

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS Fundada en 1551

FACULTAD DE INGENIERÍA ELECTRÓNICA

E.A.P. DE INGENIERÍA ELECTRÓNICA

Control neuro-difuso aplicado a una Grúa Torre

TESIS para optar el Título Profesional de: INGENIERO ELECTRÓNICO

AUTOR

JOSÉ CARLOS CHAHUARA QUISPE

ASESOR: GUILLERMO TEJADA MUÑOZ

LIMA – PERÚ 2005

De la reflexión más calmada llegará la acción más efectiva.

James Levin

Mediante tres métodos podemos obtener la sabiduría: Primero, por reflexión, que es el más noble; Segundo, por imitación, que es el más sencillo; y Tercero por la experiencia, que es el más amargo.

Confucio

AGRADECIMIENTOS

Quisiera, en primer lugar, testimoniar mi más profundo reconocimiento al profesor Guillermo Tejada, quien ha dirigido el desarrollo de esta tesis y ha guiado mis primeros pasos en la investigación. Su gran cultura científica, su disponibilidad y su simplicidad son algunos de los tantos elementos que han favorecido el desarrollo de esta tesis.

Agradezco también al Instituto de Investigación de la FIE, los meses pasados en él fueron meses de diversas experiencias enriquecedoras; el dinamismo de sus miembros y el excelente ambiente de trabajo hacen de él un lugar muy simpático y motivante.

Gracias finalmente a mis padres por haberme apoyado a lo largo de esta aventura, su paciencia y apoyo constante me han sido extremadamente útiles.

INDICE

Lista de Figuras

Lista de Tablas

Resumen

I. Introducción	1
1.1 Objetivos	3
1.2 Justificación	3
1.3 Metodología	4
1.4 Motivación	4
1.5 Antecedentes	5
1.6 Descripción del Sistema	5
1.7 Estructura de la Presentación	7
 II. Sistemas Neuro-difusos	 8
2.1 Introducción a las Redes Neuronales	8
2.1.1 La Neurona Biológica	9
2.1.2 Componentes de una Red Neuronal Artificial	11
2.1.3 Tipos de Redes Neuronales	12
2.1.4 Arquitectura de Redes Neuronales Artificiales	13
2.1.5 El Proceso de Aprendizaje	15
2.1.5.1 Tipos de Aprendizaje	15
2.1.5.2 Algoritmos de Aprendizaje	16
2.2 Introducción a los Sistemas Difusos	18
2.2.1 Conjuntos Difusos	22
2.2.2 Lógica Difusa	24
2.2.3 Sistemas Lógicos Difusos	27
2.2.3.1 Etapas de un Sistema Lógico Difuso	28
2.2.4 Sistemas Difusos Aplicados al Control de Procesos	31
2.3 Introducción a los Sistemas Neuro-difusos	32
2.3.1 Aprendizaje en un Sistema Neuro-difusos	34
2.3.2 FuzzyTech	36

2.3.2.1	Agrupamiento de Datos (<i>Clustering Module</i>)	36
2.3.2.2	Creación de Sistemas Lógicos Difusos (<i>Fuzzy Tools</i>)	39
2.3.2.3	Entrenamiento de un Sistema Difuso (<i>Neurofuzzy Module</i>)	42
2.3.2.4	Implementación de Sistemas Neuro-difusos	43
III. Bases Teóricas y Experimentales para el desarrollo del simulador de Grúa Torre		44
3.1	Ecuaciones que describen el funcionamiento de una Grúa Torre	44
3.1.1	Funcionamiento de una Grúa Torre	45
3.1.2	Coordenadas y Variables Físicas	46
3.1.3	Ecuaciones de Movimiento	50
3.2	Diseño del Simulador 3D de Grúa Torre	55
3.2.1	Implementación del modelo en Simulink	56
3.2.2	Animación	57
3.2.2.1	Elementos Geométricos	58
3.2.3	Interfaz Gráfica de Usuario	64
3.3	Validación del Simulador 3D	64
3.3.1	Algoritmos de Control desarrollados por <i>Hanafy M. Omar</i>	67
3.3.1.1	Control de Realimentación de Ganancias	67
3.3.1.2	Control Dual de Posición y Anti-oscilación	69
3.3.2	Algoritmos de Control desarrollados por <i>Amjed Al-Mousa</i>	71
IV. Extracción de Datos Experimentales para el Sistema Neuro difuso		74
4.1	Recolección de datos experimentales	74
4.2	Selección de datos experimentales más representativos	75
V. Diseño del Sistema de Automatización Neuro-difuso		76
5.1	Descripción del Problema	76
5.2	Diseño del Sistema Neuro-difuso	77
5.2.1	Derivación de Reglas a partir del Conocimiento a priori	78
5.2.1.1	Diseño de BRr I	79
5.2.1.2	Diseño de BRr II	81

5.2.1.3	Diseño de BRr III	83
5.2.2	Diseño de la Fusificación	84
5.2.3	Diseño de la Inferencia y la Defusificación	86
5.2.4	Entrenamiento del Sistema	87
5.2.5	Optimización	88
VI.	Resultados	89
6.1	Presentación de Resultados Comparativos	89
6.1.1	Neuro-difuso vs Método de Realimentación de Estados de Hanafy	89
6.1.2	Neuro-difuso vs Método de Control dual de Hanafy	92
6.1.3	Neuro-difuso vs Métodos de Control de Al-Mousa	95
6.2	Otros Resultados	98
VII.	Conclusiones	103
Bibliografía		105

LISTA DE FIGURAS

Figura 1.1 – Grúas Torre en una Edificación	1
Figura 1.2 - Sistema de Control de la Grúa Torre	5
Figura 1.3 – Sistema de Automatización Neuro-difuso	6
Figura 1.4 - Bloques de reglas incluidos en los controladores de movimientos radial y rotacional	6
Figura 2.1 - Sistema Nervioso	10
Figura 2.2 - Neurona Biológica	10
Figura 2.3 - Neurona Artificial	11
Figura 2.4 - Red Neuronal Monocapa	14
Figura 2.5 - Red Neuronal Multicapa	14
Figura 2.6 - Red Neuronal Realimentada	15
Figura 2.7 - Red Neuronal Multicapa	17
Figura 2.8 - Conceptos de Lógica difusa	21
Figura 2.9 - Funciones de Pertenencia	23
Figura 2.10 - Sistema Lógico Difuso	28
Figura 2.11 - Defusificación por Centro de Área	31
Figura 2.12 – Diagrama de bloques de un sistema Neuro-difuso	34
Figura 2.13 – Estructura de un sistema Neuro-difuso	34
Figura 2.14 – Clusterización o Agrupamiento de Datos	37
Figura 2.15 - Función de Pertenencia para expresar la similitud entre datos	39
Figura 3.1 - Movimientos de una Grúa Torre	45
Figura 3.2 - Estructura simplificada de una Grúa Torre	45
Figura 3.3 - Ubicación de los ejes coordenados	46
Figura 3.4 - Parámetros de entrada del sistema	47
Figura 3.5 - Ángulos de oscilación	47
Figura 3.6 - Vista lateral del modelo de Grúa Torre	48
Figura 3.7 - Vista frontal del modelo de Grúa Torre	48
Figura 3.8 - Vista superior del modelo de Grúa Torre	49
Figura 3.9 - Modelo físico del sistema de elevación de una Grúa Torre	53

Figura 3.10 - Modelo de compensación para el control por aceleración	55
Figura 3.11 - Implementación del modelo de Grúa Torre en Simulink	56
Figura 3.12 - Vistas de la torre de la grúa	58
Figura 3.13 - Vistas de la pluma y el contrapeso de la grúa	59
Figura 3.14 - Vistas del carro y de la cuerda de la grúa	59
Figura 3.15 - Vistas de la carga de la Grúa	60
Figura 3.16 - Vistas del señalador de destino	60
Figura 3.17 - Animación del Simulador 3D de Grúa Torre	61
Figura 3.18 - Ilustración del dibujo de formas geométricas en MATLAB	61
Figura 3.19 - Interfaz Gráfica de Usuario del Simulador 3D	64
Figura 3.20 - Controladores de Hanafy	65
Figura 3.21 - Controladores de Al-Mousa	66
Figura 3.22 - Control de Realimentación Parcial de Hanafy	67
Figura 3.23 - Comparación de Resultados usando el controlador de Realimentación parcial de Hanafy	68
Figura 3.24 - Controlador Dual de Hanafy	69
Figura 3.25 - Comparación de Resultados usando el Controlador Dual de Hanafy	70
Figura 3.26 - Controlador de Retraso en el Tiempo de Al-Mousa	71
Figura 3.27 - Controlador Difuso de Al-Mousa	72
Figura 3.28 - Comparación de resultados utilizando los algoritmos de Al-Mousa	72
 Figura 4.1 – Recolección de datos experimentales	 74
Figura 4.2 – Cabecera del archivo <i>entrenadat.xls</i>	75
 Figura 5.1 – Esquema del Sistema de Grúa Automatizado	 76
Figura 5.2 - Sistema carro péndulo con variables lingüísticas	77
Figura 5.3 – Bloques de Reglas de los controladores Radial y Rotacional	78
Figura 5.4 - Derivación de reglas utilizando la posición y el ángulo	80
Figura 5.5 - Derivación de las reglas a partir de las velocidades	82
Figura 5.6 - Términos de la variable lingüística d_r (distancia radial)	84
Figura 5.7 - Términos de la variable lingüística ϕ (ángulo de oscilación radial)	84
Figura 5.8 - Términos de la variable lingüística d'_r (velocidad radial)	84
Figura 5.9 - Términos de la variable lingüística ϕ' (velocidad angular radial)	85

Figura 5.10 - Términos de la variable lingüística d_γ (distancia rotacional)	85
Figura 5.11 - Términos de la variable lingüística d'_γ (velocidad rotacional)	85
Figura 5.12 - Términos de la variable lingüística θ (ángulo rotacional)	85
Figura 5.13 - Términos de la variable lingüística θ' (velocidad angular rotacional)	86
Figura 5.14 - Términos de la variable lingüística r'' (aceleración radial)	86
Figura 5.15 - Términos de la variable lingüística γ'' (aceleración rotacional)	87
Figura 5.16 – Esquema del Sistema de Automatización de la Grúa Torre	87
Figura 5.17 - Algunas reglas y sus pesos generadas mediante el proceso de entrenamiento	88
Figura 5.18 – Proceso de entrenamiento y optimización	88
 Figura 6.1 - Movimiento Radial	 89
Figura 6.2 - Movimiento Rotacional	90
Figura 6.3 - Movimiento Compuesto	91
Figura 6.4 - Movimiento Radial	93
Figura 6.5 - Movimiento Rotacional	93
Figura 6.6 - Movimiento Compuesto	94
Figura 6.7 - Movimiento Radial	95
Figura 6.8 - Movimiento Rotacional	96
Figura 6.9 - Movimiento Compuesto	97
Figura 6.10 - Movimiento Radial	99
Figura 6.11 - Movimiento Rotacional	99
Figura 6.12 - Movimiento Compuesto	100
Figura 6.13 - Variaciones en L	101
Figura 6.14 - Variaciones en m	102

LISTA DE TABLAS

Tabla 2.1 Computadora Von Neumann vs Sistema Neuronal	9
Tabla 2.2 Tipos de redes Neuronales	13
Tabla 2.3 Sintaxis de la Lógica Proposicional y de la Lógica Difusa	25
Tabla 2.4 Semántica de la Lógica Proposicional y de la Lógica Difusa	26
Tabla 2.5 Lógica Difusa y Redes Neuronales	32
 Tabla 4.1 Condiciones iniciales para la recolección de datos de entrenamiento	 75
 Tabla 5.1 Diseño de BRr I	 81
Tabla 5.2 Diseño de BRr II	83
Tabla 5.3 Diseño de BRr III	83
 Tabla 6.1 Movimiento Radial	 90
Tabla 6.2 Movimiento Rotacional	91
Tabla 6.3 Movimiento Compuesto	92
Tabla 6.4 Movimiento Radial	93
Tabla 6.5 Movimiento Rotacional	94
Tabla 6.6 Movimiento Compuesto	95
Tabla 6.7 Movimiento Radial	96
Tabla 6.8 Movimiento Rotacional	97
Tabla 6.9 Movimiento Compuesto	98
Tabla 6.10 Características del Controlador Neuro-difuso	100

RESUMEN

En el presente trabajo de tesis se diseña un sistema Neuro-difuso para automatizar el proceso de transporte de cargas que realiza una Grúa Torre.

Debido a que no se ha podido disponer de una Grúa Torre real ni de un prototipo a escala, se ha optado por diseñar un simulador 3D de Grúa Torre que puede ser operado tanto manualmente, a través de un panel de botones en la pantalla, como automáticamente, cuando se inserta algún algoritmo de control. Durante el estudio teórico se ha tenido acceso a trabajos de investigación de la universidad *Virginia Tech* (*Virginia Polytechnic Institute and State University – USA*) referidos al estudio de la automatización de grúas, pero se ignoran los detalles del simulador utilizado en ellos; por este motivo, se ha sometido a prueba el simulador de la presente tesis aplicando los algoritmos de control publicados por los investigadores de dicha universidad. Al haber obtenido respuestas similares a las publicadas, se verificó que el simulador propuesto es equivalente al utilizado en aquellos trabajos. Por tanto, se tuvo la certeza de haber diseñado un simulador confiable que permitiese, posteriormente, probar el desempeño del sistema de automatización Neuro-difuso.

La operación manual del simulador, emulando la labor de un obrero de construcción, ha permitido extraer datos experimentales para obtener una base de datos del funcionamiento de la Grúa Torre. Del total de valores experimentales, se seleccionaron los más representativos a través de un proceso de agrupamiento de datos o clusterización (*clustering*). Esta etapa, así como las demás etapas del diseño del algoritmo Neuro-difuso, se realizó utilizando el entorno de desarrollo fuzzyTECH del Instituto de Investigación de la FIE-UNMSM. Posteriormente, el sistema Neuro-difuso fue probado en el simulador 3D, en su modo de operación automático, y los resultados fueron comparados con los obtenidos al aplicar los algoritmos de las tesis de Post-Grado de *Virginia Tech* en el simulador.

Finalmente, se logró demostrar que un Sistema de Automatización Neuro-difuso para una Grúa Torre es tan eficiente como los desarrollados a través de métodos clásicos, con la ventaja de ser más fácil de diseñar ya que no necesita de modelos matemáticos, sino de datos de su operación manual.

I. INTRODUCCIÓN

El presente trabajo de tesis consiste en la automatización del funcionamiento de una Grúa utilizando técnicas Neuro-difusas. La Grúa es como la que se muestra en la figura 1.1, y es conocida como Grúa Torre o Grúa Pluma; su forma de 'T' característica puede ser apreciada en muchas de las construcciones que se realizan actualmente en la ciudad. La Grúa Torre es una máquina de gran importancia en la industria de la construcción contemporánea, ya que es la encargada del transporte de diversos materiales como acero y cemento dentro del área de trabajo; debido a su uso extendido y al gran interés en mejorar su desempeño, se han realizado un gran número de estudios acerca de sistemas que permitan el control automático de estas máquinas.



La tecnología Neuro-difusa forma parte de una nueva corriente en cuanto a técnicas de computación agrupadas dentro del concepto de "*Computación Flexible*" (*Soft Computing*) la cual es actualmente, sujeto de innumerables investigaciones gracias a sus capacidades de procesamiento a partir de datos imprecisos y de aprendizajes semejantes a las del cerebro biológico.

A diferencia de los métodos tradicionales de Control, el método Neuro-difuso no necesita de modelos matemáticos; esta es una característica ventajosa cuando se considera la automatización de procesos difíciles o imposibles de modelar como en el caso de una Grúa Torre. Los ingenieros encargados de automatizar procesos utilizando métodos difusos o neuro-difusos sólo necesitan conocer la operatividad (el

conocimiento del técnico operario), algunas ideas intuitivas basadas en criterios de ingeniería y la herramienta software que utilizarán para realizar el programa de automatización.

Al iniciar este trabajo, se tuvo la intención de montar una serie de sensores de gran precisión sobre una grúa real, para registrar las variables físicas involucradas en el proceso de control manual que realiza un operario mientras trabaja. Los datos recolectados representarían la operatividad de la grúa, y serían utilizados para entrenar el sistema de automatización neuro-difuso. Sin embargo, las empresas de maquinaria pesada de nuestro medio son esencialmente empresas comercializadoras, que alquilan o venden grúas a las empresas constructoras, y no tienen interés en el desarrollo de nuevas tecnologías. De esta manera, no fue posible contar con una grúa real. Por otro lado, se consideró también implementar un modelo a escala, pero el elevado costo de la estructura y los sensores de alta precisión impidieron su realización.

Debido a estos inconvenientes, se ha optado por diseñar un simulador 3D que permita aprender y analizar el comportamiento de estas máquinas. A partir de los datos experimentales obtenidos durante la operación manual del simulador, y de ideas intuitivas acerca del comportamiento de los sistemas masa-péndulo, se han extraído reglas para alimentar el sistema de decisión Neuro-difuso. En la actualidad, el empleo de simuladores es un procedimiento completamente válido, y es utilizado por la mayoría de los investigadores a nivel mundial.

Los resultados han sido comparados con aquellos obtenidos utilizando algoritmos de control propuestos en trabajos de investigación de nivel de maestría y doctorado desarrollados en la universidad *Virginia Tech (Virginia PolyTech Institute and State University - USA)* en los años 2000 y 2003.

1.1 Objetivos

Los objetivos de esta tesis son:

- 1.- Aplicar la tecnología de punta Neuro-difusa para automatizar procesos no-lineales complejos, y así evitar las complicaciones matemáticas inherentes a la Teoría Clásica de Control.
- 2.- Diseñar y validar un simulador de Grúa Torre tridimensional que sirva como herramienta experimental para el desarrollo de la tesis.
- 3.- Comparar el sistema de automatización Neuro-difuso con algoritmos de control clásicos desarrollados en otros trabajos de investigación.

1.2 Justificación

El enorme desarrollo de las Ciencias de la Computación ha permitido simplificar muchas de las tareas que el hombre suele realizar, ya sea en el diseño de circuitos electrónicos, cálculos matemáticos y almacenamiento de información entre otros. Siguiendo esta tendencia, actualmente se hace posible realizar el proceso de creación de algoritmos de control de una forma mucho más natural e intuitiva a través del uso de métodos relacionados con los conceptos de Inteligencia Artificial como son la Lógica Difusa y las Redes Neuronales. Esta nueva metodología permite evitar las complicaciones matemáticas propias de un desarrollo clásico de controladores, especialmente cuando se trata con procesos no-lineales de difícil modelamiento matemático. En el caso de una Grúa Torre, el control intuitivo realizado por un humano puede parecer obvio, pero es difícil de interpretar con métodos matemáticos convencionales, por ello, el presente estudio pretende demostrar que una solución alternativa basada en técnicas computacionales modernas es la forma más sencilla de automatizar eficientemente procesos demasiado complejos.

1.3 Metodología

El primer paso para el desarrollo de esta tesis ha sido recopilar información y realizar una revisión teórica de los conceptos que fundamentan la tecnología Neuro-difusa, los sistemas físicos masa-péndulo y el desarrollo de trabajos orientados a la automatización de grúas. Luego, se establecieron las bases matemáticas y físicas para el diseño de un simulador 3D de Grúa Torre en MATLAB®, y su comportamiento fue sometido a prueba aplicando sobre él algoritmos de control de Grúas Torre publicados en trabajos de investigación de la universidad *Virginia Tech*. De esta forma, se tuvo la certeza de contar con un simulador confiable y un marco de comparación único para el sistema de automatización Neuro-difuso y los algoritmos de control de los trabajos referenciados. Posteriormente, se siguió el procedimiento de diseño de sistemas Neuro-difusos, desde la inserción de conocimiento a priori (obvio) hasta su entrenamiento con datos experimentales obtenidos a través del manejo manual del simulador 3D. Finalmente, los desempeños del algoritmo Neuro-difuso y de los algoritmos desarrollados por investigadores del *Virginia Tech* fueron comparados luego de aplicarlos sobre el simulador 3D previamente validado. Para realizar el diseño del sistema Neuro-difuso se ha utilizado el software de desarrollo fuzzyTECH® del Instituto de Investigación FIE-UNMSM en una computadora Compaq 1200 con procesador AMD-K6 y 28,0 Mb de memoria RAM.

1.4 Motivación

La motivación de este trabajo de investigación es llevar a la práctica el método Neuro-difuso para el diseño de sistemas de automatización como aplicación de una de las nuevas tendencias, en cuanto a técnicas de computación, que genera gran interés entre investigadores de todo el mundo, gracias a su capacidad de combinar las características de la Lógica Difusa y las Redes Neuronales. Se ha considerado automatizar el funcionamiento de una Grúa Torre, debido a lo difícil que es hacerlo utilizando métodos tradicionales y a su amplio uso en la industria de la construcción.

1.5 Antecedentes

Existen diversos estudios acerca de métodos para automatizar el funcionamiento de grúas, los cuales han sido desarrollados con el fin de ayudar o reemplazar a los operadores humanos e incrementar la productividad de las empresas de construcción; esta tesis ha tomado como referencias principales dos trabajos de investigación, uno de nivel doctoral y otro de maestría presentados en *Virginia Tech (Virginia PolyTech Institute and State University - USA)* en los años 2003 y 2000 respectivamente (ver referencias [20] y [21] de la bibliografía); ambos volúmenes de tesis están disponibles en Internet.

1.6 Descripción del Sistema

En un sistema de grúa, la carga que se desea transportar está suspendida en el aire mediante un cable y los cambios en la aceleración que se producen durante el movimiento de la grúa originan oscilaciones pendulares de la carga suspendida. En esta tesis, se automatiza el transporte de la carga a través de técnicas computacionales Neuro-difusas, cuyos fundamentos se basan en la Lógica Difusa y las Redes Neuronales. Las posiciones y velocidades de los actuadores de la grúa, de la carga y del destino deseado son las entradas del sistema de automatización mientras que la salida es la magnitud de la aceleración de los actuadores de la Grúa Torre.

El diagrama de bloques del sistema se muestra en la figura 1.2.

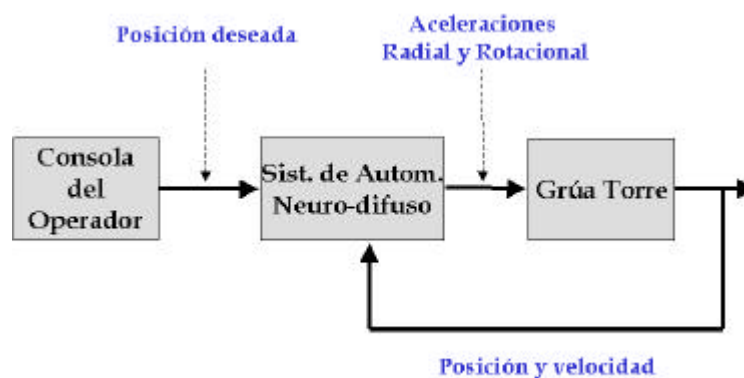


Figura 1.2 - Sistema de Control de la Grúa Torre

El Sistema de Automatización Neuro-difuso contiene los controladores de movimiento rotacional y radial (ver figura 1.3), cada uno de los cuales está compuesto de tres bloques de reglas (BR) como se puede apreciar en la figura 1.4. Los bloques de reglas contienen el conocimiento acerca del comportamiento deseado del proceso en forma de reglas SI-ENTONCES.

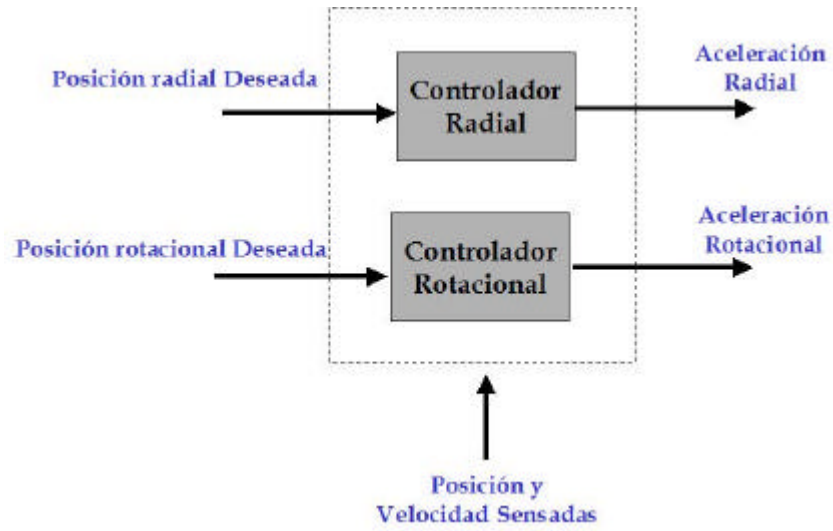


Figura 1.3 – Sistema de Automatización Neuro-difuso

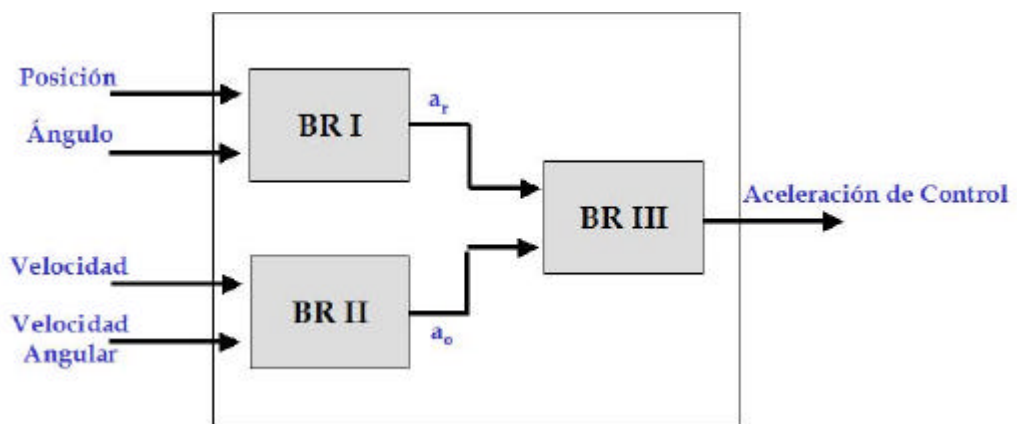


Figura 1.4 – Bloques de reglas incluidos en los controladores de movimientos radial y rotacional

Una de las ventajas que ofrece la tecnología Neuro-difusa, es que permite utilizar el conocimiento a priori del sistema para facilitar el proceso de entrenamiento. Así, antes del entrenamiento del sistema de automatización Neuro-difuso, se ingresó

conocimiento a priori (obvio) acerca del comportamiento de sistemas masa-péndulo en los BRs.

1.7 Estructura de la Presentación

El presente trabajo está organizado de la siguiente manera:

Capítulo I: Es una introducción a la tesis, con una descripción de los objetivos, las motivaciones y la metodología utilizadas.

Capítulo II: Presenta los conceptos clave que fundamentan la teoría de la Lógica Difusa y las Redes Neuronales.

Capítulo III: Presenta el desarrollo del simulador 3D de Grúa Torre y se utilizan resultados de trabajos de investigación de la universidad *Virginia Tech* para validarlo.

Capítulo IV: Describe el proceso de extracción de los datos experimentales para el entrenamiento del sistema de automatización Neuro-difuso.

Capítulo V: Consiste en el diseño del sistema de automatización Neuro-difuso y su entrenamiento con los datos obtenidos en el capítulo IV.

Capítulo VI: Se presentan los resultados de las simulaciones y se comparan los desempeños del algoritmo Neuro-difuso y los desarrollados por investigadores del *Virginia Tech*.

Capítulo VII: Presenta las conclusiones de esta tesis.

II. SISTEMAS NEURO-DIFUSOS

Los sistemas Neuro-difusos forman parte de una nueva tecnología de computación llamada computación flexible (*soft-computing*), la cual tiene su origen en la teoría de la Inteligencia Artificial. La computación flexible engloba un conjunto de técnicas que tienen en común la robustez en el manejo de la información imprecisa e incierta que existe en los problemas relacionados con el mundo real (por ejemplo: reconocimiento de formas, clasificación, toma de decisiones, etc.). En algunos casos, las técnicas de computación flexible pueden ser combinadas para aprovechar sus ventajas individuales; algunas de estas técnicas son:

- La Lógica difusa.
- Las Redes Neuronales.
- Los Algoritmos Evolutivos o Algoritmos Genéticos.
- La Teoría del Caos.
- La Teoría del Aprendizaje.
- El Razonamiento Aproximado.

En los siguientes apartados se explicarán brevemente las técnicas de Redes Neuronales, Lógica Difusa y la combinación de ambas, por ser las de principal interés para el desarrollo de esta tesis.

2.1 Introducción a las Redes Neuronales

Las Redes Neuronales Artificiales (RNAs) son la implementación en hardware y / o software de modelos matemáticos idealizados de las neuronas biológicas. Las neuronas artificiales son interconectadas unas a otras y son distribuidas en capas de tal forma que emulan en forma simple la estructura neuronal de un cerebro. Cada modelo de neurona es capaz de realizar algún tipo de procesamiento a partir de estímulos de entrada y ofrecer una respuesta, por lo que las RNA en conjunto funcionan como redes de computación paralelas y distribuidas similares a los sistemas cerebrales biológicos. Sin embargo, a diferencia de las computadoras convencionales, las cuales son programadas para realizar tareas específicas, las redes neuronales artificiales, tal como los sistemas cerebrales biológicos, deben ser entrenadas.

En la siguiente tabla se presenta una comparación entre las computadoras Von Neumann, y las redes neuronales artificiales¹:

Tabla 2.1 - Computadora Von Neumann vs Sistema Neuronal

	Computadora	Red Neuronal Artificial
Procesador	- Complejo - Alta velocidad - Uno o pocos	- Simple - Baja velocidad - Un gran número
Memoria	- Separada del procesador - Localizada - No es direccionable por el contenido.	- Integrada en el procesador distribuida - Direccionable por el contenido
Computación	- Centralizado - Secuencial - Programas Almacenados	- Distribuido Paralelo - Auto-Aprendizaje
Confiabilidad	- Muy vulnerable	- Robusta
Punto fuerte	- Manipulaciones numéricas y simbólicas	- Problemas de percepción
Ambiente operativo	- Bien definido - Bien limitado	- Pobremente definido - Ilimitado

2.1.1 La Neurona Biológica

El sistema nervioso humano, incluido el cerebro, consiste en la interconexión de millones y millones de pequeñas unidades celulares llamadas neuronas², por ejemplo, sólo la corteza cerebral humana, una capa larga y plana de neuronas de alrededor 2 a 3 milímetros de ancho con un área superficial de aproximadamente 2,200 cm², contiene alrededor de cien mil millones (10¹¹) de neuronas¹.

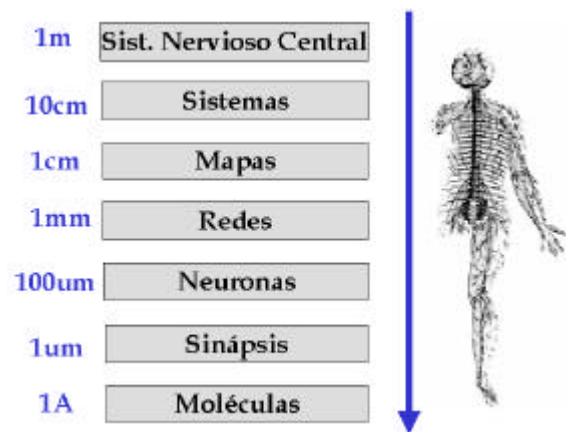


Figura 2.1 - Sistema Nervioso

En cuanto a su fisiología, las neuronas constan de un cuerpo celular o soma, en donde se aloja el núcleo de la célula. Del cuerpo de la célula salen ramificaciones de diversas fibras conocidas como dendritas y sale también una fibra más larga denominada axón. En su extremo, el axón también se ramifica en filamentos y sub-filamentos mediante los que establece conexión con las dendritas y los cuerpos de las células de otras neuronas; estas conexiones son conocidas como sinapsis. La comunicación entre neuronas ocurre como resultado de la liberación, por parte de la neurona pre-sináptica, de sustancias químicas con propiedades eléctricas llamadas neurotransmisores y la sub-secuente absorción de esas sustancias por parte de la neurona post-sináptica.

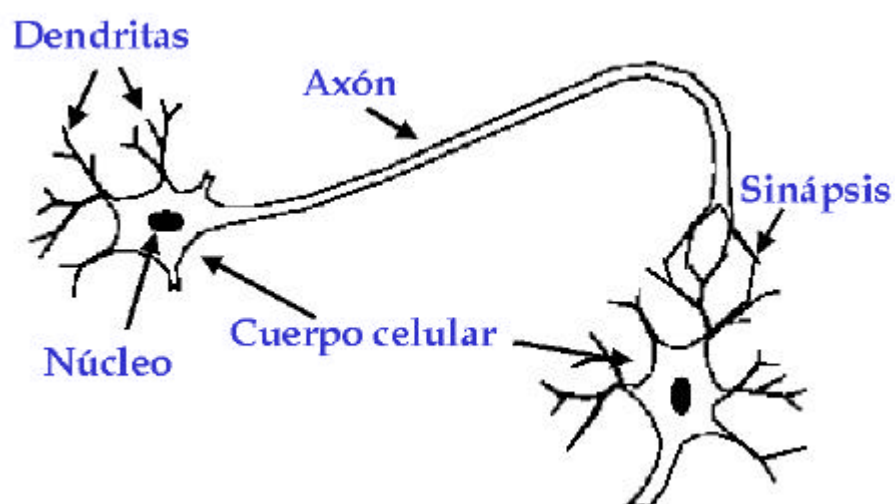


Figura 2.2 - Neurona biológica

Aunque la operación de una neurona biológica no está totalmente esclarecida, la mayor parte de los biólogos concuerdan en que la neurona después de recibir entradas a través de sus dendritas, las procesa en su soma, transmite el resultado del cálculo a través de su axón y utiliza las conexiones sinápticas para transportar la señal desde el axón a miles de dendritas pertenecientes a otras neuronas³.

2.1.2 Componentes de una Red Neuronal Artificial

Ciertamente existen computadoras que tienen la habilidad de realizar procesamiento paralelo, pero el gran número de procesadores que serían necesarios para imitar el cerebro no puede ser alcanzado con la tecnología de hardware actual; además, la estructura interna de una computadora no puede ser cambiada mientras realiza una tarea⁴. Estos hechos conducen a la utilización de un modelo idealizado de neurona biológica elemental para propósitos de simulación.

Aunque existen diferentes tipos de Redes Neuronales Artificiales todos ellos tienen casi los mismos componentes elementales. Como en el sistema nervioso biológico, una red neuronal artificial está constituida por neuronas que están unidas entre sí a través de conexiones, a las cuales se les asignan valores numéricos o pesos que representan el conocimiento de la Red Neuronal. Al cambiar los valores de los pesos se consigue imitar el cambio en la estructura de las conexiones sinápticas que ocurre durante el proceso de aprendizaje en la red neuronal biológica.

La siguiente figura muestra una neurona artificial idealizada.

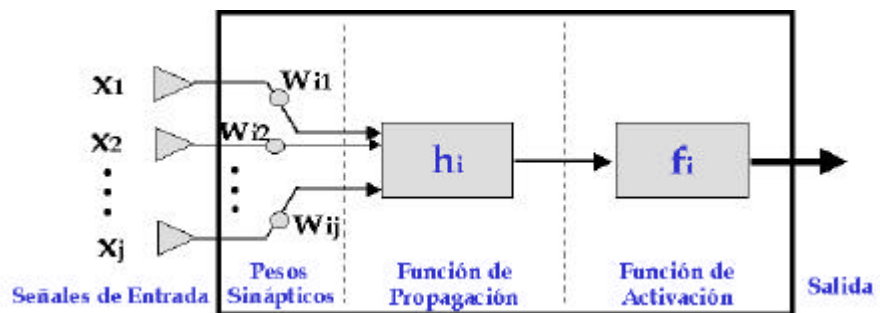


Figura 2.3 – Neurona Artificial

Como se puede observar en la figura 2.3 la neurona artificial luce similar a su contraparte biológica.

La información que conforma un conjunto de entradas $X_j(t)$ es enviada a la neurona a través de sus conexiones con pesos sinápticos W_{ij} , donde el subíndice i representa a la neurona i . Esta entrada es procesada por una función de propagación (por ejemplo: $h_i(t) = (W_{ij} \cdot X_j)$). El resultado es comparado con un valor umbral θ por la función de activación $y_i(t) = f_i(h_i(t))$ que representa simultáneamente la salida de la neurona y su estado de activación. Sólo si la entrada excede el valor umbral, la neurona se activará, en caso contrario se inhibirá.

En una RNA las neuronas suelen estar agrupadas en capas. Se conoce como capa o nivel a un conjunto de neuronas cuyas entradas provienen de la misma fuente, y cuyas salidas tienen el mismo destino. Usualmente cada neurona de una capa está conectada a todas las neuronas de las capas anterior y posterior (excepto en la capa de entrada y en la capa de salida)⁵.

Las Redes Neuronales permiten resolver problemas que no pueden ser solucionados usando algoritmos convencionales. Tales son usualmente problemas de clasificación u optimización. Los diferentes dominios en los que las redes neuronales son utilizadas, incluyen:

- Asociación de patrones.
- Clasificación de patrones.
- Procesamiento de imágenes
- Reconocimiento de voz.
- Problemas de optimización.
- Simulación.

2.1.3 Tipos de Redes Neuronales

Existen diferentes tipos de redes neuronales y cada uno tiene características especiales, por lo que cada tipo de problema tiene su propio tipo de red neuronal para

solucionarlo. Las redes neuronales pueden ser clasificadas según el tipo de aprendizaje, el tipo de aplicación y la arquitectura de la conexión. La siguiente tabla, presenta ésta clasificación.

Tabla 2.2 – Tipos de Redes Neuronales

Clasificación por el tipo de Aprendizaje	- Redes Neuronales con Aprendizaje Supervisado. - Redes Neuronales con Aprendizaje no-Supervisado. - Redes Neuronales con Aprendizaje Híbrido. - Redes Neuronales con Aprendizaje Reforzado. - Redes Neuronales con Aprendizaje Competitivo
Clasificación por el tipo de aplicación	- Redes Neuronales Aproximadoras de Funciones. - Redes Neuronales Asociativas o Memorias Asociativas. - Redes Neuronales Clasificadoras.
Clasificación por la Arquitectura de la conexión	- Redes Neuronales Monocapa. - Redes Neuronales Multicapa. - Redes Neuronales Realimentadas.

2.1.4 Arquitectura de Redes Neuronales Artificiales

Diferentes tipos de interconexión implican diferentes comportamientos de la red. Por ejemplo, las redes que tienen flujo de datos unidireccional (*feedforward*) son estáticas, es decir, producen sólo un grupo de valores de salida en lugar de una secuencia de valores de salida para una entrada dada, además sus salidas no dependen de los valores anteriores de la red. Por otro lado las redes neuronales recurrentes o realimentadas son sistemas dinámicos¹. Según la arquitectura de la conexión las redes neuronales se pueden clasificar, entre otras, como: Red Neuronal Monocapa, Red Neuronal Multicapa y Red Neuronal Realimentada.

Red Neuronal Monocapa

Las redes monocapa están formadas sólo por una capa de neuronas, y suelen utilizarse frecuentemente en tareas relacionadas con la regeneración de información incompleta o distorsionada que se presenta a la red.

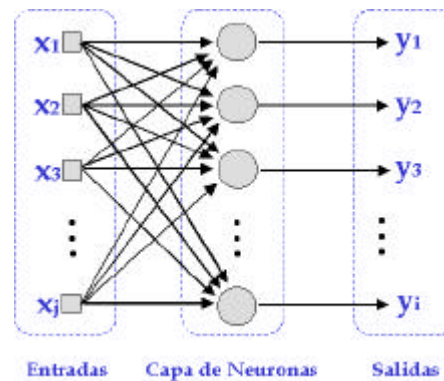


Figura 2.4 - Red Neuronal Monocapa

Red Neuronal Multicapa

Son las estructuras más comunes; como se puede apreciar en la figura 2.5, además de la capas de entrada y salida, poseen un número de capas intermedias u ocultas que mejoran su desempeño.

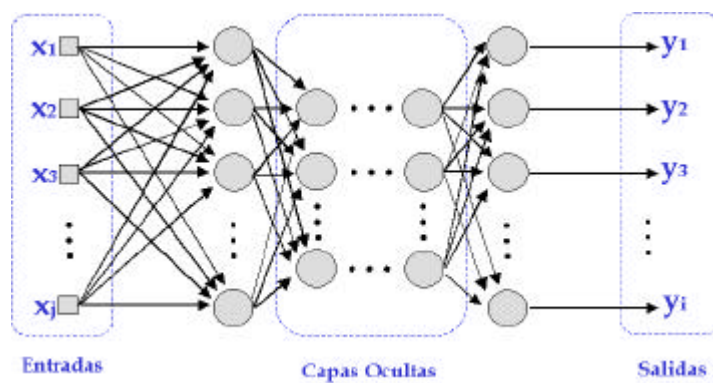


Figura 2.5 - Red Neuronal Multicapa

Red Neuronal Realimentada

Se caracteriza porque sus salidas pueden ser utilizadas como entradas. La estabilidad de la red es un importante factor a considerar en este tipo de arquitectura.

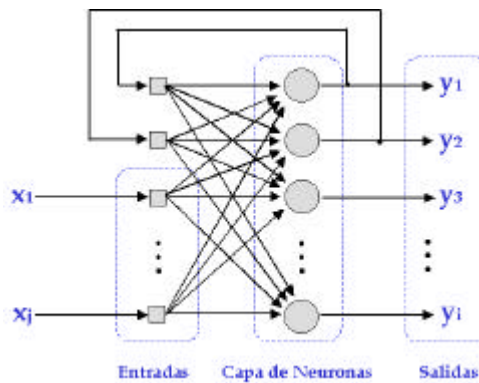


Figura 2.6 - Red Neuronal Realimentada

2.1.5 El Proceso de Aprendizaje

Como ya se mencionó, biológicamente se suele aceptar que la información memorizada en el cerebro está más relacionada con los valores sinápticos de las conexiones entre las neuronas que con las neuronas mismas, es decir, el conocimiento se encuentra en las sinapsis y todo proceso de aprendizaje consiste en la creación, modificación y destrucción de estas conexiones entre las neuronas. De forma similar el aprendizaje en las RNA consiste en determinar un conjunto de pesos sinápticos que permita a la red realizar correctamente el tipo de procesamiento deseado, esto se logra a través del entrenamiento de la red. Una vez que la red neuronal ha sido correctamente entrenada será capaz de determinar la salida deseada para las entradas que se le presenten.

2.1.5.1 Tipos de Aprendizaje

Los tipos de aprendizaje son los métodos utilizados para entrenar las RNAs, algunos de ellos son: el aprendizaje supervisado y el aprendizaje competitivo.

Aprendizaje Supervisado

En este caso un agente supervisor externo presenta a la red un conjunto de patrones característicos de entrada junto con la salida que se desea obtener e iterativamente la red ajusta sus pesos hasta que su salida tiende a ser la deseada; para realizar esta tarea la red utiliza información acerca del error que comete en cada paso de entrenamiento.

Aprendizaje Competitivo

En el aprendizaje competitivo las neuronas de salida compiten entre ellas para alcanzar el estado de activación. Como resultado sólo una unidad de salida estará activa en algún momento dado. Este procedimiento es conocido como WTA (*Winner-Take-all*).

2.1.5.2 Algoritmos de Aprendizaje

El tipo de algoritmo de aprendizaje depende esencialmente del tipo de aplicación de la red, así como de su topología. A continuación se describen los algoritmos de Corrección de Error y de Retro-propagación del Error.

Algoritmo de Corrección de Error

En el aprendizaje supervisado, se le da a la red una asociación de salidas-entradas. Durante el proceso de aprendizaje, la salida 'y' generada por la red puede no ser igual a la salida deseada 'd'. El aprendizaje por corrección de error consiste pues, en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores obtenidos de la red y los valores deseados, es decir, en función del error obtenido en la salida. Un algoritmo simple de aprendizaje por corrección de error podría ser el siguiente.

$$W_{ij} = W_{ij} + \alpha \cdot x_j (d_i - y_i) \quad (2.1)$$

Donde d_i es la salida deseada, y_i es la salida real de la neurona 'i' obtenida una iteración antes, x_j es una entrada j-ésima a la neurona 'i' y α es la tasa de aprendizaje de la red o constante de velocidad de aprendizaje de la red, el resultado W_{ij} de este cálculo es el nuevo valor que será asignado al peso en la siguiente iteración. El principio esencial de los algoritmos de corrección del error es usar la señal de error ($d-y$) para modificar los pesos en las conexiones y reducir gradualmente el error.

Algoritmo de Aprendizaje de Retro-propagación del Error

El algoritmo de aprendizaje que usa una Red Neuronal Multicapa es la Retro-propagación del Error. La importancia de este algoritmo radica en su capacidad de

modificar los pesos de las neuronas de las capas intermedias de la red durante el entrenamiento.

La idea central de la retro-propagación del error es calcular los errores para las unidades de las capas ocultas a partir de los errores en las unidades de salida, para luego propagarlos capa tras capa hacia atrás hasta llegar a la capa de entrada, modificando los pesos de las neuronas en cada paso. El algoritmo debe ajustar los parámetros de la red para calcular el gradiente de error y minimizar el error medio cuadrático entre la salida deseada y la salida de la red⁶.

La siguiente explicación del algoritmo de Retro-propagación se encuentra en la referencia [7] de la bibliografía, algunos de los subíndices han sido modificados para hacerlos congruentes con los utilizados en esta tesis.

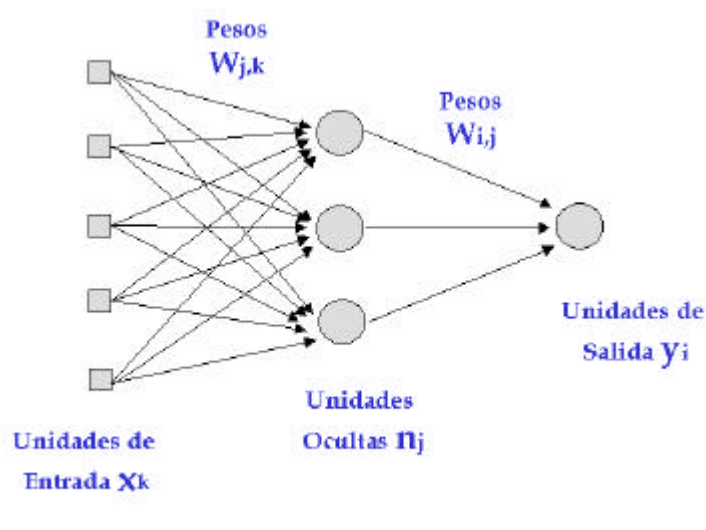


Figura 2.7– Red Neuronal Multicapa

Existen dos diferencias con respecto al algoritmo de corrección de error visto anteriormente: en vez de un valor de entrada se utiliza la activación de la unidad oculta n_j como se muestra en la figura 2.7, y la ecuación contiene un término para el gradiente de la función de activación. Si E_{rri} es el error ($d_i - y_i$) del nodo de salida, donde d_i es la salida deseada y y_i es la salida real, entonces la ecuación de actualización de los pesos del vínculo entre la unidad j y la unidad i es:

$$W_{ij} = W_{ij} + \alpha \cdot n_j \cdot E_{rri} \cdot f'(ent_i) \quad (2.2)$$

En donde f' es la derivada de la función de activación h con entrada ent_i . Si se define el nuevo término de error Δ_i como: $\Delta_i = E_{rri} \cdot f'(ent_i)$. La ecuación de actualización de los pesos se convierte entonces en:

$$W_{ij} = W_{ij} + \alpha \cdot n_j \cdot \Delta_i \quad (2.3)$$

Para actualizar las conexiones entre las unidades de entrada y las ocultas, hay que definir una cantidad análoga al término de error de los nodos de salida. Es en este momento cuando se realiza la propagación inversa del error. La idea es que el nodo oculto j es responsable de una parte del error Δ_i en cada uno de los nodos de salida con los que se conecta. Por lo tanto, los valores son Δ_i divididos de acuerdo con la intensidad de la conexión entre el nodo oculto y el nodo de salida, y se propagan hacia atrás para proporcionar los valores Δ_j del estado oculto. La regla de propagación de los valores Δ es la siguiente:

$$\Delta_j = f'(ent_j) \cdot \sum_i W_{ij} \cdot \Delta_i \quad (2.4)$$

Ahora bien, la regla de actualización de pesos correspondiente a los pesos que están entre las entradas y el nivel oculto es casi idéntica a la regla de actualización del nivel de salida:

$$W_{j,k} = W_{j,k} + \alpha \cdot x_k \cdot \Delta_j \quad (2.5)$$

2.2 Introducción a los Sistemas Difusos

Los fundamentos de los sistemas difusos se encuentran en la lógica difusa. La lógica difusa o borrosa es una técnica de computación flexible que le permite a un computador clasificar información del mundo real en una escala infinita acotada por los valores falso y verdadero; tiene por objetivo proporcionar un soporte matemático formal al razonamiento basado en el lenguaje natural, el cual se caracteriza por tratarse

de un razonamiento de tipo aproximado que hace uso de proposiciones que expresan información de carácter impreciso.

Algunas características de la lógica difusa que la hacen de tanto interés son⁸:

- Es fácil de entender, los conceptos matemáticos son bastante sencillos.
- Es flexible, su escalamiento es sencillo.
- Es tolerante a datos imprecisos.
- Puede modelar funciones no-lineales de complejidad arbitraria.
- Puede ser construida sobre la información de la experiencia de los operarios que manejan el sistema que se desea automatizar.
- Puede ser complementaria a las técnicas de control convencionales.
- Está basada en el lenguaje utilizado por los humanos.

La lógica difusa debe ser distinguida de la incertidumbre en el sentido en que la lógica difusa describe la ambigüedad de un evento, mientras que la incertidumbre la ocurrencia de un evento. Específicamente, el concepto incerteza se refiere a la incerteza estocástica, que se caracteriza por referirse a eventos bien definidos, por ejemplo,

La probabilidad de que el motor falle es de 80%

La lógica difusa trata con incertezas lingüísticas del tipo:

El motor falla constantemente.

Ambas sentencias son muy similares, sin embargo, existe una significativa diferencia en cuanto a la forma de expresar la probabilidad. Mientras en el caso de incerteza estocástica, la probabilidad es expresada en un sentido matemático, en una incerteza lingüística la probabilidad es percibida correctamente sin que haya sido cuantificada.

A continuación se explican algunos conceptos que son de gran utilidad para el estudio de la lógica difusa y la teoría de los conjuntos difusos:

Universo de Discurso

El universo de discurso denotado por U contiene todos los elementos que pueden ser tomados bajo consideración para asignar valores a las variables del sistema difuso. Los conjuntos difusos toman sus elementos del universo de discurso, por ejemplo el conjunto de gente joven podría tener a todos los seres humanos del mundo como su universo.

Función de Pertenencia

Todos los elementos dentro del universo de discurso U son miembros de algún conjunto difuso en cierto grado. La función de pertenencia es la curva que define con que grado cada elemento está incluido en el conjunto difuso. Para la definición de las funciones de pertenencia se utilizan formas estándar como la función triangular, trapezoidal, S , exponencial, singleton y π .

Variables Lingüísticas

Las variables lingüísticas son elementos fundamentales de cualquier sistema de lógica difusa. En ellas se combinan múltiples categorías subjetivas que describen el mismo concepto, así, para el caso de la variable *altura* existirán las categorías: *bajo*, *mediano*, *alto* y *muy alto*, que son llamadas términos lingüísticos y representan los posibles valores de una variable lingüística. En un lenguaje más formal, una variable lingüística se caracteriza básicamente por tres parámetros $(x, T(x), U)$ donde x es el nombre de la variable, $T(x)$ es el conjunto de términos lingüísticos de x , y U es el universo de discurso.

Grado de Pertenencia

Es el grado con el cual una entrada bien definida es compatible con una Función de Pertenencia, puede tomar valores entre 0 y 1. Por ejemplo, el Grado de Pertenencia de x al conjunto difuso *alto* (A) es representado por la función $\mu_A(x)$, donde x es un valor numérico de altura dentro del universo U ($x \in U$). El rango de μ es cualquier valor entre 0 y 1, según represente algún valor entre ningún o total grado de pertenencia al conjunto difuso.

Término

Es una categoría subjetiva de una variable lingüística, y consecuentemente, es el nombre descriptivo usado para identificar una función de pertenencia. Tal como las variables algebraicas toman valores numéricos, las variables lingüísticas toman como valores términos lingüísticos.

Entradas bien definidas (Entradas *Crisp*)

Son los diferentes valores discretos de la variable del sistema, por ejemplo las alturas medidas de un grupo de personas: 1.60m, 1.75m, 1.80m, etc. En oposición al concepto de difuso, lo *crisp*, definido, nítido o preciso no representa ninguna incerteza o imprecisión.

Rango/Dominio

Es el intervalo sobre el cual se define una Función de Pertenencia. Por ejemplo, una función de pertenencia Alto podría tener un dominio de 1.60 a 1.9 m y su rango sería de 0.3 m.

La figura 2.8 muestra gráficamente los conceptos que se acaban de revisar líneas arriba.

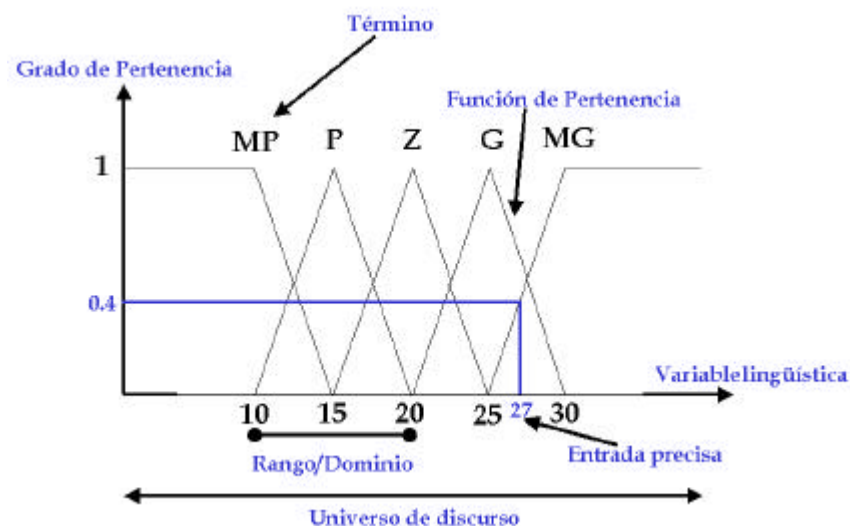


Figura 2.8 – Conceptos de lógica difusa

En las siguientes secciones se presentan algunos conceptos resaltantes de la teoría de los conjuntos difusos y la lógica difusa que son necesarios para analizar un Sistema Lógico Difuso (SLD).

2.2.1 Conjuntos Difusos

En teoría clásica de conjuntos, un conjunto tiene unos límites nítidos definidos (límites *crisp*). Por ejemplo, el conjunto A de los números más grandes que 8 se representa como:

$$A = \{ x / x > 8 \} \quad (2.6)$$

Sin embargo, según la teoría de los Conjuntos Difusos las vaguedades inherentes a los conceptos manejados por el ser humano, la transición desde “pertenecer a un conjunto” hasta “no pertenecer a un conjunto” es gradual. Así, un conjunto difuso (un conjunto sin un límite definido), contiene elementos sólo con un cierto grado de pertenencia.

A diferencia de los conjuntos clásicos que pueden ser caracterizadas ya sea por sus funciones de pertenencia, una descripción de sus elementos o un listado de sus elementos, los conjuntos difusos sólo pueden ser caracterizados por sus funciones de pertenencia; la única condición que una función de pertenencia debe satisfacer es que debe variar entre 0 y 1; la función en sí misma puede ser una curva arbitraria cuya forma la podemos definir como una función que sea agradable desde el punto de vista de simplicidad, conveniencia, velocidad y eficiencia⁸. Una forma particular para la función de pertenencia puede ser determinada sólo en el contexto de una aplicación particular pero resulta que muchas aplicaciones no son muy sensibles a las variaciones en la forma de sus funciones de pertenencia; en tales casos, es conveniente usar una forma simple, tal como la forma triangular⁹. La figura 2.9 presenta algunas funciones de pertenencia frecuentemente utilizadas:

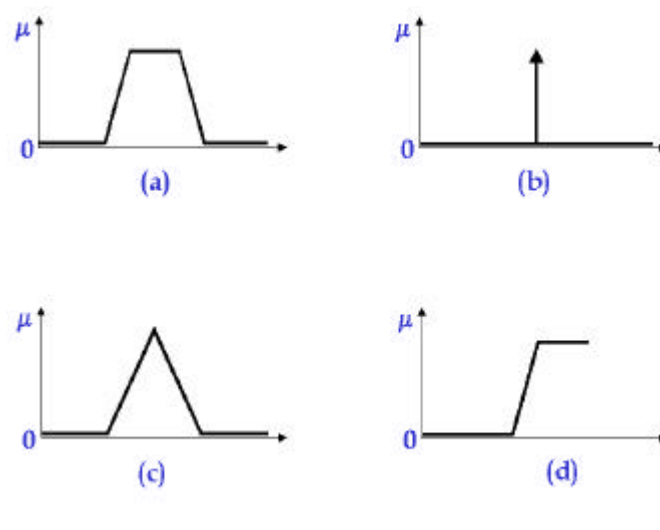


Figura 2.9 – Funciones de pertenencia. (a) Tipo trapecoidal. (b) Tipo singleton. (c) Tipo triangular. (d) Tipo z.

A los conjuntos difusos se les pueden aplicar determinados operadores, o bien puede realizarse operaciones entre ellos de la misma forma como se hace con los conjuntos crisp o clásicos. En general las funciones que definen las operaciones entre conjuntos difusos son arbitrarias en un grado sorprendente, lo que significa que uno puede crear sus propios operadores. Los operadores que califican como intersecciones difusas y uniones difusas son referidas como T-normas y T-conormas respectivamente. Por ejemplo, sean A y B dos conjuntos difusos en U con funciones de pertenencia μ_A y μ_B respectivamente, es usual definir la T-norma min, la T-conorma max, y el complemento de la siguiente manera:

- T – norma min:

$$A \cap B = \mu_A(u) \min \mu_B(u)$$

- T – conorma max:

$$A \cup B = \mu_A(u) \max \mu_B(u)$$

- Complemento de A:

$$\mu_{\bar{A}} = 1 - \mu_A(u)$$

Como lo demostró Zimmerman, los pares de T -normas y T-conormas satisfacen las propiedades conmutativas, asociativas y distributivas de los conjuntos clásicos así como las leyes de Morgan¹⁰.

La bibliografía sobre lógica difusa contiene abundante información en cuanto a la Teoría de los Conjuntos Difusos, donde se abordan temas como relación y composición difusa como extensión de sus contrapartes clásicas.

2.2.2 Lógica Difusa

Los Sistemas Lógicos Difusos son sistemas basados en reglas, las cuales son expresadas como implicaciones lógicas, es decir, en forma de sentencias **SI-ENTONCES**. La implicación refleja la relación que guarda un hecho derivado de otro y pertenece a una rama de las matemáticas conocida como lógica.

La lógica, en su variante proposicional, tiene su elemento base en la proposición o “hecho”. Una proposición es una sentencia que involucra términos definidos y debe ser apropiadamente llamada “verdadera” o “falsa”, es decir hay hechos que se cumplen o no. Sin embargo, haciendo una extensión de la teoría de conjuntos difusos, se puede decir que hay hechos que no son necesariamente ciertos o falsos, sino que tienen un cierto grado de verdad. Por ejemplo, “hoy hace calor” podría ser una oración válida en grado 0.6; así, lo que existe en el mundo no es un hecho, sino el grado de verdad de un hecho, y los seres humanos tiene un grado de certeza del hecho que puede variar entre 0 y 1(falso y verdadero).

Concretamente, se puede decir que una lógica consta de lo siguiente⁷:

- Un sistema formal para describir lo que está sucediendo en un momento determinado, y que consta de:
 - La sintaxis del lenguaje, que explica como construir oraciones
 - La semántica del lenguaje, a través de la cual cada oración expresa algo relacionado con el mundo.

- Una Teoría de demostración, que agrupe un conjunto de reglas para deducir las implicaciones de un conjunto de oraciones, y que especifique los pasos de razonamiento confiables.

Con el objetivo de mostrar la relación existente entre lógica proposicional y la lógica difusa, se presentan los conceptos referentes a sintaxis, semántica e inferencia relacionados a cada una de ellas.

Sintaxis

La sintaxis de la lógica proposicional está conformada por las constantes lógicas V (verdadero) y F (falso), los símbolos de las proposiciones tales como p y q, y los operadores lógicos conjunción ($\wedge$), disyunción ($\vee$), negación ($\sim$), equivalencia ($\leftrightarrow$) e implicación ($\rightarrow$). En lógica difusa, sin embargo, en vez de las constantes lógicas V y F se utilizan grados de verdad (grados de pertenencia a los conjuntos difusos) y los operadores lógicos son la negación difusa, las T-normas y las T-conormas.

Tabla 2.3 – Sintaxis de la lógica proposicional y de la lógica difusa

Elemento de Sintaxis	Lógica Clásica	Lógica Difusa
Constantes Lógicas	Verdadero (V) Falso (F)	Grado de verdad que varía entre 0 y 1
Símbolos de Proposiciones	p, q, r, etc.	p, q, r, etc.
Operadores Lógicos	Conjunción Disyunción Negación Implicación Equivalencia	T - normas T - conormas Negación difusa

Semántica

Al igual que en lógica proposicional, en lógica difusa, la semántica se define especificando la interpretación de los símbolos de proposición y de las constantes y especificando el significado de los conectores lógicos; a continuación se presenta una tabla de los valores semánticos de los elementos de la lógica difusa y de la lógica proposicional.

Tabla 2.4 – Semántica de la lógica proposicional y de la lógica difusa

Elemento	Significado Semántico	
	Lógica proposicional	Lógica difusa
p, q, r, etc	Un hecho. Ejm: Lima es una ciudad grande	Un grado de verdad. Ejm: Lima es una ciudad grande en un grado 0.6
Constantes lógicas	Verdadero: El hecho de la verdad Falso: No existe en el mundo	Certeza en cierto grado de verdad
Operadores lógicos	Y, no, o, etc.	Y, no, o, etc.

Inferencia

La inferencia lógica es un proceso mediante el cual se implanta la relación de implicación que existe entre oraciones⁷; es decir, si se cuenta con determinadas premisas y una posible conclusión, se puede averiguar si la conclusión es verdadera. Para ello, en lógica proposicional se construye una tabla de verdad para la oración *Premisas* $\rightarrow$ *conclusión* y se procede a verificarla.

El procedimiento que permite cerciorarse de la confiabilidad de una inferencia mediante tablas de verdad, podría emplearse a clases enteras de inferencias pues existen ciertos patrones que se presentan una y otra vez, lo que permite establecer a simple vista su confiabilidad. De esta manera se aprehende el patrón respectivo en algo que se conoce como regla de inferencia. Una vez establecida la regla, se le puede emplear para hacer más inferencias sin pasar por el tedioso procedimiento de la construcción de tablas de verdad. Se considera confiable una regla de inferencia si la conclusión es verdadera en todos aquellos casos en los que las premisas también son válidas. Algunas reglas de inferencia muy conocidas son: el Modus Ponens, el Modus Tolens, la Resolución, y la O-Introducción.

Si se desea aplicar la lógica proposicional a la ingeniería, la regla de inferencia que se ha de utilizar es el Modus Ponens, ya que satisface la Ley de Causa y Consecuencia¹⁰.

Así:

Premisa 1 (Hecho): “x es A”
 Premisa 2 (Conocimiento): “Si x es A, entonces y es B”
 Conclusión: “y es B”

En lógica difusa se utiliza el Modus Ponens Generalizado, una extensión del Modus Ponens:

Premisa 1 (Hecho): “u es A*”
 Premisa 2 (Conocimiento): “Si u es A, entonces v es B”
 Consecuencia: “v es B*”

Al comparar el Modus Ponens y el Modus Ponens Generalizado para determinar sus principales diferencias, en este último A* no es necesariamente igual al antecedente A y B* no es necesariamente igual al consecuente B. En lógica proposicional una regla será disparada sólo si la primera premisa es exactamente igual al antecedente de esa regla, y el resultado de tal activación es el consecuente real de la regla; en lógica difusa sin embargo, una regla es activada mientras no haya un grado cero de similitud entre la premisa y el antecedente de la regla, y el resultado de tal activación es un consecuente que tiene un grado de similitud diferente de cero con respecto al consecuente de la regla. Este proceder puede ser entendido como razonamiento aproximado.

2.2.3 Sistemas Lógicos Difusos

En general, un Sistema Lógico Difuso (SLD) es un mapeo no-lineal de un vector de datos de entrada con una salida escalar, es decir mapea números con números⁸. La teoría de los conjuntos difusos y la lógica difusa establecen las especificaciones de este mapeo no-lineal único, en cuanto es capaz de manejar datos numéricos y conceptos lingüísticos simultáneamente. Los SLD han sido aplicados exitosamente en campos tales como el control automático, clasificación de datos, análisis de decisiones, sistemas expertos y visión por computadora.

2.2.3.1 Etapas de un Sistema Lógico Difuso

Un SLD consta de tres etapas:

- Fusificación
- Reglas de Evaluación
- Defusificación

El esquema se muestra en la figura 2.10.

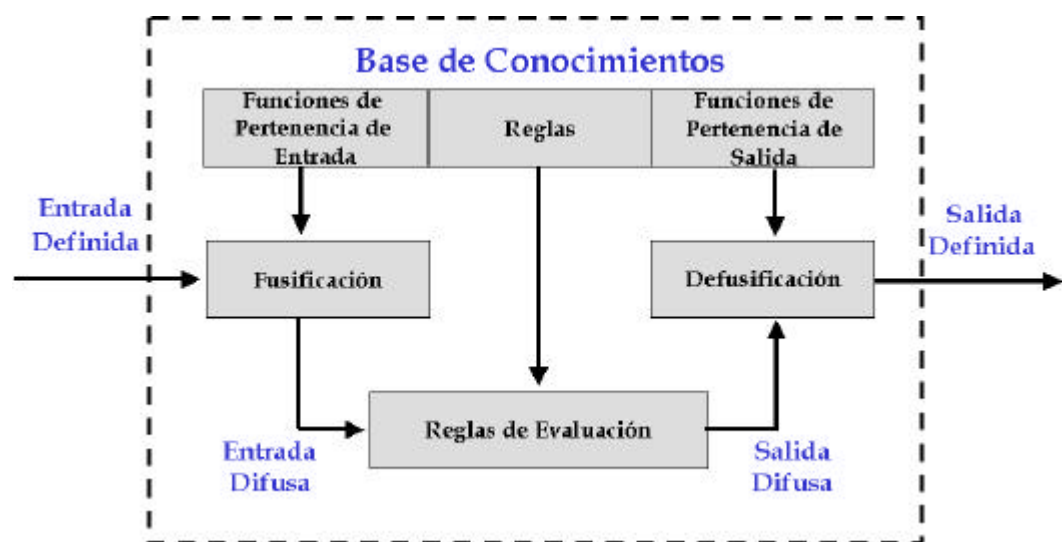


Figura 2.10 - Sistema Lógico Difuso

Fusificación

Los elementos fundamentales en esta etapa son las Funciones de Pertenencia de Entrada. La variable del proceso (entrada definida, no-difusa o *crisp*) intersecta las Funciones de Pertenencia generando las entradas difusas. Mediante este procedimiento, el fusificador establece una relación entre los puntos de entrada no difusos y sus correspondientes conjuntos difusos en el universo de discurso U .

Reglas de Evaluación

Las Reglas son sentencias **SI-ENTONCES** que describen las condiciones (antecedentes) y las acciones (consecuentes) que deben existir para tomar una decisión. La sintaxis de

las reglas es la siguiente:

SI Antecedente 1 **Y** Antecedente 2 . . . **ENTONCES** Consecuente 1 **Y** . .

El antecedente de una regla puede tener muchas partes, en tal caso todas las partes del antecedente son calculadas simultáneamente y transformadas en un número utilizando los operadores lógicos descritos en la sección precedente. El consecuente de una regla puede también tener partes múltiples y todos los consecuentes son afectados de la misma manera por el antecedente.

En el caso de lógica proposicional, evaluar las reglas **SI-ENTONCES** es sencillo. Si la premisa es verdadera, la conclusión es verdadera, por el contrario, se genera un problema cuando el antecedente es difuso, pues no se sabe como se verá afectado el consecuente. Sin embargo, de acuerdo a la teoría revisada en la sección 2.2.2, si el antecedente es verdadero en cierto grado, entonces el consecuente es verdadero en cierto grado.

Así, las reglas difusas permiten expresar el conocimiento que se tiene acerca de la relación entre los antecedentes y los consecuentes en un cierto grado de verdad. Para expresar este conocimiento de forma completa normalmente se precisa de varias reglas, con pesos asociados, que se agrupan formando una base o bloque de reglas.

Mediante la inferencia los sistemas difusos interpretan las reglas de tipo IF-THEN contenidas en su base de conocimientos, con el fin de obtener los valores de salida a partir de los valores que tienen las variables lingüísticas de entrada al sistema. Una vez que las entradas crisp han sido convertidas a variables de valores lingüísticos (fusificación), se utiliza la inferencia difusa para identificar las reglas de tipo **SI-ENTONCES** que se aplican a la situación actual y se calculan los valores lingüísticos de salida.

La inferencia es un cálculo que consiste en tres pasos: agregación de las variables lingüísticas de entrada, composición o implicación y agregación del resultado.

- **Agregación de las variables lingüísticas de entrada**

Es el primer paso que se sigue para realizar una inferencia difusa, la agregación determina el grado en el que se cumple la parte **SI** de la regla. Si la regla contiene varias premisas, estas suelen estar relacionadas por operadores lógicos difusos como T-normas y T-conormas.

- **Composición**

Es el segundo paso que se lleva a cabo para realizar la inferencia, y es conocida también como implicación difusa. Mediante la composición se comprueba la validez de la conclusión de una regla al relacionar el grado con que se cumple el antecedente de la regla, con el peso de la misma.

- **Agregación del Resultado**

Ya que las decisiones están basadas en la prueba de todas las reglas que forman un sistema de inferencia difuso los consecuentes de las reglas deben ser combinados de alguna manera para tomar una decisión. La agregación es el proceso a través del cual los conjuntos difusos que representan las salidas de las reglas son combinados en un único conjunto difuso.

Defusificación

La salida del proceso de inferencia es hasta ahora un conjunto difuso que indica la posibilidad de realizar una acción de control. Sin embargo, las aplicaciones de los sistemas difusos no pueden interpretar los valores lingüísticos obtenidos, por lo que funciones de pertenencia de salida son utilizadas para retransformar los valores difusos nuevamente en valores definidos o *crisp* mediante la defusificación.

Algunos métodos de defusificación existentes son:

- Procedimiento Máximo
- Media Ponderada
- Singleton
- Centro de Masa
- Centro de Área

El método de defusificación utilizado en el controlador Neuro-difuso que propone esta tesis es el Centro de Area. Este método consiste en hallar para cada conjunto difuso su área (A) y centro ($\bar{x}$); se suma el producto entre ellos y el resultado se divide entre la sumatoria total de las áreas como se muestra en la siguiente ecuación.

$$\text{Valor defusificado} = \bar{x} = \frac{\sum A_i \bar{x}_i}{\sum A_i} \quad (2.7)$$

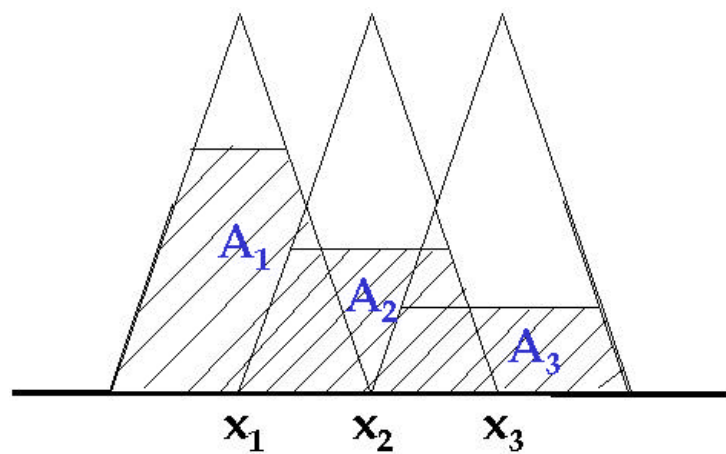


Figura 2.11 – Defusificación por Centro de Area

2.2.4 Sistemas Difusos aplicados al Control de Procesos

Desde las primeras investigaciones de Mamdani sobre una aplicación práctica para el control lógico difuso, los sistemas expertos de control difuso basados en reglas conocidos como controladores lógicos difusos o FLC (*Fuzzy Logic Controllers*) se han convertido sin duda en la aplicación más extendida de la lógica difusa².

El control lógico difuso provee un mecanismo para convertir una estrategia lingüística de control basada en el conocimiento de un operador experto en una estrategia de control automático. En este sentido, el método de control lógico difuso puede ser considerado como una aproximación entre la matemática convencional precisa y las tomas de decisiones de los humanos. Así, el lenguaje del mundo real usado en control difuso permite a los programadores incorporar la lógica ambigua de los humanos

dentro de la computadora. Además, el uso de modelos lingüísticos en lugar de modelos matemáticos mejora grandemente la transparencia del sistema y facilita las potenciales modificaciones. Los sistemas difusos pueden ser utilizados en combinación con los controladores clásicos de la Teoría de Control para lo cual existen diversos métodos; así, además del controlador difuso directo modelado líneas arriba, algunos métodos son¹¹:

- Controlador Difuso Supervisorio
- Controlador Difuso Adaptativo
- Controlador Difuso de Intervención

2.3 Introducción a los Sistemas Neuro-difusos

Como se ha visto a lo largo del presente trabajo, la Lógica Difusa y las Redes Neuronales tienen propiedades computacionales particulares que las hacen adecuadas para ciertos problemas particulares y no para otros. Por ejemplo, mientras las redes neuronales ofrecen ventajas como el aprendizaje, adaptación, tolerancia a fallas, paralelismo y generalización, no son buenas para explicar como han alcanzado sus decisiones. En cambio los sistemas difusos, los cuales razonan con información imprecisa a través de un mecanismo de inferencia bajo incertidumbre lingüística, son buenos explicando sus decisiones pero no pueden adquirir automáticamente las reglas que usan para tomarlas.

Tabla 2.5 – Lógica Difusa y Redes Neuronales

Lógica Difusa	Redes Neuronales
Permite utilizar el conocimiento disponible para optimizar el sistema directamente.	No existe un método sencillo que permita modificar u optimizar la red, ya que esta se comporta como una “caja negra”
Permite describir el comportamiento de un sistema a partir de sentencias “si – entonces”	La selección del modelo apropiado de red y el algoritmo de entrenamiento requiere de mucha experiencia
Permite utilizar el conocimiento de un experto	Permite hallar soluciones a partir de un conjunto de datos

El conocimiento es estático	Son capaces de aprender y auto-adaptarse
Existen muchas aplicaciones comerciales	Su aplicación es mayormente académica
Permiten encontrar soluciones sencillas con menor tiempo de diseño	Requieren un enorme esfuerzo computacional

Los sistemas Neuro-Difusos combinan la capacidad de aprendizaje de las RNAs con el poder de interpretación lingüística de los sistemas de inferencia difusos¹², obteniéndose los siguientes resultados:

- Aplicabilidad de los algoritmos de aprendizaje desarrollados para redes neuronales.
- Posibilidad de promover la integración de conocimiento (implícito que puede ser adquirido a través del aprendizaje y explícito que puede ser explicado y entendido).
- La posibilidad de extraer conocimiento para una base de reglas difusas a partir de un conjunto de datos.

Existen sistemas de desarrollo que han logrado unir la Lógica Difusa con las Redes Neuronales, por ejemplo se tiene:

ANFIS

ANFIS (*Adaptive Neuro Fuzzy Inference System*) es un método que permite sintonizar o crear la base de reglas de un sistema difuso, utilizando el algoritmo de entrenamiento de retro-propagación a partir de la recopilación de datos de un proceso. Su arquitectura es funcionalmente equivalente a una base de reglas tipo Sugeno¹³.

FSOM

FSOM (*Fuzzy Self-Organizing Maps*) consiste en un sistema difuso optimizado a partir de la técnica de los mapas auto-organizados de Kohonen.

NEFCLASS

El algoritmo NEFCLASS está basado en la estructura del perceptrón multicapa cuyos pesos son modelados por conjuntos difusos. Así, se preserva la estructura de una red

neuronal, pero se permite la interpretación del sistema resultante por el sistema difuso asociado, es decir, la RNA deja de ser una “caja negra”.

FuzzyTech

Es un software que propone un método de desarrollo de sistemas Neuro-difuso similar a ANFIS. FuzzyTECH es el sistema de desarrollo utilizado en esta tesis.

2.3.1 Aprendizaje en un Sistema Neuro-difuso

El Sistema Neuro-difuso consiste de un sistema difuso tradicional (ver figura 2.13) excepto que cada etapa, puede ser representada por una capa de neuronas a las que se puede proveer capacidades de aprendizaje de Redes Neuronales para optimizar el conocimiento del sistema como muestra la figura 2.12.



Figura 2.12 – Diagrama de bloques de un Sistema Neuro-difuso

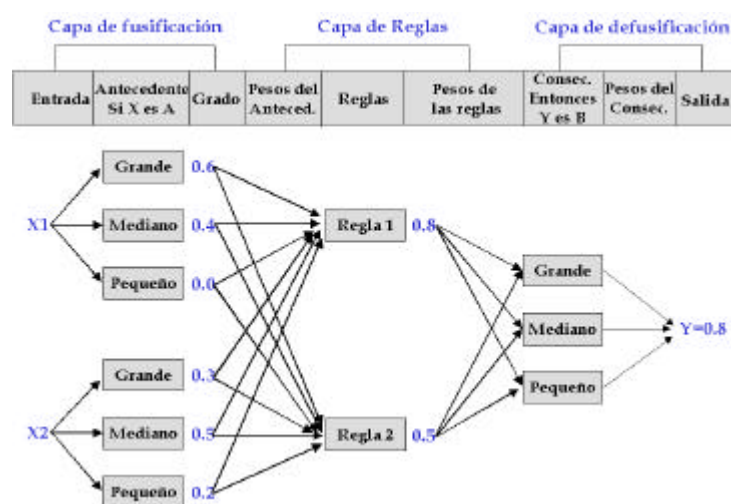


Figura 2.13 – Estructura de un Sistema Neuro-difuso

En la figura 2.13 cada pequeño rectángulo representa una neurona.

En la capa de fusificación, cada función de pertenencia de entrada del antecedente de una regla difusa representa una neurona. Los parámetros de estas neuronas, como los vértices de las funciones de pertenencia, pueden ser entrenados para determinar la forma final y la ubicación de las funciones de pertenencia.

En la figura 2.13 el grado de pertenencia que indica la certeza de “X1 es grande” es 0.6, “X1 es mediano” es 0.4, y “X1 es pequeño” es 0.0. Las salidas de estas neuronas funciones de pertenencia son conectadas a la capa de reglas difusas como lo especifiquen las reglas difusas y a través de enlaces con pesos que representan el proceso de agregación de las variable lingüísticas de entrada.

La capa de reglas difusas representa la base de reglas difusas; cada neurona representa una regla difusa de tipo SI-ENTONCES. Las salidas de las neuronas están conectadas a la capa de defusificación a través de enlaces con pesos; los pesos de estos enlaces representan la significancia relativa de las reglas asociadas con las neuronas. Sus valores pueden ser asignados de acuerdo al conocimiento a priori o inicializados como 1.0 y luego entrenadas para reflejar su importancia real para las funciones de pertenencia de salida contenidas en la capa de defusificación.

La función de la capa de defusificación es la evaluación de las reglas; en este cada consecuente “Entonces Y es B” como función de pertenencia de salida representa una neurona; la certeza de cada consecuente es calculada, y es considerada como lo bien que se ajustan las reglas que tienen el mismo consecuente (proceso de agregación del resultado). Los pesos de cada enlace de salida de estas neuronas representan los centros de área de cada función de pertenencia del consecuente y son entrenables, la salida final es entonces calculada usando algún método de defusificación.

Para realizar el entrenamiento de los sistemas neuro-difusos la estructura de la figura 2.13 puede ser configurada con valores iniciales obtenidos del conocimiento a priori, y luego, sintonizados utilizando un algoritmo de entrenamiento tal como Retro-propagación del Error, de la siguiente manera:

Paso 1: Presentar una muestra de entrada, y computar la salida correspondiente

Paso 2: Computar el error entre la salida y el valor objetivo

Paso 3: Se ajustan los pesos de conexión y las funciones de pertenencia

Paso 4: Si el error es mayor que la tolerancia, volver al paso 2, si no es así, el entrenamiento ha sido finalizado.

2.3.2 FuzzyTECH

Fuzzy TECH es un entorno de desarrollo para sistemas difusos basado en MS-Windows que permite el diseño de un sistema difuso gráficamente y la optimización del mismo utilizando un entrenamiento basado en redes neuronales. El desarrollo de un sistema Neuro-difuso en fuzzy TECH sigue los siguientes pasos¹⁴:

Paso 1: Obtener datos de entrenamiento, utilizar algún algoritmo de agrupamiento de datos si es necesario.

Paso 2: Crear un sistema lógico difuso vacío.

Paso 3: Ingresar en el sistema todo el conocimiento a priori existente para la solución.

Paso 4: Abrir los componentes del sistema lógico difuso que deberán ser entrenados.

Paso 5: Establecer los parámetros del modo Neurofuzzy

Paso 6: Entrenar con los datos muestra.

Paso 8: Evaluar la performance del sistema

Paso 9: Optimización manual.

A continuación se describen brevemente algunas características y herramientas que ofrece este programa pues es el entorno de desarrollo utilizado en la presente tesis.

2.3.2.1 Agrupamiento de Datos (*Clustering Module*)

La tecnología Neuro-difusa provee un método poderoso para convertir datos experimentales en reglas difusas. Sin embargo, en muchos casos, los datos de entrenamiento deben ser agrupados previamente, esto es necesario por dos razones:

- Eliminar los datos redundantes para alcanzar la convergencia rápidamente.

- Resolver conflictos en los datos en casos en que el conjunto de ellos sea inconsistente.

El agrupamiento o clustering es una clasificación de datos en grupos (clusters) de acuerdo a un cierto criterio de similitud¹⁵; por lo general, esto se logra maximizando la similitud intra-grupal y minimizando la similitud extra-grupal¹⁶. La similitud de los datos está especificada por la precisión (*accuracy*) de cada variable¹⁴.

Con el fin de ilustrar la función de un algoritmo de agrupamiento (*clustering*) de datos, se presenta la Figura 2.14 en la que se grafican los valores de datos en el caso de una entrada y una única salida ya que es mucho más fácil identificar clusters en espacios de datos de dos dimensiones que en espacios dimensionales más grandes.

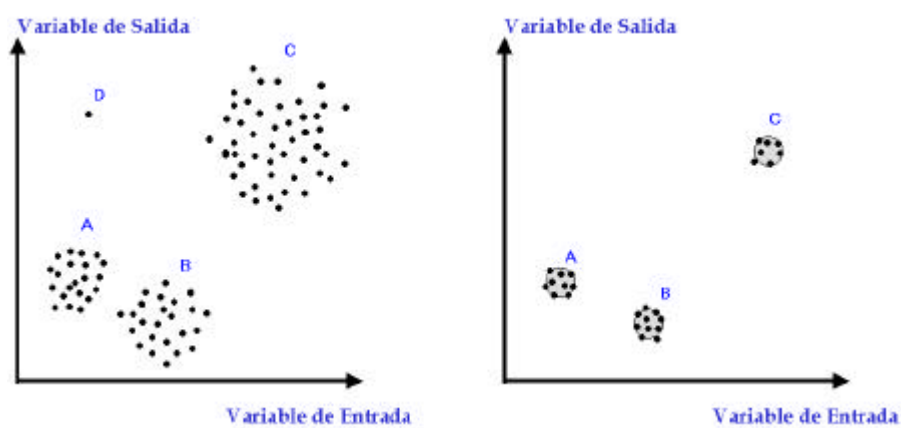


Figura 2.14 – Clusterización o agrupamiento de datos

Como se muestra en la figura 2.14, la mayor parte de los datos se agrupan con otros en tres ‘manchas’; estas manchas, son los clusters o grupos. Un dato en particular del cluster puede reemplazar al grupo para disminuir el número de datos; este es llamado dato característico o representativo del grupo; así, como resultado se obtiene un conjunto de valores característicos, que representan al conjunto de datos inicial.

FuzzyTECH utiliza dos métodos para realizar el agrupamiento de datos, uno es el IsodataCluster, y el otro es el FuzzyCluster, ambos se describen a continuación.

IsodataCluster

El algoritmo Isodata es una técnica de agrupamiento de datos muy popular. Está basado en la elección aleatoria de k centros de cluster iniciales que son actualizados de tal forma que después de un número de ciclos representan al resto de los datos tan bien como puede ser posible. Isodata es iterativo, y sigue los siguientes pasos:

- Se inicia con la asignación aleatoria de k centros de clusters iniciales
- Asigna cada dato al cluster más cercano
- Se calculan los nuevos centros de los clusters que pueden ser un número diferente a k . El cálculo se hace a partir de todos los datos individuales del cluster
- El segundo y tercer paso son repetidos hasta que el “cambio” entre las iteraciones sea pequeño. El cambio puede ser definido de varias formas, ya sea midiendo las distancias que el centro del cluster ha cambiado de una iteración a otra, o por el porcentaje de datos que han cambiado entre las iteraciones.

La función objetivo del algoritmo Isodata que debe ser minimizada es la suma de las distancias cuadradas entre cada dato y su centro de cluster asignado para minimizar la variabilidad dentro del cluster.

$$SS_{\text{dist}} = \sum_{\forall x} [x - C(x)]^2 \quad (2.8)$$

donde $C(x)$ es el centro del cluster al que el dato x está asignado. El algoritmo Isodata es muy sensible a los valores centrales iniciales de los clusters, así, es posible que dos clasificaciones del mismo grupo de datos que se inicien con valores iniciales diferentes terminen siendo clasificaciones diferentes.

FuzzyCluster

El algoritmo FuzzyCluster, como todo algoritmo difuso, no considera límites definidos (*crisp*) para determinar si un dato pertenece a un cluster o no; asocia cada variable con una función de pertenencia que determina que tan similares son dos puntos de datos, dependiendo de los valores de los parámetro delta y épsilon de la función de pertenencia (ver figura 2.15). A diferencia del Isodata cluster, la similitud de dos puntos de datos para una variable v es asignada como un grado de verdad μ_v . La distancia d_{xy} entre dos puntos es entonces computada como el mínimo negativo de las

similitudes ($d_{xy} = -\min_v \{\mu_v\}$). Los dos puntos que tienen el menor d_{xy} son considerados cercanos¹⁴.

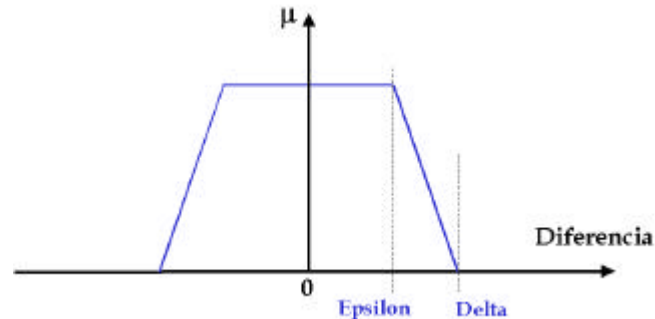


Figura 2.15 – Función de pertenencia para expresar la similitud entre datos¹⁴

La interpretación de epsilon y delta en la figura 2.15 es la siguiente:

- Epsilon es la más grande diferencia en valor de dos puntos de datos que todavía se consideran similares.
- Delta es la diferencia más pequeña en el valor de dos punto de datos que no son considerados de ninguna manera similares.

2.3.2.2 Creación de Sistemas Lógicos Difusos (*Fuzzy Tools*)

Fuzzy TECH permite la edición gráfica de las variables lingüísticas para cada una de las variables del sistema. Para esto, se pueden utilizar gran variedad de funciones de pertenencia, como las de tipo S, Z, triangular y PI. Posee también un editor gráfico de reglas, con un formato similar al de una hoja de cálculo; se pueden asociar pesos individuales a las reglas que pueden ajustarse de forma automática utilizando un entrenamiento basado en Redes Neuronales.

Todo sistema difuso desarrollado en fuzzyTECH está compuesto de tres tipos de objetos:

Texto

El texto es un objeto opcional utilizado para hacer transparente la estructura del sistema lógico difuso y describir la estructura del sistema. No tienen ninguna influencia computacional.

Variables

Las variables pueden ser de entrada, intermedias o de salida, y están ligadas cada una a una interface. Las interfaces son mostradas como pequeñas cajas que muestran el nombre de la variable y un ícono que representa el método computacional de fusificación o defusificación elegido. Las variables intermedias no están ligadas con ninguna interface y tienen influencia sólo en el bloque de reglas al que pertenecen.

Bloques de Reglas

En fuzzyTECH, las reglas individuales son contenidas dentro de bloques de reglas (BR), que conforman motores de inferencia difusos. Cada BR contiene las reglas para un conjunto de variables e implementan el proceso de inferencia difusa.

En las interfaces de entrada, ligadas a las variables lingüísticas de entrada, se permiten las siguientes herramientas para llevar a cabo la fusificación:

- Compute MBF, es el método tradicional de fusificación, el valor crisp intersecta la función de pertenencia y se determina un cierto grado de validez.
- Look up MBF, se suele usar en microcontroladores, computa las funciones de pertenencia completamente y almacena los valores en una tabla en el código generado.

En las interfaces de salida, ligadas a las variables lingüísticas de salida, se pueden utilizar las siguientes herramientas de defusificación:

- Center-of-Maximum, computa la salida crisp utilizando un promedio ponderado de los valores máximos de los términos de la variable según los resultados de la inferencia.
- Mean-of-Maximum, computa la salida sólo para el término con mayor grado de validez.
- Center-of-Area, utiliza un promedio ponderado de los valores de las áreas de los términos de las variables.

fuzzy TECH utiliza tres familias de operadores de agregación de entrada que implementan las T-normas y T-conormas (operadores lógicos difusos) del sistema:

- Min–Max (Se puede seleccionar una de ellas o una combinación de ambas).
- Min–Avg (Se puede seleccionar una de ellas o una combinación de ambas).

fuzzy TECH sólo permite el operador de composición PROD para realizar la inferencia difusa.

fuzzy TECH permite dos métodos de agregación de resultado.

- Max (*Maximum*), selecciona la regla con el mayor peso de todas las que tienen alguna relación con el término.
- Bsum (*Bounded Sum*), realiza una suma ponderada de los resultados de las reglas.

Para ilustrar el funcionamiento de las herramientas de diseño de sistemas difusos con fuzzyTECH se presenta el siguiente ejemplo de inferencia:

Supóngase que se tiene una base de conocimientos de dos reglas, y que se ha seleccionado la herramienta de agregación de entradas MAX-MIN con un parámetro de 0 (MIN), que el operador de composición es PROD y que la herramienta de agregación de salidas es MAX.

Regla 1: Si A es grande y B es frío entonces C es azul. (peso = 0.8)

Regla 2: Si A es pequeño y B es caliente entonces C es azul. (peso = 0.6)

Si el grado de verdad del antecedente A es grande es 0.2, A es pequeño es 0.8, B es frío es 0.6 y B es caliente es 0.4; entonces, el grado de validez de las premisas, obtenido mediante la agregación de las entradas utilizando el operador MIN, en la regla 1 es 0.2 y en la regla 2 es 0.4. Utilizando el operador PROD para determinar la validez del consecuente se obtiene 0.16 ($0.2 \cdot 0.8$) para la regla 1 y 0.24 ($0.6 \cdot 0.4$), y como resultado de la agregación de las salidas (operador MAX), se obtiene que la salida difusa tiene un grado de validez 0.24 azul.

2.3.2.3 Entrenamiento de un Sistema Difuso (*Neurofuzzy Module*)

El módulo Neurofuzzy de fuzzyTECH contiene herramientas para entrenar un sistema Neuro-difuso; en general, todas ellas combinan el algoritmo de entrenamiento de Retro-propagación con la idea de aprendizaje competitivo, con el fin de evitar que todas las neuronas activas que producen error sean modificadas, ya que esto podría incrementar el error del sistema al ingresar nuevas muestras.

Para realizar el entrenamiento, las herramientas de aprendizaje deben ser configuradas previamente; el módulo Neuro-difuso de fuzzy TECH permite el ajuste de los siguientes elementos: parámetros de aprendizaje, herramientas de aprendizaje, modo de selección de datos y criterios para detener el entrenamiento.

Parámetros de Aprendizaje

Los parámetros que se deben especificar antes de iniciar el aprendizaje son el ancho de paso (Step Width), y el número de neuronas ganadoras. El ancho de paso de entrenamiento es la tasa de aprendizaje del algoritmo de Retro-propagación del Error y determina que valor es añadido o sustraído a los pesos de la neurona ganadora; la determinación de este parámetro depende de la velocidad con que se desea alcanzar una solución, y de la convergencia del sistema. El número de neuronas ganadoras determina cuantas neuronas son actualizadas dentro de cada paso de iteración; mientras más neuronas ganadoras haya, es menor la protección frente al cambio de resultados de entrenamiento satisfactorios alcanzados por muestras anteriores.

Herramientas de Aprendizaje

El módulo Neurofuzzy contiene herramientas de entrenamiento estándar (REAL-METHOD y RANDOM – METHOD) y “batch” (BATCH – LEARN y BATCH – RANDOM) que utilizan el algoritmo de Retropropagación del error. La primera de cada una de estas herramientas utiliza la tasa de aprendizaje ingresada como paso de entrenamiento constante, mientras que la segunda utiliza pasos definidos de forma aleatoria a partir del intervalo [0... tasa de aprendizaje] para la actualización; además, los métodos estándar, efectúan un paso de optimización por cada muestra, mientras

que los métodos Batch, calculan los errores y gradientes para todas las muestras, y actualizan los parámetros del sistema después de cada iteración completa¹⁴.

Modo de Selección de datos

Los datos de muestra puede ser proveídos al sistema en forma secuencial o en forma aleatoria; la elección y la secuencia de las muestras influencia en el éxito del entrenamiento pues es posible que el resultado de un paso de entrenamiento, o una serie de pasos de entrenamiento, destruya el entrenamiento de los pasos de entrenamiento anteriores y vice-versa.

Criterios para detener el entrenamiento

FuzzyTECH permite tres tipos de criterios para finalizar el proceso de entrenamiento; estos son: (a) por el número máximo de iteraciones (Max. Steps), es decir detener el entrenamiento después de un número fijo de iteraciones; (b) por la desviación promedio (Avg. Dev.), criterio alcanzado si el promedio de los errores durante una iteración completa es menor que un valor umbral definido; y (c) por la máxima desviación (Max. Dev.), que compara el error de la peor muestra con un umbral de error, en este caso, las muestras con un error por debajo del umbral de error no son consideradas en el entrenamiento.

2.3.2.4 Implementación de sistemas Neuro-difusos

La implementación de sistemas difusos y neuro-difusos en fuzzyTECH se puede llevar a cabo de dos maneras: autónoma y servidor-cliente. En el primer caso el sistema neuro-difuso opera independientemente del entorno de desarrollo, para lo cual se pueden utilizar los generadores de código de fuzzyTECH, los cuales permiten generar código fuente C, Cobol, M, así como código fuente para microcontroladores, PLCs y DSPs.

De otro lado, la implementación tipo servidor-cliente permite que el kernel de fuzzyTECH trabaje como servidor de datos para MS Excel, MS VisualBasic u otras aplicaciones de MS Windows, a través de las interfaces DDE, RCU-API(DLL) y *Serial Link*.

III. BASES TEÓRICAS Y EXPERIMENTALES PARA EL DESARROLLO DEL SIMULADOR DE GRÚA TORRE

En este capítulo se presenta el fundamento teórico y el proceso de diseño del simulador de Grúa Torre 3D que permitirá analizar visualmente el funcionamiento de la Grúa Torre, y a través de cuya manipulación se extraerán datos de entrenamiento para el sistema de automatización Neuro-difuso.

Para validar el modelo matemático del simulador se aplicarán sobre él los algoritmos de control desarrollados en las tesis de la universidad *Virginia Tech* esperando obtener resultados similares. Puesto que ambos investigadores probaron exitosamente sus algoritmos sobre un prototipo real, se puede inducir que si los resultados de esta tesis son similares a los de los investigadores mencionados, el sistema de automatización propuesto en el presente trabajo también se podría implementar exitosamente en un prototipo físico.

3.1 Ecuaciones que describen el funcionamiento de una Grúa Torre

En este apartado se realiza la descripción del modelo matemático de la Grúa Torre y se derivan las ecuaciones que describen su funcionamiento. Para conseguirlo, se sigue el método de las Ecuaciones de Movimiento de Lagrange que se basa en el Principio de Conservación de la Energía¹⁷.

Muchos de los estudios realizados sobre control automático de grúas se basan en modelos lineales que permiten simplificar el análisis dinámico y el desarrollo de los controladores. Esta tesis, por el contrario, utiliza un modelo no-lineal de Grúa Torre para aprovechar las ventajas del método Neuro-difuso en la automatización de sistemas no-lineales.

Específicamente en la dinámica de la Grúa Torre se considera la relación entre su geometría y la de la carga oscilante que se encuentra suspendida por un cable.

3.1.1 Funcionamiento de una Grúa Torre

Una Grúa Torre consiste básicamente en una torre que soporta en su parte superior una pluma horizontal giratoria sobre la cual hay un carro que se desplaza radialmente (ver figuras 3.1 y 3.2). Esto permite que el punto del que pende la carga pueda moverse hacia cualquier posición horizontal deseada dentro del alcance de la Grúa utilizando sus grados de libertad horizontal y radial. La elevación y el descenso de la carga se consigue al variar la longitud del cable del mecanismo de elevación, este grado de libertad permite a la grúa iniciar y finalizar el proceso de transporte de materiales.

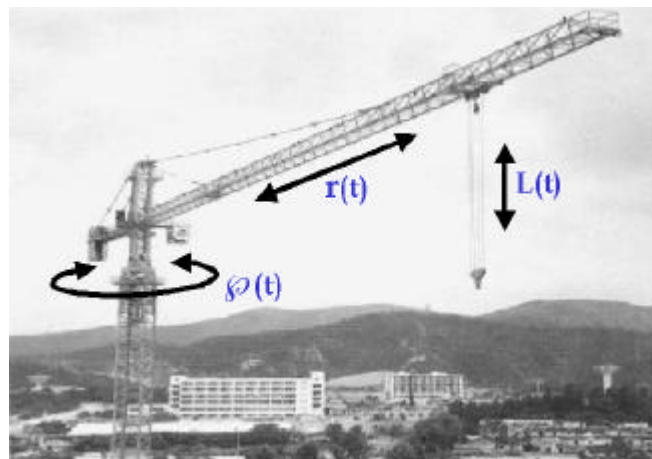


Figura 3.1 - Movimientos de una Grúa Torre

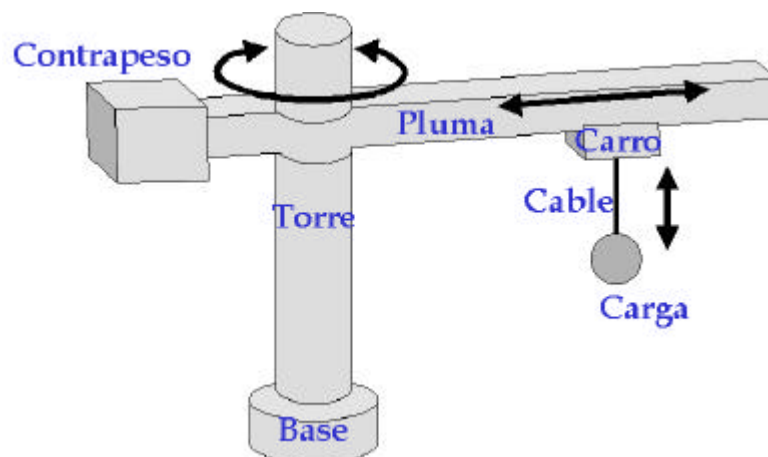


Figura 3.2 - Estructura simplificada de una Grúa Torre

La Figura 3.2 detalla la estructura simplificada de una Grúa Torre, la cual consiste de:

- Una torre que sostiene una pluma giratoria.
- La base que se fija al suelo para anclar la grúa y prevenir oscilaciones.
- La pluma de la grúa, sostenida por la torre, responsable del movimiento rotacional de la grúa.
- Un carro que se desplaza sobre la pluma.
- Un contrapeso para equilibrar la estructura.
- Un sistema mecánico de cables y poleas que hacen posible el ascenso y descenso la carga.

3.1.2 Coordenadas y Variables Físicas

Como muestra la figura 3.3 un sistema de coordenadas cartesianas es centrado en el punto de intersección de los ejes de la pluma y la torre. El sistema cartesiano se basa en la regla de la mano derecha y el eje **Z** positivo señala hacia arriba sobre la torre de la grúa. Los ejes **X** e **Y** forman el plano horizontal sobre el que se desplaza la pluma de la grúa con el eje **X** atravesando longitudinalmente la pluma.

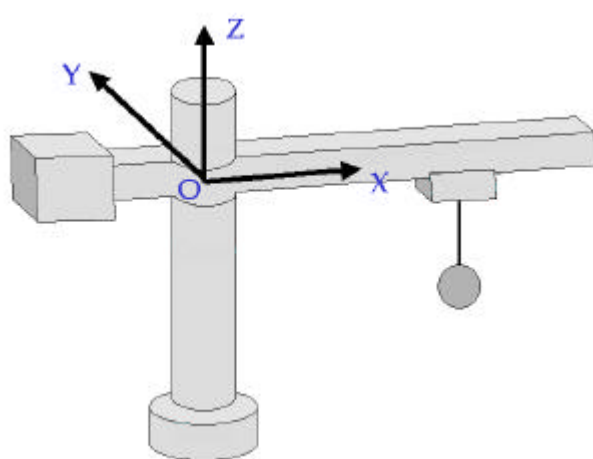


Figura 3.3 - Ubicación de los ejes coordenados

Los parámetros de entrada del sistema se pueden observar en la figura 3.4. El movimiento de rotación de la grúa traza un ángulo $\varphi(t)$; el movimiento radial del carro sobre la pluma origina un cambio en la posición del carro denotado por $r(t)$, siendo la posición inicial el origen del sistema de coordenadas **XYZ**; la altura a la que se encuentra la carga depende de la longitud $L(t)$ del cable de la Grúa Torre. En

consecuencia, para posicionar la carga $m(L)$ en algún punto del área de trabajo deben especificarse los valores de $\phi(t)$, $r(t)$ y $L(t)$; estos tres parámetros son las entradas controlables del sistema de Grúa Torre.

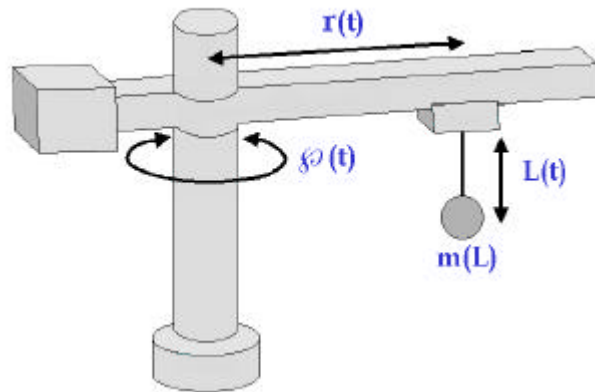


Figura 3.4 - Parámetros de entrada del sistema

Las pendulaciones de la carga están caracterizadas por los ángulos $\phi(t)$ y $\theta(t)$. El ángulo $\phi(t)$ es el ángulo que traza el cable con el eje Z sobre el plano XZ y el ángulo $\theta(t)$ es el ángulo que traza el cable con el eje Z sobre el plano YZ como se muestra en las figuras 3.5, 3.6 y 3.7. El objetivo del operador humano o sistema de automatización, es mover la carga que pende del punto Q rápidamente y manteniendo pequeños los ángulos $\theta(t)$ y $\phi(t)$.

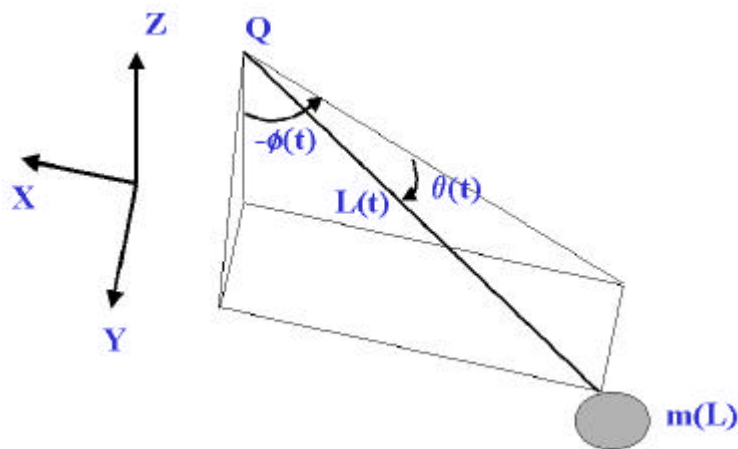


Figura 3.5 - Ángulos de oscilación

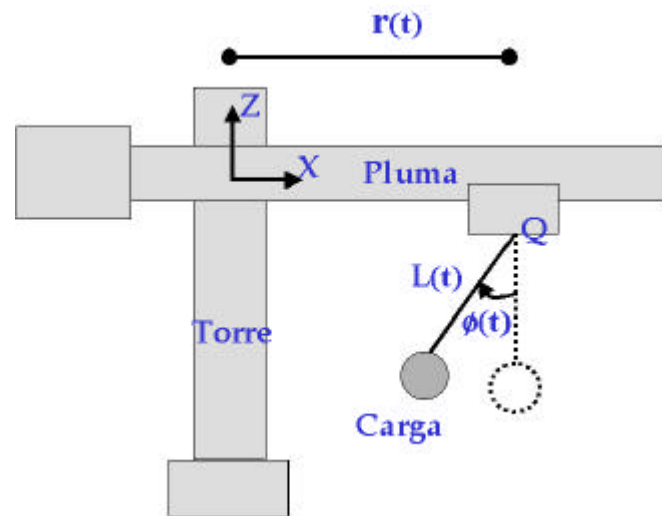


Figura 3.6 - Vista lateral del modelo de Grúa Torre

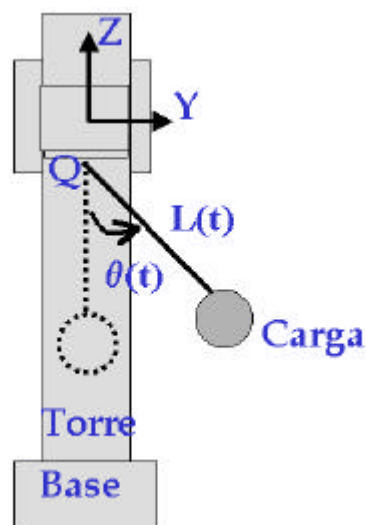


Figura 3.7 - Vista frontal del modelo de Grúa Torre

De lo visto anteriormente, se concluye que el sistema de Grúa Torre tiene 5 parámetros que determinan la ubicación espacial de la carga suspendida: $\varphi(t)$, $r(t)$, $L(t)$, $\theta(t)$ y $\phi(t)$.

Al considerar el movimiento rotacional de la grúa, se puede notar que el punto de suspensión y la carga están referidos a un sistema de coordenadas cartesiano $\mathbf{X_A Y_A Z_A}$ que rota un ángulo $\varphi(t)$ con respecto al sistema de coordenadas cartesianas inicial $\mathbf{X Y}$

Z estático, ambos sistemas están ubicados en la intersección de los ejes de la torre y la pluma.

El sistema cartesiano $\mathbf{X}_A \mathbf{Y}_A \mathbf{Z}_A$ rota con respecto al sistema cartesiano $\mathbf{X} \mathbf{Y} \mathbf{Z}$ sin movimiento de traslación relativo, y teniendo como eje de rotación el eje **Z**, común a ambos. Ambos sistemas cartesianos tienen origen común (ver figura 3.8).

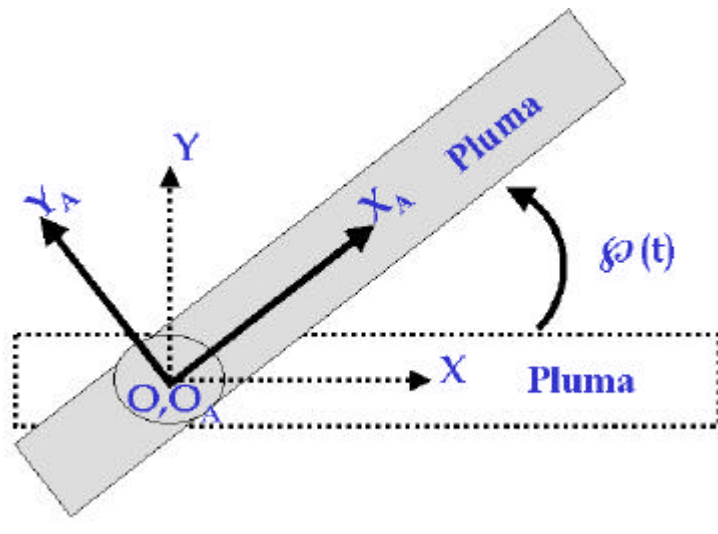


Figura 3.8 - Vista superior de la Grúa Torre

Si se consideran dos observadores: O en el sistema $\mathbf{XYZ}$ y O_A en el sistema $\mathbf{X}_A \mathbf{Y}_A \mathbf{Z}_A$, el observador O notará que el sistema $\mathbf{X}_A \mathbf{Y}_A \mathbf{Z}_A$ está rotando con velocidad angular w . Siguiendo consideraciones físicas y vectoriales es posible demostrar que la velocidad V de cualquier punto A medida por O_A en el sistema $\mathbf{X}_A \mathbf{Y}_A \mathbf{Z}_A$ con respecto al observador O en el sistema $\mathbf{XYZ}$ es¹⁸:

$$V = V_A + w \times \vec{r} \quad (3.1)$$

Donde V_A es la velocidad medida por O_A en su marco de referencia, w es la velocidad angular del sistema $\mathbf{X}_A \mathbf{Y}_A \mathbf{Z}_A$ con respecto al sistema $\mathbf{XYZ}$ y $\vec{r}$ es la distancia desde el origen del sistema $\mathbf{XYZ}$ a algún punto A, la ecuación 3.1 es de gran utilidad para hallar las ecuaciones de movimiento de la Grúa Torre.

3.1.3 Ecuaciones de Movimiento

Para derivar las ecuaciones de movimiento de la Grúa Torre se sigue el enfoque de las Ecuaciones de Movimiento de Lagrange, el cual está basado en el Principio de Conservación de la Energía. Para esto es necesario definir las coordenadas generalizadas, las fuerzas generalizadas y el Lagrangiano, para luego establecer el principio de Hamilton¹⁷.

Del análisis geométrico de la figura 3.5 se obtiene que el vector de posición de la carga $\vec{P}_L(t)$ y el vector de posición del carro $\vec{P}_C(t)$ que se desplaza sobre el eje **X**, ambos con respecto al punto de referencia del marco **XYZ**, son:

$$\vec{P}_L(t) = \{r(t) - L(t)\cos\theta(t)\sin\phi(t), L(t)\sin\theta(t), -L(t)\cos\theta(t)\cos\phi(t)\} \quad (3.2)$$

$$\vec{P}_C(t) = \{r(t), 0, 0\} \quad (3.3)$$

Ya que el marco de referencia **X_AY_AZ_A** tiene un movimiento relativo rotacional con respecto al marco **XYZ**, entonces de la ecuación 3.1, las velocidades del carro y de la carga con respecto al observador O están dadas según:

$$\vec{P}' = \frac{d\vec{P}}{dt} + \vec{w} \times \vec{P} \quad (3.4)$$

donde por consideraciones físicas¹⁸ $\vec{w}(t) = \{0, 0, \dot{\phi}'(t)\}$ es la velocidad angular para un ángulo de rotación $\dot{\phi}(t)$, y $\vec{P}(t)$ puede ser vector el de posición de la carga o el vector de posición del carro.

Ya que el modelo de Grúa Torre posee una carga, una torre y un carro, cada uno de estos elementos deberá introducir energía en el sistema, por lo que la energía cinética total del sistema estará dada por:

$$T = \frac{1}{2} m \vec{P}'L(t) \bullet \vec{P}'L(t) + \frac{1}{2} M \vec{P}'c(t) \bullet \vec{P}'c(t) + \frac{1}{2} J_o \cdot \dot{\phi}^2(t) \quad (3.5)$$

donde J_o es el momento de inercia del segmento de torre y de la pluma cuando rotan alrededor del eje **Z**.

La energía potencial depende sólo de la posición de la carga y está dada por:

$$U = -mgL(t)\cos\theta(t)\cos\phi(t) \quad (3.6)$$

El vector de coordenadas generalizadas del sistema de Grúa Torre es:

$$\vec{q} = \{ r(t), \phi(t), \varphi(t), \theta(t) \} \quad (3.7)$$

Se debe remarcar que en la ecuación 3.7 no se está considerando la longitud del cable como una coordenada generalizada a pesar de que la posición de la carga en el espacio depende de este parámetro; esto se debe a que durante la operación de las Grúas Torres reales y por motivos de seguridad, el movimiento del cable se realiza muy lentamente durante la operación de la grúa, o sólo cuando la grúa está estática y no produce oscilaciones de la carga; en consecuencia, para el modelo que se desarrolla en esta tesis se considera que la longitud de la cuerda es una función dependiente del tiempo que no afecta la energía del modelo de Grúa Torre y tiene su propio modelo dinámico que se detallará posteriormente.

Las fuerzas generalizadas correspondientes al vector generalizado de la ecuación 3.7 son:

$$\vec{F} = \{ F_x, 0, T\varphi, 0 \} \quad (3.8)$$

donde F_x es la fuerza que produce el movimiento del carro y $T\varphi$ es el torque que produce la rotación de la pluma. Las coordenadas generalizadas $\phi(t)$, $\theta(t)$, son consecuencia de la aplicación de F_x y $T\varphi$ en la grúa, por lo que se les asocian fuerzas generalizadas iguales a 0.

Además, se deben considerar dos amortiguadores viscosos que representen las fricciones existentes en el modelo de la Grúa Torre (b_1, b_2). Por lo cual, la función de disipación de Rayleigh tiene la forma siguiente:

$$D = \frac{1}{2}(b_1 \dot{r} + b_2 \dot{\phi}) \quad (3.9)$$

El Lagrangiano L de un sistema dinámico se define como la diferencia de las energías cinética y potencial:

$$L = T - U \quad (3.10)$$

Las ecuaciones de movimiento del sistema de Grúa Torre se hallan de la ecuación de Lagrange para fuerzas generalizadas que considera la disipación de energía :

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}^i} \right) - \frac{\partial L}{\partial q^i} + \frac{\partial D}{\partial \dot{q}^i} = F_i \quad (i = 1, 2, \dots, n) \quad (3.11)$$

La ecuación 3.11 es una ecuación general para un sistema de “n” coordenadas generalizadas y “n” grados de libertad.

Resolviendo la ecuación 3.11 para cada una de las coordenadas generalizadas de la expresión 3.7, las fuerzas generalizadas de la expresión 3.8 y la función de disipación 3.9, se obtienen las siguientes ecuaciones de movimiento:

$$\begin{aligned} F_x = & m\ddot{r} - 2m\dot{L}(t)\cos\theta(t)\cos\phi(t)\dot{\phi}(t) + 2mL(t)\sin\theta(t)\cos\phi(t)\dot{\theta}(t)\dot{\phi}(t) + \\ & mL(t)\cos\theta(t)\sin\phi(t)\dot{\phi}^2(t) - mL(t)\cos\theta(t)\cos\phi(t)\ddot{\phi}(t) + M\ddot{r}(t) + 2m\dot{L}(t)\sin\theta(t)\sin\phi(t)\dot{\theta}(t) + \\ & mL(t)\cos\theta(t)\sin\phi(t)\dot{\theta}^2(t) - m\dot{\phi}^2(t)r(t) + mL(t)\sin\theta(t)\sin\phi(t)\ddot{\theta}(t) - m\ddot{L}(t)\cos\theta(t)\sin\phi(t) - \\ & mL(t)\sin\theta(t)\ddot{\phi}(t) - M\dot{\phi}^2(t)r(t) - 2m\dot{L}(t)\sin\theta(t)\dot{\phi}(t) - 2mL(t)\cos\theta(t)\dot{\phi}(t)\dot{\theta}(t) + \\ & m\cos\theta(t)\sin\phi(t)L(t)\dot{\phi}^2(t) + b_1\dot{r}(t) \end{aligned} \quad (3.12)$$

$$\begin{aligned} 0 = & L(t)\cos^2\theta(t)\ddot{\phi}(t) + g\cos\theta(t)\sin\phi(t) - L(t)\cos^2\theta(t)\cos\phi(t)\sin\phi(t)\dot{\phi}^2(t) + \\ & \cos\phi(t)\cos\theta(t)r(t)\dot{\phi}^2(t) - \cos\theta(t)\cos\phi(t)\ddot{r}(t) - 2L(t)\sin\theta(t)\cos\theta(t)\dot{\phi}(t)\dot{\theta}(t) + \\ & 2L'(t)\sin\theta(t)\cos\theta(t)\cos\phi(t)\dot{\phi}(t) + 2L'(t)\cos^2\theta(t)\dot{\phi}(t) + L(t)\cos\theta(t)\cos\phi(t)\sin\theta(t)\ddot{\theta}(t) + \\ & 2L(t)\cos^2\theta(t)\cos\phi(t)\dot{\theta}(t)\dot{\phi}(t) \end{aligned} \quad (3.13)$$

$$\begin{aligned}
T_{\varphi} = & -mL(t) \text{Sen}\theta(t) r''(t) + 2m \text{Cos}\theta(t) L(t) L'(t) \text{Sen}\theta(t) \text{Cos}\phi(t) \phi'(t) - \\
& 2mL^2(t) \text{Sen}^2\theta(t) \text{Cos}\phi(t) \phi'(t) \theta'(t) - m \text{Cos}\theta(t) \text{Sen}\phi(t) L^2(t) \text{Sen}\theta(t) \phi''(t) + \\
& m \text{Cos}\theta(t) L^2(t) \text{Sen}\theta(t) \text{Cos}\phi(t) \phi''(t) - 2m L(t) L'(t) \text{Sen}\phi(t) \theta'(t) - mL^2(t) \text{Sen}\phi(t) \theta''(t) + \\
& m \text{Sen}^2\theta(t) L^2(t) \varphi''(t) + 2m \varphi'(t) L(t) L'(t) \text{Sen}^2\theta(t) + mL''(t) \text{Sen}\theta(t) r(t) + \\
& 2m \text{Cos}\theta(t) L'(t) r(t) \theta'(t) - m \text{Sen}\theta(t) r(t) L(t) \theta''(t) + m \text{Cos}\theta(t) r(t) L(t) \theta''(t) + m \varphi''(t) r^2(t) + \\
& 2m \varphi'(t) r(t) r'(t) - 2m \varphi'(t) r'(t) L(t) \text{Cos}\theta(t) \text{Sen}\phi(t) - 2m \varphi''(t) r(t) L(t) \text{Cos}\theta(t) \text{Sen}\phi(t) - \\
& 2m \varphi'(t) r(t) L'(t) \text{Cos}\theta(t) \text{Sen}\phi(t) + 2m \varphi'(t) r(t) L(t) \text{Sen}\theta(t) \text{Sen}\phi(t) \theta'(t) - \\
& 2m \varphi'(t) r(t) L(t) \text{Cos}\theta(t) \text{Cos}\phi(t) \phi'(t) + m \varphi''(t) L^2(t) \text{Cos}^2\theta(t) \text{Sen}^2\phi(t) + \\
& 2m \varphi'(t) L(t) L'(t) \text{Cos}^2\theta(t) \text{Sen}^2\phi(t) + 2M r(t) r'(t) \varphi'(t) + Mr^2(t) \varphi''(t) + J_0 \varphi''(t) + \\
& mL^2(t) \varphi'(t) \text{Cos}^2\phi(t) \text{Sen}2\theta(t) \theta'(t) + mL^2(t) \varphi'(t) \text{Cos}^2\theta(t) \text{Sen}2\phi(t) \phi'(t) + b_2 \varphi'(t) \quad (3.14)
\end{aligned}$$

$$\begin{aligned}
0 = & r''(t) \text{Sen}\theta(t) \text{Sen}\phi(t) + 2 L'(t) \theta'(t) + L(t) \theta''(t) - 2L'(t) \text{Sen}\phi(t) \varphi'(t) - L(t) \text{Sen}\phi(t) \varphi''(t) - \\
& L(t) \text{Cos}\phi(t) \varphi'(t) \phi'(t) + r(t) \text{Cos}\theta(t) \varphi''(t) + 2r'(t) \text{Cos}\theta(t) \varphi'(t) + L(t) \text{Cos}\theta(t) \text{Sen}\theta(t) \phi'^2(t) + \\
& L(t) \text{Cos}\phi(t) \text{Sen}^2\theta(t) \varphi'(t) \phi'(t) - L(t) \text{Cos}^2\theta(t) \text{Cos}\phi(t) \varphi'(t) \phi'(t) - L(t) \text{Cos}\theta(t) \text{Sen}\theta(t) \varphi'^2(t) - \\
& r(t) \text{Sen}\theta(t) \text{Sen}\phi(t) \varphi'^2(t) + L(t) \text{Cos}\theta(t) \text{Sen}\theta(t) \text{Sen}^2\phi(t) \varphi'^2(t) + g \text{Cos}\phi(t) \text{Sen}\theta(t) \quad (3.15)
\end{aligned}$$

Como fue explicado anteriormente, el sistema de elevación de la Grúa Torre representa un sistema independiente con respecto al movimiento del carro, de la pluma y del segmento rotatorio de la torre, y puede ser modelado como se muestra en la figura 3.9:

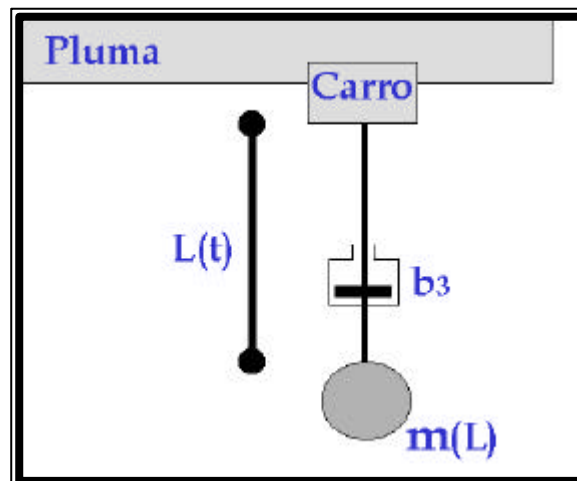


Figura 3.9 – Modelo físico del sistema de elevación de la Grúa Torre

En este modelo se considera el peso longitudinal del cable, por tanto, la Grúa Torre debe transportar la suma del peso de la carga y el peso del cable, luego:

$$m(L) = (m_c + K_{mas}L(t)) \quad (3.16)$$

Donde m_c es la masa de la carga que se debe transportar, K_{mas} es la densidad de peso longitudinal del cable en Kg/m, y $L(t)$ es la longitud del cable de la Grúa.

La energía cinética del sistema es:

$$T = \frac{1}{2}m(L) \dot{L}^2(t) \quad (3.17)$$

La energía potencial del sistema es:

$$U = -m(t) g L(t) \quad (3.18)$$

La función de densidad de Rayleigh, correspondiente al coeficiente de viscosidad b_3 es:

$$D = \frac{1}{2}b_3 \dot{L}^2(t) \quad (3.19)$$

Resolviendo la ecuación 3.11 para la única coordenada generalizada $L(t)$, se obtiene la siguiente ecuación de movimiento:

$$F_1 = K_{mas}\ddot{L}(t) + m\ddot{L}(t) - m(t)g + b_3\dot{L}(t) \quad (3.20)$$

Las ecuaciones (3.12), (3.13), (3.14), (3.15) y (3.20) expresan la posibilidad de realizar el control de los movimientos de la Grúa Torre utilizando la fuerza F_x , el torque T_ϕ y la fuerza F_1 . Sin embargo, un operador controla una grúa real a través de cambios de velocidad (aceleraciones), por lo que es adecuado incluir en el modelo un compensador de masa que permita el control indirecto de las fuerzas involucradas a través de aceleraciones. Para ello, considerando que los ángulos de oscilación son pequeños, que las tasas de cambio de $L(t)$, $r(t)$ y $\phi(t)$ son del mismo orden de magnitud que las tasas de cambio de $\phi(t)$ y $\theta(t)$, y despreciando además las variaciones en la longitud del cable y las no-linealidades, las ecuaciones (3.12), (3.13), (3.14) y (3.15) pueden ser expresadas como:

$$F_x = Mr''(t) + mg\phi(t) \quad (3.21)$$

$$0 = L(t)\phi''(t) + g\phi(t) - r''(t) \quad (3.22)$$

$$T_\gamma = (Mr^2 + J_0)\gamma''(t) + mg\theta(t)r(t) \quad (3.23)$$

$$0 = L(t)\theta''(t) + g\theta(t) + r(t)\gamma''(t) \quad (3.24)$$

Las ecuaciones (3.21), (3.22), (3.23) y (3.24) permiten la implementación del compensador de masa que se muestra enmarcado en la figura 3.10, el bloque Grúa Torre contiene las ecuaciones (3.12), (3.13), (3.14) y (3.15) inicialmente halladas, se puede observar que la fuerza F_x y el torque T_γ son funciones sencillas de las aceleraciones del carro y de la pluma generadas manualmente o por los controladores Neuro-difusos del sistema de automatización. Como para el desarrollo de esta tesis se considera que la longitud del cable se mantiene constante durante los movimientos rotacional y radial de la Grúa Torre, la ecuación 3.20, que modela el sistema de elevación de la grúa, no ha tenido que ser modificada.

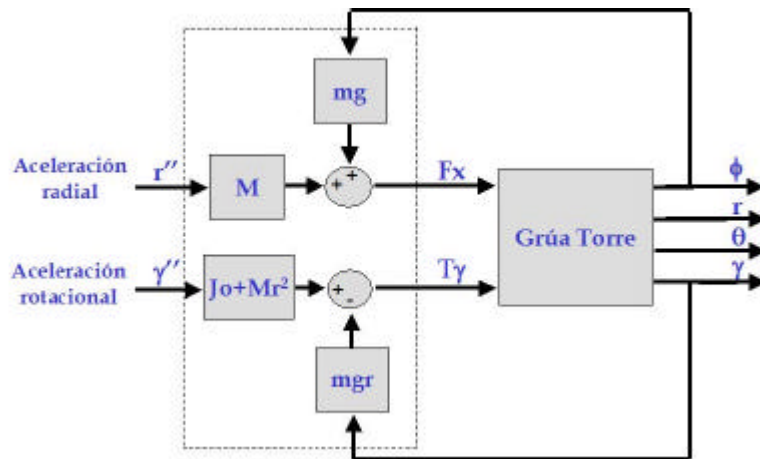


Figura 3.10 - Modelo de compensación para el control por aceleración

3.2 Diseño del Simulador 3D de Grúa Torre

Basado en el análisis realizado en el ítem 3.1, en este apartado se describe el desarrollo e implementación del simulador de Grúa Torre 3D que permitirá extraer datos de

entrenamiento para el sistema de automatización Neuro-difuso. El proceso se ha realizado en 4 pasos:

- 1) Modelado matemático de la grúa
- 2) Implementación del modelo en Simulink de Matlab
- 3) Animación
- 4) Diseño de la Interfaz Gráfica del Usuario (GUI)

El paso 1 corresponde al ítem 3.1 de este capítulo.

3.2.1 Implementación del modelo en Simulink

Simulink de MATLAB es uno de los paquetes de software más ampliamente usados en el mundo académico y en la industria para modelar y simular sistemas dinámicos.

Los modelos se implementan a partir de bloques unidos por flechas que indican el flujo de la información. Cada bloque tiene una función en el sistema, como representar una ecuación, generar señales, o permitir la visualización de los resultados. La siguiente figura muestra el modelo de Grúa Torre de las ecuaciones 3.12, 3.13, 3.14, 3.15 y 3.20 implementado en Simulink.

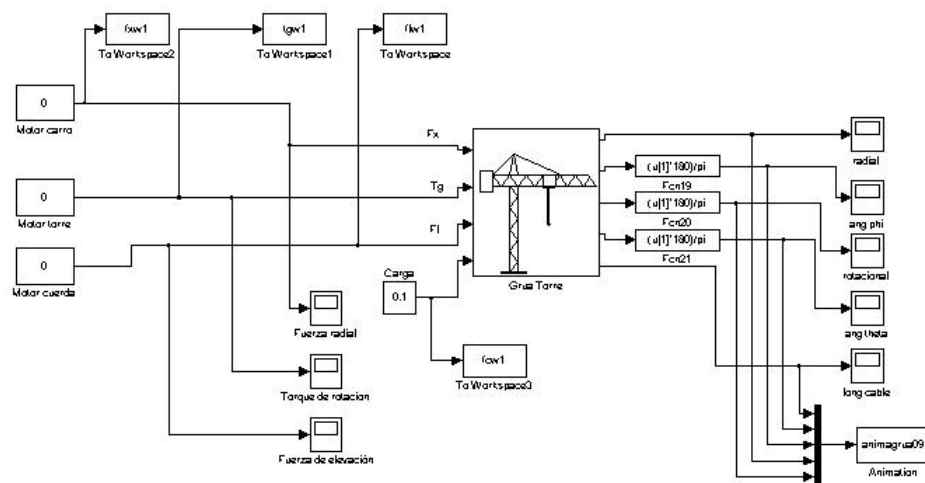


Figura 3.11 – Implementación del modelo de Grúa Torre en Simulink

La simulación de modelos en Simulink implica la integración numérica de conjuntos de ecuaciones diferenciales ordinarias. Los métodos para resolver las ecuaciones diferenciales en Simulink, pueden ocasionar resultados diferentes unos de otros, ya que debido a la diversidad de conductas de los sistemas dinámicos, ningún método simula todos los tipos de modelos de forma precisa y eficiente; por esta razón, la elección del método apropiado y la selección cuidadosa de los parámetros de simulación son consideraciones importantes cuando se desea obtener resultados rápidos y precisos. Por esta razón, luego de una revisión heurística de los demos de sistemas masa-péndulo incluidos en Matlab, se escogió el método ode 45 (Dormand-Prince) para la simulación de Grúa Torre de la presente tesis, pues es el más utilizado en este tipo de sistemas. Una explicación matemática detallada de este método de integración se puede encontrar en el manual de Simulink.

Los parámetros especificados para realizar la simulación fueron los siguientes:

- Tiempo de comienzo y finalización, 0.0 s – 10 000 s
- Paso inicial, auto
- Tamaño del paso máximo, 0.1
- Tolerancia o error relativo, 1e-6

Estos parámetros fueron determinados después de un proceso iterativo de ajuste de los parámetros de la simulación hasta obtener resultados adecuados.

3.2.2 Animación

La animación es el arte de mover imágenes estáticas de tal forma que el observador perciba un movimiento continuo. A través de la historia de la animación, los artistas han creado las imágenes a mano con pequeñas modificaciones unas con respecto a otras de tal forma que mostrándolas en una sucesión rápida han creado la ilusión de movimiento. El uso de computadoras ha simplificado enormemente la tarea de manejar el gran número de imágenes necesarias y han hecho muy sencilla su creación; además, se ha permitido crear imágenes a partir de ecuaciones y utilizar esas reglas matemáticas para moverlas en la pantalla. Así, una animación de la Grúa Torre

siguiendo este método hace posible que el comportamiento físico del sistema pueda ser observado y analizado dentro de un entorno virtual; la animación de la Grúa Torre consiste de un conjunto de objetos descritos como formas sencillas (rectángulos, círculos, etc.), una colección de datos de los puntos que constituyen una imagen, y un conjunto de instrucciones computacionales que describen como cambia cada uno de estos objetos en el tiempo; se pueden incluir cambios en la locación, orientación, geometría, o incluso color.

3.2.2.1 Elementos Geométricos

Como fue descrito en el ítem 3.1, los elementos geométricos básicos de una Grúa Torre son:

- La torre
- La pluma
- El contrapeso
- El carro
- El cable

Para poder simular el proceso de transporte de cargas, se le han agregado una carga y una marca que señale el punto de destino. Las siguientes figuras presentan individualmente cada uno de los elementos que conforman la Grúa Torre según un sistema tridimensional de ejes coordenados. Las diferentes vistas de la torre de la Grúa se muestran en la figura 3.12; así, la figura (a) presenta una vista frontal, la figura (b) presenta una vista lateral y la figura (c) presenta una vista desde un plano superior.

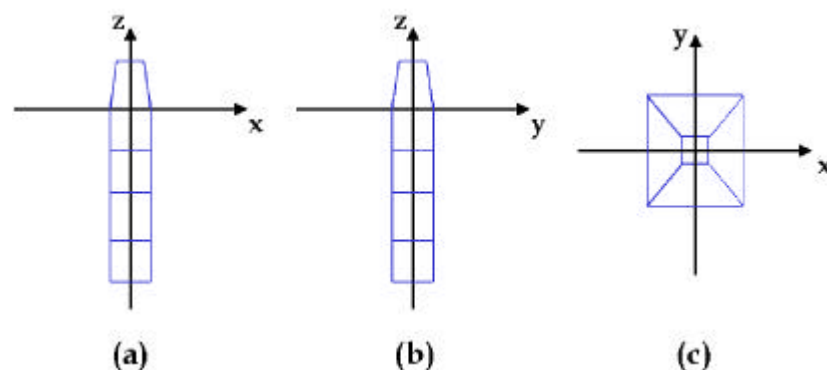


Figura 3.12 – Vistas de la torre de la grúa

Las diferentes vistas de la pluma y el contrapeso de la grúa se muestran en la figura 3.13; así, la figura (a) presenta una vista frontal, la figura (b) presenta una vista lateral y la figura (c) presenta una vista desde un plano superior.

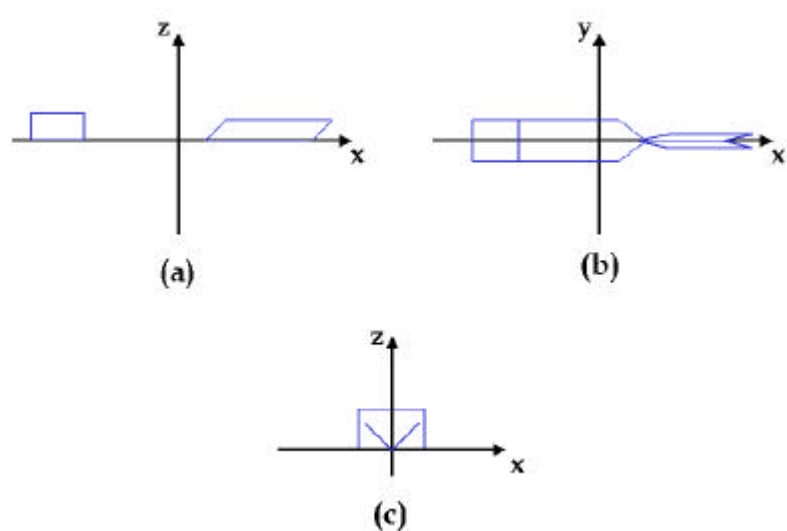


Figura 3.13 – Vistas de la pluma y el contrapeso de la grúa.

Las diferentes vistas del carro y de la cuerda de la grúa se muestran en la figura 3.14; así, la figura (a) presenta una vista frontal, la figura (b) presenta una vista lateral y la figura (c) presenta una vista desde un plano superior.

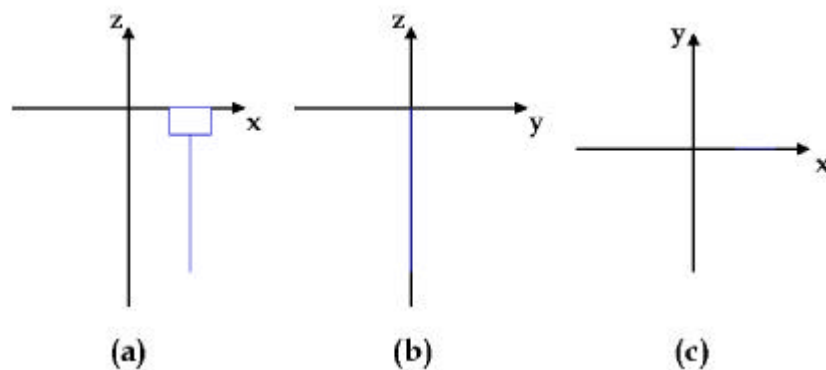


Figura 3.14 – Vistas del carro y de la cuerda de la grúa.

Las diferentes vistas del peso que la grúa debe transportar se muestran en la figura 3.15; así, la figura (a) presenta una vista frontal, la figura (b) presenta una vista lateral y la figura (c) presenta una vista desde un plano superior.

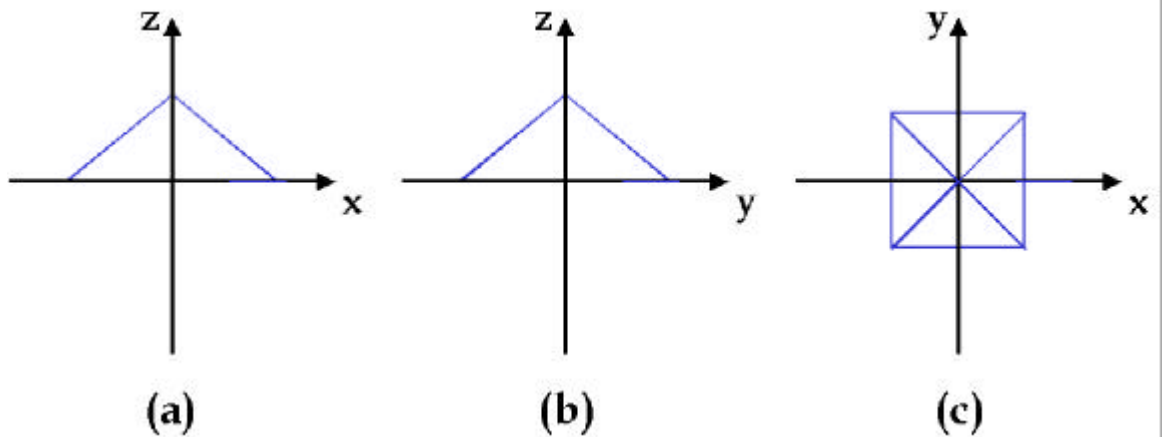


Figura 3.15 – Vistas de la carga de la grúa.

Las diferentes vistas del señalador de destino para el transporte de la carga se muestran en la figura 3.16; así, la figura (a) presenta una vista frontal, la figura (b) presenta una vista lateral y la figura (c) presenta una vista desde un plano superior.

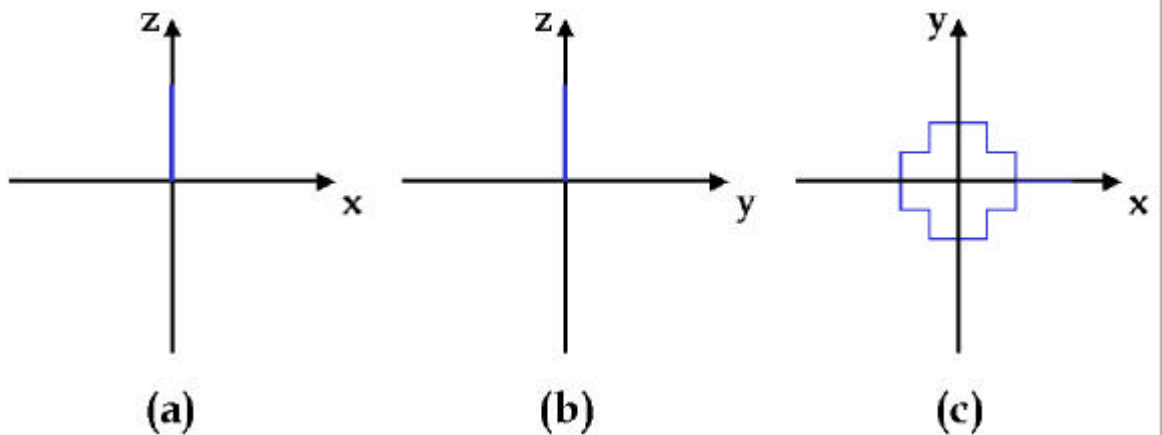


Figura 3.16 – Vistas del señalador de destino

Cada imagen de la animación de la Grúa Torre contiene todos los elementos descritos líneas arriba como se muestra en la figura 3.17; los puntos de inicio y final para el transporte de la carga son escogidos por el usuario a través del ingreso de las coordenadas de inicio y final de transporte de la carga correspondientes en el código del programa.

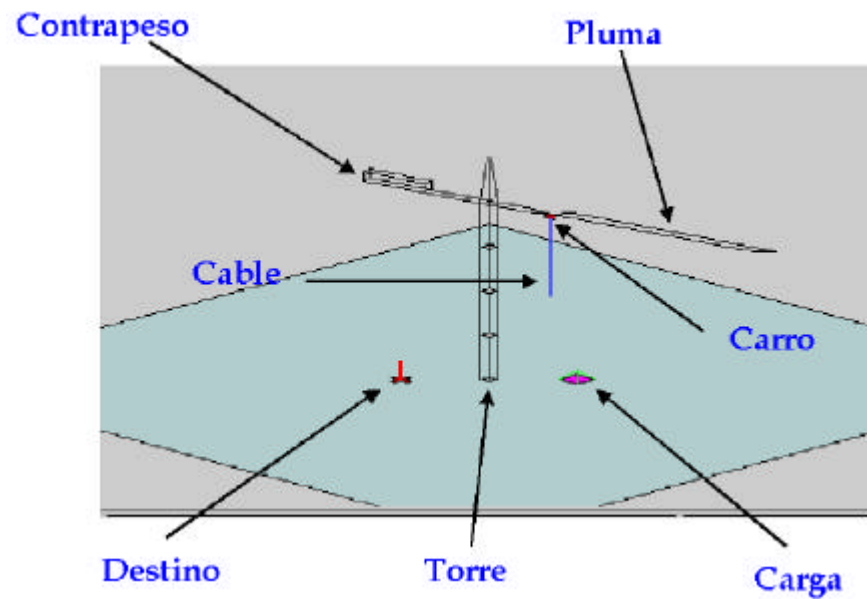


Figura 3.17 – Animación del Simulador 3D de la Grúa Torre

Para dibujar en Matlab cada una de las figuras que conforman una imagen, se necesita que los datos de la ubicación de los puntos de cada figura geométrica estén agrupados en matrices. Por ejemplo, si se desea dibujar el cuadrado de la figura 3.18 se deben considerar los vértices del mismo como los datos que determinan su ubicación. Con estos datos se forman dos vectores con sus elementos ordenados como si se estuviese dibujando el cuadrado con un lápiz, así:

$$x = [a, c, e, g, a];$$

$$y = [b, d, f, h, b];$$

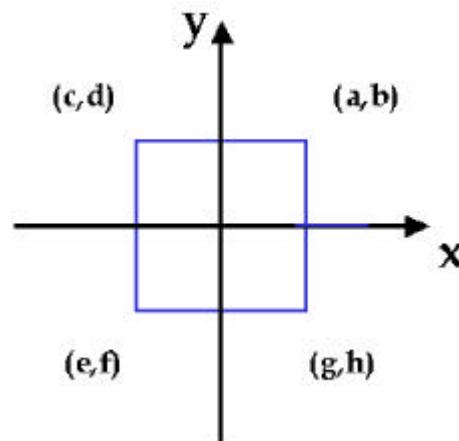


Figura 3.18 – Ilustración del dibujo de formas geométricas en MATLAB

El cuadrado es dibujado cuando se ingresa la siguiente orden:

```
cuadrado = line(x,y,'color',[0 0 0]);
```

la orden line indica que se debe trazar una línea que pase por los puntos (x,y) y que esta línea sea de color negro ([0, 0, 0]).

En el caso de la torre de la grúa, en lugar de literales como a,b,c, etc., se utilizaron las siguientes variables para describir sus dimensiones:

- Grua.torre.ancho
- Grua.torre.fondo
- Grua.torre.alto

Se debe tener en cuenta que la torre de la grúa es una reunión de formas geométricas en sí misma, por lo que debe ser dibujada por partes, utilizando los descriptores mencionados, así:

Para la parte inferior de la torre:

```
x=[-grua.torre.fondo/2    -grua.torre.fondo/2    grua.torre.fondo/2    grua.torre.fondo/2    -
grua.torre.fondo/2];
```

```
y = [-grua.torre.ancho/2 grua.torre.ancho/2 grua.torre.ancho/2 -grua.torre.ancho/2 -
grua.torre.ancho/2];
```

```
z = [-0.8*grua.torre.alto -0.8*grua.torre.alto -0.8*grua.torre.alto -0.8*grua.torre.alto -
0.8*grua.torre.alto];
```

```
hti = line(x,y,z,'color',[0 0 0],'erasemode','none');
hold on;
```

Para la parte superior de la torre:

```
hts = line(x/4,y/4,z+grua.torre.alto,'color',[0 0 0],'erasemode','none');
```

Para el lado lateral izquierdo de la torre:

```

x = [-grua.torre.fondo/2 grua.torre.fondo/2 grua.torre.fondo/2 grua.torre.fondo/8 -
grua.torre.fondo/8];

y = [-grua.torre.ancho/2 -grua.torre.ancho/2 -grua.torre.ancho/2 -grua.torre.ancho/8 -
grua.torre.ancho/8];

z = [-0.8*grua.torre.alto -0.8*grua.torre.alto 0 -0.8*grua.torre.alto+grua.torre.alto-
0.8*grua.torre.alto+grua.torre.alto];

htli = line(x,y,z,'color',[0 0 0],'erasemode','none');

```

Para el lado lateral derecho de la torre:

```

htld = line(-x,-y,z,'color',[0 0 0],'erasemode','none');

```

Para la parte frontal de la torre:

```

x = [-grua.torre.fondo/2 -grua.torre.fondo/2 -grua.torre.fondo/2 -grua.torre.fondo/8 -
grua.torre.fondo/8];

y = [grua.torre.ancho/2 -grua.torre.ancho/2 -grua.torre.ancho/2 -grua.torre.ancho/8
grua.torre.ancho/8];

z = [-0.8*grua.torre.alto -0.8*grua.torre.alto 0 -0.8*grua.torre.alto+grua.torre.alto -
0.8*grua.torre.alto+grua.torre.alto];

htf = line(x,y,z,'color',[0 0 0],'erasemode','none');

```

Para la parte trasera:

```

htt = line(-x,-y,z,'color',[0 0 0],'erasemode','none');

```

Las variables hti, hts, htli, htld, htf y htt son los “handlers” de la torre y determinan su posición en cada instante de tiempo.

Se repite el mismo procedimiento con cada uno de los elementos de la Grúa Torre; así, toda la información acerca de la posición de la Grúa Torre se encuentra contenida en estructuras matriciales de puntos. Si se desea modificar la posición de los elementos de la Grúa Torre, se deben actualizar los datos almacenados en los “handlers” de cada uno de los elementos que conforman la grúa, de acuerdo al progreso de la simulación.

3.2.3 Interfaz Gráfica de Usuario

El bloque de animación, contiene una función que recibe como parámetros las variaciones en las posiciones del carro, pluma, y la cuerda, así como la variable del tiempo de la simulación, el paso del método de integración, así como otras variables relacionadas a la presencia de una nueva ventana de animación. Esta relación entre la simulación en ejecución y la animación permite el diseño de una interfaz gráfica de usuario. El GUI (*Graphical User Interface*) o interfaz gráfica de usuario contiene los controles que el usuario utiliza para determinar el funcionamiento de la animación y por extensión, del simulador. La figura 3.19 muestra los botones que conforman la interfaz gráfica de usuario del simulador de Grúa Torre, el botón de control manual permite la operación manual del simulador y el botón de control automático permite el control automático del simulador a través de un sistema de automatización.

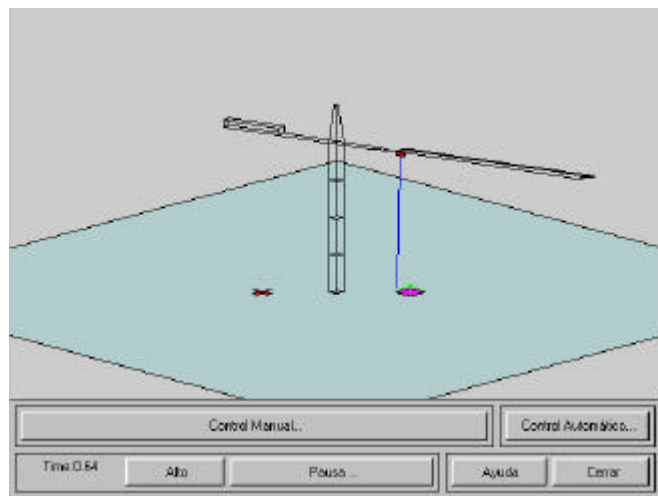


Figura 3.19 – Interfaz Gráfica de usuario del simulador 3D

3.3 Validación del Simulador 3D

Este ítem presenta la validación del simulador de la tesis, para lo cual se aplicaron sobre él algoritmos de control desarrollados por los investigadores Hanafy y Al-Mousa (*Virginia Tech - USA*). La validez del modelo será comprobada al comparar las curvas obtenidas con el simulador y las que publicaron los investigadores mencionados en sus volúmenes de tesis. Contar con un simulador validado permitirá tener la certeza de obtener resultados confiables cuando se aplique sobre él el algoritmo neuro-difuso.

Como se muestra en la figura 3.20, el trabajo desarrollado por Hanafy sigue dos enfoques. El primero, consiste en un Control de Realimentación de Estados (*Gain Scheduling Feedback Controller*), similar a un controlador proporcional clásico, en el cual el controlador de posición y el de oscilación son considerados de forma unificada. En el segundo enfoque, el proceso de transporte de la carga y el control de las oscilaciones son tratados de forma separada, lo cual requiere que se diseñen dos controladores independientes: un controlador anti-oscilación y un controlador de posición. Para controlar la posición se utiliza un controlador PD, mientras que el controlador anti-oscilación fue diseñado usando dos métodos diferentes: (a) un controlador basado en una técnica de realimentación retrasada, y (b) un controlador difuso creado a partir del modelado del controlador de realimentación retrasada.

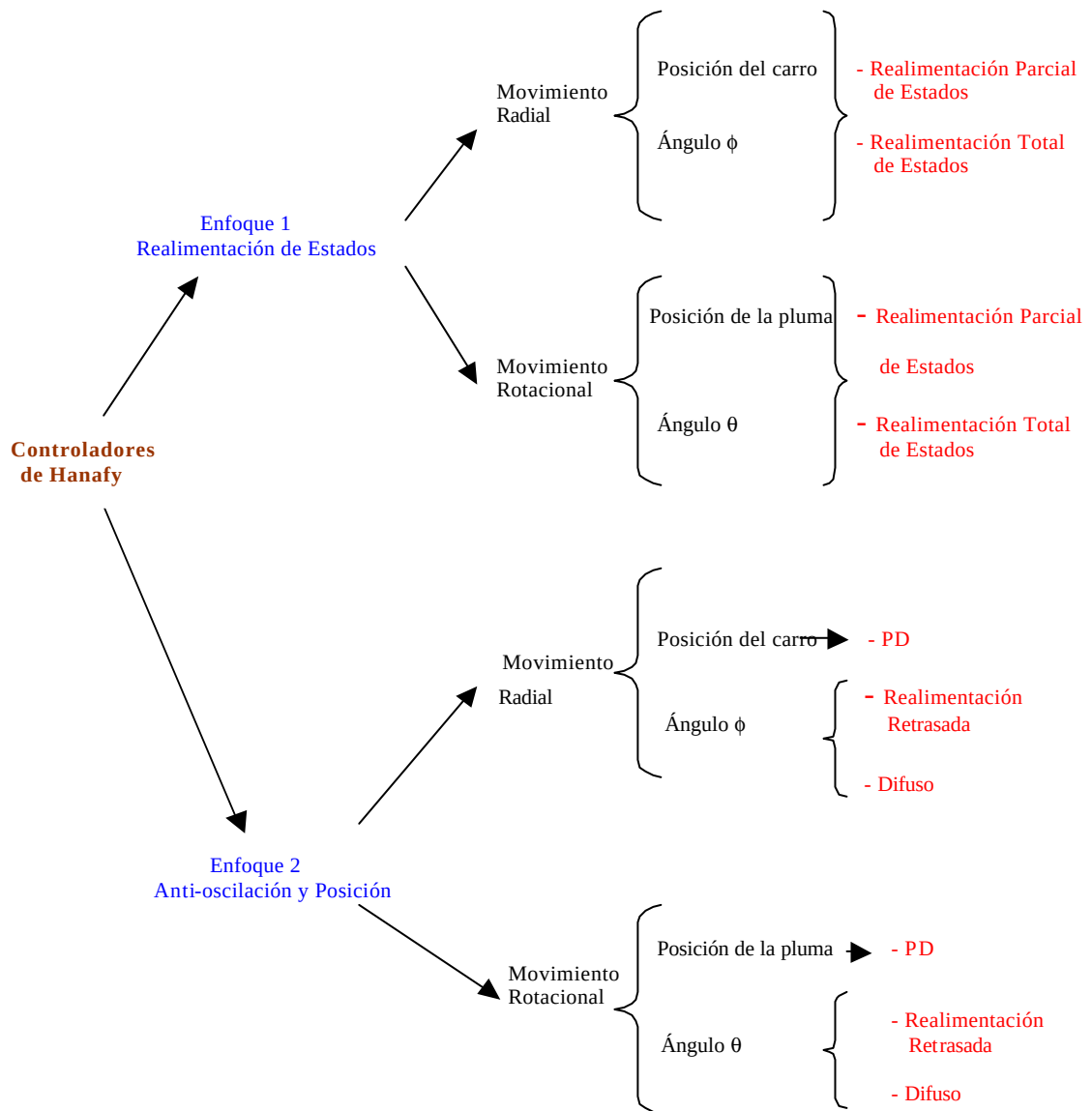


Figura 3.20 – Controladores de Hanafy

De otro lado, como muestra la figura 3.21, los controladores de Al-Mousa también fueron diseñados a partir de dos enfoque distintos, en el primer método Al-Mousa utilizó un controlador difuso creado a partir del modelado de un controlador PD, y considera diferentes controladores para el control de la posición y la oscilación. En el segundo enfoque utiliza un controlador de realimentación retrasada, que considera el control de la posición y de la oscilación de forma unificada.

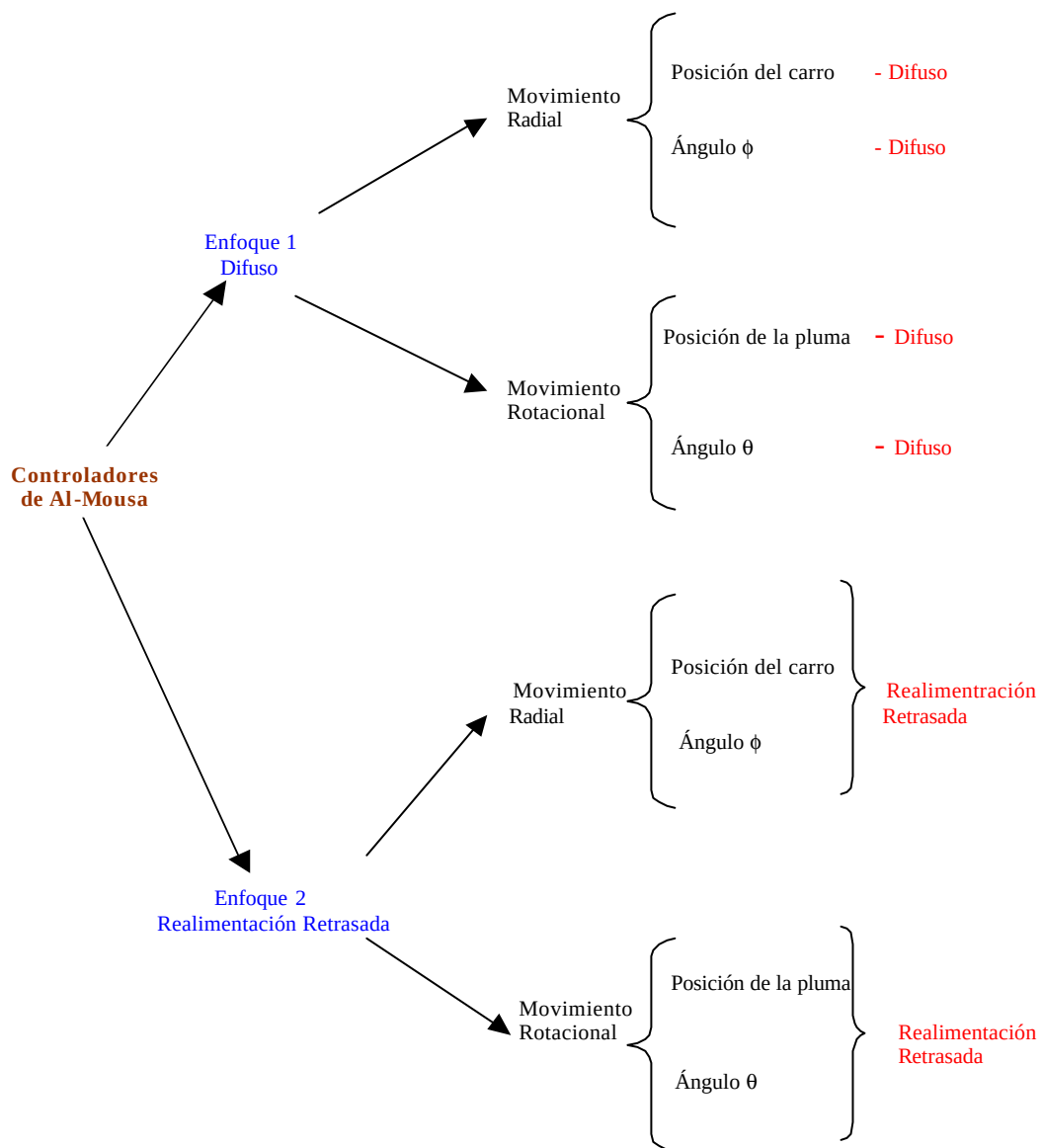


Figura 3.21 – Controladores de Al-Mousa

3.3.1 Algoritmos de Control desarrollados por Hanafy M. Omar

En el diseño de sus algoritmos de control, Hanafy consideró que la carga que transporta la Grúa Torre es siempre constante y que el cable del que pende la carga tiene un peso despreciable y longitud constante. Cabe destacar, que la tesis de Hanafy se orienta principalmente al estudio de sistemas de automatización para grúas puente, y a partir de ellos realiza extensiones y generalizaciones que permiten que estos puedan ser usados para controlar los movimientos rotacional y radial de una Grúa Torre.

3.3.1.1 Control de Realimentación de Ganancias (*Gain Scheduling Adaptive Feedback Controller*)

Esta técnica consiste en una realimentación ponderada de las variables de estado de la grúa utilizando constantes de proporcionalidad que dependen de la magnitud de la carga y longitud de la cable. Hanafy diseñó dos tipos de algoritmos de control de acuerdo a si la realimentación de las variables de estado es total o parcial; el diagrama de bloques del controlador de realimentación parcial de ganancias tiene la forma que se muestra en la figura 3.22.

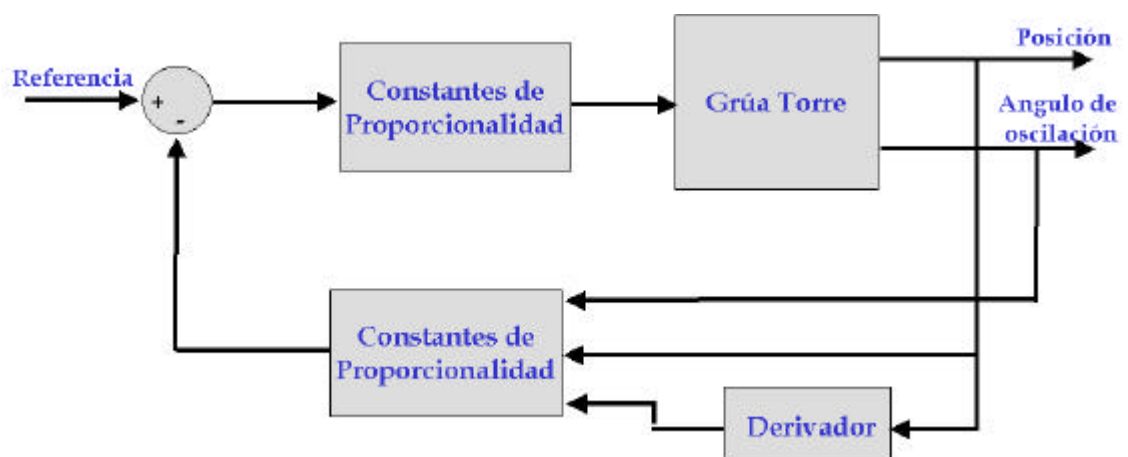
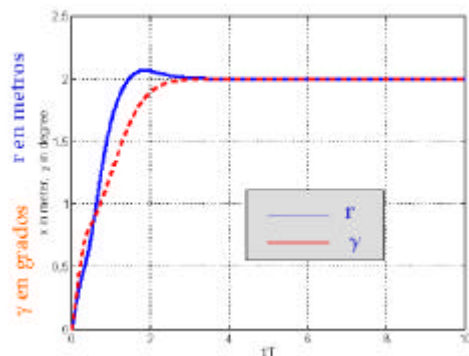
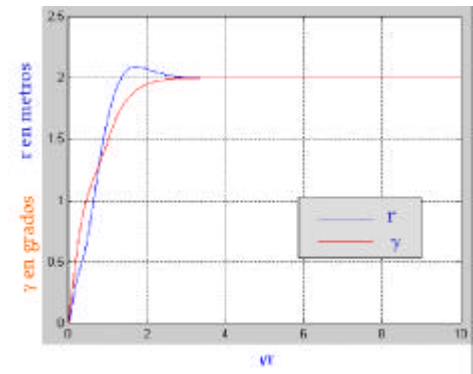


Figura 3.22 – Control de Realimentación Parcial de Hanafy

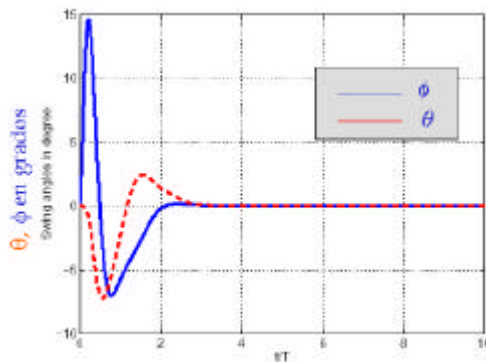
La figura 3.23 presenta las curvas publicadas por Hanafy (figuras 3.23 (a) y 3.23 (c)), y las obtenidas con el simulador 3D (figuras 3.23 (b) y 3.23 (d)).



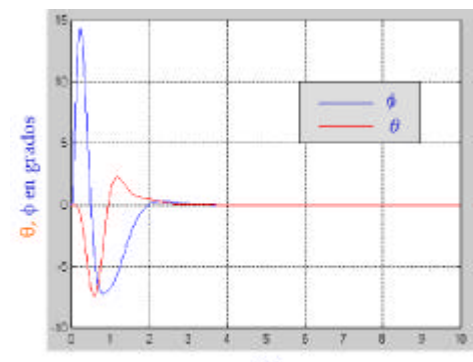
(a)



(b)



(c)



(d)

Figura 3.23 – Comparación de resultados utilizando el controlador de Realimentación Parcial de Hanafy

Como se puede apreciar, las curvas resultantes de la aplicación del controlador de realimentación parcial de estados de Hanafy sobre el simulador 3D, son muy similares a las que publicó este investigador en su tesis.

3.3.1.2 Controlador Dual de Anti-oscilación y Posición (*Anti-Swing Tracking Controller*)

En este caso el posicionamiento del actuador (carro o torre) y la compensación de la oscilación son realizados por controladores diferentes, como muestra la figura 3.24.

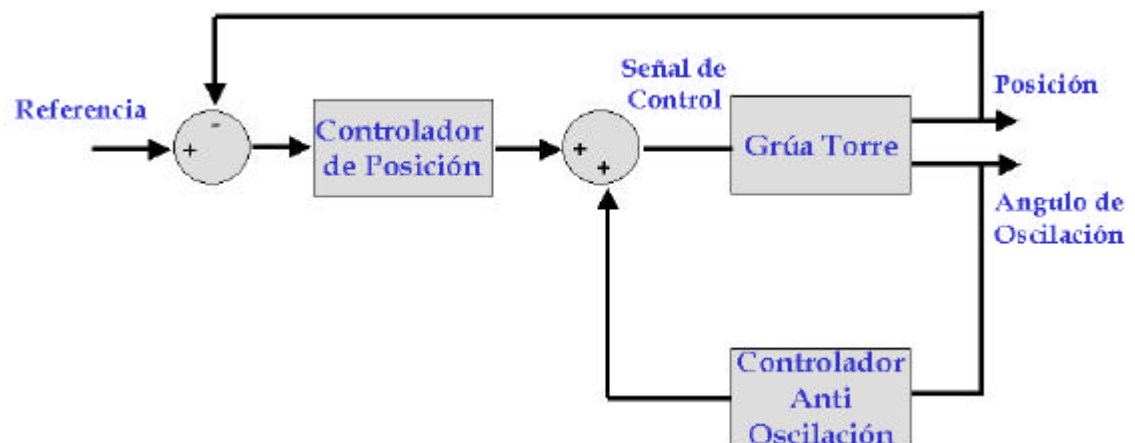
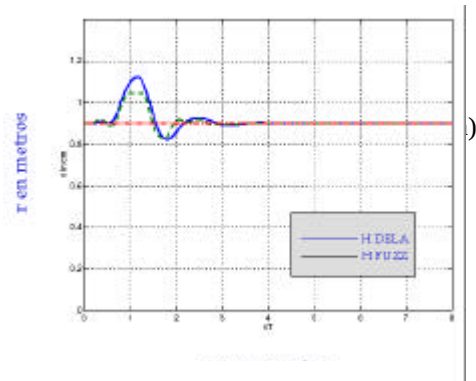


Figura 3.24 – Controlador dual de Hanafy

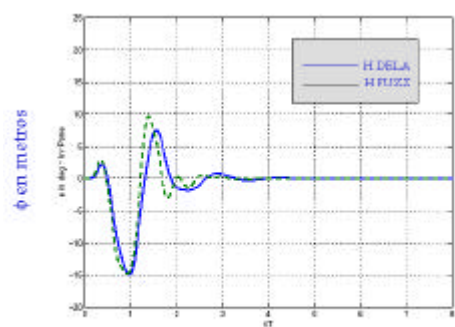
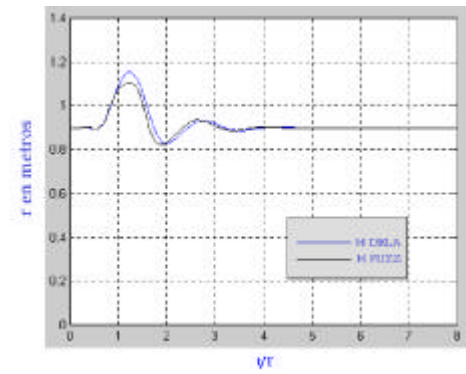
El controlador de posición desarrollado por Hanafy, es un controlador clásico Proporcional Derivativo, y para el controlador Anti-oscilación utiliza dos técnicas: control de retraso de tiempo y control difuso.

La figura 3.25 presenta algunas de las curvas obtenidas cuando se aplican los controladores PD-Delay (H DELA) y PD-difuso (H FUZZ) de Hanafy sobre el simulador 3D, y aquellas que fueron publicadas en el volumen de tesis de este investigador.

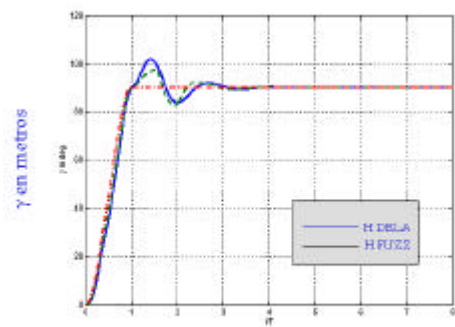
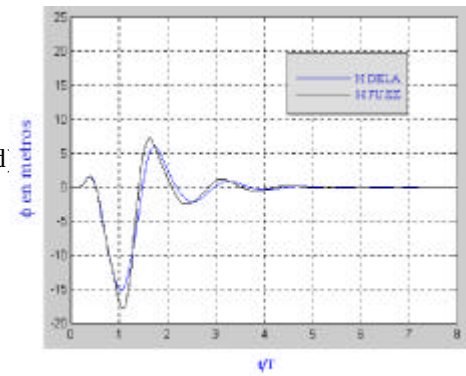
Como se puede apreciar, la similaridad que existe entre los resultados es bastante grande. Por lo que se puede decir que el comportamiento del simulador 3D de Grúa Torre de la presente tesis tiene un comportamiento dinámico similar al del simulador utilizado en *Virginia Tech*.



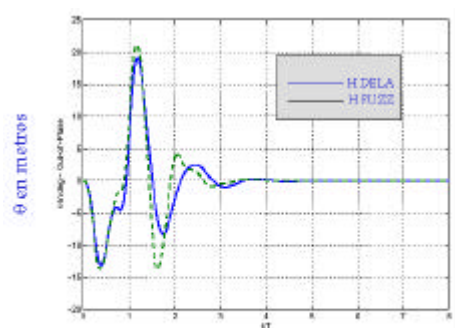
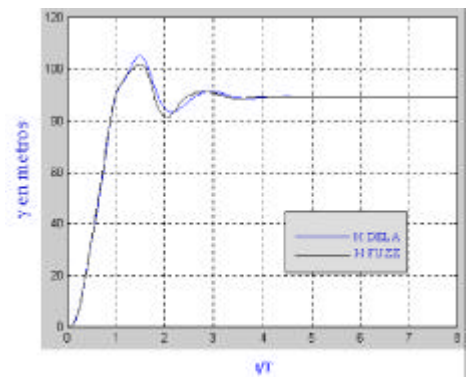
(b)



(d)



(f)



(h)

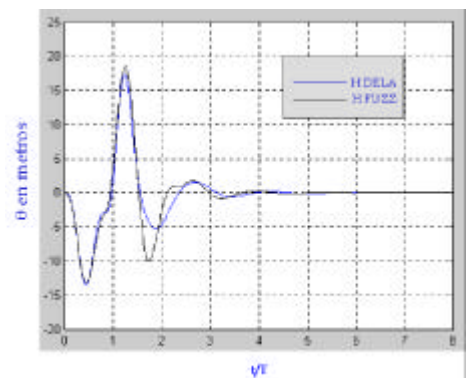


Figura 3.25– Comparación de resultados utilizando el controlador dual de Hanafy

3.3.2 Algoritmos de Control desarrollados por Amjed Al-Mousa

Los algoritmos de control desarrollados por Al-Mousa están basados en un modelo de Grúa Torre que considera que las dimensiones mecánicas de la grúa son lo suficientemente grandes como para despreciar la interacción existente entre la dinámica de la grúa y la de la carga.

Al-Mousa desarrolló dos controladores, el controlador de Realimentación Retrasada (*Delay Feedback Controller*) y el controlador difuso (*Fuzzy Controller*).

El controlador de realimentación retrasada en el tiempo, consiste en la realimentación de los estados de la Grúa retrasados un valor t en el tiempo. El diagrama de bloques de este controlador se presenta en la figura 3.26.

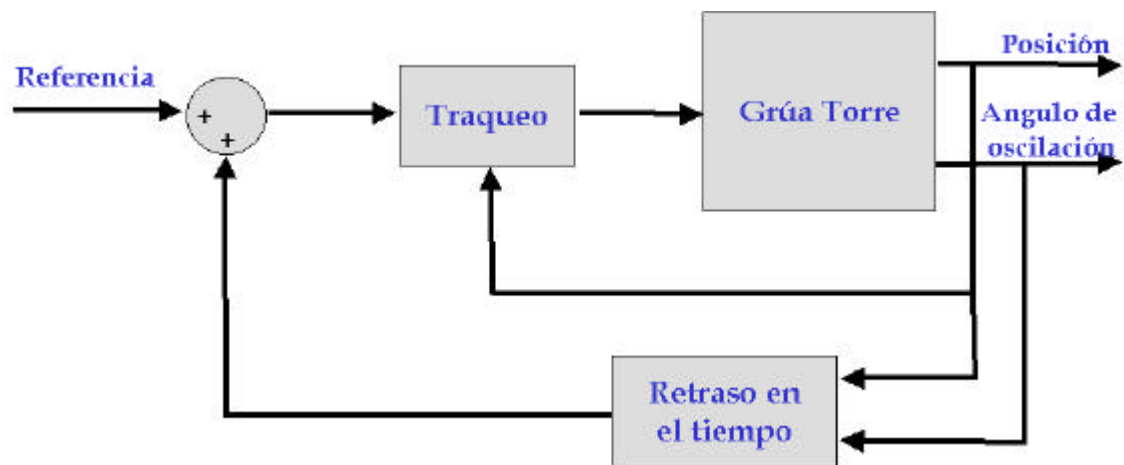


Figura 3.26 – Controlador de Realimentación Retrasada de Al-Mousa

El controlador difuso de Al-Mousa está compuesto por un controlador radial y un controlador rotacional cada uno de los cuales posee dos bloques de reglas difusas, el primero para controlar la posición, y el segundo para controlar la oscilación, ambas respuestas son luego combinadas como una suma ponderada para obtener la señal de control. Las reglas del controladores fueron obtenidas a partir del modelamiento de un controlador PD. La figura 3.27 presenta el diagrama de bloques de este controlador.

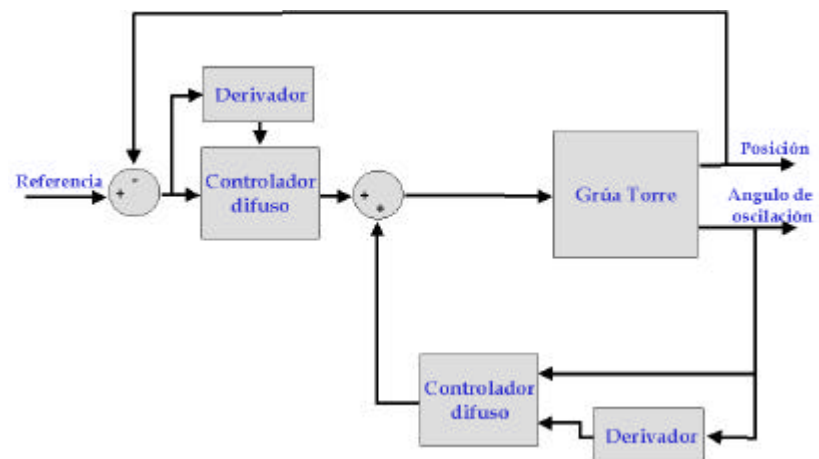
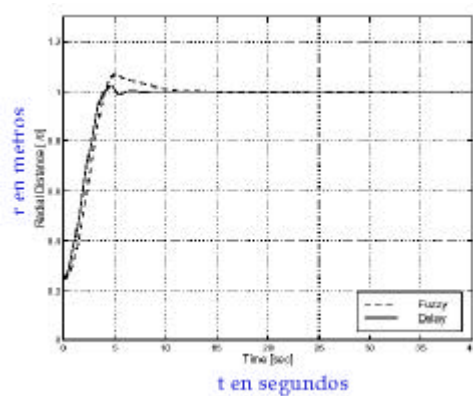
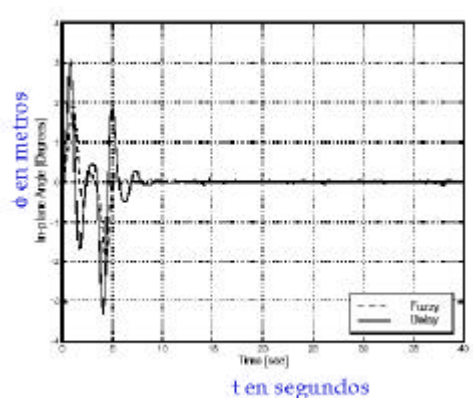
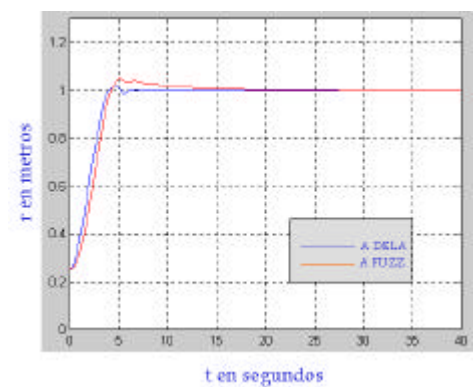


Figura 3.27 – Controlador Difuso de Al-Mousa

La figura 3.28 presenta los resultados comparativos de aplicar los controladores de Realimentación retrasada en el tiempo (A DELA) y difuso (A FUZZ) diseñados por Al-Mousa sobre el simulador 3D, y las que publicó en su volumen de tesis; como se puede apreciar, las respuestas son muy similares.



(b)



(c)

(d)

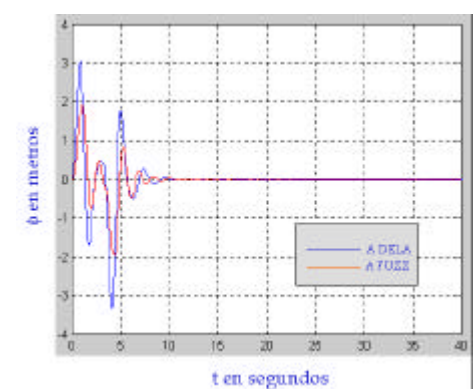


Figura 3.28 – Comparación de resultados utilizando los algoritmos de Al-Mousa

Tal como se puede apreciar en las figuras 3.23, 3.25 y 3.28, los resultados de los algoritmos de control de Hanafy y Al-Mousa en el simulador 3D son casi exactamente iguales a los obtenidos por estos investigadores en sus tesis, las pequeñas diferencias se deben a la falta de información acerca de los parámetros de simulación que utilizaron en sus trabajos de investigación (métodos de integración, tolerancias, etc.); de esta manera, se ha verificado que el simulador es equivalente a los utilizados en la universidad *Virginia Tech* y se hace posible realizar comparaciones entre las respuestas del sistema de automatización Neuro-difuso y los algoritmos de control diseñados por los investigadores mencionados al aplicarlos sobre el simulador 3D.

IV. EXTRACCION DE DATOS EXPERIMENTALES PARA ENTRENAR EL SISTEMA NEURO-DIFUSO

Como se mencionó en la introducción, debido a la imposibilidad de contar con una grúa real o un modelo a escala, la recolección de los datos experimentales se llevó a cabo utilizando el simulador 3D de Grúa Torre, diseñado en el Capítulo III, en su modo de operación manual; es decir, maniobrándolo de la misma forma como lo haría un técnico operador de estas máquinas. Luego de extraer los datos, se seleccionaron los más representativos utilizando el módulo de agrupamiento de datos de fuzzyTECH (*Clustering Module*). Estos datos más representativos serán usados para entrenar el sistema de automatización neuro-difuso.

4.1 Recolección de datos experimentales

La recolección de datos experimentales consistió en el proceso de registrar los valores de posiciones, velocidades y aceleraciones que produjo el simulador mientras era manejado manualmente a través de la interfaz gráfica presentada en el ítem 3.2.3, este proceso se ilustra en la figura 4.1. Para ello, se realizaron movimientos de la grúa con diversos desplazamientos del carro y de la torre, y diferentes condiciones iniciales (ver tabla 4.1). La información recolectada permitió crear una base de datos para representar el desempeño del sistema; esta base de datos fue almacenada en el archivo 'entrenadat.xls', y registró 2814 muestras para el movimiento radial y una cifra similar para el movimiento rotacional; cada muestra es un registro de diez variables 8 de entrada y 2 de salida, como se muestra en la figura 4.2. Las muestras fueron recogidas utilizando un tiempo de muestreo de 0.1 segundos durante la simulación.



Figura 4.1 – Recolección de datos experimentales

Tabla 4.1 – Condiciones iniciales para la recolección de dats de entrenamiento

Parámetro	Condiciones Iniciales
m (Kg)	1
Posición del carro (m)	0, 0.50, 0.75
Posición de la pluma (grados)	0, 30, 60, 90
ϕ (grados)	5, 10, 20
θ (grados)	5, 10, 20

Entradas					Salidas				
r''	γ''	d_r	ϕ	d_y	θ	d'_r	ϕ'	d'_y	θ
Acel. radial	Acel. rotac.	Dist. radial	Áng. radial	Dist. rotac.	Áng. rotac.	Veloc. radial	Veloc. angular radial	Veloc. rotac.	Veloc. angular rotac.

Figura 4.2 – Cabecera del archivo *entrenadat.xls*

4.2 Selección de Datos Experimentales más Representativos

La base de datos creada a partir de la recolección de las muestras experimentales contenía 2814 registros para los movimientos radial y rotacional, entrenar un sistema Neuro-difuso con tal cantidad de datos no resulta práctico. Para reducir la cantidad de datos de entrenamiento, se utilizó el módulo de agrupamiento de datos de fuzzyTECH que permite hallar los datos más representativos del universo de muestras experimentales. Así, el proceso de agrupamiento dió como resultado un total de 112 muestras para el movimiento radial y 51 muestras para el rotacional. Los fundamentos teóricos de este procedimiento, fueron tratados en el ítem 2.3.2.1 del capítulo II.

V. DISEÑO DEL SISTEMA DE AUTOMATIZACIÓN NEURO-DIFUSO

La figura 5.1 presenta el Sistema General de la Grúa Automatizada, constituido por: el Sistema de Automatización Neuro-difuso y el Simulador de la Grúa. Las variables de entrada al Sistema General son las posiciones deseadas del carro y de la pluma, mientras que las de salida son los ángulos de oscilación (θ, ϕ) y las posiciones actuales del carro y de la pluma (r y γ).

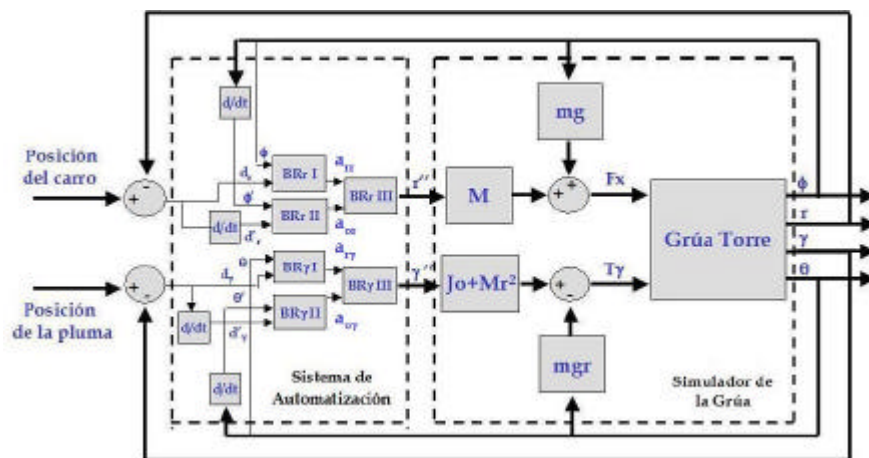


Figura 5.1 – Esquema del Sistema de Grúa Automatizado

Puesto que el Simulador de la Grúa fue descrito en el capítulo III, en este capítulo se detalla sólo el Sistema de Automatización Neuro-difuso, el cual se basa en los fundamentos teóricos descritos en el capítulo II.

5.1 Descripción del Problema

Para el movimiento radial de la grúa, como se muestra en la figura 5.2, el que una carga sea transportada a la posición ZE desde cualquier otra posición, permite definir la posición del carro en términos lingüísticos: NL (Negativo grande), PL (Positivo grande) o ZE (cero). Asimismo, el ángulo de oscilación asociado al movimiento también se puede describir en términos lingüísticos como: NL (Negativo grande), PL (Positivo grande) o ZE (cero).

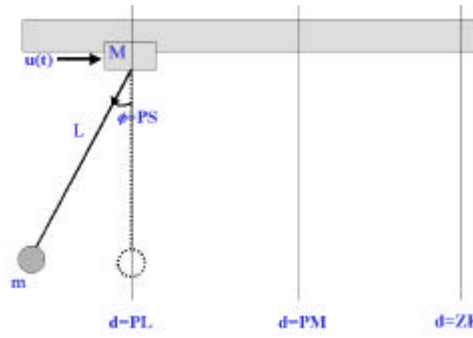


Figura 5.2 - Sistema carro péndulo con variables lingüísticas

Las mismas ideas son válidas en el caso del movimiento rotacional, excepto que en este caso los términos lingüísticos están referidos a la posición angular de la pluma de la grúa y al ángulo de oscilación asociado a este movimiento.

5.2 Diseño del Sistema Neuro-difuso

Como se mencionó, el Simulador de la Grúa fue descrito en el capítulo III, por tanto, en este ítem se detalla sólo el Sistema de Automatización Neuro-difuso, el cual consta de tres bloques de reglas para controlar el movimiento radial (BRr I, BRr II, BRr III) y otros similares para controlar el movimiento rotacional (BRγI, BRγII y BRγIII) como muestra la figura 5.1. Como se explicó en el ítem 2.2.3.1 los bloques o bases de reglas contienen las reglas y los pesos asociados que contienen el conocimiento del sistema. El número de reglas que contiene cada bloque se calcula multiplicando el número de conjuntos difusos de cada variable de entrada con los de cada variable de salida. En este caso, como se observa en la figura 5.1 para el movimiento radial, por ejemplo, se tienen cuatro variables de entrada (d_r , d'_r , ϕ , ϕ') cada una con siete conjuntos difusos (PL, PM, PS, ZE, NS, NM Y NL) y una variable de salida (r'') también con siete conjuntos difusos (PL, PM, PS, ZE, NS, NM Y NL) lo que equivale a tener $7*7*7*7*7 = 16807$ reglas en un solo bloque; sin embargo, el entorno de desarrollo fuzzyTECH sólo permite 343 reglas para cada bloque, por lo que se emplearon tres bloques con 343 reglas cada uno permitiendo contar con un máximo de 1029 reglas para controlar este movimiento. El mismo razonamiento se siguió en el caso del movimiento rotacional.

Antes del entrenamiento, los bloques de reglas de los controladores rotacional y radial, que se muestran en la figura 5.3, fueron alimentados con algunas reglas obvias (“conocimiento a priori”) que permitían describir un comportamiento próximo al deseado; a estas reglas obvias se les asignó un peso 1. Así, cada uno de los bloques BRr I, BRr II, BRγI y BRγII contenían 49 reglas obvias de peso 1 y 294 reglas de peso 0 entre sus 343 reglas iniciales; y los bloques BRr III y BRγIII contenían 13 reglas obvias de peso 1 y 330 reglas de peso 0 entre sus 343 reglas iniciales.

Durante el entrenamiento, el sistema asignó nuevos pesos a cada una de las reglas contenidas en los bloques BR I, BR II y BR III de los controladores de movimientos rotacional y radial de acuerdo a su importancia. Así, las reglas más importantes obtuvieron mayores pesos que las menos importantes.

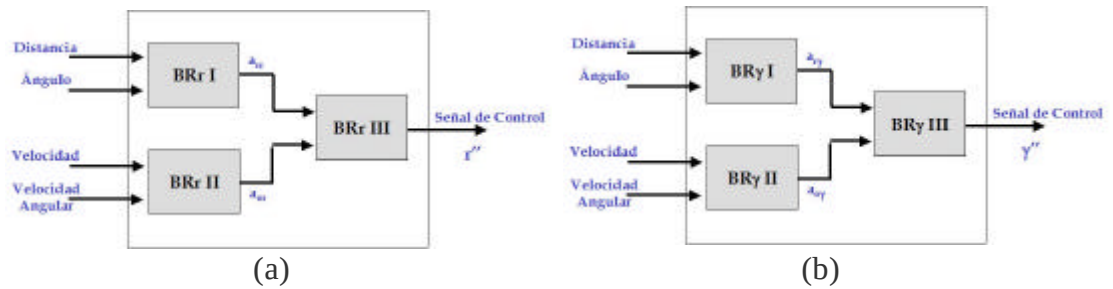


Figura 5.3 – Bloques de Reglas de los controladores (a) Radial y (b) Rotacional

5.2.1 Derivación de Reglas a partir del conocimiento a Priori

En este ítem, se presenta un proceso sencillo de derivación de reglas obvias iniciales para los bloques de reglas del sistema de automatización neuro-difuso; a continuación se detalla el caso de los bloques de movimiento radial BRr I, BRr II y BRr III de la figura 5.3 (a); el procedimiento seguido en el caso del movimiento rotacional es idéntico, por lo que no será detallado.

El bloque BRr I contiene reglas que permiten el control de un sistema masa péndulo utilizando como entradas sólo la distancia a la que se encuentra el actuador (en este caso, el carro de la grúa) de la posición deseada y el ángulo de oscilación asociado. La salida de este bloque es la aceleración a_{rr} .

El bloque BRr II contiene reglas que permiten el control de un sistema masa-péndulo utilizando como entradas las velocidades del actuador (en este caso, el carro de la grúa) y el ángulo de oscilación asociado. La salida de este bloque es la aceleración a_{or} .

El bloque de reglas BRr III, hace las veces de sumador ponderado de las dos aceleraciones (a_{rr} y a_{or}) originadas por los bloques de reglas BRr I y BRr II y contiene reglas que modelan un sumador ponderado. Su salida es la aceleración de control del actuador de la grúa (en este caso, el carro de la grúa).

Se debe remarcar que a_{rr} y a_{or} son señales internas del controlador radial y que la única señal de control generada por este es r'' que se produce a la salida del bloque de reglas BRr III.

5.2.1.1 Diseño de BRr I

Este bloque de reglas utiliza sólo la distancia a la posición deseada y el ángulo de oscilación como variables de entrada, para deducir las reglas iniciales de este bloque se sigue el siguiente procedimiento: como se muestra en la Figura 5.3 (a), en la posición de destino ZE (cero), si el ángulo de oscilación es PS (positivo pequeño), entonces el carro debe moverse en dirección negativa, pero la magnitud de su velocidad no debe ser muy grande para evitar que se mueva muy lejos de la posición ZE, la misma lógica puede ser aplicada para el caso de pendulación opuesto. Cuando el carro está lejos de la posición ZE, como PL, y el ángulo del péndulo es PL (ver figura e), el carro no deberá ser movido hacia la dirección positiva para hacer pequeño el ángulo, porque esto resultará en una gran desviación de la posición del carro; por el contrario, el carro deberá detenerse un momento hasta que el ángulo de oscilación se reduzca. En contraste, si el ángulo de oscilación es NL en la misma posición (ver figura f), el carro puede ser movido con máxima velocidad para reducir el ángulo de oscilación y mover el carro a la posición ZE simultáneamente. Cuando el carro está ubicado en NL, se puede seguir la misma lógica.

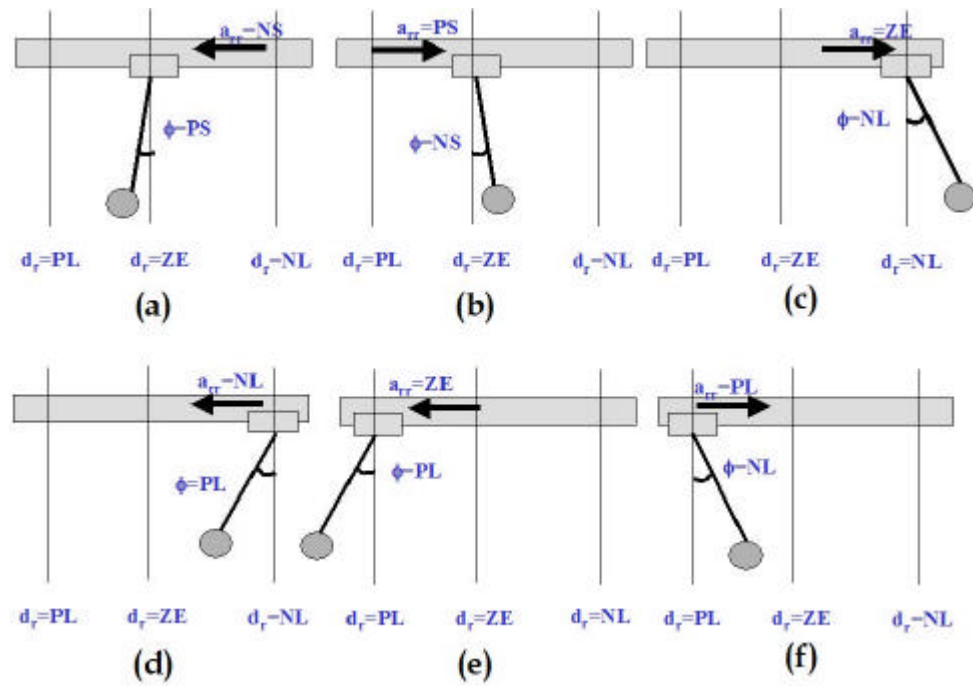


Figura 5.4: Derivación de reglas utilizando la posición y el ángulo

Según lo explicado anteriormente, las variables lingüísticas para las funciones de pertenencia de la distancia a la posición de destino, el ángulo de oscilación y la aceleración son escogidas y llamadas:

$$d_r = \{PL, PM, PS, ZE, NS, NM, NL\}$$

$$\phi = \{PL, PM, PS, ZE, NS, NM, NL\}$$

$$a_{rr} = \{PB, PM, PS, ZE, NS, NM, NB\}$$

donde:

P: Positivo, N: Negativo, L: Largo, M: Mediano, S: Pequeño, ZE: Cero,

d_r : distancia a la posición de destino, ϕ : ángulo de oscilación, a_{rr} : aceleración

A partir de lo descrito líneas arriba se puede construir la tabla de reglas lingüísticas a priori. Los antecedentes son las variables ϕ y d_r y el consecuente a_{rr} (aceleración) está ubicado en la celda correspondiente. Para completar la tabla se comienza por las reglas más obvias y se completa el resto de celdas de tal manera que no se provoquen aceleraciones bruscas que puedan perturbar el sistema.

Tabla 5.1- Diseño de BRr I

		f						
		PL	PM	PS	ZE	NS	NM	NL
d_r	PL	<i>ZE</i>	<i>PS</i>	<i>PS</i>	<i>PM</i>	<i>PM</i>	<i>PL</i>	<i>PL</i>
	PM	<i>NS</i>	<i>ZE</i>	<i>PS</i>	<i>PS</i>	<i>PM</i>	<i>PM</i>	<i>PL</i>
	PS	<i>NS</i>	<i>NS</i>	<i>ZE</i>	<i>PS</i>	<i>PS</i>	<i>PM</i>	<i>PM</i>
	ZE	<i>NM</i>	<i>NS</i>	<i>NS</i>	<i>ZE</i>	<i>PS</i>	<i>PS</i>	<i>PM</i>
	NS	<i>NM</i>	<i>NM</i>	<i>NS</i>	<i>NS</i>	<i>ZE</i>	<i>PS</i>	<i>PS</i>
	NM	<i>NL</i>	<i>NM</i>	<i>NM</i>	<i>NS</i>	<i>NS</i>	<i>ZE</i>	<i>PS</i>
	NL	<i>NL</i>	<i>NL</i>	<i>NM</i>	<i>NM</i>	<i>NS</i>	<i>NS</i>	<i>ZE</i>

5.2.1.2 Diseño de BRr II

Se puede diseñar otra base de reglas intuitiva utilizando como entradas las velocidades del carro y del péndulo y generando una oscilación para eliminar las oscilaciones pendulares. para deducir las reglas iniciales de este bloque se sigue el siguiente procedimiento: como se muestra en las Figura 5.4 (a), en algún momento t , si la velocidad angular de la carga es PL, y si el carro se mueve con una gran velocidad negativa, es preferible no acelerar el sistema. Para el caso opuesto, se puede aplicar la misma lógica. Si la carga se mueve a una velocidad ZE y el carro se mueve a una velocidad NS como se muestra en la figura 5.5 (f), el carro deberá ser acelerado con una magnitud PS para hacer pequeño el ángulo de oscilación. En contraste, si la velocidad angular a la que se mueve la carga es PL, y si el carro también se desplaza con una velocidad PL (ver imagen c), el carro puede ser acelerado al máximo para reducir el ángulo de oscilación. Cuando el carro y la carga se mueven con una velocidad NL, se sigue la misma lógica.

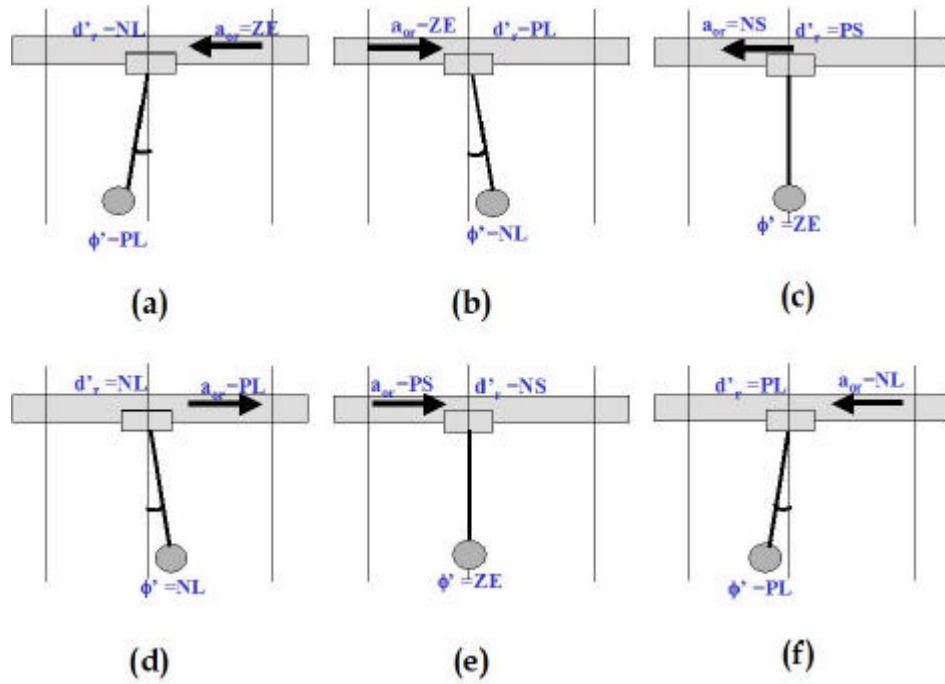


Figura 5.5: Derivación de reglas a partir de las velocidades

Según lo explicado anteriormente, las variables lingüísticas para las funciones de pertenencia son escogidas y llamadas como:

$$d'_r = \{ PL, PM, PS, ZE, NS, NM, NL \}$$

$$\phi' = \{ PL, PM, PS, ZE, NS, NM, NL \}$$

$$a_{or} = \{ PL, PM, PS, ZE, NS, NM, NL \}$$

donde:

P: Positivo, N: Negativo, L: Largo, M: Mediano, S: Pequeño, ZE: Cero, d'_r : Velocidad radial, ϕ' : Velocidad angular, a_{or} : aceleración

A partir de la información obtenida se construye la tabla de reglas de conocimiento a priori siguiendo el mismo método utilizado para completar la tabla 5.1.

Tabla 5.2 – Diseño de BRr II

		f'						
		PL	PM	PS	ZE	NS	NM	NL
d' _r	PL	<i>ZE</i>	<i>NS</i>	<i>NS</i>	<i>NM</i>	<i>NM</i>	<i>NL</i>	<i>NL</i>
	PM	<i>PS</i>	<i>ZE</i>	<i>NS</i>	<i>NS</i>	<i>NM</i>	<i>NM</i>	<i>NL</i>
	PS	<i>PS</i>	<i>PS</i>	<i>ZE</i>	<i>NS</i>	<i>NS</i>	<i>NM</i>	<i>NM</i>
	ZE	<i>PM</i>	<i>PS</i>	<i>PS</i>	<i>ZE</i>	<i>NS</i>	<i>NS</i>	<i>NM</i>
	NS	<i>PM</i>	<i>PM</i>	<i>PS</i>	<i>PS</i>	<i>ZE</i>	<i>NS</i>	<i>NS</i>
	NM	<i>PL</i>	<i>PM</i>	<i>PM</i>	<i>PS</i>	<i>PS</i>	<i>ZE</i>	<i>NS</i>
	NL	<i>PL</i>	<i>PL</i>	<i>PM</i>	<i>PM</i>	<i>PS</i>	<i>PS</i>	<i>ZE</i>

5.2.1.3 Diseño de BRr III

Considerando que un operador humano utiliza, conciente o inconcientemente, las reglas lingüísticas a priori de las tablas 5.1 y 5.2 mientras realiza su trabajo, se hace necesario encontrar una forma de combinar las dos metodologías de control diseñadas que sea capaz de aprovechar lo mejor de ambos sistemas de inferencia con el fin de obtener una respuesta óptima para automatizar la grúa. Inicialmente se modeló un sumador difuso que combina las respuestas de BRr I y BRr II, para lo cual se completaron las posiciones más obvias de la tabla que se presenta a continuación.

Tabla 5.3 – Diseño de BRr III

		a_{rr}						
		PL	PM	PS	ZE	NS	NM	NL
a_{or}	PL	<i>PL</i>						<i>ZE</i>
	PM		<i>PM</i>				<i>ZE</i>	
	PS			<i>PS</i>		<i>ZE</i>		
	ZE				<i>ZE</i>			
	NS			<i>ZE</i>		<i>NS</i>		
	NM		<i>ZE</i>				<i>NM</i>	
	NL	<i>ZE</i>						<i>NL</i>

5.2.2 Diseño de la Fusificación

Para representar los términos de las variables lingüísticas del sistema de automatización Neuro-difuso se escogieron las funciones de pertenencia triangulares que se muestran en las figuras 5.6 a 5.13, por ser las más adecuadas para el diseño de sistemas de control. Las magnitudes y los rangos de estas funciones de pertenencia son resultado del análisis de trabajos similares que forman parte de la bibliografía y del proceso de entrenamiento y optimización de las reglas.

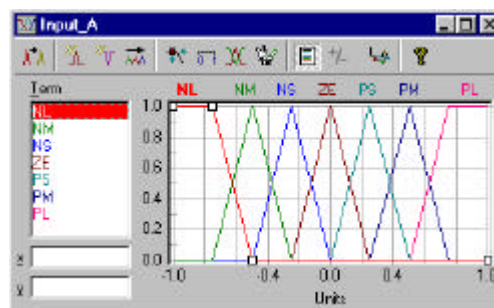


Figura 5.6 – Términos de la variable lingüística d_r (distancia radial)

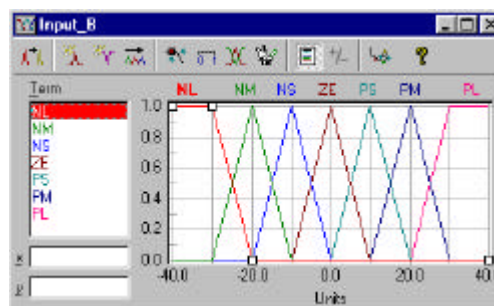


Figura 5.7 – Términos de la variable lingüística ϕ (ángulo de oscilación radial)

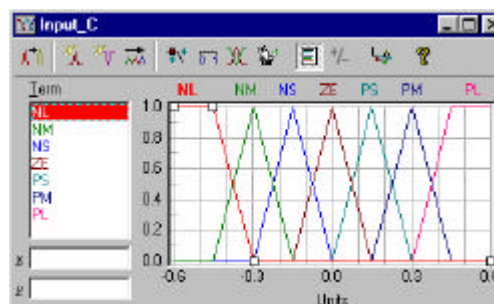


Figura 5.8 – Términos de la variable lingüística d'_r (velocidad radial)

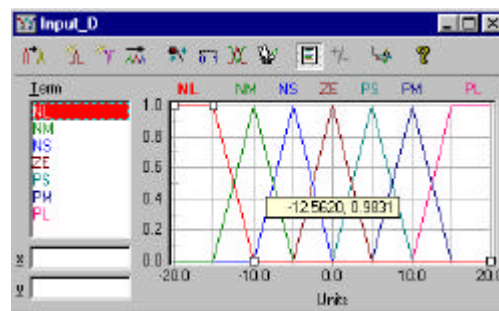


Figura 5.9 – Términos de la variable lingüística ϕ' (velocidad angular radial)

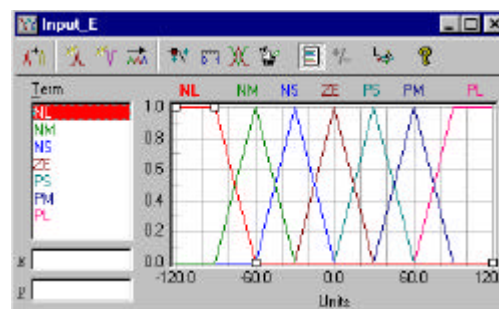


Figura 5.10 – Términos de la variable lingüística d_γ (distancia rotacional)

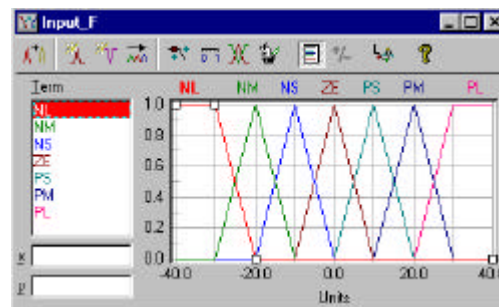


Figura 5.11 – Términos de la variable lingüística d'_γ (velocidad rotacional)

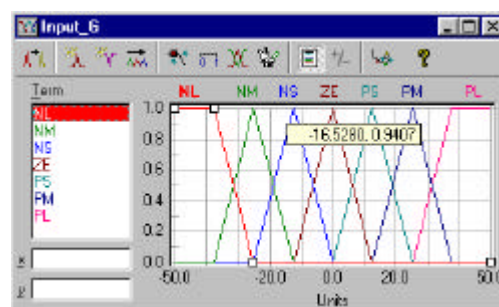


Figura 5.12 – Términos de la variable lingüística θ (ángulo rotacional)

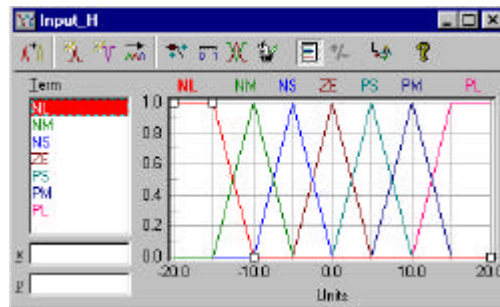


Figura 5.13 – Términos de la variable lingüística θ' (velocidad angular rotacional)

5.2.3 Diseño de la Inferencia y la Defusificación

Luego de un proceso de prueba y evaluación de resultados, para realizar la inferencia difusa, se escogió el operador MIN para la agregación de las entradas, el operador PROD para la composición y el operador BSUM para la agregación de resultados en BRIII y MAX para la agregación de resultados en BRI y BRII.

Algunos trabajos revisados mencionan que el método de defusificación más utilizado para sistemas difusos orientados al control de procesos es el método CoM (Center-of-Maximum), por este motivo se utiliza en esta tesis. Las funciones de pertenencia para la defusificación de la señal de aceleración generada por el controlador difuso se presentan en las figuras 5.14 y 5.15. Las magnitudes y los rangos de estas funciones de pertenencia son resultado del análisis de trabajos similares que forman parte de la bibliografía y del proceso de entrenamiento y optimización de las reglas.

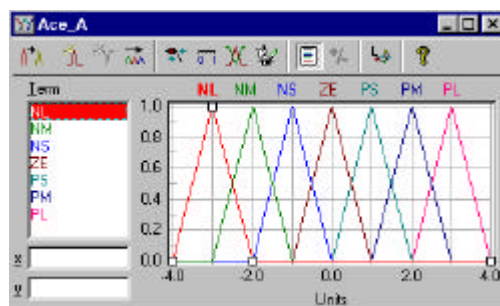


Figura 5.14 – Términos de la variable lingüística r'' (aceleración radial).

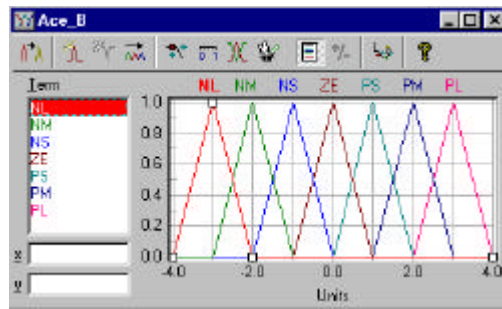


Figura 5.15 – Términos de la variable lingüística γ'' (aceleración rotacional).

5.2.4 Entrenamiento del Sistema

El entrenamiento del sistema se realizó con los datos extraídos mediante el procedimiento descrito en el Capítulo IV. Lo que diferencia al controlador rotacional del controlador radial es que cada uno de ellos fue entrenado a partir de datos característicos diferentes. La Figura 5.15 presenta el sistema neuro-difuso implementado; como se puede observar, consiste de dos controladores de movimientos (radial y rotacional), que producen las aceleraciones radial y rotacional.

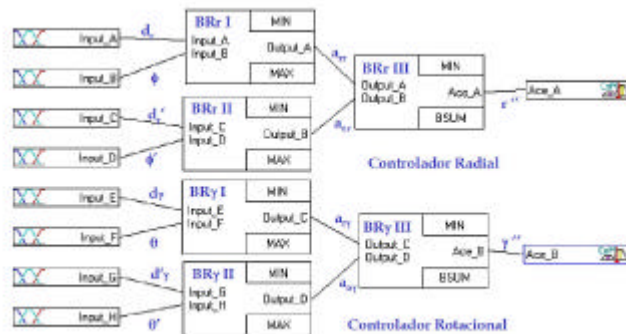


Figura 5.16 – Esquema del Sistema de Automatización de la Grúa Torre.

El entrenamiento del sistema se realizó utilizando el módulo Neurofuzzy de fuzzyTECH que permite seleccionar elementos del sistema, como funciones características o reglas, para que puedan ser entrenados. La herramienta de entrenamiento utilizada fue Random Method; la selección de datos fue aleatoria debido a la amplitud del tamaño de las muestras; las condiciones de convergencia fueron: la desviación máxima entre [50% - 1%] con un factor de 0.75 y la desviación promedio de 0.1%. La figura 5.17 presenta algunas reglas del bloque BRIII del

controlador radial con los pesos asociados generados mediante el proceso de entrenamiento.

Spreadsheet Rule Editor - RB5				
#	IF		THEN	
	Output_A	Output_B	DoS	Ace_A
41	NS	ZE	0.50	ZE
42	NS	ZE	0.02	PS
43	NS	ZE	0.07	PM
44	NS	ZE	0.09	PL
45	NS	PS	0.14	NL
46	NS	PS	0.02	NM
47	NS	PS	1.00	ZE
48	NS	PS	0.19	PS
49	NS	PS	0.08	PL
50	NS	PM	0.16	NL
51	NS	PM	0.06	NS

Figura 5.17 – Algunas reglas y sus pesos generados mediante el entrenamiento.

5.2.5 Optimización

Una de las ventajas de utilizar el método Neuro-difuso para realizar el diseño de un controlador de procesos consiste en que este puede ser modificado antes, durante y después del proceso de entrenamiento, así se reduce el número de iteraciones necesarias para alcanzar la convergencia.

El proceso de optimización del controlador Neuro-difuso fue realizado durante y al final del entrenamiento; consistió en probar el controlador en el simulador de Grúa Torre y modificar manualmente algunos de los elementos del Sistema Neuro-difuso para re-entrenarlo a fin de alcanzar los mejores resultados.

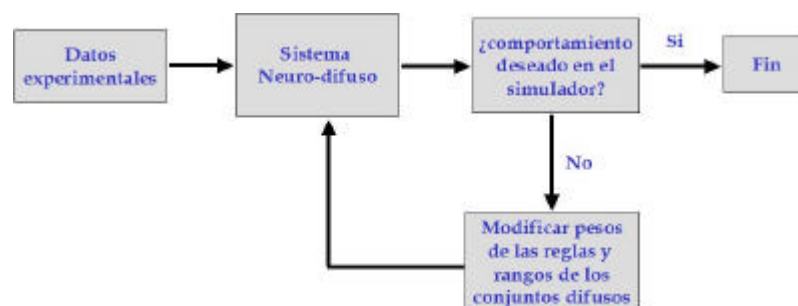


Figura 5.18 – Proceso de entrenamiento y optimización

VI. RESULTADOS

En este capítulo el sistema de automatización Neuro-difuso desarrollado a lo largo de la presente tesis es probado en el simulador 3D de Grúa Torre y sus resultados son comparados con los obtenidos al aplicar, sobre el mismo simulador, los algoritmos de control presentados en las tesis de maestría y doctorado de los investigadores Al-Mousa y Hanafy de la universidad *Virginia Tech*.

6.1 Presentación de Resultados Comparativos

Se presentan a continuación la comparación de los resultados del controlador Neuro-difuso desarrollado a lo largo de esta tesis y los desarrollados por Hanafy y Al-Mousa. Los valores físicos utilizados son: $m=1$ Kg, $M=1.97$ Kg, $J_o=7.2$ Kg m², $l=1$ m.

6.1.1 Neuro-difuso vs Método de Realimentación de Estados de Hanafy

En esta sección se comparan los desempeños del algoritmo Neuro-difuso (NF) y el algoritmo de Realimentación Total de Estados de Hanafy (H FULL). La figura 6.1 y la tabla 6.1 presentan los resultados de la comparación cuando se realiza un movimiento radial; la figura 6.2 y la tabla 6.2 presentan los resultados del movimiento rotacional; la figura 6.3 y la tabla 6.3 presentan los resultados comparativos del movimiento compuesto.

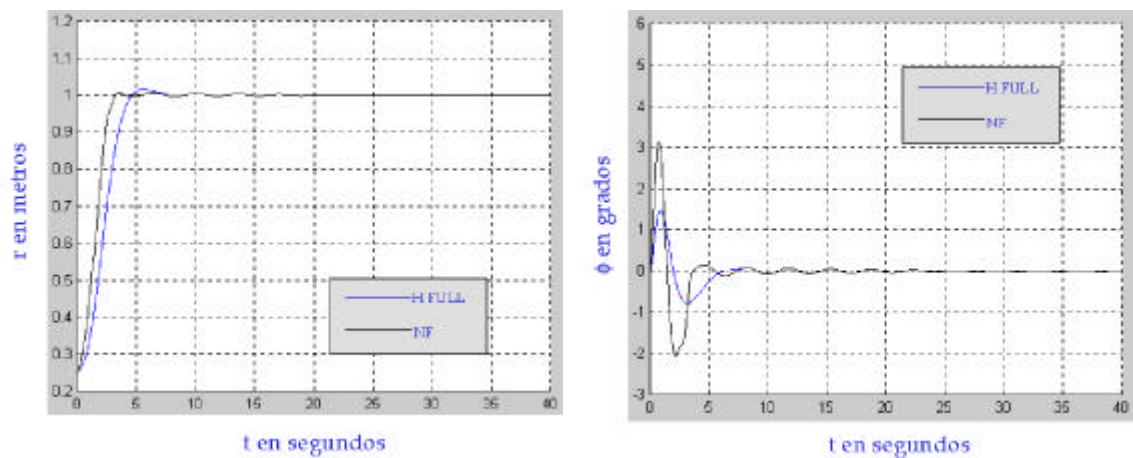
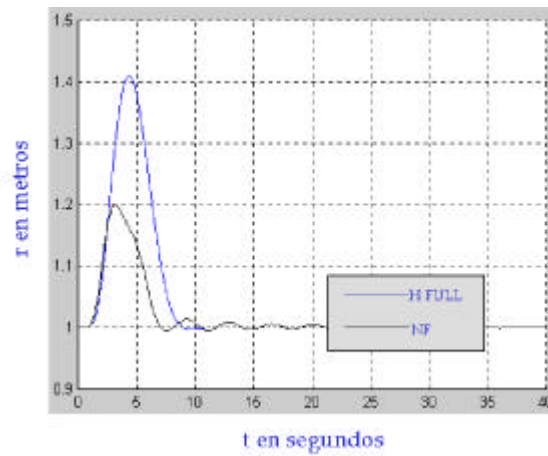


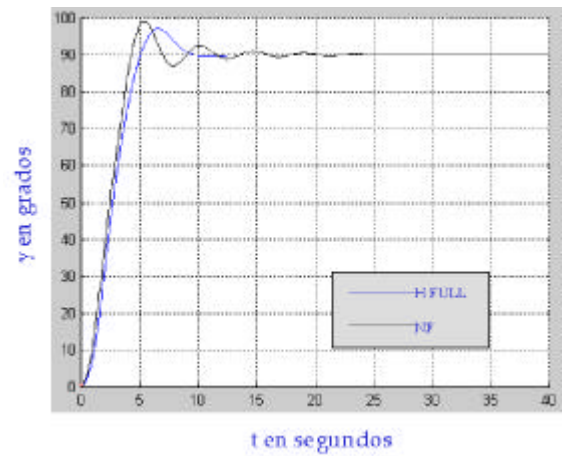
Figura 6.1 - Movimiento radial. (a) posición del carro r , (b) ángulo ϕ

Tabla 6.1 – Movimiento Radial

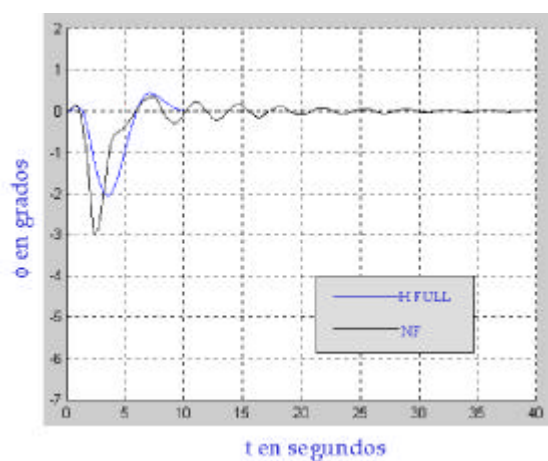
Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre-impulso máximo (m)	Tiempo de Establecimiento (s)
H FULL	1.5°	1	1.01	4.6
NF (Tesis)	3.1°	1	1	3.35



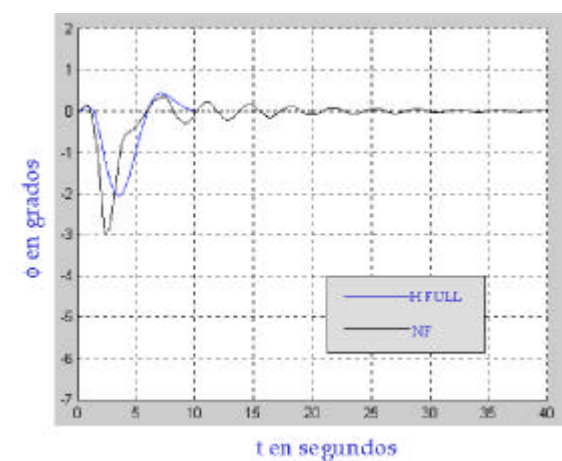
(a)



(b)



(c)

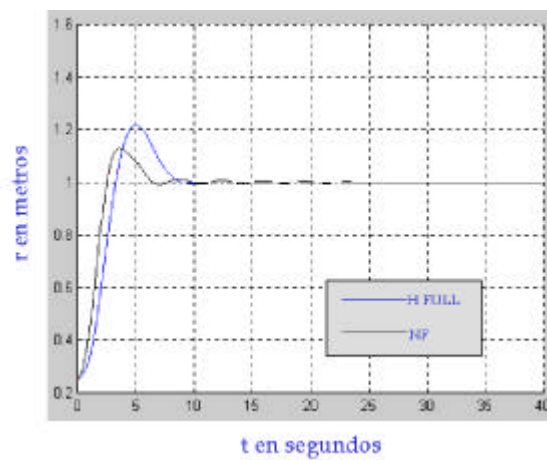


(d)

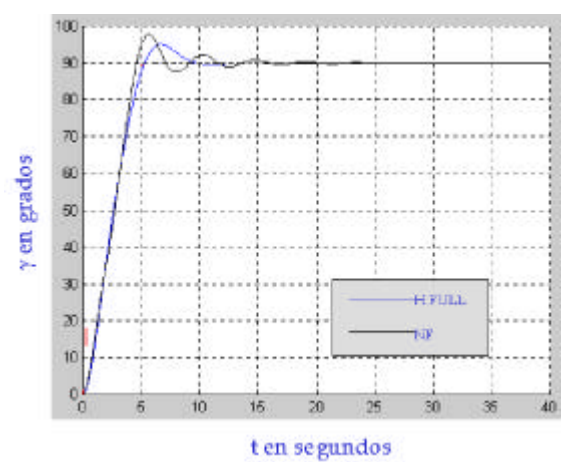
Figura 6.2 – Movimiento Rotacional. (a) Posición r del carro, (b) Posición y de la torre, (c) Ángulo ϕ , (d) Ángulo θ

Tabla 6.2 – Movimiento Rotacional

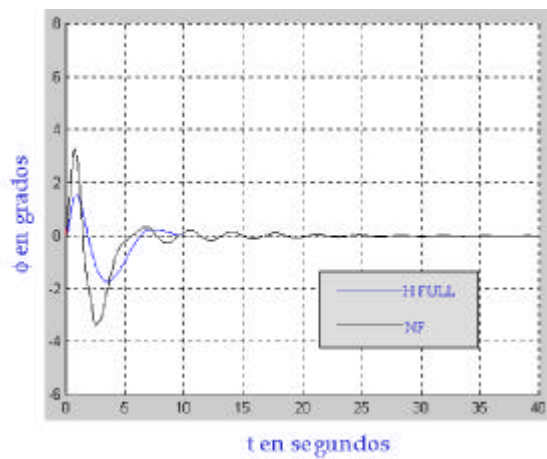
Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre-impulso máximo	Tiempo de Establecimiento (s)
H FULL	-2° (ϕ)	2 (ϕ)	1.41 (r)	8.6
	-2.7° (θ)	1 (θ)	97° (γ)	
NF (Tesis)	-3° (ϕ)	3 (ϕ)	1.2 (r)	10.8
	-3.7° (θ)	3 (θ)	99° (γ)	



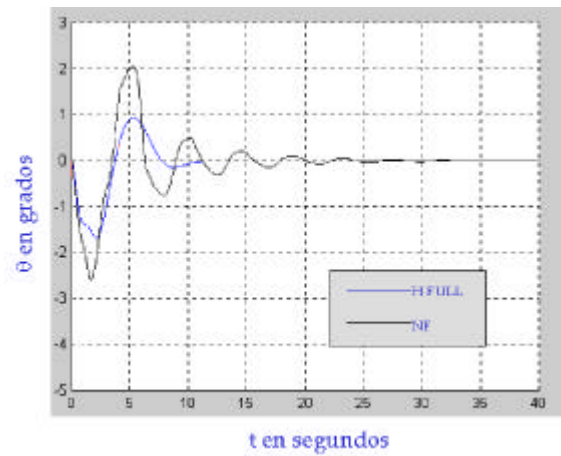
(a)



(b)



(c)



(d)

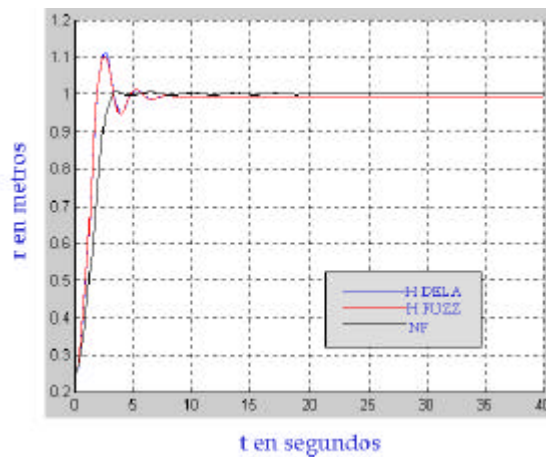
Figura 6.3 – Movimiento Compuesto. (a) Posición r del carro, (b) Posición γ de la torre, (c) Ángulo ϕ , (d) Ángulo θ

Tabla 6.3 – Movimiento Compuesto

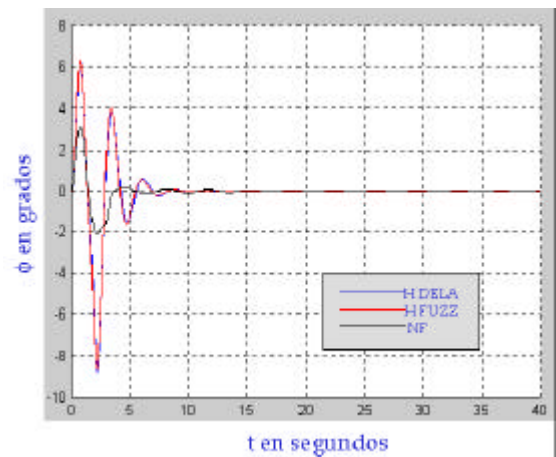
Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre-impulso máximo	Tiempo de Establecimiento (s)
H FULL	-1.75° (ϕ)	1 (ϕ)	1.22 (r)	8.2 (r)
	-17° (θ)	1 (θ)	95.2° (γ)	8.5 (γ)
NF (Tesis)	-3.4° (ϕ)	1 (ϕ)	1.13 (r)	6 (r)
	-2.6° (θ)	2 (θ)	97° (γ)	10.8 (γ)

6.1.2 Neuro-difuso vs Método de Control dual de Hanafy

En esta sección se comparan los desempeños del sistema de automatización Neuro-difuso (NF) y los algoritmos PD-Realimentación Retrasada (H DELA) y PD-Difuso (H FUZZ) de Hanafy. La figura 6.4 y la tabla 6.4 presentan los resultados de la comparación cuando se realiza un movimiento radial; la figura 6.5 y la tabla 6.5 presentan los resultados del movimiento rotacional; la figura 6.6 y la tabla 6.6 presentan los resultados comparativos del movimiento compuesto.



(a)

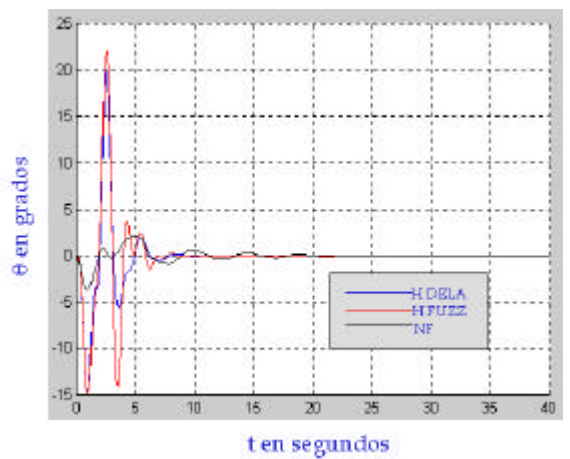
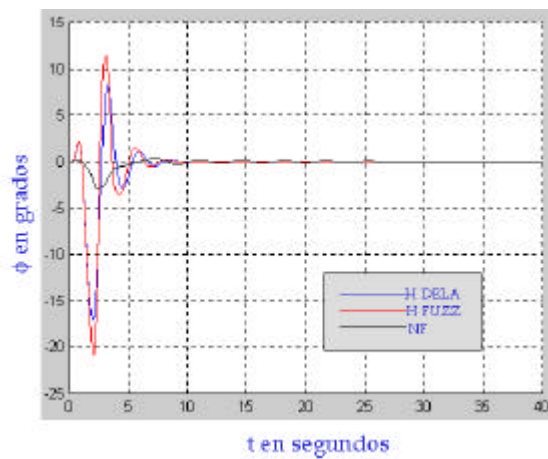
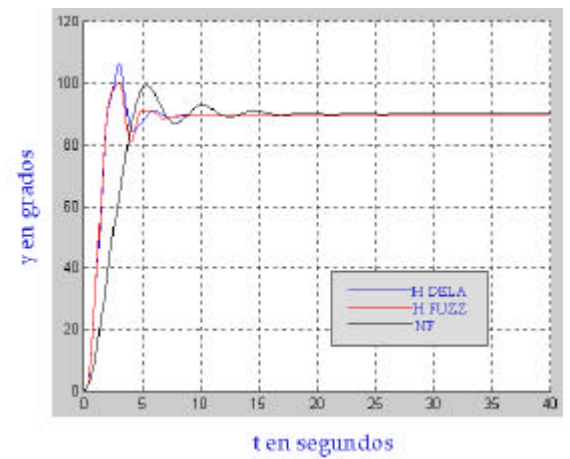
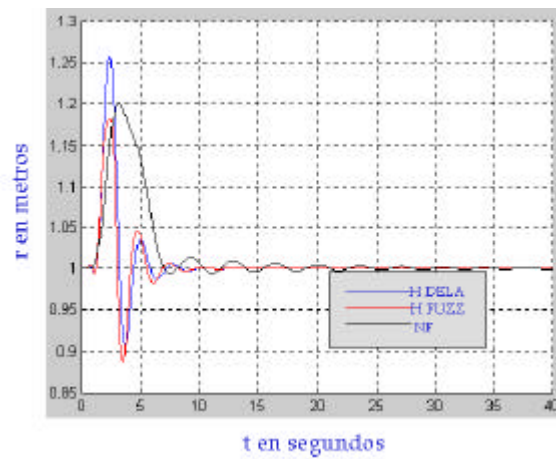


(b)

Figura 6.4 - Movimiento radial. (a) posición del carro r , (b) ángulo ϕ

Tabla 6.4 – Movimiento Radial

Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre-impulso máximo	Tiempo de Establecimiento (s)
H DELA	-8.8	3	1.1	4.5
H FUZZ	-8.6	3	1.1	4.5
NF (Tesis)	3.1	1	1	3.35



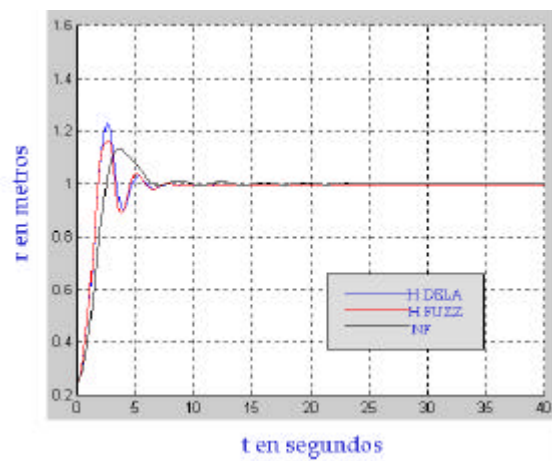
(c)

(d)

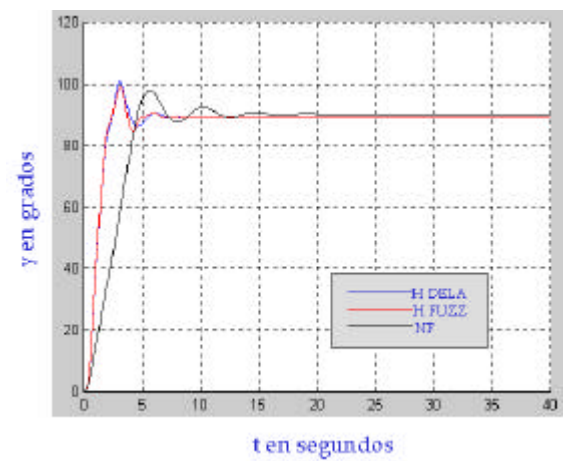
Figura 6.5 – Movimiento Rotacional. (a) Posición r del carro, (b) Posición γ de la torre, (c) Angulo ϕ , (d) Angulo θ

Tabla 6.5 – Movimiento Rotacional

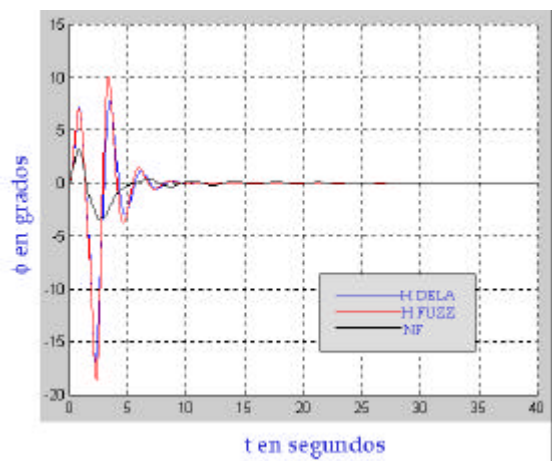
Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre- impulso máximo	Tiempo de Establecimiento
H DELA	-1.7° (ϕ)	3 (ϕ)	1.26 (r)	5.1
	-20° (θ)	2 (θ)	106° (γ)	
H FUZZ	-18.7° (ϕ)	3 (ϕ)	1.18 (r)	6.5
	-20° (θ)	3 (θ)	97° (γ)	
NF (Tesis)	-3° (ϕ)	3 (ϕ)	1.2 (r)	10.8
	-3.7° (θ)	3 (θ)	99° (γ)	



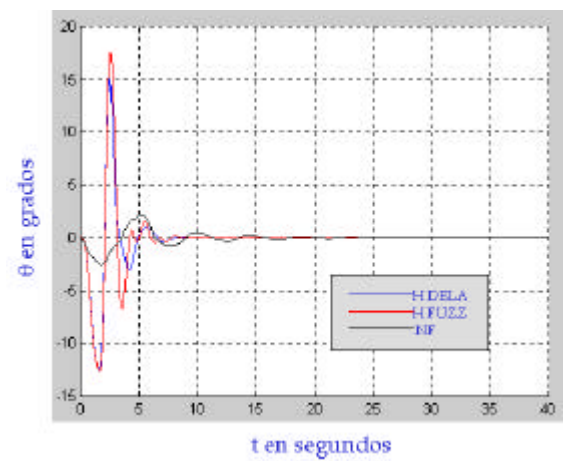
(a)



(b)



(c)



(d)

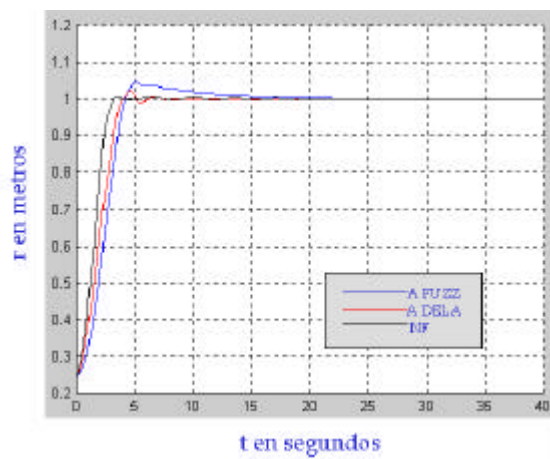
Figura 6.6 – Movimiento Compuesto. (a) Posición r del carro, (b) Posición γ de la torre, (c) Ángulo ϕ , (d) Ángulo θ

Tabla 6.6 – Movimiento Compuesto

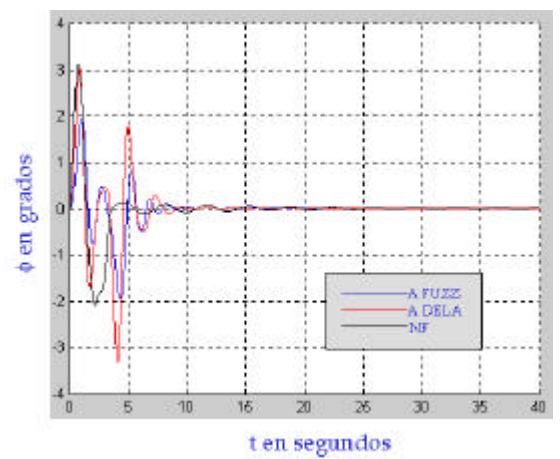
Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre- impulso máximo	Tiempo de Establecimiento
H DELA	-16.8° (ϕ)	3 (ϕ)	1.23 (r)	5.6 (r)
	15° (θ)	2 (θ)	101° (γ)	5.2 (γ)
H FUZZ	-18.6° (ϕ)	3 (ϕ)	1.17 (r)	6.7 (r)
	17.5° (θ)	3 (θ)	99.2° (γ)	4.7 (γ)
NF (Tesis)	-3.4° (ϕ)	1 (ϕ)	1.13 (r)	6 (r)
	-2.6° (θ)	2 (θ)	97° (γ)	10.8 (γ)

6.1.3 Neuro-difuso vs Métodos de Control de Al-Mousa

En esta sección se comparan los desempeños del sistema de automatización Neuro-difuso (NF) y los algoritmos de Realimentación Retrasada (A DELA) y Difuso (A FUZZ) de Al-Mousa. La figura 6.7 y la tabla 6.7 presentan los resultados de la comparación cuando se realiza un movimiento radial; la figura 6.8 y la tabla 6.8 presentan los resultados del movimiento rotacional; la figura 6.9 y la tabla 6.9 presentan los resultados comparativos del movimiento compuesto.



(a)

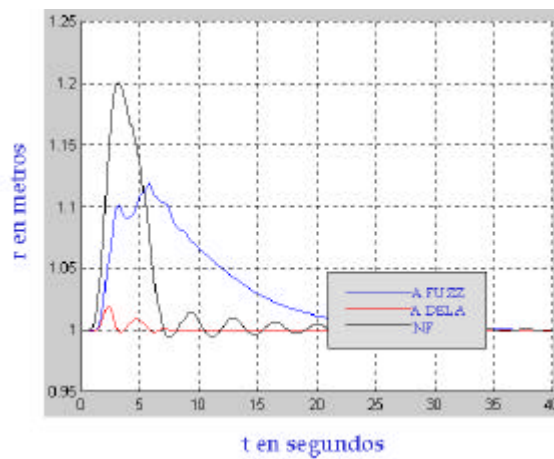


(b)

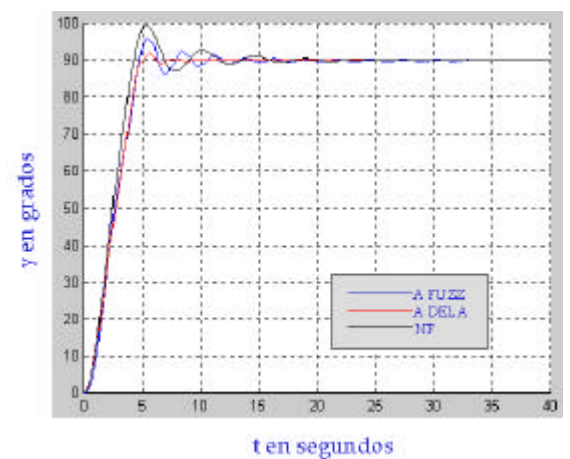
Figura 6.7 - Movimiento radial. (a) posición del carro r , (b) ángulo ϕ

Tabla 6.7 – Movimiento Radial

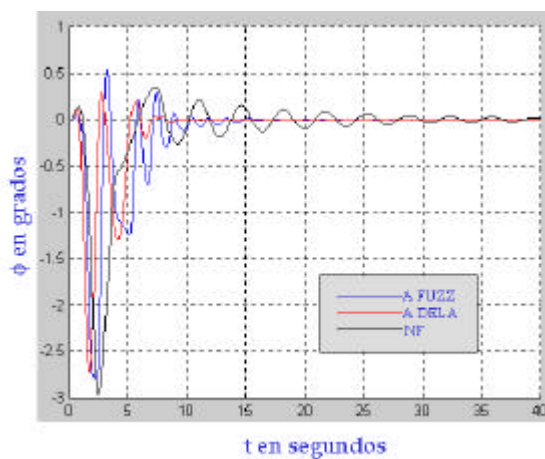
Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre-impulso máximo	Tiempo de Establecimiento
AFUZZ	1.95°	4	1.05	10
ADELA	-3.3°	4	1.025	4.8
NF (Tesis)	3.1°	1	1	3.35



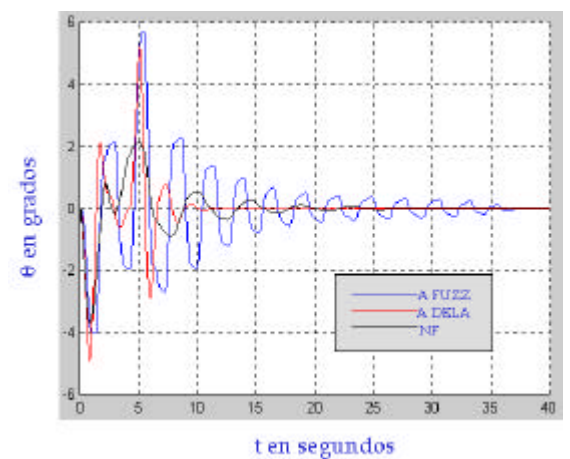
(a)



(b)



(c)

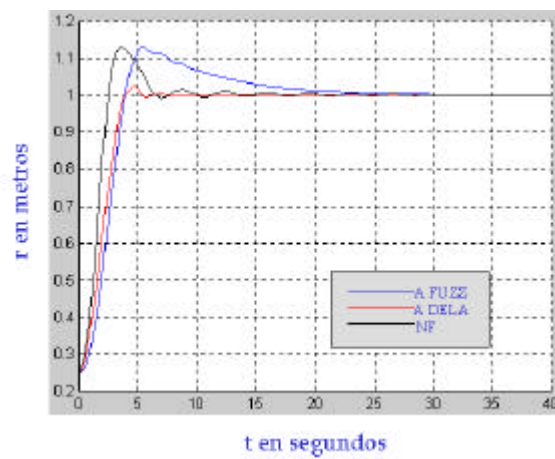


(d)

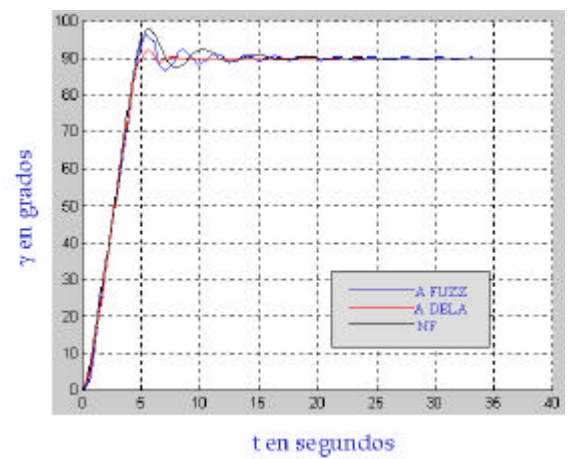
Figura 6.8 – Movimiento Rotacional. (a) Posición r del carro, (b) Posición y de la torre, (c) Ángulo ϕ , (d) Ángulo θ

Tabla 6.8 – Movimiento Rotacional

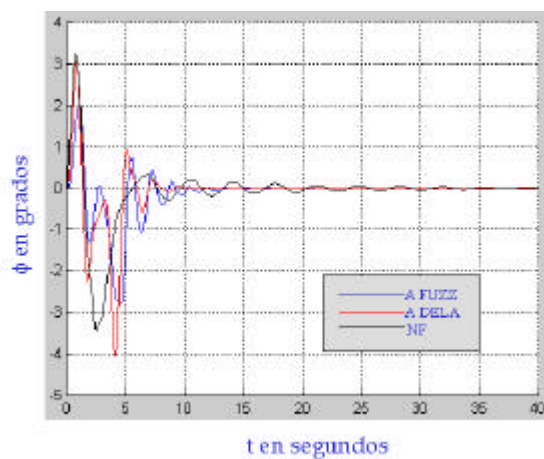
Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre- impulso máximo	Tiempo de Establecimiento
AFUZZ	$-2.82^\circ (\phi)$	4 (ϕ)	1.12 (r)	8.7
	$5.72^\circ (\theta)$	1 (θ)	$96^\circ (\gamma)$	
ADELA	$-2.72^\circ (\phi)$	3 (ϕ)	1.02 (r)	5.8
	$5.2^\circ (\theta)$	3 (θ)	$92.1^\circ (\gamma)$	
NF (Tesis)	$-3^\circ (\phi)$	3 (ϕ)	1.2 (r)	10.8
	$-3.7^\circ (\theta)$	3 (θ)	$99^\circ (\gamma)$	



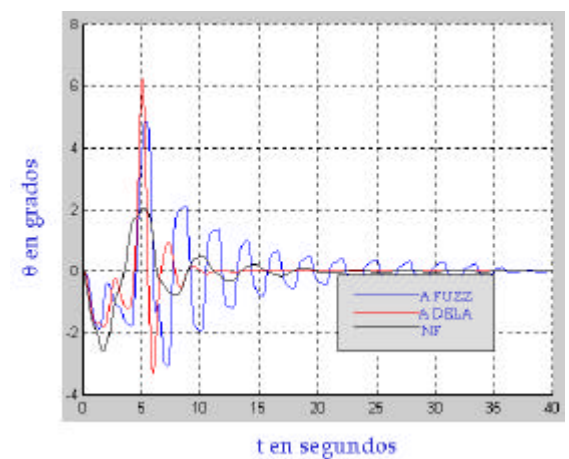
(a)



(b)



(c)



(d)

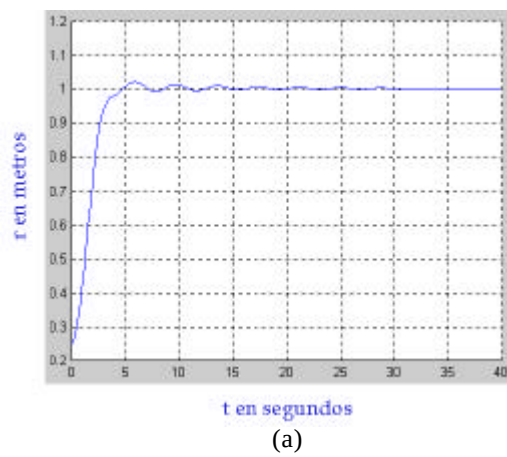
Figura 6.9 – Movimiento Compuesto. (a) Posición r del carro, (b) Posición γ de la torre, (c) Ángulo ϕ , (d) Ángulo θ

Tabla 6.9 – Movimiento Compuesto

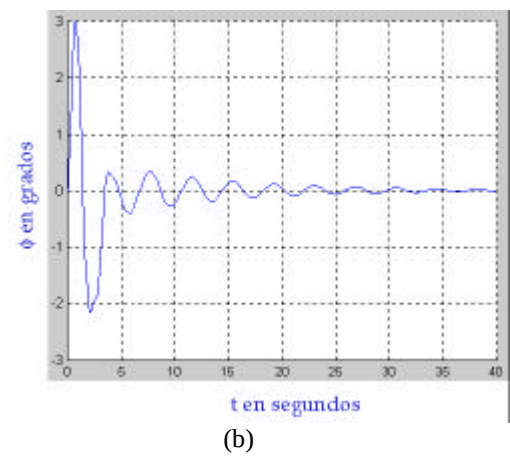
Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre- impulso máximo	Tiempo de Establecimiento
AFUZZ	-2.8° (ϕ)	5 (ϕ)	1.13 (r)	17 (r)
	4.85° (θ)	12 (θ)	96.3° (γ)	8.78 (γ)
ADELA	-4° (ϕ)	3 (ϕ)	1.03 (r)	5 (r)
	6.25° (θ)	4 (θ)	92.5° (γ)	5.8 (γ)
NF (Tesis)	-3.4° (ϕ)	1 (ϕ)	1.13 (r)	6 (r)
	-2.6° (θ)	2 (θ)	97° (γ)	10.8 (γ)

6.2 Otros Resultados

En este apartado se presentan los resultados del sistema frente a consideraciones físicas particulares como la presencia de fricciones, y las variaciones en la magnitud de la masa de la carga y la longitud de la cuerda del sistema difuso. Los valores físicos utilizados fueron $m=1.97$ Kg, $M=1.97$ Kg, $J_o=7.2$ Kg m², $l=1$ m, $\text{fric}_{\text{car}}=0.42$, $\text{fric}_{\text{tor}}=0.738$. La prueba de los controladores se realiza para tres movimientos de la grúa: el movimiento radial, el movimiento rotacional y el movimiento compuesto. El movimiento radial se ejecuta para un desplazamiento del carro a lo largo de 0.75m (0.25m-1m), el movimiento rotacional para un desplazamiento de la pluma de 90° de rotación, manteniendo constante la posición del carro a 1m del origen; y el movimiento compuesto para realizar los dos desplazamientos mencionados, simultáneamente. Los resultados obtenidos del movimiento radial se presentan en la figura 6.10; los resultados del movimiento rotacional se presentan en la figura 6.11; los resultados para el movimiento compuesto se presentan en la figura 6.12; los resultados frente a variaciones en la cuerda se presentan en la figura 6.13 y los resultados frente a variaciones de la masa de la carga en la figura 6.14. Estos resultados no son comparados con los que se podrían obtener utilizando los algoritmos desarrollados en *Virginia Tech*, ya que aquellos algoritmos no fueron diseñados para soportar variaciones en sus parámetros iniciales.

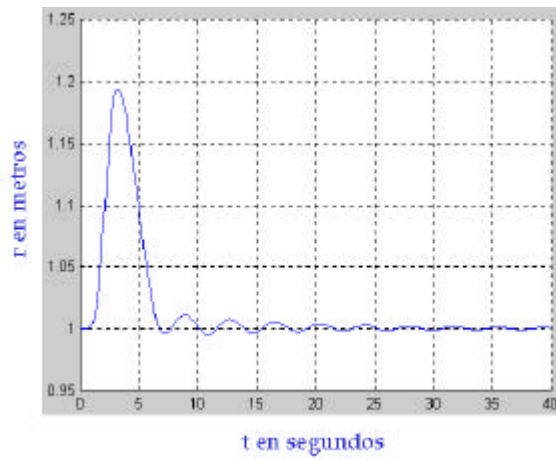


(a)

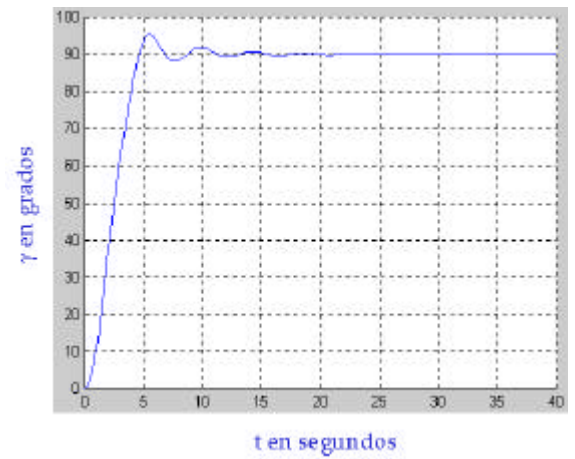


(b)

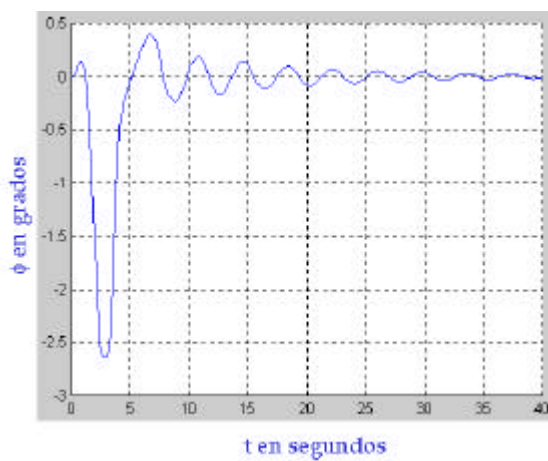
Figura 6.10 - Movimiento radial. (a) posición del carro r , (b) ángulo ϕ



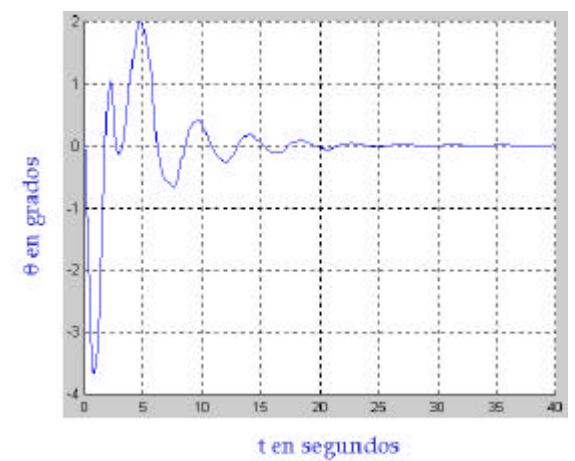
(a)



(b)

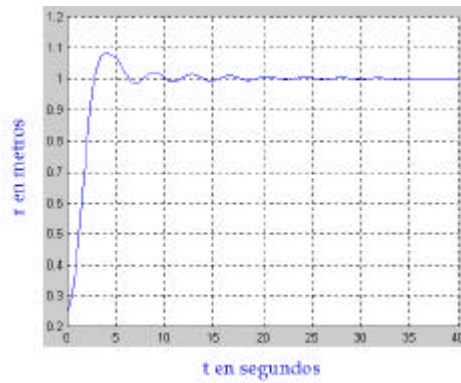


(c)

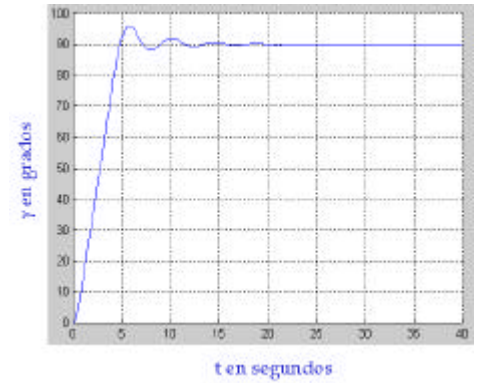


(d)

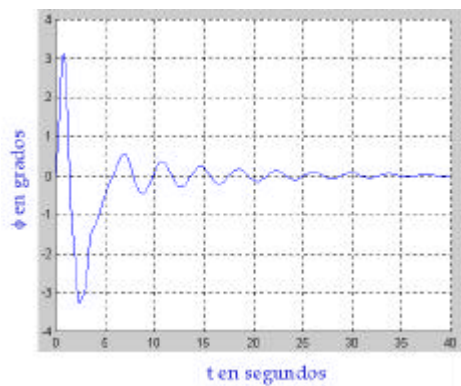
Figura 6.11 – Movimiento Rotacional. (a) Posición r del carro, (b) Posición γ de la torre, (c) Ángulo ϕ , (d) Ángulo θ



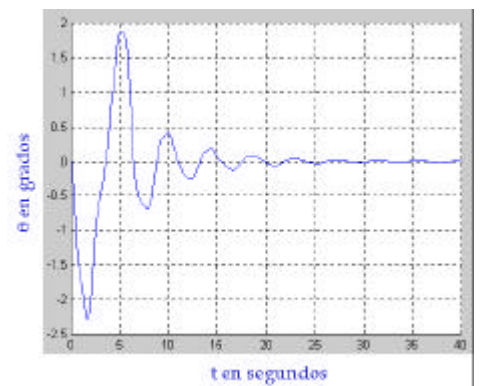
(a)



(b)



(c)

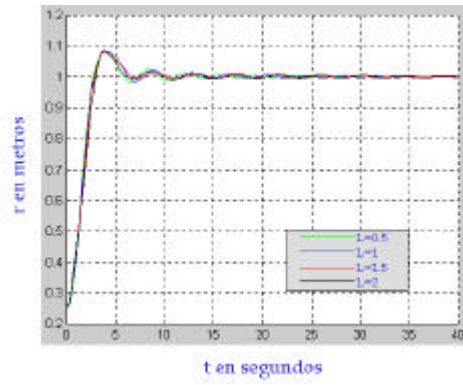


(d)

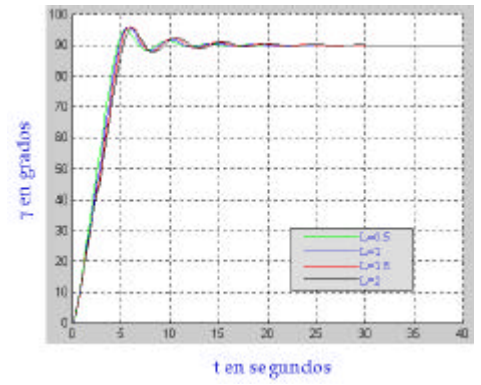
Figura 6.12 – Movimiento Compuesto. (a) Posición r del carro, (b) Posición γ de la torre, (c) Angulo ϕ , (d) Angulo θ , (e) Aceleración radial, (f) Aceleración Rotacional

Tabla 6.10 – Características del controlador Neuro-difuso

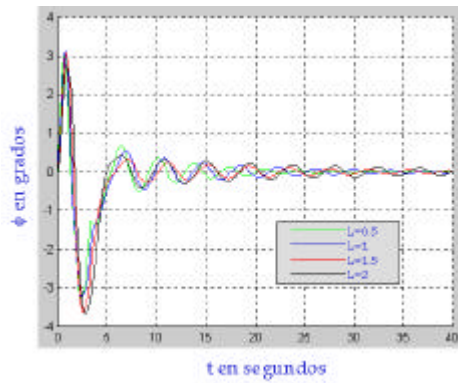
Movimiento	Oscilación Máxima	Número de Ciclos de oscilación	Sobre-impulso máximo	Tiempo de Establecimiento
Radial	3°	1	1.02 m	5.8 s
Rotacional	-2.64°(ϕ)	1 (ϕ)	1.2 m	7.8 s
	-3.66°(θ)	2.5 (θ)	95.2°	
Compuesto	-3.1°(ϕ)	1 (ϕ)	1.085 m	9.2 s
	1.9°(θ)	2 (θ)	96°	10.3 s



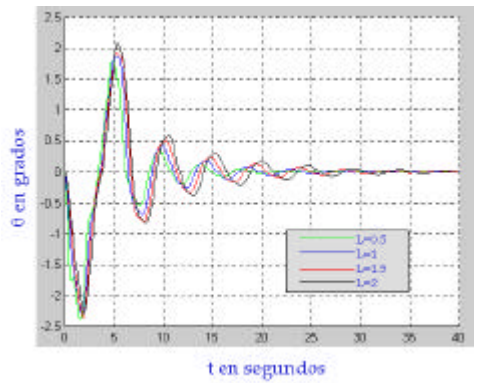
(a)



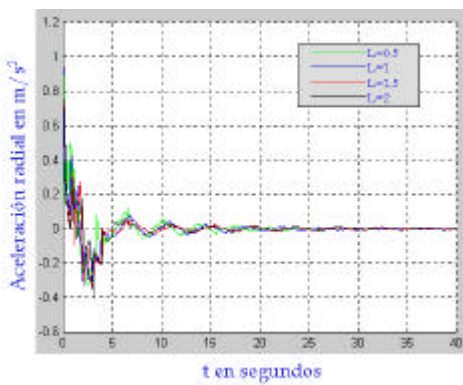
(b)



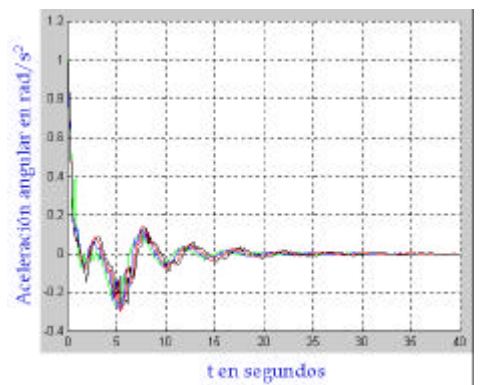
(c)



(d)

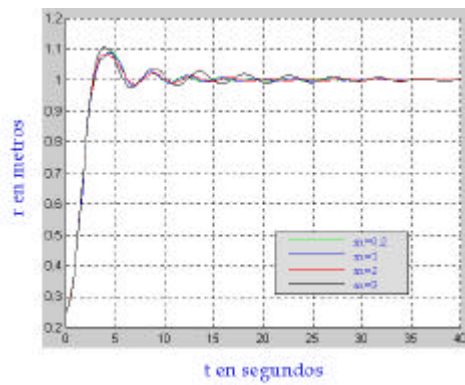


(e)

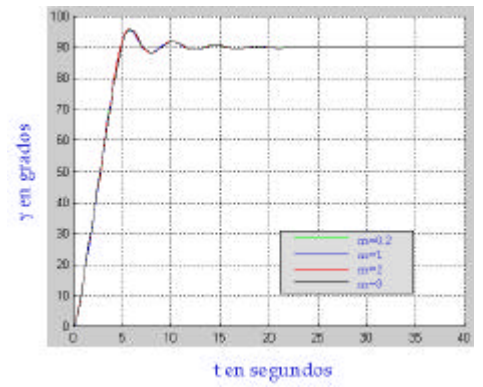


(f)

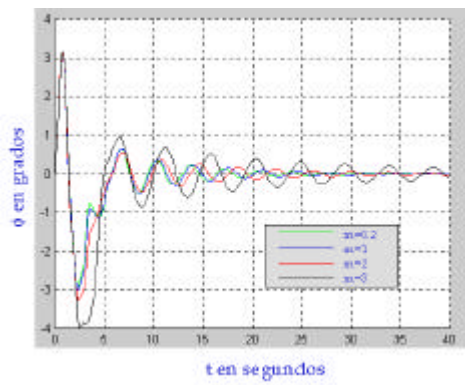
Figura 6.13 – Variaciones en L . (a) Posición r del carro, (b) Posición γ de la torre, (c) Angulo ϕ , (d) Angulo θ , (e) Aceleración radial, (f) Aceleración Rotacional



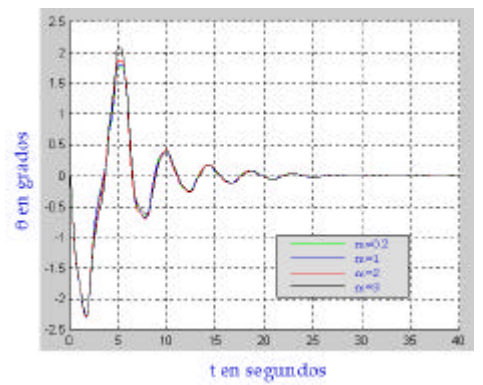
(a)



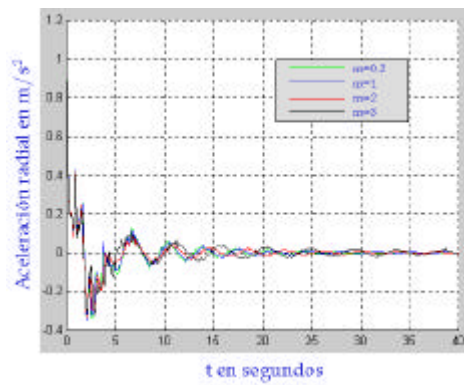
(b)



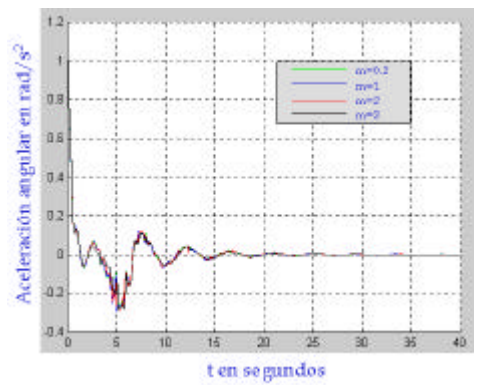
(c)



(d)



(e)



(f)

Figura 6.14 – Variaciones en m. (a) Posición r del carro, (b) Posición γ de la torre, (c) Angulo ϕ , (d) Angulo θ , (e) Aceleración radial, (f) Aceleración Rotacional

VII. Conclusiones

El simulador 3D desarrollado en esta tesis ha demostrado su utilidad porque se ha manejado manualmente, tal como lo haría un operador humano, con el objetivo de extraer conocimiento experimental para entrenar la red Neuro-difusa, y porque se ha manejado automáticamente, con los algoritmos desarrollados en *Virginia Tech* y con el algoritmo de automatización Neuro-difuso, para realizar comparaciones dentro de un mismo marco de referencia.

De los datos que se presentan en las tablas 6.1, 6.2, 6.3, 6.4, 6.7, 6.8 y 6.9, en donde se compara el desempeño del algoritmo de la presente tesis con los desarrollados por investigadores de la universidad de Virginia Tech, se puede observar que con todos los algoritmos experimentados, incluido el Neuro-difuso, los ángulos de oscilación generados debido al movimiento del carro o pluma (actuador) están dentro de un mismo orden de magnitud y sus valores no comprometen la seguridad de la carga. En cambio, como se muestra en las tablas 6.5 y 6.6, los algoritmos de Retraso y difuso de Hanafy, producen ángulos del orden de magnitud de 20° que ponen en riesgo la seguridad de la carga, a diferencia del algoritmo Neuro-difuso que mantiene los ángulos de oscilación por debajo de los 5° .

Cuando se realiza un movimiento radial (es desplazado el carro de la grúa), el sistema de automatización Neuro-difuso ha demostrado tener un tiempo de establecimiento menor que los algoritmos clásicos de Virginia Tech (ver tablas 6.1, 6.4 y 6.7).

En el caso del movimiento rotacional (es desplazada la pluma de la grúa), los algoritmos de Hanafy y Al-Mousa tienen un tiempo de establecimiento menor que el algoritmo Neuro-difuso (ver tablas 6.2 y 6.8).

En el caso del movimiento compuesto (el carro y la pluma de la grúa son desplazados al mismo tiempo), el carro se estabiliza rápidamente con el algoritmo Neuro-difuso y con el de Retraso de Tiempo de Al-Mousa (6 y 5 segundos respectivamente) en comparación con los otros algoritmos de Al-Mousa y los de Hanafy que emplean más de 8 segundos. Sin embargo, la pluma se estabiliza en 10.8 segundos con el algoritmo

Neuro-difuso mientras los otros algoritmos lo hacen en menos de 8 segundos (ver tablas 6.3 y 6.9).

Sólo en los casos que se presentan en las tablas 6.5 y 6.6, se ha observado que el tiempo de establecimiento de los algoritmos desarrollados en la universidad *Virginia Tech* son notoriamente menores que el tiempo de establecimiento del algoritmo Neuro-difuso, pero como se indicó anteriormente, las magnitudes de los ángulos de oscilación que producen ponen en riesgo la seguridad de la carga, lo cual no ocurre con el algoritmo Neuro-difuso. En consecuencia, en estos casos, el algoritmo Neuro-difuso sacrifica tiempo de establecimiento para otorgar seguridad a la carga.

El sistema Neuro-difuso ha demostrado mucha flexibilidad para mantener el mismo desempeño frente a la inclusión de nuevos parámetro físicos en el modelo de grúa, como son: las fricciones, variaciones en la longitud del cable, y variaciones en la magnitud de la masa de la carga (ver figuras 6.13 a 6.14). La flexibilidad y facilidad que tiene el algoritmo Neuro-difuso para adaptarse a cambios se contrapone a las características de los algoritmos de control clásicos, en los cuales la inclusión de nuevos parámetros implica necesariamente el rediseño de los algoritmos, tarea que en muchos casos puede llegar a ser sumamente complicada.

Finalmente, a diferencia de la Teoría de Control que hace uso de modelos matemáticos para diseñar un sistema de automatización para una Grúa Torre, el método Neuro-difuso, que combina las ventajas de la Lógica Difusa y las Redes Neuronales, ha permitido realizar esta tarea utilizando sólo el conocimiento práctico obtenido del manejo de la grúa, evitándose de esta manera las dificultades matemáticas propias de los métodos de control clásico, y mostrando un desempeño igual, y en algunos casos mejor que por métodos tradicionales. Además, ha demostrado ser mucho más flexible frente a la inclusión de nuevos parámetros físicos en el simulador 3D de grúa.

Bibliografía

- [1] A. K. Jain. **Artificial Neural Networks a Tutorial**. IEEE Computer (1999)
- [2] A. Delgado. **Inteligencia Artificial y Minirobots**. Ecoe (1998)
- [3] P. Zegers. **Speech Recognition using Neural Networks**. M.Sc. Thesis. Department of Electrical and Computer Engineering, University of Arizona - Arizona (1998)
- [4] B. Kröse y P. Van Smagt. **An Introduction to Neural Networks**. Faculty of Mathematics & Computer Sciences, University of Amsterdam - Amsterdam (1996).
http://www.fwi.uva.nl/pub/computer_systems/aut_sys/reports/neuro_intro.pdf
- [5] B. M. Del Río y A. S. Molina. **Redes Neuronales y Sistemas Difusos**. Alfaomega (2002)
- [6] F. Peralta y A. Cotrina. **Reconocedor y Analizador de Voz**. Tesis de Pre-grado. Facultad de Ingeniería Electrónica, UNMSM – Lima (2002)
- [7] S. Russell y P. Norvig. **Inteligencia Artificial un Enfoque Moderno**. Prentice Hall (1996)
- [8] The Mathworks. **Fuzzy Logic Toolbox User's Guide**. The Mathworks, Inc (1999)
- [9] G.J. Klir y B. Yuan. **Fuzzy Sets and Fuzzy Logic**. Prentice Hall (1995)
- [10] J.M. Mendel. **Fuzzy Logic Systems for engineering: A tutorial**. Proceedings of the IEEE, Vol 63 No 3, (1995)
- [11] Constantin Von Altrock. **Fuzzy Logic & Neurofuzzy Applications**. Prentice Hall. 1995

- [12] F. J. De Souza. **Modelos Neurofuzzy Hierárquicos**. Phd Thesis. Dep Eng. Electrica, PUC – Rio (1999)
- [13] J. Jatsen. **Neurofuzzy modeling**. Tech. Report No 98-H-874. Department of Automation, Technical University of Denmark – Lyngby (1998)
- [14] **fuzzyTECHÒ 5.3 User's Manual**. INFORM gMBh (1999)
- [15] O. R. Zaï ane, A. Foss, C. Lee, W. Wang **On data clustering Analysis: Scalability, Constraints and Validation**. University of Alberta – Edmonton.
<http://www.cs.ualberta.ca/~zaiane/postscript/pakddZaiane.pdf>
- [16] J. M. Buhman. **Data Clustering and Learning**. Hand book of Brain Theory and Neural Networks, MIT Press (2002)
- [17] Katsuhiko Ogata. **Dinámica de Sistemas**. Prentice Hall (1998)
- [18] Marcelo Alonso y Edward J. Finn. **Física Volumen I**. Addison-Wesley Iberoamericana (1986)
- [19] E. Larradé y A. Miravete. **Grúas**. Publidisa (1996)
- [20] Hanafy M. Omar. **Control of Gantry and Tower Cranes**. Phd Thesis. Department of Mechanics Engineering, Virginia Tech – Virginia (2003)
- [21] Amjed A. Al-Mousa. **Control of Rotary Cranes Using Fuzzy Logic and Time-Delayed Position Feedback Control**. M.Sc. Thesis. Department of Electrical Engineering, Virginia Tech – Virginia (2000)
- [22] R.M. Ghigliaza y P. Holmes. **On the dynamics of cranes, or spherical pendula with moving supports**. International Journal of Non-Linear Mechanics (2002)
<http://www.elsevier.com/locate/ijnonlinmenc>

- [23] M. Olivares y R. Rojas. **Modelado y Control Difuso**. Departamento de Ingeniería Eléctrica, Universidad de la Frontera – Temuco (2003)
- [24] P. Carbonell. **Control Fuzzy Adaptativo estructuras y estabilidad**. Actas del XI Congreso Español sobre Tecnologías y Lógica Fuzzy – León (1996).
<http://www3.unileon.es/estylf2002/>
- [25] Hsien – Chung Wu. **Fuzzy Systems and Neural Networks**. Department of Information Management, National Chi-nan University – Taiwan (2002)
<http://www.im.ncnu.edu.tw/~hcwu/pdf/neurofuzzy.pdf>
- [26] Luca Marchese. **Reti Neuronali su Personal Computer + Fuzzy Logic**. Rev. STEO.11.01. <http://www.steo.it/ici>
- [27] A. Olandosky, E. Cristina, M.E de Carvalho. **A Utilizacao Comparativa de Redes Neurais e Redes Neuro-Fuzzy no reconhecimento de padroes na análise economica financeira de empresas**. Revista de Administração Contemporânea-RAC. (2004)
- [28] C. Pellegrini. **Neurones Artificiels: Modèles et Réseaux**. AIMA Chap 20.
<http://cui.unige.ch/PI/cours/1815slides/16-reseaux-neurones.pdf>
- [29] A. Figueroa, V. Yasuda, F. Panizo y K. Villavicencio. **Control de Movimiento y Fuerza Aplicado a una Grúa (intercon)**
- [30] NIBS Pte Ltd. **Neurofuzzy Computing**. Technical Report TR-960308 (1996)
- [31] Ziyad A. Masoud. **A Control System for the Reduction of Cargo Pendulation of Ship-Mounted Cranes**. Phd Thesis. Department of Mechanics Engineering, Virginia Tech – Virginia (2000)
- [32] J. Tou, R.C. Gonzales. **Pattern Recognition Principles**. Addison Wesley (1974)

