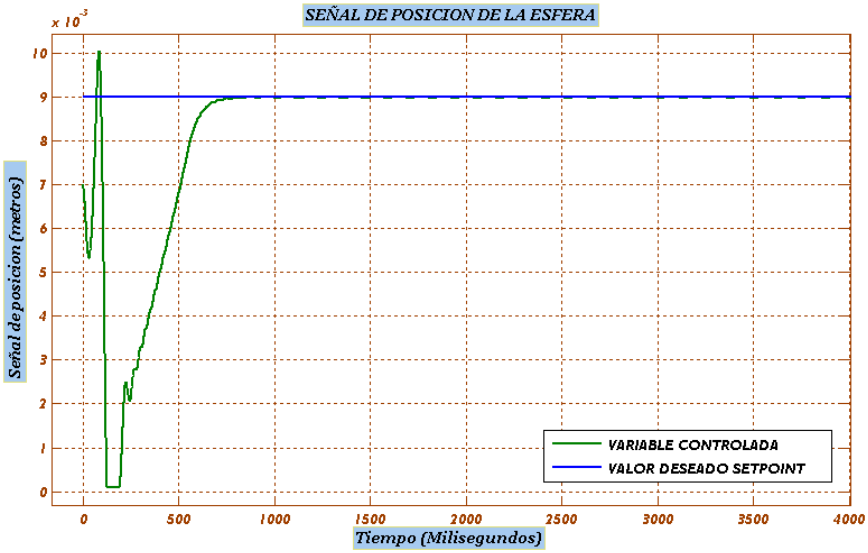



**Figura. 3.37. Impulso con ganancias de valor de Gd=0.5, Gp=18, Gi=1, Gs=2, Constante=0.3**

Se debe tener en cuenta que al variar las ganancias del sistema ayudan a modificar los universos de discurso o los rangos de las variables lingüísticas. Es por esta razón que las ganancias para los controladores con lógica difusa contienen los bloques de ganancias en los diseños. Existe un rango en el cual si se modifican las ganancias del controlador difuso deja de realizar el control de posición la esfera, dando como resultado un controlador ineficiente.

Realizar los cambios en las ganancias de los controladores difusos es muy útil al realizar la implementación en el sistema de experimentación, pero hay que tomar en cuenta que si son modificados a valores mayores o valores demasiado pequeños el controlador llega a ser ineficiente.

Se realizó un nuevo diseño a partir del controlador difuso con 27 reglas a un controlador que la variable de salida tendrá valores lingüísticos o funciones de membrecía solapados. La razón de utilizar esta técnica fue para realizar un control más exacto y presentar los resultados obtenidos.

### 3.3.2.7 Ajustes del controlador difuso PID

Al crear el nuevo diseño del controlador difuso PID con la variable de salida con funciones de membrecía solapados se espera tener un controlador lo suficientemente rápido y preciso al controlar la posición de la esfera.

Se aumentó valores lingüísticos en las entradas y en la salida se realizó la solapasión de las funciones de membrecía.

Los cambios que fueron realizados dentro del diseño del controlador difuso PID con la variable de salida solapada son:

- Definir las variables lingüísticas con nuevos valores difusos
- Definir la variable **Control o de salida** con nuevos valores difusos solapados
- Definir una nueva base de reglas

Tomando en cuenta los cambios mencionados que se realizarán, se procede con un nuevo diseño del controlador difuso PID, esperando corregir los

pequeños errores que se tienen en los anteriores diseños de controladores difusos.

### 3.3.2.7.1 Diseño del controlador difuso PID solapado

A continuación se presenta el diseño del controlador difuso PID llamado “controlador difuso PID solapado”. El nuevo controlador difuso PID solapado se presenta en la figura 3.38.

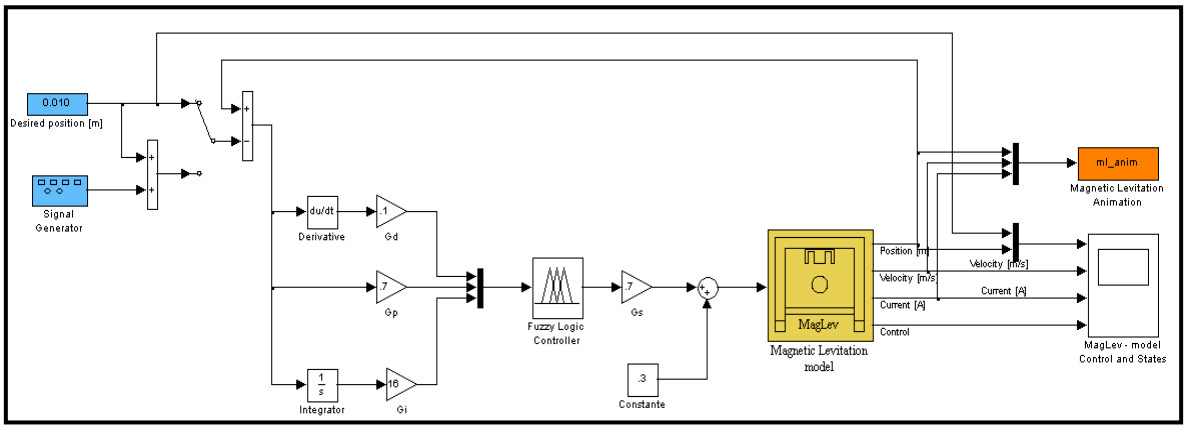



**Figura. 3.38. Controlador difuso PID solapado y sus respectivos bloques.**

Se puede observar en la figura 3.38, que las ganancias son diferentes a los anteriores controladores difusos PID, por la razón, que los universos de discurso o variables lingüísticas fueron modificados para este diseño, y por lo tanto se debe cambiar también las ganancias.

La figura. 3.38., muestra el modelo de simulación y los bloques de ganancias que fueron utilizados que más adelante se explican a detalle. La definición de las variables y valores lingüísticos se presentan a continuación en el siguiente punto.

### 3.3.2.7.2 Definición de variables lingüísticas y valores lingüísticos

Para el nuevo diseño del controlador difuso PID solapado se definen los siguientes valores lingüísticos para las variables “error”, “cambio de error”.

- Grande Negativo (GN)
- Pequeño Negativo (PN)
- Cero (Cero)
- Pequeño Positivo (PP)
- Grande Positivo (GP)

La variable lingüística “Integral” se define con tres valores lingüísticos que son:

- Pequeño Negativo (PN)
- Cero (Cero)
- Pequeño Positivo (PP)

Se define la variable lingüística con tres valores lingüísticos por la razón de hacer actuar a la “integral” solo en ciertas reglas que se consideran necesarias.

La variable lingüística “Control” se define con seis valores lingüísticos que harán o formaran una base lingüística totalmente diferente a las anteriores. Los valores lingüísticos definidos son los siguientes:

- Pequeño Negativo (PN)
- Pequeño negativo cero central (NZC)
- Cero 1 (Cero)
- Cero 2 (Cero 2)
- Pequeño positivo cero central (PZC)
- Pequeño Positivo (PP)

Son un total de 6 valores lingüísticos que se definieron para el nuevo diseño del controlador difuso PID solapado.

La variable lingüística “control” tendrá tres funciones de membrecía solapados en el valor lingüístico “cero”, esto se realiza para tener un control más exacto cuando la posición de la esfera llegue al valor deseado o llegue al valor lingüístico “cero”.

Se define el valor lingüístico “cero” con funciones de membrecía más pequeños que harán que si la posición llega a ubicarse cerca del valor deseado atraigan a la posición con exactitud hacia el centro o hacia el nuevo valor lingüístico llamado “cero 2”. Los valores lingüísticos para el valor “cero” son:

- Pequeño negativo cero central (NZC)
- Cero 2 (Cero 2)
- Pequeño positivo cero central (PZC)

Estos valores actuaran siempre y cuando la posición de la esfera y la integral estén cerca del cero.

### 3.3.2.7.3 Funciones de membrecía

Las funciones de membrecía para el nuevo diseño del controlador difuso PID solapado se presentan con el editor de funciones de membrecía de la herramienta "fuzzy editor". Figura 3.39.

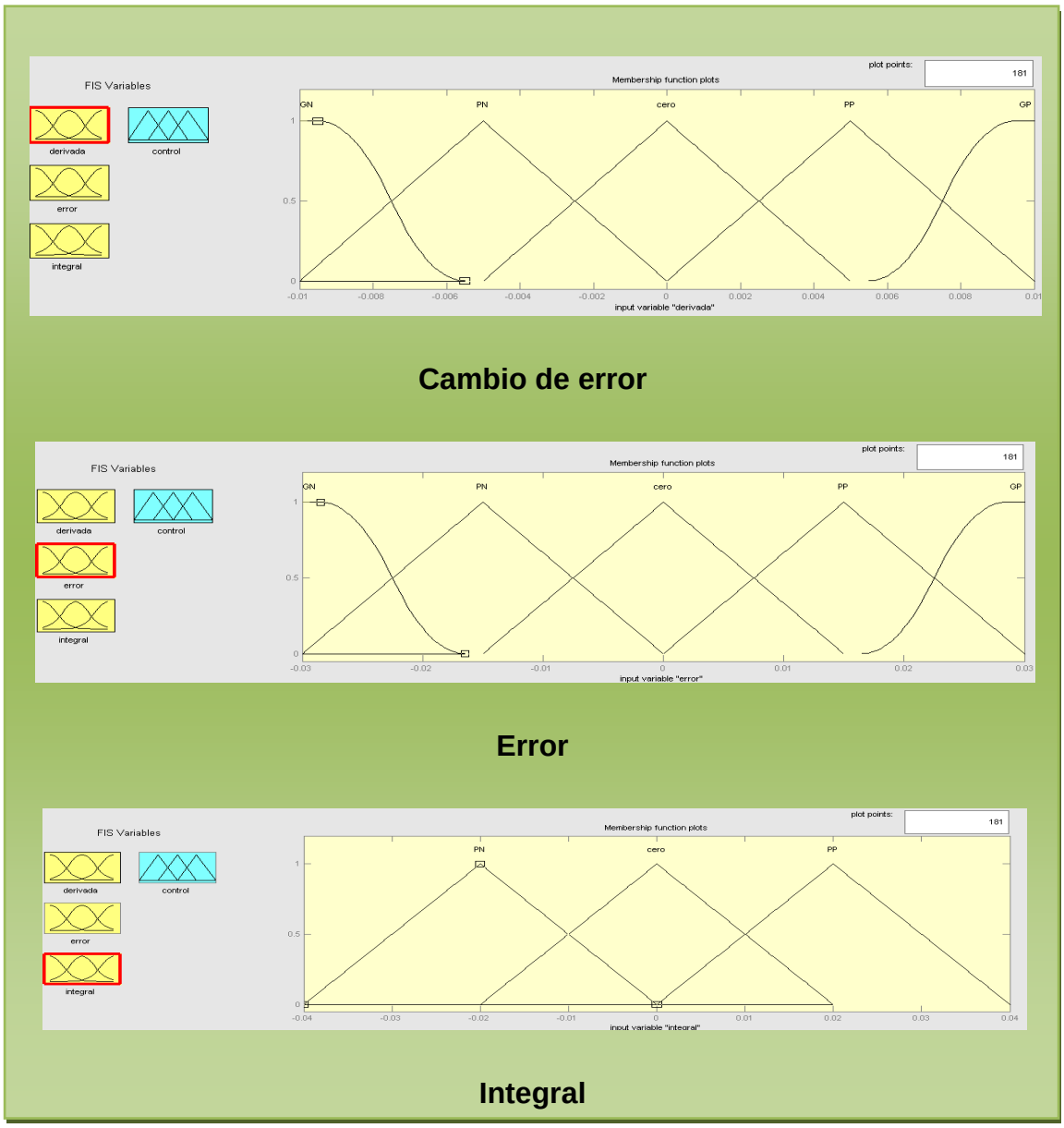



**Figura. 3.39. Funciones de membrecía para Variables de Entradas controlador difuso PID solapado**

Como se puede observar en la figura 3.39, las funciones de membrecía se encuentran definidas por los rangos que fueron definidos con anterioridad y también los valores lingüísticos para cada variable.

Para la variable lingüística “control” se definen las funciones de membrecía con el editor de funciones. Figura 3.40.

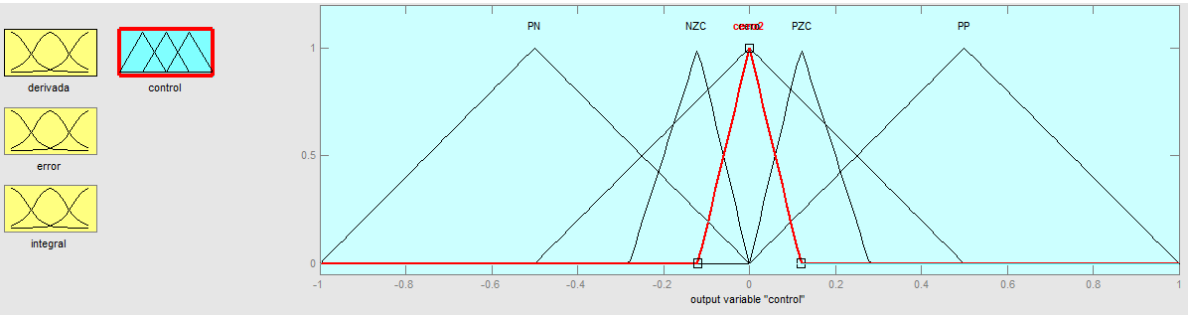



**Figura. 3.40. Funciones de membrecía Variable de Salida**

En la figura. 3.40., se puede observar que el **VALOR LINGÜÍSTICO CERO** se encuentra definido con tres valores más pequeños.

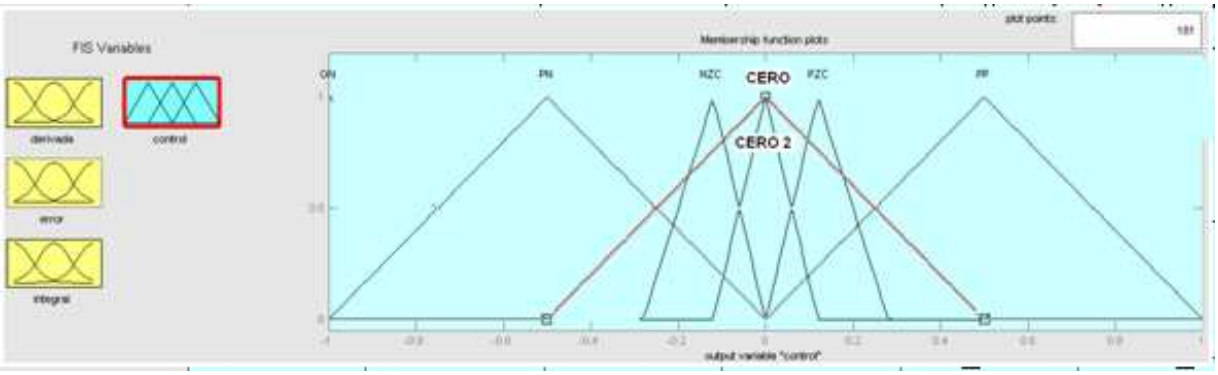



**Figura. 3.41. Conjunto difuso CERO definido en tres partes más pequeñas**

De esta manera se aumenta el número de valores lingüísticos para tener una base de reglas mejor definida y con mayores posibilidades de corregir el error en estado estacionario.

### 3.3.2.7.4 Base de reglas

La base de reglas que fue diseñada para el nuevo controlador difuso PID solapado es la siguiente:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **1** | Si | Error es Grande Negativo | Y | Cambio de error es Grande positivo | Y | Integral es cero | Entonces | Control es cero |
| **2** | Si | Error es Pequeño negativo | Y | Cambio de error es Pequeño negativo | Entonces | Control es Pequeño negativo | | |
| **3** | Si | Error es Pequeño negativo | Y | Cambio de error es cero | Entonces | Control es Pequeño negativo | | |
| **4** | Si | Error es Pequeño negativo | Y | Cambio de error es Pequeño positivo | Y | Integral es cero | Entonces | Control es cero |
| **5** | Si | Error es Cero | Y | Cambio de error es Pequeño negativo | Entonces | Control es Pequeño negativo | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **6** | Si | Error es cero | Y | Cambio de Error es cero | Y | Integral es Cero | Entonces | Control es cero 2 |
| **7** | Si | Error es Cero | Y | Cambio de Error es pequeño positivo | Entonces | Control es pequeño positivo | | |
| **8** | Si | Error es pequeño positivo | Y | Cambio de error pequeño negativo | Y | Integral es cero | Entonces | Control es cero |
| **9** | Si | Error es Pequeño Positivo | Y | Cambio de error es cero | Entonces | Control es pequeño positivo | | |
| **10** | SI | Error es Pequeño positivo | Y | Cambio de error es pequeño positivo | Entonces | Control es Pequeño positivo | | |
| **11** | Si | Error es Grande Positivo | Y | Cambio de error es Grande negativo | Y | Integral es Cero | Entonces | Control es Cero |
| **12** | Si | Error es cero | Y | Cambio de error es cero | Y | Integral es Pequeño negativo | Entonces | Control es Negativo cero central |
| **13** | Si | Error es cero | Y | Cambio de error es cero | Y | Integral Pequeño positivo | Entonces | Control es Positivo cero central |

**Tabla. 3.5. Base de reglas para el controlador difuso PID solapado**

Al realizar un estudio a la base de reglas se puede observar que solamente se hace actuar a la parte integral cuando cae dentro de valores lingüísticos críticos que hacen que la esfera se salga del punto de valor deseado. Mientras tanto, cuando las variables “error” y “cambio de error” toman valores de cero y la

integral toma valores “pequeño negativos” y “pequeño positivos” se hace actuar a los valores de control “negativo cero central” y “positivo cero central”.

Las ganancias para el sistema de control difuso PID solapado son las siguientes:

- Ganancia derivativa (Gd.) = 0.1
- Ganancia proporcional (Gp.) = 0.7
- Ganancia Integral (Gi.) = 16
- Ganancia de salida (Gs.) = 0.7
- Constante = 0.3

**3.3.2.7.5 Simulación del controlador difuso PID solapado**

Se presenta a continuación la simulación del controlador difuso PID solapado en el entorno Simulink, con todo lo mencionado acerca de las funciones de membrecía en las entradas y la salida del controlador difuso.

Las ganancias que fueron definidas se deben tomar en cuenta para realizar la simulación.

Los resultados que presenta la simulación del controlador difuso PID solapado se presenta a continuación:

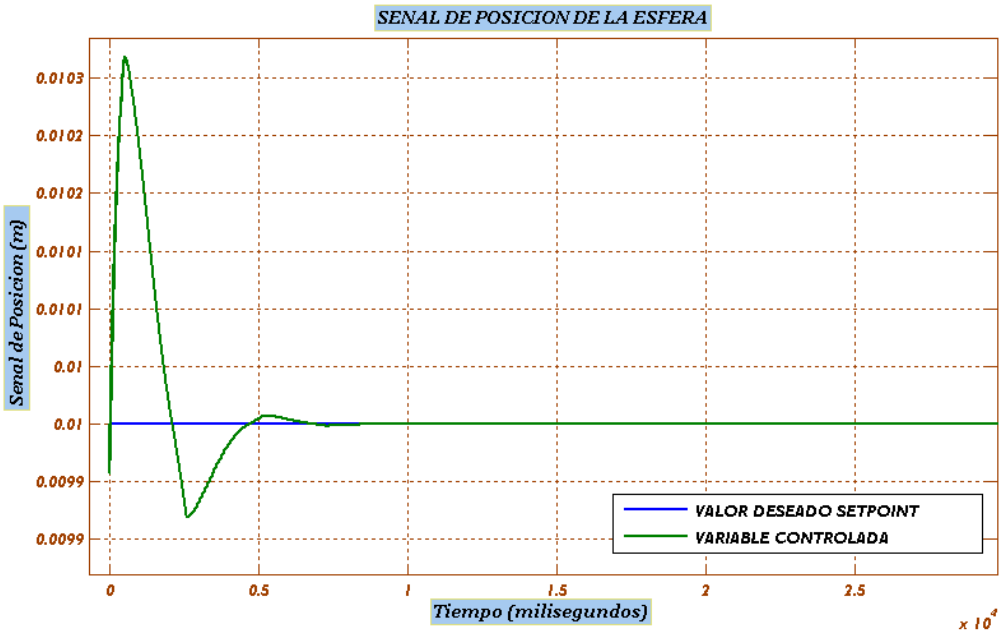



**Figura. 3.42. Simulación de Controlador difuso PID solapado.**

**Gd=0.1, Gp=0.7, Gi=16, Gs=0.7, Constante=0.3**

Se puede observar en la figura 3.42, como la posición de la esfera se ubica en el valor deseado de 0.01 m., concluyendo que el controlador difuso PID solapado tiene mejor respuesta en error en estado estable que los anteriores controladores difusos.

Se puede observar que al inicio de la simulación se tiene unas pequeñas oscilaciones alrededor del punto deseado que no influyen al controlador difuso. Figura 3.42.

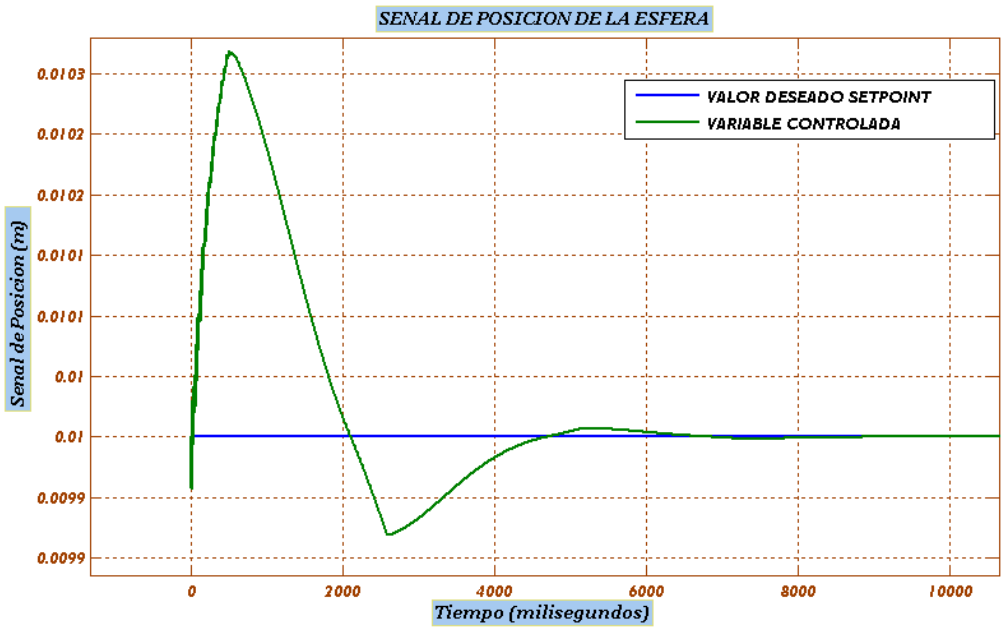



**Figura. 3.43. Sobre Impulso generado por la acción de control en el controlador difuso PID solapado**

**Gd=0.1, Gp=0.7, Gi=16, Gs=0.7, Constante=0.3**

La acción de control que se genera es parte del controlador difuso PID ya que al inicio de la simulación trata rápidamente de corregir el error en estado estable.

El tiempo de establecimiento es de 800 milisegundos para el controlador difuso PID solapado.

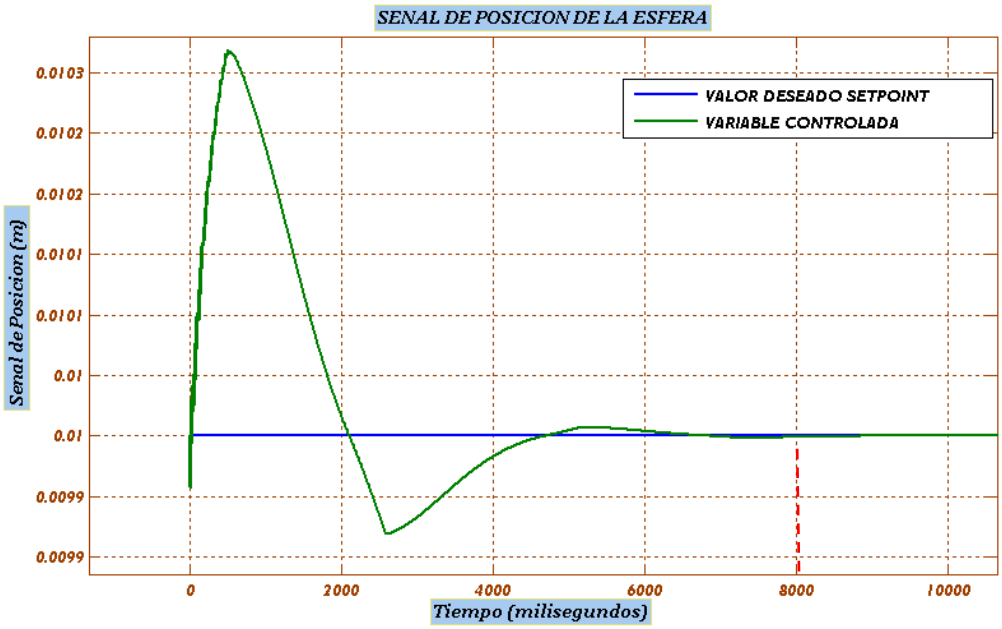



**Figura. 3.44. Tiempo de establecimiento Gd=0.1, Gp=0.7, Gi=16, Gs=0.7, Constante=0.3**

De esta manera se concluye con el diseño y simulación de los controladores difusos para el sistema de levitación magnética.

## 3.4 DISEÑO DEL MODELO DE SIMULACIÓN MEDIANTE REDES NEURONALES

En este punto del proyecto se propone realizar el control del sistema de levitación magnética con redes neuronales. Para lo cual se necesita realizar inicialmente el modelo del levitador con redes neuronales para después crear el controlador.

El algoritmo utilizado para el aprendizaje de las redes neuronales fue el de “Backpropagation”, este algoritmo fue implementado en el código que se generó para crear las redes neuronales.

El proceso que se realiza esta enfocado en utilizar las herramientas de software que se tiene, y para esto se utiliza el modelo matemático no lineal del levitador. Figura 3.45.

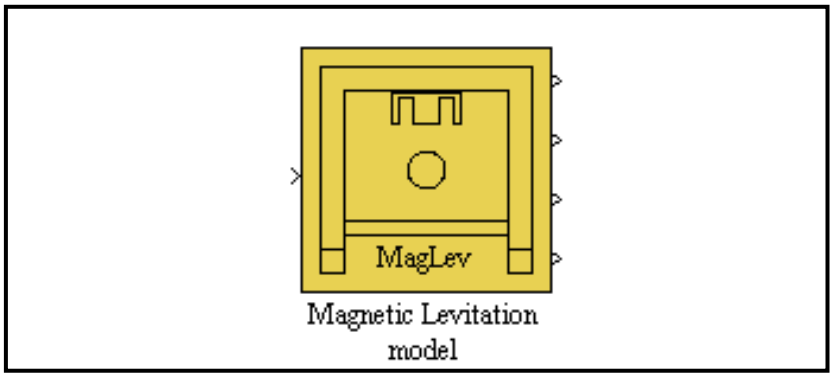



**Figura. 3.45. Modelo de simulación del levitador magnético**

Para realizar el diseño de la red neuronal del modelo del levitador se necesita utilizar el modelo de simulación, del cual la red neuronal obtendrá el conocimiento o entrenamiento.

Para que la red neuronal sea idéntica o exacta al modelo experimental se requiere que el modelo de simulación sea lo más aproximado posible. Para lo cual se ha obtenido el modelo de simulación de trabajos previos que fueron realizados en el sistema de levitación magnética. Tabla 3.6.

| PARAMETROS | VALOR | UNIDAD |
|---|---|---|
| $m$ | 0.04481 | $[kg]$ |
| $g$ | 9.81 | $\left[{}^{m}/_{s^2}\right]$ |

| | | |
|---|---|---|
| $F_{em}$ | 0.9501 | $[N]$ |
| $F_{em}P1$ | 0.02697 | [H] |
| $F_{em}P2$ | 0.004587 | [m] |
| $fi(x_1)$ | Función de posición | $[1/s]$ |
| $f_{iP1}$ | 0.0032 | $[m * s]$ |
| $f_{iP2}$ | 0.1146 | [m] |
| $C_i$ | -0.043 | [A] |
| $K_i$ | 2.6 | [A] |
| $x_{3MIN}$ | 0.0050 | [A] |
| $u_{MIN}$ | 0.00498 | |

**Tabla. 3.6. Parámetros del modelo matemático de levitación magnética**

Se presenta el modelo de simulación magnética con los valores de la tabla 3.6., en la figura 3.46.

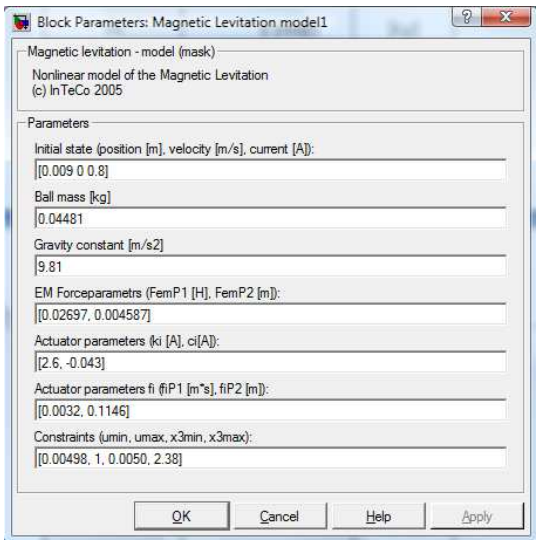



**Figura. 3.46. Parámetros del modelo matemático de levitación magnética**

### 3.4.1 Diseño del modelo de simulación con redes neuronales

Para el diseño del modelo de simulación con redes neuronales se utilizará el modelo de simulación del levitador como referencia. La red neuronal aprenderá a comportarse igual que el modelo de simulación. Para lo cual se utiliza el entrenamiento de la red con el algoritmo de "**Backpropagation**", el cual actualiza los pesos para la red neuronal en forma automática hasta conseguir que el error sea mínimo. La característica de entrenamiento de este algoritmo fue explicado en el capitulo dos.

Para empezar con el diseño de la red neuronal se empieza generando el código que contendrá la base de la red neuronal. Se empieza colocando los patrones de entrenamiento que serán los que caracterizan al modelo matemático; los valores son los siguientes:

Entrada = 0.23: 0.03: 0.5

Posición = 0.001: 0.002: 0.02

Velocidad = -100e-3: 25e-3: 100e-3

Corriente = 0.2: 0.2: 2

Se crean matrices de entrada que son los rangos de trabajo del modelo de simulación del levitador, estos valores de entrada son los valores de condiciones iniciales y las condiciones iniciales deberán ser combinadas entre sí para generar una matriz de estados iniciales.

La matriz de estados iniciales será ingresada al sistema de simulación del levitador, y obtener después de un tiempo de simulación una matriz de objetivos.

Una vez que se tiene la matriz de estados iníciales se procede a realizar la simulación del sistema en lazo abierto como se muestra en la figura. 3.47., para poder obtener una matriz después de un tiempo de simulación que va a ser la matriz de objetivos. La matriz de objetivos es necesaria para que la red neuronal se entrene y trate de llegar a obtener un error mínimo.

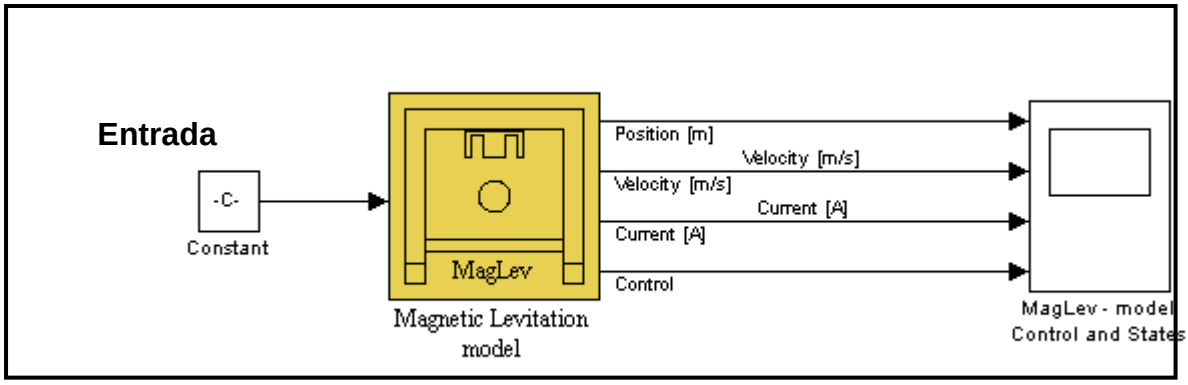



**Figura. 3.47. Modelo no lineal en lazo abierto. "Mlm"**

Se puede observar de la figura 3.47, que se tiene al modelo no lineal en lazo abierto del cual se ingresarán la matriz de estados iniciales para obtener la matriz de objetivos.

Para comenzar con la combinación de los vectores de estados iniciales se utiliza el siguiente comando dentro del código.

- **Pm = combvec(posicion,velocidad,corriente,entrada)**

Es una función de Matlab que realiza todas las combinaciones posibles de los vectores que fueron definidos al inicio, y se los guarda en una variable que contendrá los valores de estados iníciales o la matriz de estados iníciales.

Dentro del código que se generó se utiliza la abreviatura **Pm** para la **matriz de estados iniciales**. Mientras que para la **matriz de objetivos** la abreviatura de **Tm**.

Se presenta el código con el que se simula desde el editor de Matlab, para conseguir la matriz **Tm** de objetivos, realizando un lazo que recorra toda la matriz de **estados iniciales Pm** y guardarlo en la **matriz de objetivos TM**.

```
For i=1: Q
  Entrada = Pm (4, i);
   x10 = Pm (1, i);
   x20 = Pm (2, i);
   x30 = Pm (3, i);
   sim ('Mlm');
   n=numel (MLSimData.signals (1).values);
   Tm (1, i) = MLSimData.signals (1).values (n)-x10;
   Tm (2, i) = MLSimData.signals (2).values (n)-x20;
   Tm (3, i) = MLSimData.signals (3).values (n)-x30;
  End
```

Con el código generado se puede observar que la matriz PM es ingresado en el modelo no lineal llamado “Mlm” para ser simulado. El modelo “Mlm” se simula con el comando “SIM (‘modelo no lineal’)” para de esa forma obtener la matriz de objetivos.

Una vez que se termina con la simulación y se obtiene la matriz “TM” de objetivos, se continúa con la creación de la red neuronal.

### 3.4.2 Creación de la red neuronal

La función **newff ()** es usada para crear una red neuronal de tres capas con funciones no lineales del tipo tansig/purelin.

El código que se ha generado es el siguiente:

```
S1 = 8;
[S2, Q] = size(Tm);
mnet = newff (minmax (Pm), [S1 S2], {'tansig' 'purelin'},'trainlm');
```

Se usará esta red neuronal para modelar el sistema de levitación magnética donde S1 es el numero de neuronas escondidas en cada capa y S2 será el numero de capaz utilizado. Para este caso serán ocho neuronas escondidas y tres capas, y todo esto dentro de la red neuronal del tipo “Backpropagation”.

Cuando se termine con la creación de la red neuronal se continúa con el entrenamiento.

### 3.4.3 Entrenamiento de la Red Neuronal

Para realizar el entrenamiento de la red neuronal se utilizará la función "**trainlm**", es una función de entrenamiento por defecto y es bastante rápida, pero requiere de mucha memoria de procesamiento. Por lo tanto fue utilizada para obtener un entrenamiento rápido en la red neuronal.

La red se entrena con capacidad para 10000 épocas, mostrando los progresos de entrenamiento cada 10 épocas, y con un error típico de $0.00037^2$ radianes por defecto.

```
mnet.trainParam.show    = 10;
mnet.trainParam.epochs  = 10000;
mnet.trainParam.goal    = 0.00037^2;
```

El código que se presenta son las características de entrenamiento de la red neuronal. El entrenamiento será presentado cada 10 épocas, y durará 10000 épocas, y el error será de $0.00037^2$, cuando el entrenamiento llegue a este error se detendrá especificando que la red neuronal llego al error definido y por lo tanto el entrenamiento se dará por concluido. Figura 3.48.

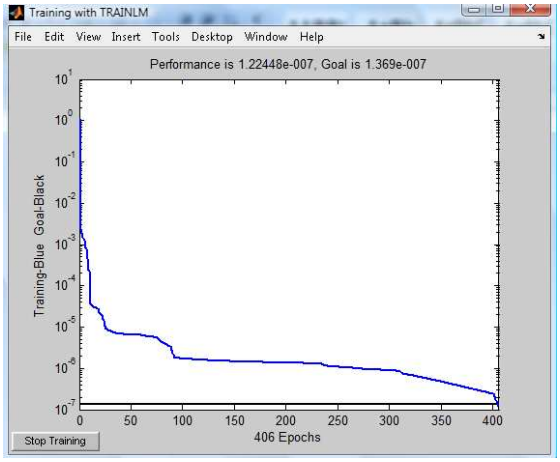



**Figura. 3.48. Entrenamiento de la red neuronal**

### 3.4.4 Simulación de la red neuronal

Cuando la red termine su entrenamiento se realiza la simulación mediante código y se presenta los resultados obtenidos. Inicialmente la respuesta del modelo del levitador. Figura 3.49.

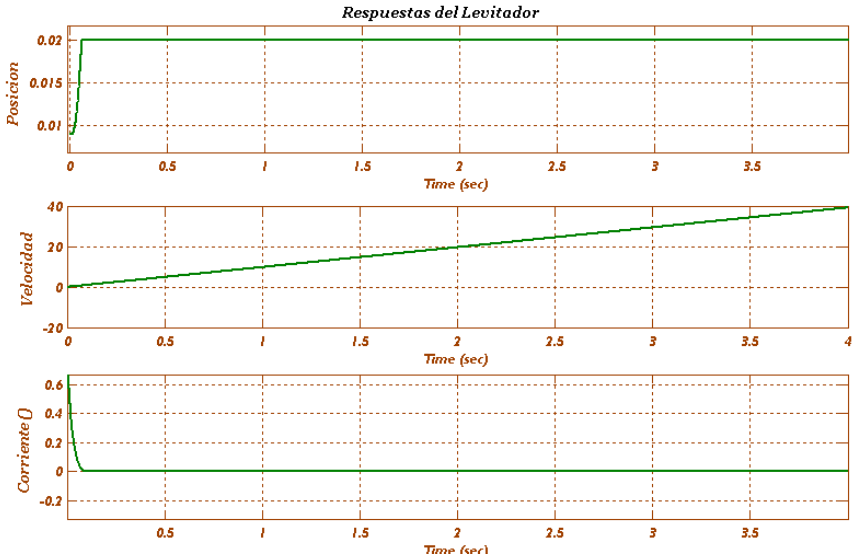



**Figura. 3.49. Respuesta del Modelo de simulación del Levitador.**

Como se puede observar en le figura. 3.49., se presentan las tres variables, posición, velocidad y corriente. La posición de la esfera cae al valor máximo del sistema, dando a entender que la esfera no fue controlada, de igual manera la velocidad se incrementa ya que la posición de la esfera no fue controlada, y por último se observa que la corriente llega a ser cero.

La segunda simulación se refiere al modelo de la red neuronal que se obtuvo en comparación al modelo del levitador. Figura 3.50.

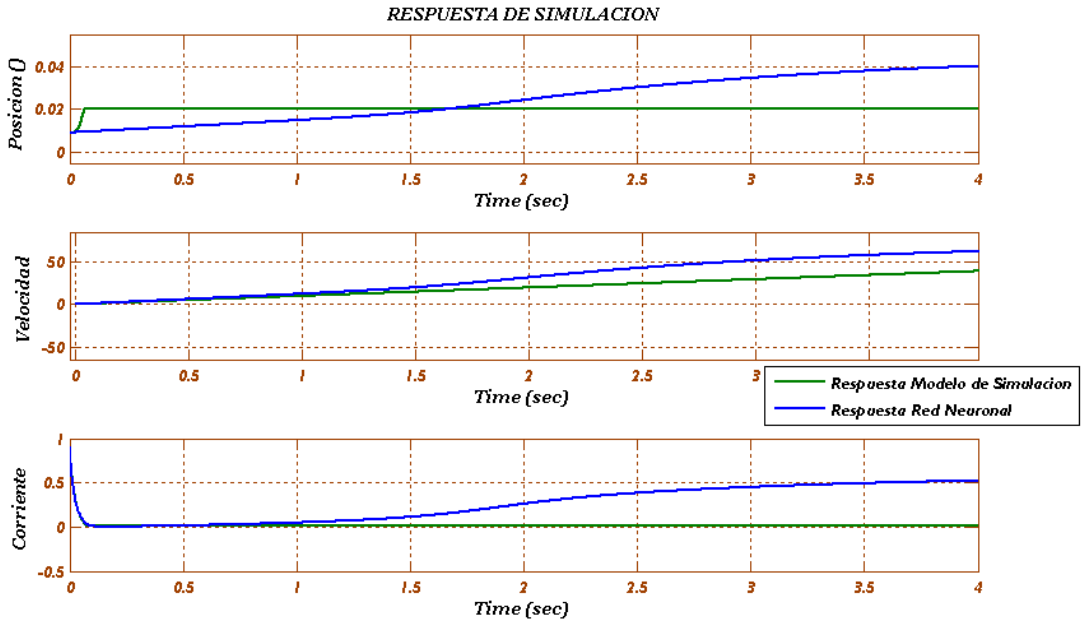



**Figura. 3.50. Comparación y Respuesta en lazo abierto de la red neuronal y modelo de levitador**

Como se puede observar en la figura. 3.50., se tiene un error de la red neuronal con respecto al modelo de referencia o modelo del levitador.

Se espera que al realizar la simulación de la red neuronal en lazo cerrado se tenga mejore resultados en cuanto a la respuesta del modelo neuronal.

De esta manera es que se realiza la simulación en lazo cerrado. Figura 3.51.

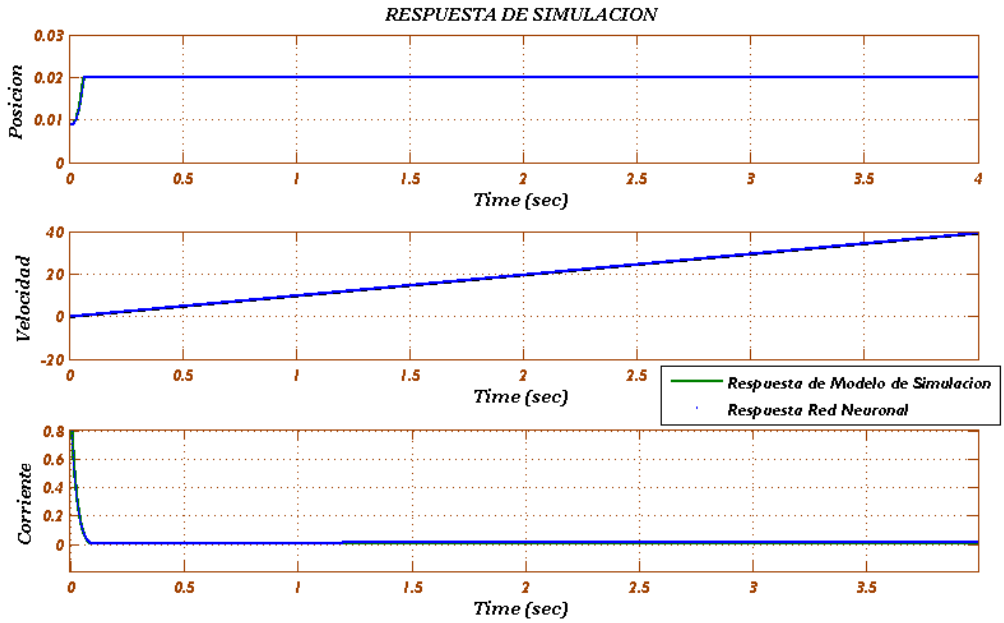



**Figura. 3.51. Comparación y Respuesta en lazo cerrado red neuronal y modelo.**

Realizando un acercamiento a la imagen. Figura 3.52.

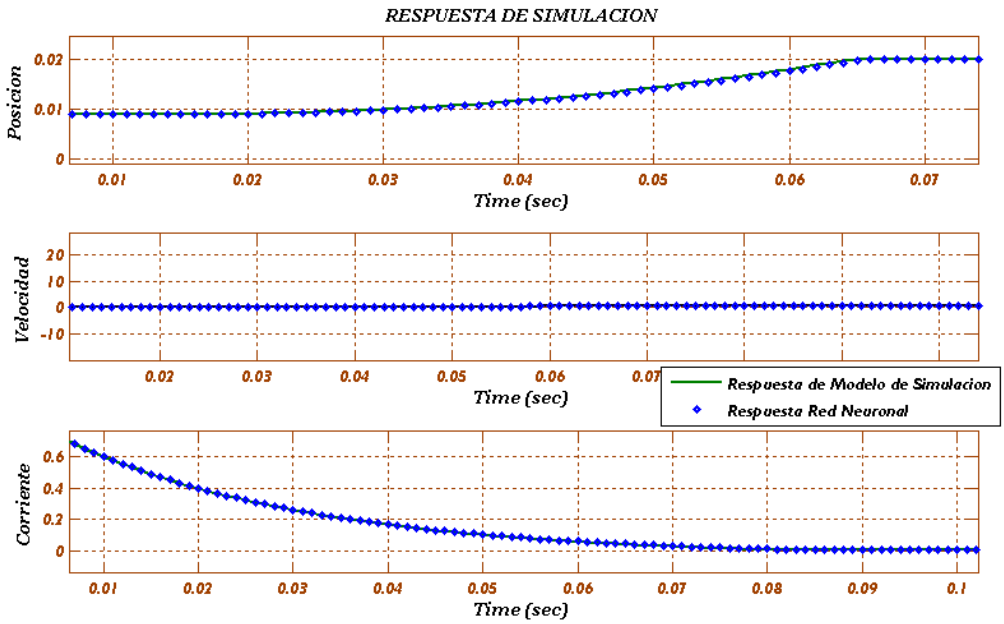



**Figura. 3.52. Acercamiento de simulación Lazo cerrado.**

La respuesta en lazo cerrado de la red neuronal con respecto al modelo del levitador está representada con color azul y el signo de un rombo.

Se puede observar en la figura 3.52, que la red neuronal es semejante frente a la simulación del modelo del levitador, esto quiere decir que la red neuronal esta comportándose igual que el modelo de referencia y se ha logrado que la red neuronal aprenda del modelo de referencia del levitador.

De acuerdo a la red diseñada con tres capas y con funciones no lineales que fue previamente diseñada, se puede observar que la red neuronal se comporta igual que el modelo de referencia y ha logrado aprender el comportamiento del sistema de levitación magnética. Por lo tanto se puede realizar un controlador para el modelo de red neuronal siguiendo el mismo procedimiento de diseño, tomando en cuenta que el modelo creado con la red neuronal no deberá aprender una vez que se cree el controlador con redes neuronales.

El controlador neuronal tendrá como referencia al PID clásico del cual obtendrá su entrenamiento. El modelo de la red neuronal que se acaba de diseñar no deberá aprender y se la debe condicionar.

## 3.5 CONTROLADOR CON REDES NEURONALES

Para realizar un controlador con redes neuronal se debe entender primeramente que es lo que necesitamos controlar, igual que el estudio que se realizó en el diseño de controladores difusos. Tomando en cuenta lo mencionado se realizará el control de posición de la esfera.

### 3.5.1 Diseño de control neuronal con redes neuronales

Un control neuronal con modelo de referencia puede ser creado donde el modelo matemático no esté disponible, todo lo que se requiere es un modelo neuronal del sistema como referencia.

Lo que va hacer el controlador neuronal es tomar como entradas a la posición, velocidad, corriente y demanda y realizar todas las combinaciones posibles para formar la matriz de estados iniciales “Pc”. Para después de un tiempo de simulación obtener la matriz de objetivos “Tc”. La simulación se realizará con el modelo en Simulink “MLmT1”. Figura 3.53.

El modelo que fue utilizado para la creación del control neuronal es el siguiente:

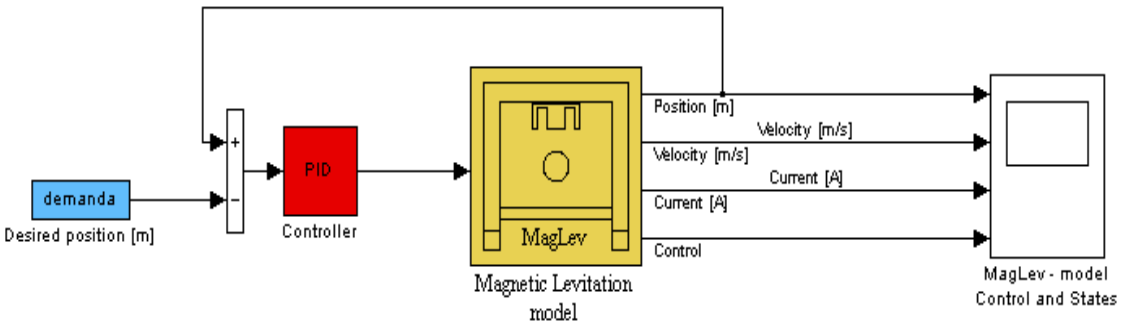



**Figura. 3.53. Modelo para el aprendizaje del controlador con redes neuronales. “MLmT1”**

Como se puede observar en la figura 3.53, posee el bloque del controlador PID clásico de donde obtendrá el entrenamiento o conocimiento de cómo controlar al modelo de simulación.

Se definen las condiciones iniciales que se necesitan para realizar las combinaciones de la matriz Pc, como se muestra a continuación.

Demanda=0.008:0.0005:0.018;

Posición=0.006:0.002:0.018;

Velocidad=-20e-3:5e-3:20e-3;

Corriente=0.6:0.1:1.4;

Igual que se hizo con el modelo de red neuronal se inicializa los vectores de estados iniciales para crear una combinación entre todos los valores y así crear una matriz de estados iniciales, pero en este caso se agrega un vector que es la **demanda** el cual es una entrada del controlador neuronal con valores de rangos de trabajo del levitador, después de esto se hace una combinación entre todos los valores con el comando:

- **Pc = combvec (posicion,velocidad,corriente,demanda)**

Esta función combinará todos los valores para conseguir una matriz de estados iniciales y después de un tiempo de simulación de 0.001 segundos obtener los siguientes estados, esto se hace, ya que el estado no cambia por una cantidad considerable en este periodo de tiempo.

La matriz de **estados iniciales** se los denomina con la siguiente abreviatura **Pc,** mientras que a los **estados de objetivos** con **Tc.**

Se presenta el código con el que se simula al modelo “MLmT1” desde el editor de Matlab para conseguir la matriz **Tc** de objetivos, realizando un lazo que recorra toda la matriz de **estados iniciales Pc.**

```
for i=1: Q

    demanda = Pc(4,i);

    x10 = Pc(1,i);

    x20 = Pc(2,i);

    x30 = Pc(3,i);

    sim('MLmT1');

    n = numel(MLSimData.time,MLSimData.signals(1).values);

    Tc(1,i) = MLSimData.signals(1).values(n) - x10;

    Tc(2,i) = MLSimData.signals(2).values(n) - x20;

    Tc(3,i) = MLSimData.signals(3).values(n) - x30;

end
```

Con este código se puede simular el modelo de referencia en Simulink “MLmT1” del cual se obtendrá la matriz de objetivos **Tc.,** para realizar el entrenamiento de la red una vez que sea creada.

### 3.5.2 Creación de la red neuronal

La función **newff ()** es usada para crear una red neuronal de cuatro capas con funciones no lineales del tipo tansig/purelin.

La red neuronal tendrá nueve neuronas en cada capa y una salida (control del levitador magnético).

```
S1 = 9;

cnet = newff (minmax (Pc), [S1 1], {'tansig' 'purelin'});
```

Como se puede observar en el código se crea una red neuronal que va ha ser almacenar en la variable **cnet** que es la variable que contendrá al controlador, también se escoge el rango de la matriz Pc., es decir se escogen el valor inicial y el valor final de toda la matriz.

### 3.5.3 Entrenamiento del controlador neuronal

Hay que tomar en cuenta el siguiente diagrama de las entradas tanto del modelo de simulación y el controlador para tener un panorama claro de lo que se va a realizar, a continuación se presenta la figura. 3.54., con la que se representa las entradas y salidas del control neuronal.

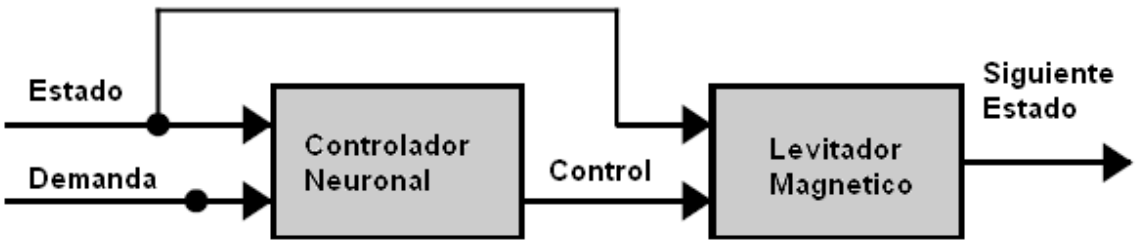



**Figura. 3.54. Diagrama del control neuronal para el sistema de levitación magnética**

Lo que se quiere conseguir es que el levitador responda con la matriz de objetivos **Tc** una vez que los 0.001 segundos hayan pasado desde que el péndulo fue inicializado con la matriz de estados iniciales **Pm,** el problema que se tiene al realizar este procedimiento es que el error entre el comportamiento del levitador real y el comportamiento deseado se produce en las salidas del levitador. ¿Cómo puede ser usado este error para ajustar el controlador?, el truco consiste en sustituir el levitador con su modelo neuronal para el propósito del entrenamiento del controlador. A continuación se presenta el diagrama del controlador neuronal con el modelo neuronal de la planta.

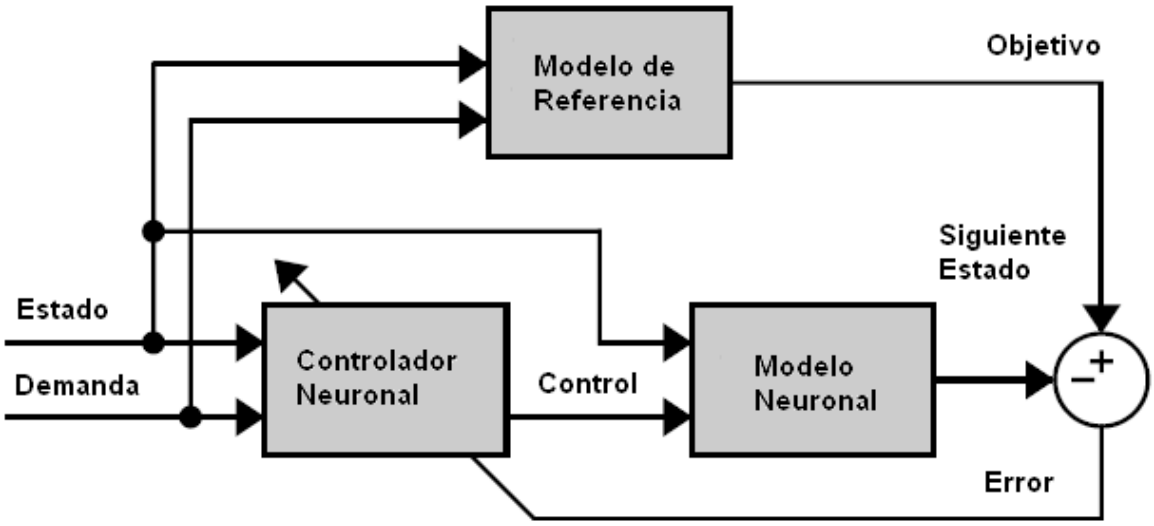



**Figura. 3.55. Diagrama del controlador neuronal con modelo de red neuronal**

Ahora el error se presenta en la salida del modelo neuronal, esto puede ser retro-propagado a través del modelo de red al control neuronal, y después retro-propagado a través del controlador y usado para ajustar los pesos del controlador, pero hay que tener en cuenta que no se cambian los pesos del modelo neuronal y así el controlador debe aprender a controlar al levitador de forma que se comporta igual que el modelo de referencia.

En el siguiente código se va a combinar y configurar el modelo neuronal con el control de red neuronal, existirán dos diferentes entradas del total de la red neuronal, que serán el estado y la demanda.

```
numInputs        = 2;
numLayers        = 4;
tnet             = network(numInputs,numLayers);
```

Como se puede ver se define el número de entradas y el número de capas, también se crea la red neuronal total **TNET** que va a ser la combinación entre el modelo neuronal “mnet” y el controlador neuronal “cnet”. Se debe tomar en cuenta que solo los pesos para el control neuronal van a ser actualizados.

Lo siguiente que se necesita es configurar las conexiones de la red.

```
tnet.biasConnect      = [1 1 1 1]';
tnet.inputConnect     = [1 0 1 0; 1 0 0 0]';
tnet.layerConnect     = [0 0 0 0; 1 0 0 0; 0 1 0 0; 0 0 1 0];
tnet.outputConnect    = [0 0 0 1];
tnet.targetConnect    = [0 0 0 1];
```

Lo que sigue es definir los rangos de las entradas, la primera entrada corresponde a los estados y la segunda entrada es la demanda o valor deseado.

```
tnet. Inputs{1}.range   = minmax(Pc(1:3,:));

tnet. Inputs{2}.range   = minmax(Pc(4,:));
```

Lo siguiente es definir el tamaño de cada capa y la función de transferencia.

```
tnet.layers{1}.size          = S1;

tnet.layers{1}.transferFcn   = 'tansig';

tnet.layers{2}.size          = 1;

tnet.layers{2}.transferFcn   = 'purelin';

tnet.layers{3}.size          = 8;

tnet.layers{3}.transferFcn   = 'tansig';

tnet.layers{4}.size          = 3;

tnet.layers{4}.transferFcn   = 'purelin';
```

Se define las cuatro capas, la primera con nueve neuronas y la función de transferencia **tansig**, la segunda capa con una neurona y la función **purelin** hasta llegar a la capa cuatro que esta con tres neuronas y función de transferencia **purelin.**

Ahora para el entrenamiento se va a utilizar el entrenamiento **Levenberg-Marquardt Backpropagation** con la función **trainlm** y para minimizar el error cuadrático medio la función **mse**.

```
tnet.performFcn          = 'mse';

tnet.trainFcn            = 'trainlm';
```

De esta manera se consigue que el error se haga mínimo al entrenarse y que la red neuronal se entrene de manera rápida.

Ahora se debe configurar los pesos de la red neuronal que corresponde al controlador **cnet** el cual irá a ser ajustado durante el proceso de entrenamiento. Para poder entrenar a la red neuronal el parámetro de aprendizaje “learn” se debe configurar en un valor de uno.

```
tnet.IW{1,1}                 = cnet.IW{1,1}(:,1:3);

tnet.inputWeights{1,1}.learn = 1;

tnet.IW{1,2}                 = cnet.IW{1,1}(:,4);

tnet.inputWeights{1,2}.learn = 1;

tnet.b{1}                    = cnet.b{1};

tnet.biases{1}.learn         = 1;

tnet.b{2}                    = cnet.b{2};

tnet.biases{2}.learn         = 1;

tnet.LW{2,1}                 = cnet.LW{2,1};

tnet.layerWeights{2,1}.learn = 1;
```

Finalmente, se debe configurar los pesos de la red neuronal **mnet** los cuales no van a ser ajustados durante el proceso de entrenamiento y para este caso se debe configurar el parámetro de entrenamiento a cero.

```
tnet.IW{3,1}                = mnet.IW{1,1}(:,1:3);
tnet.inputWeights{3,1}.learn = 0;
tnet.LW{3,2}                = mnet.IW{1,1}(:,4);
tnet.layerWeights{3,2}.learn = 0;
tnet.b{3}                   = mnet.b{1};
tnet.biases{3}.learn        = 0;
tnet.LW{4,3}                = mnet.LW{2,1};
tnet.layerWeights{4,3}.learn = 0;
tnet.b{4}                   = mnet.b{2};
tnet.biases{4}.learn        = 0;
```

Lo que sigue es entrenar a la red neuronal para que pueda realizar el control, el único que se entrenará en el proceso, es el controlador neuronal que actualizará sus pesos para ser ajustados.

El código generado es el siguiente:

```
tnet.trainParam.show        = 100;
tnet.trainParam.epochs      = 10000;
```

```
tnet.trainParam.goal        = 0.002^2;

[tnet,tr]                   = train(tnet,{Pc(1:3,:);  Pc(4,:)},{Tc});
```

El controlador es entrenado durante 10000 épocas, y cada 100 épocas se presentará el proceso de entrenamiento. Se limita un error de $0.002^2$ típico o por defecto, este error es el mínimo al que debe llegar el entrenamiento.

Como se dijo anteriormente los pesos del modelo neuronal no son actualizados y solo son los pesos del control neuronal los que deben actualizarse entonces se debe poner de regreso los nuevos pesos en el controlador neuronal.

```
cnet.IW{1,1}(:,1:3)         = tnet.IW{1,1};

cnet.IW{1,1}(:,4)           = tnet.IW{1,2};

cnet.b{1}                   = tnet.b{1};

cnet.b{2}                   = tnet.b{2};

cnet.LW{2,1}                = tnet.LW{2,1};
```

Una vez que el entrenamiento ha llegado al mínimo error este se detiene y guarda el controlador **cnet** con el que se podrá pasar al entorno de Simulink para poder utilizarlo como un bloque de simulación igual que se hizo con el bloque difuso con el siguiente comando.

```
gensim(cnet,0.001)
```

Donde este comando toma como parámetros a la red neuronal previamente creada y el tiempo de muestreo con el que se trabaja.

Con el comando “gensim” se crea el bloque de la red neuronal para poder trabajar con el entorno de Simulink como se presenta a continuación la figura. 3.56.

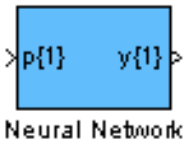



**Figura. 3.56. Red Neuronal para el controlador de levitación magnética**

Finalmente se obtiene la red neuronal para el control de levitación magnética, terminando con el proceso de diseño. En el siguiente punto se detalla los resultados de simulación de la red neuronal.

### 3.5.4 Simulación de la Red neuronal

Se realiza la implementación del controlador con redes neuronales en el entorno de Simulink para comenzar con la simulación del sistema. Figura 3.57.

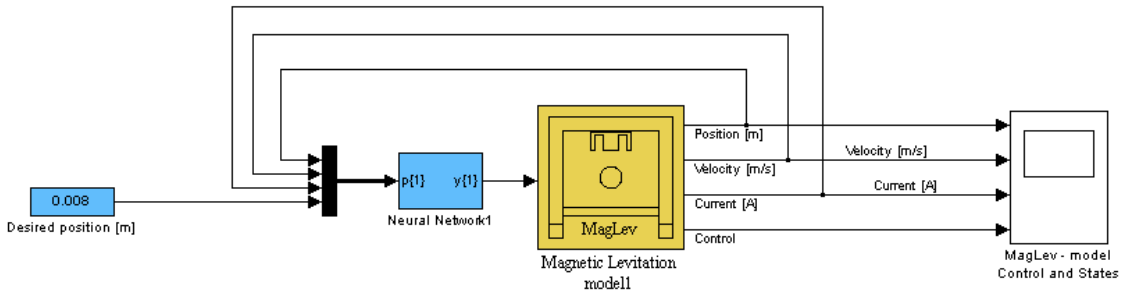



**Figura. 3.57. Controlador Neuronal con Modelo de simulación.**

Una vez creado el modelo de simulación se realiza la simulación durante 10 segundos. Figura 3.58.

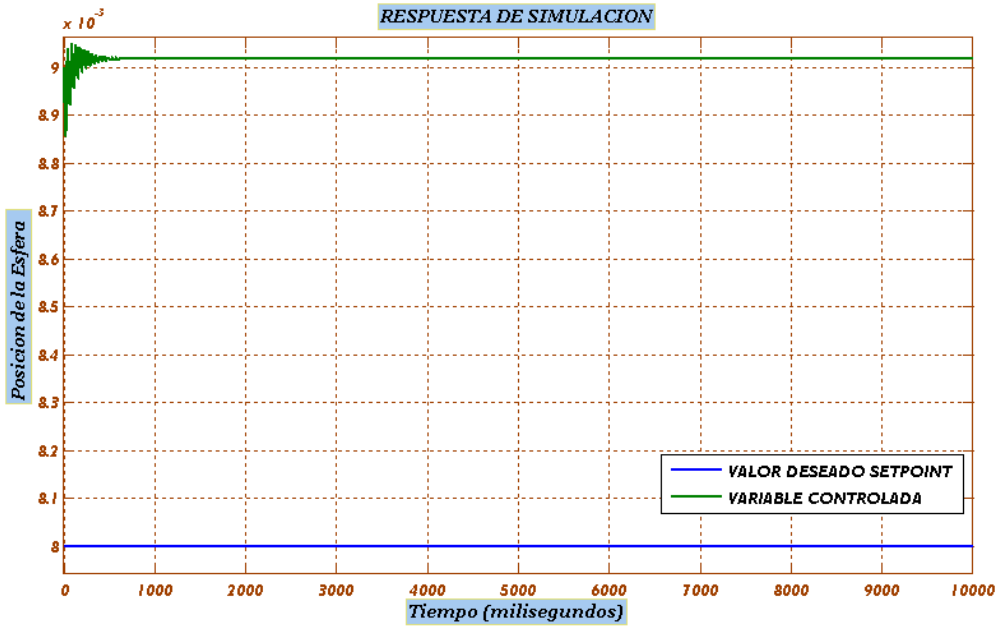



**Figura. 3.58. Posición de la esfera control neuronal**

Como se puede observar en la figura. 3.58., se tiene la respuesta del controlador neuronal con respecto del SetPoint y se puede observar que existe un error en la respuesta de la variable controlada. El error que se tiene es de 1.02 milímetros. También se puede observar que la señal de posición de la esfera no tiene oscilaciones cuando llega al valor deseado.

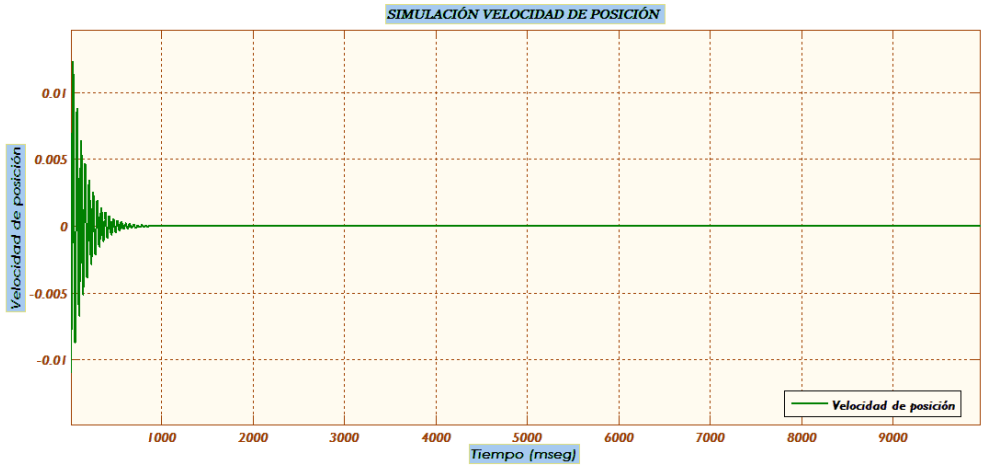



**Figura. 3.59. Velocidad de la esfera.**

La velocidad de la esfera debe llegar a cero una vez que llegue al valor deseado de posición, esto se puede apreciar en la figura 3.59 como la velocidad rápidamente se ubica en cero, pero al inicio de la señal existen variaciones de alta frecuencia que se pierden conforme al tiempo de simulación.

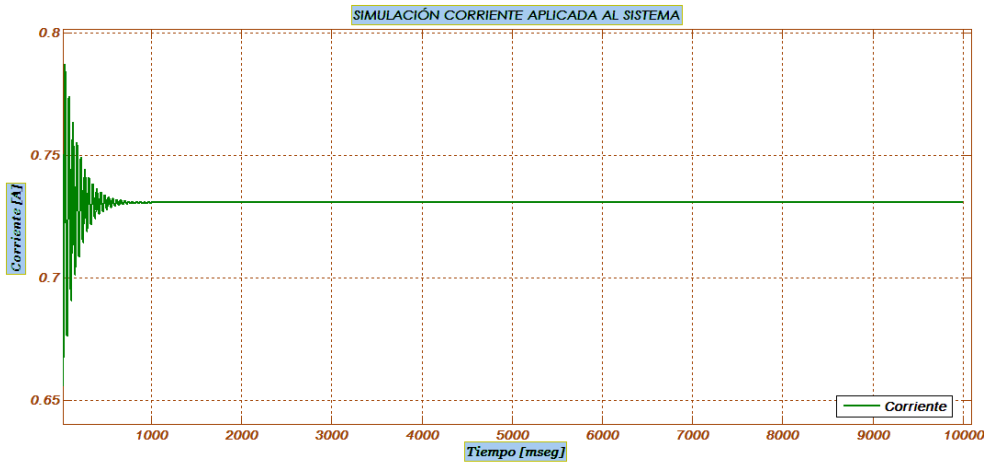



**Figura. 3.60. Corriente de la esfera**

Se puede observar en la figura 3.60, que el valor de corriente necesario para que la esfera se ubique en la posición de 9.02 milímetros es de 0.73 [A].

Una vez concluido con el diseño y simulación de los controladores inteligentes para el sistema de levitación magnética, se continúa con el siguiente capítulo donde se mostrará los resultados de los controladores inteligentes implementados en el modelo experimental.

# CAPÍTULO IV

# IMPLEMENTACION Y RESULTADOS

# CAPÍTULO IV

# IMPLEMENTACION Y RESULTADOS

En el presente capítulo que se presenta la implementación de los controladores inteligentes para el sistema de levitación magnética MLS en el modelo experimental. La implementación consiste en utilizar el modelo de experimentación del levitador magnético.

El modelo de experimentación es un bloque en Simulink que utiliza la herramienta Real Time Work para comunicarse con el actuador electromagnético y los sensores a través de una tarjeta de adquisición.

Para comenzar la implementación de los controladores inteligentes en el sistema de levitación magnética se empieza por obtener la curva característica del sensor.

## 4.1 OBTENCIÓN DE LA CARÁCTERÍSTICA DEL SENSOR

Antes de empezar con la implementación de los controladores, se debe realizar primeramente la identificación, esto se lo realiza con la herramienta de

identificación que contiene y que está previamente definido en el software. Los siguientes pasos detallan el proceso de obtener la curva característica del sensor:

1. Ingresar al software del levitador con el comando **ml_main**.

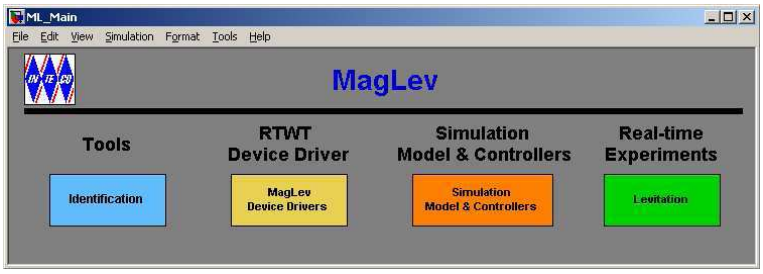



**Figura. 4. Ventana principal del software del levitador magnético**

2. Dentro de la herramienta **Identification** se encuentra cuatro pasos para la identificación, para el proyecto solo se realiza la identificación del sensor para que el sensor este debidamente ajustado a valores reales y no por defecto que ha obtenido el fabricante.

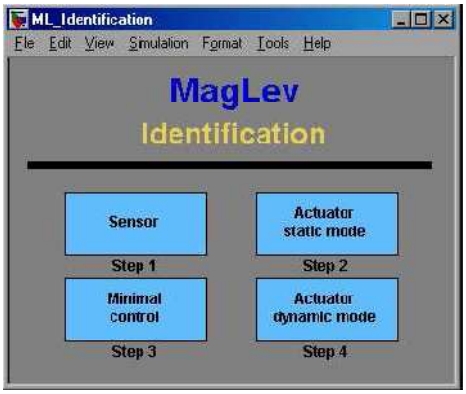



**Figura. 4.1. Ventana de Identificación**

3. En la ventana de la identificación del sensor se tiene una ventana nueva en la cual se toma los datos de la tensión entregada por el sensor desde el magneto vs la distancia y se los grafica.

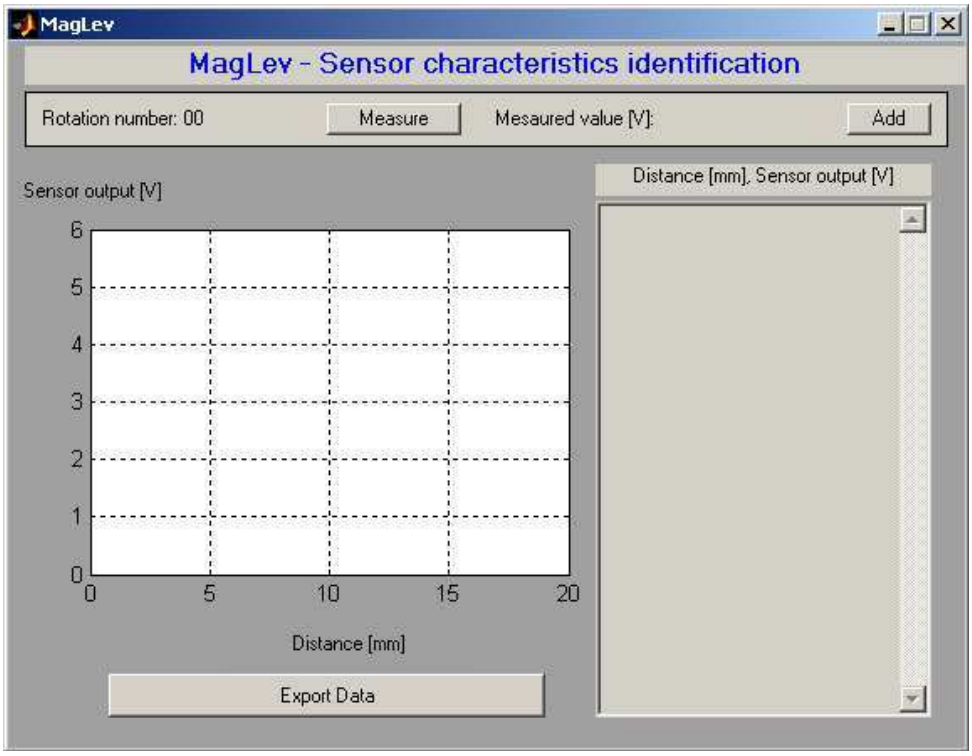



**Figura. 4.2. Señal de sensor [V] vs la distancia de la esfera desde el electro magneto [mm]**

Una vez que se llega a este punto se debe realizar las mediciones con los siguientes pasos:

a) Atornillar la esfera que contiene un tornillo al asiento

b) Mantener fija la esfera con el tornillo que se encuentra en el asiento de soporte para la esfera.

c) Subir el soporte que se encuentra fija con la esfera, de manera que se encuentre cerca del electromagnético la esfera.

d) Encender el sistema de levitación magnético y observar que la luz del sensor traspase a la esfera que está cerca del electro magneto y comenzar con la lectura de los datos, se presiona el botón **Measure** que se encuentra en la venta de la figura. 4.2.

e) Si se presiona el botón **Measure** el sensor hace una lectura de la posición de la esfera con su respectivo voltaje y lo presenta como **Measure Value [V].** Se puede corregir este valor haciendo una lectura nuevamente, posterior a esto se presiona el botón **Add** y el valor es sumado a la lista obteniendo el grafico de Voltaje vs Distancia. Figura. 4.3.

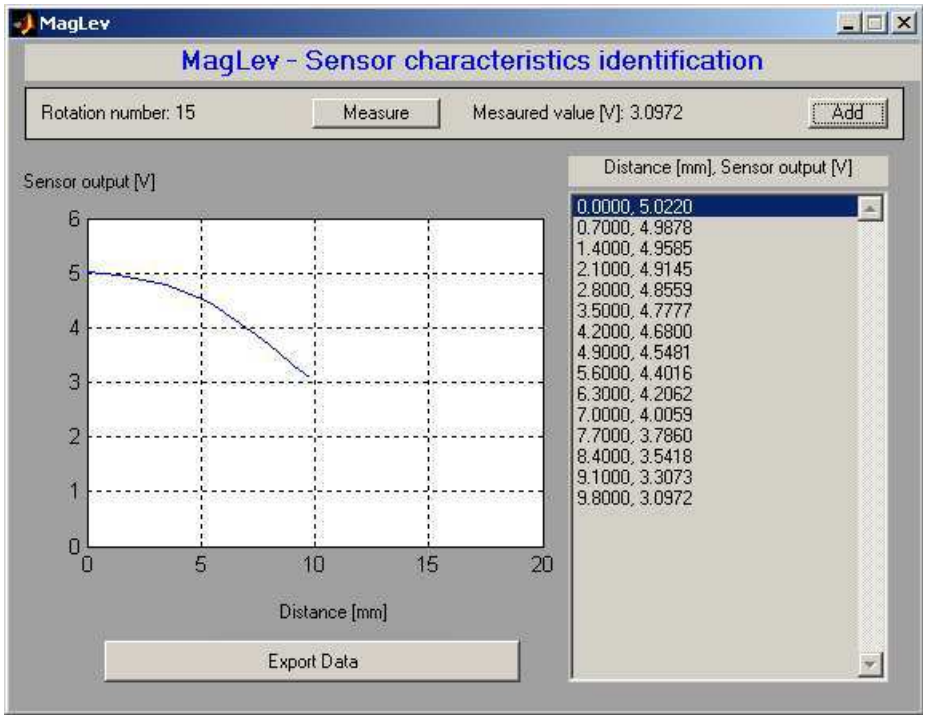



**Figura. 4.3. Características del sensor de posición de la esfera**

f) Para la siguiente lectura, la posición de la esfera debe estar un poco más alejado del electro magneto, y hacer el mismo proceso del punto e).

g) Se debe lograr que el gráfico sea una curva monotónica descendiente

h) Terminar la toma de datos hasta realizar 10 lecturas a variación de posición uniformes.

4. Una vez que se tiene los datos obtenidos se presiona en el botón **Export Data,** para guardar estos datos y los datos serán guardados en un archivo **ML_Sensor.mat** con datos del sensor estructurados con las siguientes señales: *Distancia_mm*, *Distancia_m* y *Sensor_V.*

5. Una vez que se tiene guardado los datos en el archivo ML_Sensor.mat, se debe cargar al sistema estos nuevos datos, en el entorno de trabajo de Matlab con el siguiente comando:

**Load** ML_Sensor

Una vez cargado los datos del archivo ML_Sensor, ya se puede comenzar con la implementación de los controladores utilizando la herramienta SIMULINK.

## 4.2 IMPLEMENTACIÓN DEL CONTROLADOR DIFUSO PROPORCIONAL DERIVATIVO PD EN EL MODELO EXPERIMENTAL

El controlador difuso PD fue diseñado y simulado en el entorno de Simulink con el modelo matemático de simulación de la planta.

Para realizar la implementación con el modelo experimental se necesita actuar sobre el bloque de experimentación que contiene el software en Simulink. Figura 4.4.

La implementación del controlador difuso PD en Simulink se puede apreciar en la siguiente figura 4.4.

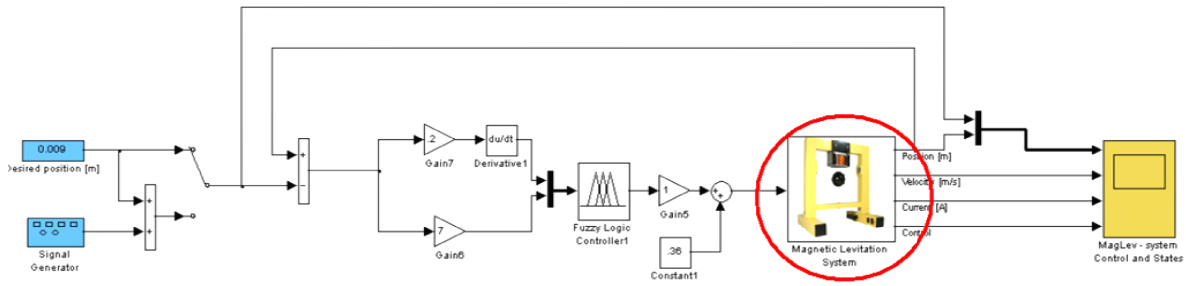



**Figura. 4.4. Sistema de levitación magnético con modelo experimental, controlador PD**

Se debe implementar el diseño del controlador difuso PD dentro del entorno de Simulink como se observa la figura 4.4.

Una vez creado el diseño se procede a cargar el archivo de extensión **.FIS** realizado en la herramienta "fuzzy editor" al bloque del controlador difuso en Simulink.

El comando que se debe aplicar en el entorno de trabajo de Matlab es el siguiente:

**Control_PD=readfis ('fuzzyPD')**

El archivo de extensión **.FIS** es el archivo que contiene el controlador con la base de reglas, valores y variables lingüísticos, que están encargados en realizar el control de la planta.

Una vez que se ha cargado la base de reglas al bloque de control difuso se debe realizar la compilación o construcción del nuevo sistema diseñado, este

proceso es necesario ya que antes de comenzar la simulación en el entorno real Matlab y el sistema de levitación magnético carga los drivers necesarios para poder conectarse con la tarjeta de interfaz.

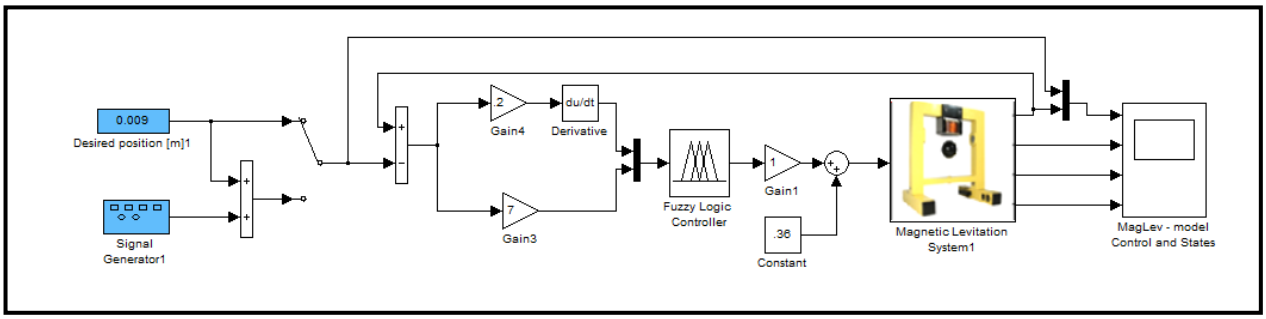



**Figura. 4.5. Controlador difuso PD con el modelo experimental**

Las funciones de membrecía que se presentan a continuación con el editor de funciones son las que fueron disenadas en el capitulo tres y son las siguientes;

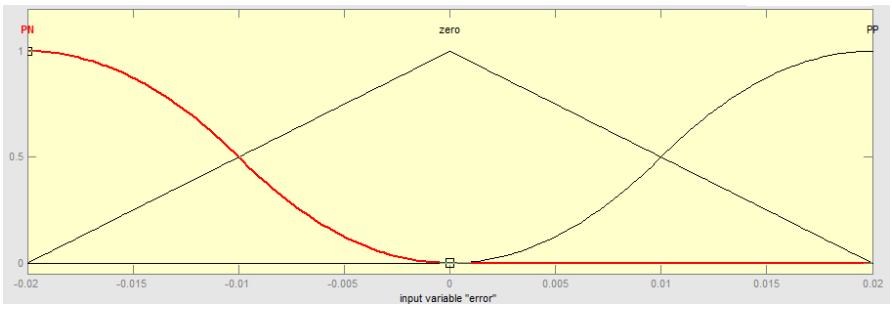



**Figura. 4.6. Funciones de membrecía para la variable Error.**

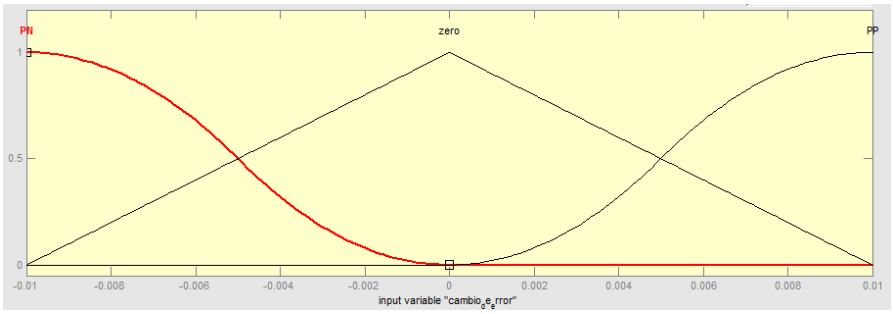



**Figura. 4.7. Funciones de membrecía para la variable Cambio de Error.**

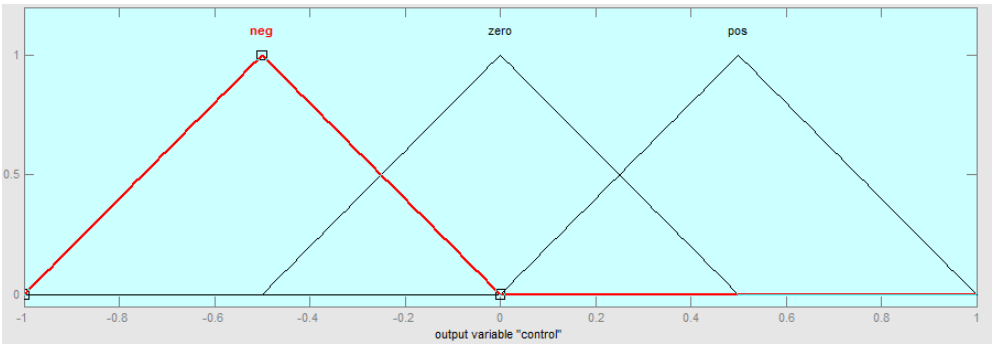



**Figura. 4.8. Funciones de membrecía para la variable Control.**

La base de reglas necesaria para controlador difuso PD es la siguiente:

| | | | | | | |
|---|---|---|---|---|---|---|
| **1** | Si | Error es Pequeño negativo | Y | Cambio de error es Pequeño negativo | Entonces | Control es Pequeño negativo |
| **2** | Si | Error es Pequeño negativo | Y | Cambio de error es Cero | Entonces | Control es Pequeño negativo |
| **3** | Si | Error es Cero | Y | Cambio de error es Pequeño negativo | Entonces | Control Pequeño negativo |
| **4** | Si | Error es Cero | Y | Cambio de Cero | Entonces | Control es Cero |
| **5** | Si | Error es Cero | Y | Cambio de error es Pequeño positivo | Entonces | Control es Pequeño positivo |
| **6** | Si | Error es Pequeño positivo | Y | Cambio de error es Cero | Entonces | Control es Pequeño positivo |
| **7** | Si | Error es Pequeño positivo | Y | Cambio de error es Pequeño positivo | Entonces | Control es Pequeño positivo |
| **8** | Si | Error es Pequeño positivo | Y | Cambio de error es Pequeño negativo | Entonces | Control es Cero |
| **9** | Si | Error es Pequeño negativo | Y | Cambio de error es Pequeño positivo | Entonces | Control es Cero |

**Tabla. 4.1. Base de 9 reglas para controlador difuso PD.**

Los resultados obtenidos en el entorno experimental con el controlador difuso PD se presentan a continuación en la figura 4.9.

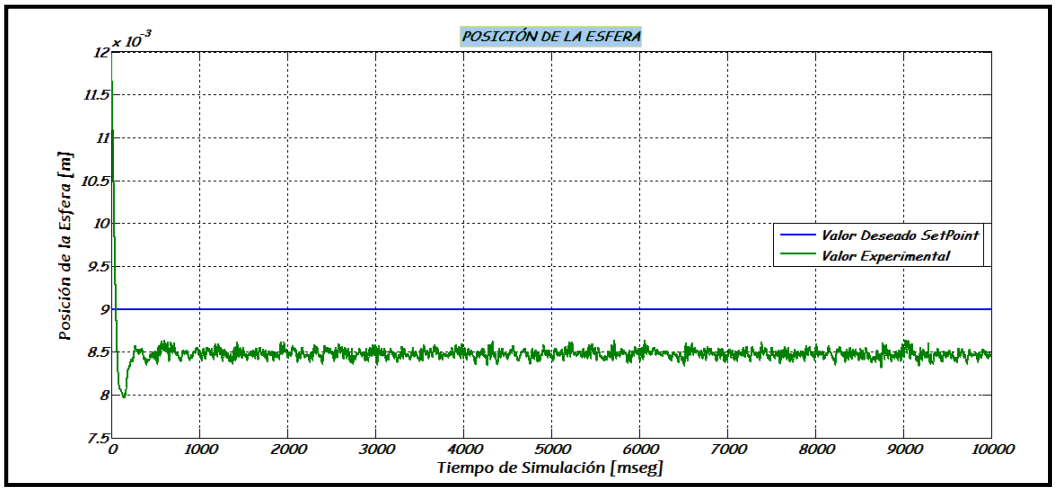



**Figura. 4.9. Posición de la Esfera del Sistema de Levitación magnética**

La respuesta de la posición de la esfera se puede determinar una vez que se realiza la implementación del controlador dentro del modelo de experimentación.

Como se puede observar en la figura 4.9., la respuesta de la posición de la esfera del controlador difuso PD con el modelo de experimentación tiene un error en estado estable de 0.5 milímetros. También se puede observar variaciones u oscilaciones de la esfera cuando llega a la posición de 8.5 milímetros que no afectan a la posición de la esfera.

Para la implementación del controlador difuso PD en el modelo de experimentación se tuvieron que realizar cambios en el diseño. Los cambios que se realizaron fueron en las ganancias del diseño del controlador difuso PD. Como se presentan a continuación la ganancia proporcional y la ganancia derivativa.

- Ganancia proporcional (GP) = 7
- Ganancia Derivativa (GD) = 0.2

Las ganancias fueron obtenidos mediante prueba error al realizar la implementación del controlador difuso PD en el modelo de experimentación, ya que originalmente con las ganancias de simulación no se obtuvieron buenos resultados.

En la implementación del controlador difuso PD en el modelo de experimentación se realizó la prueba del controlador difuso PD respecto al seguimiento de una señal senoidal. Figura. 4.10.

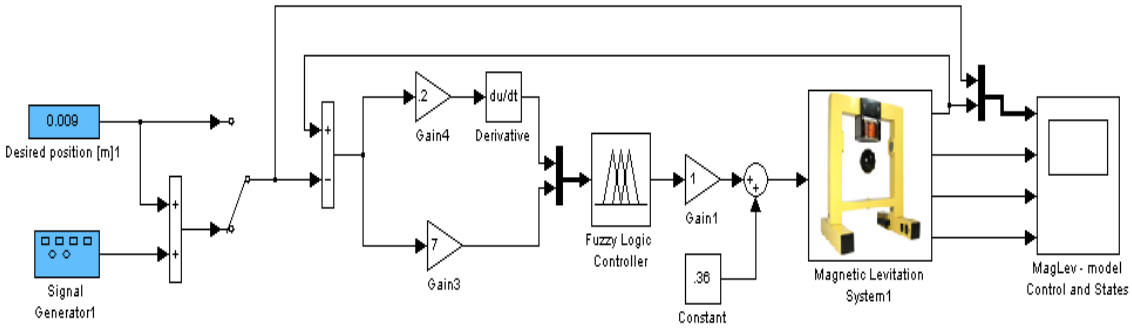



**Figura. 4.10. Seguimiento a una señal senoidal**

El generador de señales proporciona una salida de tipo Seno con un valor de 0.001 m. de amplitud como se presenta en la figura. 4.11.

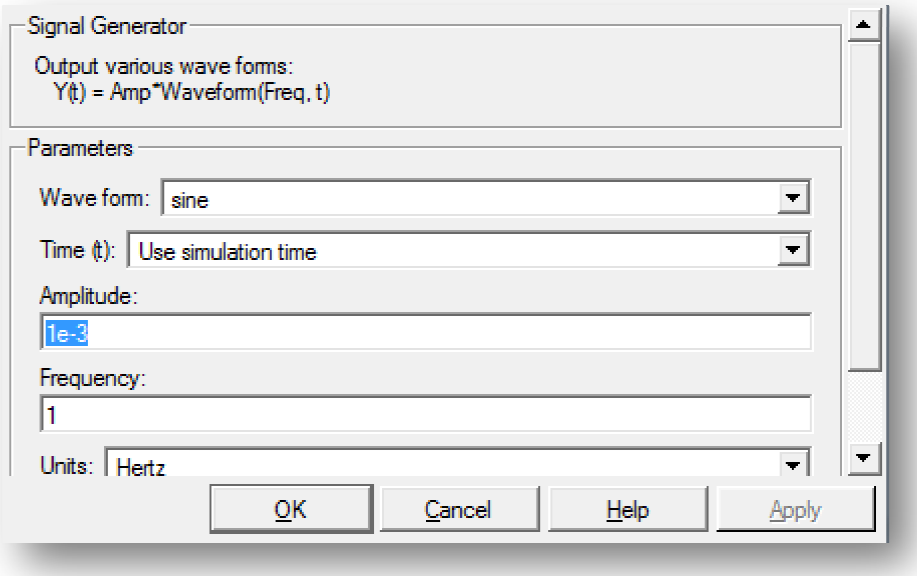



**Figura. 4.11. Valor de amplitud del generador de señales**

El resultado que se obtiene al realizar el seguimiento de la señal es el siguiente;

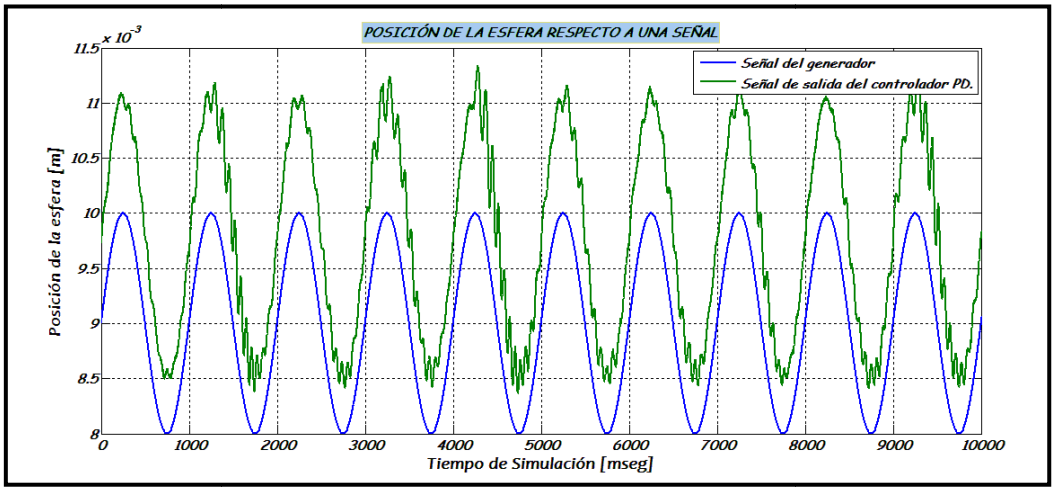



**Figura. 4.12. Respuesta del controlador respecto a una Señal de seguimiento**

Como se puede observar en la figura. 4.12., la posición de la esfera es capaz de seguir a la señal del generador y se tiene seguimiento a la señal pero

con errores. Se observa la presencia de perturbaciones en la señal lo cual se puede atribuir a la característica del sensor. Cuando la posición está más cerca de la bobina se observa que hay una pérdida de control en el sistema.

El desempeño del controlador PD está determinado tanto por el rango difuso de cada variable lingüística así también como por sus ganancias.

## 4.3 IMPLEMENTACIÓN DEL CONTROLADOR DIFUSO PROPORCIONAL INTEGRAL DERIVATIVO PID EN EL MODELO EXPERIMENTAL

La implementación del controlador difuso PID en el modelo de experimentación se lo realiza con la ayuda de Simulink como se presenta en la siguiente figura 4.13.

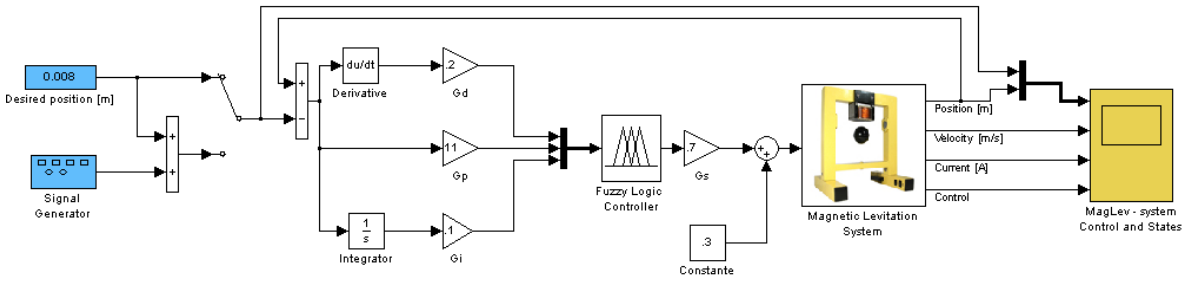



**Figura. 4.13. Controlador Difuso PID en el entorno real**

Como se puede observar en la figura 4.13, los bloques de ganancias no son los mismos que en la simulación del controlador difuso PID.

Los resultados que se presentan en este punto fueron realizados con las 27 reglas que fueron creadas en el capitulo tres.

Las funciones de membrecía que se presentan a continuación con el editor de funciones son las que fueron disenadas en el capitulo tres y son las siguientes;

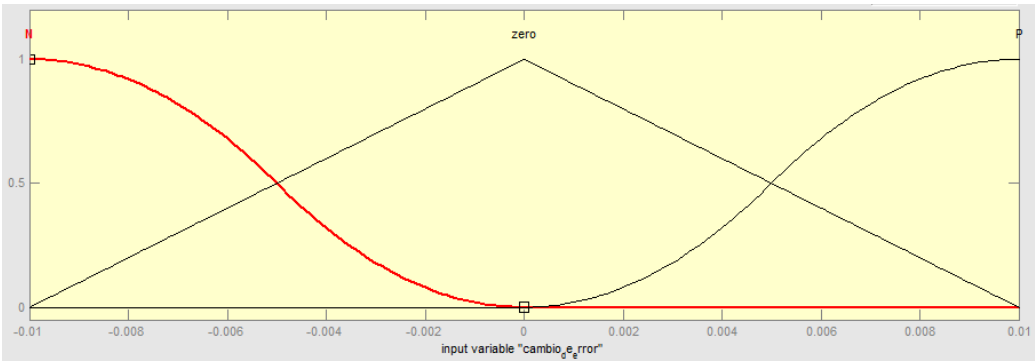



**Figura. 4.14. Funciones de membrecía para la variable Cambio de Error**

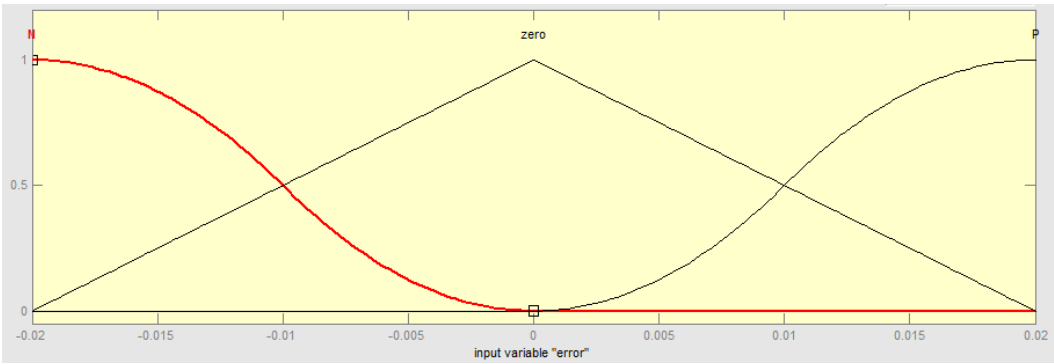



**Figura. 4.15. Funciones de membrecía para la variable Error**

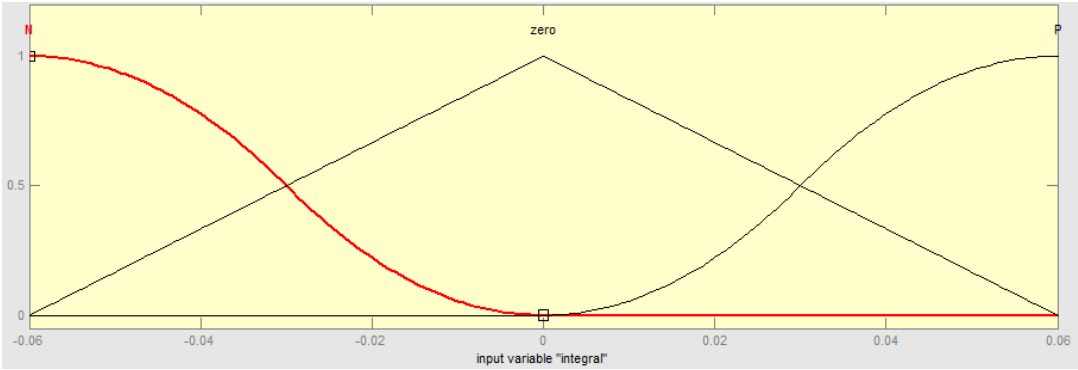



**Figura. 4.16. Funciones de membrecía para la variable Integral**

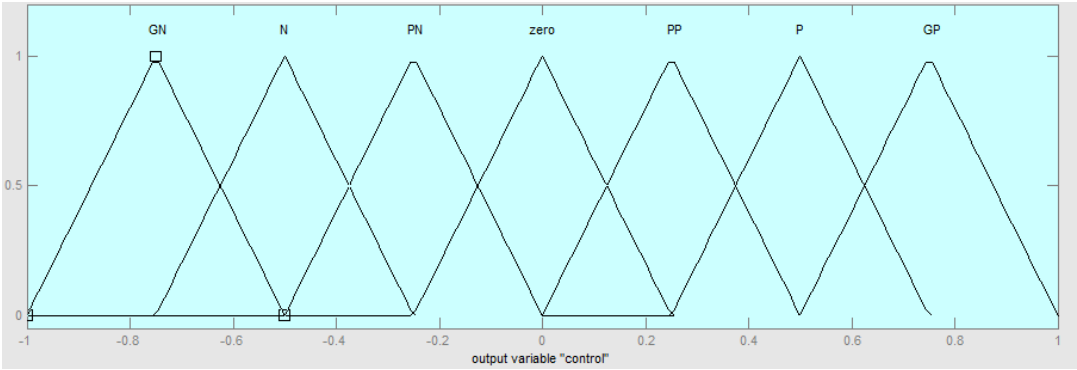



**Figura. 4.17. Funciones de membrecía para la variable de salida Control**

Como se puede observar en la figura 4.17, se tienen siete valores lingüísticos para la variable “control”, lo que da como resultado la base de 27 reglas.

La base de reglas para el diseño del controlador difuso PID es el siguiente:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **1** | Si | Cambio de Error es Positivo | Y | Error es negativo | Y | La integral es negativa | Entonces | Control es negativo |
| **2** | Si | Cambio de Error es cero | Y | Error es negativo | Y | La integral es negativa | Entonces | Control es negativo |
| **3** | Si | Cambio de Error es negativo | Y | Error es negativo | Y | La integral es negativa | Entonces | Control es Grande negativo |
| **4** | Si | Cambio de Error es Positivo | Y | Error es Cero | Y | La integral es negativa | Entonces | Control es cero |
| **5** | Si | Cambio de Error es Cero | Y | Error es cero | Y | La integral es negativa | Entonces | Control es Pequeño negativo |
| **6** | Si | Cambio de Error es negativo | Y | Error es cero | Y | La integral es negativa | Entonces | Control es negativo |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **7** | Si | Cambio de Error es Positivo | Y | Error es positivo | Y | La integral es negativa | Entonces | Control es Positivo |
| **8** | Si | Cambio de Error es Cero | Y | Error es positivo | Y | La integral es negativa | Entonces | Control es Cero |
| **9** | Si | Cambio de Error es negativo | Y | Error es positivo | Y | La integral es negativa | Entonces | Control es negativo |
| **10** | Si | Cambio de Error es positivo | Y | Error es negativo | Y | La integral es cero | Entonces | Control es cero |
| **11** | Si | Cambio de Error es Cero | Y | Error es negativo | Y | La integral es cero | Entonces | Control es Pequeño negativo |
| **12** | Si | Cambio de Error es negativo | Y | Error es negativo | Y | La integral es cero | Entonces | Control es negativo |
| **13** | Si | Cambio de Error es Positivo | Y | Error es cero | Y | La integral es cero | Entonces | Control es Pequeño Positivo |
| **14** | Si | Cambio de Error es Cero | Y | Error es cero | Y | La integral es cero | Entonces | Control es Cero |
| **15** | Si | Cambio de Error es negativo | Y | Error es cero | Y | La integral es cero | Entonces | Control es Pequeño negativo |
| **16** | Si | Cambio de Error es Positivo | Y | Error es positivo | Y | La integral es cero | Entonces | Control es Positivo |
| **17** | Si | Cambio de Error es Cero | Y | Error es positivo | Y | La integral es cero | Entonces | Control es Pequeño Positivo |
| **18** | Si | Cambio de Error es negativo | Y | Error es positivo | Y | La integral es cero | Entonces | Control es cero |
| **19** | Si | Cambio de Error es Positivo | Y | Error es negativo | Y | La integral es positivo | Entonces | Control es Positivo |
| **20** | Si | Cambio de Error es Cero | Y | Error es negativo | Y | La integral es positivo | Entonces | Control es Cero |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **21** | Si | Cambio de Error es Negativo | Y | Error es negativo | Y | La integral es positivo | Entonces | Control es negativo |
| **22** | Si | Cambio de Error es Positivo | Y | Error es cero | Y | La integral es positivo | Entonces | Control es Positivo |
| **23** | Si | Cambio de Error es Cero | Y | Error es cero | Y | La integral es positivo | Entonces | Control es Pequeño positiva |
| **24** | Si | Cambio de Error es negativo | Y | Error es cero | Y | La integral es positivo | Entonces | Control es cero |
| **25** | Si | Cambio de Error es Positivo | Y | Error es positivo | Y | La integral es positivo | Entonces | Control es Grande positivo |
| **26** | Si | Cambio de Error es cero | Y | Error es positivo | Y | La integral es positivo | Entonces | Control es Positivo |
| **27** | Si | Cambio de Error es negativo | Y | Error es positivo | Y | La integral es positivo | Entonces | Control es positivo |

**Tabla. 4.2. Base de 27 reglas para controlador difuso PID**

Como se puede observar en la Tabla 4.2, la base de 27 reglas fue escogida de la simulación para realizar la implementación del control difuso PID.

Los resultados obtenidos en el entorno experimental con el controlador difuso PID se presentan a continuación en la figura 4.18.

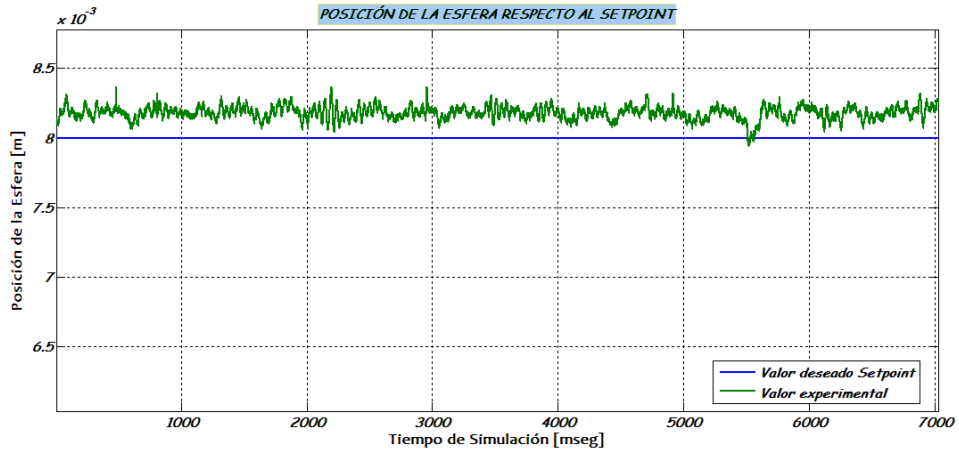



**Figura. 4.18. Posición de la Esfera del Sistema de Levitación magnética**

Como se puede observar en la figura 4.18, la posición de la esfera con respecto al valor deseado tiene un error de 0.2 milímetros demostrando así que el controlador difuso PID de 27 reglas tiene una respuesta aceptable. Pero también se tiene oscilaciones del sistema controlador.

Para la implementación del controlador difuso PID en el modelo experimental las ganancias fueron modificadas para obtener un control óptimo:

- Ganancia proporcional (GP) = 11
- Ganancia Derivativa (GD) = 0.2
- Ganancia Integral (GI) = 0.1

Los valores de ganancias que se obtienen en el controlador difuso PID fueron colocados mediante prueba error al realizar la implementación del controlador difuso PID.

Se realizó la prueba del controlador respecto al seguimiento de una señal senoidal como se observa en la figura. 4.19.

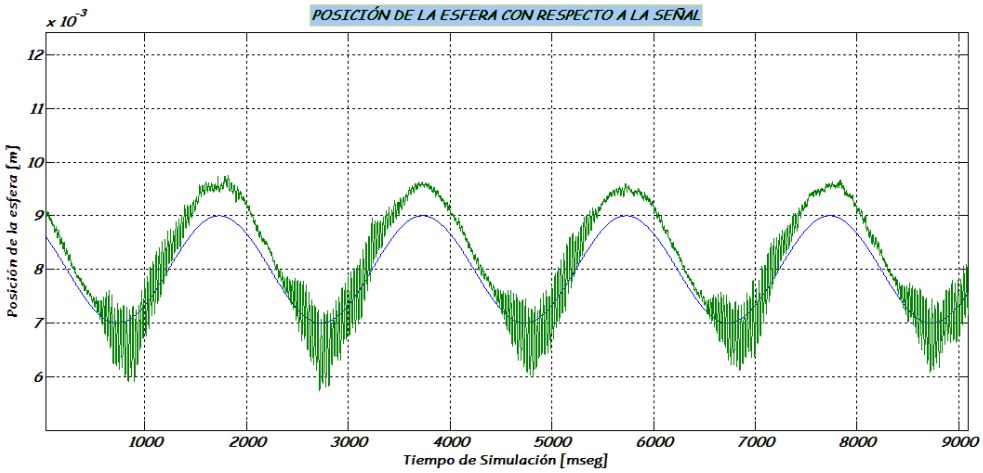



**Figura. 4.19. Respuesta del controlador respecto a una Señal de seguimiento**

Como se puede observar en la figura 4.19., existe seguimiento a la señal pero con errores. Se observa la presencia de perturbaciones de alta frecuencia en la señal lo cual se puede atribuir a la característica del sensor. Cuando la posición es más cerca de la bobina se observa que hay una pérdida de control en el sistema.

La teoría de control difuso no utiliza el modelo matemático, es una metodología que proporciona una manera simple y elegante de obtener una conclusión a partir de información de entrada, ambigua, imprecisa, o incompleta, en general la lógica difusa modela como una persona toma decisiones basada en información con las características mencionadas.

De esta manera es que se realizó la implementación de los controladores difusos en el entorno experimental del sistema de levitación magnética.

El siguiente punto, es realizar la implementación del control neuronal en el entorno experimental.

## 4.4 IMPLEMENTACIÓN DEL CONTROLADOR DE RED NEURONAL EN EL MODELO EXPERIMENTAL

La implementación del controlador de red neuronal es simple, y lo que se debe realizar es colocar el controlador de red neuronal ya previamente simulado en reemplazo del controlador PID clásico del sistema. Figura 4.20.

Se debe realizar la construcción del sistema de levitación para que Matlab carguen los respectivos drivers de comunicación con la tarjeta de interfaz.

La implementación del sistema dentro del entorno experimental se detalla en la siguiente figura. 4.20.

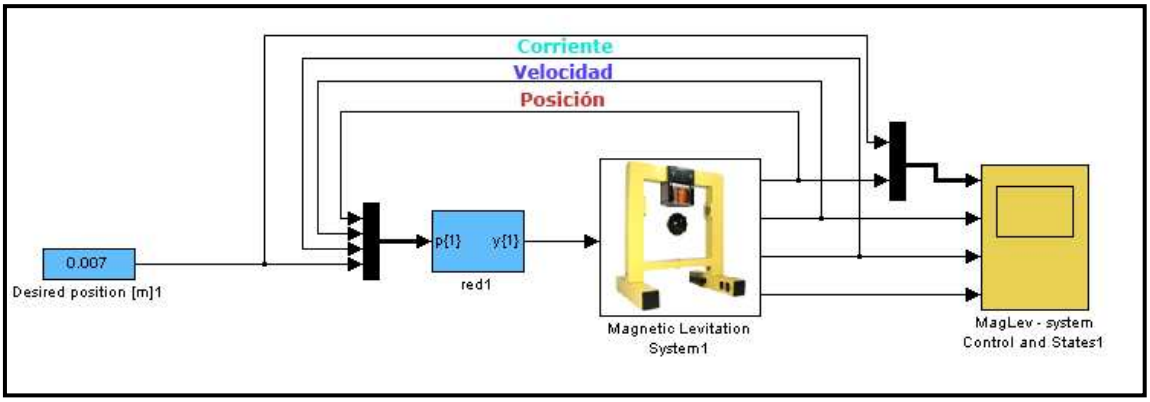



**Figura. 4.20. Controlador de Red Neuronal dentro del entorno experimental**

Como se puede observar en la figura. 4.20., el bloque de la red neuronal se encuentra diseñado con las entradas de posición, velocidad y corriente del

sistema de levitación que alimentan al controlador y que son necesarios para que el controlador pueda realizar su funcionamiento.

Al realizar la compilación de todo el diseño con la herramienta Simulink, se procede a obtener los resultados de la posición de la esfera del controlador como se observa en la figura. 4.21.

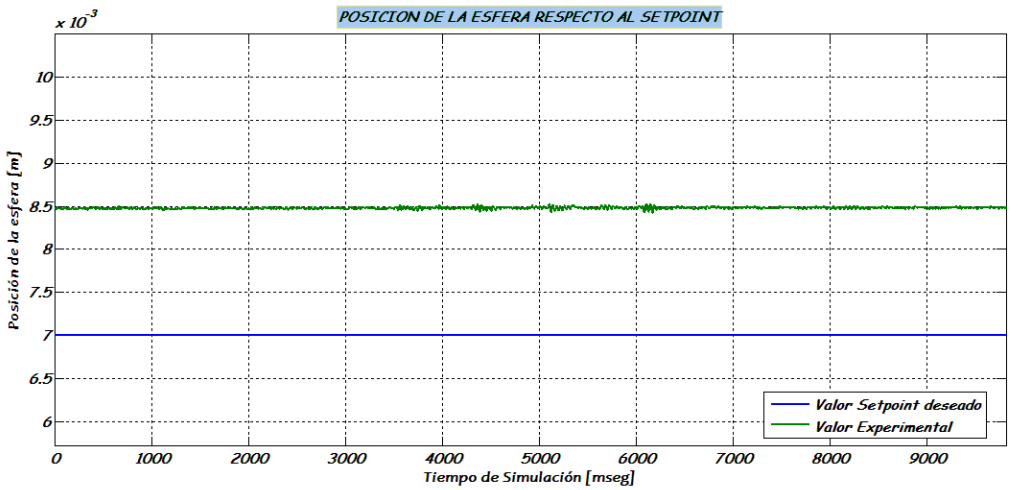



**Figura. 4.21. Respuesta del controlador respecto al SetPoint**

Como se puede observar en la figura 4.21., la respuesta de posición de la esfera del controlador tiene un error en estado estable de 1.5 milímetros respecto al SetPoint. También se puede observar que la señal de posición no tiene oscilaciones, dando como evidencia que el control con redes neuronales para el sistema de levitación que es complejo es más estable que los controladores difusos.

Se presenta la señal de respuesta de la velocidad y corriente del sistema en la figura. 4.22.

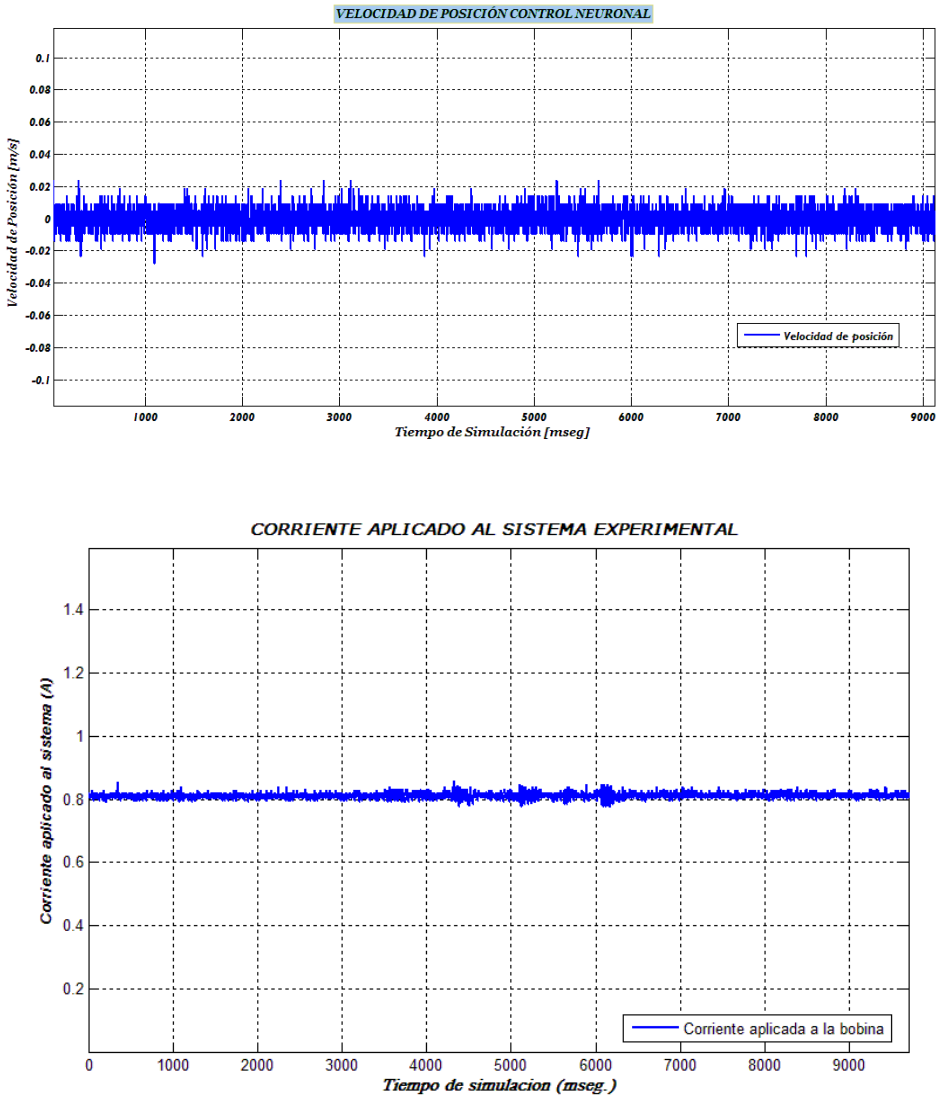



**Figura. 4.22. Respuesta de velocidad y corriente del controlador neuronal en el entorno experimental**

Se puede observar en la figura 4.22, que al realizar el control sobre la planta se tiene en la velocidad de posición oscilaciones alrededor de cero. En la figura se observa también que la corriente necesaria para estabilizar el sistema en 8.5mm es 0.8 [A].

Con esto se concluye los resultados que se obtuvieron en el proyecto esperando que sean de ayuda para futuros trabajos sobre el levitador magnético.

# CAPÍTULO V

# CONCLUSIONES Y RECOMENDACIONES

# CAPÍTULO V

# CONCLUSIONES Y RECOMENDACIONES

## 5.1 CONCLUSIONES

**1.** Los sistemas de levitación electromagnética tienen como principal reto el control de su distancia de flotación, distancias muy pequeñas producen grandes inestabilidades lo cual hacen el control casi imposible y a su vez distancias grandes requieren grandes cantidades de energía. El control se hace difícil debido a que la dinámica del sistema es muy inestable, no lineal, además de que tiene una región de estabilidad muy pequeña. Al aplicar la teoría difusa al sistema de levitación magnética se debe añadir ganancias a las entradas del controlador como son sus ganancias que por defecto alteran el universo de discurso de las variables lingüísticas. En el diseño de la red neuronal, se debe configurar los parámetros con los que se va a realizar el entrenamiento.

**2.** Se ha implementado un controlador PID difuso con el cual se ha podido obtener una respuesta satisfactoria en cuanto a seguimiento de referencia así como también una referencia constante, de igual manera se obtiene resultados positivos en cuanto a perturbaciones

dentro del diseño de controladores inteligentes como fue el caso de utilizar las tres esferas de diferente tamaño y peso dentro del sistema con lo cual no se presento problemas al realizar la levitación de las mismas, concluyendo que dentro del diseño de los controladores inteligentes no es necesario actuar sobre el modelo matemático como se lo realiza en el control convencional. Cabe destacar que el modelo matemático fue utilizado para realizar las respectivas simulaciones del sistema para después pasar al sistema experimental y es por esta razón el modelo matemático debe ser lo más aproximado al real.

**3.** La implementación de un controlador difuso PD con nueve reglas es suficiente para realizar el control sobre el sistema de levitación magnética ya que al ajustar las ganancias y los conjuntos difusos, se puede mejorar hasta un punto determinado el cual es el límite o el rango extremo a controlar, al pasar este limite el controlador se vuelve inestable y no logra controlar en todo el rango definido, es por este motivo que se debe mantener los valores propuestos dentro de este trabajo.

**4.** La implementación de un controlador difuso PID de 27 reglas fue uno de los mejores controladores y con el que se tiene mejores resultados en cuento a simulación, mientras que en el sistema real no fue tan preciso como lo fue en el simulación, y esto se puede deber a que el modelo matemático de simulación no es lo suficientemente aproximado al real, cabe recalcar que este modelo matemático de simulación fue obtenido de trabajas previos.

**5.** Para el diseño de controladores neuronales con modelo de referencia es importante realizar el modelo de simulación neuronal de la planta ya que por este medio es que la red neuronal conjuntamente con el

controlador obtienen el conocimiento del modelo al que se quiere referir que se obtenga el respectivo conocimiento.

**6.** Como se puede observar en el presente trabajo en relación a las redes neuronales; el modelo de simulación no es exactamente igual que el real ya que no siempre es posible conocer con exactitud la función de transferencia de los componentes y mucho menos de todo el sistema por lo que es necesario utilizar herramientas diferentes a las tradicionales conocidas para desarrollar el control. Una de estas herramientas son las redes neuronales, estas no necesitan conocer de manera directa el comportamiento de cada uno de los componentes del sistema para emular al sistema en su conjunto. Lo único que necesitan son datos para "aprender" el comportamiento deseado del sistema. También es cierto y obvio que las redes neuronales tienen sus limitantes, ya que no pueden presentar un comportamiento que no se les ha "enseñado".

**7.** A través de este proyecto se ha comprobado que las redes neuronales pueden emular el comportamiento de un controlador PID tradicional para controlar el sistema de levitación magnética. Se ha visto que si los datos obtenidos para entrenar la red neuronal son erróneos, la red arrojará resultados erróneos. Si los datos son adecuados, el comportamiento de la red será por lo general el esperado.

**8.** Para realizar el diseño de una red neuronal para este sistema es necesario poseer de una gran capacidad de procesamiento ya que el proceso de realizar varias interacciones en épocas largas de entrenamiento de la red neuronal almacena mucha información para poder llegar al objetivo.

**9.** La configuración de la red neuronal es también un punto clave para que una aplicación sea exitosa. Si los datos de entrenamiento son correctos,

pero la arquitectura de la red no es la adecuada para la aplicación, difícilmente la red neuronal se comportará como se espera.

**10.** En cuanto a redes neuronales se puede observar que el sistema es mucho más estable con respecto a rechazo de perturbaciones siendo capaz de utilizar las tres esferas de diferente tamaño y peso como una perturbación; la cual no fue problema para el controlador en estabilizar las esferas en el punto de referencia especificado, este tipo de controladores supone una clara alternativa a los sistemas clásicos de control tomando en cuenta que este tipo de controladores necesita de una gran capacidad de procesamiento para su entrenamiento.

**11.** La tecnología electromagnética de levitación aplicada a vehículos es muy prometedora para un futuro próximo en el que los combustibles fósiles sean escasos, el avance tecnológico en esta área del saber humano es entonces indispensable. Las ventajas que esta tecnología aporta, como minimizar la fricción; capacidad de alcanzar grandes velocidades y ser ecológicamente responsable entre otras, la hace estar a la vanguardia en sistemas de transporte de alta velocidad.

## 5.2 RECOMENDACIONES

**1.** Realizar el ajuste de identificación del sistema para no tener errores en la lectura de la posición de la esfera ya que esto es muy importante al crear un nuevo sistema de control.

**2.** Al realizar la implementación de los controladores dentro del sistema real, siempre realizar la construcción del controlador ya que por medio de esta construcción el sistema acepta o rechaza algún error que

puede existir, es decir es una compilación que realiza SIMULINK del controlador.

**3.** Permitir buena circulación de aire alrededor de la interfaz de potencia para el correcto enfriamiento de los componentes electrónicos.

# REFERENCIAS BIBLIOGRAFICAS


- HOWARD DEMUTH, Mark Beale, **Neural Network Toolbox**, Versión 3, Quinta edición, Newsgroup, Enero 1998 Mathwork, 219 páginas.

- PASSINO / YURKOVICH, Kevin M. / Stephen, **Fuzzy Control**, Versión 1, Primera edición, Addison Wesley, Febrero 1998 Wesley, 522 páginas.

- http://www.cs.bham.ac.uk/~jxb/inn.html, Introduction to neural networks.

- **http://homepage.cem.itesm.mx/aaceves/publicaciones/levitador-LEA400.pdf,** Sistema de levitación magnética controlado con lógica difusa y control clásico.

- http://recyt.fecyt.es/index.php/RIAII/article/view/RIAI.2010.03.06/7178, Aplicación de Control borroso a un sistema de Suspensión magnética.

- http://ewh.ieee.org/sb/argentina/comahue/ed1/Papers/1EdRate05/TRATE05-008.pdf, Ensayos de Controlador No lineal sobre un Levitador magnético.

# ANEXOS

# ANEXO 1

# CODIGO EN MATLAB, REDES NEURONALES PARA EL SISTEMA DE LEVITACIÓN MAGNÉTICA

# INDICE DE FIGURAS

# INDICE DE TABLAS

# Glosario

- **Conjunto,** Es una colección de elementos (reales o imaginarios) considerados como un todo.

- **Conjunto borroso,** Es aquél en que cada elemento tiene un grado de pertenencia asociado, dicho grado es un número real.

- **Conjunto clásico,** Es aquél en que cada elemento tiene asignado un grado de pertenencia, 1 si el elemento pertenece al conjunto y 0, si el elemento no pertenece a dicho conjunto.

- **Función de pertenencia,** Es una función que indica el grado de pertenencia de un elemento a un conjunto.

- **Grado de pertenencia,** Es un valor numérico, con el cual se expresa la medida en que un elemento pertenece a un conjunto.

- **Lógica Borrosa,** Es un tipo de lógica que utiliza información de entrada vaga e imprecisa para extraer conclusiones. Mediante ella se definen conceptos que no pueden ser formulados de forma precisa.

- **Universo,** Conjunto de elementos cualesquiera en los cuales se consideran una serie de características a estudiar.

- **Universo de discurso,** universo en el que se ejecuta la acción de control

- **Real-time**, sistema de tiempo real.

- **ANNs,** Red neuronal artificial (Artificial neuronal network)

# ACTA DE ENTREGA

El proyecto fue entregado al Departamento de Eléctrica y Electrónica y reposa en la Escuela Politécnica del Ejército desde:

Sangolquí, a_____

_____

Ing. Víctor Proaño

COORDINADOR DE CARRERA

_____

Sr. Cristian Noé Zúñiga Yunda