INSTITUTO TECNOLOGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY
CAMPUS MONTERREY
PROGRAMA DE GADUADOS DE LA DIVISION DE
ELECTRONICA, COMPUTACION, INFORMACION
Y COMUNICACIONES

CONTROLADOR DE ELEVACION PARA UNA ANTENA SATELITAL: EVALUACION DEL CONTROLADOR CONVENCIONAL PD, EL CONTROLADOR DIFUSO TIPO PD Y EL CONTROLADOR DESLIZANTE DIFUSO

T E S I S

PRESENTADA COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO ACADEMICO DE
MAESTRO EN CIENCIAS
ESPECIALIDAD EN SISTEMAS INTELIGENTES

LAURO RUBEN RODRIGUEZ BRAVO

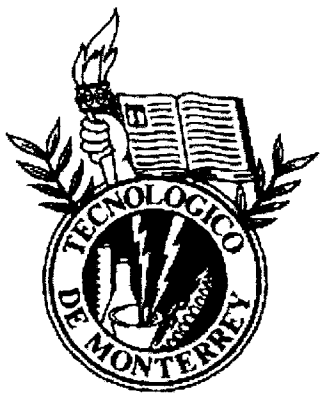
DICIEMBRE DE 2001

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

DIVISIÓN DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN,

INFORMACIÓN Y COMUNICACIONES



CONTROLADOR DE ELEVACIÓN PARA UNA ANTENA SATELITAL: EVALUACIÓN DEL CONTROLADOR CONVENCIONAL PD, EL CONTROLADOR DIFUSO TIPO PD Y EL CONTROLADOR DESLIZANTE DIFUSO.

T E S I S

PRESENTADA COMO REQUISITO PARCIAL

PARA OBTENER EL GRADO ACADEMICO DE

MAESTRO EN CIENCIAS

ESPECIALIDAD EN SISTEMAS INTELIGENTES

LAURO RUBÉN RODRÍGUEZ BRAVO

DICIEMBRE DEL 2001

Controlador de elevación para una Antena Satelital: Evaluación del controlador Convencional PD, el controlador Difuso tipo PD y el controlador deslizante Difuso

Por

Lauro Rubén Rodríguez Bravo

Tesis

Presentada al Programa de Graduados en Electrónica, Computación, Información y Comunicaciones

Del

Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Monterrey

Como requisito parcial para obtener el grado académico de

Maestro en Ciencias

Especialidad en Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Diciembre del 2001

Controlador de elevación para una Antena Satelital: Evaluación del controlador Convencional PD, el controlador Difuso tipo PD y el controlador deslizante Difuso

Lauro Rubén Rodríguez Bravo, M.C.
Instituto Tecnológico y de Estudios Superiores de Monterrey, 2001

Asesor de la tesis: Rogelio Soto, Ph. D.

El trabajo de investigación que se presenta a continuación trata el problema que tienen en la mayoría de las sedes receptoras de la Universidad Virtual del ITESM cuando desean sintonizar las antenas parabólicas que permiten recibir la señal de la Universidad (video y audio). El problema es que no conocen los equipos y no tienen los conocimientos necesarios para realizar dichos ajustes.

La solución que se propone es diseñar un controlador convencional, un controlador difuso tipo PD y un controlador deslizante difuso de dos entradas - una salida. Se evalúan los tres tipos de controladores.

La evaluación se realiza por medio de simulaciones en Matlab y con apoyo de Labwindows/CVI se deja todo lo necesario para realizar la implantación en tiempo real del controlador mejor evaluado.

Índice General

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

El capítulo 1 nos permite saber los antecedentes que nos llevan a definir el problema y la solución que se propone para resolverlo, así como el prototipo a implantar al final de esta tesis.

1.1. Antecedentes

La educación ha tenido una presencia muy importante en nuestras vidas y en el mundo en general y gracias a la tecnología cada vez más personas tienen la oportunidad de tener acceso a una mejor educación sin importar el factor de la distancia. El Instituto Tecnológico y de Estudios Superiores de Monterrey siguiendo con la misión de formar personas comprometidas con el desarrollo de su comunidad para mejorarla en lo social, lo económico y lo político y que sean competitivas internacionalmente en su área de conocimiento, crea la Universidad Virtual. La Universidad Virtual es una institución de educación superior basada en un sistema de enseñanza – aprendizaje que opera a través de las más avanzadas tecnologías de telecomunicaciones y redes electrónicas. Ofrece educación a través de innovadores modelos educativos y de vanguardia para apoyar el desarrollo de México y América Latina.

Una de las direcciones con las que cuenta la Universidad Virtual es la Dirección de Telecomunicaciones encargada de proporcionar la infraestructura y los servicios de tecnología más avanzados que permitan a la Universidad Virtual y a sus sedes establecer exitosamente sus modelos educativos y contar con la información necesaria para una operación y toma de decisiones eficientes. La Dirección cuenta con diferentes departamentos como es el Departamento de Tecnologías Interactivas cuya misión es administrar la infraestructura tecnológica de telecomunicaciones que soporta a los modelos educativos y de operación de la UV evaluando y proponiendo nuevas tecnologías de vanguardia. Dentro de sus principales funciones esta el "Apoyo y mantenimiento de equipo de transmisión y recepción satelital en Monterrey y campus receptores". Tiene la obligación de verificar que la señal que se transmite y recibe sea de la mejor calidad posible. Además de encargarse de realizar la sintonización o ajuste de las antenas para algún evento en especial o cuando la señal ya no cumple con las características deseadas.

Como parte esencial de este esquema se encuentra el sistema receptor como se muestra en la Figura 1.1, el cual típicamente utiliza un reflector parabólico, el disco, diseñado para recibir y concentrar la señal débil que se recibe del satélite. El alimentador en donde llegan todas las señales recolectadas por la antena para pasar al LNB (Low Noise Block Converter) donde la señal de RF en banda C o Ku se convierte en banda L y es enviada por medio del cable al receptor satelital donde se procesa para ser una señal que se puede alimentar a un televisor o una red de cable según sean las necesidades[2].

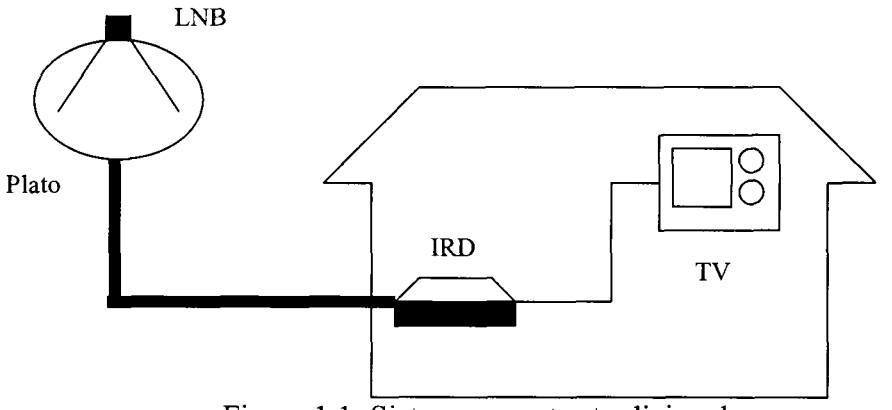


Figura 1.1: Sistema receptor tradicional.

Esto también se realiza en los demás centros receptores pero uno de los problemas que se presentan constantemente es de no contar con los conocimientos necesarios o experiencia para hacer este tipo de ajustes por lo que dependen de otras personas que se contratan por fuera y que muchas veces ni ellos tienen el conocimiento necesario para cubrir las necesidades. Debido a esto, se termina dando asesoría por teléfono desde el Campus Monterrey haciendo las cosas más difíciles y tediosas provocando en algunos momentos llegar a perder la paciencia y dejando una mala imagen ante las sedes. En algunos casos se llegan a perder a los clientes si el problema es reincidente.

Entonces el problema que se presenta es ¿cómo sintonizar una antena en términos de polaridad y posición para poder recibir señales satelitales digitales en forma automática?

Dentro del campo de la Inteligencia Artificial se tienen diferentes técnicas que nos permiten atacar una gran variedad de problemas que no se pueden resolver adecuadamente con otras herramientas. Dentro de estas técnicas están los algoritmos genéticos, redes neuronales, lógica difusa, entre otras. Una de las áreas de gran interés es la que tiene que ver con el Control Difuso el cual cuenta con la Lógica Difusa como principal fundamento teórico. Los sistemas difusos han sido aplicados en una gran variedad de campos para el control, procesamiento de señales, comunicaciones, fabricación de circuitos integrados, sistemas expertos de negocios, medicina, psicología [6].

El Control Inteligente es un sistema experto en tiempo real que implementa una parte del conocimiento o experiencia del operador en el proceso a controlar de manera automática y que los controladores PID no pueden representar pero si reglas acción / reacción. Una de las principales ventajas de la Lógica Difusa es que nos permite trabajar por medio de reglas "Sí - entonces" y cualitativas, por ejemplo:

"Sí (Error es Positivo Pequeño) y (Derivada es Cero) entonces (salida es Positivo Pequeño)"

Este tipo de reglas es lo que hace muy atractiva esta herramienta porque se puede manejar un lenguaje natural para definir la base de reglas del sistema difuso.

Tomando en cuenta todo esto, se pretende diseñar e implementar un Controlador que permita realizar el ajuste de señal de una antena satelital de banda Ku. Esto se puede hacer realizando la modificación a una antena para poder realizar el ajuste de elevación teniendo un controlador de Dos entradas - una salida con punto de referencia la calidad de la señal que se recibe por medio de un receptor análogo o digital.

En las siguientes Secciones se presenta en forma más detallada la descripción del problema, el planteamiento de la hipótesis, los antecedentes de la investigación, la metodología a utilizar, y los resultados obtenidos durante las simulaciones.

Uno de los problemas que se presenta cuando se trabaja con estos sistemas es que los equipos (receptor y antena) no se encuentran cerca uno del otro. La mayoría de las veces los equipos se instalan en lugares donde este tipo de servicio no se tiene contemplado al inicio y las instalaciones físicas no son las mejores por lo que la antena queda alejada o en el exterior del edificio donde se tienen los receptores satelitales así que se depende generalmente de dos o más personas comunicadas por radio lo que provoca que sea incomodo trabajar. Otra forma es hacer una instalación temporal usando la combinación de un receptor y un monitor de TV.

Al empezar a realizar el ajuste, la persona debe de saber exactamente la posición del lugar donde se encuentra ubicada la antena o al menos la ciudad para poder obtener una serie de datos que le permitirán realizar el ajuste de mejor manera. Esto es, su posición geográfica, le permite saber donde se encuentra ubicado el satélite con respecto a su ubicación, así como la elevación que debe de tener la antena con respecto al horizonte.

Todos estos datos se pueden sacar teóricamente por medio de fórmulas que relacionan la ubicación del satélite y la ubicación del lugar.

Ángulo de declinación

El ángulo de declinación es un factor de ajuste que permite ubicar de mejor forma el arco de satélites, es la distancia entre el piso y el disco de la antena como se puede ver en la Figura 1.2 Mientras más lejos se esta del Ecuador, más grande es este ángulo[1]. Para poder calcular este dato existen tablas o podemos usar la siguiente ecuación[1]:

$$Declinación = \tan^{-1} \frac{3964 senL}{22300+3964(1-\cos L)} \qquad (1.1)$$

Donde L es la latitud del lugar en grados. Los otros dos números de la ecuación son el radio de la tierra y la distancia entre la superficie de la tierra y el arco de satélites en Km.

Ángulo de elevación total

El ángulo de elevación como se puede ver en la Figura 1.2 es el que se ajusta inclinando el disco de la antena al mismo valor de latitud del lugar más el ángulo de declinación. El ángulo de elevación parcial se puede calcular con las siguientes ecuaciones[3]:

$$Azimuth = 180^{\circ} + \tan^{-1}\frac{\tan G}{senL} \tag{1.2}$$

$$Elevación = \tan^{-1}\left[\frac{\cos(G)\cos(L) - 0.1512}{\sqrt{1-\cos^2(G)\cos^2(L)}}\right] \tag{1.3}$$

E = Elevación de la antena parcial
S = Longitud del satélite en grados
N = Longitud del sitio en grados
L = Latitud del sitio en grados
G = S – N

Ángulo de polaridad o skew

Es el ángulo al cual se debe de girar el LNB, el encargado de hacer la conversión de frecuencia en GHz a MHz para poder recibir el canal en la polaridad adecuada. Se calcula con la siguiente ecuación[1]:

$$\cos(skew) = \frac{senL(1 - A\cos L\cos G)}{\sqrt{(\cos^2 Lsen^2 L)(A^2\cos^2 L + 1 - 2A\cos G\cos L)}} \tag{1.4}$$

Por desgracia la mayoría de las sedes contratan a personas ajenas a la sede para realizar este tipo de trabajos y los cuales han aprendido a realizar su trabajo por medio de la experiencia y con la forma tradicional de hacer muchas cosas, a prueba y error. Los problemas empiezan cuando las personas encargadas de atender este tipo de equipos no cuentan con conocimientos o tiempo necesarios para aprender por su cuenta. Una vez que se tiene la antena en la inclinación y posición adecuada, se realiza el ajuste fino de la señal moviendo la antena poco a poco sobre el arco polar de visión de los satélites y ajustando al mismo tiempo la polaridad de la señal que se desea recibir.

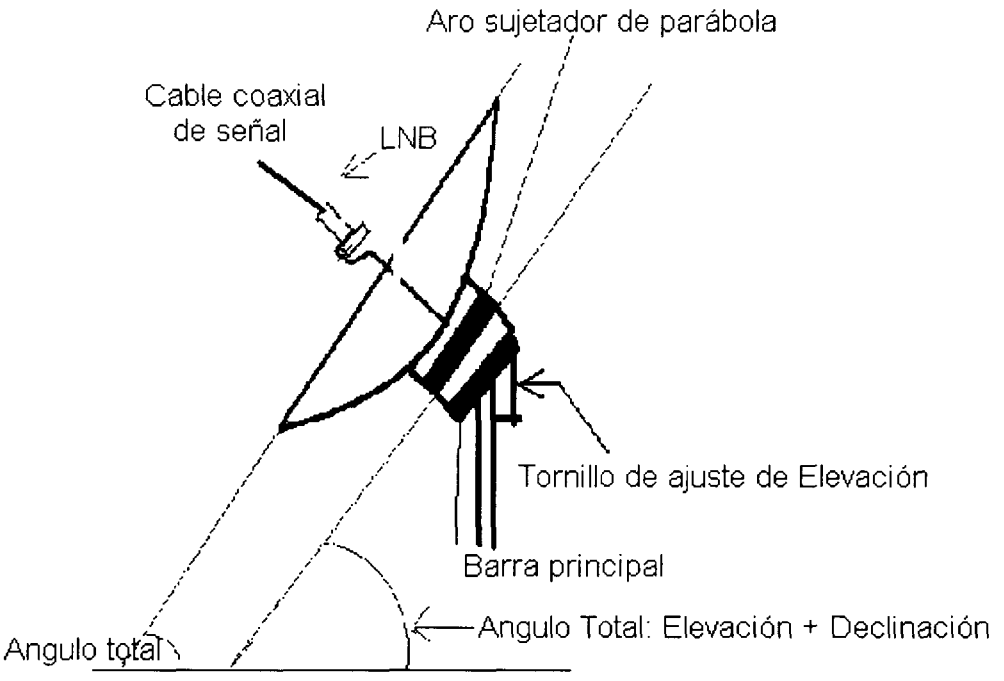


Figura 1.2: Ajuste y partes de una antena

Las señales que se pueden recibir con este tipo de equipo pueden ser de diferentes frecuencias. Una de ellas es la Banda C que son señales dentro del rango de frecuencias 3.7 a 4.2 GHz polarizadas. Esta banda es de las más utilizadas para transmitir señales análogas de televisión. La otra banda con la que se cuenta es la Banda Ku que son señales en el rango de 10.7 a 12.75 GHz polarizadas. Dependiendo de la banda donde estemos trabajando y la región geográfica, es el tamaño de antena que se utiliza para poder recibir las señales. Las antenas para banda Ku son más pequeñas y generalmente son utilizadas para aplicaciones "Directo a casa" (Direct to Home).

Las ondas polarizadas son ondas electromagnéticas que pueden ser lineales o circulares. En la polarización lineal, el campo eléctrico y magnético de la señal son perpendiculares entre ellas y permanecen en el mismo plano por el cual son originalmente transmitidas. En la polarización circular el campo eléctrico y magnético rotan en forma circular mientras viajan por el espacio[2].

Muchos de los equipos receptores análogos traen consigo una función para poder realizar el ajuste fino de la antena. Esto es relativamente fácil cuando se trabajan en banda C y cuando se sabe lo que sé esta haciendo. Cuando una persona tiene experiencia y suerte este tipo de ajustes en estas condiciones le pueden llevar más de una hora.

El problema se complica más cuando se tratan de señales digitales. Estas señales tienen la característica de recibirse lo suficientemente fuerte o no recibir nada porque no se pueden ver más que con el receptor digital. Una de las complicaciones que se dan con esta señal se debe a que la mayoría de estos equipos no tienen el mecanismo para mover la antena, por lo que se tiene que usar algún otro equipo o hacerlo manualmente. Algo que ayuda algunas veces es que por los canales donde se transmiten señales digitales también se mandan señales análogas, entonces el ajuste se hace primero basándose en la señal análoga y después sobre la digital, siempre y cuando se cuente con todo lo necesario.

Con el tiempo las personas han podido realizar estos ajustes más fácilmente y han aprendido a conocer el equipo. Con la finalidad de poder dar un mejor servicio y seguir cumpliendo con la misión de la Universidad Virtual, a partir de enero del 2000 se cambió la tecnología de transmisión, en vez de transmitir en Banda C se transmite en Banda Ku. La única desventaja que se tiene es que este tipo de ajustes se hacen más difíciles dado las características de la banda y porque el haz que se ve con la antena es más delgado siendo así una señal más selectiva.

Entonces el planteamiento es: Realizar un controlador convencional, un controlador difuso y un controlador deslizante difuso, evaluar y seleccionar cual es el mejor para sintonizar una antena satelital para una señal análoga o digital automáticamente teniendo como variable de proceso la potencia de la señal que se recibe como vemos en la figura 1.3. Este ajuste se realizara después de que la persona encargada de dicha labor haya ajustado la elevación y haya apuntado la antena más o menos al satélite para que después el controlador se encargue de realizar el ajuste restante por sí solo. El controlador esta implementado por medio de LABWINDOWS/CVI y MATLAB y junto con una computadora y una tarjeta de comunicación tiene como variable a controlar un motor, el cual mueve un pistón mecánico digital de la compañía Ultramotion que se encarga de ajustar la elevación total de la antena hasta lograr la potencia necesaria para que el equipo pueda recibir con la mejor calidad de señal, la cual es medida por medio de un receptor de datos satelital SR2000 de la compañía International Data Casting como se puede ver en la Figura 1.4. En esta Tesis se presentan todas las bases teóricas para que en trabajos o tesis posteriores se realice la implantación real de este controlador y evaluar su desempeño en tiempo real.

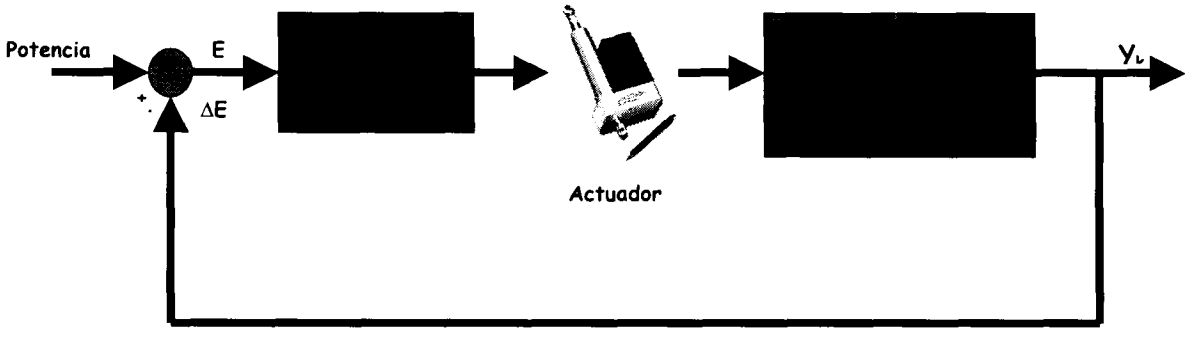


Figura 1.3: Sistema de control a bloques.

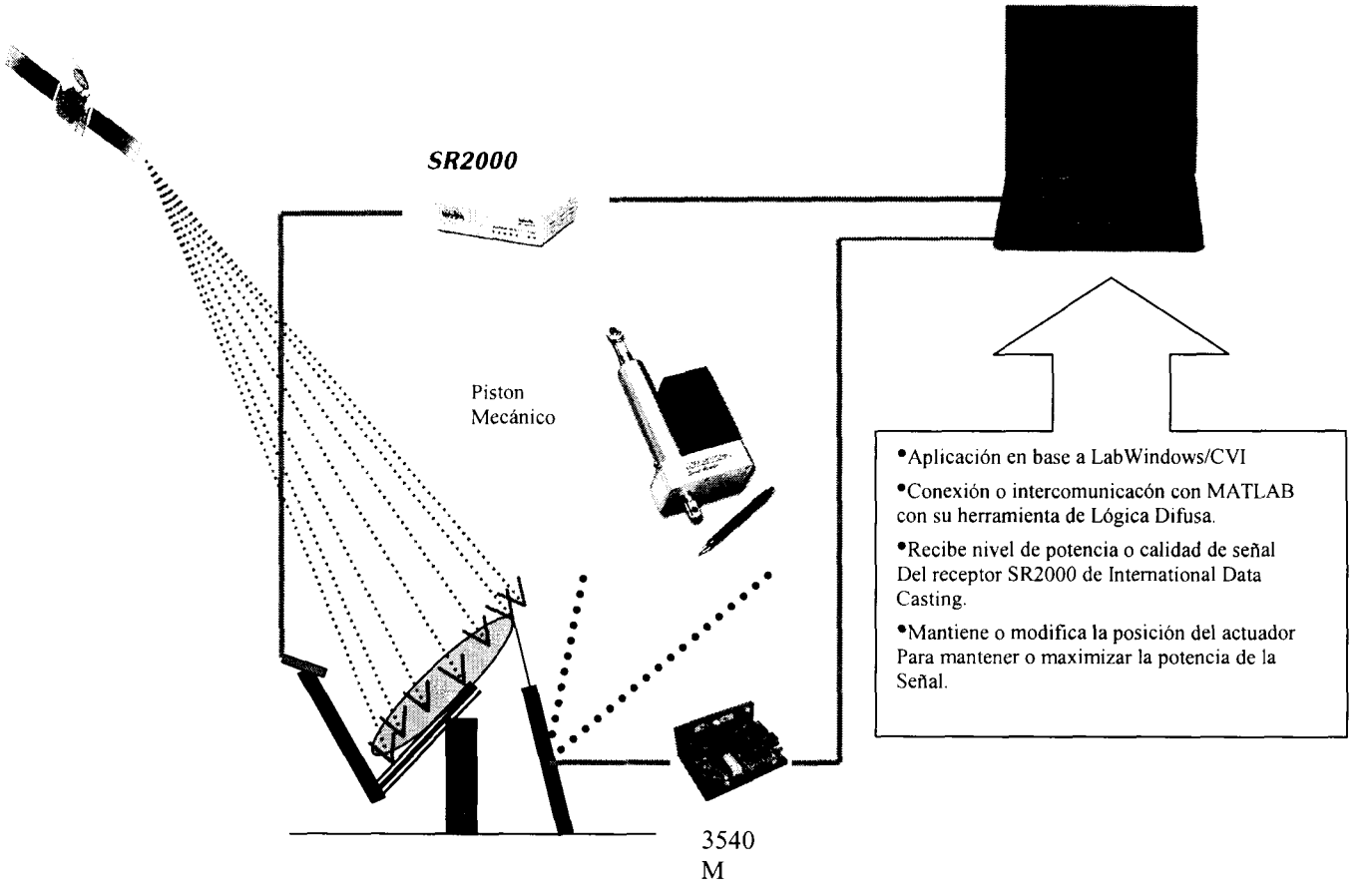


Figura 1.4: Prototipo sugerido

1.2. Objetivo

Diseñar 3 tipos de controladores: un controlador convencional PD, un controlador difuso tipo PD y un controlador deslizante difuso implementados por medio de Labwindows/CVI y Matlab e implantar el controlador con mejor desempeño.

Los objetivos particulares son los siguientes:

- Diseñar cada uno de los controladores.
- Lograr el ajuste de una antena de hasta 1.8 mts de diámetro. Controlando sólo el movimiento de inclinación.
- Realizar el ajuste teniendo una señal digital como referencia.
- Realizar el ajuste con la mayor exactitud posible.
- Realizar una interfaz que se pueda usar desde un punto remoto.

1.3. Hipótesis y contenido de la tesis

Actualmente no existen dispositivos de bajo costo que automaticen la función de posicionar una antena receptora satelital que aunado al hecho de que el posicionamiento de antenas todavía requiere de entrenamiento o conocimiento especializado, por lo que aun se vuelve necesario contar con dispositivos de este tipo.

Se propone como solución el uso de un pistón mecánico que con algoritmos de Lógica Difusa automaticen la función de posicionar la antena. Dado que se propone un controlador difuso para dicho objetivo se debe de tomar en cuenta las siguientes preguntas:

- ¿Cuál es la mejor forma de fusificar la señal de referencia?
- ¿Cuáles serán las funciones de membresía que se adecuen más al problema?
- ¿Qué tipo de arquitectura es la que más conviene?
- ¿Será necesario pensar en un esquema neuro-difuso?

Una vez contestadas dichas preguntas se podrá llegar a la hipótesis principal de esta tesis:

"El uso de un controlador difuso en el sistema de control de la antena tendrá un mejor desempeño comparado con el controlador PD convencional y el controlador deslizante difuso."

Al proponer algoritmos de Lógica Difusa para automatizar el posicionamiento de la antena la presente tesis dará los primeros pasos entregando:

En el capítulo 2, la teoría que permitirá definir las bases para el diseño de cada uno de los controladores y evaluar el desempeño de los mismos.

En el capítulo 3, el diseño de los controladores evaluados a través de Matlab por medio de su modulo de Simulink.

En el capítulo 4, la integración de Labwindows/CVI con Matlab para la implantación del prototipo en la realidad.

En el capítulo 5, las conclusiones que se obtuvieron durante el desarrollo de la tesis, así como los problemas que se presentaron durante su elaboración.

Capítulo 2

El presente capítulo presenta las bases teóricas para poder entender y diseñar los tres tipos de controladores seleccionados y establecer las bases para su evaluación.

Fundamentos Teóricos

Se le llama control convencional a aquel que esta basado principalmente en los controladores PID, controladores Proporcional – Integral – Derivativo, que por si solos pueden ayudarnos a obtener diferentes respuestas, pero que combinados han sido la base del control de procesos desde hace mucho tiempo.

Al pensar en tratar de controlar algún proceso la primera opción son los controladores Proporcional – Integral - Derivativo o cualquiera de sus variantes (P, PD, PI, PID). La selección de la acción dependerá de los parámetros de desempeño establecidos para el proceso.

La Sección 2.1 presenta las especificaciones de Respuesta Transitoria que permiten evaluar el comportamiento del controlador.

La Sección 2.2 explica cada una de las acciones básicas que tiene un controlador (acción proporcional – Integral – Derivativa) cada una por separado y después en conjunto para obtener los distintos tipos de controladores que se pueden generar con su combinación y que son evaluados a través de las especificaciones descritas en el Sección anterior.

La Sección 2.3 explica como el control por modo deslizante permite tener una estabilidad y un desempeño consistente ante aquellas imprecisiones que se pueden tener en el modelo debido a incertidumbres de la estructura o paramétricas y por dinámicas no consideradas en el modelo sobre todo cuando se tienen modelos no lineales. Se puede ver ejemplos de su comportamiento en manipuladores, vehículos submarinos, motores eléctricos, sistemas de potencia, etc. También se explican los distintos criterios para realizar el diseño y definir los parámetros que intervienen en un controlador.

La Sección 2.4 describe los sistemas difusos comúnmente usados.

La Sección 2.5 describe el control difuso que se caracteriza por ser un método basado en reglas usando sistemas expertos emulando el proceso de razonamiento de los seres humanos. La Lógica Difusa puede ser usada para controlar un proceso que puede ser controlado en forma manual con el conocimiento obtenido por la experiencia. Las reglas de control lingüísticas que el experto humano puede describir de manera intuitiva y general pueden ser trasladadas directamente a una base de reglas para un controlador difuso[4].

Por último la Sección 2.6 presenta el control deslizante difuso, el cual es un método directo para controlar sistemas sin la necesidad de un modelo matemático en forma de

ecuaciones diferenciales, en contraste al control clásico el cual es un método indirecto con un modelo matemático. El control difuso se ha implementado en muchas aplicaciones industriales. Sin embargo hay muy pocos procedimientos sistemáticos para el análisis y diseño de control difuso. El diseño de un controlador difuso con modo deslizante es propuesto para mejorar el desempeño del sistema de control. Con la adición del modo deslizante se provee un método de diseño de controladores difusos efectivo para asegurar robustez.

2.1. Especificaciones de respuesta transitoria.

En la Figura 2.1 se ve lo necesario para evaluar el comportamiento de un sistema de control, se utilizan varios parámetros de desempeño. Los más importantes son:

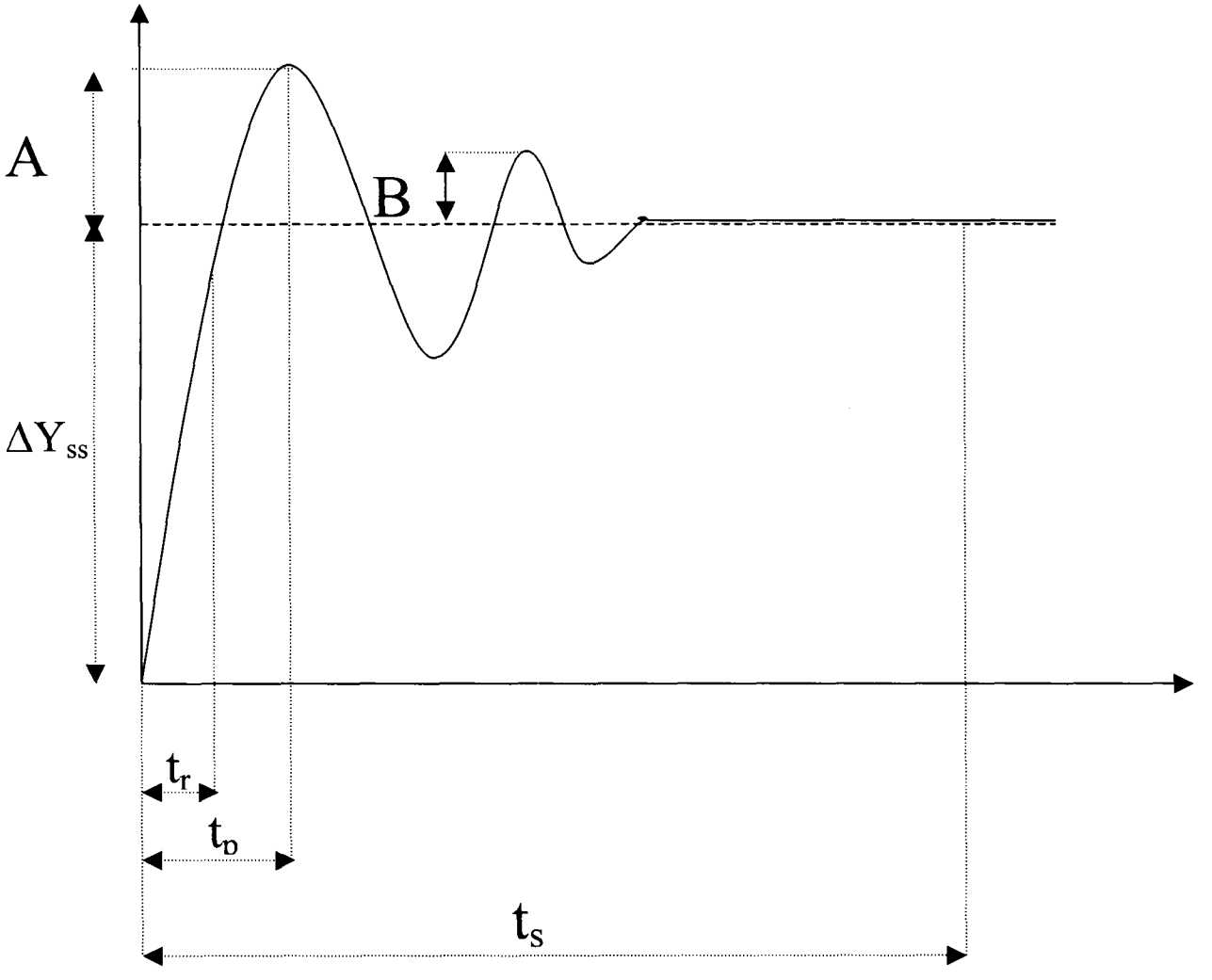


Figura 2.1: Parámetros de desempeño

Tabla 2.1. Parámetros de desempeño

t_r	Tiempo de Elevación	Tiempo para alcanzar por primera vez el valor final
t_p	Tiempo de Pico	Tiempo para alcanzar el primer pico
t_s	Tiempo de Establecimiento	Tiempo para que la magnitud de las oscilaciones sea menor o igual que el 2% de ΔY_{ss}
m_p	Sobretiro	$m_p = A/\Delta Y_{ss}$
r	Razón de decaimiento	r=B/A

2.2. Acciones básicas del controlador y PID ideal

El control PID incluye 3 acciones de control básicas:

- La **acción Proporcional** sola reduce el error pero no lo elimina cuando el proceso no tiene integrador.
- La **acción Integral** se usa para eliminar el error de estado estable. Se requiere que el número de integradores Controlador – Planta sea igual o mayor que el número de integradores de la entrada para evitar que el sistema se vuelva inestable.
- La **acción Derivativa** se usa para anticiparse a los errores. Mide la velocidad de crecimiento del error y envía una señal correctiva para evitar que crezca.

Tabla 2.2 Acciones de control

Acción	Función de Transferencia	Ecuación en el tiempo
Proporcional (P)	Kc	$m(t) = m_o + K_c e(t)$
Integral (I)	$\frac{1}{\tau_i s}$	$m(t) = m_o + \frac{1}{\tau_i}\int_0^t e(t)dt$
Derivativa (D)	$T_d s$	$m(t) = m_o + \tau_d \frac{de(t)}{dt}$
PID Ideal	$K(1 + \frac{1}{\tau_i s} + \tau_d s)$	$m(t) = m_o + K_c[e(t) + \frac{1}{\tau_i}\int_0^t e(t)dt + \tau_d \frac{de(t)}{dt}$

2.3. Control deslizante[7]

La estructura típica de un controlador robusto es compuesta por una parte nominal, similar a una linealización o una ley de control inversa y la adición de términos para lidiar con las incertidumbres del modelo. Para esta clase de sistemas, el diseño de control deslizante provee una aproximación sistemática al problema de mantener estabilidad y un desempeño consistente ante la modelación de imprecisiones en el modelo.

2.3.1 Superficie deslizante

Consideramos un sistema dinámico de una sola entrada

$$x^{(n)} = f(x) + b(x)u \tag{2.1}$$

La función f(x) es una función que no es conocida pero la imprecisión en f(x) está delimitada por una frontera superior por medio de una función continua de **x**; en forma similar, la ganancia de control b(x) tampoco es conocida con exactitud.

El problema de control es obtener un estado **x** para seguir un estado variante en el tiempo específico $\mathbf{x_d} = [x_d\ x_d\ \ldots\ x_d^{(n-1)}]^T$ en la presencia de un modelo impreciso en f(x) y b(x).

Para que la meta de seguimiento sea alcanzada usando un control finito *u*, el estado inicial $\mathbf{x_d(0)}$ debe ser tal que

$$x_d(0) = x(0) \tag{2.2}$$

2.3.2 Notación simplificada

Sea $\tilde{x} = x - x_d$ el error de seguimiento en la variable x y sea

$$\tilde{x} = x - x_d = [\tilde{x}\ \dot{\tilde{x}}\ \ldots.\tilde{x}^{(n-1)}]^T$$

el vector de error de seguimiento. Y si ahora definimos una superficie variante en el tiempo S(t) en el espacio $\mathbf{R^{(n)}}$ por una ecuación escalar s(x;t)=0, donde

$$s(x;t) = \left(\frac{d}{dt} + \lambda\right)^{n-1} \tilde{x} \tag{2.3}$$

y λ es una constante estrictamente positiva. Entonces para el caso donde n=2,

$$s = \dot{\tilde{x}} + \lambda\tilde{x}$$

Dada la condición inicial de (2.2), el problema de seguimiento $\mathbf{x} \equiv \mathbf{x_d}$ es equivalente a la parte restante de la superficie S(t) para todo t>0; $s \equiv 0$ representa una ecuación lineal diferencial cuya única solución es $\tilde{x} \equiv 0$, dadas las condiciones iniciales de (2.2). Entonces el problema de seguir un vector x_d dimensional de orden n se reduce a mantenerlo en una cantidad escalar de cero a S.

X_d puede ser remplazado por un problema de estabilización de primer orden en *s*. El problema simplificado de primer orden de mantener un escalar en cero ahora puede ser alcanzado escogiendo una ley de control *u* de (2.1) dado que fuera de S(t).

$$\frac{1}{2}\frac{d}{dt}s^2 \leq -\eta \mid s \mid \tag{2.4}$$

En otras palabras satisfaciendo la condición de la ecuación anterior o condición deslizante, hace a la superficie un conjunto invariante[1]. Además de que también implica que algunas perturbaciones o incertidumbres dinámicas en el modelo pueden ser toleradas mientras se mantenga a la superficie como un conjunto invariante.

Gráficamente se puede ver en la Figura 2.2, esto corresponde a que las trayectorias entrando a la superficie se pueden mover mientras estén apuntando hacia la superficie. S(t) en la ecuación (2.4) se refiere a una superficie deslizante y al comportamiento del sistema una vez en la superficie es llamado régimen deslizante o modo deslizante.

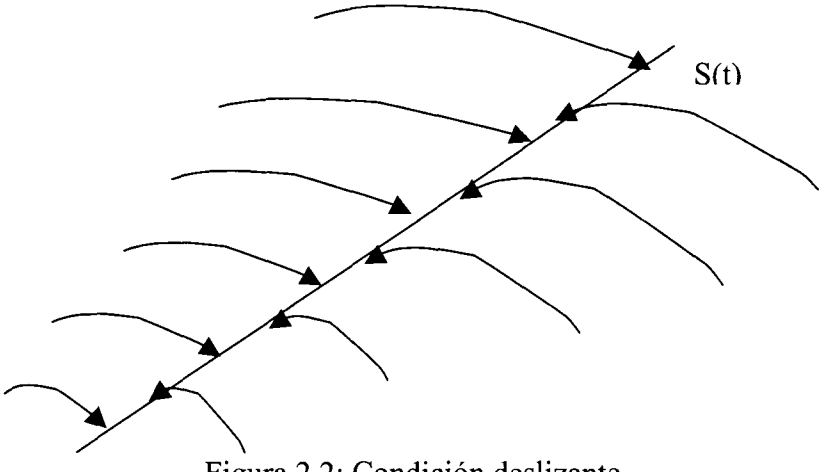


Figura 2.2: Condición deslizante

Otro aspecto interesante del conjunto invariante S(t) es que una vez en él, las trayectorias del sistema están definidas por (2.3) igualada a cero. También esta definición

[1] Un conjunto **G** es un conjunto invariante para un sistema dinámico si cada trayectoria del sistema que inicia en un punto **G** permanece en **G** para todo el tiempo futuro.

implica que una vez en la superficie, el error de seguimiento tiende exponencialmente a cero con una constante de tiempo (n-1)/λ.

Una vez satisfecha la condición de (2.4), si la condición de (2.2) no esta verificada exactamente (si x(t=0) esta fuera de x_d(t=0)) la superficie S(t) va a ser alcanzada aun así en un tiempo finito menor que $|s(t=0)|/\eta$.

Por ejemplo si n=2, la superficie es una línea en el plano de fase con pendiente de -λ y con un punto de $x_d = [x_d \dot{x}_d]^T$. Empezando desde cualquier condición inicial, el estado de trayectorias alcanza a la superficie variante en el tiempo en un tiempo finito menor que $|s(t=0)|/\eta$ y luego se desliza sobre la superficie hacia x_d en forma exponencial y con una constante de tiempo igual a 1/λ. como se puede ver en la Figura 2.3.

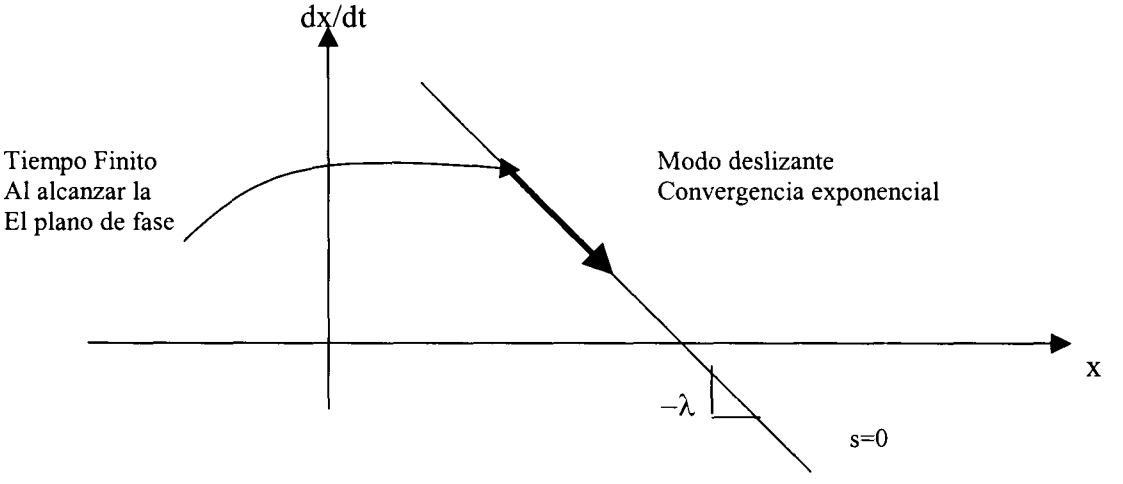


Figura 2.3: Interpretación gráfica ecuaciones (2.3) y (2.4).

Los pasos para poder construir el controlador son: primero se necesita una ley *u* de control con retroalimentación así como verificar la condición deslizante. Sin embargo para tener en cuenta las imprecisiones del modelo y las perturbaciones, la ley de control tiene que ser discontinua a través de S(t). Dado que la implementación de un controlador conmutado es imperfecta por necesidad, esto nos lleva al castañeo mostrado en la Figura 2.4. Él castañeo es una situación indeseable en la práctica dado que involucra una actividad de control alta y en un futuro puede excitar altas frecuencias dinámicas no modeladas. La discontinuidad de la ley de control *u* es lo suficientemente suave como para alcanzar un cambio entre el ancho de banda y la precisión del seguimiento: mientras el primer paso tiene que ver con incertidumbres paramétricas, el segundo tiene que ver con la robustez a dinámicas de alta frecuencia sin modelar.

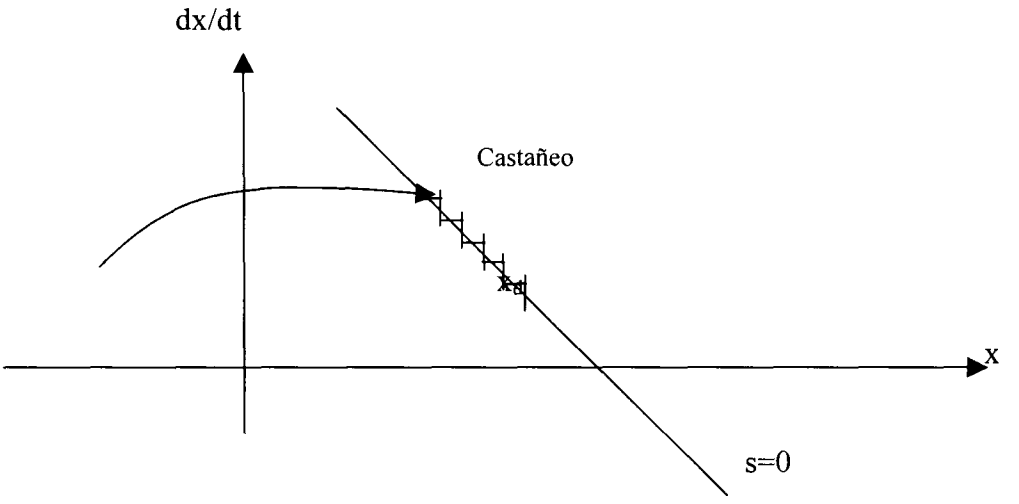


Figura 2.4: Castañeo como resultado de imperfecciones en el control.

2.3.3. Selección de λ

Generalmente la sintonía de un número escalar se hace experimentalmente en la practica. Haciendo un análisis en el diseño total podemos obtener explícitamente los factores limitantes de λ.

- Modo estructural resonante: λ debe ser menor a la frecuencia V_R del modo estructural resonante sin modelar.

$$\lambda \leq \lambda_R \approx \frac{2\Pi}{3} v_R \tag{2.5}$$

- Retraso en el tiempo no considerado: tenemos la condición

$$\lambda \leq \lambda_A \approx \frac{1}{3T_A} \tag{2.6}$$

Cuando T_A es mayor que el tiempo de retardo sin modelar.

- Tiempo de muestro: con un retraso de un periodo completo de proceso, se tiene la siguiente condición:

$$\lambda \leq \lambda_s \approx \frac{1}{5} v_{sampling} \tag{2.7}$$

Donde $v_{sampling}$ es el tiempo de muestreo.

El control de banda λ deseado es el mínimo entre estos tres. El primero depende de las propiedades mecánicas del sistema, el segundo sobre las limitaciones en el actuador y el tercero del poder computacional con el que se cuenta.

2.4 Sistemas difusos

Los sistemas difusos son bases de conocimiento o sistemas basados en reglas. El corazón de un sistema difuso es la base de conocimiento formada por unas reglas llamadas "reglas de si... entonces". Una regla "Sí - entonces" es una estructura donde las palabras son caracterizadas por funciones de membresía continuas. Por ejemplo:

"Sí (Error es Positivo Pequeño) y (Derivada es Cero) entonces (salida es Positivo Grande)"

En la Figura 2.5 podemos ver la función de membresía para Positivo Grande por ejemplo.

Para poder tener diferentes reglas en un sistema simple se utilizan diferentes principios de combinación. Los tres tipos más usados comúnmente en la literatura son[9]: Sistemas Difusos Puros, Sistemas Difusos Takagi – Sugeno, y Sistemas Difusos con Fusificador y Defusificador.

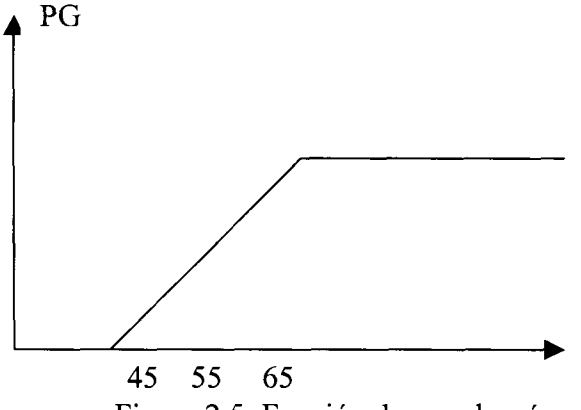


Figura 2.5: Función de membresía

2.4.1 Sistemas difusos puros.

Como se puede ver en la Figura 2.6 son los sistemas formados sólo por una base de reglas difusas y una máquina de inferencia difusa. La base de reglas representa la colección de reglas "Sí – Entonces". La máquina de inferencia combina estas reglas difusas dentro de un mapeo de conjuntos difusos en un espacio de entradas U a conjuntos de un espacio de salidas R basados en principios de Lógica Difusa. Su principal problema se debe a que sus entradas y salidas son completamente difusas, las palabras están en lenguaje natural,

cuando en los sistemas ingenieriles las entradas y salidas son variables de valor real (numéricas).

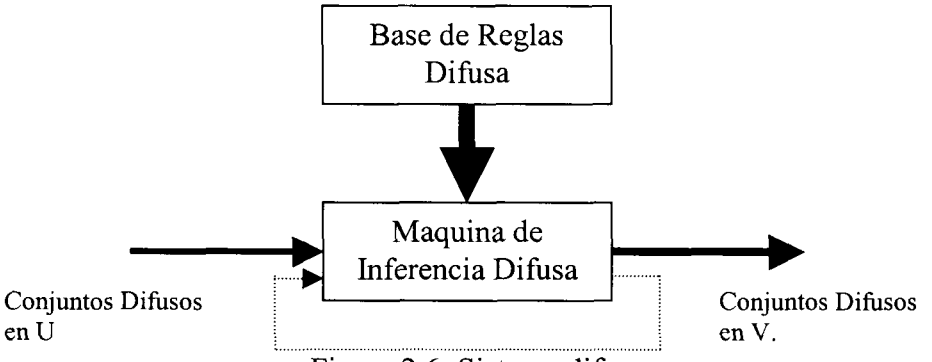


Figura 2.6: Sistema difuso puro

2.4.2 Sistemas difusos Takagi-Sugeno.

En la Figura 2.7 a diferencia del sistema anterior la parte de la regla después del "Entonces" cambia de usar una descripción en lenguaje natural a una simple formula matemática. Con estos cambios se pueden combinar mejor las reglas. Pero uno de sus problemas es que la parte matemática del "Entonces" es una formula matemática que algunas veces no puede representar el conocimiento humano por medio de reglas. Además no se tiene mucha libertad para aplicar diferentes principios en la Lógica Difusa, así que la versatilidad de los sistemas difusos no puede ser bien representada.

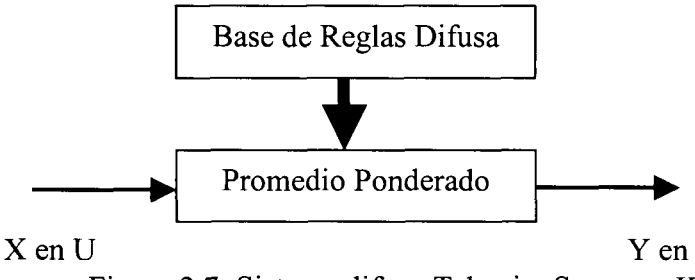


Figura 2.7: Sistema difuso Takagi – Sugeno - Kang

2.4.3 Sistemas difusos con fusificador y defusificador.

En la Figura 2.8 el fusificador transforma las variables expresadas en valores reales a un set difuso, esto es a la entrada. El defusificador transforma ese conjunto difuso a una variable de valor real, esto es en la salida. Este sistema elimina las desventajas de un sistema puro difuso y el de los Sistemas Difusos Takagi – Sugeno – Kang.

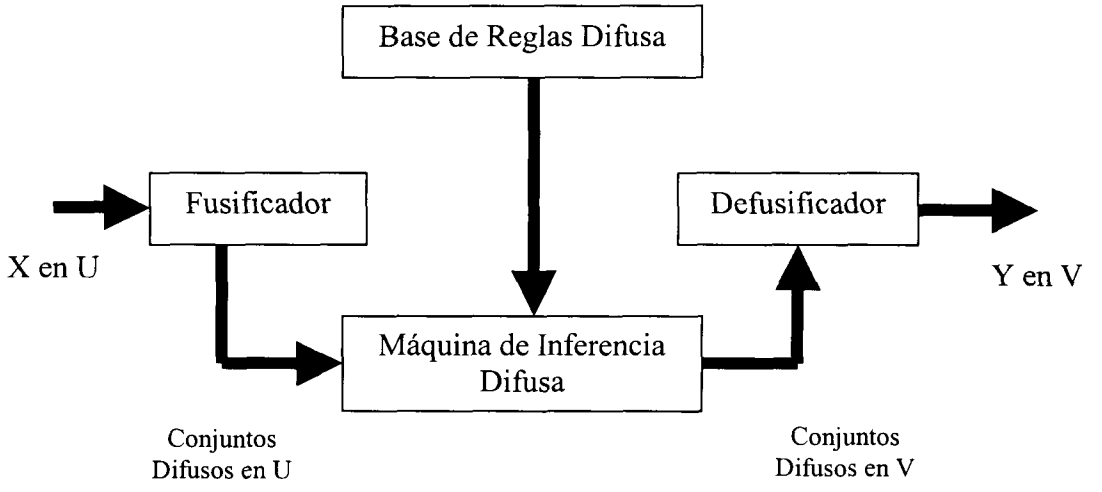


Figura 2.8: Sistema difuso con fusificador y defusificador

2.4.4 Controladores con base de conocimiento difuso

La estructura de un controlador difuso con base de conocimiento difuso como el que se muestra en la Figura 2.9 está formado por varios componentes[4].

Modulo de fusificación.

Realiza las siguientes funciones

- FM-F1: Realiza un escalamiento escalando los valores físicos de las variables de estado del proceso dentro de un dominio normalizado. También se escala los valores normalizados de los valores de salida del controlador en un dominio desnormalizado.
- FM-F2: Realiza la fusificación la cual convierte un valor nítido en un conjunto difuso para hacerlo compatible con la representación de las variables de estado del proceso en la regla-antecedente.

Base de conocimiento.

Consiste en la base de datos y reglas. La base de datos provee la información necesaria para el funcionamiento propio del módulo de fusificación, la base de las reglas y el módulo de defusificación.

Máquina de inferencia

Existen dos tipos de aproximaciones empleadas en los diseños de las máquinas de inferencia: (1) Inferencia basada en composición y (2) inferencia basada en reglas individuales. La función básica del segundo tipo de máquina de inferencia es calcular el valor total de la variable de control de salida basado en las contribuciones individuales de cada regla en la base de reglas. Cada contribución individual representa un valor de la variable de control de salida calculado por una regla. La salida del fusificador es

comparada contra cada regla antecedente, y se establece un grado de comparación con cada una. Basándose en este grado, el valor de la variable de control de salida en la regla-antecedente es modificado.

Módulo defusificador

Realiza las siguientes funciones:

- DM-F1: Realiza la defusificación la cual convierte los conjuntos modificados de los valores de salida de control en un solo valor.
- DM-F2: Obtiene la salida desnormalizada la cual escala el valor de la salida de control solamente en un dominio físico.

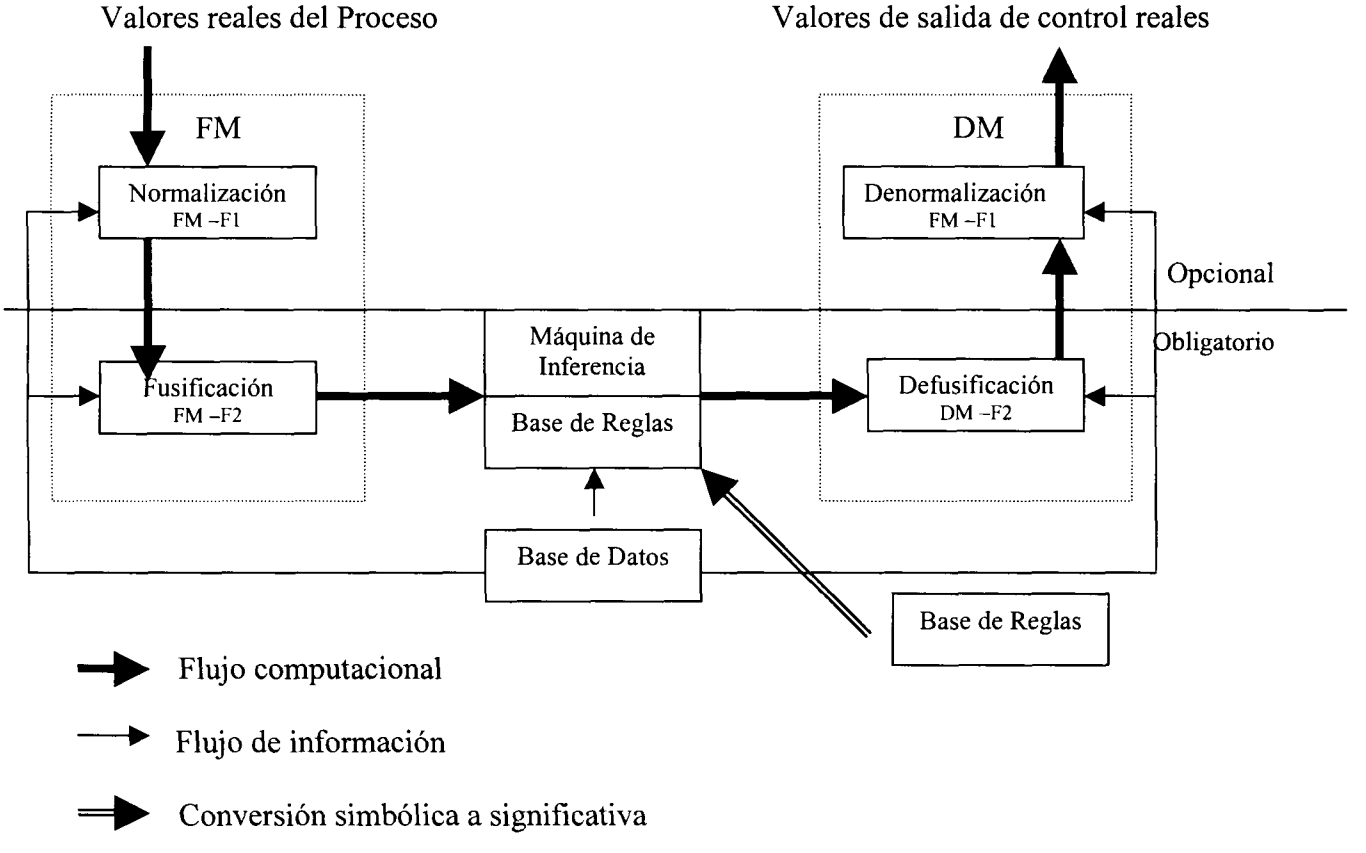


Figura 2.9: Controlador con base de conocimiento difuso

2.5 Control difuso

La Lógica Difusa fue diseñada primeramente para representar y razonar con cierta forma de conocimiento. Se asumió que el conocimiento sería expresado en forma lingüística o verbal y que no todas las operaciones deberían de ser intelectuales sino lo suficientemente poderosas para poder usar las computadoras.

2.5.1 Conjuntos difusos

Para cualquier conjunto nítido C es posible definir una función característica μ_c:→ {0,1}. La función característica es generalizada como una función de Membresía que asigna a cada $\mu \in U$ un valor del intervalo unitario {0,1}en vez de asignar un conjunto de dos valores {0,1}. El conjunto que esta definido basándose en dicha extensión de la función de membresía es llamado Conjunto Difuso.

La función de membresía μ_F de un conjunto difuso F es una función

$$\mu_F : U \rightarrow \{0,1\} \tag{2.8}$$

Así que cada elemento μ de U tiene un grado de membresía $\mu_F(\mu) \in \{0,1\}$. F esta completamente determinado por un conjunto de "tuples".

$$F = \{(\mu, \mu_F(\mu)) \mid \mu \in U\} \tag{2.9}$$

Por ejemplo supongamos que alguien quiere describir diferentes clases de autos, los cuales tienen la propiedad de ser muy caros y considerando como autos tales como BMW, Buick, Ferrari, Fiat, Lada, Mercedes, y Rolls Royce. Algunos autos como Ferrari y Rolls Royce definitivamente pertenecen a este grupo, así como los autos Fiat, Lada no pertenecen a este. Pero hay un tercer grupo de autos, donde es difícil definir si son caros o no. Usando los conjuntos difusos, el conjunto difuso de autos caros es:

{(Ferrari,1), (Rolls Royce, 1),(Mercedes,0.8),(BMW, 0.7),(Buick, 0.4)}

Zadeh propone una notación más conveniente para los conjuntos difusos. Supongamos que C es un conjunto nítido $\{\mu_1, \mu_2, \ldots \mu_n\}$ donde la notación alternativa es:

$$C = \mu_1 + \mu_2 + \ldots + \mu_n \tag{2.10}$$

Donde + denota una enumeración. Sin embargo otra notación podría ser $(\mu,\mu(u))$ o $\mu(u)/u$ donde / denota un tuple.

Por ejemplo el caso de los autos caros se podría representar de la siguiente manera:

1/Ferrari + 1/Rolls Royce + 0.8/Mercedes + 0.7/BMW + 0.4/Buick

En el control difuso, hay usualmente 4 tipos de conjuntos que están relacionados con las etiquetas "creciente", "decreciente" y dos tipos de notaciones "aproximadas". Este tipo de conjuntos son denominados conjuntos convexos. Los conjuntos difusos No-Convexos son conjuntos difusos que alternadamente crecen y decrecen en el dominio como se puede ver en la Figura 2.10

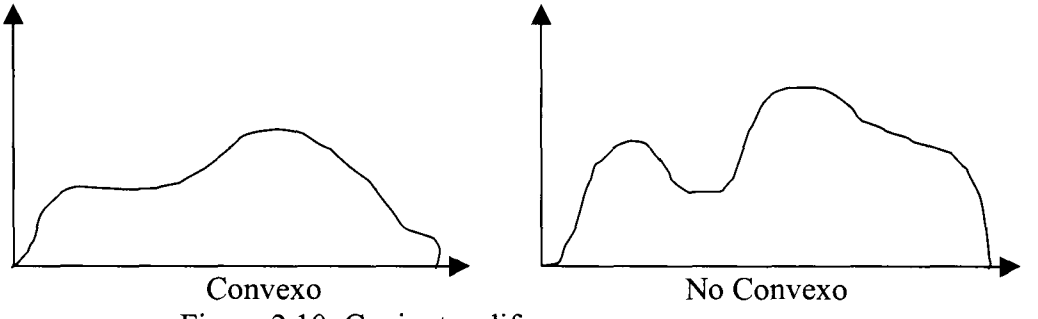


Figura 2.10: Conjuntos difusos convexos y no convexos.

Los conjuntos difusos que van creciendo se utilizan para poder describir nociones lingüísticas como "alto, gordo, caliente, viejo" en distintos dominios respectivamente. Los conjuntos decrecientes representativos para estos dominios son "bajo, flaco, frío, joven" como se ve en la Figura 2.11.

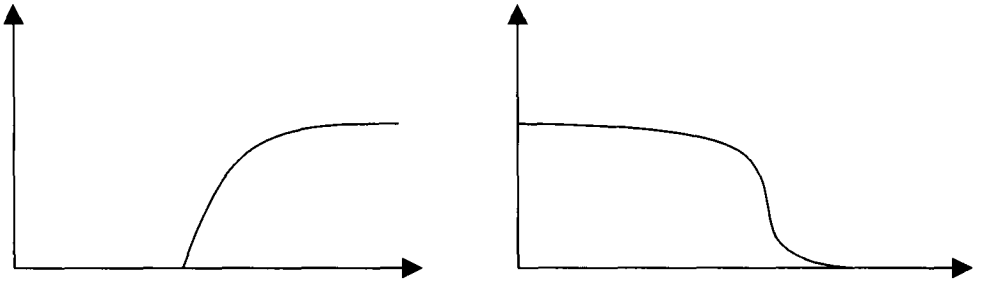


Figura 2.11: Conjuntos difusos crecientes y decrecientes

Generalmente en el control difuso se utilizan dominios de 7 elementos como pueden ser: Positivo Grande (PG), Positivo Medio(PM), Positivo Pequeño(PP), Cero(CE), Negativo Pequeño(NP), Negativo Medio(NM), y Negativo Grande(NG) lo cual se muestra en la Figura 2.12. El dominio usado es el dominio de los reales entre –6 y 6.

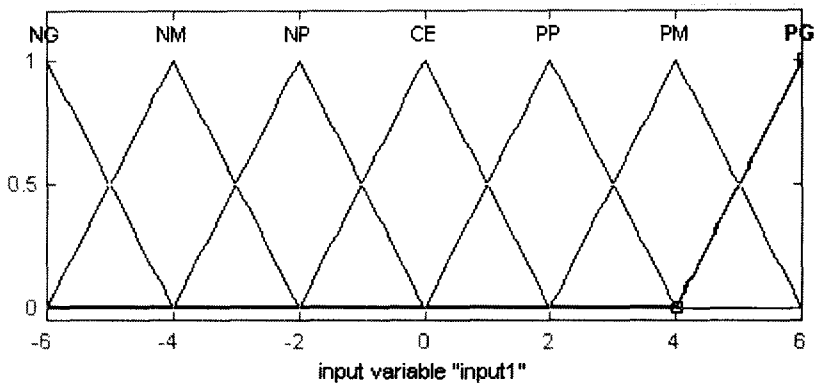


Figura 2.12: Dominio Estándar.

Si el dominio usado es en el intervalo real [-1, 1], se dice que estamos trabajando en un dominio Normalizado.

2.5.2. Enunciados "Si – Entonces" difusos

Una condicional difusa o regla "Si – Entonces" es expresada simbólicamente como:

Sí (Proposición difusa) *Entonces* (Proposición difusa)

Donde una proposición difusa es una proposición atómica o compuesta. Las reglas "Si – Entonces" describen la relación causal entre el estado del proceso y las variables de salida de control. Por ejemplo si **e** y **ė** son variables de estado de un proceso y **u** es la variable de control entonces:

Sí **e** es NG y **ė** es PG Entonces **u** es NP.

Es una expresión simbólica de la siguiente relación causal en un formato de lenguaje natural:

"Sí es el caso de que el valor de **e** sea negativo grande y el valor actual de **ė** sea positivo grande, entonces esto es la causa para que se realice un pequeño decremento en el valor previo del valor de control de salida."

El significado de una expresión simbólica:

Sí X es A, Entonces Y es B

Es representada como una relación difusa en X x Y donde X y Y son dominios de las variables lingüísticas X y Y. La construcción de estas relaciones difusas se construyen como sigue:

1. El significado de "X es A" llamada regla antecedente, es representado por un conjunto difuso $\widetilde{A} = \int_X \mu_A(x)/x$,

2. El significado de "Y es B", llamado regla consecuente, es representado por un conjunto difuso $\widetilde{B} = \int_y \mu_A(y)/y$,
3. El significado de un condicional difuso es entonces una relación difusa μ_R tal que

$$\forall x \in X \forall y \in Y : \mu_R(x,y) = \mu_A(x) * \mu_B(y) \qquad (2.11)$$

Donde "*" puede ser cualquier producto Cartesiano o cualquier operación de implicación difuso.

2.5.3 Elección de un conjunto de términos

Los valores lingüísticos miembros de un conjunto de términos son expresados como parejas en la forma <signo, magnitud>, por ejemplo <Positivo grande>, <negativo pequeño>. La componente del signo puede tomar dos valores, positivo o negativo. El componente de la magnitud puede tomar cualquier valor lingüístico que exprese magnitud (cero, pequeño, mediano, grande, etc.). El significado de una pareja, por ejemplo para el caso de un controlador PI como una base de conocimiento difuso se puede explicar de la siguiente manera:

- Los valores lingüísticos de **e** con signo negativo significan que la salida actual del proceso y tiene un valor por arriba del punto de operación y_{sp} dado que $e(k)= y_{sp}-y(k)<0$. La magnitud de un valor negativo describe la magnitud de la diferencia $y_{sp}-y$. Por otra parte los valores lingüísticos de **e** con signo positivo significan que el valor actual de y esta por debajo del punto de operación. La magnitud de tal valor positivo es la magnitud de la diferencia de $y_{sp}-y$.
- Los valores lingüísticos de **Δe** con signo negativo significan que la salida actual del proceso $y(k)$ se ha incrementado a comparación del valor previo $y(k-1)$ dado que $\Delta e(k) = -(y(k)-y(k-1))<0$. La magnitud de dicho valor negativo esta dado por la magnitud de este incremento. Los valores lingüísticos de $\Delta e(k)$ con signo positivo significan que $y(k)$ ha disminuido su valor en comparación con $y(k-1)$. La magnitud de dicho valor es la magnitud que se disminuyo.
- Un valor lingüístico de "cero" para **e** significa que la salida actual del proceso esta en el punto de operación. Un "cero" para Δe significa que la salida del proceso actual no ha cambiado desde su valor anterior, $-(y(k)-y(k-1))=0$.
- Valores lingüísticos de **Δu(k)** con signo positivo significan que el valor de la salida de control $u(k-1)$ tiene que ser aumentado para alcanzar el valor de la salida de control en el tiempo de muestro k. Un valor con signo negativo significa una disminución en el de $u(k-1)$ .La razón para esto es $u(k)= u(k-1)+ \Delta u(k)$. La magnitud del valor es la magnitud del aumento/disminución del valor $u(k-1)$.

La base de reglas se puede representar en forma de tabla como se ve en la Figura 2.13. La celda definida por la intersección del primer renglón con la primera columna representa una regla como:

Si e(k) es NG y Δe(k) es NG entonces Δu(k) es NG

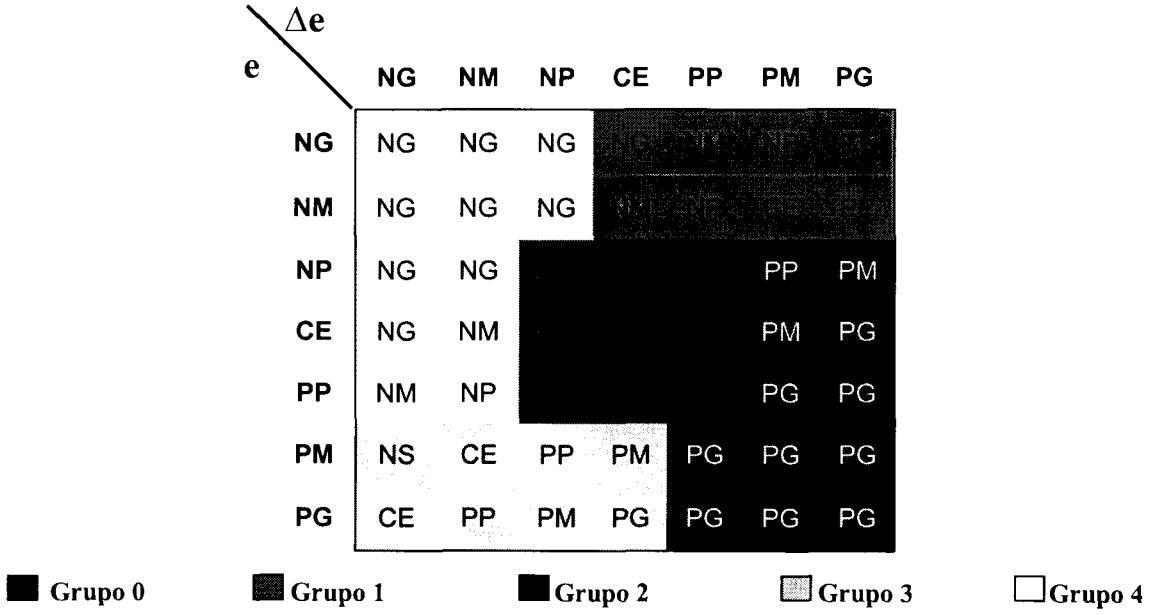


Figura 2.13: Base de reglas en forma tabular.

El conjunto de reglas pueden dividirse en 5 grupos:

- **Grupo 0:** En este grupo de reglas tanto **e** como **Δe** son (positivo o negativo) pequeños o casi cero. Esto significa que el valor de la variable de salida *y* se ha salido un poco del punto de operación pero todavía se encuentra cerca de él. La variación en el cambio Δu(k), en la salida de control previo u(k-1) generado por estas reglas también es pequeño o cero y sólo intentan corregir pequeñas desviaciones en el punto de operación. Las reglas de este grupo están relacionadas con el comportamiento del proceso en estado estable.
- **Grupo 1:** Para este grupo de reglas **e** es negativo con un valor grande o mediano el cual implica que *y(k)* esta por arriba significativamente del punto de operación. Al mismo tiempo, dado que Δe(k) es positivo, esto significa que *y* sé esta moviendo hacia el punto de operación. Que tanto cambia Δu por causa de las reglas en este grupo en la salida de control u(k-1) tiene la intención de acelerar o frenar la velocidad con que se aproxima al punto de operación. Esto se realiza cambiando la magnitud de Δu(k).
- **Grupo 2:** Para este grupo de reglas **e** esta cerca del punto de operación (PP, CE, NP) o esta abajo en forma significativa (PM,PG). Al mismo tiempo, dado que

Δe es positivo, *y* sé esta alejando del punto de operación. Entonces un cambio positivo en Δu(k) en el valor previo de la salida de control u(k-1) trata de invertir el sentido y hacer a *y* que en vez de moverse hacia fuera del punto de operación, se mueva hacia él.

- **Grupo 3:** Para este grupo de reglas **e** es positivo medio o grande lo cual significa que *y(k)* es por debajo del punto de operación significativamente. Al mismo tiempo, dado que **Δe** es negativo, *y* sé esta moviendo hacia el punto de operación. Que tanto debe cambiar Δu por causa de las reglas en la salida de control previa u(k-1) es para acelerar o frenar a *y* hacia el punto de operación.
- **Grupo 4:** Para este grupo de reglas **e** esta cerca del punto de operación (PP, CE, NP) o esta por arriba de él significativamente (NM, NG). Al mismo tiempo dado que **Δe** es negativo, *y* sé esta alejando del punto de operación. Entonces un cambio negativo en Δu(k) en la salida de control previa u(k-1) es aplicado para revertir el sentido y hacer que *y* en vez de alejarse del punto de operación se acerque hacia él.

Si uno desea mejor resolución de control alrededor del punto de operación entonces uno puede considerar un rango largo de valores lingüísticos para NP, CE, y PP, por ejemplo: CEP, cero positivo; CEN, cero negativo; PMP, positivo muy pequeño; y NMP, negativo muy pequeño. Aunque se debe tomar en cuenta que el hacer esto significa un aumento en el número de reglas.

2.5.4 Derivación de las reglas

Se tienen tres aproximaciones para la generación de las reglas de una base de conocimiento difuso. Cada uno se complementa con respecto al otro y su combinación se vuelve necesaria para poder generar una combinación efectiva de reglas.

- **Aproximación 1:** Es la más utilizada en nuestros días. Esta basada en la derivación de las reglas a partir de la experiencia del operador del proceso o del ingeniero de control. Se realiza usando dos técnicas:
  1. Introspectiva verbal de la experiencia.
  2. Interrogatorio al operador del proceso o ingeniero de control usando un cuestionario organizado cuidadosamente.
- **Aproximación 2:** Utiliza una descripción lingüística viéndolo como un modelo difuso del proceso bajo control para derivar el conjunto de reglas.
- **Aproximación 3:** Se basa en la existencia de modelo convencional del proceso, usualmente no lineal.

2.5.5 Base de datos

Los parámetros que tienen que ver con la construcción de la base de datos son:

- Selección de las funciones de membresía,
- Selección de los factores de escalamiento.

Para una eficiencia computacional, un uso eficiente de la memoria y un buen desempeño de análisis se requiere de una representación uniforme para las funciones de membresía. Esta representación uniforme puede ser alcanzada empleando funciones de membresía con curvas uniformes y paramétricas de definición funcional.

Las selecciones más funcionales de curvas para las funciones de membresía son las funciones triangulares, las trapezoidales y las de campana. Las funciones triangulares son las más usadas debido a que son las que menos esfuerzo computacional y memoria necesitan.

Una vez que selecciona la función de membresía es necesario escalar los elementos del conjunto de términos al dominio correspondiente de la variable lingüística. Este mapeo puede afectar en el desempeño de la base de conocimiento.

El uso de dominios normalizados requiere de transformaciones escalares que escalan los valores físicos de las variables de estado del proceso a un dominio normalizado. Esto se le conoce como Normalización de las entradas. La denormalización de las salidas escala los valores normalizados de las variables de control de salida a su respectivo dominio físico. Estas transformaciones son necesarias tanto en el dominio discreto como continuo.

Los factores de escalamiento que describen las entradas normalizadas y las salidas desnormalizadas juegan un rol similar al de los coeficientes de ganancia en el control convencional.

Hay dos aproximaciones para determinar los factores de escalamiento: heurística y formal(analítica).

2.6 Control deslizante difuso

Cuando se tiene un sistema no lineal se crean sistemas difusos diseñados con respecto al plano de fase determinado por el error **e** y el cambio en el error **ė** con respecto a los estados x y $\dot{x}$. Un valor difuso para una variable de control es determinado de acuerdo a valores difusos del error y el cambio en el error. La aproximación al sistema de control es la división del plano de fase en dos semiplanos por medio de una *Línea de conmutación.*

La magnitud de la salida de control depende de la distancia del vector de estados de la línea de conmutación. Para una clase específica de sistemas no lineales hay un apropiado método de control robusto llamado control por modo deslizante.

El control por modo deslizante es apropiado especialmente para el seguimiento de manipuladores de robots y para motores cuyas cargas mecánicas cambian en un rango amplio. La desventaja de este método son los cambios drásticos en la variable de control que provoca un gran estrés, como es él castañeo. Para suavizar las señales de control se introduce un plano cerca de la línea de switcheo la cual se puede ver en la Figura 2.14.

En el modo deslizante difuso en general las reglas son condicionadas de tal manera que cerca de la línea de switcheo S = 0 una salida negativa de control es generada y una señal positiva por la parte de abajo, similar al control por modo deslizante.

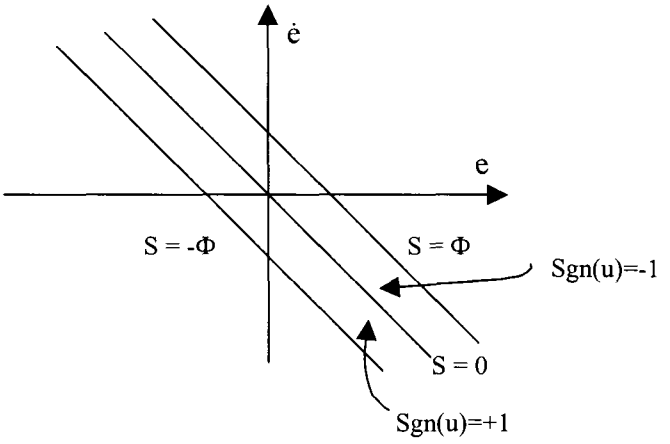


Figura 2.14: Principio de modo deslizante con frontera.

2.6.1. Aspectos cualitativos del diseño de reglas.

Para diseñar la base de conocimiento de reglas difusas se siguen los siguientes pasos:

1. u_N debe ser negativo arriba de la línea de conmutación y positivo por debajo.
2. $|u_N|$ debe de crecer conforme la distancia entre el estado actual y la línea de switcheo S=0 crece.
3. $|u_N|$ debe de crecer como la distancia *d* entre el estado actual y la línea perpendicular a la línea de switcheo por lo siguiente:
   - Discontinuidades en las fronteras del plano de fase se pueden evitar.
   - El dominio central del plano de fase puede ser alcanzado bastante rápido.
4. Los estados normalizados de $\mathbf{e_N}$ y $\dot{\mathbf{e}}_N$ que caen en el plano de fase deben de ser cubiertos por valores máximos $|\mu_N|_{max}$ con respecto al signo de μ_N.

2.7 Conclusiones

En el presente capítulo definimos los parámetros para evaluar el desempeño de un controlador, los cuales se definen por medio de la Tabla 2.1 y son: el tiempo de elevación, el tiempo de pico, tiempo de establecimiento, sobretiro, razón de decaimiento.

También se define una técnica como el control por modo deslizante que nos puede ayudar a mejorar al controlador difuso en un futuro.

También se define la teoría que permitirá como es que trabaja la Lógica Difusa y los pasos necesarios para poder diseñar el controlador tomando como base a la Lógica Difusa.

Se presenta la teoría necesaria para saber combinar a la Lógica Difusa con el control por modo deslizante y obtener un controlador más robusto en su desempeño.

Quizá la Lógica Difusa no sea suficiente y sea recomendable el optar por el controlador deslizante difuso.

Capítulo 3

Diseño y simulaciones de los controladores en MATLAB

Matlab es un paquete que por el hecho de tener una gran cantidad de funciones numéricas ya implementadas facilita su uso. Es una herramienta muy utilizada en el ITESM para el desarrollo y verificación de diseños de controladores aprovechando las ventajas de su modulo de Simulink. Es por ello que se selecciona para la realización de los diseños, su simulación y evaluación.

Este capítulo se presenta los resultados de las simulaciones tal cual como se diseñaron por medio de Matlab y los resultados obtenidos que permiten evaluar los controladores diseñados.

3.1 Modelo de antena lineal de segundo orden

El primer diseño se realiza basándose en un modelo de un problema donde el objetivo es controlar la elevación de una antena[5]. Es un modelo de segundo orden lineal, como se muestra a continuación en forma de ecuaciones diferenciales, función de transferencia y variables de estado por (3.1), (3.2), (3.3).

$$J\ddot{\theta} + B\dot{\theta} = T_c + T_d \tag{3.1}$$

J= Momento de inercia de la antena y motor
B= Amortiguamiento de la antena y motor.
T_c= Torque neto del motor
T_d= Torque debido al viento

$$\frac{B}{J} = a, \quad u = \frac{T_c}{B}, \quad w_d = \frac{T_d}{B}$$

$$\frac{B}{a}\ddot{\theta} + B\dot{\theta} = Bu + Bw_d$$

$$\frac{1}{a}\ddot{\theta} + \dot{\theta} = u + w_d$$

$$\frac{1}{a}S^2\theta(s) + S\theta(s) = u(s) + w_d(s)$$

Y si se elimina la perturbación $w_d(s)$, queda:

$$\left[\frac{1}{a}S^2 + S\right]\theta(s) = u(s)$$

$$\frac{\theta(s)}{u(s)} = \frac{1}{S[(S/a)+1]} \tag{3.2}$$

Usando la función "SS" de matlab para crear modelos en variables de estado y dándole un valor a a= 0.1 tenemos:

$$\dot{X} = \begin{bmatrix} -0.1 & 0 \\ 0.5 & 0 \end{bmatrix} X + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} u$$

$$Y = \begin{bmatrix} 0 & 0.4 \end{bmatrix} x \tag{3.3}$$

Para comprobar que el modelo es correcto se obtendrá la función de transferencia a partir del modelo de variables de estado a través de:

$$\frac{Y(s)}{U(s)} = \mathbf{H}(s\mathbf{I} - \mathbf{F})^{-1}\mathbf{G} + \mathbf{J}$$

donde:

$$\mathbf{F} = \begin{bmatrix} -0.1 & 0 \\ 0.5 & 0 \end{bmatrix} \qquad \mathbf{G} = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} 0 & 0.4 \end{bmatrix} \qquad \mathbf{J}=0$$

entonces sustituyendo:

$$\frac{Y(s)}{U(s)} = \frac{\begin{bmatrix} 0 & 0.4 \end{bmatrix}\begin{bmatrix} s & 0 \\ 0.5 & s+0.1 \end{bmatrix}\begin{bmatrix} 0.5 \\ 0 \end{bmatrix}}{s^2 + 0.1s} + 0$$

$$\frac{Y(s)}{U(s)} = \frac{0.1}{s^2 + 0.1s}$$

$$\frac{Y(s)}{U(s)} = \frac{1}{10s^2 + s}$$

$$\frac{Y(s)}{U(s)} = \frac{1}{s\left[\frac{s}{0.1} + 1\right]}$$

Para comprobar que ambos modelos son idénticos en forma gráfica se utiliza el siguiente modelo en Simulink de Matlab en la Figura 3.1 y con su respuesta en la Figura 3.2.

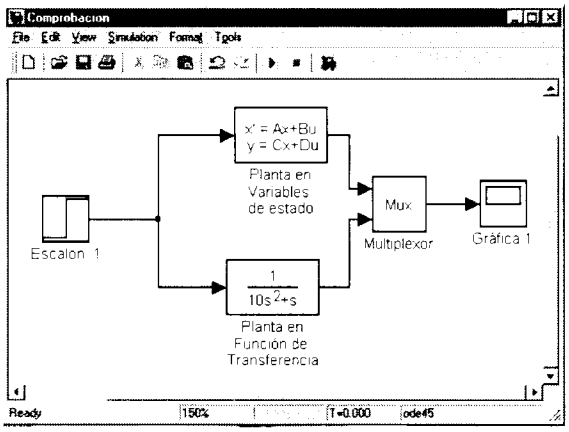


Figura 3.1: Modelo de Simulink de comprobación

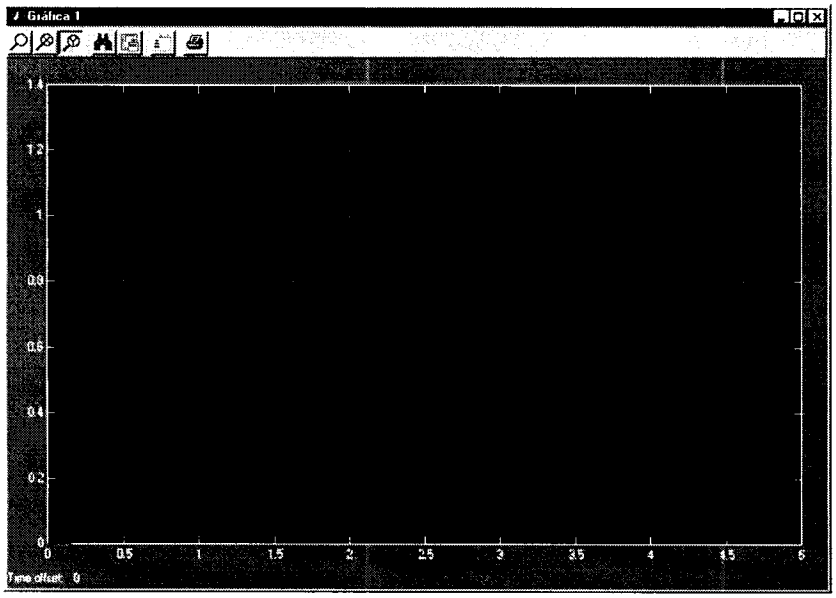


Figura 3.2: Respuesta de aplicar un escalón unitario a ambos modelos

Como se ve ambas líneas están una sobre la otra lográndose ver la misma respuesta.

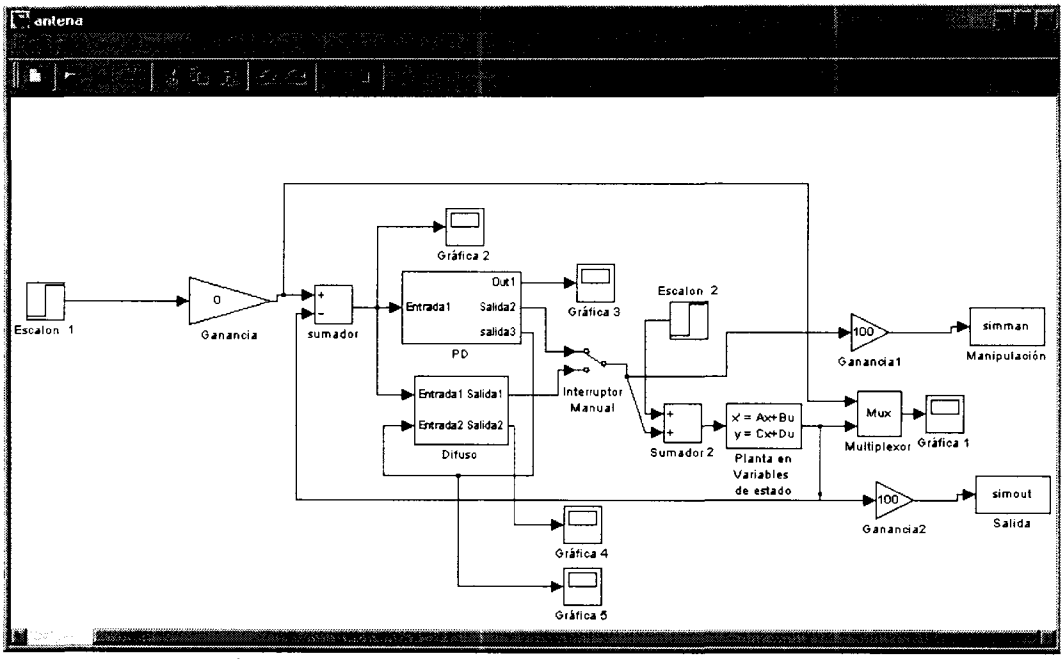


Figura 3.3: Representación en MATLAB de (3.3)

La Figura 3.3 muestra el sistema de control simulado en MATLAB por medio de Simulink utilizando el modelo de variables de estado. En el se pueden ver dos bloques, "PD" representa el controlador PD convencional mientras que "Difuso" representa el controlador Difuso.

3.1.1. Controlador PD convencional

En la Figura 3.3 el controlador PD convencional es el bloque nombrado "PD" y esta constituido por tres bloques, uno para cada acción del controlador (Proporcional, Integral, Derivativo) como se ve en la Figura 3.4, aunque la parte integral tiene un valor de cero.

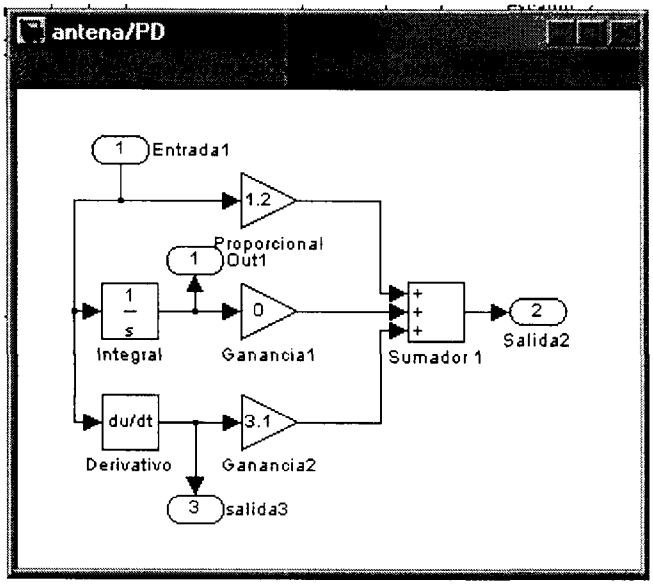


Figura 3.4: Modelo PD amplificado.

La "Entrada1" en este bloque se obtiene a través del error calculado por la relación de $\mathbf{e=Y_{Deseada}- Y_k}$. El error es alimentado a cada elemento y al final se puede sumar el efecto total para aplicarlo a la planta de pruebas por medio de "Salida2".

Para poder realizar una evaluación numérica de (3.3), se le da a la constante "a" el valor de 0.1 para definir los parámetros de la planta el cual fue sustituido en el modelo de Simulink de MATLAB. La simulación tiene una normalización en el rango de 0 a 1 para el manejo del ángulo de posición deseado. Es decir, se considera a uno como el valor deseado de elevación. Por ejemplo para el caso de Campus Monterrey la inclinación de una antena con respecto al horizontal es de aproximadamente 53°, entonces para este caso el valor de 1 en la simulación es igual a la inclinación deseada en Monterrey.

Por ser un modelo bastante sencillo el ajuste de los parámetros del controlador se realizaron a prueba y error hasta lograr un desempeño aceptable. Los valores que se encontraron por medio de esta práctica son los siguientes:

$$P = 1.2$$
$$Td = 3.1$$

La Figura 3.5 muestra la respuesta obtenida por dicho sistema en un tiempo de simulación de 70 segundos.

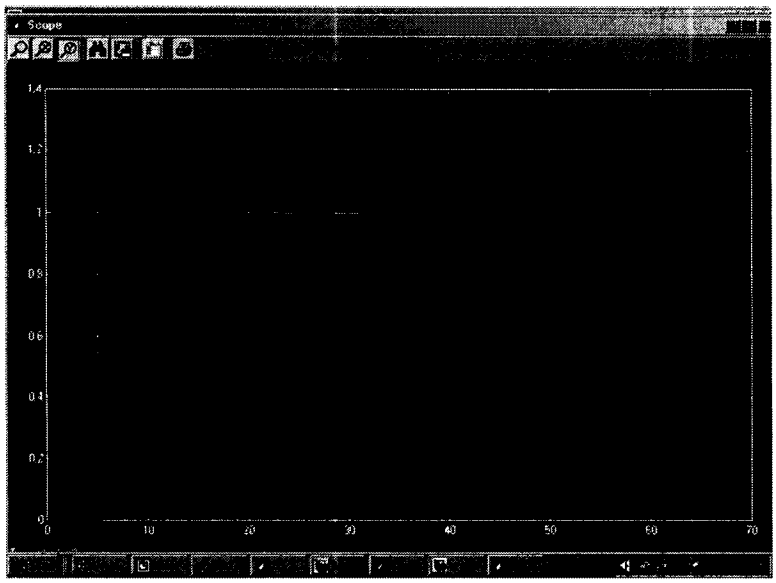


Figura 3.5: Respuesta modelo de MATLAB con la ecuación (3.3).

Como se puede observar en la simulación la respuesta es relativamente aceptable. El valor deseado se alcanza a los 30 segundos aproximadamente, el cual se puede considerar como una respuesta lenta para el tipo de modelo que se esta usando. También se puede ver que la respuesta muestra un sobretiro del 10%, el cual es muy grande y esto en un futuro puede ser de gran problema debido a que con este movimiento excesivo

podemos causar que la antena empieza a tener "juego" en las partes que la conforman causando que la calidad de la señal recibida disminuya con el tiempo.

3.1.2. Controlador difuso tipo PD

El modelo del controlador difuso esta basado en [4]. Se simuló en el mismo modelo mostrado en la Figura 3.3 pero haciendo un cambio en el interruptor manual para habilitar al bloque "Difuso" que es donde se encuentra el controlador difuso mostrado en la Figura 3.6.

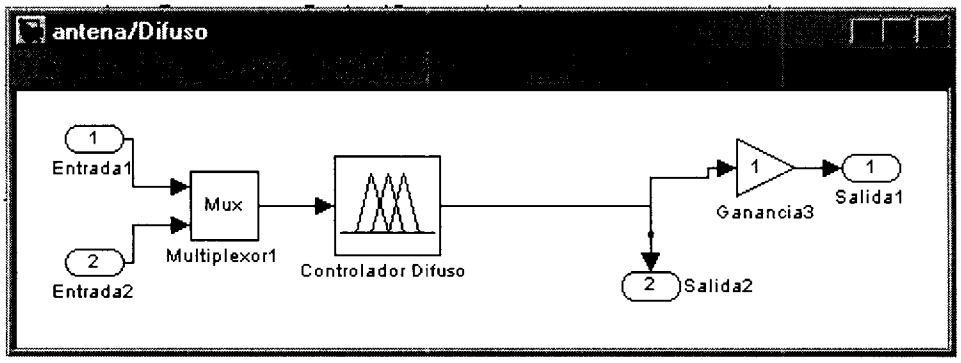


Figura 3.6: Controlador difuso

Dentro de los modelos que propone Driankov esta el uso de un controlador de dos entradas una salida. Para definir el grupo de reglas para este controlador se utilizan 5 funciones de membresía del tipo triangular para el error y para su derivada con lo que se generan un total de 25 reglas cubriendo todas las combinaciones posibles y las cuales se pueden ver en las Figuras 3.7 y 3.8.

Reglas Definidas

1. Sí (Error es NP) entonces (salida es PP)
2. Sí (Error es NM) y (Derivada es NM) entonces (salida es PM)
3. Sí (Error es NM) y (Derivada es NP) entonces (salida es PM)
4. Sí (Error es NM) y (Derivada es CE) entonces (salida es NM)
5. Sí (Error es NM) y (Derivada es PP) entonces (salida es NM)
6. Sí (Error es NM) y (Derivada es PM) entonces (salida es CE)
7. Sí (Error es NP) y (Derivada es NM) entonces (salida es NP)
8. Sí (Error es NP) y (Derivada es NP) entonces (salida es NP)
9. Sí (Error es NP) y (Derivada es CE) entonces (salida es NP)
10. Sí (Error es NP) y (Derivada es PP) entonces (salida es CE)
11. Sí (Error es NP) y (Derivada es PM) entonces (salida es PP)
12. Sí (Error es CE) y (Derivada es NM) entonces (salida es NM)
13. Sí (Error es CE) y (Derivada es NP) entonces (salida es NP)
14. Sí (Error es CE) y (Derivada es CE) entonces (salida es CE)
15. Sí (Error es CE) y (Derivada es PP) entonces (salida es PP)
16. Sí (Error es CE) y (Derivada es PM) entonces (salida es PM)
17. Sí (Error es PP) y (Derivada es NM) entonces (salida es NP)

18. Sí (Error es PP) y (Derivada es NP) entonces (salida es CE)
19. Sí (Error es PP) y (Derivada es CE) entonces (salida es PP)
20. Sí (Error es PP) y (Derivada es PP) entonces (salida es PM)
21. Sí (Error es PP) y (Derivada es PM) entonces (salida es PM)
22. Sí (Error es PM) y (Derivada es NM) entonces (salida es CE)
23. Sí (Error es PM) y (Derivada es NP) entonces (salida es PP)
24. Sí (Error es PM) y (Derivada es CE) entonces (salida es PM)
25. Sí (Error es PM) y (Derivada es PP) entonces (salida es PM)

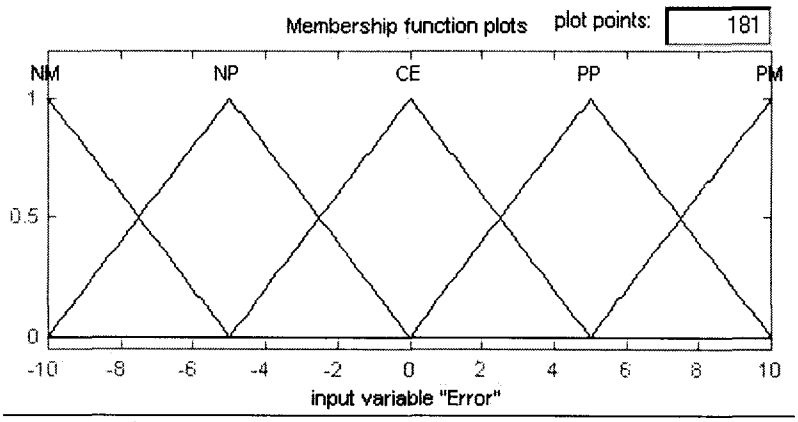


Figura 3.7: Funciones de membresía del Error, Derivada del Error, y Salida.

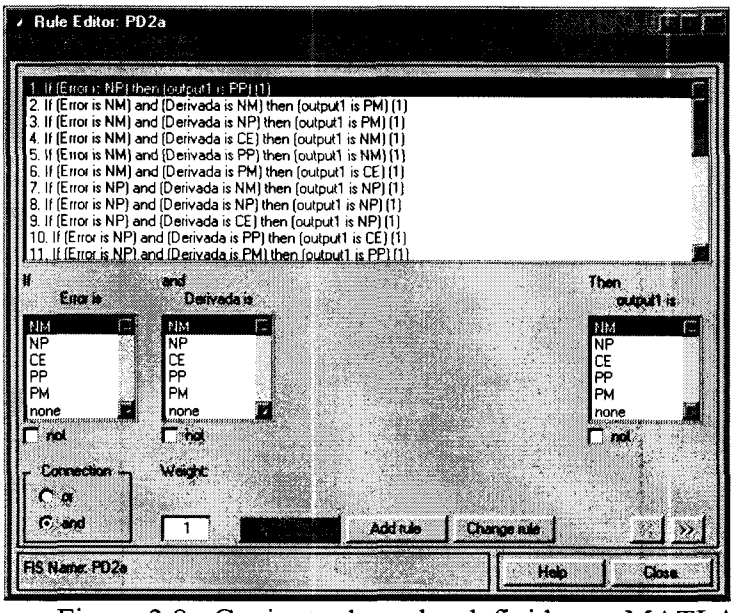


Figura 3.8: Conjunto de reglas definidas en MATLAB.

Tabla 3.1: Reglas del controlador PD difuso de 2° orden.

e \ Δe	NM	NP	CE	PP	PM
NM	PM	PM	NM	NM	CE
NP	NP	NP	NP	CE	PP
CE	NM	NP	CE	PP	PM
PP	NP	CE	PP	PM	PM
PM	CE	PP	PM	PM	PM

El rango de mapeo se define de [-10 10] y es ahí donde se distribuyen las 5 funciones de membresía.

Para poder obtener la respuesta deseada se utiliza una base de reglas tabular para definir las acciones a seguir dependiendo de las entradas como se ve en la Tabla 3.1.

En la Figura 3.9 se ve la respuesta obtenida al evaluar en las mismas condiciones de simulación del modelo de segundo orden lineal. El modelo tuvo un mejor comportamiento que el PD convencional, ya que el sobretiro se desaparece aunque el tiempo de establecimiento se va a casi tres veces más. El modelo es bueno porque el sobretiro en un futuro puede causarnos problemas con la estructura de la antena, pero si el punto de evaluación se centra en el tiempo de respuesta, pues no es el más recomendable. Para la selección de este controlador debemos de tomar en cuenta que tan crítico es para el sistema que el controlador responda lo más pronto posible, como es el caso que se busca.

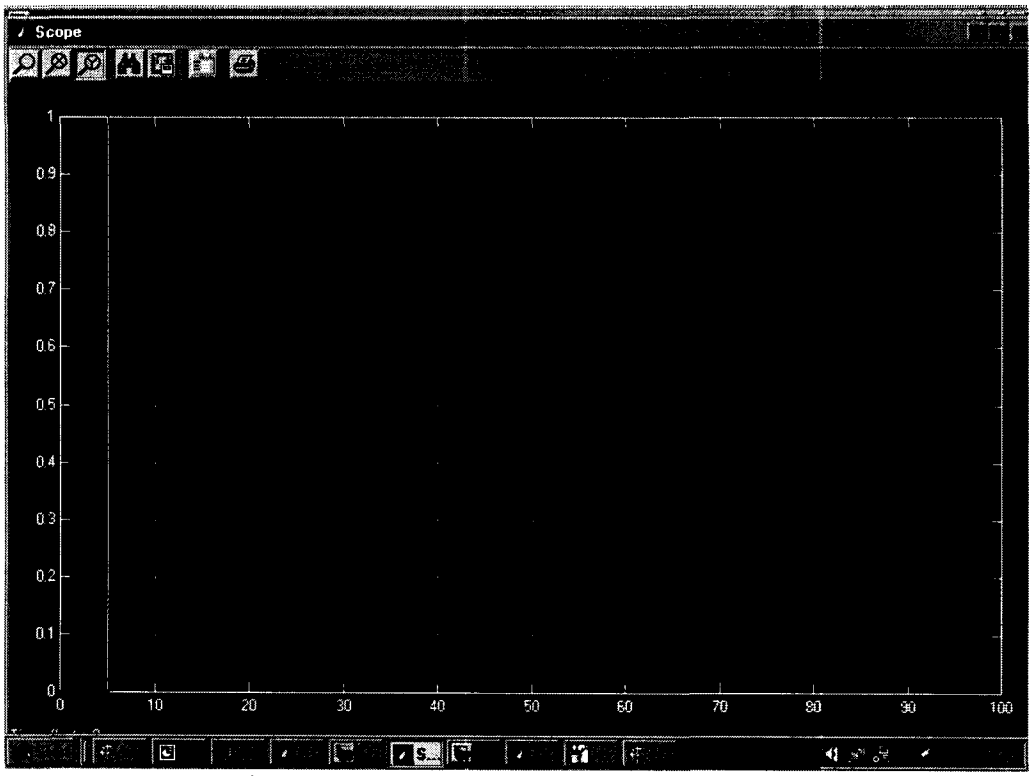


Figura 3.9: Respuesta del controlador difuso.

3.2 Modelo de antena de segundo orden parametrizado

Tratando de encontrar un modelo más acorde con la realidad se utilizó un modelo encontrado en [4].

El modelo utilizado se tomó para realizar una simulación más acorde con la realidad. La ecuación del modelo de segundo orden es la siguiente donde U(s) es la manipulación proveniente del controlador:

$$\theta(s) = \frac{1}{sN} \frac{K_{em}}{R_r(1+sT_r)sJ + K_{em}K_{me}} U(s) \tag{3.4}$$

Los datos de los parámetros que conforman dicho modelo se obtuvieron de la siguiente tabla.

Tabla 3.2: Parámetros del modelo parametrizado.

Momento Nominal	M_n	0.0195Nm
Constante Mecánico – Eléctrica	K_{me}	6mV/o/min
Constante Electromecánica	K_{em}	0.0572Nm/A
Resistencia del Rotor	Rr	1.1 ohms
Momento de Inercia	J_m	$1.6x10^{-5}$ Kgm^2
Constante de tiempo eléctrico	T_r	0.82ms
Constante de tiempo mecánico	T_m	5.3ms
Coeficiente de Transmisión	N	80
Ganancia de Control	K_c	Ajustable
Diámetro de la antena	D	1.2m
Peso	M	10Kg

La función de transferencia con los valores numéricos es:

$$\frac{49542.6829}{S^3 + 1219.5122S^2 + 23780.4878S} \tag{3.5}$$

3.2.1. Controlador PD convencional

Se selecciono este tipo de controlador debido a que el modelo muestra un componente integral que nos proporciona dicha acción.

Para realizar la sintonía de este controlador se realizó por el método de prueba y error conforme se iba corriendo la simulación.

En la Figura 3.10 podemos ver el modelo realizado en MATLAB por medio de Simulink para poder obtener la respuesta.

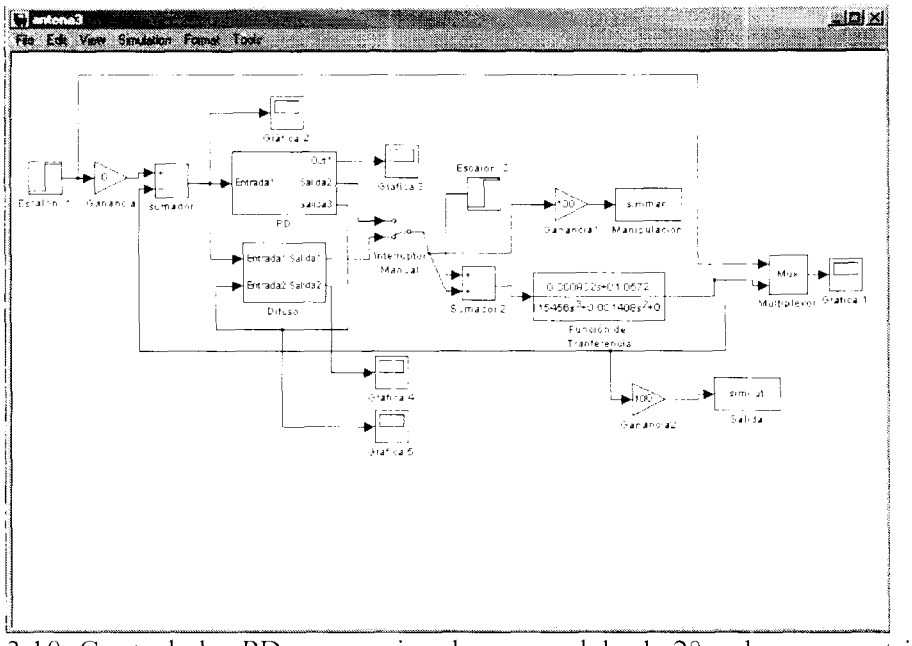


Figura 3.10. Controlador PD convencional para modelo de 2° orden parametrizado.

Los datos de sintonía para este controlador quedaron de la siguiente forma:

$$P=17$$
$$T_d=1$$

La respuesta obtenida por dicho controlador se muestra en la Figura 3.11 donde podemos ver que la respuesta es muy buena.

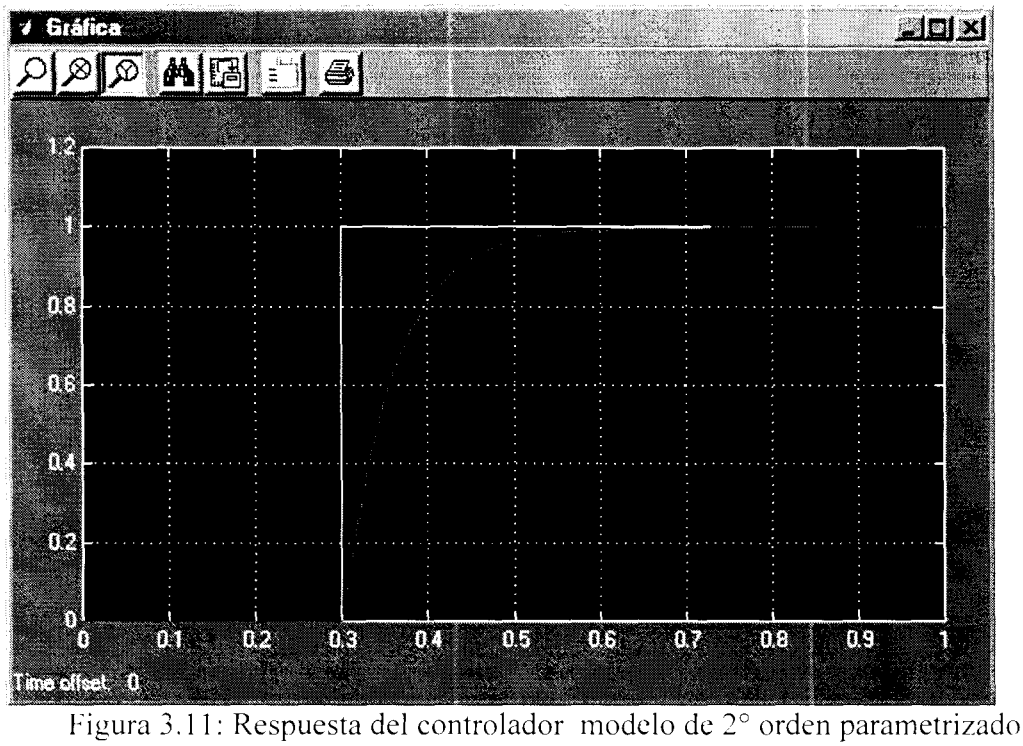


Figura 3.11: Respuesta del controlador modelo de 2° orden parametrizado.

3.2.2. Controlador difuso tipo PD

Utilizando el mismo modelo que en el punto anterior ahora se evaluara el desempeño del controlador difuso diseñado en el punto 3.1.2 con este nuevo modelo. Las reglas siguen siendo las mismas así como el diagrama utilizado en MATLAB.

La respuesta del modelo se presenta en la Figura 3.12.

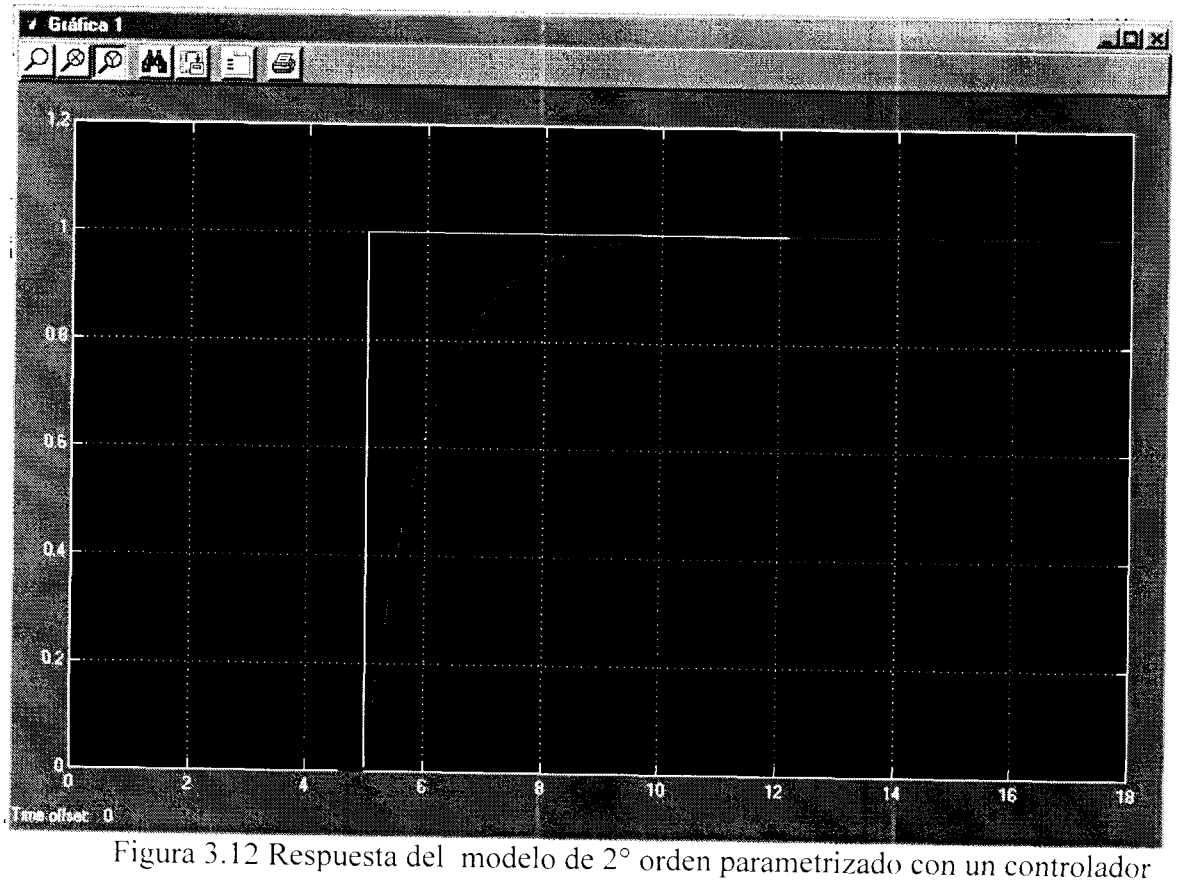


Figura 3.12 Respuesta del modelo de 2° orden parametrizado con un controlador difuso tipo PD.

Como se puede ver la respuesta es buena porque no presenta ningún sobretiro, y el sistema es llevado al punto de operación en forma lenta. El tiempo de establecimiento es de alrededor de 7 segundos.

A pesar de que la respuesta es buena aun se tiene que mejorar el tiempo de respuesta para alcanzar un desempeño al menos igual al controlador PD proporcional.

3.3 Sistema de control deslizante difuso[4]

En la Figura 3.13podemos ver los elementos del sistema de control los cuales son: θ es el ángulo de inclinación de la antena, **e= θ -θ$_d$** que es el error que se obtiene entre el ángulo de salida y el valor deseado. **Δe** es el cambio en el error, detectado como la diferencia entre el error presente y el error anterior guardado en el elemento correspondiente a la memoria. Nuestra planta llamada "Antena Satelital" sigue siendo la ecuación 3.5.

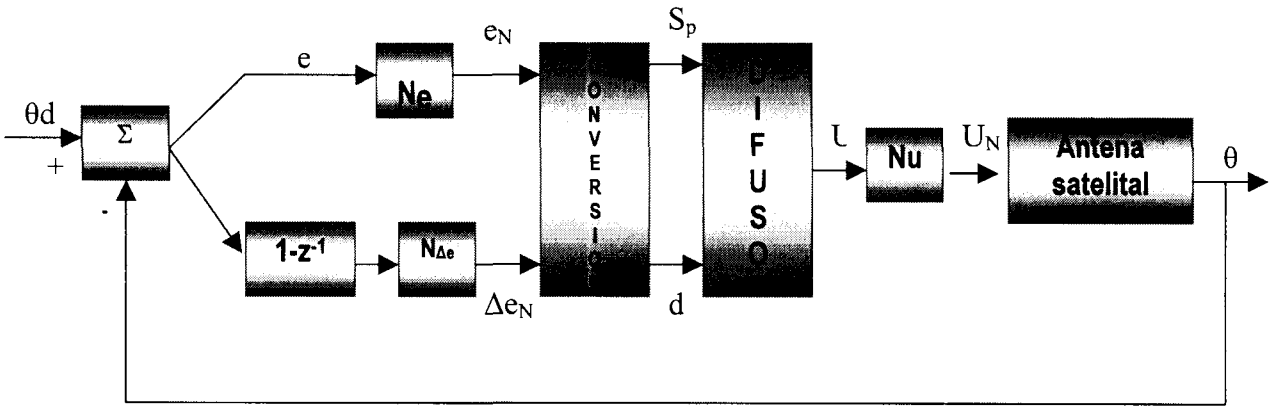


Figura 3.13 Sistema de control deslizante difuso a bloques.

La meta del controlador es mantener el ángulo de elevación de la antena satelital igual a un valor deseado.

Para determinar el comportamiento dinámico del sistema, se utilizan dos líneas P y D en el plano de fase (e_N, Δe_N):

$$P : \Delta e_N = -\lambda_N e_N \tag{3.6}$$

$$D : \Delta e_N = e_N / \lambda_N \tag{3.7}$$

El comportamiento dinámico del seguimiento del error es determinado por el parámetro λ_N como se ve en la Figura 3.14

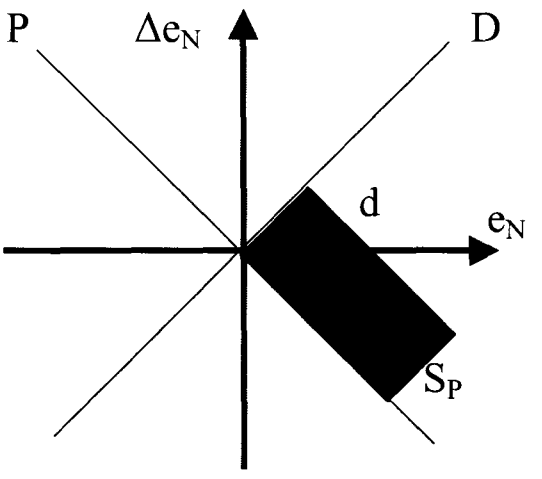


Figura 3.14 Distancias S_P, d

Basados en el error normalizado y en el cambio en el error, la distancia S_P del vector de estados (e_N, Δe_N) desde la línea P y la distancia d del vector de estados desde la línea D ortogonal a la línea P pueden ser calculados usando las siguientes ecuaciones:

$$S_P = \lambda_N e_N + \Delta e_N \tag{3.8}$$

$$d = \sqrt{|e_N|^2 - S_P^2} \tag{3.9}$$

En las Figuras 3.15 y 3.16 los valores normalizados S_{PN}, d_N son entradas del controlador difuso. Los valores de las variables lingüísticas están limitadas por un conjunto de términos lingüísticos primarios {NP, NM, NG, PP, PM, PG} para S_{PN}, u_N y {PP, PM, PG} para d_N.

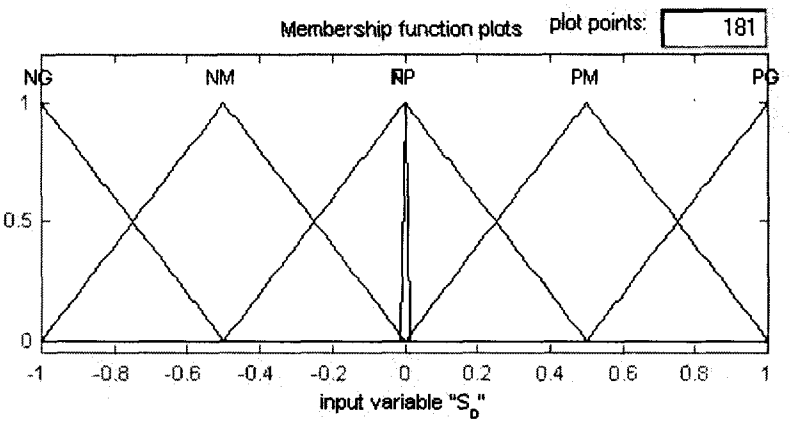


Figura 3.15: Función de membresía S_{PN}, u_N

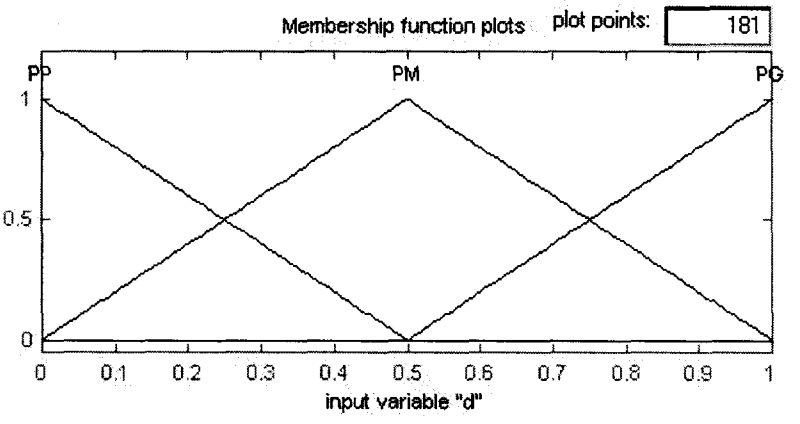


Figura 3.16: Función de membresía d_N

La estrategia de control propuesta determina la señal de control proporcional a la distancia S_{PN}, d_N de los vectores de estados de las líneas P y D. Estas leyes de control obligan al vector de estados moverse a hacia un punto de equilibrio. La base de reglas difusa que describe la ley de control es un conjunto de reglas difusas relacionadas con la descripción del proceso (S_{PN}, d_N) y la señal de control esta definida por la siguiente tabla. La defusificación que se utiliza en este controlador es el método de Centro de Gravedad.

Tabla 3.3: Tabla de reglas difusas

PG	NG	NG	NG	PG	PG	PG
PM	NG	NG	NM	PM	PG	PG
PP	NG	NM	NP	PP	PM	PG
	NG	NM	NP	PP	PM	PG

S_P

Para realizar la simulación se tuvieron que afinar varios detalles a prueba y error conforme las simulaciones se fueron dando hasta alcanzar una respuesta aceptable para el grado del problema.

En la Figura 3.17 se muestra el modelo utilizado en MATLAB para realizar la simulación por medio de Simulink.

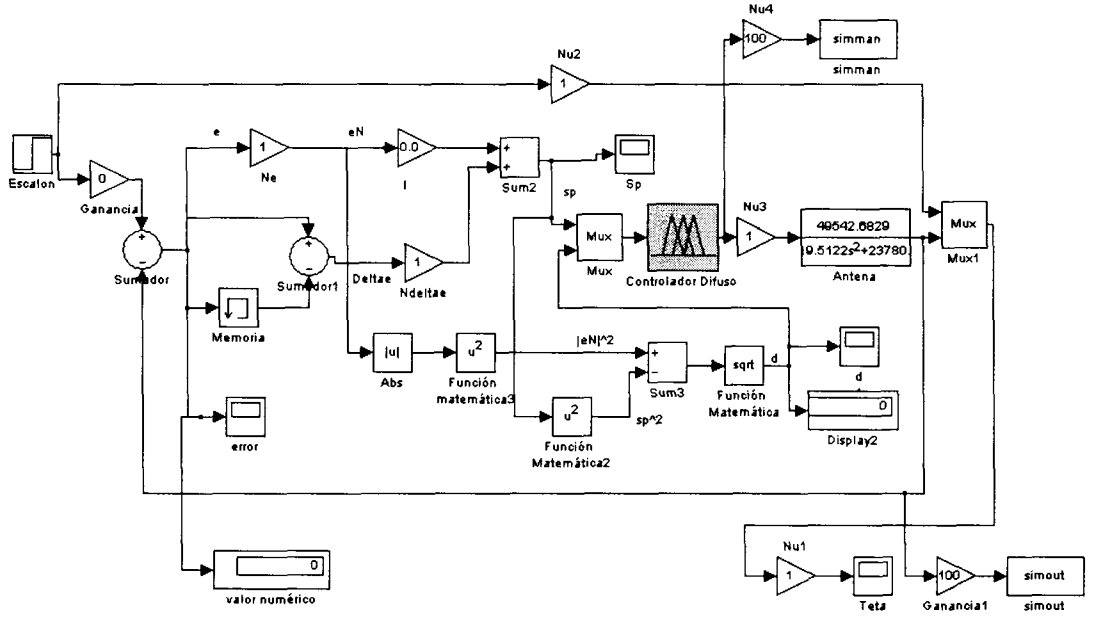


Figura 3.17: Sistema de control para el método deslizante difuso.

La respuesta que se obtuvo con este diseño fue excelente como se puede ver en la Figura 3.18.

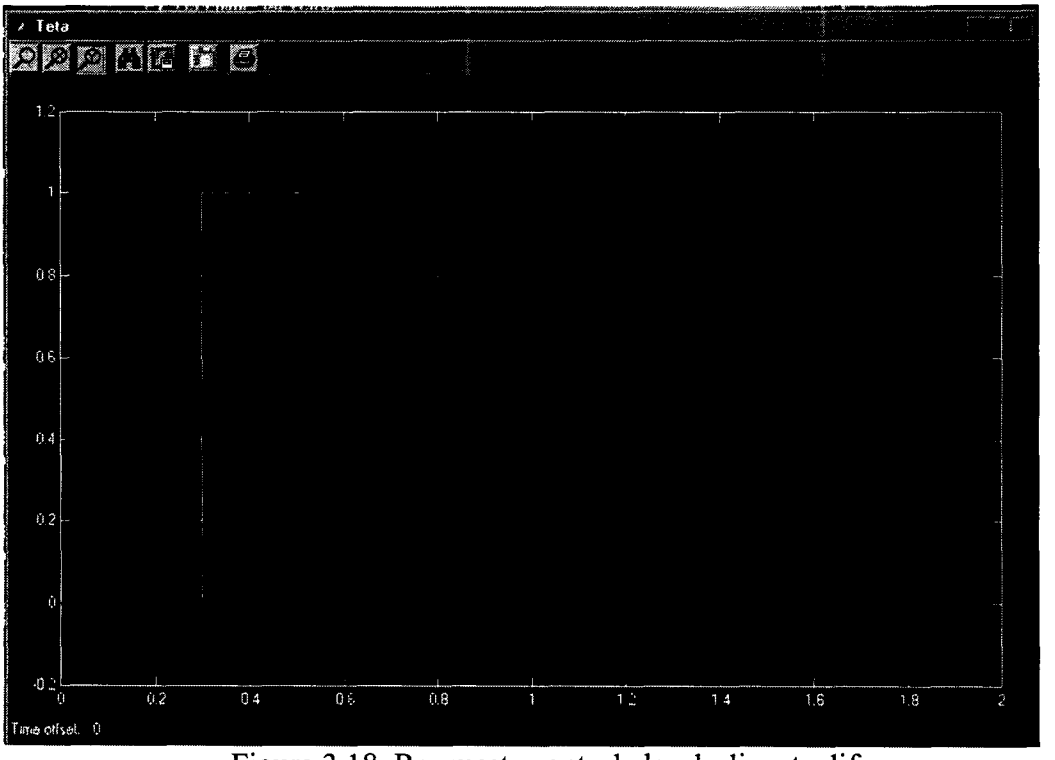


Figura 3.18. Respuesta controlador deslizante difuso.

Para poder lograr esta respuesta las funciones de membresía están en el rango de mapeo de [-10 10] con este ajuste se logro tener una respuesta aun mejor que el controlador PD convencional que se presento en la Sección 3.2. Siendo este último diseño el mejor y el que se implementara en el prototipo.

3.4 Conclusiones

Se definió un modelo teórico describiendo la dinámica de la antena y motor como base para realizar el diseño de los controladores.

Las simulaciones se hicieron bajo las mismas condiciones para los tres tipos de controladores.

Se depuró el modelo para hacerlo aun más acorde a la realidad y se verificó nuevamente la respuesta de los controladores diseñados.

A nivel simulación el controlador convencional PD es mejor en tiempo de elevación, criterio que se tomo al final para seleccionar al controlador con mejor desempeño.

Capítulo 4

Integración de Labwindows/CVI con Matlab[8]

Labwindows/CVI es una herramienta desarrollada por National Instruments y permite crear programas con librerías que permiten la adquisición de datos, pruebas y aplicaciones de medición. Labwindows/CVI es un sistema de desarrollo para programadores de C. Contiene ambiente interactivo para desarrollar programas y librerías de funciones para crear aplicaciones de adquisición de datos y instrumentos de control[11].

Contiene un ambiente interactivo para editar, compilar, ligar y depurar programas en ANSI C. Además permite usar otros compiladores de C orientados a Objetos, librerías dinámicas, librerías de C, controladores de instrumentación y archivos de código en ANSI C.

El poder de Labwindows/CVI radica en sus librerías que permiten el desarrollo de todas las fases de un sistema de control y adquisición de datos. Cuenta con 7 librerías para adquisición de datos, 3 para análisis, una para presentación por medio de su librería "Interfase de Usuario", y 4 librerías para aplicaciones de intercomunicación y comunicación por red. Por último para complementar todas estas herramientas cuenta con las librerías completas de ANSI C.

La programación de técnicas más robustas como las que ofrece las técnicas de la Inteligencia Computacional, no es una cosa fácil de hacer en C. Para desarrollar estas técnicas es necesario tener un buen conocimiento del lenguaje, poder identificar errores de compilación y poder representar estructuras altamente complejas.

Muchas técnicas de inteligencia computacional han sido desarrolladas en otros paquetes como es el caso de Matlab, paquete computacional desarrollado por Mathworks para facilitar el uso de funciones numéricas con números complejos, vectores y matrices; además permite desarrollo de sistemas para su validación y prueba. Matlab es una herramienta que se puede aprender en muy poco tiempo ya que muchos métodos numéricos ya están integrados haciendo la programación más rápida.

La intención de la integración de Matlab con Labwindows /CVI es el poder explotar las ventajas que nos cada una de estas herramientas y permitir que cualquier persona se oriente más al diseño de controladores que a programarlos.

4.1 Metodología de Integración

El problema principal al realizar esta integración esta en poder realizar comunicación entre ambas herramientas y el cómo establecer o crear los comandos para decirles que hacer a cada cual.

Para realizar la comunicación podemos apoyarnos con los programas que están en la página de web de National Instruments[12], en ella se encuentran las librerías para realizar la comunicación.

Las utilerías que se deben de agregar al programa en C son:

```
#include "matlabsrvr.h"
#include <cvirte.h>
#include <userint.h>
#include "controladortiso.h"
#include <cviauto.h>
#include "matlabUtil.h"
#include <ansi_c.h>
#include <utility.h>
#include <formatio.h>
#include <string.h>
#include <stdio.h>
```

La utilería matlabsrvr.h y matlabUtil.h son las que permiten establecer la comunicación con Matlab, como el abrir Matlab, cerrarlo, mandar "strings", mandar y recibir matrices, entre otras funciones. Estos dos archivos se deben de agregar al proyecto de Labwindows/CVI así como los archivos matlabsrvr.c, matlabsrvr.fp, matlabutil.c para que los comandos funcionen. En la Figura 4.1 se ve como se vería un proyecto con estos archivos.

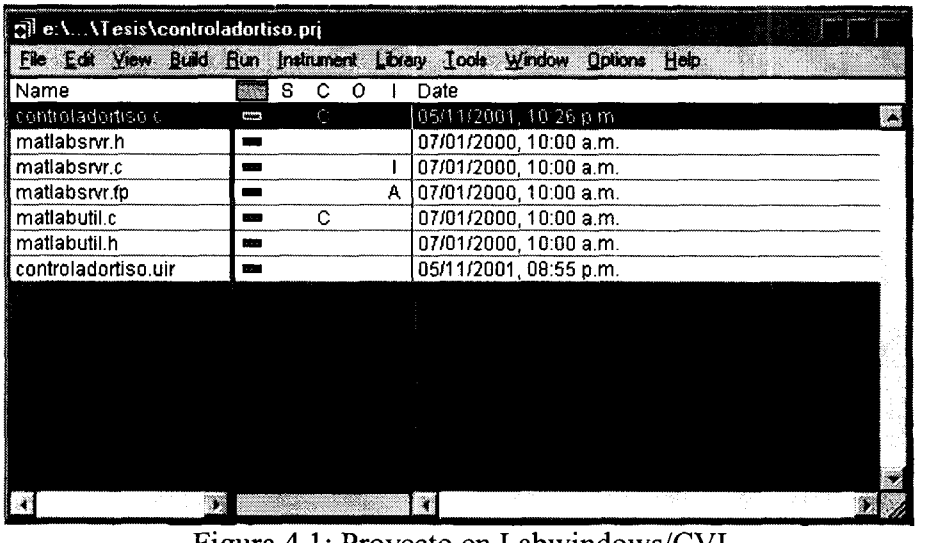


Figura 4.1: Proyecto en Labwindows/CVI

Los archivos principales del proyecto de Labwindows/CVI son Controladortiso.c y controladortiso.uir que son el código del programa y la interfaz del usuario respectivamente.

Para poder abrir y cerrar Matlab desde Labwindows/CVI utilizamos los comandos siguientes:

stat = MLApp_NewDIMLApp (NULL, &hMatlab);
result = CloseMatlab(&hMatlab);

Al ejecutarse este comando maximiza la ventana de Matlab, y las variables a las cuales están asignados los comandos permiten manejar mensajes de error en caso de que estos se presenten. Si se desea minimizar y tratar de que su uso sea de la forma más discreta posible puede minimizar la ventana por medio del comando:

MinMaxMatlab(hMatlab, 0);

Para mandar un comando y que se ejecute en Matlab, el comando que permite hacer esto es el siguiente:

RunMatlabCommand(hMatlab, command); (4.1)

Esta instrucción ejecuta el comando que esta en la variable "command". Para poder crear este comando debemos entender que al trabajar con Matlab se teclea la instrucción y hasta que se presiona ejecutar, Matlab interpreta cada una de las palabras que escribes en el área de trabajo y les da un significado.

Para poder procesar variables numéricas usadas en Labwindows/CVI en Matlab, es necesario un procesamiento para convertir la variable a una cadena de caracteres. Digamos que se desea leer la referencia que se quiere tener en un proceso desde Labwindows, por decir el nivel de un tanque y mandar este dato para que se ejecute en un modelo de Simulink que esta en Matlab, entonces se puede utilizar:

```
void crear_referenciaPID2order()
{
    //char command[300]={0},command1[300]={0};
    int k;

    for (k=0; k < 300; k++)
            {
            command[k] = 0;
            command1[k] = 0;
            }
    sprintf(command,"%f",ref);
    strcat(command1,"set_param('antena/Ganancia','Gain',' ");
    strcat(command1, command);
```

strcat(command1, "/100')") }

En "command1" se guarda el comando ya con el valor numérico convertido a una cadena de caracteres por medio de sprintf y después se concatena al comando a ejecutar en Matlab por medio de strcat. Se ejecuta este comando por medio de (4.1).

4.2 Presentación de modelos de Matlab en Labwindows/CVI

Con la integación de Matlab a Labwindows/CVI se pueden presentar los mismos modelos que en el capítulo 3 pero manipulados desde Labwindows/CVI.

La interfaz que se utiliza es la presentada en la Figura 4.2

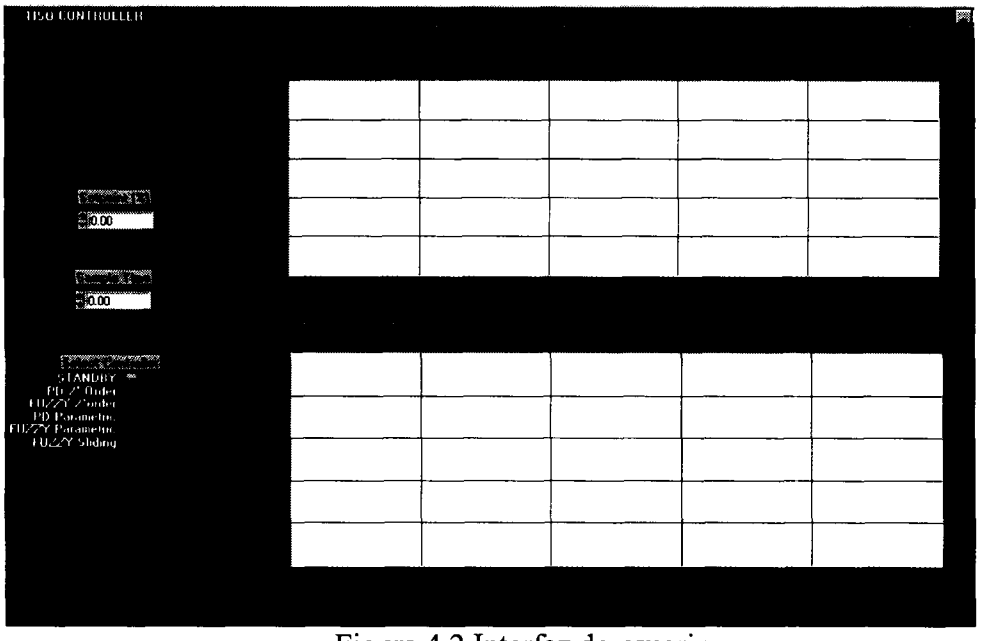


Figura 4.2 Interfaz de usuario.

Al presionar el botón de START el programa abre Matlab dejando la ventana minimizada en la barra de tareas. Al terminar pide seleccionar el controlador o modelo a ejecutar. El programa empieza a procesar la orden hasta que en Setpoint se pone el valor de potencia que se desea mantener. Automáticamente Labwindows/CVI ejecuta las instrucciones necesarias para abrir el modelo correspondiente al controlador, mandar el valor al cual se desea controlar y después de procesar los datos nos regresa los datos en forma de cadenas de caracteres gracias al siguiente comando:

result= GetMatrix(hMatlab, "simout", &simoutReal, &simoutImag, &dim1, &dim2)

;

El comando guarda los datos en una matriz que utiliza apuntadores. Por medio de una rutina llamada PLOT se grafican los datos en Labwindows/CVI.

Mientras suceden todos los cálculos, detrás de la ventana del controlador se abren las ventanas con los modelos de Simulink correspondientes al controlador que se ejecuta en ese momento. Si se ejecuta una vez cada controlador las ventanas permanecen abiertas hasta que se presiona el botón de "QUIT" que detiene todo el programa y que a la vez manda la instrucción para que Matlab se cierre.

En las Figuras 4.3, 4.4, 4.5, 4.6 y 4.7 siguientes se presentan las Figuras correspondientes a cada uno de los modelos.

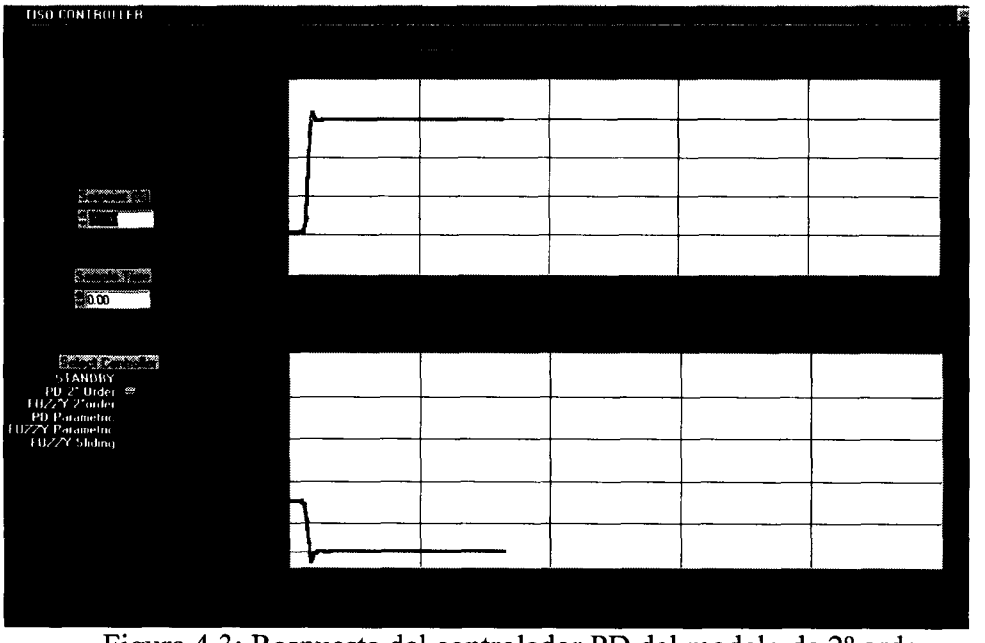


Figura 4.3: Respuesta del controlador PD del modelo de 2° orden.

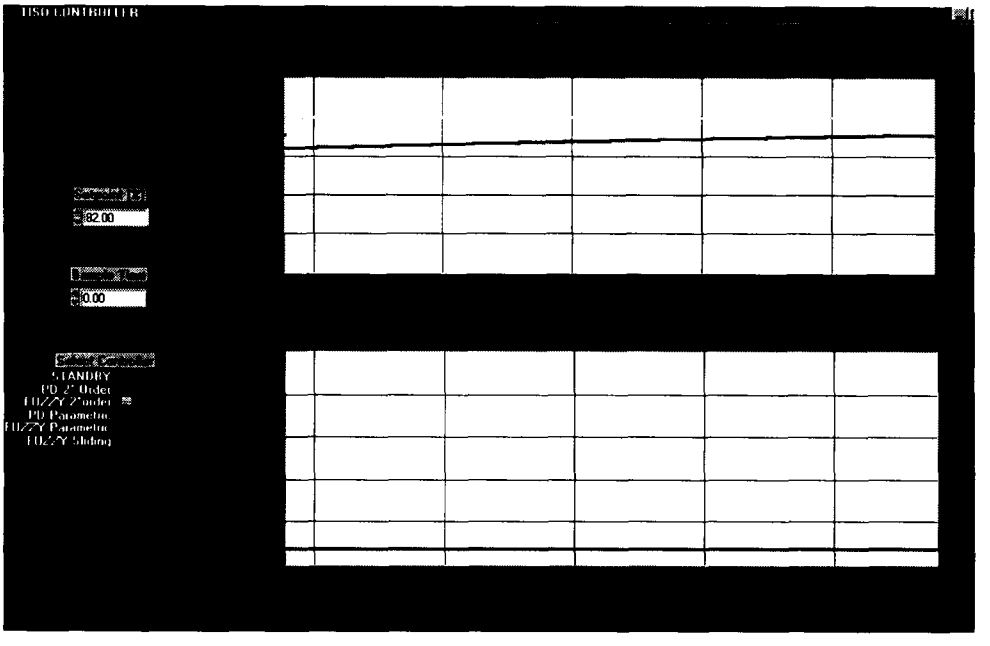


Figura 4.4: Respuesta del Controlador difuso tipo PD del modelo de 2° orden.

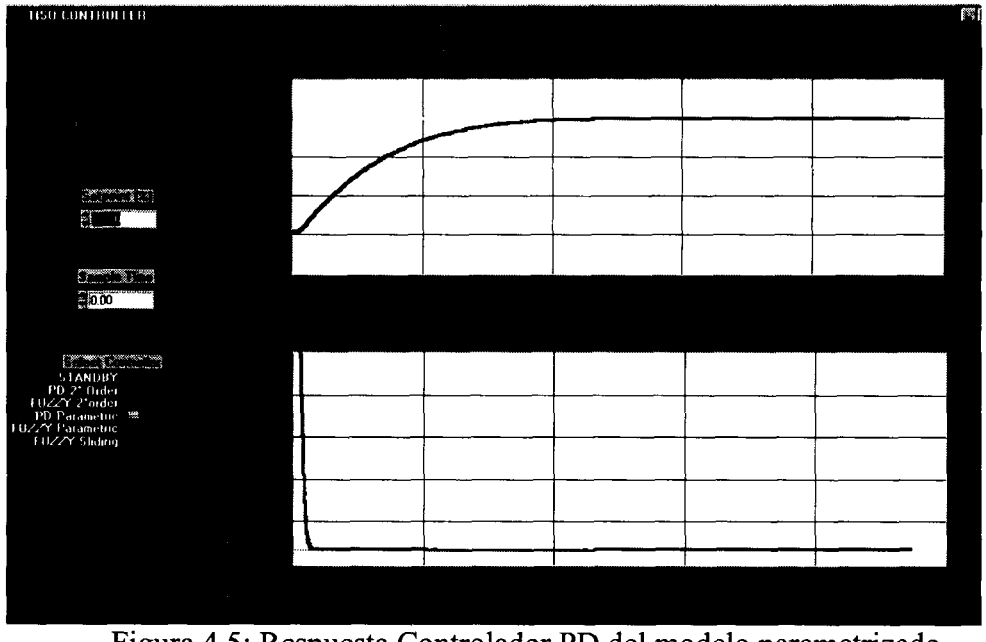


Figura 4.5: Respuesta Controlador PD del modelo parametrizado.

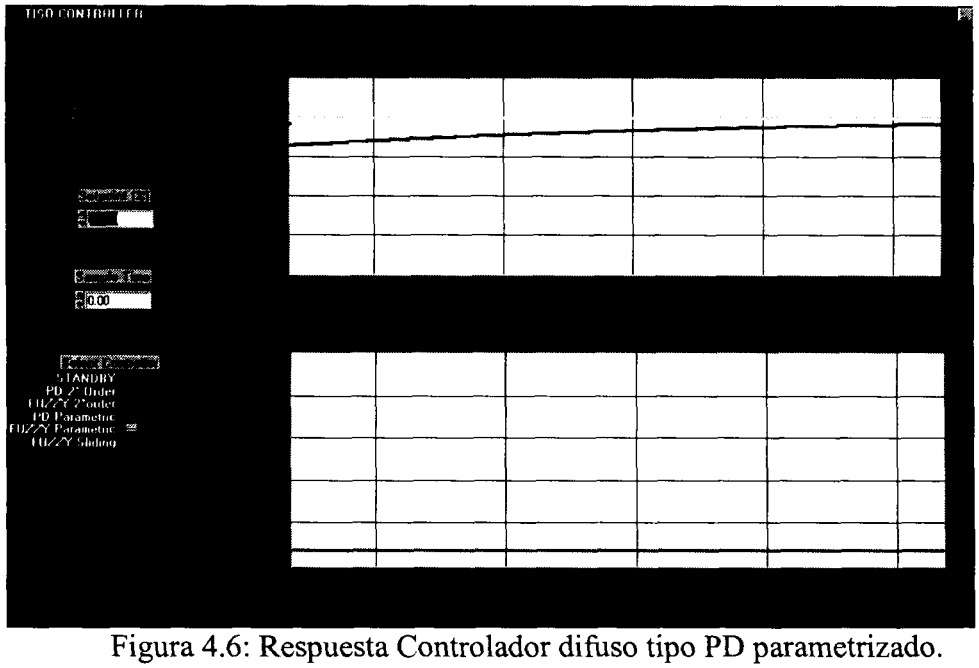


Figura 4.6: Respuesta Controlador difuso tipo PD parametrizado.

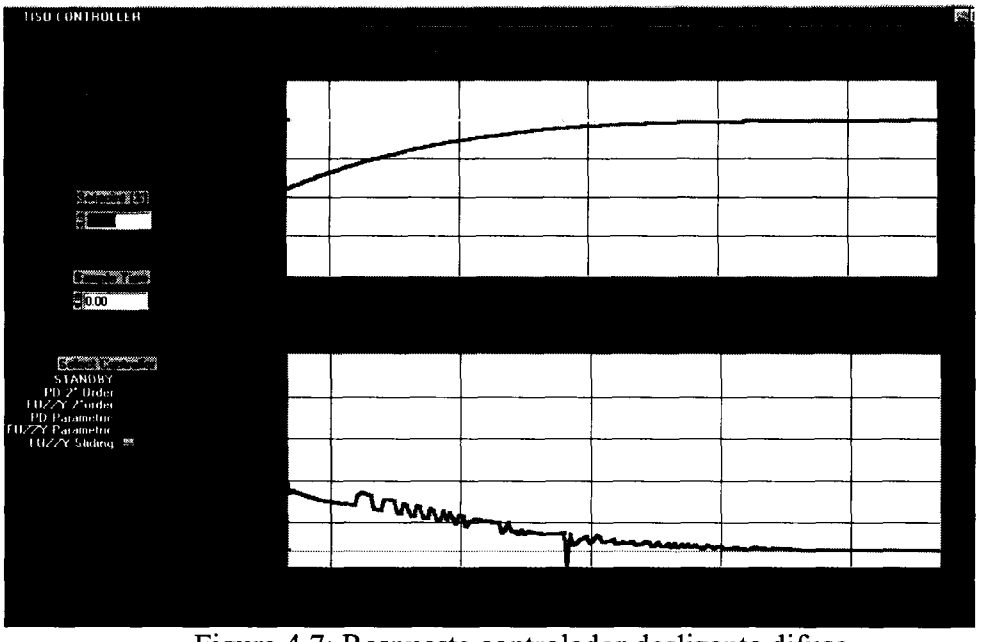


Figura 4.7: Respuesta controlador deslizante difuso.

En las Figuras se puede ver que el comportamiento visto en las simulaciones de Matlab es el mismo, el detalle que al pasar los datos a Labwindows/CVI, las gráficas no son con respecto al tiempo sino contra el número de datos o puntos calculados durante la simulación ya que la interfaz es una ventana estática que en el área de graficado utiliza un corrimiento en el área blanca para poder mostrar todos los datos.

La ventana no se dejo de un tamaño fijo debido a que la interfaz esta pensada para trabajar en tiempo real y en forma continua, entonces se espera que este presentando datos todo el tiempo por lo que la idea de manejar una ventana estática no se tomó como una opción viable.

4.3 Conclusiones

Para la comunicación entre Matlab y Labwindows/CVI se necesita el agregar los archivos mostrados en la Figura 4.1

Al trabajar con modelos de Simulink tienen la desventaja que no se puede cambiar el "Setpoint" mientras está corriendo la simulación en la versión 5.3 de Matlab. Por consiguiente este dato no se puede alterar o modificar hasta que haya terminado la presentación de los datos de manera gráfica.

Por medio de este capítulo se logra observar que Matlab ayuda a resolver ciertos problemas de forma rápida, pero también tiene sus desventajas ya que Simulink no nos evita él tener que programar cada una de las líneas de código del controlador para implementaciones en tiempo real.

Capítulo 5

Conclusiones

La contribución principal de este trabajo de tesis es el diseño y evaluación de controladores para la elevación de una antena Satelital, basándose en el resultado de comparar dichos controladores: PD convencional, un controlador difuso y un controlador deslizante difuso.

5.1 Controlador PD convencional

Los controladores convencionales son sencillos cuando se tienen modelos lineales. Para el caso del modelo lineal de la antena de segundo orden se reafirma que el ajuste o sintonía de dichos controladores puede hacerse a prueba y error y obtener una respuesta excelente a la hora de implementarlos.

En el caso del modelo de segundo orden parametrizado, el controlador convencional tiene una respuesta excelente y esto se debe a que el modelo sigue siendo lineal por lo que al controlador no se le exige demasiado y en si la parte proporcional del controlador es la que hace todo el trabajo tomando un valor grande para hacer al sistema más rápido en la respuesta.

Después de ver estos dos casos y compararlos con el controlador Difuso tipo PD y el controlador deslizante difuso, el controlador PD convencional es la mejor opción. Pero al realizar la implantación del prototipo es muy posible que no tenga el mismo comportamiento porque en este caso ya se tendrán presentes las no linealidades de la realidad donde los controladores convencionales ya no son buena opción.

5.2 Controlador difuso tipo PD

La Lógica Difusa tiene la ventaja de permitir aprovechar el conocimiento del operador para diseñar el controlador, pero también puede no ser la solución optima. La Lógica Difusa es buena opción cuando se presentan no linealidades en el modelo de la planta como es el caso de una antena satelital en la realidad.

El controlador diseñado en esta tesis fue evaluado en modelos lineales por lo que su respuesta fue buena pero no lo suficientemente rápida como para ser seleccionado como el controlador ideal. Cabe recalcar que este comportamiento puede no ser el mismo a la hora de implantar el prototipo e inclusive se puede necesitar modificar algunas reglas para obtener el desempeño deseado.

Con todo esto la hipótesis propuesta en un inicio de que la Lógica Difusa es la mejor opción para este tipo de problemas relacionados con antenas satelitales no se puede llegar a comprobar debido al tener solamente modelos lineales para su comprobación y a

la falta de un prototipo real que muestre las ventajas ante las no linealidades de la antena satelital, así como todas las no linealidades que se pueden presentar por estar expuestos al medio ambiente.

5.3 Controlador deslizante difuso

El controlador deslizante difuso puede llegar a ser el mejor controlador para implantarse en la realidad debido a que tiene las ventajas de la Lógica Difusa de mejor respuesta ante no linealidades.

Con la combinación del método de modo deslizante el controlador se vuelve aun más robusto ya que esta técnica tiene como principal característica el ser idóneo ante dinámicas no modeladas en el sistema a la hora de diseñar el controlador.

No se debe descartar este tipo de controladores a la hora de implantar el prototipo y de evaluar un controlador. Pero en el caso del diseño de [4] se debe considerar que hay normalizaciones que no se presentan o se justifican por lo que no se sabe que consideraciones se hicieron para incluirlas en el diseño.

5.4 Problemas presentados durante el desarrollo

En un principio para implantar la solución del controlador en la realidad se pensaba en Labview, paquete desarrollado también por National Instruments como una solución por la ventaja de tener un modulo de control implementado en Lógica Difusa, así que directamente se pueden definir las reglas y las funciones de membresía en forma gráfica. El contar con esta herramienta es una gran ventaja porque no se tiene que programar todo el controlador.

La desventaja de Labview es que es un lenguaje de programación gráfico (y cerrado), así que sólo tomas el bloque con la operación o función deseada y conectas la entrada y salida correspondiente. Si por alguna razón no se entiende el cómo fue hecho ese bloque, no se puede saber exactamente como es que trabaja. Labview no permite ver el código o en su defecto agregar código propio en C o ANSI C al programa por lo que esto motivó a optar por la opción de Labwindows/CVI, que aunque no cuenta con el controlador difuso implantado para solo ser configurado, si permite ver el código fuente e insertar o modificar las instrucciones que el programador considere necesarias.

El no contar con una experiencia amplia en programación provocó que el tiempo de aprendizaje aumentara por lo que se tuvo que buscar otras opciones que permitieran ser más eficientes en tiempo, por lo que se optó por la integración de Labwindows/CVI con Matlab.

Al trabajar con la integración de Matlab y Labwindows/CVI aprovechando el módulo de Simulink de Matlab, el problema presentado es que no se puede explotar al

100% dado que en la versión estudiantil de Matlab no se puede cambiar ningún parámetro del modelo cuando la simulación sé esta corriendo. Es por ello que se debe de traducir los modelos en Simulink a archivos tipo "m" de matlab y manejar todas las operaciones por instrucciones pero aprovechando ya las instrucciones que define Matlab para realizar operaciones con Lógica Difusa. Se recomienda revisar la versión profesional y sus opciones.

Aunado a todas estas variables no consideradas en un inicio, el contar con un patrocinio para poder adquirir el brazo digital para el prototipo por medio del ITESM y el proceso de compras que manejan, el tiempo de entrega sé retrasó un semestre provocando que no se contará con el prototipo real al final y tener que redefinir el alcance de la tesis como tal.

5.5 Recomendaciones para trabajos futuros

Todo el trabajo presentado en esta tesis va encaminado al diseño y construcción de una antena satelital con control automático de potencia. Una vez que se tenga el prototipo de antena automática implantado, los trabajos futuros pendientes relacionados son:

- Modificar los modelos de Simulink convirtiéndolos a archivos tipo m en Matlab para poder manejar el controlador en tiempo real.
- Modificar la arquitectura del brazo mecánico para hacer el movimiento de extensión y contracción más rápido.
- Implantar el prototipo en la realidad, es decir todo el sistema y hacer las adecuaciones necesarias.
- Evaluar si el controlador PD convencional tiene tan buena respuesta en el prototipo tomando como referencia la potencia de la señal.
- Evaluar nuevamente si el controlador PD tiene la misma respuesta satisfactoria que con el ángulo de elevación como referencia.
- Evaluar nuevamente al controlador difuso y al deslizante difuso y valorar sus índices de desempeño
- Evaluar y proponer alguna técnica para obtener un modelo más real de la antena una vez que se tiene armado ya el prototipo.
- Evaluar si el uso de redes neuronales y su entrenamiento pueden ser una opción para esta aplicación.

Hay que tomar en cuenta que este controlador propuesto lo van a utilizar personas que no tienen el mínimo conocimiento de control que están más relacionadas con el trabajo administrativo y que en las sedes y Campus del Sistema ITESM de Universidad Virtual tienen un papel donde las circunstancias los hacen hacer de todo por lo que aun queda mucho por hacer y mejorar.

Bibliografía

[1] F. Baylin, *Digital Satellite TV*, 1997. Baylin Publications, 1997.

[2] F. Baylin, B. Gale, and R. Long, *Home Satellite TV Installation and Troubleshooting Manual* , Baylin Publicatios, 1997.

[3] J.S. Beasley y J.T. PageII. *Computing Azimuth and Elevation Angles with JavaScript,* Volume 3 No.2, Spring 1999, http://web.bsu.edu/tti/3_2/3_2e.htm

[4] S. Dimitrijevic y D. Antic. *Satellite Antenna Positioning Using Two Inputs Single Output Robust Fuzzy Controller,* IEEE International Conference TELSIKS'99, October 1999, Nis, Yugoslavia, pp 13 – 15.

[5]G. F. Franklin, J.D. Powell y A. E. Nawini. *Feedback Control of Dynamic Systems,*Publicaciones Addison Wesley, pp. 583 y 584. Junio 1986.

[6] M Gupta and N. Sinha. *Intelligent Control Systems, Theory and Aplications.* IEEE PRESS, 1996

[7] J. E. Slotine y W. Li, *Applied Nonlinear Control,* Prentice Hall, pp. 276-309, 1991

[8] L. M. Torres, *Integración de LabWindows/CVI con Matlab 5.1,* Control Inteligente, Centro de Inteligencia Artificial del ITESM Campus Monterrey, Dic 1999.

[9] L. Wang. *A Course in Fuzzy System and Control,* Prentice Hall, pp. 2-7,1997

[10] J. Yen, R Langari, y L. A. Zadeh, *Industrial Applications of Fuzzy Logic and Intelligent Systems,* IEEE PRESS, 1995.

[11] *LabWindows/CVI: Getting Started with Labwindows/CVI,* National Instruments Corporation, Capítulo 1, pp. 2-5, Febrero 1998

[12] Página principal de National Instruments, Marzo del 2001, www.ni.com

Apéndice

```
#include "matlabsrvr.h"
#include <cvirte.h>
#include <userint.h>
#include "controladortiso.h"
#include <cviauto.h>
#include "matlabUtil.h"
#include <ansi_c.h>
#include <utility.h>
#include <formatio.h>
#include <string.h>
#include <stdio.h>


static int panel1;
static CAObjHandle  hMatlab = 0;
char
command[300],command1[300],command2[300],command3[300],command4[300],com
mand5[300],command6[300],command7[300],command8[300],command9[300];
double ref, tmuestreo;
int controller, controller2, pid, banderapid = 1, banderafuzzy = 1;
double      *simoutReal          =  NULL;
double      *simoutImag          =  NULL;
double      *simmanReal          =  NULL;
double      *simmanImag          =  NULL;
double      *simerrorReal        =  NULL;
double      *simerrorImag        =  NULL;
unsigned     dim1                =  0;
unsigned     dim2                =  0;
unsigned     dim3                =  0;
unsigned     dim4                =  0;
unsigned     dim5                =  0;
unsigned     dim6                =  0;




/*---------------------------------------------------------------------------*/
/* This function creates the command to change the reference  in Matlab      */
/*---------------------------------------------------------------------------*/


void crear_referenciaPID2order()
{
```

```
        int k;

        for (k=0; k < 300; k++)
                {
                command[k] = 0;
                command1[k] = 0;
                }
        sprintf(command,"%f",ref);
        strcat(command1,"set_param('antena/Ganancia','Gain',' ");
        strcat(command1, command);
        strcat(command1, "/100')");
}



void crear_referenciaFUZZY2order()
{

        int k;

        for (k=0; k < 300; k++)
                {
                command2[k] = 0;
                command3[k] = 0;
                }
        sprintf(command2,"%f",ref);
        strcat(command3,"set_param('antena1/Ganancia','Gain',' ");
        strcat(command3, command);
        strcat(command3, "*0.35212/100')");
}

void crear_referenciaPIDparametric()
{

        int k;

        for (k=0; k < 300; k++)
                {
                command4[k] = 0;
                command5[k] = 0;
                }
        sprintf(command4,"%f",ref);
        strcat(command5,"set_param('antena2/Ganancia','Gain',' ");
        strcat(command5, command);
        strcat(command5, "/100')");
```

```
}

void crear_referenciaFUZZYparametric()
{

        int k;

        for (k=0; k < 300; k++)
                {
                command6[k] = 0;
                command7[k] = 0;
                }
        sprintf(command6,"%f",ref);
        strcat(command7,"set_param('antena3/Ganancia','Gain',' ");
        strcat(command7, command);
        strcat(command7, "/100')");
}

void crear_referenciaFUZZYsliding()
{

        int k;

        for (k=0; k < 300; k++)
                {
                command8[k] = 0;
                command9[k] = 0;
                }
        sprintf(command8,"%f",ref);
        strcat(command9,"set_param('saptiso/Ganancia','Gain',' ");
        strcat(command9, command);
        strcat(command9, "/100')");
}


/*-------------------------------------------------------------------------*/
/* This functions open models in Matlab, runs the simulation and           */
/* leaves the resuls in a variable called simout and simman at workspace   */
/*-------------------------------------------------------------------------*/


int runpid2order()
{       int    result  =  0;
```

```
            ClearStripChart(PANEL1, PANEL1_Setpoint_Power);
            ClearStripChart(PANEL1, PANEL1_Manipulation_Error);
            RunMatlabCommand(hMatlab, "open_system('antena')");
        RunMatlabCommand(hMatlab, "load FUZZYLAURO");
        RunMatlabCommand(hMatlab, command1);
        RunMatlabCommand(hMatlab, "sim('antena')");
            RunMatlabCommand(hMatlab, "set_param('antena/Ganancia','Gain','0')");
            RunMatlabCommand(hMatlab, "save_system('antena')");
        result= GetMatrix(hMatlab, "simout", &simoutReal, &simoutImag, &dim1,
&dim2);
        if ((result != SUCCESS) || (simoutReal == NULL))
        {
          MessagePopup ("ERROR", "Error de MATLAB o variable en NULL");
          return 0;
        }

        result= GetMatrix(hMatlab, "simman", &simmanReal, &simmanImag, &dim3,
&dim4);
        if (result != SUCCESS)
        {
          MessagePopup ("ERROR", "Error in closing MATLAB");
          return 0;
        }
        result= GetMatrix(hMatlab, "simerror", &simerrorReal, &simerrorImag, &dim5,
&dim6);
        if (result != SUCCESS)
        {
          MessagePopup ("ERROR", "Error in closing MATLAB");
          return 0;
        }


  return 1;
}

int runpidparametric()
{       int    result =  0;



            ClearStripChart(PANEL1, PANEL1_Setpoint_Power);
            ClearStripChart(PANEL1, PANEL1_Manipulation_Error);
            RunMatlabCommand(hMatlab, "open_system ('antena2')");
        RunMatlabCommand(hMatlab, "load FUZZYLAURO");
        RunMatlabCommand(hMatlab, command5);
        RunMatlabCommand(hMatlab, "sim('antena2')");
```

```
                RunMatlabCommand(hMatlab,
"set_param('antena2/Ganancia','Gain','0')");
                RunMatlabCommand(hMatlab, "save_system('antena2')");
        result= GetMatrix(hMatlab, "simout", &simoutReal, &simoutImag, &dim1,
&dim2);
        if ((result != SUCCESS) || (simoutReal == NULL))
        {
          MessagePopup ("ERROR", "Error de MATLAB o variable en NULL");
          return 0;
        }

        result= GetMatrix(hMatlab, "simman", &simmanReal, &simmanImag, &dim3,
&dim4);
        if (result != SUCCESS)
        {
          MessagePopup ("ERROR", "Error in closing MATLAB");
          return 0;
        }
        result= GetMatrix(hMatlab, "simerror", &simerrorReal, &simerrorImag, &dim5,
&dim6);
        if (result != SUCCESS)
        {
          MessagePopup ("ERROR", "Error in closing MATLAB");
          return 0;
        }


  return 1;
}


int runfuzzy2order()
{       int    result =  0;

                ClearStripChart(PANEL1, PANEL1_Setpoint_Power);
                ClearStripChart(PANEL1, PANEL1_Manipulation_Error);
                RunMatlabCommand(hMatlab, "open_system ('antena1')");
                RunMatlabCommand(hMatlab, "load FUZZYLAURO");
        RunMatlabCommand(hMatlab, command3);
        RunMatlabCommand(hMatlab, "sim('antena1')");
                RunMatlabCommand(hMatlab,
"set_param('antena1/Ganancia','Gain','0')");
                RunMatlabCommand(hMatlab, "save_system('antena1')");
        result= GetMatrix(hMatlab, "simout", &simoutReal, &simoutImag, &dim1,
&dim2);
        if ((result != SUCCESS) || (simoutReal == NULL))
```

```
        {
          MessagePopup ("ERROR", "Error de MATLAB o variable en NULL");
          return 0;
        }

        result= GetMatrix(hMatlab, "simman", &simmanReal, &simmanImag, &dim3,
&dim4);
        if (result != SUCCESS)
        {
          MessagePopup ("ERROR", "Error in closing MATLAB");
          return 0;
        }


  return 1;
}



int runfuzzyparametric()
{       int    result =  0;

                ClearStripChart(PANEL1, PANEL1_Setpoint_Power);
                ClearStripChart(PANEL1, PANEL1_Manipulation_Error);
                RunMatlabCommand(hMatlab, "open_system ('antena3')");
                RunMatlabCommand(hMatlab, "load FUZZYLAURO");
        RunMatlabCommand(hMatlab, command7);
        RunMatlabCommand(hMatlab, "sim('antena3')");
                RunMatlabCommand(hMatlab,
"set_param('antena3/Ganancia','Gain','0')");
                RunMatlabCommand(hMatlab, "save_system('antena3')");
        result= GetMatrix(hMatlab, "simout", &simoutReal, &simoutImag, &dim1,
&dim2);
        if ((result != SUCCESS) || (simoutReal == NULL))
        {
          MessagePopup ("ERROR", "Error de MATLAB o variable en NULL");
          return 0;
        }

        result= GetMatrix(hMatlab, "simman", &simmanReal, &simmanImag, &dim3,
&dim4);
        if (result != SUCCESS)
        {
          MessagePopup ("ERROR", "Error in closing MATLAB");
          return 0;
        }
```

```
  return 1;
}


int runfuzzysliding()
{       int    result =  0;

                ClearStripChart(PANEL1, PANEL1_Setpoint_Power);
                ClearStripChart(PANEL1, PANEL1_Manipulation_Error);
                RunMatlabCommand(hMatlab, "open_system ('saptiso')");
                RunMatlabCommand(hMatlab, "load FUZZYLAURO");
        RunMatlabCommand(hMatlab, command9);
        RunMatlabCommand(hMatlab, "sim('saptiso')");
                RunMatlabCommand(hMatlab, "set_param('saptiso/Ganancia','Gain','0')");
                RunMatlabCommand(hMatlab, "save_system('saptiso')");
        result= GetMatrix(hMatlab, "simout", &simoutReal, &simoutImag, &dim1,
&dim2);
        if ((result != SUCCESS) || (simoutReal == NULL))
        {
          MessagePopup ("ERROR", "Error de MATLAB o variable en NULL");
          return 0;
        }

        result= GetMatrix(hMatlab, "simman", &simmanReal, &simmanImag, &dim3,
&dim4);
        if (result != SUCCESS)
        {
          MessagePopup ("ERROR", "Error in closing MATLAB");
          return 0;
        }


  return 1;
}



void PlotPID ()
{
        double trace[2], trace2[2];
        int k;
        if (simoutReal != NULL)
```

```
        {
                for (k=0; k < dim1; k=k+3)
                {
                        trace[0]= (double) simoutReal[k];
                        trace[1]= ref;
                        trace2[0]= (double) simmanReal[k];
                        PlotStripChart (PANEL1, PANEL1_Setpoint_Power,  trace, 2, 0,
0, VAL_DOUBLE);
                        PlotStripChart (PANEL1, PANEL1_Manipulation_Error, trace2, 2,
0, 0, VAL_DOUBLE);
                }
        }
}



void PlotFUZZY()
{
        double trace[2], trace2[2];
        int k;

        if (simoutReal != NULL)
        {
                for (k=0; k < dim1; k=k+3)
                {
                        trace[0]= (double) simoutReal[k];
                        trace[1]= ref;
                        trace2[0]= (double) simmanReal[k];
                        PlotStripChart (PANEL1, PANEL1_Setpoint_Power,  trace, 2, 0,
0, VAL_DOUBLE);
                        PlotStripChart (PANEL1, PANEL1_Manipulation_Error, trace2, 2,
0, 0, VAL_DOUBLE);
                }
        }
}




/*---------------------------------------------------------------------------*/
/* This function reports the Automation error associated with the code.      */
/* errCode is the Automation error code.                                     */
/* errMesg is the error message that MATLAB returns in some functions. You   */
/* can pass NULL for this.                                                   */
```

```
/*---------------------------------------------------------------------*/
void ErrFunc(HRESULT errCode, char *errMesg)
{
  char      errStr[200];

  CA_GetAutomationErrorString (errCode, errStr, 200);
  MessagePopup ("ERROR MESSAGE", errStr);
  if (errMesg)
  {
    MessagePopup ("MATLAB ERROR MESSAGE", errMesg);
    CA_FreeMemory (errMesg);
  }
}

/*---------------------------------------------------------------------*/
/* This function closes MATLAB if it is running.                       */
/* hMatlabPtr is a pointer to the handle of the MATLAB Application Object.  */
/*---------------------------------------------------------------------*/
int CloseMatlab(CAObjHandle *hMatlabPtr)
{
  HRESULT    stat  =  0;

  /* Exit MATLAB if it is running */
  if (*hMatlabPtr && hMatlabPtr)
  {
    stat = MLApp_DIMLAppQuit (*hMatlabPtr, NULL);
    if (stat < 0)
    {
      ErrFunc(stat, NULL);
      return (int)stat;
    }
    else
    {
      stat = CA_DiscardObjHandle (*hMatlabPtr);
      if (stat < 0)
      {
        ErrFunc(stat, NULL);
        return (int)stat;
      }
      *hMatlabPtr=0;
      return SUCCESS;
    }
  }
  else
    return ERROR_WRONG_HANDLE;
```

```
}


int main (int argc, char *argv[])
{
        if (InitCVIRTE (0, argv, 0) == 0)       /* Needed if linking in external compiler;
harmless otherwise */
                return -1;      /* out of memory */
        if ((panel1 = LoadPanel (0, "controladortiso.uir", PANEL1)) < 0)
                return -1;
        DisplayPanel (panel1);
        RunUserInterface ();
        return 0;
}

int CVICALLBACK abrematlab (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        HRESULT stat;
        char String1;char String2;char String3;char WriteString;

        switch (event)
                {
                case EVENT_COMMIT:
                        /* Instantiate a MATLAB application object & maximize the
window */
        stat = MLApp_NewDIMLApp (NULL, &hMatlab);
        //MessagePopup ("Wait", "MATLAB IS OPENING...");
        if (stat < 0) ErrFunc(stat, NULL);
        MinMaxMatlab(hMatlab, 1);
        MinMaxMatlab( hMatlab, 0);
        MessagePopup ("READY", "SELECT CONTROLLER");


                        break;
                case EVENT_VAL_CHANGED:

                        break;
                }
        return 0;
}

int CVICALLBACK cerrarmatlab (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        int    result =  0;
```

```
        switch (event)
                {
                case EVENT_COMMIT:
                        /* Exit MATLAB if it is running */
        result = CloseMatlab(&hMatlab);
        if (result != SUCCESS)
        {
          MessagePopup ("ERROR", "Error in closing MATLAB");
          return 0;
        }
                        QuitUserInterface (0);
                        break;
                case EVENT_VAL_CHANGED:

                        break;
                }
        return 0;
}

int CVICALLBACK Mandareferencia (int panel, int control, int event,
                void *callbackData, int eventData1, int eventData2)
{
        int     result      =  0;

        switch (event)
                {
                case EVENT_COMMIT:

                        break;
                case EVENT_VAL_CHANGED:
                        GetCtrlVal (PANEL1, PANEL1_Referencia, &ref);
                        crear_referenciaPID2order();
                        crear_referenciaFUZZY2order();
                        crear_referenciaPIDparametric();
                        crear_referenciaFUZZYparametric();
                        crear_referenciaFUZZYsliding();

                        if (controller2 == 1)
                        {
                                runpid2order();
                                PlotPID();

                        }

                        if (controller2 == 2)
```

```
                    {

                            runfuzzy2order();
                            PlotFUZZY();

                    }

                    if (controller2 == 3)
                            {
                            MessagePopup ("Warning", "Select Controller");
                            }

                    if (controller2 == 4)
                    {
                            runpidparametric();
                            PlotPID();

                    }

                    if (controller2 == 5)
                    {

                            runfuzzyparametric();
                            PlotFUZZY();

                    }

                    if (controller2 == 6)
                    {

                            runfuzzysliding();
                            PlotFUZZY();

                    }



                    break;
            }
        return 0;
}


int CVICALLBACK mandatmuestreo (int panel, int control, int event,
            void *callbackData, int eventData1, int eventData2)
```

```
{
	switch (event)
		{
		case EVENT_COMMIT:

			break;
		case EVENT_VAL_CHANGED:
			GetCtrlVal (PANEL1, PANEL1_tmuestreo, &tmuestreo);
			SetCtrlAttribute (PANEL1, PANEL1_timerlauro,
ATTR_INTERVAL,tmuestreo);
			// GetCtrlVal (PANEL1, PANEL1_tmuestreo, &tmuestreo);
			 //crear_tiempomuestreo();
			break;
		}
	return 0;
}

int CVICALLBACK Timerlauro (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_TIMER_TICK:

			break;
		}
	return 0;
}

int CVICALLBACK SelectController2 (int panel, int control, int event,
		void *callbackData, int eventData1, int eventData2)
{
	switch (event)
		{
		case EVENT_COMMIT:

			break;
		case EVENT_VAL_CHANGED:
			GetCtrlVal (PANEL1, PANEL1_RINGSLIDE, &controller2);


			break;
		}
	return 0;
}
```