ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRICA

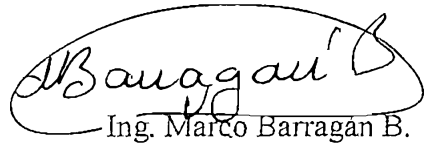
SIMULACIÓN GRÁFICA EN WINDOWS DE UN ELEVADOR EMPLEANDO TÉCNICAS DE CONTROL DIFUSO

TESIS PREVIA A LA OBTENCION DEL TITULO DE INGENIERO EN ELECTRÓNICA Y CONTROL

ALBERTO ESTEBAN SÁNCHEZ TERÁN

Quito, Agosto de 1999.

Certifico que la presente Tesis
ha sido realizada en su totalidad por el
Sr. Alberto Esteban Sánchez Terán.



Ing. Marco Barragán B.
DIRECTOR DE TESIS

DEDICATORIA

A Dios, a mi familia y
en especial a mis Padres
quienes han sido mi luz,
mi guía y ejemplo
durante toda mi vida.

La ciencia no sirve más que para darnos cuenta de la extensión de nuestra ignorancia
(Lamennais)

AGRADECIMIENTO

A mi director de Tesis, destacado profesional, maestro y amigo, Ing. Marco Barragán B., MSc., por su ayuda, paciencia y acertada dirección, factores determinantes en la elaboración del presente trabajo.

A la Escuela Politécnica Nacional y en especial a la Facultad de Ingeniería Eléctrica, por su noble labor educativa.

A todos mis amigos y compañeros que durante todo este tiempo han colaborado con su apoyo e ideas, y en especial al Ing. Francisco Salvador por su asesoramiento en programación multimedia.

ÍNDICE

CAPITULO I

INTRODUCCIÓN

I.1 IMPORTANCIA

Los elevadores, son máquinas utilizadas a diario en muchos edificios del mundo. Su uso es tan extensivo que se estima que un elevador en un edificio comercial puede llegar a recorrer una distancia equivalente a tres vueltas alrededor del mundo durante su vida útil. Los avances tecnológicos de los últimos tiempos han permitido que los constructores de estos mecanismos logren brindar al usuario de mayor confort, confiabilidad, y seguridad; así como de ser lo suficientemente versátiles para adaptarse a innovaciones arquitectónicas. Por ejemplo, hubiese sido imposible construir edificios tan altos como las torres gemelas en Nueva York, sin el desarrollo de un sistema eficiente de transportación vertical.

Debido a las distintas clases de servicio que estos mecanismos pueden prestar y los lugares donde están instalados, existen distintas configuraciones de modo que una gran cantidad de parámetros internos como de condiciones de operación pueden variar, lo que no debería reflejarse en un desmejoramiento en la calidad del servicio.

Es por eso que además de las innovaciones del tipo arquitectónico y mecánico en estos mecanismos, se hace necesario realizar también innovaciones en su sistema de control, pues corresponde a gran parte del costo de un elevador.

Dentro de las muchas estrategias modernas de control de sistemas utilizadas en la actualidad, talvez una de las que más éxito está demostrando es el control por lógica difusa, ya que para su diseño no es necesario la obtención de modelos matemáticos de gran precisión, sino más bien un conocimiento heurístico de su dinámica, lo que implícitamente reduce costos.

A pesar de que el control difuso ha demostrado ser la solución a muchos problemas (no todos), existe aún mucho escepticismo en su uso, especialmente en aplicaciones en donde vidas humanas corren riesgo, debido a que no se ha podido desarrollar una teoría completa que garantice su desempeño.

I.2 JUSTIFICACION

Los elevadores son mecanismo muy complejos, que poseen parámetros variantes, perturbaciones externas debido a cambios de carga, efectos no – lineales por la presencia de fuerzas de rozamiento, efectos de saturación, y en los mecanismos de transmisión (engranajes y reductores mecánicos), lo que lo convierte en un problema muy apropiado para el estudio del desempeño de controles.

El análisis de este tipo de mecanismos demanda de un conocimiento integrado de varias herramientas, las cuales presentan limitaciones cuando la complejidad del sistema se incrementa, aún para controles convencionales.

Adicionalmente, hasta donde se ha podido consultar, no existe información a nivel nacional ni internacional, respecto a su modelo. Es entonces, que desde este punto de vista, es necesario y científicamente interesante el obtener el modelo.

Por otro lado, a pesar de que la teoría de la lógica difusa está totalmente sustentada, la teoría del control difuso es todavía muy limitada, compleja, y aplicable únicamente a un número muy restringido de casos particulares y conlleva conceptos tan básicos como los de la lógica matemática hasta conceptos muy avanzados como los de estabilidad de sistemas no – lineales. Por las razones anteriores, se conoce que siempre la simulación es una alternativa. Por lo último, es necesario la implementación de un programa de simulación que permita realizar rápidas modificaciones y de la manera más fácil posible.

Con base a los anterior, en el capítulo II se presenta un compendio de la teoría matemática de la lógica difusa incluyendo algunos resultados y sugerencias útiles para el diseño de controles difusos, que ayudan a entender la gran cantidad de alternativas existentes para su diseño.

Luego, en la primera parte del capítulo III, se realiza una introducción de las características básicas de los elevadores: como funcionan, cuales son sus partes constitutivas, y sus configuraciones más comunes. Se prosigue inmediatamente con una breve información sobre los fenómenos de rozamiento, y los modelos más comunes utilizados para su representación. La segunda parte del capítulo abarca la modelación del elevador, que sirve para realizar su análisis en la última parte del mismo, y cuyos resultados son de útiles para una comprensión detallada de su funcionamiento y de como sus distintas características afectan en su desempeño.

El capítulo IV comprende el diseño, control y simulación del elevador, con técnicas de control difuso y convencionales, con fines de comparación. Los controles convencionales ensayados son el PID y realimentación de estado con control integral, donde se han utilizado criterios de control conocidos para su diseño y en la medida de lo posible. Para el control difuso existen en realidad algunas alternativas, razón por la cual se proponen tres esquemas de control, PD, PI y PID difusos. Por último, se explica el programa desarrollado para la simulación y animación del elevador.

El capítulo V se refiere a los resultados y conclusiones más importantes relativas al trabajo desarrollado.

En los anexos, se incluyen el manual del usuario del programa, su listado y los flujogramas de las rutinas más importantes.

CAPITULO II

CONTROL DIFUSO

II.1 LINEAMIENTOS GENERALES

Conforme la tecnología avanza, los sistemas han incrementado su complejidad y tamaño. El diseño de controladores, empleando técnicas clásicas y modernas, se convierte en un proceso extremadamente difícil y laborioso para sistemas complejos. En la gran mayoría de casos es necesario un conocimiento preciso de la planta, lo cual es costoso y a menudo imposible de conseguir. Existe sin embargo, el conocimiento heurístico en la experiencia de cada persona que trabaje con tales plantas, que lo llevan a tener un dominio sobre la manera en la que se deben operar. Este conocimiento, que no puede ser cuantificado directamente, es la base de lo que se conoce como el ***Control Difuso***.

El Control Difuso se cimienta en la generalización de la teoría convencional de conjuntos introducida por Zadeh en 1965, como una manera matemática de representar la ***ambigüedad en la cotideanidad*** [23]. Asimilar la idea básica de lo que representa un conjunto difuso es bastante sencillo. Por ejemplo en el caso de un conductor de automóvil: si este advirtiera que la ***velocidad*** a la que está viajando es ***muy rápida***, entonces

presionaría con *mucha fuerza* el *freno*. Pero mas aún, si para otro conductor la *velocidad* era *medianamente rápida* y simplemente, a su criterio, se debe presionar el *freno* con *mediana fuerza*. Esto nos conduce a preguntar cuál de los dos conductores tiene la razón? En realidad, la razón la pueden tener los dos, debido a la inexactitud de los términos empleados, y por lo tanto sus sugerencias sobre cuanto presionar el freno serían también correctas.

El control difuso ha brindado pruebas de ser una muy buena alternativa cuando los procesos son muy complejos para un análisis por medio de técnicas convencionales, o cuando las fuentes de información proveen datos de manera cualitativa, inexacta o incierta, a pesar de que en la actualidad no existe un procedimiento sistemático de diseño de tales controladores. [3]

El Control Difuso emplea la ***Lógica Difusa***, la cual brinda una base teórica estructurada para el análisis de procesos lógicos generalizados. En la lógica tradicional una proposición podría ser únicamente verdadera (1) ó falsa (0). En la lógica difusa, una proposición podría ser tanto verdadera como falsa. La lógica difusa está más cerca del razonamiento humano y del lenguaje natural en comparación con la lógica tradicional, lo cual brinda una mayor aproximación a la naturaleza inexacta del mundo real [3]. Los sistemas difusos han demostrado también ser una manera fácil de representar sistemas de datos continuos de manera digital [6], permitiendo a los diseñadores construir controladores aún cuando su conocimiento del comportamiento matemático del sistema sea limitado. El empleo de esta técnica ha logrado también reducir costos debido a que no requiere de sensores de alta

precisión, presenta mayor robustez al ruido ambiental, y su implementación se la puede realizar con microcontroladores de hasta 4 bits u 8 bits. [4]

En esencia, los sistemas de lógica difusa, proveen de un algoritmo para convertir el conocimiento del control expresado lingüísticamente en un control automático.

II.2 LÓGICA DIFUSA

Fue Lofti Zadeh quien en 1965 comenzó a escribir una serie de artículos sobre conjuntos difusos y análisis de sistemas desde un punto de vista lingüístico. Esto fundó la base para que Mamdani y sus colegas empezarán a realizar las primeras investigaciones sobre control difuso en 1974. [3]

Pero en general, la lógica difusa ha encontrado grandes campos de aplicación no solamente en control, sino también en distintas ciencias tales como:

- Identificación y estimación de sistemas
- Reconocimiento y procesamiento de patrones
- Economía

En la actualidad gran parte de las investigaciones en lo referente al control difuso se están desarrollando en las áreas relacionadas con el control difuso adaptivo, y sistemas neuro-difusos.

II.2.1 CONJUNTOS DIFUSOS Y FUNCIONES DE MEMBRESÍA

En la teoría convencional de conjuntos, un conjunto quedará definido enumerando todos sus elementos (definición por extensión, $x \in A$) o por medio de las propiedades que estos cumplan (definición por comprensión, $A = \{x / x \text{ cumple cierta propiedad}\}$). Otra alternativa para definir un conjunto es la de introducir una ***función de membresía*** también llamada función característica o discriminante, del tipo cero-uno. Por ejemplo para un conjunto arbitrario A, se emplea la función de membresía denotada como $\mu_A(x)$, definida de la siguiente manera:

$$A \Rightarrow \quad \mu_A(x)$$

$$x \mapsto \mu_A(x) = \begin{cases} 1 \text{ si } x \in A \\ 0 \text{ si } x \notin A \end{cases} \qquad \text{[ecuación 2.1]}$$

Se dice entonces que el conjunto A es matemáticamente equivalente a su función de membresía, en el sentido de que conociendo $\mu_A(x)$, es lo mismo que conocer al conjunto A. Un conjunto difuso **F** definido en un universo **U**, estará caracterizado entonces por una función de membresía $\mu_F(x)$ que toma valores en el intervalo [0,1]. Es decir, éste es una generalización de un conjunto convencional (cuya función de membresía puede tomar únicamente los valores 0 y 1).

A partir de esta definición, se puede observar que un elemento x podría pertenecer a más de un conjunto difuso, por lo que se dice que la función de membresía provee de una medida del grado de pertenencia de un elemento del universo U en un conjunto difuso. Un conjunto difuso F en U, puede ser representado por una colección de pares ordenados de un elemento genérico x y su grado de pertenencia: $F = \{(x, \mu_F(x)) / x \in U\}$.

Cuando U es continuo, se representa a F generalmente como $F = \int_U \mu_F(x)/x$, y similarmente cuando U es discreto como: $F = \sum_U \mu_F(x)/x$. En estas ecuaciones, tanto el signo integral como la sumatoria no denotan sus operaciones respectivas sino que representan una colección de todos los puntos $x \varepsilon U$ con funciones de membresía asociadas $\mu_F(x)>0$.

Las funciones de membresía más comunes empleadas en control son las trapezoidales, triangulares, gaussianas, sigmoidales y singleton, que a continuación se definen:

Triangular: En la tabla 2.1, se muestra la caracterización matemática de la función triangular, donde se puede variar tanto su centro (c) como el ancho de su base (w). Además existen dos variaciones de la función triangular definidas como saturada izquierda(u^L) y saturada derecha (u^R). Para estas últimas c^L y c^R, representan los puntos de saturación, así mismo w^L y w^R representan la base del triángulo completo. La figura 2.1 muestra las gráficas de 5 funciones de membresía triangulares con distintos centros y anchos de base.

Funciones de membresía triangulares	
Saturada izquierda	$u^L(u) = \begin{cases} 1 & \text{si } u \le c^L \\ \max\left\{0,\ 1 + \dfrac{c^L - u}{0.5w^L}\right\} & \text{cualquier otro} \end{cases}$
Normal	$u^C(u) = \begin{cases} \max\left\{0,\ 1 + \dfrac{u - c}{0.5w}\right\} & \text{si } u \le c \\ \max\left\{0,\ 1 + \dfrac{c - u}{0.5w}\right\} & \text{cualquier otro} \end{cases}$
Saturada derecha	$u^L(u) = \begin{cases} \max\left\{0,\ 1 + \dfrac{u - c^R}{0.5w^R}\right\} & \text{si } u \le c^R \\ 1 & \text{cualquier otro} \end{cases}$

Tabla 2.1 Caracterización matemática de Funciones de Membresía Triangulares

Tabla 2.2 Caracterización matemática de Funciones de Membresía Gaussianas

$$g(x) = \begin{cases} 1, & x = x \\ 0, & \text{cualquier otro} \end{cases}$$

El crear una función de membresía adecuada, era hasta hace algunos años un proceso

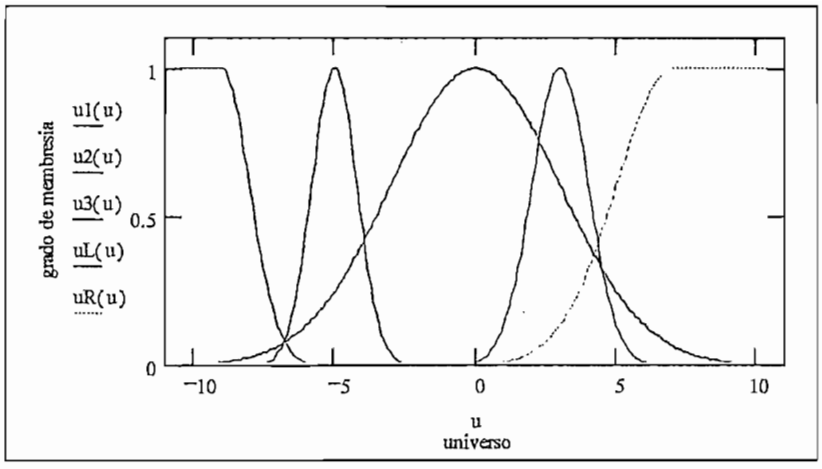


Figura 2.2 Funciones de Membresía Gaussianas para distintos valores de σ y c.

Singleton: La función singleton o impulso, es talvez una de las más empleadas en la lógica difusa, para procesos de fusificación como se verá más adelante. Su caracterización matemática es la siguiente.

$$g(x) = \begin{cases} 1, & x = \bar{x} \\ 0, & \text{cualquier otro} \end{cases}$$

El escoger una función de membresía adecuada, era hasta hace algunos años un proceso arbitrario que se basaba en la práctica y la experiencia. Sin embargo, recientemente, existen ya criterios técnicos para escoger la función de membresía basados en procedimientos de optimización.[16]

Es importante diferenciar, en especial con la función gaussiana, que las funciones de membresía no representan ni están relacionadas de manera alguna con funciones de

densidad de probabilidad, y no existe relación alguna entre la teoría de la lógica difusa con la teoría de probabilidades.

Algunas recomendaciones para la disposición de las funciones de membresía son:

- El número de funciones de membresía debe variar entre 5 y 9 para cada variable lingüística y siempre en números impares.
- El sobrelapamiento debe estar comprendido entre el 10% y el 50% del dominio de cada función.
- La suma de los puntos verticales, en el intervalo de sobrelapamiento, debe ser menor o igual a 1.
- Es conveniente que exista una mayor densidad de funciones de membresía en las cercanías del punto de control óptimo y a medida que se alejen de este, la densidad disminuya.[4]

II.2.2 VARIABLES LINGÜÍSTICAS

Una variable lingüística es aquella que describe semánticamente, en un lenguaje natural o artificial, alguna variable real. Es decir, una variable lingüística es aquella que toma valores definidos en términos lingüísticos. Por ejemplo, la variable **Temperatura** podría tomar los siguientes valores: {**Muy Frío, Frío, Normal, Caliente, Muy Caliente**}. Una definición formal es la siguiente:

Si una variable emplea palabras como sus valores, se dice que es una variable lingüística.

Una variable lingüística está caracterizada por el cuarteto: {x,T(x),U,S}

donde:

x es el nombre de la variable (ej. Temperatura)

T(x) son las etiquetas de la variable x (ej. ...frío, normal, caliente...)

U es el conjunto universo de la variable física (ej. rango de temperatura de [0,100] °C)

S regla semántica que asigna un conjunto difuso en U a cada valor lingüístico T(x).

Por ejemplo, se asigna a las etiquetas frío, normal y caliente a las funciones de membresía $\mu_{frío}(x)$, $\mu_{normal}(x)$, y $\mu_{caliente}(x)$ respectivamente[1].

En la figura 2.3, se muestra un ejemplo de asignación de etiquetas a distintas funciones de membresía.

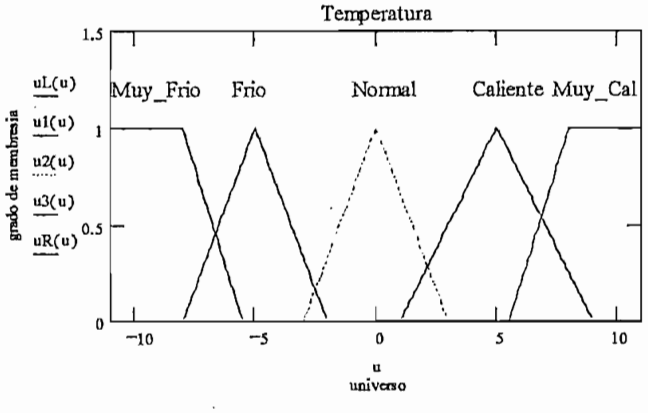


Figura 2.3 Asignación de valores lingüísticos a funciones de membresía.

Como se observa, las funciones de membresía de los distintos conjuntos difusos se encuentran sobrelapadas. Esta característica es necesaria para que exista ambigüedad en la

[1] Hablar de funciones de membresía es matemáticamente equivalente a hablar de conjuntos difusos, ya que una función de membresía caracteriza a un conjunto difuso.

definición lingüística de una variable y como se verá más adelante, generará una superficie suave y estable [16].

Es importante también señalar, que cuando las funciones de membresía pertenecen a una variable de entrada, estas podrán ser del tipo saturadas, más si caracterizan a una variable de salida, no lo podrán ser, ya que como se verá en la sección 2.3.3 el universo de salida debe ser finito.

II.2.3 OPERACIONES FUNDAMENTALES DE LA LÓGICA DIFUSA

La lógica difusa es una generalización de la lógica tradicional. Se presenta a continuación una discusión sobre los resultados fundamentales de la lógica tradicional y a no ser que se indique expresamente, estos resultados son válidos para la lógica difusa.

La lógica matemática emplea proposiciones. Una proposición es un enunciado del que se puede obtener su valor de verdad, por ejemplo:

"*Hoy es miércoles* "

Existen además proposiciones compuestas, que se forman de la combinación de proposiciones simples:

"*Hoy es miércoles y hace mucho frío*"

Las proposiciones compuestas además de estar conformadas de 2 o más proposiciones simples, emplean conectivos lógicos que unen a las distintas proposiciones. Estos

conectivos lógicos, modifican el valor de verdad de la proposición compuesta. De aquí en adelante los conectivos lógicos serán referidos como operaciones lógicas.

Entre las operaciones más importantes están la conjunción "y", negación "no", y la disyunción "o". La simbología normalizada empleada en la lógica y como se lee en el lenguaje natural es como se la muestra en la tabla 2.3, donde p y q son proposiciones.

Se escribe:	Se lee:
$p \wedge q$	"p y q"
$p \vee q$	"p o q"
$\sim p$	"no p"

Tabla 2.3 Operaciones Lógicas comunes

Existen sin embargo más operaciones lógicas y una de las más relevantes es la implicación. La implicación, establece que a partir de un *antecedente* se tiene un *consecuente*, como se muestra a continuación:

$p \rightarrow q$, y se lee como "p implica q" o "Si p entonces q"

Cada proposición posee un valor de verdad, y las proposiciones pueden ser combinadas a través de conectivos lógicos obteniendo una proposición compuesta con un nuevo valor de verdad. En la tabla 2.4 se muestra como los valores de verdad de las proposiciones compuestas se obtienen para los distintos conectivos lógicos. [20]

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	1

Tabla 2.4. Tablas de Verdad para distintas operaciones lógicas

La lógica proposicional brinda el sustento a la teoría de conjuntos. Las distintas operaciones entre conjuntos son definidas por medio de la lógica y además son isomorfas entre ellas, y con el álgebra Booleana. Esto significa que cualquier teorema o definición en cualquiera de las tres teorías (lógica proposicional, teoría de conjuntos, o álgebra boolena) tendrá su equivalente en las otras dos. Las operaciones fundamentales entre conjuntos son la unión, intersección, y el complemento y son equivalentes a la disyunción, conjunción y a la negación respectivamente.

Como se mostró anteriormente, un conjunto convencional o difuso, puede ser definido por su función de membresía y además el resultado de cualquier operación entre conjuntos (unión, intersección o complemento) es otro conjunto, por lo que este nuevo conjunto también podrá ser caracterizado por una nueva función de membresía. Las siguientes fórmulas generan la función de membresía para los conjuntos resultantes de las operaciones antes expuestas.

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} \qquad \text{[ecuación 2.2]}$$

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \qquad \text{[ecuación 2.3]}$$

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \qquad \text{[ecuación 2.4]}$$

Otra definición de la intersección muy utilizada en control es la intersección por producto definida como:

$$\mu_{A\cap B}(x) = \mu_A(x) \cdot \mu_B(x) \qquad \text{[ecuación 2.5]}$$

Es importante señalar que estas maneras de definir las operaciones a través de funciones de membresía no son únicas; sin embargo, son de gran utilidad para aplicaciones de control. En general a los operadores definidos para la intersección se los conoce como normas-t y para la unión como conormas-t o normas-s. Otras formas de caracterizar a estas operaciones a través de estas operaciones se encuentran en [3, 9 y 16].

Es fácil verificar la validez de estas fórmulas con conjuntos convencionales, así por ejemplo para la unión, $\mu_{A\cup B}(x)=1$ si $x \varepsilon A$ o $x \varepsilon B$; similarmente para la intersección: $\mu_{A\cap B}(x)=1$ si $x \varepsilon A$ y $x \varepsilon B$; para el complemento $\mu_{\bar{A}}(x)=1$ si $x \notin A$, pues como se sabe en el caso de conjuntos convencionales las funciones de membresía toman valores de 0 ó 1. Con conjuntos difusos las funciones de membresía toman valores en el rango [0,1] [16]. El siguiente ejemplo ilustra el empleo de estas definiciones con conjuntos difusos:
Supónganse las siguientes condiciones para el elemento genérico x^*.

$\mu_A(x^*)=0.6$

$\mu_B(x^*)=0.7$

Entonces, se presentan los siguientes ejemplos de funciones de membresía para las tres operaciones de conjuntos antes definidas:

$$\mu_{A\cap B}(x^*)=\min\{\mu_A(x^*),\mu_B(x^*)\}=\min\{0.6,0.7\}=0.6$$

o

$$\mu_{A\cap B}(x^*)=\mu_A(x^*)\cdot\mu_B(x^*)=0.6\cdot 0.7=0.42$$

$$\mu_{A\cup B}(x^*) = \max\{\mu_A(x^*), \mu_B(x^*)\} = \max\{0.6, 0.7\} = 0.7$$

$$\mu_{\bar{A}}(x^*) = 1 - \mu_A(x^*) = 1 - 0.6 = 0.4$$

Otra operación importante en la teoría de conjuntos es el producto cartesiano que se define como:

$$A_1 \times \ldots \times A_n = \{(x_1, \ldots, x_n) / x_i \in A_i\} \qquad \text{[ecuación 2.6]}$$

Donde A_i son conjuntos contenidos en los universos U_i respectivamente, y su producto cartesiano está contenido en el espacio $U_1 \times \ldots \times U_n$. Esta operación genera un vector ordenado, donde cada elemento $x_i \varepsilon A_i$. Además, como se observa, los A_i no son necesariamente subconjuntos del mismo universo U. Utilizando funciones de membresía se representa como:

$$\mu_{A_1 x \ldots x A_n}(x_1, \ldots, x_n) = \min\{\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)\}$$

o [ecuación 2.7] y [ecuación 2.8]

$$\mu_{A_1 x \ldots x A_n}(x_1, \ldots, x_n) = \mu_{A_1}(x_1) \cdot \ldots \cdot \mu_{A_n}(x_n)$$

En general un producto cartesiano es una operación de relación, que se define de manera más general como sigue:

$$R(U_1, \ldots, U_n) = \{((u_1, \ldots, u_n), \mu_R(u_1, \ldots, u_n)) \mid (u_1, \ldots, u_n) \in U_1 \times \ldots \times U_n\} \qquad \text{[ecuación 2.9]}$$

Las operaciones que se llevan a cabo entre relaciones son conocidas como operaciones de composición. Una definición formal de una operación de composición entre relaciones de conjuntos difusos es:

Sean R y S dos relaciones en UxV y VxW respectivamente. La composición de R(U,V) y S(V,W) es una relación difusa denotada como $R \circ S$, y con función de membresía:

$$\mu_{R \circ S}(x,z) = \sup_{y \in V}[\mu_R(x,y) * \mu_S(y,z)] \qquad \text{[ecuación 2.10]}$$

Donde * simboliza una norma-t (mínimo o producto). En conjuntos difusos esta operación de composición recibe el nombre de composición supremo-asterisco. Una particularización de esta operación empleada ya para procesos de inferencia de control difuso es el caso en que la relación R es un solo conjunto difuso, de tal manera que $\mu_R(x,y)$ se transforma en $\mu_R(x)$. Esto significa que V=U y la función de membresía resultante es tan solo función de z, y la ecuación 2.10 se escribe como:

$$\mu_{R \circ S}(z) = \sup_{x \in U}[\mu_R(x) * \mu_S(x,z)] \qquad \text{[ecuación 2.11]}$$

Queda sin embargo, por definir el equivalente de la implicación lógica en la teoría de conjuntos. La implicación lógica tiene gran trascendencia en el control difuso ya que es por medio de ésta que se codificará el conocimiento heurístico empleado para el control.
Por ejemplo en la siguiente implicación:

$$p \rightarrow q$$

Al ser la lógica matemática una herramienta para describir procesos de razonamiento, ésta tendrá que poseer ciertos mecanismos para poder llevar a cabo tal tarea. Un proceso de razonamiento o inferencia consiste en poder concluir algún juicio en base a un grupo de premisas. En general se conoce al conjunto de premisas como *antecedentes* y a la conclusión como *consecuente*.

particularidades que ya no se cumplen en la misma, como por ejemplo las dos leyes fundamentales de la teoría convencional de conjuntos:

1) Ley de Contradicción: $A \cup \overline{A} = U$

2) Ley de Exclusión: $A \cap \overline{A} = \phi$

II.2.4 REGLAS DE CONOCIMIENTO Y PROCESOS DE INFERENCIA

Al ser la lógica matemática una herramienta para describir procesos de razonamiento, ésta tendrá que poseer ciertos mecanismos para poder llevar a cabo tal tarea. Un proceso de razonamiento o inferencia consiste en poder concluir algún juicio en base a un grupo de premisas. En general se conoce al conjunto de premisas como ***antecedentes*** y a la conclusión como ***consecuente***.

Las premisas son proposiciones simples o compuestas que para el caso del control difuso, pueden estar expresadas en la forma:

SI <*variable lingüística*> **es** <*etiqueta*> **ENTONCES** <*variable lingüística*> **es** <*etiqueta*>

Es decir, en la forma de una implicación lógica "***Si*** *p* ***Entonces*** *q*". Este tipo de sentencias, en Control Difuso, son conocidas como reglas de conocimiento, y su función es la de codificar el conocimiento de las acciones de control.

Para poder llevar a cabo un proceso de razonamiento, la lógica provee de ciertos mecanismos conocidos como reglas de inferencia. Para el caso de aplicaciones de control

difuso la regla de inferencia más importante es la del *modus ponens generalizado*, que utiliza la implicación (regla de conocimiento). El modus ponens en la lógica tradicional realiza el siguiente razonamiento:

premisa 1: (x es A)
premisa 2: (x es A) $\rightarrow$ (y es B)

---

conclusión: (y es B)

El modus ponens generalizado en cambio establece que:

premisa 1: (x es A^*)
premisa 2: (x es A) $\rightarrow$ (y es B)

---

conclusión: (y es B^*)

Donde A, A^*, B y B^*, son conjuntos difusos denotados por sus respectivas etiquetas.

La diferencia entre el modus ponens tradicional y el generalizado, es que en el tradicional la premisa 1 debe ser idéntica al antecedente de la premisa 2 y la conclusión es el consecuente de la premisa 2; mientras que en el generalizado, la premisa 1 no es la misma que el antecedente de la premisa 2, y la conclusión tampoco es igual al consecuente de la premisa 2; sin embargo A y A^* son similares al igual que B y B^*.

Es decir que en el modus ponens generalizado, la regla de conocimiento (premisa 2) será válida (activa) mientras exista un grado de similaridad mayor a cero entre la premisa 1 y el antecedente de la regla, y el resultado tendrá un grado de similaridad mayor a cero con el consecuente de la regla. La regla de inferencia de la implicación difusa está basada en las

reglas de inferencia composicionales para razonamiento aproximado sugerida por Zadeh en 1973.

Otra manera de expresar el modus ponens es a través de la siguiente proposición lógica:

$$(x \text{ es } A^*) \wedge (x \text{ es } A) \rightarrow (y \text{ es } B) \qquad \text{[ecuación 2.14]}$$

Esta es una operación de composición ya que la extensión de ambas proposiciones a la teoría de conjuntos, resultan ser relaciones. Al ser conjuntos difusos, la composición a emplear es la de supremo-asterisco donde una de la relaciones es un solo conjunto difuso, entonces se cumple la ecuación 2.11:

$$\mu_{B^*}(y) = \sup_{x \in A^*} [\mu_A(x) * \mu_{A \rightarrow B}(x, y)] \qquad \text{[ecuación 2.15]}$$

Se concluye de esta manera, que el modus ponens generalizado es una operación de composición. Para mostrar como se aplica la ecuación 2.15 y con fines de encontrar una expresión más sencilla, considérese que $\mu_{A^*}(x)=1$ únicamente cuando $x=x^*$, y es cero para cualquier otro x, que como se explicará más adelante es un fusificador singleton; y utilizando además el producto para la implicación y el mínimo para la composición, puede escribirse entonces:

$$\begin{aligned} \mu_{B^*}(y) &= \sup_{x \in A^*} [\mu_{A^*}(x) * \mu_{A \rightarrow B}(x, y)] \\ &= \sup_{x \in A^*} [\mu_{A^*}(x^*) * \mu_{A \rightarrow B}(x^*, y)] \\ &= \sup_{x \in A^*} [1 * \mu_{A \rightarrow B}(x^*, y)] \end{aligned}$$

$$= \sup_{x \in A^*} \left[\min\left\{1, \mu_A(x^*) \cdot \mu_B(y)\right\} \right]$$
$$= \sup_{x \in A^*} \left[\mu_A(x^*) \cdot \mu_B(y) \right] = \mu_A(x^*) \cdot \mu_B(y)$$

Independientemente de si se emplea el mínimo o el producto para la composición, el supremo se elimina en la medida de que se emplee un fusificador singleton. Se tiene entonces los siguientes resultados, dependiendo de la norma-t que se utilice:

$$\mu_{B'}(y) = \mu_A(x^*) \cdot \mu_B(y)$$
$$\mu_{B'}(y) = \min\left\{\mu_A(x^*), \mu_B(y)\right\}$$

[ecuación 2.16] y [ecuación 2.17]

II.3 SISTEMAS DE LÓGICA DIFUSA

El propósito de esta sección es el de explicar las partes constitutivas y como trabaja un sistema de lógica difusa. En las secciones anteriores se explicaron las herramientas necesarias de la lógica difusa que se aplicarán en el análisis de los sistemas difusos; sin embargo, hasta la actualidad no existe un procedimiento bien definido para el diseño de sistemas difusos, y lo único que se puede establecer son recomendaciones probadas que en la mayoría de los casos brindan un buen resultado en la operación en-línea de los sistemas difusos.

Matemáticamente, un sistema de lógica difusa (SLD) es un mapeo no-lineal de un vector de entrada en un escalar o un vector de salida. Para llevar a cabo tal mapeo, los sistemas .

difusos se constituyen de 4 procesos fundamentales: Fusificación, Mecanismo de Inferencia, Base de Conocimiento, y Defusificación, como se muestra en la figura 2.4.

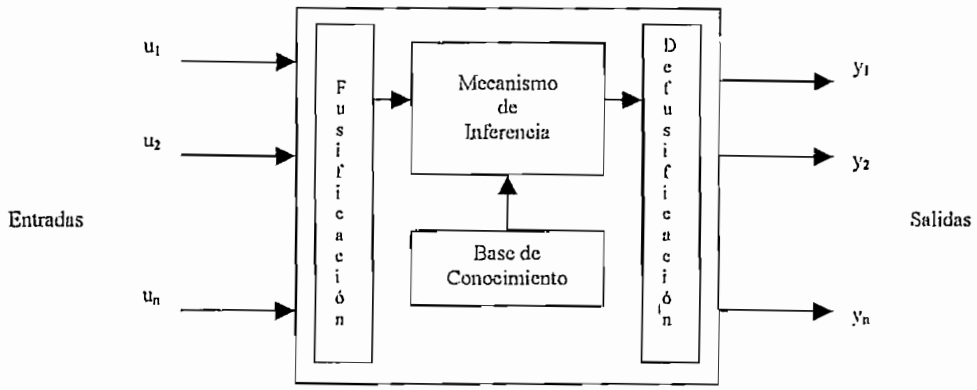


Figura 2.4 Sistema Difuso (Controlador)

II.3.1 FUSIFICACION

Los sistemas difusos, tales como el mostrado en la figura 2.4, trabajan con conjuntos difusos y no valores numéricos ($u_1,...,u_n$). El proceso que se encarga de convertir o de mapear los valores numéricos en conjuntos difusos es conocido como Fusificación.

Si U_i^* es el conjunto de todos los posibles conjuntos difusos definidos en U_i y dado $u_i \varepsilon U_i$, la fusificación mapeará a u_i a un conjunto difuso $A_i^{fuz} \subset U_i^*$.

Esta transformación se puede definir matemáticamente como una función de la siguiente manera:

$F: U_i \rightarrow U_i^*$

donde

$F(u_i) = A_i^{fuz}$

En aplicaciones de control, la fusificación singleton es muy empleada debido a que como se mostró, simplifica mucho el cálculo de la inferencia, eliminando el supremo de la operación de composición facilitando su implementación práctica.

Esta fusificación genera un conjunto difuso $A_i^{fuz} \subset U_i^*$ con una función de membresía dada por:

$$\mu_{A_i^{fuz}}(x) = \begin{cases} 1, & x = u_i \\ 0, & \text{cualquier otro} \end{cases} \qquad \text{[ecuación 2.18]}$$

Existen otros tipos de fusificación, así por ejemplo la Gaussiana y la Triangular. Estas se han caracterizado por una mayor inmunidad al ruido[16], pero su uso, debido a su complejidad matemática, no se ha llegado a justificar del todo en la práctica [19], si se desea una discusión más amplia sobre el empleo de otro tipo de fusificadores se recomienda consultar en [16].

Ya para una implementación práctica el proceso de fusificación singleton se reduce a evaluar cada función de membresía de la variable lingüística en el valor numérico de la variable de entrada correspondiente.

Y se puede simplificar aún más evaluando únicamente las funciones de membresía a cuyo dominio pertenece el valor numérico. Si las funciones de membresía no están sobrelapadas, a cada lado, en un porcentaje mayor al 50%, se basta con evaluar un máximo de dos funciones de membresía por variable lingüística.

Un ejemplo gráfico de fusificación singleton se muestra a continuación:

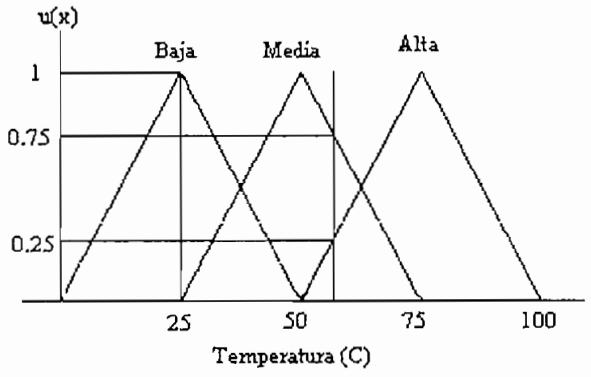


Figura 2.5 Fusificación singleton

En este ejemplo, se están fusificando 2 valores numéricos de la variable Temperatura. El primero (rojo), corresponde a una temperatura de 25°C y su correspondiente grado de membresía en el conjunto difuso etiquetado por Baja es de 1. El segundo valor (azul), que corresponde a una temperatura de 60°C, posee dos grados de membresía en dos distintos conjuntos difusos etiquetados por Media y Alta, de 0.25 y 0.75 respectivamente.

II.3.2 BASE DE CONOCIMIENTO Y MECANISMO DE INFERENCIA

La colección de reglas de conocimiento se conoce como la Base de Conocimiento, y es el corazón del sistema difuso, ya que contiene todas las posibles acciones de control que realizará el sistema de lógica difusa para controlar la planta.

Es común que las premisas sean proposiciones compuestas, dependiendo del número de entradas, entonces el antecendente podrá tener más de una variable lingüística, así por ejemplo:

SI *<variable lingüistica(1)>* **es** *<etiqueta>* **Y** *<variable lingüistica(2)>* **es** *<etiqueta>* **Y...Y** *<variable lingüistica(n)>* **es** *<etiqueta>* **ENTONCES** *<variable lingüistica(S)>* **es** *<etiqueta>*

El conectivo "Y" (conjunción) empleado en el ejemplo, puede ser también un conectivo "O"; sin embargo, no suele presentarse el caso de su uso en la premisa. La premisa puede ser cuantificada empleando cualquiera de las normas-t o conormas-t dependiendo del conectivo. Así mismo, el consecuente también podrá tener más de una variable lingüística de salida; sin embargo por la propiedad de distribución se cumple que:

$$p \rightarrow (q * r) \equiv (p \rightarrow q) * (p \rightarrow r)$$

Donde el asterisco representa una norma-t o una conorma-t. Lo que es equivalente a tener 2 reglas de conocimiento. Es decir que, desde el punto de vista de control, esto representa que un sistema MIMO puede ser analizado como dos o más sistemas MISO.

En general, no todas las reglas de conocimiento deben incluir en su antecedente a todas las variables lingüísticas de entrada.

Si n representa el número de variables lingüísticas de entrada y N_i es el número de etiquetas (funciones de membresía) por variable lingüística, al hacer todas las posible combinaciones encontramos que el número total de reglas será:

$$\#\,\text{posible de reglas} = R = \prod_{i=1}^{n} N_i \qquad \text{[ecuación 2.19]}$$

Existen dos propiedades fundamentales de las bases de conocimiento que son: completez y consistencia. Se dice que una base de conocimiento es "completa" si existe una conclusión para cada posible combinación de entrada y es "consistente" si no existe contradicción entre las conclusiones para distintas combinaciones de las entradas. Las reglas de conocimiento pueden ser organizadas en una matriz o varias matrices, dependiendo de número de entradas, y son conocidas como "Memoria Asociativa Difusa" (FAM). Para mostrar uno de los métodos para generar una base de conocimiento se presenta el siguiente ejemplo[2]: Supóngase el problema de control del péndulo invertido, como el que se muestra en la figura 2.6.

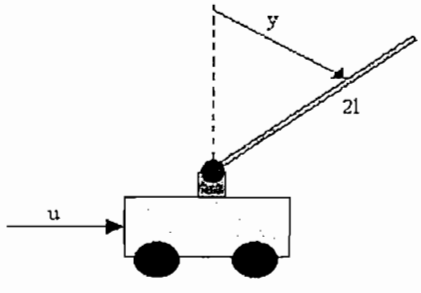


Figura 2.6 Sistema de péndulo invertido

[2] Ejemplo tomado de [19].

Se pretende balancear el péndulo por medio del movimiento del móvil. En la figura 2.6 se denota por "y" a la posición angular del péndulo con respecto a la posición deseada, y "u" denota la fuerza aplicada al móvil para lograr equilibrar el péndulo.

Si a una persona se le encargara el control del péndulo, este debería basar sus decisiones en observaciones del comportamiento del sistema, como se muestra en la figura 2.7.

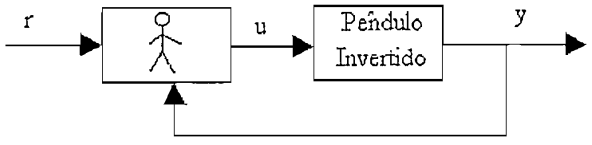


Figura 2.7 Control humano de un péndulo invertido

La persona podría escoger como variables observadas al "error" $(e(t)=r(t)-y(t))$, que posee el péndulo con respecto a la posición deseada (r) y además, al "cambio del error" $\left(\frac{d}{dt}e(t)\right)$.

Alternativamente, se puede remplazar a esta persona con un controlador que lleve a cabo las mismas acciones de control, basadas en el conocimiento que tiene el individuo sobre el comportamiento deseado del péndulo. Este último sería el mecanismo de control que emplea un controlador difuso, como se muestra en la figura 2.8.

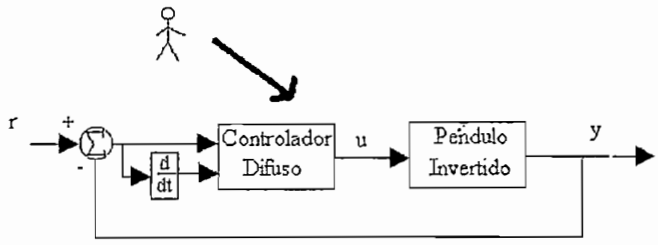


Figura 2.8 Controlador Difuso de un péndulo invertido.

Entonces, el controlador podrá emplear, al igual que el individuo, las dos variables "error" y "cambio del error" (variables lingüísticas) para realizar las acciones de control. Asumiendo que el individuo a definido cinco etiquetas (funciones de membresía) para cada variable lingüística de entrada (error y cambio del error) y de salida (fuerza), se podría expresar su conocimiento en una memoria asociativa (FAM) como la de la tabla 2.5.

"fuerza" u		"Cambio en el error" $\left(\dot{e}\right)$				
		neg_grande	neg_pequeño	Cero	pos_pequeño	pos_grande
"error" e	neg_grande	pos_grande	pos_grande	pos_grande	pos_pequeño	cero
	neg_pequeño	pos_grande	pos_grande	pos_pequeño	cero	neg_pequeño
	Cero	pos_grande	pos_pequeño	Cero	neg_pequeño	neg_grande
	pos_pequeño	pos_pequeño	cero	neg_pequeño	neg_grande	neg_grande
	pos_grande	cero	neg_pequeño	neg_grande	neg_grande	neg_grande

Tabla 2.5 Memoria asociativa (FAM) para un sistema de control de un péndulo invertido.

Debido a que el error se obtiene mediante e = r-y, y como se pretende balancear al péndulo en la parte superior, entonces r = 0 y el "error" será e = -y, y el "cambio en el error": $\dot{e} = -\dot{y}$. Por lo anterior, se interpreta que un "error" "pos_grande" (positivo grande) indica que el péndulo se encuentra alejado a la izquierda de la posición deseada, mientras que un

"error" "neg_pequeño" (negativo pequeño) se interpretaría como que el péndulo se encuentra no muy lejos de la posición deseada y a la derecha de esta. De manera similar, el "cambio en el error" será positivo (pos_grande, pos_pequeño) cuando el péndulo esté viajando en el sentido contrario a las manecillas del reloj, y negativo (neg_grande, neg_pequeño) cuando el péndulo viaje en el sentido de las manecillas del reloj.

Una manera muy útil de representar las etiquetas para implementaciones prácticas, es la de reemplazar las palabras por números como se muestra en la tabla 2.6.

	neg_grande	neg_pequeño	cero	pos_pequeño	pos_grande
Alternativa 1	-2	-1	0	1	2
Alternativa 2	0	1	2	3	4

Tabla 2.6 Equivalencias de las etiquetas útiles para implementaciones prácticas

Cuando se tiene más de dos variables de entrada, existirá una memoria asociativa relacionando dos de las variables lingüísticas por cada combinación de las etiquetas de las demás. En general las memorias asociativas son de gran utilidad hasta para 3 variables, para un mayor número de variables se emplea Matrices Programables de Lógica Difusa (PFLA). Estas representaciones de la base de conocimiento son mapeos multidimesionales, que permiten visualizar varias entradas y salidas simultáneamente. Para mayor información en este tipo de estructuras matriciales se puede consultar en [6].

Como se explicó en la sección II.3.1, cada valor continuo de la variable de entrada puede ser mapeado en un máximo de dos conjuntos difusos. Si existiesen dos variables de entrada, entonces el máximo número de reglas de conocimiento activas serían cuatro; sin embargo,

se puede generalizar al caso multidimensional en el sentido de que en cada memoria asociativa no existirán más de cuatro reglas activas al mismo tiempo.

Al tener más de una regla de conocimiento activa, cada una aportará con una conclusión no necesariamente diferente. Debido a la naturaleza ambigua de la lógica difusa, todas estas conclusiones son válidas, y deben ser tomadas en cuenta. El mecanismo que obtiene las conclusiones de cada regla activa se conoce como mecanismo de inferencia.

El mecanismo de inferencia calcula mediante la regla composicional de inferencia (ecuación 15 o 16) la conclusión de cada regla activa.

Para ilustrar el mecanismo de inferencia, supóngase el siguiente ejemplo empleando la memoria asociativa de la tabla 2.5 y que se han obtenido los siguientes valores de las funciones de membresía de los conjuntos difusos de entrada luego de un proceso de fusificación singleton para las dos variables x1 ≡ "error" y x2 ≡ "cambio del error":

$\mu_{pos\_grande}(x1^*) = 0.7$ y $\mu_{pos\_pequeno}(x1^*) = 0.3$

$\mu_{Cero}(x2^*) = 0.5$ y $\mu_{neg\_pequeno}(x2^*) = 0.5$

Observando en la memoria asociativa, se nota que están activas 4 reglas de conocimiento, que son:

1) **Si** *error* **es** *pos_grande* **y** *cambio de error* **es** *cero* **Entonces** *Fuerza* **es** *neg_grande*.
2) **Si** *error* **es** *pos_grande* **y** *cambio del error* **es** *neg_pequeño* **Entonces** *Fuerza* **es** *neg_pequeño*.
3) **Si** *error* **es** *pos_pequeño* **y** *cambio del error* **es** *cero* **Entonces** *Fuerza* **es** *neg_pequeño*.
4) **Si** *error* **es** *pos_pequeño* **y** *cambio del error* **es** *neg_pequeño* **Entonces** *Fuerza* **es** *cero*.

Para calcular los respectivos valores inferidos de cada regla se emplean el producto para la conjunción de las premisas, y para la implicación:

Para la regla 1:

$$\mu_{neg\_grande'}(y) = \mu_{pos\_grande}(x1^*) \cdot \mu_{cero}(x2^*) \cdot \mu_{neg\_grande}(y)$$
$$= (0.3) \cdot (0.5) \cdot \mu_{neg\_grande}(y)$$
$$= 0.15 \cdot \mu_{neg\_grande}(y)$$

Para la regla 2:

$$\mu_{neg\_pequeno'}(y) = \mu_{pos\_grande}(x1^*) \cdot \mu_{neg\_pequeno}(x2^*) \cdot \mu_{neg\_pequeno}(y)$$
$$= (0.3) \cdot (0.5) \cdot \mu_{neg\_pequeno}(y)$$
$$= 0.15 \cdot \mu_{neg\_pequeno}(y)$$

Para la regla 3:

$$\mu_{neg\_pequeno'}(y) = \mu_{pos\_pequeno}(x1^*) \cdot \mu_{cero}(x2^*) \cdot \mu_{neg\_pequeno}(y)$$
$$= (0.7) \cdot (0.5) \cdot \mu_{neg\_pequeno}(y)$$
$$= 0.35 \cdot \mu_{neg\_pequeno}(y)$$

Para la regla 4:

$$\mu_{cero'}(y) = \mu_{pos\_pequeno}(x1^*) \cdot \mu_{neg\_pequeno}(x2^*) \cdot \mu_{cero}(y)$$
$$= (0.7) \cdot (0.5) \cdot \mu_{cero}(y)$$
$$= 0.35 \cdot \mu_{cero}(y)$$

Es así que se obtendrán tres conjuntos difusos de salida inferidos. Una interpretación gráfica de los conjuntos inferidos se muestra en la figura 2.6.

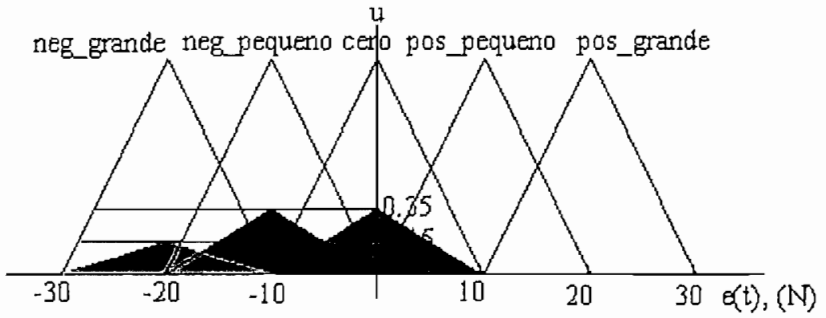


Figura 2.6. Conjuntos Difusos de Salida inferidos, empleando el producto para la implicación

Al emplear el producto para implicación, lo que se obtiene es un escalamiento vertical de las funciones de membresía de salida, como se observa en la figura 2.6. Si se hubiera empleado el mínimo para la implicación, se hubiesen obtenido las mismas funciones de membresía pero truncadas, como se muestra en la figura 2.7.

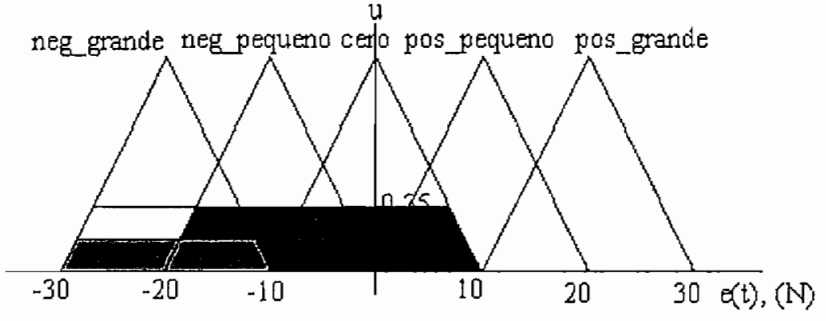


Figura 2.7. Conjuntos Difusos de Salida inferidos, empleando el mínimo para la implicación

Para obtener una sola respuesta numérica ya en el universo de salida, se utiliza el proceso de defusificación que se explicará en la siguiente sección.

II.3.3 MÉTODOS DE DEFUSIFICACIÓN

El mecanismo de inferencia produce un resultado en un conjunto difuso $\mu_{Bi}^{*}(y)$. El proceso de defusificación, mapeará este resultado en el universo de salida (u(t)), produciéndose así la salida de control de la planta. Este mapeo puede ser llevado a cabo mediante algunos métodos; sin embargo, existen dos cuyo uso se ha extendido en aplicaciones de control y de ingeniería, en general, debido a su simplicidad y fácil implementación [19]. Estas dos estrategias son: defusificación por centro de gravedad (COG) y la defusificación por Centro Promedio (COA). Para conocer sobre otras estrategias de defusificación se puede consultar en [9 y 16] .

Centro de Gravedad(COG): Este método calcula el centro de gravedad del área de las funciones de membresía inferidas, ya sean estas escaladas (implicación por producto) o truncadas (implicación por mínimo). La ecuación general del COG es la siguiente:

$$y = \frac{\sum_{i=1}^{R} c_i \int \mu_{Bi}^{*}(y) \cdot dy}{\sum_{i=1}^{R} \int \mu_{Bi}^{*}(y) \cdot dy} \qquad \text{[ecuación 2.20]}$$

Donde:

R	es el número total de reglas inferidas
c_i	es el valor para el cual la función de membresía alcanza su valor máximo
$\int \mu_{Bi}^{*}(y) \cdot dy$	es el área debajo de la función $\mu_{Bi}^{*}(y)$

Como condición necesaria para aplicar este método, se debe cumplir que $\sum_{i=1}^{R} \int \mu_{Bi}^{*}(y) \cdot dy \neq 0$. Esto se asegura en la medida de que exista una posible salida para cada posible entrada, esto demanda la existencia de al menos una regla activa para cada posible combinación de la entrada. Además, la $\int \mu_{Bi}^{*}(y) \cdot dy$ debe ser posible de calcular y esto se asegura mediante el uso de funciones de membresía de salida que posean un universo finito; es decir, no se deben emplear funciones de membresía saturadas.

Cuando las funciones de membresía son triangulares es muy sencillo calcular esta integral, llevando a cabo las siguientes simplificaciones:

- Si se usa el producto para la implicación, la integral se reduce al cálculo del área de un triángulo de base w, y altura $\mu_{Bi}^{*}(y)$, con la siguiente ecuación:

$$\int \mu_{Bi}^{*}(y) \cdot dy = (0.5) \cdot w \cdot \mu_{Bi}^{*}(y) \quad \text{[ecuación 2.21]}$$

- Si se usa el mínimo para la implicación, la integral corresponde al área de un triángulo truncado de base w y altura $h = \min\{\mu_A(x^*), \mu_B(y)\}$, cuya área se calcula por medio de:

$$\int \mu_{Bi}^{*}(y) \cdot dy = w \cdot \left(h - \frac{h^2}{2} \right) \quad \text{[ecuación 2.22]}$$

Cuando las funciones de membresía son gaussianas; se realiza una integración numérica del área debajo de la función. Como ejemplo de este método se realiza el proceso de

defusificación con los datos del ejemplo de la sección II.3.2., y asumiendo el mínimo para la implicación:

$$y = \frac{-55.5 - 27.75 - 57.75 + 0}{2.775 + 2.775 + 5.775 + 5.775} = \frac{-135.225}{17.1} = -7.907[N]$$

Debido a que se usa el mínimo para la implicación es necesario utilizar la ecuación 2.22 para el cálculo del área. Si se utiliza el producto para la implicación se obtiene el siguiente resultado:

$$y = \frac{-20 \cdot (1.5) - 10 \cdot (1.5) - 10 \cdot (3.5) + 0 \cdot (3.5)}{1.5 + 1.5 + 3.5 + 3.5} = \frac{-80}{10} = -8[N]$$

Como se ve, el resultado no difiere mucho, ya sea que se use el mínimo o el producto para la implicación.

<u>Centro Promedio (COA)</u>: La salida de control es calculada utilizando el centro de cada una de la funciones de membresía y la máxima certeza de cada consecuente representada a través del conjunto difuso inferido. La ecuación 2.23 permite calcular de esta forma la salida.

$$y = \frac{\sum_{i=1}^{R} c_i \cdot \sup\left\{\mu_{Bi}^{*}(y)\right\}}{\sum_{i=1}^{R} \sup\left\{\mu_{Bi}^{*}(y)\right\}} \qquad \text{[ecuación 2.23]}$$

Se demuestra en [19] que la ecuación 2.23 puede ser simplificada a:

$$y = \frac{\sum_{i=1}^{R} c_i \cdot \mu_{Ai}(x_1^*, x_2^*, \ldots, x_n^*)}{\sum_{i=1}^{R} \mu_{Ai}(x_1^*, x_2^*, \ldots, x_n^*)} \qquad \text{[ecuación 2.24]}$$

Esta expresión indica que basta con calcular la función de membresía de la premisa de cada regla activa, ponderarla con c_i, para luego sumar los resultados. Este método de defusificación es bastante sencillo y fácil de implementar, ya que no involucra de ninguna manera la forma de las funciones de membresía de salida.

A manera de demostración del uso de este método y para comparar con el del COG, se muestra a continuación el mismo ejemplo utilizado para la demostración del método anterior.

$$y = \frac{-20 \cdot (0.15) - 10 \cdot (0.15) - 10 \cdot (0.35) + 0 \cdot (0.35)}{0.15 + 0.15 + 0.35 + 0.35} = \frac{-8}{1} = -8[N]$$

Como se puede ver, se obtuvo el mismo resultado que en el ejemplo anterior que utilizaba el producto para la implicación. Esto es de esperarse ya que el área de un triángulo es proporcional a su altura.

II.3.4 REPRESENTACIÓN MATEMÁTICA DE SISTEMAS DE LÓGICA DIFUSA

El obtener la representación matemática ayudará a comprender la generación de la superficie no-lineal que los sistemas de lógica difusa intrínsecamente implementan.

Debido a que existen muchas formas de implementar un SLD, dependiendo de que operación se escoge para la conjunción, implicación, o que método se usa para la defusificación y qué forma toman las funciones de membresía, los sistemas de lógica difusa podrán tener varias representaciones matemáticas.

Para obtener la representación matemática de un SLD se puede partir de cualquiera de los métodos de defusificación, así por ejemplo el método de Centro-Promedio:

$$y = \frac{\sum_{i=1}^{R} b_i \cdot \mu_i}{\sum_{i=1}^{R} \mu_i}$$

Para ser más explícitos en la última ecuación, se requiere definir μ_i, en términos de las funciones de membresía individuales que son parte de cada una de las premisas. Suponiendo que se usa el producto para la conjunción de cada uno de los términos de la premisa de cada regla, y funciones de membresía triangulares definidas como en la tabla 2.1, donde el $\mu_j(x_j)$ representa una función de membresía para el j-ésimo universo de la variable de entrada; y además c_j^i (w_j^i) representa el centro (ancho de la base) de la i-ésima función de membresía triangular en el j-ésimo universo de la variable de entrada.

Supóngase además que se usan todas las posibles combinaciones de las funciones de membresía de entrada para conformar las reglas y que cada premisa tiene un término asociado para cada y todos los universos de las variables de entrada. Haciéndose todas estas consideraciones se puede escribir la última ecuación de una manera más detallada como:

$$y = \frac{b_1 \prod_{j=1}^{n} \mu_j^{L}(x_j) + b_2 \mu_1^{C_1}(x_1) \prod_{j=2}^{n} \mu_j^{L}(x_j) + \cdots}{\prod_{j=1}^{n} \mu_j^{L}(x_j) + \mu_1^{C_1}(x_1) \prod_{j=2}^{n} \mu_j^{L}(x_j) + \cdots} \qquad \text{[ecuación 2.25]}$$

En la ecuación 2.25 se presentan todas las posibles combinaciones de las premisas. Debido a que se utiliza una notación inapropiada para las funciones de membresía (L, C_i, R), la ecuación 2.25 es difícil de interpretar con facilidad; pero si en lugar de usar esta notación, se numeran las funciones de membresía de izquierda a derecha como $1,2,...,N_j$, donde N_j es el número de funciones de membresía en el j-ésimo universo de las variables de entrada y se usa a la nupla de índices $(j,k,...,l)$ para identificar a cada etiqueta de las variables de entrada, y se nota como $b^{(j,k,...,l)}_i$ a los centros de cada función de membresía de salida y donde el índice "i" denota a cada una de las reglas; se obtiene entonces la siguiente representación simplificada:

$$y = \frac{\sum_{i=1}^{R} b^{(j,k,...,l)_i} \mu_1^j \mu_2^k \cdots \mu_n^l}{\sum_{i=1}^{R} \mu_1^j \mu_2^k \cdots \mu_n^l} \qquad \text{[ecuación 2.26]}$$

donde los índices deben variar en los siguientes rangos:

$1 \le j \le N_1$; $1 \le k \le N_2$,..., $1 \le l \le N_n$

Como se han utilizado todas las posible combinaciones, entonces según la ecuación 2.19 existirán $R = \prod_{j=1}^{n} N_j$ reglas, y esto requerirá de $\sum_{j=1}^{n} 2N_j + \prod_{j=1}^{n} N_j$ parámetros [15] para describir al sistema, debido a que existen dos parámetros por cada función de membresía de entrada y R centros de funciones de membresía de salida.[19]

Suponiendo un caso simple en el que existan dos entradas y una salida, una vez que se ha seleccionado las alternativas que se van a utilizar para los distintos procesos internos del sistema difuso, se puede calcular un valor de salida para cada combinación de valores de entrada. Si se gráfica estos puntos se generará una superficie no-lineal en un espacio de tres dimensiones. A este superficie normalmente se la conoce como "superficie de control".

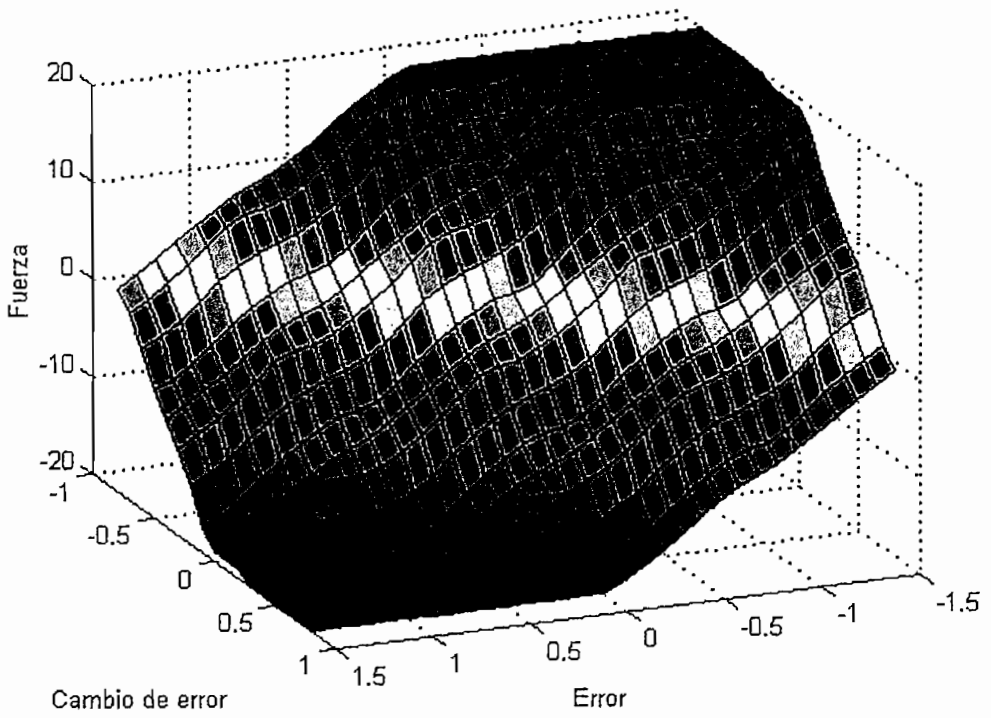


Figura 2.8 Superficie no-lineal generada por el sistema de lógica difusa para el control de un péndulo invertido.

II.3.5. SISTEMAS DIFUSOS DE TAKAGI-SUGENO

Los sistemas difusos analizados hasta el momento son conocidos como Sistemas Difusos de Mamdani o Standard. Un tipo diferente de sistemas difusos fue sugerida por Takagi y Sugeno en 1985. Estos sistemas, a su vez, son derivados de una clase más generalizada de sistemas difusos conocidos como " sistemas funcionales".

Un sistema difuso funcional, utiliza fusificación singleton, y una i-ésima regla tiene la siguiente forma:

$$\text{SI} <u_j \text{ es } A_i^j> \text{Y } <u_j \text{ es } A_i^k> \text{Y...Y } <u_l \text{ es } A_n^l> \text{ ENTONCES } b_i = g_i(\bullet)$$

Donde • representa simplemente el argumento de la función g_i y b_i no es el centro de las funciones de membresía de salida. La diferencia entre este tipo de funciones y los de Mamdani, es que el consecuente de la regla ya no tiene una etiqueta asociada a una variable lingüística sino una función $b_i = g_i(\bullet)$. En la mayoría de casos el argumento de g(•) contiene a las funciones de membresía de entrada, sin embargo también puede contener otras variables.

Cuando la función de salida tiene la siguiente forma:

$$b_i = g_i(\bullet) = a_{i,0} + a_{i,1}\, u_1 + \ldots + a_{i,n} u_n, \qquad \text{con} \qquad (a_{i,n} \varepsilon \Re)$$

Se conoce al sistema funcional como un sistema de Takagi-Sugeno. Si se desea obtener mayor información sobre este tipo de sistema se puede consultar en [9] y [19].

II.3.6. PROPIEDAD DE APROXIMACION UNIVERSAL

Se dice que cualquier sistema que sea capaz de aproximar una función desconocida, posee una propiedad de aproximación universal. Así por ejemplo si se escoge el método de defusificación por Centro-Promedio y el producto para la conjunción de la premisas y la implicación, y funciones de membresía Gaussianas, y se nota a este sistema difuso por f(u), entonces para cualquier función real $\psi(u)$ que sea contínua y acotada en un intervalo I, existe un sistema difuso f(u) tal que:

$$\sup_{U} |f(u) - \psi(u)| < \varepsilon$$

Donde ε es un valor real mayor a cero. Sin embargo esta propiedad no indica como encontrar este sistema. El aporte más significativo de esta propiedad tiene que ver con que indica que es posible conseguir ciertos requerimientos de desempeño. En especial para control, esta propiedad sugiere que es posible ajustar la superficie no-lineal generada por el sistema, pero no garantiza de manera alguna estabilidad u otras características propias de los sistemas de control.[19]

CAPITULO III

MODELACIÓN Y ANÁLISIS

III.1 CARACTERÍSTICAS BÁSICAS DE LOS ELEVADORES

Los elevadores o ascensores se hacen cada día más importantes a medida que las ciudades crecen verticalmente. Tal situación puede ejemplificarse en ciudades como Nueva York, Chicago y en general las grandes metrópolis del mundo, donde existen edificios con un número de pisos mayor a cien. Innovaciones en los elevadores han sido necesarias debido a nuevas tendencias arquitectónicas, y de ingeniería.

Los elevadores, como tales, no son una idea de nuestro tiempo sino que se remontan a culturas tan antiguas como la romana. En aquellas épocas estos eran accionados por la fuerza de animales o de personas y recorrían una distancia, guiados a veces por rieles. A comienzos del siglo XIX aparecieron los primeros elevadores accionados por máquinas a vapor, que a menudo causaban tragedias debido a que utilizaban sogas de cáñamo y cuando se rompían, no había forma de frenarlos. Una concepción moderna de los elevadores, los define como vehículos cuyo propósito es el de elevar personas o materiales, con la suficiente comodidad y la mayor seguridad posible.[22] Vehículos que cumplieran con este

tipo de requisitos no aparecieron sino hasta 1853, cuando Otis introdujo el primer elevador con sistema de seguridad para prevención de accidentes y que brindaba la suficiente comodidad a la gente que transportaba.

Las máquinas de vapor fueron rápidamente remplazadas por sistemas hidráulicos que utilizaban un ariete movido por agua. Este llegó a predominar durante bastante tiempo debido a que podía recorrer mayores distancias y a velocidades de alrededor de 3 [m/s], lo que era mucho más rápido comparado con el de vapor que alcanzaba una velocidad máxima de 0.2 [m/s].

En 1889 fue instalado el primer elevador eléctrico en el edificio Demarest de Nueva York, y que prestó servicio hasta 1920, cuando el edificio fue demolido. Este sistema básicamente difería del de vapor en que la máquina propulsora era eléctrica, pero el sistema mecánico se mantenía. Poco a poco los sistemas hidráulicos fueron remplazados por sistemas eléctricos con mejores características, así hasta llegar a 1903 cuando se instaló el primer elevador eléctrico a tracción, cuya introducción fue decisiva para llegar a los actuales sistemas.

En estos elevadores, el medio para trasmitir la fuerza elevadora a los cables es la tracción generada por el rozamiento entre las ranuras de la polea motriz de la máquina y los cables que la envuelven. Para conseguir tal rozamiento, se utilizan ranuras en forma de V, las cuales aprisionan al cable como se muestra en la figura 3.1.

Figura 3.1 Ranurado de una polea motriz.

Un extremo de los cables se conecta a la cabina y el otro a un contrapeso, para equilibrar fuerzas sobre la polea. Los sistemas de tracción dieron también mayor seguridad al pasajero, ya que la cabina no se suspendía de un solo cable, sino de varios.

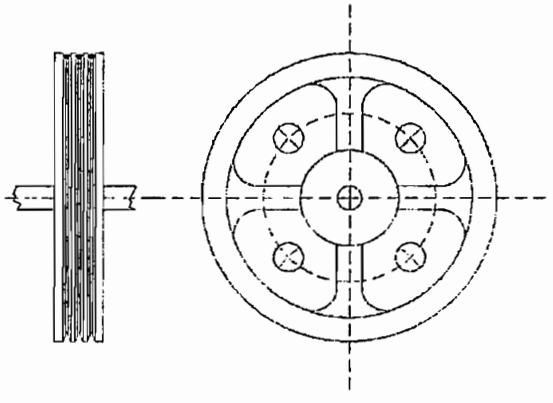


Figura 3.2 Polea de tracción común en elevadores.

La introducción de los sistemas eléctricos propulsores, brindaron un control más preciso del movimiento de la cabina, lo que permitió un manejo más exacto de la posición de la cabina y de su velocidad, lo que a su vez se reflejó en un conocimiento del tiempo de respuesta a una llamada.

Los primeros elevadores a tracción se acoplaban directamente al eje de la máquina con la polea motriz, por lo que se los conoce como elevadores de tracción directa. Este tipo de

sistema necesitaba para su operación de un motor eléctrico de DC de poca velocidad (50 a 200 rpm) de 4 a 8 polos, acoplado a una polea de 0.75 a 1.2 m de diámetro. Los motores de bajas revoluciones eran también necesarios para conseguir diámetros de poleas motrices adecuados al radio de flexión de los cables de acero. Se exige generalmente, incluso hoy, que el diámetro de la polea sea como mínimo 40 veces el del cable empleado.

La tracción directa es todavía muy empleada en los elevadores de gran recorrido y que avanzan a altas velocidades (>1.75 m/s). Algunas desventajas se presentan en este tipo de sistemas propulsores, y entre ellas están el de que una máquina de bajas revoluciones es bastante voluminosa, y las condiciones de operación de la misma son extremas debido a que debe ser capaz de arrancar con la cabina cargada, lo que demanda un alto torque de arranque.

Para solucionar estos inconvenientes se introdujo el uso de reductores mecánico del tipo sinfín-corona, que están acoplados entre el eje de la máquina y la polea motriz. Esta configuración permite el uso de motores de altas revoluciones, que a través del reductor disminuyen la velocidad de la cabina y aumentan considerablemente el torque. A su vez, con esto se permitió el uso de motores de alterna, que estaban condicionados por las altas revoluciones a las que operan.

Los motores de alterna, debían tener un doble bobinado. El bobinado de menor número de polos era empleado para viajar a la velocidad nominal, y el de mayor número de polos para el proceso de desaceleración. El funcionamiento normal de este sistema es el de arrancar con la velocidad alta, mantener esta durante el viaje, cambiar a la velocidad baja a una

determinada distancia del punto de destino, desconectar el motor en el punto de llegada e inmediatamente aplicar el freno mecánico, todo esto en una distancia de 3.6 m (altura media entre dos plantas).

Cuando la máquina era de contínua se utilizaba un grupo Ward-Leonard para su maniobra. El grupo Ward-Leonard consiste en un grupo motor(AC o DC) - generador(DC) conectado al inducido de la máquina motriz. Al variarse el campo del generador, varía el voltaje aplicado al inducido del motor y, en consecuencia, la velocidad y el torque del mismo. En contraste al sistema de alterna, este permite mejorar las características de arranque y parada. Permite además detener la cabina eléctricamente antes de aplicar el freno mecánico, mejorando la precisión, que a su vez, se refleja en un comportamiento más suave del sistema para el usuario. A pesar de todas estas ventajas, los sistemas de DC son muy complicados y poco versátiles, debido a la gran cantidad de mandos electromecánicos necesarios, el empleo de motores y generadores de DC que requieren de bastante mantenimiento, y el bajo factor de potencia del sistema. Este accionamiento todavía es muy utilizado, especialmente en elevadores anteriores a la década de los 90. La figura 3.3. muestra un típico sistema de tracción de motor DC, reductor y polea motriz.

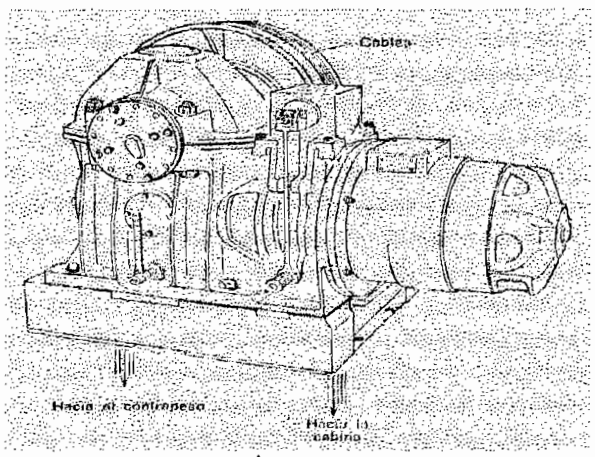


Figura 3.3 Máquina de tracción con reductor y polea motriz.

Actualmente, con la introducción de nuevas técnicas de control electrónico de potencia y su aplicación al control de máquinas, el uso de motores de inducción ha resurgido notablemente, ya que se permite una variación casi contínua en un rango muy amplio de velocidades con altos torques, y bajo factor de potencia. El empleo de estos sistemas reduce considerablemente el costo de mantenimiento y de operación, pero el costo de los controladores para estas máquinas es todavía alto.

En cuanto al sistema de frenos, esto no ha cambiado mayormente con el transcurso del tiempo. Este actúa sobre un tambor acoplado al eje del motor, mediante resortes y abierto eléctricamente con una bobina, como se muestra en la figura 3.4.

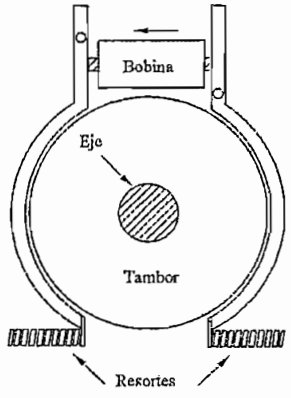


Figura 3.4 Freno mecánico empleado en elevadores.

En lo que respecta al sistema de poleas y cables, en realidad existen varias configuraciones dependiendo de la localización del cuarto de máquinas y del uso que vaya a prestar el elevador. Sin embargo, en la mayoría de los casos, el cuarto de máquinas se encuentra en la

última planta de los edificios, y las dos configuraciones más empleadas son las que se muestran en la figura 3.5.

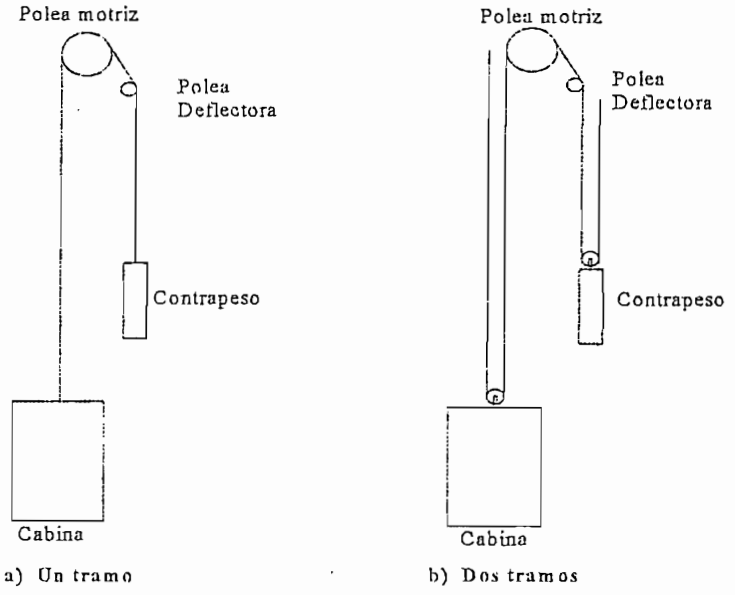


Figura 3.5 Dos configuraciones comunes del sistema de poleas.

La configuración de un tramo es la más empleada debido al menor recorrido de los cables, y sirve para levantar cargas de hasta 2230 N. La disposición de dos tramos se usa especialmente en elevadores de carga y materiales, debido a que la máquina motriz tendrá que generar únicamente la mitad de la fuerza con lo que puede elevar mayores cargas, pero necesita mayor longitud de cables por lo tanto es poco práctica para edificios altos, además la máquina puede trabajar a mayores revoluciones. Para los fines de este trabajo se considerará la configuración de un tramo debido a que es la configuración más común, e incluye todos los aspectos básicos de las demás configuraciones.

Por último, los sistemas modernos de elevadores procuran tener un incremento de velocidad lineal durante un corto lapso de tiempo hasta llegar a su velocidad nominal, la

cual la mantienen durante casi todo el recorrido, para luego desacelerar al sistema de una manera suave y controlada. La trayectoria que la posición describe en la mayoría de los elevadores actuales se presenta en la figura 3.6, y la velocidad en la figura 3.7.

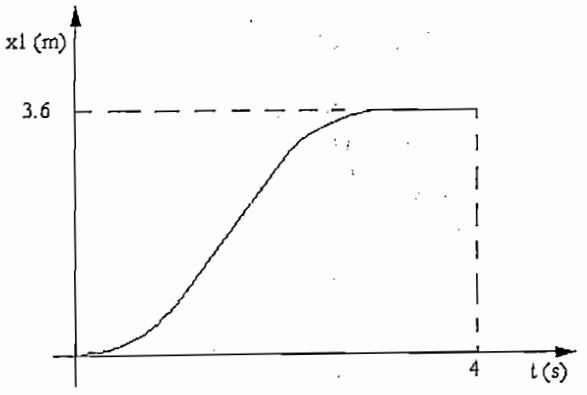


Figura 3.6 Trayectoria de posición de un elevador.

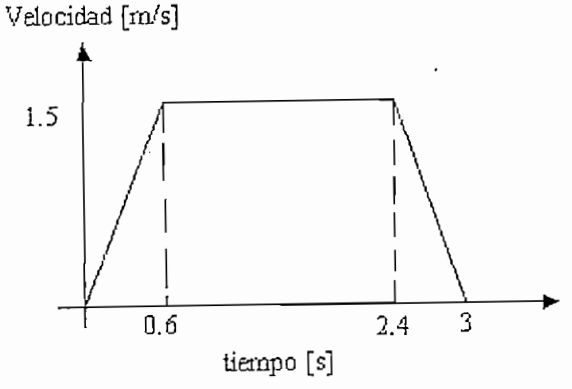


Figura 3.7 Trayectoria de la velocidad de un elevador.

III.2.1 GENERALIDADES SOBRE LOS FENÓMENOS DE ROZAMIENTO

El rozamiento es un fenómeno necesario y a la vez indeseable que está presente en el sistema de un elevador. Como se explicó, la potencia del motor es trasmitida a los cables de sujeción a través de un fuerza de tracción fruto de un fenómeno de rozamiento entre los cables y la superficie de la polea motriz, por lo que es de interés del diseñador maximizar este efecto, en esta circunstancia. Sin embargo el elevador se encuentra guiado dentro de la fosa por medio de rieles, y el contacto de estos rieles con los soportes del elevador producen una fuerza de rozamiento que se opone al movimiento.

Los fenómenos de rozamiento son los causantes de gran parte del calentamiento de las máquinas, y del desgaste de sus partes mecánicas. En todos los casos reales en que exista un deslizamiento entre materiales en contacto, existirán fuerzas de rozamiento que dan lugar a una pérdida de energía por disipación en forma de calor. El rozamiento es un fenómeno que se presenta de muchas formas y por lo tanto, existen algunos tipos, los más comunes son el rozamiento seco y el rozamiento fluido (o viscoso). [7, 15 y 17]

Los primeros estudios del fenómeno de rozamiento seco fueron realizados por Coulomb en 1781 y por Morin entre 1831 y 1834. El rozamiento seco se produce entre dos superficies no lubricadas en contacto que se están deslizando o con tendencia a deslizar. La naturaleza física del rozamiento se ha atribuido a muchos factores, tales como las irregularidades de la

superficie de los materiales e incluso a fuerzas de atracción de orden molecular. A pesar de que no existe una teoría muy detallada de este fenómeno, se han establecido modelos matemáticos suficientes para abordar la mayor parte de problemas.[17]

Se ha determinado experimentalmente que el rozamiento seco es esencialmente función de la fuerza normal a la superficie de contacto, que se presenta sobre un cuerpo como reacción[3] a la fuerza aplicada sobre ésta. Así por ejemplo en la figura 3.8, se muestra un diagrama de fuerzas de un bloque cualquiera, donde la fuerza normal se produce por reacción al peso del bloque.

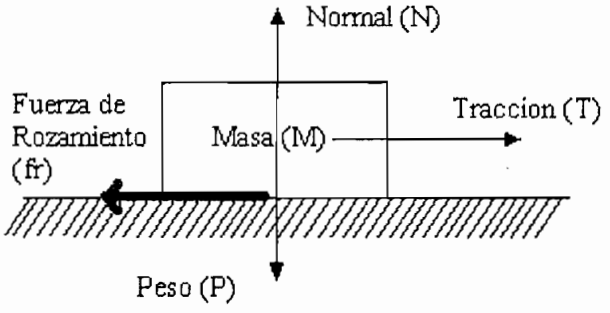


Figura 3.8 Diagrama de fuerzas sobre un bloque

Si al bloque se aplica una fuerza de tracción, que aumenta linealmente, éste tenderá a moverse, pero no lo hará sino hasta que se supere un valor límite del rozamiento, y que viene dado por $Fr_{c\ max} = \mu_{Emax} \cdot N$. Donde μ_{Emax} se conoce como el coeficiente de rozamiento estático máximo. A esta fase, en la que no existe todavía movimiento, sino una tendencia a moverse, se conoce como el dominio del rozamiento estático, y la magnitud de la fricción está determinada por las ecuaciones de equilibrio de fuerzas. [17]

[3] Las fuerzas de reacción se producen de acuerdo a la 2da Ley de Newton de acción y reacción.

Una vez que se ha superado el límite impuesto por el rozamiento estático máximo, el cuerpo empezará a moverse; sin embargo, el rozamiento no desaparece, pero experimenta un descenso rápido como se muestra en la figura 3.9. A esta fase se la conoce como el dominio del rozamiento cinético o dinámico. El rozamiento cinético obedece a la misma ley del rozamiento estático en cuanto a su proporcionalidad con la componente normal, pero el coeficiente de rozamiento dinámico es ligeramente menor al estático y se nota por μ_K.

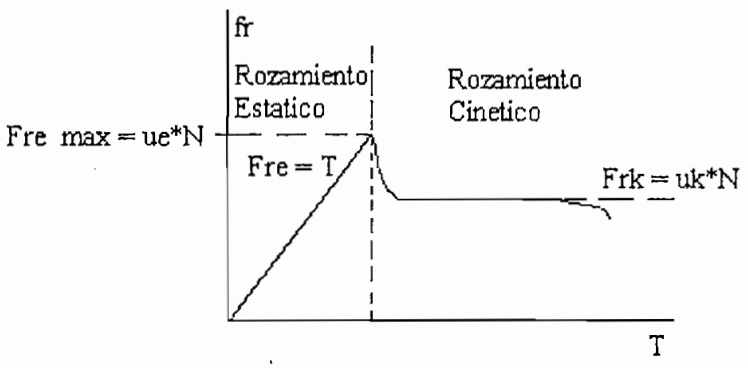


Figura 3.9 Dominios del rozamiento seco.

Luego del descenso en magnitud, el rozamiento se mantendrá constante durante un cierto tiempo, para luego disminuir aún más al aumentar la velocidad [17]. Se acostumbra considerar dos modelos para representar al rozamiento cinético, y que se muestran en la figura 3.10. [2, y 7]

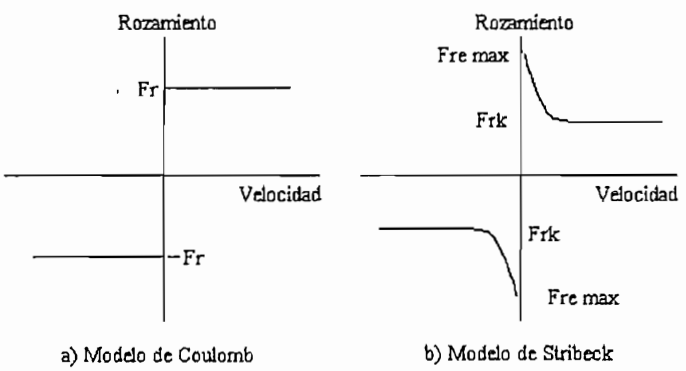


Figura 3.10 Modelos de rozamiento seco.

La expresión matemática para el modelo de la figura 3.10a es el siguiente:

$$fr_K = \begin{cases} -\mu_k \cdot N & \dot{x} < 0 \\ 0 & \dot{x} = 0 \\ \mu_k \cdot N & \dot{x} > 0 \end{cases}$$

Para la figura 3.10b, el modelo es: (Boa y Paveses 1982)

$$fr_K = \begin{cases} N\left(\Delta\mu \cdot e^{-\sigma \cdot \dot{x}} + \mu_K\right) & \dot{x} > 0 \\ 0 & \dot{x} = 0 \\ -N\left(\Delta\mu \cdot e^{\sigma \cdot \dot{x}} + \mu_K\right) & \dot{x} < 0 \end{cases}$$

donde:

$\Delta\mu = \mu_E max - \mu_K$ y σ determina la velocidad de decrecimiento desde el rozamiento estático máximo al cinético. Este parámetro presenta mucha incertidumbre y su determinación es experimental. Para fines de simulación se asumirá igual a 25 [$s \cdot m^{-1}$].

Por otro lado, estos modelos del rozamiento son, aplicables a todas las superficies secas en contacto hasta un cierto límite, e incluso a superficies móviles parcialmente lubricadas [17].

Los coeficientes de rozamiento dependen en gran parte de la condición exacta en que se encuentren las superficies, así como de la velocidad e igualmente se encuentran sujetos a un gran margen de incertidumbre. En la tabla 3.1. se presenta los valores de los coeficientes de rozamiento para superficies de acero sobre acero, y de acero sobre acero engrasado.

Coeficiente	Estático (μ_E)	Cinético (μ_K)
acero sobre acero (engrasado)	0.1	0.05
acero sobre acero	0.6	0.4

Tabla 3.1 Coeficientes de rozamiento estático y cinético para el acero.

Es importante además considerar el rozamiento fluido que se presenta cuando existe un movimiento relativo entre dos superficies separadas por algún líquido (lubricante) o gas, pues las superficies de piezas mecánicas se encuentran generalmente separadas por una película de lubricante. Cuando esto ocurre, se presenta una lubricación parcial, que es una condición intermedia entre el rozamiento seco y el fluido, y es difícil obtener un modelo fidedigno, por lo que se recurre a medidas experimentales. Se ha determinado sin embargo, que el rozamiento viscoso es una función lineal de la velocidad relativa entre las dos superficies[7], y se representa matemáticamente como: $Fr_v = Cv$. Donde C es un coeficiente de viscosidad y "v" es la velocidad relativa. Un gráfico de esta ecuación se muestra en la figura 3.11.

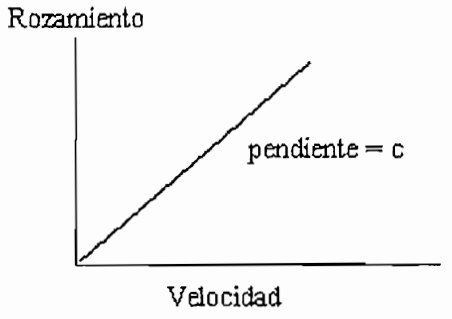


Figura 3.11 Fuerza de rozamiento viscosa

Los modelos del Rozamiento de seco propuestos, son aplicables a las fuerzas existentes en los rieles del elevador, mientras que el rozamiento viscoso aparecerá especialmente en los rodamientos del mecanismo.

III.2.2 FUERZA DE TRACCION EN UN ELEVADOR

La potencia generada por el motor es trasmitida a los cables a través de la polea, por medio de las fuerzas de rozamiento entre estos. Por la polea, dependiendo del elevador, pasarán al menos 3 cables de acero. Los cables que se utilizan para este tipo de vehículos están regidos por la norma ANSI 17.1 de la ASME. Estos cables están conformados por una parte central, o alma, de yute, que soporta y lubrica a los torones. Los cables empleados para elevadores son designados como cables de arrastre o tracción. Estos cables se encuentra sobredimensionados en factores de 7 a 12 veces la carga nominal con la que trabajarán [21]. De aquí se puede deducir que los efectos de elasticidad en ellos serán prácticamente nulos, y se los puede considerar rígidos, de manera que no presentarán alargamientos ni contracciones apreciables.

La polea de tracción o motriz posee ranuras en forma de V, como se muestra en la figura 3.1, de manera que aprisionan mejor los cables, aumentando el rozamiento y a la vez minimizando el deslizamiento entre estos y la superficie de la polea, para poder trasmitir la mayor potencia posible. Los ángulos de las ranuras deben ser tales que la fuerza para desencajar los cables de las ranuras no sea excesiva.[21]

En operación normal, no debe existir un deslizamiento entre la superficie de la polea y los cables, por lo que el sistema estará en equilibrio dinámico. En la figura 3.12 se muestra como un cable envuelve a la polea en un ángulo de contacto total β, donde el extremo de la tensión T_1 sujeta a la cabina y el de T_2 avanza a la polea deflectora y luego al contrapeso.

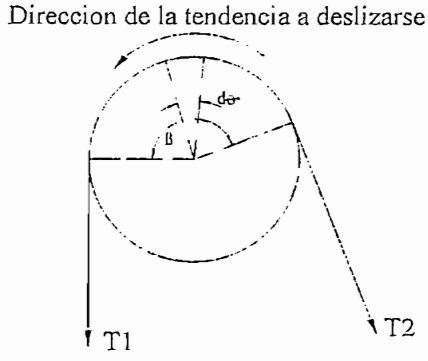


Figura 3.12 Tendencia al deslizamiento entre un cable y la polea de tracción

El cable sobre la polea se verá aprisionado en la ranura por acción de las tensiones, y las superficies inclinadas de la ranura ejercerán dos fuerzas normales en reacción a las tensiones como se muestra en la figura 3.13, donde se ha escogido un elemento diferencial dθ del ángulo de contacto total, por lo que todas las fuerzas también serán diferenciales..

Como el cable tiende a moverse en la dirección antihoraria (figuras 3.12 y 3.13), aparecerá una fuerza de rozamiento (dfr) en la dirección contraria y tangencial a la superficie de contacto, y en reacción a esta, también existirá una fuerza dT en dirección contraria.

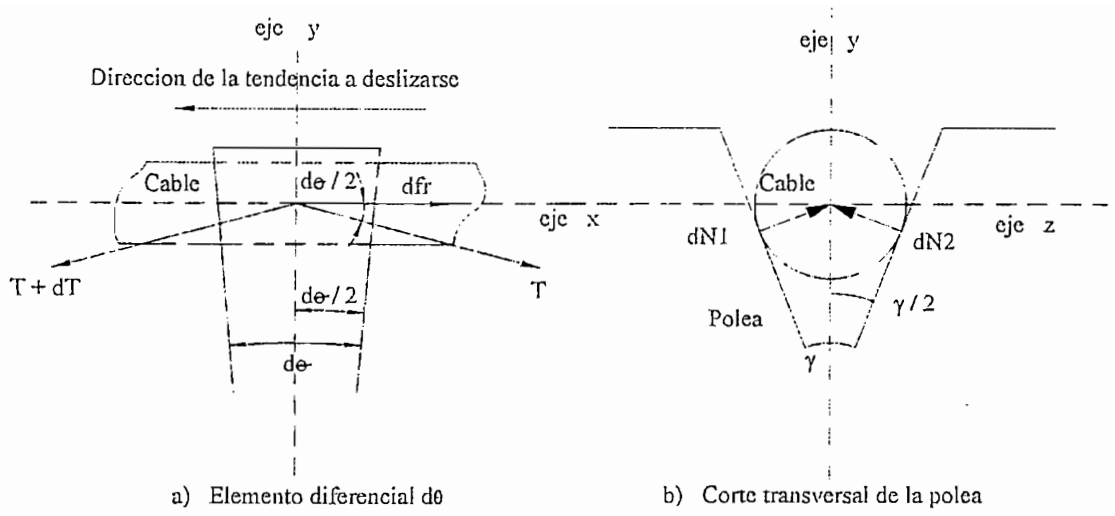


Figura 3.13 Fuerzas actuantes en un elemento diferencial $d\theta$ desde distintas vistas.

Escribiendo la ecuación de equilibrio en el sentido del eje y:

$$+\uparrow \sum F_y = 0$$

$$\Rightarrow (dN_1 + dN_2)\operatorname{sen}\left(\frac{\gamma}{2}\right) - T\operatorname{sen}\left(\frac{d\theta}{2}\right) - (T + dT)\operatorname{sen}\left(\frac{d\theta}{2}\right) = 0 \qquad \text{[ecuación 3.1]}$$

Las ecuación de equilibrio en la dirección del eje x será:

$$\overset{+}{\rightarrow} \sum F_x = 0$$

$$\Rightarrow T\cos\left(\frac{d\theta}{2}\right) + dfr - (T + dT)\cos\left(\frac{d\theta}{2}\right) = 0 \qquad \text{[ecuación 3.2]}$$

Como $d\theta$ es muy pequeño y tiende a cero se puede asumir que:

$$\cos\left(\frac{d\theta}{2}\right)=1;\ \ \text{sen}\left(\frac{d\theta}{2}\right)\approx\frac{d\theta}{2};\ \ dT\cdot d\theta=0$$

y decir que en magnitud

$$dN_1 = dN_2 = dN$$

Por lo que las ecuaciones 3.1 y 3.2 se pueden escribir respectivamente como:

$$dN=\frac{T}{2\,\text{sen}\left(\frac{\gamma}{2}\right)}d\theta \qquad \text{[ecuación 3.3]}$$

$$dT = dfr \qquad \text{[ecuación 3.4]}$$

Como se pretende encontrar la condición límite en la que se produciría un deslizamiento, la fuera de rozamiento (dfr) sería igual al diferencial del rozamiento estático máximo, así:

$$dfr = 2\mu_E max\cdot dN$$

Remplazando esta expresión en la ecuación 3.4, y a su vez la ecuación 3.3 en la ecuación del rozamiento, se tiene que:

$$\frac{dT}{T}=\frac{\mu_E max}{\text{sen}(\gamma/2)}d\theta$$

Integrando los dos miembros de la ecuación en sus respectivos límites:

$$\int_{T_2}^{T_1}\frac{dT}{T}=\int_0^{\beta}\frac{\mu_E max}{\text{sen}(\gamma/2)}d\theta$$

$$\Rightarrow \ln\left|\frac{T_1}{T_2}\right|=\frac{\mu_E max\cdot\beta}{\text{sen}(\gamma/2)}$$

$$\Rightarrow T_1 = T_2 e^{\left(\frac{\mu_E max\cdot\beta}{\text{sen}(\gamma/2)}\right)} \qquad \text{[ecuación 3.5]}$$

De esta última ecuación se puede concluir que:

- El rozamiento será mayor a medida de que γ sea más pequeño.
- Para que no exista deslizamiento se debe cumplir al menos que $T_1 < T_2 e^{\left(\frac{\mu_E max \cdot \beta}{sen(\gamma/2)}\right)}$.
- Mientras mayor sea el ángulo de contacto β, mayor será el rozamiento.

Se ha considerado en este análisis un solo cable sometido a una tensión T; en la realidad existen "n" (n =0-5) cables, por lo que la tensión sobre cada uno, asumiendo una distribución uniforme de fuerza, es de: Tensión/cable=T_i/n; i = 1,2

Este análisis es importante en la medida, de que en el arranque del sistema, existe un pequeño deslizamiento debido a las demás fuerzas de rozamiento y a la carga del elevador, durante un intervalo corto de tiempo. Durante este intervalo, la máquina de tracción debe alcanzar a vencer estas fuerzas, y este efecto se ve traducido en las condiciones iniciales del sistema, y se lo aborda en la siguiente sección.

III.2.3 MODELO DINÁMICO DEL ELEVADOR

Para poder encontrar el modelo dinámico del sistema se hace un análisis individual de cada componente del mismo. Para tal efecto, se toman las siguientes consideraciones, y aproximaciones:

- Se desprecia cualquier efecto secundario de la polea deflectora sobre los cables de sujeción, ya que su única función es la de alejar lo suficiente al contrapeso de la cabina.
- Una vez en movimiento el elevador y de la sección anterior, no existe deslizamiento entre la polea motriz y los cables, y estos no presentan características elásticas.
- Para el modelo del rozamiento entre los rieles y los soportes de la cabina se considera que esta fuerza se distribuye por igual.
- Para el contrapeso, se hace la misma consideración que en el punto anterior.
- Durante el trayecto no se presentan oscilaciones laterales de la cabina o del contrapeso.
- Todos los cables están sometidos a la misma tensión, y se consideran fuerzas netas en el análisis.
- Cabe indicar que se han considerado para el análisis, que los elementos principales del elevador son: cabina, polea motriz, polea deflectora y contrapeso.
- No se consideran no linealidades existentes en los sistemas de engranajes y trasmisión.

Cabina:

La cabina, al igual que el contrapeso, se desplaza verticalmente, guiada por rieles sujetos a las paredes de la fosa del elevador, y que a su vez encajan en soportes adheridos a sus respectivas estructuras. Un diagrama de estos se presenta en la figura 3.14.

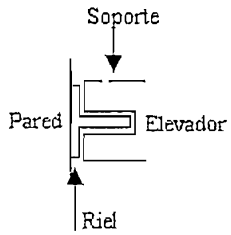


Figura 3.14 Disposición de un riel y un soporte vistos desde la parte superior.

En el espacio entre los rieles y los soportes existe una película de grasa, que se asume uniforme, por lo que se presenta una condición de lubricación límite, que puede ser representada por los modelos descritos en la sección III.2., para rozamiento seco.

La figura 3.15 muestra el diagrama de cuerpo libre de la cabina, desplazándose hacía arriba:

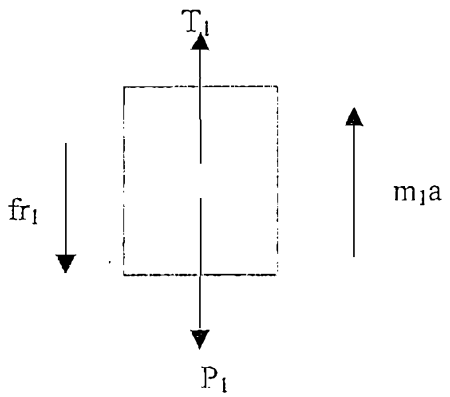


Figura 3.15 Diagrama de cuerpo libre de la cabina de un elevador.

Donde:

T_1: Es la tensión del cable

fr_j: Es la fuerza de rozamiento neta entre rieles y soportes.

P_j: Es el peso.

m_j: Es la masa del vehículo.

Antes de entrar en movimiento, la tensión T_1 deberá vencer la fuerza de rozamiento estática entre los rieles y los soportes, y el peso de la cabina, por lo que escribiendo la ecuación de equilibrio de fuerzas en para la condición de rozamiento estático máximo se encuentra que:

$$+\uparrow \sum Fy = 0$$
$$T_1 - P_1 - \underset{E}{Fr_1}\,\max = 0$$
$$T_1 = P_1 + \underset{E}{Fr_1}\,\max \qquad \text{[ecuación 3.6]}$$

Una vez que $T_1 > P_1 + \underset{E}{Fr_1}\,\max$, el elevador empezará a moverse. Como se mostró en la sección III.2, la fuerza de rozamiento para condiciones inferiores a la del rozamiento máximo está determinada por la ecuación de equilibrio. Sobrepasado este límite se puede escribir la ecuación dinámica como sigue:

$$+\uparrow \sum Fy = m_1 a$$
$$T_1 - P_1 - fr_1 = m_1 a$$
$$\Rightarrow T_1 = m_1 a + P_1 + fr_1 \qquad \text{[ecuación 3.7]}$$

En la ecuación 3.7 la tensión "T_1", la aceleración "a" y la fuerza de rozamiento son funciones implícitas del tiempo.

Polea de Motriz o de Tracción:

Considerando positivo el sentido horario, se escribe la ecuación dinámica de la polea de la siguiente manera:

$$\sum M_0 = I_1 \cdot \alpha$$
$$T_2 \cdot R + \Gamma_L' - T_1 \cdot R = I_1 \cdot \alpha \qquad \text{[ecuación 3.8]}$$

Donde:

Γ_L': Es el torque mecánico en el secundario del reductor.

R: Es el radio de la polea motriz.

I_j: Momento de inercia de una polea

α: Aceleración angular

De los resultados de la cinemática rotacional se conoce que $\alpha = \frac{a}{R}$, que remplazando en la ecuación 3.8, y despejando la aceleración se encuentra que:

$$a = \frac{(T_2 - T_1)R^2 + R \cdot \Gamma_L'}{I_1} \qquad \text{[ecuación 3.9]}$$

Ahora, del análisis de la cabina, se dedujo que ésta no se desplazará hasta que se cumpla que $T_1 > P_1 + \underset{E}{Fr_1}\,max$, y antes de esto la velocidad de ascenso (v) es nula, por lo que a = 0, que sustituyendo en la ecuación 3.9 se deduce que:

$$\Gamma_L' > \left(\underset{E}{Fr_1}\,max + P_1 - T_2 \right) \cdot R \qquad \text{[ecuación 3.10]}$$

Este resultado indica que la máquina debe primero generar un torque Γ_L' que cumpla con la ecuación 3.10, para que el sistema entre en movimiento.

Polea Deflectora:

Un diagrama de las fuerzas que actúan sobre la polea deflectora se presenta en la figura 3.16.

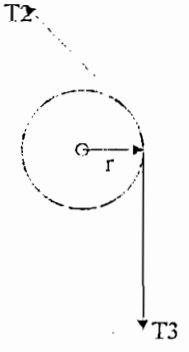


Figura 3.16 Fuerzas sobre una polea deflectora.

Para la ecuación de dinámica de la polea, se considera positivo el sentido horario, entonces:

$$\sum M = I_2 \alpha$$

$$T_3 \cdot r - T_2 \cdot r = I_2 \cdot \frac{a}{r}$$

$$\Rightarrow T_2 = T_3 - I_2 \frac{a}{r^2} \qquad \text{[ecuación 3.11]}$$

Al igual que para 3.10, en la condición de movimiento inminente se cumple que:

$$T_2 = T_3 \qquad \text{[ecuación 3.12]}$$

Contrapeso:

El diagrama de cuerpo libre se presenta en la figura 3.17.

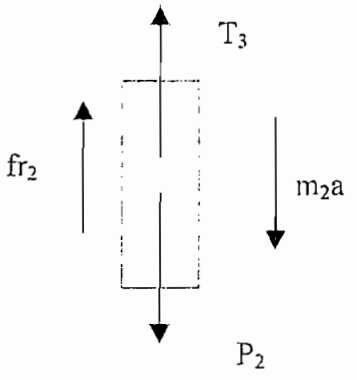


Figura 3.17 Diagrama de cuerpo libre del contrapeso.

La condición de movimiento inminente se calcula de igual manera que para la cabina y se obtiene que:

$$T_3 < P_2 - \underset{E}{Fr_2}\max \qquad \text{[ecuación 3.13]}$$

La ecuación dinámica, de similar manera es:

$$T_3 = P_2 - m_2 a - fr_2 \qquad \text{[ecuación 3.14]}$$

Para establecer la condición global de movimiento inminente se reemplazan las ecuaciones 3.12, 3.13 en la 3.10, obteniéndose:

$$\Gamma_L' > \left(\sum_E Fr\max - \Delta P\right)\cdot R \qquad \text{[ecuación 3.15]}$$

Donde :

$\Delta P = P_2 - P_1$

$\sum_E Fr\max = \underset{E}{Fr_1}\max + \underset{E}{Fr_2}\max$

Esta ecuación indica que debido a la ayuda del contrapeso, el torque inicial puede ser cero cuando $\Delta P > \sum_E Fr\max$, ya que esta fuerza es suficiente para vencer la fuerza de rozamiento. Cuando $\Delta P < \sum_E Fr\max$, ocurre lo contrario, debe existir un torque de arranque suficiente para vencer el torque generado por $\sum_E Fr\max - \Delta P$.

Es por esto, que la condición de movimiento inminente permite establecer la condición inicial del torque que será:

$$\Gamma_{Lo}' = \begin{cases} 0 & \text{cuando} \quad \Delta P \geq \sum_E Fr\,max \\ \left(\sum_E Fr\,max - \Delta P\right)\cdot R & \text{cuando} \quad \Delta P < \sum_E Fr\,max \end{cases}$$

La ecuación dinámica del elevador, válida una vez que T_L' cumple con la condición anterior, se obtiene remplazando 3.7, 3.11, y 3.14 en 3.9, encontrándose que:

$$a = \frac{\left(\Delta P - \sum fr\right)\cdot R^2 + R\cdot \Gamma_L'}{I_1 + I_2\left(\frac{R}{r}\right)^2 + R^2 \sum m} \qquad \text{[ecuación 3.16]}$$

Donde:

$$\sum fr = fr_1 + fr_2$$

$$\sum m = m_1 + m_2$$

III.2.4 MODELO DE LA MAQUINA

La mayoría de los elevadores, construidos en las décadas anteriores, emplean motores de DC controlados por voltaje de armadura. Ultimamente han empezado a utilizarse los motores de AC. Estos requieren de un control de la magnitud del voltaje aplicado y de su frecuencia, en una relación constante (V/f=cte.) logrando así variar su velocidad en todo el rango. Se requiere que la relación V/f sea constante, para mantener el flujo del entrehierro constante y evitar efectos de saturación. Los efectos de saturación, producen elevaciones innecesarias en las corrientes de fase de la máquina.

Ya sea que la máquina sea de contínua o de alterna, ésta estará conectada a un conversor estático de potencia. El conjunto conversor – motor – carga es conocido como un accionamiento de máquina, y se muestra en la figura 3.18.

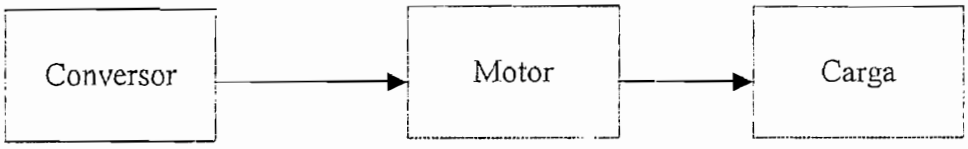


Figura 3.18 Diagrama de bloques de un accionamiento de máquinas.

Un modelo detallado del motor no es necesario para este trabajo, debido a que no es de interés analizar el comportamiento interno del mismo, sino más bien el conocer de manera más general su comportamiento ante cambios de la magnitud de su entrada.

Un motor, puede ser representado por un sistema eléctrico en cascada con un sistema mecánico, como se muestra en la figura 3.19.

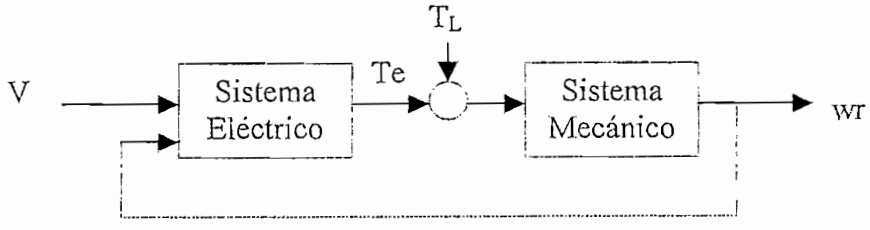


Figura 3.19 Diagrama de bloques de un motor de inducción

La parte eléctrica, en realidad, es un sistema no-lineal, ya que existen, por ejemplo, multiplicaciones entre las funciones de voltaje V y la velocidad w_r, efectos de saturación y otros. Para efectos de este análisis y por la razón antes expuesta, se lo representa por una ecuación diferencial lineal de primer orden, como la que sigue:

$$\dot{\Gamma e} + \frac{1}{\tau_e}\Gamma e = Ke \cdot V \qquad \text{[ecuación 3.17]}$$

Donde:

Γ_e es el torque electromagnético efectivo de la máquina.

V es el voltaje rms a los terminales de la máquina.

τ_e constante de tiempo de la máquina que es aproximadamente igual a:

$$\tau_e = \frac{Le}{Re} = \frac{L_1 + L_2}{Re}.$$

Las inductancias L_1 y L_2 corresponden a las inductancias del estator y rotor (referida al estator) respectivamente. La resistencia Re, es la resistencia equivalente vista desde del estator.

En gran medida esta formulación corresponde a observaciones del comportamiento de la máquina más que a un estudio detallado de la misma.

Como se mostró en la sección anterior, la carga del motor es altamente inercial, por lo que su respuesta es lenta comparada con el sistema eléctrico, implicando que la constante eléctrica es mucho menor que la mecánica.

El sistema mecánico del motor está descrito por una ecuación diferencial de primer orden, que representa la sumatoria de torques en el eje del motor; de esta manera la ecuación tiene la forma:

$$Te - T_L = J\frac{d}{dt}wr \qquad \text{[ecuación 3.18]}$$

Donde:

T_L es el torque de carga.

J es la inercia de la máquina.

wr es la velocidad rotórica del motor.

Esta ecuación ya fue utilizada en la sección anterior concluyendo en la ecuación 3.16. Sin embargo la ecuación 3.16 se encuentra en el secundario del reductor, y la aceleración "a" es tangencial. Para poder referir la ecuación 3.17 al secundario, bastará con dividir la Te por la relación de acoplamiento del reductor N definida como:

$$N = \frac{T_1}{T_2} = \frac{w_2}{w_1}$$

Donde los subíndices 1 y 2 denotan el primario y el secundario del reductor respectivamente.

Por lo tanto se tiene la siguiente equivalencia:

$$\Gamma'_L = \frac{1}{N} \cdot Te \qquad \text{[ecuación 3.19]}$$

El modelo del elevador quedará definido entonces por las ecuaciones 3.16, 3.17, y la definición 3.19.

III.2.5 SENSORES DE POSICIÓN Y VELOCIDAD

Existen algunos tipos de sensores de posición y velocidad que se utilizan en elevadores modernos. Este se escoge según algunos factores importantes, tales como:

- Costo.
- Precisión.
- Rango.
- Condiciones de operación.

Para sensar la velocidad, suelen utilizarse dos alternativas.

- Tacogenerador.
- Codificador digital.

El tacogenerador, es de menor costo. Este genera una señal de voltaje cuya magnitud es proporcional a la velocidad del eje del motor. La desventaja de este sensor es que no provee información sobre la posición de la cabina y entrega además una señal analógica, que debe ser acondicionada para luego ser digitalizada.

Los elevadores modernos emplean codificadores, que brindan la facilidad de entregar una señal digital, que simplifica su uso en un sistema microprocesado; sin embargo, su costo es elevado. Existen 2 tipos de codificadores: rotatorios y lineales.

Independientemente del tipo de codificador, estos producirán dos tipos de señales:

- La primera señal será una medida de la velocidad con la que esta rotando el motor o desplazándose un objeto.
- La segunda señal provee un código relativo a la posición angular o lineal, dependiendo del codificador que se use.

Para los sistemas de elevadores, los codificadores lineales, son los más utilizados y consisten en una lámina tendida a lo largo de todo el foso del elevador. Esta lámina posee perforaciones en todo su trayecto. Las perforaciones, a su vez, son capturadas por un sistema digital que incrementa o decrementa un registro, determinando la posición de la cabina.

En la medida de que el código que entrega el codificador para una posición cualquiera es único, se puede aproximar este dispositivo a una función lineal de la forma:

$$V_p = K_p \cdot Y$$

Donde:

V_p Voltaje de salida.

Y Posición vertical.

K_p Constante de proporcionalidad.

La información de la velocidad viene dada en forma de un cierto número de pulsos por segundo, dependiendo de la velocidad con la que se mueva la cabina, lo que también se asemeja a una aproximación lineal de la velocidad con la frecuencia de salida de los pulsos. Así se puede escribir que:

$$V_v = K_v \cdot \frac{dy}{dt}$$

Donde:

V_v Voltaje de salida.

$\frac{d}{dt}y$ Velocidad de la cabina.

K_v Constante de proporcionalidad.

III.3 ANÁLISIS GENERAL DEL MODELO

Esta sección presenta un análisis del modelo del elevador. Este, a su vez, brindará las pautas necesarias para el diseño de los distintos controladores y proveerá de un conocimiento heurístico del comportamiento del elevador para el diseño del controlador difuso.

III.3.1 DIAGRAMA DE BLOQUES

Considerando que:

x_1 Posición vertical de la cabina.

x_2 Velocidad de ascenso la cabina, y es igual a $\dot{x}_1$.

x_3 Torque electromagnético del motor.

u Voltaje de entrada al motor.

El sistema, entonces, expresado en variables de estado se presenta como:

$$\dot{x}_1 = x_2 \qquad \text{[ecuación 3.20]}$$

$$\dot{x}_2 = \frac{(\Delta P - \sum fr)\cdot R^2 + R/N \cdot x_3}{I_1 + I_2\left(\frac{R}{r}\right)^2 + R^2 \sum m} \qquad \text{[ecuación 3.21]}$$

$$\dot{x}_3 = -\frac{1}{\tau_e} x_3 + Ke \cdot u \qquad \text{[ecuación 3.22]}$$

$$y = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad \text{[ecuación 3.23]}$$

Si se agrupan constantes, de la siguiente manera:

$$K_1 = \frac{R^2}{J}$$

$$K_2 = \frac{R}{N \cdot J}$$

$$K_3 = \frac{1}{\tau_e}$$

$$K_4 = Ke$$

$$T_L = K_1 \cdot \Delta P$$

$$J = I_1 + I_2 \left(\frac{R}{r}\right)^2 + R^2 \cdot \sum m$$

Y notando a:

$$\sum fr = f(x_2)$$

Se tiene entonces, al sistema escrito de manera simplificada como:

$$\dot{x}_1 = x_2 \qquad \text{[ecuación 3.20]}$$

$$\dot{x}_2 = K_1 \cdot \Delta P - K_1 \cdot f(x_2) + K_2 \cdot x_3 \qquad \text{[ecuación 3.24]}$$

$$\dot{x}_3 = -K_3 x_3 + K_4 \cdot u \qquad \text{[ecuación 3.25]}$$

$$y = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \qquad \text{[ecuación 3.26]}$$

Con condiciones iniciales iguales a:

$$\dot{X}o = \begin{pmatrix} 0 \\ 0 \\ \Gamma_{Lo}' \cdot N \end{pmatrix}$$

Donde Γ_{Lo}' se definió en la sección anterior como:

$$\Gamma_{Lo}' = \begin{cases} 0 & \text{cuando} \quad \Delta P \geq \sum_E Fr\max \\ \left(\sum_E Fr\max - \Delta P\right) \cdot R & \text{cuando} \quad \Delta P < \sum_E Fr\max \end{cases}$$

El sistema, en lazo abierto, puede ser entonces representado por el diagrama de bloques de la figura 3.20.

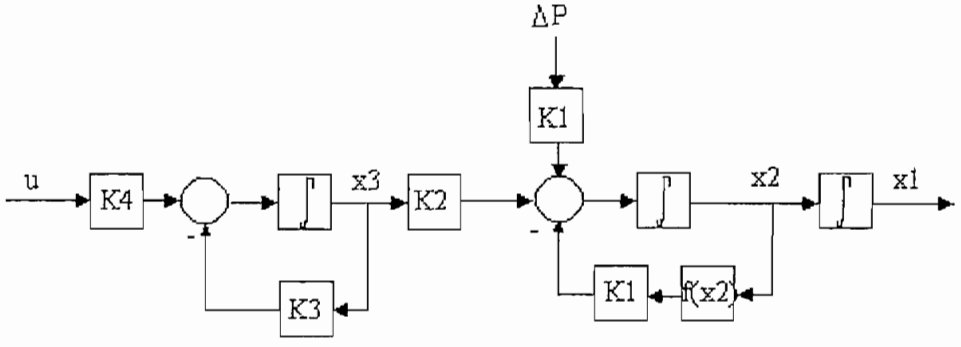


Figura 3.20 Diagrama de bloques del elevador en lazo abierto.

III.3.2 ESPECIFICACIONES DE UN ELEVADOR

Dependiendo del elevador, este poseerá distintas especificaciones de operación. Se ha escogido un elevador típico de un edificio de seis pisos situado en la Av. Juan de Dios Martínez N34 – 467 y Portugal, edificio Terrazas Portela.

Se realizó la recolección de datos de placa y mediciones de distintos elementos del mecanismo, y se presentan a continuación.

Relación de acoplamiento del Reductor (primario a secundario): 1/27

Radio de la polea motriz (R): 0.31 m

Radio de la polea deflectora (r): 0.55 m

Peso del contrapeso: 8202.6 N (837 kg-f)

Cabina:

- Peso de carga: 5145 N (525 kg-f)
- Peso de cabina: 5782 N (590 kg-f)
- Carga máxima: 11867.8 N (1211 kg-f)
- Velocidad: 1.5 m/s
- Velocidad máxima: 1.8 m/s

Peso de la polea deflectora: 393.96 N (40.2 kg-f)

Peso de la polea motriz: 470.4 N (48 kg-f)

Tiempo aproximado de ascenso de una planta a otra (3.6m): 3.5 - 4 s.

Tiempo aproximado de descenso de una planta a otra (3.6m): 3.5 - 4 s.

Se estima que la parte eléctrica del sistema posee una constante de tiempo de τ_e=0.027s.

A partir de estos datos se ha podido determinar las distintas constantes, y sus rangos de variación, que se encuentran tabulados a continuación.

Carga	ΔP (N)	K_1 (kg^{-1})	K_2 ($m^{-1}kg^{-1}$)	K_3 (s^{-1})	K_4
Mínima	2420.6	0.00067976	0.05920526	37.037	138.889
Máxima	-2724.4	0.00050098	0.04363352	37.037	138.889
Nominal	0	0.00058204	0.05069371	37.037	138.889

Tabla 3.2 Rangos de las constantes del sistema.

Debido a que el sistema posee parámetros variantes, es conveniente establecer un modelo nominal, con parámetros que reflejen las características normales de funcionamiento.

Por las especificaciones del fabricante, se toma como criterio para la selección de los parámetros nominales, la condición en que la masa de la cabina sea igual a la masa del contrapeso, obteniéndose así el siguiente conjunto de parámetros nominales.

K_{1n}	K_{2n}	K_{3n}	K_{4n}
0.00058204	0.05069371	37.037	138.889

Tabla 3.3 Parámetros nominales del modelo del elevador.

Este sistema posee algunas peculiaridades importantes, que son:

- Es no-lineal.
- Es un sistema con perturbación externa.
- Es un sistema de parámetros variantes.

Por simplicidad, para la fuerza de rozamiento neta en los rieles, se ha escogido el modelo de Coulomb, extensiones de los resultados son posibles para el modelo de Stribeck. En cuanto a la fuerza de rozamiento viscoso, se asume que el coeficiente de rozamiento "C" es constante y siempre está presente.

Un elemento fundamental para poder realizar los análisis posteriores es la determinación del punto de equilibrio del sistema. La determinación de éste se la realiza por medio de la condición:

$$\dot{X} = 0$$

En lazo abierto el sistema no tiene punto de equilibrio, pero en lazo cerrado (figura 3.21), el punto de equilibrio existe y es:

$$x_e = \begin{bmatrix} \frac{K_3 K_1}{K_2 K_4} \\ 0 \\ -\frac{K_1}{K_2} \end{bmatrix} \cdot \Delta P + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot \dot{r}$$

Como se observa éste es dependiente de la perturbación y de la entrada de referencia.

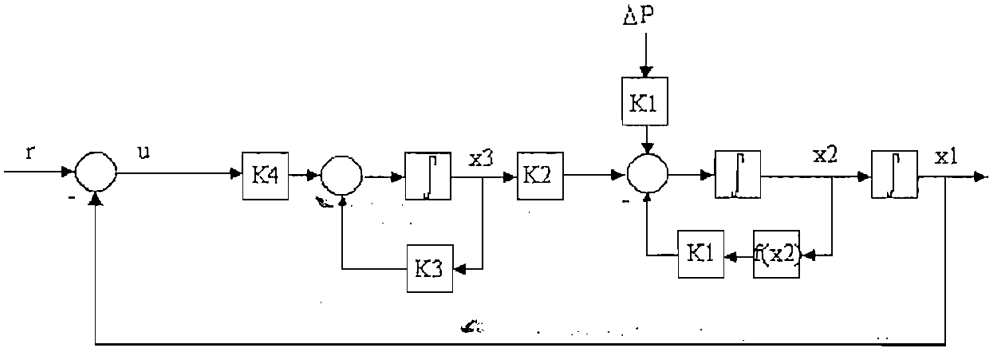


Figura 3.21 Diagrama de bloques del sistema en lazo cerrado.

Como se mostró en secciones anteriores, el término $f(x_2)$ representa las fuerzas de rozamiento de seco y viscoso, cuya expresión es:

$$f(x_2) = C \cdot x_2 + fr_K$$

Donde C es el coeficiente de rozamiento viscoso que se asume igual a 0.0531 [Ns/m], para todo el análisis, y fr_k (rozamiento de Coulomb) se definió anteriormente como:

$$fr_K = \begin{cases} -\mu_k \cdot N & \dot{x} < 0 \\ 0 & \dot{x} = 0 \\ \mu_k \cdot N & \dot{x} > 0 \end{cases}$$

Únicamente para simplificar la notación, se hace que la magnitud de la fuerza de rozamiento ($\mu_k \cdot N$) se represente por Fr_k.

Un gráfico de las trayectorias que describe el sistema en el espacio de estado, puede ser de gran utilidad para una visualización más clara de su comportamiento. Éstas pueden ser trazadas en un espacio tridimensional, para lo cual se asume que las condiciones iniciales están en el origen, y se consideran tres estados de carga: mínima, nominal y máxima, con referencia nula y un valor de $K_1 Fr_K = 0.5$, y cuyos trazos se muestran en la figura 3.22.

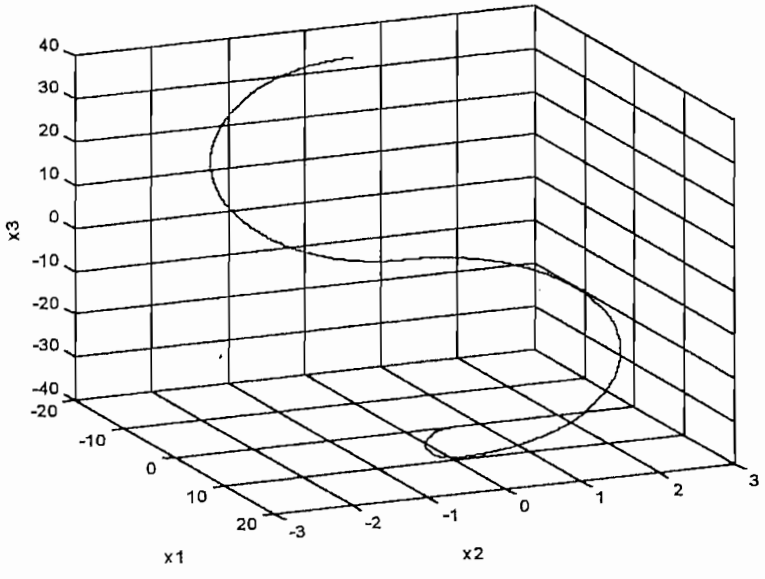


Figura 3.22 Trayectorias del sistema para condiciones de carga mínima(azul), nominal, máxima(magenta), para una $K_1 Fr_k = 0.5$.

En realidad, la figura muestra tres trayectorias, donde la trayectoria correspondiente a la carga nominal es un punto en el origen, y los tres puntos de equilibrio son:

$$\text{Carga mínima}: x_e = \begin{bmatrix} 7.41 \\ 0 \\ -27.78 \end{bmatrix}$$

$$\text{Carga nominal}: x_e = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Carga máxima}: x_e = \begin{bmatrix} -8.341 \\ 0 \\ 31.28 \end{bmatrix}$$

Si la magnitud de la fuerza de rozamiento disminuye, se observa que el sistema exhibe un ciclo límite, como se muestra en la figura 3.23, donde $K_1 Fr_k$ se ha hecho igual a 0.15, y con carga mínima.

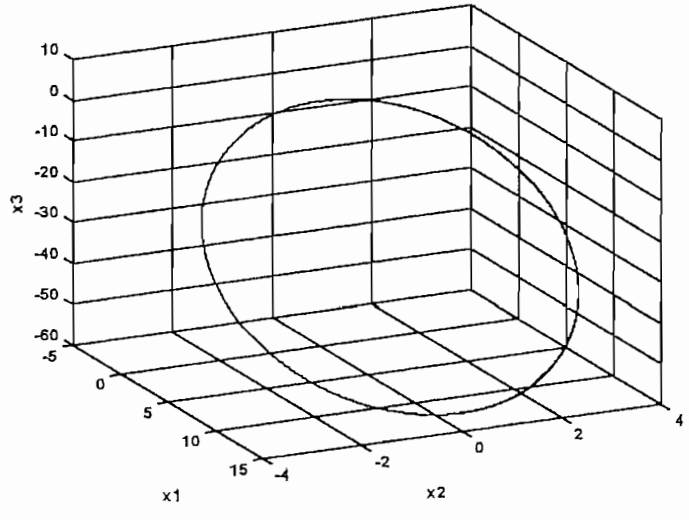


Figura 3.23 Trayectoria para: $K_1 Fr_k = 0.15$, para carga mínima.

Por lo visto, es necesario realizar un análisis por función descriptiva para poder entender de mejor manera los efectos de la variación de la magnitud del rozamiento, y que se lo presenta posteriormente.

Se puede obtener mayor información de la no-linealidad analizando el comportamiento en el plano de fase, de la parte del sistema que contiene a la discontinuidad (figura 3.24).

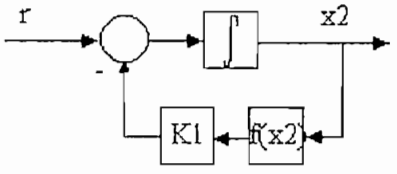


Figura 3.24 Parte del sistema que contiene la no-linealidad.

La ecuación diferencial de este diagrama de bloques es:

$$\dot{x}_2 = r - K_1 c \cdot x_2 - K_1 \cdot fr_K$$

El gráfico de $\dot{x}_2$ en función de x_2 se muestra en la figura 3.25, para una referencia constante.

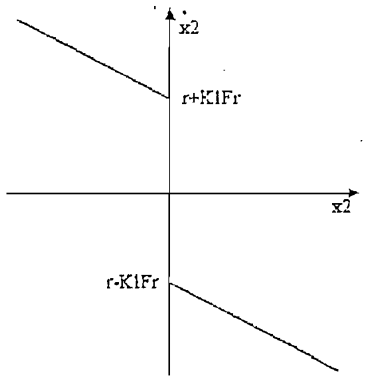


Figura 3.25 Diagrama de $\dot{x}_2$ vs x_2.

III.3.4 ANÁLISIS DEL MODELO LINEALIZADO

Para linearizar el sistema, se realiza una expansión en series de Taylor de las ecuaciones 3.20, 3.24 y 3.25 donde $u = K_4 - x_1$ (sistema en lazo cerrado) alrededor del conjunto de puntos de equilibrio. En la expansión se desprecian los términos de orden superior, y se obtiene la siguiente matriz:

$$A^* = \begin{pmatrix} 0 & 1 & 0 \\ 0 & -K_1 c & K_2 \\ -K_4 & 0 & -K_3 \end{pmatrix}_{P.E.}$$

La ecuación característica del sistema, obtenida mediante $|\lambda I - A^*| = 0$ es:

$$|\lambda I - A^*| = \lambda^3 + [K_1 c + K_3]\lambda^2 + K_1 K_3 c\lambda + K_2 K_4 = 0$$

Se ha comprobado que para cualquier condición de carga del sistema, la ecuación característica tiene como soluciones una raíz real negativa y dos raíces complejas conjugadas en el semiplano derecho.

Lamentablemente, como se observa, el modelo linealizado no brinda ninguna información adicional sobre los efectos del rozamiento en el desempeño del sistema; sin embargo, provee información sobre ciertas pautas para poder minimizar los efectos de la perturbación, y los cambios en los parámetros.

Si al sistema linealizado (en lazo abierto) se aplica la transformada de Laplace se obtienen las siguientes ecuaciones:

$$X_1(s) = \frac{K_1 \Delta P(s) + K_2 X_3(s)}{s(s + K_1 c)} \qquad \text{[ecuación 3.27]}$$

$$\frac{X_3(s)}{U(s)} = \frac{K_4}{s + K_3} \qquad \text{[ecuación 3.28]}$$

El diagrama de bloques descrito por las ecuaciones anteriores se muestra en la figura 3.26.

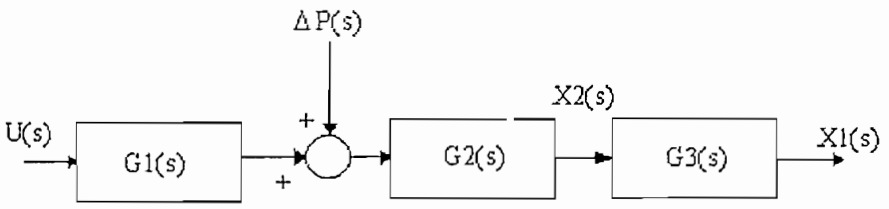


Figura 3.26 Diagrama de bloques del sistema con modelo de función de transferencia.

Donde, $G_1(s)$, $G_2(s)$ y $G_3(s)$ se definen como:

$$G_1(s) = \frac{K_2 K_4 / K_1}{(s + K_3)}$$

$$G_2(s) = \frac{K_1}{(s + K_1 c)}$$

$$G_3(s) = \frac{1}{s}$$

Reemplazando los parámetros por sus valores nominales se obtiene:

$$G_1(s) = \frac{12096.76085}{(s + 37.037)}$$

$$G_2(s) = \frac{0.00058204}{(s + 0.0000309)}$$

De aquí en adelante, en caso de que no se indique lo contrario, el modelo a emplearse será el nominal.

La perturbación $\Delta P(s)$, representa la diferencia de peso entre la cabina y el contrapeso. Esta perturbación permanece constante durante el recorrido del elevador, por lo que puede ser representada por una entrada de tipo paso.

Se pretende encontrar de que manera manera afecta esta perturbación al desempeño del sistema y como minimizarlo.

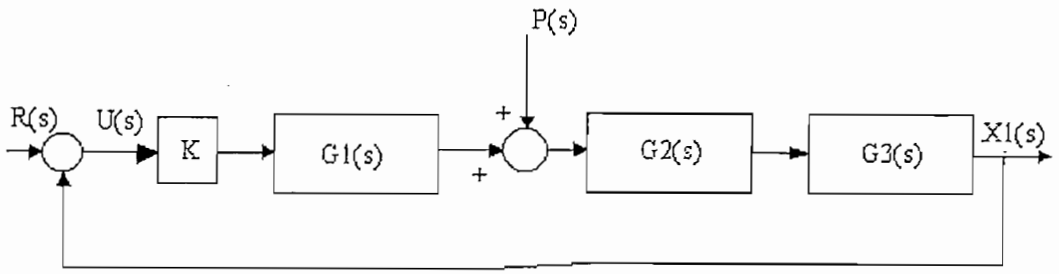


Figura 3.27 Sistema linealizado en lazo cerrado

Para encontrar la función de transferencia en lazo cerrado del sistema mostrado en la figura 3.27, se usa la regla de Mason y se encuentra que:

$$\frac{X_1(s)}{\Delta P(s)} = \frac{G_2G_3(s)}{1+KG_1G_2G_3(s)} = \frac{G_2G_3(s)}{1+L(s)}$$

$$T(s) = \frac{KG_1G_2G_3(s)}{1+KG_1G_2G_3(s)} = \frac{KG_1G_2G_3(s)}{1+L(s)}$$

Este resultado indica que es conveniente maximizar la ganancia $KG_1(s)$, para minimizar el efecto de la perturbación.

Como la planta posee parámetros variables, un análisis de sensibilidad de la planta con respecto a cambios en la planta $G(s)=G_1G_2G_3(s)$, y a cambios en la ganancia estática K es conveniente, y es así como:

$$S_G^T = \frac{G}{T} \cdot \frac{dT}{dG} = \frac{1}{1+L(s)}$$

$$S_K^T = S_G^T \cdot S_K^G$$

$$S_K^G = 1$$

$$\rightleftharpoons S_K^T = S_G^T$$

Puede entonces concluirse tres criterios que son útiles para el diseño de los controladores:

- Alta ganancia de lazo L(s) para minimizar S_G^T y S_K^T.
- Alta ganancia L(s) para minimizar el efecto de $\Delta P(s)$ en la salida.
- Ancho de banda suficiente, para obtener una buena reproducción de la entrada en la salida.

El Lugar Geométrico de las Raíces del sistema nominal, se muestran en la figura 3.28.

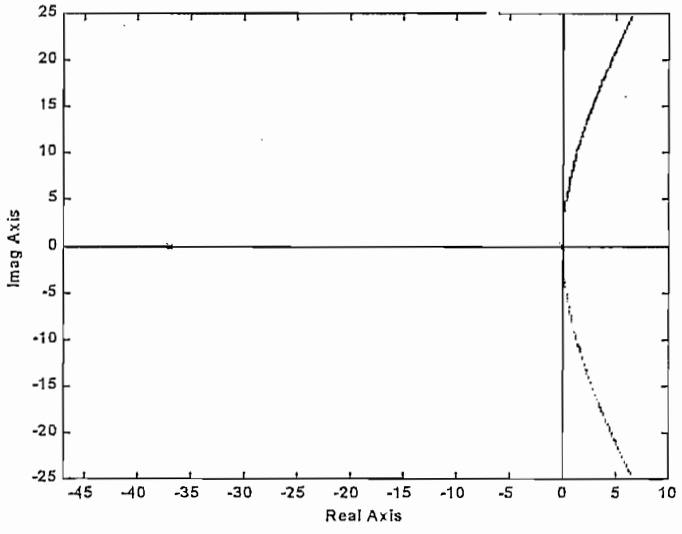


Figura 3.28 a)LGR del sistema del elevador sin compensación.

Existe un polo en el origen y otro muy cercano al mismo, por lo que el sistema posee una ganancia crítica, a partir de la cual es inestable, y que se la encuentra utilizando el criterio de Routh-Hurwitz, obteniéndose que $K_{crítico}$=0.042386 y el cruce del LGR con el eje imaginario es s = ±j $1.0697x10^{-5}$, por lo que el sistema linealizado es naturalmente inestable.

III.3.5 ANÁLISIS POR FUNCION DESCRIPTIVA

Para incluir los efectos no–lineales en el modelo linealizado, es posible encontrar la función descriptiva de la no-linealidad. Ésta representación permite establecer la existencia de ciclos límites, determinar su frecuencia, amplitud y estabilidad de los mismos. El diagrama de bloques del sistema, en estas circunstancias se muestra en la figura 3.29.

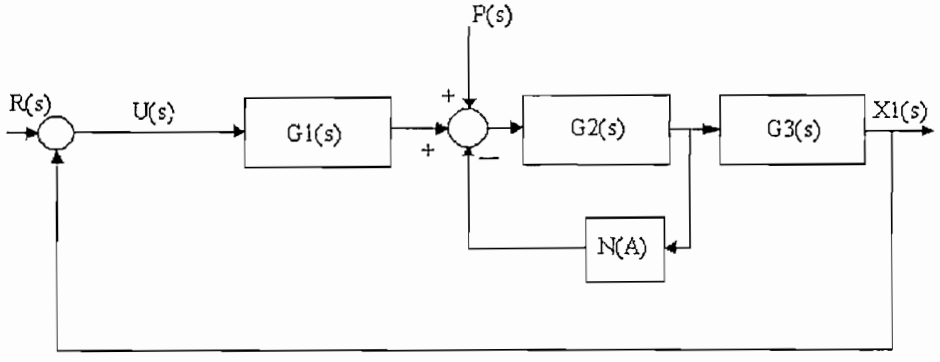


Figura 3.29 Diagrama de bloques del sistema, con representación de la no-linealidad con función descriptiva.

Donde:

- $G_1(s)$, $G_2(s)$, y $G_3(s)$, están definidas como en la sección anterior.
- N(A) es la función descriptiva de la fuerza de rozamiento de Coulomb.

La función descriptiva se obtiene calculando la amplitud de la primera armónica de la respuesta temporal de la no-linealidad a una entrada senoidal de la forma Asin(wt) (A>0), y diviendo para la amplitud de la entrada, por lo que primero es necesario encontrar la respuesta temporal de la no-linealidad, como sigue:

La fuerza de rozamiento de Coulomb se definió anteriormente como:

$$fr_K = \begin{cases} -Fr_K & \dot{x} < 0 \\ 0 & \dot{x} = 0 \\ Fr_K & \dot{x} > 0 \end{cases}$$

Ésta tiene una respuesta temporal a una entrada senoidal de amplitud A, como la mostrada en la figura 3.30.

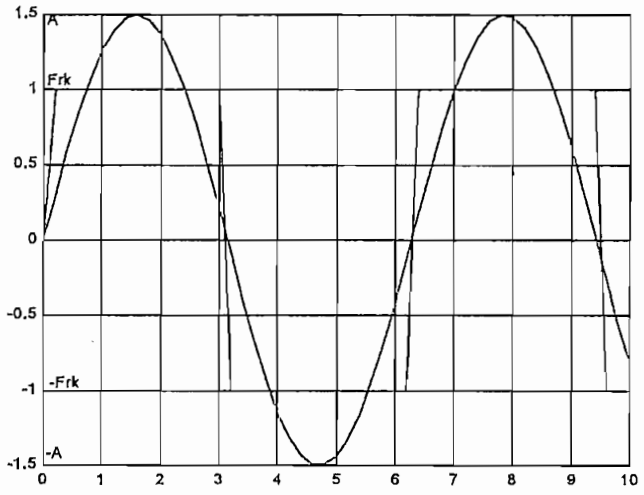


Figura 3.30 Respuesta temporal de la fuerza de rozamiento de Coulomb a una entrada senoidal Asin(wt).

Entonces, se puede proceder a calcular los coeficientes de Fourier.

Como la función es impar, el coeficiente de Fourier $a_1 = 0$, pero para el coeficiente de b_1:

$$b_1 = \frac{2}{T}\int_0^T Fr_K \cdot \sin(wt)\cdot d(wt) = \frac{1}{\pi}\left[\int_0^{\pi} Fr_K \cdot \sin(wt)\cdot d(wt) - \int_{\pi}^{2\pi} Fr_K \cdot \sin(wt)\cdot d(wt)\right]$$

$$b_1 = \frac{4\cdot Fr_K}{\pi}$$

La función descriptiva es entonces:

$$N(A) = \frac{4Fr_k}{\pi A}$$

Una vez encontrada la función descriptiva, se continua a encontrar la función de transferencia del sistema en lazo cerrado; que se realiza usando la regla de Mason en el diagrama de bloques de la figura 3.29, es así como:

$$T(s)=\frac{G_1\cdot G_2\cdot G_3}{1+G_1\cdot G_2\cdot G_3+N(A)\cdot G_2}$$

Dividiendo el numerador y denominador para $1+G_1G_2G_3$ se obtiene:

$$T(s)=\frac{\dfrac{G_1\cdot G_2\cdot G_3}{1+G_1\cdot G_2\cdot G_3}}{1+N(A)\cdot\left(\dfrac{G_2}{1+G_1\cdot G_2\cdot G_3}\right)}$$

En esta última expresión se define una función equivalente G_{eq} dada por:

$$G_{eq}=\frac{G_2}{1+G_1\cdot G_2\cdot G_3}(s)=\frac{K_1s(s+K_3)}{s^3+(K_1C+K_3)s^2+(K_1\cdot K_3\cdot C)s+K_2\cdot K_4}$$

Cuyo denominador, es idéntico al del sistema en lazo cerrado linealizado deducido en la sección anterior, y por lo tanto siempre tendrá una raíz real negativa y dos raíces complejas conjugadas en el semiplano derecho. Debido a éste particular el sistema no es de mínima fase por lo que el criterio de Nyquist de estabilidad no es aplicable.

Debido a que la función descriptiva es real, se puede realizar un LGR del sistema variando a N(A) de 0 a ∞. Entonces, la ecuación característica del sistema es:

$$1+N(A)\cdot G_{eq}(s)=s^3+(K_1C+K_3)s^2+(K_1\cdot K_3\cdot C)s+K_2\cdot K_4+N(A)\cdot K_1s(s+K_3)=0$$

Por facilidad de notación se hacen los siguientes reemplazos:

$$\overline{N}(A)=K_1\cdot N(A)$$

$$a=K_3$$

$$b=K_1C+K_3$$

$$c=K_1K_3C$$

$$d=K_2\cdot K_4$$

Sustituyendo estas equivalencias en la ecuación característica y reorganizando términos, la ecuación característica puede ser escrita con mayor facilidad como:

$$s^3+(b+\overline{N})s^2+(c+a\overline{N})s+d=0$$

El Lugar Geométrico para las condiciones de carga mínima, nominal y máxima, se muestra en la figura 3.31a, y un acercamiento al mismo en la figura 3.31b.

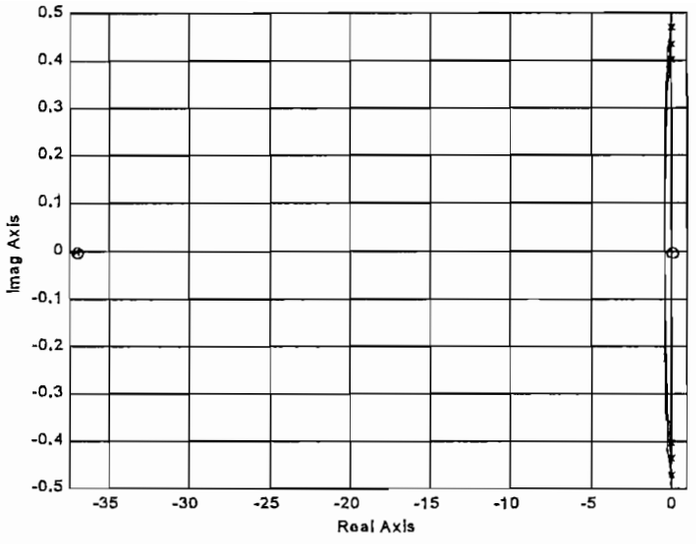


Figura 3.31a LGR de $1+NG_{eq}$ variando $\overline{N}$ de 0 a ∞, para carga mínima, nominal y máxima.

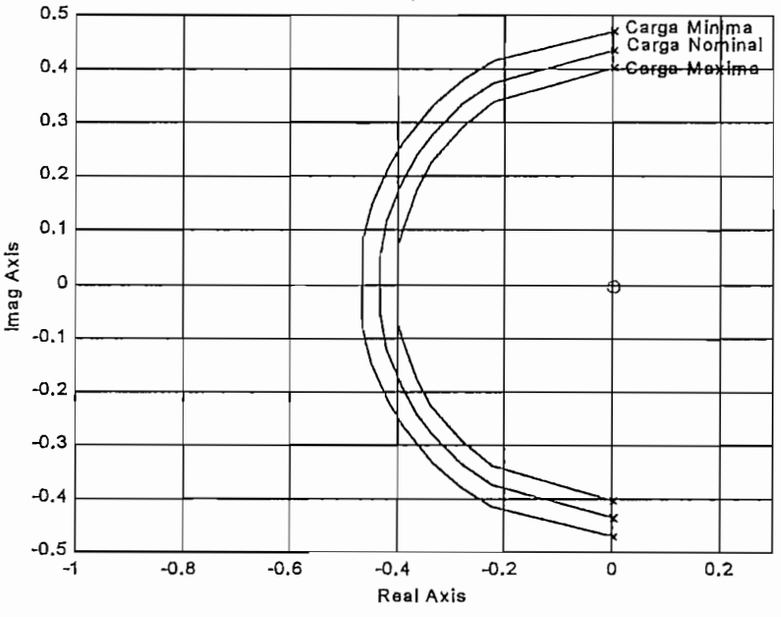


Figura 3.31b Acercamiento del LGR para cargas mínima,nominal y máxima variando $\overline{N}$ de 0 a ∞.

W (rad/s)	0.471151579	0.435973998	0.404482598

Tabla 3.4 Valores de la función descriptiva, para los cúales existe ciclo límite, y su frecuencia de oscilación.

El cruce del LGR con el eje imaginario es el punto en el cual existe un ciclo límite. Es interesante observar que si $\overline{N}$ aumenta por encima de $\overline{N}_{crítico}$, ya sea por una disminución de A o aumento de Fr_K, el sistema es asintóticamente estable, mientras que si $\overline{N}$ disminuye por debajo de $\overline{N}_{crítico}$, por aumento de A o disminución de Fr_K, el sistema es inestable. Estos resultados concuerdan con el análisis en el espacio de estado, donde al reducir la magnitud de la fuerza de rozamiento, el sistema exhibía en un ciclo límite. Además como el punto de cruce con el eje imaginario (jw) disminuye al aumentar la carga, la frecuencia de oscilación del ciclo también se reduce.

Para encontrar el valor de $\overline{N}$ con el que se alcanza esta condición ($\overline{N}_{crítico}$) se aplica el criterio de Routh-Hurwitz, y se encuentra que:

$$\overline{N}_{crítica}^2 + \left(\frac{ab+c}{a}\right)\cdot \overline{N}_{crítica} + \left(\frac{bc-d}{a}\right) = 0$$

De donde se debe escoger el valor positivo. Se ha calculado el $\overline{N}_{crítico}$, el $N_{crítico}$ y la frecuencia de oscilación del ciclo, para las tres condiciones de carga, que se tabulan a continuación:

	Mínima	**Nominal**	**Máxima**
$\overline{N}_{crítica}$	0.005957467	0.005101121	0.004390765
N	8.764030363	8.764235434	8.764405551
w (rad/s)	0.471151579	0.435975998	0.404482598

Tabla 3.4 Valores de la función descriptiva, para los cúales existe ciclo límite, y su frecuencia de oscilación.

En resumen, el sistema tiene un punto de equilibrio que depende tanto de la entrada de referencia como de la condición de carga. Dependiendo, de la magnitud de la fuerza de rozamiento, el sistema puede exhibir un ciclo límite inestable, cuya frecuencia puede ser determinada por un análisis de función descriptiva. Todo esto lleva a concluir que en la ausencia de fuerza de rozamiento de Coulomb el sistema oscila cuando existe cualquier perturbación.

CAPITULO IV

CONTROL Y SIMULACIÓN DEL ELEVADOR

IV.1 CONTROL DEL ELEVADOR

Se presentan dos técnicas convencionales de control para el elevador. Su propósito es el de comparar el procedimiento de diseño así como los resultados de desempeño de estas con las del control difuso. Para los diseños del control PID y realimentación de estado se considera para el diseño la planta linealizada con parámetros nominales.

IV.1.1 CONTROLADOR PID

Se ha escogido al control PID, debido a que muchos de los sistemas industriales y entre ellos los elevadores modernos utilizan este tipo de control. El control PID se caracteriza por ser de fácil ajuste y muy robusto, por lo que se presenta como una muy buena contraparte de comparación con el control difuso.

Existen muchas alternativas para el diseño de controladores PID. En este trabajo se realiza el diseño por medio de criterios de control óptimo empleando la minimización de u índice de funcionamiento. Se ha escogido el índice ITAE o integral del tiempo por error absoluto, cuya expresión es:

$$ITAE = \int_0^T t \cdot |e(t)| \cdot dt$$

Para obtener un funcionamiento óptimo, es requisito además que la función de transferencia en lazo cerrado tenga la siguiente forma: [5]

$$T(s) = \frac{Y(s)}{R(s)} = \frac{b_o}{s^n + b_{n-1}s^{n-1} + \cdots + b_1 s + b_o}$$

Debido a que la planta posee un polo en el origen y otro muy cercano a él, se puede utilizar un lazo secundario de realimentación de velocidad para mejorar sus características, como se muestra en la figura 4.1.

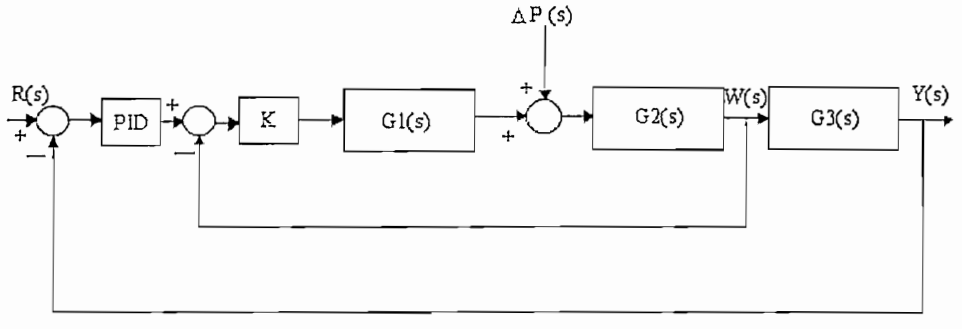


Figura 4.1 Esquema de control PID, con lazo secundario de realimentación.

Donde $G_1(s)$, $G_2(s)$ y $G_3(s)$ se definen como:

$$G_1(s) = \frac{12096.76085}{(s+37.037)}$$

$$G_2(s) = \frac{0.00058204}{(s+0.0000309)}$$

$$G_3(s) = \frac{1}{s}$$

$$PID = \frac{q_1 s^2 + q_2 s + q_3}{s}$$

y la ganancia K, se ha escogido igual a 42.608, lo que mueve los polos del lazo secundario a s_1=-11.966 y s_2=-25.071, mejorando las características del sistema. Bajo estas condiciones la función de transferencia de la planta es:

$$G(s) = \frac{299.99}{s(s+11.966)(s+25.071)}$$

Entonces, las variables desconocidas son los coeficientes del control PID. Para encontrar sus valores, el índice ITAE provee de los coeficientes óptimos de la ecuación característica en lazo cerrado. Entonces, el procedimiento de diseño consiste en encontrar la ecuación en lazo cerrado del sistema compensado e igualarlos con la ecuación de coeficientes óptimos.

Así entonces, la ecuación del sistema compensado es:

$$s^4 + 37.0370309 \cdot s^3 + (300 + 299.99 \cdot q_1) \cdot s^2 + 299.99 \cdot q_2 \cdot s + 299.99 \cdot q_3 = 0$$

La ecuación característica con coeficientes óptimos, de acuerdo con el índice ITAE, para un sistema de cuarto orden con entrada paso es:

$$s^4 + 2.1 \cdot \omega_n \cdot s^3 + 3.4 \cdot \omega_n{}^2 \cdot s^2 + 2.7\omega_n{}^3 \cdot s + \omega_n{}^4 = 0$$

Se requiere entonces que $2.1w_n \approx 37.0370309$, por lo que se escoge w_n=17.637, que reemplazando en la ecuación de coeficiente óptimos, se encuentra que la característica óptima es:

$$s^4 + 37.0370309 \cdot s^3 + 1057.5786 \cdot s^2 + 14812.0227 \cdot s + 96753.6765 = 0$$

Igualando coeficientes y despejando las constantes q_1, q_2, y q_3, se encuentra que el controlador PID es:

$$PID = \frac{2.5253 \cdot s^2 + 49.37 \cdot s + 322.523}{s}$$

Para que el funcionamiento sea óptimo, se debe cumplir la segunda condición: "la función de transferencia en lazo cerrado no debe tener ceros". Para cumplir con esta condición, se inserta un filtro de entrada tal que cancele a los ceros del PID y cuya ganancia en DC sea de 1. El filtro que cumple con estas características es:

$$Gp = \frac{322.523}{2.5253 \cdot s^2 + 49.37 \cdot s + 322.523}$$

El LGR del sistema compensado se muestra en la figura 4.2, donde se observa que el sistema ideal (sin rozamiento) será estable para cualquier valor de ganancia.

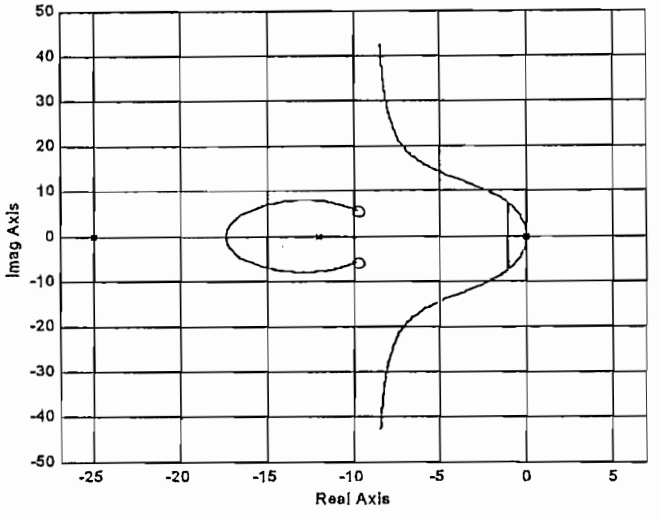


Figura 4.2 LGR del sistema con control PID, para carga nominal, sin rozamiento.

La respuesta de este control para una entrada paso unitario y sin rozamiento, considerando tres estados de carga, mínima, máxima y nominal se muestra en la figura 4.3.

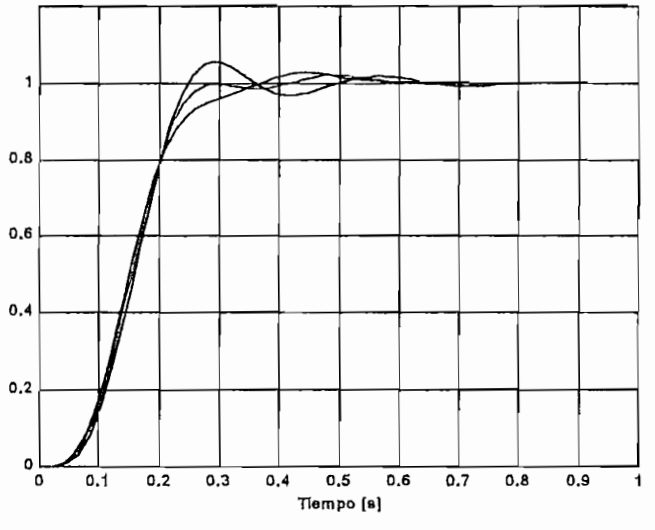


Figura 4.3 Respuesta del sistema con control PID, sin rozamiento, para cargas: mínima (azul), nominal (magenta), máxima (negro).

Se observa que el efecto de la perturbación sobre el desempeño del controlador es bastante incidente sobre todo en la magnitud del sobreimpulso. Uno de los inconvenientes del uso de este esquema es la magnitud de la señal de entrada a la planta, esta se presenta en la figura 4.4, para condición de carga máxima.

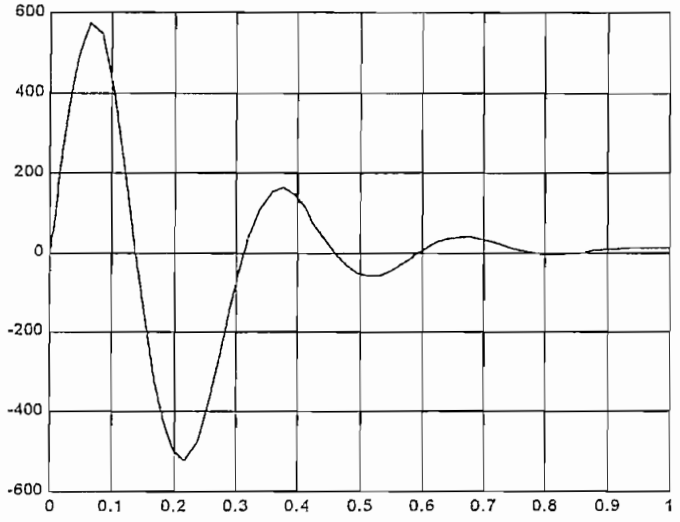


Figura 4.4 Señal de entrada a la planta para carga máxima, con control PID.

Falta ahora analizar los efectos de la fuerza de rozamiento. En la figura 4.5 se muestra la respuesta a una entrada paso unitaria, para las tres condiciones de carga, cuando la fuerza de rozamiento tiene una magnitud de 100 N, cuya magnitud es un estimado de la fuerza de rozamiento. Como se observa, el efecto de la fuerza de rozamiento de Coulomb es casi inapreciable, en gran medida por la robustez del control PID y por el tipo del sistema compensado, sin embargo, se observó que el efecto del rozamiento fue el de disminuir el sobreimpulso.

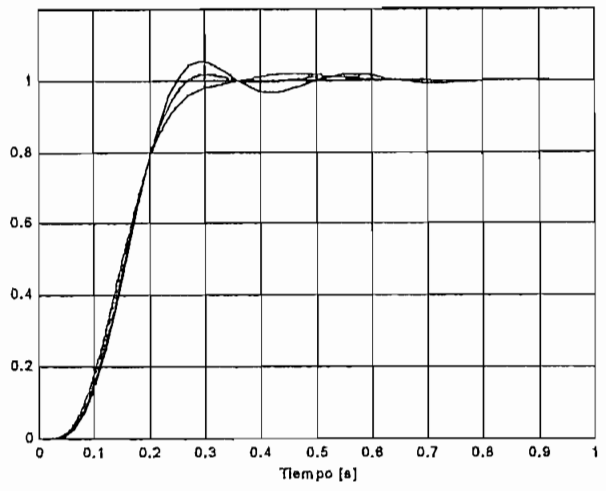


Figura 4.5 Respuesta del sistema con control PID, con rozamiento de Coulomb (Fr=100 [N]) para cargas: mínima (azul), nominal (magenta), máxima (negro).

Todas estas simulaciones brindan una buena idea de la calidad del sistema de control, sin embargo, es importante simularlo con la entrada de referencia que se desea que el sistema describa ya en operación, y que se introdujo en la sección III.1. La posición del sistema se presenta en la figura 4.6, para la condición de carga nominal y con una fuerza de rozamiento de 100 [N].

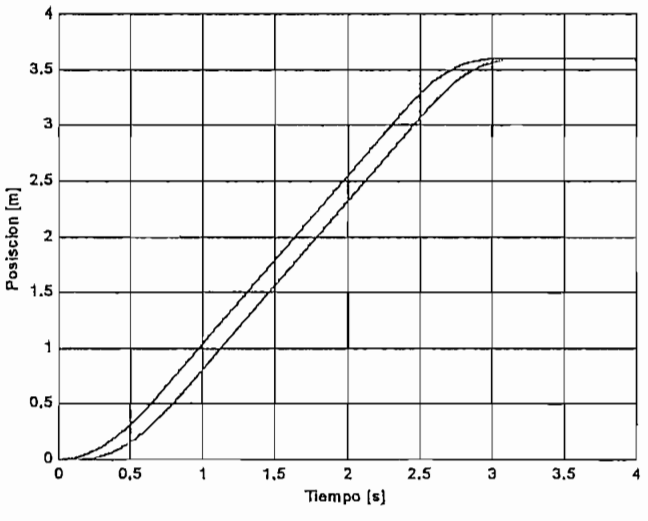


Figura 4.6 Posición de la cabina del elevador (magenta), Señal de referencia (negro).

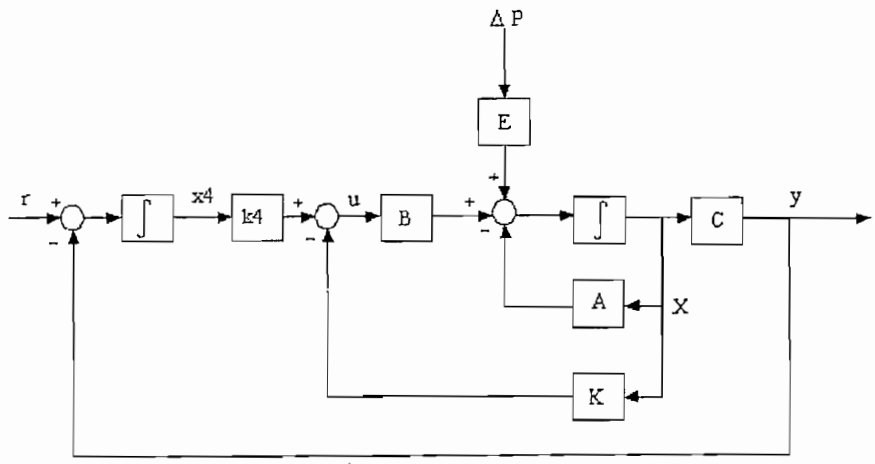


Figura 4.8 Esquema de control de realimentación de estado.

Donde:

$$A=\begin{pmatrix}0 & 1 & 0\\ 0 & -K_1c & K_2\\ 0 & 0 & -K_3\end{pmatrix}=\begin{pmatrix}0 & 1 & 0\\ 0 & -0.0000309 & 0.05069371\\ 0 & 0 & -37.037\end{pmatrix}$$

$$B=\begin{pmatrix}0\\ 0\\ K_4\end{pmatrix}=\begin{pmatrix}0\\ 0\\ 138.889\end{pmatrix}$$

$$C=\begin{pmatrix}1 & 0 & 0\end{pmatrix}$$

$$E=\begin{pmatrix}0\\ K_1\\ 0\end{pmatrix}=\begin{pmatrix}0\\ 0.00058204\\ 0\end{pmatrix}$$

$$K=\begin{pmatrix}k_1 & k_2 & k_3\end{pmatrix}$$

Las ecuaciones dinámicas del sistema son entonces:

$$\dot{X} = A \cdot X(t) + B \cdot u(t) + E \cdot \Delta P$$

$$x_4 = r(t) - y(t)$$

$$y(t) = C \cdot X(t)$$

$$u(t) = -K \cdot X(t) + k_4 \cdot x_4$$

Estas ecuaciones generan un sistema de orden n+1, cuya representación equivalente es:

$$\dot{\bar{X}} = \left(\bar{A} - \bar{B} \cdot \bar{K}\right) \cdot \bar{X} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} r(t) + \bar{E} \cdot \Delta P$$

Donde:

$$\bar{A} = \begin{pmatrix} A & 0 \\ -1 & 0 \end{pmatrix} \qquad \bar{B} = \begin{pmatrix} B \\ 0 \end{pmatrix} \qquad \bar{E} = \begin{pmatrix} E \\ 0 \end{pmatrix}$$

$$\bar{K} = \begin{pmatrix} K & -k_4 \end{pmatrix}$$

Los n+1 valores propios del sistema pueden ser colocados arbitrariamente si y solo sí el nuevo sistema es totalmente controlable. La matriz de controlabilidad MC es entonces:

$$MC = \left(\bar{B} \quad \bar{A} \cdot \bar{B} \quad \bar{A}^2 \cdot \bar{B} \quad \bar{A}^3 \cdot \bar{B}\right)$$

$$MC = \begin{pmatrix} 0 & 0 & 7.040 & -260.77 \\ 0 & 7.040 & -260.7702 & 9658.1448 \\ 138.889 & -5144.03 & 190519.509 & -7056271.063 \\ 0 & 0 & 0 & 7.040 \end{pmatrix}$$

Cuyo rango es 4, por lo que el sistema es controlable.

La ecuación característica del sistema se obtiene mediante:

$$|sI - \overline{A} + \overline{B} \cdot \overline{K}| = 0$$

$$s^4 + (37.0370309 + 138.889k_3)s^3 + (0.001144 + 0.00429k_3 + 7.040k_2)s^2 + 7.040k_1 s + 7.040k_4 = 0$$

Entonces, siguiendo un procedimiento similar al del controlador PID, se igualan los coeficientes a los de la ecuación característica óptima utilizada en el diseño anterior, obteniéndose que:

$k_1 = 2103.77$

$k_2 = 150.0748$

$k_3 = 0$

$k_4 = 13742.05356$

La figura 4.9 muestra la respuesta del sistema a una entrada paso unitaria para las tres condiciones de carga y despreciando la fuerza de rozamiento de Coulomb.

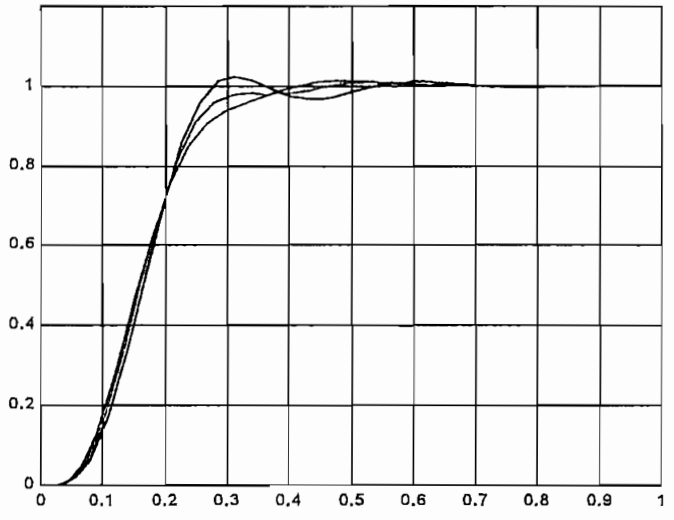


Figura 4.9 Respuesta del sistema a entrada paso unitaria para cargas:mínima (azul), nominal (magenta), máxima (negro), despreciando el rozamiento de Coulomb.

La respuesta de este control es similar a la del PID, pero logra disminuir el sobreimpulso aún más. Es de interés además observar la magnitud de la señal de control, que se muestra en la figura 4.10.

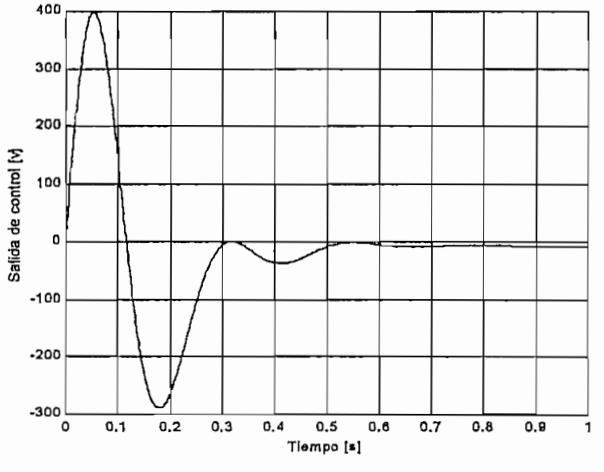


Figura 4.10 Salida de control para condiciones de carga mínima.

Al igual que en el control PID, la señal de control de la planta tiene una magnitud bastante alta.

Se incluye a continuación, el rozamiento de Coulomb, y se observa que sus efectos son mínimos, al igual que en el control PID. En la figura 4.10 se muestra la respuesta a una entrada paso unitaria del sistema para las tres condiciones de carga y donde se ha incluido una fuerza de rozamiento de 100 [N].

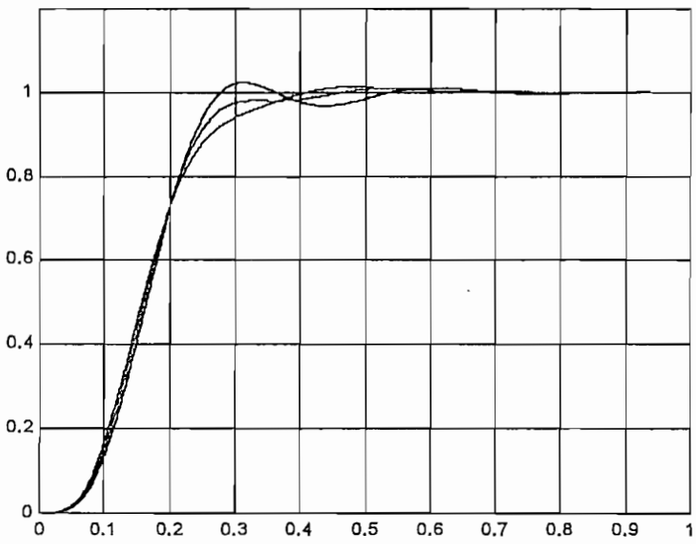


Figura 4.10 Respuesta del sistema con realimentación de estado, con rozamiento de Coulomb (Fr=100 [N]) para cargas: mínima (azul), nominal (magenta), máxima (negro).

Igualmente que en el caso del PID, es necesario observar la respuesta del sistema a la trayectoria deseada. Los gráficos de la posición y de la velocidad se muestran en la figura 4.11 y 4.12 respectivamente, para carga nominal.

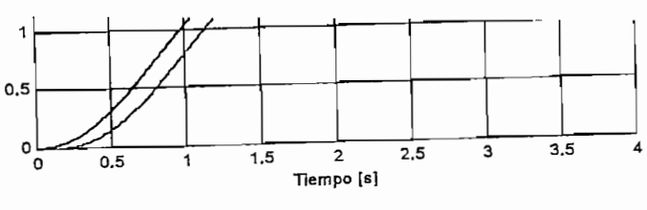
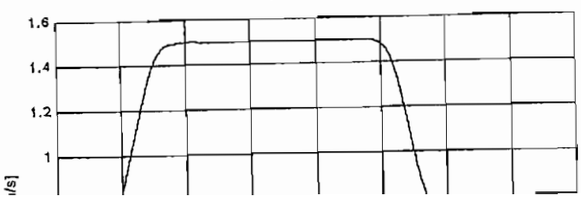


Figura 4.11 Posición de la cabina del elevador (magenta), Señal de referencia (negro).



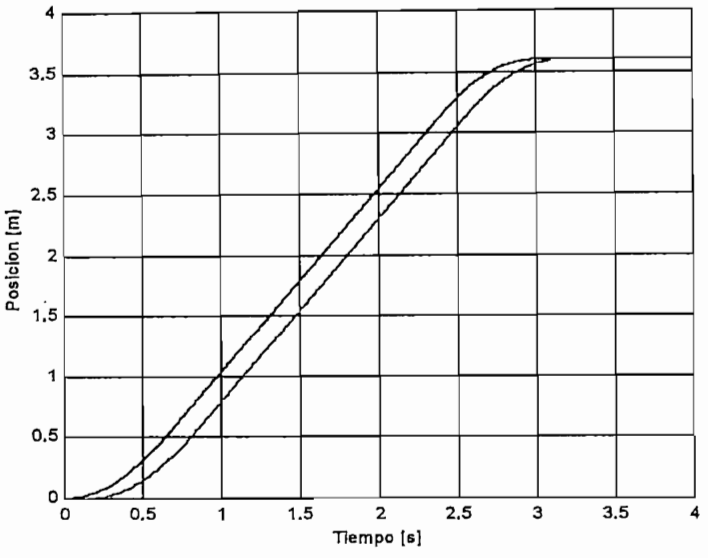


Figura 4.11 Posición de la cabina del elevador (magenta), Señal de referencia (negro).

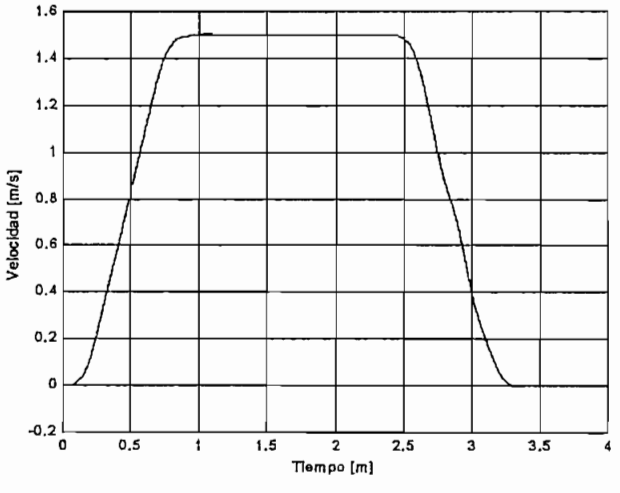


Figura 4.12 Velocidad de la cabina para carga nominal.

Se llevó a cabo además simulaciones del sistema con los demás estados de carga y se encontró que este no produce diferencias apreciables en la trayectoria de la posición así como de la velocidad.

IV.1.3 CONTROLADORES DIFUSOS

Se proponen en este trabajo tres controles difusos: PD, PI y PID. Los controles toman estos nombres debido a que utilizan como entradas al error, derivada del error e integral del error de posición. Los dos primeros realizan un mapeo sobre una superficie no-lineal tridimensional. El tercer control, debido a sus tres entradas, realiza un mapeo multidimensional. En el siguiente capítulo se presentan los resultados fundamentales que se obtuvieron con estos controles.

No existe un proceso sistemático para el diseño de un controlador difuso; sin embargo, cualquier intento de diseño debe estar precedido por cierto conocimiento heurístico del comportamiento de la planta que se quiere controlar.

En los estudios de las secciones anteriores, se ha podido deducir y recabar información que es de mucha utilidad para el diseño del controlador difuso. Así, se sabe que el sistema modelado es altamente inercial; es decir, el acelerar el sistema, cuando éste se encuentra cargado, es un trabajo bastante pesado para la máquina; más aún, una vez en movimiento, el frenar la máquina y su carga también es un proceso difícil que el control debe ser capaz de manejar. En cuanto a los efectos del rozamiento, de los controles antes diseñados, se observó que son despreciables.

Es ahora importante seleccionar las variables de control. Se presenta entonces las opciones de utilizar al error de posición, su derivada y su integral, y realizar un control PD, PI ó PID. En la figura 4.13, 4.14 y 4.15 se muestran los esquemas de los controles ensayados.

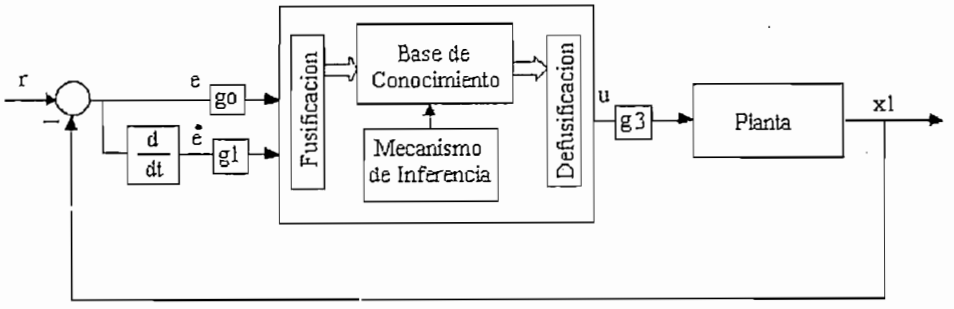


Figura 4.13 Control Difuso PD.

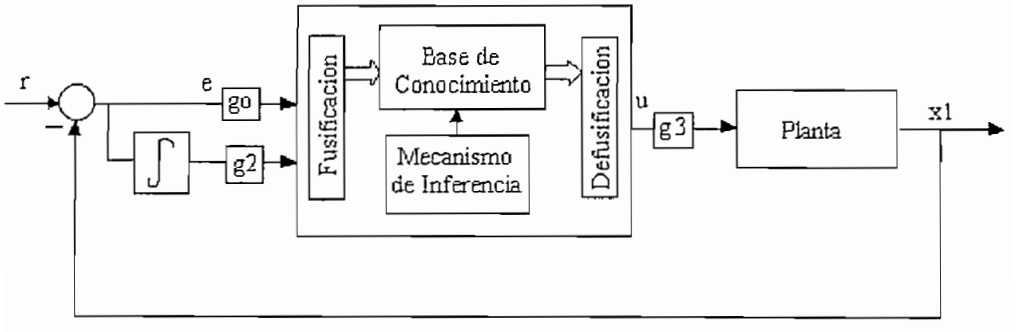


Figura 4.14 Control Difuso PI.

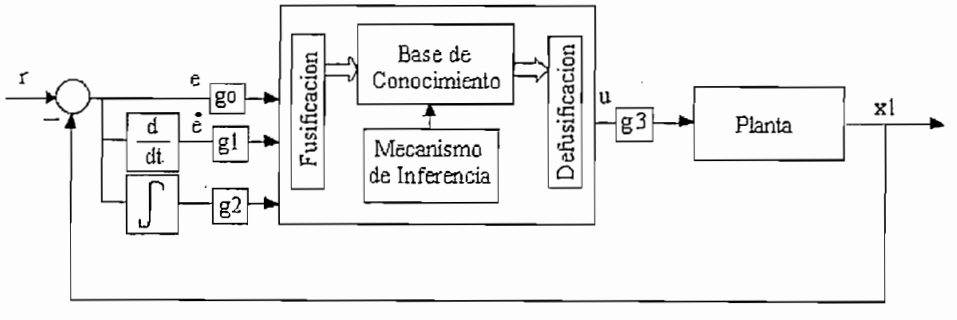


Figura 4.15 Control Difuso PID.

Las constantes g_o, g_1, g_2, y g_3 permiten ajustar la sensibilidad del control a cada una de las variables. El efecto de estas constantes sobre los dominios de las funciones de membresía de las variables es el siguiente:

- Si g_i es <1, el efecto es el de disminuir uniformemente el dominio de las funciones de membresía en un factor de g_i.
- Si g_i es >1, el efecto es el de aumentar uniformemente el dominio de las funciones de membresía en un factor de g_i.

Las funciones de membresía toman su forma segúr lo que se quiera expresar o como se lo quiera cuantificar. Para este caso, la cuantización del error de la posición, su derivada e integral pueden ser representados adecuadamente con funciones triangulares. El número de funciones de membresía tiene más bien que ver con cuantos niveles se desea cuantificar a las variables. Es conveniente; sin embargo, comenzar por un número bajo, e ir aumentando este de acuerdo con resultados de simulación o pruebas. Un mayor número de funciones de membresía posiblemente producirá mayor precisión, pero esta precisión se verá incrementada en tiempo de computo. Para estos diseños, se propone un número de siete funciones de membresía por variable, que para el caso de los controles PD y PI implica el tener una memoria asociativa (FAM) de 49 reglas, mientras que para el caso del PID se tienen 343 reglas.

CONTROL DIFUSO PD:

El primer paso para el diseño es el de dimensionar los universos de entrada y salida. Gran parte de este proceso, se realizó en base a resultados de simulación, hasta obtener los datos que se presentan, sin embargo, se parte fijando los límites laterales de acuerdo a un conocimiento del rango máximo en que deben variar las magnitudes, para luego ir ajustándolas.

Para ello, se definió primeramente un universo normalizado de –100 a 100, para luego definir el rango de cada una de las variables. Así, el universo de salida, correspondiente al voltaje aplicado a la máquina, puede variar en un rango máximo de entre –120 y 120 voltios. Si la referencia es positiva, los valores negativos de la salida se interpretan como frenado de la máquina. Para las variables de entrada se ha seleccionado que el universo del error comprenda un rango efectivo entre –0.1 y 0.1, y de su derivada en un rango de –1 a 1.

Entonces, las constantes de entrada g_0, g_1 y g_4, son 0.001, 0.01, y 1.0 respectivamente.
Las funciones de membresía de cada variable son:

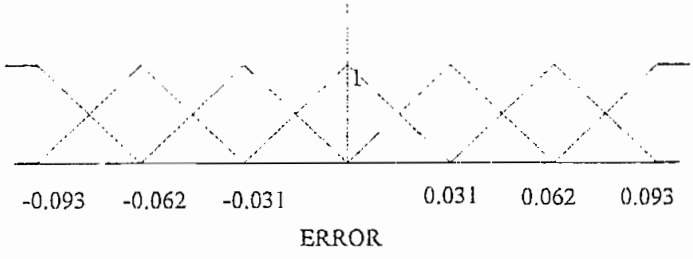


Figura 4.16 Funciones de membresía del error.

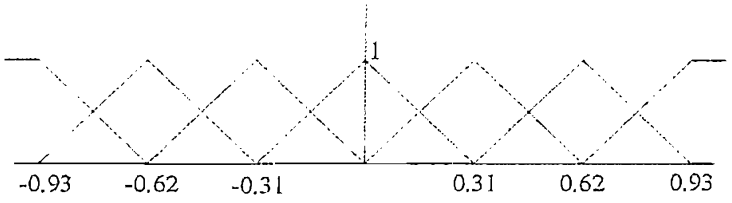


Figura 4.17 Funciones de membresía de la derivada del error.

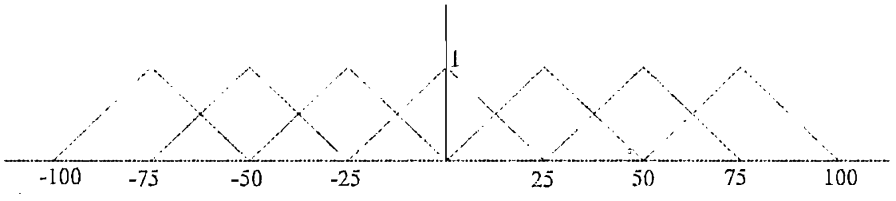


Figura 4.18 Funciones de membresía del voltaje.

Por facilidad numérica, se han etiquetado a las funciones de membresía por medio de índices positivos y negativos. La interpretación lingüística de estos índices se muestra en la siguiente tabla:

Negativo Grande	Negativo Mediano	Negativo Pequeño	Cero	Positivo Pequeño	Positivo Mediano	Positivo Grande
-3	-2	-1	0	1	2	3

Tabla 4.1 Interpretación lingüística de los etiquetas numéricas.

El siguiente paso consiste en establecer las reglas de control, que luego de algunos ensayos de simulación, la memoria asociativa (FAM) que mejores resultados demostró se muestra en la tabla 4.2.

Voltaje		Derivada del Error						
		-3	-2	-1	0	1	2	3
Error	-3	-3	-3	-3	-3	-3	-2	0
	-2	-3	-3	-3	-3	-2	0	2
	-1	-3	-3	-3	-2	0	2	3
	0	-3	-3	-2	0	2	3	3
	1	-3	-2	0	2	3	3	3
	2	-2	0	2	3	3	3	3
	3	0	2	3	3	3	3	3

Tabla 4.2 Memoria Asociativa (FAM) del control PD.

La superficie de control, bajo estas condiciones y utilizando el mínimo para la conjunción y defusicación por centro de gravedad se muestra en la figura 4.19.

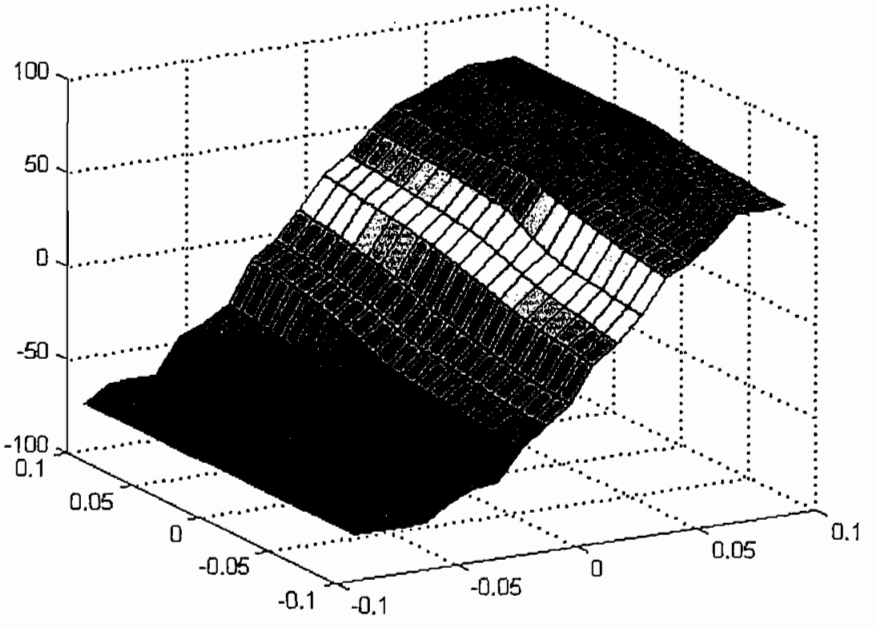


Figura 4.19 Superficie de control del control PD.

Queda ahora por realizar un análisis del sistema de control; sin embargo, los métodos de análisis de sistemas de control difuso son todavía muy limitados, en lo relativo a la teoría y la complejidad matemática. Adicionalmente, un sistema de control difuso como el propuesto, presenta dos no linealidades, además de tener múltiples entradas (MISO), razones por las cuales la mayoría de los métodos de análisis no son aplicables.

Debido a esto resta únicamente la simulación, que permite medir la calidad del sistema de control. A continuación se presentan los resultados de las simulaciones de este control a una entrada de tipo paso, para las tres condiciones de carga consideradas:

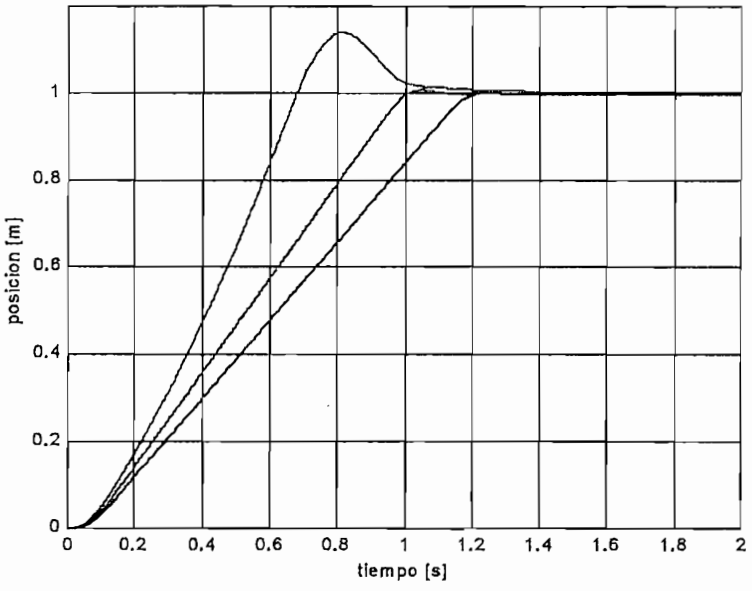


Figura 4.20 Respuesta del control difuso PD a una entrada paso. Carga mínima (azul), carga nominal (magenta), carga máxima (negro).

Muchos autores llaman al gráfico del error y su derivada como un plano de fase difuso. Este gráfico brinda una idea del orden en que se activan las reglas y cuales son a las que

más se accede. Este gráfico se muestra en la figura 4.21, para el mismo control PD antes mencionado, para carga máxima.

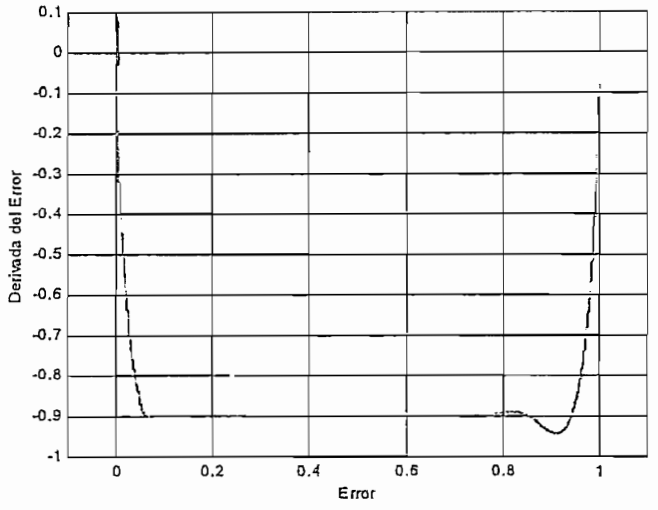


Figura 4.21 Plano de fase difuso, para el control PD y condiciones de carga máxima.

Se observa claramente que el error tiende a descender, por lo que converge rápidamente al centro de la memoria asociativa (FAM), mientras que su derivada pasa un buen tiempo en un valor de -0.9, considerado como un valor " Negativo Grande" (-3), para luego disminuir rápidamente a "cero" (0).

Por último, se realizaron pruebas de sensibilidad del control a variaciones tanto en las constantes de escalamiento g_0, g_1 y g_4 como a cambios en la memoria asociativa (FAM), y se encontró, que el sistema resulta ser muy sensible a cambios en la constante de la derivada del error g_1. En cuanto a la memoria asociativa el punto más sensible es el correspondiente a un error "Grande Positivo" (3) y una derivada del error "Negativa Grande" (-3), cuyos resultados se muestran en la figura 4.22.

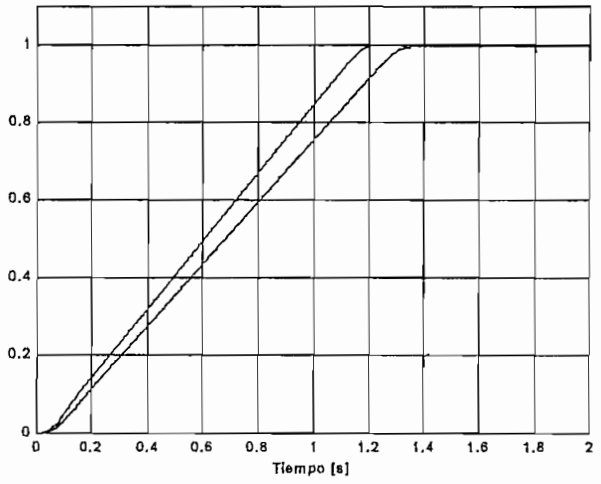


Figura 4.22 Respuesta a entrada paso, variando la FAM en la posición (3,-3). Carga mínima (azul), carga máxima (magenta).

La curva en azul representa la condición de carga mínima, mientras que las curva en magenta corresponde a la condición de carga máxima. Se observa que al reemplazar el conjunto inferido "Cero" (0) por un conjunto inferido de "Negativo Pequeño" (-1), el sobreimpulso es totalmente eliminado. Sin embargo, esto repercute en el desempeño de los otros estados de carga. Así por ejemplo, en el estado de carga máxima, el tiempo de subida se incrementa de 1.2 segundos a 1.3 segundos.

Debido a que el universo de salida del controlador difuso está limitado, la señal de control no debe exceder el rango establecido. Este resultado se muestra en la figura 4.23, donde se presenta la salida del controlador sometido a un paso unitario y para la condición de carga mínima.

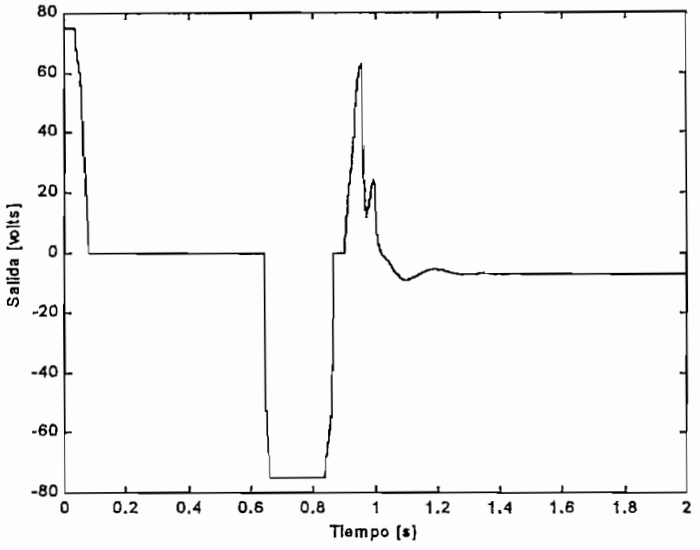


Figura 4.23 Salida del controlador difuso PD en condiciones de carga mínima.

En cuanto a la fuerza de rozamiento, en la figura 4.24 se muestran los resultados de simulación del sistema para las condiciones de carga mínima, máxima y nominal con una fuerza de rozamiento de 100 [N].

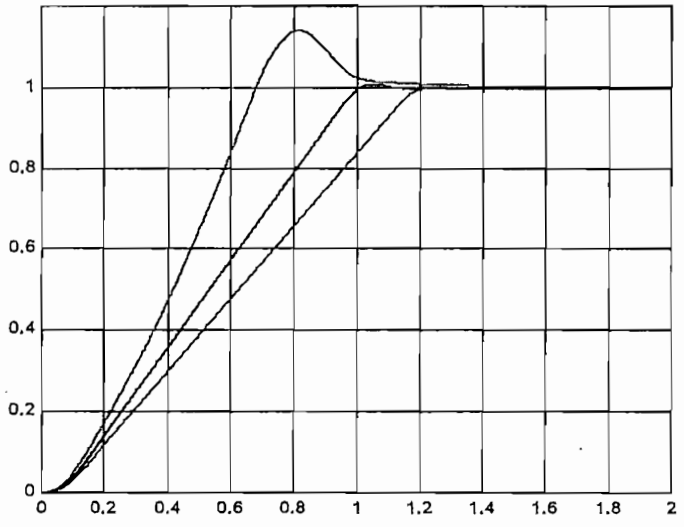


Figura 4.24 Respuesta del control difuso PD a una entrada paso. Carga mínima (magenta), carga nominal (negro), carga máxima (azul), con una fuerza de rozamiento de 100 [N] con el modelo de Coulomb.

Como se observa, igualmente que en los controles convencionales, los efectos del rozamiento son despreciables. Es importante, además ver el comportamiento del sistema a la señal de referencia deseada, y que se muestra en la figura 4.25 y 4.26, para la posición y la velocidad respectivamente.

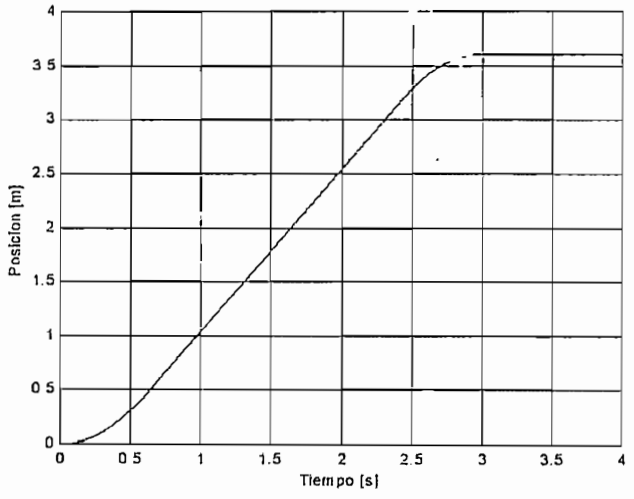


Figura 4.25 Posición de la cabina con control difuso PD, carga nominal y con la entrada de referencia de trayectoria deseada.

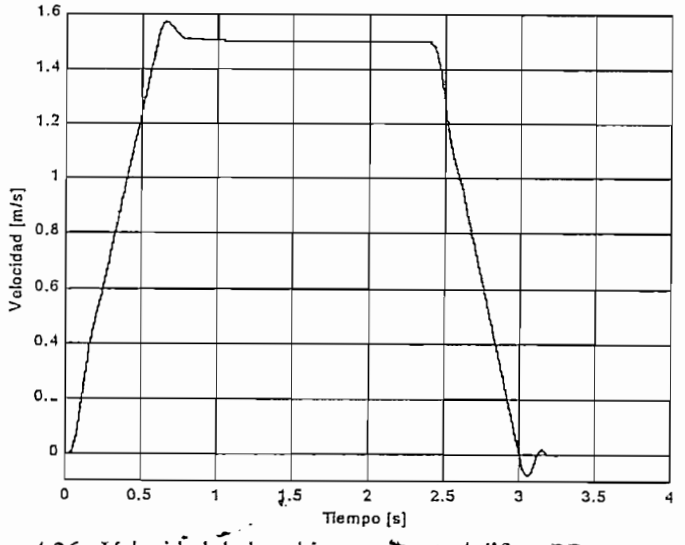


Figura 4.26 Velocidad de la cabina con control difuso PD, carga nominal y con la entrada de referencia de trayectoria deseada.

Es claro que la curva de posición sigue a la referencia de manera casi perfecta, pero la curva de velocidad deja ver que existen pequeños cambios de dirección en el movimiento, cuando la cabina se dispone a parar.

CONTROL DIFUSO PI:

El control difuso PI, mostró no ser adecuado para el control de este sistema, ya que no se logró estabilizarlo de ninguna forma. Sin embargo, este resultado no implica que no sea posible, simplemente indica que es muy difícil conseguirlo. La figura 4.25 muestra una respuesta típica del sistema bajo un control de este tipo.

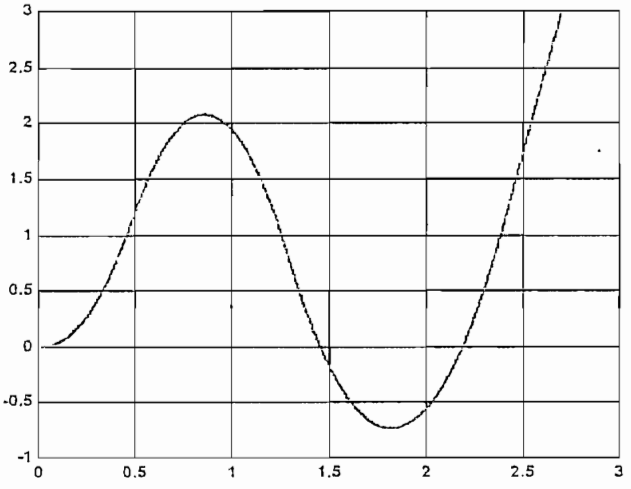


Figura 4.25 Respuesta del sistema con un control difuso PI.

CONTROL DIFUSO PID:

El control difuso PID, tiene tres entradas, el error , su derivada y su integral, lo que implica que en el caso de tener siete funciones por variable, se tendrán 343 reglas posibles. Esto le provee al diseñador de mayor información para emitir un criterio sobre el tipo de acción a realizar, pero a su vez el ajuste del controlador es mucho más laborioso.

Al igual que en los demás controles, primero se estable el rango efectivo del universo de cada variable. Se han seleccionado g_0=0.001, g_1=0.0098, g_2=0.1 y g_3=1.0, y siete funciones de membresía por variables de tal manera de tener los siguientes rangos:

error =]-0.1,0.1[

derivada del error =]-0.98, 0.98[

integral del error =]-10,10[

voltaje = [-100,100][1]

Las figuras 4.26, 4.27, 4.28, y 4.29 muestran la disposición de las funciones de membresía en sus respectivos universos.

[1] Los corchetes invertidos indican que se usan funciones de membresia que incluyen saturación.

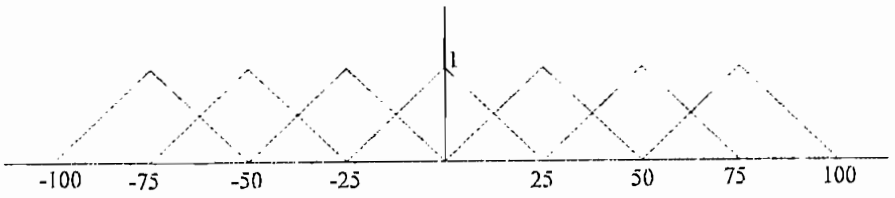


Figura 4.26 Funciones de membresía del voltaje de salida.

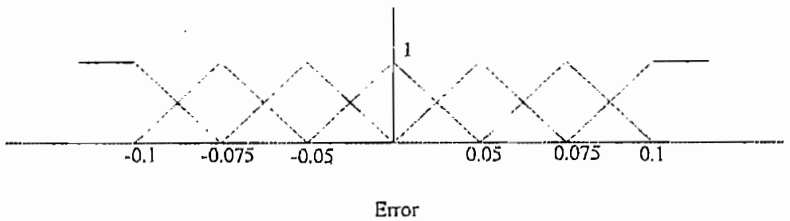


Figura 4.27 Funciones de membresía del error.

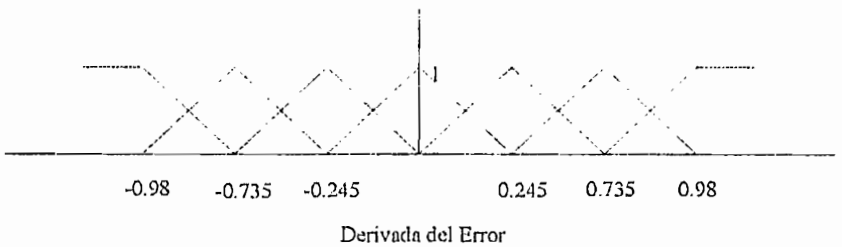


Figura 4.28 Funciones de membresía de la derivada del error.

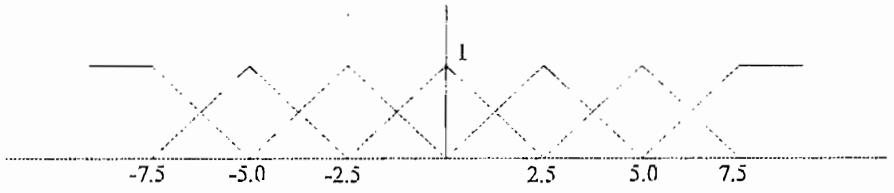


Figura 4.29 Funciones de membresía de la integral del error.

Para llenar la memoria asociativa (FAM), se han realizado algunos ensayos hasta que se logró obtener una con resultados muy parecidos a los del control PD. Debido a que con tres variables de entrada se tiene un mapeo multidimensional, es necesario descomponer a la memoria asociativa en 7 matrices, que se muestran en las tablas 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, y 4.9.

Integral del Error "-3"		Derivada del Error						
		-3	**-2**	**-1**	**0**	**1**	**2**	**3**
Error	**-3**	-3	-3	-3	-3	-3	-3	0
	-2	-3	-3	-3	-3	-3	0	1
	-1	-3	-3	-3	-3	0	1	1
	0	-3	-3	-3	0	0	1	2
	1	-3	-3	0	1	1	1	2
	2	-3	0	1	1	1	2	3
	3	0	1	1	1	2	3	3

Tabla 4.3 Memoria Asociativa (FAM) del control PID.

Integral del Error "-2"		Derivada del Error						
		-3	-2	-1	0	1	2	3
Error	-3	-3	-3	-3	-3	-3	-2	0
	-2	-3	-3	-3	-3	-2	0	1
	-1	-3	-3	-3	-2	0	1	1
	0	-3	-3	-2	0	1	1	2
	1	-3	-2	0	1	1	2	2
	2	-2	0	1	1	2	2	3
	3	1	1	1	2	2	3	3

Tabla 4.4 Memoria Asociativa (FAM) del control PID.

Integral del Error "-1"		Derivada del Error						
		-3	-2	-1	0	1	2	3
Error	-3	-3	-3	-2	-2	-2	-1	0
	-2	-3	-2	-2	-2	-1	0	1
	-1	-2	-2	-2	-1	0	1	1
	0	-2	-2	-1	0	1	1	2
	1	-2	-1	0	1	1	2	2
	2	-1	0	1	1	2	2	3
	3	0	1	1	2	2	3	3

Tabla 4.5 Memoria Asociativa (FAM) del control PID.

Integral del Error "0"		Derivada del Error						
		-3	-2	-1	0	1	2	3
Error	-3	-3	-3	-3	-3	-3	-2	0
	-2	-3	-3	-3	-3	-2	0	3
	-1	-3	-3	-3	-3	0	2	3
	0	-3	-3	-2	0	2	3	3
	1	-3	-2	0	3	3	3	3
	2	-2	0	2	3	3	3	3
	3	0	2	3	3	3	3	3

Tabla 4.6 Memoria Asociativa (FAM) del control PID.

Integral del Error "1"		Derivada del Error						
		-3	-2	-1	0	1	2	3
Error	-3	-3	-3	-3	-3	-2	-1	0
	-2	-3	-3	-3	-3	-1	0	2
	-1	-3	-3	-3	-3	0	2	3
	0	-3	-3	-3	0	2	3	3
	1	-3	-3	0	2	3	3	3
	2	-1	0	2	3	3	3	3
	3	0	2	3	3	3	3	3

Tabla 4.7 Memoria Asociativa (FAM) del control PID

Integral del Error "2"		Derivada del Error						
		-3	-2	-1	0	1	2	3
Error	-3	-3	-3	-3	-3	-2	-1	0
	-2	-3	-3	-3	-3	-1	0	1
	-1	-3	-3	-3	-1	0	1	2
	0	-3	-3	-1	0	1	2	2
	1	-3	-3	0	1	2	2	2
	2	-1	0	1	2	2	2	3
	3	0	1	2	2	2	3	3

Tabla 4.8 Memoria Asociativa (FAM) del control PID

Integral del Error "3"		Derivada del Error						
		-3	-2	-1	0	1	2	3
Error	-3	-3	-3	-2	-1	-1	-1	0
	-2	-3	-3	-3	-1	-1	0	1
	-1	-2	-3	-3	-1	0	1	2
	0	-1	-1	-1	0	2	3	3
	1	-1	-1	0	2	3	3	3
	2	-1	0	1	3	3	3	3
	3	0	1	2	3	3	3	3

Tabla 4.9 Memoria Asociativa (FAM) del control PID

En estas condiciones, en la figura 4.30 se muestra la respuesta del sistema a una entrada paso, sin rozamiento, para las tres condiciones de carga, y donde se ha escogido el mínimo para la conjunción y el método de defusificación por centro de gravedad.

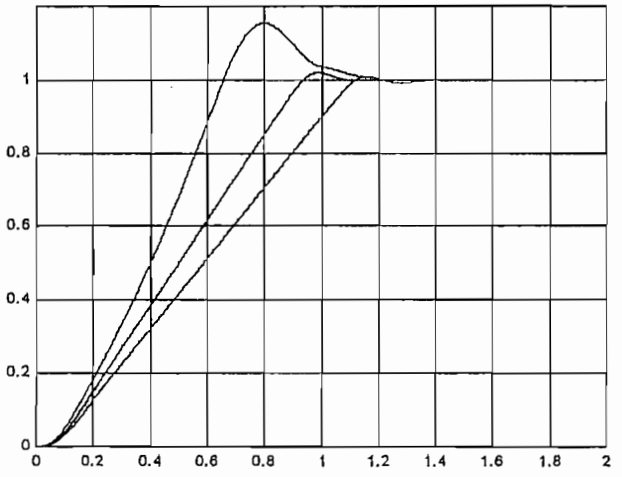


Figura 4.30 Respuesta a entrada paso, para carga mínima (azul), nominal (negro) y máxima (magenta), sin rozamiento.

Se observa que el sistema, bajo este tipo de control, tiene una respuesta muy parecida al control PD. La señal de control a la planta, bajo este esquema de control también posee una salida limitada, como se muestra en la figura 4.31.

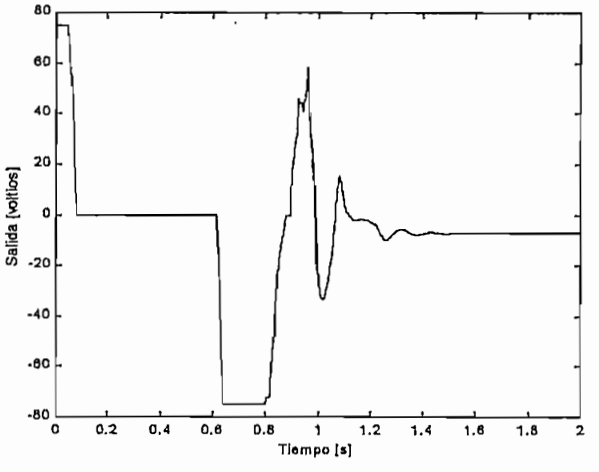


Figura 4.31 Salida de control del PID difuso para carga mínima.

Un análisis de sensibilidad de la Memoria Asociativa (FAM) , mostró resultados muy parecidos a los del control PD, cuando la integral del error cae sobre el conjunto "Positivo Pequeño" (1) (tabla 4.7), y el error es "Positivo Grande" (3), y la derivada del error es "Negativa Grande" (-3), entonces cualquier cambio en esta celda produce efectos muy apreciables sobre el sobreimpulso del sistema como se puede observar en la figura 4.32.

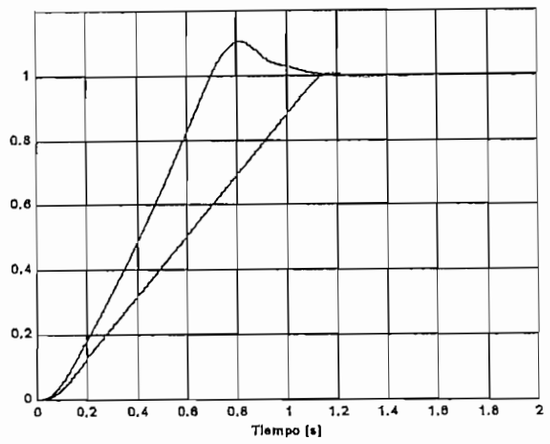
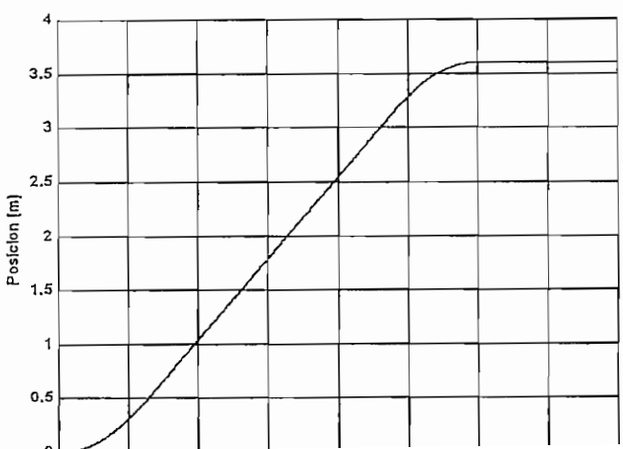


Figura 4.32 Respuesta a entrada paso, variando la FAM en la posición (3,-3,1) a -1, para carga mínima (azul) y máxima (magenta).

Es bastante notorio, que la sensibilidad a cambios en esta posición en la Memoria Asociativa (FAM) ha disminuido bastante con respecto al control PD.

Nuevamente, para tener una idea de la secuencia de activación de las reglas, así como para saber cuales son las que más se activan es útil visualizar la trayectoria que describen las variables de entrada al sistema de lógica difusa, esta vez en un espacio tridimensional. La figura 4.33 muestra la trayectoria para el caso de carga mínima.



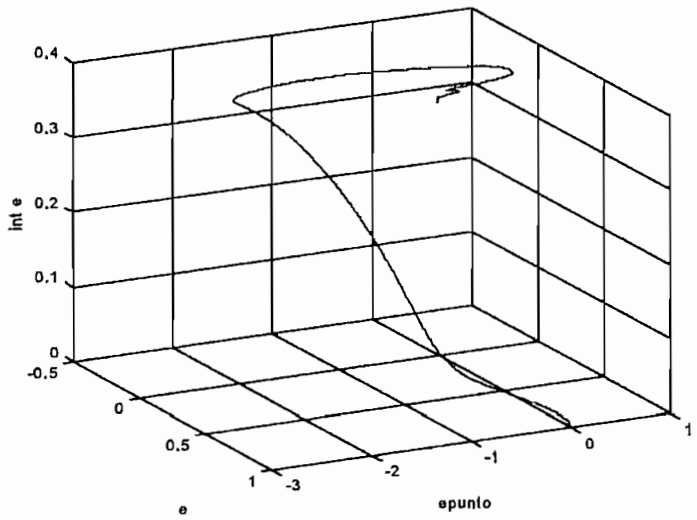


Figura 4.33 Trayectoria del error, derivada del error e integral del error, para el caso de carga mínima.

Se ha realizado también una simulación del sistema sometido a la señal de referencia con la trayectoria deseada, y se presentan los resultados en las figura 4.34 para la posición y en la figura 4.35 para la velocidad.

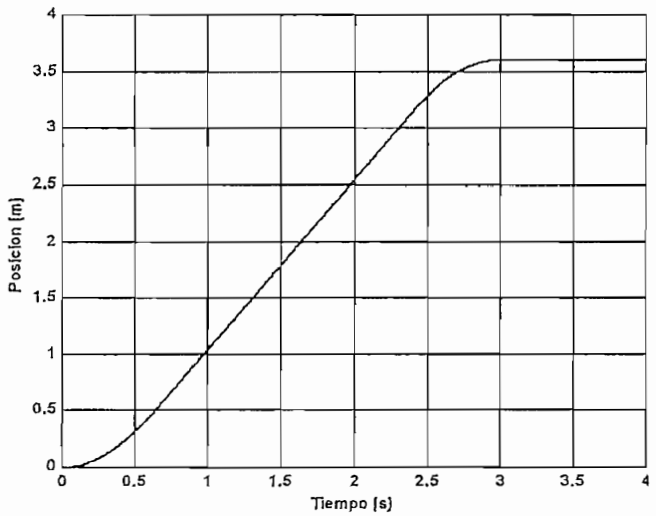


Figura 4.34 Posición de la cabina con señal de referencia deseada para carga nominal.

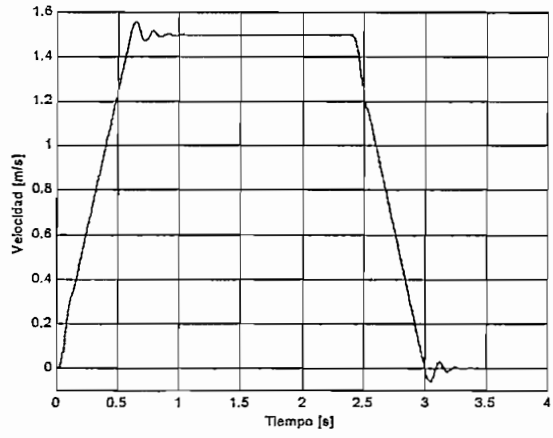


Figura 4.35 Velocidad de la cabina con señal de referencia deseada para carga nominal.

Se ve claramente que el seguimiento es casi perfecto para la posición, mientras que para la velocidad se presentan pequeñas oscilaciones al final de la rampa de ascenso y cuando el elevador se detiene.

Para finalizar este análisis, queda por visualizar los efectos del rozamiento. La figura 4.34 muestra la respuesta del sistema con carga mínima, nominal y máxima para una fuerza de rozamiento de 100 [N], y es claro que sus efectos son despreciables.

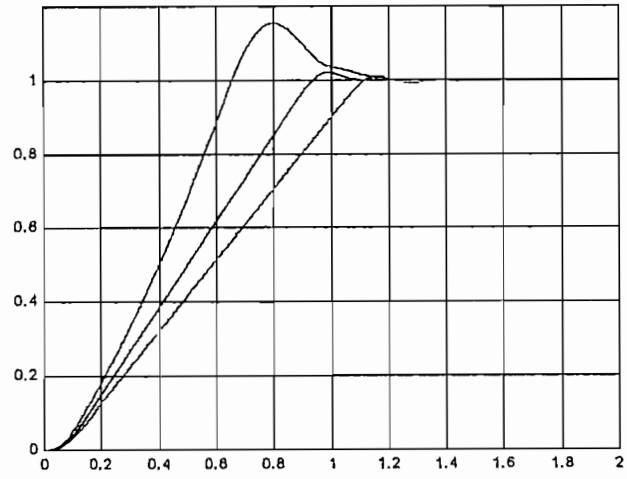


Figura 4.34 Respuesta del sistema a entrada paso, para carga mínima (azul), nominal (negro), y máxima (magenta), con una fuerza de rozamiento de 100 [N].

En cuanto a los métodos de conjunción, y defusicación. En el capítulo II, se mostró que cuando se utiliza el producto para la conjunción, el método del centro de gravedad produce el mismo resultado que el método del centro promedio, ya sea que se utilice el mínimo o el producto en este último. Con este precedente, el único cambio significativo podría darse en el caso de que se utilice el mínimo como operador de la conjunción en el método del centro de gravedad. Las pruebas de simulación mostraron que las diferencias fueron insignificantes, casi inapreciables, por lo que no se las muestra.

Antes de terminar esta sección, es importante aclarar que posiblemente se pueden obtener mejores desempeños de los esquemas de control difuso ensayados calibrando de mejor manera los parámetros del sistema utilizando algoritmos computacionales de ajuste. [19]

IV.2 PROGRAMA DE SIMULACIÓN GRÁFICA EN WINDOWS

IV.2.1 PAUTAS GENERALES

El programa de simulación gráfica para controles difusos, contiene tres tipos de control. Estos controles son: PD, PI y PID. Tanto el control PD como el PI, tienen un máximo de cuarenta y nueve reglas, y el PID tiene un número máximo de trescientas cuarenta y tres reglas, lo que brinda al diseñador de un campo bastante amplio de posibles modificaciones.

Además de las reglas de conocimiento, el programa de simulación gráfica es lo suficientemente versátil para poder realizar cambios en los demás parámetros, así por ejemplo: disposición de las funciones de membresía, magnitudes de los universos de entrada y salida, y operadores para la conjunción, implicación y defusificación. Dentro del gran número de posibles alternativas que se presentan para cada uno de los parámetros de los sistemas de lógica difusa (SLD) discutidos en el capítulo II, los siguientes han demostrado ventajas sobresalientes para aplicaciones de control, y se han implementado en este trabajo.

Funciones de Membresía: Utiliza funciones de membresía triangulares, cuyo número puede ser de 3, 5 o 7 por variable. Permite variar la forma de cada función de membresía, e introducir un escalamiento a los dominios de las funciones de membresía.

Operadores de Conjunción e Implicación: Los dos operadores más comunes en aplicaciones de control para la conjunción y la implicación son el mínimo y el producto.

Métodos de Defusificación: Para sistemas de control difusos continuos los dos métodos comúnmente empleados que son el Centro de Gravedad (COG) y el Centro Promedio (COA).

Mediante todas estas alternativas, se espera proveer al usuario de la mayor cantidad de herramientas para que pueda seleccionar y simular rápidamente cualquier control difuso de los tipos antes señalados.

El problema abordado en esta tesis, ha mostrado que los parámetros de la planta utilizada son dependientes de sus condiciones de carga. En esta medida, el programa de simulación gráfica permite al usuario ingresar la condición de carga del sistema, y el modelo es recalculado una vez que se ejecuta la simulación. Otro componente son los efectos de las fuerzas de rozamiento, para ello se ha incluido la posibilidad de usar los modelos de Coulomb y el de Stribeck, para los cuales se permite variar su magnitud, a través de la componente normal.

Con todas estas consideraciones se cree que se abarca un marco bastante amplio para el diseño y simulación de controladores difusos aplicados a elevadores.

IV.2.2 SELECCIÓN DEL COMPILADOR

De todos los requisitos antes mencionados y de la gran cantidad de cálculos involucrados se desprende dos conclusiones importantes para la selección del compilador:

- Debe ser capaz de realizar una gran cantidad de cálculos de una manera eficiente y rápida.
- Debe proveer de facilidades para programación gráfica en Windows.

Existen algunas alternativas para el desarrollo del sistema que cumplen con estas dos características. Primeramente, uno de los compiladores más eficientes y rápidos es el C++

de Borland o el Visual C++ de Microsoft. La diferencia está en que la versión de Borland compila aplicaciones en 16 bits y Visual C++ en 32 bits, mejorando así su velocidad de ejecución y además de que Visual C++ provee de las librerías con las clases fundamentales de Microsoft (MFC) para programación en Windows, que podrían sobrellevar la segunda característica necesaria.

Una segunda alternativa se refiere a que en los últimos años se han presentando entornos de programación especiales para el desarrollo de aplicaciones multimedia. Uno de los más utilizados es Director de Macromedia. Este entorno de programación contiene un lenguaje híbrido llamado Lingo que brinda la capacidad recursos gráficos interactivos y de fácil uso. En general Lingo es un híbrido entre Visual Basic y Visual C++. Pero su gran limitación es las de no poder llevar a cabo cálculos tan sofisticados como los que se necesitan.

Una última alternativa es la de utilizar MATLAB, que posee bibliotecas para la generación de interfaces gráficas e incluye además una gran cantidad de algoritmos para simulación de sistemas de control. Sin embargo, su principal desventaja es la de que el usuario deberá tener instalado MATLAB y el Toolbox de Control Fuzzy para poder ejecutar este programa.

En base a estas alternativas se decidió utilizar una mezcla, en el sentido del uso de Director y Lingo para el manejo de las interfaces gráficas con el usuario y la presentación de la

animación, mientras que para los cálculos se usa C++ de Borland versión 4.0. Es importante señalar además que el no uso de MATLAB se debió también a pruebas comparativas en la velocidad de simulación, en las cuales MATLAB presentó grandes tiempos de simulación.

Si bien C++ es un lenguaje orientado a objetos, no es necesario recurrir a este tipo de programación, ya que la programación estructurada es lo suficientemente eficiente. El programa en Director y Lingo, debido a la complejidad en el manejo gráfico, sí requiere de programación orientada a objetos, por lo que no tiene una secuencia de ejecución global determinada.

El control del sistema está a cargo del programa realizado en Director, quien llama al programa en C++ para ejecutar los cálculos y entregar los resultados. El enlace entre los dos programas se hace en base de dos archivos de textos, como se muestra en la figura 4.35. El primero denominado fuzzyin.txt, que contiene toda la información ingresada por el usuario, y construido por el programa en Lingo. El segundo archivo de texto, denominado fuzzyout.txt, entrega siete columnas de datos, con los valores del tiempo, posición , velocidad, señal de control, error de posición, derivada del error, e integral del error; que a su vez son leídos por el programa en Director para graficar las curvas y llevar a cabo la animación en tiempo real. El formato del archivo fuzzyin.txt se muestra en el anexo.

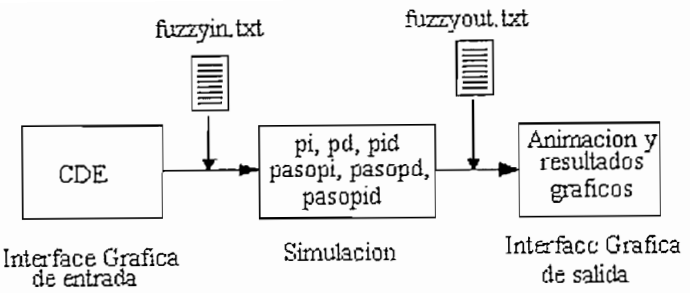


Figura 4.35 Estructura del programa de simulación gráfica.

Además, para obtener una mejor representación gráfica de las curvas, de posición, velocidad, error, derivada del error, integral del error, y salida del sistema de lógica difusa se han escrito las siguientes rutinas .m para MATLAB:

fpos.m: Gráfica de posición en función de la velocidad.

fvel.m: Gráfica de velocidad en función de la velocidad.

pfd2d.m: Gráfica del plano de fase difuso para controles PD y PI.

pfd3d.m: Gráfica del plano de fase difuso para control PID.

Los requerimientos mínimos del sistema para poder ejecutar el programa son:

- CD-ROM: Para poder cargar el programa.
- Pentium de 200 MHz recomendable.
- 16 Mb de memoria RAM.
- 20 MB de memoria libre en disco duro.
- Microsoft Windows 95
- DOS 6.0 o superior

Para el cálculo de la salida de control se utiliza el siguiente pseudocódigo:

1. Obtener los valores de e, $\dot{e}$, y $\int e$.
2. Calcular e.gdm[i], epunto.gdm[j], inte.gdm[k] para todo i, j, k.

   (Encontrar los grados de membresía para todas las funciones)
3. Calcular premisa[i,j,k] para todo i, j, k.

   (Encontrar el valor de la premisa ya sea por mínimo o por producto).
4. Calcular implicación[i,j,k]=areaimplicada[regla[i,j,k]], premisa[i,j,k] para todo i, j, k.
5. numerador = 0, denominador = 0.
6. Calcular numerador y denominador. (COG o COA)
7. u = numerador/denominador.

Los diagramas de flujo más importantes de las funciones para la simulación del sistema se incluyen en el anexo al igual que su código en C++.

CAPITULO V

RESULTADOS Y CONCLUSIONES

V.1 RESULTADOS

Primeramente se presentan los resultados referentes al desempeño de los controladores difusos, para luego realizar una comparación de estos con los otros dos controles.

V.1.1 CONTROLES DIFUSOS

De los controles difusos ensayados, únicamente el control PI mostró no ser apropiado para el control del elevador, ya que su respuesta fue siempre inestable, por lo que no se lo toma en cuenta para las siguientes comparaciones.

El primer factor importante a analizar entre los otros dos esquemas de control, es decir PD y PID, es su calidad. Para medir la calidad de cada uno de estos controles se utilizó su respuesta a una señal paso. En las figura 5.1, 5.2 y 5.3 se muestran estas respuestas para las tres condiciones de carga, es decir mínima, nominal y máxima.

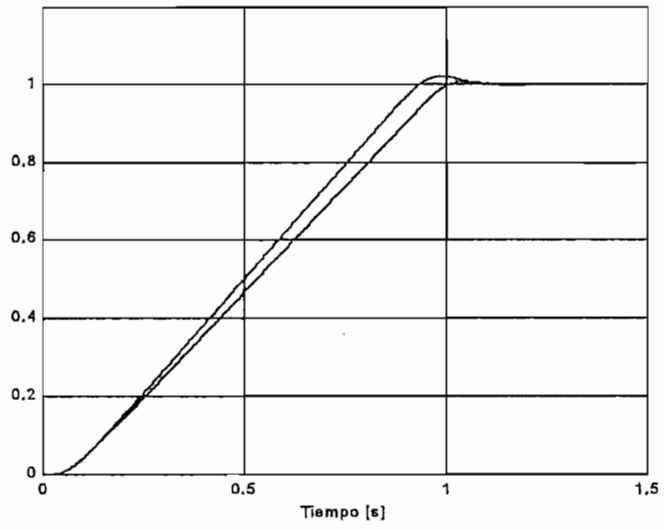


Figura 5.1 Respuesta paso de los controles difusos PD (magenta) PID (azul) para condiciones de carga nominales.

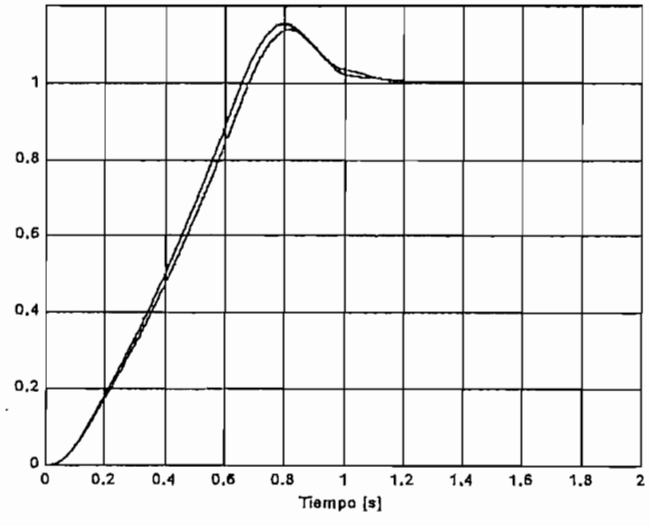


Figura 5.2 Respuesta paso de los controles difusos PD (magenta) PID (azul) para condiciones de carga mínimas.

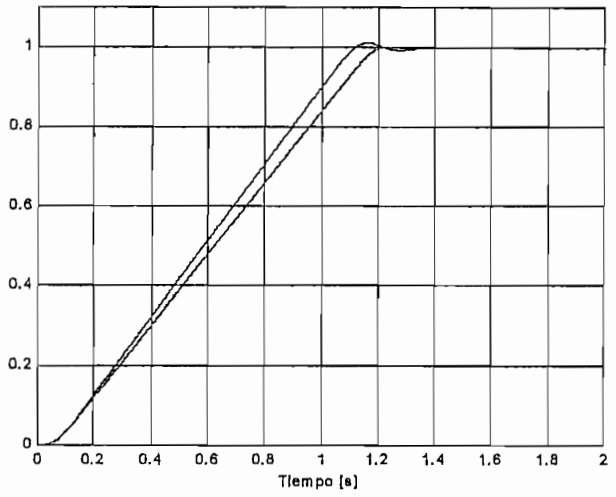


Figura 5.3 Respuesta paso de los controles difusos PD (magenta) PID (azul) para condiciones de carga máximas.

Estos resultados indican que el control difuso PD responde ligeramente mejor que el PID, aún cuando el PID tiene más opciones de calibración. Debido a que los resultados no son muy distintos, no se puede decir que el PD es mejor que el PID, pero si implica que se pueden obtener desempeños muy similares con menos trabajo computacional.

Es evidente además que cualquiera de los dos controles diseñados son muy sensibles a variaciones de la carga; es así que con carga mínima el sistema experimenta un sobre impulso del 15%, un tiempo de establecimiento de 1.1 segundos y un tiempo de subida de 0.7 segundos, mientras que con carga máxima el sobreimpulso desaparece, pero el sistema tiene un tiempo de establecimiento de 1.2 segundos y un tiempo de subida de 1.1 segundos. Se mostró además que el sobreimpulso, así como estos tiempos son muy sensibles a cambios en las posiciones correspondientes a un error "Positivo Grande" y una derivada del error "Negativa Grande" de la memoria asociativa.

En cuanto al seguimiento a la señal de referencia deseada, tanto el PID difuso así como el PD difuso demostraron tener un error de posición máximo de un milímetro para cualquier condición de carga. Debido a esto, las curvas prácticamente están sobrepuestas por lo que las diferencias no son apreciables y no se las muestra.

A diferencia de la posición, la trayectoria de la velocidad del elevador presentó ciertas oscilaciones al final de la rampa de subida y al final de la rampa de bajada. Estas oscilaciones se ven incrementadas en magnitud cuando la carga tiende a aumentar, para cualquiera de los dos controles difusos PD o PID. Los gráficos de estos resultados se presentan en las figuras 5.4 y 5.5 para el control PID y PD respectivamente.

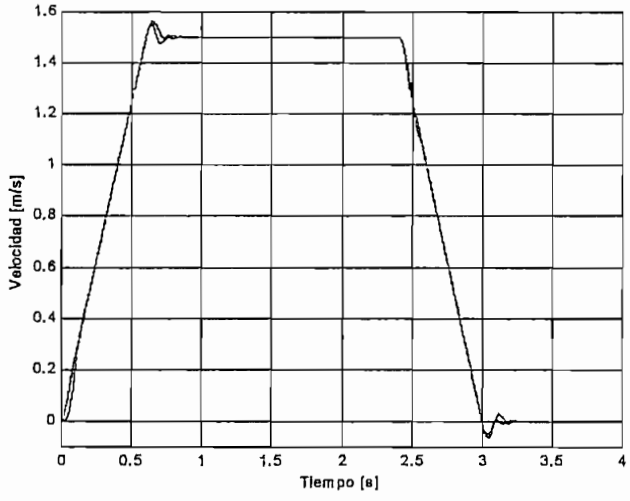


Figura 5.4 Trayectoria de velocidad para el control difuso PID.
Carga máxima (magenta), carga mínima (azul).

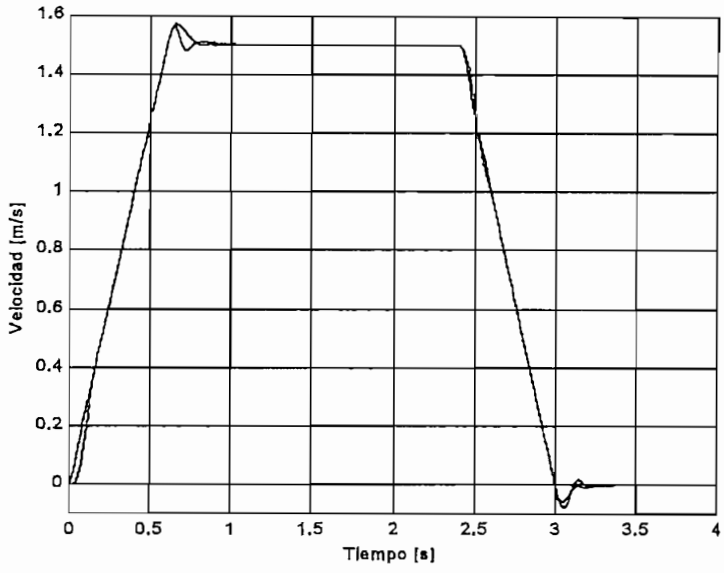


Figura 5.5 Trayectoria de velocidad para el control difuso PD. Carga máxima (magenta), carga mínima (azul).

Estos resultados llevan a concluir que los controles difusos aquí presentados, responden bastante bien a señales de referencia suaves y no discontinuas, pues sus características mejoraron con respecto a las de una entrada paso, que es una señal brusca.

Por otro lado está el rozamiento, que mediante simulación, se encontró que sus efectos son los de reducir los sobreimpulsos y aumentar el error en la posición; sin embargo, estas diferencias son muy poco apreciables, pues están en el orden de las décimas de milímetros.

V.1.2 COMPARACION ENTRE LOS CONTROLES CONVENCIONALES Y LOS DIFUSOS

La primera gran diferencia que se presenta entre el control difuso y el control convencional, además de su sustento teórico, es el método de diseño. Es claro que para el diseño de los controles difusos fue necesario nada más que un conocimiento heurístico de las acciones de control que se deben tomar dependiendo de las variables de control seleccionadas, mientras que para el diseño de los controles convencionales efectivamente se requirió del empleo del modelo matemático del elevador.

Ahora se abordan los resultados que cada uno de los controles produjo. Debido a que tanto el PID difuso como PD difuso tuvieron características muy similares, se toma como referencia al control PD para los resultados comparativos, por simplicidad.

Las figuras 5.6, 5.7 y 5.8 muestran la respuesta paso del control PD difuso, del PID convencional y de la realimentación de estado, para la condición de carga mínima, nominal y máxima respectivamente.

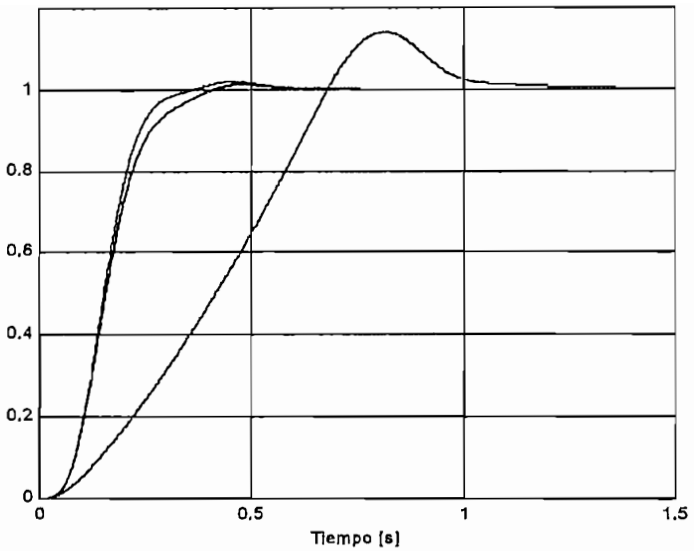


Figura 5.6 Respuesta a un paso unitario de los controles ensayados:PID (azul), realimentación de estado (negro), difuso (magenta), para carga mínima.

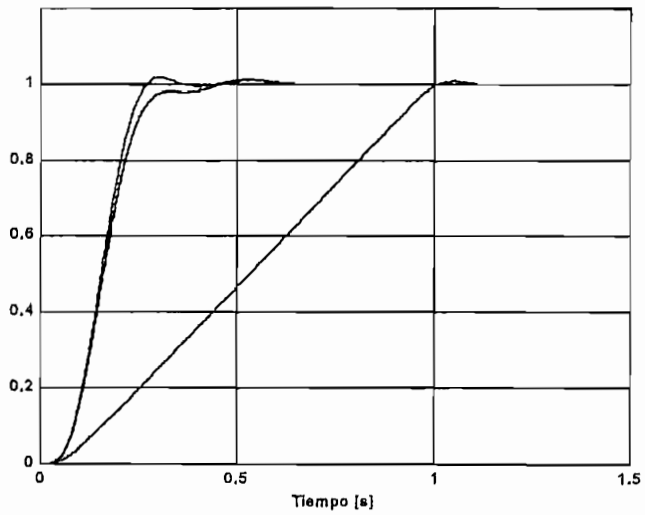


Figura 5.7 Respuesta a un paso unitario de los controles ensayados:PID (azul), realimentación de estado (negro), difuso (magenta), para carga nominal.

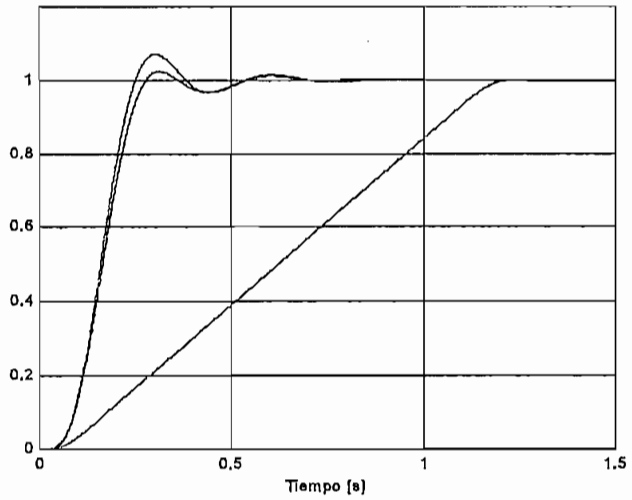


Figura 5.8 Respuesta a un paso unitario de los controles ensayados:PID (azul), realimentación de estado (negro), difuso (magenta), para carga máxima.

Claramente se observa que el comportamiento de los sistemas de control convencionales son más rápidos y menos dependientes de la condición de carga, por lo que presentan mayor robustez. La mayor rapidez de estos sistemas se puede explicar por la magnitud de la señal de salida que producen. Es así como en el caso del PID convencional y de la realimentación de estado, estos generan una señal de salida del orden de los 600 voltios(figuras 4.4 y 4.10), mientras que el control difuso genera una señal de salida del orden de los 80 voltios(figuras 4.23 y 4.31), esto debido a que el universo de salida se encuentra limitado. Este resultado es una ventaja de los controles difusos estándar sobre los convencionales; es decir, el universo de discurso permite incorporar fácilmente las restricciones físicas. Cuando se trata de realizar algo similar en los controles convencionales, aparecen efectos no deseados como inestabilidad, y a costo de introducir no linealidades en los lazos.

Falta ahora analizar, comparativamente, cuán capaces son estos controles de seguir a la señal de referencia deseada. Las figuras 5.9, 5.10 y 5.11 muestran los resultados de las simulaciones para condiciones de carga mínima, nominal y máxima para los tres controles.

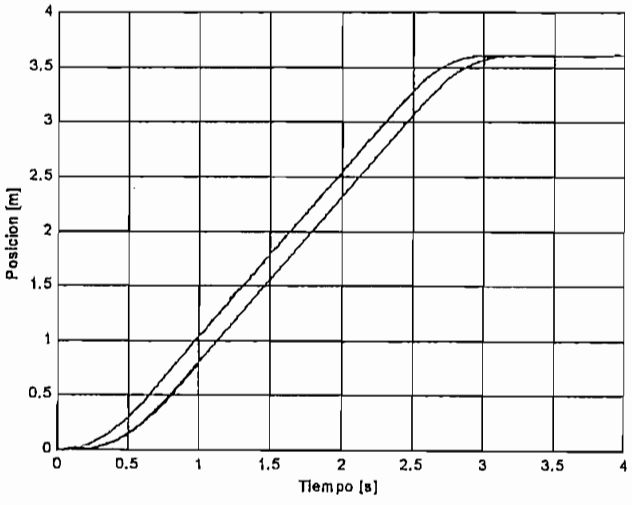


Figura 5.9 Trayectorias de posición del elevador con control:PID (azul), realimentación de estado (negro), difuso (magenta), con carga mínima, y referencia en líneas de trazos.

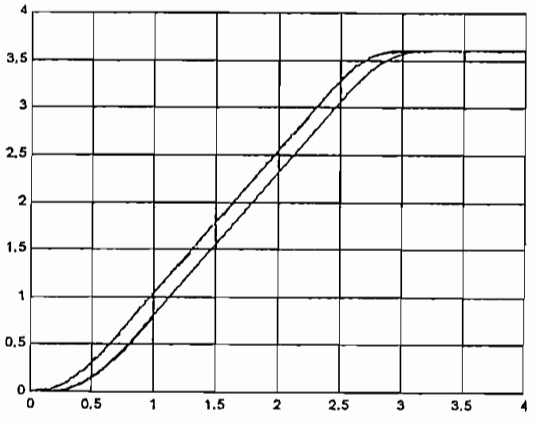


Figura 5.10 Trayectorias de posición del elevador con control:PID (azul), realimentación de estado (negro), difuso (magenta), con carga nominal, y referencia en línea de trazos.

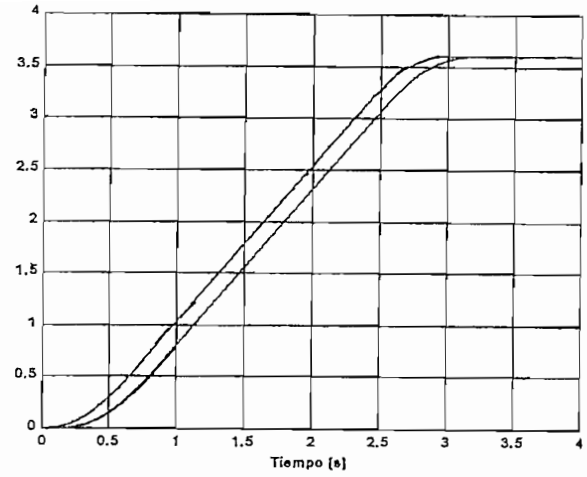


Figura 5.11 Trayectorias de posición del elevador con control:PID (azul), realimentación de estado (negro), difuso (magenta), con carga máxima, y referencia en línea de trazos.

Se puede ver, que el control difuso es el que mejor sigue a la trayectoria; aunque los tres controles presentan un error nulo en estado estable. En cuanto a la velocidad, la figura 5.12 muestra la curva de la velocidad para los tres controles con carga máxima únicamente, debido a que como se mostró, no se presentan mayores variaciones ante cambios de carga al seguir la trayectoria deseada. Con esto se comprueba que efectivamente los controles difusos, presentan mejores características de seguimiento.

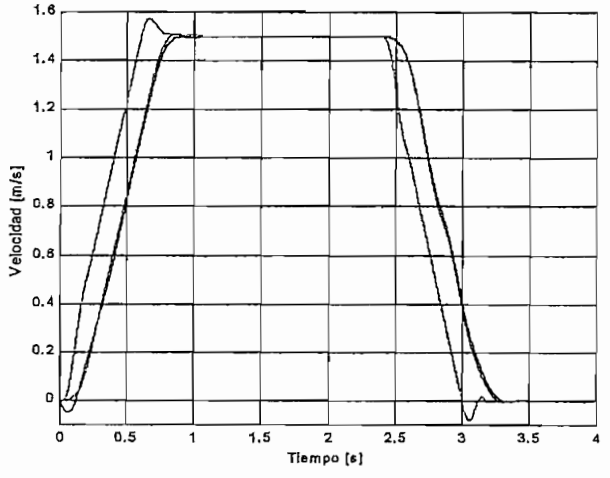


Figura 5.12 Trayectorias de velocidad del elevador con control:PID (azul), realimentación de estado (negro), difuso (magenta), con carga máxima.

Se nota además que para todos los controles ensayados, a excepción del PI difuso, la fuerza de rozamiento no es un factor que incida de manera determinante en el desempeño del sistema.

Es importante también recordar que un elevador debe ser capaz de ir en dos direcciones: hacia arriba o hacia abajo. A pesar de que se dedujo el modelo asumiendo que el elevador sube, este también es válido cuando el elevador baja. La figura 5.13 muestra la respuesta del sistema a una señal paso negativa para los tres controles para la condición de carga nominal, y en la cual se observa una respuesta idéntica a la de la figura 5.7, pero invertida

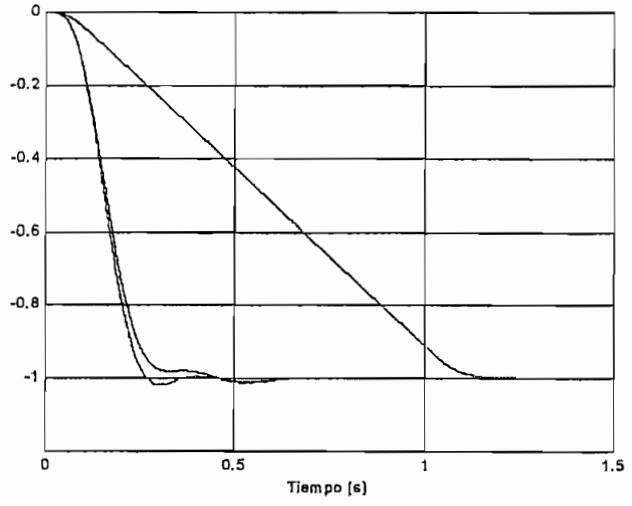


Figura 5.13 Respuesta a un paso unitario negativo de los controles ensayados:PID (azul), realimentación de estado (negro), difuso (magenta), para carga nominal.

Por último faltarían comparar los métodos de análisis de los sistema de control convencionales con los de los sistema de lógica difusa. Entonces, volviendo al sustento teórico abordado en el capítulo II, los sistemas de control difuso entran en el campo de los sistemas no-lineales. Se han desarrollado ciertas herramientas para el análisis de sistemas

no-lineales (ej: función descriptiva y estabilidad de Lyapunov); sin embargo su aplicabilidad está limitada a un conjunto muy reducido de ellos, que presentan características muy simples, y por esta razón, únicamente ciertos sistemas difusos pueden ser analizados. Una discusión muy interesante sobre los esfuerzos realizados para desarrollar herramientas de análisis para sistemas de lógica difusa se presenta en [15]. Por esto, muchas veces se recurre a la simulación.

En cuanto al modelo matemático deducido, es importante señalar que no existe ningún precedente o investigación previa que comprenda esta planta, por lo que se ha procurado analizar e incluir la mayor parte de sus componentes. Se cree entonces que el modelo matemático obtenido en este trabajo es lo suficientemente completo para el propósito deseado.

A manera de recomendación, mayor investigación puede ser realizada en este problema, empleando otras técnicas de control difuso. Por ejemplo, se mostró que la calibración de los controles es bastante laboriosa, entonces posiblemente la elaboración de un algoritmo de auto ajuste sugerido en [15] pueda mejorar el desempeño del sistema y liberar al diseñador de trabajo.

Además, la idea de que un sistema con dinámica como el PID puede obtener buenos resultados, puede estar sugiriendo el uso de un control difuso del tipo Takagi-Sugeno con interpolación entre sistemas lineales, o un esquema de control supervisorio; a su vez, la alta sensibilidad a cambios de carga de los controles difusos permite intuir el uso de un control difuso adaptivo, para mejores resultados.

Aparentemente, quedan por responder algunas preguntas, pero por un lado, no es el objetivo central del trabajo, y por otro, un trabajo de tesis debe tener limitaciones por razones de alcance en la investigación.

V.2 CONCLUSIONES

En base a todo lo anterior, se pueden establecer las siguientes conclusiones:

1. En lo que respecta a los lenguajes de programación:

- La elaboración de un programa para simulación de controladores difusos PI, PD y PID utilizando el lenguaje C++ resulta ser beneficiosa debido a que el tiempo de ejecución es bastante menor en comparación con el Toolbox de Control Fuzzy de MATLAB, a más de las mencionadas en el capítulo IV.
- El uso de lenguajes de programación multimedia tales como Director brinda muchas facilidades para la programación gráfica y de animaciones en tiempo real.

2. En lo que respecta al modelo del elevador:

- El problema del control del elevador, muestra ser un problema ideal para el desarrollo de muchas técnicas de control, ya que contiene no-linealidades, una perturbación externa, y parámetros variantes.
- El modelo desarrollado contiene las características más importantes de este tipo de mecanismos.

- Las fuerzas de rozamiento tienen el efecto de estabilizar al modelo cuando funciona en lazo cerrado.
- El punto de equilibrio del sistema depende de la perturbación y de la referencia, cuando no existe compensación.

3. En cuanto a los controles ensayados:

- El diseño de un control difuso requiere de un conocimiento heurístico del funcionamiento del elevador, mas no de un conocimiento matemático preciso.
- Los controladores PD y PID difusos muestran características muy similares de desempeño, mientras que el PI no se presenta favorable para el control del elevador, debido a que es muy difícil conseguir estabilizarlo.
- Los controles difusos PD y PID presentan un buen desempeño al controlar el elevador, cuando la señal de referencia es suave y continua, ya que son capaces de seguirla sin error para cualquier condición de carga.
- Una característica de los controles difusos estándar (Mamdani) es el de limitar fácilmente su señal de salida, al rango impuesto por las restricciones físicas del problema.
- El ajuste de un sistema de lógica difusa mediante conocimientos heurísticos, es un proceso bastante laborioso, y complicado debido a la gran cantidad de parámetros involucrados.
- Los controladores difusos PD y PID ensayados en este trabajo, son muy sensibles a variaciones de carga, ya que su respuesta desmejora cuando el sistema es sometido a señales bruscas tales como una señal paso.

- Los métodos de defusicación: Centro de Gravedad y Centro Promedio, no presentan diferencias apreciables en los resultados de simulación.
- El rozamiento no muestra ser un problema considerable en el control del elevador, pero sí en el diseño de los controles convencionales. Así, muchas veces al querer usar la Función Descriptiva, la función de transferencia equivalente es de fase no mínima, por lo que no es posible utilizar algunos métodos de diseño de sistemas de control.
- Del estudio comparativo entre los controles difusos y los convencionales, es claro que a pesar de que los difusos mostraron características inferiores en sus respuestas a señales de tipo paso, sus características de respuesta fueron superiores cuando las señales de referencia eran las que se espera que el elevador describa.

4. Ventajas, desventajas, y limitaciones:

- El programa de simulación desarrollado provee de un entorno gráfico fácil de usar para el ajuste rápido de los controles difusos y una simulación rápida, con la presentación de los resultados en tiempo real.
- En cuanto a las limitaciones del programa de simulación, está la de que no se pudo incorporar una opción integrada para la impresión de las curvas debido a limitaciones del lenguaje director.
- A pesar de que el modelo considerara la no linealidad del rozamiento, se han despreciado otras no linealidades tales como la saturación, debido a que de los resultados de los análisis, sus efectos son mínimos.

5. Posibles estudios posteriores:

- Se sugiere realizar futuros estudios en la aplicación del control difuso al problema del elevador empleando sistemas del tipo Takagi – Sugeno, sistemas difusos adaptivos y sistemas difusos supervisorios.
- En lo que se refiere al diseño de controles difusos, se sugiere que se realicen igualmente estudios en cuanto a algoritmos de auto ajuste y entrenamiento de sistemas difusos.
- Debido a la gran complejidad matemática en la representación de sistema de lógica difusa, se recomienda realizar un estudio sobre herramientas para el análisis de sistemas difusos como el de estabilidad y función descriptiva, ya que en la actualidad esta limitaciones son la razón por la cual su uso no es extensivo.
- Buscar nuevos modelos matemáticos de elevadores en los que se incluya un mayor número de no-linealidades.

BIBLIOGRAFÍA

[1] L. Allis, et al, "Inside Director 6 with Lingo", New Riders, In 1997.

[2] D.P. Atherton, "Nonlinear Control Engineering", Van Nostrand Reinhold, Student Edition, New York, 1982.

[3] Chuen Chien Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.20 No.2, pp. 404, March/April 1990

[4] E. Cox, "Fuzzy Fundamentals", *IEEE Spectrum*, pp.58-61, October 1992.

[5] R.C. Dorf, R.H. Bishop, "Modern Control Systems", Adisson Wesley Longman, 7th edition, October 1994.

[6] N. Govind, "Fuzzy Logic Control with the Intel 8XC196 Embedded Microcontroller", Intel Corporation, Chandler, AZ, pp.1

[7] H. Harrison, J. Bollinger, "Introduction to Automatic Controls", Scranton Pennsylvania, pp. 11-12, 324-325, 1970.

[8] R.C. Hibbeler, "Engineering Mechanics: Statics and Dinamics", Macmillan Publishing Company, 5th. edition, pp. 355-356, New York, 1989.

[9] M. Johansson, "A Primer on Fuzzy Control", *Department of Automatic Control – Lund Institute of Technology*, pp. 1-60, 1996.

[10] H.K. Khalil, "Nonlinear Systems", Macmillan Publishing Company, New York, 1992.

[11] P.C. Krause, "Analysis of Electrical Machinery", McGraw-Hill, New York, 1986.

[12] Macromedia, "Macromedia Director 6: Using Director", March 1997.

[13] Macromedia, "Macromedia Director 6: Using Lingo", March 1997.

[14] Macromedia, "Macromedia Director 6: Lingo Dictionary", March 1997.

[15] D. McGill, W. King, "Engineering Mechanics: Statics", PWS-Kento Pucblishing Co., pp. 357-360, 398-399, Boston, 1989.

[16] J.M. Mendel, "Fuzzy Logic Systems for Engineering: A Tutorial", *Proceedings of the IEEE*, Vol.83, No.3, pp.345-377, March 1995.

[17] J.L. Meriam, "Estática", Reverté, 2da. edición, pp.268-278, Barcelona, 1991.

[18] C. Pappas, W. Murray, "Manual de Borland C++ 4.0", McGrawHill, Madrid, 1994.

[19] K. Passino, S. Yurkovich, "Fuzzy Control", Addison Wesley Longman, 1998.

[20] R. Proaño Viteri, "Lógica-Conjuntos-Estructuras", Editora Luz de América, 2da. edición, Quito, 1992.

[21] J. Shigley, "Diseño en Ingeniería Mecánica", McGraw-Hill, 4ta. edición, pp.755-756, 775-782, México, 1990.

[22] G. Strakosch, "Transporte Vertical", Marcombo S.A., Barcelona, 1973.

[23] L.A. Zadeh, "Fuzzy Sets", *Information and Control*, vol.8, pp. 338-352, 1965

Referencias Adicionales:

- Schindler, Planning Guide, 1996.
- Schindler, Geared Traction Elevators Layout Data.
- http://www.macromedia.com

ANEXOS

ANEXO A

MANUAL DEL PROGRAMA

ANEXO A

MANUAL DEL PROGRAMA

INTRODUCCION: CDE es un programa de diseño y simulación de controles difusos estándar PD, PI y PID aplicados a elevadores, en un ambiente de Windows 95. Este Manual del Usuario contiene los requerimientos del sistema, el procedimiento de instalación y una familiarización con el uso del programa.

REQUERIMIENTOS DEL SISTEMA: Para que el programa pueda ser ejecutado sin problemas el sistema debe tener las siguientes características:

- Microsoft Windows 95.
- DOS 6.0 o superior.
- Unidad de CD-ROM.
- 20 Mb de espacio libre en disco.
- 16 Mb de memoria RAM.
- Procesador mínimo de 200 Mhz .

INSTALACION: Para realizar la instalación del programa realice los siguientes pasos:

1. En el menú "Inicio" de Windows 95, escoja la opción "Ejecutar"
2. Escriba en el cuadro de texto la siguiente instrucción:
   G:\Instalar.exe
3. Siga las instrucciones que el programa de instalación le indica.

Importante: En el paso "2", en lugar de "G" coloque el nombre de su unidad de CD-ROM.

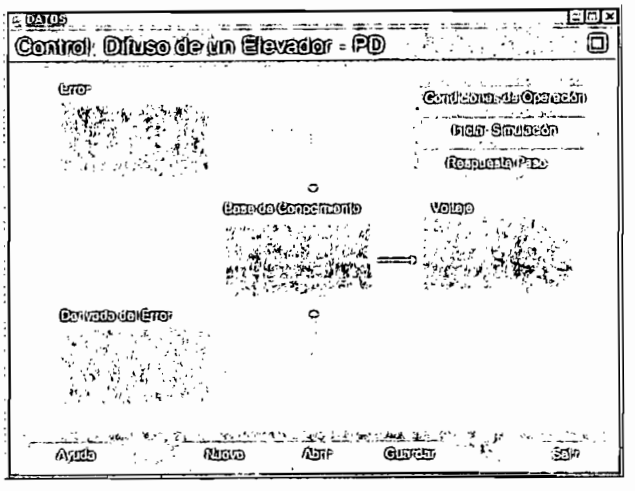
El programa de instalación, copiará los archivos del programa a su unidad de disco duro, y creará los accesos directos en el menú de "Inicio" y de "Programas".

FAMILIARIZACION CON CDE (CONTROL DIFUSO DE UN ELEVADOR):

CDE es un programa simple de utilizar. El primer paso consiste en ingresar al programa, para ello basta con escoger la opción "Programa CDE" en el menú inicio o de programas.

Una vez dentro, la primera pantalla del programa, le permite escoger entre los tres controles difusos propuestos: PD, PI o PID, permitiéndole visualizar sus esquemas respectivos. Cuando haya escogido con que control desea trabajar, haga clic en "Aceptar", caso contrario haga clic en "Salir" con lo que abandonará el programa.

Una vez seleccionado el control, aparece la pantalla de diseño del control, como la mostrada a continuación.



Independientemente de que control haya escogido, esta pantalla siempre presenta los siguientes botones en la parte inferior, y cuya función se describe a continuación:

ABRIR: Abre un archivo de texto donde se han guardado datos de un trabajo anterior.
GUARDAR: Guarda un archivo con los datos actuales de cada variable.
SALIR: Abandona el programa.
NUEVO: Resetea todos los valores a sus valores por omisión.
AYUDA: Provee de ayuda sobre el manejo del programa.

En la parte superior derecha existen tres botones más, cuyas funciones son:

CONDICIONES DE OPERACIÓN: Se pasa a otra pantalla en la cual se ingresan las condiciones de operación del elevador: carga y rozamiento en los rieles.

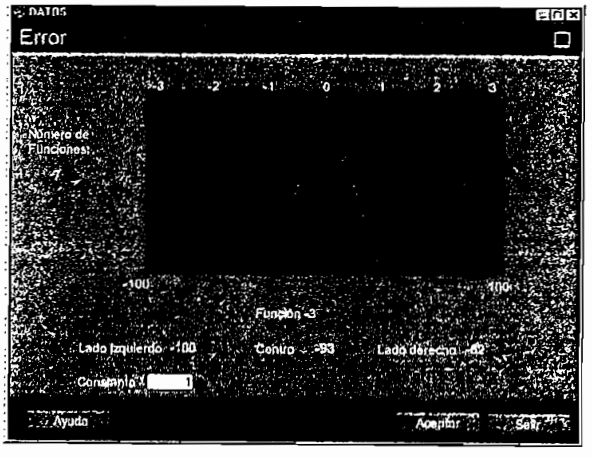
SIMULACION: Realiza la simulación del sistema con la señal de referencia de la trayectoria deseada, luego de la cual se pasa inmediatamente a la pantalla de resultados donde se muestra la animación y las curvas de posición y velocidad del elevador.

RESPUESTA PASO: Realiza la simulación del sistema sometido a una señal de tipo Paso Unitario, luego de la cual se pasa inmediatamente a la pantalla de resultados donde se muestra la animación y las curvas de posición y velocidad del elevador.

La parte central de la pantalla contiene los componentes del control difuso escogido. Para poder modificar sus propiedades basta con hacer clic sobre cualquiera de ellos:

Importante: Antes de poder ingresar a la Base de Conocimiento se deben haber establecido las propiedades de los demás componentes.

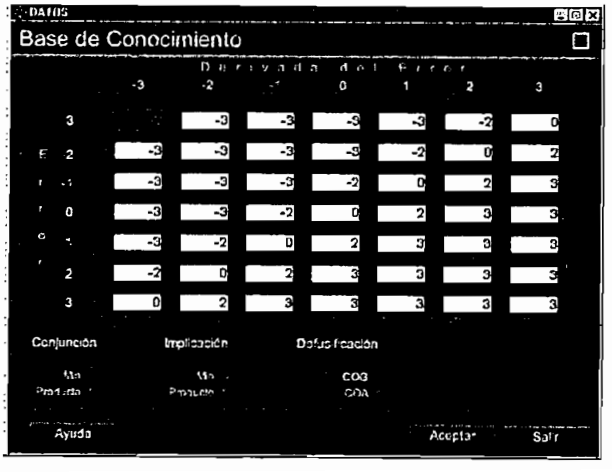
Las pantalla de edición de las variables lucen como en la siguiente figura:



Se permite realizar las siguientes acciones en esta pantalla:

- Escoger el número de funciones de membresía.
- Modificar la disposición física de cada función de membresía.
- Escalar el universo de la variable.

Una vez ingresados las propiedades de cada variable, se procede a ingresar la Base de Conocimiento, cuya pantalla es como sigue:



En esta pantalla además de poder modificar la memoria asociativa difusa, se pueden escoger las distintas formas de realizar las operaciones lógicas y el método de defusificación.

Una vez ingresados todos estos parámetros se puede iniciar cualquiera de las opciones de simulación.

Si Ud. tiene MATLAB ver. 5.0 o superior!

Si Ud. tiene MATLAB se han incluido pequeñas rutinas para poder realizar impresiones gráficas de los resultados de simulación que se cargan automáticamente el momento de la instalación. Estas rutinas son:

fpos: Gráfica de posición del elevador.
fvel: Gráfica de la velocidad del elevador.
salida: Gráfica de la señal de control.
pfd2d: Gráfica del plano de fase difuso para los controles PD o PI.
pfd3d: Gráfica del error, su derivada e integral en un espacio tridimensional, solo para el control PID.

Para poder ejecutar estas rutinas realice las siguientes acciones:

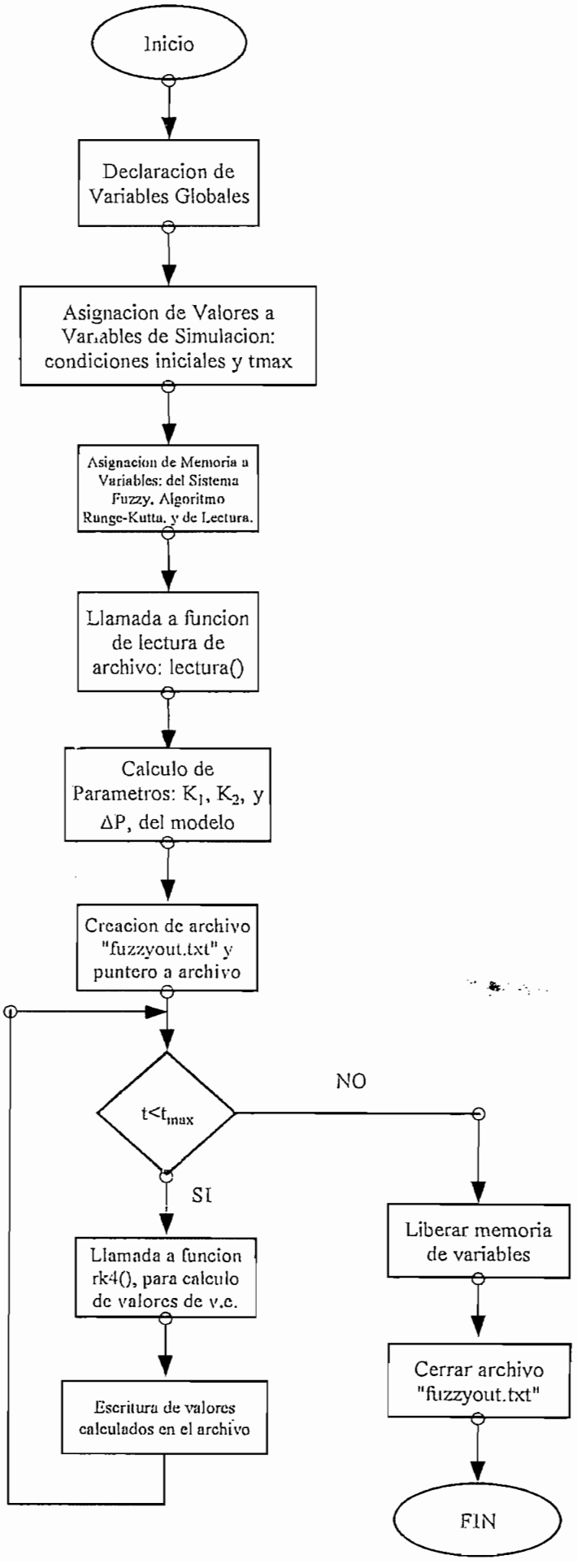
1. Cargue MATLAB
2. En el menú FILE escoja la opción "Set Path..."
3. Presione el botón "Browse"
4. Busque el directorio CDE y haga doble clic sobre él.
5. Presione "Close"

De esta manera basta con escribir el nombre de la rutina desde la línea de comandos de MATLAB y Ud. podrá visualizar las curvas de simulación.

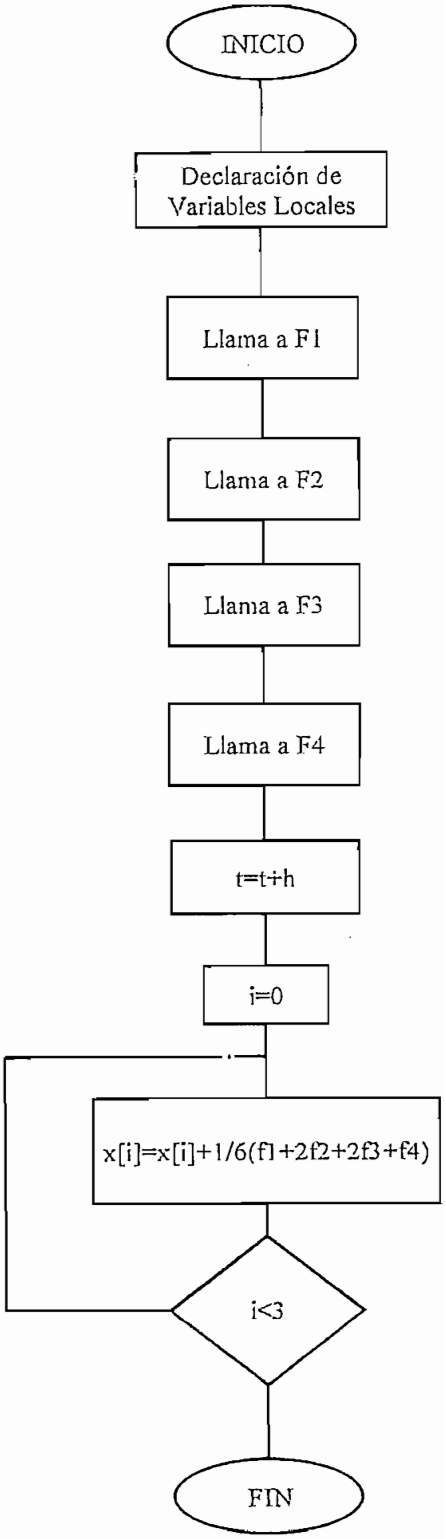
ANEXO B

DIAGRAMAS DE FLUJO

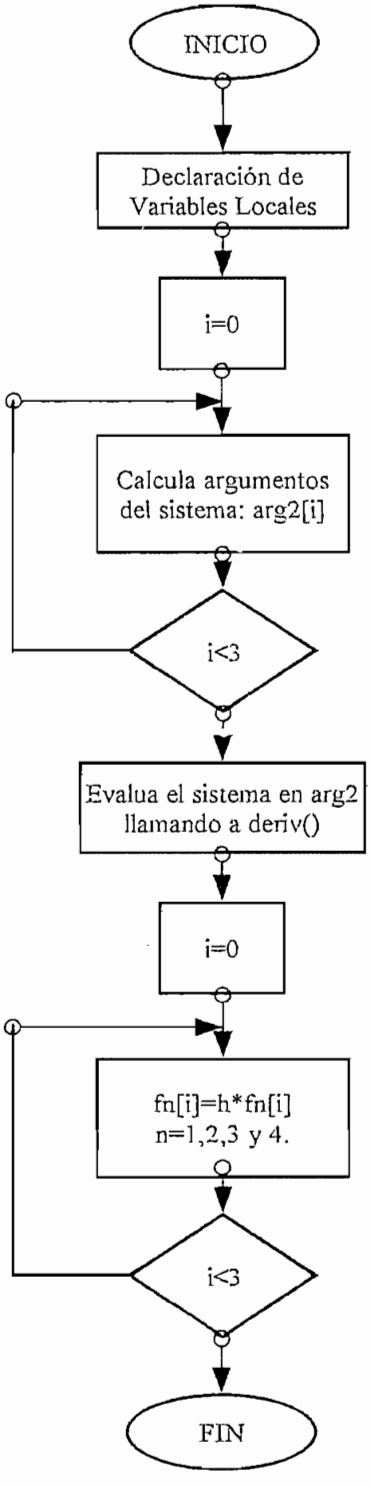
Modulo: main()


Inicio
Declaracion de Variables Globales
Asignacion de Valores a Variables de Simulacion: condiciones iniciales y tmax
Asignacion de Memoria a Variables: del Sistema Fuzzy, Algoritmo Runge-Kutta, y de Lectura.
Llamada a funcion de lectura de archivo: lectura()
Calculo de Parametros: K1, K2, y ΔP, del modelo
Creacion de archivo "fuzzyout.txt" y puntero a archivo
t<tmax
NO
SI
Llamada a funcion rk4(), para calculo de valores de v.e.
Escritura de valores calculados en el archivo
Liberar memoria de variables
Cerrar archivo "fuzzyout.txt"
FIN

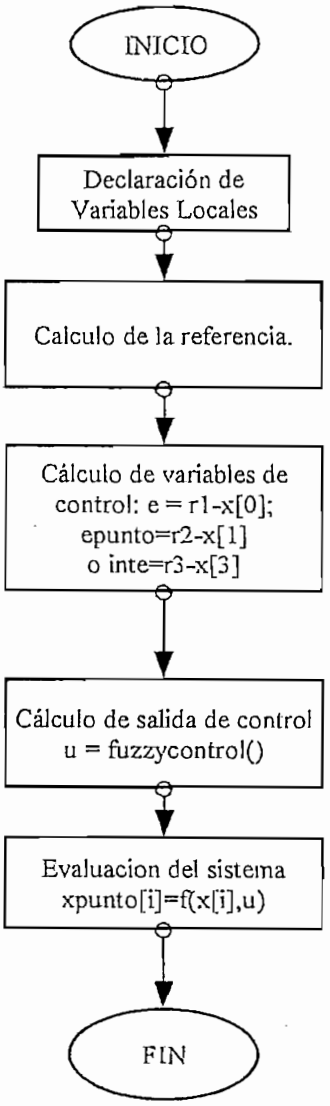
Funcion: rk4()
Tipo: void
Llamada desde: main()



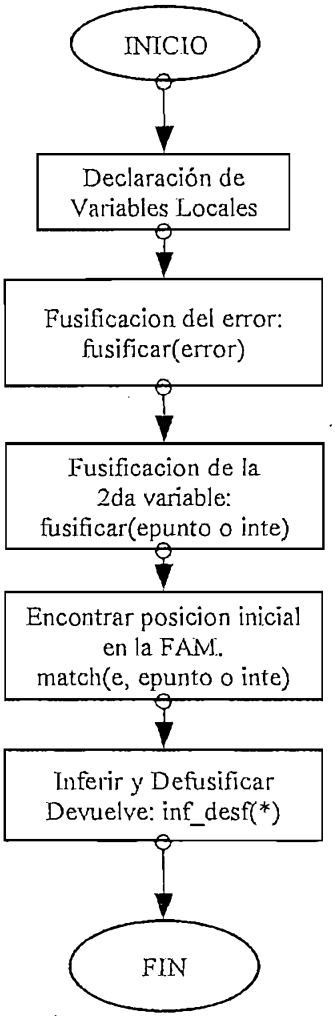
Funcion: F1(),F2(),F3(), y F4()
Tipo: void
Llamada desde: rk4()



Funcion: deriv()
Tipo: void
Llamada desde: F1(), F2(), F3() y F4()

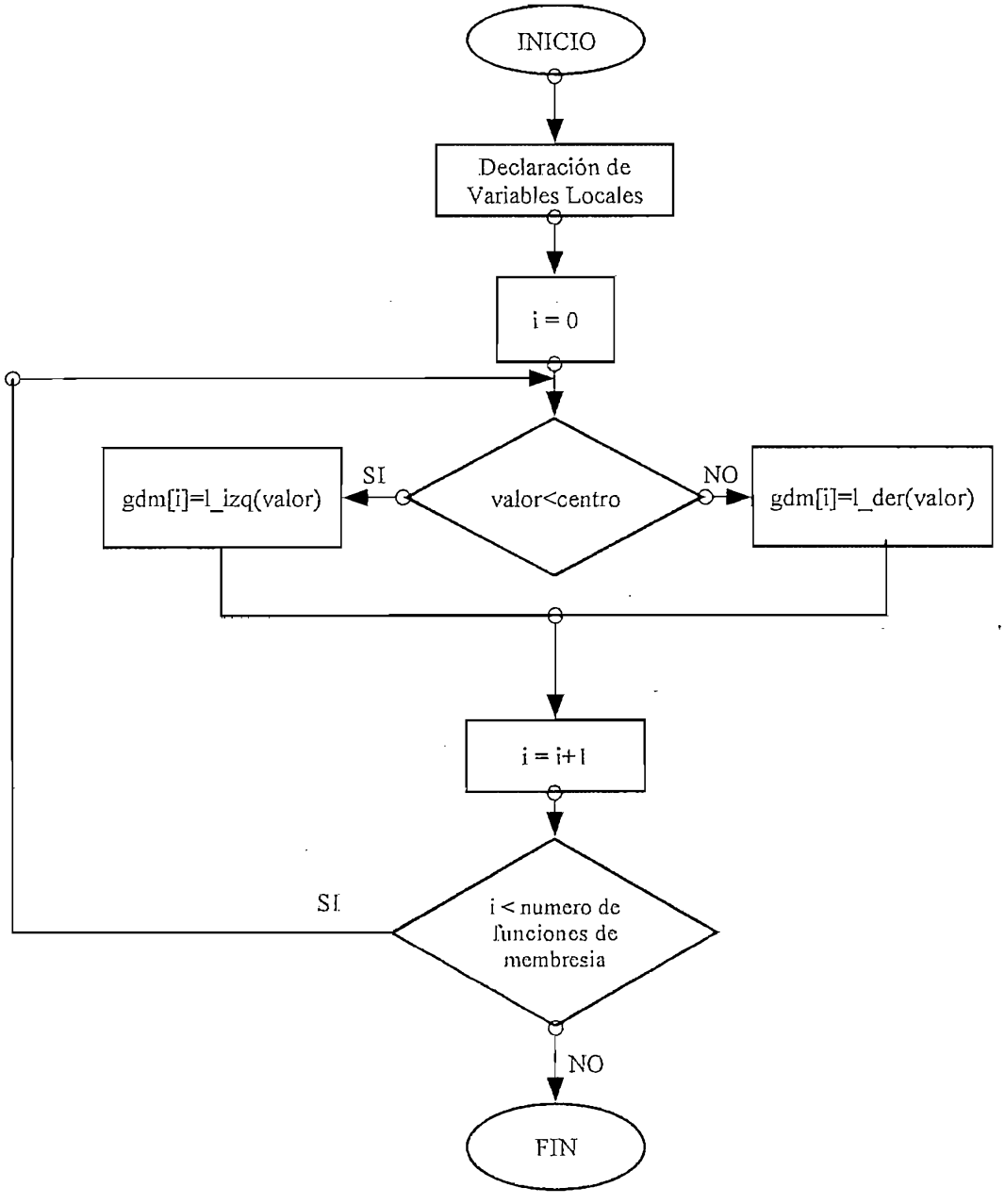


Funcion: fuzzycontrol(*)
Tipo: double
Llamada desde: deriv(*)

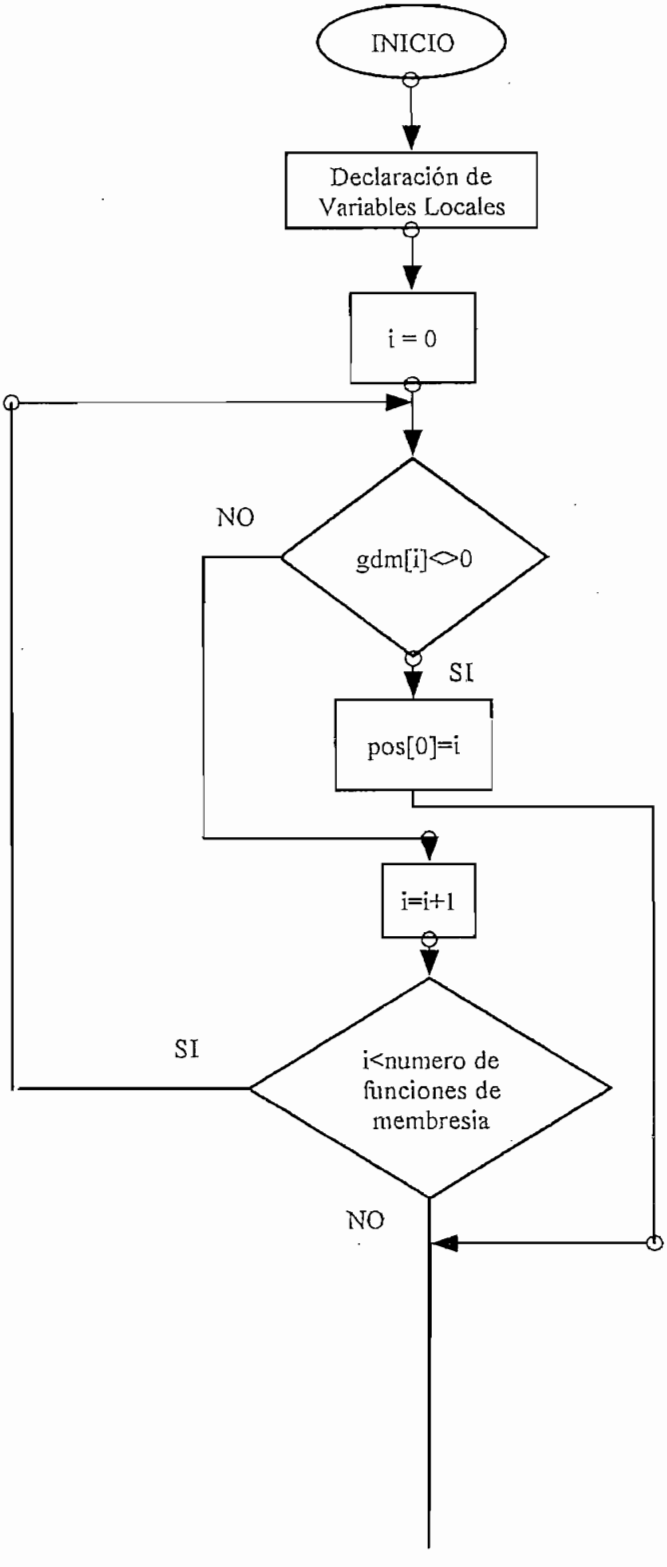


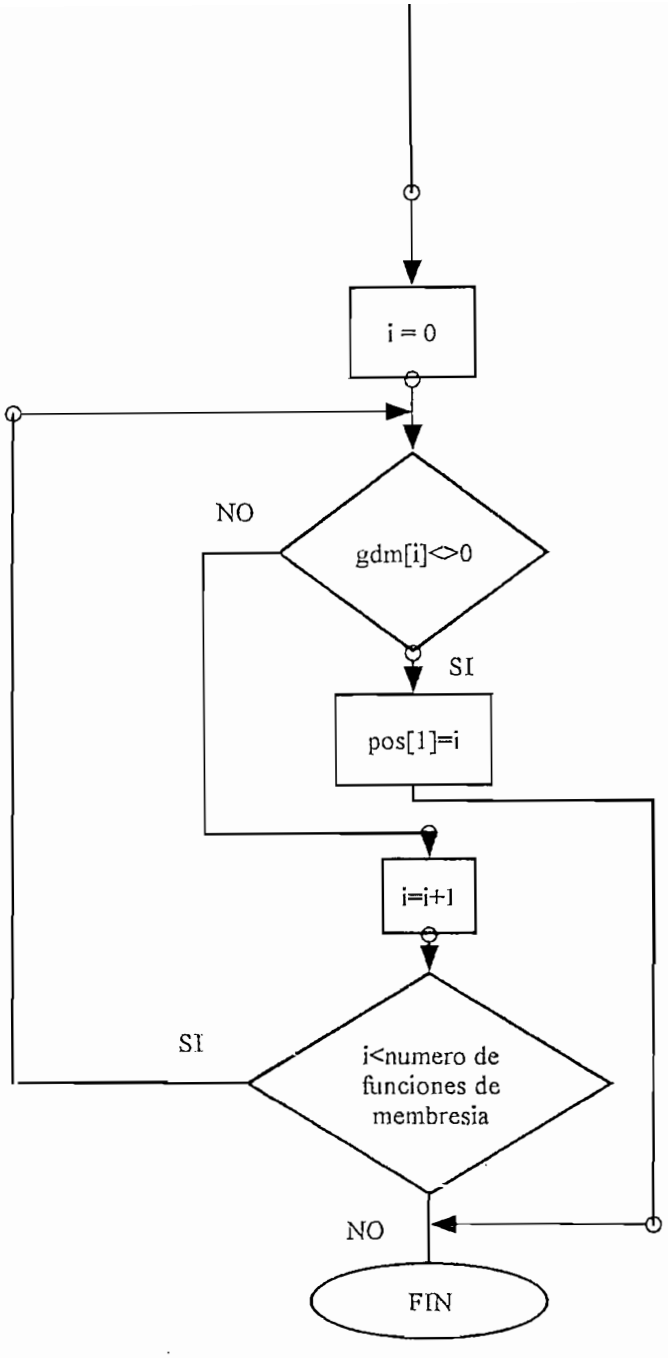
Nota: En el caso del control PID, se fusifica las tres variables e, epunto e inte, e igualmente la funcion match encuentra la posicion inicial en la FAM para las tres variables.

Funcion: fusificar(*)
Tipo: void
Llamada desde: fuzzycontrol(*)



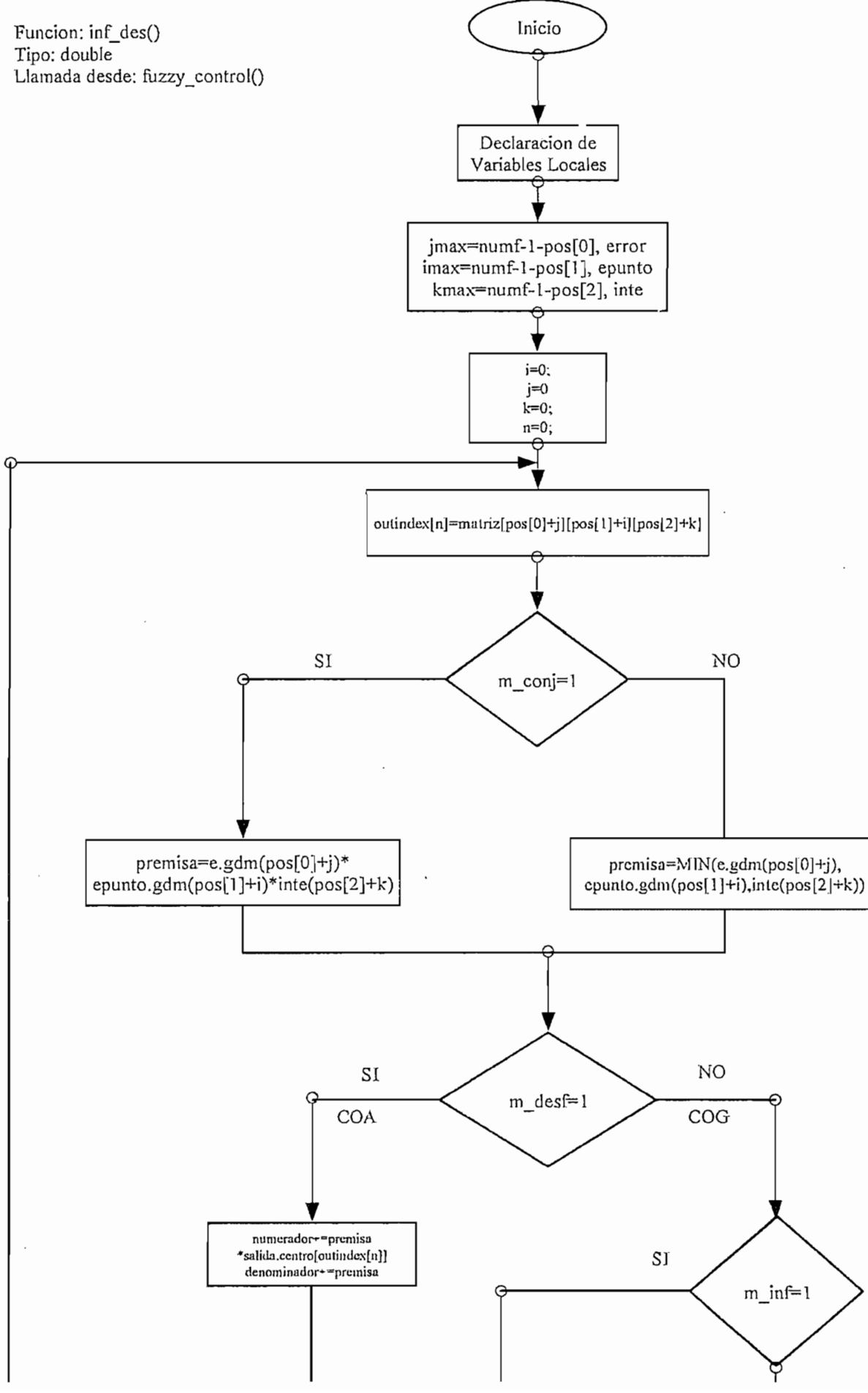
Funcion: match(*)
Tipo: void
Llamada desde: fuzzycontrol(*)

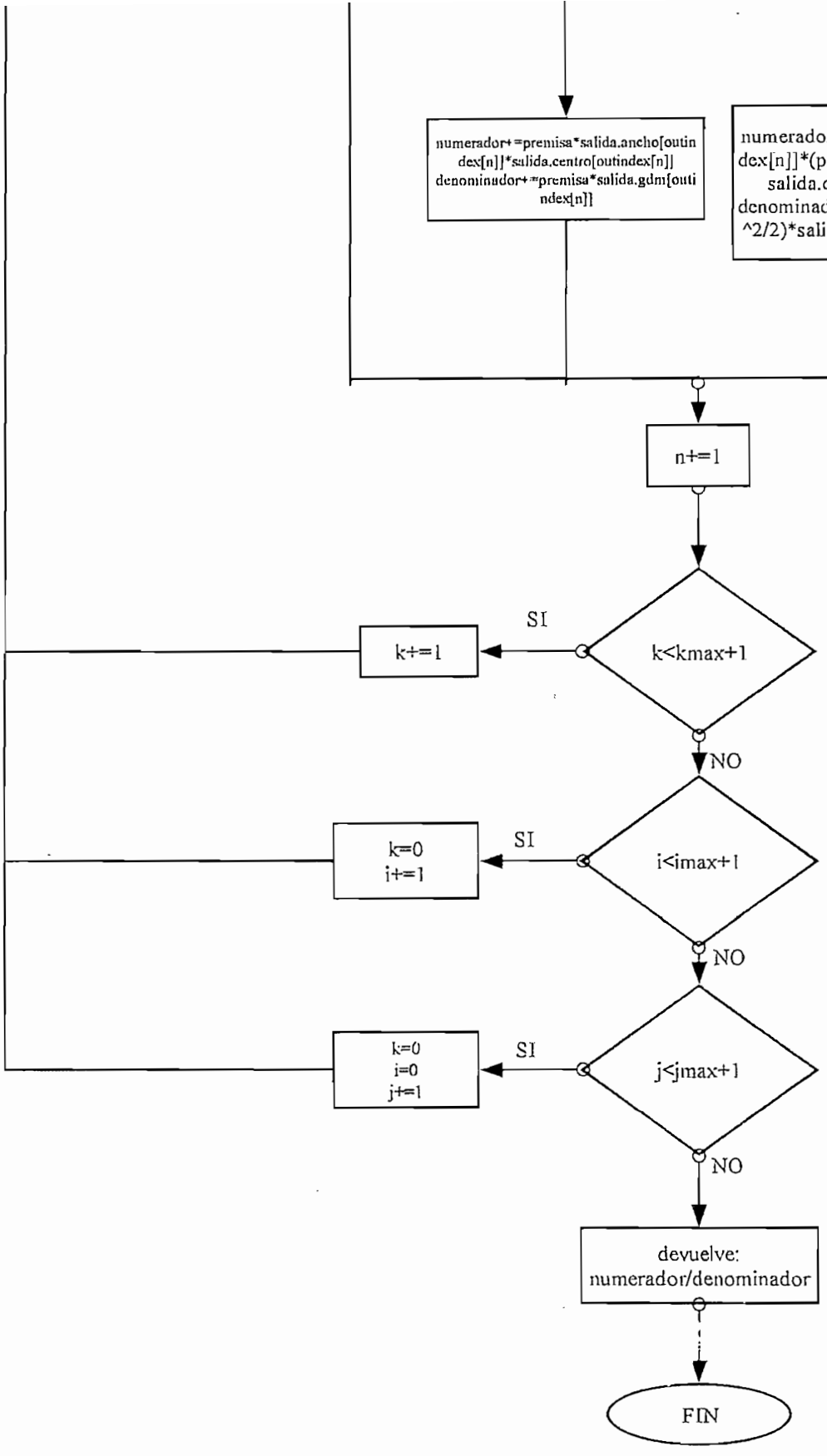




Nota: En el caso del control PID, se busca la posicion inicial en la FAM para una variables mas y se la guarda en la variable pos[2].

Funcion: inf_des()
Tipo: double
Llamada desde: fuzzy_control()





NOTA: Este flujograma pertenece al caso de tres variables de entrada (PID).

ANEXO C

CODIGO DEL PROGRAMA

El formato del archivo de texto fuzzyin.txt es como el que se muestra a continuación:

```
*a                                  // Identificador del error
5                                   // Numero de funciones del error
-300  -100  -50                     // Datos de las funciones:
-100  -50   0.0                     // izquierdo, centro, derecho
-50   0.0   50
0.0   50    100
50    100   300
*b                                  // Identificador del cambio de error
5                                   // Numero de funciones del cambio de error
-200  -100  -50                     // Datos de las funciones:
-100  -50   0.0                     // izquierdo, centro, derecho
-50   0.0   50
0.0   50    100
50    100   200
*c                                  // Identificador de voltaje de salida
5                                   // Numero de funciones de salida
-100  -66.66      -33.33           // Datos de las funciones:
-66.66      -33.33      0.0         // izquierdo, centro, derecho
-33.33      0.0   33.33
0.0   33.33 66.66
33.33 66.66 100
*d                                  // Identificador de FAM
-2    -2    -2    -1    0         // FAM: eje horizontal: cambio de error
-2    -2    -1    0     1         // eje vertical: error
-2    -1    0     1     2
-1    0     1     2     2
0     1     2     2     2
*e                                  // Identificador de parámetros
1                                   // Conjunción: 1= mínimo, 2= producto
1                                   // Implicación: 1 = mínimo, 2= producto
1                                   // Defusificación: 1 = cog, 2 = coa
0                                   // Carga del elevador en kg. (0-525)
2                                   // Modelo de rozamiento
12                                  // Fuerza Normal
0.001                               // go
0.01                                // g1
0.3                                 // g2
```

PROGRAMA EN BORLAND C++ 4.0

```
// fuzzy.h
// Control Difuso de un Elevador
// Realizado por: Alberto Sanchez Teran
// Director: Ing. Marco Barragan
// Facultad de Ingenieria Electrica
// Escuela Politecnica Nacional


#include <fstream.h>
#include <stdlib.h>

#define MAX(A,B) ((A)>(B)?(A):(B))
#define MIN(A,B) ((A)<(B)?(A):(B))

int matriz[9][9];


// Area de estructuras

typedef struct modelo{
    double deltap;
    double p1;
    double k1;
    double k2;
    double roz;
    int tipo;
} MODELO;

typedef struct punto{
    double  *p1;
    double  *p2;
    double  *p3;
    double  g0;
    double  g1;
    double  g2;
}PUNTO;

typedef struct mfin{
    double *centro;
    double *hwr;
    double *hwl;
    double *gdm;
    int    numf;
} MFIN;

typedef struct mfout{
    double *centro;
    double *ancho;
    int    numf;
} MFOUT;


typedef struct param {
    int m_conj;
    int m_inf;
    int m_desf;
} PARAM;


typedef struct sis_fuzzy {
    MFIN *e;
    MFIN *epunto;
    MFOUT *salida;
    PARAM *parametro;
} SISTEMA_DIFUSO;


// Area de prototipado de funciones

void asig(PUNTO *mem);
void lectura(SISTEMA_DIFUSO *sisfuzzy, PUNTO *mem, MODELO *model);
void asigfuzzyin(MFIN *mem, int numero);
void asigfuzzyout(MFOUT *mem, int numero);
void liberar(PUNTO *mem);                    1
```

```
                                numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->anc
ho[outindex[n]]*sisfuzzy->salida->centro 2utindex[n]];
```

```
void asignar_memoria(SISTEMA_DIFUSO *sisfuzzy);
void liberar_memoria(SISTEMA_DIFUSO *sisfuzzy);
void fusificar(double valor, MFIN *mem);
double saturada_izq(double u, double c, double w);
double saturada_der(double u, double c, double w);
double l_der(double u, double c, double w);
double l_izq(double u, double c, double w);
void match(const MFIN *e, const MFIN *epunto, int *pos);
double inf_desf(SISTEMA_DIFUSO *sisfuzzy,int *pos, int *outindex);
double fuzzy_control(double e, double epunto, SISTEMA_DIFUSO *sisfuzzy);

// Area de implementacion de funciones

void liberar(PUNTO *mem){
    delete(mem->p1);
    delete(mem->p2);
    delete(mem->p3);
}

double fuzzy_control(double e, double epunto, SISTEMA_DIFUSO *sisfuzzy){
    int pos[2];
    int outindex[4];

    fusificar(e,sisfuzzy->e);
    fusificar(epunto,sisfuzzy->epunto);
    match(sisfuzzy->e, sisfuzzy->epunto, pos);
    return inf_desf(sisfuzzy, pos, outindex);
}


double inf_desf(SISTEMA_DIFUSO *sisfuzzy, int *pos, int *outindex){
    int i, imax;
    int j, jmax;
    int n=0;
    double premisa,
         numerador=0,
         denominador=0;

    jmax=(sisfuzzy->e->numf-1)-pos[0];       // j lleva a e.
    imax=(sisfuzzy->epunto->numf-1)-pos[1];  // i lleva a epunto.

// Caso (a): imax y jmax son mayores que cero.

    if(imax>0 && jmax>0){
        for(j=0;j<2;j++){
            for(i=0;i<2;i++){
                outindex[n]=matriz[pos[0]+j][pos[1]+i];  //almacena conjunto inferid
o
                if(sisfuzzy->parametro->m_conj==2)    //conj. por producto
                    premisa=sisfuzzy->e->gdm[pos[0]+j]*sisfuzzy->epunto->gdm[pos[1]+
i];
                else                                  //conj. por minimo
                    premisa=MIN(sisfuzzy->e->gdm[pos[0]+j],sisfuzzy->epunto->gdm[pos
[1]+i]);

                //DESFUSIFICACION

                if(sisfuzzy->parametro->m_desf==2){    //COA
                    denominador+=premisa;
                    numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
                }
                else {
                    if(sisfuzzy->parametro->m_inf==2){
                        denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]];
                        numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*sisf
uzzy->salida->centro[outindex[n]];
                    }
                    else{
                        denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->a
ncho[outindex[n]];
                        numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->anc
ho[outindex[n]]*sisfuzzy->salida->centro[outindex[n]];
```

```
                    }
                }
                n+=1;
            }
        }
    }
// Caso (b): imax>0 jmax=0

    else if(imax>0 && jmax==0){
        for(i=0;i<2;i++){
            outindex[n]=matriz[pos[0]][pos[1]+i];
            if(sisfuzzy->parametro->m_conj==2)     //conj. por producto
                premisa=sisfuzzy->e->gdm[pos[0]]*sisfuzzy->epunto->gdm[pos[1]+i];
            else                                   //conj. por minimo
                premisa=MIN(sisfuzzy->e->gdm[pos[0]],sisfuzzy->epunto->gdm[pos[1]+i]
);

            //DESFUSIFICACION

            if(sisfuzzy->parametro->m_desf==2){    //COA
                denominador+=premisa;
                numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
            }
            else {                                 //COG
                if(sisfuzzy->parametro->m_inf==2){
                    denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]];
                    numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*sisfuzzy
->salida->centro[outindex[n]];
                }
                else{
                    denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->ancho
[outindex[n]];
                    numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->ancho[o
utindex[n]]*sisfuzzy->salida->centro[outindex[n]];
                }
            }
            n+=1;
        }
    }
// Caso (c):  imax=0 y jamx>0

    else if(imax==0 && jmax>0){
        for(j=0;j<2;j++){
            outindex[n]=matriz[pos[0]+j][pos[1]];
            if(sisfuzzy->parametro->m_conj==2)     //conj. por producto
                premisa=sisfuzzy->e->gdm[pos[0]+j]*sisfuzzy->epunto->gdm[pos[1]];
            else                                   //conj. por minimo
                premisa=MIN(sisfuzzy->e->gdm[pos[0]+j],sisfuzzy->epunto->gdm[pos[1]]
);

            //DESFUSIFICACION

            if(sisfuzzy->parametro->m_desf==0){    //COA
                denominador+=premisa;
                numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
            }
            else {                                 //COG
                if(sisfuzzy->parametro->m_inf==2){
                    denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]];
                    numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*sisfuzzy
->salida->centro[outindex[n]];
                }
                else{
                    denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->ancho
[outindex[n]];
                    numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->ancho[o
utindex[n]]*sisfuzzy->salida->centro[outindex[n]];
                }
            }
            n+=1;
        }
    }
//  Caso (d): imax=0 y jmax=0
    else {
```

```
        outindex[0]=matriz[pos[0]][pos[1]];
        if(sisfuzzy->parametro->m_conj==2)     //conj. por producto
            premisa=sisfuzzy->e->gdm[pos[0]]*sisfuzzy->epunto->gdm[pos[1]];
        else                                   //conj. por minimo
            premisa=MIN(sisfuzzy->e->gdm[pos[0]],sisfuzzy->epunto->gdm[pos[1]]);

        //DESFUSIFICACION

        if(sisfuzzy->parametro->m_desf==2){    //COA
            denominador+=premisa;
            numerador+=premisa*sisfuzzy->salida->centro[outindex[0]];
        }
        else {                                 //COG
            if(sisfuzzy->parametro->m_inf==2){
                denominador+=premisa*sisfuzzy->salida->ancho[outindex[0]];
                numerador+=premisa*sisfuzzy->salida->ancho[outindex[0]]*sisfuzzy->sa
lida->centro[outindex[0]];
            }
            else{
                denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->ancho[out
index[0]];
                numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->ancho[outin
dex[0]]*sisfuzzy->salida->centro[outindex[0]];
            }
        }
    }

    return numerador/denominador;
}


void match(const MFIN *e, const MFIN *epunto, int *pos){
    int i;

    for(i=0;i<e->numf;i++){
        if(e->gdm[i]!=0){
            pos[0]=i;
            break;
        }
    }
    for(i=0;i<epunto->numf;i++){
        if(epunto->gdm[i]!=0){
            pos[1]=i;
            break;
        }
    }
}



void fusificar(double valor, MFIN *mem) {

    int i;
    int r=mem->numf;

    mem->gdm[0]=saturada_izq(valor,mem->centro[0],mem->hwr[0]);
    mem->gdm[r-1]=saturada_der(valor,mem->centro[r-1],mem->hwl[r-1]);

    for(i=1;i<r-1;i++){
        if(valor<mem->centro[i])
            mem->gdm[i]=l_izq(valor,mem->centro[i],mem->hwl[i]);
        else
            mem->gdm[i]=l_der(valor,mem->centro[i],mem->hwr[i+1]);
    }
}


double l_der(double u, double c, double w){
    return MAX(0,(1-(u-c)/w));
}

double l_izq(double u, double c, double w){
    return MAX(0,(1-(c-u)/w));
}
```

```
double saturada_izq(double u, double c, double w) {
    if(u<c)
        return 1.0;
    else
        return MAX(0,(1-(u-c)/w));
}

double saturada_der(double u, double c, double w) {
    if(u>c)
        return 1.0;
    else
        return MAX(0,(1-(c-u)/w));
}



void asignar_memoria(SISTEMA_DIFUSO *sisfuzzy) {

// Asignacion de memoria para las variables de la estructura de
// tipo SISTEMA_DIFUSO

    sisfuzzy->e=new MFIN[1];
    sisfuzzy->epunto=new MFIN[1];
    sisfuzzy->salida=new MFOUT[1];
    sisfuzzy->parametro=new PARAM[1];

}


void liberar_memoria(SISTEMA_DIFUSO *sisfuzzy) {

    delete(sisfuzzy->e->centro);
    delete(sisfuzzy->e->hwl);
    delete(sisfuzzy->e->hwr);
    delete(sisfuzzy->e->gdm);

    delete(sisfuzzy->epunto->centro);
    delete(sisfuzzy->epunto->hwl);
    delete(sisfuzzy->epunto->hwr);
    delete(sisfuzzy->epunto->gdm);

    delete(sisfuzzy->salida->centro);
    delete(sisfuzzy->salida->ancho);

    delete(sisfuzzy->e);
    delete(sisfuzzy->epunto);
    delete(sisfuzzy->salida);
    delete(sisfuzzy->parametro);
}



void lectura(SISTEMA_DIFUSO *sisfuzzy, PUNTO *mem,MODELO *model){
    int numero,i,j,dato;
    char    marcador1,marcador2;

    ifstream fin("fuzzyin.txt");

    fin>>marcador1>>marcador2;
    if(marcador1=='*' && marcador2=='a'){   //Datos de las mf's del error
        fin>>numero;
        asigfuzzyin(sisfuzzy->e,numero); //asig. memoria del error

        // lectura de datos del error, calculo y ubicacion en
        // sus respectivas matrices y variables.
        for(i=0;i<numero;i++){
            fin>>mem->p1[i]>>mem->p2[i]>>mem->p3[i];
            sisfuzzy->e->hwl[i]=mem->p2[i]-mem->p1[i];
            sisfuzzy->e->hwr[i]=mem->p3[i]-mem->p2[i];
            sisfuzzy->e->centro[i]=mem->p2[i];
        }
```

```
        fin>>marcador1>>marcador2;
        }
        if(marcador1=='*' && marcador2=='b'){  // Datos de las mf's epunto
            fin>>numero;
            asigfuzzyin(sisfuzzy->epunto,numero);

            // lectura de datos del c. de error, calculo y ubicacion en
            // sus respectivas matrices y variables.
            for(i=0;i<numero;i++){
                fin>>mem->p1[i]>>mem->p2[i]>>mem->p3[i];
                sisfuzzy->epunto->hwl[i]=mem->p2[i]-mem->p1[i];
                sisfuzzy->epunto->hwr[i]=mem->p3[i]-mem->p2[i];
                sisfuzzy->epunto->centro[i]=mem->p2[i];
            }
            fin>>marcador1>>marcador2;
        }
        if(marcador1=='*' && marcador2=='c'){ // Datos de las mf's de salida
            fin>>numero;
            asigfuzzyout(sisfuzzy->salida,numero);

            // lectura de datos de salida, calculo y ubicacion en
            // sus respectivas matrices y variables.
            for(i=0;i<numero;i++){
                fin>>mem->p1[i]>>mem->p2[i]>>mem->p3[i];
                sisfuzzy->salida->ancho[i]=mem->p3[i]-mem->p1[i];
                sisfuzzy->salida->centro[i]=mem->p2[i];
            }
            fin>>marcador1>>marcador2;
        }
        if(marcador1=='*' && marcador2=='d'){ // Datos de la matriz
            // lectura de datos de la matriz

            for(i=0;i<sisfuzzy->e->numf;i++){
                for(j=0;j<sisfuzzy->epunto->numf;j++){
                    fin>>dato;
                    matriz[i][j]=dato+(numero-1)/2;
                }
            }
            fin>>marcador1>>marcador2;
        }
        if(marcador1=='*' && marcador2=='e'){ // Datos de sistema fuzzy y modelo
            fin>>sisfuzzy->parametro->m_conj;
            fin>>sisfuzzy->parametro->m_inf;
            fin>>sisfuzzy->parametro->m_desf;
            fin>>model->p1;
            fin>>model->tipo;
            fin>>model->roz;
            fin>>mem->g0;
            fin>>mem->g1;
            fin>>mem->g2;
        }
        else
            exit(1);

        for(i=0;i<sisfuzzy->e->numf;i++){
            sisfuzzy->e->centro[i]*=mem->g0;
            sisfuzzy->e->hwl[i]*=mem->g0;
            sisfuzzy->e->hwr[i]*=mem->g0;
        }
        for(i=0;i<sisfuzzy->epunto->numf;i++){
            sisfuzzy->epunto->centro[i]*=mem->g1;
            sisfuzzy->epunto->hwl[i]*=mem->g1;
            sisfuzzy->epunto->hwr[i]*=mem->g1;
        }
        for(i=0;i<sisfuzzy->salida->numf;i++){
            sisfuzzy->salida->centro[i]*=mem->g2;
            sisfuzzy->salida->ancho[i]*=mem->g2;
        }

        fin.close();
    }

    void asig(PUNTO *mem){
    //  mem=new PUNTO[1];
```

```
    mem->p1=new double[7];
    mem->p2=new double[7];
    mem->p3=new double[7];
}

void asigfuzzyin(MFIN *mem,int numero){

    mem->numf=numero;
    mem->centro=new double[numero];
    mem->gdm=new double[numero];
    mem->hwl=new double [numero];
    mem->hwr=new double [numero];

}

void asigfuzzyout(MFOUT *mem,int numero){

    mem->numf=numero;
    mem->centro=new double[numero];
    mem->ancho=new double [numero];
}
```

```
// Fuzzypid.h
// Control Difuso de un Elevador
// Realizado por: Alberto Sanchez Teran
// Director: Ing. Marco Barragan
// Facultad de Ingenieria Electrica
// Escuela Politecnica Nacional

#include <fstream.h>
#include <stdlib.h>

#define MAX(A,B) ((A)>(B)?(A):(B))
#define MIN(A,B) ((A)<(B)?(A):(B))

int matriz[9][9][9];


// Area de estructuras

typedef struct modelo{
    double deltap;
    double p1;
    double k1;
    double k2;
    double roz;
    int tipo;
} MODELO;

typedef struct punto{
    double  *p1;
    double  *p2;
    double  *p3;
    double  g0;
    double  g1;
    double  g2;
    double  g3;
}PUNTO;

typedef struct mfin{
    double *centro;
    double *hwr;
    double *hwl;
    double *gdm;
    int    numf;
} MFIN;

typedef struct mfout{
    double *centro;
    double *ancho;
    int    numf;
} MFOUT;


typedef struct param {
    int m_conj;
    int m_inf;
    int m_desf;
} PARAM;


typedef struct sis_fuzzy {
    MFIN *e;
    MFIN *epunto;
    MFIN *inte;
    MFOUT *salida;
    PARAM *parametro;
} SISTEMA_DIFUSO;


// Area de prototipado de funciones

void asig(PUNTO *mem);
void lectura(SISTEMA_DIFUSO *sisfuzzy, PUNTO *mem, MODELO *model);
void asigfuzzyin(MFIN *mem, int numero);
void asigfuzzyout(MFOUT *mem, int numero);
```

```
void liberar(PUNTO *mem);

void asignar_memoria(SISTEMA_DIFUSO *sisfuzzy);
void liberar_memoria(SISTEMA_DIFUSO *sisfuzzy);
void fusificar(double valor, MFIN *mem);
double saturada_izq(double u, double c, double w);
double saturada_der(double u, double c, double w);
double l_der(double u, double c, double w);
double l_izq(double u, double c, double w);
void match(const MFIN *e,const MFIN *inte, const MFIN *epunto, int *pos);
double inf_desf(SISTEMA_DIFUSO *sisfuzzy,int *pos, int *outindex);
double fuzzy_control(double e, double inte, double epunto, SISTEMA_DIFUSO *sisfuzzy)
;

// Area de implementacion de funciones

void liberar(PUNTO *mem){
    delete(mem->p1);
    delete(mem->p2);
    delete(mem->p3);
}

double fuzzy_control(double e, double inte, double epunto, SISTEMA_DIFUSO *sisfuzzy)
{
    int pos[3];
    int outindex[343];

    fusificar(e,sisfuzzy->e);
    fusificar(inte,sisfuzzy->inte);
    fusificar(epunto,sisfuzzy->epunto);
    match(sisfuzzy->e, sisfuzzy->inte,sisfuzzy->epunto, pos);
    return inf_desf(sisfuzzy, pos, outindex);
}



double inf_desf(SISTEMA_DIFUSO *sisfuzzy, int *pos, int *outindex){
    int i, imax;
    int j, jmax;
    int k, kmax;
    int n=0;
    double premisa,
         numerador=0,
         denominador=0;

    jmax=(sisfuzzy->e->numf-1)-pos[0];       // j lleva a e.
    imax=(sisfuzzy->epunto->numf-1)-pos[1];  // i lleva a epunto.
    kmax=(sisfuzzy->inte->numf-1)-pos[2];    // k lleva a inte.

// Caso (a): imax, jmax, y kmax son mayores que cero.

    if(imax>0 && jmax>0 && kmax>0){
        for(j=0;j<(jmax+1);j++){
            for(i=0;i<(imax+1);i++){
                for(k=0;k<(kmax+1);k++){

                    outindex[n]=matriz[pos[0]+j][pos[1]+i][pos[2]+k];  //almacena co
njunto inferido
                    if(sisfuzzy->parametro->m_conj==2)    //conj. por producto
                        premisa=sisfuzzy->e->gdm[pos[0]+j]*sisfuzzy->epunto->gdm[pos
[1]+i]*sisfuzzy->inte->gdm[pos[2]+k];
                    else{                                  //conj. por minimo
                        premisa=MIN(sisfuzzy->e->gdm[pos[0]+j],sisfuzzy->epunto->gdm
[pos[1]+i]);
                        premisa=MIN(premisa,sisfuzzy->inte->gdm[pos[2]+k]);
                    }

                    //DESFUSIFICACION

                    if(sisfuzzy->parametro->m_desf==2){    //COA
                        denominador+=premisa;
                        numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
                    }
                    else {
```

```
                        if(sisfuzzy->parametro->m_inf==2){
                            denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]
];
                            numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*
sisfuzzy->salida->centro[outindex[n]];
                        }
                        else{
                            denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salid
a->ancho[outindex[n]];
                            numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida-
>ancho[outindex[n]]*sisfuzzy->salida->centro[outindex[n]];
                        }
                    }
                    n+=1;
                }
            }
        }
    }

// Caso (b): imax>0, jmax=0, y kmax>0

    else if(imax>0 && jmax==0 && kmax>0){
        for(i=0;i<(imax+1);i++){
            for(k=0;k<(kmax+1);k++){
                outindex[n]=matriz[pos[0]][pos[1]+i][pos[2]+k];
                if(sisfuzzy->parametro->m_conj==2)     //conj. por producto
                    premisa=sisfuzzy->e->gdm[pos[0]]*sisfuzzy->epunto->gdm[pos[1]+i]
*sisfuzzy->inte->gdm[pos[2]+k];
                else{                                          //conj. por minimo
                    premisa=MIN(sisfuzzy->e->gdm[pos[0]],sisfuzzy->epunto->gdm[pos[1
]+i]);
                    premisa=MIN(premisa,sisfuzzy->inte->gdm[pos[2]+k]);
                }

                //DESFUSIFICACION

                if(sisfuzzy->parametro->m_desf==2){     //COA
                    denominador+=premisa;
                    numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
                }
                else {                                         //COG
                    if(sisfuzzy->parametro->m_inf==2){
                        denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]];
                        numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*sisf
uzzy->salida->centro[outindex[n]];
                    }
                    else{
                        denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->a
ncho[outindex[n]];
                        numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->anc
ho[outindex[n]]*sisfuzzy->salida->centro[outindex[n]];
                    }
                }
                n+=1;
             }
        }
    }

// Caso (c):  imax=0  jmax>0 kmax>0

    else if(imax==0 && jmax>0 && kmax>0){
        for(j=0;j<(jmax+1);j++){
            for(k=0;k<(kmax+1);k++){
                outindex[n]=matriz[pos[0]+j][pos[1]][pos[2]+k];
                if(sisfuzzy->parametro->m_conj==2)     //conj. por producto
                    premisa=sisfuzzy->e->gdm[pos[0]+j]*sisfuzzy->epunto->gdm[pos[1]]
*sisfuzzy->inte->gdm[pos[2]+k];
                else{                                          //conj. por minimo
                    premisa=MIN(sisfuzzy->e->gdm[pos[0]+j],sisfuzzy->epunto->gdm[pos
[1]]);
                    premisa=MIN(premisa,sisfuzzy->inte->gdm[pos[2]+k]);
                }

                //DESFUSIFICACION
```

```
                if(sisfuzzy->parametro->m_desf==0){     //COA
                    denominador+=premisa;
                    numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
                }
                else {                                  //COG
                    if(sisfuzzy->parametro->m_inf==2){
                        denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]];
                        numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*sisf
uzzy->salida->centro[outindex[n]];
                    }
                    else{
                        denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->a
ncho[outindex[n]];
                        numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->anc
ho[outindex[n]]*sisfuzzy->salida->centro[outindex[n]];
                    }
                }
                n+=1;
            }
        }
    }

//  Caso (d): imax=0  jmax=0 kmax>0
    else if(imax==0 && jmax==0 && kmax>0){
        for(k=0;k<(kmax+1);k++){
            outindex[n]=matriz[pos[0]][pos[1]][pos[2]+k];
            if(sisfuzzy->parametro->m_conj==2)     //conj. por producto
                premisa=sisfuzzy->e->gdm[pos[0]]*sisfuzzy->epunto->gdm[pos[1]]*sisfu
zzy->inte->gdm[pos[2]+k];
            else{                                   //conj. por minimo
                premisa=MIN(sisfuzzy->e->gdm[pos[0]],sisfuzzy->epunto->gdm[pos[1]]);
                premisa=MIN(premisa,sisfuzzy->inte->gdm[pos[2]+k]);
            }

            //DESFUSIFICACION

            if(sisfuzzy->parametro->m_desf==2){     //COA
                denominador+=premisa;
                numerador+=premisa*sisfuzzy->salida->centro[outindex[0]];
            }
            else {                                  //COG
                if(sisfuzzy->parametro->m_inf==2){
                    denominador+=premisa*sisfuzzy->salida->ancho[outindex[0]];
                    numerador+=premisa*sisfuzzy->salida->ancho[outindex[0]]*sisfuzzy
->salida->centro[outindex[0]];
                }
                else{
                    denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->ancho
[outindex[0]];
                    numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->ancho[o
utindex[0]]*sisfuzzy->salida->centro[outindex[0]];
                }
            }
            n+=1;
        }
    }

// Caso (e):  imax>0  jmax>0 kmax==0

    else if(imax>0 && jmax>0 && kmax==0){
        for(j=0;j<(jmax+1);j++){
            for(i=0;i<(imax+1);i++){
                outindex[n]=matriz[pos[0]+j][pos[1]+i][pos[2]];
                if(sisfuzzy->parametro->m_conj==2)     //conj. por producto
                    premisa=sisfuzzy->e->gdm[pos[0]+j]*sisfuzzy->epunto->gdm[pos[1]+
i]*sisfuzzy->inte->gdm[pos[2]];
                else{                                   //conj. por minimo
                    premisa=MIN(sisfuzzy->e->gdm[pos[0]+j],sisfuzzy->epunto->gdm[pos
[1]+i]);
                    premisa=MIN(premisa,sisfuzzy->inte->gdm[pos[2]]);
                }

                //DESFUSIFICACION
```

```
                if(sisfuzzy->parametro->m_desf==0){     //COA
                    denominador+=premisa;
                    numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
                }
                else {                                  //COG
                    if(sisfuzzy->parametro->m_inf==2){
                        denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]];
                        numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*sisf
uzzy->salida->centro[outindex[n]];
                    }
                    else{
                        denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->a
ncho[outindex[n]];
                        numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->anc
ho[outindex[n]]*sisfuzzy->salida->centro[outindex[n]];
                    }
                }
                n+=1;
            }
        }
    }

// Caso (f):  imax>0  jmax==0 kmax==0

    else if(imax>0 && jmax==0 && kmax==0){
        for(i=0;i<(imax+1);i++){
                outindex[n]=matriz[pos[0]][pos[1]+i][pos[2]];
                if(sisfuzzy->parametro->m_conj==2)    //conj. por producto
                    premisa=sisfuzzy->e->gdm[pos[0]]*sisfuzzy->epunto->gdm[pos[1]+i]
*sisfuzzy->inte->gdm[pos[2]];
                else{                                 //conj. por minimo
                    premisa=MIN(sisfuzzy->e->gdm[pos[0]],sisfuzzy->epunto->gdm[pos[1
]+i]);
                    premisa=MIN(premisa,sisfuzzy->inte->gdm[pos[2]]);
                }

                //DESFUSIFICACION

                if(sisfuzzy->parametro->m_desf==0){     //COA
                    denominador+=premisa;
                    numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
                }
                else {                                  //COG
                    if(sisfuzzy->parametro->m_inf==2){
                        denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]];
                        numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*sisf
uzzy->salida->centro[outindex[n]];
                    }
                    else{
                        denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->a
ncho[outindex[n]];
                        numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->anc
ho[outindex[n]]*sisfuzzy->salida->centro[outindex[n]];
                    }
                }
                n+=1;
        }
    }

// Caso (g):  imax=0  jmax>0 kmax==0
    else if(imax==0 && jmax>0 && kmax==0){
        for(j=0;j<2;j++){
                outindex[n]=matriz[pos[0]+j][pos[1]][pos[2]];
                if(sisfuzzy->parametro->m_conj==2)    //conj. por producto
                    premisa=sisfuzzy->e->gdm[pos[0]+j]*sisfuzzy->epunto->gdm[pos[1]]
*sisfuzzy->inte->gdm[pos[2]];
                else{                                 //conj. por minimo
                    premisa=MIN(sisfuzzy->e->gdm[pos[0]+j],sisfuzzy->epunto->gdm[pos
[1]]);
                    premisa=MIN(premisa,sisfuzzy->inte->gdm[pos[2]]);
                }

                //DESFUSIFICACION       5
```

```
                if(sisfuzzy->parametro->m_desf==0){     //COA
                    denominador+=premisa;
                    numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
                }
                else {                                  //COG
                    if(sisfuzzy->parametro->m_inf==2){
                        denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]];
                        numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*sisf
uzzy->salida->centro[outindex[n]];
                    }
                    else{
                        denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->a
ncho[outindex[n]];
                        numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->anc
ho[outindex[n]]*sisfuzzy->salida->centro[outindex[n]];
                    }
                }
                n+=1;
        }
    }
// Caso (h): imax=0 jmax=0  kmax=0
    else{
                outindex[0]=matriz[pos[0]][pos[1]][pos[2]];
                if(sisfuzzy->parametro->m_conj==2)     //conj. por producto
                    premisa=sisfuzzy->e->gdm[pos[0]]*sisfuzzy->epunto->gdm[pos[1]]*s
isfuzzy->inte->gdm[pos[2]];
                else{                                   //conj. por minimo
                    premisa=MIN(sisfuzzy->e->gdm[pos[0]],sisfuzzy->epunto->gdm[pos[1
]]);
                    premisa=MIN(premisa,sisfuzzy->inte->gdm[pos[2]]);
                }

                //DESFUSIFICACION

                if(sisfuzzy->parametro->m_desf==0){     //COA
                    denominador+=premisa;
                    numerador+=premisa*sisfuzzy->salida->centro[outindex[n]];
                }
                else {                                  //COG
                    if(sisfuzzy->parametro->m_inf==2){
                        denominador+=premisa*sisfuzzy->salida->ancho[outindex[n]];
                        numerador+=premisa*sisfuzzy->salida->ancho[outindex[n]]*sisf
uzzy->salida->centro[outindex[n]];
                    }
                    else{
                        denominador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->a
ncho[outindex[n]];
                        numerador+=(premisa-premisa*premisa/2)*sisfuzzy->salida->anc
ho[outindex[n]]*sisfuzzy->salida->centro[outindex[n]];
                    }
                }
    }

    return numerador/denominador;
}

void match(const MFIN *e, const MFIN *inte,const MFIN *epunto, int *pos){
    int i;

    for(i=0;i<e->numf;i++){
        if(e->gdm[i]!=0){
            pos[0]=i;
            break;
        }
    }
    for(i=0;i<epunto->numf;i++){
        if(epunto->gdm[i]!=0){
            pos[1]=i;
            break;
        }
    }
    for(i=0;i<inte->numf;i++){
```

```
            if(inc>gdm[i]){
                pos[2]=i;
                break;
            }
        }
    }
}


void fusificar(double valor, MFIN *mem) {

    int i;
    int r=mem->numf;

    mem->gdm[0]=saturada_izq(valor,mem->centro[0],mem->hwr[0]);
    mem->gdm[r-1]=saturada_der(valor,mem->centro[r-1],mem->hwl[r-1]);

    for(i=1;i<r-1;i++){
        if(valor<mem->centro[i])
            mem->gdm[i]=l_izq(valor,mem->centro[i],mem->hwl[i]);
        else
            mem->gdm[i]=l_der(valor,mem->centro[i],mem->hwr[i+1]);
    }
}


double l_der(double u, double c, double w){
    return MAX(0,(1-(u-c)/w));
}

double l_izq(double u, double c, double w){
    return MAX(0,(1-(c-u)/w));
}

double saturada_izq(double u, double c, double w) {
    if(u<c)
        return 1.0;
    else
        return MAX(0,(1-(u-c)/w));
}

double saturada_der(double u, double c, double w) {
    if(u>c)
        return 1.0;
    else
        return MAX(0,(1-(c-u)/w));
}



void asignar_memoria(SISTEMA_DIFUSO *sisfuzzy) {

// Asignacion de memoria para las variables de la estructura de
// tipo SISTEMA_DIFUSO

    sisfuzzy->e=new MFIN[1];
    sisfuzzy->inte=new MFIN[1];
    sisfuzzy->epunto=new MFIN[1];
    sisfuzzy->salida=new MFOUT[1];
    sisfuzzy->parametro=new PARAM[1];

}


void liberar_memoria(SISTEMA_DIFUSO *sisfuzzy) {

    delete(sisfuzzy->e->centro);
    delete(sisfuzzy->e->hwl);
    delete(sisfuzzy->e->hwr);
    delete(sisfuzzy->e->gdm);

    delete(sisfuzzy->inte->centro);
```

```
    delete(sisfuzzy->inte->hwl);
    delete(sisfuzzy->inte->hwr);
    delete(sisfuzzy->inte->gdm);

    delete(sisfuzzy->epunto->centro);
    delete(sisfuzzy->epunto->hwl);
    delete(sisfuzzy->epunto->hwr);
    delete(sisfuzzy->epunto->gdm);

    delete(sisfuzzy->salida->centro);
    delete(sisfuzzy->salida->ancho);

    delete(sisfuzzy->e);
    delete(sisfuzzy->inte);
    delete(sisfuzzy->epunto);
    delete(sisfuzzy->salida);
    delete(sisfuzzy->parametro);
}

void lectura(SISTEMA_DIFUSO *sisfuzzy, PUNTO *mem,MODELO *model){
    int numero;
    int     i;
    int j;
    int k;
    int dato;
    char    marcador1;
    char    marcador2;

    ifstream fin("fuzzyin.txt");

    fin>>marcador1>>marcador2;
    if(marcador1=='*' && marcador2=='a'){   //Datos de las mf's del error
        fin>>numero;
        asigfuzzyin(sisfuzzy->e,numero); //asig. memoria del error

// lectura de datos del error, calculo y ubicacion en
// sus respectivas matrices y variables.

        for(i=0;i<numero;i++){
            fin>>mem->p1[i]>>mem->p2[i]>>mem->p3[i];
            sisfuzzy->e->hwl[i]=mem->p2[i]-mem->p1[i];
            sisfuzzy->e->hwr[i]=mem->p3[i]-mem->p2[i];
            sisfuzzy->e->centro[i]=mem->p2[i];
        }
    fin>>marcador1>>marcador2;
    }
    if(marcador1=='*' && marcador2=='b'){  // Datos de las mf's epunto
        fin>>numero;
        asigfuzzyin(sisfuzzy->epunto,numero);

        // lectura de datos del c. de error, calculo y ubicacion en
        // sus respectivas matrices y variables.
        for(i=0;i<numero;i++){
            fin>>mem->p1[i]>>mem->p2[i]>>mem->p3[i];
            sisfuzzy->epunto->hwl[i]=mem->p2[i]-mem->p1[i];
            sisfuzzy->epunto->hwr[i]=mem->p3[i]-mem->p2[i];
            sisfuzzy->epunto->centro[i]=mem->p2[i];
        }
        fin>>marcador1>>marcador2;
    }
    if(marcador1=='*' && marcador2=='*'){   //Datos de las mf's del ie
        fin>>numero;
        asigfuzzyin(sisfuzzy->inte,numero); //asig. memoria del error

        // lectura de datos del error, calculo y ubicacion en
        // sus respectivas matrices y variables.
        for(i=0;i<numero;i++){
            fin>>mem->p1[i]>>mem->p2[i]>>mem->p3[i];
            sisfuzzy->inte->hwl[i]=mem->p2[i]-mem->p1[i];
            sisfuzzy->inte->hwr[i]=mem->p3[i]-mem->p2[i];
            sisfuzzy->inte->centro[i]=mem->p2[i];
        }
    fin>>marcador1>>marcador2;
    }
```

```
    if(marcador1=='*' && marcador2=='c'){ // Datos de las mf's de salida
        fin>>numero;
        asigfuzzyout(sisfuzzy->salida,numero);

        // lectura de datos de salida, calculo y ubicacion en
        // sus respectivas matrices y variables.
        for(i=0;i<numero;i++){
            fin>>mem->p1[i]>>mem->p2[i]>>mem->p3[i];
            sisfuzzy->salida->ancho[i]=mem->p3[i]-mem->p1[i];
            sisfuzzy->salida->centro[i]=mem->p2[i];
        }
        fin>>marcador1>>marcador2;
    }
    if(marcador1=='*' && marcador2=='d'){ // Datos de la matriz
        // lectura de datos de la matriz
        for(k=0;k<sisfuzzy->inte->numf;k++){
            for(i=0;i<sisfuzzy->e->numf;i++){
                for(j=0;j<sisfuzzy->epunto->numf;j++){
                    fin>>dato;
                    matriz[i][j][k]=dato+(numero-1)/2;
                }
            }
        }
        fin>>marcador1>>marcador2;
    }
    if(marcador1=='*' && marcador2=='e'){ // Datos de sistema fuzzy y modelo
        fin>>sisfuzzy->parametro->m_conj;
        fin>>sisfuzzy->parametro->m_inf;
        fin>>sisfuzzy->parametro->m_desf;
        fin>>model->p1;
        fin>>model->tipo;
        fin>>model->roz;
        fin>>mem->g0;
        fin>>mem->g1;
        fin>>mem->g3;
        fin>>mem->g2;
    }
    else
        exit(1);

    for(i=0;i<sisfuzzy->e->numf;i++){
        sisfuzzy->e->centro[i]*=mem->g0;
        sisfuzzy->e->hwl[i]*=mem->g0;
        sisfuzzy->e->hwr[i]*=mem->g0;
    }

    for(i=0;i<sisfuzzy->epunto->numf;i++){
        sisfuzzy->epunto->centro[i]*=mem->g1;
        sisfuzzy->epunto->hwl[i]*=mem->g1;
        sisfuzzy->epunto->hwr[i]*=mem->g1;
    }
    for(i=0;i<sisfuzzy->inte->numf;i++){
        sisfuzzy->inte->centro[i]*=mem->g2;
        sisfuzzy->inte->hwl[i]*=mem->g2;
        sisfuzzy->inte->hwr[i]*=mem->g2;
    }
    for(i=0;i<sisfuzzy->salida->numf;i++){
        sisfuzzy->salida->centro[i]*=mem->g3;
        sisfuzzy->salida->ancho[i]*=mem->g3;
    }
    fin.close();
}

void asig(PUNTO *mem){
    mem->p1=new double[7];
    mem->p2=new double[7];
    mem->p3=new double[7];
}

void asigfuzzyin(MFIN *mem,int numero){

    mem->numf=numero;
    mem->centro=new double[numero];
    mem->gdm=new double[numero];
```

```
    mem->hwl=new double [numero];
    mem->hwr=new double [numero];
}

void asigfuzzyout(MFOUT *mem,int numero){

    mem->numf=numero;
    mem->centro=new double[numero];
    mem->ancho=new double[numero];
}
```

```
// fuzzypd.cpp
// Control Difuso de un Elevador
// Realizado por: Alberto Sanchez Teran
// Director: Ing. Marco Barragan
// Facultad de Ingenieria Electrica
// Escuela Politecnica Nacional

#include <iostream.h>
#include <math.h>
#include "fuzzy.h"

#define h 0.001

// variables globales

SISTEMA_DIFUSO sis;
double input,er,erp;

// Area de definicion de estructuras

typedef struct rk_struct{
    double *f1;
    double *f2;
    double *f3;
    double *f4;
    double *arg2;
} RK_STRUCT;

// Area de prototipo de funciones

void rk4(RK_STRUCT *rks, double *t, double *x, MODELO *model);
void deriv(const double *x, double *xpunto,MODELO *model, double t);
void F1(const double *x,double *f1, MODELO *model, double t);
void F2(const double *x,double *f1, double *f2, double *arg2, MODELO *model, double
t);
void F3(const double *x,double *f2, double *f3, double *arg2, MODELO *model, double
t);
void F4(const double *x,double *f3, double *f4, double *arg2, MODELO *model, double
t);
void rk_init(RK_STRUCT *rks);
void rk_free(RK_STRUCT *rks);
void calmod(MODELO *model);
double froz(int m, double N, double v);
double R1(double t);
double R2(double t);

// Area de implementacion de funciones
double R1(double t){
    if(t<=0.6){
        return 1.25*t*t;
    }
    else if(t>0.6 && t<=2.4){
        return 1.5*t-0.45;
    }
    else if(t>2.4 && t<=3){
        return -1.25*t*t+7.5*t-7.65;
    }
    else
        return 3.6;
}

double R2(double t){
    if(t<=0.6){
        return 2.5*t;
    }
    else if(t>0.6 && t<=2.4){
        return 1.5;
    }
    else if(t>2.4 && t<=3){
        return -2.5*t+7.5;
    }
    else
        return 0;
}
```

```
double froz(int m, double N, double v){

    if(m==1){
        if(v>0)
            return 9.8*N*0.05;
        else if(v<0)
            return -9.8*N*0.05;
        else
            return 0.0;
    }
    else {
        if(v>0)
            return 9.8*N*(0.05*exp(-25*v)+0.05);
        else if(v<0)
            return -9.8*N*(0.05*exp(-25*v)+0.05);
        else
            return 0.0;
    }
}

void calmod(MODELO *model){
    double J;

    model->deltap=(247-model->p1)*9.8;
    J=4.23801+0.0961*(1427+model->p1);
    model->k1=0.0961/J;
    model->k2=8.37/J;
}
void deriv(const double *x, double *xpunto,MODELO *model,double t){
    double e,edot,u;
    double r1,r2;

    r1=R1(t);
    r2=R2(t);

    e=r1-x[0];
    edot=r2-x[1];
    er=e;
    erp=edot;
    u=fuzzy_control(e,edot,&sis);
    input=u;
    xpunto[0]=x[1];
    xpunto[1]=model->k1*model->deltap-model->k1*froz(model->tipo,model->roz,x[1])-mo
del->k1*0.0531*x[1]+model->k2*x[2];
    xpunto[2]=-37.037*x[2]+138.889*u;

}

void rk4(RK_STRUCT *rks, double *t, double *x, MODELO *model){
    int i;

    F1(x,rks->f1,model,*t);
    F2(x,rks->f1,rks->f2,rks->arg2,model,*t);
    F3(x,rks->f2,rks->f3,rks->arg2, model,*t);
    F4(x,rks->f3,rks->f4,rks->arg2, model,*t);

    *t+=h;

    for(i=0;i<3;i++)
        x[i]+=(1.0/6.0)*(rks->f1[i]+2*(rks->f2[i])+2*(rks->f3[i])+rks->f4[i]);
}

void F1(const double *x, double *f1, MODELO *model,double t){
    int i;
    deriv(x,f1,model,t);
    for(i=0; i<3;i++)
        f1[i]=h*f1[i];
}

void F2(const double *x, double *f1, double *f2, double *arg2,MODELO *model, double
t){
    int i;
```

```
    for(i=0; i<3;i++)
        arg2[i]=x[i]+0.5*f1[i];

    deriv(arg2,f2,model,(t+0.5*h));

    for(i=0;i<3;i++)
        f2[i]=h*f2[i];
}

void F3(const double *x, double *f2, double *f3, double *arg2,MODELO *model, double
t){
    int i;

    for(i=0;i<3;i++)
        arg2[i]=x[i]+0.5*f2[i];

    deriv(arg2,f3,model,(t+0.5*h));

    for(i=0;i<3;i++)
        f3[i]=h*f3[i];
}

void F4(const double *x, double *f3, double *f4, double *arg2,MODELO *model, double
t){
    int i;

    for(i=0; i<3;i++)
        arg2[i]=x[i]+f3[i];

    deriv(arg2,f4,model,(t+h));

    for(i=0;i<3;i++)
        f4[i]=h*f4[i];
}

void rk_init(RK_STRUCT *rks){

    rks->f1=new double[3];
    rks->f2=new double[3];
    rks->f3=new double[3];
    rks->f4=new double[3];
    rks->arg2=new double[3];
}

void rk_free(RK_STRUCT *rks){
    delete(rks->f1);
    delete(rks->f2);
    delete(rks->f3);
    delete(rks->f4);
    delete(rks->arg2);
}

void main(void){

    double
        t=0,
        x[3],
        tmax=4;

    RK_STRUCT rks;
    PUNTO   mem;
    MODELO model;

    rk_init(&rks);
    asignar_memoria(&sis);
    asig(&mem);
    lectura(&sis,&mem,&model);
    calmod(&model);

    x[0]=0;
    x[1]=0;
    if(model.deltap>=0.1*model.roz)
        x[2]=0;
    else
```

```
        x[2]=(-model.deltap+0.1*model.roz)*0.31/27;

    ofstream fout("fuzzyout.txt");
    while(t<tmax){
        rk4(&rks,&t,x,&model);
        fout<<t<<" "<<x[0]<<" "<<x[1]<<" "<<input<<" "<<er<<" "<<erp<<"\n";
    }
    fout.close();
    liberar(&mem);
    liberar_memoria(&sis);
    rk_free(&rks);

}
```

```
// fuzzypid.cpp
// Control Difuso de un Elevador
// Realizado por: Alberto Sanchez Teran
// Director: Ing. Marco Barragan
// Facultad de Ingenieria Electrica
// Escuela Politecnica Nacional

#include <iostream.h>
#include <math.h>
#include "fuzzypid.h"

#define h 0.001

// variables globales

SISTEMA_DIFUSO sis;
double input,er,erp,ier;

// Area de definicion de estructuras

typedef struct rk_struct{
    double *f1;
    double *f2;
    double *f3;
    double *f4;
    double *arg2;
} RK_STRUCT;

// Area de prototipo de funciones

void rk4(RK_STRUCT *rks, double *t, double *x, MODELO *model);
void deriv(const double *x, double *xpunto,MODELO *model, double t);
void F1(const double *x,double *f1, MODELO *model, double t);
void F2(const double *x,double *f1, double *f2, double *arg2, MODELO *model, double
t);
void F3(const double *x,double *f2, double *f3, double *arg2, MODELO *model, double
t);
void F4(const double *x,double *f3, double *f4, double *arg2, MODELO *model, double
t);
void rk_init(RK_STRUCT *rks);
void rk_free(RK_STRUCT *rks);
void calmod(MODELO *model);
double froz(int m, double N, double v);
double R1(double t);
double R2(double t);
double R3(double t);

// Area de implementacion de funciones

double R1(double t){
    if(t<=0.6){
        return 1.25*t*t;
    }
    else if(t>0.6 && t<=2.4){
        return 1.5*t-0.45;
    }
    else if(t>2.4 && t<=3){
        return -1.25*t*t+7.5*t-7.65;
    }
    else
        return 3.6;
}

double R2(double t){
    if(t<=0.6){
        return 2.5*t;
    }
    else if(t>0.6 && t<=2.4){
        return 1.5;
    }
    else if(t>2.4 && t<=3){
        return -2.5*t+7.5;
    }
    else
```

```
        return 0;
}

double R3(double t){
    if(t<=0.6){
        return 0.41667*t*t*t;
    }
    else if(t>0.6 && t<=2.4){
        return 0.75*t*t-0.45*t+0.09;
    }
    else if(t>2.4 && t<=3){
        return -0.41667*t*t*t+3.75*t*t-7.65*t+5.85;
    }
    else
        return 3.6*t-5.4;
}

double froz(int m, double N, double v){

    if(m==1){
        if(v>0)
            return 9.8*N*0.05;
        else if(v<0)
            return -9.8*N*0.05;
        else
            return 0.0;
    }
    else {
        if(v>0)
            return 9.8*N*(0.05*exp(-25*v)+0.05);
        else if(v<0)
            return -9.8*N*(0.05*exp(-25*v)+0.05);
        else
            return 0.0;
    }
}

void calmod(MODELO *model){
    double J;

    model->deltap=(247-model->p1)*9.8;
    J=4.28801+0.0961*(1427+model->p1);
    model->k1=0.0961/J;
    model->k2=8.37/J;
}
void deriv(const double *x, double *xpunto,MODELO *model,double t){
    double e,ie,ep,u;
    double r1,r2,r3;

    r1=R1(t);
    r2=R2(t);
    r3=R3(t);
    e=r1-x[0];
    ie=r3-x[3];
    ep=r2-x[1];
    er=e;
    ier=ie;
    erp=ep;

    u=fuzzy_control(e,ie,ep,&sis);
    input=u;
    xpunto[0]=x[1];
    zpunto[1]=model->k1*model->deltap-model->k1*froz(model->tipo,model->roz,x[1])-mo
del->k1*0.0531*x[1]+model->k2*x[2];
    xpunto[2]=-37.037*x[2]+138.889*u;
    xpunto[3]=x[0];

}

void rk4(RK_STRUCT *rks, double *t, double *x, MODELO *model){
    int i;

    F1(x,rks->f1,model,*t);
    F2(x,rks->f1,rks->f2,rks->arg2,model,*t);
```

```
    F3(x,rks->f2,rks->f3,rks->arg2, model,*t);
    F4(x,rks->f3,rks->f4,rks->arg2, model,*t);

    *t+=h;

    for(i=0;i<4;i++)
        x[i]+=(1.0/6.0)*(rks->f1[i]+2*(rks->f2[i])+2*(rks->f3[i])+rks->f4[i]);
}

void F1(const double *x, double *f1, MODELO *model,double t){
    int i;
    deriv(x,f1,model,t);
    for(i=0; i<4;i++)
        f1[i]=h*f1[i];
}

void F2(const double *x, double *f1, double *f2, double *arg2,MODELO *model, double
t){
    int i;

    for(i=0; i<4;i++)
        arg2[i]=x[i]+0.5*f1[i];

    deriv(arg2,f2,model,(t+0.5*h));

    for(i=0;i<4;i++)
        f2[i]=h*f2[i];
}

void F3(const double *x, double *f2, double *f3, double *arg2,MODELO *model, double
t){
    int i;

    for(i=0;i<4;i++)
        arg2[i]=x[i]+0.5*f2[i];

    deriv(arg2,f3,model,(t+0.5*h));

    for(i=0;i<4;i++)
        f3[i]=h*f3[i];
}

void F4(const double *x, double *f3, double *f4, double *arg2,MODELO *model, double
t){
    int i;

    for(i=0; i<4;i++)
        arg2[i]=x[i]+f3[i];

    deriv(arg2,f4,model,(t+h));

    for(i=0;i<4;i++)
        f4[i]=h*f4[i];
}

void rk_init(RK_STRUCT *rks){

    rks->f1=new double[4];
    rks->f2=new double[4];
    rks->f3=new double[4];
    rks->f4=new double[4];
    rks->arg2=new double[4];
}

void rk_free(RK_STRUCT *rks){
    delete(rks->f1);
    delete(rks->f2);
    delete(rks->f3);
    delete(rks->f4);
    delete(rks->arg2);
}

void main(void){
```

```
    double
        t=0,
        x[4],
        tmax=4;

    RK_STRUCT rks;
    PUNTO   mem;
    MODELO model;

    rk_init(&rks);
    asignar_memoria(&sis);
    asig(&mem);
    lectura(&sis,&mem,&model);
    calmod(&model);

    x[0]=0;
    x[1]=0;
    if(model.deltap>=0.1*model.roz)
        x[2]=0;
    else
        x[2]=(-model.deltap+0.1*model.roz)*0.31/27;
    x[3]=0;

    ofstream fout("fuzzyout.txt");
    while(t<tmax){
        rk4(&rks,&t,x,&model);
        fout<<t<<" "<<x[0]<<" "<<x[1]<<" "<<input<<" "<<er<<" "<<erp<<" "<<ier<<"\n"
;
    }
    fout.close();
    liberar(&mem);
    liberar_memoria(&sis);
    rk_free(&rks);

}
```

```
// pasopd.cpp
// Control Difuso de un Elevador
// Realizado por: Alberto Sanchez Teran
// Director: Ing. Marco Barragan
// Facultad de Ingenieria Electrica
// Escuela Politecnica Nacional

#include <iostream.h>
#include <math.h>
#include "fuzzy.h"

#define h 0.001

// variables globales

SISTEMA_DIFUSO sis;
double input,er,erp;

// Area de definicion de estructuras

typedef struct rk_struct{
    double *f1;
    double *f2;
    double *f3;
    double *f4;
    double *arg2;
} RK_STRUCT;

// Area de prototipo de funciones

void rk4(RK_STRUCT *rks, double *t, double *x, MODELO *model);
void deriv(const double *x, double *xpunto,MODELO *model, double t);
void F1(const double *x,double *f1, MODELO *model, double t);
void F2(const double *x,double *f1, double *f2, double *arg2, MODELO *model, double
t);
void F3(const double *x,double *f2, double *f3, double *arg2, MODELO *model, double
t);
void F4(const double *x,double *f3, double *f4, double *arg2, MODELO *model, double
t);
void rk_init(RK_STRUCT *rks);
void rk_free(RK_STRUCT *rks);
void calmod(MODELO *model);
double froz(int m, double N, double v);

// Area de implementacion de funciones

double froz(int m, double N, double v){

    if(m==1){
        if(v>0)
            return 9.8*N*0.05;
        else if(v<0)
            return -9.8*N*0.05;
        else
            return 0.0;
    }
    else {
        if(v>0)
            return 9.8*N*(0.05*exp(-25*v)+0.05);
        else if(v<0)
            return -9.8*N*(0.05*exp(-25*v)+0.05);
        else
            return 0.0;
    }
}

void calmod(MODELO *model){
    double J;

    model->deltap=(247-model->p1)*9.8;
    J=4.23801+0.0961*(1427+model->p1);
    model->k1=0.0961/J;
    model->k2=8.37/J;
}
```

```
void deriv(const double *x, double *xpunto,MODELO *model,double t){
    double e,edot,u;
    double r1,r2;

    r1=1;
    r2=0;
    e=r1-x[0];
    edot=r2-x[1];
    er=e;
    erp=edot;
    u=fuzzy_control(e,edot,&sis);
    input=u;
    xpunto[0]=x[1];
    xpunto[1]=model->k1*model->deltap-model->k1*froz(model->tipo,model->roz,x[1])-mo
del->k1*0.0531*x[1]+model->k2*x[2];
    xpunto[2]=-37.037*x[2]+138.889*u;

}

void rk4(RK_STRUCT *rks, double *t, double *x, MODELO *model){
    int i;

    F1(x,rks->f1,model,*t);
    F2(x,rks->f1,rks->f2,rks->arg2,model,*t);
    F3(x,rks->f2,rks->f3,rks->arg2, model,*t);
    F4(x,rks->f3,rks->f4,rks->arg2, model,*t);

    *t+=h;

    for(i=0;i<3;i++)
        x[i]+=(1.0/6.0)*(rks->f1[i]+2*(rks->f2[i])+2*(rks->f3[i])+rks->f4[i]);
}

void F1(const double *x, double *f1, MODELO *model,double t){
    int i;
    deriv(x,f1,model,t);
    for(i=0; i<3;i++)
        f1[i]=h*f1[i];
}

void F2(const double *x, double *f1, double *f2, double *arg2,MODELO *model, double
t){
    int i;

    for(i=0; i<3;i++)
        arg2[i]=x[i]+0.5*f1[i];

    deriv(arg2,f2,model,(t+0.5*h));

    for(i=0;i<3;i++)
        f2[i]=h*f2[i];
}

void F3(const double *x, double *f2, double *f3, double *arg2,MODELO *model, double
t){
    int i;

    for(i=0;i<3;i++)
        arg2[i]=x[i]+0.5*f2[i];

    deriv(arg2,f3,model,(t+0.5*h));

    for(i=0;i<3;i++)
        f3[i]=h*f3[i];
}

void F4(const double *x, double *f3, double *f4, double *arg2,MODELO *model, double
t){
    int i;

    for(i=0; i<3;i++)
        arg2[i]=x[i]+f3[i];

    deriv(arg2,f4,model,(t+h));
```

```
    for(i=0;i<3;i++)
        f4[i]=h*f4[i];
}

void rk_init(RK_STRUCT *rks){

    rks->f1=new double[3];
    rks->f2=new double[3];
    rks->f3=new double[3];
    rks->f4=new double[3];
    rks->arg2=new double[3];
}

void rk_free(RK_STRUCT *rks){
    delete(rks->f1);
    delete(rks->f2);
    delete(rks->f3);
    delete(rks->f4);
    delete(rks->arg2);
}

void main(void){

    double
        t=0,
        x[3],
        tmax=4;

    RK_STRUCT rks;
    PUNTO   mem;
    MODELO model;

    rk_init(&rks);
    asignar_memoria(&sis);
    asig(&mem);
    lectura(&sis,&mem,&model);
    calmod(&model);

    x[0]=0;
    x[1]=0;
    if(model.deltap>=0.1*model.roz)
        x[2]=0;
    else
        x[2]=(-model.deltap+0.1*model.roz)*0.31/27;

    ofstream fout("fuzzyout.txt");
    while(t<tmax){
        rk4(&rks,&t,x,&model);
        fout<<t<<" "<<x[0]<<" "<<x[1]<<" "<<input<<" "<<er<<" "<<erp<<"\n";
    }
    fout.close();
    liberar(&mem);
    liberar_memoria(&sis);
    rk_free(&rks);

}
```

```
// pasopid.cpp
// Control Difuso de un Elevador
// Realizado por: Alberto Sanchez Teran
// Director: Ing. Marco Barragan
// Facultad de Ingenieria Electrica
// Escuela Politecnica Nacional

#include <iostream.h>
#include <math.h>
#include "fuzzypid.h"

#define h 0.001

// variables globales

SISTEMA_DIFUSO sis;
double input,er,erp,ier;

// Area de definicion de estructuras

typedef struct rk_struct{
        double *f1;
        double *f2;
        double *f3;
        double *f4;
        double *arg2;
} RK_STRUCT;

// Area de prototipo de funciones

void rk4(RK_STRUCT *rks, double *t, double *x, MODELO *model);
void deriv(const double *x, double *xpunto,MODELO *model, double t);
void F1(const double *x,double *f1, MODELO *model, double t);
void F2(const double *x,double *f1, double *f2, double *arg2, MODELO *model, double t);
void F3(const double *x,double *f2, double *f3, double *arg2, MODELO *model, double t);
void F4(const double *x,double *f3, double *f4, double *arg2, MODELO *model, double t);
void rk_init(RK_STRUCT *rks);
void rk_free(RK_STRUCT *rks);
void calmod(MODELO *model);
double froz(int m, double N, double v);

// Area de implementacion de funciones

double froz(int m, double N, double v){

        if(m==1){
                if(v>0)
                        return 9.8*N*0.05;
                else if(v<0)
                        return -9.8*N*0.05;
                else
                        return 0.0;
        }
        else {
                if(v>0)
                        return 9.8*N*(0.05*exp(-25*v)+0.05);
                else if(v<0)
                        return -9.8*N*(0.05*exp(-25*v)+0.05);
                else
                        return 0.0;
        }
}

void calmod(MODELO *model){
        double J;

        model->deltap=(247-model->p1)*9.8;
```

```
        J=4.23801+0.0961*(1427+model->p1);
        model->k1=0.0961/J;
        model->k2=8.37/J;
}
void deriv(const double *x, double *xpunto,MODELO *model,double t){
        double e,ie,ep,u;
        double r1,r2,r3;

        r1=1;
        r2=t;
        r3=0;
        e=r1-x[0];
        ie=r2-x[3];
        ep=r3-x[1];
        er=e;
        ier=ie;
        erp=ep;

        u=fuzzy_control(e,ie,ep,&sis);
        input=u;
        xpunto[0]=x[1];
        xpunto[1]=model->k1*model->deltap-model->k1*froz(model->tipo,model->roz,x[1])-model-
>k1*0.0531*x[1]+model->k2*x[2];
        xpunto[2]=-37.037*x[2]+138.889*u;
        xpunto[3]=x[0];

}

void rk4(RK_STRUCT *rks, double *t, double *x, MODELO *model){
        int i;

        F1(x,rks->f1,model,*t);
        F2(x,rks->f1,rks->f2,rks->arg2,model,*t);
        F3(x,rks->f2,rks->f3,rks->arg2, model,*t);
        F4(x,rks->f3,rks->f4,rks->arg2, model,*t);

        *t+=h;

        for(i=0;i<4;i++)
                x[i]+=(1.0/6.0)*(rks->f1[i]+2*(rks->f2[i])+2*(rks->f3[i])+rks->f4[i]);
}

void F1(const double *x, double *f1, MODELO *model,double t){
        int i;
        deriv(x,f1,model,t);
        for(i=0; i<4;i++)
                f1[i]=h*f1[i];
}

void F2(const double *x, double *f1, double *f2, double *arg2,MODELO *model, double t){
        int i;

        for(i=0; i<4;i++)
                arg2[i]=x[i]+0.5*f1[i];

        deriv(arg2,f2,model,(t+0.5*h));

        for(i=0;i<4;i++)
                f2[i]=h*f2[i];
}

void F3(const double *x, double *f2, double *f3, double *arg2,MODELO *model, double t){
        int i;

        for(i=0;i<4;i++)
                arg2[i]=x[i]+0.5*f2[i];
```

```
        deriv(arg2,f3,model,(t+0.5*h));

        for(i=0;i<4;i++)
                f3[i]=h*f3[i];
}

void F4(const double *x, double *f3, double *f4, double *arg2,MODELO *model, double t){
        int i;

        for(i=0; i<4;i++)
                arg2[i]=x[i]+f3[i];

        deriv(arg2,f4,model,(t+h));

        for(i=0;i<4;i++)
                f4[i]=h*f4[i];
}

void rk_init(RK_STRUCT *rks){

        rks->f1=new double[4];
        rks->f2=new double[4];
        rks->f3=new double[4];
        rks->f4=new double[4];
        rks->arg2=new double[4];
}

void rk_free(RK_STRUCT *rks){
        delete(rks->f1);
        delete(rks->f2);
        delete(rks->f3);
        delete(rks->f4);
        delete(rks->arg2);
}

void main(void){

        double
                t=0,
                x[4],
                tmax=4;

        RK_STRUCT rks;
        PUNTO mem;
        MODELO model;

        rk_init(&rks);
        asignar_memoria(&sis);
        asig(&mem);
        lectura(&sis,&mem,&model);
        calmod(&model);

        x[0]=0;
        x[1]=0;
        if(model.deltap>=0.1*model.roz)
                x[2]=0;
        else
                x[2]=(-model.deltap+0.1*model.roz)*0.31/27;
        x[3]=0;

        ofstream fout("fuzzyout.txt");
        while(t<tmax){
                rk4(&rks,&t,x,&model);
                fout<<t<<" "<<x[0]<<" "<<x[1]<<" "<<input<<" "<<er<<" "<<erp<<" "<<ier<<"\n";
        }
        fout.close();
        liberar(&mem);
```

```
        liberar_memoria(&sis);
        rk_free(&rks);

}
```

PROGRAMA EN DIRECTOR - LINGO 6.5

```
global lista,num_funci,funci,que_tri,deltax,ladoder
global yy2,yy1,xx2,xx1

******** PARA GRAFICAR TODOS LOS TRIANGULOS ********
on grafica_lista es_voltaje
  set spt=20
  set n=integer(the text of member "funci")
  set y2=the top of sprite 3+5
  set y1=the bottom of sprite 3-1

  repeat with i=21 to 41
    set the loc of sprite i to point(10000,10)
    updatestage
  end repeat
  if lista=[] then

    if not es_voltaje then
      set temp=integer(((the width of sprite 3)-30)/(n-1))
      set x1=the left of sprite 3+15
      set x2=the right of sprite 3-15
      set deltax=integer(float(x2-x1)/float(n-1))
      repeat with i=1 to n
        if i=1 then
          setat lista,1,[point(x1-14,the top of sprite 3+5),
point(the left of sprite 3+15,the top of sprite 3+5),point(t
he left of sprite 3+15+deltax,the bottom of sprite 3)]
        else if i=n then
          setat lista,i,[point(x1+(deltax*(i-2)),y1),point(x
1+(deltax*(i-1)),y2),point(x2+14,y2)]
        else
          setat lista,i,[point(x1+(deltax*(i-2)),y1),point(x
1+(deltax*(i-1)),y2),point(x1+(deltax*(i)),y1)]
        end if
      end repeat
    else
      set x1=the left of sprite 3
      set x2=the right of sprite 3
      set deltax=integer(float(x2-x1)/float(n+1))

      repeat with i=1 to n
        setat lista,i,[point(x1+(deltax*(i-1)),y1),point(x1+
(deltax*(i)),y2),point(x1+(deltax*(i+1)),y1)]
      end repeat
    end if

  else
    case que_tri of
      1:global error
        set lista=error
      2:
```

```
        global d_error
        set lista=d_error
      3:
        global voltaje
        set lista=voltaje
      4:
        global otro
        set lista=otro

    end case
  end if


  set cuenta=1
  repeat with i=1 to n
    drawLine (spt+cuenta), getat(getat(lista,i),1),getat(get
at(lista,i),2) , "dere", "izq"
    set cuenta=cuenta+1
    drawLine (spt+cuenta), getat(getat(lista,i),2),getat(get
at(lista,i),3) , "dere", "izq"
    set cuenta=cuenta+1
  end repeat
  set temp=35


  repeat with i=1 to n
    set the member of sprite (temp-1+i) to member (i+9-(n-1)
/2) of castlib "textos"
    set tempo=(getat(getat(lista,i),2))-point(27,20)
    set the loc of sprite (temp-1+i) to tempo
  end repeat
  setat (num_funci,que_tri,n)
  setup_lista
end

on funcion cuanto
  set funci=funci+cuanto
  set temp=getat(num_funci,que_tri)
  if funci<((temp-1)/2)-temp+1 then set funci=((temp-1)/2)-t
emp+1
  if funci>((temp-1)/2) then set funci=((temp-1)/2)
  set the member of sprite 43 = member (30+funci) of castlib
 "textos"
  recoge_datos ((temp-1)/2)+funci+1
end

on setup_lista
  set temp=getat(num_funci,que_tri)
  set funci=((temp-1)/2)-temp+1
  set the member of sprite 43 = member (30+funci) of castlib
```

```
 "textos"
  recoge_datos 1
end

on recoge_datos quien
  set centro=funcion_x(the loch of (getat(getat(lista,quien)
,2)))
  set izquierdo=funcion_x(the loch of getat(getat(lista,quie
n),1))
  set derecho=funcion_x(the loch of getat(getat(lista,quien)
,3))

  set the text of member "centro"=string(centro)
  set the text of member "izquierdo"=string(izquierdo)
  set the text of member "derecho"=string(derecho)
  updatestage
end

on lado cual, cuanto
  set temp=getat(num_funci,que_tri)
  set pos=((temp-1)/2)+funci+1
  case cual of
    #izquierdo: set punto = the loch of getat(getat(lista,po
s),1)
    #derecho: set punto = the loch of getat(getat(lista,pos)
,3)
    #centro: set punto =the loch of getat(getat(lista,pos),2
)
  end case
  set topeizq= the left of sprite 3
  set topeder = the right of sprite 3

  if cual=#izquierdo and punto+cuanto > topeizq and punto+cu
anto<topeder then
    -- and ladoizq+cuanto>=0 and ladoizq+cuanto<deltax/2
    set lado=integer(the text of member string(cual))+cuanto
    set the text of member string(cual)=string(lado)
    set the loch of getat(getat(lista,pos),1)=¬
the loch of getat(getat(lista,pos),1)+cuanto
    dibuja pos
  end if

  if cual=#derecho and punto+cuanto > topeizq and punto+cuan
to<topeder then
    -- and ladoder+cuanto<=0 and ladoder+cuanto>deltax/-2
    set lado=integer(the text of member string(cual))+cuanto
    set the text of member string(cual)=string(lado)
    set the loch of getat(getat(lista,pos),3)=¬
the loch of getat(getat(lista,pos),3)+cuanto
    dibuja pos
```

```
  end if

  if cual=#centro  and punto+cuanto > topeizq and punto+cuan
to<topeder then
    --  and ladocen+cuanto<=deltax/2 and ladocen+cuanto>delt
ax/-2
    set lado=integer(the text of member string(cual))+cuanto
    set the text of member string(cual)=string(lado)
    set the loch of getat(getat(lista,pos),2)=¬
the loch of getat(getat(lista,pos),2)+cuanto
    dibuja pos
  end if
end

on dibuja cuenta
  set spt=20
  drawLine (spt+(cuenta*2)-1), getat(getat(lista,cuenta),1),
getat(getat(lista,cuenta),2) , "dere", "izq"
  drawLine (spt+(cuenta*2)), getat(getat(lista,cuenta),2),ge
tat(getat(lista,cuenta),3) , "dere", "izq"
  updatestage
end


on sube cuanto
  set tempo=integer(the text of member "funci")
  set tempo=tempo+(cuanto*2)
  if tempo>7 then set tempo=7
  if tempo<3 then set tempo=3
  set the text of member "funci"=string(tempo)
  set lista=[]
  case que_tri of
    1,2,4:
      grafica_lista 0
    3:
      grafica_lista 1
  end case
end



on funcion_y x
  return integer(((yy2-yy1)/200)*(x+100)+yy1)
end

on funcion_x y
  return integer((((y-yy1)*200)/(yy2-yy1))-100)
end

on drawLine aSprite, startPoint, endPoint, leaningRightCast,
```

```
leaningLeftCast

  if not(the puppet of sprite aSprite) then puppetsprite aSp
rite, true
  set the member of sprite asprite to member leaningRightCas
t
  if not (the type of member (the member of sprite aSprite)
= #shape) then
    -- this won't work so exit
    exit
  end if

  -- startPoints and endPoints are passed as points; list "g
etat" syntax is used
  -- to get the values out

  -- bottomUpCast and topDowncast are passed as either membe
rnames or numbers

  -- first we have to determine WHICH of two shape cast memb
ers to use
  -- the one leaning left or the one leaning right

  if getat(startPoint, 1) < getat(endPoint, 1) then
    -- the points are in left to right order
    -- (we still have to determine top and bottom)

  else
    -- reverse the points
    set tempHolder = startPoint
    set startPoint = endPoint
    set endPoint = tempHolder

  end if

  set left = getat(startPoint, 1)
  set right = getat(endPoint, 1)

  -- now check for a "no size" effect where the x's are the
same
  -- add 1 to the right side if the numbers are identical
  if left = right then set right = right + 1


  -- now for top and bottom
  if getat(startPoint, 2) < getat(endPoint, 2) then
    -- the left side is higher than the right
    set top = getat(startPoint, 2)
    set bottom = getat(endPoint, 2)
    put leaningLeftCast into whichCast
```

```
  else
    set top = getat(endPoint, 2)
    set bottom = getat(startPoint, 2)
    put leaningRightCast into whichCast
  end if

  -- now check for top and bottom
  if top = bottom ther set bottom = bottom + 1
  if stringp(whichCast) then set whichCast = the membernum o
f member whichCast

  set the membernum of sprite aSprite = whichCast

  spritebox aSprite, left, top, right, bottom
end



on highlight spritelist, on, apagado
  repeat with n = 1 to count(spriteList)
    set the member of sprite getAt(spriteLIst,n) = member ap
agado
    set the member of sprite (the clickOn) = member on
  end repeat
  puppetSound "bip.aif"
  updateStage
end


global que_tri,lista,error,d_error,voltaje,condiciones,archi
vo,control,otro
global filas,columnas,maximo,minimo,g0,g1,g2,g3
global cuenta_matriz,matriz,matriz1,matriz2,matriz3,matriz4,
matriz5,matriz6,matriz7


on muevete donde,quien
  case the frame of
    35:
      set the cond of control=true
      if not integerp(integer(field "peso")) or integer(fiel
d "peso")<0 or integer(field "peso")>525 then
        alert "Por favor verifique que el valor del peso est
é correctamente asignado"
        exit
      else
        setat(condiciones,2,integer(the text of member "peso
"))
      end if
      if not floatp(float(field "normal")) then
```

```
      alert "Por favor verifique que el valor de la normal
esté correctamente asignado"
      exit
    end if

  25:
    revisa_matriz matriz
    set the base of control=true

  56:
    global result
    closefile (resultad)
    repeat with i=1 to 120
      puppetsprite i,false
    end repeat


  15:
    global control,que_tri
    case que_tri of
      1:
        set the error of control=true
        set g0=field "constante"
      2:
        set the d_error of control=true
        set g1=field "constante"
      3:
        set the voltaje of control=true
        set g2=field "constante"
      4:
        set the otro of control=true
        set g3=field "constante"
    end case

    corta_lineas "constante","textos"
    if not floatp(float(field "constante")) or (float(fiel
d "constante") <0 or float (field "constante")>10) then
      alert "Por favor verifique que el valor de la consta
nte esté correctamente asignado"
      exit
    end if

    case integer(field "funci") of
      3:
        set maximo=1
        set minimo=-1
      5:
        set maximo=2
        set minimo=-2
      7:
```

```
          set maximo=3
          set minimo=-3
        end case

    end case

    case donde of
      "inicio":
        repeat with i=21 to 46
          set the loc of sprite i to point(1000,200)
          puppetsprite i,false
        end repeat
        updatestage

        if que_tri= 1 or que_tri=2 or que_tri=3 then
          setat(num_funci,que_tri,integer(the text of member "
funci"))
        end if

        case que_tri of
          1:
            repeat with i=1 to count(lista)
              setat (error,i,getat(lista,i))
            end repeat
          2:
            repeat with i=1 to count(lista)
              setat (d_error,i,getat(lista,i))
            end repeat
          3:
            repeat with i=1 to count(lista)
              setat (voltaje,i,getat(lista,i))
            end repeat
          4:
            repeat with i=1 to count(lista)
              setat (otro,i,getat(lista,i))
            end repeat

        end case
        set lista=[]


      "error":
        set que_tri=quien
        case quien of
          1:
            if error<>[] then
              repeat with i=1 to count(error)
                setat (lista,i,getat(error,i))
              end repeat
            end if
```

```
        2:
          if d_error<>[] then
            repeat with i=1 to count(d_error)
              setat (lista,i,getat(d_error,i))
            end repeat
          end if
        3:
          if voltaje<>[] then
            repeat with i=1 to count(voltaje)
              setat (lista,i,getat(voltaje,i))
            end repeat
          end if

        4:
          if otro<>[] then
            repeat with i=1 to count(otro)
              setat (lista,i,getat(otro,i))
            end repeat
          end if

      end case

    "datos":
      set the member of sprite 20 to member "bot_up"
      updatestage

    "base":
      controla
  end case
  go frame donde
end

on controla
  global control
  if the voltaje of control=false then
    alert "Por favor, ingrese los datos del voltaje"
    abort
  else if the error of control=false then
    alert "Por favor, ingrese los datos del error"
    abort
  else if the d_error of control=false then
    alert "Por favor, ingrese los datos de la derivada del e
rror"
    abort
  else if the otro of control=false then
    alert "Por favor, ingrese los datos de la integral"
    abort

  end if
end
```

```
on controla2
  if the cond of control=false then
    alert "Por favor, ingrese las condiciones de operación"
    abort
  end if
  if the base of control=false then
    alert "Por favor, ingrese correctamente los rangos de la
 Base de Conocimiento"
    abort
  end if
end


on ayuda cual
  global vpanel,que_ayuda,que_tri
  set que_ayuda=cual
  set vpanel to window "ayuda"
  set x1=0 --the stageleft
  set y1=0 --the stagetop
  set x2=0 --the stageright
  set y2=0 --the stagetop
  set xi=160
  set yi=120
  set ancho=320
  set alto=240
  set the rect of vpanel to rect(x1+xi,y1+yi,x1+xi+ancho,y1+
yi+alto)
  set the filename of vpanel to "ayuda"
  set the titlevisible of vpanel to false
  case the frame of
    5: set que_ayuda=1
    15:
      case que_tri of
        1:set que_ayuda=2
        2:
          if the moviename contains "pd" or the moviename co
ntains "pid" then
            set que_ayuda=3
          else
            set que_ayuda=7
          end if
        3:set que_ayuda=4
        4:set que_ayuda=7
      end case
    25:set que_ayuda=5
    35:set que_ayuda=6
  end case
  open vpanel
end
```

```
on nuevo
  global num_funci,error,tiene_datos,d_error,lista,otro
  global voltaje,opciones,condiciones,control
  global control,g0,g1,g2,g3,otro,matriz
  global yy2,yy1
  set yy2=553
  set yy1=153

  openfile(archivo,the moviepath&"fuzzyin.txt",1)
  delete archivo
  if the moviename contains "pid" then
    set control=[#error:false,#d_error:false,#voltaje:false,
#cond:false,#base:false,#otro:false]
  else
    set control=[#error:false,#d_error:false,#voltaje:false,
#cond:false,#base:false,#otro:true]
  end if

  set lista=[]
  set the text of member "funci" to "7"
  if the moviename contains "pid" then
    set num_funci=[7,7,7,7]
  else
    set num_funci=[7,7,7,1]
  end if
  set tiene_datos=[0,0,0,0]
  set error=[]
  set otro=[]
  set d_error=[]
  set voltaje=[]
  set opciones=[1,1,1]
  set condiciones=[2,500]
  set g0=1
  set g1=1
  set g2=1
  set g3=1
  set matriz=[]
  set matriz1=[]
  set matriz2=[]
  set matriz3=[]
  set matriz4=[]
  set matriz5=[]
  set matriz6=[]
  set matriz7=[]
end

on matriz cuanto
  coloca_matriz matriz,cuenta_matriz
```

```
  revisa_matriz matriz
  set cuenta_matriz=cuenta_matriz+cuanto
  if cuenta_matriz<=0 then set cuenta_matriz=1
  if cuenta_matriz>getat(num_funci,4) then set cuenta_matriz
=getat(num_funci,4)
  set the member of sprite 10=member ("matriz"&string(cuenta
_matriz))
  case cuenta_matriz of
    1:set matriz=matriz1
    2:set matriz=matriz2
    3:set matriz=matriz3
    4:set matriz=matriz4
    5:set matriz=matriz5
    6:set matriz=matriz6
    7:set matriz=matriz7
  end case

  recoge_matriz matriz
end

on revisa_matriz matriz
  global cuenta_matriz
  repeat with i=1 to filas
    repeat with j=1 to columnas
      set temp=field ("valor"&i&j)
      corta_lineas ("valor"&i&j),"textos"
      coloca_matriz matriz,cuenta_matriz
      if integerP(integer(temp)) = FALSE or (integer(temp)>m
aximo or integer(temp)<minimo) then
        alert "Por favor verifique que los datos ingresados
sean valores numéricos y estén dentro de los rangos asignado
s para el voltaje"
        abort
      end if
    end repeat
  end repeat
end


on activa_matriz
  set filas=getat(num_funci,1)
  set columnas=getat(num_funci,2)
  repeat with i=1 to filas
    set tempo1=[]
    set tempo2=[]
    set tempo3=[]
    set tempo4=[]
    set tempo5=[]
    set tempo6=[]
    set tempo7=[]
```

```
      repeat with j=1 to columnas
        setat (tempo1,j,0)
        setat (tempo2,j,0)
        setat (tempo3,j,0)
        setat (tempo4,j,0)
        setat (tempo5,j,0)
        setat (tempo6,j,0)
        setat (tempo7,j,0)
      end repeat
      setat (matriz1,i,tempo1)
      setat (matriz2,i,tempo2)
      setat (matriz3,i,tempo3)
      setat (matriz4,i,tempo4)
      setat (matriz5,i,tempo5)
      setat (matriz6,i,tempo6)
      setat (matriz7,i,tempo7)
    end repeat
  end


  on recoge_matriz matriz
    set filas=getat(num_funci,1)
    set columnas=getat(num_funci,2)
    set spt=11
    repeat with i=1 to filas
      repeat with j=1 to columnas
        set the text of member ("valor"&string(i)&string(j)) o
f castlib "textos"=string(getat(getat(matriz,i),j))
        set the member of sprite spt = member ("valor"&string(
i)&string(j)) of castlib "textos"
        set spt=spt+1
      end repeat
      set spt=11+(i*7)
    end repeat
  end


  on coloca_matriz matriz,cual
    set filas=getat(num_funci,1)
    set columnas=getat(num_funci,2)
    set spt=11
    repeat with i=1 to filas
      set tempo=getat(matriz,i)
      repeat with j=1 to columnas
        setat (tempo,j,field ("valor"&string(i)&string(j)) )
      end repeat
    end repeat

  end
```



```
chivo),bajo)]
    else if cuenta=getat(num_funci,1) then
```

```
on abrir
  setfiltermask (archivo,"Archivos de Texto")
  set temp= displayopen(archivo)
  if voidp(temp) then exit
  openfile(archivo,temp,0)
  setposition(archivo,0)
  set elarchivo= readfile(archivo)
  set lineas=the number of lines of elarchivo-1
  closefile(archivo)
  global num_funci,error,tiene_datos,d_error,lista,otro
  global voltaje,opciones,condiciones,control
  global control,g0,g1,g2,g3
  set control=[#error:true,#d_error:true,#voltaje:true,#cond
:true,#base:true,#otro:true]

  set tiene_datos=[0,0,0]
  set error=[]
  set otro=[]
  set d_error=[]
  set voltaje=[]
  if the moviename contains "pid" then
    set cte=0
  else
    set cte=1
  end if

  set opciones=[integer(line lineas-9+cte of elarchivo),inte
ger(line lineas-8+cte of elarchivo),integer(line lineas-7+ct
e of elarchivo)]
  set condiciones=[integer(line lineas-5+cte of elarchivo),i
nteger(line lineas-6+cte of elarchivo)]
  set the text of member "normal"=string(float(line lineas-4
+cte of elarchivo))
  set g0=float(line lineas-3+cte of elarchivo)
  set g1=float(line lineas-2+cte of elarchivo)
  set g2=float(line lineas-1+cte of elarchivo)
  set g3=float(line lineas+cte of elarchivo)
  set linea_a=3
  set cuenta=1
  setat(num_funci,1,integer(line 2 of elarchivo))
  set alto=80+5
  set bajo=287-1
  repeat while word 1 of line linea_a of elarchivo<>"*b"
    if cuenta=1 then
      set temp=[point(funcion_y(word 1 of line linea_a of el
archivo),alto),point(funcion_y(word 2 of line linea_a of ela
rchivo),alto),point(funcion_y(word 3 of line linea_a of elar
chivo),bajo)]
    else if cuenta=getat(num_funci,1) then
```

```
      set temp=[point(funcion_y(word 1 of line linea_a of el
archivo),bajo),point(funcion_y(word 2 of line linea_a of ela
rchivo),alto),point(funcion_y(word 3 of line linea_a of elar
chivo),alto)]
    else
      set temp=[point(funcion_y(word 1 of line linea_a of el
archivo),bajo),point(funcion_y(word 2 of line linea_a of ela
rchivo),alto),point(funcion_y(word 3 of line linea_a of elar
chivo),bajo)]
    end if
    setat(error,cuenta,temp)
    set cuenta=cuenta+1
    set linea_a=linea_a+1
  end repeat

  if the moviename contains "pid" then
    set cuenta=1
    setat(num_funci,4,integer(word 1 of line linea_a+1 of el
archivo))
    set linea_a=linea_a+2
    repeat while word 1 of line linea_a of elarchivo<>"**"
      if cuenta=1 then
        set temp=[point(funcion_y(word 1 of line linea_a of
elarchivo),alto),point(funcion_y(word 2 of line linea_a of e
larchivo),alto),point(funcion_y(word 3 of line linea_a of el
archivo),bajo)]
      else if cuenta=getat(num_funci,4) then
        set temp=[point(funcion_y(word 1 of line linea_a of
elarchivo),bajo),point(funcion_y(word 2 of line linea_a of e
larchivo),alto),point(funcion_y(word 3 of line linea_a of el
archivo),alto)]
      else
        set temp=[point(funcion_y(word 1 of line linea_a of
elarchivo),bajo),point(funcion_y(word 2 of line linea_a of e
larchivo),alto),point(funcion_y(word 3 of line linea_a of el
archivo),bajo)]
      end if
      setat(otro,cuenta,temp)
      set cuenta=cuenta+1
      set linea_a=linea_a+1
    end repeat
  end if


  set cuenta=1
  setat(num_funci,2,integer(word 1 of line linea_a+1 of elar
chivo))
  set linea_a=linea_a+2
  repeat while word 1 of line linea_a of elarchivo<>"*c"
    if cuenta=1 then
```

```
      set temp=[point(funcion_y(word 1 of line linea_a of el
archivo),alto),point(funcion_y(word 2 of line linea_a of ela
rchivo),alto),point(funcion_y(word 3 of line linea_a of elar
chivo),bajo)]
    else if cuenta=getat(num_funci,2) then
      set temp=[point(funcion_y(word 1 of line linea_a of el
archivo),bajo),point(funcion_y(word 2 of line linea_a of ela
rchivo),alto),point(funcion_y(word 3 of line linea_a of elar
chivo),alto)]
    else
      set temp=[point(funcion_y(word 1 of line linea_a of el
archivo),bajo),point(funcion_y(word 2 of line linea_a of ela
rchivo),alto),point(funcion_y(word 3 of line linea_a of elar
chivo),bajo)]
    end if
    setat(d_error,cuenta,temp)
    set cuenta=cuenta+1
    set linea_a=linea_a+1
  end repeat


  set cuenta=1
  setat(num_funci,3,integer(word 1 of line linea_a+1 of elar
chivo))
  set linea_a=linea_a+2
  repeat while word 1 of line linea_a of elarchivo<>"*d"
    set temp=[point(funcion_y(word 1 of line linea_a of elar
chivo),bajo),point(funcion_y(word 2 of line linea_a of elarc
hivo),alto),point(funcion_y(word 3 of line linea_a of elarch
ivo),bajo)]
    setat(voltaje,cuenta,temp)
    set cuenta=cuenta+1
    set linea_a=linea_a+1
  end repeat

  case getat(num_funci,3) of
    3:
      set maximo=1
      set minimo=-1
    5:
      set maximo=2
      set minimo=-2
    7:
      set maximo=3
      set minimo=-3
  end case


  set linea_a=linea_a+1
  set filas=getat(num_funci,1)
```

```
  set columnas=getat(num_funci,2)
  repeat with k=1 to getat(num_funci,4)
    repeat with i=1 to filas
      case k of
        1: set temp1=[]
        2: set temp2=[]
        3: set temp3=[]
        4: set temp4=[]
        5: set temp5=[]
        6: set temp6=[]
        7: set temp7=[]
      end case
      repeat with j=1 to columnas
        case k of
          1: setat(temp1,j,word j of line linea_a of elarchi
vo)
          2: setat(temp2,j,word j of line linea_a of elarchi
vo)
          3: setat(temp3,j,word j of line linea_a of elarchi
vo)
          4: setat(temp4,j,word j of line linea_a of elarchi
vo)
          5: setat(temp5,j,word j of line linea_a of elarchi
vo)
          6: setat(temp6,j,word j of line linea_a of elarchi
vo)
          7: setat(temp7,j,word j of line linea_a of elarchi
vo)
        end case
      end repeat
      case k of
        1: setat(matriz1,i,temp1)
        2: setat(matriz2,i,temp2)
        3: setat(matriz3,i,temp3)
        4: setat(matriz4,i,temp4)
        5: setat(matriz5,i,temp5)
        6: setat(matriz6,i,temp6)
        7: setat(matriz7,i,temp7)
      end case
      set linea_a=linea_a+1
    end repeat
  end repeat
end


on guardar
  controla
  controla2
  set temp=displaysave(archivo,"Nombre del Archivo","datos.t
```

```
xt")
  if not voidp(temp) then
    openfile(archivo,temp,1)
    delete (archivo)
    guarda temp
    guarda (the moviepath&"fuzzyin.txt")
  end if
end



on guarda temp
  closefile(archivo)
  createfile (archivo,temp)
  openfile(archivo,temp,0)
  setposition(archivo,0)
  set salto=numtochar(13)&numtochar(10)

  writestring(archivo,"*a"&salto)
  writestring(archivo,string(getat(num_funci,1))&salto)
  repeat with i=1 to getat(num_funci,1)
    setposition(archivo,getlength(archivo))
    set centro=funcion_x(the loch of (getat(getat(error,i),2
)))
    set izquierdo=funcion_x(the loch of getat(getat(error,i)
,1))
    set derecho=funcion_x(the loch of getat(getat(error,i),3
))
    writestring(archivo,izquierdo&TAB&centro&TAB&derecho&sal
to)
  end repeat


  writestring(archivo,"*b"&salto)
  writestring(archivo,string(getat(num_funci,2))&salto)
  repeat with i=1 to getat(num_funci,2)
    setposition(archivo,getlength(archivo))
    set centro=funcion_x(the loch of (getat(getat(d_error,i)
,2)))
    set izquierdo=funcion_x(the loch of getat(getat(d_error,
i),1))
    set derecho=funcion_x(the loch of getat(getat(d_error,i)
,3))
    writestring(archivo,izquierdo&TAB&centro&TAB&derecho&sal
to)
  end repeat

  if the moviename contains "pid" then
    writestring(archivo,"**"&salto)
    writestring(archivo,string(getat(num_funci,4))&salto)
```

```
    repeat with i=1 to getat(num_funci,4)
      setposition(archivo,getlength(archivo))
      set centro=funcion_x(the loch of (getat(getat(otro,i),
2)))
      set izquierdo=funcion_x(the loch of getat(getat(otro,i
),1))
      set derecho=funcion_x(the loch of getat(getat(otro,i),
3))
      writestring(archivo,izquierdo&TAB&centro&TAB&derecho&s
alto)
    end repeat
  end if


  writestring(archivo,"*c"&salto)
  writestring(archivo,string(getat(num_funci,3))&salto)
  repeat with i=1 to getat(num_funci,3)
    setposition(archivo,getlength(archivo))
    set centro=funcion_x(the loch of (getat(getat(voltaje,i)
,2)))
    set izquierdo=funcion_x(the loch of getat(getat(voltaje,
i),1))
    set derecho=funcion_x(the loch of getat(getat(voltaje,i)
,3))
    writestring(archivo,izquierdo&TAB&centro&TAB&derecho&sal
to)
  end repeat

  set filas=getat(num_funci,1)
  set columnas=getat(num_funci,2)
  writestring(archivo,"*d"&salto)
  repeat with k=1 to getat(num_funci,4)
    case k of
      1:set matriz=matriz1
      2:set matriz=matriz2
      3:set matriz=matriz3
      4:set matriz=matriz4
      5:set matriz=matriz5
      6:set matriz=matriz6
      7:set matriz=matriz7
    end case
    repeat with i=1 to filas
      set linea=""
      repeat with j=1 to columnas
        set linea= linea&getat(getat(matriz,i),j)
        if j<>columnas then
          set linea=linea&TAB
        else
          set linea=linea&salto
        end if
```

```
      end repeat
      writestring(archivo,linea)
    end repeat
  end repeat


  --  setposition(archivo,getlength(archivo))
  writestring(archivo,"*e"&salto)
  writestring(archivo,string(getat(opciones,1))&salto)
  writestring(archivo,string(getat(opciones,2))&salto)
  writestring(archivo,string(getat(opciones,3))&salto)
  writestring(archivo,string(getat(condiciones,2))&salto)
  writestring(archivo,string(getat(condiciones,1))&salto)
  writestring(archivo,string(field "normal")&salto)
  writestring(archivo,string(g0)&salto)
  writestring(archivo,string(g1)&salto)
  writestring(archivo,string(g2)&salto)
  writestring(archivo,string(g3)&salto)
  closefile(archivo)
end

on simula quien
  global ejecuta
  controla
  controla2
  set ejecuta=quien
  closefile(archivo)
  openfile(archivo,the moviepath&"fuzzyin.txt",0)
  delete (archivo)
  closefile(archivo)
  guarda (the moviepath&"fuzzyin.txt")


  go frame "chao"
end

on corta_lineas cual,libreria
  set tempo=the text of member cual of castlib libreria
  if the number of lines of tempo> 1 then ¬
set the text of member cual of castlib libreria= ¬
line 1 of tempo
end


on iniciar
  global contador
  set contador=0
  set the timeoutLength to 3
  go frame "ini_sim"
end
```

```
******** SIMULACION **********

on exitFrame
  -- la listado toma velocidades cada tiempo determinado
  -- por timeoutlength

  global resultad,pasada
  set pasada=1
  set resultad=new(xtra "fileio")
  openfile (resultad,the moviepath&"fuzzyout.txt",1)

  set the floatPrecision to 5


  global resultad,temp
  set temp=readfile(resultad)
  global listado
  set listado=[]
  set contador=1
  set cont=1
  repeat with i=1 to the number of lines of temp/50
    set cuadro=[float(word 3 of line contador of temp),float
(word 2 of line contador of temp)]
    setat(listado,cont,cuadro)
    set contador=contador+50
    set cont=cont+1
  end repeat
  cursor -1
   closefile (resultad)
end



on exitFrame
  global pendiente,y2,y1,x2,x1,contador,tempo
  set y2=the top of sprite 2+float(the height of sprite 3/2)
  set y1=the bottom of sprite 2-float(the height of sprite 3
/2)
  set tempo=[]
  set vel=[]

  -- recoger datos de espacio
  repeat with i=1 to count(listado)
    add (tempo,getat(getat(listado,i),2))
    add (vel,getat(getat(listado,i),1))
  end repeat
  set x2=max(tempo)
```

```
  set x1=min(tempo)
  set vel_max=max(vel)
  set the text of member "ymax"=string(x2)
  set the text of member "tmax"=string(float(0.05*count(list
ado)))
  set the text of member "vmax"=string(vel_max)
  set pendiente=float((y2-y1)/(x2-x1))


  puppetsprite 9,true

  -- Datos para la gráfica

  global unix_x_seg,uniy_x_seg,uniy2_x_seg
  set unix_x_seg=integer(the width of member "eje_tiempo" /c
ount(listado))*1.3
  set uniy_x_seg=integer(the height of member "eje_vel" /vel
_max)
  set uniy2_x_seg=integer(the height of member "eje_vel" /x2
)

end


on exitFrame
  go to the frame
end

on timeout
  global listado,pendiente,y2,x2,contador
  if contador<count(listado) then
    set tempo=y2+(pendiente*(getat(getat(listado,contador+1)
,2)-x2))
    set the locv of sprite 3 to tempo

    --   grafico

    global unix_x_seg,uniy_x_seg ,uniy2_x_seg,tempo
    set the loch of sprite 9 to the left of sprite 118+(unix
_x_seg*contador)
    set the locv of sprite 9 to the bottom of sprite 117-((g
etat(getat(listado,contador+1),1))*uniy_x_seg)

    set the loch of sprite 8 to the left of sprite 113+(unix
_x_seg*contador)
    set the locv of sprite 8 to the bottom of sprite 111-(ge
tat(tempo,contador+1)*uniy2_x_seg)


    updatestage
```

```
    set contador=contador+1
  else
    go to the frame +1
    exit
  end if
end

****** BASE DE CONOCIMIENTO *******
on exitFrame
  global num_funci,opciones
  global filas,columnas,num_funci,matriz,cuenta_matriz
  global matriz1,matriz2,matriz3,matriz4,matriz5,matriz6,mat
riz7
  set cuenta_matriz=1
  set the member of sprite 10= member "matriz1"

  if matriz1=[] then activa_matriz
  set matriz=matriz1

  recoge_matriz matriz

  set spt=67
  repeat with i=1 to filas
    set the member of sprite (spt-1+i) to member (i+9-(filas
-1)/2) of castlib "textos"
  end repeat
  set locy=(the locv of sprite(spt+filas-1))+30
  set spt=60
  repeat with i=1 to columnas
    set the member of sprite (spt-1+i) to member (i+9-(colum
nas-1)/2) of castlib "textos"
  end repeat
  set locx= (the loch of sprite (spt+columnas-1))+60
  set the member of sprite 74 = member "recuadro"
  set the rect of sprite 74 to rect( the left of sprite 74,
the top of sprite 74,locx,locy)
  coloca_boton getat(opciones,1),[76,77]
  coloca_boton getat(opciones,2),[78,79]
  coloca_boton getat(opciones,3),[80,81]

  updatestage
end

on coloca_boton opcion,lista
  if opcion<>0 then
    set the member of sprite getat(lista,opcion)= member "bo
t_down"
    updatestage
  end if
end
```

```
*****   INICIAR FUNCIONES DE TRIANGULOS *******

on exitFrame
  global que_tri,yy2,yy1,xx2,xx1,error,d_error,voltaje,g0,g1
,g2,g3,otro
  set the text of member "funci"=string(getat(num_funci,que_
tri))
  if the text of member "funci"="1" then set the text of mem
ber "funci"="7"
  set yy2=553
  set yy1=153
  set xx2=100
  set xx1=-100


  case que_tri of
    1:
      set lista=error
      grafica_lista 0
      set temp="terror"
      set the text of member "constante"=string(g0)
    2:
      set lista=d_error
      grafica_lista 0
      set temp="tdelta"
      set the text of member "constante"=string(g1)
    3:
      set lista=voltaje
      grafica_lista 1
      set temp="tvolt"
      set the text of member "constante"=string(g2)
    4:
      set lista=otro
      grafica_lista 0
      set temp="totro"
      set the text of member "constante"=string(g3)
  end case

  set the member of sprite 2 to member temp of castlib "text
os"
end


******* OBJETO DE LOS BOTONES *********

global idioma,actores

property libreria,hay_idioma,idiom
```

```
property mb_soltar,exe_soltar
property hay_rollover,snd_rollover,canal_snd_roll,exe_rollov
er
property hay_aplastado,snd_aplastado,canal_snd_apls,exe_apla
stado
property snd_saliente,canal_snd_slnt
property spt,cursor1,cursor2,exe_salir



on getpropertydescriptionlist
  set descr=[:]
  addprop descr,#hay_idioma,[#default:true,#format:#boolean,
#comment:"Este miembro está sujeto al cambio de idioma:"]
  addprop descr,#hay_rollover,[#default:true,#format:#boolea
n,#comment:"Existe Rollver:"]
  addprop descr,#hay_aplastado,[#default:true,#format:#boole
an,#comment:"Existe aplastado:"]
  addprop descr,#mb_soltar,[#default:"",#format:#string,#com
ment:"Cast Member:"]
  addprop descr,#libreria,[#default:"internal",#format:#lib,
#comment:"Librería del miembro:"]
  addprop descr,#snd_rollover,[#default:"audio rollover",#fo
rmat:#string,#comment:"Audio Rollver:"]
  addprop descr,#canal_snd_roll,[#default:1,#format:#integer
,#range:[#min:1,#max:2],#comment:"Canal de Audio para Rollve
r:"]
  addprop descr,#snd_aplastado,[#default:"audio aplastado",#
format:#string,#comment:"Audio aplastadoo:"]
  addprop descr,#canal_snd_apls,[#default:1,#format:#integer
,#range:[#min:1,#max:2],#comment:"Canal de Audio para aplast
ado:"]
  addprop descr,#snd_saliente,[#default:"audio salir",#forma
t:#string,#comment:"Audio al Salir:"]
  addprop descr,#canal_snd_slnt,[#default:1,#format:#integer
,#range:[#min:1,#max:2],#comment:"Canal de Audio para Salir:
"]
  addprop descr,#exe_aplastado,[#default:"",#format:#string,
#comment:"Ejecución al Aplastar:"]
  addprop descr,#exe_rollover,[#default:"",#format:#string,#
comment:"Ejecución al Rollver:"]
  addprop descr,#exe_soltar,[#default:"",#format:#string,#co
mment:"Ejecución al Soltar:"]
  addprop descr,#exe_salir,[#default:"",#format:#string,#com
ment:"Ejecución al Salir:"]
  addprop descr,#cursor1,[#default:"",#format:#string,#comme
nt:"Cursor 1:"]
  addprop descr,#cursor2,[#default:"",#format:#string,#comme
nt:"Cursor 2:"]
```

```
  return descr
end



on beginsprite me
  if hay_idioma=false then
    set idiom=""
  else
    set idiom=idioma
  end if


  if actores=void then set actores=[#nada:0]
  addprop actores,mb_soltar,[#yo:me,#canal:the spritenum of
me]
  puppetsprite spt,true
  set spt=the spritenum of me
  set the member of sprite spt=member (mb_soltar&"n") of cas
tlib (libreria&idiom)
  updatestage
end

on endsprite
  puppetsprite spt,false
end

on mousedown me
  if not the doubleclick then
    if hay_aplastado then
      set the member of sprite spt=member (mb_soltar&"d") of
 castlib (libreria&idiom)
      updatestage
    end if
    if snd_aplastado<>"" then
      puppetsound canal_snd_apls,snd_aplastado
    end if
    do exe_aplastado
  end if

end

on mouseenter me
  if hay_rollover then
    set the member of sprite spt=member (mb_soltar&"r") of c
astlib (libreria&idiom)
  end if
  do exe_rollover
  if cursor1<>"" then
```

```
    if cursor2<>"" then
      cursor  [member cursor1,member cursor2]
    else
      cursor (member cursor1 of castlib "internal")
    end if
  end if
  if snd_rollover<>"" then
    puppetsound canal_snd_roll,snd_rollover
  end if
end

on mouseleave me
  if snd_saliente<>"" then
    puppetsound canal_snd_slnt,snd_saliente
  end if
  if exe_salir<>"" then do exe_salir
  set the member of sprite spt=member (mb_soltar&"n") of cas
tlib (libreria&idiom)
  cursor -1
end

on mouseup me
  if not the doubleclick then
    if rollover(spt) and hay_rollover then
      set tempo="r"
    else
      set tempo="n"
    end if
    set the member of sprite spt=member (mb_soltar&tempo) of
 castlib (libreria&idiom)
  end if
  do exe_soltar
end
```