INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

AUTOSINTONIZACIÓN POR BÚSQUEDA TABÚ DEL CONTROL VECTORIAL DIFUSO DE VELOCIDAD PARA UN MOTOR DE INDUCCIÓN.

TESIS

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS CON ESPECIALIDAD EN INGENIERÍA ELÉCTRICA

PRESENTA

JUAN JOSÉ MUÑOZ CÉSAR

MÉXICO D.F. OCTUBRE DE 2005.

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA DE CESIÓN DE DERECHOS

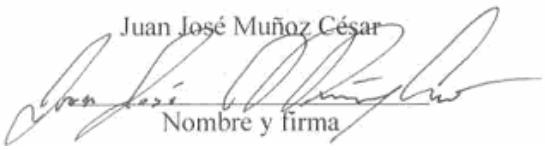
En la ciudad de México, Distrito Federal, el día 21 del mes de Octubre del año 2005, el que suscribe Juan José Muñoz César alumno del Programa de Maestría en Ciencias en Ingeniería Eléctrica , opción en control con número de registro A030625, adscrito a la Sección de estudios de Posgrado e Investigación de la ESIME Unidad Zacatenco, manifiesta que es autor intelectual del presente Trabajo de Tesis bajo la dirección del Dr. David Romero Romero y cede los derechos del trabajo intitulado: **Autosintonización Por Búsqueda Tabú Del Control Vectorial Difuso De Velocidad Para Un Motor de Inducción**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir al contenido textual, graficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección:

mucjuj@hotmail.com o dromero@ipn.mx

Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Juan José Muñoz César



Nombre y firma

SIP-14

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D. F. siendo las 11:00 horas del día 21 del mes de Octubre del 2005 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de la E. S. I. M. E. para examinar la tesis de grado titulada:

"AUTOSINTONIZACIÓN POR BÚSQUEDA TABÚ DEL CONTROL VECTORIAL DIFUSO DE VELOCIDAD PARA UN MOTOR DE INDUCCIÓN"

Presentada por el alumno:

MUÑOZ	CÉSAR	JUAN JOSÉ
Apellido paterno	materno	nombre(s)

Con registro: A 0 3 0 6 2 5

Aspirante al grado de:

MAESTRO EN CIENCIAS

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***SU APROBACIÓN DE LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Director de tesis

DR. DAVID ROMERO ROMERO

DR. DANIEL OLGUÍN SALINAS

DR. JAIME ROBLES GARCÍA

DR. RAÚL ANGEL CORTES MATEOS

DR. JAIME JOSÉ RODRÍGUEZ RIVAS

M. EN C. JESUS REYES GARCÍA

EL PRESIDENTE DEL COLEGIO

DR. JAIME ROBLES GARCÍA

SECCION DE ESTUDIOS DE POSGRADO E INVESTIGACION

RESUMEN

En este trabajo se presenta la autosintonización por Búsqueda Tabú del control vectorial difuso de velocidad para un motor de inducción por medio de simulación digital.

Las pruebas se realizan con los datos de un motor de inducción de 3 hp, trifásico de cuatro polos, utilizando control vectorial directo por flujo orientado del rotor.

El control se realiza con cuatro controles difusos uno de velocidad, uno de flujo y dos para las corrientes, que se sintonizan manualmente, usando 5 y 7 funciones de membresía de tipo triangular para las entradas, mientras que en la salidas se utilizan tipo rodilla a los extremos del universo del discurso. Como entradas se eligen el error y el cambio del error, lo que produce memorias asociativas difusas de 25 y 49 reglas respectivamente. En este trabajo se usa el mínimo para representar el "and" en la premisa y la implicación y el Centro de Gravedad para dedifusificación.

Se propone un esquema para convertir la definición de las reglas difusas a un problema de optimización usando Búsqueda Tabú. Además se desarrollan las estructuras necesarias para el manejo de la memoria y las características de la búsqueda. Finalmente se utiliza la Búsqueda Tabú para sintonizar óptimamente la base de reglas de los controles difusos, en la cual se mantiene la lógica de la base de reglas tanto para definir las vecindades de búsqueda, como para los movimientos y cambios de las reglas.

La investigación se realizo por medio de programación y simulación digital, usando el modelo no lineal de 5to orden del motor de inducción. Los programas se realizaron en el lenguaje Borland C++ ® Versión 5.0A y Borland C++ Builder ® Versión 6.0. Las pruebas realizadas en la simulación digital muestran que se obtienen mejores resultados utilizando la optimización de las reglas difusas por Búsqueda Tabú.

ABSTRACT

This work presents the self-tuning by Tabu Search of Speed Fuzzy Vector Control for an induction motor by means of digital simulation.

The tests are made with data of a 3 HP induction motor, three-phase, four poles, and utilizing rotor-flux-oriented direct vector control.

The control is made with four fuzzy controllers one for speed, one for flux and two for the currents, that are tuned manually, using 5 and 7 triangle type member functions for the inputs, whereas in the outputs knee membership functions for the ends of the discourse universe. As inputs the error and the change of the error are chosen, which respectively produce fuzzy associative memories of 25 and 49 rules. In this work the minimum is used to represent "and" in the premise and the implication and the Center of gravity for defuzzification.

A scheme is developed to change the definition of the diffuse rules into an optimization problem for Tabu Search. Besides the necessary structures for the managing of memory and the characteristics of the search are developed. Finally Tabu Search is used to optimally tune the rule base of the fuzzy controls, in which the logic of the base of rules is maintained to define the neighborhood search, as well for the rules movements and changes.

The research was performed by means of programming and digital simulation using the nonlinear 5^{th} order model of the induction motor. The programs were made in Borland C++ ® Version 5.0A y Borland C++ Builder ® Version 6.0. (Build 10.157). The tests in the digital simulation show that improve results are obtained using the optimization of the fuzzy rules by tabu search.

CONTENIDO

ÍNDICE DE FIGURAS

CAPÍTULO 2 CONTROL DEL MOTOR DE INDUCCIÓN

CAPÍTULO 3 CONTROL DIFUSO

CAPÍTULO 4 BÚSQUEDA TABÚ Y LÓGICA DIFUSA

CAPÍTULO 5 PRUEBAS Y RESULTADOS

ÍNDICE DE TABLAS

CAPÍTULO 3 CONTROL DIFUSO

CAPÍTULO 4 BÚSQUEDA TABÚ Y LÓGICA DIFUSA

APÉNDICE A

GLOSARIO DE TÉRMINOS

α,β	Sistema de coordenadas ortogonal estacionario.
d-q	Sistema de coordenadas ortogonal que gira a la velocidad angular de las corrientes de fase.
b_i	Centro de las funciones de membresía.
C.A.	Corriente Alterna
C.D.	Corriente Directa
$i_{s\alpha,\beta}$	α,β - Corrientes del estator.
$i_{r\alpha,\beta}$	α,β - Corrientes del rotor.
i_{sd}	Componente de corriente en el eje directo (componente producida por el flujo).
i_{sq}	Componente de corriente en el eje de cuadratura (componente producida por el par).
$i_{sa,b,c}$	Corrientes de las tres fases del motor.
i_{mr}	Corriente de magnetización.
J	Momento de inercia (Kg-m^2).
L_s	Inductancia de fase del estator.
L_r	Inductancia de fase del rotor.
L'_s	Inductancia transitoria de fase del estator.
L'_r	Inductancia transitoria de fase del rotor.
L_m	Inductancia mutua (estator a rotor).
P	Número de pares de polos.
R_s	Resistencia de fase del estator.
R_r	Resistencia de fase del rotor.

T_e	Par electromagnético (Nm).
T_L	Par de carga.
$T_r = \frac{L_r}{R_r}$	Constante de tiempo del rotor.
$T_s = \frac{L_s}{R_s}$	Constante de tiempo del estator.
T´s	Constante de tiempo transitoria del estator.
T´r	Constante de tiempo transitoria del rotor.
$u_{s\alpha,\beta}$	α,β - Voltajes del estator.
$u_{r\alpha,\beta}$	α,β - Voltajes del rotor.
u_{sd}	Componente del eje de cuadratura de voltaje.
u_{sq}	Componente del eje directo de voltaje.
u_{dd}	Componente de desacople de voltaje de estator en eje director.
u_{dq}	Componente de desacople de voltaje de estator en eje de cuadratura.
$\hat{u}_{sd}$	Componente lineal de estator en el eje directo.
$\hat{u}_{sq}$	Componente lineal de estator en el eje de cuadratura.
ω_r	Velocidad eléctrica del rotor.
ω_e	Velocidad sincronía (r/s).
ω_m	Velocidad angular del rotor.
ω_r	Velocidad eléctrica del rotor.
ω_{mr}	Velocidad angular del vector espacial del flujo de enlace del rotor.
X_{ls}	Reactancia de pérdidas del estator.
X_{lr}	Reactancia de pérdidas del rotor.

X_m	Reactancia Mutua.
θ_{campo}	Posición del flujo del rotor.
$\psi_{s\alpha,\beta}$	α,β - Flujo magnético del estator.
$\psi_{r\alpha,\beta}$	α,β - Flujo magnético del rotor.
$\psi_{rd,q}$	d-q - Flujo magnético del rotor.
$\sigma = 1 - \frac{L_m^2}{L_s L_r}$	Constante resultante de dispersión
μ	Función de membresía.
e	Error.
Δe	Cambio del error.
Kp	Constante proporcional del contro PI.
Ki	Constante Integral del control PI.
$a_{i,j}$	Posición de la regla en la memoria asociativa difusa.
r	renglón en la base de reglas.
c	columna en la base de reglas.
ISE	Indice de comportamiento integral del error al cuadrado.
rpm	Revoluciones por minuto.
T_B	Torque base.
I_B	Corriente base.
P	Pares de polos.

CAPÍTULO 1

INTRODUCCIÓN

1.1 INTRODUCCIÓN

En este capítulo se presenta el objetivo de la tesis, la justificación, el estado del arte de las metaheurísticas y sistemas de autosintonización en trabajos internacionales y trabajos realizados en la Sección de Estudios de Posgrado e Investigación de la ESIME Zacatenco, además de las aportaciones y el contenido de la tesis.

1.2 OBJETIVO DE LA TESIS

Sintonizar óptimamente la tabla de reglas lingüísticas del control vectorial difuso tipo Mamdani por el método de Búsqueda Tabú, para usarse en el control de velocidad de la Máquina de inducción.

1.3 JUSTIFICACIÓN

En los sistemas electrónicos de potencia es difícil definir los modelos a emplear [4]. Aun si el modelo de la planta es bien conocido existen problemas con la variación de los parámetros. Algunas veces el proceso es multivariable, complicado y no lineal, como el modelo dinámico d-q de la máquina de corriente alterna. El control vectorial de flujo orientado puede superar este problema, pero desarrollar un control preciso es difícil, y existen varios problemas con la variación de parámetros en el sistema [4], [17]. Por otro lado en el control difuso no es indispensable un modelo matemático de la planta [19], [20]. Este esta basado en la experiencia del operador de la planta y la heurística. El control difuso es básicamente un control no lineal y adaptable, el cual provee un comportamiento robusto para sistemas lineales y no lineales con variación de parámetros [4], [18]. De hecho el control difuso es posiblemente el mejor control adaptable [4]. Como el control del sistema esta basado en la experiencia del operador o un experto y tomando en cuenta que estos tienen la capacidad de reconocer pocas reglas claramente (9 reglas) y que estas reglas se hacen confusas o incomprensibles incluso para el experto [21], no se asegura que la sintonización de las reglas sea óptima, un ejemplo es el control de la máquina

de inducción donde en todos los casos encontrados hasta este momento, se utilizan tablas de reglas simétricas.

Al no asegurar en este caso que la base de reglas sea óptima existe la necesidad de utilizar un método que optimice estas tablas. Existen métodos de optimización tales como los de relajación, evolutivos y redes neuronales, de los cuales algunos son utilizados para la optimización de las funciones de membresía. Estos métodos no han sido utilizados para la optimización de la base de reglas lingüísticas pues se necesita un método de optimización combinatoria [2] [21], entre estos se encuentra los métodos de búsqueda y constructivos, de estos se elige el de Búsqueda Tabú su diferencia principal es la agresividad del método para encontrar soluciones optimas [1], [13].

1.4 ESTADO DEL ARTE

Algunos de los tipos fundamentales de metaheurísticas que actualmente son utilizadas son las metaheurísticas para los métodos de relajación, las metaheurísticas para los procesos constructivos, las metaheurísticas para las búsquedas por entornos y las metaheurísticas para los procedimientos evolutivos

Las metaheurísticas de relajación se refieren a procedimientos de resolución de problemas que utilizan relajaciones del modelo original (es decir, modificaciones del modelo que hacen al problema más fácil de resolver), cuya solución facilita la solución del problema original. Entre las metaheurísticas de relajación se encuentran los métodos de relajación lagrangiana [24], o de restricciones subordinadas.

Las metaheurísticas constructivas se orientan a los procedimientos que tratan de la obtención de una solución a partir del análisis y selección paulatina de las componentes que la forman. Dentro de este tipo de metaheurística, destaca la aportación de la metaheurística GRASP [23], que, en la primera de sus dos fases, incorpora a la estrategia greedy pasos aleatorios con criterios adaptables para la selección de los elementos a incluir en la solución.

Las metaheurísticas de búsqueda guían los procedimientos que usan transformaciones o movimientos para recorrer el espacio de soluciones alternativas y explotar las estructuras de entornos asociadas. Estos son los tipos de metaheurísticas más importantes, que establecen estrategias para recorrer el espacio de soluciones del problema transformando de forma iterativa soluciones de partida. Las búsquedas evolutivas se distinguen de estas en que es un conjunto de soluciones, generalmente llamado población de búsqueda, el que evoluciona sobre el espacio de búsqueda. La concepción primaria de heurística más frecuente era la de alguna regla inteligente para mejorar la solución de un problema que se aplicaba iterativamente mientras fuera posible obtener nuevas mejoras. Tales procesos se conocen como búsquedas monótonas (descendentes o ascendentes), algoritmos escaladores (hill-climbing) o búsquedas locales. Esta última denominación obedece a que la mejora se obtiene en base al análisis de soluciones similares a la

que realiza la búsqueda; denominadas soluciones vecinas. Estrictamente hablando, una búsqueda local es la que basa su estrategia en el estudio de soluciones del vecindario o entorno de la solución que realiza el recorrido. Las metaheurísticas de búsqueda local son las estrategias o pautas generales para diseñar métodos de búsqueda local, como la estrategia voraz o greedy. Esta metaheurística establece como pauta, una vez consideradas cuales son las soluciones que intervienen en el análisis local, elegir iterativamente la mejor de tales soluciones mientras exista alguna mejora posible.

Sin embargo, se suele asumir que las búsquedas locales solo modifican la solución que realiza el recorrido mediante una mejora en su propio entorno. El principal inconveniente de estas búsquedas locales es que se quedan atrapadas en un óptimo local, una solución que no puede ser mejorada por un análisis local. Por ello, el propósito fundamental de las primeras metaheurísticas era extender una búsqueda local para continuarla más allá de los óptimos locales, denominándose Búsqueda Global.

Las metaheurísticas de búsqueda global incorporan pautas para tres formas básicas de escapar de los óptimos locales de baja calidad como volver a iniciar la búsqueda desde otra solución de arranque, modificar la estructura de entornos que se está aplicando y permitir movimientos o transformaciones de la solución de búsqueda que no sean de mejora.

Surgen así, respectivamente, las metaheurísticas de arranque múltiple, las metaheurísticas de entorno variable y las metaheurísticas de búsqueda no monótona. Las metaheurísticas de arranque múltiple [28] establecen pautas para reiniciar de forma inteligente las búsquedas descendentes. Las metaheurísticas de entorno variable modifican de forma sistemática el tipo de movimiento con el objeto de evitar que la búsqueda se quede atrapada por una estructura de entornos rígida. Las búsquedas que también aplican movimientos de no mejora durante el recorrido de búsqueda se denominan búsquedas no monótonas.

Las metaheurísticas para búsquedas no monótonas controlan los posibles movimientos de empeoramiento de la solución mediante criterios de aceptación estocásticos o utilizando la memoria del proceso de búsqueda. Las metaheurísticas de búsqueda estocásticas establecen pautas para regular la probabilidad de aceptar transformaciones que no mejoren la solución.

El Recocido Simulado [29] es el exponente más importante de este tipo de metaheurísticas donde la probabilidad de aceptación es una función exponencial del empeoramiento producido. Las metaheurísticas de búsqueda con memoria utilizan información sobre el recorrido realizado para evitar que la búsqueda se concentre en una misma zona del espacio.

Fundamentalmente se trata de la Búsqueda Tabú [1], [11] cuya propuesta original prohíbe temporalmente soluciones muy parecidas a las últimas soluciones del recorrido.

Las metaheurísticas evolutivas están enfocadas a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones. Los algoritmos genéticos y meméticos [25] y los de estimación de distribuciones [26] emplean fundamentalmente procedimientos aleatorios.

Algunas metaheurísticas surgen combinando metaheurísticas de distinto tipo, como la metaheurística GRASP *(*Greedy Randomized Adaptive Search Procedure*)* [23], que combina una fase constructiva con una fase de búsqueda de mejora. Otras metaheurísticas se centran en el uso de algún tipo de recurso computacional o formal especial como las redes neuronales, los sistemas de hormigas o la programación por restricciones y no se incluyen claramente en ninguno de los cuatro tipos anteriores. Por otro lado, de una u otra forma, todas las metaheurísticas se pueden concebir como estrategias aplicadas a procesos de búsqueda, donde todas las situaciones intermedias en el proceso de resolución del problema se interpretan como elementos de un espacio de búsqueda, que se van modificando a medida que se aplican las distintas operaciones diseñadas para llegar a la resolución definitiva. Por ello, y porque los procesos de búsqueda heurística constituyen el paradigma central de las metaheurísticas, es frecuente interpretar que el término metaheurística es aplicable esencialmente a los procedimientos de búsqueda sobre un espacio de soluciones alternativas.

La historia de la aplicación del control difuso desde el desarrollo de la teoría de Zadeh en 1965, hizo su primera aplicación de control en un proceso dinámico, el cual fue reportado por Mamdani en 1974, y por Mamdani y Assilian en 1975. Pero no fue hasta 1987 cuando Dasilva et al. desarrollaron un control difuso adaptable y lo aplicaron a un accionamiento que funcionaba en los cuatro cuadrantes por primera vez, y no fue sino hasta 1989 cuando se aplicó la lógica difusa en electrónica de potencia y accionamiento de motores cuando Li y Lau aplicaron el método a un microprocesador para el control de un servomotor, asumiendo un amplificador de potencia lineal [4]. Desde entonces la literatura ha puesto mucha atención en el potencial de los controles difusos en aplicaciones de control de máquinas, junto con el control vectorial, demostrando que tiene un mejor comportamiento dinámico en presencia de disturbios en la carga y variación de parámetros [6], [9], [10].

En años recientes se ha empezado a utilizar la Búsqueda Tabú para sintonizar la base de reglas de los controles difusos por ejemplo:

En el año de 1999 se realizó un trabajo de aprendizaje de reglas difusas con Búsqueda Tabú, para el problema de estacionar un trailer, a nivel de simulación en la universidad de Milano [2].

En el año 2001 se presenta una implementación de un control difuso para un servomotor de C.D. usando Búsqueda Tabú [16].

En el año 2002 se presenta una tesis para sintonizar las reglas del control difuso de un generador sincrono, utilizando Búsqueda Tabú a nivel simulación en la SEPI ESIME ZAC [21].

Existen estudios de Búsqueda Tabú y lógica difusa en otras áreas de la ingeniería [30], [31] y diversos estudios de Búsqueda Tabú [1], [11], [22], [24].

En la Sección de Estudios de posgrado e Investigación (SEPI) de la Escuela Superior de Ingeniería Mecánica y Eléctrica (ESIME) de la unidad profesional "Adolfo López Mateos" del Instituto Politécnico Nacional (IPN) Zacatenco se han desarrollado trabajos de doctorado y maestría sobre lógica difusa [32], [33], Algoritmos Genéticos[34] y evolucionarías [35], Redes Neuronales [36], [37] y [38], de sistemas híbridos como lo son los Sistemas Neurodifusos [39], [40] y [41] y solo un trabajo con Búsqueda Tabú y Lógica Difusa [21].

1.5 APORTACIONES

Desarrollo del algoritmo computacional del Control Vectorial Difuso Tabú, haciendo énfasis en la lógica que mantiene la estructura de la tabla de reglas. En los trabajos investigados la base tabú es convertida a una lista, modificando de esta manera la vecindad natural de la regla. En este trabajo la base de reglas no pierde su forma.

Se desarrolla un nuevo esquema de búsqueda en las vecindades de la regla o solución actual, al no utilizar migrar la búsqueda hacia los mejores valores encontrados, sin no a aquellos cercanos al mejor valor. El método hace hincapié en la lógica de la búsqueda al elegir la regla en la cual se continuará la búsqueda, y en los movimientos realizados.

Aplicación de la Búsqueda Tabú a un sistema de control multivariable como es el control vectorial difuso de la máquina de inducción.

Desarrollo del software que simula el proceso de Búsqueda Tabú utilizando el sistema de control vectorial difuso del motor de inducción, para la autosintonización de la memoria asociativa difusa del control de velocidad.

Metodología para la obtención de los valores iniciales para las funciones de membresía y universo del discurso en los controles difusos, basados en el comportamiento de los controles convencionales.

Se sintoniza óptimamente la tabla de reglas difusas general para los controles difusos utilizados en el control de velocidad de la máquina de inducción.

1.6 CONTENIDO DE LA TESIS

El trabajo de tesis se encuentra estructurado en 6 capítulos y 3 apéndices los cuales contienen lo siguiente:

El capítulo dos contiene una descripción general del esquema de control de velocidad autosintonizable de la máquina de inducción propuesto en este trabajo con la descripción del modelo de flujo del rotor y la descripción del circuito de desacople además contiene información de cómo sintonizar los controles clásicos del esquema vectorial para la máquina de inducción.

El capítulo tres trata de los controles difusos utilizados en el esquema propuesto y la sintonización de los universos de los discursos.

El capítulo cuatro trata de la Búsqueda Tabú y lógica difusa, en la conversión de la definición de las bases de reglas difusas a un problema de optimización utilizando la Búsqueda Tabú, describe las estructuras utilizadas para el manejo de memoria y contiene la descripción del método utilizado para sintonizar la base de reglas en este trabajo, además de las características de la Búsqueda Tabú que fueron implementadas.

El capítulo cinco muestra las pruebas realizadas al sistema en presencia de transitorios, usando el control vectorial con controles clásicos PI, control vectorial difuso (PI) y control difuso autosintonizable por Búsqueda Tabú, por último comparando los resultados obtenidos.

El capítulo seis presenta las conclusiones, aportaciones y sugerencias para trabajos futuros.

El apéndice A contiene el modelo del motor utilizado para simular el sistema y los datos del mismo.

El apéndice B contiene la descripción de las transformaciones de Clarke y Park utilizadas en este trabajo.

El apéndice C contiene el programa de simulación y todas las funciones necesarias para su funcionamiento con una breve descripción de estas.

CAPÍTULO 2

CONTROL DEL MOTOR DE INDUCCIÓN

2.1 INTRODUCCIÓN

Los motores de corriente alterna (C.A.) tienen estructuras muy acopladas, no lineales y de múltiples variables, en comparación con las estructuras desacopladas, mucho más simples de los motores de corriente directa (C.D.) con excitación separada [6] y [4]. Por otra parte los motores de C.A. tienen varias ventajas: son ligeros (de 20 a 40% más ligeros que los motores de C.D. equivalentes), son poco costosos y tienen poco mantenimiento en general, en comparación con los motores de C.D.

Se requiere controlar en ellos la frecuencia, el voltaje y la corriente, para aplicaciones de velocidad variable [6]. Los convertidores de frecuencia, los inversores y los controladores de voltaje de C.A. pueden controlar la frecuencia, el voltaje o la corriente llenando estos requisitos. Estos convertidores de potencia, que son relativamente complejos y más costosos, requieren técnicas más avanzadas de control por retroalimentación, como por ejemplo, control por referencia al modelo, control adaptable, control en modo deslizante y control por campo orientado.

Los motores trifásicos de inducción son los que se usan con mayor frecuencia con accionamientos de velocidad variable y están sustituyendo a los de C.D. en muchas de las aplicaciones industriales y domésticas.

Con el control vectorial de la máquina de inducción se obtiene una mejor respuesta del sistema que con controles convencionales, pero no logra tener un buen comportamiento cuando existen variaciones de parámetros en el sistema, además de que desarrollar un control preciso se torna complicado [4], [17]. Por este motivo se propone agregar la característica de controles difusos al sistema ya que estos han demostrado ser capaces de proveer un comportamiento robusto para sistemas lineales y no lineales con variación de parámetros, el control difuso es básicamente un control no lineal y adaptable [19], [20]. Otra de sus características es que no es indispensable un modelo matemático de la planta [3], puesto que esta basado en la experiencia del operador de la planta y la heurística, este conocimiento se pone dentro de una memoria asociativa difusa que utiliza el control difuso para poder generar acciones de control.

Como el control del sistema esta basado en la experiencia del operador o un experto y tomando en cuenta que estos tienen la capacidad de reconocer pocas reglas claramente (9 reglas) y que estas reglas se hacen confusas o incomprensibles incluso para el experto [21], no se asegura que la sintonización de las reglas sea óptima, por lo que es necesario utilizar una técnica de optimización que sea capaz de trabajar con la base de reglas del control difuso. Con esto se propone un control vectorial directo por flujo orientado del rotor difuso tipo Mamdani autozintonizable de velocidad por Búsqueda Tabú para la máquina de inducción.

2.2 ESQUEMA DE CONTROL

Para desarrollar la estructura del control propuesta en este trabajo es necesario implementar las características del control vectorial directo por flujo orientado del rotor, utilizando las transformaciones de Clarke y Park, además del modelo de flujo del rotor y el circuito de desacople [4], [6] y [17].

Los controles convencionales PI se sintonizan utilizando las características del control vectorial, de desacoplamiento del motor reduciendo el problema a ecuaciones diferenciales lineales de primer orden para los lazos internos, pero no así para los lazos de control externos, obteniendo para el lazo de velocidad externo una ecuación característica de tercer orden y para el control de flujo dos bloques de primer orden con constantes de tiempo diferentes [42], que se presentan en la sección 2.3.

Para el control difuso de la máquina se utilizan 4 controles difusos tipo Mamdani que sustituyen a cada uno de los controles convencionales, sintonizando individualmente los universos de los discursos, la sintonización se basa en el comportamiento del control convencional aunque también es posible desarrollarla basándose en el comportamiento de la máquina bajo transitorios y por último se realiza una sintonización general.

Después de sintonizar manualmente el espacio del discurso, se utiliza la Búsqueda Tabú para optimizar la memoria asociativa difusa del control de forma general.

El diagrama a bloques del control propuesto se observa en la figura 2.1.

A continuación se muestra el procedimiento que se sigue para desarrollar el control empleado.

- Medir las cantidades del motor (corrientes y voltajes de fase).

- Transformarlas al sistema de 2 fases (α, β) usando la transformación de Clarke. Ver apéndice B.

- Calcular el vector espacial del flujo del rotor posición y magnitud.

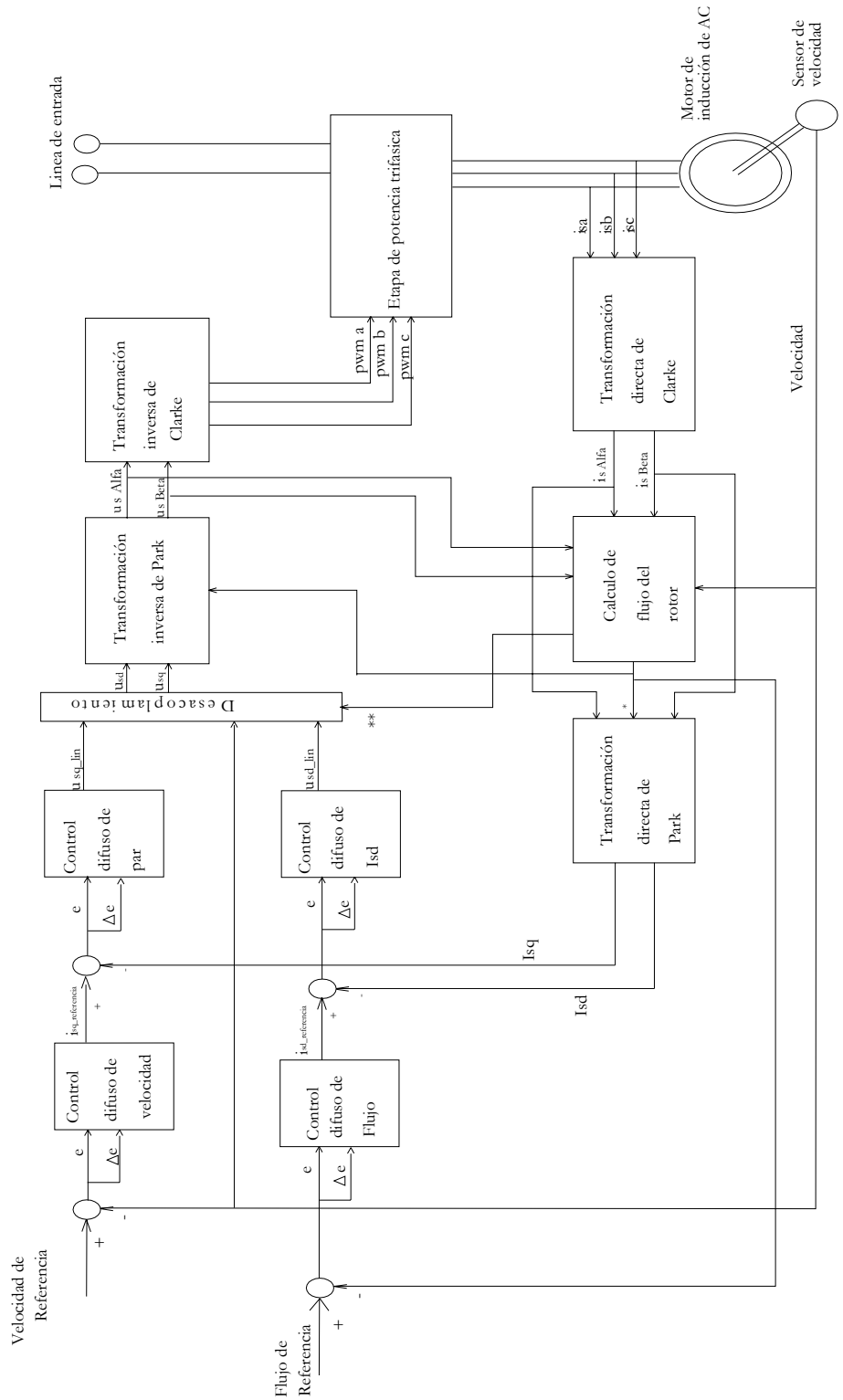


Figura 2. 1 Diagrama a bloques del control vectorial de la máquina de inducción.

* Posición del flujo del rotor

** ψ_{rd}

- Transformar las corrientes del estator al sistema de coordenadas d-q usando la transformación de Park. Ver apéndice B.

- Las componentes de corriente del estator producidas por el par (i_{sq}) y el flujo (i_{sd}) son controladas por separado con el uso de los controles difusos tipo Mamdani, optimizando la base de reglas utilizando Búsqueda Tabú.

- El vector espacial de la salida de voltaje del estator es calculado usando el bloque de desacople.

- El vector espacial del voltaje del estator es transformado usando la transformación inversa de Park del sistema d - q al sistema de dos fases arreglado para el estator.

- Las tres salidas del voltaje son generadas con la transformación inversa de Clarke.

2.3 CONTROL VECTORIAL

El control vectorial es una de las técnicas más usadas [4], [6] y [9] de control para motores de inducción. En marcos de referencia especial, la expresión para el par electromagnético del entrehierro liso, es similar a la expresión de par para la máquina de C.D. con excitación separada [6]. En el caso de las máquinas de inducción, el control es desarrollado en el marco de referencia d-q ligado al vector espacial del flujo del rotor. Esta es la razón por la que la implementación del control vectorial requiere información acerca del módulo y posición del vector espacial del flujo. Las corrientes del estator de la máquina de inducción están separadas en componentes producidas por el flujo y el par, al utilizar la transformación del sistema de coordenadas d-q, cuyo eje directo (d) es alineado con el vector espacial del flujo del rotor. Lo que significa que la componente en el eje q del vector espacial del flujo es siempre cero [4], [6], [9] y [17].

$$\psi_{rq} = 0 \quad \text{y también} \quad \frac{d}{dt}\psi_{rq} = 0 \qquad (2.1)$$

Las componentes $i_{s\alpha}$ e $i_{s\beta}$ calculadas con la transformación de Clarke, están ligadas al marco de referencia del estator α, β. En control vectorial esto es necesario para tener todas las cantidades expresadas en el mismo cuadro de referencia. El cuadro de referencia del estator no es factible para el proceso de control. El vector espacial $\bar{i}_s$ esta rotando en una razón igual a la frecuencia angular de las corrientes de fase. Las componentes $i_{s\alpha}$ e $i_{s\beta}$ dependen del tiempo y velocidad. Es posible transformar estas componentes del marco de referencia del estator al marco de referencia d-q que gira a la misma velocidad angular de la frecuencia de las corrientes de fase. Así las componentes i_{sd} e i_{sq} no son dependientes de la velocidad o del tiempo.

2.3.1 Modelo del flujo del rotor

El conocimiento de la magnitud y ángulo (posición) del vector espacial del flujo del rotor es la principal información para el control vectorial del motor de inducción. Con el vector espacial del flujo magnético del rotor, es posible establecer el sistema de coordenadas rotativo (d-q).

La implementación del modelo del flujo del rotor se calcula utilizando la medición de los voltajes, corrientes del estator y la velocidad del rotor.

Si los efectos de saturación son despreciados, la ecuación del voltaje del estator puede definirse de la siguiente forma:

$$u_s = R_S i_s + L_S \frac{di_s}{dt} + L_m \frac{di_r}{dt} \qquad (2.3.1.1)$$

Donde i_s e i_r son los fasores espaciales de corriente del estator y el rotor respectivamente en el marco de referencia estacionario.

Si la ecuación 2.3.1.1 es expresada en términos de la corriente de magnetización del rotor, (i_{mr}), las siguientes ecuaciones de voltaje son obtenidas para el estator:

$$u_s = R_s i_s + \sigma L_S \frac{di_s}{dt} + (1-\sigma) L_s \frac{di_{mr}}{dt} \qquad (2.3.1.2)$$

Donde σ es la constante resultante de dispersión. Así de la ecuación 2.3.1.2

$$(1-\sigma) T_s \frac{d\,\mathrm{i}_{mr}}{dt} = \frac{u_s}{R_s} - i_s - T'_S \frac{di_s}{dt} \qquad (2.3.1.3)$$

Donde Ts y T'_s son la constante de tiempo del estator y la constante de tiempo transitoria del estator respectivamente. Por lo que al añadir a esto a la ecuación de la corriente de magnetización 2.3.1.4.

$$T_r \frac{di_{mr}}{dt} = i_s - i_{mr} + j\omega_r T_r i_{mr} \qquad (2.3.1.4)$$

Se obtiene:

$$\frac{di_{mr}}{dt}[T_r + T_s(1-\sigma)] = \frac{u_s}{R_s} + (j\omega_r T_r - 1) i_{mr} - T'_S \frac{di_s}{dt} \qquad (2.3.1.5)$$

Para poder calcular el flujo del rotor es necesario resolver esta ecuación dentro de sus componentes reales e imaginarias.

El vector espacial del flujo del rotor en el cuadro de referencia estacionario (α,β) se obtiene al resolver las siguientes ecuaciones diferenciales [6].

$$\begin{aligned}\left[(1-\sigma)T_s + T_r\right]\frac{d\psi_{r\alpha}}{dt} &= \frac{L_m}{R_s}u_{s\alpha} - \psi_{r\alpha} - \omega_r Tr\,\psi_{r\beta} - \sigma L_m T_s\frac{di_{s\alpha}}{dt}\\ \left[(1-\sigma)T_s + T_r\right]\frac{d\psi_{r\beta}}{dt} &= \frac{L_m}{R_s}u_{s\beta} + \omega Tr\,\psi_{r\alpha} - \psi_{r\beta} - \sigma L_m T_s\frac{di_{s\beta}}{dt}\end{aligned} \qquad (2.3.1.6)$$

Estas a altas velocidades resultan ser independientes de la velocidad y el efecto en el cambio de temperatura es despreciable. Sin embargo a bajas velocidades pueden tenerse errores de consideración, debido al cambio en la resistencia del estator causada por la variación de la temperatura.

Es posible obtener una mayor precisión si se utiliza el valor de alta temperatura del estator, o si la temperatura es medida y un circuito de compensación es implementado, o utilizando un modelo térmico para corregir la resistencia del estator.

2.3.2 Circuito de desacople

Para propósitos de control vectorial por flujo orientado del rotor, la corriente i_{sd} en el eje directo (componente producida por el flujo del rotor) y la corriente del estator i_{sq} en el eje de cuadratura (componente producida por el par) deben ser controladas independientemente.

El acoplamiento es debido a que la componente del eje directo u_{sd} depende de i_{sq} y la componente del eje de cuadratura u_{sq} depende también de i_{sd}. Las componentes del eje del estator i_{sd} e i_{sq} se controlan independientemente (control desacoplado) si las ecuaciones del rotor están desacopladas y las componentes de corriente del estator i_{sd} e i_{sq} son indirectamente controladas al controlar el voltaje en las terminales del motor de inducción [6].

Las ecuaciones tienen la siguiente forma:

$$\begin{aligned}T'_S\frac{di_{sd}}{dt} + i_{sd} &= \frac{u_{sd}}{R_s} + \omega_{mr}T'_S\,i_{sq} - (T_s - T'_s)\frac{d\,|i_{mr}|}{dt}\\ T'_S\frac{di_{sq}}{dt} + i_{sq} &= \frac{u_{sq}}{R_s} - \omega_{mr}T'_S\,i_{sd} - (T_s - T'_s)\omega_{mr}\,|i_{mr}|\end{aligned} \qquad (2.3.2.1)$$

Las componentes de voltaje son las salidas de los controles las cuales controlan las componentes i_{sd} e i_{sq}. Estas se adicionan a las componentes de voltaje de desacople. De esta manera es posible obtener las componentes directa y de cuadratura de las terminales de salida de voltaje. Esto significa que las salidas de los controladores son:

$$\hat{u}_{sd} = R_s i_{sd} + L'_S \frac{di_{sd}}{dt} \qquad (2.3.2.2)$$

$$\hat{u}_{sq} = R_s i_{sq} + L'_S \frac{di_{sq}}{dt}$$

Y las componentes de desacople si se considera constante i_{mr} son:

$$u_{dd} = -\omega_{mr} L'_S \, i_{sq} \qquad (2.3.2.3)$$

$$u_{dq} = \omega_{mr} L'_S \, i_{sd} + (L_s - L'_S)\omega_{mr} \left| i_{mr} \right|$$

Donde la corriente de magnetización del rotor es:

$$i_{mr} = \frac{\psi_{r\psi r}}{L_m} \qquad (2.3.2.4)$$

Como se observa el algoritmo de desacople transforma el modelo no lineal del motor a ecuaciones lineales que pueden ser controladas por un PI general o un PID en lugar de controladores complicados.

2.4 DISEÑO Y SINTONIZACIÓN DE LOS CONTROLES CLÁSICOS PARA EL MOTOR DE INDUCCIÓN

El diseño y sintonización de los controles es una de las partes más importantes en la simulación e implementación del control vectorial del motor de inducción, además de ser una de las etapas más complejas a desarrollar. En esta sección se propone un esquema para sintonizar los controles clásicos usados en el control vectorial.

2.4.1 Diseño de los lazos de control de corriente

El control esta diseñado basado en el diagrama de bloques mostrado en la figura 2.4.1.1 [4]. Los voltajes u_{sq} y u_{sd} son generados basados en las siguientes ecuaciones, que son derivadas de las ecuaciones 2.3.2.1.

$$u_{sd} = (Salida\ PI) - \omega_{mr} L'_S\, i_{sq} \qquad (2.4.1.1)$$

$$u_{sq} = (Salida\ PI) + \omega_{mr} L'_S\, i_{sd} + (L_s - L'_S)\omega_{mr} \left| i_{mr} \right|$$

Las salidas de los controles PI son los voltajes de referencia. Los términos de acoplamiento son cancelados por el circuito de desacople, por lo que la respuesta de los lazos de corriente es descrita por:

$$Salida\ PI = R_s i_{sd} + L'_S \frac{di_{sd}}{dt} \qquad (2.4.1.2)$$

$$Salida\ PI = R_s i_{sq} + L'_S \frac{di_{sq}}{dt}$$

La estructura para los lazos de control de corriente se muestra en la figura 2.4.1.1 para los ejes d y q [4]. En el diagrama el retardo debido a la conmutación del inversor es despreciado. El diseño es valido, asegurando que el retardo del lazo de control de corriente es al menos 5 veces mayor que el lazo equivalente del inversor [42].

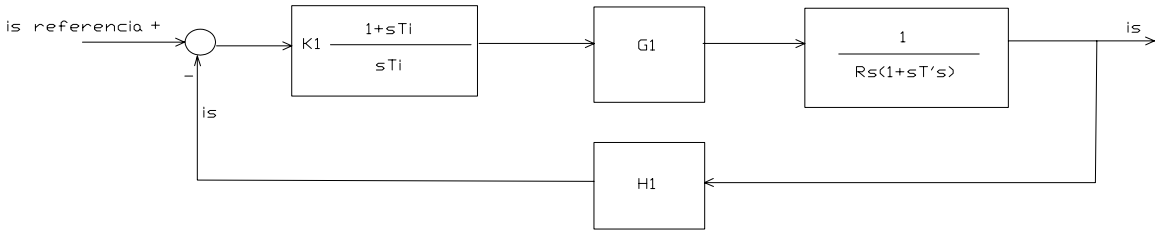


Figura 2.4.1.1 Lazo de control de corriente

Un método de simplificar el control es poner la constante de tiempo igual a la constante de tiempo transitoria del estator i.e., Ti=T´s y K1=KRd/G1H1. Con esto la respuesta en lazo cerrado del controlador actual se simplifica a un retardo de primer orden con T_{cl} =T's/K [42].

H1 y G1 son ganancias unitarias en este trabajo y pueden cambiar dependiendo de los parámetros del sistema.

En el trabajo se utiliza el valor K=2, es importante resaltar que es posible elegir un valor diferente para variar el valor del retardo de primer orden.

De esta forma se asegura la estabilidad para los lazos de corriente.

2.4.2 Diseño del control de velocidad

El diagrama a bloques del control de velocidad se muestra en la figura 2.4.2.1 [4], El control de velocidad se basa en este diagrama, [42].

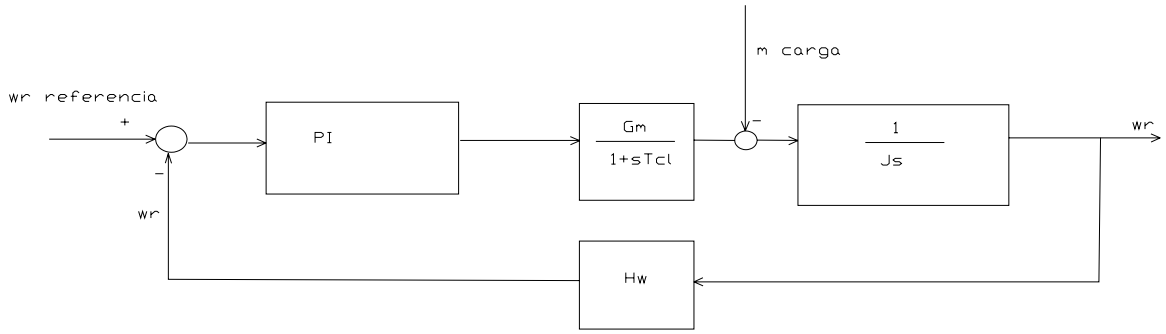


Figura 2.4.2.1 Lazo de control de velocidad.

Dado que la planta tiene un integrador y un retardo, la función de transferencia resultante en lazo cerrado tiene tres raices. La ecuación característica resulta:

$$s^3 JT_\omega T_{cl} + s^2 JT_\omega + sT_\omega K_\omega G_\omega H_\omega + K_\omega G_\omega H_\omega = 0 \qquad (2.4.2.1)$$

El control es posible diseñarlo forzando las tres raices reales e iguales [42].

$$K_\omega = (1/3)J / G_\omega H_\omega T_{cl}$$
$$T_\omega = 9T_{cl} \qquad (2.4.2.2)$$

2.4.3 Diseño del control de flujo

La estructura del lazo de control de flujo se muestra en la figura 2.4.3.1 [4]. La constante del control PI se escoge de tal forma que cancele el retardo Tr del rotor. T_{cl} es pequeña comparada con Tr y se desprecia. El diagrama de bloques del control de flujo se reduce a un retardo de primer orden de Tr/Kf, [42]. En este trabajo el valor usado para Kf es 4.

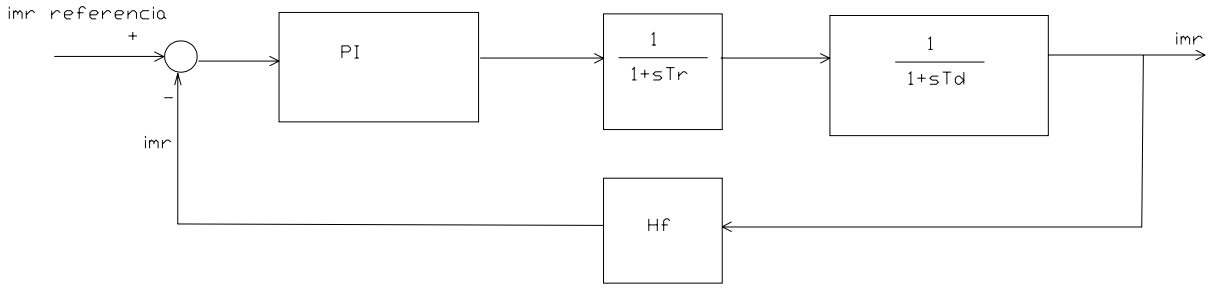


Figura 2.4.3.1 Lazo de control de flujo.

2.5 CONTROL DIFUSOS

Los controles difusos utilizan la señal de error y el cambio del error como entradas y el cambio en la acción como salida del control, se desarrollan controles difusos tipo Mamdani con tablas de 25 y 49 reglas, utilizando funciones de membresía tipo rodilla para los extremos de las señales de entrada y triangulares en la zona intermedia, para el cambio en la acción se utilizan todas las funciones tipo triangular, las funciones de membresía se proponen simétricas, se utiliza el mínimo para el operador "and" tanto en la premisa como para la implicación, para dedifusificación se emplea el centro de gravedad (COG).

2.6 OPTIMIZACIÓN UTILIZANDO BÚSQUEDA TABÚ

La optimización de la memoria asociativa difusa de los controles se desarrolla por medio de una Búsqueda Tabú, que utiliza una tabla de reglas codificada por medio de los índices utilizados para numerar las reglas, la memoria de la búsqueda se realiza por medio de estructuras que guardan la historia de la búsqueda, la historia es usada para evitar soluciones que se han encontrado en el pasado reciente además de utilizarse para diversificar la búsqueda negociando zonas difíciles.[1], [2], [11] y [15]. La búsqueda se realiza al evaluar subespacios del conjunto total de reglas por lo que es necesario incluir un valor de caos, el cual se utiliza para dejar que la búsqueda avance sin permitir valores que perjudiquen la búsqueda, hasta un valor especifico de iteraciones donde la búsqueda toma todas sus características [2].

CAPÍTULO 3

CONTROL DIFUSO

3.1 INTRODUCCIÓN

El control difuso ha demostrado tener buenos resultados con modelos lineales, no lineales y frente a variación de parámetros [4], para su implementación se requiere una base de reglas (un conjunto de reglas Si - Entonces), las cuales contienen una cuantificación difusa lógica de la descripción lingüística de los expertos de cómo lograr un buen control, un mecanismo de inferencia (también llamado inferencia difusa) o módulo que emula las decisiones del experto haciendo una interpretación y aplicando el conocimiento acerca de cómo realizar el mejor control de la planta, una interfase de difusificación, que convierte las entradas de los controles a información que el mecanismo de inferencia puede usar fácilmente para activar y aplicar reglas y finalmente una interfase de dedifusificación, que convierte las conclusiones del mecanismo de inferencia en las entradas actuales del proceso [3]. Estos elementos se muestran en la figura 3.1.

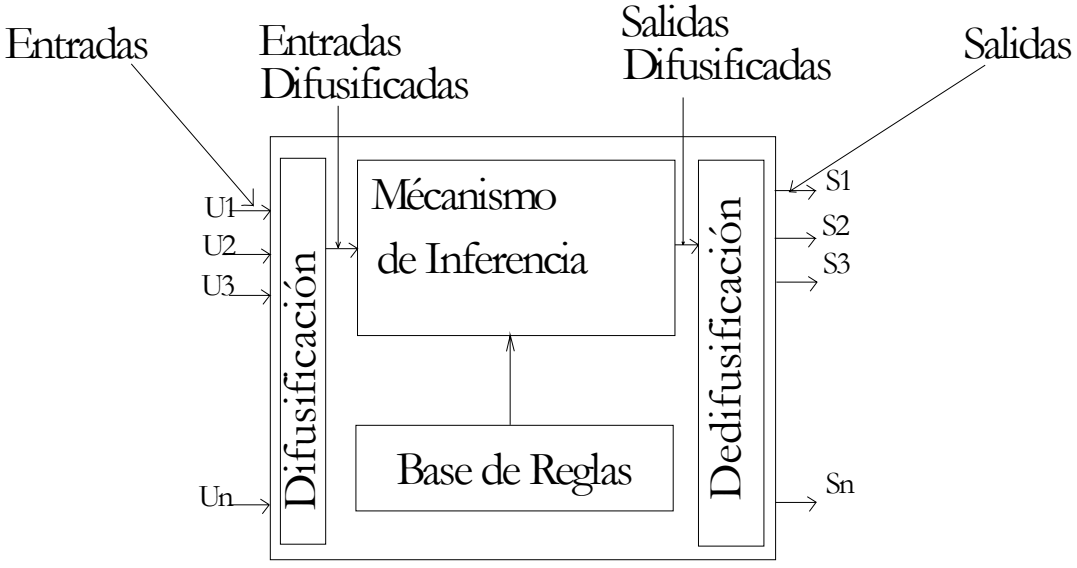


Figura 3. 1 Elementos del control difuso.

3.2 CONTROL DIFUSO PROPORCIONAL E INTEGRAL (PI)

La ecuación dada a un controlador convencional PI es:

$$u = K_p e + K_I \int e dt \qquad (3.2.\ 1)$$

Donde K_P y K_I son las constantes de proporcionalidad e integral respectivamente. Por lo tanto la ecuación anterior puede ser transformada a una expresión equivalente, con el propósito de eliminar la integral de error, quedando dicha expresión de la siguiente manera:

$$\dot{u} = K_p \dot{e} + K_I e \qquad (3.2.\ 2)$$

Con la ecuación anterior se considera a la e y Δe las variables de entrada, y la ú la variable de salida del control difuso PI.

Si e(k) es <propiedad difusa> y Δe(k) es <propiedad difusa> entonces Δu(k) es <acción>.

3.3 CONTROL DIFUSO PARA EL MOTOR DE INDUCCIÓN

En este trabajo se utilizan 4 controles, uno para la velocidad, uno para el flujo y dos para las corrientes como se observa en la figura 3.3.1, extraída del diagrama de bloques de la figura 2.1. Todos los controles se diseñaron con las mismas características por lo que se expone de forma general el control difuso utilizado en este trabajo para los cuatro controles.

Las entradas del control difuso son el error y el cambio del error, que se calculan de la siguiente manera:

$$\begin{aligned} error &= valor_{referencia} - valor_{real} \\ \Delta error &= error_{anterior} - error_{real} \end{aligned} \qquad (3.3.\ 1)$$

Donde:

$valor_{referencia}$ = Valor deseado o de referencia.

$Valor_{real}$ = Valor real medido del motor.

La salida del control difuso será la acción requerida o valor de referencia para el siguiente control.

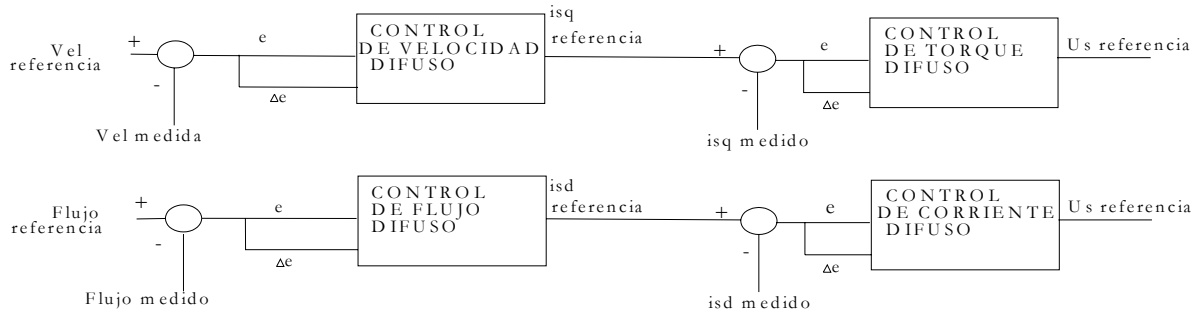


Figura 3.3. 1 Diagrama del control difuso para el motor de inducción.

Para crear la base de reglas se utiliza el desarrollo lingüístico para especificar un conjunto de reglas, que capturan el conocimiento del experto de cómo controlar el sistema, las reglas se usan de la siguiente manera:

Si e(k) es <propiedad difusa> y Δe(k) es <propiedad difusa> entonces Δu(k) es <acción>.

Que corresponde a una configuración de control proporcional integral difuso.

3.3.1 Tabla de reglas

La memoria asociativa difusa o base de reglas contiene el conocimiento del experto de cómo controlar el sistema [3]. Para este problema, con 2 entradas y 5 valores lingüísticos para cada uno de los controles, existen al menos $5^2 = 25$ posibles reglas (Todas las posibles combinaciones de la premisa lingüística para las dos entradas).

La tabla de reglas se genera a continuación. La simetría se propone como una tabla inicial para el inicio de la búsqueda.

e Δe	*Negativo grande*	*Negativo pequeño*	*Cero*	*Positivo pequeño*	*Positivo grande*
Negativo grande	Bajar mucho	Bajar mucho	Bajar mucho	Bajar poco	Nada
Negativo pequeño	Bajar mucho	Bajar mucho	Bajar poco	Nada	Subir poco
Cero	Bajar mucho	Bajar poco	Nada	Subir poco	Subir mucho
Positivo pequeño	Bajar poco	Nada	Subir poco	Subir mucho	Subir mucho
Positivo Grande	Nada	Subir poco	Subir mucho	Subir mucho	Subir mucho

Tabla 3.3.1. 1 Tabla de reglas, 5 funciones de membresía para e y 5 para Δe, la acción esta representada en las casillas de color.

De igual forma en este problema, con 2 entradas y 7 valores lingüísticos para cada uno de los controles, existen al menos $7^2 = 49$ posibles reglas (Todas las posibles combinaciones de la premisa lingüística para las dos entradas). La tabla de regla se genera a continuación utilizando las características de simetría como en el caso de 25 reglas.

e Δ*e*	*Negativo grande*	*Negativo mediano*	*Negativo pequeño*	*cero*	*Positivo pequeño*	*Positivo mediano*	*Positivo grande*
Negativo grande	Bajar Mucho	Bajar mucho	Bajar mucho	Bajar medianamente	Bajar poco	Bajar poco	Nada
Negativo mediano	Bajar mucho	Bajar mucho	Bajar medianamente	Bajar poco	Bajar poco	Nada	Subir poco
Negativo pequeño	Bajar mucho	Bajar medianamente	Bajar poco	Bajar poco	Nada	Subir poco	Subir poco
Cero	Bajar medianamente	Bajar poco	Bajar poco	Nada	Subir poco	Subir poco	Subir medianamente
Positivo pequeño	Bajar poco	Bajar poco	Nada	Subir poco	Subir poco	Subir medianamente	Subir mucho
Positivo mediano	Bajar poco	Nada	Subir poco	Subir poco	Subir medianamente	Subir mucho	Subir mucho
Positivo grande	Nada	Subir poco	Subir poco	Subir medianamente	Subir mucho	Subir mucho	Subir mucho

Tabla 3.3.1. 2 Tabla de reglas, 7 funciones de membresía para e y 7 para e', la acción esta representada en las casillas de color.

3.3.2 Funciones de membresía

Por otra parte es necesario cuantificar el conocimiento del experto de cómo controlar el sistema en forma abstracta, las funciones de membresía cuantifican el significado de los valores lingüísticos.

A continuación se muestran las funciones de membresía usadas en este trabajo donde el eje horizontal es llamado universo del discurso para la entrada e(t), dado que provee el rango de valores del error que pueden ser cuantificados con la lingüística y los conjuntos difusos. Se especifican ahora las funciones de membresía para los 5 valores lingüísticos, 5 para cada entrada y 5 para la salida. Los cuatro controles utilizan el mismo tipo de funciones.

El control se realiza con 3 funciones de membresía de tipo triangular y 2 tipo rodilla a los extremos para las entradas y 5 funciones tipo triangular para la salida del control, las cuales se grafican a continuación. Observe que en el caso de las funciones de membresía propuestas para la salida se tiene la habilidad de limitar el rango de la salida del controlador al centro de cada función de membresía de las orillas, esto se debe a que el centro de gravedad nunca calculara un valor fuera de este rango [3].

Negativo Grande Negativo pequeño cero Positivo pequeño Positivo grande .

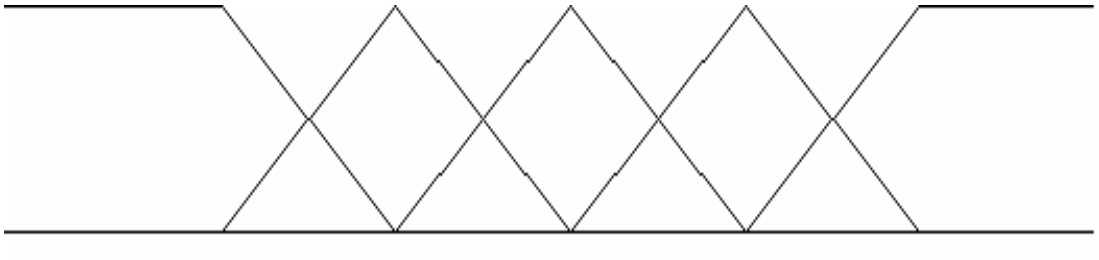


Figura 3.3.2. 1 Funciones de membresía para e(t).

Negativo Grande Negativo pequeño cero Positivo pequeño Positivo grande

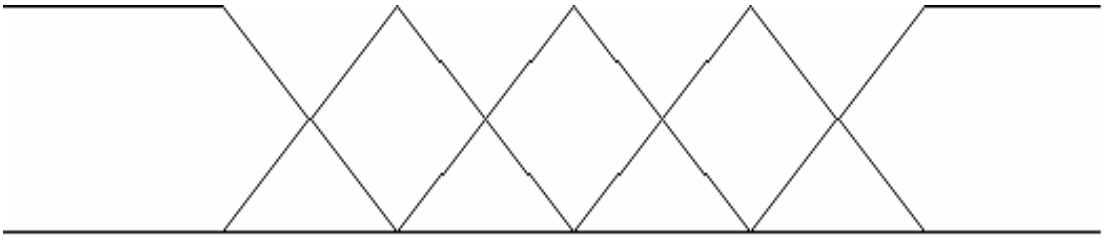


Figura 3.3.2. 2 Funciones de membresía para Δe(t)

Negativo Grande Negativo pequeño cero Positivo pequeño Positivo Grande .

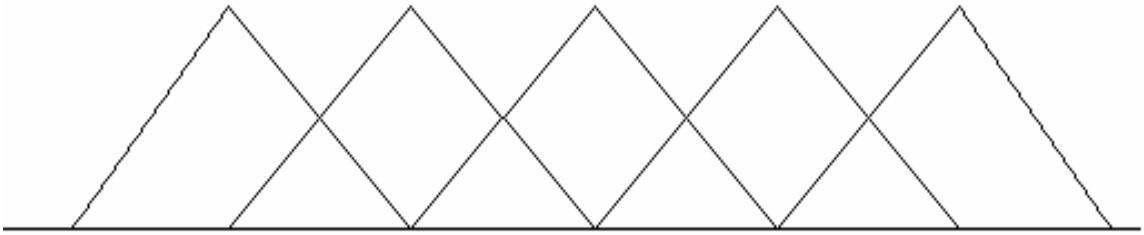


Figura 3.3.2. 3 Funciones de membresía para Δu(t).

Funciones de membresía para el control de velocidad difuso; Error e(t), figura 3.3.2.1; Cambio del error é(t), figura 3.3.2.2; Cambio en la salida de control Δu(t), figura 3.3.2.3.

A cada centro de las funciones de membresía se le es asignado un valor específico como se muestra en la figura 3.3.2.4. Para la entrada del error del control de velocidad se tiene:

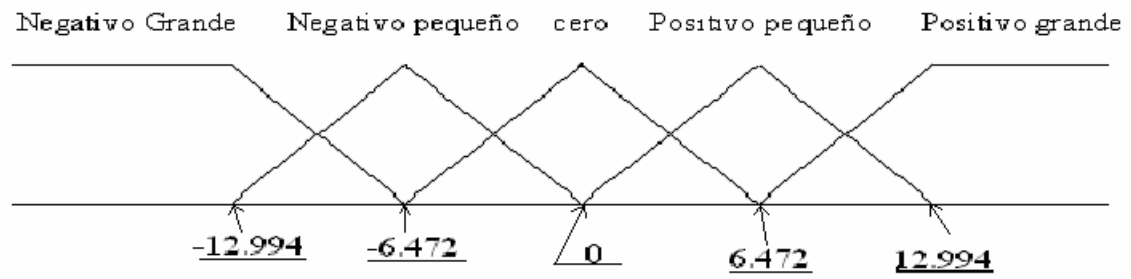


Figura 3.3.2. 4 Asignación de los centros a las funciones de membresía a la entrada de error del control de velocidad.

Los valores asignados a cada centro de las funciones de membresía para los controles difusos se presentan en las siguientes tablas:

Control de velocidad	Rangos para el error	Rangos para el cambio del error	Rangos para el cambio en la salida de control
Negativo grande	-12.994	-0.0406	-0.04509
Negativo pequeño	-6.472	-0.0203	-0.022545
Cero	0	0	0
Positivo pequeño	6.472	0.0203	0.022545
Positivo grande	12.994	0.0406	0.04509

Tabla 3.3.2. 1 Tabla de valores del control difuso de velocidad.

Control de flujo	Rangos para el error	Rangos para el cambio del error	Rangos para el cambio en la salida de control
Negativo grande	-0.16	-0.00002724	-0.00245
Negativo pequeño	-0.08	-0.00001362	-0.001225
Cero	0	0	0
Positivo pequeño	0.08	0.00001362	0.001225
Positivo grande	0.16	0.00002724	0.00245

Tabla 3.3.2. 2 Tabla de valores del control difuso de flujo.

Controles de corriente	Rangos para el error	Rangos para el cambio del error	Rangos para el cambio en la salida de control
Negativo grande	-40	-0.054	-0.034
Negativo pequeño	-20	-0.027	-0.017
Cero	0	0	0
Positivo pequeño	20	0.027	0.017
Positivo grande	40	0.054	0.034

Tabla 3.3.2. 3 Tabla de valores de los controles difusos de corriente.

Al utilizar estos valores fue necesario escalar los universos del discurso para obtener una mejor sintonización, a continuación se enlistan los controles con los respectivos universos del discurso que fueron escalados.

Control de flujo
Cambio del error = 0.1.
Control de velocidad
Error = 2.
Incremento en la salida de control = 1.012.

Todos los demás universos del discurso permanecen sin escalado.

3.3.3 Difusificación

Es el proceso de obtener un valor de una entrada variable (e.g. e(t)) encontrando el valor numérico de las funciones de membresía que están definidas para la variable. Por ejemplo para el control de velocidad si e(t)=1.5 y $\Delta e = 0.005$ los valores encontrados son:

$\mu_{\text{positivo pequeño}}(e(t))=1$ con los demás cero.

$\mu_{\text{cero}}(\Delta e(t))= \mu_{\text{positivo pequeño}}(\Delta e(t))=0.5$

3.3.4 Inferencia

El proceso de inferencia se puede dividir en 2 pasos [3].

1.- Las premisas de todas las reglas son comparadas con las entradas del controlador para determinar cual regla aplicar para la situación actual. Este proceso envuelve la determinación de la certeza de cada regla aplica, y típicamente se toma la que mayor fuerza tenga de acuerdo con la certeza de aplicarla a la situación actual.

Cuantificación de la premisa vía lógica difusa.

Observando las premisas siguientes:

Si el error es cero y cambio en el error es positivo pequeño entonces el cambio en la salida es positivo pequeño.

Lo principal en este paso es cuantificar el operador lógico "y" que combina el significado de los dos términos lingüísticos utilizando para este trabajo el Mínimo.

Utilizando de nuevo los valores definidos para el control de velocidad con e(t)= 0.75 y Δe(t)= 0.0025

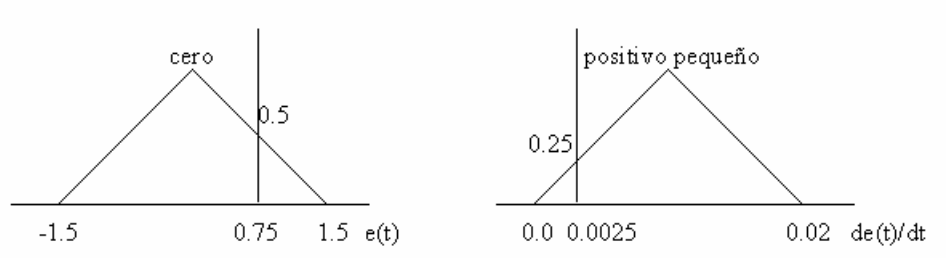


Figura 3.3.4. 1 Premisas de la regla.

$\mu_{cero}(e(t))=0.5$ y $\mu_{positivo\ pequeño}(e(t))=0.25$

Se utiliza el mínimo definido como:

Mínimo: Define $\mu_{premisa} = \min\{0.5,0.25\}=0.25$ es decir que es definido usando el mínimo de los dos valores de membresía.

Es posible utilizar otras definiciones para el operador lógico como el producto.

Determinación de que reglas están encendidas.

El mecanismo de inferencia busca que reglas están encendidas para encontrar cuales son relevantes en la situación actual combinando las recomendaciones de las reglas para obtener una conclusión.

En este caso e(t) = 0 y Δe(t) = 0.0075

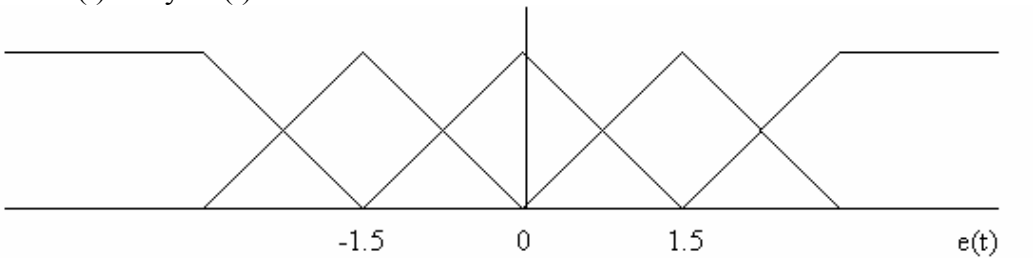


Figura 3.3.4. 2 Función de membresía μ_{cero} (e(t))=1

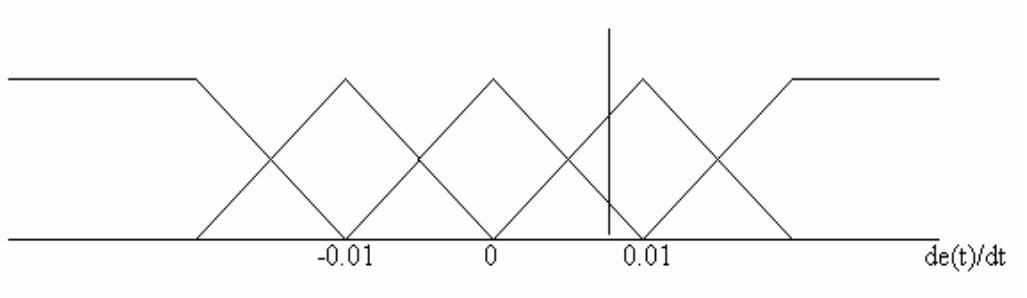


Figura 3.3.4. 3 Funciones de membresía $\mu_{cero}(\Delta e(t))=0.25$ y $\mu_{positivo\ pequeño}(\Delta e(t))=0.75$

Note que μ_{cero} (e(t))=1 y que las otras funciones de membresía para e(t) están apagadas es decir sus valores son cero. Para la entrada $\Delta e(t)$ se observa que $\mu_{cero}(\Delta e(t))=0.25$ y $\mu_{positivo\ pequeño}(\Delta e(t))=0.75$ y todas las demás están apagadas.

Esto implica que las premisas son:

"Error es cero"
"Cambio en el error es cero"
"Cambio en el error es positivo pequeño"

Usando la base de reglas encontramos que las reglas que están encendidas son:

Si el error es cero y el cambio en el error es cero entonces el cambio en la salida es cero.

Si el error es cero y el cambio en el error es positivo pequeño el cambio en la salida es positivo pequeño.

En este caso no se verán más de 4 reglas activadas al mismo tiempo y solo se tendrán 1,2 ó 4 reglas al mismo tiempo.

2.- La conclusión (¿Qué acciones de control se tomaran?), son determinadas usando las reglas que han sido determinadas a aplicar en el tiempo actual. Las conclusiones son caracterizadas con un conjunto difuso (o conjuntos) que representan la certeza de que la entrada a la planta podrá tomar varios valores.

Determinación de la conclusión.

Consiste en decidir que conclusión se aplicara con las reglas encendidas, para regresar al punto de operación. Para hacerlo se consideran las recomendaciones de cada regla independientemente. Después se combinan todas las recomendaciones de todas las reglas para determinar la acción.
Recomendación de una regla.

Si el error es cero y el cambio en el error es cero entonces el cambio en la salida es cero. (Regla #1)

$\mu_{premisa}(1)$=mínimo(0.25,1)=0.25

La conclusión se representa por:

$\mu(1)(\mu)$=mínimo$\{0.25,\mu_{cero}(\mu)\}$

Como se menciona anteriormente es posible utilizar el operador producto para representar la implicación.

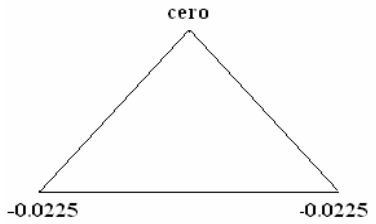


Figura 3. 3.4. 4. Función de membresía consecuente.

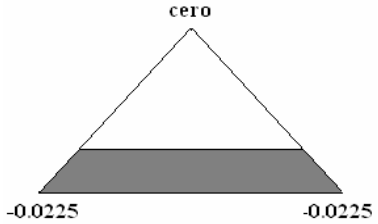


Figura 3.3.4. 5 Implicación del conjunto con la función de membresía $\mu(1)(\mu)$ para la regla 1.

Recomendación de la otra regla.

Si el error es cero y el cambio del error es positivo pequeño entonces el cambio en la salida es positivo pequeño. (Regla 2).

$\mu_{premisa}(2)$=mínimo(0.75,1)=0.75

Y la conclusión es:

$\mu(2)(\mu)$=mínimo$\{0.75,\mu_{positivo\ pequeño}(\mu)\}$

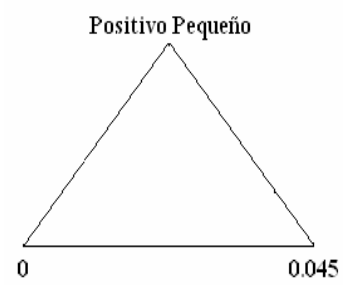


Figura 3.3.4. 6 Función de membresía consecuente.

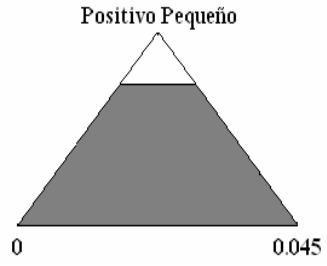


Figura 3.3.4. 7 Implicación del conjunto difuso con la función de membresía $\mu(2)(\mu)$ para la regla 2.

3.3.5 Dedifusificación

Es el proceso de convertir las decisiones en acciones y es la componente final del control difuso. Este paso trabaja en la implicación de los conjuntos difusos producidos por el mecanismo de inferencia y combinan sus efectos para proveer la mayor certeza en la salida del controlador (entrada al sistema).

La salida se denota como μ^{crisp} y desarrollando el proceso para las conclusiones determinadas anteriormente se tiene:

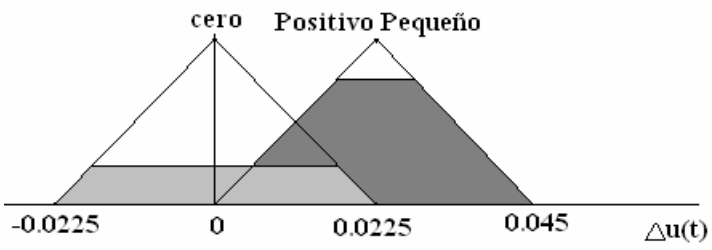


Figura 3.3.5. 1 Conjuntos difusos implicados.

En este trabajo se utiliza en "Centro de Gravedad" (COG), para combinar las recomendaciones representadas por los conjuntos difusos implicados de todas las reglas que se define de la siguiente forma:

$$u^{crisp} = \frac{\sum_i b_i \int u(i)}{\sum_i \int u(i)} \qquad (3.3.5.\ 1)$$

Esta es la forma clásica para calcular el centro de gravedad y en este caso utilizada para calcular el Centro de Gravedad de los conjuntos difusos implicados.

Como se ha descrito en este trabajo se usa el mínimo para representar el "and" en la premisa y la implicación y el Centro de Gravedad para dedifusificación.

Todos los elementos del control difuso anteriormente mencionados tienen las mismas características cuando se utiliza una base de 49 reglas.

3.4 SINTONIZACIÓN DE LOS CONTROLES DIFUSOS

Una parte esencial en el diseño de un control difuso es la sintonización de las funciones de membresía [3] y [10]. Existen tres formas de sintonizar los controles difusos, utilizando el conocimiento del experto, observando el comportamiento de la máquina o utilizando los resultados de los controles ya implementados.

En este trabajo se utilizan los valores generados por los controles PI clásicos para sintonizar las funciones de membresía de los controles difusos.

La sintonización inicial de las funciones de membresía de todos los controles difusos utilizados en este trabajo se realiza basada en la respuesta de los controles convencionales utilizados en el control vectorial, simulando disturbios en la carga para obtener el tamaño del universo del discurso donde deben permanecer las funciones de membresía.

3.4.1 Diseño del control de velocidad

El diseño del control de velocidad se basa en la observación de los resultados obtenidos con el control convencional de velocidad utilizando el control vectorial directo. Las pruebas se realizan con los datos de un motor de inducción de 3 hp, trifásico de cuatro polos, utilizando control vectorial por flujo orientado del rotor para mayor información sobre los datos del motor ver apéndice A.

Se utilizan los valores obtenidos de las gráficas de la simulación bajo disturbio en la carga y el archivo de datos generado para encontrar los valores del universo del discurso para las 5 y 7 funciones de membresía.

El sistema se encuentra inicialmente trabajando sin carga, hasta los 2 segundos donde se aplica la carga nominal, se deja que se estabilice el sistema y a los 4 segundos se retira la carga, este disturbio es el mismo para todas las gráficas y archivos de datos generados para la sintonización de los controles, este error se obtiene de simular al motor a la velocidad de 200 rad/s.

El error de velocidad se obtiene de restar el valor medido del valor deseado de la velocidad, la gráfica de la figura 3.4.1.1 muestra el valor máximo que se presenta en la simulación en este caso es de 2.752893 cuyo valor es usado para establecer el universo del discurso de las funciones de membresía utilizadas para el error.

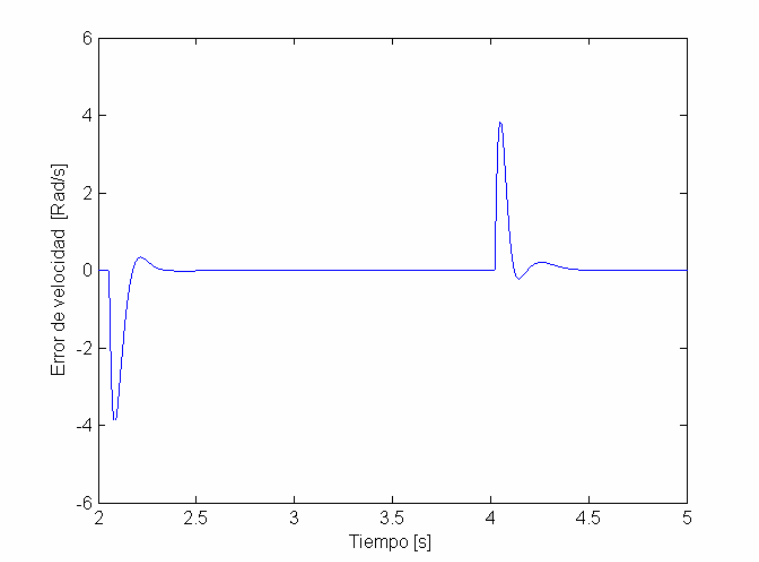


Figura 3.4.1. 1 Gráfica del error de velocidad.

El cambio en el error se obtiene de la observación del error en cada muestreo tomando en cuenta que el máximo cambio en el error se presentará cuando se inicia el disturbio, el valor se obtiene al restar el valor del error actual menos el error anterior y se obtiene del archivo de datos generado por la simulación en este caso se encontró un valor máximo de cambio de 0.016672 que es usado para definir el universo del discurso para el cambio del error.

La salida del control de velocidad es la corriente de cuadratura que se observa en la figura 3.4.1.2.

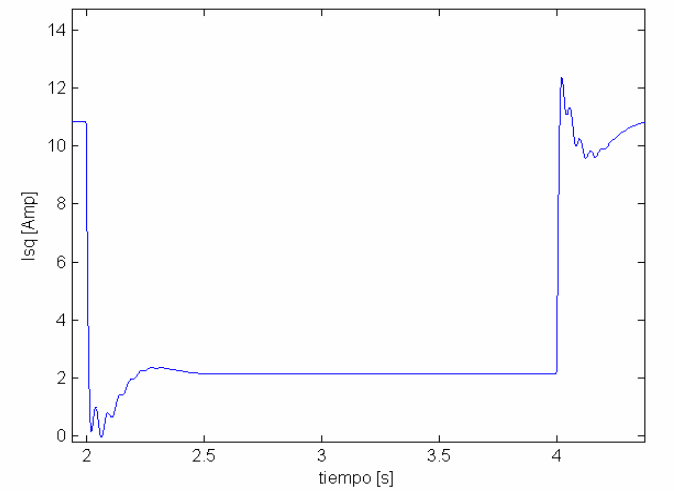


Figura 3.4.1. 2 Gráfica de la salida del control de velocidad i_{sq}

La acción es obtenida del archivo de datos generado por el programa suponiendo que el cambio máximo en la salida sucede cuando se encuentra el cambio del error máximo, en este caso el cambio máximo en la salida es de 0.054825 Amperes, este valor es utilizado para definir el universo del cambio en la salida de control.

3.4.2 Diseño del control de flujo

Primero se obtiene la gráfica del comportamiento del error en el valor del flujo.

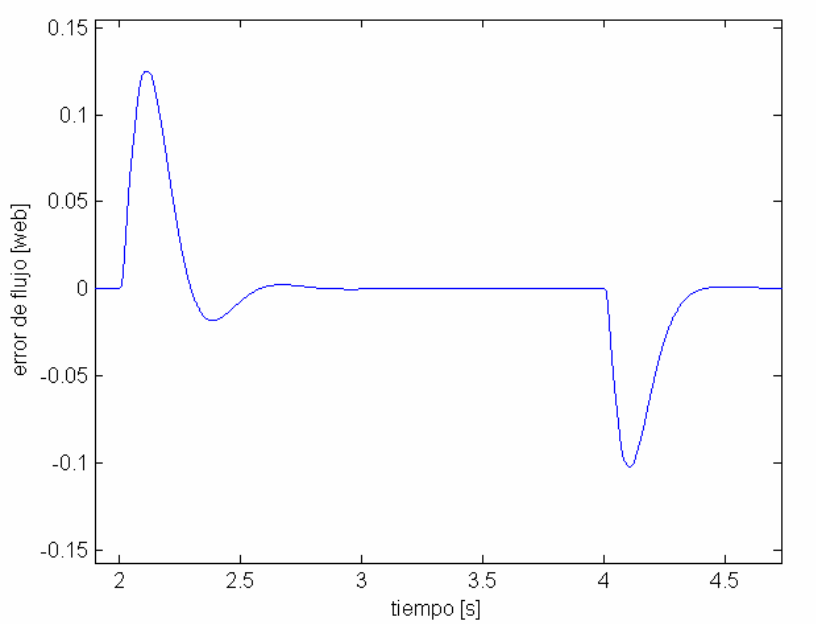


Figura 3.4.2. 1 Gráfica del error de flujo.

De la gráfica 3.4.2.1 que muestra la respuesta del error en el valor del flujo de donde se obtiene lo siguiente: el error máximo de flujo se obtiene en el pico de la señal del disturbio y se calcula restando el valor deseado del valor medido de flujo en ese instante, en este caso se obtiene un valor de 0.1248 de error máximo permitido por el control convencional en el esquema de control vectorial. El valor del cambio del error en el flujo se obtiene al inicio del disturbio y se toma el mismo valor tanto para el valor positivo como para el valor negativo, este se calcula con el archivo de datos generado por la simulación y se obtiene del valor actual menos el valor anterior y el resultado obtenido para este caso es de 0.000261. Estos valores son utilizados para definir el tamaño del universo del discurso para las entradas del control de flujo. Para obtener el cambio en la acción de control es necesario obtener la gráfica de la salida del control y el archivo generado por la simulación.

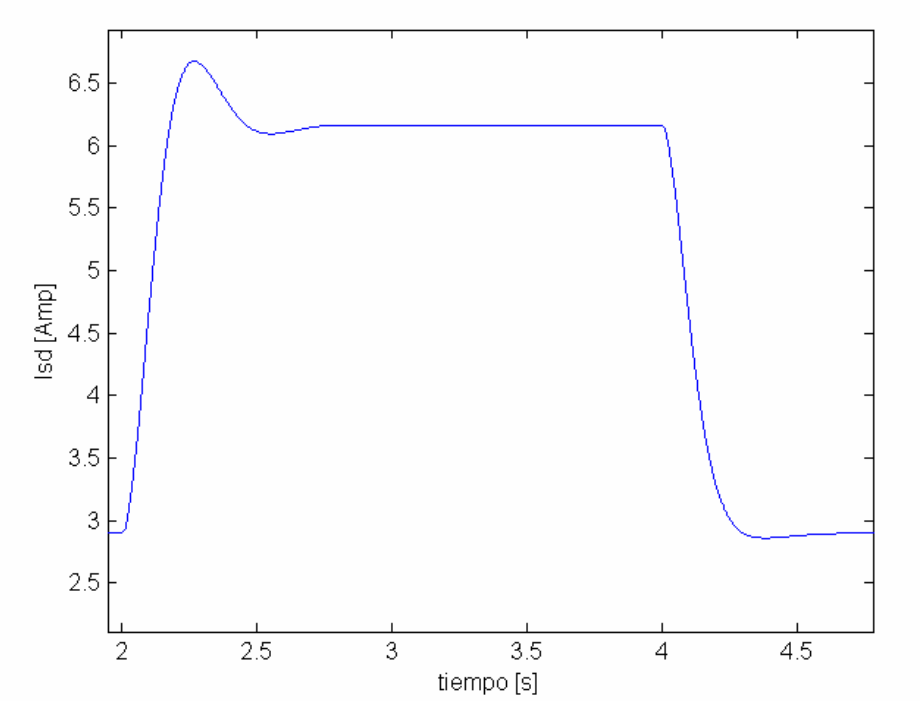


Figura 3.4.2. 2 Gráfica de la salida del control de flujo i_{sd}.

De la gráfica de la figura 3.4.2.2 y del archivo de datos generado por la simulación de la salida del control de flujo que corresponde a la señal de referencia i_{sd}, se obtiene el valor del cambio en la acción de control, es posible modificar el programa para obtener directamente el valor del cambio en la salida si se toma en cuenta que el cambio en la acción de control es el valor actual de la corriente i_{sd} menos el valor anterior, al realizar esta modificación se obtienen valores máximos para el cambio de la corriente de referencia i_{sd} de 0.00167 Amperes que es utilizado para definir el universo del discurso para el cambio en la acción de control.

3.4.3 Diseño de los controles de corriente

Debido a que en los controles convencionales de las corrientes en el control vectorial se utilizan las mismas constantes de control, se proponen los mismos universos del discurso para las funciones de membresía de los controles de corriente difusos.

A continuación se muestra la gráfica de la señal de referencia y el valor medido, recuérdese que el valor de referencia es la salida de los controles de velocidad y de flujo, por esto en la gráfica la señal de referencia no permanece constante. Las salidas de los controles de las corrientes son los voltajes en los ejes d y q.

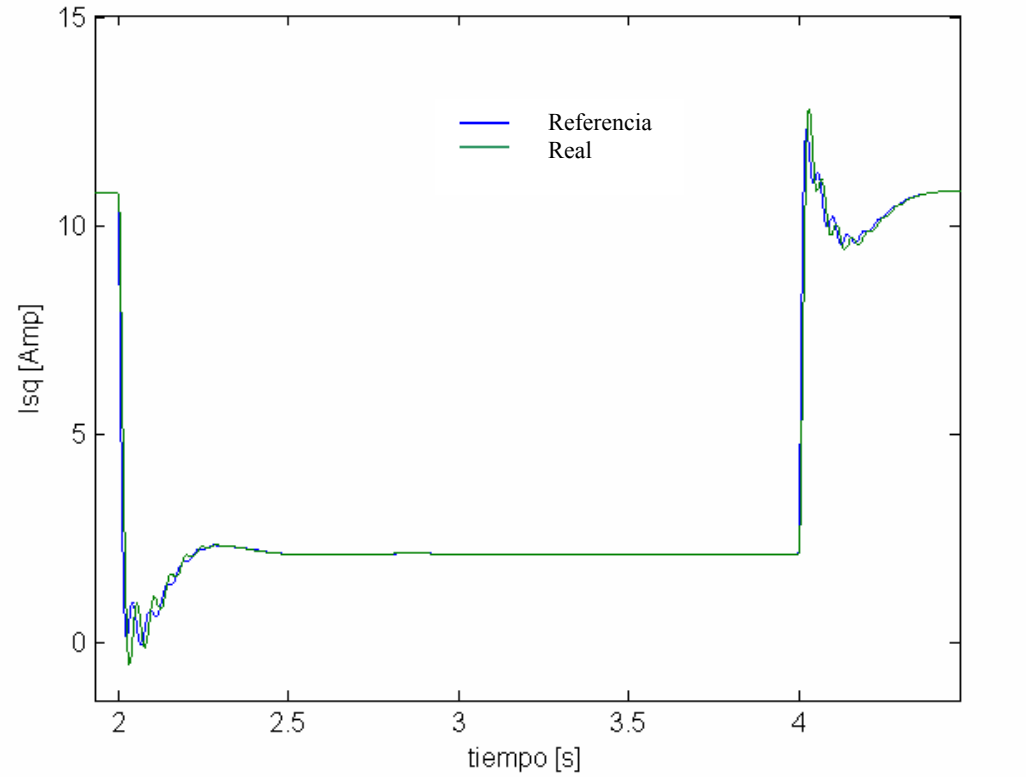


Figura 3.4.3. 1 Gráfica de los valores de i_{sq} de referencia e i_{sq} medida.

Como se observa en la gráfica de la figura 3.4.3.1 no se obtiene fácilmente el valor de error de la corriente i_{sq}, por lo que es preferible gráficar el error que se presenta en el lazo de control de la corriente utilizando el valor de referencia menos el valor medido del sistema, de esta manera puede observarse el error máximo que se presenta con la configuración actual del control.

La gráfica del error en el valor de i_{sq} se observa en la figura 3.4.3.2 de la cual podemos obtener el valor máximo del error y confirmándolo con el archivo de datos generado, para este caso tenemos un valor máximo del error de 3.2 Amperes, para encontrar el cambio en el error se observan los datos del archivo generado encontrando el valor de 0.05476.

La gráfica de la salida de control se muestra en la figura 3.4.3.3 y es utilizada para observar el comportamiento de la señal de control en el sistema, además de ayudar a obtener los valores para el universo del discurso que se utiliza en este trabajo en valor máximo encontrado es de 0.0482 Volts.

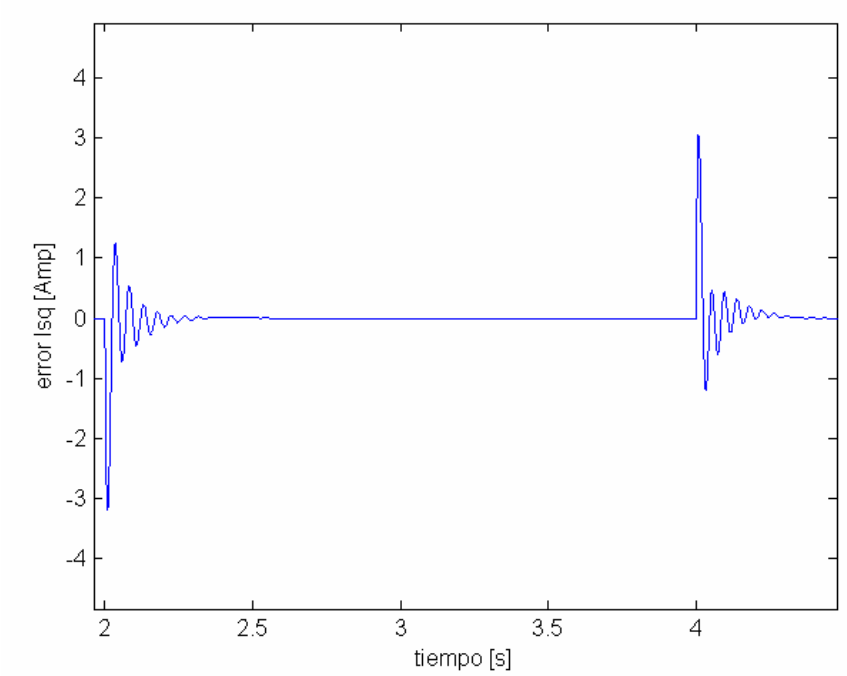


Figura 3.4.3. 2 Gráfica del error de i_{sq}.

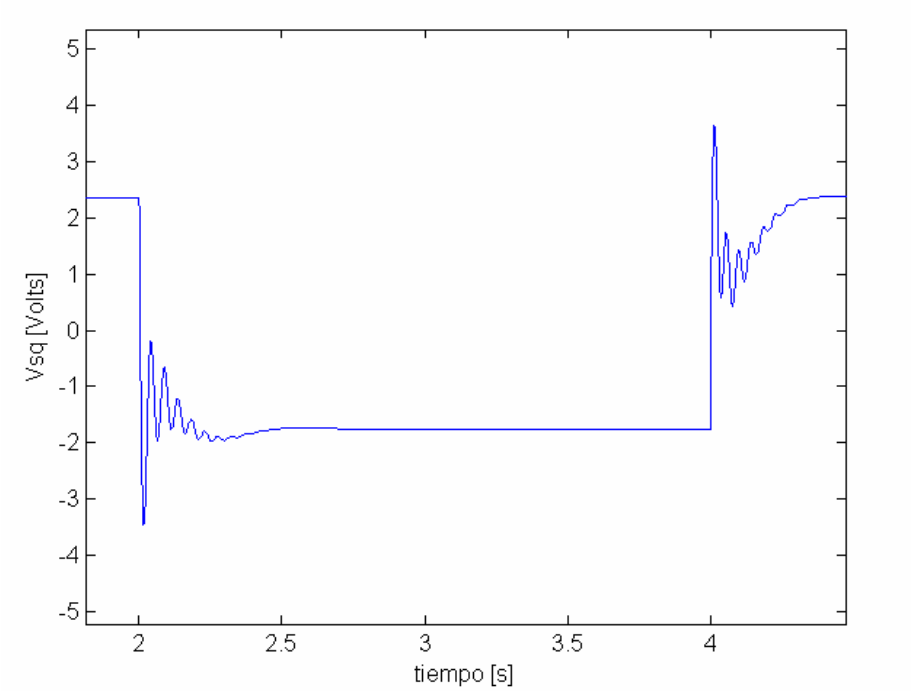


Figura 3.4.3. 3 Gráfica del comportamiento de la salida del control i_{sq}.

Con estos valores es posible definir los universos del discurso iniciales para todos los valores de entrada, salida de los 4 controles difusos, generando las tablas que contiene los datos de los centros de cada función de membresía para cada control. Después de realizar este proceso se recomienda realizar una sintonización fina observando el comportamiento de los controles y variando los universos del discurso.

3.4.4 Generación de las tablas de valores para las funciones de membresía

Las funciones de membresía se proponen simétricas, esto hace referencia a que si se obtuvo un error máximo de 3 en la gráfica del error de velocidad y se tienen 5 funciones de membresía y el valor asignado a la función de membresía negativo grande es de -3 y el valor de la función de membresía negativo pequeño es de -1.5, los valores correspondientes positivos de las funciones de membresía positivo pequeño y positivo grande serán respectivamente de 1.5 y 3 como se muestra en la figura 3.4.4.1. De esta forma es posible proponer los valores iniciales para todas las funciones de membresía.

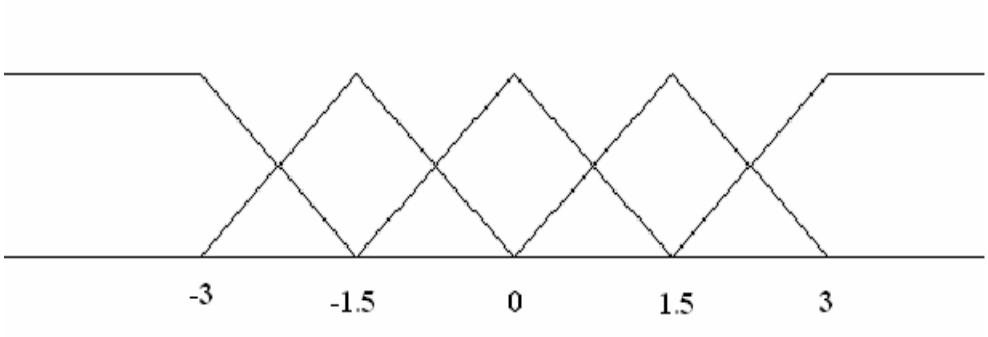


Figura 3.4.4. 1 Funciones de membresía para el error de velocidad.

Utilizando este procedimiento se desarrollaron las tablas para cinco funciones de membresía.

Control de velocidad	Rangos para el error	Rangos para el cambio del error	Rangos para el cambio en la salida de control
Negativo grande	-3	-0.02	-0.045
Negativo pequeño	-1.5	-0.01	-0.0225
Cero	0	0	0
Positivo pequeño	1.5	0.01	0.0225
Positivo grande	3	0.02	0.045

Tabla 3.4.4. 1 Tabla de valores del control difuso de velocidad.

Control de flujo	Rangos para el error	Rangos para el cambio del error	Rangos para el cambio en la salida de control
Negativo grande	-0.15	-0.00028	-0.0012
Negativo pequeño	-0.075	-0.00014	-0.0006
Cero	0	0	0
Positivo pequeño	0.075	0.00028	0.0006
Positivo grande	0.15	0.00014	0.0012

Tabla 3.4.4. 2 Tabla de valores del control difuso de flujo.

Controles de corriente	Rangos para el error	Rangos para el cambio del error	Rangos para el cambio en la salida de control
Negativo grande	-3.2	-0.056	-0.034
Negativo pequeño	-1.6	-0.028	-0.017
Cero	0	0	0
Positivo pequeño	1.6	0.028	0.017
Positivo grande	3.2	0.056	0.034

Tabla 3.4.4. 3 Tabla de valores de los controles difusos de corriente.

El caso de tener 7 funciones de membresía simétricas, y tomando en consideración el valor máximo del error obtenido para el lazo de control de flujo encontrado en la sección 3.4.2 y asignarlo a la función de membresía positivo grande es posible generar los valores de las otras funciones de membresía. El valor encontrado en la sección 3.4.2 es mayor al de la función positivo grande, esto se debe a que las tablas presentadas en este trabajo después de utilizar este método fueron sintonizadas manualmente para mejorar su comportamiento.

Los valores generados se presentan a continuación en la figura 3.4.4.2

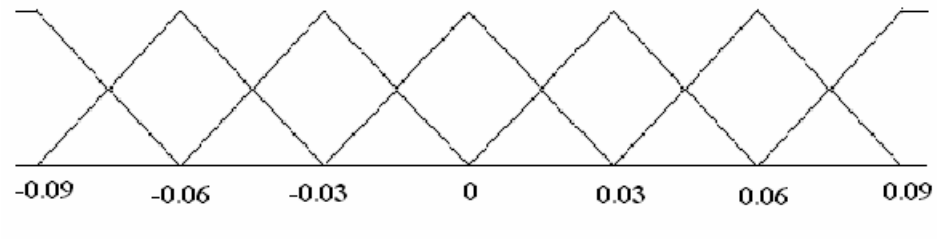


Figura 3.4.4. 2 Funciones de membresía para el error de flujo.

Del mismo modo se generan las tablas de valores para 7 funciones de membresía presentadas a continuación.

Control de velocidad	Rangos para el error	Rangos para el cambio del error	Rangos para el cambio en la salida de control
Negativo grande	-5.31	-0.021	-0.045
Negativo mediano	-3.54	-0.014	-0.030
Negativo pequeño	-1.77	-0.007	-0.015
Cero	0	0	0
Positivo pequeño	1.77	0.007	0.015
Positivo mediano	3.54	0.014	0.030
Positivo grande	5.31	0.021	0.045

Tabla 3.4.4. 4 Tabla de valores del control difuso de velocidad

Control de flujo	Rangos para el error	Rangos para el cambio del error	Rangos para el cambio en la salida de control
Negativo grande	-0.09	-0.00027	-0.00135
Negativo mediano	-0.06	-0.00018	-0.00090
Negativo pequeño	-0.03	-0.00009	-0.00045
Cero	0	0	0
Positivo pequeño	0.03	0.00009	0.00045
Positivo mediano	0.06	0.00018	0.00090
Positivo grande	0.09	0.00027	0.00135

Tabla 3.4.4. 5 Tabla de valores del control difuso de flujo.

Controles de corriente	Rangos para el error	Rangos para el cambio del error	Rangos para el cambio en la salida de control
Negativo grande	-0.135	-0.06	-0.0375
Negativo mediano	-0.090	-0.04	-0.025
Negativo pequeño	-0.045	-0.02	-0.0125
Cero	0	0	0
Positivo pequeño	0.045	0.02	0.0125
Positivo mediano	0.090	0.04	0.025
Positivo grande	0.135	0.06	0.0375

Tabla 3.4.4. 6 Tabla de valores de los controles difusos de corriente.

Es importante mencionar que este método no asegura un comportamiento adecuado del sistema pero si provee valores iniciales para las funciones de membresía y los universos del discurso. Con estos valores es posible sintonizar los controles teniendo valores iniciales.

CAPÍTULO 4

BÚSQUEDA TABÚ Y LÓGICA DIFUSA

4.1 INTRODUCCIÓN

La Búsqueda Tabú tiene sus antecedentes en métodos diseñados para cruzar cotas de factibilidad u optimalidad local tratadas como barreras en procedimientos clásicos, e imponer y eliminar cotas sistemáticamente para permitir la exploración de regiones no consideradas en otro caso [1]. El nombre y la terminología de Búsqueda Tabú vienen de una característica distintiva de este procedimiento, es el uso de memoria adaptable y de estrategias especiales de resolución de problemas.

Búsqueda Tabú es el origen del enfoque basado en memoria y estrategia intensiva en la literatura de las metaheurísticas, en contraposición con los métodos que no tienen memoria o que sólo usan una débil memoria basada en herencia. Búsqueda Tabú es también responsable de enfatizar el uso de los diseños estructurados para explotar los patrones históricos de la búsqueda, de forma opuesta a los procesos que confían casi exclusivamente en la aleatorización.

Los principios fundamentales de la Búsqueda Tabú fueron elaborados en una serie de artículos a finales de los años 80 y principios de los 90, y han sido unificados en el libro "Tabú Search" [1]. El destacable éxito de la Búsqueda Tabú para resolver problemas de optimización difíciles (especialmente aquellos que surgen en aplicaciones del mundo real) ha causado una explosión de nuevas aplicaciones de Búsqueda Tabú durante los últimos años.

La filosofía de la Búsqueda Tabú es derivar y explotar una colección de estrategias inteligentes para la resolución de problemas, basadas en procedimientos implícitos y explícitos de aprendizaje. El marco de memoria adaptable de Búsqueda Tabú no sólo explota la historia del proceso de resolución del problema, sino que también exige la creación de estructuras para hacer posible tal explotación.

La historia de resolución del problema se extiende a la experiencia ganada tras resolver múltiples instancias de una clase de problema usando Búsqueda Tabú. Las estructuras de memoria de la Búsqueda Tabú funcionan mediante referencia a cuatro dimensiones principales, consistentes en

la propiedad de ser reciente, en frecuencia, en calidad, y en influencia. Estas dimensiones se fijan contra unos antecedentes de conectividad y estructuras lógicas.

Para poder utilizar la Búsqueda Tabú en este trabajo hay que recordar que los controles difusos utilizan una base de reglas y el primer problema es definir esta tabla lingüística [1] y [15].

Sin embargo no es posible utilizar la base de reglas lingüística como tal, por lo que para optimizar la base de reglas de los controles difusos es necesario, transformar el problema de la definición de la tabla lingüística a un problema de optimización, ya sea de cambios o de transformaciones.

En la Búsqueda Tabú la función objetivo f(x), donde $x \in X$, se evalúa de forma ordinaria. Pero el espacio de búsqueda X esta dividido en vecindades, una vecindad N(x) esta formada por los elementos alrededor del punto seleccionado x [1], [11] y [15]. Este punto y su vecindad son parte de una posible solución, la vecindad es de tamaño n, donde n es el posible número de soluciones x dentro de la vecindad N(x). La vecindad tiene que estar dentro del espacio de búsqueda $N(x) \subset X$, y cada solución x tiene una vecindad asociada.

El proceso de búsqueda va de una solución a otra hasta satisfacer un criterio determinado. La mejor de la solución de la vecindad x', es la solución x que obtuvo el mejor resultado en la evaluación de la función objetivo durante la búsqueda, cada solución x que consigue ser la mejor solución de la vecindad x' lo hace por medio de una operación que se llama movimiento [1], [2], [11] y [15].

La Búsqueda Tabú utiliza memoria a corto plazo y a largo plazo [1], [2], [11], [15] y [21]. La memoria a corto plazo en Búsqueda Tabú tiene por objeto identificar aquellas soluciones que se han visitado en el pasado reciente y darles un atributo tabú el cual no permite que las soluciones vuelvan a ser utilizadas por un corto periodo de tiempo, mientras que la memoria de largo plazo trabaja en un horizonte mayor es decir guarda la historia de la búsqueda para poder diversificarla o intensificarla, dando penalizaciones a aquellas regiones que se han cambiado o participado en la búsqueda, durante todo el proceso.

La Búsqueda Tabú es entonces una búsqueda basada en las vecindades o en los entornos, que utiliza los movimientos para elegir sitios donde continuar el procedimiento, creando nuevas soluciones, dependiendo del entorno donde se realiza la búsqueda. La búsqueda desarrollada propone mantener la estructura de la base de reglas, manteniendo la lógica de esta a diferencia de los trabajos donde la base de reglas se modifica para poder ser utilizada en el proceso de búsqueda poniéndola en forma de listas o definiendo de manera especial el espacio de búsqueda, haciendo las vecindades no simétricas. En los siguientes parrafos encontrará la definición de la función objetivo, como se utiliza la tabla de reglas en el presente trabajo, y la descripción general de la Búsqueda Tabú desarrollada en este trabajo.

4.2 FUNCIÓN OBJETIVO

La función objetivo se presenta en forma matemática, con esta función se pueden determinar los valores mínimos o máximos, de algún o algunos parámetros con ciertos límites dados [21], son también llamados parámetros de decisión y los limites del problema son las restricciones de la solución [1], [11], [15] y [21]. Los límites definen los valores que no pueden ser evaluados por la función objetivo, estos límites deben ser expresados en los parámetros de decisión. Algunos límites se representan como igualdades o desigualdades [13].

La función objetivo en este trabajo esta compuesta por el error de velocidad al cuadrado y el cambio del error de velocidad al cuadrado.

$$\text{Función Objetivo} = \sum (\text{error}^2 + \Delta\text{error}^2). \qquad (4.2.1)$$

La función objetivo es utilizada cada vez que un movimiento en la vecindad ha sido desarrollado. La vecindad puede ser modificada dependiendo de la trayectoria de la búsqueda y algunos elementos pueden ser tabú, en estos casos el resultado de la función objetivo no tendrá efecto al elegirse las soluciones, siempre y cuando no se cumpla el criterio de aspiración.

4.3 ESQUEMA DE LA BÚSQUEDA TABÚ

El esquema de optimización por Búsqueda Tabú, se implementa desarrollando diferentes bloques que contienen las características del procedimiento, estos utilizan estructuras especiales para guardar la historia de la búsqueda, estas estructuras sirven para crear atributos y penalizaciones sobre la memoria a corto plazo y a largo plazo, con el propósito de diversificar o intensificar la búsqueda. El diagrama de flujo del esquema de Búsqueda Tabú propuesto se muestra en la figura 4.3.1.

El apéndice C contiene el programa desarrollado para realizar la simulación del sistema con una base de 25 reglas, para los diferentes disturbios.

Las funciones desarrolladas se describen brevemente a continuación y se basan en las estructuras mostradas en la sección 4.6, mientras que las características que presenta este programa se observan en la sección 4.7.

Las estructuras utilizan memorias asociativas difusas simplificadas mostradas en la sección 4.4 para realizar los movimientos, las evaluaciones de la vecindad de la regla, y elegir las soluciones encontradas por la Búsqueda Tabú.

Las vecindades de búsqueda son definidas en la sección 4.5 donde se presentan todas las posibles vecindades para la memoria asociativa difusa utilizada en este trabajo.

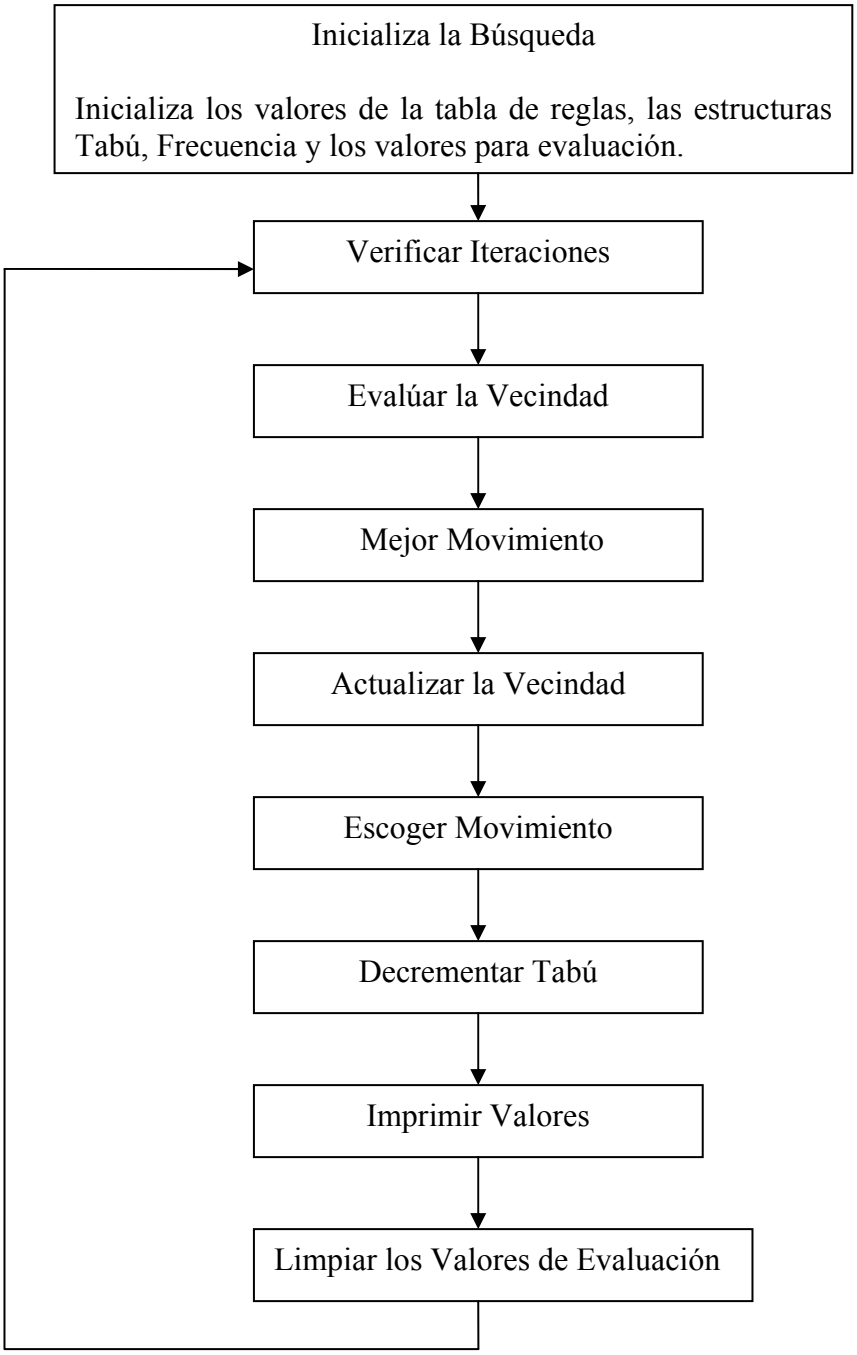


Figura 4.3.1 Diagrama de flujo del esquema de Búsqueda Tabú.

Inicializar la Búsqueda. Inicializa los valores de la tabla lingüística de forma simétrica, inicializa la estructura tabú, la estructura de frecuencia y la tabla de valores de evaluación con ceros.

Evaluar la Vecindad. Identifica el elemento y el tipo de vecindad a evaluar, evalúa la vecindad del elemento utilizando la función objetivo y transformando la base de reglas del sistema de control vectorial difuso, la forma de evaluación se realiza con cambios o transformaciones, cambiando el valor de cada vecindad por el valor del elemento o solución actual en otras palabras transformando el valor de la vecindad por el valor del elemento actual. Al terminar de evaluar cada vecindad regresa el valor que poseía la tabla antes de realizar la transformación, por lo que este bloque no afecta las vecindades de la tabla de reglas de forma permanente. La evaluación de las vecindades se guarda en una lista de candidatos, esta lista contiene los valores de los resultados de la evaluación de la función objetivo de cada vecindad y los atributos y penalizaciones de las vecindades.

Mejor Movimiento. Elige la siguiente vecindad a evaluar y el mejor movimiento encontrado, utilizando los valores generados por el bloque de evaluación contenidos en la lista de candidatos, tomando en cuenta el atributo tabú y las penalizaciones de frecuencia además del valor de caos. La mejor transformación no es elegida para continuar la búsqueda, pues esta se convierte tabú para atributos de memoria de corto plazo.

Actualizar la Vecindad. Da de alta los movimientos elegidos, la estructura tabú y frecuencia. Cambia el valor del elemento de la tabla de reglas, por el mejor encontrado por el bloque Mejor Movimiento, actualiza el valor tabú para el atributo de memoria a corto plazo con el valor especificado de dos iteraciones e incrementa el valor de frecuencia en uno, debido al cambio en la vecindad.

Escoger Movimiento. Usa los valores encontrados por el bloque mejor movimiento para obtener los nuevos valores para continuar la búsqueda, es decir decodifica la información de la lista de candidatos del mejor valor encontrado para utilizarla en la base de reglas.

Decrementar Tabú. Este bloque decrementa en uno todos los valores utilizados por la búsqueda en el pasado reciente de la estructura tabú, cada iteración los valores se decrementan en uno hasta llegar al valor de cero, para que se les permita participar de nuevo en la búsqueda.

Imprimir Valores. Imprime en pantalla los valores encontrados por la búsqueda tanto del mejor movimiento encontrado como el del elegido para continuar la búsqueda, además de imprimir en pantalla los valores actualizados de la estructura tabú para cada elemento y los valores de frecuencia de cada regla.

Limpiar los Valores de Evaluación. Pone a cero todos los valores de la lista de candidatos para ser usados en la siguiente iteración.

Para poder llevar acabo la implementación en simulación de la Búsqueda Tabú es necesario simplificar la memoria asociativa difusa a una forma que el ordenador pueda trabajar, además de definir las vecindades a utilizar en la búsqueda ya que dependiendo de la posición del elemento que se elige para evaluar la vecindad, esta vecindad tendrá una forma y un tamaño especifico, es necesario construir las estructuras necesarias para el funcionamiento del esquema de optimización. A continuación se presenta la forma propuesta en este trabajo para la utilización de la Búsqueda Tabú, y las características de esta en la optimización de la base de reglas de los controles vectoriales difusos.

4.4 SIMPLIFICANDO LA TABLA DE REGLAS UTILIZANDO SUBÍNDICES

La simplificación de la memoria asociativa difusa se basa en la proposición de índices que definan cada premisa o conclusión, en este trabajo se propone empezar con una numeración desde cero, por ejemplo para cinco funciones de membresía cero será Negativo grande, 1 corresponde a Negativo pequeño, 2 a cero y así sucesivamente. Esto se hace para facilitar la programación del algoritmo y convertir la tabla de reglas en una matriz.

Con la propuesta anterior es posible definir las premisas y las consecuencias de la siguiente manera:

Para el error y el cambio en el error.

0	Negativo Grande	NG
1	Negativo Pequeño	NP
2	Cero	C
3	Positivo Pequeño	PP
4	Positivo Grande	PG

Para la consecuencia.

0	Bajar mucho	BM
1	Bajar Poco	BP
2	Nada	N
3	Subir Poco	SP
4	Subir Mucho	SM

La tabla de reglas propuesta con 25 reglas se muestra a continuación:

e Δe	*Negativo grande*	*Negativo pequeño*	*Cero*	*Positivo pequeño*	*Positivo grande*
Negativo grande	Bajar mucho	Bajar mucho	Bajar mucho	Bajar poco	Nada
Negativo pequeño	Bajar mucho	Bajar mucho	Bajar poco	Nada	Subir poco
Cero	Bajar mucho	Bajar poco	Nada	Subir poco	Subir mucho
Positivo pequeño	Bajar poco	Nada	Subir poco	Subir mucho	Subir mucho
Positivo Grande	Nada	Subir poco	Subir mucho	Subir mucho	Subir mucho

Tabla 4.4. 1 Tabla de 25 reglas.

Utilizando los índices mencionados para numerar las premisas y las consecuencias, la tabla de reglas puede representarse de forma simplificada, como se muestra en la tabla 4.4.2 si se elimina la primera fila y la primera columna que solo son las etiquetas del error y el cambio del error.

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	4	4
2	3	4	4	4

Tabla 4.4. 2 Tabla simplificada de 25 reglas.

Repitiendo el procedimiento para 7 funciones de membresía.

Para el error y el cambio en el error.

0	Negativo Grande	NG
1	Negativo Mediano	NM
2	Negativo Pequeño	NP
3	Cero	C
4	Positivo Pequeño	PP
5	Positivo Mediano	PM
6	Positivo Grande	PG

Para el cambio en la acción.

0	Bajar Mucho	BMO
1	Bajar Medianamente	BME
2	Bajar Poco	BP
3	Nada	N
4	Subir Poco	SP
5	Subir Medianamente	SME
6	Subir Mucho	SMO

La tabla de reglas propuesta con 49 reglas se muestra a continuación:

e Δe	*Negativo grande*	*Negativo mediano*	*Negativo pequeño*	*cero*	*Positivo pequeño*	*Positivo mediano*	*Positivo grande*
Negativo grande	Bajar Mucho	Bajar mucho	Bajar mucho	Bajar medianamente	Bajar poco	Bajar poco	Nada
Negativo mediano	Bajar mucho	Bajar mucho	Bajar medianamente	Bajar poco	Bajar poco	Nada	Subir poco
Negativo pequeño	Bajar mucho	Bajar medianamente	Bajar poco	Bajar poco	Nada	Subir poco	Subir poco
Cero	Bajar medianamente	Bajar poco	Bajar poco	Nada	Subir poco	Subir poco	Subir medianamente
Positivo pequeño	Bajar poco	Bajar poco	Nada	Subir poco	Subir poco	Subir medianamente	Subir mucho
Positivo mediano	Bajar poco	Nada	Subir poco	Subir poco	Subir medianamente	Subir mucho	Subir mucho
Positivo grande	Nada	Subir poco	Subir poco	Subir medianamente	Subir mucho	Subir mucho	Subir mucho

Tabla 4.4. 3 Tabla de 49 reglas.

Utilizando los índices del mismo modo que en el caso de 25 reglas para numerar las premisas y las consecuencias, la tabla de reglas puede representarse como se muestra en la tabla 4.4.4 si se elimina la primera fila y la primera columna que solo son las etiquetas del error y el cambio del error.

0	0	0	1	1	2	3
0	0	1	1	2	3	4
0	1	1	2	3	4	5
1	1	2	3	4	5	5
1	2	3	4	5	5	6
2	3	4	5	5	6	6
3	4	5	5	6	6	6

Tabla 4.4. 4 Tabla de 49 reglas simplificada.

En adelante solo se utilizara la forma simplificada de la tabla de reglas.

La Búsqueda Tabú puede ser convenientemente caracterizada mediante referencia a la búsqueda en el entorno, aunque se insiste en que la búsqueda en el entorno tiene un significado más amplio en la Búsqueda Tabú que en algunas otras partes de la literatura de las metaheurísticas. Por ejemplo, Búsqueda Tabú incluye procedimientos constructivos y destructivos entre los procesos que dirige por memoria adaptable, mientras que tales procedimientos y sus combinaciones son a menudo excluidos de la definición de búsqueda en el entorno en otros enfoques.

4.5 DEFINICIÓN DE LA VECINDAD EN LA BÚSQUEDA TABÚ

Una representación conveniente de búsqueda en el entorno, identifica para cada solución $x \in X$, un conjunto asociado de vecinos, $N(x) \subset X$, llamado entorno de x. En Búsqueda Tabú, los entornos normalmente se asumen simétricos, es decir, x' es un vecino de x si y sólo si x es un vecino de x'.

Para poder aplicar lo anterior se observa que tenemos una tabla de nxn (tabla de reglas simplificada), donde n corresponde al número de particiones difusas, de aquí es posible manejar la base de reglas como una matriz, y cada regla tiene una vecindad lingüística N(x) donde las soluciones x dentro de la vecindad son las acciones lingüísticas.

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	4	4
2	3	4	4	4

Tabla 4.5. 1 Vecindad de la regla 2,2.

En la tabla 4.5.1 se observa el tamaño máximo de la vecindad que en este trabajo es posible obtener suponiendo que la vecindad solo esta conformada por un nivel de acciones alrededor de la regla, cada regla tiene una posición (i, j) en donde i corresponde al renglón y j a la columna, así que cada acción tiene una vecindad de tamaño fijo dependiendo de la posición de la conclusión de la regla, los tamaños de la vecindad solo pueden caer en los valores 3 cuando están en las esquinas de la matriz, 5 en cualquiera de las orillas y 8 cuando están lejos de estas.

4.5.1 Vecindad caso 1

Las reglas que pertenecen al caso 1, son aquellas que se encuentran lejos de las orillas o de las esquinas de la matriz, si se toma en cuenta que se tiene una matriz de n x n donde n es el número de renglones y es igual al número de columnas (matriz cuadrada), en este trabajo en la numeración de las funciones de membresía se utilizó el cero, por lo que los subíndices de los elementos incluyen el cero, considerando esto las reglas que pertenecen al primer caso se define como aquellos elementos:

$$\text{Reglas caso 1} = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n-1} \\ \vdots & \ddots & \vdots \\ a_{n-1,1} & \cdots & a_{n-1,n-1} \end{pmatrix}$$

Recuerde que se utiliza el elemento 0,0 como parte de la matriz por lo que en este caso el elemento $a_{0,0}$ es una esquina, por este motivo la submatriz para el caso 1 empieza en el elemento $a_{1,1}$. Para el caso de la tabla de 25 reglas lingüísticas los elementos que pertenecen a este caso se observan en la zona oscura de la tabla 4.5.1.1

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	4	4
2	3	4	4	4

Tabla 4.5.1. 1 Elementos del caso 1.

En el caso 1 el tamaño de la vecindad es de 8, por ejemplo la vecindad del elemento, (i,j) que se encuentra lejos de la esquinas u orillas de la matriz, tiene la vecindad definida por:

$$\text{Vecindad del elemento } a_{i,j} = \begin{pmatrix} a_{i-1,j-1} & \cdots & a_{i,j+1} \\ \vdots & \ddots & \vdots \\ a_{i+1,j} & \cdots & a_{i+1,j+1} \end{pmatrix}$$

Sin el elemento $a_{i,j}$.

Para el caso de 25 reglas lingüísticas, la vecindad del elemento $a_{2,2}$ se observa en la tabla 4.5.1.2

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	4	4
2	3	4	4	4

Tabla 4.5.1. 2 Vecindad de la regla $a_{2,2}$.

4.5.2 Vecindad caso 2

Las vecindades del caso 2 son aquellas que se encentran en las esquinas, que para todas las matrices de nxn son los elementos $a_{0,0}$, $a_{0,n}$, $a_{n,0}$ y $a_{n,n}$ solo cuatro elementos (cuatro esquinas). Para el caso de 25 reglas lingüísticas los elementos que pertenecen a este caso se observan en la tabla 4.4.2.1

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	4	4
2	3	4	4	4

Tabla 4.5.2. 1 Elementos caso 2.

En la tabla 4.5.2.2 se observa que cada esquina tiene una vecindad diferente, a continuación se define cada vecindad para cada caso

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	4	4
2	3	4	4	4

Tabla 4.5.2. 2 Vecindades posibles para las reglas del caso 2.

Para el caso del elemento $a_{0,0}$ la vecindad queda definida como:

$$\text{Vecindad de } a_{0,0} = \begin{pmatrix} a_{0,0} & \cdots & a_{0,1} \\ \vdots & \ddots & \vdots \\ a_{1,0} & \cdots & a_{1,1} \end{pmatrix}$$

Sin incluir el valor de $a_{0,0.}$

Esto especifica que la vecindad en este caso esta hacia abajo y a la derecha, puesto que no existen elementos a la izquierda o hacia arriba.

Para el caso del elemento $a_{0,n}$ la vecindad queda definida como:

$$\text{Vecindad de } a_{0,n} = \begin{pmatrix} a_{0,n-1} & \cdots & a_{0,n} \\ \vdots & \ddots & \vdots \\ a_{1,n-1} & \cdots & a_{1,n} \end{pmatrix}$$

Sin incluir el valor de $a_{0,n.}$

Este elemento tiene entorno a la izquierda y hacia abajo sin tener vecindad arriba o a la derecha.

Para el caso del elemento $a_{n,0}$ la vecindad queda definida como:

$$\text{Vecindad de } a_{n,0} = \begin{pmatrix} a_{n-1,0} & \cdots & a_{n-1,1} \\ \vdots & \ddots & \vdots \\ a_{n,0} & \cdots & a_{n,1} \end{pmatrix}$$

Sin incluir el valor de $a_{n,0.}$

Para este elemento se tiene vecindad arriba y a la derecha, sin entorno a la izquierda o abajo.

Para el caso del elemento $a_{n,n}$ la vecindad queda definida como:

$$\text{Vecindad de } a_{n,n} = \begin{pmatrix} a_{n-1,n-1} & \cdots & a_{n-1,n} \\ \vdots & \ddots & \vdots \\ a_{n,n-1} & \cdots & a_{n,n} \end{pmatrix}$$

Sin incluir el valor de $a_{n,n.}$

Se observa que este elemento no tiene una vecindad a la derecha o abajo, su vecindad esta limitada a la izquierda y arriba.

Como es posible ver, el tamaño de la vecindad se limita a tres para todos los elementos que pertenecen al caso 2.

4.5.3 Vecindad caso 3

El caso 3 de vecindades se refiere a los contornos de aquellos elementos que se encuentra a las orillas de la matriz, el tamaño de la vecindad de estos elementos es de 5 y existen cuatro diferentes definiciones de vecindad para estos elementos (debido a 4 orillas) dependiendo de la posición, muy parecidas a las expuestas en el caso 2 donde se trataron y definieron las esquinas. Los elementos de la matriz de la tabla de 25 reglas que pertenece a este caso se observa en la tabla 4.5.3.1 en las casillas sombreadas.

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	4	4
2	3	4	4	4

Tabla 4.5.3. 1 Elementos del caso 3.

La definición de la vecindad para los elementos de la orilla en la columna cero.

$$\text{Vecindad Columna cero} = \begin{pmatrix} a_{i-1,j} & \cdots & a_{i-1,j+1} \\ \vdots & \ddots & \vdots \\ a_{i+1,j} & \cdots & a_{i+1,j+1} \end{pmatrix}$$

Sin incluir el valor $a_{i,j}$. Se observa que no tiene vecindad a la izquierda de la matriz.

La definición de la vecindad para los elementos de la orilla del renglón cero.

$$\text{Vecindad Renglon cero} = \begin{pmatrix} a_{i,j-1} & \cdots & a_{i,j+1} \\ \vdots & \ddots & \vdots \\ a_{i+1,j-1} & \cdots & a_{i+1,j+1} \end{pmatrix}$$

Sin incluir el valor $a_{i,j}$. Se observa que no tiene vecindad hacia arriba pues son los valores superiores de la matriz.

La definición de la vecindad para los elementos de la orilla del renglón n.

$$\text{Vecindad Renglon n} = \begin{pmatrix} a_{i-1,j-1} & \cdots & a_{i-1,j+1} \\ \vdots & \ddots & \vdots \\ a_{i,j-1} & \cdots & a_{i,j+1} \end{pmatrix}$$

Sin incluir el valor $a_{i,j.}$ Se observa que no tiene vecindad hacia abajo puesto que es el último renglón de la matriz.

La definición de la vecindad para los elementos de la orilla en la columna n.

$$\text{Vecindad Columna n} = \begin{pmatrix} a_{i-1,j-1} & \cdots & a_{i-1,j} \\ \vdots & \ddots & \vdots \\ a_{i+1,j-1} & \cdots & a_{i+1,j} \end{pmatrix}$$

Sin incluir el valor $a_{i,j.}$ Se observa que no tiene vecindad a la derecha de la matriz por ser la última columna.

En la tabla 4.5.3.2 se observan las vecindades para los elementos 0,2 y 4,2 estos elementos pertenecen a las orillas renglón cero y renglón n respectivamente.

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	4	4
2	3	4	4	4

Tabla 4.5.3. 2 Ejemplo de vecindades para las reglas de las orillas de la matriz.

No es necesario definir esto para la tabla de 49 reglas, pues el comportamiento de las vecindades por ser una matriz es idéntico.

4.6 ESTRUCTURAS NECESARIAS PARA LA CREACIÓN DE MEMORIA

Los atributos de movimientos almacenados son a menudo usados en Búsqueda Tabú para imponer restricciones, que evitan que sean elegidos movimientos que invertirían los cambios representados por estos atributos. Una restricción tabú se activa típicamente sólo en el caso en el que sus atributos hayan ocurrido dentro de un número limitado de iteraciones anteriores a la iteración presente (creando una restricción basada en lo reciente) o hayan ocurrido con una cierta frecuencia sobre un período de iteraciones más largo (creando una restricción basada en la frecuencia).

Como se menciona anteriormente es necesario crear estructuras para poder explotar las restricciones y atributos de movimiento, las estructuras creadas para este trabajo constan de dos tablas, una tabú y una tabla de frecuencia.

4.6.1 Estructura Tabú

La tabla tabú contiene los estados tabú de cada regla para utilizar la característica de memoria a corto plazo.

1	0	0	0	0
2	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Tabla 4.6.1. 1 Tabla de la estructura tabú.

4.6.2 Estructura de frecuencia

La tabla de frecuencia guarda las veces que la regla ha sido modificada o usada por la búsqueda para utilizar características de diversificación.

5	2	0	1	0
0	1	0	1	0
0	0	1	0	0
3	1	0	0	0
2	2	3	0	0

Tabla 4.6.2. 1 Tabla de la estructura de frecuencia.

4.6.3 Estructura de evaluación o lista de candidatos

Para poder usar todas las estructuras es necesario crear una tabla que contiene los resultados de la evaluación de la función objetivo en cada regla de la vecindad, con su respectivo valor tabú y su respectivo valor de frecuencia. El número de elementos que aparecen en la estructura depende de la zona evaluada.

LISTA DE CANDIDATOS

No. REGLA	VALORES	TABÚ	FRECUENCIA
Valor 0	27072.076913	1	5
Valor 1	28006.110926	0	2
Valor 2	6111.062818	2	0
Valor 3	38174.913637	0	1
Valor 4	0	0	0
Valor 5	0	0	0
Valor 6	0	0	0
Valor 7	0	0	0
Valor 8	0	0	0

Tabla 4.6.3. 1 Lista de candidatos de la vecindad evaluada.

En la tabla anterior implícitamente es posible notar que la vecindad es de una de las reglas de las esquinas, puesto que solo se obtienen cuatro valores de vecindad, por ejemplo para la primera regla (valor 0) evaluada se tiene un valor de función objetivo de 27072.076913 y un valor tabú de 1 que implica que esta regla es tabú por una iteración, además en la columna de frecuencia para esta regla encontramos un valor de 5 que indica que esta regla ha sido modificada o utilizada por la búsqueda 5 veces.

4.7 SINTONIZACIÓN DE LA MEMORIA ASOCIATIVA DIFUSA

En este caso la búsqueda esta referida a imponer restricciones para guiar el proceso, y por otra parte para negociar regiones difíciles de alcanzar. Estas restricciones operan en varias formas, ambas por exclusión directa de alternativas de búsqueda clasificadas como “Prohibidas” y también por la translación de evaluaciones modificadas y probabilidad de elección.

La filosofía de Búsqueda Tabú es derivar y explotar una colección de principios de inteligencia resolviendo problemas. Un elemento fundamental de la Búsqueda Tabú es el uso de memoria flexible. Desde el punto de vista de Búsqueda Tabú conjunta el proceso de crear y explotar estructuras para tomar ventaja de la historia.

4.7.1 Definición inicial de la base de reglas

En la mayoría de los casos la solución inicial puede ser construida aleatoriamente, sin embargo es preferible hacerlo de una forma inteligente, es decir tomando ventajas de las estructuras de los problemas [2] y [11], en este caso se elige una base de reglas simétrica tanto para la tabla de 25 reglas como para la de 49 reglas. El proceso se ejemplifica con la tabla de 25 reglas, con los valores iniciales obtenidos en la sección 3.4 sin realizar la sintonización manual para el inicio de la autosintonización final. La tabla simplificada simétrica para 25 reglas se presenta en la tabla 4.7.1.1.
Búsqueda en 2,2.

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	4	4
2	3	4	4	4

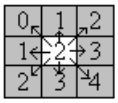


Tabla 4.7.1. 1 Vecindad de la regla 2,2.

4.7.2 Inicio de la búsqueda

El método opera bajo la consideración que una vecindad puede ser construida para identificar soluciones adyacentes que pueden ser alcanzadas desde el punto de búsqueda actual o solución actual [1], [2], [11] y [15]. En este caso las vecindades se definen como aquellas reglas a donde es posible mudar la búsqueda y son aquellas que rodean la regla, en este problema los cambios además de ser usados para viajar de una solución a otra, son usados para elegir subespacios de búsqueda. La Búsqueda Tabú inicia en el centro de la base de reglas elemento 2,2 aunque es posible iniciar en cualquier otra regla. Debido a que se incluye el cero en los subíndices el número del renglón es dos y como se tiene el mismo número de funciones de membresía para el error y el cambio del error el número de columna es dos, la vecindad de esta regla puede verse en la tabla 4.7.1.1 y los movimientos en el cuadro extraído de la tabla a la derecha.

4.7.3 Evaluación de la vecindad

Al evaluar la vecindad de esta regla tenemos asociado a cada cambio un valor de movimiento, el cual representa el valor de la función objetivo como resultado del proceso de cambio. Los valores del movimiento generalmente proveen una base fundamental para evaluar la calidad de un movimiento [2], [11] y [15]. Los valores mostrados en esta sección son los obtenidos al utilizar el programa de simulación.

No. REGLA	VALORES	TABÚ	FRECUENCIA
Valor 0	27072.076913	0	0
Valor 1	28006.110926	0	0
Valor 2	6111.062818	0	0
Valor 3	38174.913637	0	0
Valor 4	6111.062818	0	0
Valor 5	24675.404344	0	0
Valor 6	6111.062818	0	0
Valor 7	11286.949668	0	0
Valor 8	1740.774619	0	0

Tabla 4.7.3. 1 Lista de candidatos resultante al evaluar la vecindad de la regla 2,2.

Mejor valor encontrado 8, 1740.774619
Valor elegido para la búsqueda 7

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	2	4
2	3	4	4	4

Tabla 4.7.3. 2 Tabla de reglas después de elegir el movimiento.

4.7.4 Clasificación Tabú

Un mecanismo para explotar la memoria en Búsqueda Tabú es clasificar un conjunto de movimientos en una vecindad como prohibidos (o Tabú). La clasificación depende de la historia de la búsqueda, particularmente manifestada en frecuencia o reciente a esto se le llama atributo, y es la participación en búsquedas o en la generación de soluciones pasadas [1], [2], [11] y [15]. Por ejemplo un atributo de un cambio es el identificado por el elemento que cambia y el utilizado para realizar la búsqueda.

ESTRUCTURA TABÚ

0	0	0	0	0
0	0	0	0	0
0	0	2	0	0
0	0	0	2	0
0	0	0	0	0

Tabla 4.7.4. 1 Estructura que muestra los elementos tabú resultantes del primer movimiento.

TABLA DE FRECUENCIA

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	0

Tabla 4.7.4. 2 Tabla de frecuencia resultante del primer movimiento.

Como la base para prevenir que la búsqueda repita cambios que se realizaron en el pasado reciente, revirtiendo potencialmente los efectos de movimientos previos por cambios que puedan

regresar a la configuración previa, se clasifican tabú todas las reglas que se hallan cambiado o utilizado para realizar la búsqueda, por 2 iteraciones siguientes como se observa en la estructura tabú de la tabla 4.7.4.1 mientras que la tabla de frecuencia incrementa en uno cada vez que la regla participe en la búsqueda o sea cambiada.

4.7.5 Memoria a corto plazo

Cada celda de la tabla estructura tabú contiene el número de iteraciones que deben permanecer las reglas, hasta que se les permita ser cambiadas o utilizadas otra vez para realizar una búsqueda, es decir si la regla 1,1 tiene un valor de cero esta puede ser utilizada libremente para realizar una búsqueda o ser cambiada. Por otro lado la regla la regla 3,3 tiene un valor de 2 que no permite que esta sea utilizada en las siguientes dos iteraciones esto es llamado memoria basada en lo reciente e indica las reglas que fueron utilizadas en las dos iteraciones anteriores.

La forma utilizada es de alta influencia pues se elige el valor que mejore más la solución actual de los valores tanto para continuar la búsqueda como para realizar el movimiento. Los elementos que son iguales al valor que se esta utilizando para evaluar la vecindad se hacen tabú, solo por el instante de evaluación, es decir no es posible mudar la búsqueda a una regla que tenga el valor de prueba.

4.7.6 Criterio de aspiración

Al implementar restricciones tabú existen consideraciones importantes, estas no son inviolables bajo todas las circunstancias, cuando un movimiento tabú puede resultar en una solución mejor de las que se han visitado antiguamente su clasificación tabú puede ser anulada [1], [12], [13] y [15]. Una condición que permite que tal sustitución ocurra es llamado criterio de aspiración en este problema se utiliza el llamado “la mejor solución”.

La siguiente iteración utiliza este criterio para violar una restricción tabú, al evaluar la vecindad de r = 3 , c = 2.

Búsqueda en 3,2.

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	2	4
2	3	4	4	4

Tabla 4.7.6. 1 Tabla de reglas indicando la vecindad en celdas sombreadas.

No. REGLA	VALORES	TABÚ	FRECUENCIA
Valor 0	12657.355443	0	0
Valor 1	11979.954709	2	1
Valor 2	1740.774619	0	0
Valor 3	1841.495052	0	0
Valor 4	1740.774619	0	0
Valor 5	1521.767552	2	1
Valor 6	1740.774619	0	0
Valor 7	1988.133567	0	0
Valor 8	6948.547237	0	0

Tabla 4.7.6. 2 Lista de candidatos resultante de evaluar la vecindad.

Valor elegido para la búsqueda 3
Mejor valor encontrado 5, 1521.767552

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	3	4
2	3	4	4	4

Tabla 4.7.6. 3 Tabla de reglas resultante al utilizar el criterio de aspiración.

La solución inicial tiene un valor en la función objetivo de 6111.062818 y en el inicio la estructura tabú tiene todos los valores en cero, que indica que todas las reglas pueden ser utilizadas por la búsqueda, en este punto la función objetivo tiene un valor de 1740.774619 y se observa que la estructura tabú tiene dos elementos tabú, el 3,3 coincide con un valor de mejora nunca antes encontrado por lo cual cumple con el criterio de aspiración y aunque es considerado tabú la búsqueda anula este estado y lo elige como cambio tabla 4.7.6.3.

ESTRUCTURA TABÚ

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	2	2	0
0	0	0	0	0

Tabla 4.7.6. 4 Estructura resultante al utilizar el criterio de aspiración.

TABLA DE FRECUENCIA

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	1	2	0
0	0	0	0	0

Tabla 4.7.6. 5 Tabla resultante al utilizar el criterio de aspiración.

Al valor cambiado se le asigna de nuevo un estado tabú de dos iteraciones, note que el elemento 2,2 ha disminuido su estado tabú a 1 iteración.

Evaluando la vecindad de r = 3 c = 1 que es la regla a donde se muda la búsqueda, se observa que en este momento ninguno de los candidatos (incluyendo los mejores) tienen un valor de movimiento que mejore la función objetivo.

Búsqueda en 3,1.

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	3	4
2	3	4	4	4

Tabla 4.7.6. 6 Tabla que muestra la vecindad de la regla.

No. REGLA	VALORES	TABÚ	FRECUENCIA
Valor 0	27400.781603	0	0
Valor 1	1939.333316	0	0
Valor 2	1521.767552	1	1
Valor 3	1522.704365	0	0
Valor 4	1521.767552	0	0
Valor 5	3101.488711	2	1
Valor 6	1521.767552	0	0
Valor 7	1524.323608	0	0
Valor 8	23136.819124	0	0

Tabla 4.7.6. 7 Lista de candidatos que no posee un valor de mejora.

Valor elegido para la búsqueda 3
Mejor valor encontrado se mantiene el anterior no hay mejora 1521.767552

4.7.7 Valor CAOS

Como se utilizan subespacios de búsqueda es necesario utilizar un valor llamado caos, el cual tiene la función para dejar que la búsqueda avance en el espacio sin que se realice un movimiento destructivo, pero cuando este valor de un número especifico de iteraciones es sobrepasado, la búsqueda empieza a realizar movimientos de no mejora. El valor de caos utilizado en este trabajo es dos.

El mejor movimiento de no mejora es el que hace permanecer la función objetivo con el mismo valor. En algunas implementaciones de búsqueda por tabú, donde una gran proporción de movimientos en la vecindad tienen un valor de no mejora en la función objetivo, los investigadores han encontrado conveniente prohibir la selección de movimiento que hacen que la función objetivo de cambio tenga el mismo valor [15], pero en este problema la búsqueda permite que la búsqueda se mude al siguiente movimiento sin hacer cambio, esto hace que la solución actual tenga el mismo valor de función objetivo. Se realiza este procedimiento debido a que se trabaja con subespacios de la base de reglas, por lo que el encontrar un valor de no mejora en la función objetivo con la regla de prueba no significa que en la vecindad de la regla no existan otras reglas de prueba que mejoren la función objetivo. Sin embargo esto se realiza por un número definido de iteraciones (caos) para evitar caer en óptimos locales, después de cierto número de no mejoras se empiezan a aceptar movimientos destructivos o de no mejora para poder escapar de este óptimo local y diversificar la búsqueda.

ESTRUCTURA TABÚ

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	2	1	1	0
0	0	0	0	0

Tabla 4.7.7. 1 Estructura tabú resultante de un movimiento de no mejora.

TABLA DE FRECUENCIA

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	1	1	2	0
0	0	0	0	0

Tabla 4.7.7. 2 Estructura de frecuencia después de utilizar el criterio de aspiración.

Como se observa en la estructura tabú solo un elemento tiene un valor tabú de 2, esto indica que la búsqueda no cambio ningún valor en la base de reglas y que solo se hizo tabú el valor utilizado para probar los movimientos.

Evaluando la vecindad de r= 3 c= 0.

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	2	3	3	4
2	3	4	4	4

Tabla 4.7.7. 3 Tabla de reglas que indica la vecindad de la regla.

No. REGLA	VALORES	TABÚ	FRECUENCIA
Valor 0	17872.696038	0	0
Valor 1	1521.767552	0	0
Valor 2	1521.767552	0	0
Valor 3	1370.126192	2	1
Valor 4	46041.255574	0	0
Valor 5	1527.977057	0	0
Valor 6	0.000000	0	0
Valor 7	0.000000	0	0
Valor 8	0.000000	0	0

Tabla 4.7.7. 4 Lista de candidatos con sus respectivos valores tabú y frecuencia.

Valor elegido para la búsqueda 5
Mejor valor encontrado 3, 1370.126192

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	1	3	3	4
2	3	4	4	4

Tabla 4.7.7. 5 Tabla resultante de la evaluación actual.

Aquí encontramos otra vez un valor que cumple con el criterio de aspiración, justificando el criterio para utilizar un valor de caos, por lo que la búsqueda lo elige para realizar el cambio,

actualizando su valor tanto en la tabla de reglas como en la estructura tabú y la estructura de frecuencia.

4.7.8 Uso de penalizaciones

La estrategia de imponer penalizaciones solo bajo condiciones particulares se usa para preservar la agresividad de la búsqueda. Las funciones de penalización en general se diseñan para justificar no sólo frecuencias sino también valores de movimientos y ciertas medidas de influencia.

ESTRUCTURA TABÚ

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
2	2	0	0	0
0	0	0	0	0

Tabla 4.7.8. 1 Estructura con los nuevos elementos tabú.

TABLA DE FRECUENCIA

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
1	2	1	2	0
0	0	0	0	0

Tabla 4.7.8. 2 Tabla de frecuencia con valores actualizados.

Se muda a la siguiente regla, es posible notar que no se utiliza hasta este punto una característica diferente a las ya mencionadas anteriormente por lo que el proceso continúa.

Evaluando la vecindad de r= 4 c= 1

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
1	1	3	3	4
2	3	4	4	4

Tabla 4.7.8. 3 Tabla de reglas que indica la vecindad de la regla.

No. REGLA	VALORES	TABÚ	FRECUENCIA
Valor 0	1348.876826	2	1
Valor 1	1554.318775	2	2
Valor 2	1370.126192	0	1
Valor 3	1350.526777	0	0
Valor 4	1370.126192	0	0
Valor 5	1529.504743	0	0
Valor 6	0.000000	0	0
Valor 7	0.000000	0	0
Valor 8	0.000000	0	0

Tabla 4.7.8. 4 Lista de candidatos resultante de la evaluación.

Valor elegido para la búsqueda 3, 1350.526777

Mejor valor encontrado 0

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
3	1	3	3	4
2	3	4	4	4

Tabla 4.7.8. 5 Tabla de reglas resultante de la evaluación.

En este punto el criterio de aspiración se cumple de nuevo y existen dos valores uno a donde muda la búsqueda y otro que cambia la regla.

ESTRUCTURA TABÚ

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
2	1	0	0	0
0	2	0	0	0

Tabla 4.7.8. 6 Estructura que indica los elementos tabú.

TABLA DE FRECUENCIA

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
2	2	1	2	0
0	1	0	0	0

Tabla 4.7.8. 7 Tabla con valores de frecuencia actualizados.

Evaluando la vecindad de la regla 4,0.

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
3	1	3	3	4
2	3	4	4	4

Tabla 4.7.8. 8 Base de reglas que indica la vecindad de la regla.

No. REGLA	VALORES	TABÚ	FRECUENCIA
Valor 0	1356.155504	2	2
Valor 1	1512.557824	1	2
Valor 2	1348.876826	0	0
Valor 3	1368.416810	2	1
Valor 4	0.000000	0	0
Valor 5	0.000000	0	0
Valor 6	0.000000	0	0
Valor 7	0.000000	0	0
Valor 8	0.000000	0	0

Tabla 4.7.8. 9 Lista de candidatos resultante de evaluar la vecindad.

Valor elegido para la búsqueda 0
Mejor valor encontrado. No se encuentra un valor que mejore la búsqueda.

ESTRUCTURA TABÚ

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
1	0	0	0	0
2	1	0	0	0

Tabla 4.7.8. 10 Estructura con nuevos valores tabú.

TABLA DE FRECUENCIA

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
2	2	1	2	0
1	1	0	0	0

Tabla 4.7.8. 11 Tabla que indica las reglas utilizadas.

Hasta este punto no se ha utilizado la estructura tabú puesto que el valor de caos apenas esta alcanzando su valor máximo permitido, en las ocasiones anteriores lo inactivó el criterio de aspiración, se observa que el movimiento 0, el correspondiente con la regla 3,0 es el que menos empeora la búsqueda así es posible continuar la búsqueda en ese lugar.

Evaluando la vecindad de r= 3 c= 0

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
3	1	3	3	4
2	3	4	4	4

Tabla 4.7.8. 12 Vecindad de la regla.

No. REGLA	VALORES	TABÚ	FRECUENCIA
Valor 0	15465.749509	0	0
Valor 1	16346.988435	0	0
Valor 2	1348.876826	1	2
Valor 3	1678.572035	0	2
Valor 4	1356.478618	2	1
Valor 5	1348.876826	1	1
Valor 6	0.000000	0	0
Valor 7	0.000000	0	0
Valor 8	0.000000	0	0

Tabla 4.7.8. 13 Lista de candidatos.

En este punto se ha superado el valor de caos por lo que empiezan a aparecer elecciones de movimientos de no mejora.

ESTRUCTURA TABÚ

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
2	2	0	0	0
1	0	0	0	0

Tabla 4.7.8. 14 Elementos tabú.

TABLA DE FRECUENCIA

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
3	3	1	2	0
1	1	0	0	0

Tabla 4.7.8. 15 Elementos utilizados en la búsqueda.

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
0	1	2	3	4
3	3	3	3	4
2	3	4	4	4

Tabla 4.7.8. 16 Base de reglas con el primer movimiento de no mejora.

4.7.9 Diversificación

Los contadores de frecuencia muestran la distribución de movimientos a través de todas las iteraciones. Se usan estos contadores para diversificar la búsqueda, conduciéndola a nuevas regiones, esta influencia de diversificación se restringe a operar sólo sobre ocasiones particulares. En este caso, se seleccionaron aquellas ocasiones donde no existen movimientos de mejora admisibles, tomando en cuenta que después de pasar el valor de caos de no encontrar mejora esto ocurrirá.

En este punto se ha superado el valor de caos por lo que empiezan a aparecer elecciones de movimientos de no mejora.

Como se observa en la tabla de valores no existen movimientos posibles de mejora así que se acepta el movimiento que menos empeore el valor de la función objetivo pero que no tenga un valor tabú, (criterio de baja influencia), en este caso es el número 3, por lo que el nuevo valor para la función objetivo en la búsqueda es 1678.572035 que cambia la regla 3,1.

El siguiente subespacio para continuar la búsqueda será una regla que no tenga estado tabú y que su valor sea superior al de la función objetivo actual en este caso el número cero, por lo que la búsqueda continua en 2,0.

El uso de la información de frecuencia penalizará movimientos de no mejora mediante la asignación de una penalización mayor a cambios de reglas con mayores contadores de frecuencia. Típicamente, estos contadores serian normalizados, por ejemplo mediante la división por el número total de iteraciones o su máximo valor, en este trabajo el valor de penalización no solo depende del valor del contador de frecuencia sino también del valor de la estructura tabú para elegir el movimiento, las penalizaciones dependen de los valores enteros de estos valores. Es posible desarrollar intensificación con este concepto, así dando puntos de apoyo fundamentales en memoria de largo plazo en Búsqueda Tabú.

En la iteración 15.

Evaluando la vecindad de r= 1 , c= 1.

TABLA DE REGLAS

0	0	0	1	2
0	0	1	2	3
1	1	2	3	4
3	1	3	3	4
3	3	4	4	4

Tabla 4.7.9. 1 Tabla de reglas que indica la vecindad de la regla.

Se encuentran los siguientes valores:

No. REGLA	VALORES	TABÚ	FRECUENCIA
Valor 0	1329.558580	0	0
Valor 1	1329.558580	0	0
Valor 2	1329.558580	0	0
Valor 3	1329.558580	0	0
Valor 4	1329.558580	0	0
Valor 5	31008.873738	0	0
Valor 6	1356.478618	0	5
Valor 7	50881.639536	2	3
Valor 8	51667.298341	0	1

Tabla 4.7.9. 2 Lista de candidatos que requiere el uso de diversificación.

Valor elegido para la búsqueda 5
Mejor valor encontrado: No se encuentra un valor que mejore la búsqueda.

De la estructura anterior se observa que no existe ningún movimiento que mejore la búsqueda pues el valor actual de la función objetivo es de 1329.558580 lo cual implica que para esta iteración estos valores tienen un estado tabú, entra el criterio de multas el cual, suma los valores de las estructuras tabú y las de frecuencia para dar penalización mayor a las reglas que han sido utilizadas muy recientemente y aquellas que han participado más en la búsqueda.

ESTRUCTURA TABÚ

0	0	0	0	0
0	2	0	0	0
0	1	0	0	0
0	1	0	0	0
0	0	0	0	0

Tabla 4.7.9. 3 Estructura tabú que indica los elementos prohibidos.

TABLA DE FRECUENCIA

0	0	0	0	0
0	1	0	0	0
5	3	1	0	0
5	6	1	2	0
2	1	0	0	0

Tabla 4.7.9. 4 Estructura de frecuencia.

TABLA DE REGLAS

0	0	1	1	2
0	0	1	2	2
1	1	2	3	4
3	1	3	3	4
3	3	4	4	4

Tabla 4.7.9. 5 Tabla de reglas resultante al utilizar la Búsqueda Tabú.

Valor de la función objetivo 1319.152935.

La tabla 4.7.9.5 es la configuración con el mejor valor encontrado al usar la búsqueda, y el mejor valor encontrado es 1319.152935.

CAPÍTULO 5

PRUEBAS Y RESULTADOS

5.1 INTRODUCCIÓN

En este capítulo se presentan los resultados de la autosintonización de la tabla de reglas lingüísticas del control difuso tipo Mamdani, con el método Búsqueda Tabú. Las pruebas se realizan con los datos de un motor de inducción de 3 hp, trifásico de cuatro polos, utilizando control vectorial directo por flujo orientado del rotor para mayor información de los parámetros del motor ver apéndice A. Este capítulo se divide en 8 partes, la segunda parte (5.2) define las pruebas a las cuales fue sometido el sistema y que índice de comportamiento fue utilizado; La tercera parte (5.3) muestra los resultados de la autosintonización de la tabla de 25 reglas, a diferentes disturbios en la carga y a cambios de velocidad con la misma tabla utilizada por los 4 controles, observando los efectos de la optimización en el comportamiento de las señales de velocidad, flujo del rotor, y las corrientes en el marco de referencia d-q; La cuarta parte (5.4) muestra los resultados de la autosintonización de la tabla de 49 reglas, a diferentes disturbios en la carga y a cambios de velocidad, comparando los efectos de la optimización en el comportamiento de las señales de las corrientes en el marco de referencia d-q, flujo del rotor y velocidad. Las secciones 5.5 y 5.6 presentan los índices de funcionamiento para los controles con 25 y 49 reglas para los diferentes disturbios.

La séptima parte (5.7) presenta el análisis de resultados obtenidos con controles de 25 reglas mientras que la octava parte (5.8) presenta el análisis de los resultados obtenidos con controles de 49 reglas, ambos utilizando la misma tabla de reglas para los cuatro controles, y esta misma autosintonizada por la Búsqueda Tabú.

5.2 AUTOSINTONIZACIÓN CON 25 Y 49 REGLAS LINGÜÍSTICAS

En la presentación de los resultados se muestran las gráficas de la velocidad, flujo del rotor, corrientes del estator en los ejes d-q y el índice de comportamiento de la velocidad, este índice se realiza mediante la integral del error al cuadrado [44], ecuación 5.2.1.

$$ISE = \int_0^t e^2(t)dt \qquad (5.2.1)$$

La memoria asociativa difusa inicial se propone simétrica para tablas de 25 y 49 reglas, la optimización se inicia en el centro de la tabla, con la opción de iniciar la búsqueda en cualquier parte de la memoria asociativa, las pruebas realizadas para 25 y 49 reglas fueron las mismas. Primero se sintonizan las reglas con respecto al primer disturbio que es quitar la carga del sistema y se presentan los resultados solo para la optimización de este disturbio sin importar lo que sucede en los diferentes disturbios.

La siguiente prueba consiste en llevar el sistema de carga cero al valor de carga nominal y sintonizar para esta prueba usando la Búsqueda Tabú iniciando desde las reglas simétricas. La última sintonización se realiza partiendo de las reglas optimizadas para el primer disturbio, pero optimizando el segundo disturbio, la tabla obtenida es utilizada para las simulaciones de los disturbios en la carga y las variaciones de velocidad.

5.3 AUTOSINTONIZACIÓN DE 25 REGLAS LINGÜÍSTICAS

En esta sección se presentan los resultados obtenidos al simular el sistema con el control PI convencional, el control difuso y los controles difusos tabú estos últimos con una base de 25 reglas. Para la primera prueba se aplica un disturbio al quitar la carga nominal y llevarla al valor de cero. La segunda prueba se realiza al llevar el sistema de carga cero al valor de carga nominal. Por último se utiliza la tabla sintonizada para el primer disturbio utilizando la Búsqueda Tabú para sintonizar el segundo disturbio y la tabla resultante se emplea en la simulación de todos disturbios propuestos y los cambios en la velocidad de referencia.

Se presentan además los diagramas de fase tiempo y lingüísticos, con la base de reglas no optimizada y optimizada para observar los cambios de las trayectorias. Se incluyen además de las gráficas de velocidad, las gráficas de flujo y corrientes en los ejes d-q, para cada disturbio en particular y al final de forma general.

Los índices de funcionamiento del control vectorial convencional, difuso y difuso tabú para las tres pruebas se presentan juntos en la sección 5.5.

5.3.1 Autosintonización de 25 reglas 1er disturbio

La Búsqueda Tabú es utilizada para sintonizar la tabla de 25 reglas, cuando el sistema es llevado de carga nominal al valor de carga cero, se muestran los resultados de velocidad, flujo y corriente, además de los diagramas de fase y lingüísticos antes y después del uso de la Búsqueda Tabú.

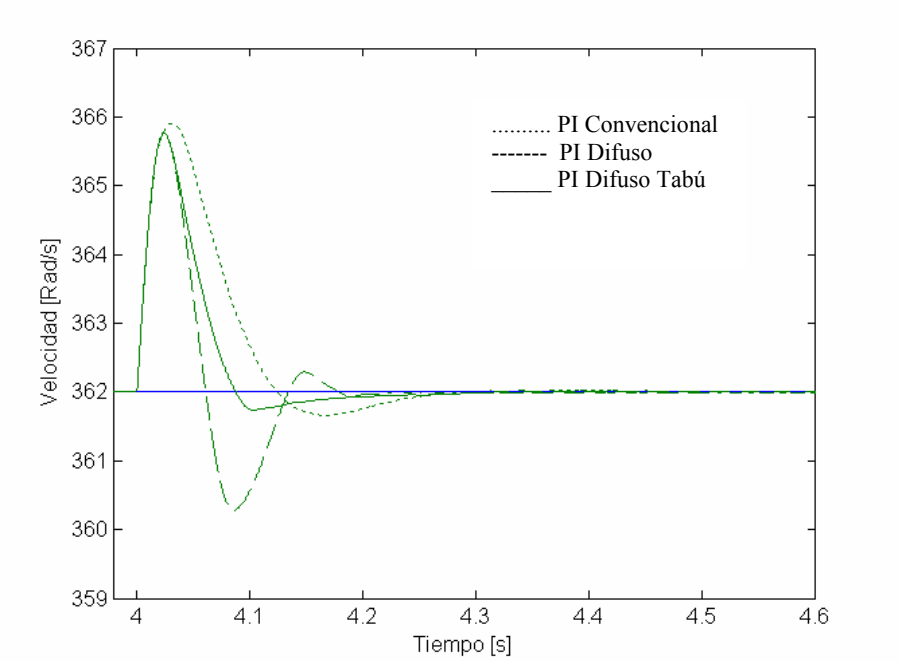


Figura 5.3. 1 Resultados de velocidad de las simulaciones al retirar la carga nominal del sistema.

En la figura 5.3.1 se observan los resultados de velocidad de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú. El disturbio es producido en t = 4 segundos al retirar la carga nominal (11.9 Nm) y llevarla hasta el valor de cero.

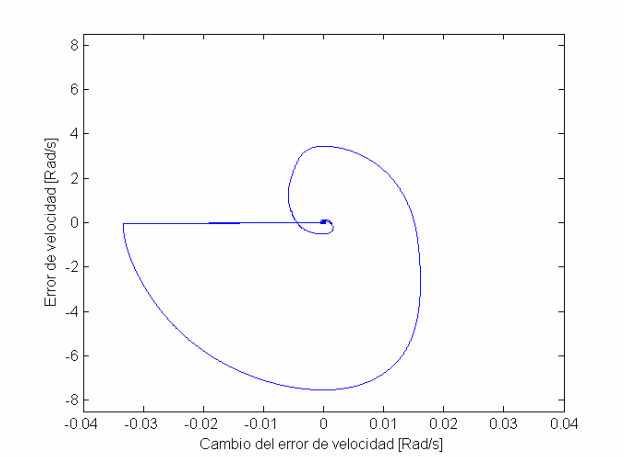


Figura 5.3. 2 Diagrama de fase tiempo.

BM	MB	BM	BP	N
BM	BM	BP	N	SP
BM	BP	N	SP	SM
BP	N	SP	SM	SM
N	SP	SM	SM	SM

Figura 5.3. 3 Diagrama de fase lingüístico.

La figura 5.3.2 muestra el diagrama de fase tiempo resultante del sistema con el control difuso y la figura 5.3.3 muestra el diagrama de fase lingüístico del control difuso ambas con la base de reglas sin optimizar.

El diagrama de fase de tiempo comparado con el diagrama de fase lingüístico es diferente debido a la forma en la cual fueron elegidas las reglas, ver tabla 3.3.1.1 donde los valores negativos aparecen a la izquierda para el error, y para el cambio del error la parte negativa se encuentra en la parte superior de la tabla, mientras que para el diagrama de fase tiempo las partes negativas coinciden en la parte inferior izquierda de la tabla.

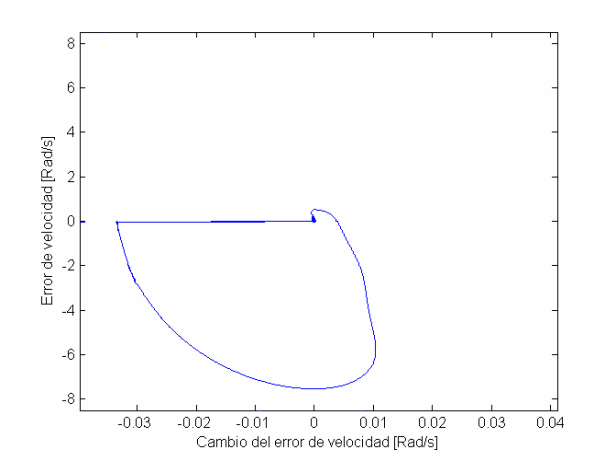


Figura 5.3. 4 Diagrama de fase tiempo.

BM	BM	BM	BP	N
BM	BM	BP	SP	SM
BM	BP	N	SP	SM
BP	N	SP	SM	SM
N	SP	SM	SM	SM

Figura 5.3. 5 Diagrama de fase lingüístico.

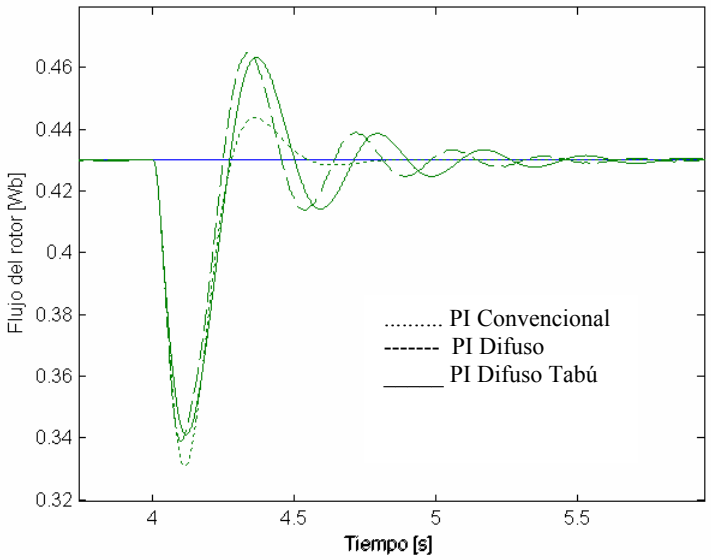


Figura 5.3. 6 Resultados del flujo del rotor de las simulaciones al retirar la carga nominal del sistema.

En la figura 5.3.4 se muestra el diagrama fase tiempo del control difuso optimizado, y en la figura 5.3.5 se observa el diagrama de fase lingüístico resultante del sistema con la memoria asociativa optimizada.

Comparando los resultados se observa que solo cambian dos reglas, que disminuyen la inercia del sistema al regresar a la velocidad de referencia, por lo que la regla con acción "Nada" cambia a "Subir Poco".

En la figura 5.3.6 se observan los resultados del flujo del rotor de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú.

El disturbio es producido en t = 4 segundos al retirar la carga nominal (11.9 Nm) y llevarla hasta el valor de cero. Los controles difusos utilizan la misma memoria asociativa difusa de 25 reglas.

El flujo del rotor como se observa, no es afectado al utilizar la Búsqueda Tabú para optimizar el control de velocidad, esto se debe a que el flujo tiene un sobre impulso contrario al de la respuesta de la velocidad del sistema para el mismo disturbio.

A continuación se presentan las corrientes obtenidas con los diferentes controles propuestos.

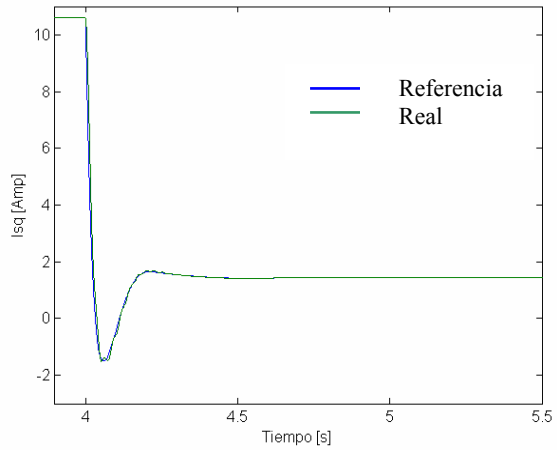


Figura 5.3.7. 1 Resultados de I_{sq} del control convencional.

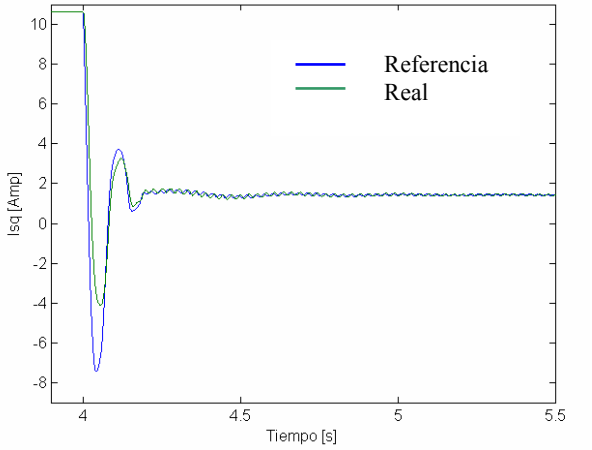


Figura 5.3.7. 2 Resultados de I_{sq} del control difuso.

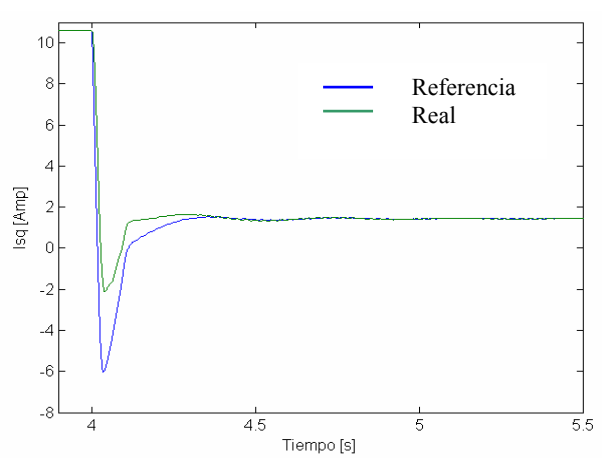


Figura 5.3.7. 3 Resultados de I_{sq} del control difuso optimizado.

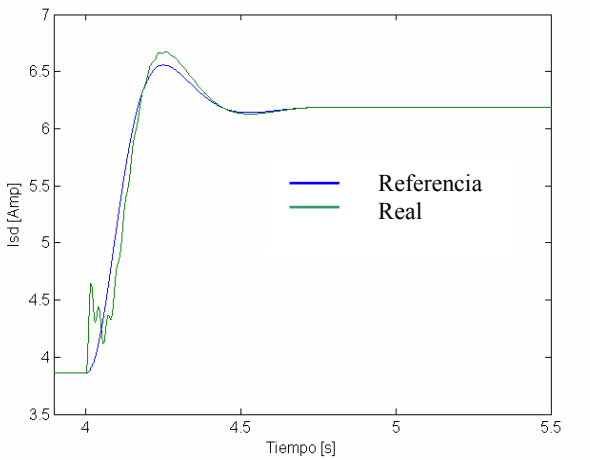


Figura 5.3.8. 1 Resultados de I_{sd} del control convencional.

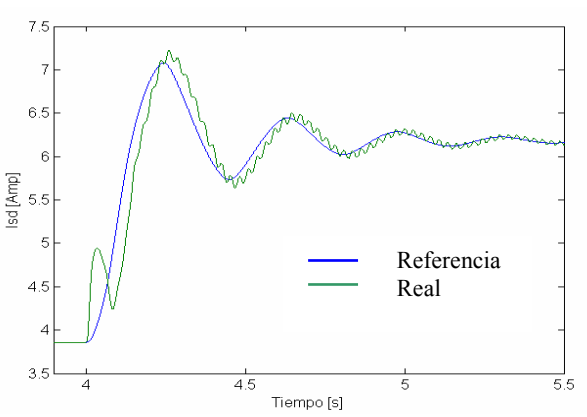


Figura 5.3.8. 2 Resultados de I_{sd} del control difuso.

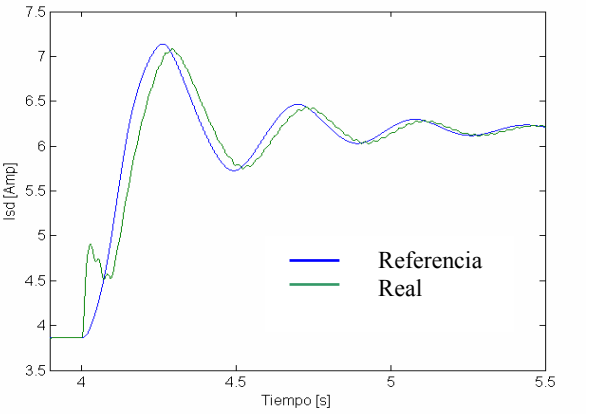


Figura 5.3.8. 3 Resultados de I_{sd} del control difuso optimizado.

En las figuras 5.3.7 se observan las señales de referencia de i_{sq} y los valores resultantes de i_{sq} para en control vectorial convencional, el control difuso y el control difuso optimizado por Búsqueda Tabú respectivamente.

En las figuras 5.3.8 se muestran los resultados de las señales de referencia para i_{sd} y los resultados de i_{sd} para los controles propuestos con tablas de 25 reglas.

Las corrientes obtenidas con el control usado no presentan una mejora sobre el control convencional, ni siquiera después de haber usado la búsqueda por tabú para mejorar el comportamiento del sistema. Esto se debe a que se optimiza con respecto a la señal de velocidad y no con respecto a las señales de corriente.

5.3.2 Autosintonización de 25 reglas 2do disturbio

La Búsqueda Tabú es utilizada para sintonizar la tabla de 25 reglas, cuando el sistema es llevado de carga cero al valor de carga nominal, se muestran los resultados de velocidad, flujo y corriente, además de los diagramas de fase y lingüísticos antes y después del uso de la Búsqueda Tabú.

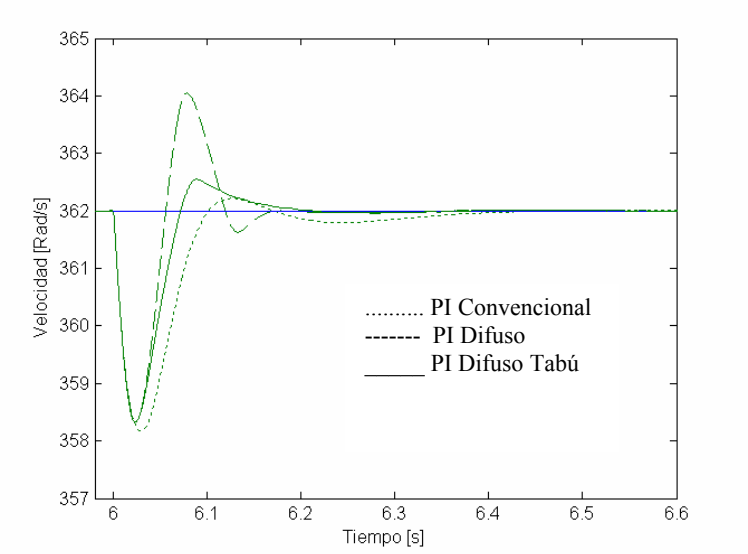


Figura 5.3.9 Resultados de velocidad de las simulaciones al retirar la carga nominal del sistema.

En la figura 5.3.9 se observan los resultados de velocidad de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú. El disturbio es producido en t = 6 segundos al llevar el sistema de carga cero al valor de carga nominal (11.9 Nm).

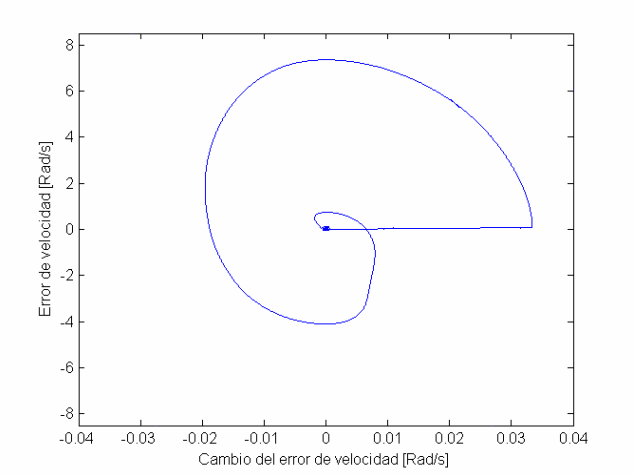


Figura 5.3.10 Diagrama de fase tiempo.

BM	BM	BM	BP	N
BM	BM	BP	N	SP
BM	BP	N	SP	SM
BP	N	SP	SM	SM
N	SP	SM	SM	SM

Figura 5.3.11 Diagrama de fase lingüístico.

La figura 5.3.10 muestra el diagrama de fase tiempo resultante del sistema con el control difuso y la figura 5.3.11 muestra el diagrama de fase lingüístico del control difuso sin optimizar.

Es posible observar las oscilaciones en el sistema al notar que en los diagramas de fase tiempo y lingüístico, la trayectoria rodea el centro de los diagramas de fase, lo que corresponde con los resultados de velocidad en el sistema.

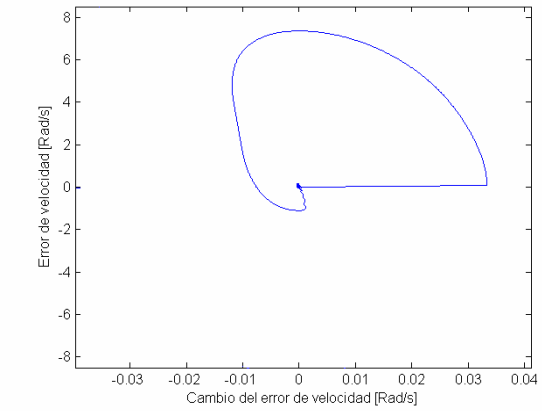


Figura 5.3.12 Diagrama de fase tiempo.

BM	BM	BM	BP	N
BM	BM	BP	N	SP
BM	BP	N	SP	SM
BM	BP	SP	SM	SM
N	SP	SM	SM	SM

Figura 5.3.13 Diagrama de fase lingüístico

En la figura 5.3.12 se muestra el diagrama fase tiempo del control difuso optimizado, y en la figura 5.3.13 se observa el diagrama de fase lingüístico resultante del sistema con la memoria asociativa optimizada. Comparando los resultados se observa que solo cambian dos reglas, que disminuyen la inercia del sistema al regresar a la velocidad de referencia, por lo que la regla con acción “Nada” cambia a “Bajar Poco”.

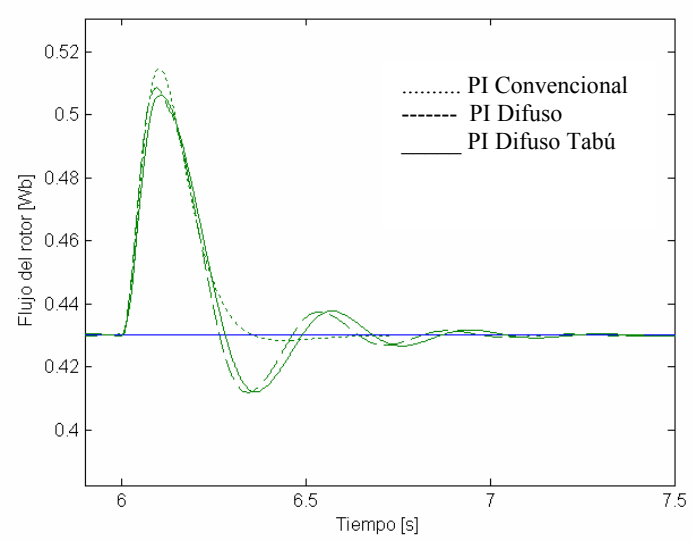


Figura 5.3. 14 Resultados del flujo del rotor de las simulaciones al aplicar carga nominal al sistema.

En la figura 5.3.14 se observan los resultados del flujo del rotor de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú. El disturbio es producido en t = 6 segundos al llevar la carga de cero hasta el valor de carga nominal (11.9 Nm). Los controles difusos utilizan la misma memoria asociativa difusa de 25 reglas.

El flujo del rotor como se observa, no es afectado al utilizar la Búsqueda Tabú para optimizar el control de velocidad, esto se debe a que el flujo tiene un sobre impulso contrario al de la respuesta de la velocidad del sistema para el mismo disturbio.

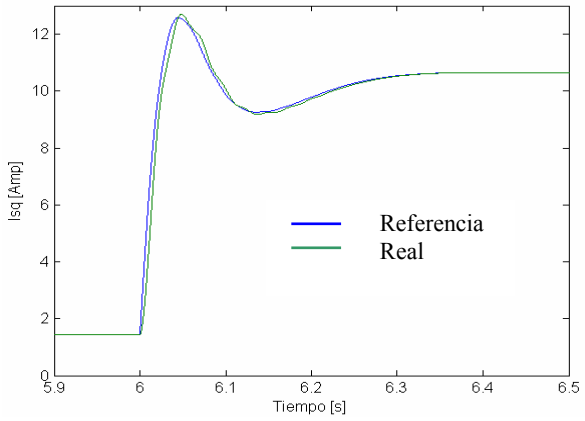


Figura 5.3.15. 1 Resultados de I_{sq} del control convencional.

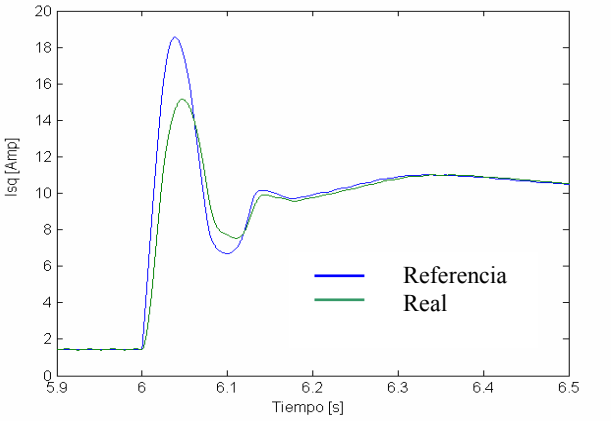


Figura 5.3.15. 2 Resultados de I_{sq} del control difuso.

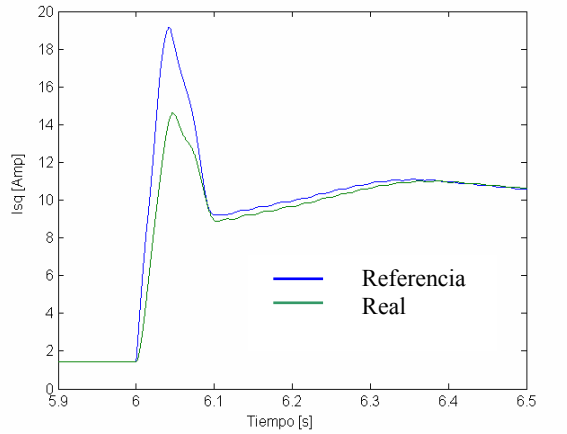


Figura 5.3.15. 3 Resultados de I_{sq} del control difuso. optimizado.

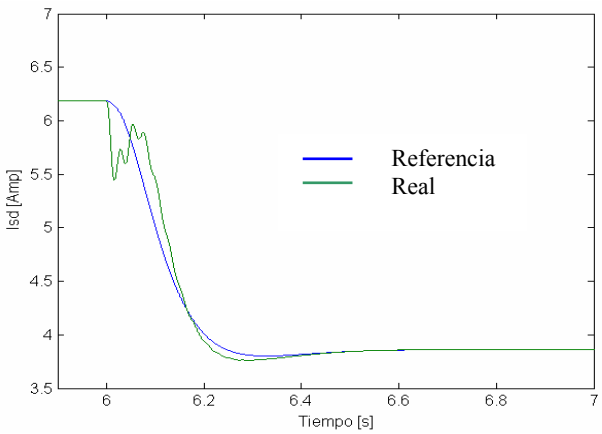


Figura 5.3.16. 1 Resultados de I_{sd} del control convencional.

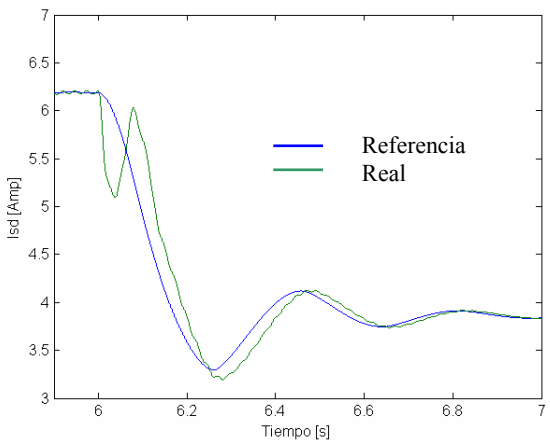


Figura 5.3.16. 2 Resultados de I_{sd} del control difuso.

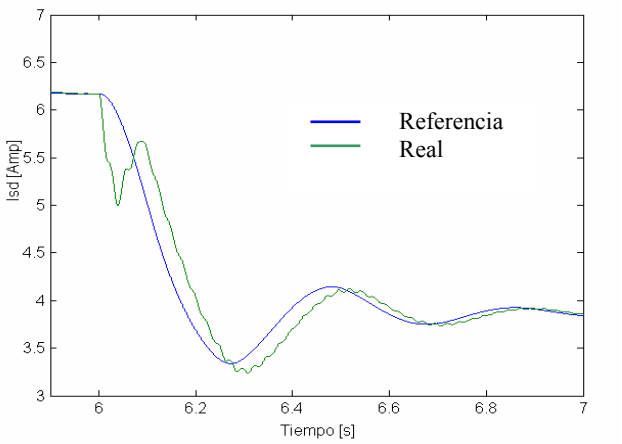


Figura 5.3.16. 3 Resultados de I_{sd} del control difuso optimizado.

En las figuras 5.3.15 se observan las señales de referencia de i_{sq} y los valores resultantes de i_{sq} para el control vectorial convencional, el control difuso y el control difuso optimizado por Búsqueda Tabú respectivamente. En las figuras 5.3.16 se muestran los resultados de las señales de referencia para i_{sd} y los resultados de i_{sd} para los controles propuestos con tablas de 25 reglas.

5.3.3 Autosintonización de 25 reglas a diferentes disturbios

La Búsqueda Tabú es utilizada para sintonizar la tabla de 25 reglas, cuando el sistema es llevado de carga nominal al valor de carga cero (primer disturbio), con la base de reglas obtenida de la autosintonización del primer disturbio, se somete el sistema a un nuevo cambio en la carga, llevando el sistema de carga cero a carga nominal (segundo disturbio) obteniendo así la tabla final optimizada para los dos disturbios. Se presentan los resultados de velocidad, flujo y corriente, además de los diagramas de fase y lingüísticos antes y después del uso de la Búsqueda Tabú.

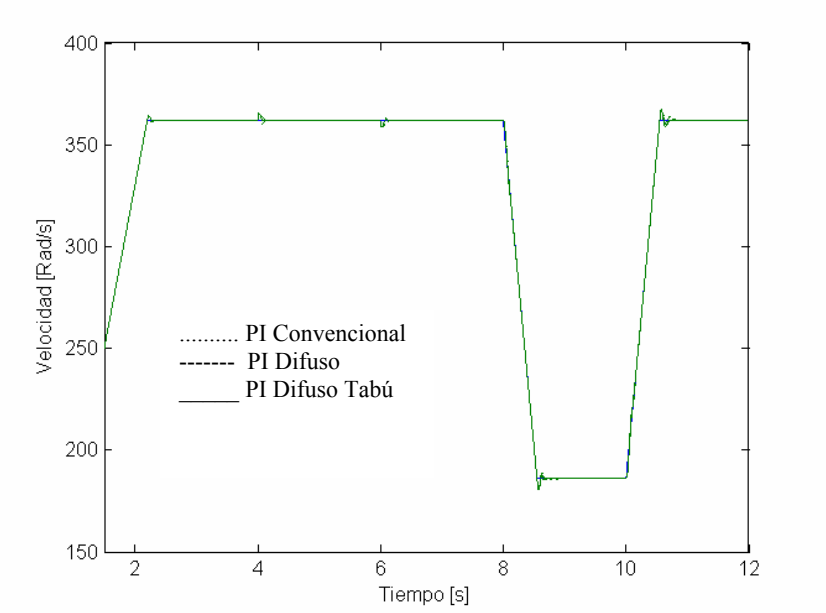


Figura 5.3.17 Resultados de velocidad de las simulaciones a plena carga, sin carga, y decremento de velocidad e incremento de velocidad comportamiento del control PI convencional, PI difuso y PI difuso Tabú.

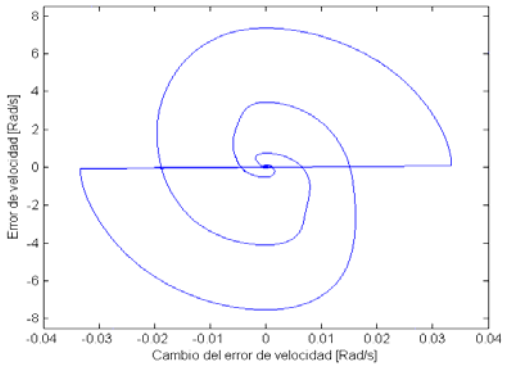
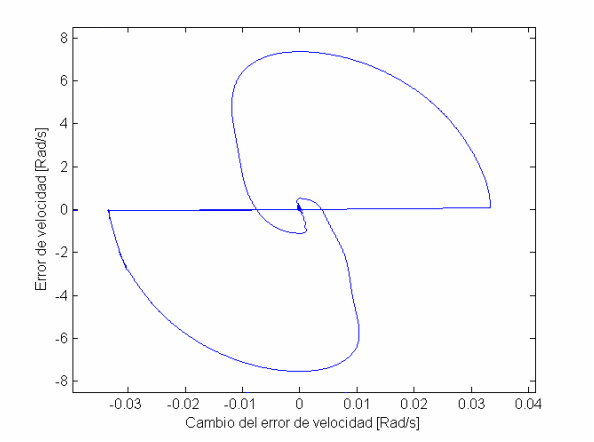


Figura 5.3.18 Diagrama de fase tiempo.

BM	BM	BM	BP	N
BM	BM	BP	N	SP
BM	BP	N	SP	SM
BP	N	SP	SM	SM
N	SP	SM	SM	SM

Figura 5.3.19 Diagrama de fase lingüístico.

En la figura 5.3.17 se observan los resultados de velocidad de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú. El primer disturbio es producido en t = 4 segundos al retirar la carga nominal (11.9 Nm) y llevarla hasta el valor de cero, el segundo disturbio se genera al aplicar carga nominal en t = 6 segundos en t =8 segundos se da un decremento en la velocidad de referencia y se estabiliza en t = 8.55 segundos, por último en t = 10 segundos se incrementa la velocidad hasta 362 Rad/s. La figura 5.3.18 muestra el diagrama de fase tiempo resultante del sistema con el control difuso y la figura 5.3.19 muestra el diagrama de fase lingüístico del control difuso ambos sin optimizar solo para los disturbios de cambio de carga.



BM	BM	BM	BP	N
BM	BM	BP	SP	SM
BM	BP	N	SP	SM
BM	BP	SP	SM	SM
N	SP	SM	SM	SM

Figura 5.3.20 Diagrama de fase tiempo. **Figura 5.3.21 Diagrama de fase lingüístico.**

En la figura 5.3.20 se muestra el diagrama fase tiempo del control difuso optimizado, y en la figura 5.3.21 se observa el diagrama de fase lingüístico resultante del sistema con la memoria asociativa optimizada ambas para los disturbios en la carga. Es posible observar el cambio de cuatro reglas que mejoran los diagramas de fase, al llevar al centro las trayectorias generadas por el sistema con menor oscilación.

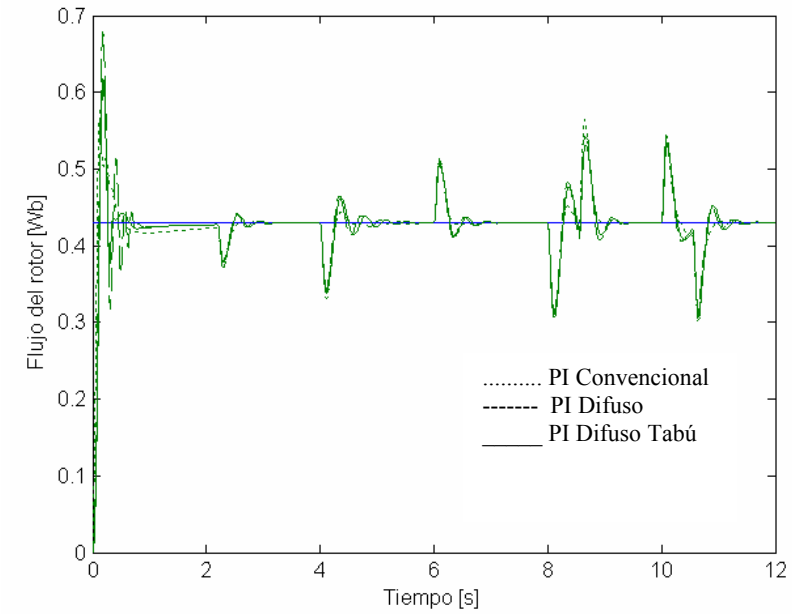


Figura 5.3.22 Resultados del flujo del rotor de las simulaciones a diferentes disturbios.

En la figura 5.3.22 se observan los resultados del flujo del rotor de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú. Los controles difusos utilizan la misma memoria asociativa difusa de 25 reglas.

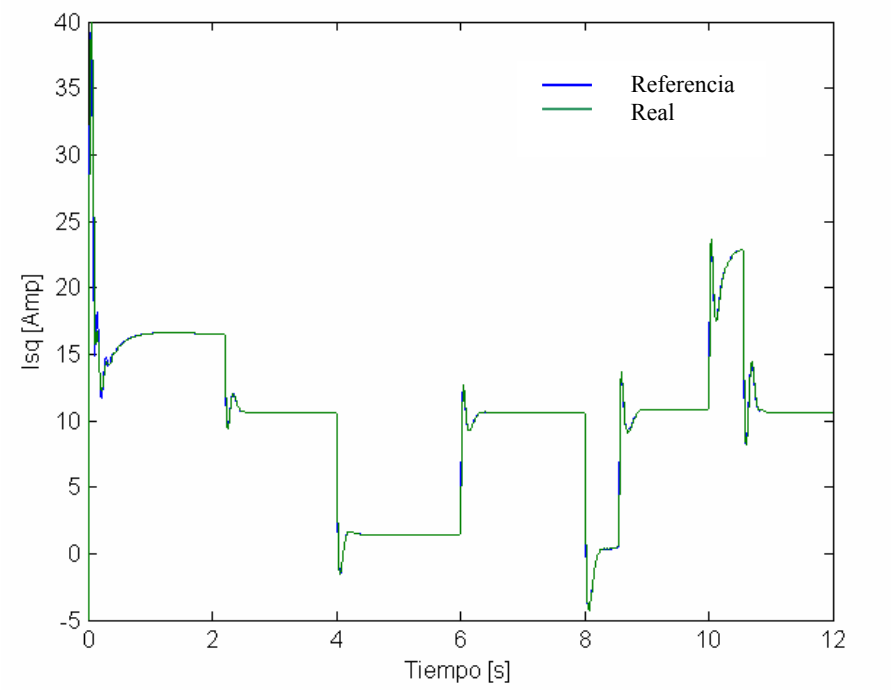


Figura 5.3.23. 1 Resultados de I_{sq} del control convencional.

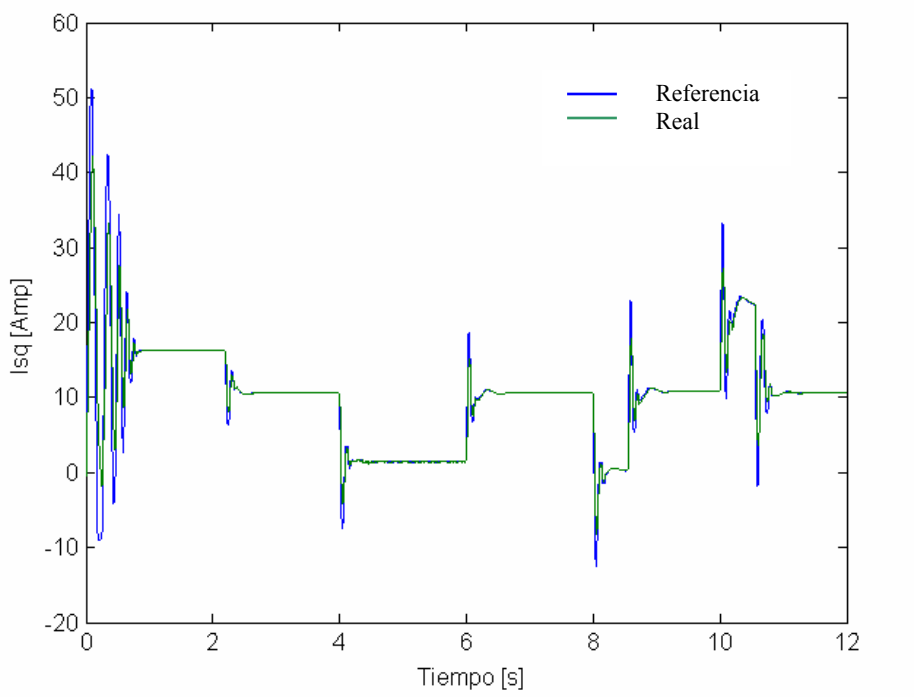


Figura 5.3.23. 2 Resultados de I_{sq} del control difuso.

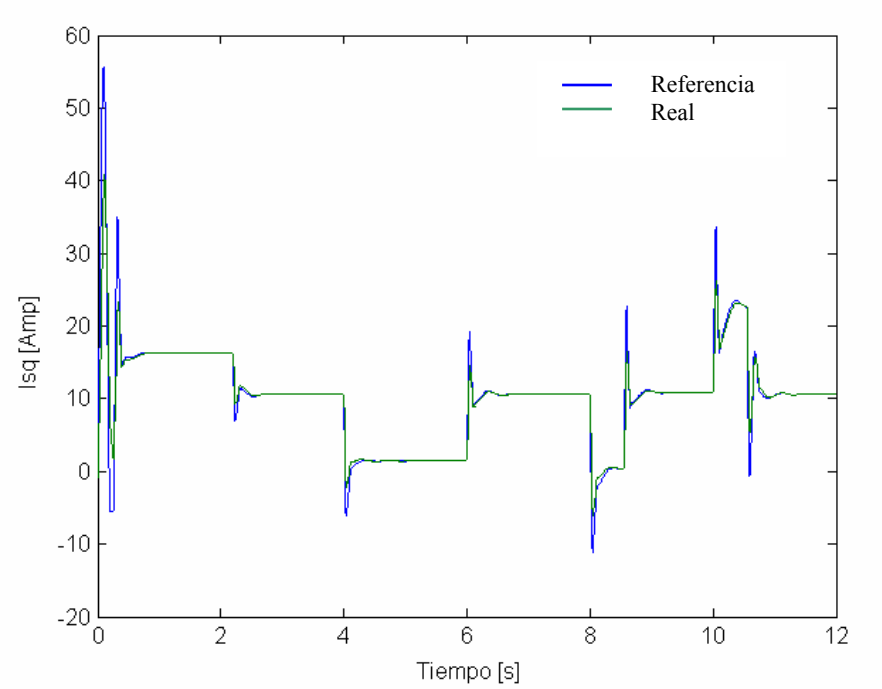


Figura 5.3.23. 3 Resultados de I_{sq} del control difuso optimizado

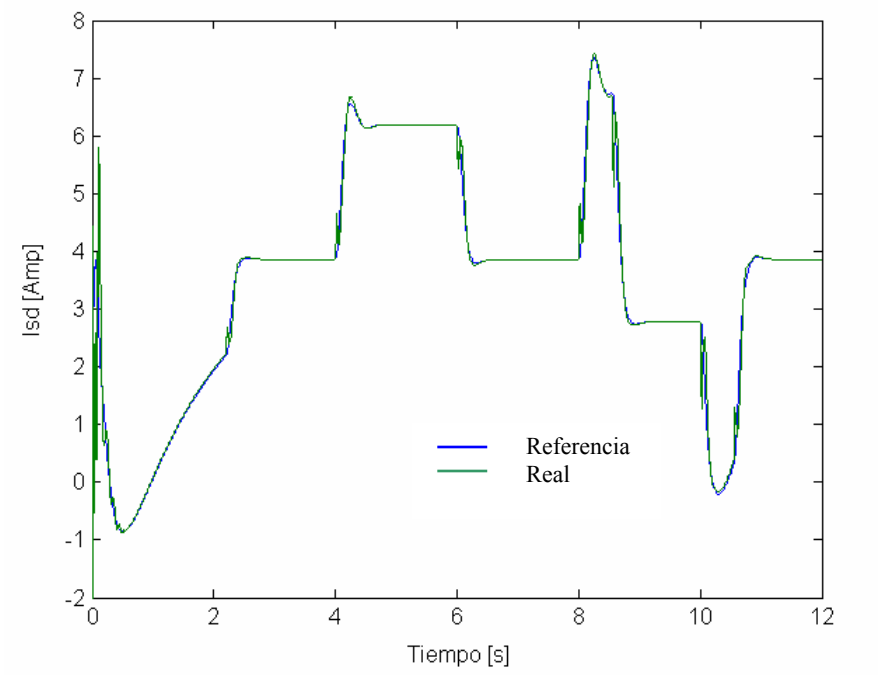


Figura 5.3.24. 1 Resultados de I_{sd} del control convencional.

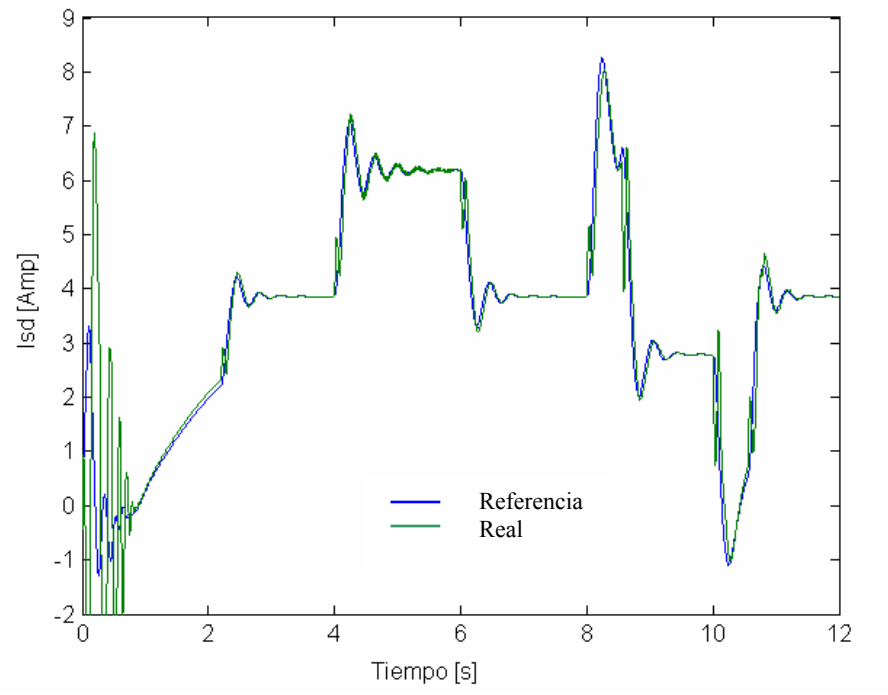


Figura 5.3.24. 2 Resultados de I_{sd} del control difuso.

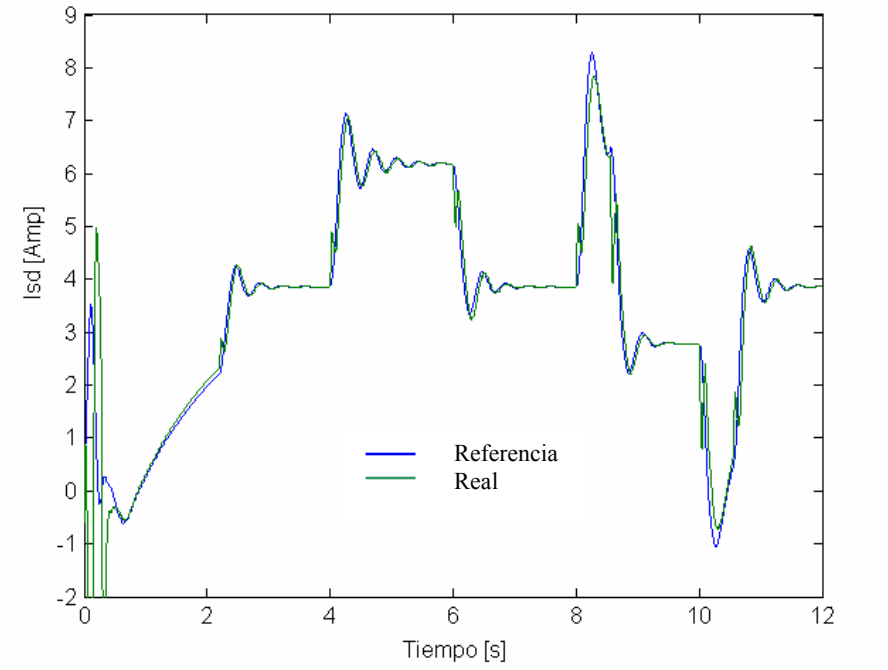


Figura 5.3.24. 3 Resultados de I_{sd} del control difuso optimizado.

En las figuras 5.3.23 se observan las señales de referencia de i_{sq} y los valores resultantes de i_{sq} para el control vectorial convencional, el control difuso y el control difuso optimizado por Búsqueda Tabú para los disturbios en la carga respectivamente.

En las figuras 5.3.24 se muestran los resultados de las señales de referencia para i_{sd} y los resultados de i_{sd} para el control vectorial, control difuso, y optimizado por Búsqueda Tabú respectivamente este último propuesto con tablas de 25 reglas .

5.4 AUTOSINTONIZACIÓN DE 49 REGLAS LINGÜÍSTICAS

En esta sección se presentan los resultados obtenidos al simular el sistema con el control PI convencional, el control difuso y los controles difusos tabú estos últimos con una base de 49 reglas. Para la primera prueba se aplica un disturbio al quitar la carga nominal y llevarla al valor de cero.

La segunda prueba se realiza al llevar el sistema de carga cero al valor de carga nominal. Por último se utiliza la tabla autosintonizada por la Búsqueda Tabú para el primer disturbio y se autosintoniza el sistema para el segundo disturbio y la tabla resultante se emplea en la simulación de todos disturbios propuestos y los cambios en la velocidad de referencia.

Se presentan además los diagramas de fase tiempo y lingüísticos, con la base de reglas no optimizada y optimizada para la observación en los cambios de las trayectorias.

Se incluyen además de las gráficas de velocidad las gráficas de flujo y corrientes en los ejes d-q, para cada disturbio en particular y al final de forma general.

Los índices de funcionamiento del control vectorial convencional, difuso y difuso tabú para las tres pruebas se presentan juntos en la sección 5.6.

5.4.1 Autosintonización de 49 reglas 1er disturbio

La Búsqueda Tabú es utilizada para sintonizar la tabla de 49 reglas, cuando el sistema es llevado de carga nominal al valor de carga cero, se presentan los resultados de velocidad, flujo y corriente, además de los diagramas de fase y lingüísticos antes y después del uso de la Búsqueda Tabú.

En las gráficas se presentan los resultados del control vectorial, control difuso y el control difuso optimizado con la Búsqueda Tabú. Los resultados en la memoria asociativa difusa de la sintonización con respecto al primer disturbio son usados posteriormente como una tabla de reglas inicial para poder sintonizar la base de reglas del segundo disturbio, completando el procedimiento para la sintonización de diferentes disturbios con una sola base de reglas.

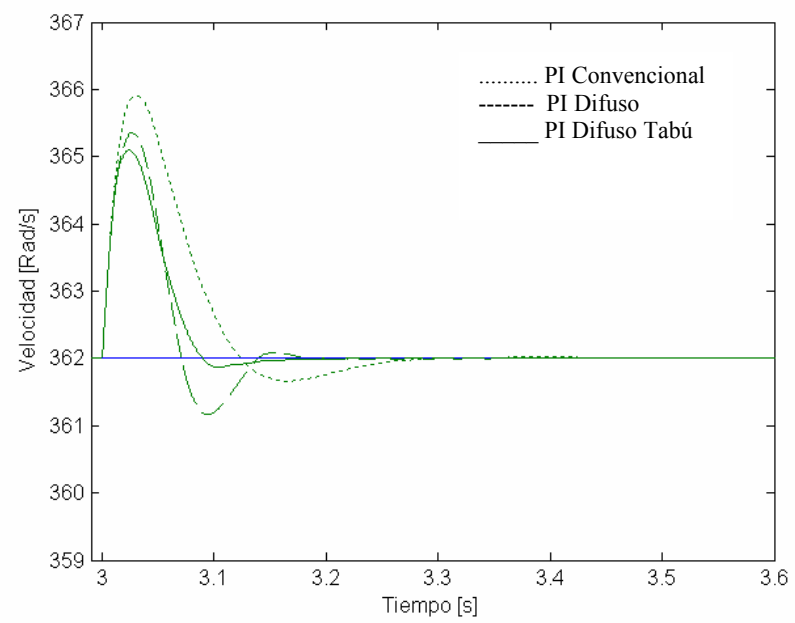
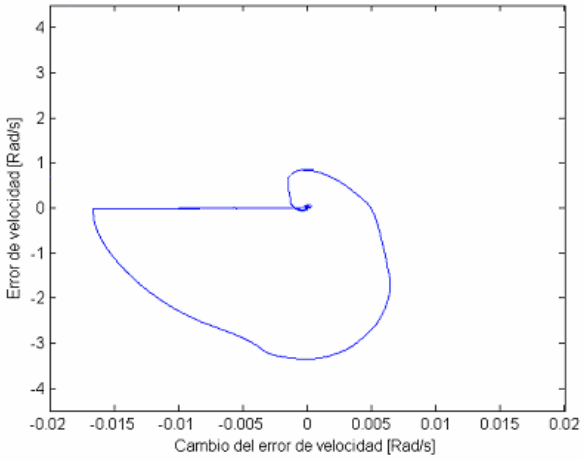


Figura 5.4. 1 Resultados de velocidad de las simulaciones al retirar la carga nominal del sistema.

En la figura 5.4.1 se observan los resultados de velocidad de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú. El disturbio es producido en t = 3 segundos al retirar la carga nominal (11.9 Nm) y llevarla hasta el valor de cero.



BMO	BMO	BMO	BME	BP	BP	N
BMO	BMO	BME	BP	BP	N	SP
BMO	BME	BP	BP	N	SP	SP
BME	BP	BP	N	SP	SP	SME
BP	BP	N	SP	SP	SME	SMO
BP	N	SP	SP	SME	SMO	SMO
N	SP	SP	SME	SMO	SMO	SMO

Figura 5.4. 2 Diagrama de fase tiempo.

Figura 5.4. 3 Diagrama de fase lingüístico.

La figura 5.4.2 muestra el diagrama de fase tiempo resultante del sistema con el control difuso y la figura 5.4.3 muestra el diagrama de fase lingüístico del control difuso sin optimizar.

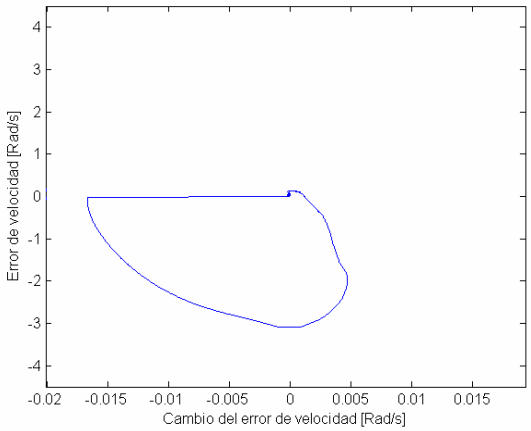


Figura 5.4. 4 Diagrama de fase tiempo.

BMO	BMO	BMO	BME	BP	BP	N
BMO	BMO	BME	BP	BP	N	SP
BMO	BME	BP	BP	SP	SP	SP
BME	BP	BP	N	SP	SP	SME
BP	BP	N	SP	SME	SME	SMO
BP	N	SP	SP	SME	SMO	SMO
N	SP	SP	SME	SMO	SMO	SMO

Figura 5.4. 5 Diagrama de fase lingüístico.

En la figura 5.4.4 se muestra el diagrama fase tiempo del control difuso optimizado, y en la figura 5.4.5 se observa el diagrama de fase lingüístico resultante del sistema con la memoria asociativa optimizada. Es posible notar si se compara la figura 5.4.3 con la figura 5.4.5 que la regla que cambio de más influencia es “Nada” por “Subir Poco”.

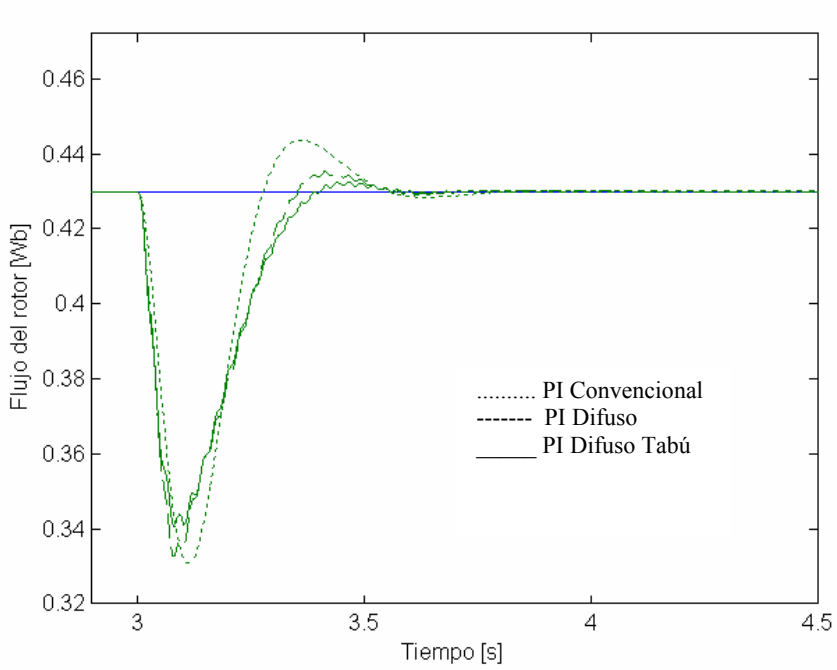


Figura 5.4. 6 Resultados del flujo del rotor de las simulaciones al retirar la carga nominal del sistema.

En la figura 5.4.6 se observan los resultados del flujo del rotor de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú. El disturbio es producido en t = 3 segundos al retirar la carga nominal (11.9 Nm) y llevarla hasta el valor de cero. Los controles difusos utilizan la misma memoria asociativa difusa de 49 reglas.

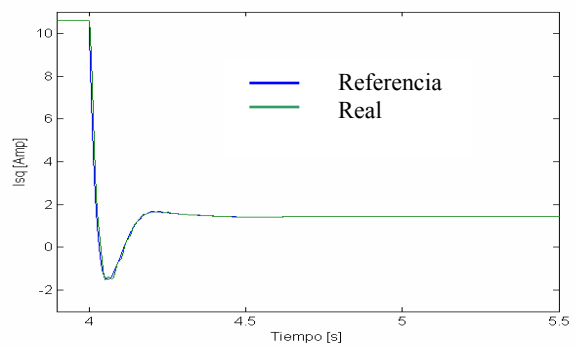


Figura 5.4.7. 1 Resultados de I_{sq} del control convencional.

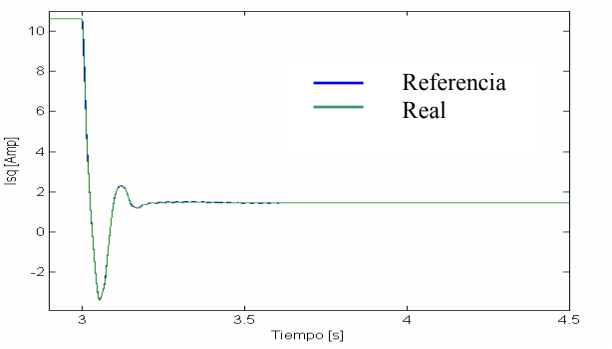


Figura 5.4.7. 2 Resultados de I_{sq} del control difuso.

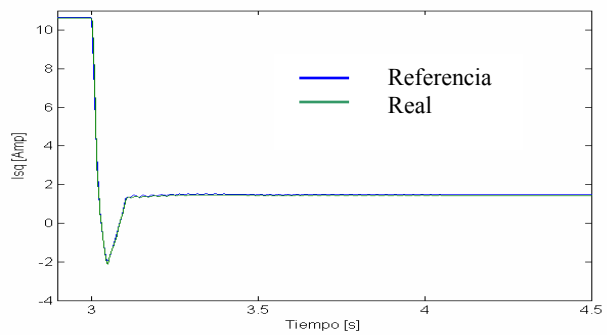


Figura 5.4.7. 3 Resultados de I_{sq} del control difuso optimizado.

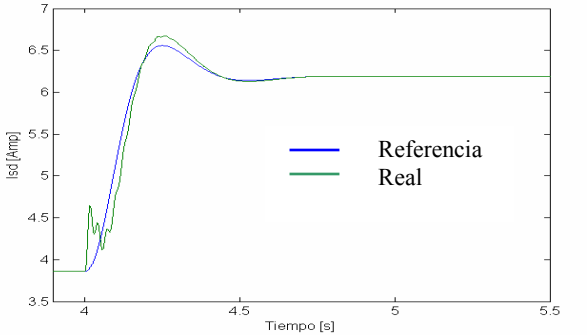


Figura 5.4.8. 1 Resultados de I_{sd} del control convencional.

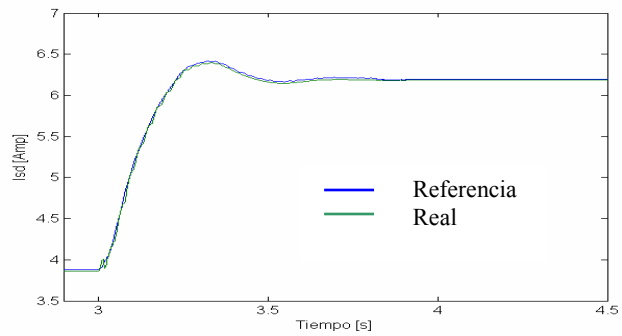


Figura 5.4.8. 2 Resultados de I_{sd} del control difuso.

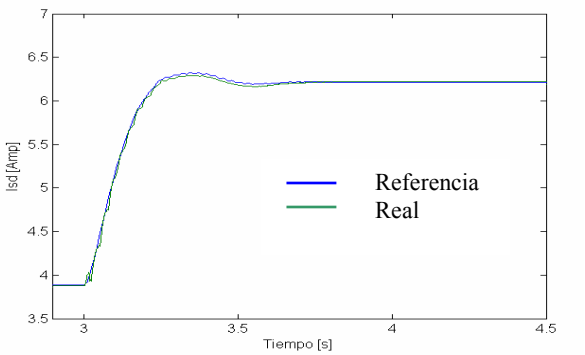


Figura 5.4.8. 3 Resultados de I_{sd} del control difuso optimizado.

En las figuras 5.4.7 se observan las señales de referencia de $_{isq}$ y los valores resultantes de i_{sq} para en control vectorial convencional, el control difuso y el control difuso optimizado por Búsqueda Tabú respectivamente. En las figuras 5.4.8 se muestran los resultados de las señales de referencia para i_{sd} y los resultados de i_{sd} para los controles propuestos con tablas de 49 reglas.

En estas pruebas el control difuso muestra gran capacidad para seguir la señal de referencia, en contraste con los controles convencionales.

5.4.2 Autosintonización de 49 reglas 2do disturbio

La Búsqueda Tabú es utilizada para sintonizar la tabla de 49 reglas, cuando el sistema es llevado de carga cero al valor de carga nominal, se presentan los resultados de velocidad, flujo y corriente, además de los diagramas de fase y lingüísticos antes y después del uso de la Búsqueda Tabú.

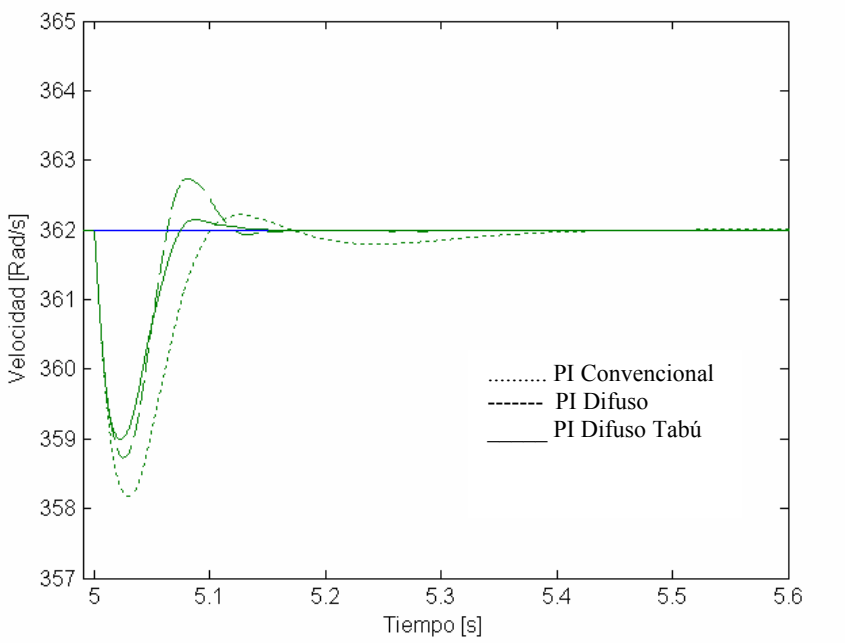


Figura 5.4.9 Resultados de velocidad de las simulaciones al retirar la carga nominal del sistema.

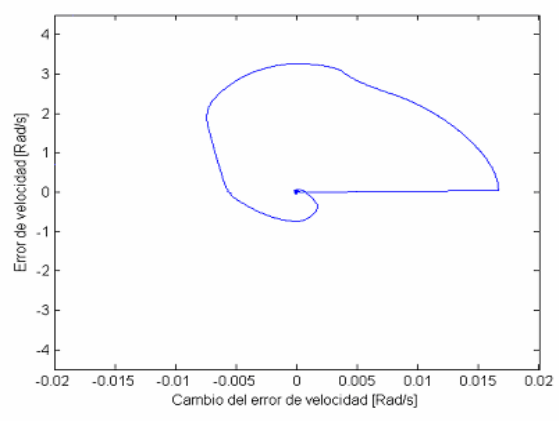


Figura 5.4.10 Diagrama de fase tiempo.

BMO	BMO	BMO	BME	BP	BP	N
BMO	BMO	BME	BP	BP	N	SP
BMO	BME	BP	BP	N	SP	SP
BME	BP	BP	N	SP	SP	SME
BP	BP	N	SP	SP	SME	SMO
BP	N	SP	SP	SME	SMO	SMO
N	SP	SP	SME	SMO	SMO	SMO

Figura 5.4.11 Diagrama de fase lingüístico.

En la figura 5.4.9 se observan los resultados de velocidad de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú. El disturbio es producido en t = 5 segundos al llevar el sistema de carga cero al valor de carga nominal (11.9 Nm). La figura 5.4.10 muestra el diagrama de fase tiempo resultante del sistema con el control difuso y la figura 5.4.11 muestra el diagrama de fase lingüístico del control difuso sin optimizar.

En la figura 5.4.9 es posible observar a simple vista la mejora de que se obtiene con el control difuso y la mejora que realiza la optimización del sistema con la búsqueda por tabú.

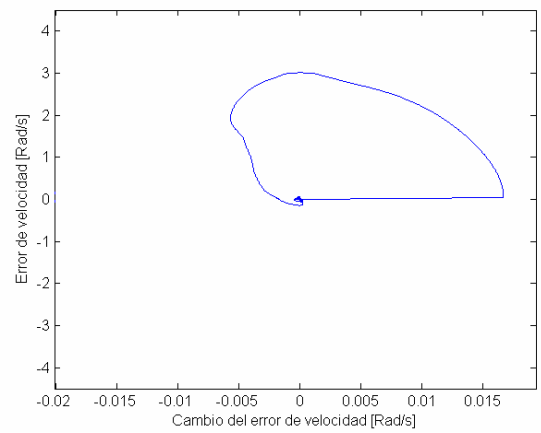


Figura 5.4.12 Diagrama de fase tiempo.

BMO	BMO	BMO	BME	BP	BP	N
BMO	BMO	BME	BP	BP	N	SP
BMO	BME	BME	BP	N	SP	SP
BME	BP	BP	N	SP	SP	SME
BP	BP	BP	SP	SP	SME	SMO
BP	N	SP	SP	SME	SMO	SMO
N	SP	SP	SME	SMO	SMO	SMO

Figura 5.4.13 Diagrama de fase lingüístico

En la figura 5.4.12 se muestra el diagrama fase tiempo del control difuso optimizado, y en la figura 5.4.13 se observa el diagrama de fase lingüístico resultante del sistema con la memoria asociativa optimizada.

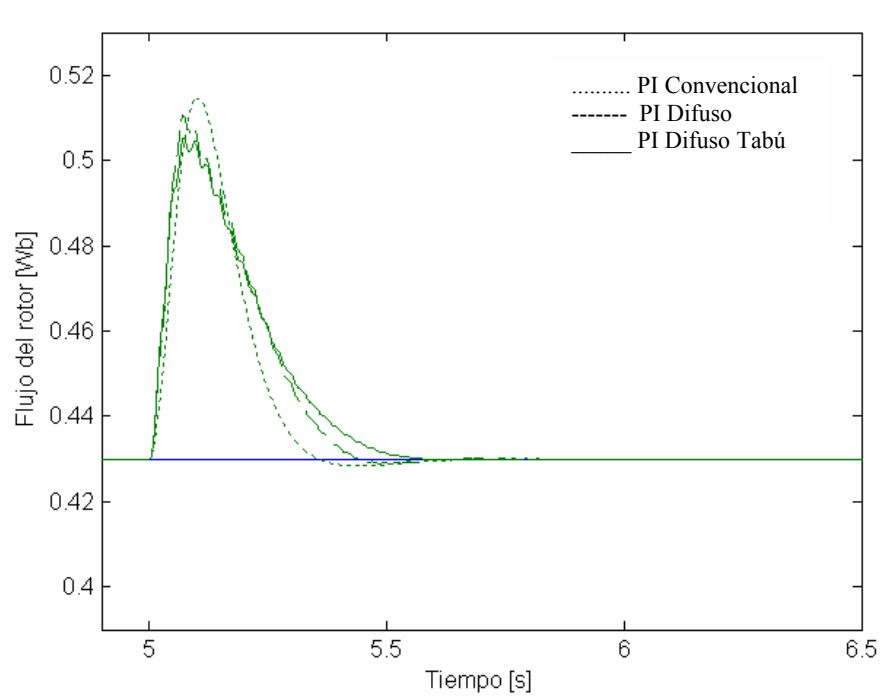


Figura 5.4. 14 Resultados del flujo del rotor de las simulaciones al aplicar carga nominal al sistema.

En la figura 5.4.14 se observan los resultados del flujo del rotor de la simulación del sistema con el control convencional, el control difuso y el control difuso auto sintonizado con la Búsqueda Tabú. El disturbio es producido en t = 5 segundos al llevar la carga de cero hasta el valor de carga nominal (11.9 Nm). Los controles difusos utilizan la misma memoria asociativa difusa de 49 reglas.

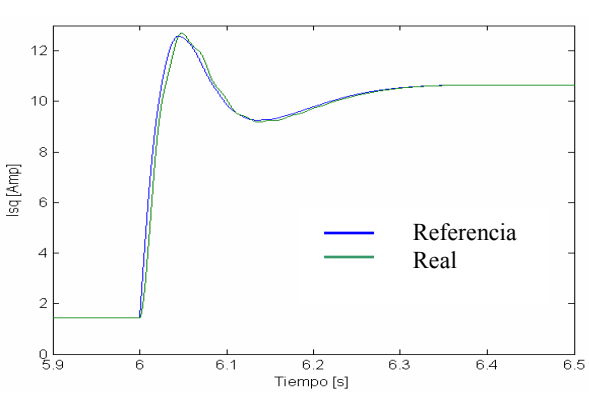


Figura 5.4.15. 1 Resultados de $I_{sq.}$ del control convencional.

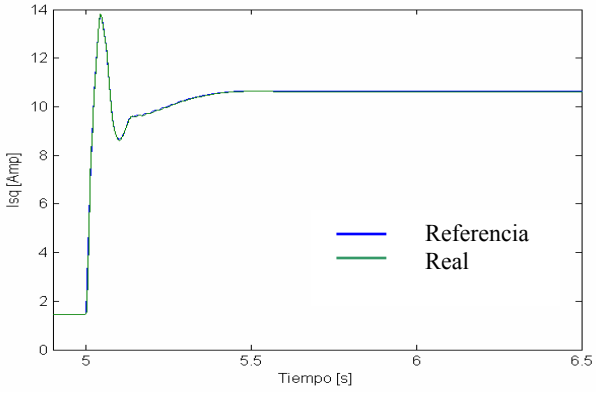


Figura 5.4.15. 2 Resultados de $I_{sq.}$ Del control difuso.

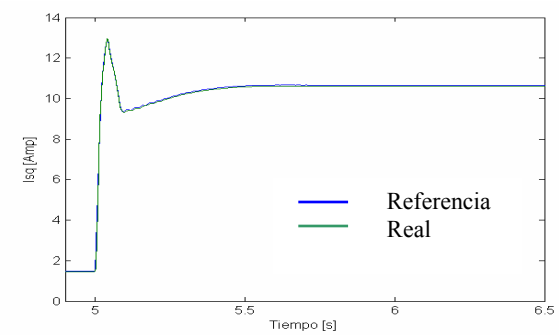


Figura 5.4.15. 3 Resultados de $I_{sq.}$ del control. difuso optimizado

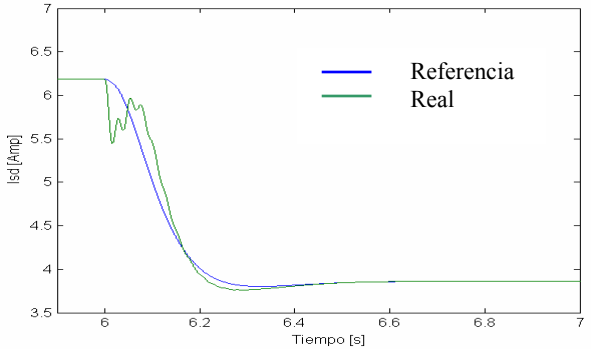


Figura 5.4.16. 1 Resultados de I_{sd} del control convencional.

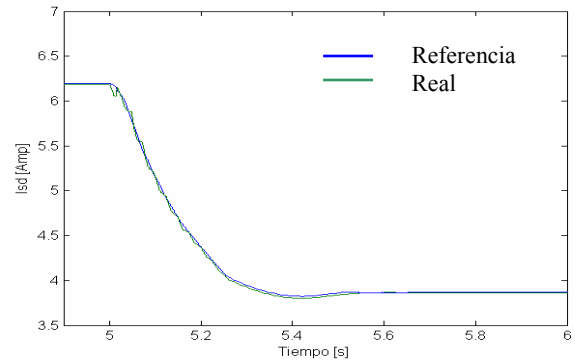


Figura 5.4.16. 2 Resultados de I_{sd} del control difuso.

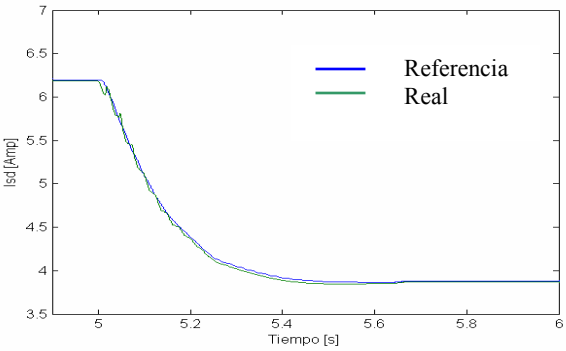


Figura 5.4.16. 3 Resultados de I_{sd} del control difuso optimizado.

En las figuras 5.4.15 se observan las señales de referencia de $i_{sq.}$ y los valores resultantes de $i_{sq.}$ para el control vectorial convencional, el control difuso y el control difuso optimizado por Búsqueda Tabú. En las figuras 5.4.16 se muestran los resultados de las señales de referencia para i_{sd} y los resultados de i_{sd} para los controles propuestos con tablas de 49 reglas. Se observa un mejor seguimiento de la señal de referencia de los controles difusos, que del control convencional.

5.4.3 Autosintonización de 49 reglas a diferentes disturbios

La Búsqueda Tabú es utilizada para sintonizar la tabla de 49 reglas, cuando el sistema es llevado de carga nominal al valor de carga cero (primer disturbio), con la base de reglas obtenida de la autosintonización del primer disturbio, se somete el sistema a un nuevo cambio en la carga, llevando el sistema de carga cero a carga nominal (segundo disturbio) obteniendo así la tabla final optimizada para los dos disturbios. Se presentan los resultados de velocidad, flujo y corriente, además de los diagramas de fase y lingüísticos antes y después del uso de la Búsqueda Tabú

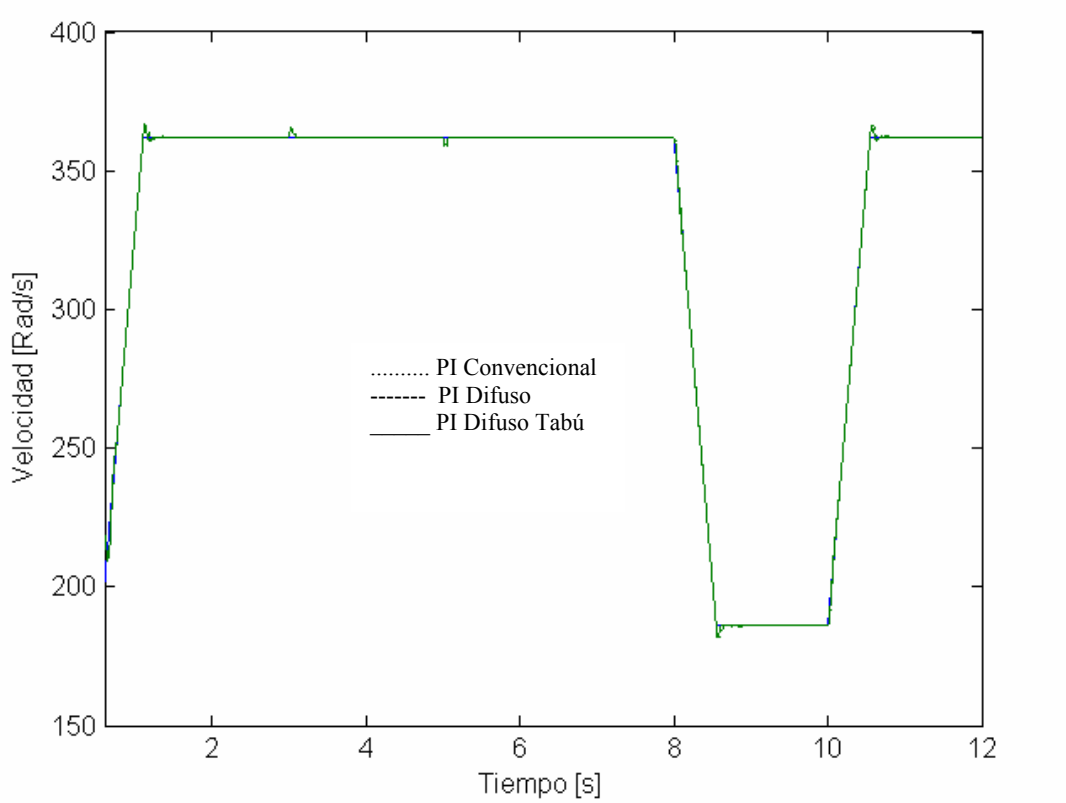


Figura 5.4.17 Resultados de velocidad de las simulaciones a plena carga, sin carga, decremento de velocidad e incremento de velocidad, comportamiento del control PI convencional, PI difuso y PI difuso Tabú.

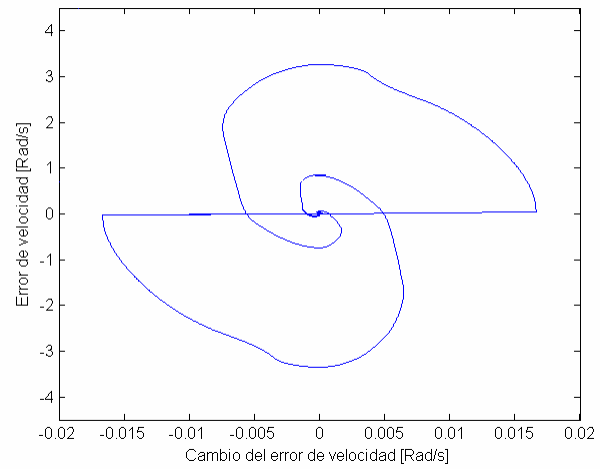


Figura 5.4.18 Diagrama de fase tiempo.

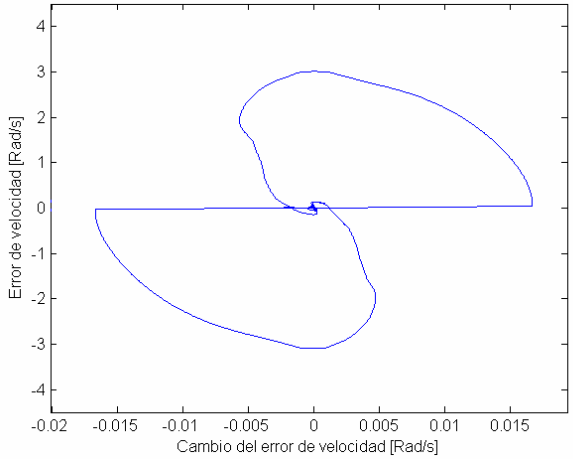
BMO	BMO	BMO	BME	BP	BP	N
BMO	BMO	BME	BP	BP	N	SP
BMO	BME	BP	BP	N	SP	SP
BME	BP	BP	N	SP	SP	SME
BP	BP	N	SP	SP	SME	SMO
BP	N	SP	SP	SME	SMO	SMO
N	SP	SP	SME	SMO	SMO	SMO

Figura 5.4.19 Diagrama de fase lingüístico.

En la figura 5.4.17 se observan los resultados de velocidad de la simulación del sistema con el control convencional, el control difuso y el control difuso auto sintonizado con la Búsqueda Tabú.

El primer disturbio es producido en t = 3 segundos al retirar la carga nominal (11.9 Nm) y llevarla hasta el valor de cero, el segundo disturbio se genera al aplicar carga nominal en t = 5 segundos en t =8 segundos se decremento la velocidad de referencia y se estabiliza en t = 8.55 segundos, por último en t = 10 segundos se incrementa la velocidad hasta 362 Rad/s.

La figura 5.4.18 muestra el diagrama de fase tiempo resultante del sistema con el control difuso y la figura 5.4.19 muestra el diagrama de fase lingüístico del control difuso ambos sin optimizar solo para los disturbios del cambio de carga.



BMO	BMO	BMO	BME	BP	BP	N
BMO	BMO	BME	BP	BP	N	SP
BMO	BME	BME	BP	SP	SP	SP
BME	BP	BP	N	SP	SP	SME
BP	BP	BP	SP	SME	SME	SMO
BP	N	SP	SP	SME	SMO	SMO
N	SP	SP	SME	SMO	SMO	SMO

Figura 5.4.20 Diagrama de fase tiempo. **Figura 5.4.21 Diagrama de fase lingüístico.**

En la figura 5.4.20 se muestra el diagrama fase tiempo del control difuso optimizado, y en la figura 5.4.21 se observa el diagrama de fase lingüístico resultante del sistema con la memoria asociativa optimizada ambas para los disturbios en la carga. Se observa al comparar la figura 5.4.19 y la figura 5.4.21 que las reglas que cambian se encuentran en la trayectoria que sigue el sistema, por lo cual la modifican.

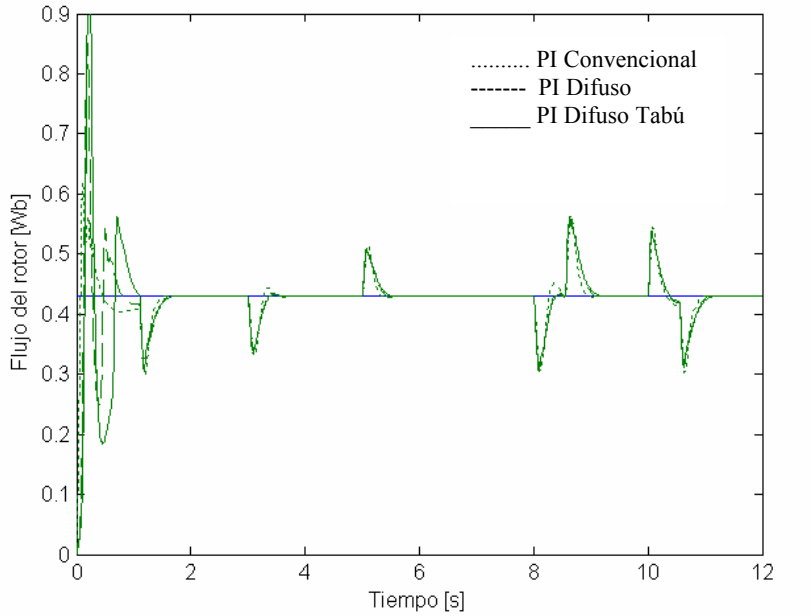


Figura 5.4.22 Resultados del flujo del rotor de las simulaciones a diferentes disturbios.

En la figura 5.4.22 se observan los resultados del flujo del rotor de la simulación del sistema con el control convencional, el control difuso y el control difuso autosintonizado con la Búsqueda Tabú. Los controles difusos utilizan la misma memoria asociativa difusa de 49 reglas.

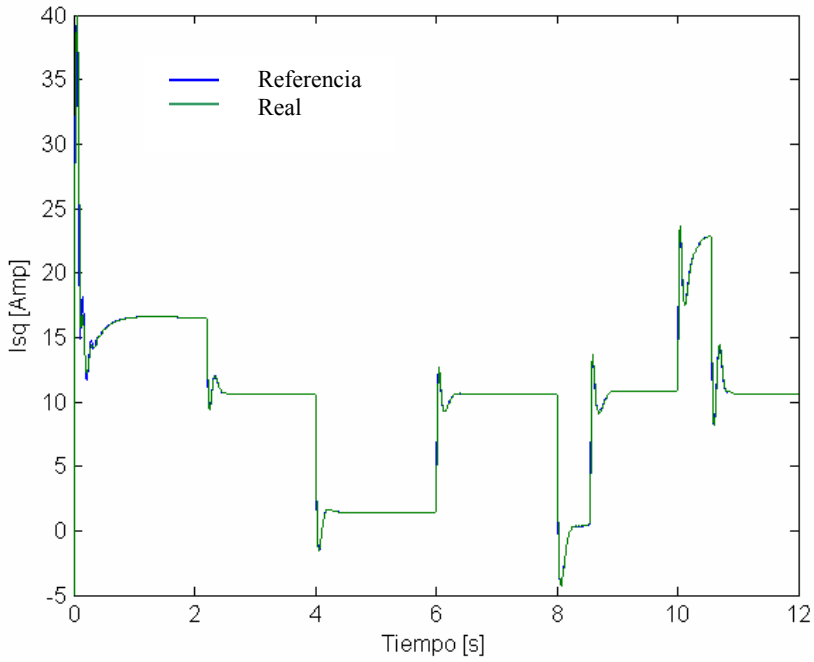


Figura 5.4.23. 1 Resultados de I_{sq} del control convencional.

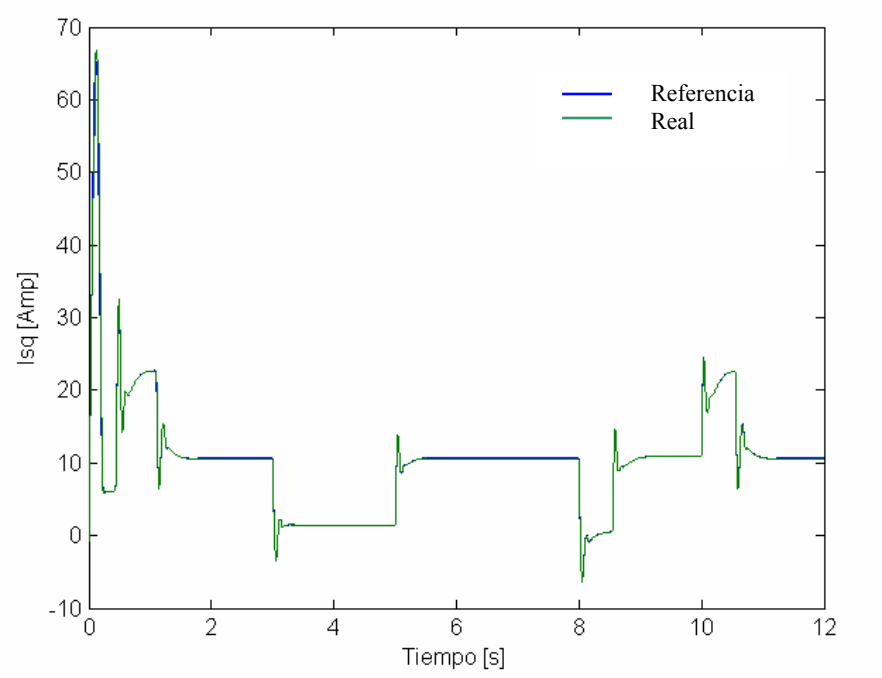


Figura 5.4.23. 2 Resultados de I_{sq} del control difuso.

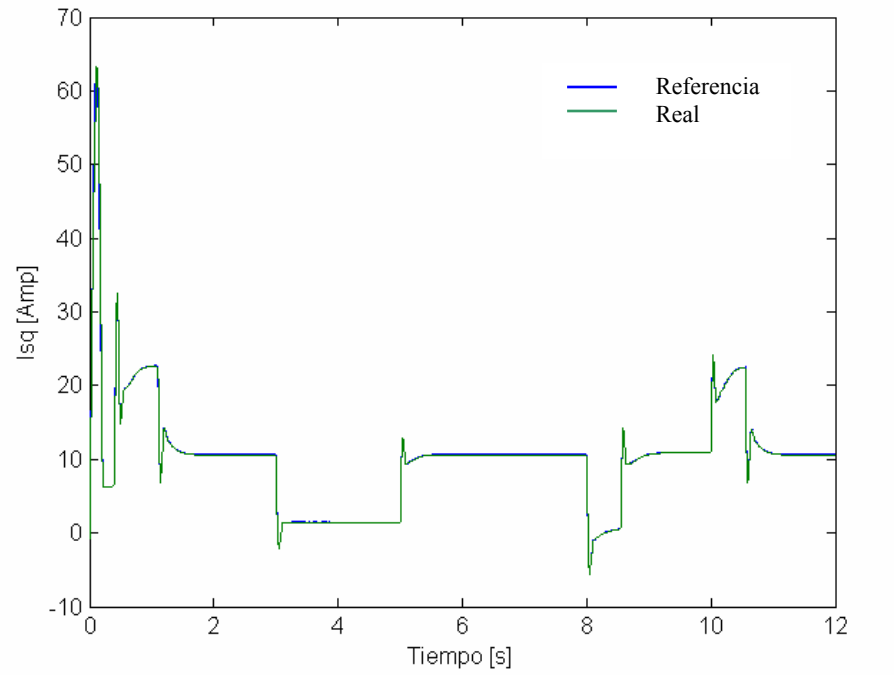


Figura 5.4.23. 3 Resultados de I_{sq} del control difuso optimizado

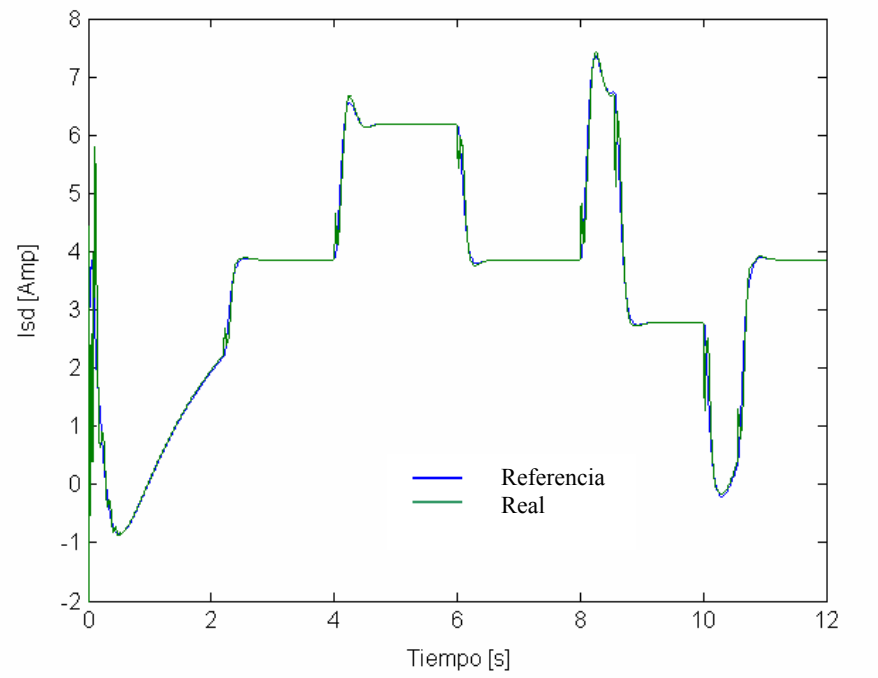


Figura 5.4.24. 1 Resultados de I_{sd} del control convencional.

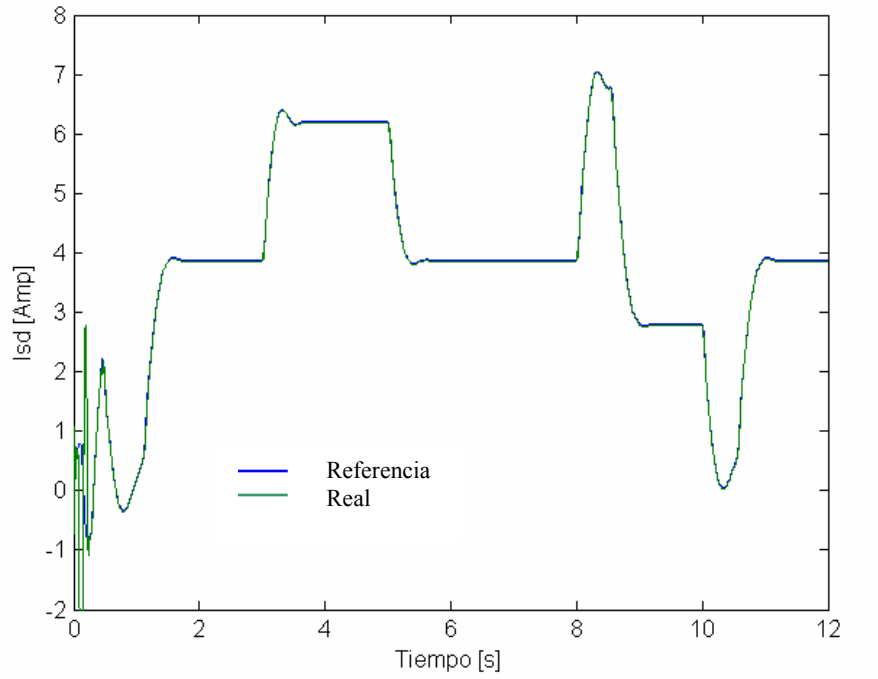


Figura 5.4.24. 2 Resultados de I_{sd} del control difuso.

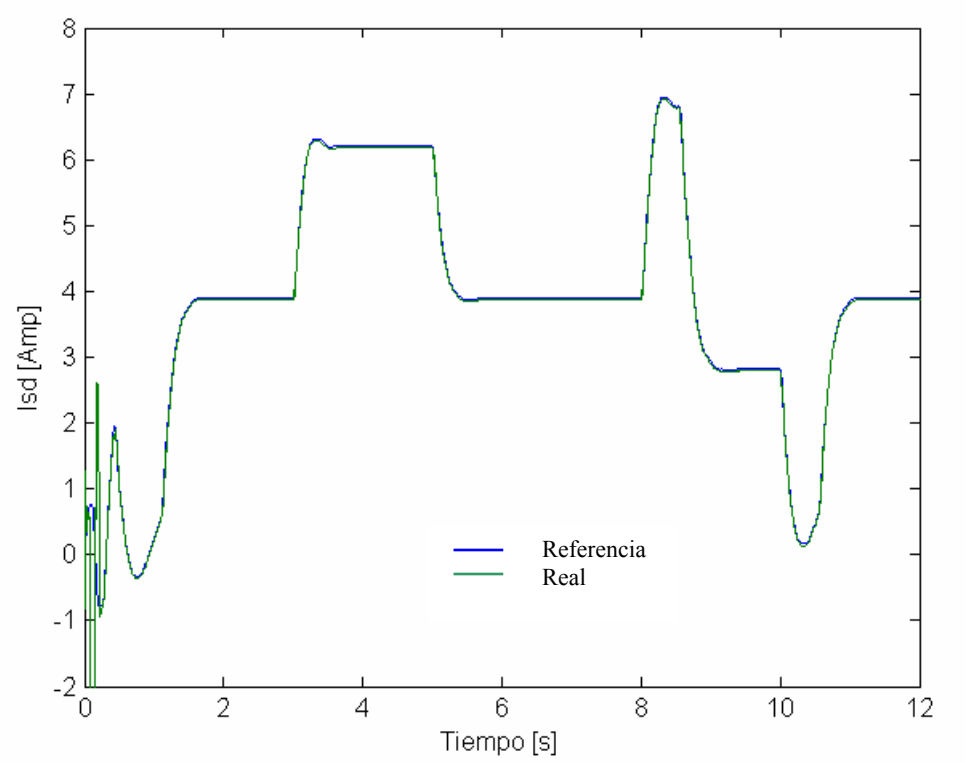


Figura 5.4.24. 3 Resultados de I_{sd} del control difuso optimizado.

En las figuras 5.4.23 se observan las señales de referencia de i_{sq} y los valores resultantes de i_{sq} para el control vectorial convencional, el control difuso y el control difuso optimizado por Búsqueda Tabú para los disturbios en la carga respectivamente. En las figuras 5.4.24 se muestran los resultados de las señales de referencia para i_{sd} y los resultados de i_{sd} para el control vectorial, control difuso, y optimizado por Búsqueda Tabú respectivamente este último propuesto con tablas de 49 reglas.

En los diagramas de fase lingüísticos se observan las memorias asociativas difusas optimizadas por la Búsqueda por Tabú, por lo que no se presentan de manera individual.

5.5 ÍNDICES DE COMPORTAMIENTO OBTENIDOS AL SINTONIZAR LA BASE DE 25 REGLAS

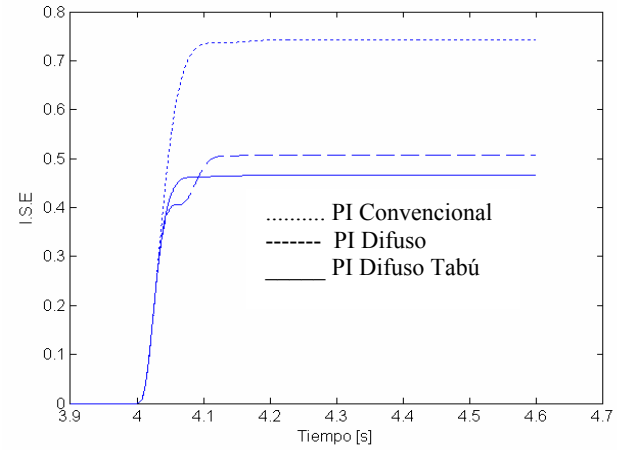


Figura 5.5. 1 I.S.E de velocidad 1er disturbio.

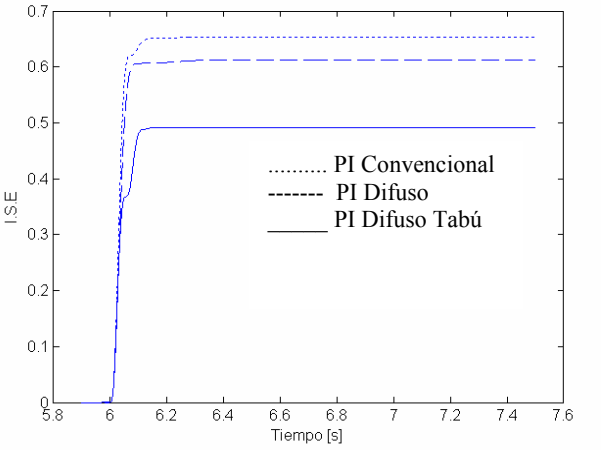


Figura 5.5. 2 I.S.E de velocidad 2do disturbio.

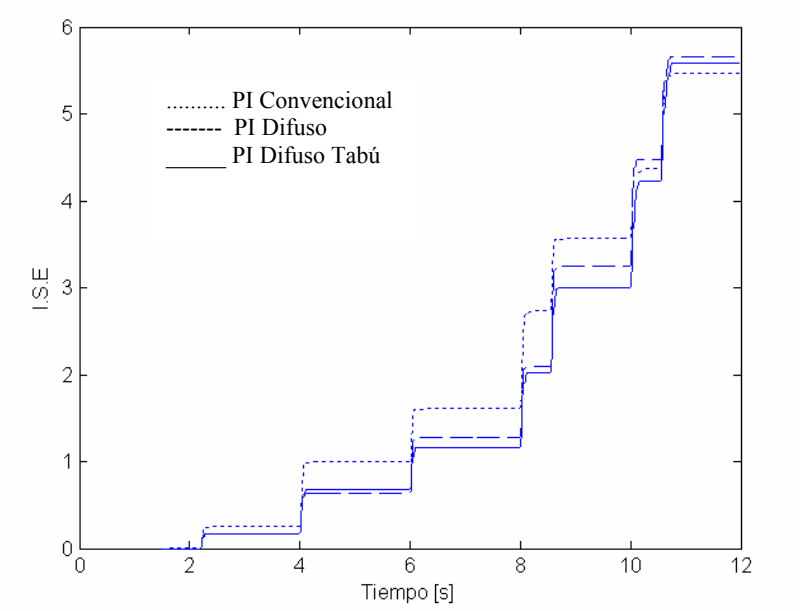


Figura 5.5. 3 I.S.E de velocidad con la simulación de todos los disturbios.

En las figuras 5.5 se muestran los índices de funcionamiento del control convencional, difuso y difuso tabú con bases de 25 reglas, para los disturbios presentados anteriormente. Se presentan los índices con el fin de enfatizar en las diferencias que se observan en los resultados de las gráficas de velocidad presentados en la sección 5.3.

5.6 ÍNDICES DE COMPORTAMIENTO OBTENIDOS AL SINTONIZAR LA BASE DE 49 REGLAS

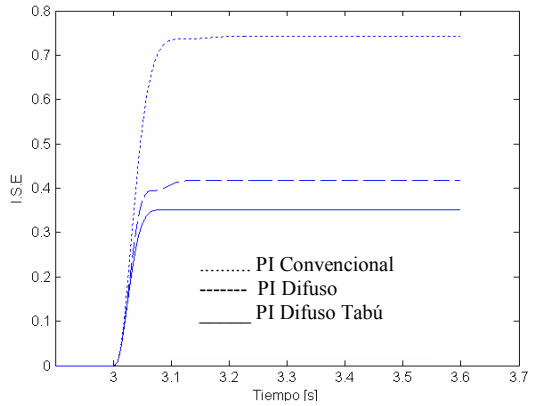


Figura 5.6. 1 I.S.E de velocidad 1er disturbio.

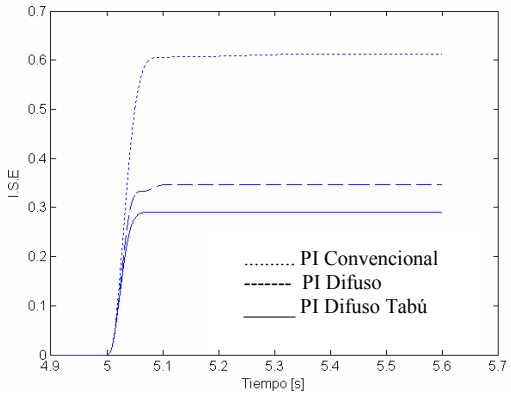


Figura 5.6. 2 I.S.E de velocidad 2do disturbio.

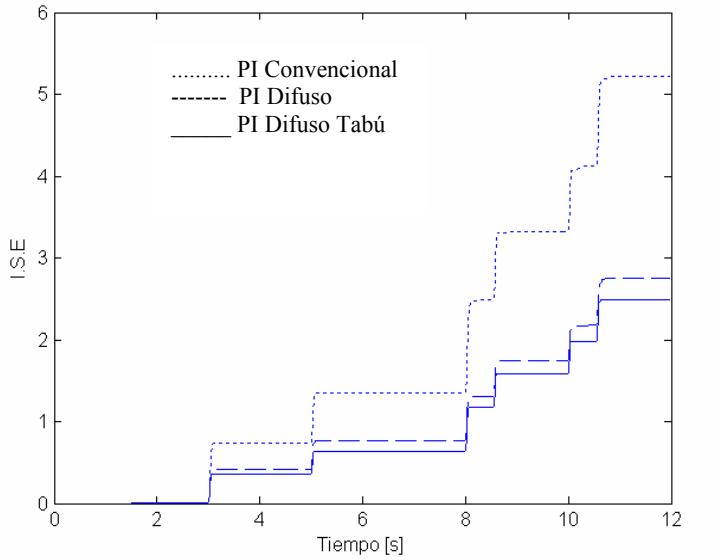


Figura 5.6. 3 I.S.E de velocidad utilizando todos los disturbios.

En las figuras 5.6 se muestran los índices de funcionamiento del control convencional, difuso y difuso tabú con bases de 49 reglas. En la figura 5.6.1 se presentan los resultados para el primer disturbio (cuando es retirada la carga). En la figura 5.6.2 se presentan los resultados del segundos disturbio (se lleva el sistema de cero a plena carga). Por último en la figura 5.6.3 se presentan los resultados cuando el sistema se somete a todos los disturbios con la tabla sintonizada con la Búsqueda Tabú.

La simulación del sistema completo se lleva acabo en 12 segundos los cuales son suficientes para presentar los disturbios en la carga, y los cambios en la señal de referencia de velocidad.

5.7 ANÁLISIS DE LOS RESULTADOS OBTENIDOS AL SINTONIZAR LA BASE DE 25 REGLAS

En las gráficas del índice de comportamiento de velocidad para las dos primeras pruebas, se puede observar que los controles difusos con una base de 25 reglas tienen un mejor comportamiento con respecto al control convencional, mientras que el control que utiliza la base de reglas optimizada por la Búsqueda Tabú es superior a los dos mencionados anteriormente.

En las gráficas de velocidad se observa que se mejora la respuesta del sistema evitando el segundo sobre impulso, es decir cuando la señal se acerca de nuevo a la de referencia esta lleva inercia, por lo que produce el siguiente sobre impulso, el control con la base de reglas optimizada disminuye este sobre impulso llevándolo de forma más rápida a la señal de referencia.

En la figura 5.6.3 se observa que el comportamiento de los índices cambia para las zonas que representan los cambios en la referencia de velocidad, no conservando la mejora del sistema. Para las zonas donde se encuentran los cambios de carga el control difuso con reglas simétricas resulta con mejores resultados que el convencional, y el control que utiliza la tabla de reglas sintonizada con la Búsqueda Tabú, presenta mejoras en comparación con el control difuso con base de reglas simétricas. Por otra parte en las gráficas del flujo del rotor no se observa un cambio importante, esto era de esperarse puesto que la optimización se realizó con respecto a la velocidad, no con respecto al flujo. Otra justificación para esto es que si comparamos los sobreimpulsos del flujo y la velocidad notamos que son contrarios por lo que sus diagramas de fase serian diferentes. Con respecto a las corrientes los mejores resultados los obtuvieron los controles convencionales.

5.8 ANÁLISIS DE LOS RESULTADOS OBTENIDOS AL SINTONIZAR LA BASE DE 49 REGLAS

En las gráficas del índice de comportamiento de velocidad para todas las pruebas, se puede observar que los controles difusos con una base de 49 reglas tienen un mejor comportamiento con respecto al control convencional, mientras que el control que utiliza la base de reglas optimizada por la Búsqueda Tabú es superior a los dos mencionados anteriormente.

En las gráficas de velocidad se observa que se mejora la respuesta del sistema disminuyendo el tamaño del segundo sobreimpulso, es decir cuando la señal se acerca de nuevo a la de referencia esta lleva inercia, por lo que produce el siguiente sobre impulso, el control con la base de reglas optimizada lo disminuye llevándolo de forma más rápida a la señal de referencia. Además se observa otra mejora puesto que el control difuso con tabla de reglas simétricas disminuye el

tamaño del sobre impulso, y el control autosintonizado por medio de la Búsqueda Tabú lo disminuye aun más.

Por otra parte en las gráficas del flujo del rotor no se observa un cambio importante, esto era de esperarse puesto que la optimización se realizo con respecto a la velocidad, no con respecto al flujo. Otra justificación para esto fue mencionada en el análisis de 25 reglas. Con respecto a las corrientes los resultados mejoraron notablemente si se compara contra el sistema de 25 reglas el más notable fue el cambio que se obtiene al seguir i_{sd} puesto que los sigue la señal mejor que el control convencional.

CAPÍTULO 6

CONCLUSIONES Y RECOMENDACIONES

6.1 INTRODUCCIÓN

Este capítulo se encuentra dividido en 4 secciones, la segunda sección contiene las conclusiones realizadas, la tercera sección presenta las sugerencias para trabajos futuros y la última presenta las aportaciones de este trabajo.

6.2 CONCLUSIONES

- Al utilizar la Búsqueda Tabú para sintonizar la memoria asociativa difusa del control de velocidad, se mejoran los resultados con respecto a controles difusos con bases de reglas simétricas un 20% y también con respecto a controles convencionales un 40% según los valores obtenidos con el valor del índice del error al cuadrado.

- Las reglas que tienen influencia en el comportamiento del sistema son activadas por la trayectoria observada en los diagramas de fase lingüístico, debido a esto son las reglas que sufren alteración al utilizar la búsqueda, sin embargo existen reglas que cambian aunque no se encuentren en la trayectoria debido a que se utiliza una tabla de reglas general para optimizar el sistema.

- En este caso el afectar las bases de reglas de todos los controles, con respecto al mejoramiento de velocidad produce mejoras en el comportamiento general del sistema.

- El proceso algunas veces termina cuando el cambio de una regla no lleva a la estabilidad al sistema, no en el número especificado de iteraciones. En estos casos es posible utilizar otros lenguajes que permitan seguir con la evaluación a pesar de singularidades o utilizar la base de reglas optimizada previamente y cambiar el inicio de la búsqueda, se observa que es el mismo comportamiento que se obtendría en una aplicación práctica.

- Con bases de 25 reglas no se asegura un mejor comportamiento para disturbios diferentes a los que fue sometido el sistema cuando fue optimizado, sin embargo al optimizar la

memoria asociativa difusa de 49 reglas se puede asegurar un mejor comportamiento aun para disturbios, para los cuales el sistema no fue optimizado.

- El resultado de la configuración de las reglas en la base tabú, depende de los valores que se utilizan en las funciones de membresía y el tamaño del discurso, si esto cambia la configuración de la base de reglas cambiara.

- La Búsqueda Tabú en los casos en que los valores de las funciones de membresía y el espacio del discurso, no poseen una sintonización que supere el comportamiento del control convencional, encuentra mejoras con respecto al control convencional en disturbios determinados.

- El uso del valor caos usado en este trabajo, permite la evaluación de varios subespacios de la tabla de reglas, sin que se permitan movimientos de no mejora esto es especialmente útil en sistemas que tienden a empeorar más que a mejorar, es decir cuando se tiene un sistema ya sintonizado es más probable que se empeoren los resultados, puesto que existe un número mayor de combinaciones que de un mal comportamiento y un número menor de combinaciones que mejoren el sistema.

- Al realizar la Búsqueda Tabú se tienen mejores resultados cuando la búsqueda migra a un valor diferente de la regla, que obtuvo la mejor evaluación y fue cambiada, esto disminuye la agresividad en el cambio de reglas al llevarlas lejos de su posición original. Los mejores resultados fueron encontrados al mudar la búsqueda a reglas con gran probabilidad en la mejora del sistema, en este caso se utiliza el segundo mejor valor encontrado para continuar la búsqueda, por otro lado estos valores siempre se encuentran cerca del mejor valor.

6.3 SUGERENCIAS PARA TRABAJOS FUTUROS

- Autosintonizar por medio de la Búsqueda Tabú todos los controles individualmente y simultáneamente.

- Crear una tabla de reglas conjuntando todos los controles individualmente y utilizar la Búsqueda Tabú.

- Realizar control difuso con 2 controles para la máquina de inducción y sintonizar óptimamente los controles con Búsqueda Tabú (sustituir 4 y proponer 2).

- Aplicar Búsqueda Tabú en otros esquemas de control vectorial.

- Aplicar la Búsqueda Tabú a otros sistemas de control multivariable.

- Simular el sistema tomando en cuenta la saturación electromagnética.
- Usar la Búsqueda Tabú para la sintonización de controles PI.
- Usar la Búsqueda Tabú para sintonizar controles difusificados PI.
- Simular el sistema utilizando el modelo del accionamiento.
- Desarrollar una Búsqueda Tabú para sintonización de las funciones de membresía.
- Implementar el esquema de control en un sistema de prueba.
- Desarrollar e implementar un sistema que sintonice las funciones de membresía, y después la base de reglas difusas automáticamente.

6.4 APORTACIONES

Desarrollo del algoritmo computacional del Control Vectorial Difuso Tabú, haciendo énfasis en la lógica que mantiene la estructura de la tabla de reglas. En los trabajos investigados la base tabú es convertida a una lista, modificando de esta manera la vecindad natural de la regla. En este trabajo la base de reglas no pierde su forma.

Se desarrolla un nuevo esquema de búsqueda en las vecindades de la regla o solución actual, al no utilizar migrar la búsqueda hacia los mejores valores encontrados, sino a aquellos cercanos al mejor valor. El método hace hincapié en la lógica de la búsqueda al elegir la regla en la cual se continuará la búsqueda, y en los movimientos realizados.

Aplicación de la Búsqueda Tabú a un sistema de control multivariable como es el control vectorial difuso de la máquina de inducción.

Desarrollo del software que simula el proceso de Búsqueda Tabú con el sistema de control vectorial difuso del motor de inducción, para la autosintonización de la memoria asociativa difusa del control de velocidad en forma general.

Metodología para la obtención de los valores iniciales para las funciones de membresía y universo del discurso en los controles difusos basados en el comportamiento de los controles convencionales.

Se sintoniza óptimamente la tabla de reglas difusas general para los controles difusos utilizados en el control de velocidad de la máquina de inducción.

REFERENCIAS

[1] Glover F. and Laguna M., "*Tabu Search*", Kluwer Academic Publisher, 1997.

[2] Maurizio Denna, Giancarlo Mauri and Anna Maria Zanaboni.," *Learning Fuzzy Rules Whit Tabu Search An Application to control*", IEEE Trans. Fuzzy Systems Vol. 7 No.2 April 1999.

[3] Kevin M. Passino, Stephen Yurkovich, "*Fuzzy Control*", Addison Wesley, 1998.

[4] Bose, Bilmal K., "*Modern Power Electronics and AC drivers*", Prentice Hall PTR, 2002.

[5] J. Murphy, F. Turnbull, "*Power Electronics Control of AC Motor*", Pergamon Press, Great Britain, 1988.

[6] Peter Vas., " *Sensorless Vector and Direct Torque Control*", Oxford University Press, 1998.

[7] Paul C. Krause, Oleg Wasynczuk, Scott D. Sudhoff, "*Analysis of Electric Machinery and Drive Systems*",IEEE Press – Wiley Interscience, 2002.

[8] Bose, Bilmal K. "*Power Electronics and Variable Frequency Drives*", IEEE Press, 1997.

[9] Z. V. Lakaparampil, K.A. Fathima, V. T. Ranganathan., "*Design, Modelling, Simulation and Implementation of Vector Controlled Induction Motor Drive*",

[10] Y. Miloud, A. Draou, "*Fuzzy Logic Speed Control of an Indirect Field-Oriented Induction Machine Drive*", The 27th Annual Conference of the IEEE Industrial Electronics Society, 2001.

[11] Fred Glover, Belén Melián, "*Tabu Search*", Inteligencia Artificial, Revista Iberoamericana de inteligencia Artificial. No 19,2003.

[12] Fred Glover, James P. Kelly, Manuel Laguna, "*Practical Introduction To Simulation Optimization*", Winter Simulation Conference, 2003.

[13] Belén Melián, José A. Moreno Pérez, J. Marcos Moreno Vega, "*Metaheuristics: A gloval view*", Inteligencia Artificial, Revista Iberoamericana de inteligencia Artificial. No 19, 2003.

[14] B. Kosko, "*Neural Network and Fuzzy Systems*", Englewood Cliffs, NJ: Prentice Hall, 1992.

[15] M. Laguna "*A Guide to Implementing Tabu Search*", Investigación Operativa. Vol. 4 No.1 1994.

[16] Gi-Hyun Hwang, J.H. Park, Dong-He Lee, Dong-Wang Kim, J.I. Woo "*Implementation Of Flc For Speed Control Of A DC Servo Using Real-Type Tabu Search*", IEEE ISIE Pusan, Korea, 2001

[17] Brian Hebert, Longya Xu and Yifan Tang, "*Fuzzy Logic Enhanced Speed Control of an indirect Field-Oriented Induction Machine Drive*", IEEE Trans. Power Electronics. Vol.12 No.5 September 1997.

[18] C.M. Liaw and S. Y. Cheng, "*Fuzzy Two Degres-of-fredom Speed Controller for Motor*", Drives. IEEE Trans. Industrial Electronics. Vol. 7 Febuary 1999.

[19] Rejani K. Mudi and Nikhil R. Pal, "*A Robust selft-Tuning Scheme for PI and PD Type Fuzzy Controllers*",. IEEE Trans, Fuzzy Systems, Vol. 7, February 1999.

[20] Mao-Fu Lai, Michio Nakano and Guan-Chyun Hsieh. "*Application of Fuzzy Logic in the Phase-Locked Loop Speed Control of Induction Motor Drive*", IEEE Trans. Industrial Electronics. Vol. 43 No. 6 December 1999.

[21] Ruíz R. H. "*Control Difuso Autosintonizable De Voltaje Por Medio De La Búsqueda Por Tabú Para Un Generador Sincrono*", SEPI ESIME ZAC. Abril 2002.

[22] Glover F., Nelly P. J., Laguna M. "*Practical Introduction To Simulation Optimization*", Proceedings of the 2003 winter Simulation Conference. S,chick, P. J. Sánchez, D. Ferrin, and D.J. Morrice, eds.

[23] M. Resende y J. L. González- Valerde. "*GRASP: Procedimientos de búsqueda miopes aleatorizados y adaptativos. Inteligencia artificial*", Vol 19. Revista Iberoamericana de Inteligencia Artificial, Vol 19, 2003.

[24] J. E. Beasley Lagrangian Relaxation en C.R. Reeves (ed). "*Modern Heuristic Techniques for combinatorial Problems",* Pag. 243-303, Blackwell Scientific Publications 1993.

[25] P. Moscato y C. Cotta-Porras. "*Una introducción a los algoritmos meméticos"*, Revista Iberoamericana de inteligencia Artificial. Vol 19, 2003.

[27] Larrañaga, J. A. Lozano y H. Mühlen Bein. "*Algoritmos de estimación de distribuciones en problemas de optimizacion combinatoria",* Revista Iberoamericana de inteligencia Artificial. Vol 19, 2003.

[28] R. Martí y J. M. Moreno-Vega. "*Métodos multi-arranque*", Revista Iberoamericana de inteligencia Artificial. Vol 19, 2003.

[29] K. Dowsland y B. A. Diaz. "*Diseño de heuristicas y fundamentos de recocido simulado*", Revista Iberoamericana de inteligencia Artificial. Vol 19, 2003.

[30] Hiroyuki Mori, and Yasuyuki Sone, "*A Simplified fuzzy Inference Method whit Tabu Search for Short-term Load Forecasting in Power Systems*", Procedings of the 37th IEEE Conference on Decision & Control. Tampa, Florida USA, December 1998.

[31] Fushuan Wen, and C. S. Chang., "*A Fuzzy Abductive Inference Model for Diagnostic Problem Solving Using Tabu Search Aproach*", IEEE Symposium on Circuits and Systems, Hong Kong, June 9-12 1997.

[32] Ing. Gerardo Celso Hernandez Mendoza. " *Control Difuso De Velocidad De Un Motor De Corriente Directa*", 25 de febrero de 1997.

[33] Ing, Marco Antonio Sobrerilla González. "*Diseño De Un Estabilizador Del Sistema Electrico De Potencia Utilizando Control Difuso*", 16 de Diciembre de 1997.

[34] José Alberto Gomez Hernández. "*Optimización De La Confiabilidad En La Transmisión En Sistemas Electricos De Potencia Utilizando Algoritmos Geneticos*", Julio de 1997.

[35] M. C. José Alberto Gómez Hernández. "*Optimización De La Confiabilidad En Sistemas Eléctricos De Potencia Compuestos Utilizando Algoritmos Evolucionarios*", Noviembre 2001.

[36] Ing. Marco Vinicio Magallon Valderrama. "*Sintonizador De Un Estabilizardor De Sistemas De Potencia Por Medio De Redes Neuronales*", 30 Abril de 1998.

[37] Ing. Gabriel Mendoza Figueroa. "*Control De Velocidad De Un Motor De Inducción Aplicando El Mapa Auto- Organizable De Kohonen*". Noviembre de 1998.

[38] Jorge César Landa Hernández. "*Control Neuronal De Sistemas De Excitación De Un Genrador Sincrono*", Agosto del 2001.

[39] Ing. Julian Mendoza Fernandez. "*Compensación Por Avance De Fase Neurodifuso Para La Estabilización De Sistemas Electricos De Potencia*", 16 de octubre de 1998.

[40] Ing. Miguel A. Gama Valdez. "*Control Directo Del Par y Flujo Del Motor De Inducción Utilizazndo Una Neuro-Difuso*", 2004.

[41] Ing. Marco A. Carretero Reyes "*Desarrollo De Un Controlador Prealimentado Neuro-Difuso Tipo Anfis Para Una Unidad Turbotas*", 2002.

[42] Lakaparampil Z.V., Fátima K.A., Ranganathan "*Desing, modelling, simulation and implementation of vector control induction motor drive*", Indian Institute of Science

[43] Aström K. & Hägglund T., "*PID Controllers: Theory, Design, and Tuning*", Instrument Society of America, 2nd Edition, 1995.

[44] Nagrath I., Gopal M. *"Control Systems Engineering"* A Ahsted Press Book. 2nd Edition.

APÉNDICE A

PARÁMETROS Y MODELO DEL MOTOR DE INDUCCIÓN

A.1 PARÁMETROS DEL MOTOR DE INDUCCIÓN

Los parámetros están representados en ohms usando una frecuencia de 60 Hz para el cálculo de las reactancias [7].

hp	Volts	rpm	T_B (N.m)	$I_{B(abc)}$ (amps)	R_s (ohms)	X_{ls} (ohms)	X_m (ohms)	X'_{lr} (ohms)	R'_r (ohms)	J ($kg.m^2$)
3	220	1710	11.9	5.8	0.435	0.754	26.13	0.754	0.816	0.089

Tabla A. 1 Parámetros utilizados para la simulación del motor de inducción.

El motor tiene las siguientes características 4 polos, 60 Hz, 3 fases.

A.2 ECUACIONES DE ESTADO DEL MODELO DINÁMICO DEL MOTOR DE INDUCCIÓN

El modelo dinámico en espacio de estados es importante para análisis transitorio, particularmente en estudios de simulación por computadora. Se presentan las ecuaciones de estado de la máquina en el cuadro de referencia rotatorio con los flujos de enlace como las principales variables [4].

$$\frac{dF_{sq}}{dt} = \omega_b \left[u_{sq} - \frac{\omega_e}{\omega_b} F_{sd} - \frac{R_S}{X_{ls}} \left(F_{sq} - F_{mq} \right) \right] \qquad \text{(A. 1)}$$

$$\frac{dF_{sd}}{dt} = \omega_b \left[v_{sd} + \frac{\omega_e}{\omega_b} F_{sq} - \frac{R_S}{X_{ls}} \left(F_{sd} - F_{md} \right) \right] \qquad \text{(A. 2)}$$

$$\frac{dF_{rq}}{dt} = -\omega_b \left[\frac{\left(\omega_e - \omega_r \right)}{\omega_b} F_{rd} + \frac{R_r}{X_{lr}} \left(F_{rq} - F_{mq} \right) \right] \qquad \text{(A. 3)}$$

$$\frac{dF_{rd}}{dt} = -\omega_b \left[-\frac{\left(\omega_e - \omega_r \right)}{\omega_b} F_{rq} + \frac{R_r}{X_{lr}} \left(F_{rd} - F_{md} \right) \right] \qquad \text{(A. 4)}$$

$$T_e = \frac{3}{2}\left(\frac{P}{2}\right)\frac{1}{\omega_b}\left(F_{sd}i_{sq} - F_{sq}i_{sd}\right) \quad \text{(A. 5)}$$

$$T_e = T_L + J\frac{d\omega_m}{dt} = T_L + \frac{2}{P}J\frac{d\omega_r}{dt} \quad \text{(A. 6)}$$

Las ecuaciones A.1 – A.6 describen completamente el modelo de la máquina de inducción donde F_{qs}, F_{ds}, F_{qr} y F_{dr} son las variables de estado [4]. Este modelo tiene como entradas a u_{sq} y u_{sd} y la frecuencia ω_s y con las ecuaciones del flujo de enlace se calculan las corrientes i_{qs} e i_{ds}.

Donde:

$$F_{md} = \frac{X_{ml}}{X_{ls}}F_{sd} + \frac{X_{ml}}{X_{lr}}F_{rd} \quad \text{(A. 7)}$$

$$F_{mq} = \frac{X_{ml}}{X_{ls}}F_{sq} + \frac{X_{ml}}{X_{lr}}F_{rq} \quad \text{(A. 8)}$$

$$X_{ml} = \frac{1}{\left(\frac{1}{X_m} + \frac{1}{X_{ls}} + \frac{1}{X_{lr}}\right)} \quad \text{(A. 9)}$$

La forma de calcular las corrientes se muestra a continuación:

$$i_{sq} = \frac{F_{sq} - F_{mq}}{X_{ls}} \quad \text{(A. 10)}$$

$$i_{rq} = \frac{F_{rq} - F_{mq}}{X_{lr}} \quad \text{(A. 11)}$$

$$i_{sd} = \frac{F_{sd-}F_{md}}{X_{ls}} \quad \text{(A. 12)}$$

$$i_{rd} = \frac{F_{rd-}F_{md}}{X_{lr}} \quad \text{(A. 13)}$$

Los Flujos de enlace se definen como:

$$F_{sq} = \omega_b\Psi_{sq} = X_{ls}i_{sq} + F_{mq} \quad \text{(A. 14)}$$

$$F_{rq} = \omega_b\Psi_{rq} = X_{lr}i_{rq} + F_{mq} \quad \text{(A. 15)}$$

$$F_{sd} = \omega_b\Psi_{sd} = X_{ls}i_{sd} + F_{md} \quad \text{(A. 16)}$$

$$F_{rd} = \omega_b\Psi_{rd} = X_{lr}i_{rd} + F_{md} \quad \text{(A. 17)}$$

APÉNDICE B

TRANSFORMACIONES DE CLARKE Y PARK

B.1 TRANSFORMACIÓN DE CLARKE

Esta transformación lleva del sistema trifásico a, b, c a un sistema de coordenadas α,β de dos fases referido al estator. Esto se obtiene de la proyección de las componentes trifásicas, sobre ejes de cuadratura α,β.

Se asume que el eje de la fase "a" es colineal con el eje α, las corrientes de cuadratura de fase del estator $i_{s\alpha}$, e $i_{s\beta}$, se relacionan de la siguiente manera con las corrientes de fase a, b, c.

$$i_{s\alpha} = k\left[i_{sa} - \frac{1}{2}i_{sb} - \frac{1}{2}i_{sc}\right]$$
$$i_{s\beta} = k\frac{\sqrt{3}}{2}\left(i_{sb} - i_{sc}\right) \qquad \text{(B. 1)}$$

Para una potencia invariante en el tiempo la constante de transformación k=2/3. En este caso, las cantidades i_{sa} e $i_{s\alpha}$ son iguales. Si se asume que $i_{sa} + i_{sb} + i_{sc} = 0$, las componentes de fase de cuadratura pueden expresarse utilizando solo dos fases de las 3 del sistema.

$$i_{s\alpha} = i_{sa}$$
$$i_{s\beta} = \frac{1}{\sqrt{3}}i_{sa} + \frac{2}{\sqrt{3}}i_{sb} \qquad \text{(B. 2)}$$

La transformación inversa de Clarke regresa del sistema α, β al sistema trifásico de fases a, b, c. La constante k = 2/3, esta dada por las siguientes ecuaciones:

$$i_{sa} = i_{s\alpha}$$
$$i_{sb} = -\frac{1}{2}i_{s\alpha} + \frac{\sqrt{3}}{2}i_{s\beta} \qquad \text{(B. 3)}$$
$$i_{sc} = -\frac{1}{2}i_{s\alpha} - \frac{\sqrt{3}}{2}i_{s\beta}$$

B.2 TRANSFORMACIÓN DE PARK

Transforma las componentes del cuadro de referencia del estator al cuadro de referencia d-q que gira a la misma velocidad de la frecuencia angular del las corrientes de fase. Se considera el eje d alineado con el flujo del rotor, las componentes i_{sq} e i_{sd} del vector espacial de corriente en el cuadro de referencia d-q se determinan como sigue:

$$\begin{aligned} i_{sd} &= i_{s\alpha}\cos\theta_{campo} + i_{s\beta}\sin\theta_{campo} \\ i_{sq} &= -i_{s\alpha}\sin\theta_{campo} + i_{s\beta}\cos\theta_{campo} \end{aligned} \qquad \text{(B. 4)}$$

Donde la componente i_{sd} es llamada componente del eje directo (componente producida por el flujo) e i_{sq} es llamada componente de cuadratura (componente producida por el torque). Son invariantes en el tiempo además de que el control del flujo y torque con estas resulta sencillo.

Para lograr utilizar funciones trigonométricas de forma sencilla, se calcula directamente su valor usando la división, que se define con las siguientes ecuaciones:

$$\psi_{rd} = \sqrt{{\psi_{r\alpha}}^2 + {\psi_{r\beta}}^2} \qquad \text{(B. 5)}$$

$$\begin{aligned} \sin\theta_{campo} &= \frac{\psi_{r\beta}}{\psi_{rd}} \\ \cos\theta_{campo} &= \frac{\psi_{r\alpha}}{\psi_{rd}} \end{aligned} \qquad \text{(B. 6)}$$

La transformación inversa de Park lleva el sistema de coordenadas d-q a α, β y esta dado por las siguientes ecuaciones:

$$\begin{aligned} i_{s\alpha} &= i_{sd}\cos\theta_{campo} - i_{sq}sen\theta_{campo} \\ i_{s\beta} &= i_{sd}sen\theta_{campo} + i_{sq}\cos\theta_{campo} \end{aligned} \qquad \text{(B. 7)}$$

APÉNDICE C

PROGRAMA DE SIMULACIÓN DIGITAL

Programa que realiza la autosintonización de la memoria asociativa difusa general del control vectorial de velocidad difuso tipo Mamdani, para un motor de inducción.

El programa fue desarrollado en el lenguaje computacional Borland C++ ® Versión 5.0A y Borland C++ Builder ® Versión 6.0.

C.1. Funciones relacionadas con el control vectorial

ControllerPI: Contiene el algoritmo de control convencional PI.

CptrfmClarke: Obtiene la transformación directa de Clarke.

CptrfmClarkeInv: Obtiene la transformación inversa de Clarke.

CptrfmPark: Obtiene la transformación directa de Park.

CptrfmParkInv: Obtiene la transformación inversa de Park.

Decoupling: Obtiene las componentes de desacople utilizadas por el control vectorial.

Dqestabl: Establece el sistema que gira con el flujo del rotor.

Fluxmodel: Estima el flujo del rotor.

Funx1 – Funx5: Utilizadas para simular el modelo de la máquina de inducción.

Motor: Función que engloba el control convencional y el difuso en un sola función.

Value_Error: Obtiene el valor de la función objetivo.

SvmStd: Esta función calcula el ciclo de trabajo apropiado necesario para generar un voltaje de estator de referencia dado usando la modulación de vectores espaciales con

un ciclo de trabajo nulo del estado de swicheo de los estados O000 y 0111 en cada sector del hexágono.

InvStd: Recibe los tiempos de conmutación y provee los voltajes de las fases del modelo del motor.

C.2. Funciones relacionadas al control difuso.

Fuzzy_control: Reúne todas las funciones del control difuso.

Fuzzy_free: Libera la memoria asignada para todas las posibles reglas que se inicializan con el uso del control difuso.

Fuzzy_init1 – **fuzzy_init3**: Inicializa los valores del espacio del discurso y los centros de las funciones de membresía.

Fuzzyify: Esta función difusifica la entrada u para determinar el grado de membresía para cada función.

Int_defuzz: Modulo de inferencia y dedifusicación.

Leftall: Determina el grado de membresía para la función de membresía rodilla izquierda.

Mach: Determina cuales reglas están activadas.

Rightall: Determina el grado de membresía para la función de membresía rodilla derecha.

Triangle: Determina el grado de membresía para la función de membresía triángulo.

C.3 Funciones relacionadas con el algoritmo de la Búsqueda Tabú

Best_move: Elige la siguiente vecindad a evaluar y el mejor movimiento encontrado, utilizando los valores generados por el bloque de evaluación contenidos en la lista de candidatos, tomando en cuenta el atributo tabú y las penalizaciones de frecuencia además del valor de caos. La mejor transformación no es elegida para continuar la búsqueda, pues esta se convierte tabú para atributos de memoria de corto plazo.

Chose_move: Usa los valores encontrados por el bloque mejor movimiento para obtener los nuevos valores para continuar la búsqueda, es decir decodifica la

información de la lista de candidatos del mejor valor encontrado para utilizarla en la base de reglas.

Cleareval: Prepara la lista de candidatos para la siguiente búsqueda.

Dectabu: Este bloque decrementa en uno todos los valores utilizados por la búsqueda en el pasado reciente de la estructura tabú, en cada iteración los valores se decrementan en uno hasta llegar al valor de cero, para que se les permita participar de nuevo en la búsqueda.

Eval_neighborhood: Identifica el elemento y el tipo de vecindad a evaluar, evalúa la vecindad del elemento utilizando la función objetivo y transformando la base de reglas del sistema de control vectorial difuso, la forma de evaluación se realiza con cambios o transformaciones, cambiando el valor de cada vecindad por el valor de la elemento o solución actual en otras palabras transformando el valor de la vecindad por el valor del elemento actual. Al terminar de evaluar cada vecindad regresa el valor que poseía la tabla antes de realizar la transformación, por lo que este bloque no afecta las vecindades de la tabla de reglas de forma permanente. La evaluación de las vecindades se guarda en una lista de candidatos, esta lista contiene los valores de los resultados de la evaluación de la función objetivo de cada vecindad y los atributos y penalizaciones de las vecindades.

Update_neighborhood: Identifica el elemento y el tipo de vecindad a evaluar, evalúa la vecindad del elemento utilizando la función objetivo y transformando la base de reglas del sistema de control vectorial difuso, la forma de evaluación se realiza con cambios o transformaciones, cambiando el valor de cada vecindad por el valor de la elemento o solución actual en otras palabras transformando el valor de la vecindad por el valor del elemento actual. Al terminar de evaluar cada vecindad regresa el valor que poseía la tabla antes de realizar la transformación, por lo que este bloque no afecta las vecindades de la tabla de reglas de forma permanente. La evaluación de las vecindades se guarda en una lista de candidatos, esta lista contiene los valores de los resultados de la evaluación de la función objetivo de cada vecindad y los atributos y penalizaciones de las vecindades.

VYJ y VYJH: Imprimen los datos en pantalla.

C.4 El programa de simulación digital

Este programa permite elegir la columna y el renglón de la memoria asociativa difusa, en el cual se desea inicializar la búsqueda, además de poder elegir el número de iteraciones en las cuales se desea que el proceso termine.

Al correr el programa se despliegan los resultados generados en cada iteración por el programa para cada evaluación de la vecindad.

El programa es capaz de simular en control vectorial convencional y el control difuso tabú, para esto es necesario que en la función "motor" se elijan cuales serán los controles a utilizarse.

El programa es capaz de generar un archivo de datos en cada iteración, que tendrá los datos del comportamiento de la velocidad, en cada iteración será borrado y vuelto a escribir con el fin de observar el comportamiento a cada iteración. No solo es posible guardar los datos del comportamiento de la velocidad con respecto al tiempo, también es posible guardar los datos del flujo, las corrientes, los valores del error, el cambio del error, entre otros.

A continuación se presenta el programa utilizado para realizar la simulación.

```
/*****************************************
*   Nombre del archivo:
*ControlDifusoVectorialAutosintonizableTabu_MI_3
F.
*
*   Descripción:
*Simulación del sistema de control vectorial difuso
*autozintonizable de velocidad por Búsqueda Tabú
*para la máquina de inducción.
*
*****************************************
#include <string.h>
#include <iostream.h>
#include <fstream.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#define I_KT_C  double(4.12787358483)
#define KL_R_KT_C double(.657711084236)
#define KL_T_KT_C double(.00259395763597)
#define TR_KT_C          double(.360735402864)
#define T_SAMPLE         double(0.0000625)
#define PSI_ZERO double(0.0000001)/* valor
minimo del flujo magnetico */
#define LM_TR_C double(.793114064684) /*
constante dependiente del motor  */
#define MAX(A,B)  ((A) > (B) ? (A) : (B))
#define MIN(A,B)  ((A) < (B) ? (A) : (B))
#define SIZE 5   /*tamaño de la tabla de raglas n*n*/
/*****************************************
* Estructuras para los algoritmos del desacoplamiento
vectorial
*****************************************/
typedef struct
{
        double PhaseA; /*corriente de la fase A*/
  double PhaseB; /*corriente de la fase B*/
  double PhaseC; /*corriente de la fase C*/
} mc_s3PhaseSystem;

typedef struct
{
        double alpha; /*valor en la fase alfa
(corriente,voltaje,flujo)*/
  double beta;  /*valor en la fase beta
(corriente,voltaje,flujo)*/
} mc_sPhase;

typedef struct {
   double alpha;
   double beta;
} mc_sPhase32;

typedef struct
{
        double d_axis; /*valor en el eje d*/
  double q_axis; /*valor en el eje q*/
} mc_sDQsystem;
```

```
typedef struct
{
        double psi_Rd;
        double omega_field;
        double i_Sd;
        double i_Sq;
} mc_sDQEstabl;

typedef struct
{
        double sine;        /*valor del seno*/
  double cosine;  /*valor del coseno*/
} mc_sAngle;

typedef struct {
  mc_sPhase32 XAlphaBeta; /* modelo de flujo
variable de estado 1 */
  mc_sPhase dXAlphaBeta; /* modelo de estado
variable de estado 2 */
        double I_KT; /* constante dependiente del
motor 1 */
        double KL_R_KT; /* constante dependiente
del motor 2 */
        double KL_T_KT; /* constante dependiente
del motor 3 */
        double TR_KT; /* constante dependiente del
motor 4 */
        double T; /* periodo de muestreo */
} fluxmodel_sState;

typedef struct {
        double psi_Rd_zero; /* minimo valor del
flujo magnetico del rotor */
        double LM_TR; /* constante dependiente del
motor */
} dqestabl_sState;

typedef struct
{
  double Gain_P;
  double Gain_I;
  double LimitPos;
  double LimitNeg;
  double PortionI_K_1;
  double Desired;
}mc_sPIparams;

typedef struct
{
   double ULimit;      /*salida de voltaje maximo del
estator limite*/
   double LM_LR_TR;  /* constante dependiente del
motor 1*/
   double LM_LR;     /* constante dependiente del
motor 2 */
   double KL;        /* constante dependiente del motor
3 */
} decoupling_sState;

/****************************************
* Estructuras para el control difuso
****************************************/
typedef struct in_mem
{
 double width;/* Entrada funcion de membresia (1/2
de la base del triangulo).*/
 double *center;/* Centro de cada funcion de
membresia de entrada.*/
 double *dom;/*Grado de membresia para cada una
de las funciones de membresia*/
} IN_MEM;

typedef struct out_mem
{
 double width;/* Ancho de la funcion de membresia
de salidad (1/2 de la base

            del triangulo). */
 double *center;       /* Centro de cada funcion de
membresia. */
} OUT_MEM;

typedef struct fuz_sys
{
 IN_MEM  *emem;/* Agrupar todos los parametros
del sistema difuso en una unica

             variable. */
 IN_MEM  *edotmem;
 OUT_MEM *outmem;
} FUZ_SYS;

typedef struct
{
            double e1;
  double e2;
  double e3;
  double e4;
  double edot;
```

```
  double aux;
  double uact;
  double udot;
  double u;
}INOUT_PI;

/****************************************
* Estructuras para la Búsqueda Tabú
****************************************/
typedef struct
{
 double obj_fun1;
 double value;
 int    limleft;
 int    limright;
} val_fun;

typedef struct
{
          int i;
  int j;
  int a;
  int b;
  double aux;
} i_matrix;

typedef struct
{
          double best;
  //double curr;
  int chaos;
  int    shift;
}prueba;


double funx1(double,double,double,double,double);
/*Fsq,Fsd,Frq,Vsq,Ws*/
double funx2(double,double,double,double,double);
/*Fsq,Fsd,Frq,Vsd,Ws*/
double funx3(double,double,double,double,double);
/*Fsq,Frq,Frd,W,Ws*/
double funx4(double,double,double,double,double);
/*Fsd,Frq,Frd,W,Ws*/
double funx5(double,double);                /*Te,Tl*/

void cptrfmClarke ( mc_sPhase *pAlphaBeta,
mc_s3PhaseSystem *p_abc);
void cptrfmClarkeInv ( mc_s3PhaseSystem *p_abc,
mc_sPhase *pAlphaBeta);
void cptrfmPark (mc_sDQsystem *pDQ, mc_sPhase
*pAlphaBeta, mc_sAngle *pSinCos);
void cptrfmParkInv(mc_sPhase
*pAlphaBeta,mc_sDQsystem *pDQ, mc_sAngle
*pSinCos);
void sincos(mc_sAngle *pSinCos,double t);
void fluxmodel (mc_sPhase *pPsiRAlphaBeta,
fluxmodel_sState *pState,

          mc_sPhase *pUsAlphaBeta, mc_sPhase
*pIsAlphaBeta,double omega);
void dqestabl( mc_sAngle *pThetaField,
mc_sDQEstabl *pDQdata,

          mc_sPhase *pIsAlphaBeta, mc_sPhase
*pPsiRAlphaBeta, double omega,

          dqestabl_sState *pDQstate );
double controllerPI(double DesiredValue, double
MeasuredValue,

mc_sPIparams *pParams);

void decoupling( decoupling_sState *sState,
mc_sDQEstabl *pDQdata,

          mc_sDQsystem *pU_S_ref, double omega,
mc_sDQsystem *pU_S );
void svmStd (mc_sPhase *pAlphaBeta,
mc_s3PhaseSystem *p_svmabc);
void svmPwmlct (mc_sPhase *pAlphaBeta,
mc_s3PhaseSystem *p_svmabc);
void invStd(mc_s3PhaseSystem *p_svmabc,
mc_s3PhaseSystem *p_PwmABC,
                                        double
countupdown);
/****************************************
funciones prototipo para el control
****************************************/
void fuzzy_init1(FUZ_SYS *fuzzy_system);
void fuzzy_init2(FUZ_SYS *fuzzy_system);
void fuzzy_init3(FUZ_SYS *fuzzy_system);
void fuzzy_free(FUZ_SYS *fuzzy_system);
double fuzzy_control(double e, double edot,
FUZ_SYS *fuzzy_system,

          int rules[SIZE][SIZE]);
void fuzzyify(double u, IN_MEM *mem);
double leftall(double u, double w, double c);
double rightall(double u, double w, double c);
```

```
double triangle(double u, double w, double c);
void match(const IN_MEM *emem, const IN_MEM
*edotmem, int *pos);
double inf_defuzz(IN_MEM *emem, IN_MEM
*edotmem, OUT_MEM *outmem, int *pos,

int rules[SIZE][SIZE]);
double motor (int rules[SIZE][SIZE]);
/****************************************
funciones prototipo para la busqueda tabu
****************************************/
void value_Error_Fun(double e, double edot, val_fun
*value, double t);
void eval_neighborhood (int r, int c,double
values[9][3], int rules[SIZE][SIZE],

                    int frec[SIZE][SIZE],int
tabu[SIZE][SIZE]);
int best_move (prueba *value, double values[9][3]);
void update_neighborhood (int r, int c, int initer, int
rules[SIZE][SIZE],

                              int frec[SIZE][SIZE], int
tabu[SIZE][SIZE]);
void chose_move (i_matrix *index, int moiter);
void VYJ (int matrix [SIZE][SIZE]);
void VYJH(double matrix [9][3]);
void CLEAREVAL (double matrix [9][3]);
void dectabu (int matrix [SIZE][SIZE]);

void main(void)
{
  int i=0,iter,caos,best_initer;
  int rules[SIZE][SIZE]={{0,0,0,1,2},
                         {0,0,1,2,3},
                         {0,1,2,3,4},
                         {1,2,3,4,4},
                  {2,3,4,4,4}};
  int tabu [SIZE][SIZE]={{0},{0}};
  int frec [SIZE][SIZE]={{0},{0}};

printf("\n ¿En que columna desea iniciar la
Búsqueda?\n"); scanf("%lf",&a);
  printf("\n ¿En que renglón desea iniciar la
Búsqueda?\n"); scanf("%lf",&b);
  printf("\n ¿Cuantas iteraciones desea que el
programa ejecute?\n"); scanf("%lf",&c);
  iter =0;
  i_matrix index;
  prueba value;
  index.a=a;
  index.b=b;
    value.best=6111.062817824257764;
  value.chaos=0;

   for(iter=0;iter<=c;iter++){
 eval_neighborhood
(index.a,index.b,values,rules,frec,tabu);
   best_initer=best_move(&value, values);
   printf("\n Valor elegido para la busqueda
%d\n",value.shift);
   if (best_initer!=10){
   value.best=values[best_initer][0];
   }
   update_neighborhood
(index.a,index.b,best_initer,rules,frec,tabu);
   tabu[index.a][index.b]=3;
   frec[index.a][index.b]+=1;
   chose_move(&index, value.shift);
   printf(" \n Mejor valor encontrado %d %f
\n",best_initer, value.best);

  dectabu (tabu);
     printf("\n VALORES              TABU
          FRECUENCIA\n");
     VYJH(values);
     printf("\n ESTRUCTURA TABU \n");
     VYJ(tabu);
     printf("\n FRECUENCIA \n");
     VYJ(frec);
     printf("\n BASE DE REGLAS \n");
     VYJ(rules);
     printf("\n Evaluando la vecindad de r= %d  c=
%d\n", index.a,index.b);
     CLEAREVAL(values);

    /*  for (i=0;i<=8;i++)
    {
   //printf("Valor %d %f\n",i,values[i][0]);
    values[i][0]=0;
    } */
    getch();
  }
  }
/****************************************
* Modulo eval_neighborhood().
*
* Descripción:
* Evalua la vecindad de la regla elegida.
*
```

```
*
*   Devuelve:    Los valores de la tabla usada para
*elegir el mejor   movimiento.
*
*   Argumentos:
*
*   entrada: -   index_a   - Renglon de la regla a
evaluar.
*
* -   index_b        - Columna de la regla a evaluar.
*
*                      -   rules            -Base de reglas.
*
*                      -   tabu             -Estructura tabu.
*
*                      -   frec             -Tabla valores de
*frecuencia
*
*   salida: -          values    - tabla donde se acumulan
*los valores de frecuencia
* tabu y la evaluacion  la funcion objetivo de la
*           vecindada de la regla.
*
****************************************/
void eval_neighborhood(int r, int c, double
values[9][3], int rules[SIZE][SIZE],

                       int frec[SIZE][SIZE],int
tabu[SIZE][SIZE])
   {
   int i,j,k=0;
   double aux;

      if(r<4 && c==0){
      if(r!=0){
     for (i=(r-1);i<=(r+1);i++){
                       for (j=(c);j<=(c+1);j++){
      if (rules[i][j]<rules[r][c]+3){
          aux=rules[i][j];
          rules[i][j]=rules[r][c];
       values[k][1]=tabu[i][j];
       values[k][2]=frec[i][j];
                       values[k][0]= motor(rules);
          rules[i][j]=aux;
          printf("Value %d %f \n",k,values[k][0]);
       }
          k++;
       }
       }
      }
}
if(r==0 && c<4){
if(c!=0){
for (i=r;i<=(r+1);i++){
                       for (j=(c-1);j<=(c+1);j++){
 if (rules[i][j]<rules[r][c]+3){
      aux=rules[i][j];
      rules[i][j]=rules[r][c];
 values[k][1]=tabu[i][j];
 values[k][2]=frec[i][j];
                                values[k][0]= motor(rules);
      rules[i][j]=aux;
      printf("Value %d %f \n",k,values[k][0]);
 }
      k++;
 }
 }
 }
}
if(r==0 && c==0){
 for (i=r;i<=(r+1);i++){
                       for (j=c;j<=(c+1);j++){
 if (rules[i][j]<rules[r][c]+3){
      aux=rules[i][j];
      rules[i][j]=rules[r][c];
                       values[k][1]=tabu[i][j];
 values[k][2]=frec[i][j];
                       values[k][0]= motor(rules);
      rules[i][j]=aux;
      printf("Value %d %f \n",k,values[k][0]);
 }
      k++;
 }
 }
}
if(r==0 && c==4){
for (i=r;i<=(r+1);i++){
                       for (j=(c-1);j<=(c);j++){
 if (rules[i][j]<rules[r][c]+3){
      aux=rules[i][j];
      rules[i][j]=rules[r][c];
                       values[k][1]=tabu[i][j];
 values[k][2]=frec[i][j];
                       values[k][0]= motor(rules);
      rules[i][j]=aux;
      printf("Value %d %f \n",k,values[k][0]);
 }
      k++;
 }
 }
```

```
    }
    if(r==4 && c==0){
    for (i=(r-1);i<=r;i++){
                    for (j=c;j<=(c+1);j++){
     if (rules[i][j]<rules[r][c]+3){
         aux=rules[i][j];
         rules[i][j]=rules[r][c];
                    values[k][1]=tabu[i][j];
     values[k][2]=frec[i][j];
                    values[k][0]= motor(rules);
         rules[i][j]=aux;
         printf("Value %d %f \n",k,values[k][0]);
     }
         k++;
     }
    }
    }
    if(r==4 && c==4){
    for (i=(r-1);i<=r;i++){
                    for (j=(c-1);j<=c;j++){
     if (rules[i][j]<rules[r][c]+3){
         aux=rules[i][j];
         rules[i][j]=rules[r][c];
                    values[k][1]=tabu[i][j];
     values[k][2]=frec[i][j];
                    values[k][0]= motor(rules);
         rules[i][j]=aux;
         printf("Value %d %f \n",k,values[k][0]);
     }
         k++;
     }
    }
    }
   if(r==4 && c<4){
   if(c!=0){
    for (i=(r-1);i<=r;i++){
                    for (j=(c-1);j<=(c+1);j++){
     if (rules[i][j]<rules[r][c]+3){
         aux=rules[i][j];
         rules[i][j]=rules[r][c];
                    values[k][1]=tabu[i][j];
     values[k][2]=frec[i][j];
                    values[k][0]= motor(rules);
         rules[i][j]=aux;
         printf("Value %d %f \n",k,values[k][0]);
     }
         k++;
     }
     }
    }
    }
    if(r<4 && c==4){
    if(r!=0){
    for (i=(r-1);i<=(r+1);i++){
                    for (j=(c-1);j<=c;j++){
     if (rules[i][j]<rules[r][c]+3){
         aux=rules[i][j];
         rules[i][j]=rules[r][c];
                    values[k][1]=tabu[i][j];
     values[k][2]=frec[i][j];
                    values[k][0]= motor(rules);
         rules[i][j]=aux;
         printf("Value %d %f \n",k,values[k][0]);
     }
         k++;
     }
     }
    }
    }
   if(r>0 && c>0){
    if (r<4 && c<4){
    for (i=(r-1);i<=(r+1);i++){
                    for (j=(c-1);j<=(c+1);j++){
     if (rules[i][j]<rules[r][c]+3){
         aux=rules[i][j];
         rules[i][j]=rules[r][c];
                    values[k][1]=tabu[i][j];
     values[k][2]=frec[i][j];
                    values[k][0]= motor(rules);
         rules[i][j]=aux;
         printf("Value %d %f \n",k,values[k][0]);
     }
         k++;
         //getch();
      }
         }
    }
   }
  }

/*****************************************
* Modulo best_move().
*
* Descripción:
* Elige el mejor movimieto y la sigueinte vecindad a
evaluar.
*
*   Devuelve:  El indice del mejor movimiento
*encontrado, con el i indice de la regla a evaluar.
*
```

```
*   Argumentos:
*
*   entrada: -values   -Tabla que contiene los valores
*tabú,  frecuencia, y las evaluaciones de la vecindad.
*
*   salida:  -value     -Apuntador a los valores de caos,
*y la regla a  utilizar para evaluar su vecindad.
*
****************************************/
 int best_move(prueba *value, double values[9][3])
 {
   int i,b;
   double aux;

   if (value->chaos <2){
   b=10;
   aux=7000;
   value->shift=0;
   for (i=0;i<=8;i++)
   {
    if(values[i][0]<(value->best) && values[i][0] !=
0){
    if(aux>values[i][0]){
    aux=values[i][0];
    b=i;        // error pues se queda con el último
encontrado
   }
    //printf("Value %d %f\n",b,val);
    }
   }
   for (i=1;i<=8;i++){
   if (values[i][0]>value->best &&
values[i][0]<values[value->shift][0]){
   value->shift=i;
    //printf("Shift %d\n",value->shift);
    }
   }
   if (value->shift == 0){
   for (i=1;i<=8;i++){
   if (values[i][0]<value->best &&
values[i][0]>values[value->shift][0]){
   value->shift=i;
    //printf("Shift %d\n",value->shift);
     }
    }
   }
   if(b==10){
    value->chaos+=1;
    }
   }
   else{

   /******
   ******/
   b=10;
   aux=values[0][0];
   value->shift=10;
   for (i=0;i<=8;i++)
   {
    if (values [i][1] < 1 || values [i][0]<(value->best)){
     if (values[i][0] != 0 && values[i][0] <=aux){
     if(values [i][0]!= value->best){
       aux=values[i][0];
       b=i;
     }            // esta parte revisa si se cumple el
criterio de aspiracion
    }                                    //printf("Value
%d %f\n",b,val);
    }
   }
   aux=values[0][0];
   for (i=0;i<=8;i++)
   {
    if (values [i][1] < 1 || values [i][0]<(value->best)){
     if (values[i][0] != 0 && values[i][0] <=aux){
     if(values [i][0]!= value->best && i != b){
       aux=values[i][0];
       value->shift=i;
     }
    }
    }
   }
   if(value->shift == 10){//Si no existe valor para
mudar la busqueda
   aux=7000;             //se utiliza frecuencia
   for(i=0;i<=8;i++){
   if((values[i][2]+values[i][1]) < aux && values
[i][0]!= 0){
   if(i != b && values[i][0]!=value->best){
   aux=values[i][2];
   value->shift=i;
    }
   }
   }
   }
   }
   return(b);
 }
/****************************************
* Modulo update_neighborhood.
```

```
*
* Descripción:
* Da de alta los movimientos elegidos, la estructura
*tabú y la  frecuencia.
*
*   Devuelve:
*
*   Argumentos:
*
*           entrada: - c    -Columna de la regla elegida.
*
*                    - r    -Renglon de la regla elegida.
*
*   salida: -  rules    -Base de reglas.
*
*           -  frec     -Valores de ffrecuencia
*
*           -  tabu     -Estructura tabú
*
*           -   initer  -Mejor valor encontrado.
*
****************************************/
void update_neighborhood(int r, int c,int initer, int
rules[SIZE][SIZE],

                         int frec[SIZE][SIZE], int tabu
[SIZE][SIZE])
{
                    int k=0;
    int i,j;

      if(r<4 && c==0){
      if(r!=0){
     for (i=0;i<=2;i++){
                    for (j=0;j<=1;j++){
         if(initer==k){
                    rules[i+(r-1)][j]=rules[r][c];
       tabu[i+(r-1)][j]=3;
       frec[i+(r-1)][j]+=1;
       }
      k++;
      }
      }
     }
     }
     if(r==0 && c<4){
     if(c!=0){
     for (i=0;i<=1;i++){
                    for (j=0;j<=2;j++){
         if(initer==k){
                    rules[i][j+(c-1)]=rules[r][c];
       tabu[i][j+(c-1)]=3;
       frec[i][j+(c-1)]+=1;
       }
        k++;
      }
      }
     }
     }
     if(r==0 && c==0){
     for (i=0;i<=1;i++){
                    for (j=0;j<=1;j++){
       if(initer==k){
                    rules[i][j]=rules[r][c];
       tabu[i][j]=3;
       frec[i][j]+=1;
       }
        k++;
      }
      }
     }
     if(r==0 && c==4){
     for (i=0;i<=1;i++){
                    for (j=0;j<=1;j++){
       if(initer==k){
                    rules[i][j+(c-1)]=rules[r][c];
       tabu[i][j+(c-1)]=3;
       frec[i][j+(c-1)]+=1;

       }
        k++;
      }
      }
     }
     if(r==4 && c==0){
     for (i=0;i<=1;i++){
                    for (j=0;j<=1;j++){
       if(initer==k){
                    rules[i+(r-1)][j]=rules[r][c];
       tabu[i+(r-1)][j]=3;
       frec[i+(r-1)][j]+=1;
       }
        k++;
      }
      }
     }
     if(r==4 && c==4){
     for (i=0;i<=1;i++){
                    for (j=0;j<=1;j++){
       if(initer==k){
```

```
                rules[i+(r-1)][j+(c-1)]=rules[r][c];
      tabu[i+(r-1)][j+(c-1)]=3;
      frec[i+(r-1)][j+(c-1)]+=1;
      }
       k++;
    }
   }
  }
if(r==4 && c<4){
if(c!=0){
  for (i=0;i<=1;i++){
                for (j=0;j<=2;j++){
       if(initer==k){
                rules[i+(r-1)][j+(c-1)]=rules[r][c];
      tabu[i+(r-1)][j+(c-1)]=3;
      frec[i+(r-1)][j+(c-1)]+=1;
      }
       k++;
    }
   }
  }
 }
 if(r<4 && c==4){
 if(r!=0){
 for (i=0;i<=2;i++){
                for (j=0;j<=1;j++){
       if(initer==k){
                rules[i+(r-1)][j+(c-1)]=rules[r][c];
      tabu[i+(r-1)][j+(c-1)]=3;
      frec[i+(r-1)][j+(c-1)]+=1;
      }
       k++;
    }
   }
  }
 }
 if(r>0 && r>0){
 if (r<4 && r<4){
 for (i=0;i<=2;i++){
                for (j=0;j<=2;j++){
  if(initer==k){
                rules[i+(r-1)][j+(c-1)]=rules[r][c];
      tabu[i+(r-1)][j+(c-1)]=3;
      frec[i+(r-1)][j+(c-1)]+=1;
      }
   k++;
        }
  }
 }
 }
  }
/*****************************************
* Modulo chose_move().
*
* Descripción:
*       Con los valores encontrados obtiene los nuevos
*valores para continuar la busqueda, decodifica la
*informacion de la lista para utlizarla en la base de
*reglas.
*
*   Devuelve: Los nuevo indeices para continuar la
*evaluacion.
*
*   Argumentos:
*
*   entrada: - moiter  - Regla para continuar la
busqueda.
*
*
*    salida: - index  -apuntador a la estructura que
contiene los
*                   indices de la busqueda,
*
*****************************************/
void chose_move (i_matrix *index, int moiter)
{
                int r,c,k=0;
    int i,j;

    r=index->a;
    c=index->b;

      if(r<4 && c==0){
      if(r!=0){
     for (i=0;i<=2;i++){
                for (j=0;j<=1;j++){
         if(moiter==k){
                index->a=i+(r-1);
       index->b=j;
       }
      k++;
      }
      }
     }
    }
    if(r==0 && c<4){
    if(c!=0){
    for (i=0;i<=1;i++){
                for (j=0;j<=2;j++){
         if(moiter==k){
```

```
                index->a=i;
        index->b=j+(c-1);
        }
          k++;
     }
     }
    }
   }
  if(r==0 && c==0){
   for (i=0;i<=1;i++){
                for (j=0;j<=1;j++){
         if(moiter==k){
                index->a=i;
        index->b=j;
        }
          k++;
     }
    }
   }
   if(r==0 && c==4){
   for (i=0;i<=1;i++){
                for (j=0;j<=1;j++){
         if(moiter==k){
                index->a=i;
        index->b=j+(c-1);
        }
          k++;
     }
    }
   }
   if(r==4 && c==0){
   for (i=0;i<=1;i++){
                for (j=0;j<=1;j++){
         if(moiter==k){
                index->a=i+(r-1);
        index->b=j;
        }
          k++;
     }
    }
   }
   if(r==4 && c==4){
   for (i=0;i<=1;i++){
                for (j=0;j<=1;j++){
         if(moiter==k){
                index->a=i+(r-1);
        index->b=j+(c-1);
        }
          k++;
     }
    }
   }
  if(r==4 && c<4){
  if(c!=0){
    for (i=0;i<=1;i++){
                for (j=0;j<=2;j++){
         if(moiter==k){
                index->a=i+(r-1);
        index->b=j+(c-1);
        }
          k++;
     }
     }
    }
   }
   if(r<4 && c==4){
   if(r!=0){
   for (i=0;i<=2;i++){
                for (j=0;j<=1;j++){
         if(moiter==k){
                index->a=i+(r-1);
        index->b=j+(c-1);
        }
          k++;
     }
     }
    }
   }
   if(r>0 && r>0){
   if (r<4 && r<4){
   for (i=0;i<=2;i++){
                for (j=0;j<=2;j++){
    if(moiter==k){
                index->a=i+(r-1);
        index->b=j+(c-1);
        }
     k++;
          }
    }
   }
   }
  }
/*****************************************
************
* Modulo dectabu().
*
* Descripción:
*Decrementa en uno el estado tabu en la estructura
*sin el valor tabu
* de la regla es mayor que cero
```

```
*
*   Argumentos:
*
*   entrada:   -tabu                                -
*Estructura tabú.
*
****************************************/
void dectabu (int matrix [SIZE][SIZE])
{
int i,j;
            for (i=0;i<=4;i++){
                        for (j=0;j<=4;j++){
        if(matrix[i][j]>0)
        matrix[i][j]=matrix[i][j]-1;
      }
    }
}
/****************************************
* Modulo VYJH().
*
* Descripción:
*Imprime en pantalla los valores agrupados de tabu,
*frecuencia y  las evaluaciones
*
*   Argumentos:
*
*   entrada:  - values       - matrix tablas de valores
*tabu, frecuencia
*           y evaluaciones de la vecindad.
*
****************************************/
void VYJH (double matrix [9][3])
{
int i,j;
            for (i=0;i<=8;i++){
                        for (j=0;j<=2;j++){
        printf("%d %f                        ",i ,matrix[i][j]);
        if(j==2)
        printf("\n");
      }
    }
}
/****************************************
* Modulo CLEAREVAL().
*
* Descripción:
*            Limpia los valores agrupados de tabu,
frecuencia y las
*evaluaciones para utilizar la estructura en la
siguiente evañuación
*
*   Argumentos:
*
*   entrada:  - values       - matrix tabls de valores
tabu, frecuencia y
*

            evaluaciones de la vecindad.
*
****************************************/
void CLEAREVAL (double matrix [9][3])
{
int i,j;
            for (i=0;i<=8;i++){
                        for (j=0;j<=2;j++){
        matrix[i][j]=0;
      }
    }
}

/****************************************
* Modulo VYJ().
*
* Descripción:
* Imprime en pantalla la matriz selecciona ya sea
*base de reglas, estructructura tabu o frecuencia.
*
*
*   Argumentos:
*
*   entrada:       -matrix   -  Matriz de 4x4
*
****************************************/
void VYJ (int matrix [SIZE][SIZE])
{
int i,j;
            for (i=0;i<=4;i++){
                        for (j=0;j<=4;j++){
        printf("%d ",matrix[i][j]);
        if(j==4)
        printf("\n");
      }
    }
}

/****************************************
* Modulo motor().
*
* Descripción:
```

```
* Sistema utilizado para obtener la evalucion de la
*funcion objetivo
*   contiene el modelo del motor, controles difusos, PI
*clasicos y las  transformaciones necesarias para
*utlizar el control vectorial.
*
*   Devuelve:  -Evaluacion del sistema con la regla
*modificada.
*
*   Argumentos:
*
*   entrada:   -rules -Base de reglas utilizada en los
controles
*                       difusos.
/
double motor(int rules[SIZE][SIZE])
  {
  int i,band;
  double t,h,tf,m=0.5,hh,n,h2,A,B,C,countupdown;
  double k1_x1,k1_x2,k1_x3,k1_x4,k1_x5;
  double k2_x1,k2_x2,k2_x3,k2_x4,k2_x5;
  double k3_x1,k3_x2,k3_x3,k3_x4,k3_x5;
  double k4_x1,k4_x2,k4_x3,k4_x4,k4_x5;
  double
u1,u2,isa,isb,isA,isB,isC,x1,x2,x3,x4,x5,Fmq,Fmd,isq
,isd,irq,ird,Te,
                        Tl,Ws,Wm;
  double VPhaseA,VPhaseB,VPhaseC;
  double VAlpha,VBeta;
  double omega;
  double Waux,ant,act;
  mc_s3PhaseSystem      p_abc;
  mc_s3PhaseSystem  p_svmabc;
  mc_s3PhaseSystem  p_PwmABC;
        mc_sPhase
        pAlphaBeta;
  mc_sPhase                     pUsAlphaBeta;
        mc_sPhase
        pIsAlphaBeta;
  mc_sPhase                     pPsiRAlphaBeta;
  mc_sPhase                     i_AlphaBeta;
  mc_sPhase                     Psi_R;
  mc_sPhase       pI_S;
        mc_sDQsystem            pDQ;
  mc_sDQsystem                  pDQf;
  mc_sDQsystem     pU_S;
  mc_sDQsystem                  pU_S_ref;
        mc_sAngle                       pSinCos;
  mc_sAngle                     pSinCosf;
  mc_sAngle                     pThetaField;
  fluxmodel_sState  pState;
  mc_sPIparams  i_SD_controllerParams; /*
Parametros de controlador PI

                                                de la
corriente del estator componente D*/
  mc_sPIparams  i_SQ_controllerParams; /*
Parametros de controlador PI

                                                de la
corriente del estator componente Q*/
  mc_sPIparams  FLUX_controllerParams; /*
Parametros de controlador PI

                                                de la
corriente del estator componente D*/
  mc_sPIparams  W_controllerParams; /* Parametros
de controlador PI

                                                de la
corriente del estator componente Q*/
  INOUT_PI                              InOutPI;
  FUZ_SYS fuzzy_system;
  val_fun   value;
  mc_sDQEstabl pDQdata;   /* data structure of the
output quantities */
  dqestabl_sState   pDQstate;
  decoupling_sState sState;

  FILE *fi;
  //clrscr();

        fi=fopen ( "MI12.txt", "w");
printf("****************** EVALUANDO LA
VECINDAD  ************************\n");

  h=0.0000625;
  t=0;
  tf=9.5;
  hh=tf-t;
  n=hh/h;
  h2=h/2;
  band=1;
  u2=0;
  i=0,band=0;
   m=0.5,A=0,B=0,C=0,countupdown=0;
   k1_x1=0,k1_x2=0,k1_x3=0,k1_x4=0,k1_x5=0;
   k2_x1=0,k2_x2=0,k2_x3=0,k2_x4=0,k2_x5=0;
   k3_x1=0,k3_x2=0,k3_x3=0,k3_x4=0,k3_x5=0;
   k4_x1=0,k4_x2=0,k4_x3=0,k4_x4=0,k4_x5=0;
```

```
u1=0,u2=0,isa=0,isb=0,isA=0,isB=0,isC=0,x1=0,x2=
0,x3=0,x4=0,x5=0,Fmq=0;

Fmd=0,isq=0,isd=0,irq=0,ird=0,Te=0,Tl=0,Ws=0,W
m=0;
   VPhaseA=0,VPhaseB=0,VPhaseC=0;
   VAlpha=0,VBeta=0;
   omega=0;
   Waux=0,ant=0,act=0;
         pState.I_KT = I_KT_C; /*constante
dependiente del motor 1*/
         pState.KL_R_KT = KL_R_KT_C; /*
constante dependiente del motor 2*/
         pState.KL_T_KT = KL_T_KT_C; /*
constante dependiente del motor 3 */
         pState.TR_KT = TR_KT_C; /* constante
dependiente del motor 4 */
  pState.T = T_SAMPLE; /* tiempo de muestreo*/
  pState.XAlphaBeta.alpha=0;
  pState.XAlphaBeta.beta=0;
  pAlphaBeta.alpha = 0;
  pAlphaBeta.beta = 0;
         pIsAlphaBeta.alpha = 0;
  pIsAlphaBeta.beta = 0;
  i_AlphaBeta.alpha = 0;
  i_AlphaBeta.beta = 0;
         Psi_R.alpha = 0;
  Psi_R.beta = 0;
  pI_S.alpha = 0;
  pI_S.beta = 0;
  pUsAlphaBeta.alpha = 0;   //se cargan los valores a
la estructura
  pUsAlphaBeta.beta =0;//de voltajes de forma
probicional para probar el modelo
  pPsiRAlphaBeta.alpha =0;//-10.4575830;//0;//-
123.1032648;
  pPsiRAlphaBeta.beta =0;//-2.3168692;//0;//-
27.3116907;
         pDQ.d_axis = 0;
  pDQ.q_axis = 0;
         pDQf.d_axis = 0;
  pDQf.q_axis = 0;
         pU_S.d_axis = 0;
  pU_S.q_axis = 0;
  pU_S_ref.d_axis= 0;
  pU_S_ref.q_axis= 0;
  pSinCos.sine = 0;
  pSinCos.cosine = 0;
  pSinCosf.sine = 0;
  pSinCosf.cosine = 0;
  pThetaField.sine = 0;
  pThetaField.cosine = 0;
  InOutPI.e1=0;
  InOutPI.e2=0;
  InOutPI.e3=0;
  InOutPI.e4=0;
  InOutPI.edot = 0;
  InOutPI.aux = 0;
  InOutPI.uact = 0;
  InOutPI.udot = 0;
  InOutPI.u=0;
  pDQdata.psi_Rd = 0;;
  pDQdata.omega_field = 0;
         pDQdata.i_Sd = 0;
         pDQdata.i_Sq = 0;
  pU_S_ref.q_axis=0;
  value.obj_fun1 = 0;
  value.limleft=8;
  value.limright = value.limleft + 1;
 pDQstate.psi_Rd_zero = PSI_ZERO;/*valor minimo
del flujo magnetico del rotor*/
         pDQstate.LM_TR = LM_TR_C; /* constante
dependiente del motor */
         x1=.0410146114;
         x2=.1789363581;
         x3=.0398642984;
         x4=.1739178334;
         x5=.377;
   sState.ULimit= 176.9;     /*salida de voltaje
maximo del estator limite*/
   sState.LM_LR_TR=14139.30936;/*constante
dependiente del motor 1*/
   sState.LM_LR= 34.655172;  /*constante
dependiente del motor 2 */
   sState.KL=-2.39996195;
   /*i_SD (corriente del estator componente D )
inicialización del control */
   i_SD_controllerParams.Gain_P          = .87;//
   i_SD_controllerParams.Gain_I          =
95.9579129*(.0000625);//
   i_SD_controllerParams.PortionI_K_1     = 0;
   i_SD_controllerParams.LimitPos         = 0;
   i_SD_controllerParams.LimitNeg         = 0;
   i_SD_controllerParams.Desired          = 0;
   /* i_SQ (corriente del estator componente Q )
inicialización del control */
   i_SQ_controllerParams.Gain_P          = .87;//
   i_SQ_controllerParams.Gain_I          =
95.9579129*(.0000625);//
```

```
   i_SQ_controllerParams.PortionI_K_1       = 0;
   i_SQ_controllerParams.LimitPos           = 0;
   i_SQ_controllerParams.LimitNeg           = 0;
   i_SQ_controllerParams.Desired            = 0;
   /*FLUX (magnitud del flujo del rotor )
inicialización del control */
   FLUX_controllerParams.Gain_P             = 4;//
   FLUX_controllerParams.Gain_I             =
183.087041*(0.0000625);
   FLUX_controllerParams.PortionI_K_1       = 0;
   FLUX_controllerParams.LimitPos           = 0;
   FLUX_controllerParams.LimitNeg           = 0;
   FLUX_controllerParams.Desired            = .43;
   /* W (velocidad del rotor ) inicialización del
control */
   W_controllerParams.Gain_P             = 1.77212807
;//
   W_controllerParams.Gain_I             = 40.1004572
*(.0000625);//
   W_controllerParams.PortionI_K_1       = 0;
   W_controllerParams.LimitPos           = 0;
   W_controllerParams.LimitNeg           = 0;
   W_controllerParams.Desired            =
10;//362.01946197066002;
  printf("Numero de pasos de integración:
%f\n\n\n",n);
  //printf("\nRUNNING SIMULATION\n");
      p_abc.PhaseA=0;
      p_abc.PhaseB=0;
      p_abc.PhaseC=0;
      p_PwmABC.PhaseA=0;
      p_PwmABC.PhaseB=0;
      p_PwmABC.PhaseC=0;

p_svmabc.PhaseA=.1796*cos(377*t);//p_svmabc.Pha
seA;//
            p_svmabc.PhaseB=.1796*cos(377*t-
2.0943951); //p_svmabc.PhaseB;//

p_svmabc.PhaseC=.1796*cos(377*t+2.0943951);//p_
svmabc.PhaseC;//
      Waux=.377;

  for (i=0;i<=n;i++)
  {

    //Ws=377;
          VPhaseA= p_svmabc.PhaseA;
          VPhaseB= p_svmabc.PhaseB;
                              VPhaseC=
p_svmabc.PhaseC;
      //fprintf(fi,"%f   %f   %f
%f\n",t1,VPhaseA,VPhaseB,VPhaseC);
  VAlpha = VPhaseA;//Se consideran en esta parte
los ejes positivos hacia abajo
  VBeta = -1/sqrt(3)*VPhaseB  +
1/sqrt(3)*VPhaseC;/*para generar el voltaje del


                                        estator positivo*/
      //fprintf(fi,"%f   %f   %f
%f\n",t1,VAlpha,VBeta,pUsAlphaBeta.beta);

      C = sqrt((VAlpha*VAlpha)+(VBeta*VBeta));
                              A=-VBeta/C;
      B=VAlpha/C;
      u1 = VAlpha*B - VBeta*A;
      u2 = VAlpha*A + VBeta*B;
      ant=act;
      act=atan(-VBeta/VAlpha);
      Ws=(act-ant)/0.0000625;
      if (Ws<=0)
         Ws=Waux;
        else
        Waux=Ws;
      pUsAlphaBeta.alpha = VPhaseA;  //se cargan
los valores a la estructura
      pUsAlphaBeta.beta = 1/sqrt(3) * VPhaseA +
2/sqrt(3) * VPhaseB;//de
      //voltajes de forma probicional para probar el
modelo
      //de flujos pues aun no se utiliza el modelo
      //de desacople y como esntrada al modelo de
flujo se necesitan
                  //como entrada los valores de la
estructura.
      //fprintf(fi,"%f   %f   %f  %f\n",t,u1,u2,Ws);
    if(t<.1)              /*ENTRADA ESCALON*/
     Tl=0.0;
    else
          if (t<2)
          Tl= 11.87;
      else
          if(t<8)
        Tl=0;
        else
                  Tl=11.87;
    if(t<.55)
```

```
            W_controllerParams.Desired =
W_controllerParams.Desired+.04;
        else
            W_controllerParams.Desired = 362;
      Fmq=0.492888670918*x1+0.492888670918*x3;
      Fmd=0.492888670918*x2+0.492888670918*x4;
     isq=(x1-Fmq)/0.754;
     irq=(x3-Fmq)/0.754;
     isd=(x2-Fmd)/0.754;
     ird=(x4-Fmd)/0.754;
     Te=0.007957559681*(x2*isq-x1*isd);//calculo
del par electromagnetico
     //fprintf(fi,"%f   %f   %f   %f   %f
%f\n",t,isq,irq,isd,ird,Te);
     k1_x1=h*funx1(x1,x2,x3,u1,Ws);//Aplicacion de
Runge Kutta de cuarto orden
     k1_x2=h*funx2(x1,x2,x3,u2,Ws);
     k1_x3=h*funx3(x1,x3,x4,x5,Ws);
     k1_x4=h*funx4(x2,x3,x4,x5,Ws);
     k1_x5=h*funx5(Te,Tl);

k2_x1=h*funx1(x1+m*k1_x1,x2+m*k1_x2,x3+m*k
1_x3,u1,Ws);

k2_x2=h*funx2(x1+m*k1_x1,x2+m*k1_x2,x3+m*k
1_x3,u2,Ws);

k2_x3=h*funx3(x1+m*k1_x1,x3+m*k1_x3,x4+m*k
1_x4,x5,Ws);

k2_x4=h*funx4(x2+m*k1_x2,x3+m*k1_x3,x4+m*k
1_x4,x5,Ws);
     k2_x5=h*funx5(Te,Tl);

k3_x1=h*funx1(x1+m*k2_x1,x2+m*k2_x2,x3+m*k
2_x3,u1,Ws);

k3_x2=h*funx2(x1+m*k2_x1,x2+m*k2_x2,x3+m*k
2_x3,u2,Ws);

k3_x3=h*funx3(x1+m*k2_x1,x3+m*k2_x3,x4+m*k
2_x4,x5,Ws);

k3_x4=h*funx4(x2+m*k2_x2,x3+m*k2_x3,x4+m*k
2_x4,x5,Ws);
     k3_x5=h*funx5(Te,Tl);

k4_x1=h*funx1(x1+k3_x1,x2+k3_x2,x3+k3_x3,u1,
Ws);

k4_x2=h*funx2(x1+k3_x1,x2+k3_x2,x3+k3_x3,u2,
Ws);

k4_x3=h*funx3(x1+k3_x1,x3+k3_x3,x4+k3_x4,x5,
Ws);

k4_x4=h*funx4(x2+k3_x2,x3+k3_x3,x4+k3_x4,x5,
Ws);
     k4_x5=h*funx5(Te,Tl);
     x1=x1+(k1_x1+2*k2_x1+2*k3_x1+k4_x1)/6;
     x2=x2+(k1_x2+2*k2_x2+2*k3_x2+k4_x2)/6;
     x3=x3+(k1_x3+2*k2_x3+2*k3_x3+k4_x3)/6;
     x4=x4+(k1_x4+2*k2_x4+2*k3_x4+k4_x4)/6;
     x5=x5+(k1_x5+2*k2_x5+2*k3_x5+k4_x5)/6;
//fprintf(fi,"%.7f   %.7f   %.7f   %.7f   %.7f
%.7f\n",t,x1/377,x2/377,x3/377,x4/377,x5);
     isa = isq*B+isd*A;
     isb = -isq*A+isd*B;
       //fprintf(fi,"%f   %f   %f\n",t,isa,isb);
     isA= isa;
     isB= -0.5*isa - isb * sqrt(3)/2;
     isC= -0.5*isa + isb * sqrt(3)/2;
     //fprintf(fi,"%f   %f   %f   %f
%f\n",t,isA,isB,isC,Wm);
       p_abc.PhaseA = isA;
       p_abc.PhaseB = isB;
       p_abc.PhaseC = isC;
       cptrfmClarke (&pAlphaBeta, &p_abc);
           //fprintf(fi,"%f   %f   %f
%f\n",t,pAlphaBeta.alpha,pAlphaBeta.beta,isb);
          pIsAlphaBeta.alpha = pAlphaBeta.alpha;//se
cargan los valores a la estructura
  pIsAlphaBeta.beta = pAlphaBeta.beta;//de
corrientes que se utilizan en la
       //funcion del calculo del flujo
  Wm=(x5*3600)/754;//calculo para obtener la
velocidad mecanica del rotor en rpm
    //fprintf(fi,"%f  %f\n",t,Wm);
  omega = x5;     // se asigna el valor de la velocidad
electrica del rotor a omega
//para ser utilizada en la función de calculo del flujo
Wm esta en rpm el
//estimador utiliza en rad/segundo se supone que la
velocidad en estado estable
//es casi igual a la velocidad sincrona es decir 377
pero la velodiad mecanica
//(no se utiliza)del rotor por ser un motor de 4 polos
es la mitad es decir
//velocidad maxima en rad/seg es de 188.5.
```

```
//fprintf(fi,"%f %f %f %f %f
%f\n",t,pUsAlphaBeta.alpha,pUsAlphaBeta.beta,
                                    //
          pIsAlphaBeta.alpha,pIsAlphaBeta.beta,omeg
a);
                                    //IMPRESION EN
ARCIVO PARA VERIFICAR LOS DATOS QUE
ENTRAN EN LA FUNCION
      fluxmodel (&pPsiRAlphaBeta, &pState,
&pUsAlphaBeta, &pIsAlphaBeta, omega);
     //llamada a la funcion del calculo del flujo.
//fprintf(fi,"%.7f %.7f %.7f
%.7f\n",t,pPsiRAlphaBeta.alpha,

          //pPsiRAlphaBeta.beta,VPhaseA);
      dqestabl (&pThetaField, &pDQdata,
&pIsAlphaBeta, &pPsiRAlphaBeta, omega,
       &pDQstate);// llamada a la función para
establecer el sistema en d-q.
//fprintf(fi"%f %f %f %.14f\n",t,pDQdata.psi_Rd,
pDQdata.i_Sq, pDQdata.i_Sd);
// impresion de las corriente en el cuadro de
referencia del flujo del rotor
//fprintf(fi,"%f %f %.14fx1= %.14fx2= %.14fx3=
%.14fx4= %.14fx5= %f %f %f %f %f\n",
//t,VPhaseA,x1,x2,x3,x4,x5,pPsiRAlphaBeta.alpha,p
PsiRAlphaBeta.beta,pDQdata.psi_Rd,
// pDQdata.i_Sq, pDQdata.i_Sd);
    //fprintf(fi,"%f %f
%f\n",t,W_controllerParams.Desired,omega );
      InOutPI.aux = InOutPI.e2;
      InOutPI.e2 = (FLUX_controllerParams.Desired-
pDQdata.psi_Rd);
      InOutPI.edot = (InOutPI.e2 -InOutPI.aux);//*.95;
      fuzzy_init2(&fuzzy_system);
      InOutPI.udot =
fuzzy_control(InOutPI.e2,InOutPI.edot,&fuzzy_syste
m, rules);
      i_SD_controllerParams.Desired =
i_SD_controllerParams.Desired + InOutPI.udot;
//printf("%f %f %f\n", InOutPI.e2, InOutPI.edot,
i_SD_controllerParams.Desired);
      //getch();
      //fprintf(fi,"%f %f
%f\n",t,FLUX_controllerParams.Desired,pDQdata.psi
_Rd);
      fuzzy_free(&fuzzy_system);
      InOutPI.aux =InOutPI.e4*.5;
      InOutPI.e4 = (i_SD_controllerParams.Desired-
pDQdata.i_Sd)*.5;//*1.315;
      InOutPI.edot = (InOutPI.e4 -InOutPI.aux);//*.41;
      fuzzy_init3(&fuzzy_system);
      InOutPI.udot =
fuzzy_control(InOutPI.e4,InOutPI.edot,&fuzzy_syste
m, rules);
      pU_S_ref.d_axis = pU_S_ref.d_axis +
InOutPI.udot;
      //fprintf(fi,"%f %f %f\n", InOutPI.e4,
InOutPI.edot, InOutPI.udot);
      //getch();
      //fprintf(fi,"%f %f
%f\n",t,i_SD_controllerParams.Desired,pDQdata.i_S
d);
      fuzzy_free(&fuzzy_system);
      InOutPI.aux =InOutPI.e1;
      InOutPI.e1 = W_controllerParams.Desired -
omega;
      InOutPI.edot = InOutPI.e1 -InOutPI.aux;
      value_Error_Fun(InOutPI.e1, InOutPI.edot,
&value, t);
      fuzzy_init1(&fuzzy_system);
      InOutPI.udot =
fuzzy_control(InOutPI.e1,InOutPI.edot,&fuzzy_syste
m, rules);
      i_SQ_controllerParams.Desired =
i_SQ_controllerParams.Desired + InOutPI.udot;
      fprintf(fi,"%f %f %f %f\n",t, InOutPI.e1,
InOutPI.edot, InOutPI.udot);
      //fprintf(fi,"%f %f\n",t, value.value);
      //getch();
      fuzzy_free(&fuzzy_system);
      InOutPI.aux =InOutPI.e3*0.5;
      InOutPI.e3 = (i_SQ_controllerParams.Desired-
pDQdata.i_Sq)*.5;//*1.315;
      InOutPI.edot = (InOutPI.e3 - InOutPI.aux);//*.41;
      fuzzy_init3(&fuzzy_system);
      InOutPI.udot =
fuzzy_control(InOutPI.e3,InOutPI.edot,&fuzzy_syste
m, rules);
      pU_S_ref.q_axis = pU_S_ref.q_axis +
(InOutPI.udot);//*.95);
      //printf("%f %f %f\n", InOutPI.e3, InOutPI.edot,
pU_S_ref.q_axis);
      //getch();
      //fprintf(fi,"%f %f
%f\n",t,i_SQ_controllerParams.Desired,pDQdata.i_S
q);
      fuzzy_free(&fuzzy_system);
       /*
```

```
    i_SD_controllerParams.Desired =
controllerPI(FLUX_controllerParams.Desired,
    pDQdata.psi_Rd, &FLUX_controllerParams);
    pU_S_ref.d_axis=
controllerPI(i_SD_controllerParams.Desired,
pDQdata.i_Sd,
    &i_SD_controllerParams);//40.12612853763480;
    i_SQ_controllerParams.Desired =
controllerPI(W_controllerParams.Desired,
    omega, &W_controllerParams);
    pU_S_ref.q_axis =
controllerPI(i_SQ_controllerParams.Desired,
pDQdata.i_Sq,

&i_SQ_controllerParams);//175.06014340386338 +
2.74079461766844;
      */
      //fprintf(fi,"%f
%f\n",t,i_SQ_controllerParams.Desired-
pDQdata.i_Sq);
      //fprintf(fi,"%f %f
%f\n",t,i_SD_controllerParams.Desired-
pDQdata.i_Sd,
      //i_SQ_controllerParams.Desired-
pDQdata.i_Sq);
      //fprintf(fi,"%f %f\n",t,pU_S_ref.q_axis-
countupdown);
      countupdown = pU_S_ref.q_axis;
   decoupling (&sState, &pDQdata,&pU_S_ref,
omega,&pU_S);
    //fprintf(fi,"%f %f
%f\n",t,pU_S.d_axis,pU_S.q_axis);
   pSinCos.sine  = pThetaField.sine; //sin(Ws*t);//
   pSinCos.cosine          = pThetaField.cosine;
//cos(Ws*t);//
    pDQ.d_axis
          =pU_S.d_axis;//pUsAlphaBeta.alpha*pSinC
os.cosine +
    //pUsAlphaBeta.beta*pSinCos.sine;
//pU_S.d_axis;//  179.6;//
    pDQ.q_axis              =pU_S.q_axis;//-
pUsAlphaBeta.alpha*pSinCos.sine +
    //pUsAlphaBeta.beta*pSinCos.sine;
//pU_S.q_axis;//0;//
    //fprintf(fi,"%f %f
%f\n",t,pDQ.d_axis,pDQ.q_axis);
   cptrfmParkInv (&pAlphaBeta, &pDQ,
&pSinCos);
    //fprintf(fi,"%f %f
%f\n",t1,pAlphaBeta.alpha,pAlphaBeta.beta);
   svmPwmlct (&pAlphaBeta, &p_svmabc);
   //svmStd (&pAlphaBeta, &p_svmabc);
//fprintf(fi,"%f %f %f
%f\n",t,p_svmabc.PhaseA,p_svmabc.PhaseB,p_svma
bc.PhaseC);
   //invStd(&p_svmabc, &p_PwmABC,
countupdown);
//fprintf(fi,"%f %f %f
%f\n",t1,p_PwmABC.PhaseA,p_PwmABC.PhaseB,p
_PwmABC.PhaseC);
 /*
 if(countupdown<0.0013888 && band == 1)
 countupdown=countupdown+h;
 if(countupdown>0.0013888)
 band=0;
 if(countupdown>0 && band == 0)
 countupdown=countupdown-h;
 if(countupdown<=0)
 band=1;
 */
   if(countupdown<= 0.001388 && band == 1)
 countupdown=countupdown+h;
 if(countupdown>0.001388){
 band=0;
 countupdown=countupdown-h;
 }
 if(countupdown>0 && band == 0)
 countupdown=countupdown-h;
 if(countupdown<=0)
 band=1;
  t=t+h;
 //fprintf(fi,"%f %f \n",t1,countupdown);


 }

 fclose(fi);
 printf("READY\n");
 printf("DATAFILE: 'MI12.txt'\n");
 //getche();
 return(value.value);
}

  /*Definición de las ecuaciones diferenciales del
modelo de estado del MI*/

  double funx1(double x1, double x2, double x3,
double u1, double Ws)
  {
    double funx1;
```

```
    funx1=-110.296714076*x1-
Ws*x2+107.203285924*x3+377*u1;//
    return(funx1);
  }
  double funx2(double x1, double x2, double x3,
double u2, double Ws)
  {
    float funx2;
    funx2=Ws*x1-
110.296714076*x2+107.2032859224*x3+377*u2;
    return(funx2);
  }
  double funx3(double x1, double x3, double x4,
double x5,double Ws )
  {
    double funx3;
    funx3=201.098577735*x1-
206.901422265*x3+x4*(-Ws+x5);
    return(funx3);
  }
        double funx4(double x2, double x3, double
x4, double x5, double Ws)
  {
    double funx4;
    funx4=201.098577735*x2+x3*(Ws-x5)-
206.901422265*x4;
    return(funx4);
  }
        double funx5(double Te, double Tl)
  {
    double funx5;
    funx5=22.4719101124*(Te-Tl); //22.47191011
    return(funx5);
  }
/****************************************
*
*  Modulo: cptrfmClarke ()
*
*  Descripcion:    Esta funcion calcula la
*transformación de Clarke,
* usada para transformar los valores (corriente,
*voltaje, flujo)
*del sistema trifasico estacionario al sistema
*ortogonal
*          estacionario alfa-beta.
*
*Devuelve:        Modifica la estructura apuntada por
el apuntador
* pAlphaBeta de acuerdo a las siguientes ecuaciones:
*
*                    alpha = a
*                    beta  = 1/sqrt(3)*a + 2/sqrt(3)*b
*
*          Argumentos:       La funcion tiene dos
argumentos:
*entrada - p_abc                - Apuntador a la
*estructura que contiene los datos del sistema
*trifasico estacionario.
*salida   - pAlphaBeta  - Apuntador a la estructura
*que contiene los datos de las dos fases del sistema
*ortogonal estacionario.
*
****************************************/
void cptrfmClarke ( mc_sPhase *pAlphaBeta,
mc_s3PhaseSystem *p_abc)
{
pAlphaBeta->alpha = p_abc->PhaseA;
pAlphaBeta->beta = 1/sqrt(3) * p_abc->PhaseA +
2/sqrt(3) * p_abc->PhaseB;
}
/****************************************
*
*          Modulo:       cptrfmClarkeInv()
*
*  Descripcion:    Esta funcion calcula la
*transformacion inversa de
* Clarke que es usada para transformar los valores
*(corriente,voltaje,flujo) del sistema de coordenadas
*ortogonal
* estacionario alfa-beta al sistema de coordenadas
*trifasico
* estacionario.
*  Returns:                        Modifica la
*estructura apuntada * por el apuntador p_abc de
*acuerdo con las siguientes ecuaciones:
*                    a = alpha
*                    b = -0.5*alpha + sqrt(3)/2*beta
*                    c = -(a + b)
*
*Argumentos: La funcion tiene dos argumentos:
*entran  -pAlphaBeta       -apuntador a la estructura
*que contiene
* datos del sistema ortogonal de dos fases
*
*salida            -p_abc   -apuntador a la estrucutra
*que contiene datos del sitema estacionario trifasico
*
*
****************************************/
```

```
void cptrfmClarkeInv ( mc_s3PhaseSystem *p_abc,
mc_sPhase *pAlphaBeta)
{
p_abc->PhaseA = pAlphaBeta->alpha;
p_abc->PhaseB = -0.5* pAlphaBeta->alpha +
sqrt(3)/2* pAlphaBeta->beta;
p_abc->PhaseC = -(p_abc->PhaseA + p_abc-
>PhaseB);
}
/*****************************************
*
*          Modulo:                cptrfmPark()
*
* Descripción:   Esta funcion calcula la
*transformacion de Park que
*   es usada para transformar los valores (corriente,
*voltaje, flujos)
* del sistema de coordenadas ortogonal estacionario
*alfa-beta al
* sistema de coordenadas ortogonal rotatorio d-q.
*
*Devuelve: Modifica la estructura apuntada por el
*apuntador pDQ
*                de acuerdo con las siguientes
*ecuaciones:
*                d_axis = alpha*cos(theta) + beta
*sin(theta)
*                q_axis = beta *cos(theta) -
*alpha*sin(theta)
*
*          Argumentos: La funcion tiene tres
*argumentos:
*salida   -dDQ     -apuntador a la estructura que
*contiene datos
*                del sistema ortogonal de dos fases
*estacionario
*                 DQ.
*
*entran -pAlphaBeta -      Apuntador a la estructura
*que contiene
*                          datos del sistema ortogonal
*rotatorio de
*                          dos fases.
*entran -pSinCos -Apuntador a la estructura donde los
*valores
*                          del seno y el coseno son
cargados
*
*
*****************************************/
void cptrfmPark (mc_sDQsystem *pDQf, mc_sPhase
*pIsAlphaBeta, mc_sAngle *pSinCosf)
{
pDQf->d_axis = pIsAlphaBeta->alpha * pSinCosf-
>cosine + pIsAlphaBeta->beta * pSinCosf->sine;
pDQf->q_axis = -pIsAlphaBeta->alpha * pSinCosf-
>sine + pIsAlphaBeta->beta * pSinCosf->cosine;
}
/*****************************************
*
*          Modulo:            cptrfmParkInv()
*
*Descripcion: Funcion que calcula la transformada
*inversa de Park
*que es usada  para transformar los valores
*(corriente,voltaje,flujo)
* del sistema de coordenadas ortogonal rotatorio d-q
*al sistema de
* coordenadas ortogonal estacionario alfa-beta
*
*Devuelve: Modifica la estructura apuntada por el
*apuntador alpha-
*          beta de acuerdo a las siguientes ecuaciones:
*          alpha = d_axis*cos(theta) - q_axis*sin(theta)
*            beta  = d_axis*sin(theta) +
*q_axis*cos(theta)
*
*          Argumentos:       La funcion tiene tres
*argumentos:
* salida -pAlfhaBeta -Apuntador a la estructura que
*contiene datos
*                    del sistema de coordenadas
*ortogonal de dos
*                fases estcionario
*
*entra  -pDQ       -Apuntador a la estructura que
*contiene datos
*                del sistema de coordenadas
*ortogonal de do‡
*                fases DQ rotatorio
*entra    -pSinCos            -apuntador a la estructura
*con los
*                           valores  de seno y coseno
*
*
*****************************************/
void cptrfmParkInv (mc_sPhase *pAlphaBeta,
mc_sDQsystem *pDQ, mc_sAngle *pSinCos)
{
```

```
pAlphaBeta->alpha = pDQ->d_axis * pSinCos-
>cosine - pDQ->q_axis*pSinCos->sine;
pAlphaBeta->beta = pDQ->d_axis * pSinCos->sine +
pDQ->q_axis*pSinCos->cosine;
}
//void sincos(mc_sAngle *pSinCos,float t)
//{
//pSinCos->sine = sin(377*t=;
//pSinCos->cosine = cos(377*t);
//}
/*****************************************
*
* MOdulo:          fluxmodel()
*
* Descripcion:
*Esta funcion calcula las dos componentes de los ejes
*del flujo de  enlace del rotor en el cuadro de
*referencia estacionario alfa_beta.
* El calculo del flujo esta basado en el modelo
*matematico del        * motor de induccion de AC. La
*funcion  resuleve el sistema de dos
*  ecuaciones diferenciales por el método de Runge
*Kutta de 4to
*
* orden:
*
*  Los parametros del motor:
*   sigma=1-Lm*Lm/(Ls*Lr);
*   KT=Tr+Ts*(1-sigma);
*   I_KT=1/KT;
*   KL_R=Lm/Rs;
*   KL_T=sigma*Lm*Ts;
*   TR=Pp*Lr/Rr;
*
* Argumentos:
*  entrada/salida:   pState   -Apuntador a
*fluxmodel_sState
*                    -Variables de estado de la
*funcion,constantes del
*                    motor, coeficientes del modelo de
*flujo
*
* entrada:     pUsAlphaBeta -apunta a la estructura
*mc_sPashe,
*                              componentes alfa-beta  del
voltaje del
*                              estator
* entrada:     pIsAlphaBeta   -apunta a la estructura
*mc_sPashe, componentes alfa-beta  de corriente del
*estator
*
* entrada:     omega-Velocidad medida del rotor
*
* salida:      pPsiRAlphaBeta -apuntador a la
*estructura mc_sPhase.
*componentes alfa beta calculadas del flujo del
*                    rotor
*
*****************************************/
void fluxmodel (mc_sPhase *pPsiRAlphaBeta,
fluxmodel_sState *pState,

          mc_sPhase *pUsAlphaBeta, mc_sPhase
*pIsAlphaBeta, double omega)
{
 double PsiRpredictAlpha;
 double PsiRpredictBeta;
 double dXpredictAlpha;
 double dXpredictBeta;
 double IsComp32Alpha,IsComp32Beta;
 /* First step - prediction: */
/* PsiRalphapredict =  Xalpha[k-1] + T * dXalpha[k-
1] - KL_T_KT * Isalpha */
 IsComp32Alpha =  pState-
>KL_T_KT*pIsAlphaBeta->alpha;
 PsiRpredictAlpha = pState-
>XAlphaBeta.alpha+(pState-
>dXAlphaBeta.alpha*pState->T)-IsComp32Alpha;
 /* PsiRbetapredict  =  Xbeta[k-1] + T * dXbeta[k-1] -
KL_T_KT * Isbeta */
 IsComp32Beta=pState->KL_T_KT * pIsAlphaBeta-
>beta;
 PsiRpredictBeta = pState-
>XAlphaBeta.beta+(pState-
>dXAlphaBeta.beta*pState->T)-IsComp32Beta;
 /* dXalphapredict = KL_R_KT * uSalpha - I_KT *
PsiRalphapredict - omega * TR_KT *
PsiRbetapredict */
 dXpredictAlpha= pState-
>KL_R_KT*pUsAlphaBeta->alpha -(pState-
>I_KT*PsiRpredictAlpha)-
(omega*PsiRpredictBeta*pState->TR_KT);
/* dXbetapredict  = KL_R_KT * uSbeta + omega *
TR_KT * PsiRalphapredict - I_KT * PsiRbetapredict
*/
 dXpredictBeta=(pState->KL_R_KT*pUsAlphaBeta-
>beta)-(pState-
>I_KT*PsiRpredictBeta)+(omega*PsiRpredictAlpha
*pState->TR_KT);
 /* Second step - trapezoidal method: */
```

```
 /* Xalpha = Xalpha[k-1] + T * (dXalpha[k-1] +
dXalphapredict) / 2 */
 pState->XAlphaBeta.alpha=pState-
>XAlphaBeta.alpha+(pState->T*(pState-
>dXAlphaBeta.alpha+ dXpredictAlpha)/2);
 /* Xbeta  = Xbeta[k-1] + T * (dXbeta[k-1] +
dXbetapredict) / 2  */
 pState->XAlphaBeta.beta = pState-
>XAlphaBeta.beta + (pState->T*(pState-
>dXAlphaBeta.beta+dXpredictBeta))/2;
 /* pdXAlphaBeta = pdXpredictAlphaBeta */
 pState->dXAlphaBeta.alpha=dXpredictAlpha;
 pState->dXAlphaBeta.beta=dXpredictBeta;
 /* Final step - Flux calculation */
 /* PsiRalpha =  Xalpha - KL_T_KT * Isalpha */
 pPsiRAlphaBeta->alpha= pState-
>XAlphaBeta.alpha-IsComp32Alpha;
 /* PsiRbeta  =  Xbeta - KL_T_KT * Isbeta */
 pPsiRAlphaBeta->beta=pState->XAlphaBeta.beta-
IsComp32Beta;
}
/****************************************
Modulo: dqestabl()
*
*Descripción:
*Esta funcion establece el sistema de coordenadas d-q
*(definido por el vector del flujo magnetico del rotor)
*y calcula la  velocidad de rotacion del campo del
*rotor.
*
* El codigo de desarrolla usando las siguientes
*ecuaciones:
*
*          a) Calculo de la magnitud del flujo
magnetico del rotor
*
*yrd = sqrt((YrA*YrA)+(YrB*YrB)); // Magnitud del
*flujo del
* rotor
*b) Calculo del seno de teta de campo y del coseno de
*teta de
* campo.
*       sinef =    YrB/Yrd;
*       cosinef =  YrA/Yrd;
*
*c) Calculo de las componentes del eje directo y de
cuadratura.
*
*                    d_axis = alpha*cos(theta) + beta
*sin(theta)
*                    q_axis = beta *cos(theta) -
*alpha*sin(theta)
*
*          d) Calculo de la omega de campo
*
*      omega_field = ((LM_TR * i_Sq[k]) / psi_Rd) +
*omega
*
*donde:   LM_TR es la constante derivada de los
parametros del
* motor:            inductancia_mutua / constante de
*tiempo del rotor
*
*Argumentos:
*
*Salida:  pThetaField- apuntador a la estructura
mc_sAngle - seno y coseno de la posición del flujo
*magnetico del rotor.
*          Salida:  pDQdata- apuntador ala estructurura
*mc_sDQEstabl
*           flujo del rotor, omega de campo, corrientes
*del estator en
*          d-q
*  entrada: pIsAlphaBeta - apuntador a la estructura
*mc_sPhase componentes de corriente del estator
*alfa, beta.
*
*
*entrada: pPsiRAlphaBeta - apuntador a la estructura
*mc_sPhase -
*          componentes del flujo magnetico del rotor
*alfa, beta.
*
*          entrada: omega - velocidad medida del rotor
*
*entrada: pDQstate - apuntador a la estructura
*dqestabl_sState
*variables de estado de la función, constantes
*dependientes del motor.
*
****************************************/
void dqestabl( mc_sAngle *pThetaField,
mc_sDQEstabl *pDQdata,

          mc_sPhase *pIsAlphaBeta, mc_sPhase
*pPsiRAlphaBeta,
           double omega, dqestabl_sState *pDQstate)
{
pDQdata->psi_Rd = sqrt((pPsiRAlphaBeta-
>alpha*pPsiRAlphaBeta->alpha)+(pPsiRAlphaBeta-
```

```
>beta*pPsiRAlphaBeta->beta));// Magnitud del flujo
del rotor
pThetaField->sine = pPsiRAlphaBeta-
>beta/pDQdata->psi_Rd;
pThetaField->cosine = pPsiRAlphaBeta-
>alpha/pDQdata->psi_Rd;
pDQdata->i_Sd = pIsAlphaBeta->alpha *
pThetaField->cosine + pIsAlphaBeta->beta *
pThetaField->sine;
pDQdata->i_Sq = -pIsAlphaBeta->alpha *
pThetaField->sine + pIsAlphaBeta->beta *
pThetaField->cosine;
pDQdata->omega_field = ((pDQstate->LM_TR *
pDQdata->i_Sq) / pDQdata->psi_Rd) + omega;
}
/****************************************
* Modulo: ControlPI
*
* Descripción: Funcion del control convencional PI
*
* Argumentos:
*
*          entrada: Valor deseado
*
*          entrada: Valor medido
*
*entrada/salida: apuntador a pParams de la estructura
*                    mc_tPIparams que contiene los
parametros del
*                    controlador.
****************************************/
double controllerPI (double DesiredValue, double
MeasuredValue,

            mc_sPIparams *pParams)
{
double PortionK,PortionI,e,PIoutput;
  e = DesiredValue - MeasuredValue;
  PortionK = pParams -> Gain_P * e;
  PortionI = pParams -> Gain_I * e;
  PortionI = PortionI + pParams -> PortionI_K_1;
  pParams -> PortionI_K_1 =  PortionI;
  PIoutput = PortionK + PortionI; /* salida del
controlador */
  return(PIoutput);
}

/****************************************
* Modulo: decoupling()
*
*Descripción:
*Esta funcion calcula las componentes de desacople,
*entonces el
* sitema no lineal (motor de inducción de CA)esta
*completado y la
* salida de voltaje esta limitada dentro de los limites
*deseados.
*
*La tecnica de desacople es usada para transformar el
*sistema no
* lineal a uno lineal. Tal transformacion abilida el uso
*de controles
* lineales (PI,PID ...) y asi la tarea de control se
*facilita.
*
* uSd = uSd_ref - (omega_field * i_Sq * KL +
*LM_LR_TR *
* psi_Rd)
* uSq = uSq_ref + omega_field * i_Sd * KL + omega
* LM_LR *
* psi_Rd
*
* Si el modulo del voltaje del estator es mas alto que
*el limite
*(ULimit)
*entonces uSd y uSq estan limitados. El algoritmo
*limita el modulo
* del vector
*de voltaje. La componente-d esta limitada dentro de
*los mismos
* limites que el vector del modulo. Esto permite  un
*apropiado
* control de psi_Rd. Entonces los limites para la
*componente-q son
* calculados de forma semejante que el modulo del
*vector
* resultante no exeda los limites.
*Algunos calculos intermedios son  hechos en 32 bit
*para obtener  una mayor exactitud.
*
*          Devuelve: ninguno
*
*          Argumentos:
*          entrada: sState - apuntador a
decouplin_sState
* - constante dependiente del motor y limite del
voltaje del estator
*(LM_LR_TR,LM_LR,KL,ULimit)
*  entrada:          pDQdata - apuntador a la estructura
```

```
*                       mc_sDQEstabl - flujo del rotor
*(psi_Rd), omega *de campo (omega_field),
*corrientes *del
*           estator en d-q (i_Sd,i_Sq)
*           entrada: pU_S_ref - apuntador a
*mc_sDQsystem referencia del voltaje del estator
*           (uSd_ref, uSq_ref)
*entrada: omega - medicion de la velocidad de la
flecha del motor
*salida:  pU_S - apuntador a la estructura de salida
*mc_sDQsystem
*- resultante del voltaje del estator (uSd, uSq)
*
****************************************/
void decoupling (decoupling_sState *sState,
mc_sDQEstabl *pDQdata,
mc_sDQsystem *pU_S_ref, double omega,
mc_sDQsystem *pU_S)
{
double SqLimitMax,SqLimitMin;
//pU_S->d_axis = pU_S_ref->d_axis - ( pDQdata-
>omega_field * pDQdata->i_Sq * sState->KL +
sState->LM_LR_TR * pDQdata->psi_Rd);
//pU_S->q_axis = pU_S_ref->q_axis + ( pDQdata-
>omega_field * pDQdata->i_Sd * sState->KL +
omega * sState->LM_LR * pDQdata->psi_Rd);

pU_S->d_axis = pU_S_ref->d_axis -
(.0039439165399 * pDQdata->omega_field *
pDQdata->i_Sq );// sin considerar el retardo del
inversor.
pU_S->q_axis = pU_S_ref->q_axis +
((.0039439165399 * pDQdata->omega_field *
pDQdata->i_Sd ) + (.971953510646 * pDQdata-
>omega_field * pDQdata->psi_Rd));

//pU_S->d_axis = pU_S_ref->d_axis - (-
2.39996195006 * pDQdata->omega_field * pDQdata-
>i_Sq )-(pU_S_ref->q_axis*pDQdata-
>omega_field*0.0000625);//Considerando el sistema
de desacople y  del sistema de retardo de
accionamiento
//pU_S->q_axis = pU_S_ref->q_axis + ((-
2.39996195006* pDQdata->omega_field * pDQdata-
>i_Sd ) + (34.65517482759426 * pDQdata-
>omega_field * pDQdata->psi_Rd))+(pU_S_ref-
>d_axis*pDQdata->omega_field*0.0000625);

//pU_S->d_axis = pU_S_ref->d_axis - ((-
2.39996195006+(0.0000625*0.435))* pDQdata-
>omega_field * pDQdata->i_Sq)-(pDQdata-
>omega_field*pDQdata->omega_field)*(((-
2.39996195006*0.0000625)*pDQdata-
>i_Sd)+((0.0000625*(0.002+2.39996195006))*(pDQ
data->psi_Rd / 0.06931034)));  //tomando en cunta
desacople dentro del acionamiento
//pU_S->q_axis = pU_S_ref->q_axis + ((-
2.39996195006+(0.0000625*0.435))* pDQdata-
>omega_field * pDQdata->i_Sd)+ (pDQdata-
>omega_field*pDQdata->omega_field)*(0.0000625
* (-2.39996195006))* (pDQdata->i_Sq);
}
/****************************************
*
*           Modulo: svmPwmIcl()
*
*Descripción: Esta función calcula los ciclos de
trabajo necesarios
* para generar el voltaje del estator requerido usando
la tecnica
* general de Modulación senoidal.
*
* Devuelve: Llena las variables de la fase_A, fase_B
*y fase_C  en la
* estructura de datos mc_s3PhaseSystem y regresa el
*numero del  sector en el que reside el vector de
*voltaje.
*
*           Argumentos:
*           entrada(s):
*     -p_AlphaBeta - Directo (alfa) y cuadratura (beta)
*componente del vector de voltaje del estator en el
*cuadro de referencia estacionario.
*
*salida(s):
*-p_abc    - apuntador al ciclo de trabajo de la fase_A,
*fase_B, fase_C.
*
****************************************/
 void svmPwmlct (mc_sPhase *pAlphaBeta,
mc_s3PhaseSystem *p_svmabc)
 {
  double Uref1,Uref2,Uref3,sec,seci;
  int a,b,c;
/*transformada inversa de Clarke Modificada*/
       Uref1  = pAlphaBeta->beta;
       Uref2  = pAlphaBeta->alpha*sqrt(3)/2-
pAlphaBeta->beta/2;
       Uref3  = -pAlphaBeta->alpha*sqrt(3)/2-
pAlphaBeta->beta/2;
```

```
 if (Uref1>0)
    a=1;
    else
    a=0;
 if (Uref2>0)
                    b=2;
    else
    b=0;
 if (Uref3>0)
                    c=4;
    else
    c=0;
seci=a+b+c;
if (seci=1)
                    sec=2;
if (seci=2)
                    sec=6;
if (seci=3)
                    sec=1;
if (seci=4)
                    sec=4;
if (seci=5)
                    sec=3;
 if (seci=6)
                    sec=5;
/*Utilizar PARA OBTENER LOS CICLOS DE
TRABAJO DEL ACCIONAMIENTO
p_svmabc->PhaseA = 89.8 + (pAlphaBeta->alpha/2);
p_svmabc->PhaseB = 89.8 +(pAlphaBeta-
>beta*sqrt(3)/4 - pAlphaBeta->alpha/4);
//p_svmabc->PhaseC = -(p_svmabc->PhaseA +
p_svmabc->PhaseB);
p_svmabc->PhaseC = 89.8+(-pAlphaBeta-
>beta*sqrt(3)/4 - pAlphaBeta->alpha/4);
                                        */
/*Por ahora solo se utiliza  la parte de la
transformacion inversa d park para alimentar el
modelo*/

p_svmabc->PhaseA = pAlphaBeta->alpha;
p_svmabc->PhaseB = pAlphaBeta->beta*sqrt(3)/2 -
pAlphaBeta->alpha/2;
//p_svmabc->PhaseC = -(p_svmabc->PhaseA +
p_svmabc->PhaseB);
p_svmabc->PhaseC = -pAlphaBeta->beta*sqrt(3)/2 -
pAlphaBeta->alpha/2;
}
/*****************************************
*
*          Modulo: svmStd()
*
*Descripción: Esta funcion calcula el ciclo de trabajo
*apropiado necesario
*          para generar un voltaje de estator de
*referencia dado usando la modulación de
*          vectores espaciales con un ciclo de trabajo
*nulo del estado de swicheo de los
*          estados O000 y 0111 en cada sector del
*exagono.
*
* Devuelve:   La función llena las variables de la
*fase_A, fase_B y fase_C  en la
*                    estructura de datos
*mc_s3PhaseSystem y regresa el numero de sector
del
*        vector de voltaje del estator residente dentrm.
*
*Argumentos:
*                    entrada(s):
*                                        -
*p_AlphaBeta - Directo (alfa) y cuadratura (beta)
*componente del vector de voltaje del estator en el
*cuadro de referencia estacionario,
*

*          salida(s)
*-p_abc  - apuntador al ciclo de trabajo de la fase_A,
*fase_B, fase_C.
*
*****************************************/
void svmStd (mc_sPhase *pAlphaBeta,
mc_s3PhaseSystem *p_svmabc)
 //para regresar el valor del sector aculculado usar int
{
double Uref1,Uref2,Uref3,sec;

/*transformada inversa de Clarke Modificada*/
       Uref1  = pAlphaBeta->beta;
       Uref2  = pAlphaBeta->alpha*sqrt(3)/2-
pAlphaBeta->beta/2;
       Uref3  = -pAlphaBeta->alpha*sqrt(3)/2-
pAlphaBeta->beta/2;
/* Identificacion del sector  */

   if (Uref3 > 0 && Uref2 > 0)              /*sector 5*/
    {
   /* Sector 5 calculation(s) */
    p_svmabc->PhaseB = (179.6+pAlphaBeta-
>beta)/2;
```

```
    p_svmabc->PhaseA = p_svmabc-
>PhaseB+pAlphaBeta->alpha*sqrt(3)/2-pAlphaBeta-
>beta/2;
    p_svmabc->PhaseC = p_svmabc->PhaseA-
pAlphaBeta->alpha*sqrt(3)/2-pAlphaBeta->beta/2;
    sec     = 5;
     }

   if (Uref3 > 0 && Uref2 <= 0 && Uref1 > 0)
/*sector 3*/
     {
    /* Sector 3 calculation(s) */
    p_svmabc->PhaseA = (179.6+pAlphaBeta-
>alpha*sqrt(3)/2-pAlphaBeta->beta/2)/2;
    p_svmabc->PhaseC = p_svmabc->PhaseA-
pAlphaBeta->alpha*sqrt(3)/2-pAlphaBeta->beta/2;
    p_svmabc->PhaseB = p_svmabc-
>PhaseC+pAlphaBeta->beta;
    sec     = 3;
     }

    if (Uref3 > 0 && Uref2 <= 0 && Uref1 <= 0)
/*sector 4 */
    {
   /* Sector 4 calculation(s) */
   p_svmabc->PhaseA = (179.6+pAlphaBeta-
>alpha*sqrt(3)/2+pAlphaBeta->beta/2)/2;
   p_svmabc->PhaseB = p_svmabc->PhaseA-
pAlphaBeta->alpha*sqrt(3)/2+pAlphaBeta->beta/2;
   p_svmabc->PhaseC = p_svmabc->PhaseB-
pAlphaBeta->beta;
   sec     = 4;
     }

    if (Uref3 <= 0 && Uref2 > 0 && Uref1 > 0)
/*sector 1   */
    {

   /* Sector 1 calculation(s) */

   p_svmabc->PhaseC = (179.6-pAlphaBeta-
>alpha*sqrt(3)/2-pAlphaBeta->beta/2)/2;
   p_svmabc->PhaseB = p_svmabc-
>PhaseC+pAlphaBeta->beta;
   p_svmabc->PhaseA = p_svmabc-
>PhaseB+pAlphaBeta->alpha*sqrt(3)/2-pAlphaBeta-
>beta/2;
   sec     = 1;

    }

    if (Uref3 <= 0 && Uref2 > 0 && Uref1 <= 0)
/*sector 6*/
    {
   /*Sector 6 calculation(s) */
   p_svmabc->PhaseB = (179.6-pAlphaBeta-
>alpha*sqrt(3)/2+pAlphaBeta->beta/2)/2;
   p_svmabc->PhaseC = p_svmabc->PhaseB-
pAlphaBeta->beta;
   p_svmabc->PhaseA = p_svmabc-
>PhaseC+pAlphaBeta->alpha*sqrt(3)/2+pAlphaBeta-
>beta/2;
   sec     = 6;
    }

    if (Uref3 <= 0 && Uref2 <= 0) /*sector 2*/
    {
   /* Sector 2 calculation(s) */
   p_svmabc->PhaseC = (179.6-pAlphaBeta->beta)/2;
   p_svmabc->PhaseA = p_svmabc-
>PhaseC+pAlphaBeta->alpha*sqrt(3)/2+pAlphaBeta-
>beta/2;
   p_svmabc->PhaseB = p_svmabc->PhaseA-
pAlphaBeta->alpha*sqrt(3)/2+pAlphaBeta->beta/2;
   sec = 2;
    }

   /* Limit outputs */

   if (p_svmabc->PhaseA < 0 )
   p_svmabc->PhaseA = 0;

   if (p_svmabc->PhaseB < 0 )
   p_svmabc->PhaseB = 0;

   if (p_svmabc->PhaseC < 0 )
   p_svmabc->PhaseC = 0;
}

void invStd(mc_s3PhaseSystem *p_svmabc,
mc_s3PhaseSystem *p_PwmABC,
                                    double
countupdown)
{

double contupdown, timer,h,i,A,B,C;

A= p_svmabc->PhaseA/179.6 *0.0013888;
B= p_svmabc->PhaseB/179.6 *0.0013888;
C= p_svmabc->PhaseC/179.6 *0.0013888;
```

```
timer=countupdown;

            if (A >= timer )
   p_PwmABC->PhaseA = 1;
   else
   p_PwmABC->PhaseA = 0;

   if (B >= timer )
   p_PwmABC->PhaseB = 1;
   else
   p_PwmABC->PhaseB = 0;

   if (C >= timer )
   p_PwmABC->PhaseC = 1;
   else
   p_PwmABC->PhaseC = 0;

/*    timer=countupdown;


            if (A >= timer )
   p_PwmABC->PhaseA = 1;
   else
   p_PwmABC->PhaseA = 0;

   if (B >= timer )
   p_PwmABC->PhaseB = 1;
   else
   p_PwmABC->PhaseB = 0;

   if (C >= timer )
   p_PwmABC->PhaseC = 1;
   else
   p_PwmABC->PhaseC = 0;

  */
}
void invStd(mc_s3PhaseSystem *p_svmabc,
mc_s3PhaseSystem *p_PwmABC, float
countupdown)
{

float contupdown, timer,i,A,B,C;

A= p_svmabc->PhaseA*0.00003125;
B= p_svmabc->PhaseB*0.00003125;
C= p_svmabc->PhaseC*0.00003125;
timer=countupdown;

            if (A >= timer )
   p_PwmABC->PhaseA =1;
   else
   p_PwmABC->PhaseA = -1;

   if (B >= timer )
   p_PwmABC->PhaseB =1;
   else
   p_PwmABC->PhaseB = -1;

   if (C >= timer )
   p_PwmABC->PhaseC = 1;
   else
   p_PwmABC->PhaseC = -1;

}
/****************************************
Rutinas del control difuso.
****************************************/
/****************************************
* Modulo: fuzzy_init1()
*
*Descripción:
*Asigna memoria para todas las posibles reglas y
inicializa el ancho
* y los centros para cada funcion de membresia.
*
*Esta inicializacion es para los valores del control de
velocidad.
*
*          Devuelve: ninguno
*
*          Argumentos:
*          entrada: fuzzy_system -  Apuntador a
FUZZY_SYS.
*
*salida:  fuzzy system - apuntador a la estructura de
*salida
* FUZZY_SYS - valores  de width, center, para e,
*edot, outmem.
*
****************************************/
void fuzzy_init1(FUZ_SYS *fuzzy_system) {
  int i;
  if (!(fuzzy_system->emem = (IN_MEM *)
malloc(sizeof(IN_MEM)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
```

```
  if (!(fuzzy_system->edotmem = (IN_MEM *)
malloc(sizeof(IN_MEM)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->outmem = (OUT_MEM *)
malloc(sizeof(OUT_MEM)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->emem->center = (double *)
malloc(5*sizeof(double)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->emem->dom = (double *)
malloc(5*sizeof(double)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->edotmem->center = (double *)
malloc(5*sizeof(double)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->edotmem->dom = (double *)
malloc(5*sizeof(double)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->outmem->center = (double *)
malloc(5*sizeof(double)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }

  /* Inicialización para las funciones de membresia
del control para el motor. */
  fuzzy_system->emem->width = 1.5;  /* Ancho
definido como 1/2 de la base del triangulo. */
  fuzzy_system->edotmem->width = 0.01;
  fuzzy_system->outmem->width = 0.03;

  for (i=0; i<5; i++) {    /* localizacion de los centros
de las funciones de membresia*/
    fuzzy_system->emem->center[i] = (-3 + i*1.5);
    fuzzy_system->edotmem->center[i] = (-0.02 +
i*0.01);
    fuzzy_system->outmem->center[i] = (-0.045 +
i*0.0225);
  }
}
/*****************************************
* Modulo: fuzzy_init2()
*
*Descripción:
*Asigna memoria para todas las posibles reglas y
*inicializa el ancho  y los centros para cada funcion
*de membresia.
*
*          Esta inicializacion es para los valores del
*control de flujo.
*
*          Returs: ninguno
*
*          Argumentos:
*          entrada: fuzzy_system -  Apuntador a
*FUZZY_SYS.
*
*          salida:    fuzzy system - apuntador a la
*estructura de salida FUZZY_SYS - valores
*                    de width, center, para e, edot,
*outmem.
*
*****************************************/
void fuzzy_init2(FUZ_SYS *fuzzy_system) {

  int i;
  if (!(fuzzy_system->emem = (IN_MEM *)
malloc(sizeof(IN_MEM)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->edotmem = (IN_MEM *)
malloc(sizeof(IN_MEM)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->outmem = (OUT_MEM *)
malloc(sizeof(OUT_MEM)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->emem->center = (double *)
malloc(5*sizeof(double)))) {
    printf("Error allocating memory.\n");
    exit(1);
  }
  if (!(fuzzy_system->emem->dom = (double *)
malloc(5*sizeof(double)))) {
```

```
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->edotmem->center = (double *)
malloc(5*sizeof(double)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->edotmem->dom = (double *)
malloc(5*sizeof(double)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->outmem->center = (double *)
malloc(5*sizeof(double)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }

  /* Inicialización para las funciones de membresia
del control para el motor. */
  fuzzy_system->emem->width = 0.075;  /* Ancho
definido como 1/2 de la base del triangulo. */
  fuzzy_system->edotmem->width = 0.00014;
  fuzzy_system->outmem->width = .0006;

  for (i=0; i<5; i++) {
   fuzzy_system->emem->center[i] = (-0.15 +
i*.075);
   fuzzy_system->edotmem->center[i] = (-0.00028 +
i*0.00014);
   fuzzy_system->outmem->center[i] = (-0.0012 +
i*0.0006);
  }
}

/****************************************
* Modulo: fuzzy_init3()
*
*Descripción:
*Asigna memoria para todas las posibles reglas y
inicializa el ancho
* y los centros para cada funcion de membresia.
*
*Esta inicializacion es para los valores de los
controles de corriente.
*
*          Devuelve: ninguno
*
*          Argumentos:
*          entrada: fuzzy_system -  Apuntador a
FUZZY_SYS.
*
*          salida:    fuzzy system - apuntador a la
*estructura de salida FUZZY_SYS - valores
*                    de width, center, para e, edot,
*outmem.
*
****************************************/
void fuzzy_init3(FUZ_SYS *fuzzy_system) {

  int i;
  if (!(fuzzy_system->emem = (IN_MEM *)
malloc(sizeof(IN_MEM)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->edotmem = (IN_MEM *)
malloc(sizeof(IN_MEM)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->outmem = (OUT_MEM *)
malloc(sizeof(OUT_MEM)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->emem->center = (double *)
malloc(5*sizeof(double)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->emem->dom = (double *)
malloc(5*sizeof(double)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->edotmem->center = (double *)
malloc(5*sizeof(double)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->edotmem->dom = (double *)
malloc(5*sizeof(double)))) {
   printf("Error allocating memory.\n");
   exit(1);
  }
  if (!(fuzzy_system->outmem->center = (double *)
malloc(5*sizeof(double)))) {
   printf("Error allocating memory.\n");
```

```
   exit(1);
  }

  /* Inicialización para las funciones de membresia
del control para el motor. */
  fuzzy_system->emem->width = 1.6;  /* Ancho
definido como 1/2 de la base del triangulo. */
  fuzzy_system->edotmem->width = 0.028;
  fuzzy_system->outmem->width = 0.017;

  for (i=0; i<5; i++) {
    fuzzy_system->emem->center[i] = (-3.2 + i*1.6);
    fuzzy_system->edotmem->center[i] = (-0.056 +
i*0.028);
    fuzzy_system->outmem->center[i] = (-0.034 +
i*.017);
  }
}
/****************************************
* Modulo: fuzzy_free()
*
*Descripción:
*          Libera la memoria asignada para todas las
posibles reglas
*          que se inicializan con el uso del control
*difuso.
*
*
*          Returs: ninguno
*
*          Argumentos:
*          entrada: fuzzy_system -  Apuntador a
*FUZZY_SYS.
*
*          salida:    fuzzy system - apuntador a la
*estructura de salida FUZZY_SYS - valores
*                     de width, center, para e, edot,
*outmem.
*
****************************************/
void fuzzy_free(FUZ_SYS *fuzzy_system) {

  free(fuzzy_system->emem->center);
  free(fuzzy_system->emem->dom);
  free(fuzzy_system->edotmem->center);
  free(fuzzy_system->edotmem->dom);
  free(fuzzy_system->outmem->center);
  free(fuzzy_system->emem);
  free(fuzzy_system->edotmem);
  free(fuzzy_system->outmem);
}
/****************************************
* Modulo: fuzzy_control()
*
*Descripción:
*Modulo principal del control difuso el cual llama
*todas los modulos requeridos para el calculo de la
*salida de control.
*
*
*          Regresa: valor de la salida de control.
*
*          Argumentos:
*entrada: e -  error obtenido en el sistema e = valor
deseado - valor
*  real.
*entrada:edot -  cambio del error en el sistema ce = e
actual- e
* anterior.
*entrada:fuzzy system - apuntador a la estructura
*FUZZY_SYS -
* valores de width, center, e, edot, outmem.
*
****************************************/
double fuzzy_control(double e, double edot,
FUZ_SYS *fuzzy_system, int rules[SIZE][SIZE]) {

  int pos[2];

  fuzzyify(e, fuzzy_system->emem);
  fuzzyify(edot, fuzzy_system->edotmem);
  match(fuzzy_system->emem, fuzzy_system-
>edotmem, pos);
  return inf_defuzz(fuzzy_system->emem,
fuzzy_system->edotmem,
fuzzy_system->outmem, pos, rules);
}
/****************************************
* Modulo: fuzzyfy()
*
*Descripción:
*Esta funcion difusifica la entrada u para determinar
*el grado de
* membresía para cada funcion (in_mem).
*
*Se asumen 5 funciones de membresia, con la
*primera y ultima
* funcion de membresia tipo ridilla, la demas son
*triangulares.
*
```

```
*
*           Returs: ninguno
*
*           Argumentos:
*           entrada: u -  valor obtenido del sistema e o
edot.
*
*salida:  mem - apuntador a la estructura de salida
*IN_MEM - valores
*resultantes del grado de membresia para cada una de
*las funciones.
*
****************************************/
void fuzzyify(double u, IN_MEM *mem) {

  int i;

  mem->dom[0] = leftall(u, mem->width, mem-
>center[0]);
  for (i=1; i<4; i++)
    mem->dom[i] = triangle(u, mem->width, mem-
>center[i]);
  mem->dom[4] = rightall(u, mem->width, mem-
>center[4]);
}
/****************************************
* Modulo: leftall()
*
*Descripción:
*Determia el grado de membresia para la funcion de
*membresia.
* rodilla izquierda
*
*Regresa: Grado de membresia para la funcion de
membresia
* rodilla izquierda.
*           Argumentos:
*           entrada: u -  valor obtenido del sistema e o
*edot.
*
*           entrada: c - centro de la funcion de
*membresia.
*
*           entrada: w - ancho de la funcion de
*membresia.
*
****************************************/
double leftall(double u, double w, double c)
{
  if (u < c)
    return 1.0;
  else
    return MAX(0,(1-(u-c)/w));
}
/****************************************
* Modulo: rightall()
*
*Descripción:
*Determia el grado de membresia para la funcion de
*membresia
* rodilla derecha.
*
*Regresa: Grado de membresia para la funcion de
*membresía
* rodilla derecha.
*
*           Argumentos:
*           entrada: u -  valor obtenido del sistema e o
*edot.
*
*           entrada: c - centro de la funcion de
*membresia.
*
*entrada: w - ancho de la funcion de *membresia.
*
****************************************/
double rightall(double u, double w, double c)
{
  if (u >= c)
        return 1.0;
  else
        return MAX(0,(1-(c-u)/w));
}
/****************************************
* Modulo: triangle()
*
*Descripción:
*Determia el grado de membresia para las funciones
*de membresia  triangulares.
*
*           Regresa: Grado de membresia para la
*funcion triangular.
*
*           Argumentos:
*           entrada: u -  valor obtenido del sistema e o
*edot.
*
*           entrada: c - centro de la funcion de
*membresia.
*
```

```
*           entrada: w - ancho de la funcion de
*membresia.
*
****************************************/
double triangle(double u, double w, double c)
{
 if (u >= c)
   return MAX(0,(1-(u-c)/w));
 else
   return MAX(0,(1-(c-u)/w));
}
/****************************************
* Modulo: match()
*
*Descripción:
*           Determina cuales reglas estan activada.
*
*
*           Regresa: Ninguna.
*
*           Argumentos:
*           entrada: emem -  apuntador a la estructura
*IN_MEM.
*
*           entrada: edotmem - apuntador a la estructura
*IN_MEM.
*
*salida: pos - apuntador al arreglo pos donde se
*guarda que regla  fue activada.
****************************************/
void match(const IN_MEM *emem, const IN_MEM
*edotmem, int *pos) {

 int i;

 for (i=0; i<5; i++) {
  if(emem->dom[i] != 0) {
    pos[0] = i;
    break;
  }
 }
 for (i=0; i<5; i++) {
  if(edotmem->dom[i] != 0) {
    pos[1] = i;
    break;
  }
 }
}
/****************************************
* Modulo: inf_defuzz()
*
*Descripción:
*           Modulo de inferencia y dedifusicacción.
*
*
*           Regresa: Grado de membresia para la
*funcion de membresia rodilla izquierda.
*
*           Argumentos:
*           entrada: emem - .
*
*           entrada: edotmem - .
*
*           entrada: pos - .
*
*           salida: outmem -
*
****************************************/
double inf_defuzz(IN_MEM *emem, IN_MEM
*edotmem, OUT_MEM *outmem, int *pos,

           int rules[SIZE][SIZE]) {

 double outdom, area, Atot = 0, WAtot = 0;
 int i, j, k, l, out_index;

 for(i=0; i<2; i++) {
  for(j=0; j<2; j++) {
   if (((pos[0]+i)<5) && ((pos[1]+j)<5)) { /*
Observe que los limites no son
    exedidos. */
     outdom = 0;

        k=pos[0]+i;
        l=pos[1]+j;

       // printf("pos0= %d,
pos1=%d\n",pos[0],pos[1]);

        out_index = rules[k][l];
      // printf("out_index %d\n",out_index);


      /* Determina la certeza de la premiza*/
      outdom = MIN((emem->dom[pos[0]+i]),
(edotmem->dom[pos[1]+j]));
      // printf("outdom %f\n",outdom);

      /* Dedifusificación */
```

```
      area = 2*outmem->width*(outdom -
(outdom*outdom)/2);
      Atot += area;
      WAtot += area*outmem->center[out_index];
        //printf("Atot %f, WAtot %f\n",Atot, WAtot);
    }
   }
  }
  /* Regresa el valor crisp. El signo menos es
requerido para dar una salida
  correcta del sistema del motor de induccion, note
que este signo menos actualmente
  asegura que la tabla de indices trabajada es como la
trabajada fuera.
   */
  return (WAtot/Atot);
}

/*****************************************
Rutinas de la busqueda por tabu para el universo del
discurso.
*****************************************/

/*****************************************
* Modulo value_Error_fun().
*
* Descripción:
*          Calcula el valor de la funcion objetivo.
*
*
*   Devuelve:    La evaluación de la regla en la
*vecindad
*
*   Argumentos:
*
*   entrada:    -  e             -Valor del error de
*velocidad.
*
*  - edot    -Cambio en el error de velocidad.
*
* -   t     - tiempo.
*
*   salida:      - value   - Valor obtenido de la función
*objetivo.
*****************************************/
void value_Error_Fun(double e, double edot, val_fun
*value, double t)
{
double a = value->limright + 0.0000625;
                    if (t>value->limleft && t<=(value-
>limright)){
    value->obj_fun1 += sqrt(e*e) + sqrt(edot*edot);
    }
    if (t>value->limright && t < a){
    value->value= value->obj_fun1;
    value->obj_fun1 = 0;
    }
}
```