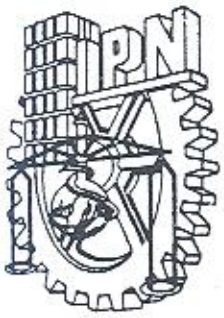


INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE INGENIERÍA
MECÁNICA Y ELÉCTRICA

SECCIÓN DE ESTUDIOS DE POSGRADO
E INVESTIGACIÓN

DESARROLLO DE UN REGULADOR DE VOLTAJE PARA UN GENERADOR SÍNCRONO USANDO LA TEORÍA DE RESONANCIA ADAPTABLE

T E S I S

QUE PARA OBTENER EL GRADO DE:

Maestro en Ciencias
con especialidad en
Ingeniería Eléctrica

PRESENTA

JACOB EFRAÍN DÍAZ LAVARIEGA

MÉXICO, D. F.

JUNIO 2001

CGPI-14

INSTITUTO POLITECNICO NACIONAL

COORDINACION GENERAL DE POSGRADO E INVESTIGACION

ACTA DE REVISION DE TESIS

En la Ciudad de México, D.F. siendo las 17:30 horas del día 5 del mes de abril del 2001 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:

"DESARROLLO DE UN REGULADOR DE VOLTAJE PARA UN GENERADOR SINCRONO UTILIZANDO LA TEORIA DE RESONANCIA ADAPTABLE"

Presentada por el alumno:

DIAZ	LAVARIEGA	JACOB EFRAIN
Apellido paterno	materno	nombre(s)

Con registro: 9 7 0 7 5 6

aspirante al grado de:

MAESTRO EN CIENCIAS

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

LA COMISION REVISORA

Director de tesis

DR. DAVID ROMERO ROMERO

DR. DANIEL OLGUIN SALINAS

DR. RAUL CORTES MATEOS

DR. JAIME ROBLES GARCIA

M. en C. DOMITILO LIBREROS

EL PRESIDENTE DEL COLEGIO

DR. DAVID ROMERO ROMERO

A mis queridos padres:

Efraín Díaz Heraz†

Celia Ursula Lavariega León

Por enseñarme el valor para la vida y la fidelidad a los justos principios…

A mis hermanas:

Mayra Victoria

Ascención Nunila

Clara Aleyda

…las amo

A toda la Familia Lavariega

…por el carácter que me transmitieron y mostrarme que la unión hace la fuerza

AGRADECIMIENTOS

Agradezco al Dr. David Romero Romero que con sus valiosos consejos pude llevar con éxito esta investigación; a todos los profesores de la Sección Eléctrica que mediante su enseñanza me han permitido adquirir un método de trabajo diligente y el poder de la autoenseñanza, y que me enseñaron que todo se logra con empeño y constancia.

Agradezco el apoyo económico que me proporcionó el Consejo Nacional de Ciencia y Tecnología y el Instituto Politécnico Nacional durante mis estudios, y que me permitieron llevar a cabo este trabajo.

Agradezco a toda la Familia Martínez Díaz por su apoyo moral brindado, ya que me abrieron las puertas de su hogar y me dieron su confianza en momentos de incertidumbre, sin pedir absolutamente nada a cambio. A la señora Lilia Díaz por haber creído en mí.

Agradezco a la Sra. Alba Díaz por su apoyo económico y moral incondicional, por recibirme con todo cariño al visitar su casa; le agradezco por no dudar nunca que tendría éxito en mis estudios.

Agradezco a mi amigo Jaime García García por los momentos tan amenos que pasamos, y que hicieron menos tediosas las sesiones nocturnas de estudio. A mi amigo Guillermo Cacho López por su amistad, y por sus conocimientos brindados para el desarrollo de programa de flujos de potencia.

Agradezco a la Srita. Miriam Martínez Ruiz por su apoyo moral, por su apoyo en la escritura de la tesis, y por la impresora, ya que de lo contrario hubiera sido menos que imposible.

RESUMEN

En este trabajo se desarrolla una estructura neuro-difusa, y se aplica a la regulación de voltaje de un generador síncrono conectado a un bus infinito. El diseño de esta estructura se basa en la Teoría de Resonancia Adaptable; esta teoría es el resultado de investigaciones enfocadas a sistemas competitivos que pudieran funcionar en ambientes, en donde se pueden recibir eventos inesperados y que sea necesario codificarlos, sin necesidad de perder la información ya codificada. La Teoría de Resonancia Adaptable surgió como una respuesta a la necesidad de tener sistemas inteligentes, capaces de aprender independientemente y sin perder estabilidad durante el aprendizaje.

La aplicación de la estructura neuro-difusa diseñada, como regulador de voltaje de un generador síncrono, es una aplicación específica y requirió de un entrenamiento con algunas variantes con respecto a las aplicaciones propuestas originalmente para esta estructura. Una vez completado el entrenamiento de la estructura, se obtienen todos los elementos necesarios de un control difuso tradicional, con la sintonización correcta de las funciones de membresía lograda mediante algoritmos internos de ajuste.

El desempeño del sistema neuro-difuso en esta aplicación, es satisfactorio. Permite al generador síncrono recuperar su estado normal de funcionamiento con menos oscilaciones al compararse con un sistema de regulación ST1, amplia sus márgenes de tiempos de liberación de falla, logrando una mayor robustez del sistema de generación.

ABSTRACT

A neuro-fuzzy structure is developed in this thesis, and it is applied to a synchronous generator voltage regulator connected to an infinite bus. Its design is based on Adaptive Resonance Theory which was obtained from research about competitive systems that could work in enviroments where there are unexpected events, and it is necesary to code without lossing the previous information. Adaptive Resonance Theory is a response to the need of intelligent systems that are able to learn independently, and without losing stability during learning.

The aplication of the designed structure, as a synchronuos generator voltage regulator, requires a specific training that differs from those developed originally for this structure. Once concluded its training, all the elements that conform a traditional fuzzy control are provided, even the correct tunning of the membership functions is obtained from inner adjustment algorithms.

Neuro-fuzzy system performance for this application is satisfactory. The synchronous machine can return to its normal state faster, and with fewer oscillations compared with a ST1 voltage regulator. Moreover, the fault liberation time is increased. As a result, it provides to the generation system with a more robust voltage regulation system.

CONTENIDO

LISTA DE FIGURAS

APÉNDICE A

LISTA DE TABLAS

CAPÍTULO UNO

GENERALIDADES

1. 1 Introducción

El uso unificado de redes neuronales artificiales y Teoría Difusa en el diseño de controles, como el desarrollado en este trabajo, da como resultado la formación de arquitecturas eficientes en su desempeño para el cual se diseñan.

El éxito de éstas arquitecturas neurodifusas, se fundamenta en que incorporan en sus diseños, algunas de las características que los seres humanos utilizamos durante el proceso de aprendizaje, como: la atención, la organización, la evaluación, entre otras.

Artefactos como cámaras fotográficas y lavadoras, ya incorporan a sus controles la Teoría Difusa. Los sistemas telefónicos utilizan filtros basados en redes neuronales, y aplicaciones más complejas, como el control de dirección de un auto durante la etapa de estacionamiento, y predicción de series caóticas de tiempo[1] conjuntan ambas teorías para lograr resultados en aplicaciones más complejas.

El sistema de regulación desarrollado en este trabajo, se aplica en la regulación de voltaje de excitación en un generador síncrono conectado a un bus infinito durante condiciones de falla. La regulación de voltaje en el generador síncrono es de vital importancia, ya que de ello depende en gran parte la estabilidad de un

[1] Ref. 5

sistema eléctrico de potencia. El sistema de regulación desarrollado, efectúa una regulación de voltaje de excitación efectiva durante las pruebas en condiciones de falla. La falla se aplica en el bus y se mantiene durante un tiempo determinado para observar si la máquina conectada al bus se mantiene estable durante la falla. Se lleva a cabo una comparación de índices de desempeño, entre un sistema de regulación ST1 y la estructura desarrollada en este trabajo. En el total de los casos se obtuvo mejor comportamiento de la estructura desarrollada.

1. 2 Objetivo.

El objetivo de esta tesis es desarrollar una arquitectura neuro-difusa basada en la Teoría de Resonancia Adaptable para la regulación de voltaje de un generador síncrono.

1. 3 Justificación de la Tesis.

Por la necesidad que actualmente existe de proporcionar seguridad y confiabilidad en el funcionamiento de un sistema eléctrico de potencia, se presenta este tipo de control. Los cuales basados en redes neuronales autoorganizables y teoría difusa, ofrecen una mejor respuesta al sistema ante la presencia de fallas severas.

1. 4 Estado del Arte.

El mejoramiento de la estabilidad de los sistemas eléctricos de potencia es uno de los aspectos de mayor importancia en el control de los sistemas eléctricos de potencia, y "...es bien sabido que el control a través del lazo de excitación del generador síncrono es una de las formas más efectivas para estabilizar sistemas eléctricos de potencia..."[2]. El sistema de regulación desarrollado por investigadores de la Universidad de Singapur y de la República de China, fue desarrollado con técnicas de inteligencia artificial en 1996.

Este sistema esta basado en control robusto y es aplicado en un modelo de máquina síncrona-bus infinito linealizado. Este sistema es comparado con un sistema de control de retroalimentación directa linealizada y también con un control basado en el estabilizador de potencia. El sistema fue probado con la aplicación de una falla sobre la línea que conecta la máquina con el bus, liberando

[2] Ref. 18, col. 1, párr. 1

la falla en 0.1 seg. Durante las pruebas el sistema diseñado demostró un índice de comportamiento superior al de los sistemas con los que fue comparado[3].

En el departamento de Ingeniería Electrónica de la Universidad de Hong Kong, los investigadores desarrollaron un controlador neuro difuso en 1995[4]. El controlador diseñado, es aplicado a la regulación de convertidores de cd-cd. Estos dispositivos son difíciles de controlar debido a su no linealidad y a la incertidumbre en sus parámetros. El controlador utiliza el error del voltaje y el cambio del error del voltaje como entradas, y su salida es el switch PWM para el control del convertidor.

Este controlador esta compuesto por los elementos tradicionales de un controlador difuso: Capa de entrada, capa de nodos difusificadores, capa de base de reglas, y la capa de nodos dedifusificadores. El aprendizaje de este sistema neuro difuso, es del tipo supervisado y utiliza el algoritmo de propagación del error para corregir sus parámetros. Las simulaciones se realizaron a un modelo linealizado del convertidor, y se comparó con un control PI. El sistema diseñado, logró una mejoría substancial del comportamiento del convertidor durante las pruebas, superando al control PI.

El Laboratorio politécnico Interdepartamental de mecatrónica de Torino, Italia ha diseñado un sistema híbrido donde se combina la tecnología digital y los sistemas neuro difusos en 1998[5]. El sistema consiste en la utilización de un TMS320C31 Texas DSP, el cual puede soportar 256 canales A/D y D/A, codificadores PWM, E/S digitales y expansión de memoria. El sistema Neuro difuso diseñado, se encarga de controlar la planta, mientras que el DSP permite lograr un aprendizaje en línea y sistemas de control adaptable, filtrar señales, y administrar el sistema en su totalidad. El sistema creado, ya es comercializado en el mercado pudiendo, mediante su diseño, aceptar el acople de sensores, actuadores y otros dispositivos. El campo de aplicación de este sistema, es en vehículos todo terreno teledirigidos, se busca que estos vehiculos sean ligeros, simples y con un costo muy bajo.

---

[3] Ref. 18
[4] Ref. 19
[5] Ref. 20

Viajando en la autopista cerca de Pittsburgh a una confortable velocidad de 55mph, el hombre al volante parece relajado. En realidad debe estarlo, durante las ya recorridas 90 millas, no ha tenido que tocar en absoluto el volante, el freno o el acelerador. El conductor real es un sistema robótico, que conjunta las señales de videocámaras, sonares y rayos láser con la experiencia aprendida a partir de entrenamientos de cómo conducir un automóvil (Pomerleau, 1993)[6].

Durante 1991, el Departamento de sistemas Neuronales y Cognitivos de la Universidad de Boston fue designado para el desarrollo de redes neuronales capaces de clasificar patrones análógicos. La oficina de la Fuerza Aérea Americana a través de su oficina para la Investigación científica, El departamento de Investigación del Ejército, Petróleo Británico y la Fundación Nacional para la Ciencia proporcionan los recursos suficientes para el desarrollo de este tipo de redes; como resultado de las investigaciones, se obtiene la red basada en la Teoría de resonancia Adaptable para clasificar patrones analógicos llamada FUZZY ART. Además se crean otras redes como el ART1 y el sistema ARTMAP[7] que clasifican patrones binarios y binarios/analógico, respectivamente. Estas arquitecturas son el resultado de investigaciones iniciadas en la década de 1970, con los investigadores Stephen Grossberg, Christoph vonder Malsburg, Leon Cooper, Shun-ichi Amari y Teuvo Kohonen. Las redes creadas, han sido utilizados por otros científicos para aplicaciones en el área de control[8], y la medicina[9]. Se han creado variantes de estas estructuras[10], han encontrado aplicación en el área de investigación de procesamiento de imágenes[11]. El campo de los sistemas de redes neuronales se vió revolucionado con el desarrollo de la Teoría de resonanca Adaptable.

Estos son algunos ejemplos de Inteligencia Artificial que existen hoy en día, de los cuales este trabajo forma una pequeñisima parte.

---

[6] Ref. 8
[7] Ref. 3 y 4
[8] Ref. 5
[9] Ref. 1
[10] Ref. 21
[11] Ref. 24

1. 5 Aportaciones de la tesis

- Se presenta el desarrollo de una estructura que contiene los elementos básicos y lleva a cabo las funciones de un controlador difuso tradicional que puede ser aplicable a plantas con un comportamiento dinámico.
- Se presenta una propuesta de entrenamiento para este tipo de aplicación que puede ser aplicado a otro tipo de plantas.
- Se presenta la aplicación de este controlador como un regulador de voltaje de campo en el modelo de una máquina síncrona.

1. 6 Estructura de la tesis

En el **capítulo dos** se presentan los fundamentos de la Teoría de Resonancia Adaptable (ART, por sus siglas en inglés Adaptive Resonace Theory), que es la base para el desarrollo de la estructura neuro-difusa desarrollada en esta tesis. Con la presentación de este material, se pretende bosquejar las bases de los mecanismos con los que la estructura lleva a cabo el reconocimiento de patrones[12].

En el **capítulo tres** se presenta la red adaptable de control difuso basada en la ART (FALCON-ART, por sus siglas en inglés, "ART-Based Fuzzy Adaptative Control Network"[13]), se explica cómo se efectúa la clasificación de patrones y los diferentes sentidos de la propagación de las señales durante su etapa de entrenamiento, así como también durante su etapa de producción, y se muestran las funciones realizadas por los diferentes mecanismos inherentes a la red.

En el **capítulo cuatro** se presenta el desarrollo del entrenamiento de la FALCON-ART, y los resultados arrojados por esta estructura en su etapa de entrenamiento y su etapa de producción. También se presentan los resultados obtenidos después de acoplar la FALCON-ART al modelo de máquina síncrona[14], y que funciona en sustitución del regulador de voltaje de la máquina síncrona. Estos resultados son pruebas efectuadas al modelo de máquina síncrona en diferentes condiciones iniciales de operación durante una falla aplicada en el bus en rangos de funcionamiento estipulados con anterioridad.

---

[12] Para mayor información, el lector puede consultar Ref. 1,3 y 6
[13] Ref. 5
[14] Ver Apéndice C

En el **capítulo cinco** se presentan algunas conclusiones de este trabajo, así como también algunas sugerencias que pueden ser útiles para trabajos posteriores que pueden ser desprendidos a partir del presente trabajo.

Además se incluyen los apéndices siguientes:

El **Apéndice A** incluye la presentación del software diseñado para la observación del sistema desarrollado en este trabajo durante las pruebas efectuadas.

El **Apéndice B** incluye el modelo de la planta a controlar, en este caso se presenta el modelo de una máquina síncrona. En el modelo se aplica el controlador neurodifuso desarrollado en este trabajo.

El **Apéndice C** incluye el código del programa, que integra el modelo de la planta a controlar, y el regulador neurodifuso. En el Apéndice se detallan cada uno de los pasos que se realizan durante la ejecución del programa, así como también se definen las variables que se utilizaron.

El **Apéndice D** incluye las condiciones iniciales que se generaron, ara la simulación de la máquina síncrona, estas condiciones iniciales se utilizaron en el entrenamiento del sistema neurodifuso, así como también en el periodo de producción.

El **Apéndice E** incluye el programa que se utilizó para la generación de las condiciones iniciales del Apéndice D, se introducen también algunos de los fundamentos que se necesitan conocer para desarrollar un programa de este tipo.

El **Apéndice F** presenta los fundamentos de los sistemas de redes neuronales artificiales, para una mejor comprensión presentado en este trabajo.

El **Apéndice G** presenta los fundamentos de la teoría difusa, los elementos que conforman un sistema de control difuso, y las ecuaciones básicas de desarrollo para un control de este tipo.

CAPÍTULO DOS

FUNDAMENTOS DE LA TEORÍA DE RESONANCIA ADAPTABLE

2. 1 Introducción.

Al diseñar máquinas inteligentes que sean capaces de aprender independientemente y de llevar a cabo un buen desempeño en trabajos complejos, los diseñadores enfrentan las siguientes preguntas que son básicas dentro de la ciencia computacional:

- "¿Por qué ponemos atención?
- ¿Por qué aprendemos a hacer suposiciones de nuestro alrededor?
- ¿Cómo enfrentamos con éxito situaciones inesperadas?
- ¿Cómo superamos dichos eventos aún sin tener a nadie que nos indique que hacer?
- ¿Cómo logramos combinar factores útiles diferentes para salvar una situación relevante determinada?
- ¿Cómo reconocemos hechos familiares, aún cuando hemos acumulado muchos otros tipos de información?
- ¿Cómo relacionamos conocimiento del mundo exterior con nuestras necesidades, de forma que podamos tomar decisiones que permitan satisfacerlas?

- Y finalmente, ¿Qué tienen todas estas propiedades en común?"[9]

Entre los modelos de redes neuronales artificiales, existen varios que han incorporado a sus arquitecturas, características que logran cubrir algunas de estas preguntas, pero aún mostrando deficiencias en sus desempeños (ver Tabla 1). La Teoría de Resonancia Adaptable (ART, por sus siglas en inglés) planteada por Stephen Grossberg[10], es la base fundamental para muchas de las estructuras creadas por este investigador, que dan una respuestas a las interrogantes anteriores, y forman en sí un avance muy importante dentro del campo de la inteligencia artificial.

2. 2 Modelos competitivos de aprendizaje[11].

Los modelos ART provienen del análisis de modelos competitivos de aprendizaje más simples, que fueron desarrollados en los inicios de los 70´s por Christop Von der Malsburg y Stephen Grossberg. Estos modelos consisten generalmente en dos capas F1 y F2, cuyo funcionamiento se describe en forma breve: Al presentarse un vector de entrada $\boldsymbol{I}$ a la capa F1, se normaliza y se convierte en un vector $\boldsymbol{X}$. Este último vector se propaga a la capa F2, donde cada nodo de dicha capa evalúa el vector $\boldsymbol{X}$, y se obtiene un nodo "ganador-toma-todo", eligiéndose este como el nodo que codificará la entrada $\boldsymbol{I}$.

Se "...ha probado matemáticamente..."[12] que estos modelos alcanzan la estabilidad en su aprendizaje cuando no reciben la suficiente información. Sin embargo, "...existen varios ejemplos..."[13] en los que se demuestra el hecho en que estos modelos pueden llegar a inestabilizarse durante el aprendizaje "...en otras palabras, la adaptabilidad o plasticidad de la red, contribuye a borrar el aprendizaje adquirido anteriormente..."[14].

Este problema de inestabilidad se presenta no únicamente en los modelos competitivos, ya que es un problema general en el modelado de redes. Por lo

[9] Ref. 1, col. 1, pág. 77
[10] Para mayor información, el lector puede consultar Ref. 1,2,3 y 4.
[11] Ref. 1
[12] Ref. 1, col. 3, pág. 78
[13] Ver nota 12

anterior, existe la posibilidad de que estos modelos *no* puedan ser utilizados en sistemas autónomos en donde puedan presentarse eventos inesperados de aprendizaje. " La Teoria de Resonancia Adaptable fue desarrollada en 1976, con la finalidad de incluir ciertos modelos competitivos de aprendizaje dentro de las estructuras con auto-control..."[15]

Tabla 2.1 Ventajas de las estructuras ART & otras estructuras

Arquitecturas ART	**Alternativas de aprendizaje**
Aprendizaje en tiempo real	Aprendizaje fuera de linea
Ambiente no estacionario	Ambiente estacionario
Auto-organizables (No supervisado)	Un guía proporciona la respuesta correcta (supervisado)
Memoria auto-estable en respuesta a muchas entradas arbitrarias.	Caos en memoria debido a muchas entradas arbitrarias
Uso efectivo de la capacidad de memoria	Utilización parcial de la capacidad de memoria
Estado plástico ante eventos inesperados	Control externo del estado plástico para evitar un colapso en la estructura
Aprenden a hacer suposiciones dentro de la estructura arriba-abajo	Se impone en forma externa la magnitud del costo de la suposición
Una atención activa regula el aprendizaje	Aprendizaje pasivo
Aprendizaje rápido y lento	Aprendizaje lento o se corre el riesgo de un colapso
Aprenden enfasando aproximación-igualación	Aprenden tomando en cuenta la desigualdad de las entradas
Rápida adaptación para buscar la mejor igualación.	Búsqueda por árboles.
Acceso directo a eventos familiares	El tiempo de reconocimiento aumenta en respuesta a entradas más complejas

[14] Ver nota 12

[15] Ref. 1, col. 1,pág. 79.

2. 3 Dilema de Plasticidad-Estabilidad.

Este dilema es "...enfrentado por todos los sistemas inteligentes capaces de adaptarse autónomamente en tiempo real a cambios inesperados de nuestro entorno."[16], y plantea las preguntas siguientes:

- "¿CÓMO PUEDE DISEÑARSE UN SISTEMA DE APRENDIZAJE ARTIFICIAL QUE PUEDA MANTENERSE PLÁSTICO (O ADAPTABLE), EN RESPUESTA A EVENTOS SIGNIFICATIVOS, Y AÚN PERMANECER ESTABLE A SITUACIONES IRRELEVANTES?
- ¿CÓMO PUEDE SABER EL SISTEMA CUANDO CAMBIAR ENTRE SU MODO PLÁSTICO SIN CAOS Y SU MODO ESTABLE SIN RIGIDEZ?,
- ¿CÓMO PUEDE MANTENER EL SISTEMA EL CONOCIMIENTO APRENDIDO Y CONTINUAR APRENDIENDO COSAS NUEVAS?, Y FINALMENTE
- ¿QUÉ PREVIENE QUE EL NUEVO APRENDIZAJE NO BORRE EL CONOCIMIENTO PREVIO?"[17]

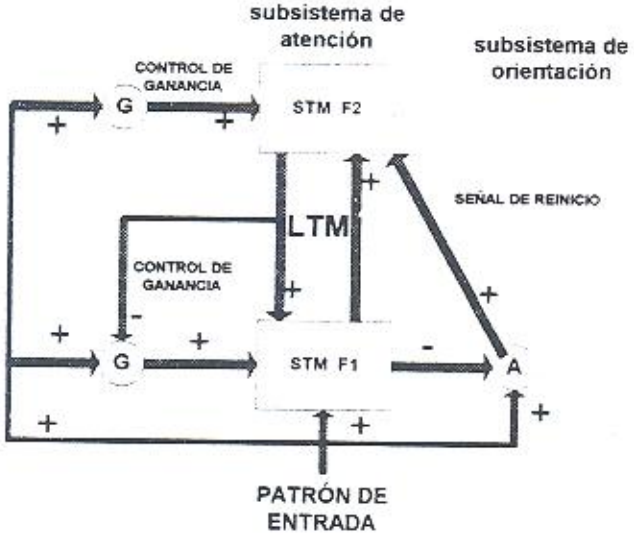


Figura 2. 1 Estructura general de una red basada en la ART.

El autor de la teoría ART, ha desarrollado varias estructuras que dan respuesta a estas preguntas, y que ya son aplicadas en el campo de la inteligencia artificial. Estas estructuras son las siguientes: ART1[18], ART difuso[19] y ARTMAP[20]; en la tabla 2.1 se mencionan las ventajas que las estructuras ART pueden

[16] Ref. 1, col. 2, pág. 77
[17] Ref. 1, col. 3, pág. 77
[18] Ver Ref. 1
[19] Ver Ref. 2
[20] Ver Ref. 3 y 4

presentar en comparación con otras redes como "...autoasociadores, máquina de Boltzmann y Retropropagación..."[21].

2. 4 Estructura general de las redes ART.

Estas estructuras están conformadas por dos elementos principalmente: Las capas F1 y F2 (subsistema de atención), y un sistema de reinicio (subsistema de orientación), que pueden ser representados como en la Fig. 2.1:

Donde :

- LTM(por sus siglas en inglés, *Long Time Memory*) = Memoria de largo plazo
- STM(por sus siglas en inglés, *Short Time Memory*) = Memoria de corto plazo
- A es la señal de reiniciar o suspender la búsqueda en la capa F2, dependiendo de la respuesta de la red (llamado subsistema de orientación).
- F1 es la capa donde se efectúa la comparación de tramas durante la presentación de un patrón en la etapa de aprendizaje.
- F2 es la capa donde se encuentran los nodos que van codificando los patrones de entrada.
- El control de ganancia G, permite que ambas capas perciban mayor o menor diferencia entre los patrones codificados y los patrones que se van presentando en la entrada de la red.

Las tramas que se generan en ambas capas durante la presentación de cada patrón de entrada conforman la STM, y los pesos adaptables que se encuentran en los nodos de las capas F1 y F2, representan la LTM. Una vez que se presenta un patrón de entrada, se inicia un ciclo de búsqueda para poder determinar la clase que en donde se puede codificar el patrón correspondiente a esa entrada, esto se realiza al menos una vez por cada patrón de entrada.

[21] Ref. 1, col. 3, pág. 80

2. 5 Funcionamiento de una estructura ART.

Dentro del conjunto de arquitecturas ART, el modelo ART1 es frecuentemente utilizado para enseñar los fundamentos de la teoría en la que se basan (ART); la razón de dicha utilidad es que en este modelo se plasma en su totalidad la Teoría de Resonancia Adaptable. A continuación describe el funcionamiento de este modelo:

Inicialmente en la capa F1, se presenta un patrón de entrada ***I***. A cada uno de los nodos de F1, le corresponde un componente de este vector. Igualmente, el vector ***I*** se envía al subsistema de orientación, nodo A, y al sistema de ganancia, nodo G. Como condición inicial, la capa F1, generará una trama ***X1***, idéntica al vector de entrada ***I***, que será utilizada para nulificar el efecto excitatorio de este vector en el subsistema de atención, y evitar que este sistema se active. En este momento, la capa F1 está recibiendo dos señales de entrada, la del vector de entrada ***I*** y la del control de ganancia G. Esta secuencia se muestra en la figura 2.2.

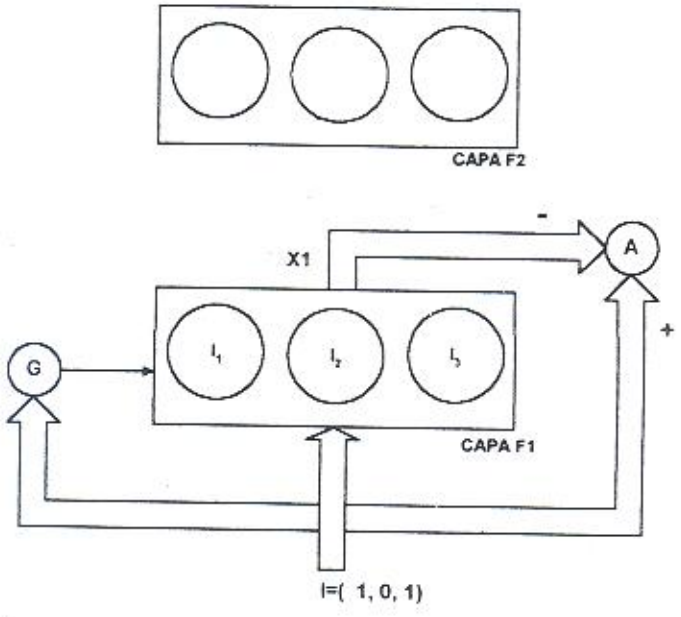


Figura 2. 2 Hacia el intento de reconocer una trama.

Tanto la capa F1 como la capa F2, necesitan de manera imprescindible, dos señales para poder activarse. Este requisito se denomina regla de los dos tercios, y se explica más adelante.

La trama generada ***X1***, se transmite a todos y cada uno de los nodos de la capa F2. En esta capa, la trama ***X1*** es evaluada y se determina el nodo que contiene

la trama que más se asemeja o que es igual al vector de entrada ***I***. En este momento la capa F2, recibe dos estímulos, la de la trama ***X1*** y la del sistema de ganancia, como se muestra en la figura 2.3.

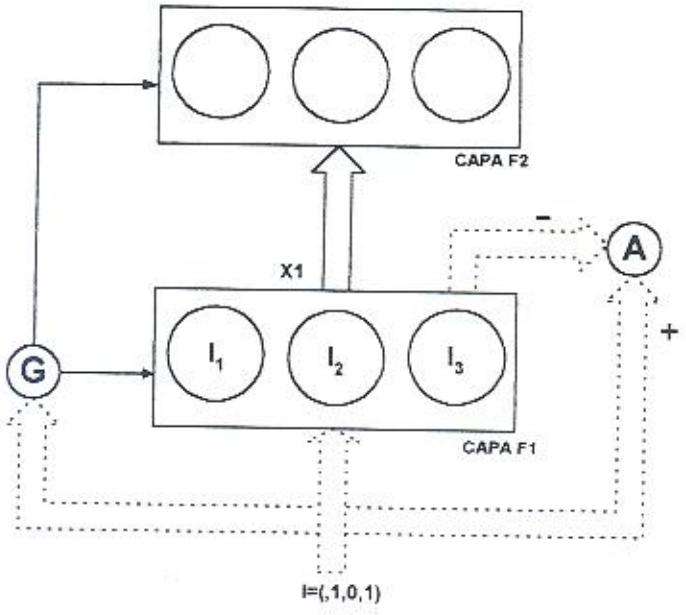


Figura 2. 3 Envío de trama X1 hacia la capa F2.

Una vez seleccionado el nodo ganador (nodo asurado en la figura 2.4), se genera ahora una trama de salida en la capa F2, esta trama de salida ***X2***, se envía a la capa F1 y se inhibe al mismo tiempo el sistema de ganancia ***G***. En la capa F1 se genera una trama de salida ***V***, que se compara con el vector de entrada ***I*** en el subsistema de orientación. En este momento, la capa F1 recibe 2 señales de excitación, la trama ***X2*** y el vector de entrada ***I***, ver figura 2.4

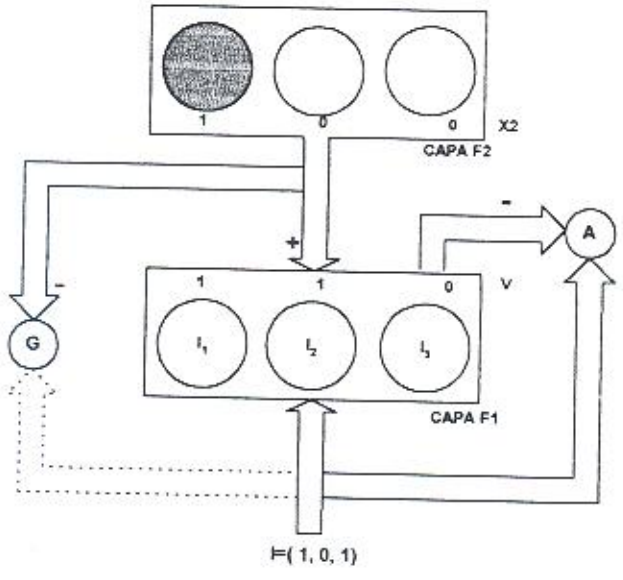


Figura 2. 4 Se lleva a cabo el reconocimiento de la trama de entrada.

Si la evaluación realizada en el nodo ***A***, indica un error en la selección del nodo ganador en la capa F2, el nodo ***A*** enviará una señal a la capa F2 para inhibir el nodo ganador seleccionado en esa iteración y lo mantendrá así mientras la capa F2, no proporcione la predicción correcta para el nodo de entrada ***I*** como se muestra en la figura 2.5.

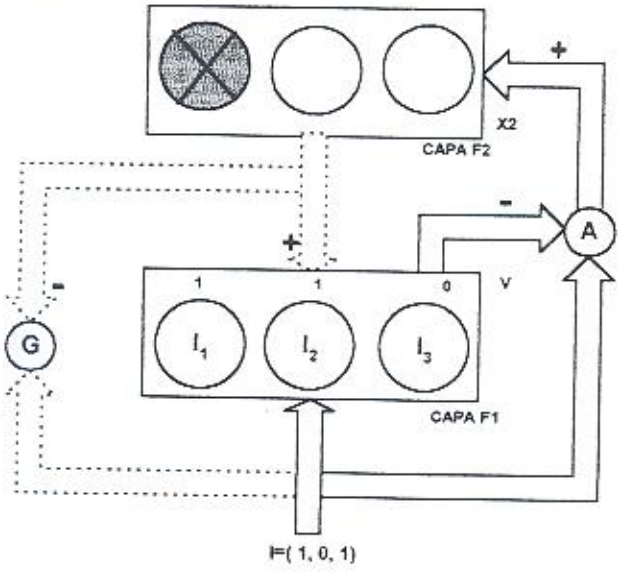


Figura 2. 5 Se inhibe la actividad del nodo seleccionado con la clase incorrecta.

Una vez inhibido el nodo ganador anterior, se comienza nuevamente la búsqueda de un nodo ganador, el esquema del sistema ART será el siguiente:

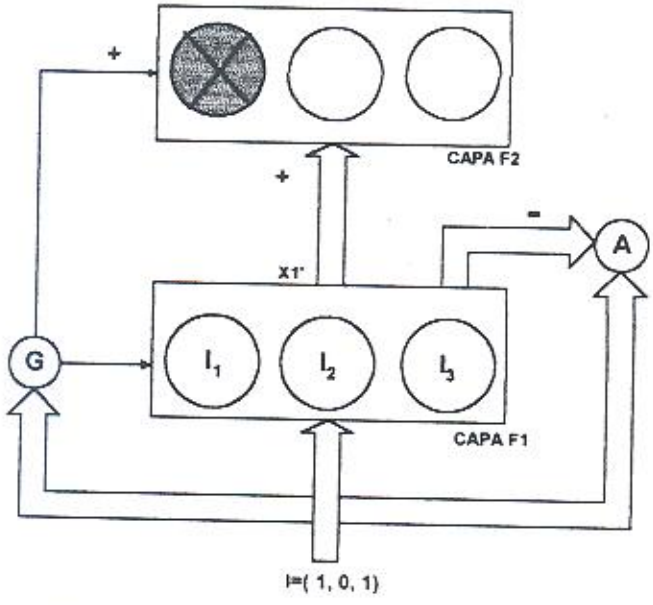


Figura 2. 6 Se reinicia la búsqueda del nodo correcto.

La búsqueda del nodo continuará hasta que la coincidencia del vector de entrada ***I***, con la clase codificada en el nodo ganador de F2, cumpla con las

condiciones de los sistemas de atención y de orientación que se encuentran vigilando las predicciones del sistema ART.

2. 6 Regla de los 2/3 y Control de Ganancia.

Estos dos mecanismos forman una parte importante dentro de las estructuras ART, y su objetivo es evitar que tanto la capa F2, como F1, no realicen un funcionamiento incorrecto debido a señales provenientes de otras direcciones.

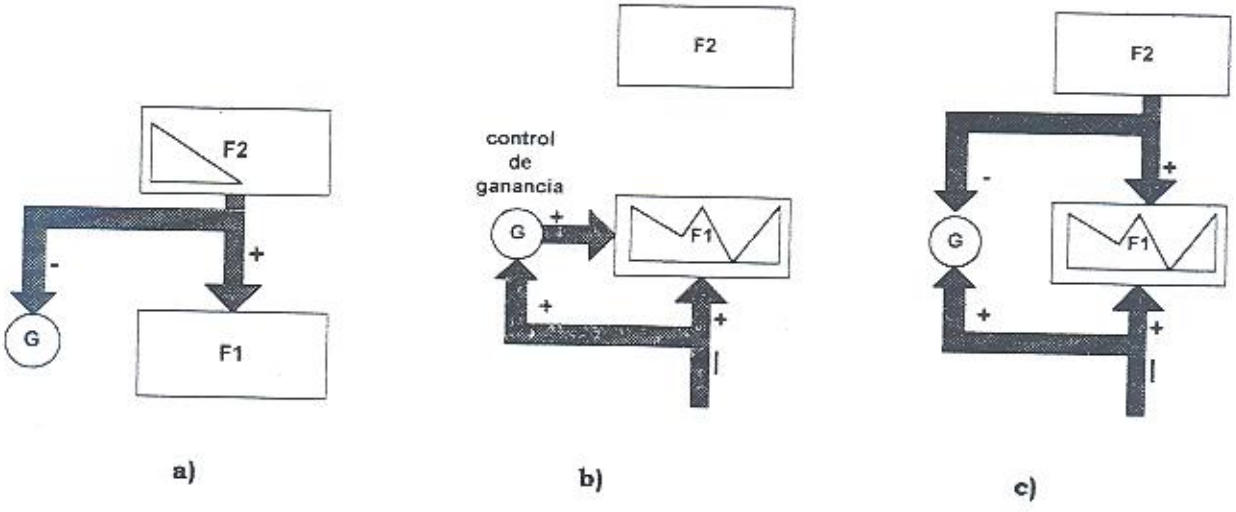


Figura 2. 7 Secuencia de flujo de señales

Si F2 enviara una señal a F1, sin que esta última hubiera recibido ninguna excitación inferior preliminar, "...F2 estaría indicando lo que espera que sea la señal de entrada, antes que la trama inferior llegue realmente a F1"[22].

El control de ganancia **G**, ver la Fig. 2.7 a), que es un sistema que activa todos los nodos de la capa F1, inhibe su actividad cuando recibe una señal de F2. Si se aplica la regla de los 2/3, es decir, que se reciban al menos 2 señales de activación sobre la capa F1, "... la inhibición de **G** significa que la señal descendente, de F2 no puede por si misma, desencadenar una salida procedente de F1"[23].

Es necesario, por lo tanto, que F1 reciba una señal de la parte inferior, para que pueda iniciarse un ciclo de búsqueda. Si el sistema **G** es estimulado por una señal inferior (Fig. 2.7 b), su señal es enviada hacia cada uno de los nodos de F1. De acuerdo a la regla de los 2/3, los nodos de F1 estarían recibiendo entonces, 2

[22] Ref. 7, párr. 3, pág. 313.

señales de excitación, la inferior y la proveniente de **G**. "G y la regla de los 2/3 se combinan para permitir que la capa F1 distinga entre una señal de anticipación procedente de la capa superior, y una señal de entrada procedente de la parte inferior..."(Fig.2.7c)[24]

El procedimiento anterior es seguido, con ligeras variaciones, por todas las estructuras ART. En ellas existirán sistemas de vigilancia que impedirán acciones incorrectas en su estructura. Para la estructura ART difusa por ejemplo, existe el parámetro de vigilancia que define estrictamente a que clase, ya codificada, pertenece el patrón actual de entrada. Para el ARTMAP difuso, existirán dos ART difusas que contendrán cada una de ellas su parámetro de vigilancia, existirá también un parámetro que controle un campo de mapeo de la red y que modificará, cada vez que exista una falla de predicción por parte de la estructura, al parámetro de vigilancia de la ART difusa que está codificando los patrones de entrada. Finalmente, la estructura FALCON-ART que se presenta en este trabajo, contiene el mismo tipo de parámetros del ARTMAP difuso; sin embargo, su sistema de mapeo se gobierna por una regla diferente de aprendizaje.

[23] Ref. 7, párr. 1, pág. 314
[24] Ref. 7, párr. 3, pág. 314

CAPITULO TRES

RED DE CONTROL DE APRENDIZAJE ADAPTABLE DIFUSO BASADO EN LA TEORÍA DE RESONANCIA ADAPTABLE

3. 1 Introducción.

La Red de Control de Aprendizaje Adaptable Difuso basado en la Teoría de Resonancia Adaptable ó "ART-based fuzzy adaptive learning control network"[26] (FALCON-ART, por sus siglas en inglés) es una arquitectura de conexiones múltiples y realiza la función de un controlador tradicional de lógica difusa. Su aprendizaje estructural y de parámetros puede llevarse a cabo en línea, el primero se logra mediante la *Teoría de Resonancia Adaptable* y el segundo se lleva a cabo mediante el aprendizaje de *Retropropagación del error*. La red lleva a cabo una partición de los espacios de entrada y de salida, utilizando hipercajas difusas irregulares que van de acuerdo con la distribución de los datos de entrenamiento, ver Fig. 3.1.

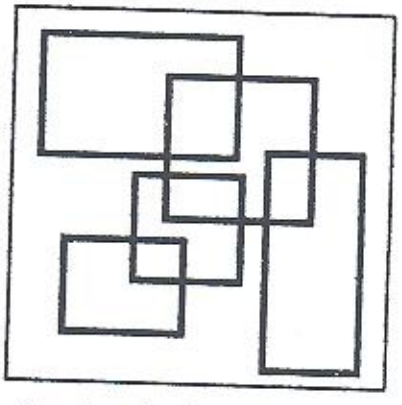


Figura 3. 1 Partición flexible difusa en hipercajas

Inicialmente, dentro de la red no existen funciones de membresía, divisiones difusas ni reglas lógicas difusas, estas se van creando y comienzan a crecer como los patrones de entrenamiento se van presentando. De esta forma el usuario no necesita

dar ningún tipo de conocimiento o información inicial dentro de la red. La estructura FALCON-ART particiona el espacio de entrada y salida, sintoniza las funciones de membresía y encuentra las reglas lógicas difusas durante su aprendizaje en línea.

3.2 Normalización y Codificación del complemento de los patrones de entrada[27].

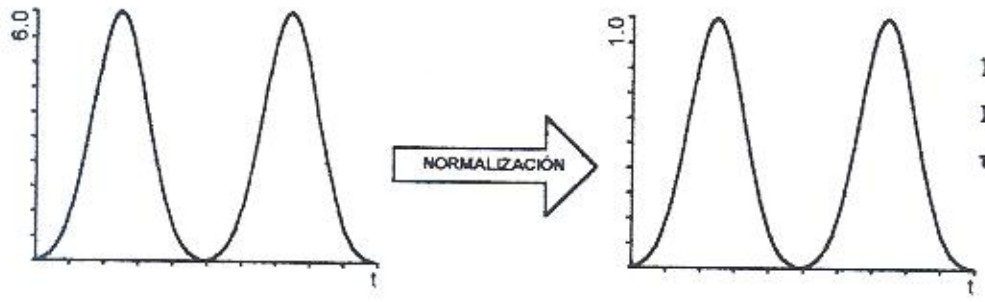


Figura 3. 2 Normalización de una señal

La Fig. 3.2 esquematiza la transformación llamada *Normalización*, que debe realizarse a una señal que se desea codificar en el sistema FALCON-ART. La codificación del complemento es un proceso que reescala un vector n-dimensional $\boldsymbol{x}=(x_1,x_2,\ldots,x_n)$ en $\Re^n$, hacia su complemento en su 2n dimensión en $[0,1]^{2n}$, x', de tal forma que:

$$x' \equiv (\bar{x}_1, \bar{x}_1^C, \bar{x}_2, \bar{x}_2^C, \ldots, \bar{x}_n, \bar{x}_n^C) = (\bar{x}_1, 1-\bar{x}_1, \bar{x}_2, 1-\bar{x}_2, \ldots, \bar{x}_n, 1-\bar{x}_n) \qquad \textbf{3. 1}$$

Donde

$$\frac{x}{|x|} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) \qquad \textbf{3. 2}$$

y x_i^c es el complemento de x_i ; es decir, $x_1^C = 1 - \bar{x}_1$.

En la Fig. 3.2, se puede apreciar que el valor máximo de la señal normalizada ya no es 6.0 sino 1.0. Esta transformación se hace con el objetivo de evitar la proliferación de categorías, ya que dentro de los *"...sistemas difusos o neuro difusos los espacios de entrada y salida son particionados en una* ***malla****, sin embargo, conforme el número de variables entrada/salida se incrementa, el número de mallas divididas se incrementa combinatoriamente"*[28], ver Fig. 3.3.

---

[26] Ref. 5
[27] Las ecuaciones de este segmento son tomadas de la Ref. 5
[28] Ref. 5, col. 2, pág. 477

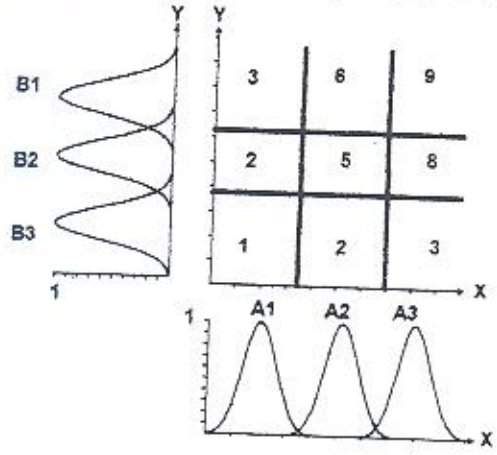


Figura 3. 3 Partición difusa tipo malla

3. 3 Estructura y Aprendizaje de la FALCON-ART[29].

Su arquitectura puede representarse como una estructura de cinco capas como se muestra en la Figura 3.4. Los nodos de la capa 1 son los nodos de entrada (*nodos lingüísticos*), y representan las variables lingüísticas de entrada. La capa 5 es la capa de salida, se tienen dos nodos lingüísticos para cada variable de salida, uno de ellos es para los datos de entrenamiento (salida deseada) la cual será proporcionada por la red durante su entrenamiento y el segundo para cuando la red ya se encuentra en su etapa de producción. Los nodos de las capas 2 y 4 son los *nodos término*, estos actúan como las funciones de membresía para representar los términos de la variable lingüística respectiva. Cada uno de los nodos de la capa 3 es un *nodo regla*, y cada uno representa una regla lógica difusa, el conjunto total de nodos regla forma lo que se llama la *base de reglas difusas*.

Los enlaces existentes entre la capa 2 y 3, y los enlaces entre la capa 3 y 4 forman, respectivamente, las precondiciones y consecuencias de los nodos regla. Los enlaces en la capa 2 y 5 están completamente conectados entre los nodos lingüísticos y sus correspondientes nodos términos. Las flechas indican la dirección normal de flujo de las señales dentro de la red cuando la red ya se encuentra en operación.

CAPA UNO: La función de los nodos lingüísticos de entrada en la capa 1, es la de transmitir la señal de entrada idénticamente como la reciben, es decir, su función de activación es:

[29] Los textos, las ecuaciones y las figuras de este segmento fueron tomados de la Ref. 5

$$f(\bar{x}_i, \bar{x}_i^c) = (x_i, \bar{x}_i^c) = (x_i, 1.0 - x_i) \qquad \textbf{3. 3}$$

$$\text{y} \qquad a[f(\bullet)] = f(\bullet)$$

Donde $a[f(\bullet)] = f(\bullet)$ representa la función de activación del nodo correspondiente, los pesos iniciales de estos nodos son $w_i^{(1)} = 1$.

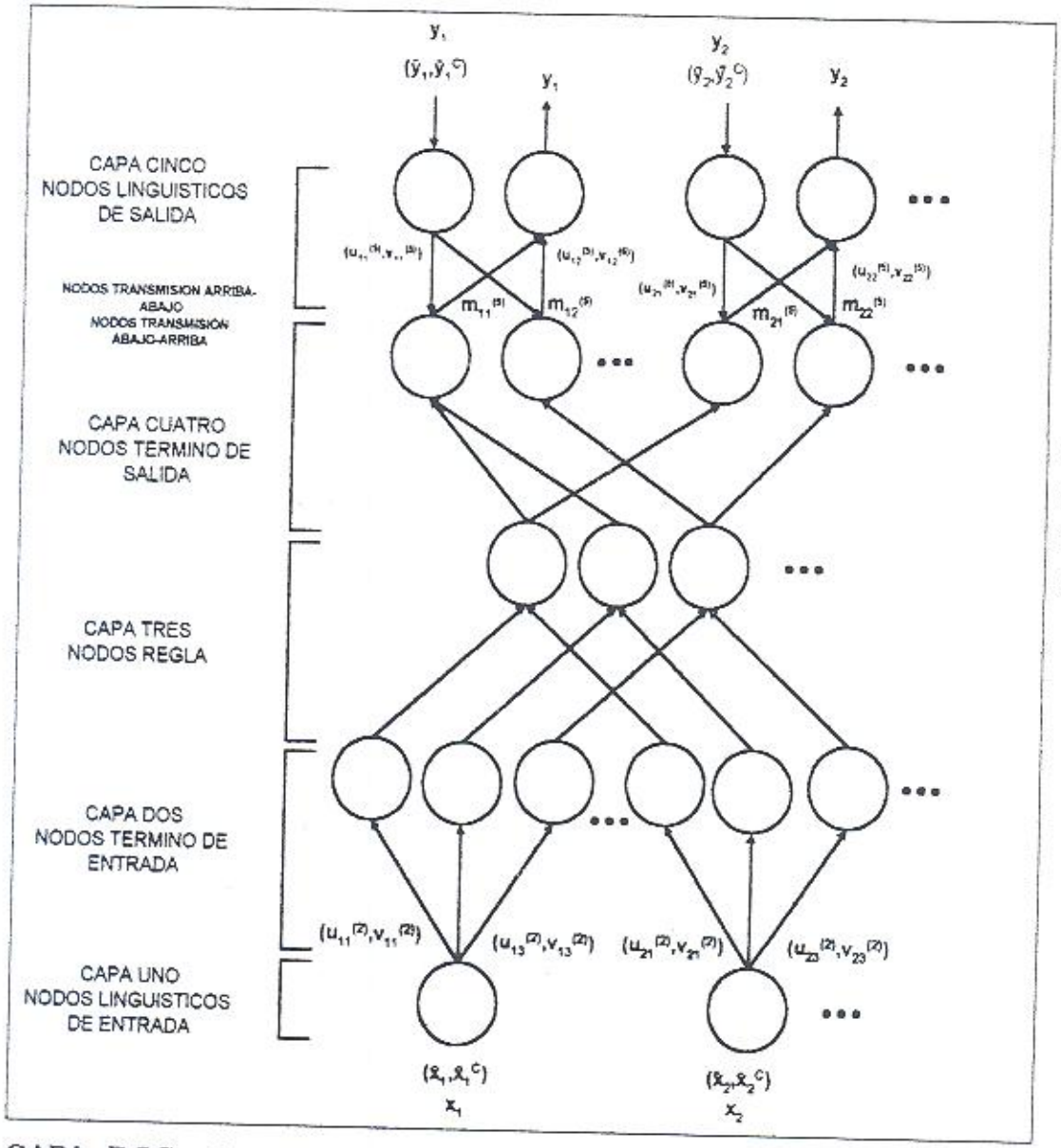


Figura 3. 4 Estructura FALCON-ART.

CAPA DOS: En esta capa se encuentran los *nodos término del espacio de entrada*, y cada uno representa un término de la variable lingüística de entrada y actúa como una función de membresía unidimensional. Para la formación de las funciones de membresía se utiliza la siguiente función:

$$f[z_{ij}^{(2)}] = \left\{1.0 - g[z_{ij}^{(2)} - v_{ij}^{(2)}, \gamma] - g[u_{ij}^{(2)} - z_{ij}^{(2)}, \gamma]\right\} \qquad \textbf{3. 4}$$

$$\text{y} \quad a[f(\bullet)] = f(\bullet)$$

Donde $u_{ij}^{(2)}$ y $v_{ij}^{(2)}$, son la pendiente izquierda y derecha, respectivamente, de la función de membresía del *j*-ésimo nodo término del *i*-ésimo nodo lingüístico, ver Fig. 3.5.

$z_{ij}^{(2)}$ es la entrada al *j*-ésimo nodo término de entrada desde el *i*-ésimo nodo lingüístico (i.e.: $z_{ij}^{(2)} = x_i$), y:

$$g(s,\gamma) = \begin{cases} 1 & \text{si } s\gamma > 1 \\ s\gamma & \text{si } 0 \le s\gamma \le 1 \\ 0 & s\gamma < 0 \end{cases} \qquad \textbf{3. 5}$$

El parámetro γ es el parámetro de sensitividad que regula la difusividad de la función de membresía. Si γ es grande se logran conjuntos difusos rígidos. Si γ es pequeño se logran conjuntos difusos más flexibles. Un conjunto de nodos, define una función de membresía n-dimensional en el espacio de entrada. Por esta razón es que cada nodo lingüístico tenga el mismo número de nodos término. Esto también se cumple para el espacio de variables lingüísticas de salida.

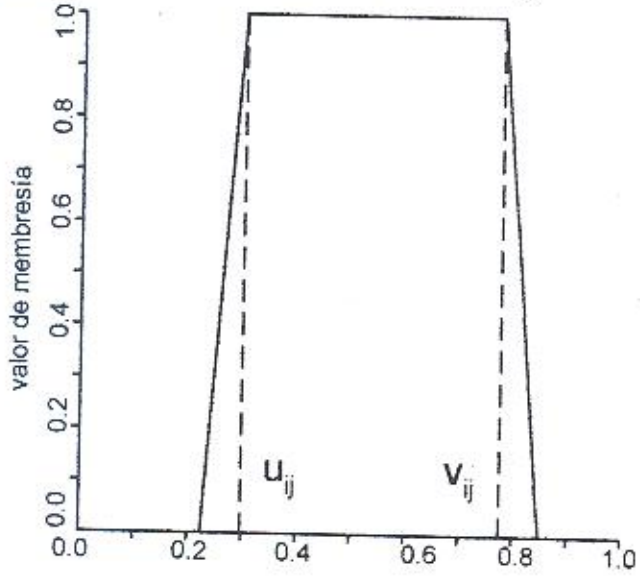


Figura 3. 5 Representación de las funciones de membresía.

En cada conexión de la capa dos existen dos pesos, $u_{ij}^{(2)}$ y $v_{ij}^{(2)}$, que corresponden a cada *j*-ésimo nodo de cada *i*-ésimo nodo lingüístico de entrada, ver Fig. 3.4. Estos

mismos pesos, son los que definen la función de membresía. El peso $u_{ij}^{(2)}$ corresponde a x_i y $v_{ij}^{(2)}$ corresponde a x_i^c, que son las entradas al *i*-ésimo nodo lingüístico. Durante el ajuste de pesos y la operación normal de la red, solo se utilizará x_i.

CAPA TRES. Cada nodo regla tiene conectado un conjunto de *n* nodos término de la capa dos, uno por cada nodo lingüístico de entrada. Es por ello que existen tantos nodos regla como nodos término existen para un nodo lingüístico de entrada. Cada variable lingüística tiene el mismo número de términos en la FALCON-ART.

Las precondiciones de las reglas lógicas difusas, están representadas por las conexiones de la capa tres. Los nodos regla realizan la operación siguiente:

$$f[z_i^{(2)}] = \prod_{i=1}^{n} z_i^{(3)} \qquad \textbf{3. 6}$$

$$\text{y } a[f(\bullet)] = f(\bullet)$$

Donde $z_i^{(3)}$ es la i-ésima entrada a un nodo de la capa 3 el producto se efectúa sobre las entradas a este nodo, los pesos en estos nodos son unitarios.

El conjunto de nodos término de cada nodo lingüístico, forman lo que se llama una hipercaja o hiperfunción de membresía. Las esquinas de la hipercaja están definidas por los pesos $u_{ij}^{(2)}$ y $v_{ij}^{(2)}$ de todos los nodos término de un determinado nodo *i*.

El conjunto de nodos regla está conectado a conjuntos de *m* nodos término de salida en la capa 4, uno por cada variable lingüística de salida. Este conjunto de nodos término de salida forman una hiperfunción de membresía *m*-dimensional en el espacio de salida que especifican las consecuencias de un nodo regla.

CAPA CUATRO. En esta capa se encuentran los nodos término de salida, los cuales tienen dos formas de operación: Transmisión abajo-arriba ó Transmisión arriba-abajo.

En la primer forma de transmisión, los enlaces de la capa 4 realizan la operación difusa OR sobre los nodos que tienen la misma consecuencia y su función de activación es:

$$f[z_i^{(4)}] = \max[z_1^{(4)}, z_2^{(4)}, \ldots, z_p^{(4)}] \qquad \textbf{3. 7}$$

$$\text{y } a[f(\bullet)] = f(\bullet)$$

Donde $z_i^{(4)}$ es la i-ésima entrada a un nodo de la capa 4 y p es el número de entradas a este nodo desde los nodos regla en la capa 3. En el segundo modo de transmisión, los nodos de esta capa y los enlaces de la capa 5, representan los nodos término de salida y son funciones de membresía unidimensionales en el espacio de salida, y tienen también los pesos adaptables $u_{ij}^{(5)}$ y $v_{ij}^{(5)}$ en cada uno de los enlaces arriba-abajo en la capa 5; es decir, realizan la misma función que los nodos y las conexiones de la capa 2.

CAPA CINCO. Cada nodo lingüístico de salida corresponde a una variable lingüística de salida. Hay dos tipos de nodo en la capa 5, unos que realizan la transmisión arriba-abajo para los datos de entrenamiento, para alimentar la red, de igual manera que los nodos lingüísticos de entrada. Donde:

$$f(y_i, y_i^c) = (y_i, y_i^c) = (y_i, 1.0 - y_i) \qquad \textbf{3. 8}$$

Donde y_i es el i-ésimo elemento del vector deseado normalizado. El segundo tipo de nodo realiza la transmisión abajo-arriba para realizar la señal de salida. Estos nodos y las conexiones asociadas a ellos actúan como dedifusificadores.

Si $u_{ij}^{(5)}$ y $v_{ij}^{(5)}$ son los vértices de la hipercaja del j-ésimo término de la i-ésima variable lingüística de salida y_i entonces podemos caracterizar las funciones siguientes para obtener el método de dedifuzificación centro del área:

$$f[z_j^{(5)}] = \sum_j w_{ij}^{(5)} z_j^{(5)} = \sum_j m_{ij}^{(5)} z_j^{(5)} \qquad \textbf{3. 9}$$

$$\text{y} \quad a[f(\cdot)] = \frac{f(\cdot)}{\sum_j z_j^{(5)}} \qquad \textbf{3. 10}$$

Donde $z_i^{(5)}$ es la entrada al i-ésimo nodo lingüístico de salida desde su j-ésimo nodo término y $m_{ij}^{(5)} = [u_{ij}^{(5)} + v_{ij}^{(5)}]/2$ denota el valor central de la función de membresía del j-ésimo término de la i-ésima variable lingüística de salida.

El centro de una región difusa está definida como el valor absoluto más pequeño entre todos los otros puntos en la región en la cual la función de membresía es igual a uno. El peso $w_{ij}^{(5)}$ en una conexión de transmisión abajo-arriba en la capa cinco, está definida por:

$$w_{ij}^{(5)} = m_{ij}^{(5)} = [u_{ij}^{(5)} + v_{ij}^{(5)}]/2 \qquad \textbf{3. 11}$$

Donde $u_{ij}^{(5)}$ y $v_{ij}^{(5)}$ son los pesos en la unión correspondiente a la transmisión arriba-abajo en la misma capa 5. Para determinar los vértices de la hipercaja en la capa 2 y 4, se sigue un algoritmo basado en la *Teoría de Resonancia Adaptable*, y para el ajuste de parámetros se utiliza el algoritmo de retropropagación.

En la presentación de cada patrón se realizan los pasos siguientes:

1. Primero se organiza la estructura mediante la partición difusa apropiada de los espacios tanto de entrada como de salida y para encontrar la formación de los nodos regla.
2. Se supervisa el ajuste óptimo de las funciones de membresía para obtener la salida deseada.

Para el inicio del entrenamiento el usuario sólo necesita los patrones de entrenamiento (***x*** & ***y*** en la Fig. 3.4), sin ser necesario que aporte información acerca de las funciones de membresía, de las reglas lógicas difusas, es decir que inicialmente la red no contiene ningún *nodo término*, ni de entrada ni de salida, y ningún *nodo regla*. Estos nodos se van formando conforme el aprendizaje avanza, y sólo se necesitan los nodos lingüísticos de entrada, y posteriormente se añadirán los demás nodos.

3.3. 1 Aprendizaje de la organización de la estructura.

En esta etapa se utilizan el patrón de entrada $x_i(t), i = 1,\ldots,n$ (donde n es el número de componentes del vector de entrada ***x***), y el patrón de salida deseado $y_i(t), i = 1,\ldots,m$ (donde m, es el número de componentes del vector de entrada ***y***). Estos vectores que se encuentran en la parte inferior y superior, respectivamente, de la Fig. 3.4. El objetivo del sistema FALCON-ART es crear:

- Las particiones difusas,
- Las funciones de membresía, y
- Las reglas lógicas difusas.

La red trabajará en ambos sentidos en la capa 4, los patrones de entrada y salida se envían hacia la red en forma simultánea. Este proceso se lleva a cabo en tres pasos de aprendizaje:

- Proceso de agrupación difusa del espacio de entrada.
- Proceso de agrupación difusa del espacio de salida.
- Proceso de mapeo.

3.3.1. 1 Proceso de agrupación difusa del espacio de entrada[30].

Para cada vector $\boldsymbol{x}$ (vector normalizado y complementado), se calculan las funciones de selección:

$$T_j(\boldsymbol{x}) = \frac{|x' \wedge w_j|}{\alpha + |w_j|} \quad j = 1,2,\ldots,N \qquad \textbf{3. 12}$$

donde "$\wedge$" es el operador mínimo realizado entre cada entrada del los dos vectores, α es el parámetro de selección, N es el número de nodos existentes en la capa y $\boldsymbol{w_j}$ es el vector de pesos complementado. Este último vector de pesos se define como: $\mathbf{w}_j \equiv \{[u_{1j}, 1-v_{1j}],\ldots,[u_{ij}, 1-v_{ij}],\ldots,[u_{nj}, 1-v_{nj}]\}$. La función de selección indica la similaridad que existe entre el vector de entrada $\boldsymbol{x}$ y el vector de pesos complementado $\boldsymbol{w_j}$. La categoría que se elige es:

$$T_J = \max\{T_j : j = 1,\ldots, N\} \qquad \textbf{3. 13}$$

La resonancia ocurre cuando la categoría seleccionada cumple con el criterio de similaridad siguiente:

$$\frac{|\mathbf{x}' \wedge \mathbf{w}_J|}{|\mathbf{x}'|} \geq \rho \qquad \textbf{3. 14}$$

En caso contrario, se reinicia la búsqueda de otra categoría que cumpla con el criterio de similaridad. En este caso la función de selección T_J se hace igual a cero durante la presentación del patrón de entrada respectivo, para prevenir la selección de la misma categoría durante la búsqueda. Se escoge un nodo nuevo J, y se continua el proceso hasta cumplir con el criterio de similaridad.

Si ningún nodo es el idóneo, se crea una nueva hipercaja, añadiéndose un grupo de *n nodos-término nuevos*, uno por cada variable lingüística de entrada, y uniendo los nuevos nodos-término con los nodos lingüísticos; además, el vector de

[30] Todas las ecuaciones de este segmento, están tomadas de la Ref. 5

pesos de este nodo recién creado es el vector actual $\boldsymbol{x}$. Este nuevo conjunto de nodos definen la nueva hipercaja, y así, una nueva categoría en el espacio de entrada, este nuevo nodo se define como en nodo J-ésimo.

3.3.1. 2 Proceso de agrupación difusa del espacio de salida[31].

Este proceso es idéntico al proceso anterior, con excepción de las capas 4 y 5, que trabajan en el modo arriba-abajo, y se utiliza el vector de entrenamiento de salida deseado $y(t)$. Y el nodo escogido y de recién creación se conoce como el K-ésimo nodo.

La hipercaja seleccionada se define por el vector de pesos complementado:

$$\mathbf{w}_K \equiv \{[u_{1j}, 1-v_{1j}],\ldots,[u_{ij}, 1-v_{ij}],\ldots,[u_{mj}, 1-v_{mj}]\} \qquad \textbf{3. 15}$$

Los dos procesos anteriores de agrupación producen la J-ésima hipercaja de entrada y la K-ésima hipercaja de salida. Si la J-ésima hipercaja de entrada no es de nueva creación, quiere decir que ya existe un nodo que corresponde a este subíndice; sí el J-ésimo nodo es de recién creación, entonces se agrega un nuevo nodo regla en la capa 3 y se conecta a los nodos término de recién creación también.

3.3.1. 3 Proceso de mapeo[32].

Una vez que las dos hipercajas han sido seleccionadas, una en el espacio de entrada y la otra en el espacio de salida, se tiene que realizar un mapeo entre los nodos de la capa 3 y los nodos de la capa 4, o lo que es lo mismo desarrollar las consecuencias de las reglas lógicas difusas. Este mapeo consiste en conectar el nodo ganador J a la hipercaja ganadora K, es decir conectar el nodo regla J a los nodos término que forman la hipercaja K en el espacio de salida. El algoritmo es el siguiente:

Paso 1) si el nodo regla J es de nueva formación entonces se conecta dicho nodo a la hipercaja de salida K.

Paso 2) de otra forma si originalmente el nodo regla J no está conectado a una hipercaja de salida K entonces desactivar el nodo J y buscar otro nodo que cumpla con el criterio de similaridad mencionado anteriormente (ir al paso 1)

Paso 3) de otra forma ningún cambio en la estructura es necesario.

[31] Todas las ecuaciones de este segmento, y los textos están tomados de la Ref. 5

En el proceso de mapeo, las hipercajas K y J se van ajustando por medio de la actualización de sus vectores de pesos, de acuerdo con:

$$w_J^{nuevo} = x' \wedge w_J^{viejo}$$
$$w_K^{nuevo} = y' \wedge w_K^{viejo} \qquad \textbf{3. 16}$$

En el proceso de mapeo la consecuencia de un nodo regla se decide en el proceso de mapeo, y no se cambia en absoluto en lo futuro. En resumen, para el aprendizaje de la estructura se utiliza el algoritmo de los sistemas ART y se realiza con los pasos siguientes:

1. Se calculan las funciones de selección T_j.
2. Se escoge la T_j de mayor magnitud.
3. Se verifica el criterio de resonancia. $\frac{|x \wedge w_J|}{|x|} \geq \rho$
   a) Se cumple: se realiza el aprendizaje, se va al paso 6
   b) No se cumple: se hace 0 la función de selección del punto 2.
4. Se escoge la Tj inmediata inferior.
5. Se va al paso 3.
6. Se lee un nuevo patrón, se va al paso 1.

3.3. 2 Aprendizaje del ajuste de parámetros[33].

Una vez aprendida la estructura, se entra al siguiente paso, que es ajustar óptimamente los parámetros de las funciones de membresía, con el mismo patrón de entrenamiento. La idea básica del algoritmo de retropropagación es corregir el error de salida de cada nodo en cada capa.

El objetivo es minimizar la función de error:

$$E = \frac{1}{2}[y(t) - \hat{y}(t)]^2 \qquad \textbf{3. 17}$$

[32] Todas las ecuaciones de este segmento, están tomadas de la Ref. 5
[33] La totalidad de ecuaciones de este segmento está tomada de la Ref. 5

Donde $y(t)$ es la salida deseada y $\hat{y}(t)$ es la salida actual de la red. El algoritmo inicia a los nodos de salida y se procede regresivamente a los nodos de las capas inferiores para calcular $\frac{\partial E}{\partial y}$.

Durante el aprendizaje de parámetros se utiliza x y y en lugar de los vectores complementados. Supongamos que sea w el parámetro ajustable en un nodo, la regla general de aprendizaje será:

$$w(t+1) = w(t) + \eta(-\frac{\partial E}{\partial w}) \qquad \textbf{3. 18}$$

donde: $\frac{\partial E}{\partial w} = \frac{\partial E}{\partial f}\frac{\partial f}{\partial w} = \frac{\partial E}{\partial a}\frac{\partial a}{\partial f}\frac{\partial f}{\partial w}$ y η es el parámetro de velocidad de aprendizaje.

Se derivan a continuación las regla de aprendizaje capa por capa para el ajuste de los vértices $u_{ij}'s$ y $v_{ij}'s$, que son los parámetros ajustables en esta estructura.

CAPA CINCO. La regla de aprendizaje para actualizar los vértices de una hiperfunción de membresía $v_j^{(5)}$ es:

$$\frac{\partial E}{\partial v_j^{(5)}} = \frac{\partial E}{\partial a^{(5)}}\frac{\partial a^{(5)}}{\partial v_j^{(5)}} = -\left[y^d(t) - y(t)\right]\frac{z_j^{(5)}}{2\sum_j z_j^{(5)}} \qquad \textbf{3. 19}$$

por lo que el vértice $v_j^{(5)}$ se actualiza como:

$$v_j^{(5)}(t+1) = v_j^{(5)}(t) + \eta\left[y^d(t) - y(t)\right]\frac{z_j^{(5)}}{2\sum_j z_j^{(5)}} \qquad \textbf{3. 20}$$

De igual forma, la regla de aprendizaje para actualizar el vértice $u_j^{(5)}$ es:

$$\frac{\partial E}{\partial u_j^{(5)}} = \frac{\partial E}{\partial a^{(5)}}\frac{\partial a^{(5)}}{\partial u_j^{(5)}} = -\left[y^d(t) - y(t)\right]\frac{z_j^{(5)}}{2\sum_j z_j^{(5)}} \qquad \textbf{3. 21}$$

por lo que el vértice $u_j^{(5)}$ se actualiza como:

$$u_j^{(5)}(t+1) = u_j^{(5)}(t) + \eta\left[y^d(t) - y(t)\right]\frac{z_j^{(5)}}{2\sum_j z_j^{(5)}} \qquad \textbf{3. 22}$$

el error que se propaga de la capa anterior es:

$$\delta^{(5)} = -\frac{\partial E}{\partial a^{(5)}} = -\frac{\partial E}{\partial y} = y^d(t) - y(t) \qquad \textbf{3. 23}$$

CAPA CUATRO. En el modo de transmisión abajo-arriba, en esta capa no existen parámetros que deban ser ajustados. Se calcula únicamente la señal de error $\delta_i^{(4)}$ y se propaga hacia las capas inferiores. Esta señal de error puede ser calculada:

$$\delta_i^{(4)} = -\frac{\partial E}{\partial a^{(4)}} = -\frac{\partial E}{\partial a^{(5)}}\frac{\partial a^{(5)}}{\partial a^{(4)}} = \delta_i^{(5)}\frac{m_i^{(5)}\sum_i z_i^{(5)} - \sum_i m_i^{(5)} z_i^{(5)}}{(\sum_i z_i^{(5)})^2} \qquad \textbf{3. 24}$$

En el caso de múltiples salidas, los cálculos en las capas 4 y 5 son iguales para todas las variables lingüísticas de salida.

CAPA TRES. Al igual que en la capa 4, únicamente se calcula la señal del error, y se puede obtener de la forma siguiente:

$$\delta_i^{(3)} = -\frac{\partial E}{\partial a^{(3)}} = -\frac{\partial E}{\partial a^{(4)}}\frac{\partial a^{(4)}}{\partial f^{(4)}}\frac{\partial f^{(4)}}{\partial a^{(3)}} = \delta_i^{(4)}\frac{z_i^{(4)}}{z_{max}^{(4)}} \qquad \textbf{3. 25}$$

Donde $z_{max}^{(4)} = max($ de las entradas al nodo término de salida $j)$. El término

$$z_i^{(4)} \;/\; z_{max}^{(4)} \qquad \textbf{3. 26}$$

normaliza el error que es calculado en las reglas que se activan con la misma consecuencia. Si existen múltiples salidas, entonces la señal de error se convierte en

$$\delta_i^{(3)} = \sum_k \delta_k^{(4)} z_k^{(4)} / z_{max}^{(4)} \qquad \textbf{3. 27}$$

Donde la sumatoria se realiza sobre las consecuencias de una nodo regla, es decir, que el error de un nodo regla es la suma de los errores de sus consecuencias.

CAPA DOS. La regla de aprendizaje para el vértice $v_{ij}^{(2)}$ se deriva como sigue:

$$-\frac{\partial E}{\partial v_{ij}^{(2)}} = -\frac{\partial E}{\partial a^{(3)}}\frac{\partial a^{(3)}}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial v_{ij}^{(2)}} \qquad \textbf{3. 28}$$

donde:

$$\frac{\partial a^{(3)}}{\partial a^{(2)}} = 1$$

$$\frac{\partial a^{(2)}}{\partial v_{ij}^{(2)}} = \begin{cases} \gamma / n & \text{si } 0 \le (x_i - v_{ij}^{(2)})\gamma \le 1 \\ 0 & \text{de otra forma} \end{cases} \qquad \textbf{3. 29}$$

y la regla de actualización de $v_{ij}^{(2)}$ es como sigue:

$$v_{ij}^{(2)}(t+1) = v_{ij}^{(2)}(t) + \eta \delta_i^{(3)} \frac{\partial a^{(2)}}{\partial v_{ij}^{(2)}} \qquad \textbf{3. 30}$$

De forma similar, la regla de aprendizaje para el vértice $u_{ij}^{(2)}$ se deriva como sigue:

$$-\frac{\partial E}{\partial u_{ij}^{(2)}} = -\frac{\partial E}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial u_{ij}^{(2)}} \qquad \textbf{3. 31}$$

donde:

$$\frac{\partial a^{(3)}}{\partial a^{(2)}} = 1$$

$$\frac{\partial a^{(2)}}{\partial u_{ij}^{(2)}} = \begin{cases} -\gamma / n & \text{si } 0 \le (u_{ij}^{(2)} - \bar{x}_t)\gamma \le 1 \\ 0 & \text{de otra forma} \end{cases} \qquad \textbf{3. 32}$$

y la regla de actualización de $u_{ij}^{(2)}$ es como sigue:

$$u_{ij}^{(2)}(t+1) = u_{ij}^{(2)}(t) + \eta \delta_i^{(3)} \frac{\partial a^{(2)}}{\partial u_{ij}^{(2)}} \qquad \textbf{3. 33}$$

El sistema FALCON-ART es un modelo general de un sistema de control difuso, que puede aprender en línea particionando los espacios de entrada y salida, y encontrando las reglas lógicas difusas y las funciones de membresía apropiadas en forma dinámica. Este sistema previene el crecimiento combinatorio de la memoria asociativa difusa de sistemas complejos de control.

CAPÍTULO CUATRO

ENTRENAMIENTO Y APLICACIÓN DEL CONTROL FALCON-ART COMO REGULADOR DE VOLTAJE EN UNA MÁQUINA SÍNCRONA.

4. 1 Introducción.

En este segmento se presentan los aspectos contemplados para llevar a cabo el entrenamiento de la estructura FALCON-ART como un regulador de voltaje en un generador síncrono. El esquema de entrenamiento para cada caso particular en donde se apliquen redes como la FALCON-ART, es responsabilidad y dependerá de la creatividad del diseñador, ya que la bibliografía puede resultar escasa y superficial, conteniendo aplicaciones que no proveen la información suficiente para aplicaciones específicas.

El entrenamiento se llevó a cabo sobre 30 condiciones iniciales del generador síncrono, tomadas de un total de 200, y las restantes 170, se utilizaron en la etapa de producción. Aquí se presentan algunos de los resultados obtenidos a partir de esas pruebas y que demostraron que el entrenamiento llevado a cabo, resultó ser el adecuado para este caso de aplicación en particular.

Los problemas que se tuvieron que resolver durante el diseño de la red FALCON-ART, fue el desarrollo y la comprobación de un buen funcionamiento en todas sus partes componentes, ya que su estructura envuelve desarrollar la red ART difusa[34], la ART 1[35] y el ARTMAP difuso[36]. Estas estructuras fueron desarrolladas y verificadas de acuerdo a los ejercicios sugeridos por los autores de estas.

[34] Ver Ref. 1

Posteriormente, el siguiente problema a resolver fue el acople de la estructura FALCON-ART dentro del modelo de máquina síncrona, determinar cuales serían sus señales de entrada, y la forma en que el programa computacional debía ejecutarse para tomar en cuenta el cambio de las variables de entrada a la red. En seguida, la sintonización de los parámetros que determinan el comportamiento de la red, debían hallarse a través de las pruebas, que sumaron una cantidad considerable antes de poder obtener resultados satisfactorios del sistema en conjunto: Maquina, red FALCON-ART, parámetros del FALCON-ART, secuencia de entrenamiento, almacenamiento y análisis de datos. Así, el diseño de redes y su aplicación a casos específicos ponen a prueba la paciencia y la imaginación de autores y usuarios.

Para la aplicación del sistema FALCON-ART como un regulador de voltaje en un generador síncrono, se realizaron diferentes secuencias de entrenamiento, tomando en cuenta por ejemplo: No. de épocas de entrenamiento, el número y tipo de entradas al sistema FALCON-ART, No. de condiciones iniciales del modelo de generador síncrono que se tomaban en cuenta para el entrenamiento, y método de solución de las ecuaciones del modelo de generador síncrono.

Se llevó un registro de pruebas para tenerlo como guía cuando se obtenían resultados positivos, para, de ser así, seguir en esa tendencia, si no, se decidía un cambio radical en la configuración de todo el sistema. La batería de pruebas se llevó a cabo en el área de digitales de la Sección de Posgrado e Investigación de la Sección Eléctrica.

4. 2 APRENDIZAJE SUPERVISADO, regla de aprendizaje mediante la cual se entrenó el sistema FALCON-ART.

En este esquema de entrenamiento, la respuesta deseada ***d(t)*** es proporcionada al sistema. En el aprendizaje supervisado, se proporcionan una serie de pares ordenados: $(\boldsymbol{x}^{(1)},\boldsymbol{d}^{(1)})$, $(\boldsymbol{x}^{(2)},\boldsymbol{d}^{(2)}),\ldots,(\boldsymbol{x}^{(k)},\boldsymbol{d}^{(k)})$ constituidos por entradas y salidas deseadas. Cuando cada vector de entrada $\boldsymbol{x}^{(k)}$ se proporciona al sistema, al

[35] Ver Ref. 2

[36] Ver Ref. 4

mismo tiempo se proporciona la salida deseada $d^{(k)}$. En la Fig. 4.1, se mide la diferencia entre la salida actual $y(k)$ y la salida deseada $d(t)$, esta señal de error se envía al sistema neuro-difuso para que efectúe el cambio en sus pesos, y la salida actual sea corregida para aproximarse a la salida deseada. En este esquema de aprendizaje, se supone que el valor deseado $d(k)$ es conocido con anterioridad en la presentación de cada patrón de entrada.

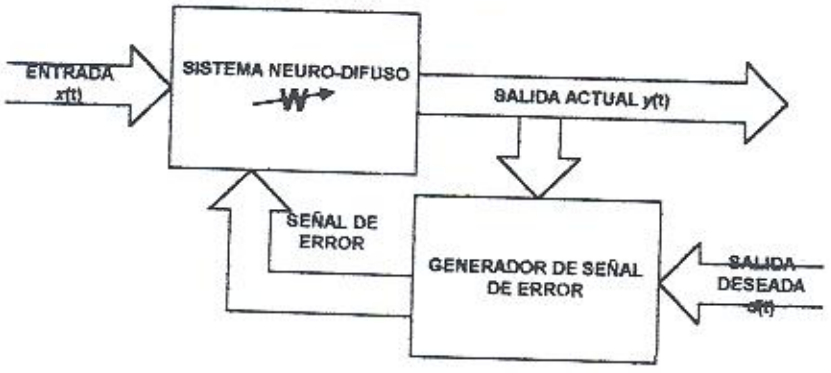


Figura 4.1
Aprendizaje
Supervisado.

4. 3 Selección de las variables de entrada al sistema FALCON-ART.

La selección de variables de entrada a un sistema de control es de importancia vital, ya que de ellas dependerá el éxito del diseño de un controlador. Para la estructura FALCON-ART se describe a continuación algunos de los factores que determinaron, finalmente, la selección de las variables de entrada a este sistema.

Durante la investigación, se efectuaron pruebas al sistema FALCON-ART en las que el vector de entrada no se consideraba la variable que se deseaba controlar (voltaje de excitación); es decir, se utilizaba el sistema FALCON-ART en lazo abierto. Posteriormente, se incluyó esta variable en el vector de entrada, reflejándose inmediatamente en las pruebas subsecuentes con resultados positivos, pero sin ser aún satisfactorios.

Así también, se contempló como variable de entrada, el vector completo de variables de estado de la planta a controlar y la variación del error del voltaje terminal. La inclusión de la primera variable en el vector de entrada, ocasionaba un aumento considerable en el número de operaciones llevadas a cabo por la red; y la

segunda variable proporcionaba un mejoramiento casi nulo en el comportamiento de la planta durante condiciones de prueba.

La selección de variables de entrada a los espacios de entrada y salida se definieron a partir de las pruebas realizadas, en las que se tomó en cuenta, finalmente, las variables siguientes: *el voltaje terminal, el voltaje de excitación de campo, el error del voltaje y el error del voltaje de excitación.*

4.3. 1 Espacio de Entrada.

4.3.1. 1 Entrenamiento.

El espacio de entrada se define como el espacio vectorial en donde se conjuntan todos los patrones que se recibirán como entradas durante el periodo de producción de la red. Para este espacio se tomaron en cuenta las entradas siguientes:

- Voltaje terminal $V_{ter}(t)$.
- Voltaje de excitación $V_{exc}(t)$.
- Error del voltaje terminal $E_{ter}(t)$
- Error del voltaje de excitación $E_{exc}(t)$.

Este conjunto de entradas se leen en un tiempo determinado **t** y se envían a la estructura FALCON-ART , ver Fig. 4.2 a).

4.3.1. 2 Producción.

Se mantienen las mismas variables de entrada de la etapa de entrenamiento. Las variables son leídas en un tiempo ***t***, y se envían a la estructura FALCON-ART, la señal generada por la red se envía al modelo de generador síncrono sustituyendo la señal enviada por el sistema de regulación ST1.

4.3. 2 Espacio de Salida.

4.3.2. 1Entrenamiento.

El espacio de salida, se define como el espacio vectorial donde se conjuntan todos los patrones que serán entregados por la red en la etapa de producción, estas entradas se leen en un tiempo ***t+1*** y se envían a la estructura FALCON-ART. Las entradas a este espacio son:

- Voltaje de excitación $V_{exc}(t)$.
- Error del voltaje de excitación $E_{exc}(t)$.

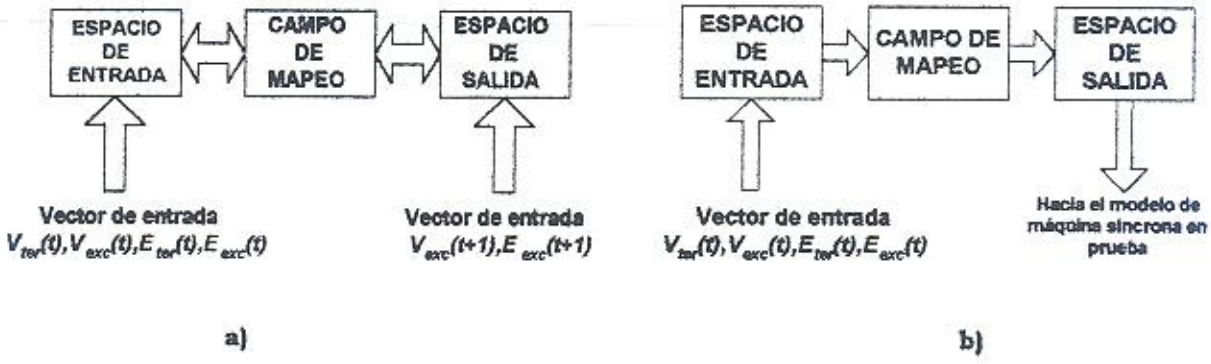


Figura 4. 2 Flujo de señales generalizado durante: a) etapa de entrenamiento; b) etapa de producción, en la estructura FALCON-ART

4.3.2. 2 Producción.

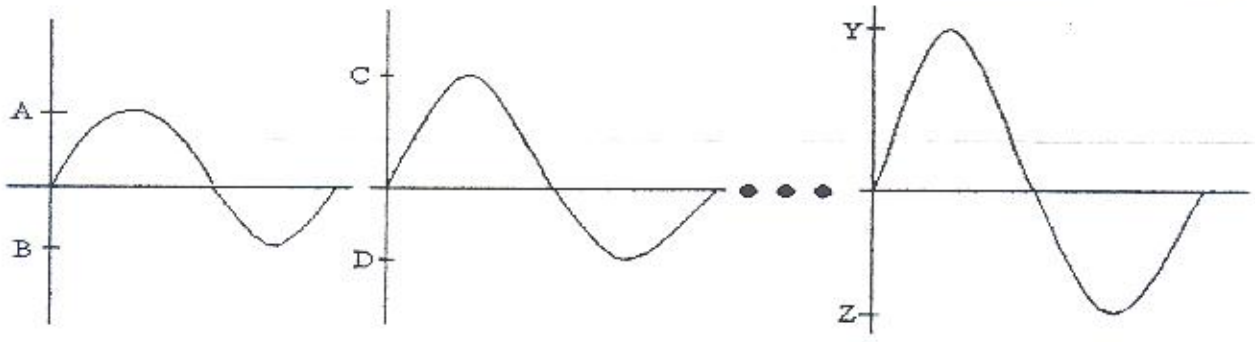
Durante esta etapa de producción se bloquean las entradas al espacio de salida, ver Fig. 4.2b), ya que la estructura FALCON-ART ya ha clasificado y estructurado ambos espacios de entrada.

4. 4 Normalización y Complementación de las entradas al Sistema FALCON-ART.

El preprocesamiento de las señales de entrada a los espacios vectoriales de entrada, es necesario para limitar el excesivo número de clases que pudieran formarse en ambos espacios.

La normalización de las entradas para el entrenamiento de la estructura FALCON-ART en este caso en particular de aplicación, se realizó con una metodología diferente a la propuesta en la Ec. 3.2. La razón de esta metodología es debido a que en este caso, la normalización propuesta en la Sección 3.2 origina la creación de una cantidad exagerada de clases tanto en el espacio de entrada como en el de salida.

Así, se consideran N condiciones iniciales del generador síncrono en donde se considera, que la señal a normalizar pudiera tener valores y comportamientos como se muestran en la Fig. 4.3:



Cond. Inic. 1 **Cond.Inic. 2** . . . **Cond. Inic. N**

Figura 4. 3 Normalización de señales a ser codificadas por el sistema FALCON-ART

Durante la simulación de la *Cond. Inic. 1*, ver Fig. 4.3, las siglas A y B en la ec. 4.1, son tomadas como valor máximo local y valor mínimo local, respectivamente.

Una vez hallados estos límites, se realiza una simulación adicional *(misma condición inicial)* donde se utilizan estos máximos y mínimos locales en la ecuación siguiente:

$$\text{variable}_{\text{normalizada}} = \frac{\text{variable}_{\text{valor real}} - B}{|A - B|} \qquad 4.1$$

Lo que daba como resultado, una señal normalizada que puede codificarse correctamente en la estructura FALCON-ART. Respectivamente, se siguió la misma metodología en la *Cond. Inic. 2* hasta la *Cond. Inic. N.*

Además, fue necesario crear un máximo global y un mínimo global que pudieran normalizar todas las condiciones iniciales entrenadas para la etapa de prueba. En la Fig. 4.3, este máximo y mínimo global es Y y Z.

La complementación del vector de entrada al sistema FALCON-ART se basa en la ecuación 3.1 y se lleva a cabo en el interior de la subrutina control FALCON-ART.

4. 5 Propuesta de Entrenamiento y producción.

Una vez diseñada la estructura FALCON-ART, se llevó a cabo un programa donde se acopla la estructura FALCON-ART y el modelo de generador síncrono en el

que se desea controlar el voltaje terminal mediante la regulación del voltaje de campo. Esta propuesta se conforma de dos etapas:

- Entrenamiento y,
- Producción.

4.5. 1 Entrenamiento.

Este bloque de código contiene la obtención del máximo y mínimo local *(Simulación uno, Fig. 4.4)* que se utilizan en el entrenamiento *(Simulación dos, Fig. 4.4)* del sistema FALCON-ART. La Simulación uno, consiste en efectuar las condiciones de falla en el generador síncrono y monitorear los valores de las señales que se desean codificar, para obtener su valor máximo y mínimo alcanzado. Las variables monitoreadas en esta etapa son: El voltaje terminal, el error y el error del voltaje de excitación. La razón de monitorear éstas variables únicamente, es que se desconoce el valor máximo y mínimo que pueden alcanzar durante la prueba. La variable de entrada que es el voltaje de campo tiene límites previamente determinados, por lo que se descartan en esta etapa de la propuesta.

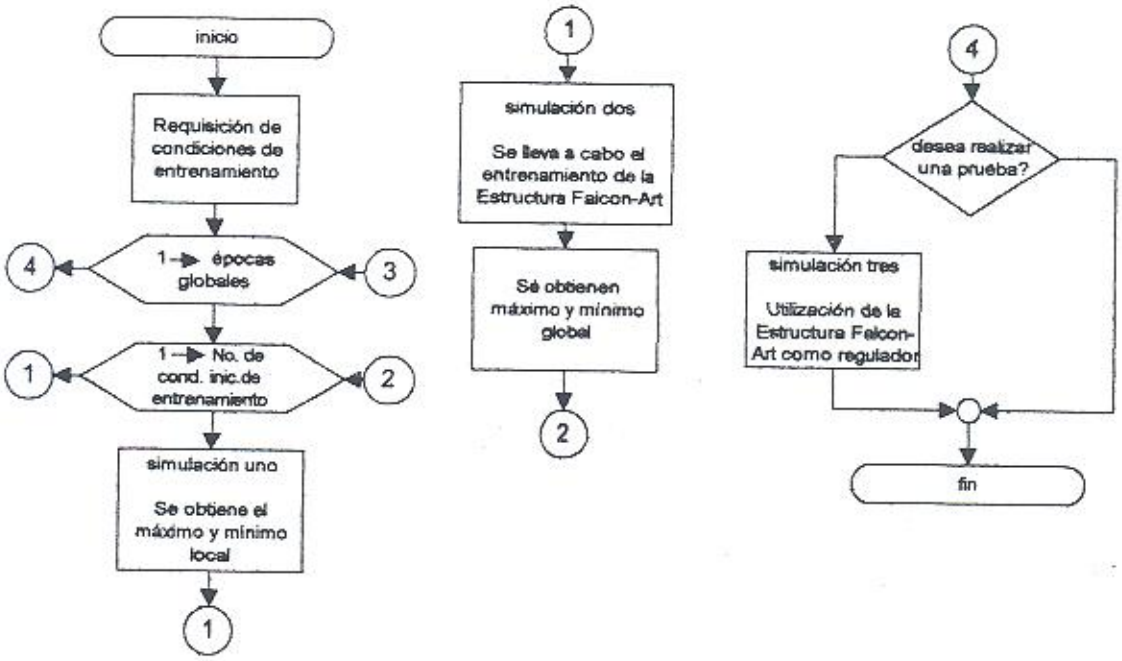


Figura 4. 4 Propuesta de entrenamiento y prueba para la estructura FALCON-ART

La simulación dos permite llevar a cabo el entrenamiento de la estructura FALCON-ART, en esta se envían las señales ya normalizadas, tanto del espacio de salida como de entrada, para que sean codificadas en los espacios correspondientes.

4.5. 2 Producción.

Esta etapa se contempla en la tercera parte del diagrama de flujo de la Fig. 4.4, en ella se pueden simular las condiciones iniciales con las que se entrenó, así como también las condiciones iniciales restantes [37].

4. 6 Ajuste de parámetros de la red FALCON-ART.

Muchas de las clases de redes neuronales artificiales, contienen en su arquitectura, parámetros que intervienen directamente en su comportamiento durante y después del entrenamiento. El sistema FALCON-ART contiene 5 parámetros que realizan esta función y se denominan de la siguiente manera:

- Parámetro de vigilancias para el espacio de entrada: ρ_A
- Parámetro de vigilancia para el espacio de salida: ρ_B
- Velocidad de aprendizaje: η
- No. de entradas al espacio de entrada: ENT
- No. de entradas al espacio de salida: ENT_B

Las pruebas realizadas con la propuesta de entrenamiento anterior, se enfocaron principalmente en la variación de ρ_A, ρ_B y η. El parámetro ρ_A nos ayuda a determinar el grado de pertenencia que tiene un patrón determinado en una clase ya codificada en el espacio de entrada y también nos ayuda a limitar el tamaño de la clase misma. Ambas restricciones anteriores se evalúan durante las pruebas.

Igualmente el parámetro ρ_B realiza la misma función pero para el espacio de salida, el parámetro η permite determinar el grado de modificación que se les aplica

[37] Ver Apéndice D.

a los pesos contenidos en ambos espacios, entrada y salida, por lo que con ello se aumenta o reduce la velocidad de aprendizaje.

Las pruebas realizadas revelaron que los parámetros ρ_A y ρ_B determinan completamente la exactitud de los resultados, y para en este caso en particular, sus valores quedaron establecidos como: $\rho_A = 0.5$ $\rho_B = 0.9$

El parámetro de velocidad de aprendizaje η, se determinó en un valor de η=0.001. Este parámetro puede tener valores que dependen, del tipo de aplicación y del tipo de patrones que se estén manejando en las pruebas. Un valor muy bajo de η ocasionaba que la señal de salida de la estructura FALCON-ART, no fuera alcanzada por más iteraciones que se ejecutaban, un valor muy alto ocasionaba un movimientos repentinos de ajuste. Así, este parámetro es de vital importancia para el aprendizaje.

Los valores de los parámetros mencionados anteriormente, se determinan por ensayo y error; así, una vez diseñada la estructura FALCON-ART y establecida la propuesta de entrenamiento, el programador debe ejecutar un número considerable de pruebas que le permitan desarrollar un sentido común, que lo dirijan hacia resultados que satisfagan las condiciones de comportamiento del sistema.

4. 7 Generación de condiciones iniciales.

Las condiciones iniciales fueron generadas mediante un programa[38] de flujos de cargas. A este programa de flujo de carga se le proveía con el valor de:

Voltaje Terminal: Rango de variación 0.98 - 1.06

Potencia real: Rango de variación 0.5 – 1.0

Dentro estos rangos se generaban valores aleatoriamente, que se utilizaban posteriormente para calcular el ángulo en terminales de la máquina, al igual que la potencia reactiva generada por esta.

Se generaron 200 condiciones iniciales, con las 30 primeras se realizó el entrenamiento, y con las 170 siguientes se llevó a cabo la producción; sin embargo,

[38] Ver Apéndice E

4. 8 Desarrollo del entrenamiento.

Una vez que se determinaron las condiciones iniciales anteriores, se desarrolló en entrenamiento en donde se tomaron las condiciones siguientes:

- Parámetro de vigilancia espacio de entrada: $\rho_A = 0.5$
- Parámetro de vigilancia espacio de salida: $\rho_B = 0.9$
- Parámetro de aprendizaje: $\eta = 0.001$
- Tiempo de simulación: 5 seg.
- Inicio de la falla: 1.0 seg. después de iniciada la simulación.
- Tiempo de liberación: 4.0 ciclos
- Patrones de entrenamiento en cada simulación: 500
- Épocas de entrenamiento: 5
- Condiciones iniciales entrenadas: 30

Durante el entrenamiento del sistema FALCON-ART se logra desarrollar las variables lingüísticas y las reglas difusas sin necesidad de proveer al sistema ningún tipo de dato, además se logra sintonizar los dominios que corresponden a cada una de las etiquetas lingüísticas, siendo esto una gran ventaja sobre la sintonización a ensayo y error. La función del sistema FALCON-ART es desarrollar las correspondencias entre las clases de entradas y las clases de salidas para que el sistema funcione correctamente, para esta aplicación en particular se obtuvo la configuración de la Fig. 4.5.

Esta estructura comprende en sus diferentes capas, los nodos que configuran el sistema FALCON-ART y que le proporcionan sus características propias. Estos nodos son los siguientes:

- 4 variables de entrada (capa 1)
- seis etiquetas lingüísticas para cada una de las 4 variables de entrada (capa 2)
- seis reglas difusas (capa 3)
- tres etiquetas lingüísticas para cada una de las 2 variables de salida (capa 4), y
- 2 variables de salida (capa 5)

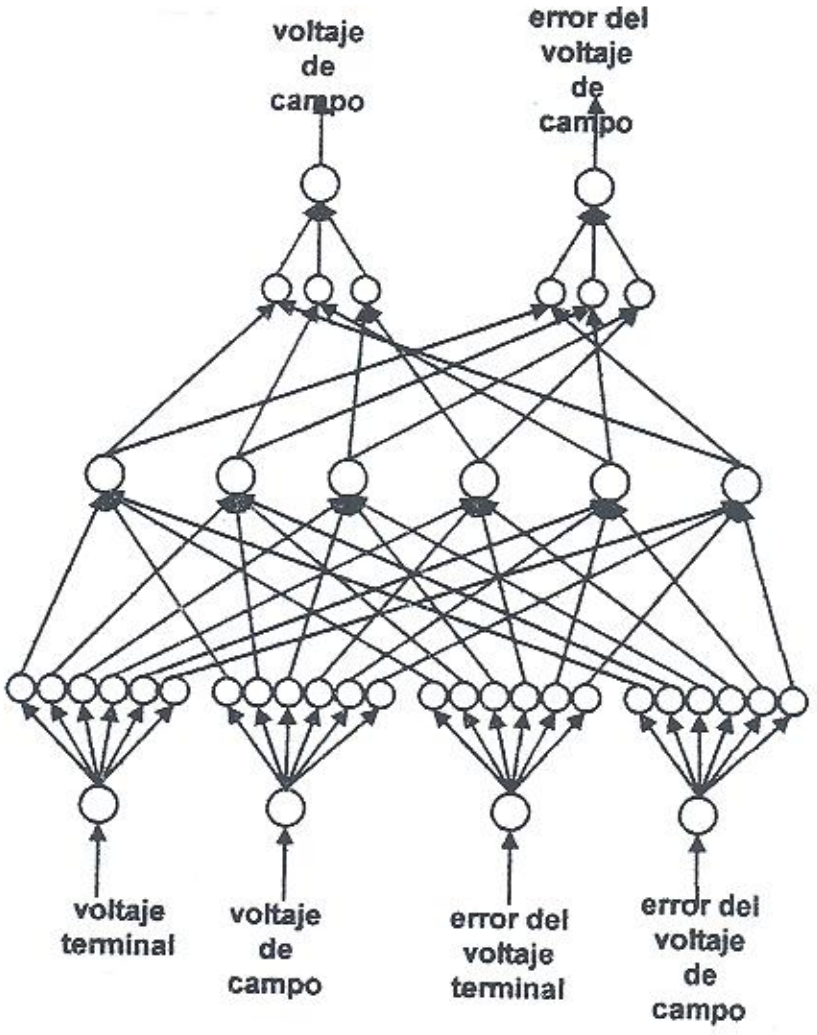


Figura 4. 5 Configuración final del sistema FALCON-ART

Haciendo un bosquejo de la relación existente entre las clases de entrada y salida pueden representarse como en la Fig. 4.6, en donde se esquematizan ambos espacios vectoriales, que contienen clases codificadas dentro de ellos, y que guardan una correspondencia única entre ellas, esta se mantiene sin cambio durante la etapa de producción del sistema FALCON-ART.

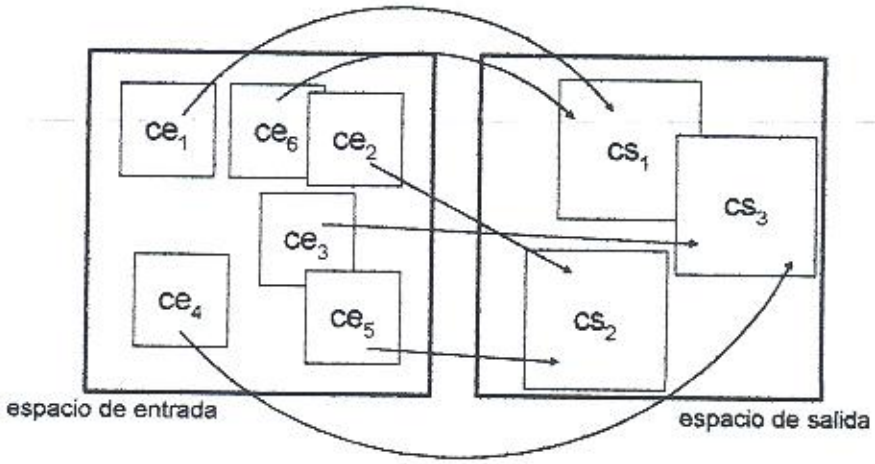


Figura 4. 6 Correspondencia de las clases existentes en el espacio de entrada y el espacio de salida

La relación entre las clases difusas puede describirse como reglas difusas, estas son determinadas por el entrenamiento del sistema FALCON-ART, y pueden enunciarse como sigue:

1. Si $\boldsymbol{x}$ es ce_1 entonces $\boldsymbol{y}$ es cs_1
2. Si $\boldsymbol{x}$ es ce_2 entonces $\boldsymbol{y}$ es cs_2
3. Si $\boldsymbol{x}$ es ce_3 entonces $\boldsymbol{y}$ es cs_3
4. Si $\boldsymbol{x}$ es ce_4 entonces $\boldsymbol{y}$ es cs_3
5. Si $\boldsymbol{x}$ es ce_5 entonces $\boldsymbol{y}$ es cs_2
6. Si $\boldsymbol{x}$ es ce_6 entonces $\boldsymbol{y}$ es cs_1

Donde : ce = clase de entrada; cs = clase de salida

Las reglas nos indican las diferentes clases en las que se puede codificar a las señales de entrada. Esta correspondencia se mantiene sin cambio alguno durante la etapa de producción; por ejemplo: si la entrada $\boldsymbol{x}$ es de la clase de entrada ce_5 el vector de salida $\boldsymbol{y}$, se encontrará en la clase codificada cs_2. O si $\boldsymbol{x}$ es de la clase de entrada ce_4, el vector de salida $\boldsymbol{y}$ se hallará en la clase codificada cs_3.

4. 9 Clases codificadas en los espacios de entrada y salida.

En este inciso se presenta la representación tradicional de las etiquetas lingüísticas en un sistema difuso y que se utiliza para conocer los dominios lingüísticos de cada una de estas etiquetas.

Se presentan las demarcaciones en cada una de las etiquetas que se encuentran en los espacios vectoriales, se da una ubicación sobre los planos XY para ver su ubicación sobre el universo del discurso, y se da una ubicación de las etiquetas lingüísticas en forma tridimensional para poder apreciar mejor su ubicación ya que algunas de ellas se encuentran sobre puestas.

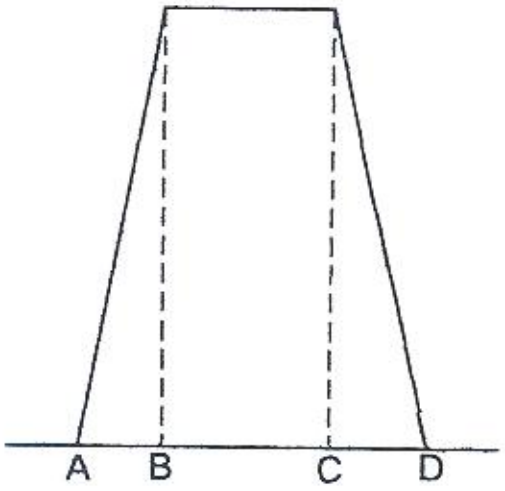
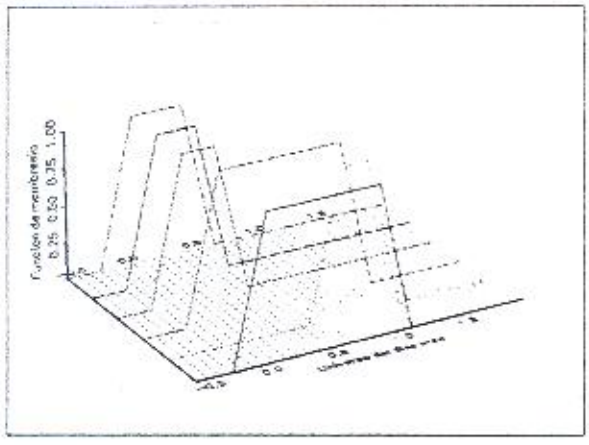


Figura 4. 7 Diagrama esquemático de una etiqueta lingüística sobre el universo

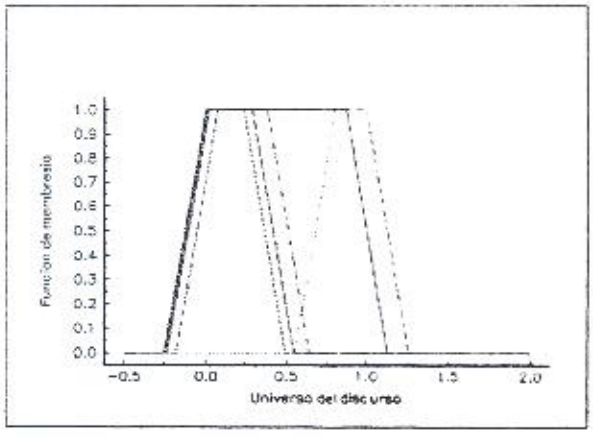
En la Fig. 4.7 se esquematiza una etiqueta lingüística, en donde se muestran las siglas: A, B, C y D. Éstas siglas representan los puntos importantes de la etiqueta; las siglas B y C definen la función de membresía y son obtenidos durante el entrenamiento del sistema FALCON-ART. Los límites A y D se obtienen en la graficación de las funciones de membresía. Los valores de éstas siglas se dan para cada una de las etiquetas lingüísticas de las variables de entrada y de salida que se encuentran distribuidas sobre el universo del discurso correspondiente, estos valores se dan para una mayor utilidad de los resultados obtenidos en el entrenamiento de la estructura FALCON-ART.

4.9. 1 Espacio de entrada.

4.9.1. 1 Error del voltaje terminal.

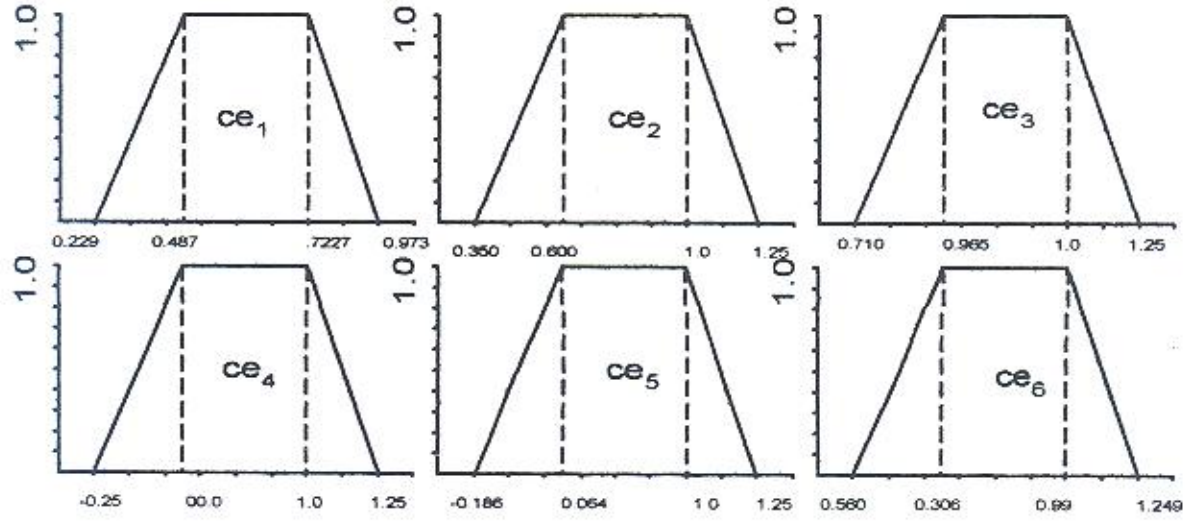


a)



b)

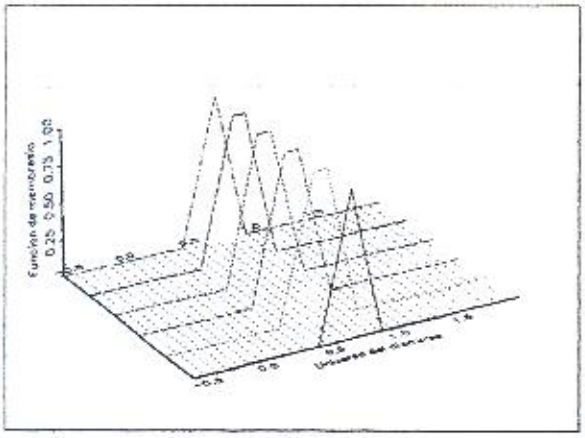
De acuerdo a la Fig. 4.5 donde se muestra la configuración final del FALCON-ART, las variables de entrada (parte inferior de la Fig. 4.5) tendrá, cada una de ellas, 6 funciones de membresía que se formaron durante el entrenamiento. En estas Figuras se puede ver la distribución de las funciones de membresía para el error del voltaje terminal. También se puede apreciar los valores en los cuales se establecen los dominios de cada una de las funciones de membresía



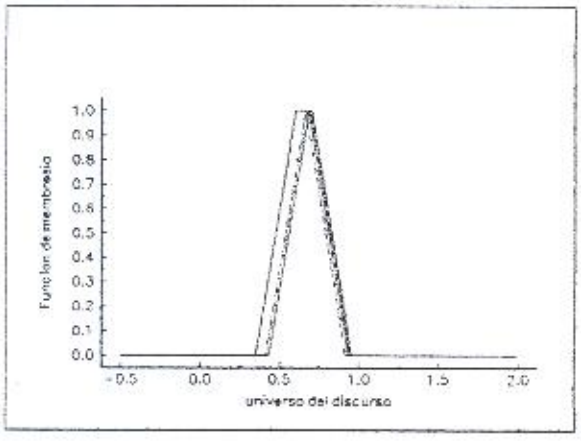
c)

Figura 4. 8 Funciones de membresía para el error del voltaje terminal en el espacio de entrada: a) Vista 3D, b) Vista XY, c) Dominio de las funciones de membresía.

4.9.1. 2 *Error del voltaje de excitación.*



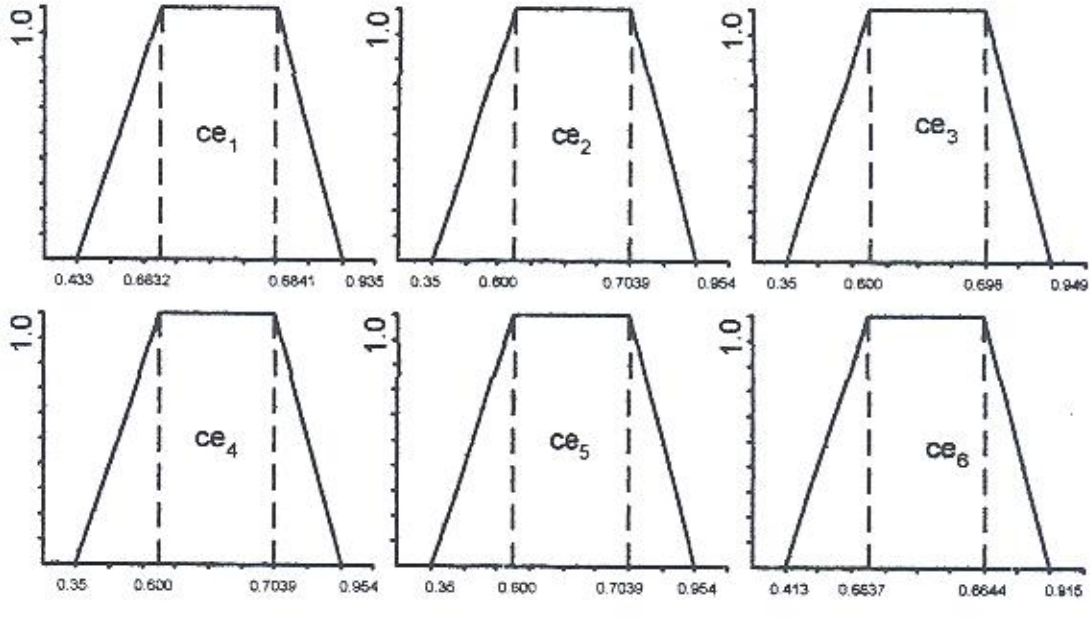
a)



b)

El Error del voltaje de excitación forma parte del vector de entrada, por lo tanto también se formarán 6 funciones de membresía. Para esta variable, sus funciones de membresía tienen un alto grado de traslapamiento, por lo que la vista 3D permite apreciar mejor la distribución de éstas en el Universo de discurso de la variable.

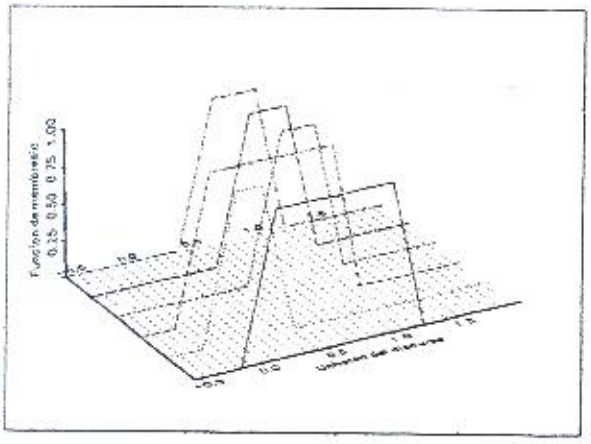
También se proporcionan los valores donde se ubican los límites de las funciones de membresía para esta variable.



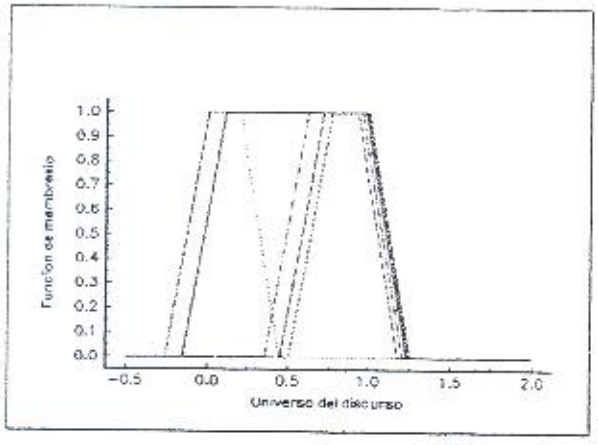
c)

Figura 4. 9 Funciones de membresía para el error del voltaje de excitación en el espacio de entrada: a) Vista 3D, b) Vista XY, c) Dominio de las funciones de membresía.

4.9.1. 3 *Voltaje terminal*



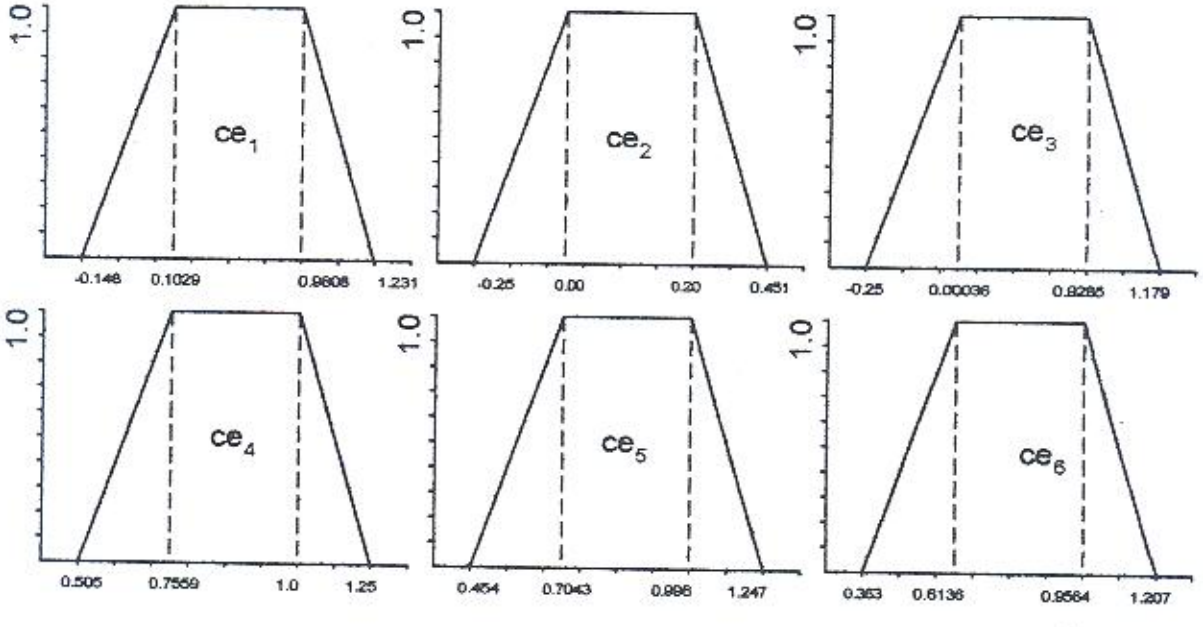
a)



b)

El sistema FALCON-ART forma tambíen 6 funciones de membresía para el voltaje terminal, de acuerdo a la Fig. 4.5. Las figuras 4.10 a) y b) muestran la distribución de las funciones de membresía en el Universo de discurso de la variable.

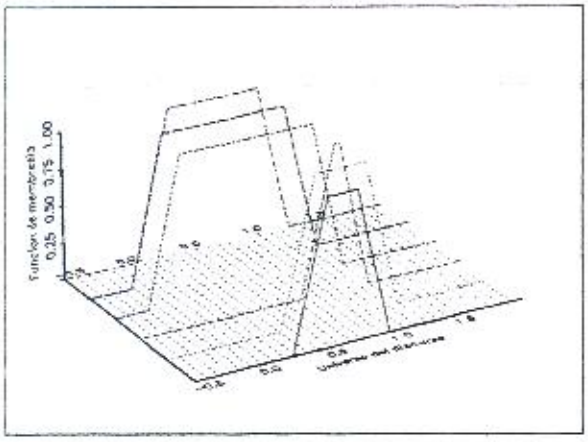
También se proporcionan los valores donde se ubican los límites de las funciones de membresía para esta variable.



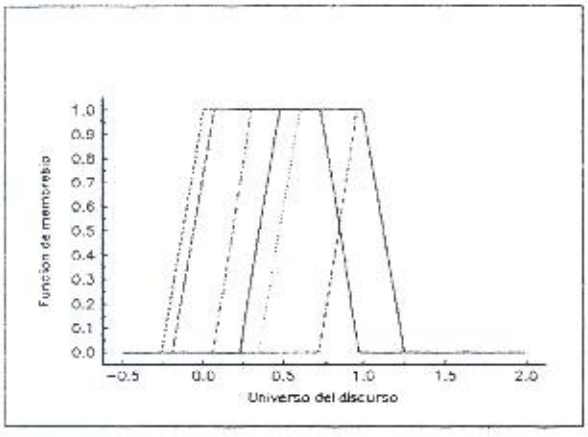
c)

Figura 4. 10 Funciones de membresía para el voltaje terminal en el espacio de entrada: a) Vista 3D, b) Vista XY, c) Dominio de las funciones de membresía.

4.9.1. 4 *Voltaje de campo*



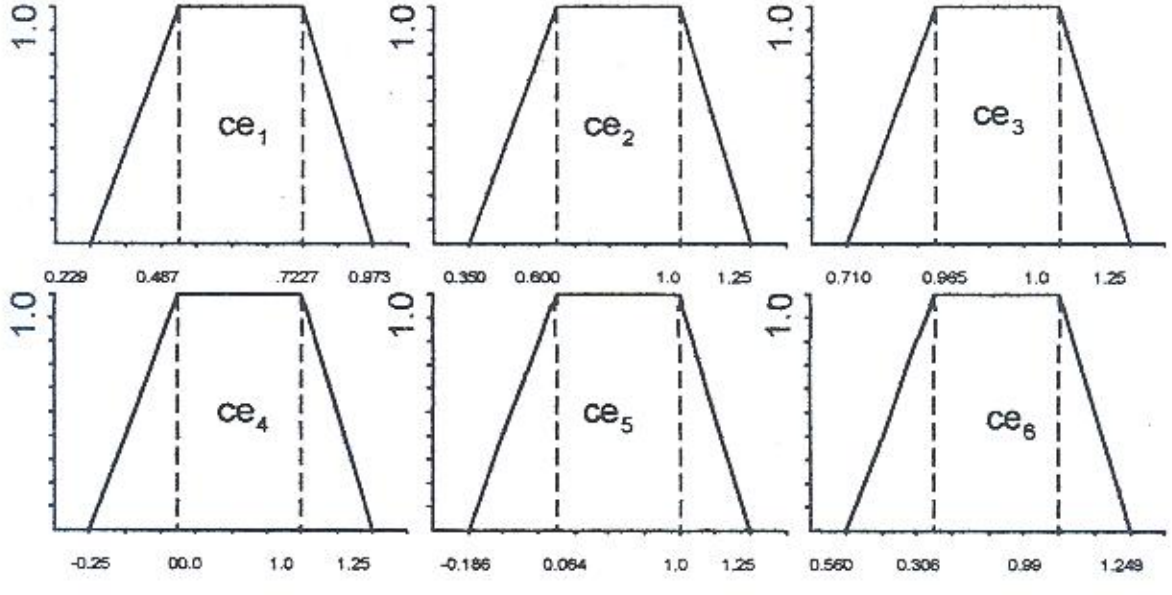
a)



b)

El sistema FALCON-ART forma tambien 6 funciones de membresía para el voltaje de campo, de acuerdo a la Fig. 4.5. Las figuras 4.10 a) y b) muestran la distribución de las funciones de membresía en el Universo de discurso de la variable.

También se proporcionan los valores donde se ubican los límites de las funciones de membresía para esta variable.

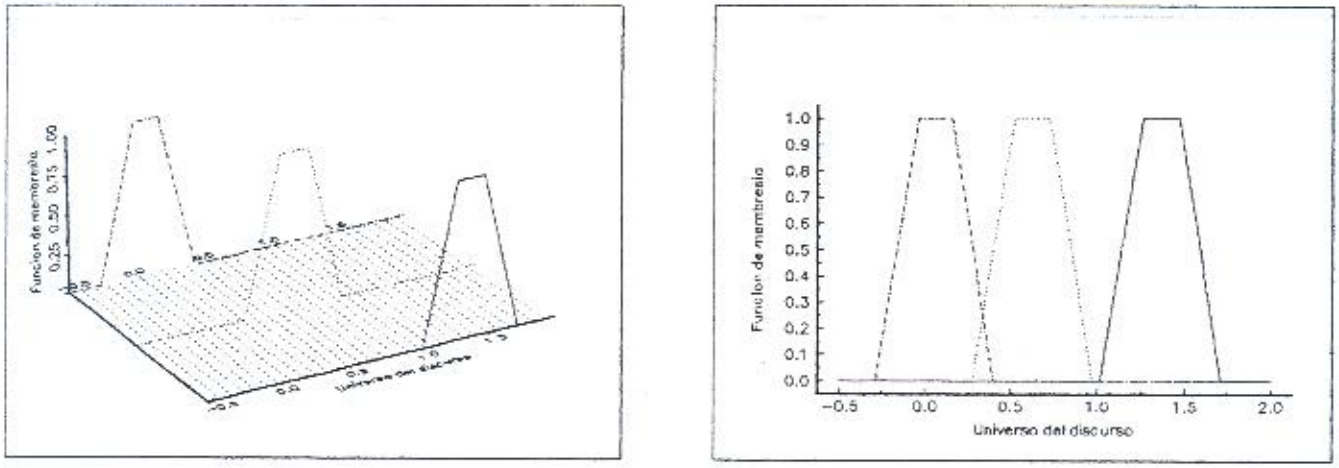


c)

Figura 4. 11 Funciones de membresía para el voltaje de campo en el espacio de entrada: a) Vista 3D, b) Vista XY, c) Dominio de las funciones de membresía.

4.9. 2 Espacio de salida.

4.9.2. 1 Voltaje de campo.



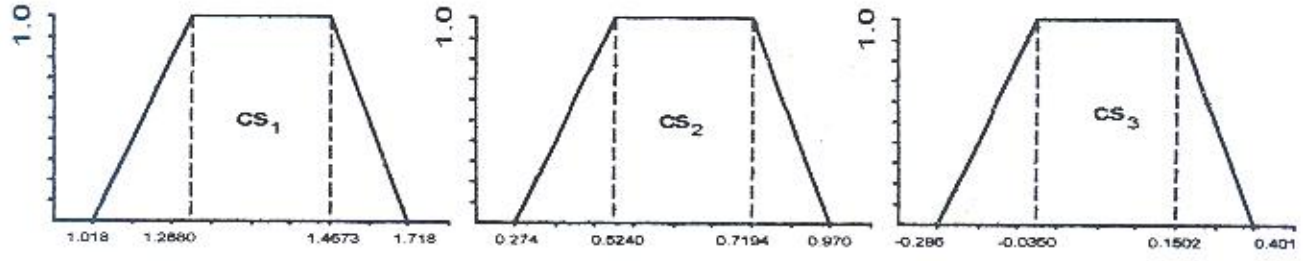
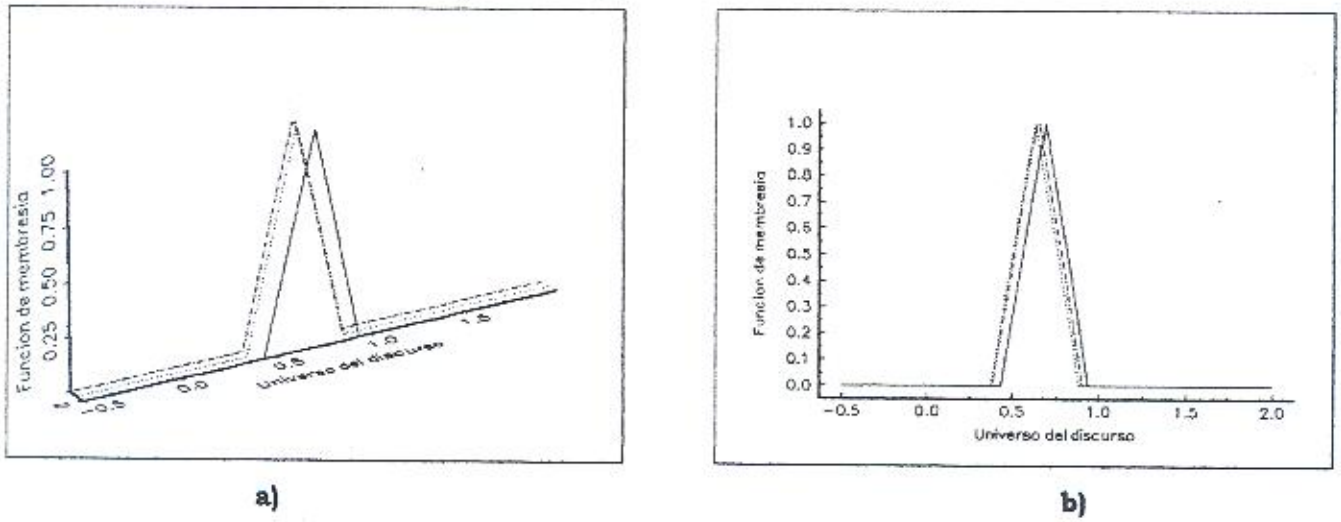
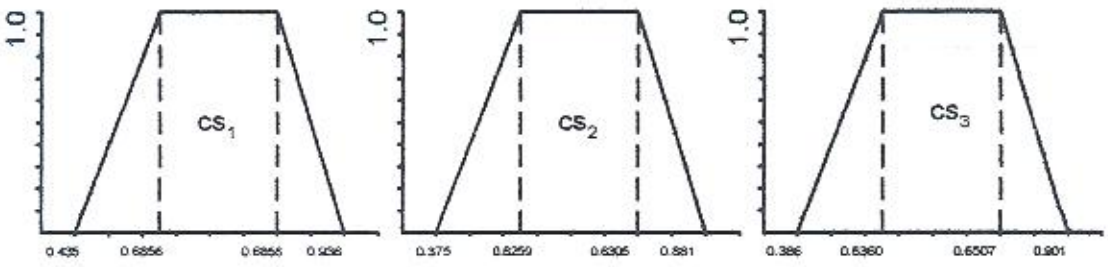


Figura 4. 12 Funciones de membresía para el voltaje de campo en el espacio de salida: a) Vista 3D, b) Vista XY, c) Dominio de las funciones de membresía.

4.9.2. 2 Error del voltaje de campo.





c)

Figura 4. 13 Funciones de membresía para el error del voltaje de campo en el espacio de salida: a) Vista 3D, b) Vista XY, c) Dominio de las funciones de membresía.

Para el espacio de salida, donde se codifican los patrones de las variables voltaje de campo y error del voltaje de campo, se forma solo tres funciones de membresía para cada variable. Las figuras 4.12 y 4.13 permiten apreciar la distribución de estas variables en sus Universos respectivos.

4. 10 Aplicación de la estructura FALCON-ART en la regulación de voltaje de un generador síncrono.

Para la aplicación del sistema FALCON-ART se tomaron dos conjuntos de condiciones iniciales, el primero de ellos, Tabla 4.1, se probó para fallas de 6 y 8 ciclos de duración. La falla se aplicó al bus en un 1 seg. después de haber iniciado la simulación, y se liberó una vez cumplido el tiempo de falla. Las pruebas realizadas incluyen la comparación del sistema de regulación FALCON-ART con un sistema de regulación ST1 para evaluar su índice de desempeño. Para la prueba de 6 y 8 ciclos se grafican: el voltaje terminal, el voltaje de campo o excitación, la potencia activa y reactiva, el ángulo de la máquina, el índice de desempeño del voltaje terminal y el índice de desempeño del ángulo de la máquina.

El segundo conjunto de condiciones iniciales, se probó para una falla con una duración de 18 ciclos. En este grupo de condiciones iniciales, se intenta mostrar los limites de regulación del sistema FALCON-ART. Para cada una de las condiciones iniciales se grafica únicamente las variables: el voltaje terminal, el voltaje de campo

variables: el voltaje terminal, el voltaje de campo o excitación, la potencia activa y reactiva, y el ángulo de la máquina. Se excluye la graficación de los índices de desempeño de este grupo, ya que el sistema de regulación ST1, deja de regular en la mayoría de condiciones iniciales a partir de fallas con un tiempo de duración de 12 ciclos.

A continuación se presentan las tablas que contienen los grupos de condiciones iniciales probados. La tabla 4.1 contiene las condiciones iniciales para una falla de 6 y 8 ciclos de duración, y la tabla 4.2 contiene las condiciones iniciales para una falla de duración de 18 ciclos.

Tabla 4. 1 Condiciones iniciales simuladas para fallas de 6 y 8 ciclos de duración

*No.	Voltaje Terminal	Ángulo en terminales	Potencia Activa	Potencia activa
1	.1000000e+01	.6000000e+01	.8805000e+00	-.1361000e+00
2	.1000000e+01	.6807718e+01	.1000000e+01	-.1473878e+00
15	.9896917E+00	.6534585E+01	.9319504E+00	-.2272967E+00
20	.9904293E+00	.6707360E+01	.9592349E+00	-.2236688E+00
27	.1010917E+01	.6016429E+01	.9117899E+00	-.4590218E-00

Tabla 4. 2 Condiciones iniciales simuladas para una falla de 18 ciclos de duración.

*No.	Voltaje Terminal	Ángulo en terminales	Potencia Activa	Potencia activa
144	.1001754E+01	.6535971E+01	.9645268E+00	-.1294009E+00
145	.9823618E+00	.3885064E+01	.5284195E+00	-.2413188E+00
191	.1051076E+01	.2873440E+01	.5347080E+00	.3667168E+00
192	.1023007E+01	.3378494E+01	.5463802E+00	.1060993E+00
197	.1038097E+01	.6002964E+01	.9833057E+00	.1881852E+00

*La numeración de las tablas, corresponde a la número consecutivo que tienen las condiciones iniciales en el apéndice D.

4.10. 1 CONDICIONES INICIALES CON UNA DURACIÓN DE 6 CICLOS.

4.10.1. 1 Simulación de la condición inicial No. 1.

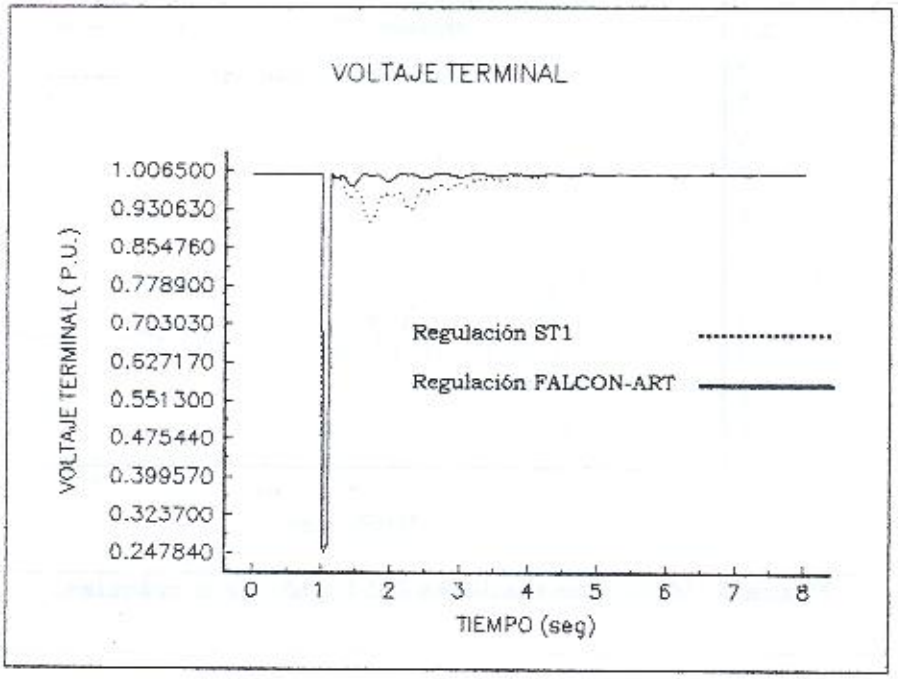


Figura 4. 14 Comportamiento del voltaje terminal con valor inicial de 1.0 p.u.

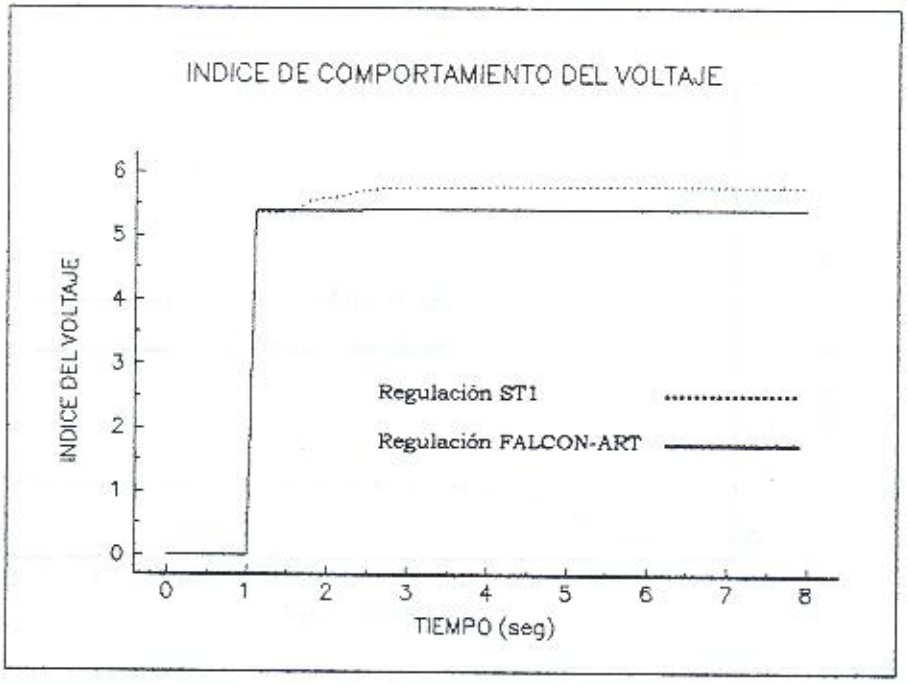


Figura 4. 15 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1.

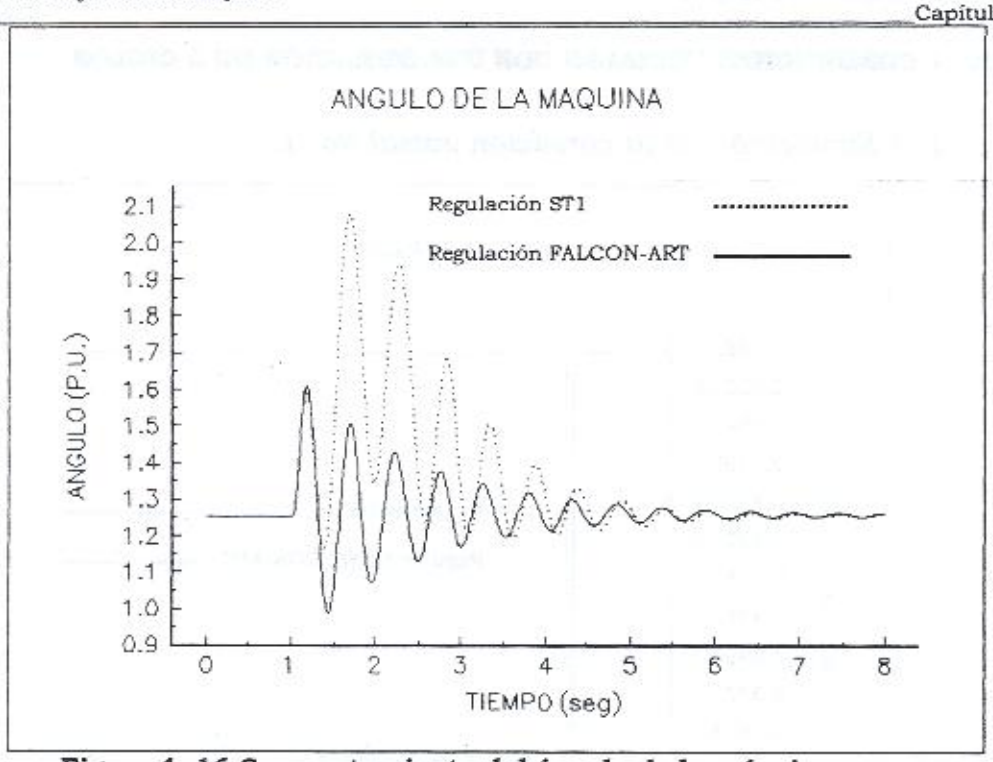


Figura 4. 16 Comportamiento del ángulo de la máquina.

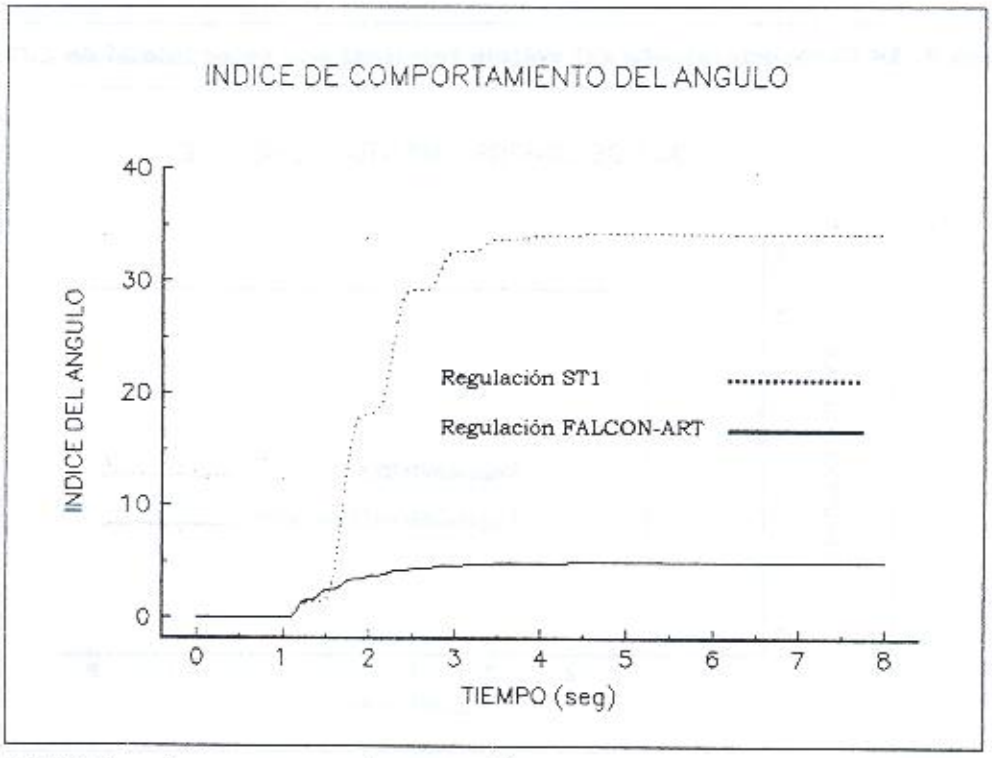


Figura 4. 17 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

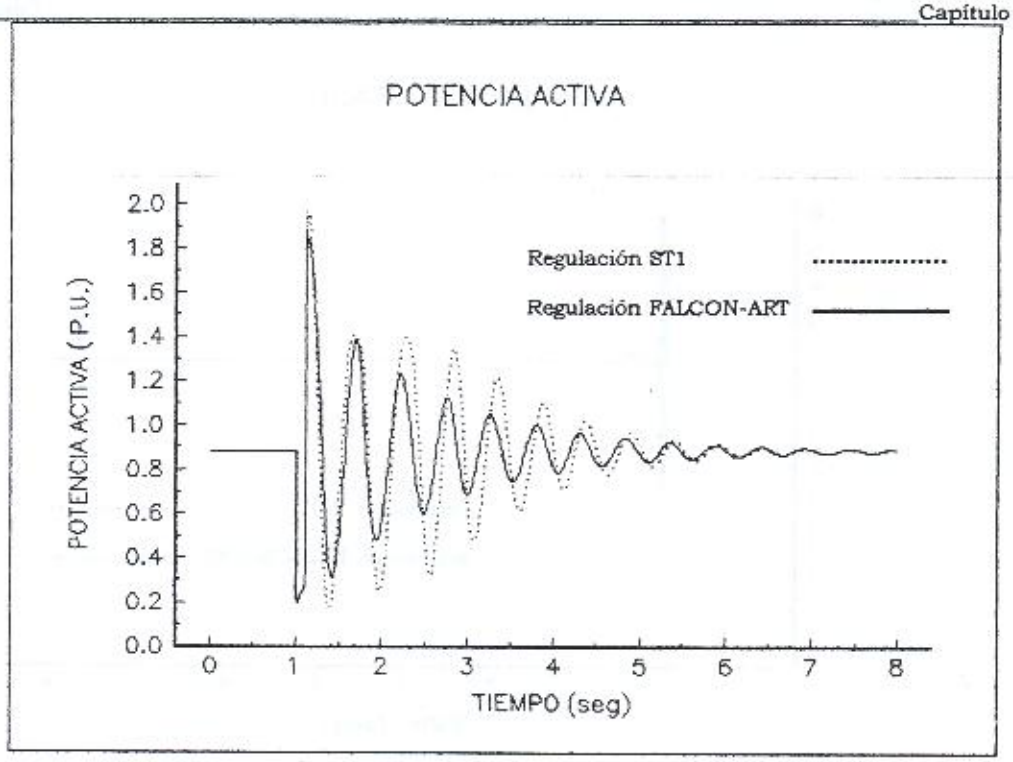


Figura 4. 18 Comportamiento de la potencia activa con valor inicial de 0.8805 p.u.

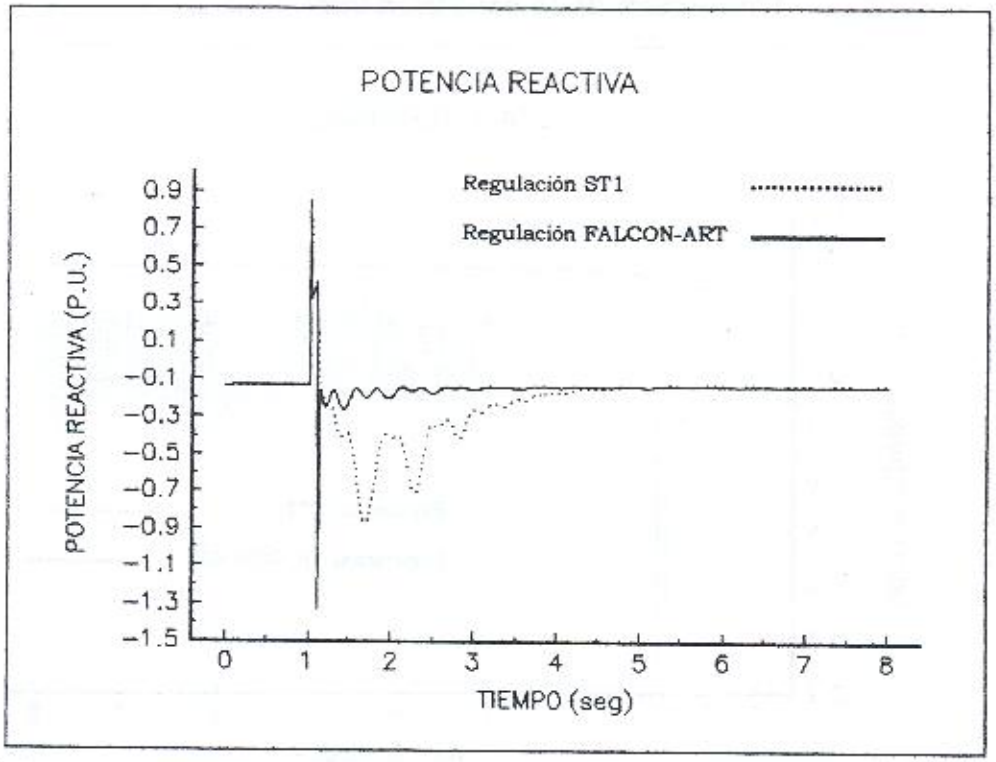


Figura 4. 19 Comportamiento de la potencia reactiva con un valor inicial de −.1361 p.u.

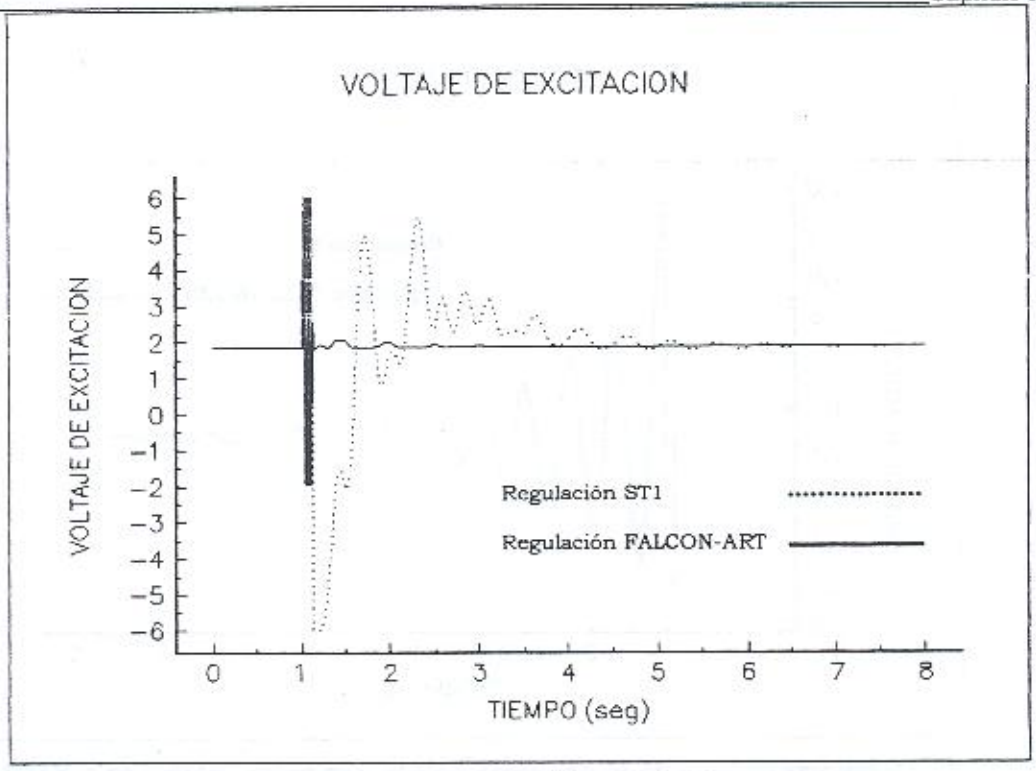


Figura 4. 20 Comportamiento del voltaje de campo.

4.10.1. 2 Simulación de la condición inicial No. 2.

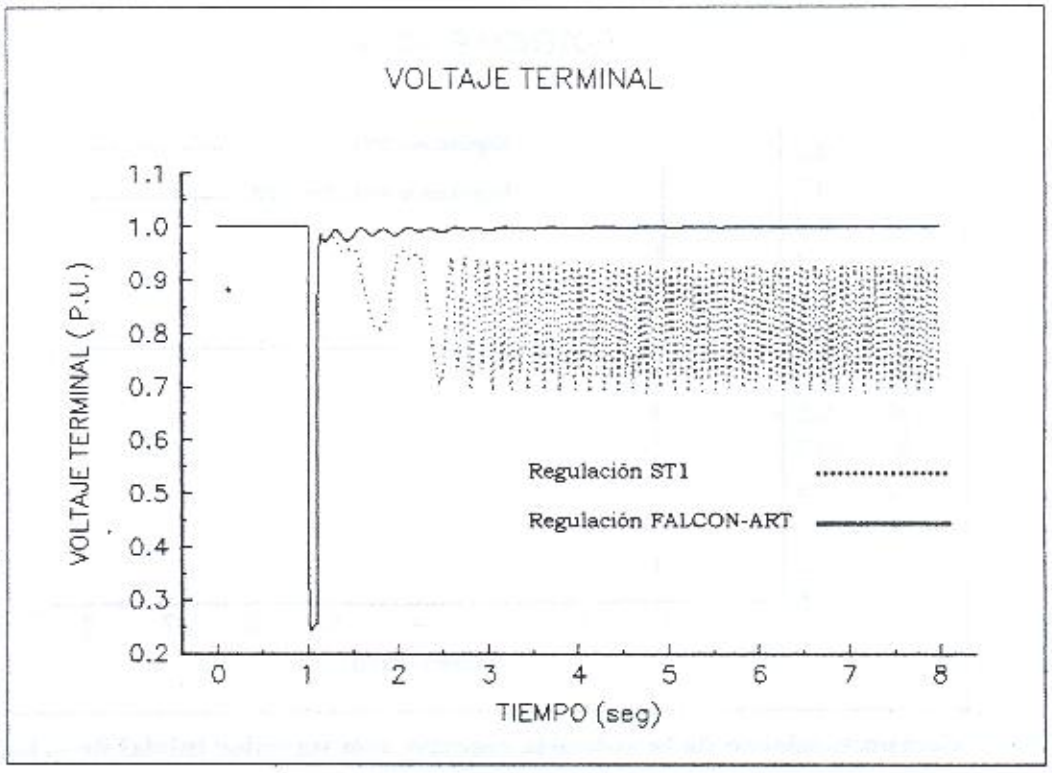


Figura 4. 21 Comportamiento del voltaje terminal con valor inicial de 1.0 p.u.

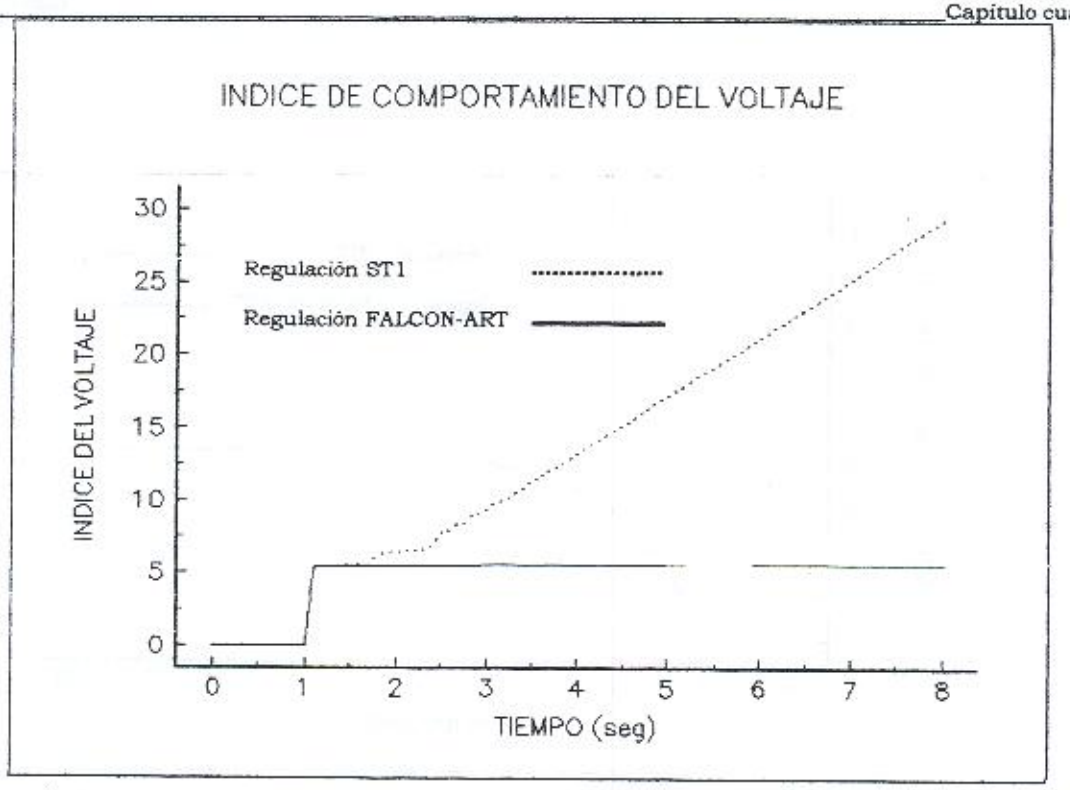


Figura 4. 22 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1

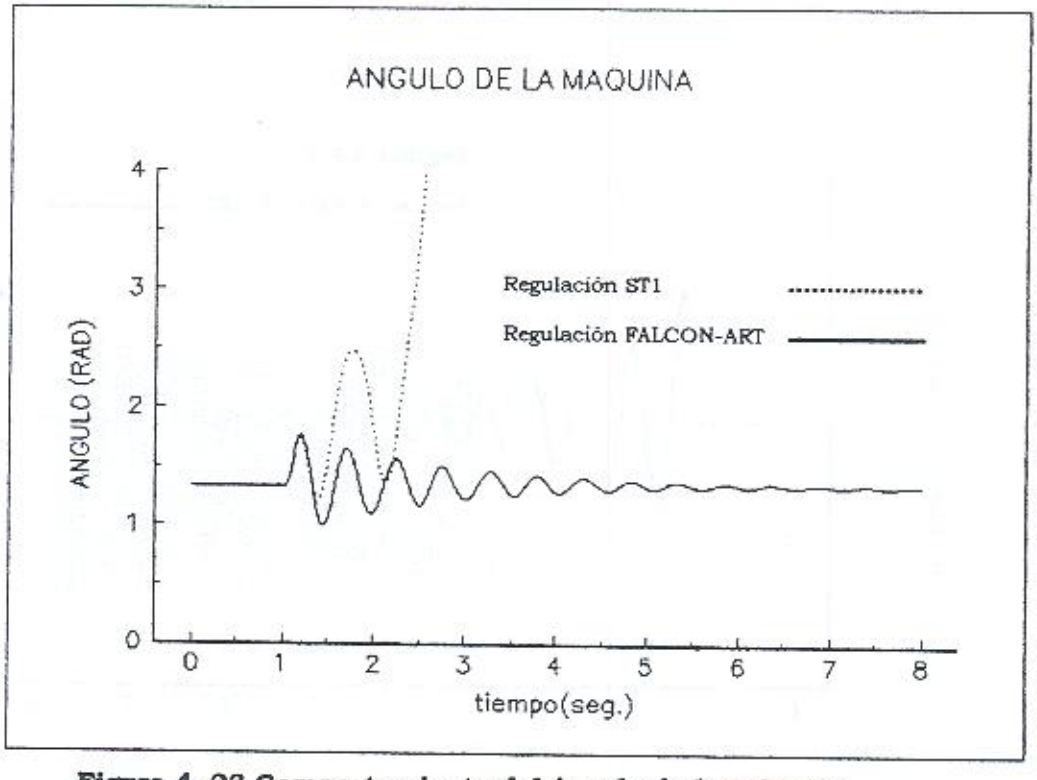


Figura 4. 23 Comportamiento del ángulo de la máquina.

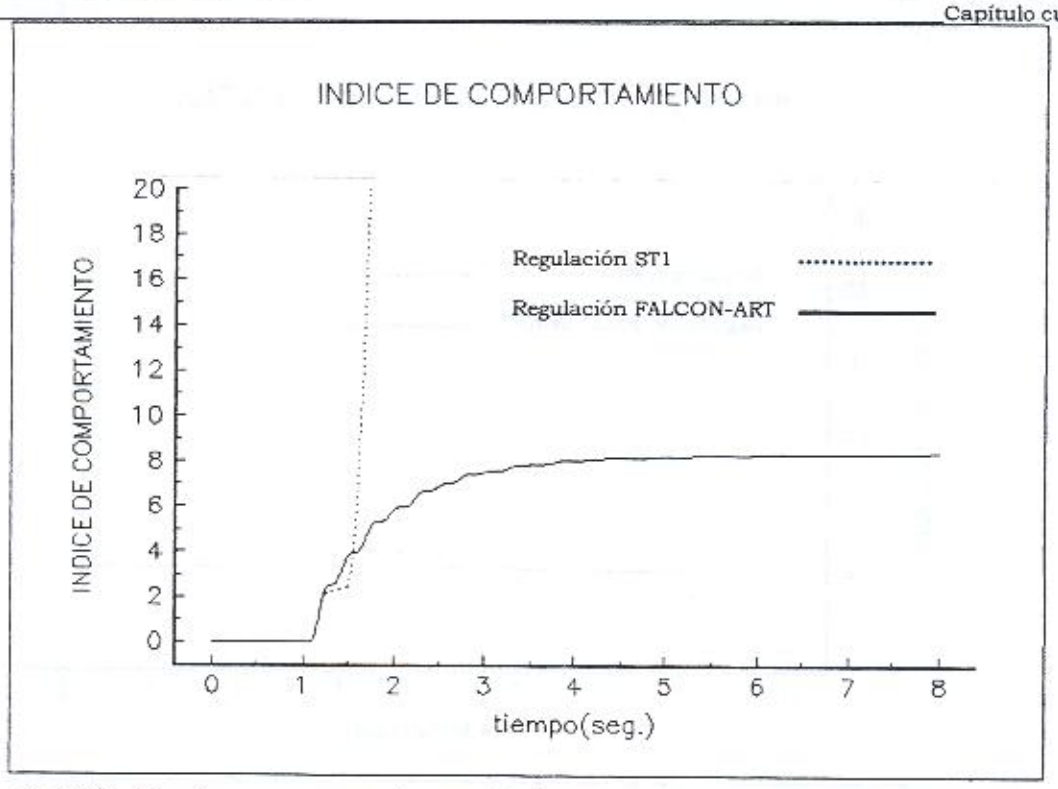


Figura 4. 24 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

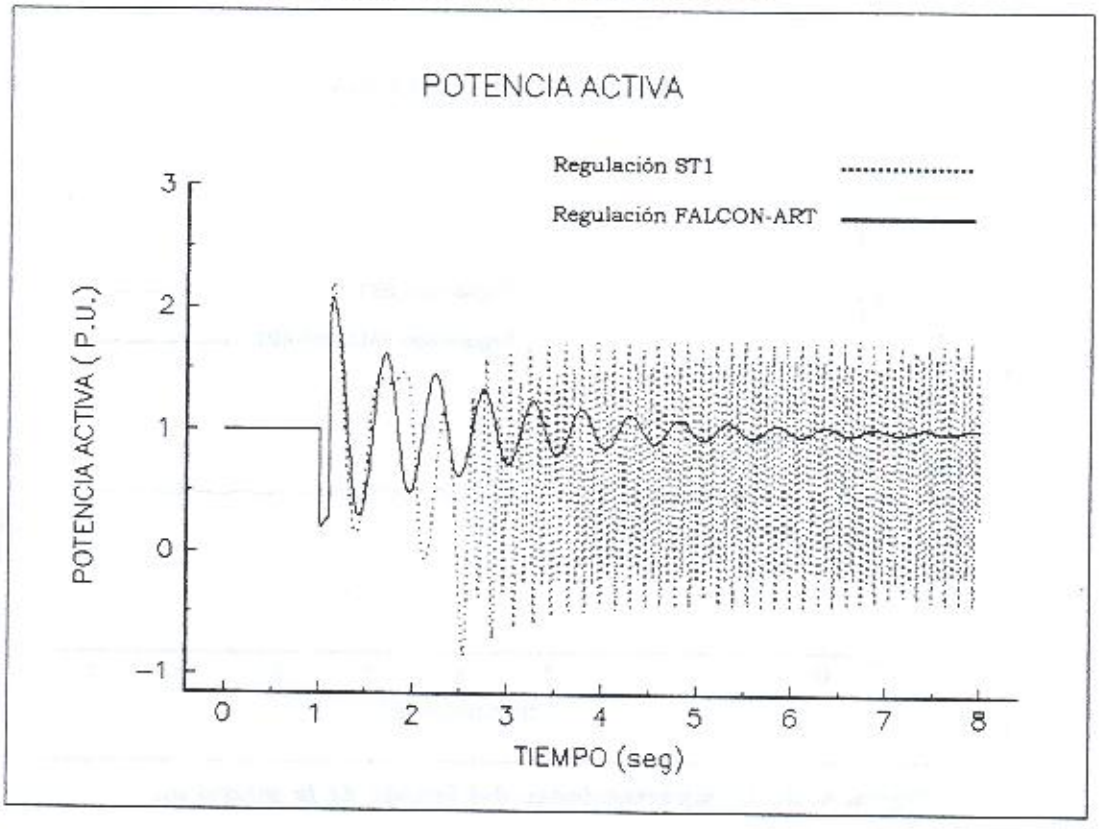


Figura 4. 25 Comportamiento de la potencia activa con valor inicial de 1.0 p.u.

4.10.1. 3 Simulación de la condición inicial No. 15.

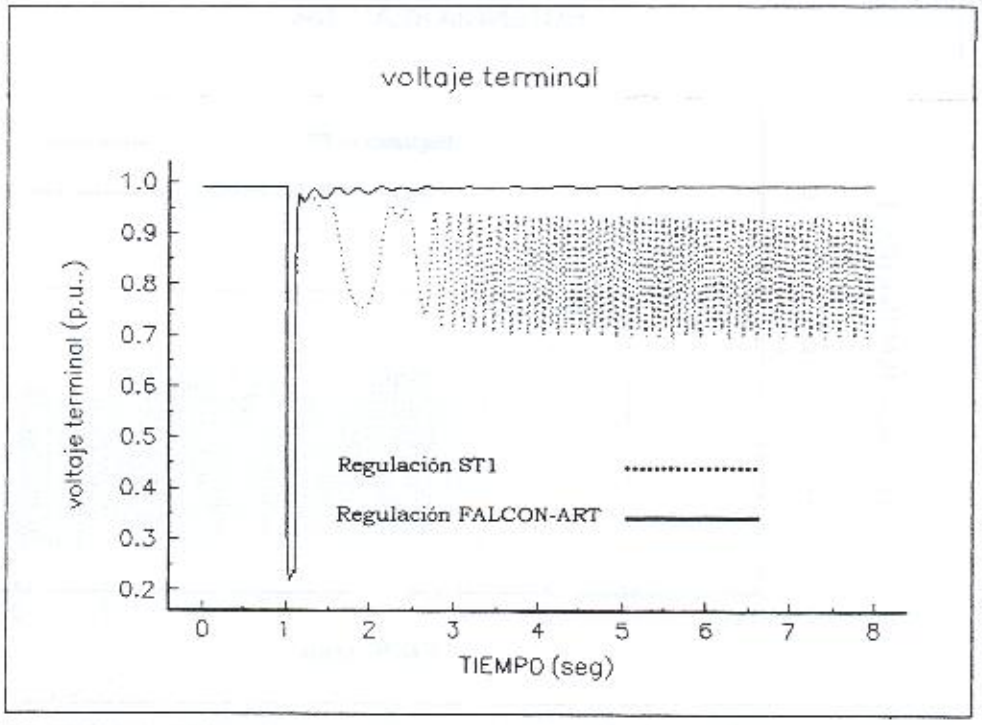


Figura 4. 28 Comportamiento del voltaje terminal con valor inicial de .989691 p.u.

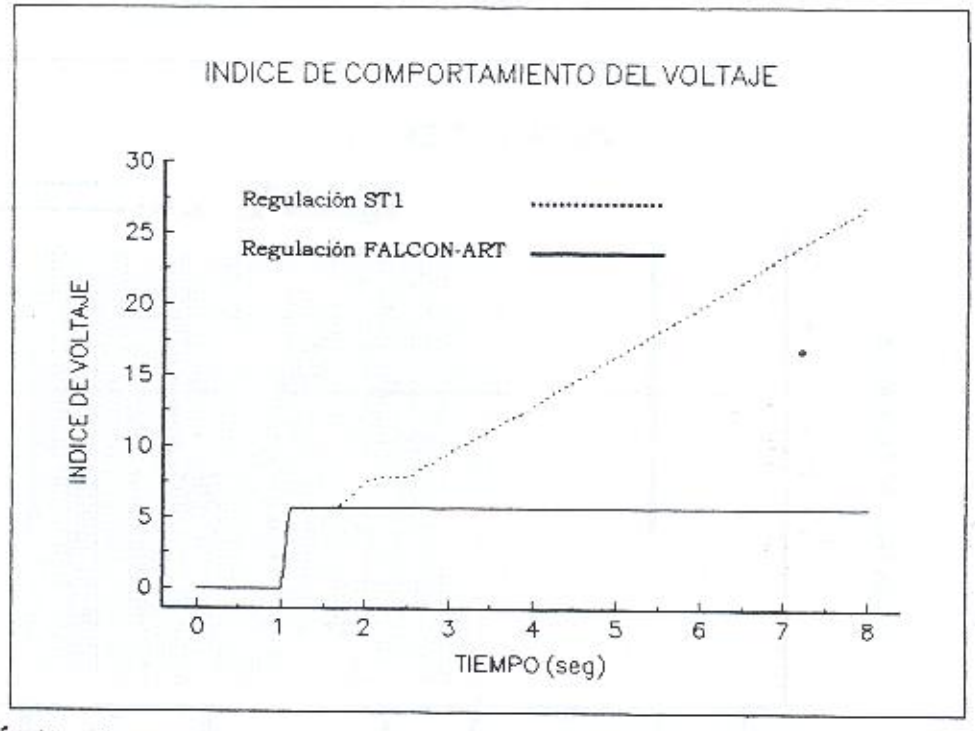


Figura 4. 29 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1

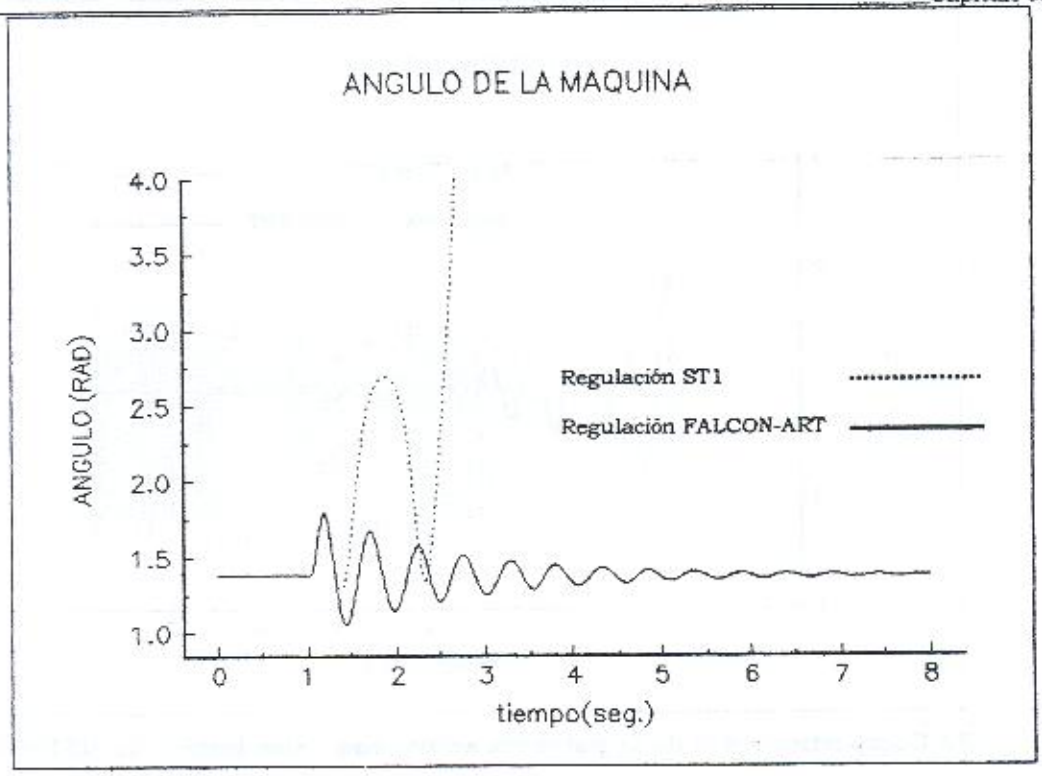


Figura 4. 30 Comportamiento del ángulo de la máquina.

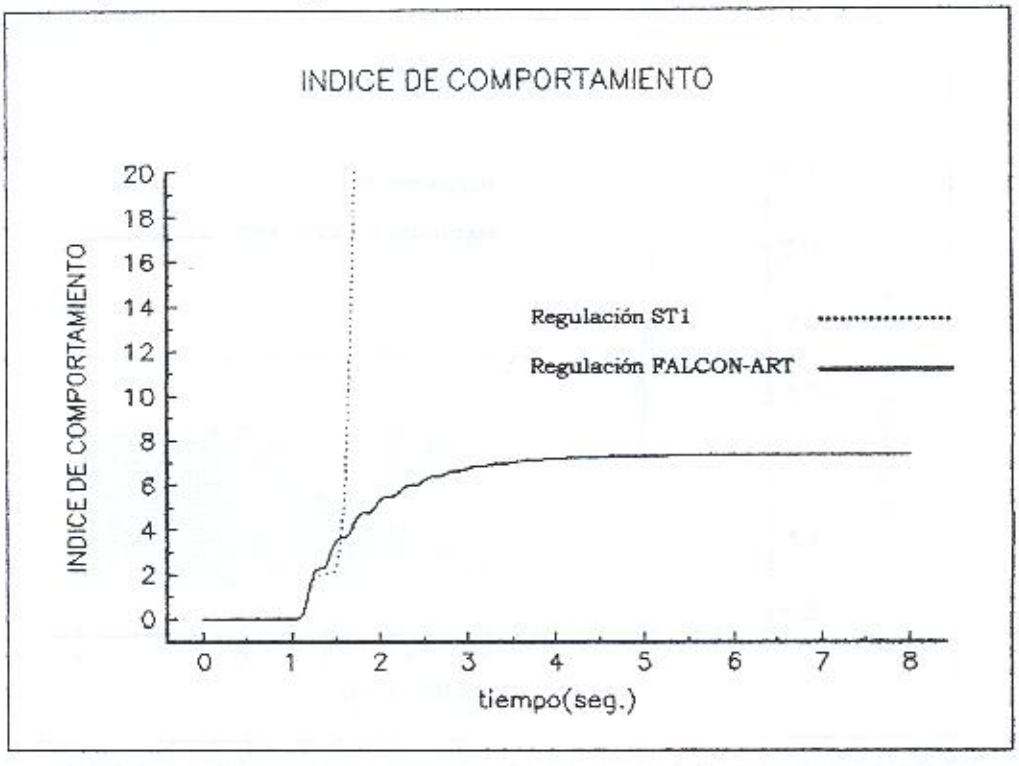


Figura 4. 31 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

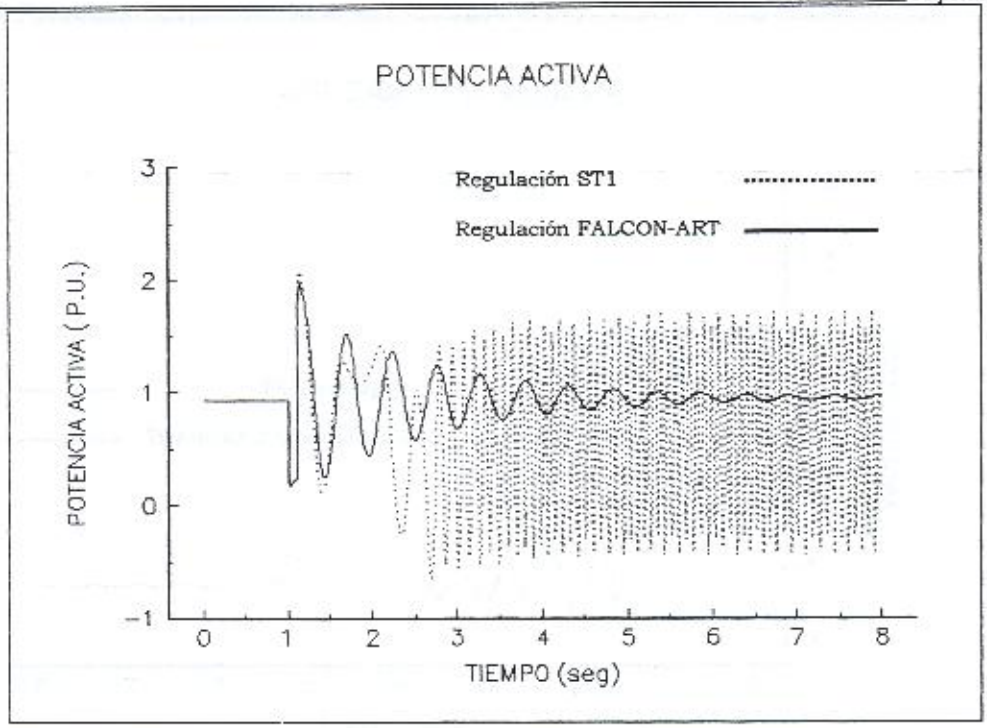


Figura 4. 32 Comportamiento de la potencia activa con valor inicial de .931950 p.u.

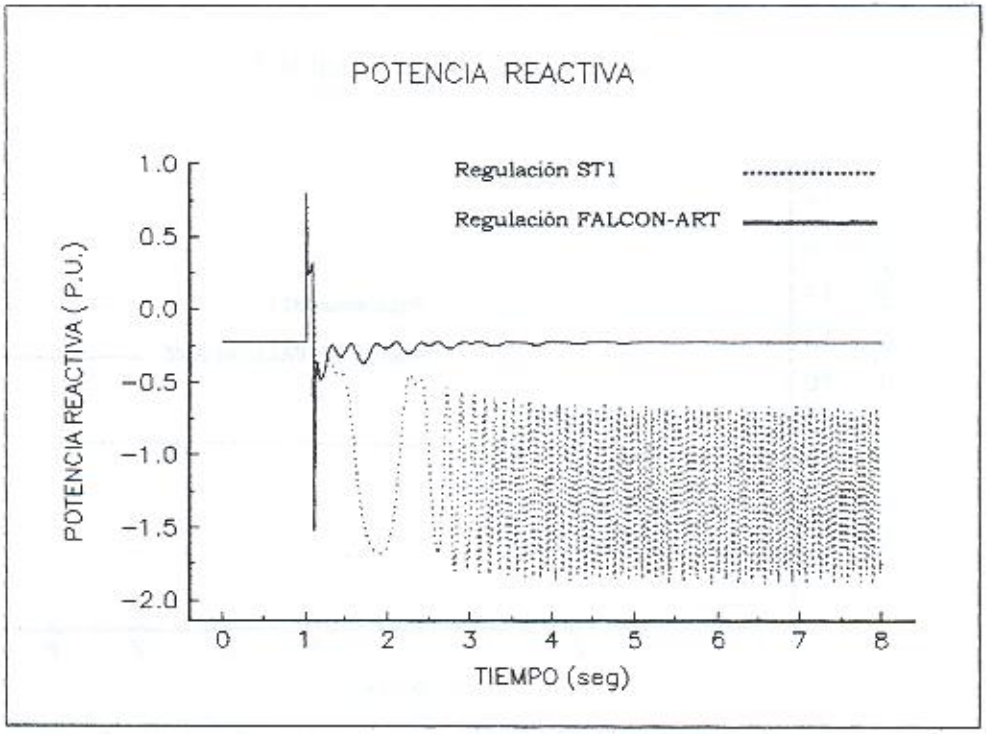


Figura 4. 33 Comportamiento de la potencia reactiva con un valor inicial de -.227296 p.u.

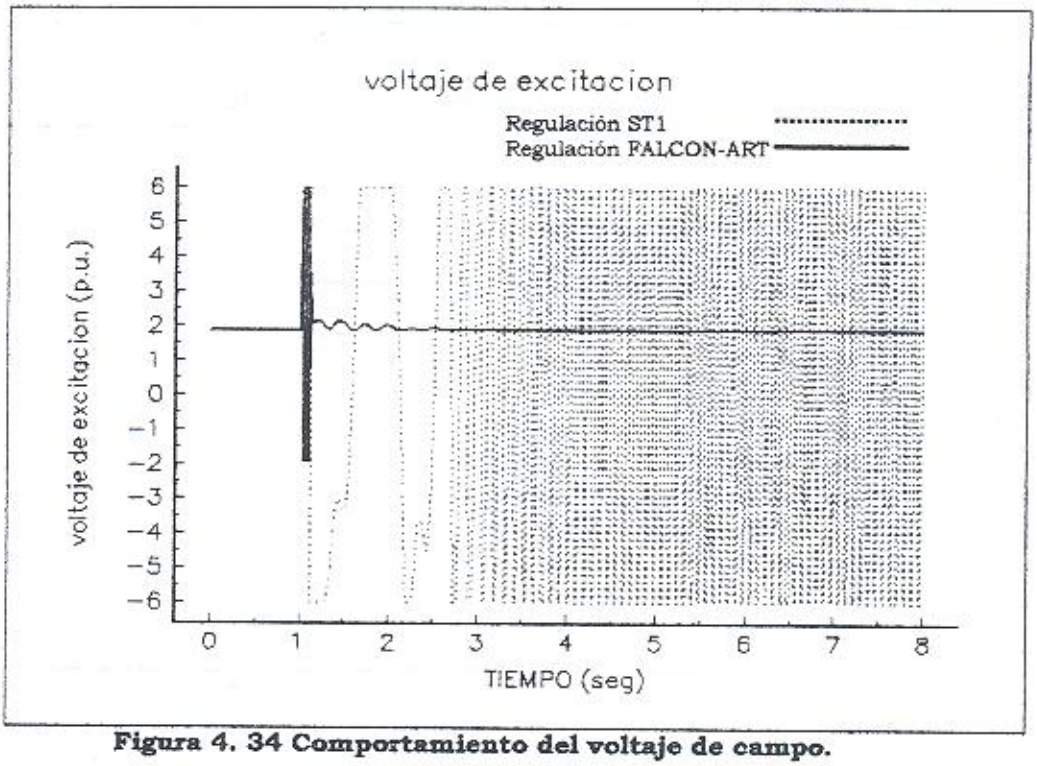


Figura 4. 34 Comportamiento del voltaje de campo.

4.10.1. 4 Simulación de la condición inicial No. 20.

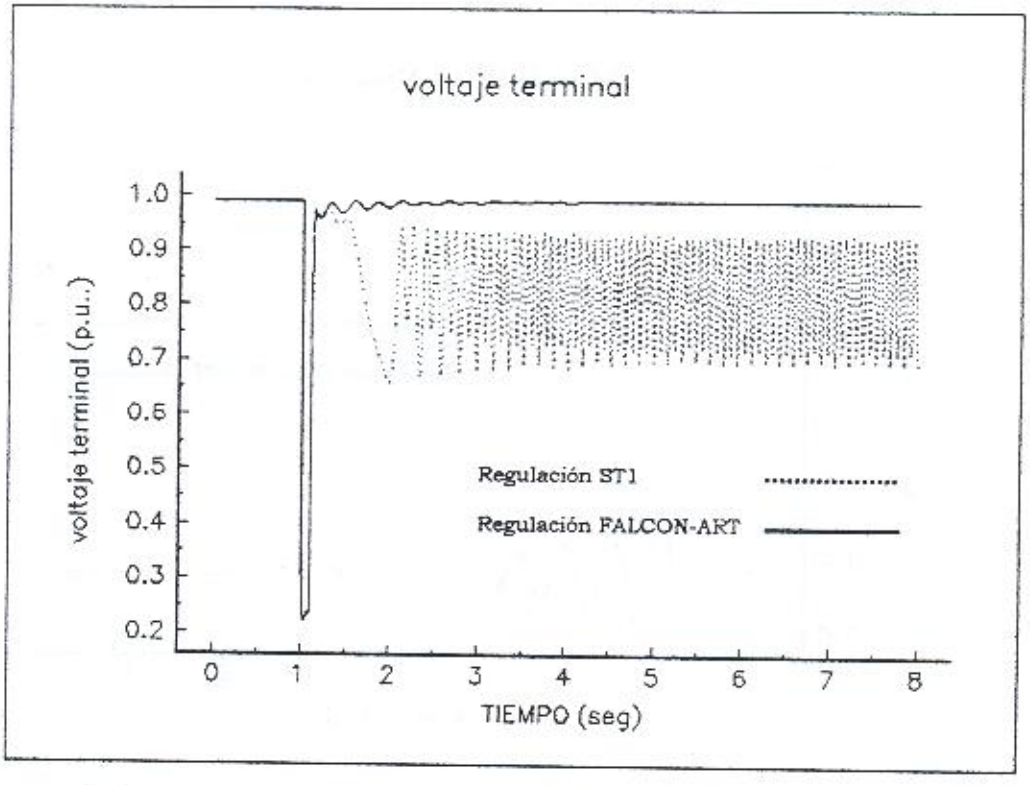


Figura 4. 35 Comportamiento del voltaje terminal con valor inicial de 0.9904

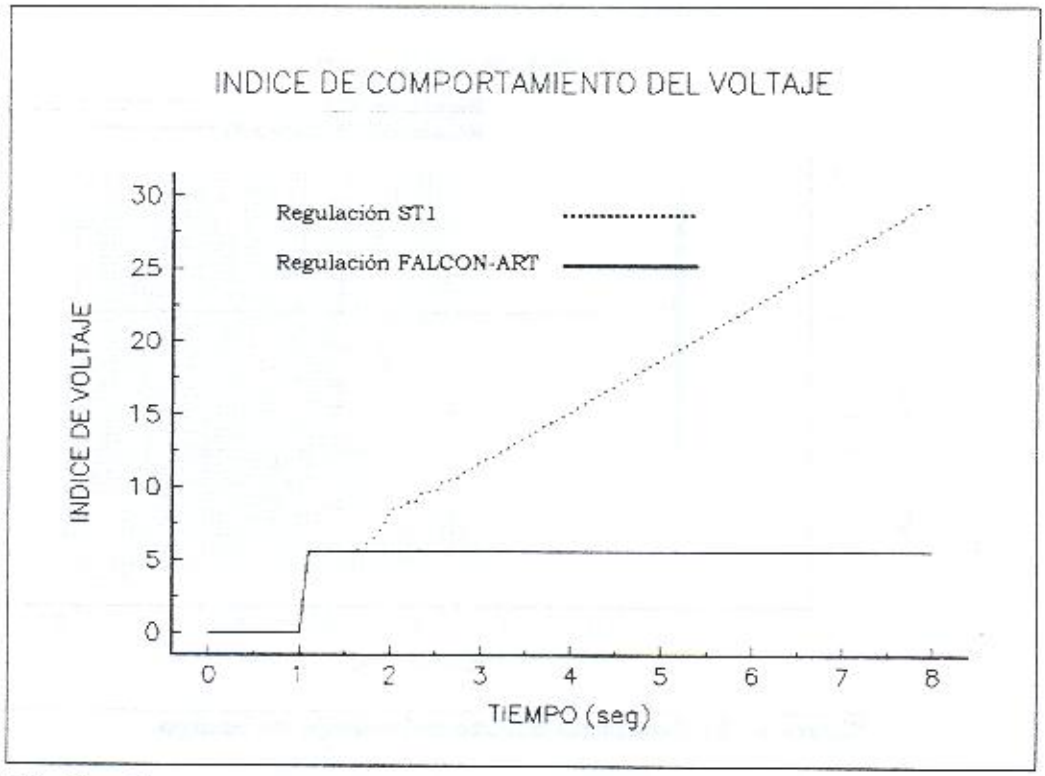


Figura 4. 36 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1

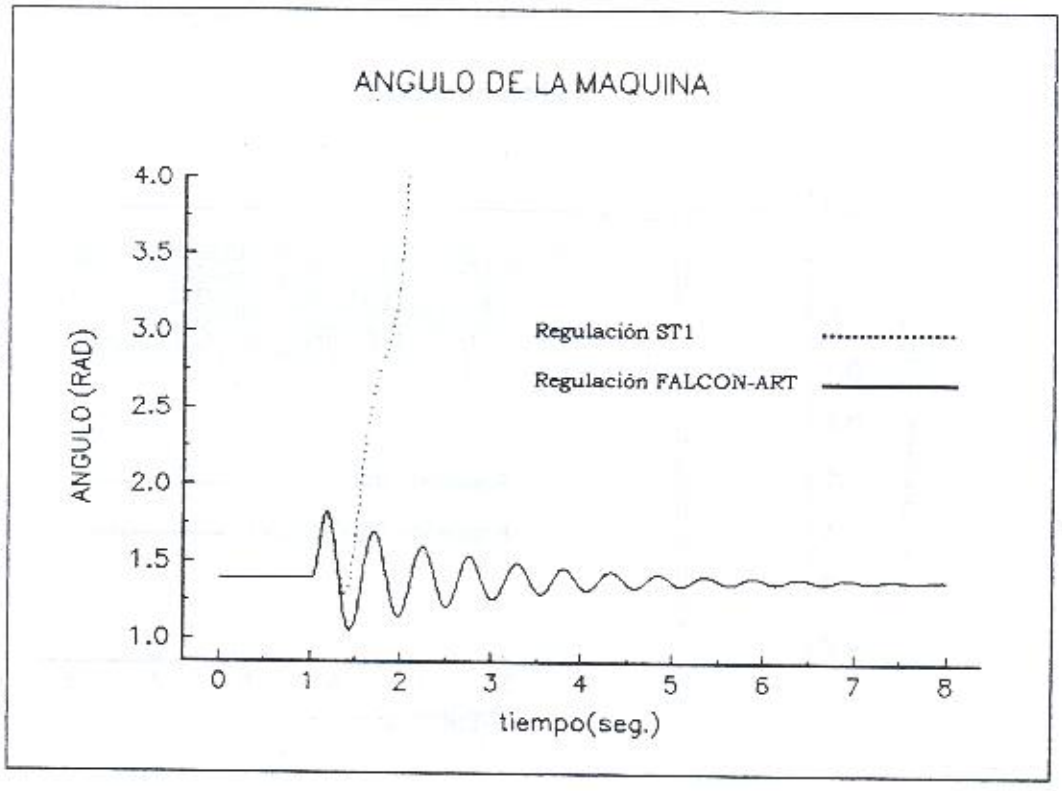


Figura 4. 37 Comportamiento del ángulo de la máquina.

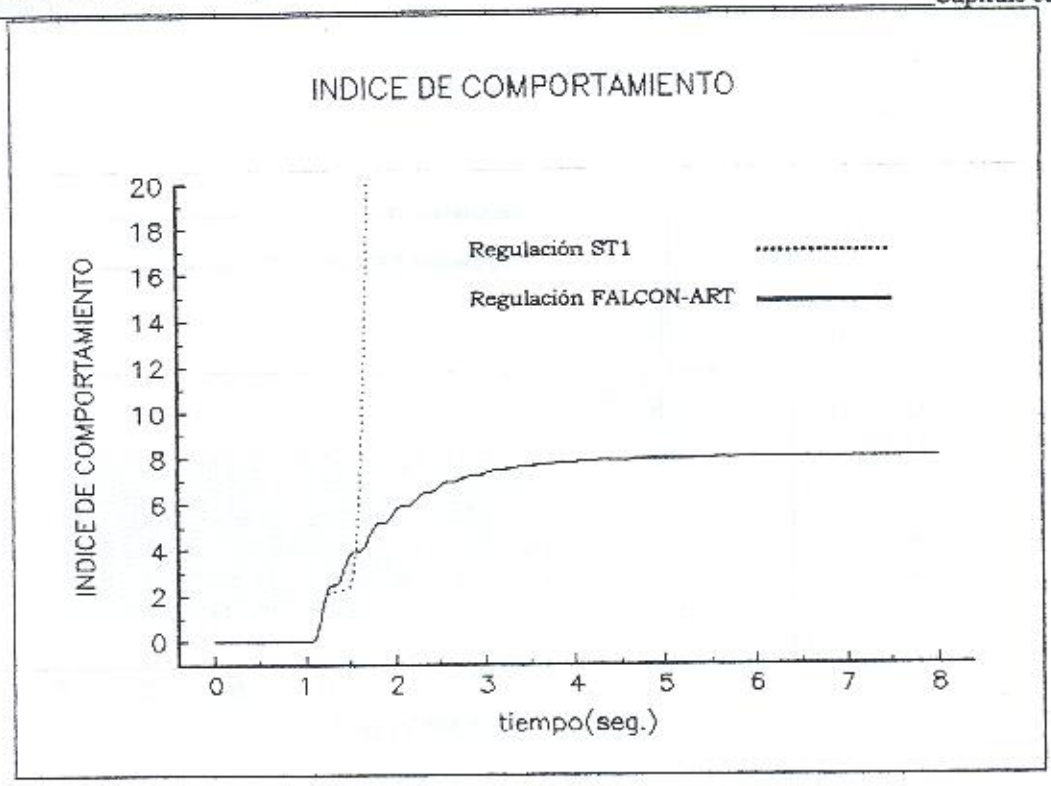


Figura 4. 38 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

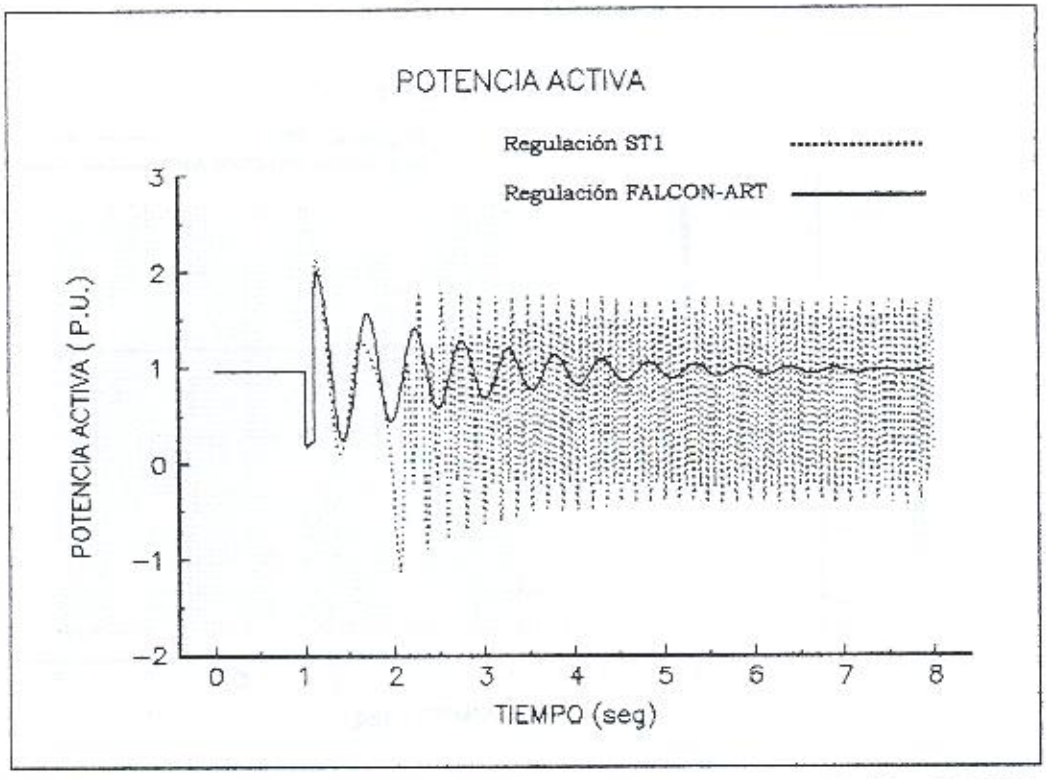


Figura 4. 39 Comportamiento de la potencia activa con valor inicial de .959234

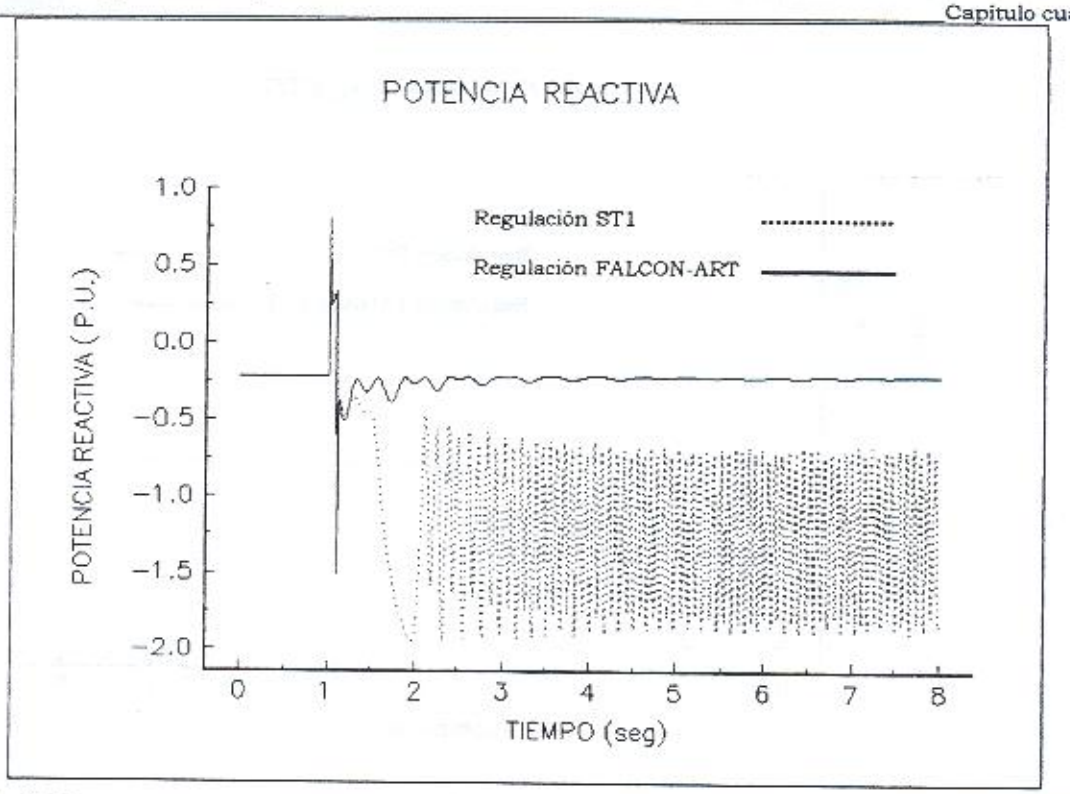


Figura 4. 40 Comportamiento de la potencia reactiva con un valor inicial de –.223668

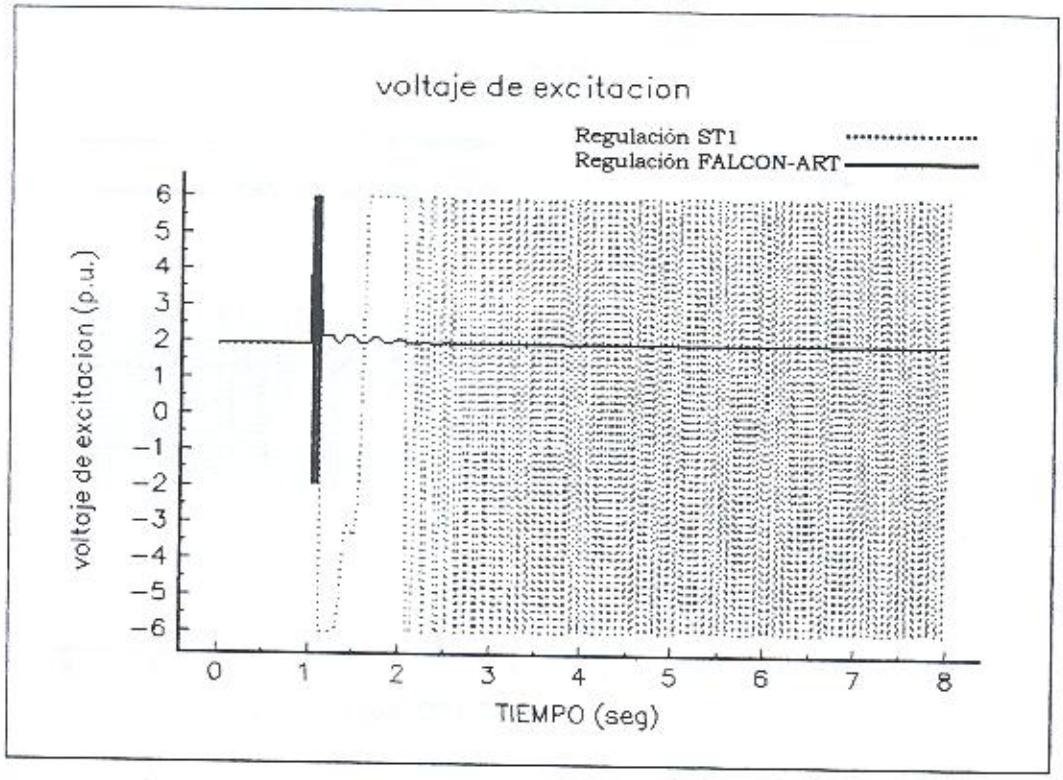


Figura 4. 41 Comportamiento del voltaje de campo.

4.10.1. 5 Simulación de la condición inicial No. 27.

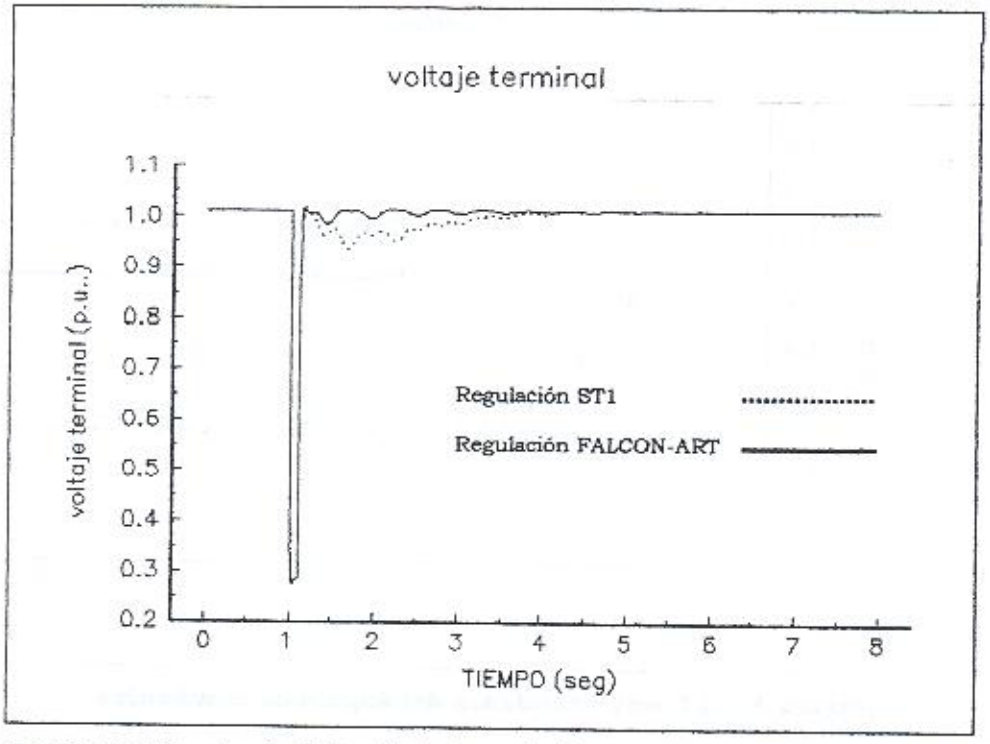


Figura 4. 42 Comportamiento del voltaje terminal con valor inicial de 1.0109 p.u.

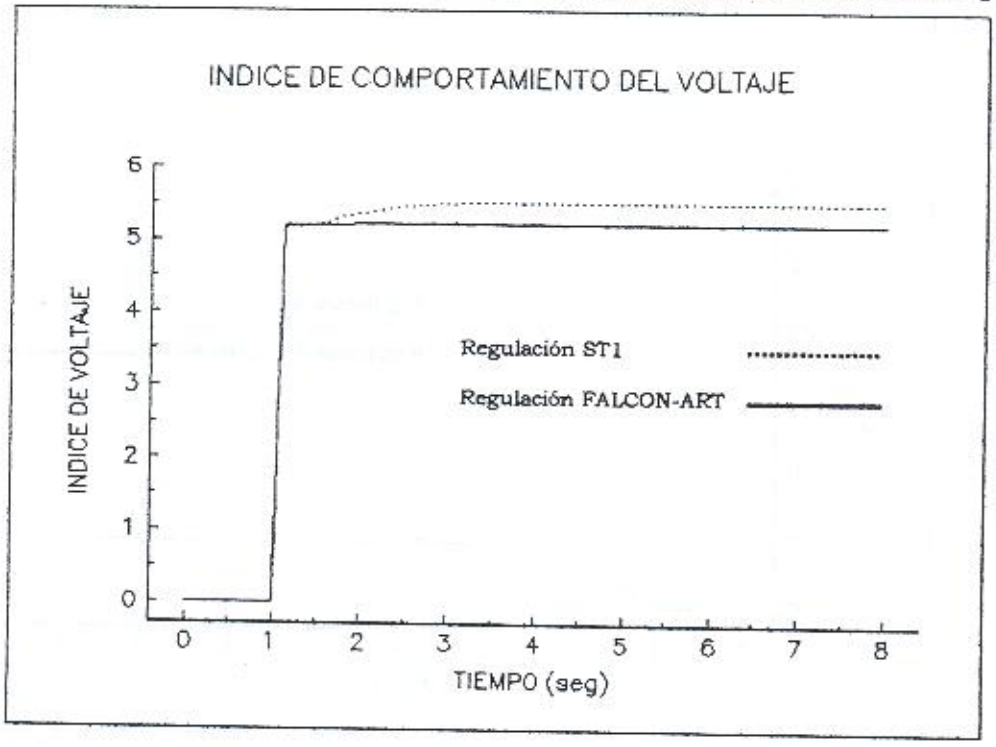


Figura 4. 43 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1

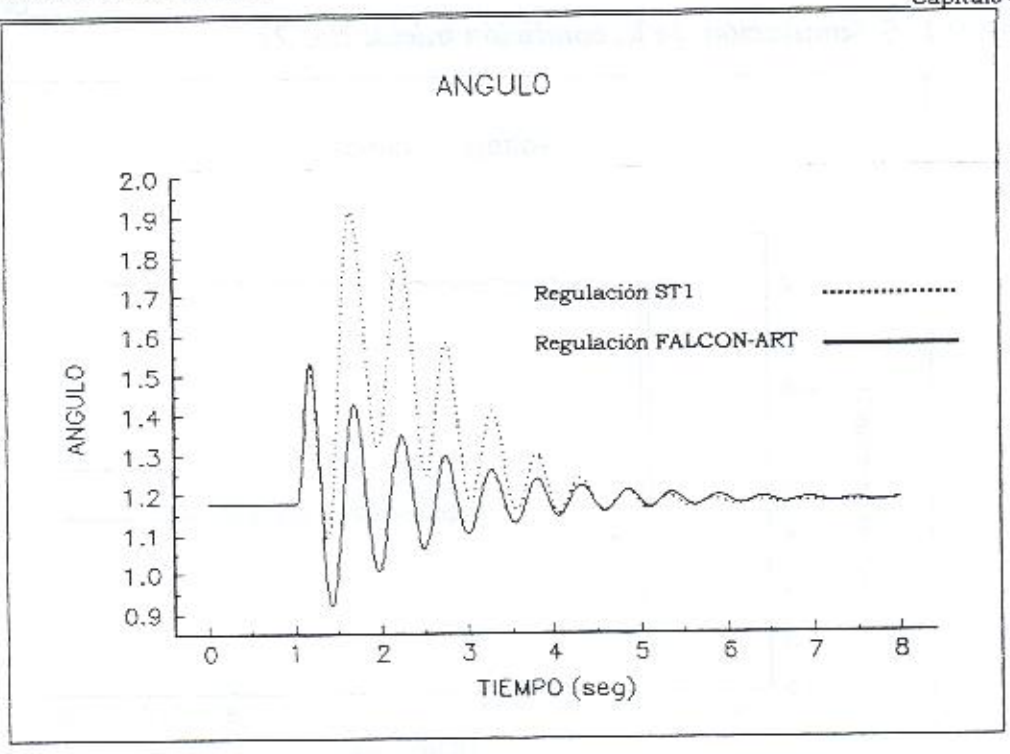


Figura 4. 44 Comportamiento del ángulo de la máquina.

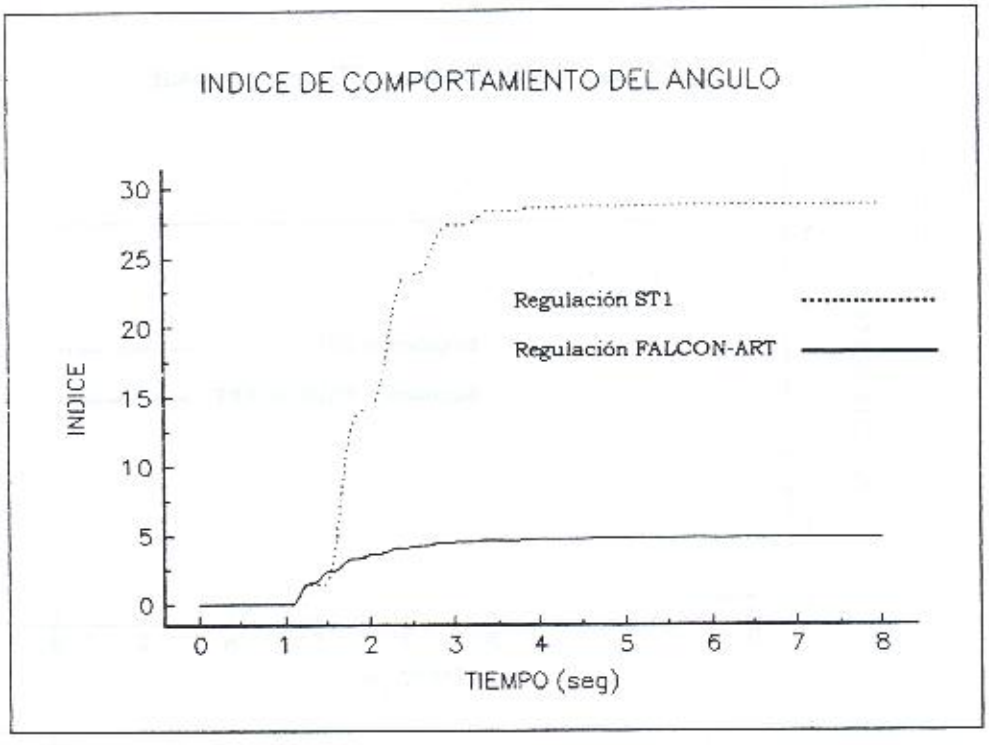


Figura 4. 45 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

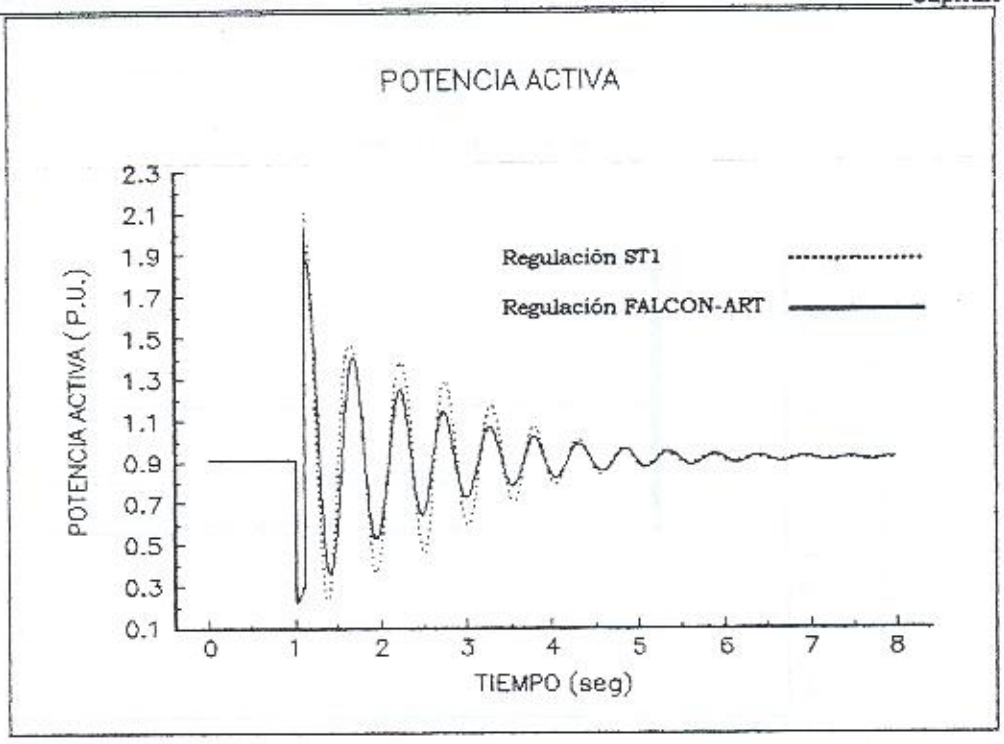


Figura 4. 46 Comportamiento de la potencia activa con valor inicial de .911789 p.u.

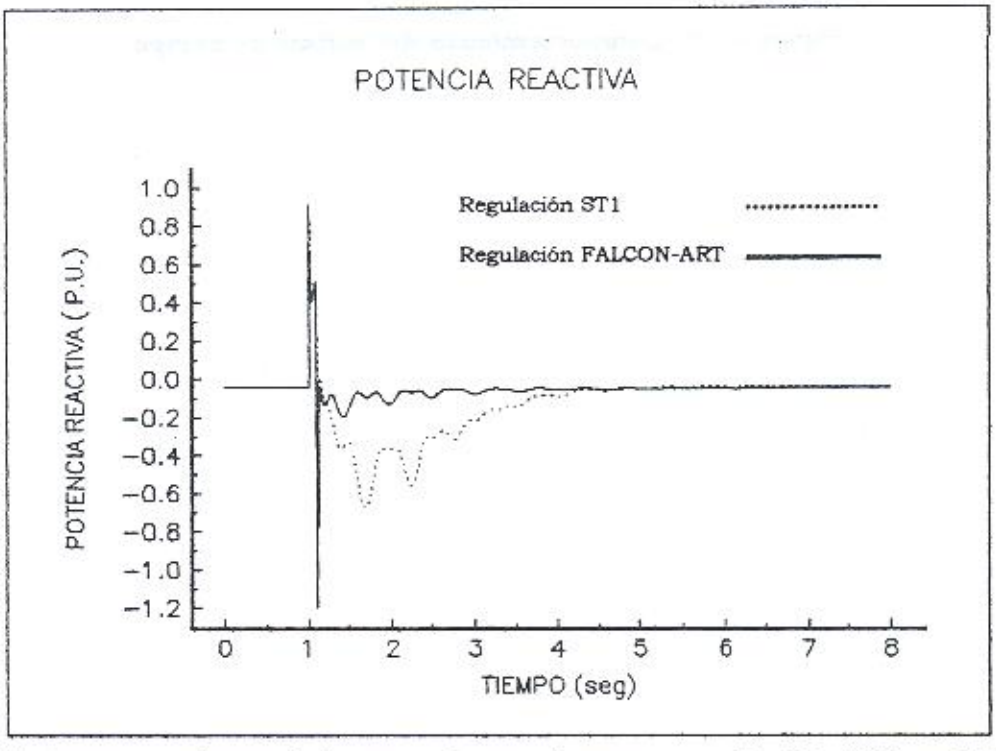


Figura 4. 47 Comportamiento de la potencia reactiva con un valor inicial de −.459021 p.u.

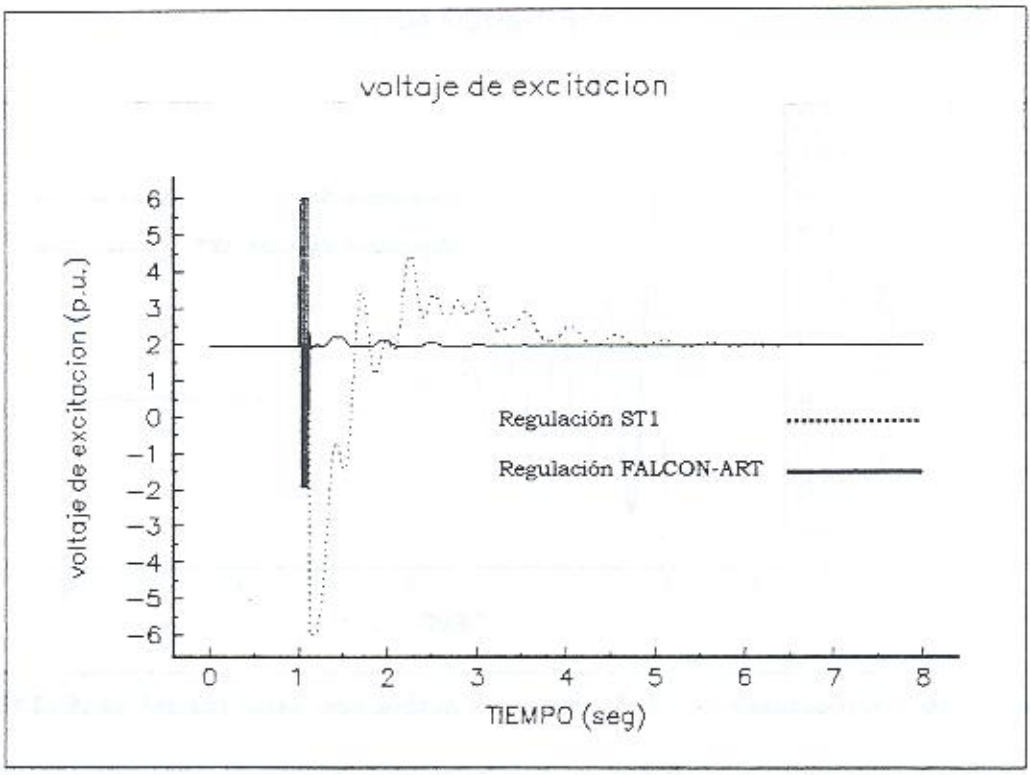


Figura 4. 48 Comportamiento del voltaje de campo.

4.10.2 CONDICIONES INICIALES CON UNA FALLA DE 8 CICLOS DE DURACIÓN.

4.10.2. 1 Simulación de la condición inicial No.1.

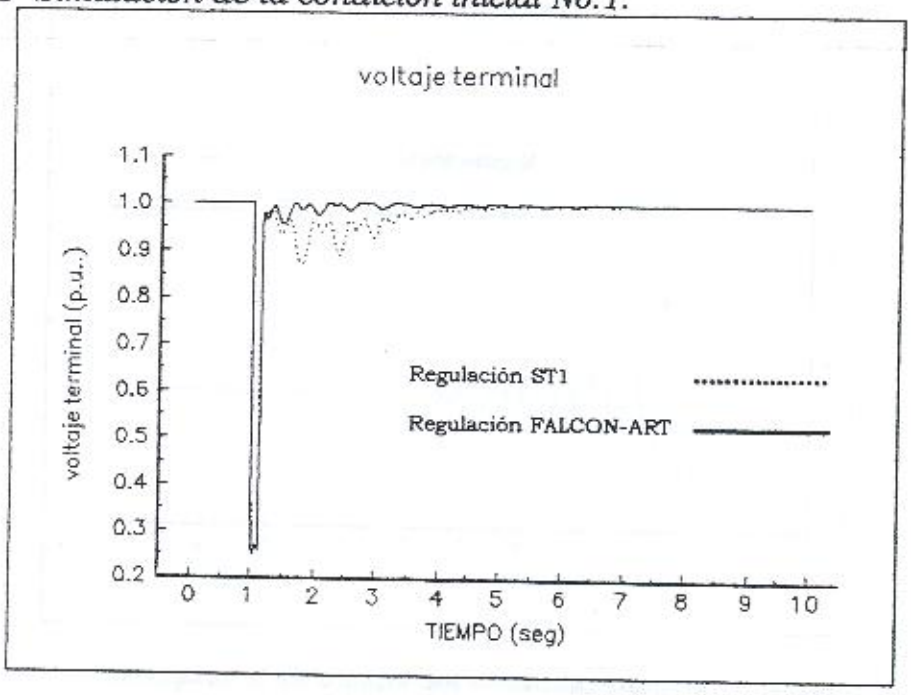


Figura 4. 49 Comportamiento del voltaje terminal con valor inicial de 1.0 p.u.

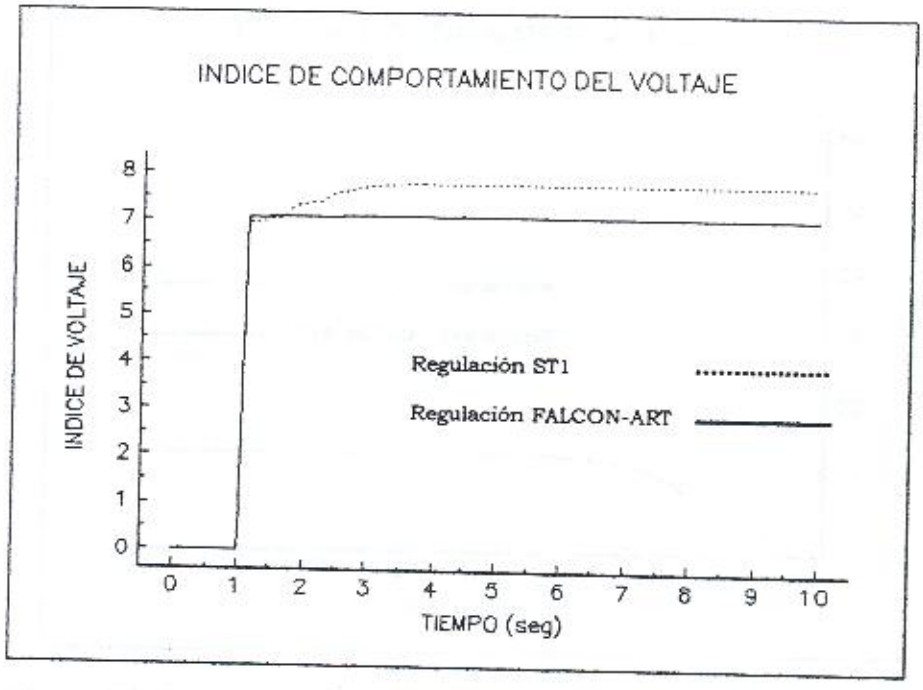


Figura 4. 50 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1.

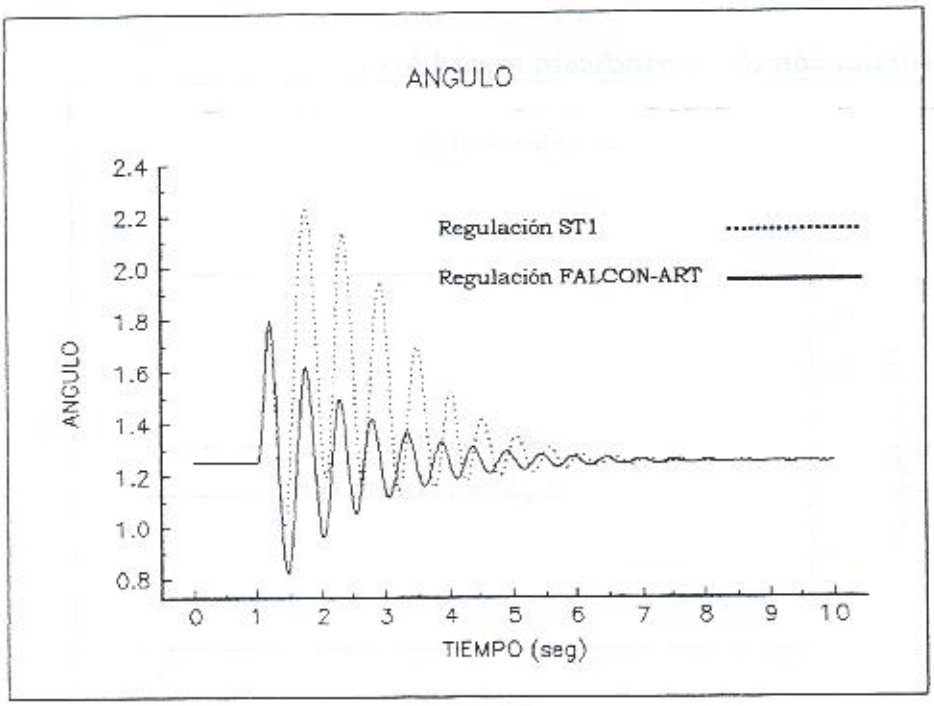


Figura 4. 51 Comportamiento del ángulo de la máquina.

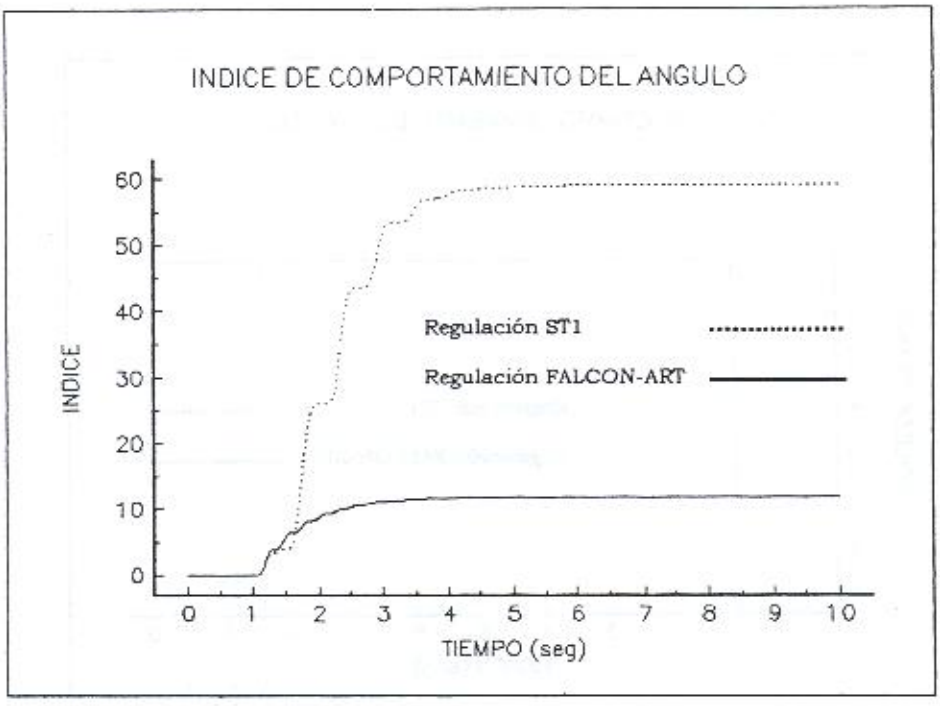


Figura 4. 52 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

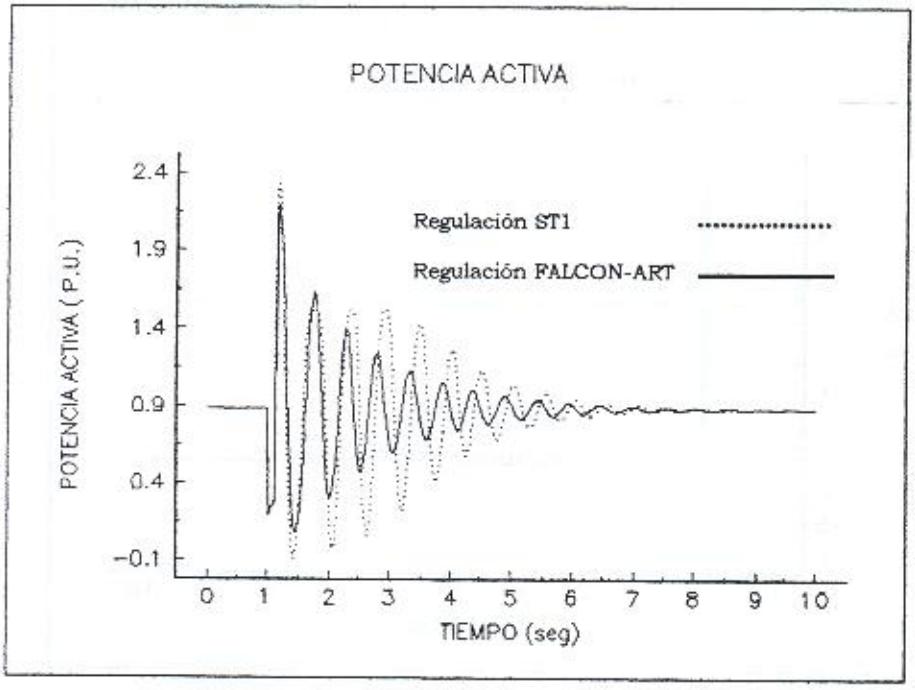


Figura 4. 53 Comportamiento de la potencia activa con valor inicial de 0.8805 p.u.

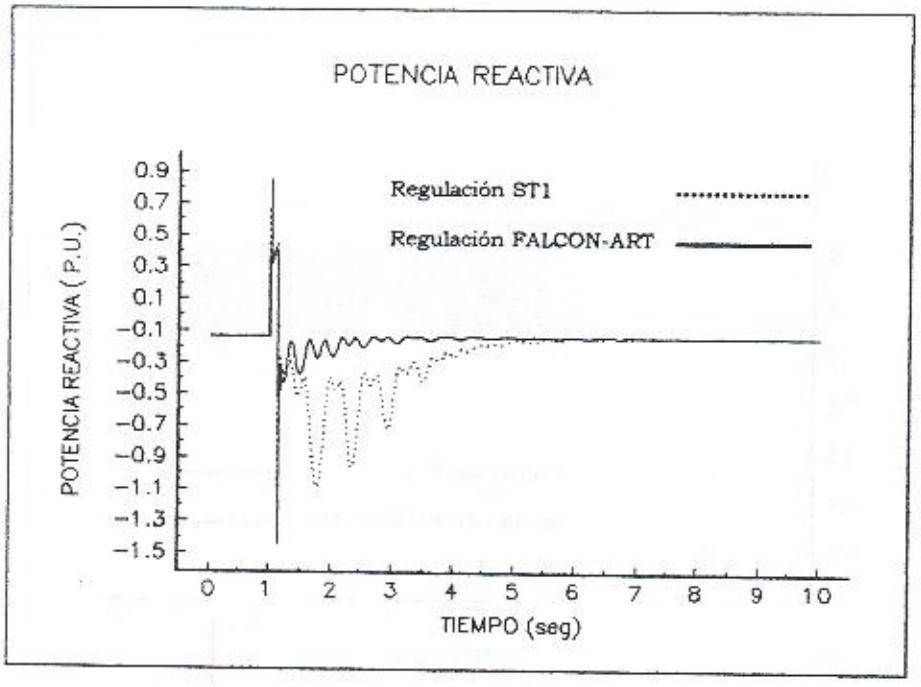


Figura 4. 54 Comportamiento de la potencia reactiva con un valor inicial de –.1361 p.u.

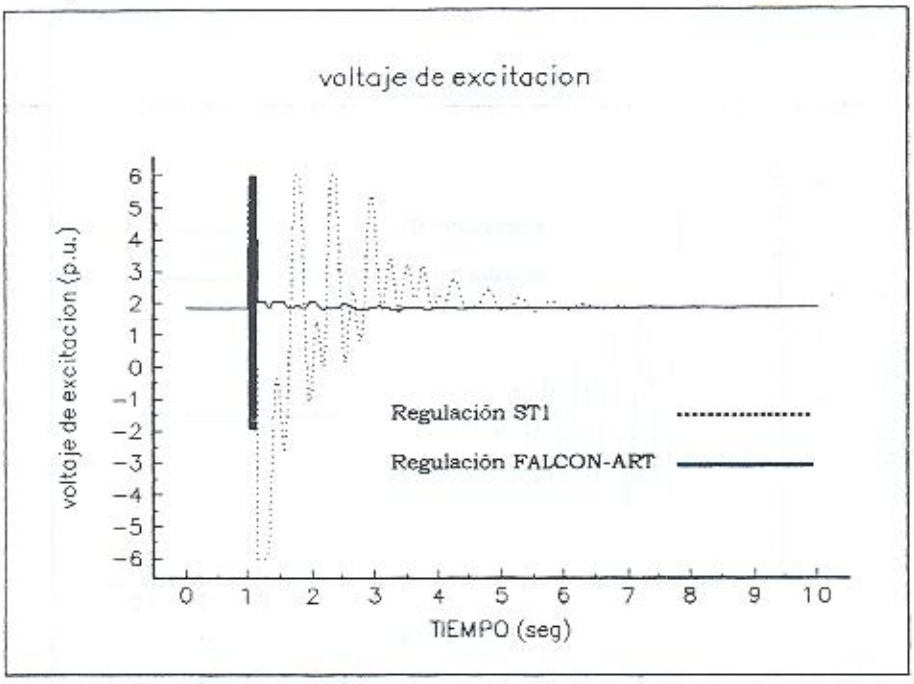


Figura 4. 55 Comportamiento del voltaje de campo.

4.10.2. 2 Simulación de la condición inicial No. 2.

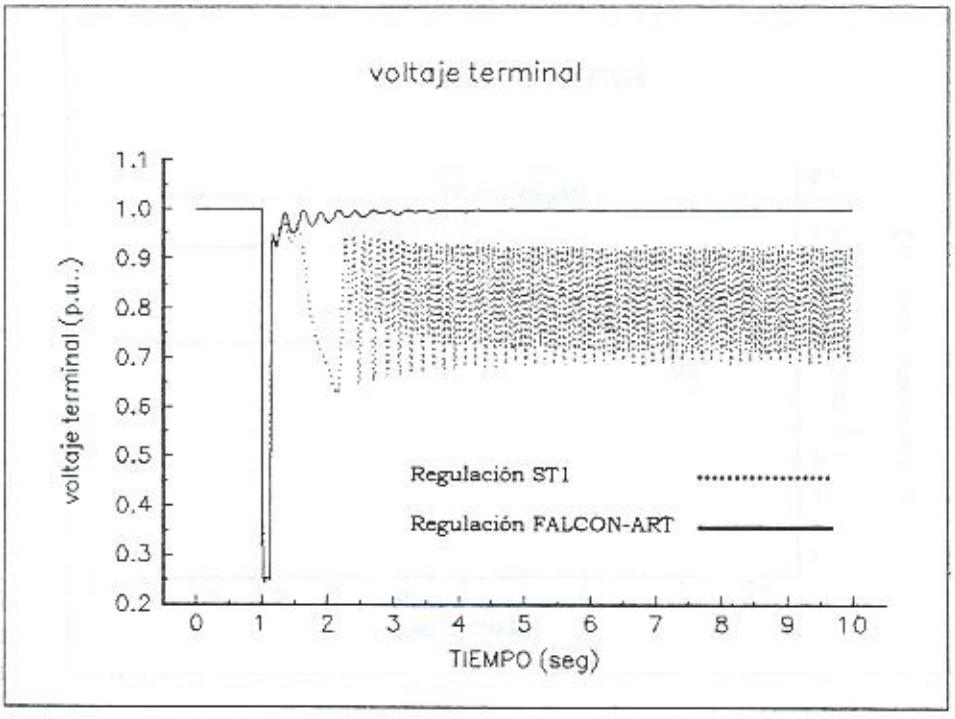


Figura 4. 56 Comportamiento del voltaje terminal con valor inicial de 1.0 p.u.

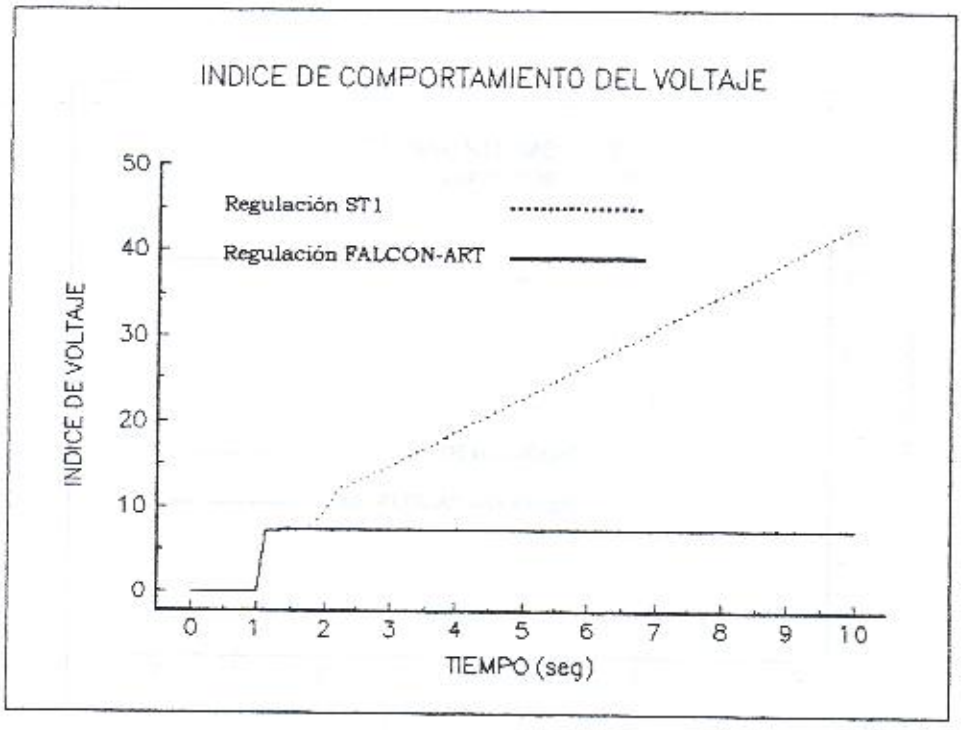


Figura 4. 57 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1.

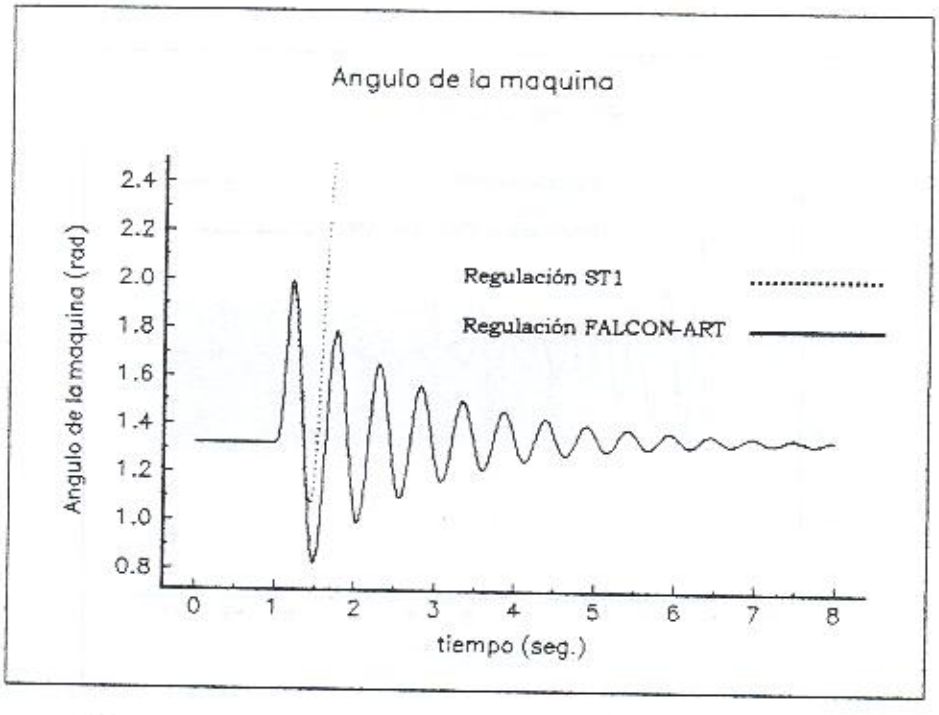


Figura 4. 58 Comportamiento del ángulo de la máquina.

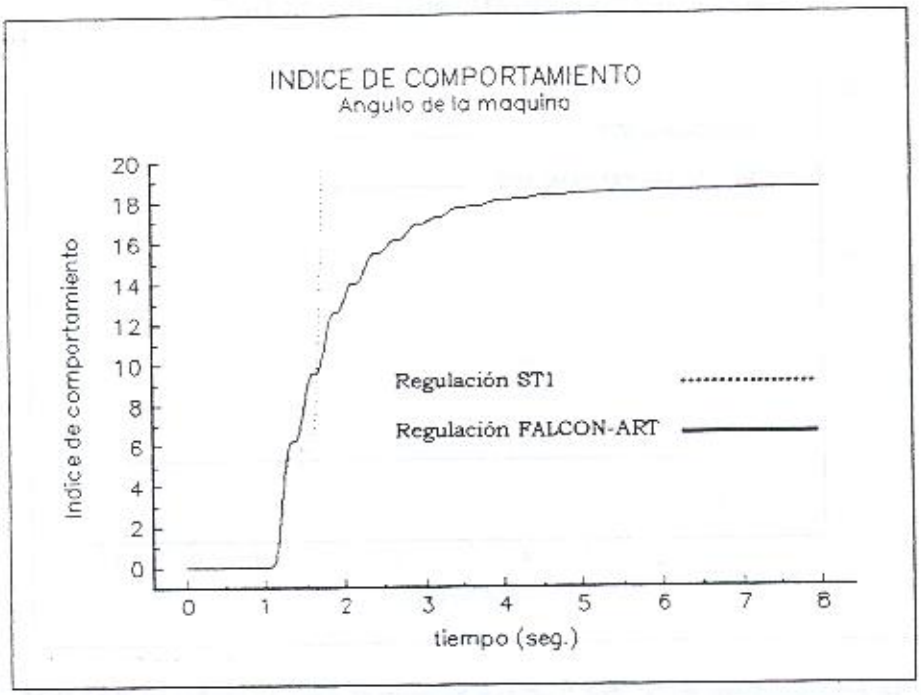


Figura 4. 59 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

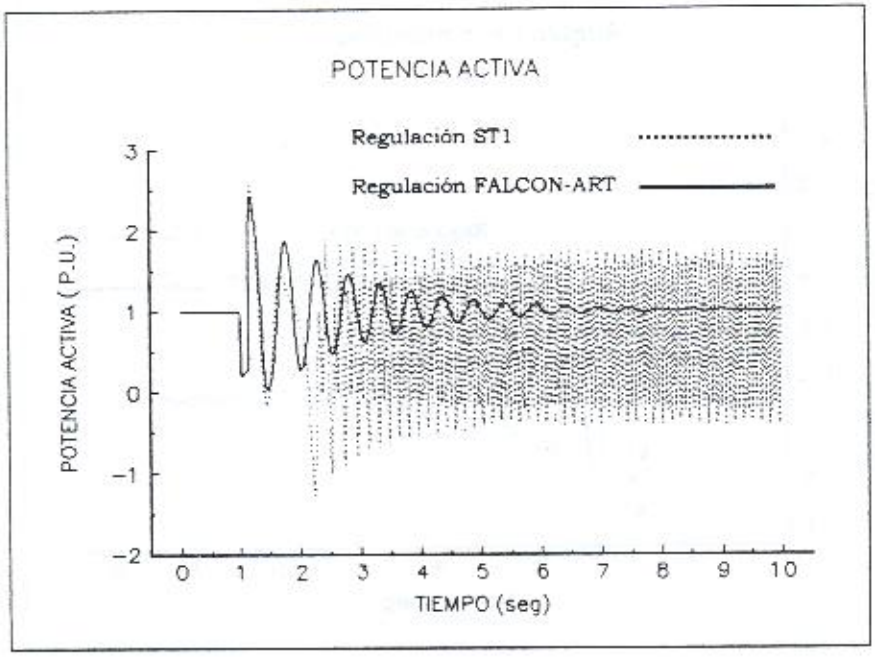


Figura 4. 60 Comportamiento de la potencia activa con valor inicial de 1.00 p.u.

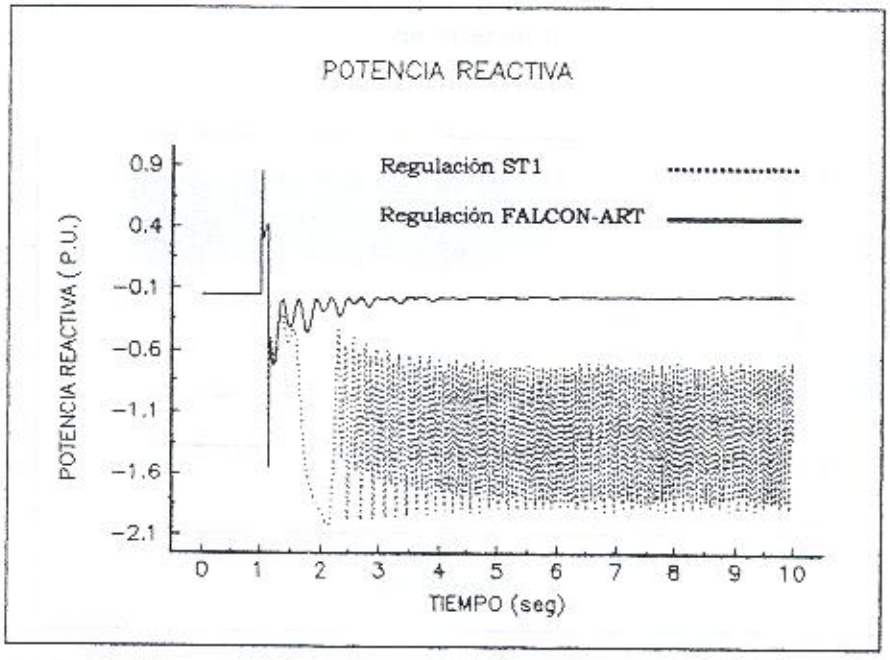


Figura 4. 61 Comportamiento de la potencia reactiva con un valor inicial de –.1473 p.u.

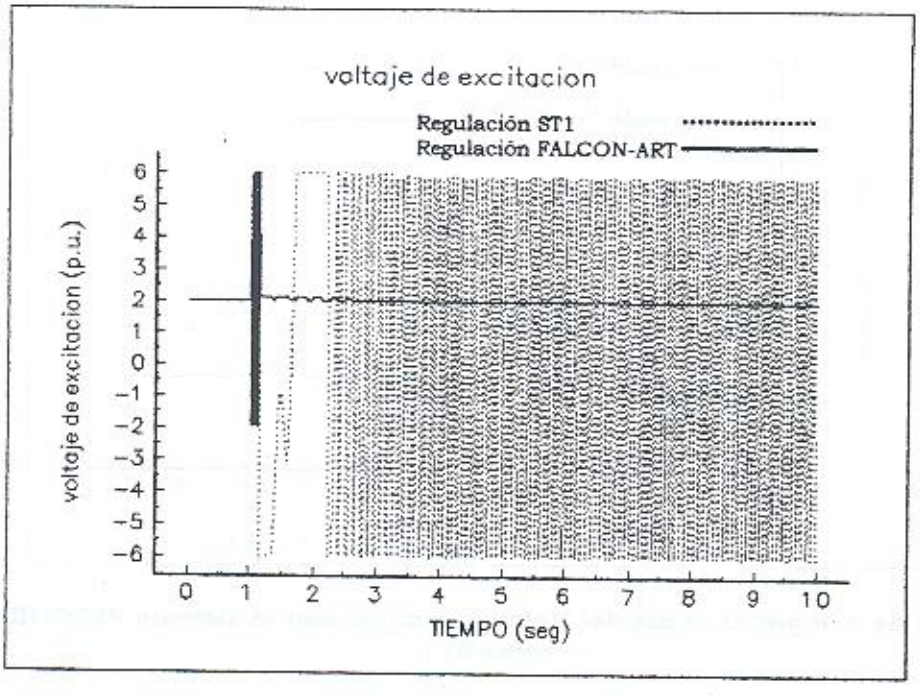


Figura 4. 62 Comportamiento del voltaje de campo.

4.10.2. 3 Simulación de la condición inicial No. 15.

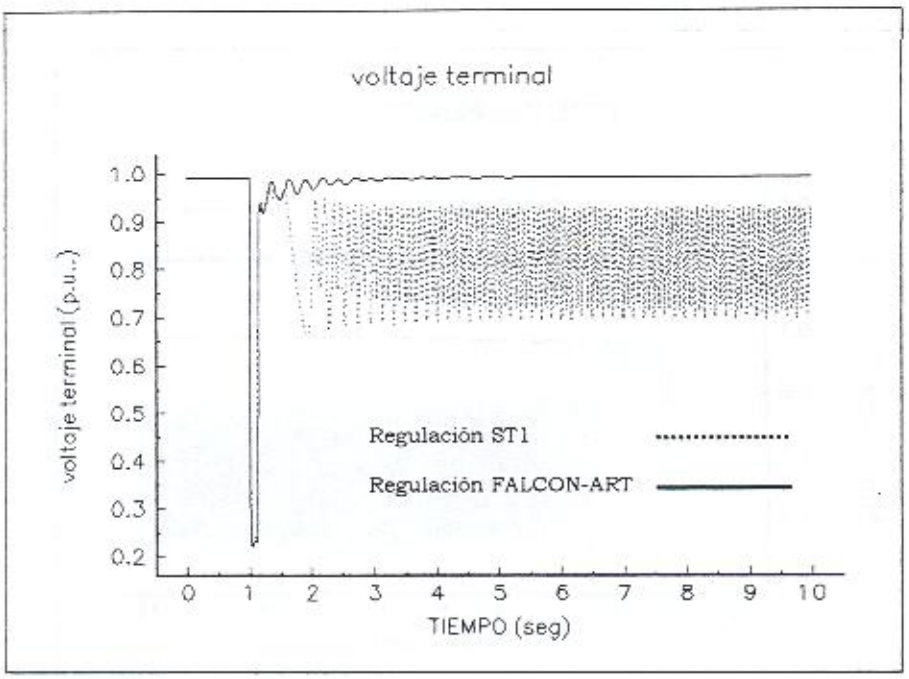


Figura 4. 63 Comportamiento del voltaje terminal con valor inicial de 0.9896 p.u.

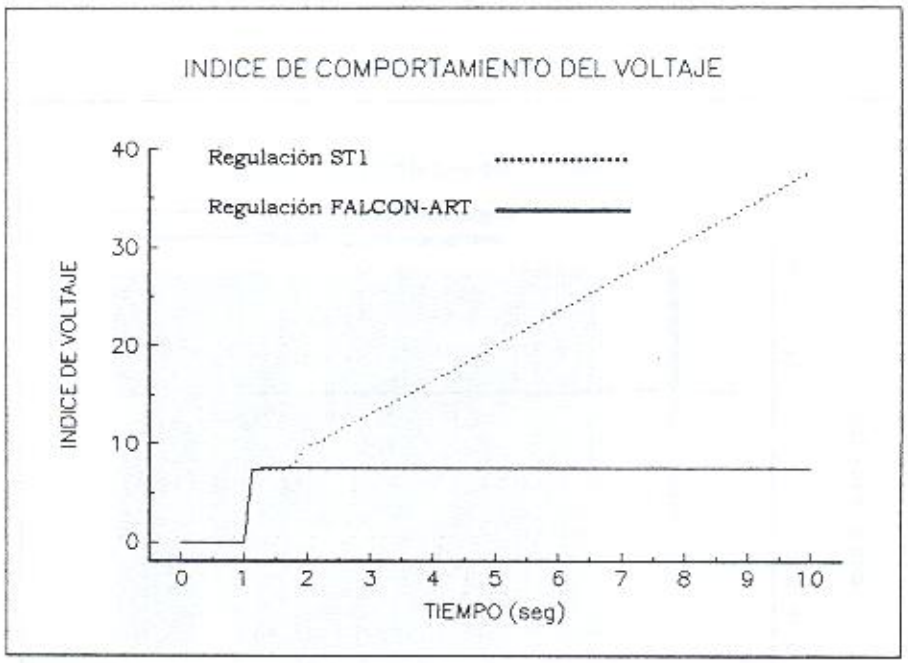


Figura 4. 64 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1.

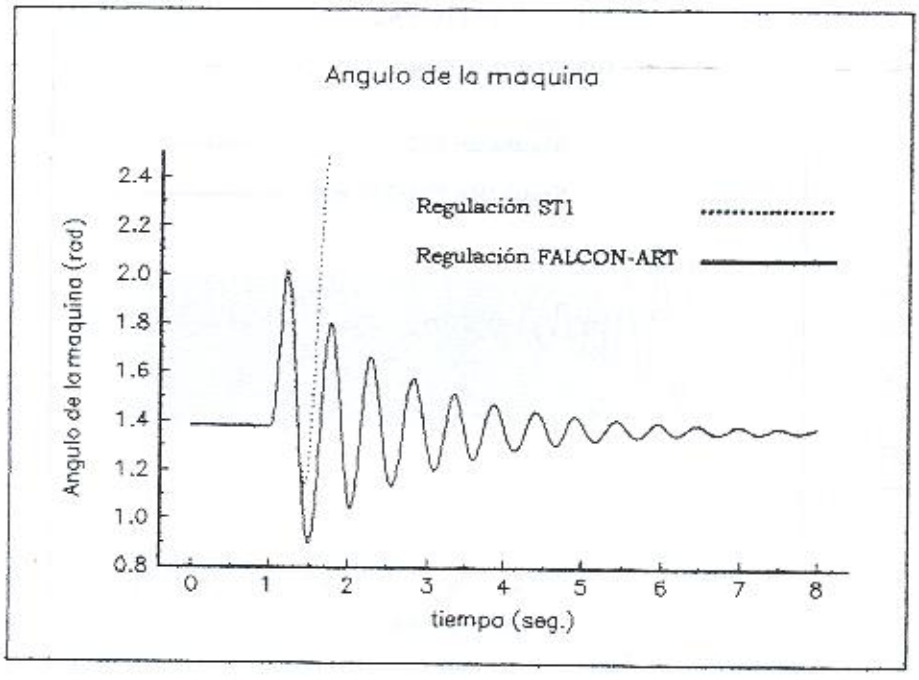


Figura 4. 65 Comportamiento del ángulo de la máquina.

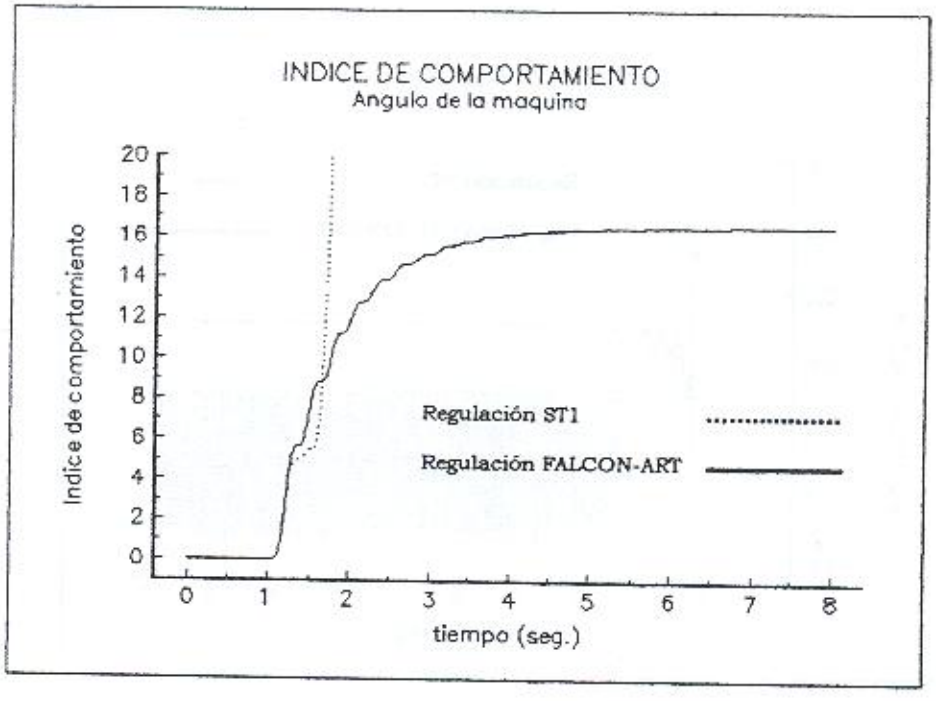


Figura 4. 66 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

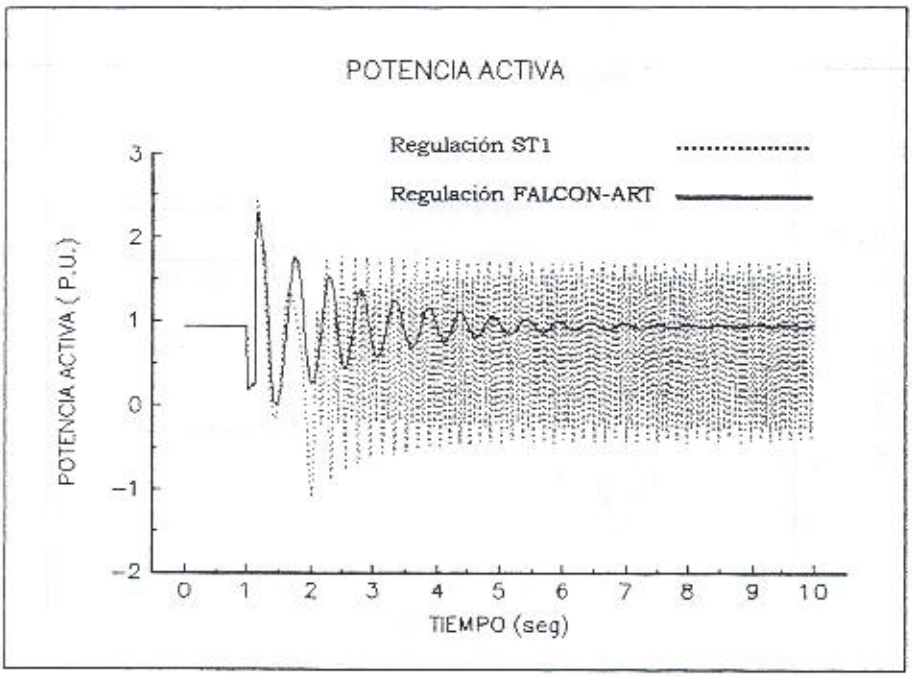


Figura 4. 67 Comportamiento de la potencia activa con valor inicial de 0.9319 p.u.

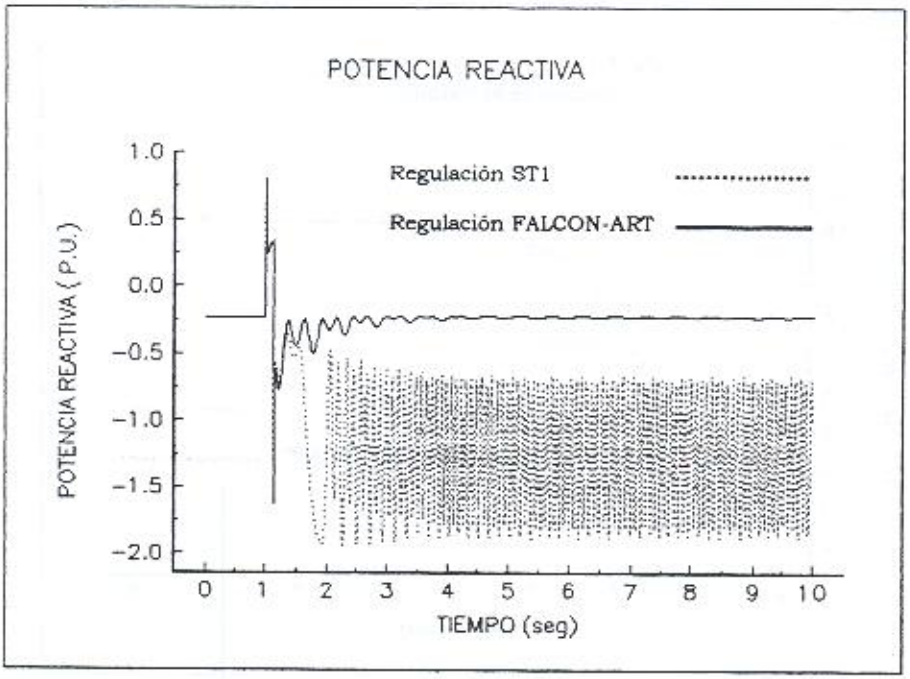


Figura 4. 68 Comportamiento de la potencia reactiva con un valor inicial de −.2272 p.u.

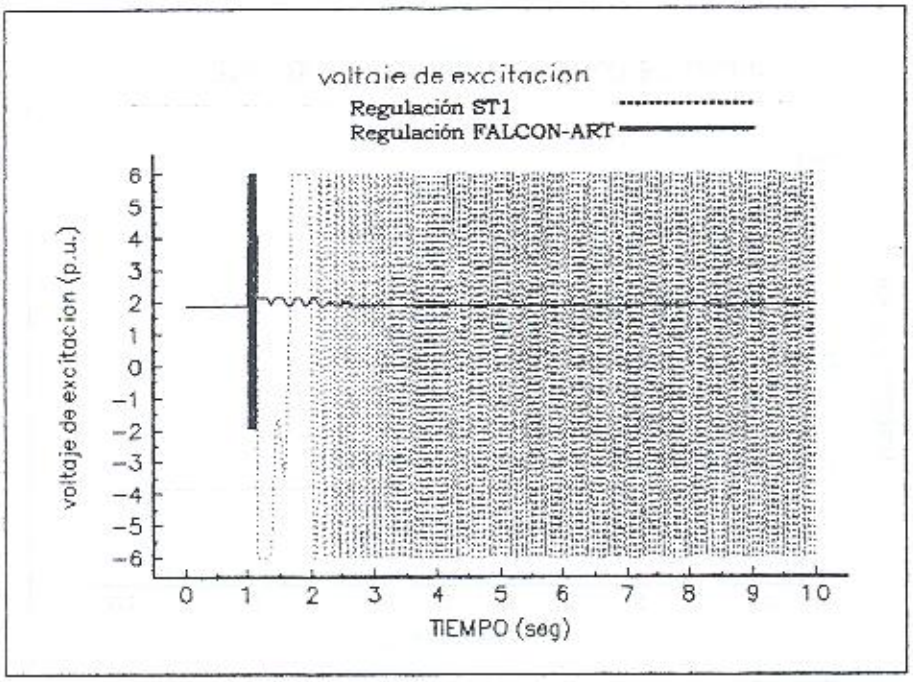


Figura 4. 69 Comportamiento del voltaje de campo.

4.10.2. 4 Simulación de la condición inicial No. 20.

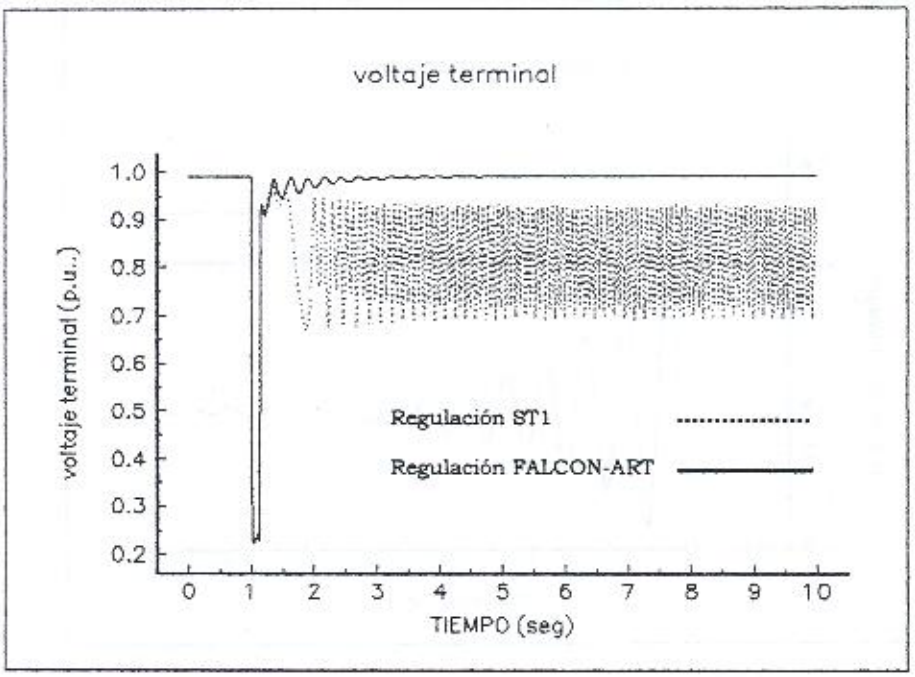


Figura 4. 70 Comportamiento del voltaje terminal con valor inicial de 0.9904 p.u.

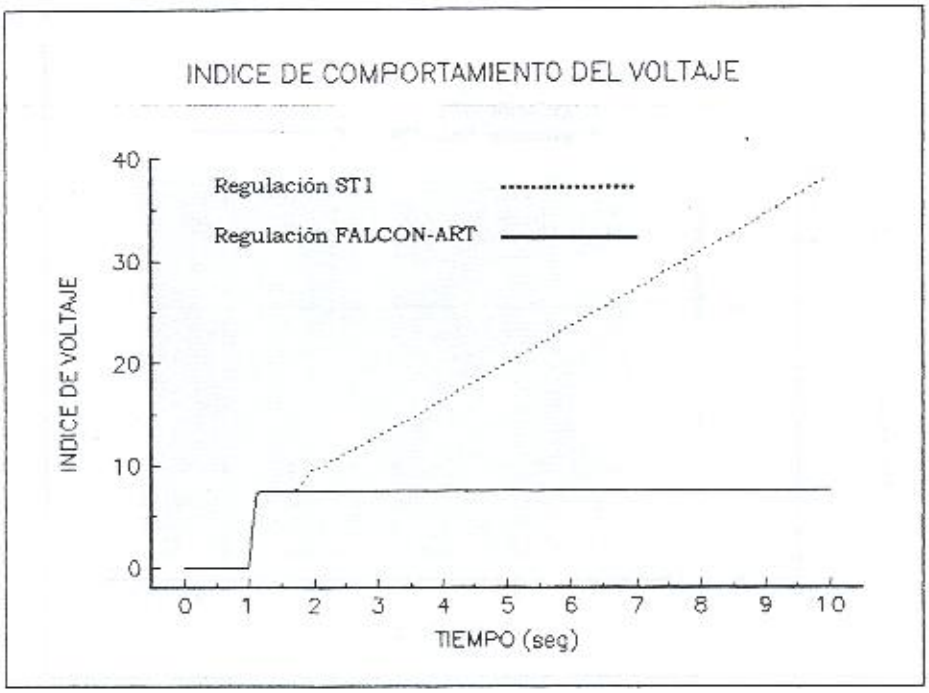


Figura 4. 71 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1.

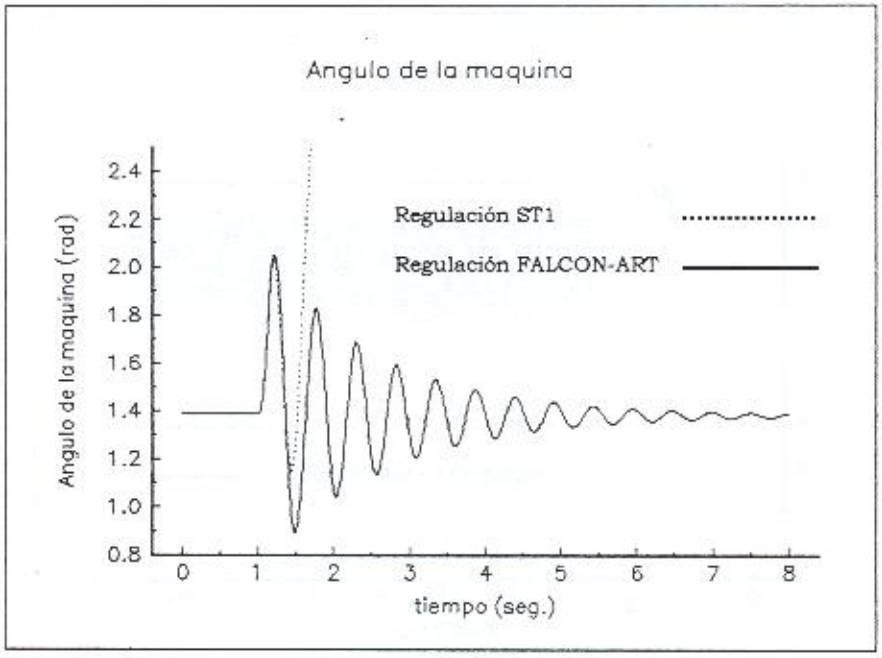


Figura 4. 72 Comportamiento del ángulo de la máquina.

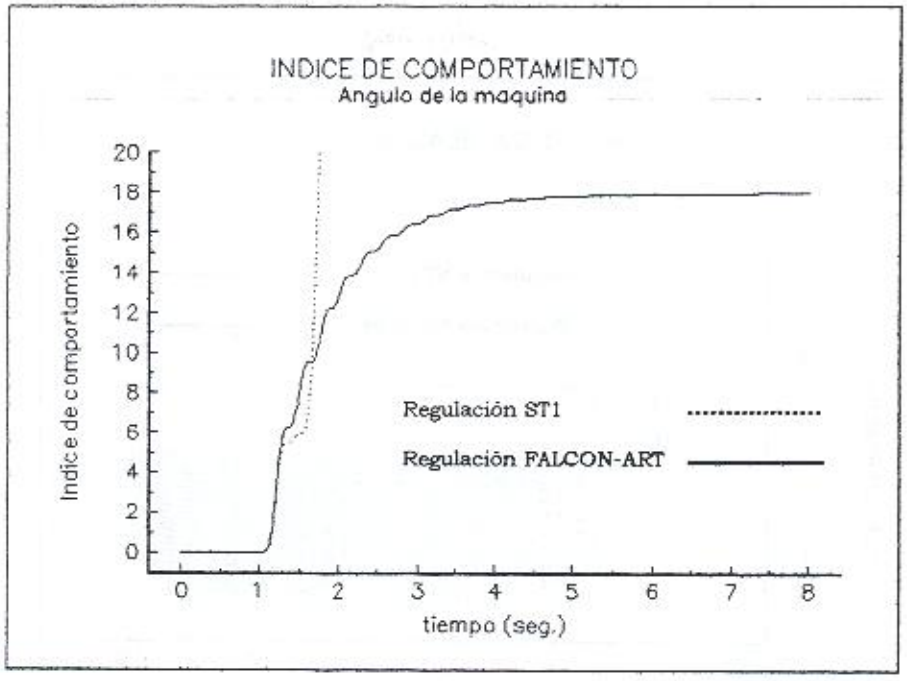


Figura 4. 73 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

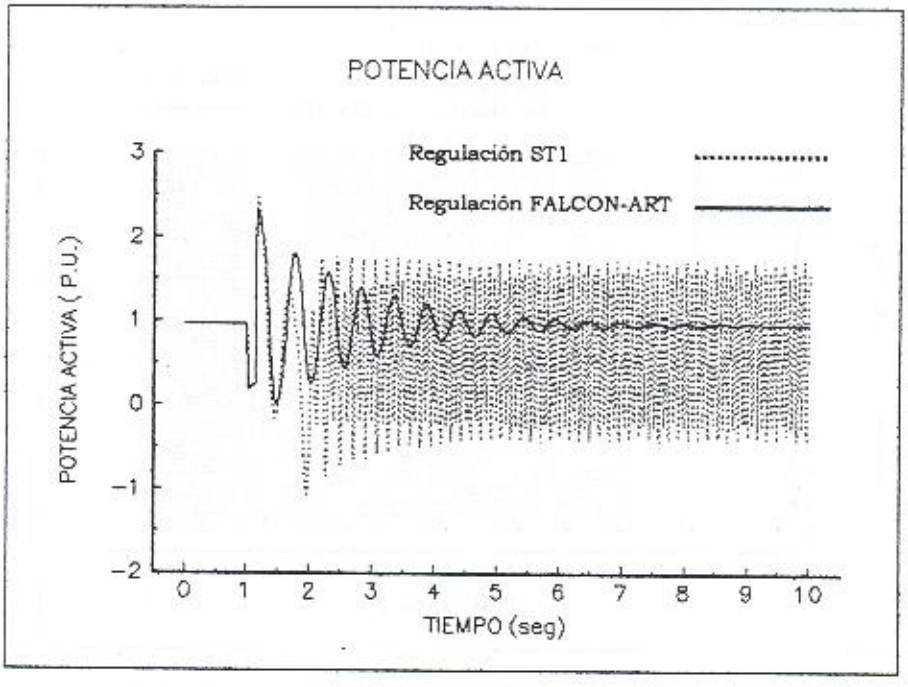


Figura 4. 74 Comportamiento de la potencia activa con valor inicial de 0.9592 p.u.

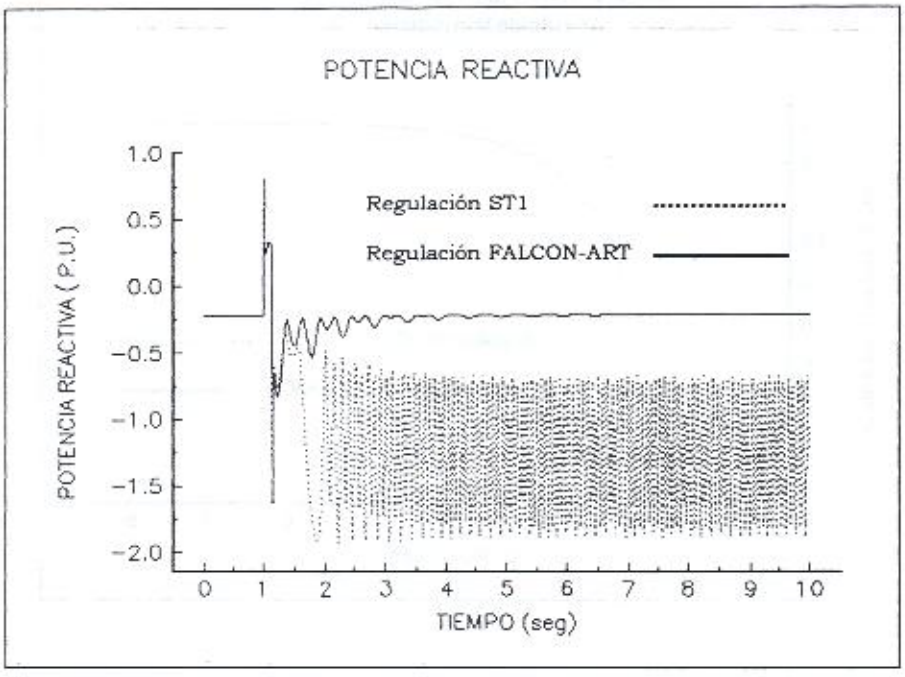


Figura 4. 75 Comportamiento de la potencia reactiva con un valor inicial de −.2236 p.u.

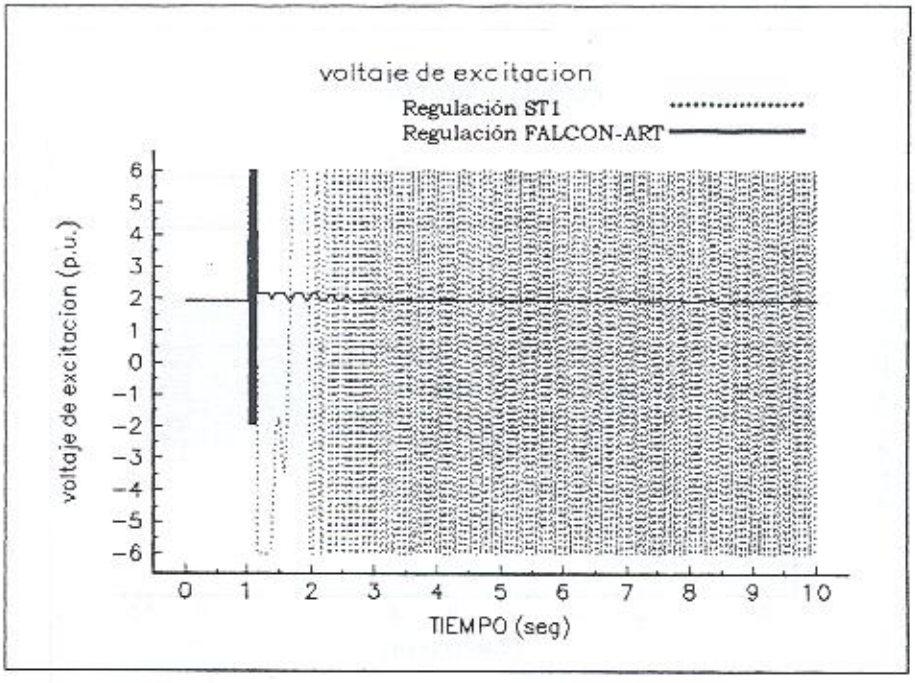


Figura 4. 76 Comportamiento del voltaje de campo.

4.10.2. 5 Simulación de la condición inicial No. 27.

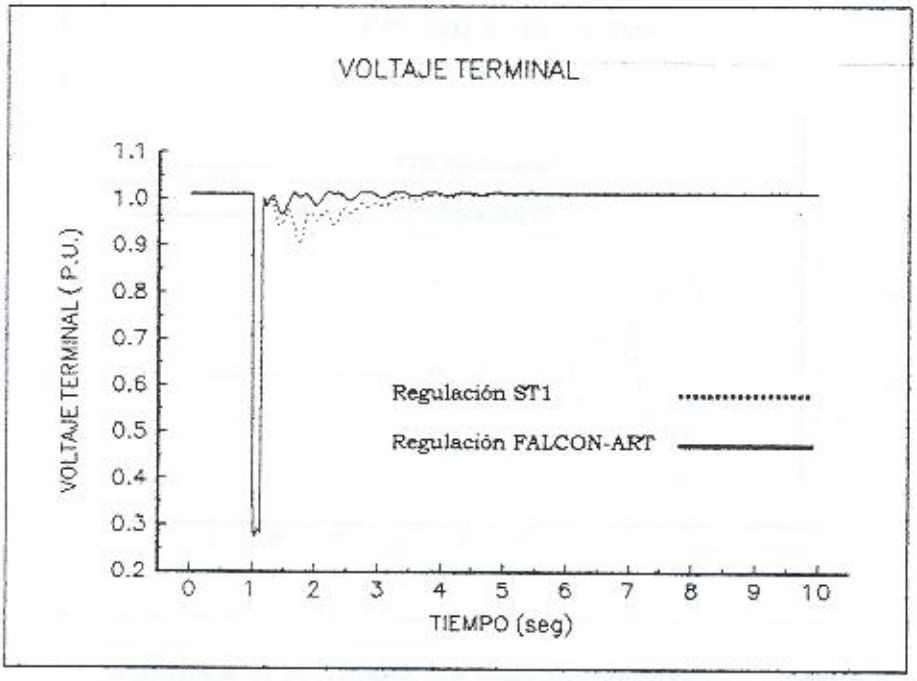


Figura 4. 77 Comportamiento del voltaje terminal con valor inicial de 1.0109 p.u.

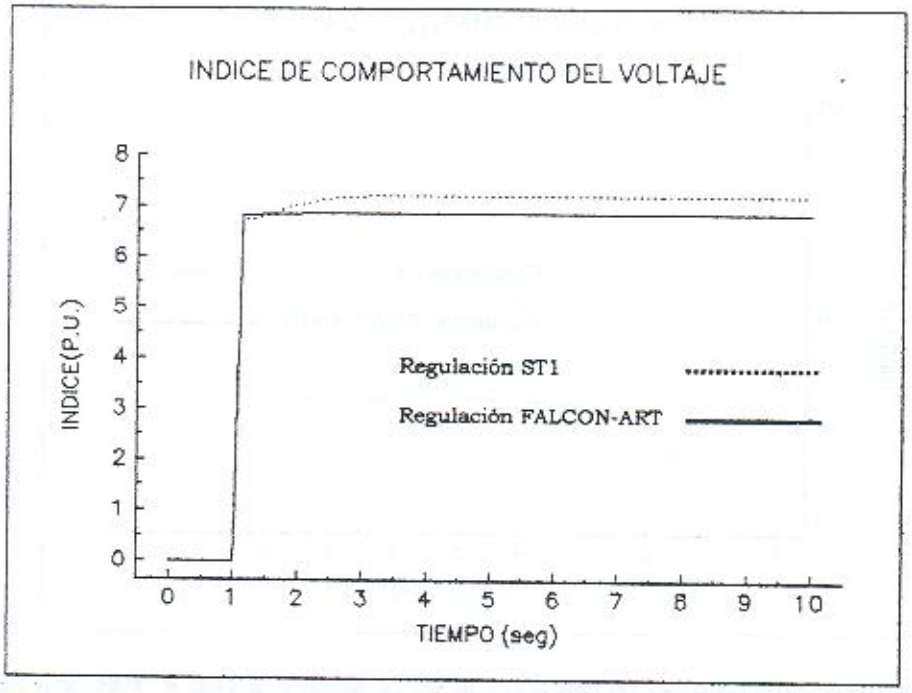


Figura 4. 78 Índice de comportamiento del Voltaje terminal con el sistema FALCON-ART y con el sistema ST1.

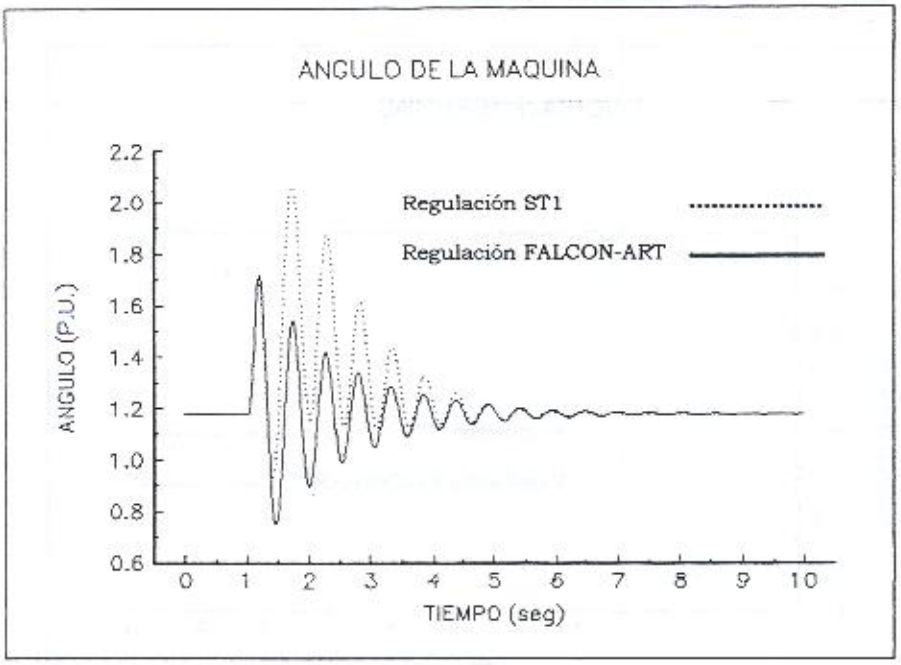


Figura 4. 79 Comportamiento del ángulo de la máquina.

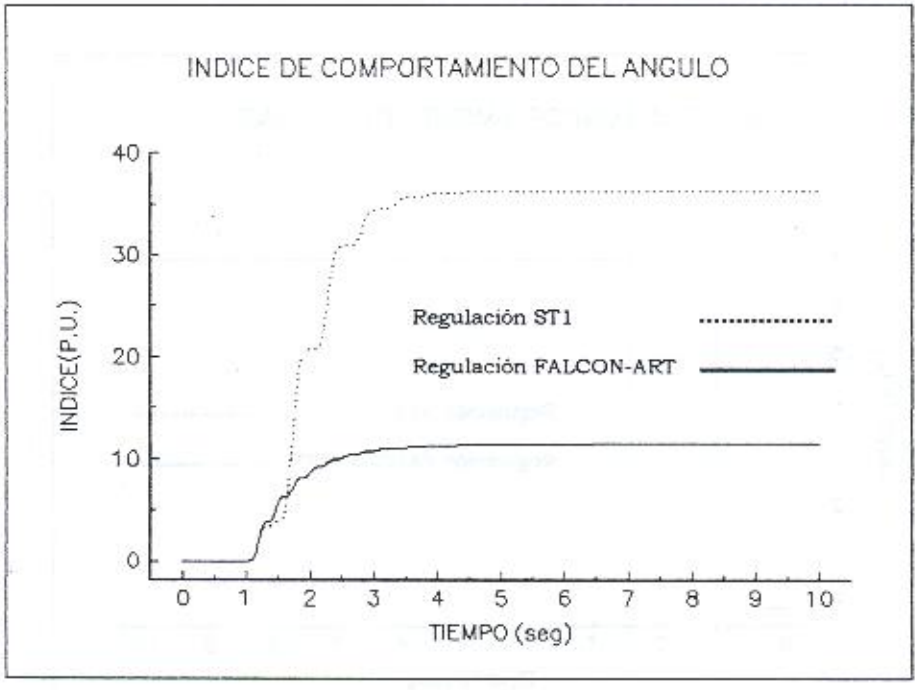


Figura 4. 80 Índice de comportamiento del Ángulo de la máquina con el sistema FALCON-ART y con el regulador ST1.

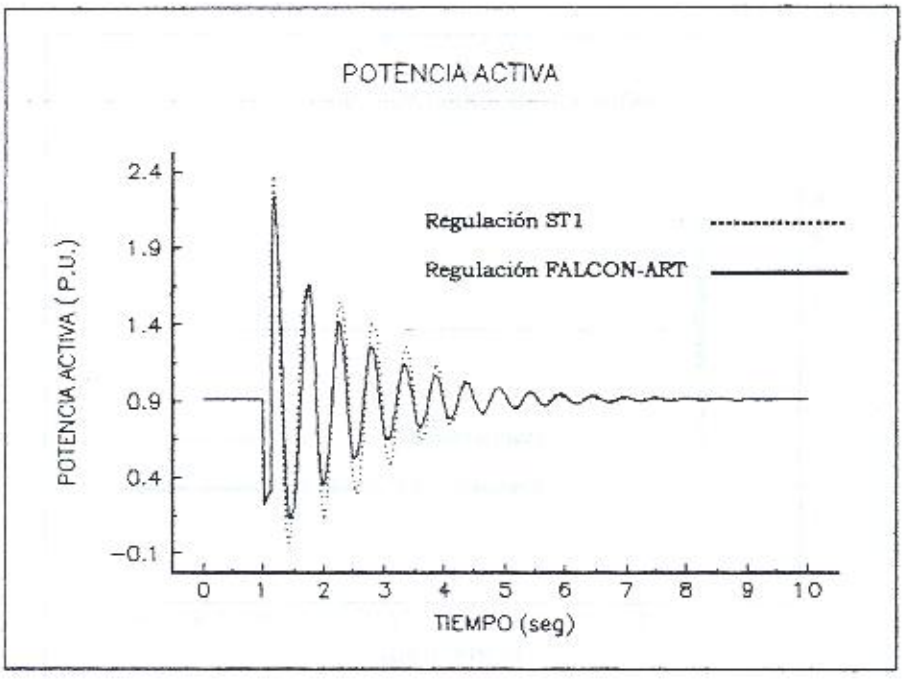


Figura 4. 81 Comportamiento de la potencia activa con valor inicial de 0.9117 p.u.

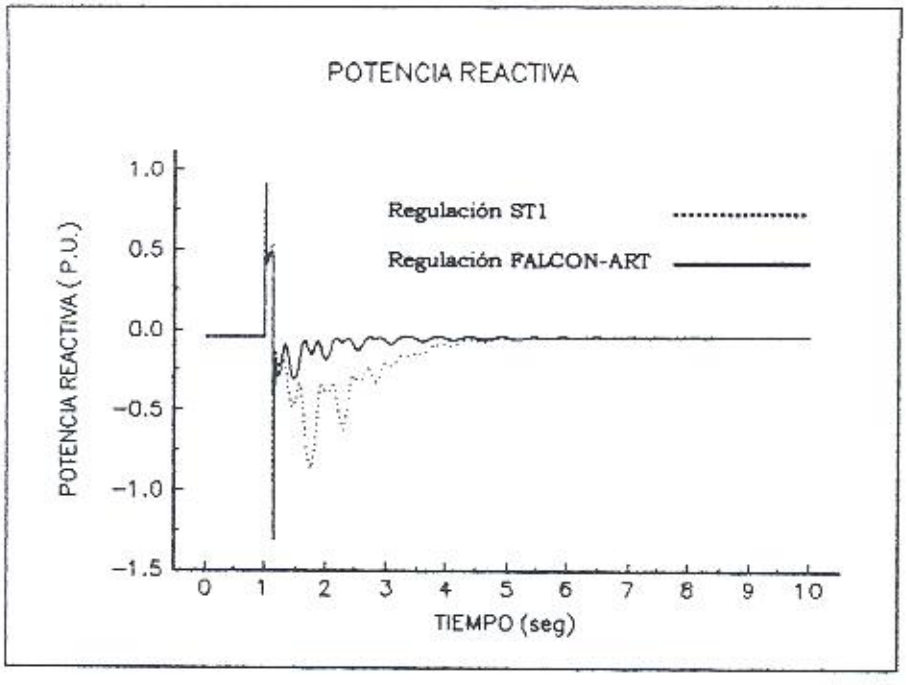


Figura 4. 82 Comportamiento de la potencia reactiva con un valor inicial de −.4590 p.u.

4.10.3 CONDICIONES INICIALES CON UNA FALLA DE 18 CICLOS DE DURACIÓN.

4.10.3. 1 Simulación de la condición inicial No.144.

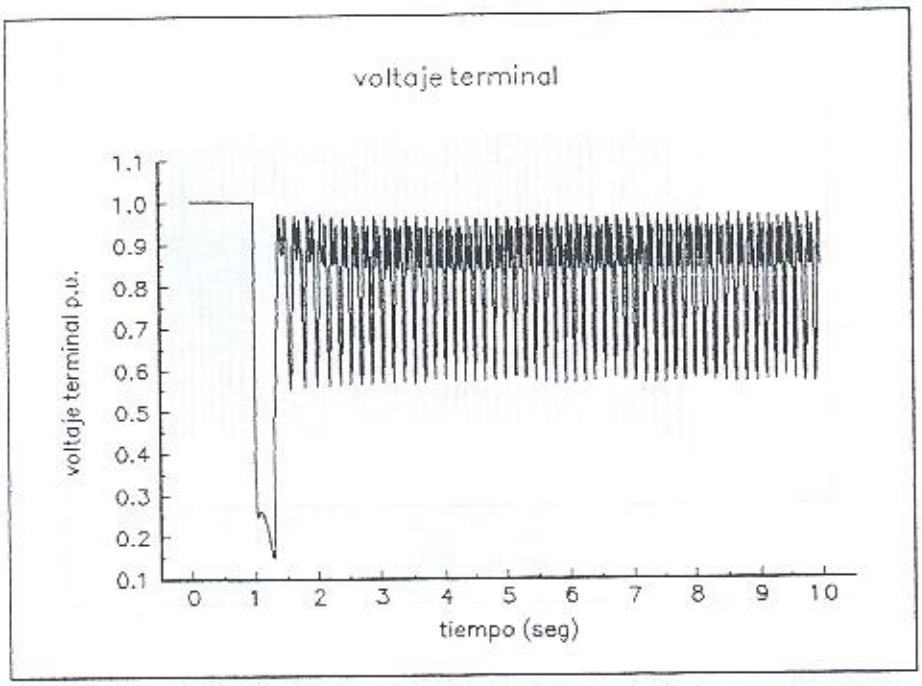


Figura 4. 84 Comportamiento del voltaje terminal con valor inicial de 1.001 p.u.

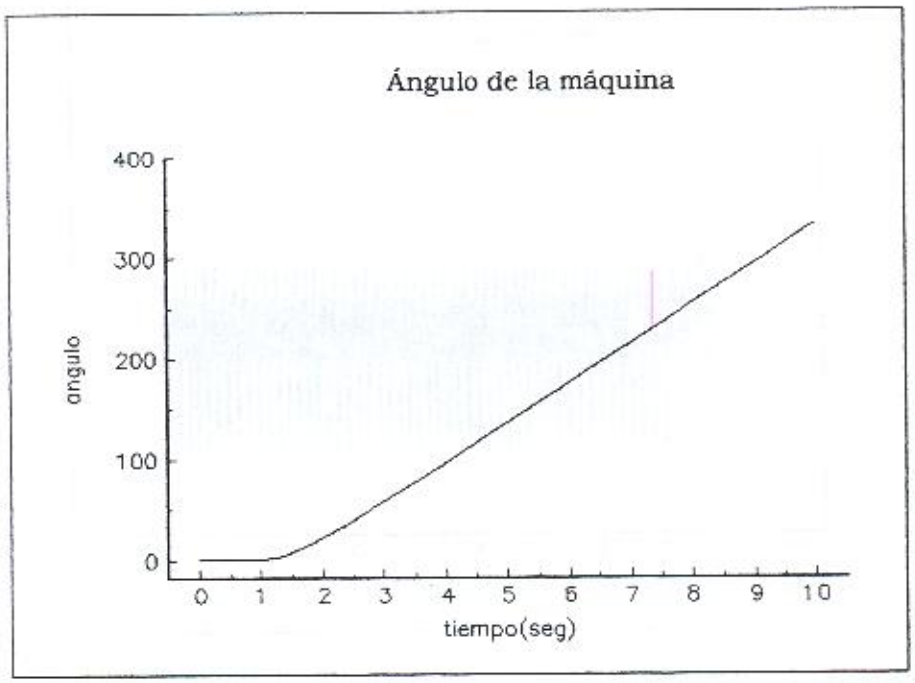


Figura 4. 85 Comportamiento del ángulo de la máquina.

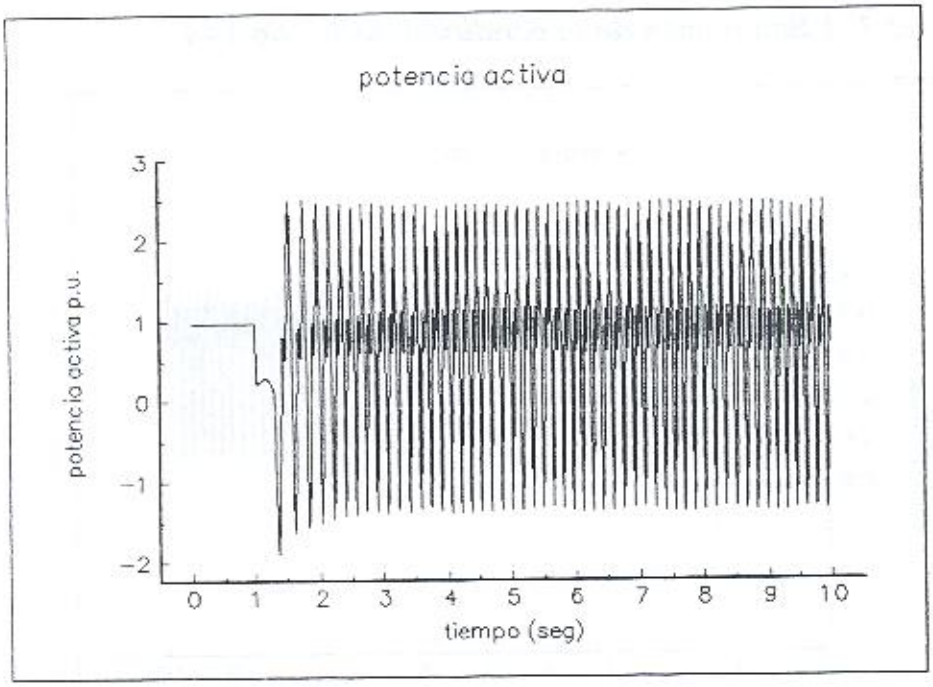


Figura 4. 86 Comportamiento de la potencia activa con valor inicial de 0.9645 p.u.

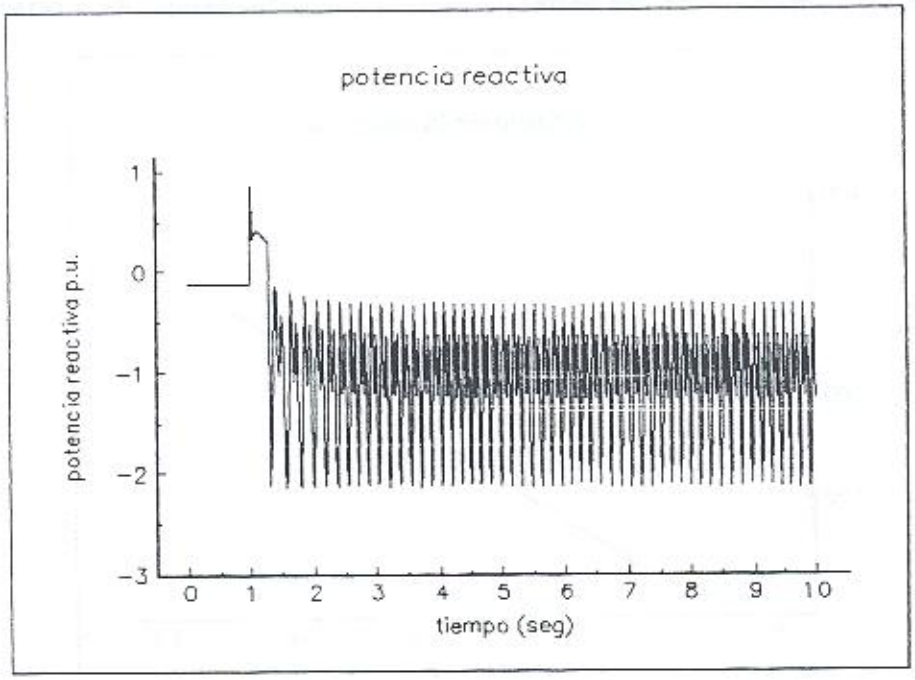


Figura 4. 87 Comportamiento de la potencia reactiva con un valor inicial de −.1294 p.u.

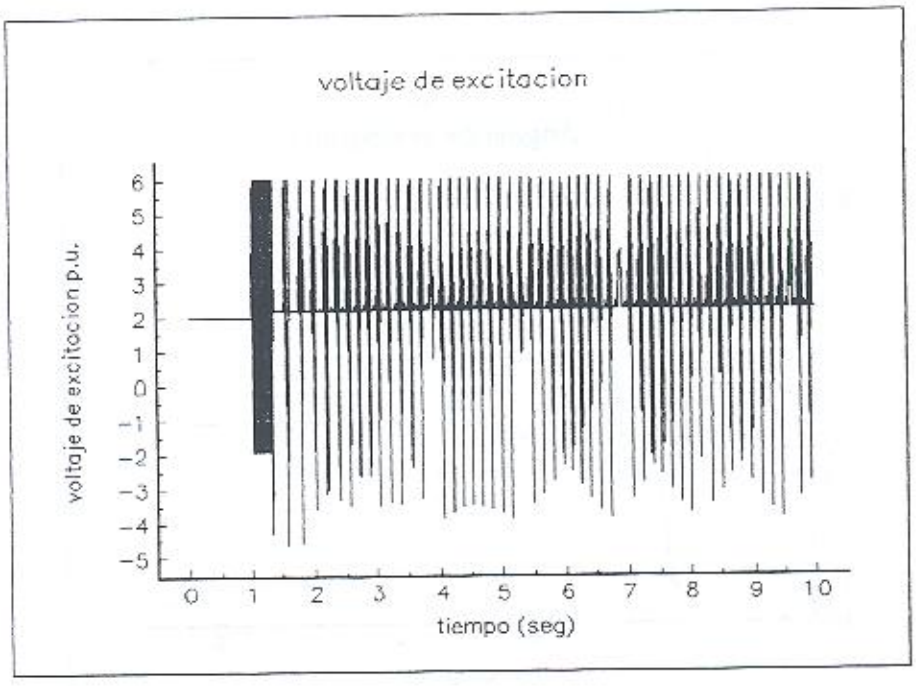


Figura 4. 88 Comportamiento del voltaje de campo.

4.10.3. 2 Simulación de la condición inicial No.145.

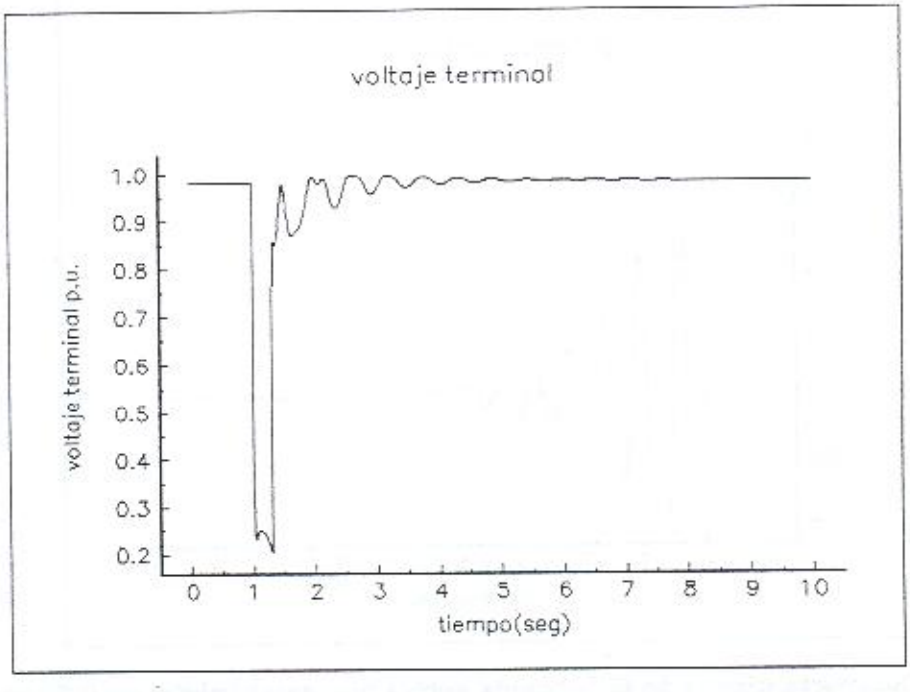


Figura 4. 89 Comportamiento del voltaje terminal con valor inicial de .9823 p.u.

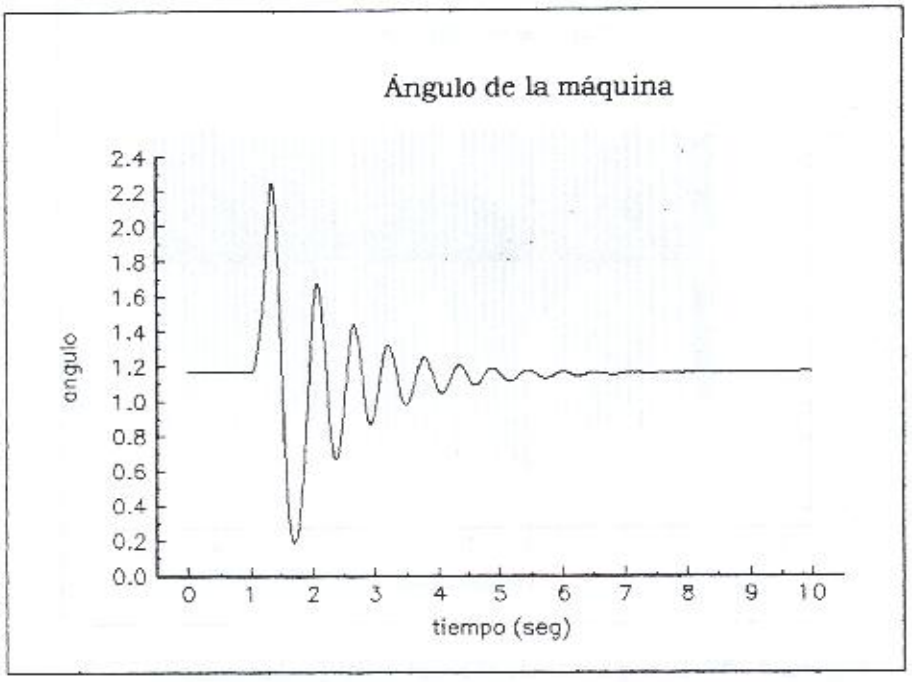


Figura 4. 90 Comportamiento del ángulo de la máquina.

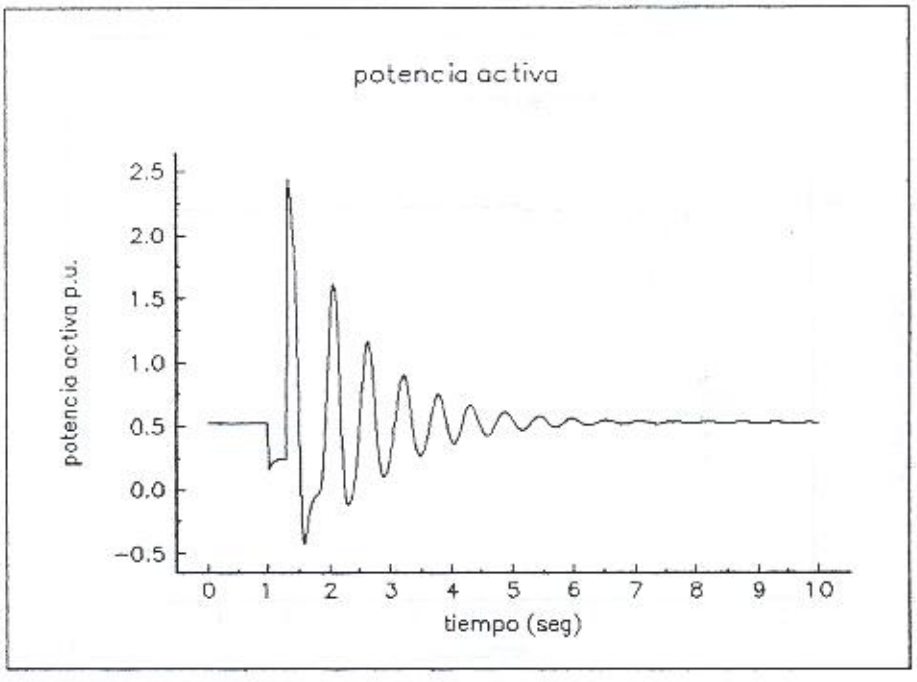


Figura 4. 91 Comportamiento de la potencia activa con valor inicial de 0.5284 p.u.

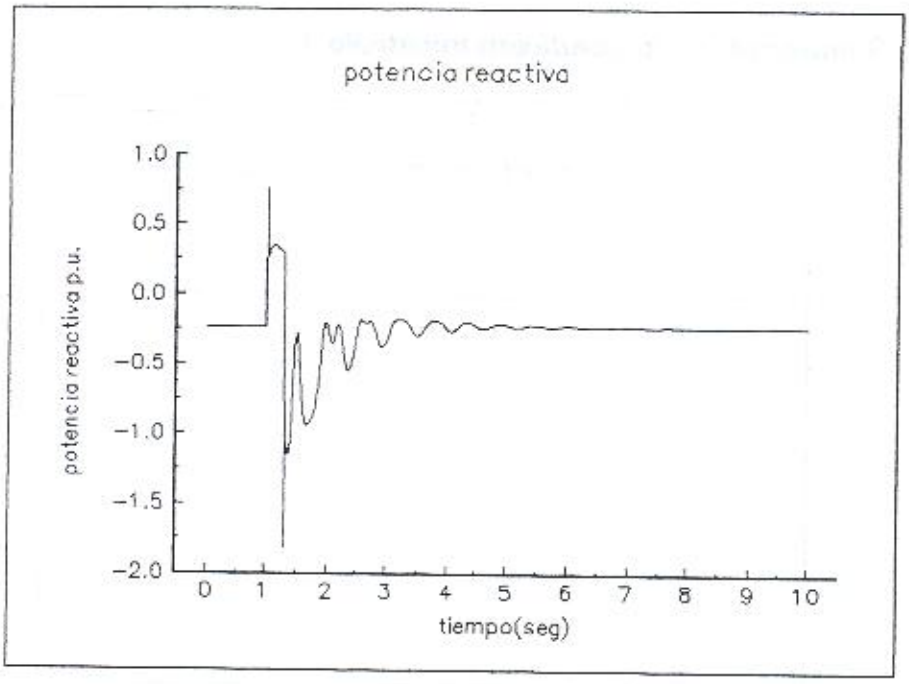


Figura 4. 92 Comportamiento de la potencia reactiva con un valor inicial de –.2413 p.u.

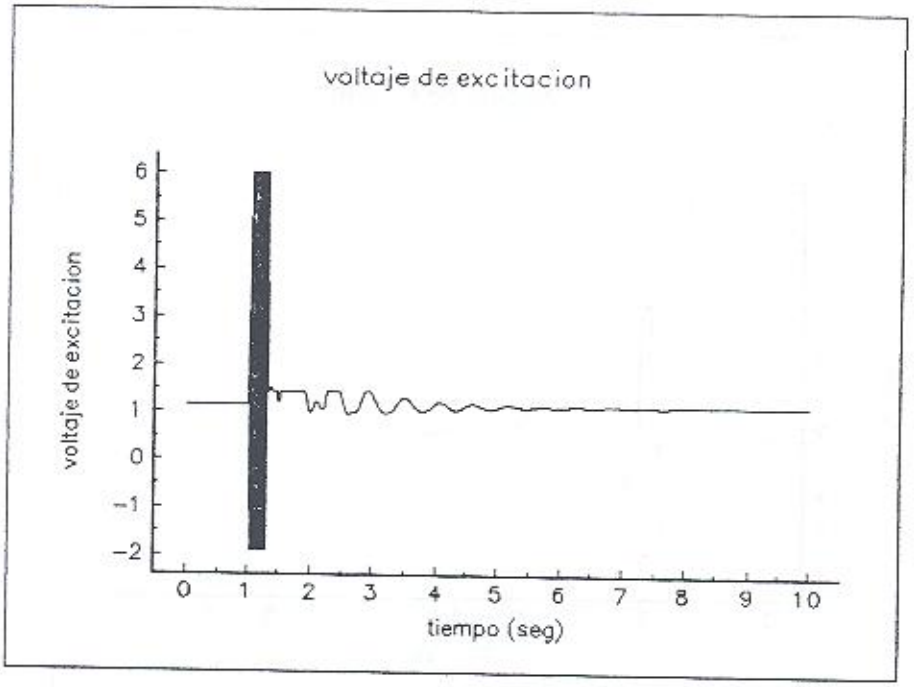


Figura 4. 93 Comportamiento del voltaje de campo.

4.10.3. 3 Simulación de la condición inicial No.191.

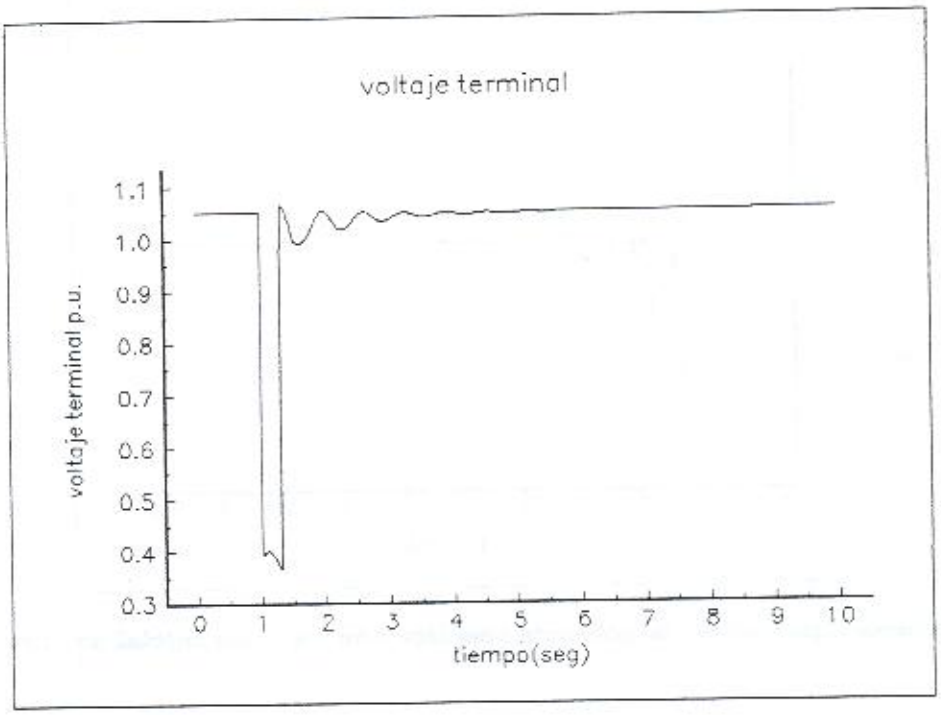


Figura 4. 94 Comportamiento del voltaje terminal con valor inicial de 1.0510 p.u.

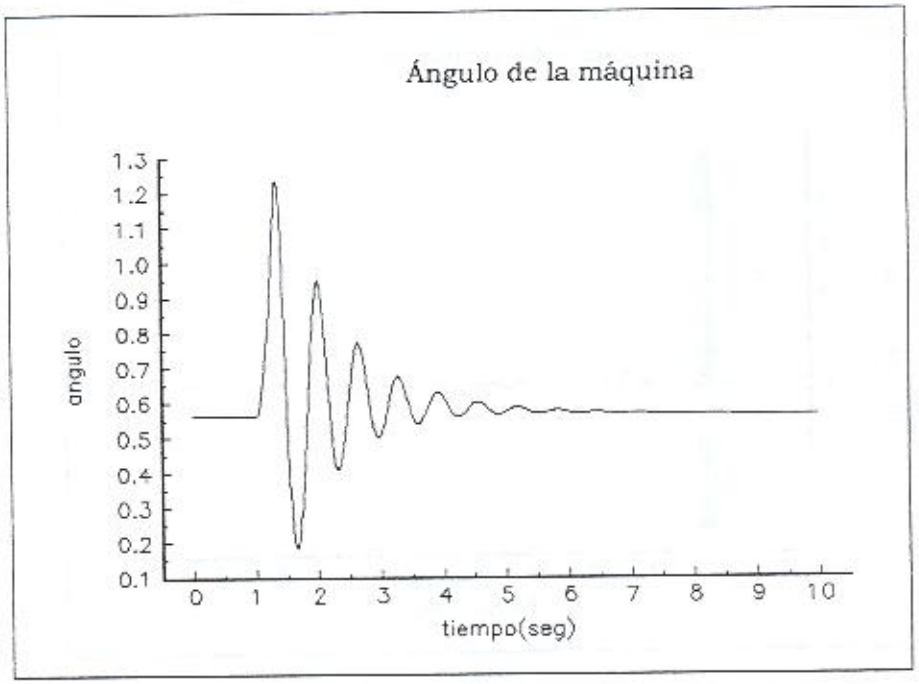


Figura 4. 95 Comportamiento del ángulo de la máquina.

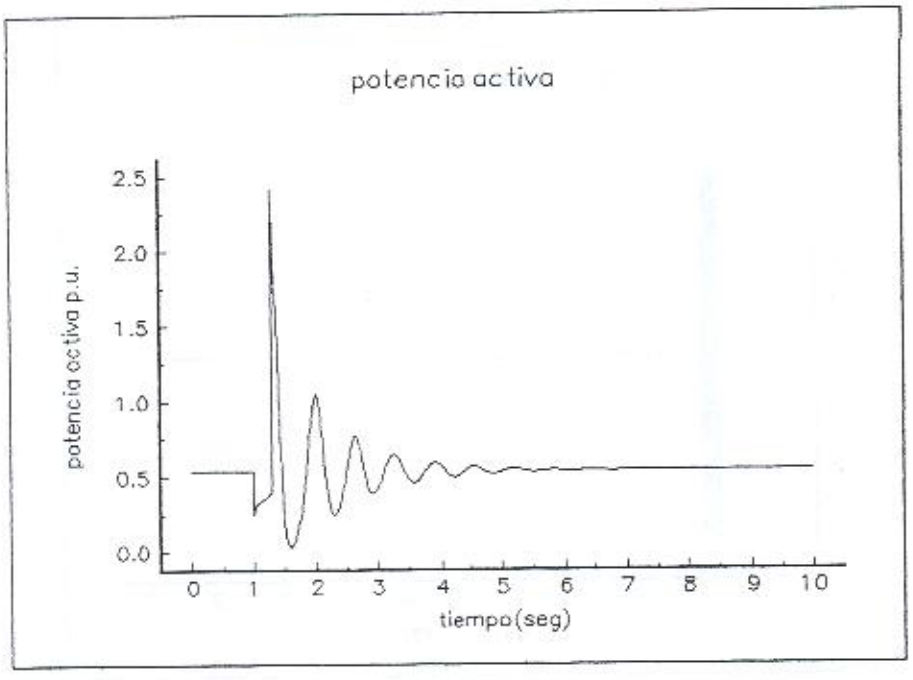


Figura 4. 96 Comportamiento de la potencia activa con valor inicial de 0.5347 p.u.

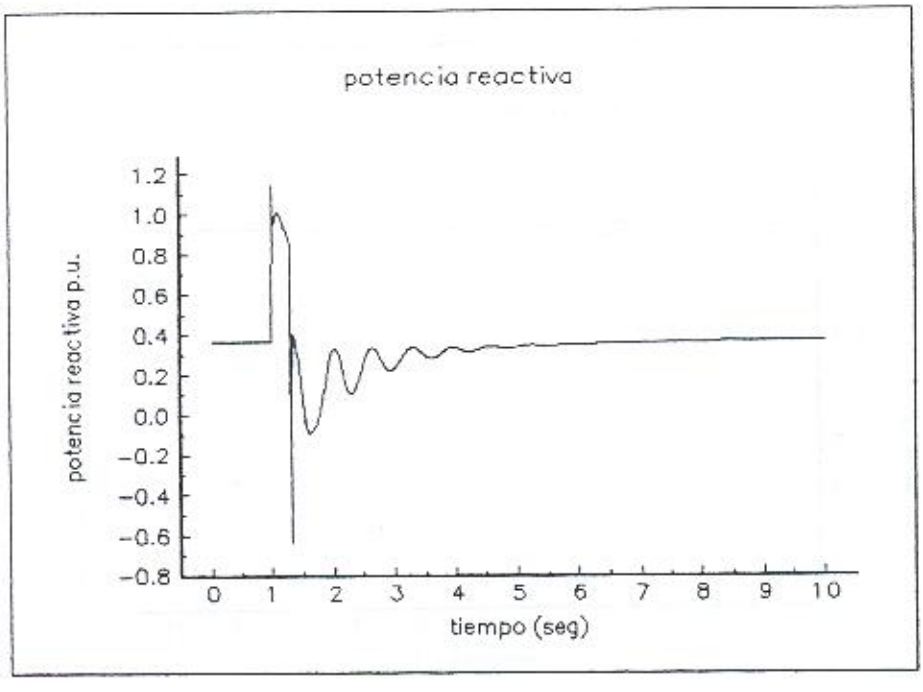


Figura 4. 97 Comportamiento de la potencia reactiva con un valor inicial de 0.3667 p.u.

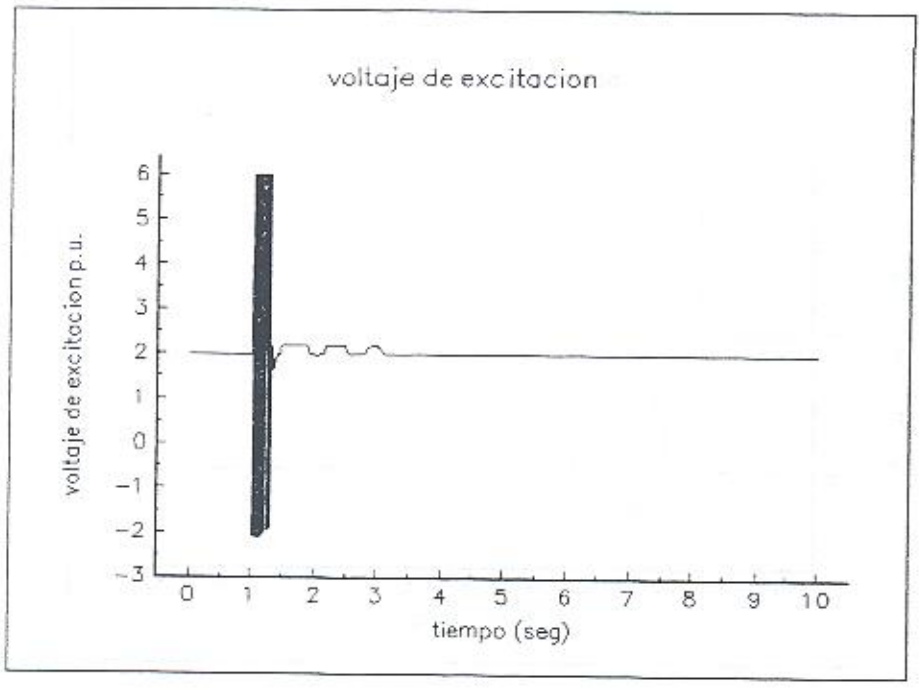


Figura 4. 98 Comportamiento del voltaje de campo.

4.10.3. 4 Simulación de la condición inicial No.192.

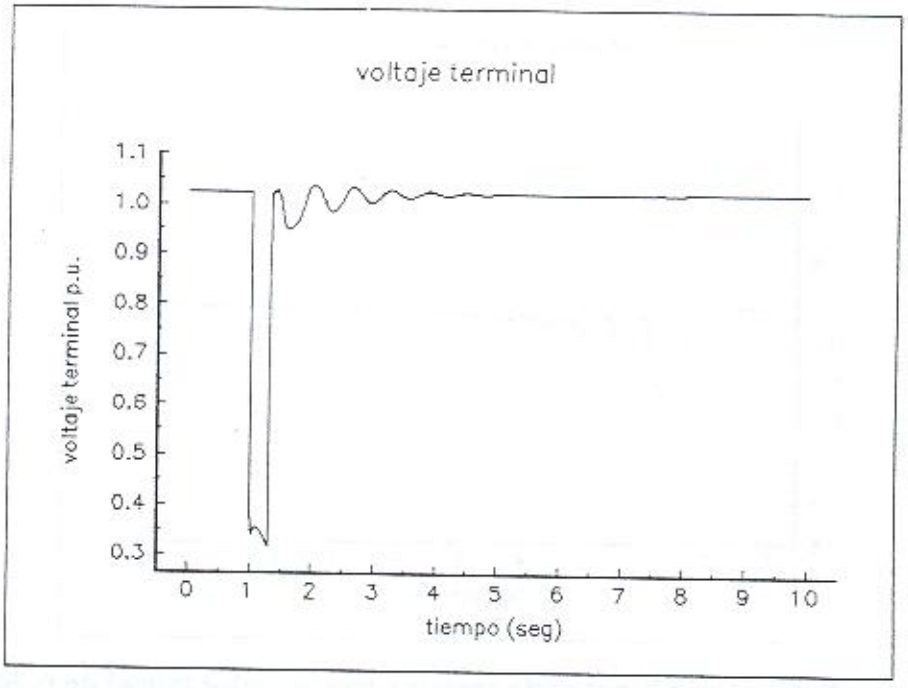


Figura 4. 99 Comportamiento del voltaje terminal con valor inicial de 1.0230 p.u.

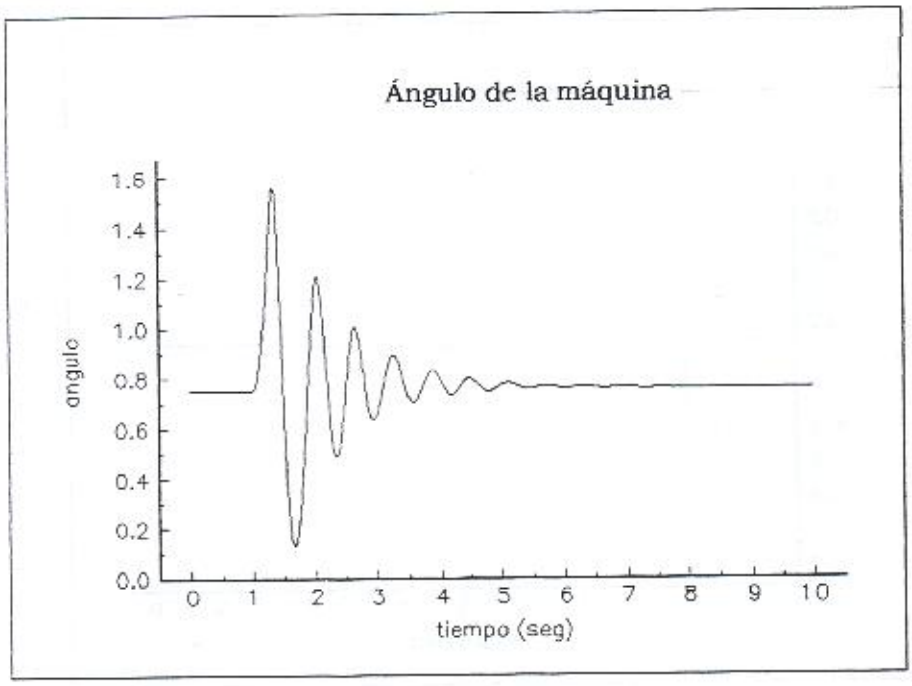


Figura 4. 100 Comportamiento del ángulo de la máquina.

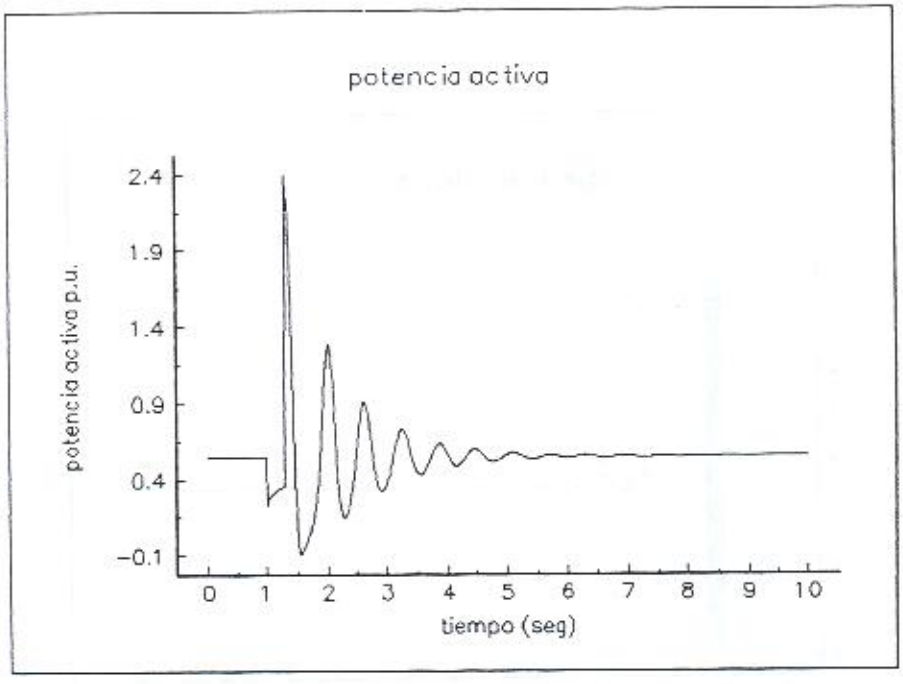


Figura 4. 101 Comportamiento de la potencia activa con valor inicial de 0.5463 p.u.

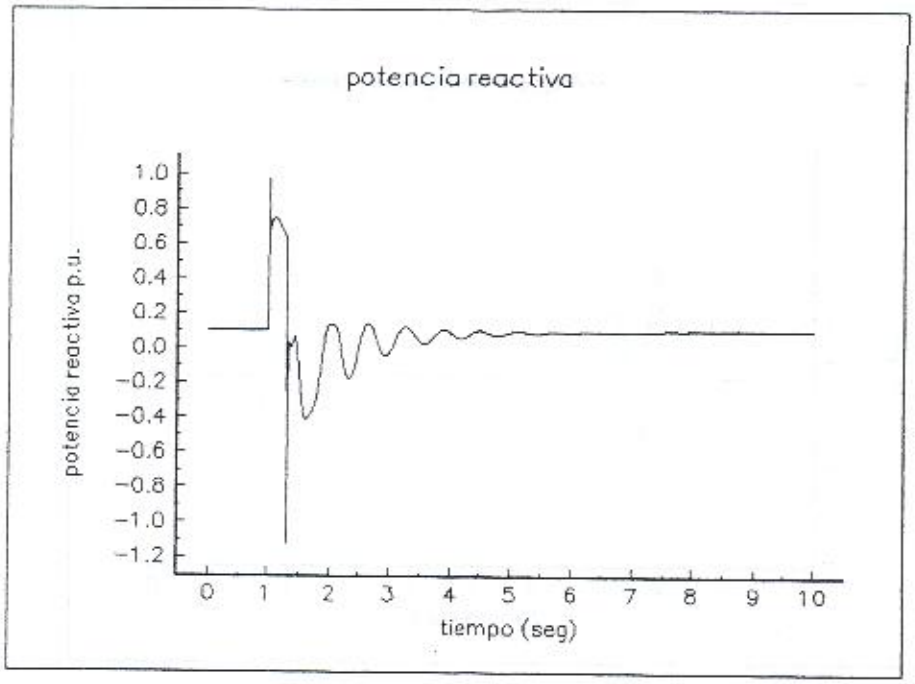


Figura 4. 102 Comportamiento de la potencia reactiva con un valor inicial de 0.1060 p.u.

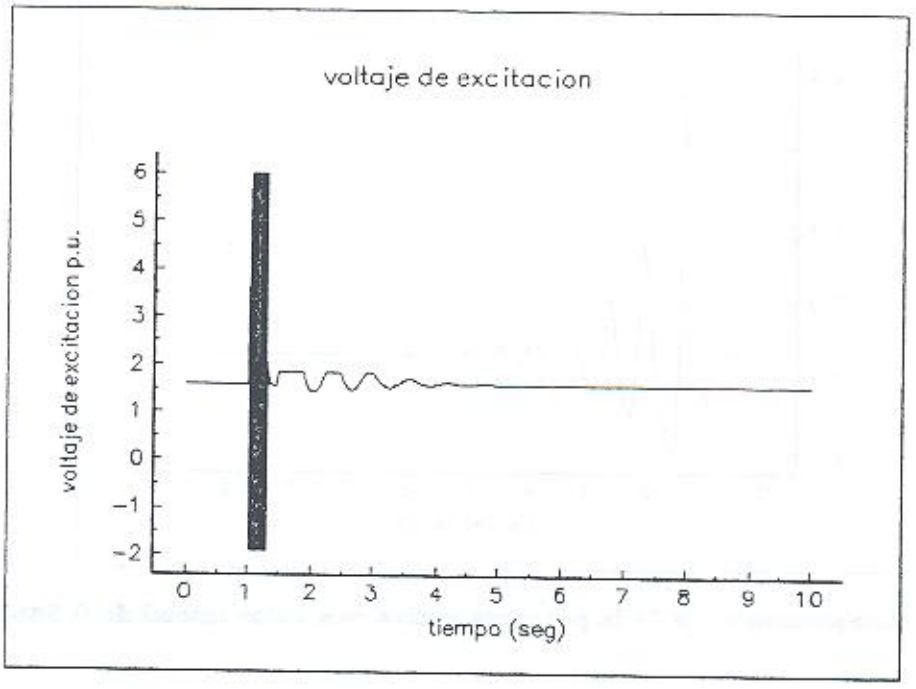


Figura 4. 103 Comportamiento del voltaje de campo.

4.10.3. 5 Simulación de la condición inicial No.197.

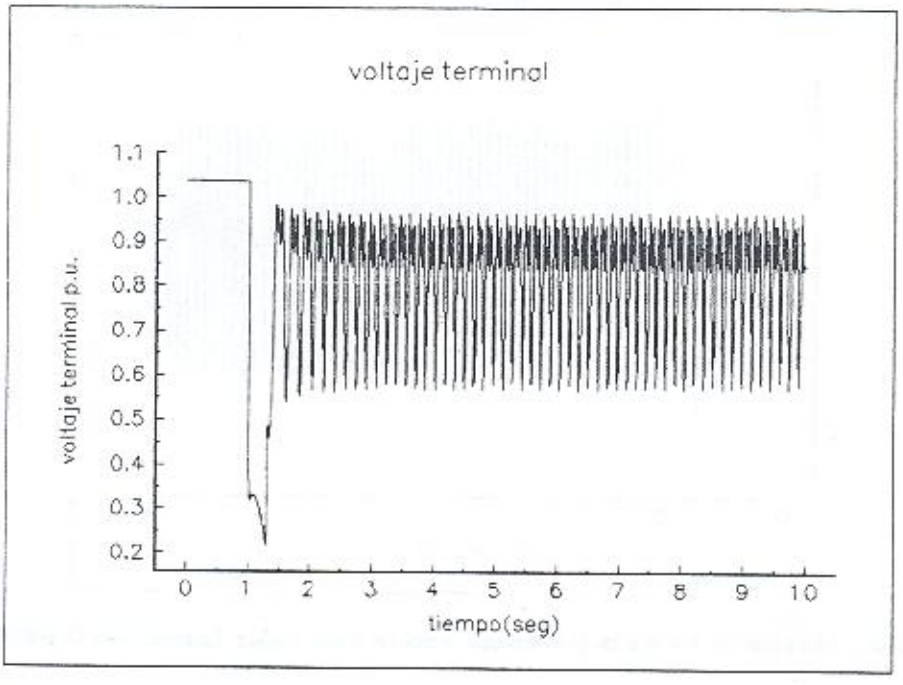


Figura 4. 104 Comportamiento del voltaje terminal con valor inicial de 1.038 p.u.

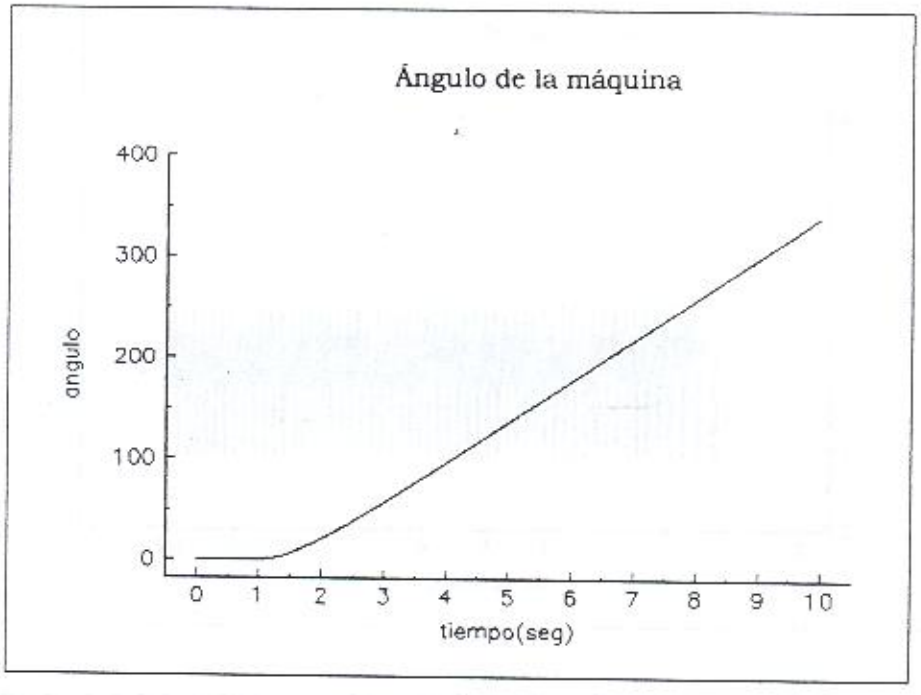


Figura 4. 105 Comportamiento del ángulo de la máquina.

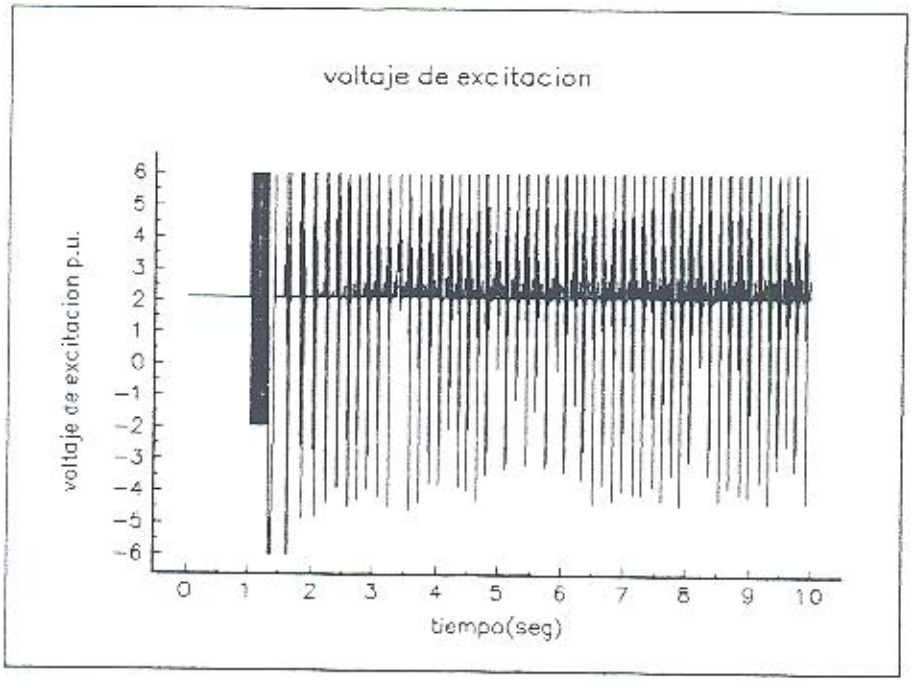


Figura 4.108 Comportamiento del voltaje de campo.

CAPITULO 5

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

5. 1 Conclusiones.

5.1. 1 Conclusiones generales.

1. Se presentó los fundamentos de la Teoría de Resonancia Adaptable que representan una solución para el diseño de sistemas que pudieran desempeñarse autónomamente en ambientes en los que se presenten eventos inesperados.

2. Se presentó la estructura FALCON-ART que contiene las propiedades de la Teoría de Resonancia Adaptable que permite construir las clases en dicha estructura, mediante sistemas de orientación y atención que evitan que el sistema se colapse.

3. Se presentó la propuesta de entrenamiento para la aplicación de la estructura FALCON-ART como un regulador de voltaje de campo de un modelo de máquina síncrona en condiciones de falla.

4. Se diseñó la subrutina CONTROL_FALCON_ART descrita en el Apéndice D, que contiene la estructura FALCON-ART en Lenguaje Fortran, y que puede ser aplicada como un sistema de control a variados tipos de plantas.

5.1. 2 Sobre la pruebas efectuadas.

Las pruebas demostraron que el sistema de regulación FALCON-ART, mejora substancialmente la regulación de voltaje del modelo de máquina utilizado, al ser comparado con el sistema de regulación ST1.

La siguiente tabla presenta los resultados de pruebas realizadas al sistema. En todos los casos el sistema FALCON-ART reguló el 100% de las condiciones iniciales presentadas en el apéndice F.

Condiciones iniciales simuladas	Tiempo de aplicación de falla (ciclos)	Cond. Inic. Reguladas por el sistema ST1	Cond. Inic. Reguladas por el sistema FALCON-ART.
200	2.0	198	200
200	3.0	196	200
200	4.0	187	200
200	5.0	177	200
200	6.0	172	200
200	7.0	171	200
200	8.0	168	200

Tabla 5.1 Concentrado de pruebas.

Como demuestran las gráficas del capitulo anterior, las oscilaciones producidas por la falla en las variables graficadas, son menores en número y amplitud, lo que permitiría disminuir los esfuerzos eléctricos y mecánicos en el sistema de generación.

5. 2 Aportaciones.

1. Se provee una estructura que obtiene los elementos necesarios para un control difuso tradicional, sin la necesidad de proporcionarle etiquetas lingüísticas o reglas difusas.
2. Se aporta un control de voltaje de campo para una máquina síncrona, que es más robusto en condiciones de falla, y que además, puede ser aplicado a otro tipo de plantas.

3. Se aporta un esquema de entrenamiento que es sencillo de implementar en otras plantas, y que podría permitir corroborar resultados de trabajos relativos al trabajo presentado.

5. 3 Trabajos futuros.

1. Uno de los trabajos que se proponen por el autor es la elaboración de una estructura FALCON-ART que contenga eliminación de reglas en línea, ya que la estructura presentada en este trabajo, posiblemente contendrá clases contenidas en otras, pudiendo de esta manera aumentar el número de clases, sin que hubiera necesidad de ello.

2. Un trabajo que pudiera ser útil también, sería la aplicación de la estructura FALCON-ART a otro tipo de plantas a controlar para verificar su eficacia en otras condiciones.

3. Se propone así mismo, un programa didáctico en el que se pueda conjuntar la serie de redes (y otras) que fue necesario desarrollar para lograr la estructura FALCON-ART y que no fueron presentadas por razones de tiempo y espacio. Esto con la finalidad de enseñar a las generaciones futuras una alternativa de desarrollo de trabajos.

Fuentes consultadas.

[1] Gail A. Carpenter y Stephen Grossberg, "**The ART of Adaptive Pattern Recognition by a Self-Organizing Neural network**", IEEE Trans. Fuzzy Syst., Vol. 1, pág. 77-88, Feb. 1988. Boston University.

[2] Gail A. Carpenter, Stephen Grossberg y David B. Rosen, "**Fuzzy ART: An Adaptive Resonance Algorithm for rapid stable classification of analog patterns**", IEEE Transaction on Computer System, Vol. 2, pág. 411-416, 1991.

[3]Gail A. Carpenter,Stephen Grossberg, "**A Self-Organizing Neural Network for Supervised Learning, Recognition, and predicction**", IEEE Communication Magazine Vol. 2, págs. 38-49, 1992.

[4] Gail A. Carpenter,Stephen Grossberg, David B. Rosen, Natalya Markuzon, and John H. Reynolds "**Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps**", IEEE Transaction on Neural Networks., Vol. 3, pág. 698-713, Sep.1992.

[5] Chen-Jian Lin, and Ching-Teng Lin "**An ART-based Fuzzy Adaptive Learning Control Network**" IEEE Transactions on Fuzzy Systems, Vol. 5 No. 4, November 1997.

[6] GAIL A. Carpenter, STEPHEN Grossber "**A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine**" Center for Adaptative Systems, Departament of Mathematics, Boston University, Boston. March 3, 1986.

[7]FREEMAN A. James y SKAPURA M. David.(1996) "**REDES NEURONALES, algoritmos, aplicaciones y técnicas de programación**" Editorial Addison-Wesley/Díaz de Santos. 1a. Edición. E.U.A.(índice).

[8] Stuart Russell, and Peter Norvig "**Artificial Intelligence, A modern Approach**" Editorial Prentice Hall 2ª. Edición E.U.A. 932 págs. 1998.

[9] Jose´Rámon Hilera González y Víctor José Martínez Hernández "**Redes Neuronales Artificiales. Fundamentos, Modelos y aplicaciones**" Editorial Addison-Wesley Iberoamericana 2ª. Edición E.U.A. 373 págs. 1995.

[10]Kevin M. Passino y Stephen Yurkovich "**Fuzzy Control**" Editorial Adisson Wesley. 2a. Edición. E.U.A. 475 págs. 1998

[11]Simon Haykin "**NEURAL NETWOKS, A comprehensive foundation**" Editorial Prentice Hall. 2ª. Edición E.U.A.

[12] Ching-Teng Lin y George Lee "**Neural Fuzzy Systems**" Editorial Prentice Hall. 2a. Edición. E.U.A. 797 págs.

[13] Julián Mendoza Fernández "**Compensación por avance de fase neurodifuso para la estabilidad de sistemas eléctricos de potencia**" Tesis de maestría. 1998. SEPI-ESIME-IPN

[14] Francisco Javier Villanueva Magaña "**Estimación robusta de parámetros para control adaptable de Excitación de generadores síncronos**". Tesis de maestria. 1997. SEPI-ESIME-IPN.

[15] Marco Vinicio Magallón Valderrama "**Sintonización de un estabilizador de sistemas eléctricos de potencia por medio de redes neuronales**". Tesis de maestría. 1997. SEPI-ESIME-IPN.

[16] Daozi Xia y G.T. Heydt "**Self-Tunning controller for generator excitation control**". IEEE Transactions on Power Apparatus and Systems, Vol PAS-102, No. 6 June 1983.

[17]Allen J. Wood y Bruce F. Wollenberg **"Power Generation, Operation and Control"** Editorial Wiley Interscience 2ª. Edición. E.U.A. 569 págs.

[18]G. Guo, Y. Wang, K.-Y. Lim y L. Gao **"Robust nonlinear controller for power system transient stability enhancement with voltage regulation"**, IEEE Procedures.- Generation, Transmission and Distribution. Vol. 143, No. 5, September 1996.

[19]L.K. Wong, F.H.F. Leung y P.K. Tam **"A Neuro-fuzzy controller Applying to a Cuk Converter"** Industrial Electronics, Control and Instrumentation, 1995. Proceedings of the 1995 IEEE IECON. 21st. International Conference on Volume 1, págs. 446-450.

[20]B. Bona, S. Carabelli, y M. Chiaberge **"Hybrid Neuro-Fuzzy System for Control of Complex Plants"** Industrial Electronics, 1998. Proceedings ISIE '98 IEEE International Symposium on published 1998. Vol. 1, págs, 87-92.

[21] Taylor, Y; greenhough, M. **"S/sub-ART: a modified ART 2ª algorith with rapid intermediate learning capabilities"** Neural Networks, 1994. IEEE World Congress on computacional Intelligence Vol. 2, pages 606-611

[22] Goergiopoulos, M; Dagher, Y; Heileman, G.L.; Bebis G. " **Properties of learning of a fuzzy ART variant"** Neural Networks, 1997. International conference, Vol. 3, pages 2012-2016.

[23] Frank, T; Kraiss, K-F; Kuhlen, T. **"Comparative analysis of fuzzy ART and ART-2ª network clustering performance"** Neural Networks, IEEE Transaction on May 1998, Vol: 93, pag. 544-559.

[24] Penny Pei Chen, Wei-Chung Lin; Hai-Lung Hong " **Multi-resolution fuzzy ART neural networks"** Neural Networks, 1999, IJCNN '99. International Joint conference, vol. 3, pages 1993-1978.

APÉNDICE A

SOFTWARE

A. 1 Introducción.

La realización de la presente tesis, requirió el diseño de un software que proporcionara la facilidad de observar el comportamiento del sistema diseñado al mismo tiempo se ejecutaban las pruebas. Durante el diseño de redes neuronales y durante su aplicación a casos específicos, pueden generarse gran cantidad de datos que deben ser analizados. El tiempo de cómputo que se requiere para este análisis puede convertirse en un obstáculo para lograr un trabajo de investigación.

La versatilidad que presenta los diferentes compiladores en la actualidad, permiten construir un software para cada caso específico. Es así, como se fue construyendo el software que se presenta en esta sección. Su diseño se fue realizando de acuerdo a las necesidades que debían ser cubiertas para una mayor rapidez en el análisis de las pruebas realizadas. La adquisición de datos, la visualización de estos en un medio gráfico era de vital importancia en esta tesis.

La gran cantidad de pruebas realizadas para esta aplicación, su registro y el tiempo de computo se fue optimizando conforme el software se iba ajustando de acuerdo a las necesidades del autor.

Cabe mencionar que el software, sólo se utiliza como un indicador para poder decidir si la prueba ejecutada es de utilidad. El software proporciona el comportamiento del sistema en una forma generalizada. Las gráficas obtenidas y presentadas en el capítulo 4 de la tesis, fueron realizadas con un graficador comercial.

A. 2 Desarrollo.

El software se diseño en su totalidad en Microsoft Developer Studio, y su ejecución permite apreciar el comportamiento del generador durante condiciones de falla. La falla puede ser iniciada en el tiempo que el usuario lo desee, siendo recomendable un tiempo de aplicación entre 0.1 y 1.0 seg. La razón de esto es que el tiempo total de simulación es de 10 seg. y el comportamiento de la máquina puede llegar a ser inadvertido si se aplica una falla después de estos límites de tiempo. El usuario también puede proporcionar al programa el número de ciclos al cual desea liberar la falla. El programa proporcionó buenos resultados durante la realización de pruebas, las cuales se limitaron a un tiempo de liberación de 18 ciclos; después de haberse depurado el software, no se presentaron fallas durante su ejecución.

El proyecto recibió el nombre de PROYECTO_FINAL_DE_TESIS, y al ser ejecutado, en la pantalla aparecerán las preguntas siguientes:

- ¿ Cuántas condiciones iniciales desea simular?
- ¿ Cuál es su tiempo inicial de falla?
- ¿ En cuantos ciclos desea eliminar la falla?

A.2. 1 ¿Cuántas condiciones iniciales desea simular?

En esta interrogante, el usuario proporciona un número entre 1 y 200 condiciones iniciales. En caso de que el usuario proporcione una cantidad menor a 1, el programa continuará con las preguntas siguientes, y finalmente enviará la pregunta siguiente:

DESEA REPETIR LA PRUEBA NUEVAMENTE SI[1] NO [2] ?

Si el usuario presiona 1, la pregunta anterior se seguirá imprimiendo en pantalla; por lo tanto, el usuario deberá presionar 2, para que el programa se termine. Si el usuario da una cantidad mayor a 200, el programa ejecutará todas las condiciones iniciales, y una vez que haya finalizado la condición inicial No. 200, enviará un mensaje de error donde indica que el archivo de datos ha finalizado y cerrará el programa en ejecución.

A.2. 2 ¿ Cuál es su tiempo inicial de falla?

En esta pregunta, el usuario proporciona el momento de inicio de falla, una vez iniciada la simulación. Se recomienda usar valores entre 0.01 y 1.0 seg.

A.2. 3 ¿ En cuantos ciclos desea eliminar la falla?

En esta pregunta, se le da al programa el número de ciclos en los que desee liberar la falla, se recomienda utilizar valores entre 3.0 y 15.0 ciclos.

A. 3 Ventanas mostradas por el software

El software genera un grupo de pantallas durante su ejecución, el total de estas es 5. Se presenta una pantalla inicial donde aparecen todas las preguntas del segmento anterior, y se presenta otras 4 pantallas donde se grafican: el voltaje terminal, el voltaje de campo, la potencia activa y la potencia reactiva. Estas gráficas muestran el comportamiento de variables anteriores con los sistemas de regulación FALCON-ART y el sistema ST1, simultáneamente; es decir, en una sola pantalla puede apreciarse el voltaje terminal con el sistema ST1 (graficado en rojo) y con el sistema FALCON-ART (graficado en azul).

La pantalla inicial aparece así una vez que las preguntas de la sección anterior han sido contestadas:

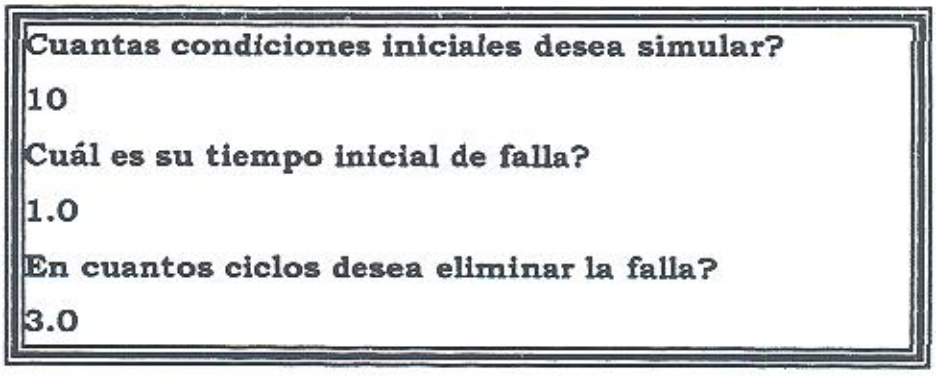


Figura A. 1 Pantalla inicial del programa PROYECTO_FINAL_DE_TESIS.

En ella se muestran como respuestas del usuario: 10 condiciones iniciales a simular, un tiempo inicial de falla de 1.0 seg. y un tiempo de liberación de 3.0 ciclos.

Una vez introducidos estos datos, el programa desplegará 4 ventanas más. Estas ventanas llevan los título siguientes:

- COMPORTAMIENTO DEL VOLTAJE DE CAMPO
- COMPORTAMIENTO DEL VOLTAJE TERMINAL
- COMPORTAMIENTO DE LA POTENCIA ACTIVA
- COMPORTAMIENTO DE LA POTENCIA REACTIVA.

En estas ventanas, se aprecia el comportamiento de la variable mencionada en el título de la ventana; es decir, la ventana titulada COMPORTAMIENTO DEL VOLTAJE DE CAMPO, mostrará el comportamiento del voltaje de campo, y así con las demás ventanas.

El programa proporciona el comportamiento del sistema FALCON-ART y del sistema de regulación ST1, por lo que en cada condición inicial se aprecia la diferencia de comportamiento de las variables mencionadas anteriormente con cada uno de estos sistemas.

Durante el simulado de cada condición inicial, el usuario puede intercambiar las ventanas para apreciar el comportamiento de las variables. Al terminarse de graficar todas las variables en su ventana respectiva, el programa realizará una pausa para permitir la revisión de las ventanas. Esta pausa se indicará en la pantalla inicial, y aparecerá con los textos siguientes:

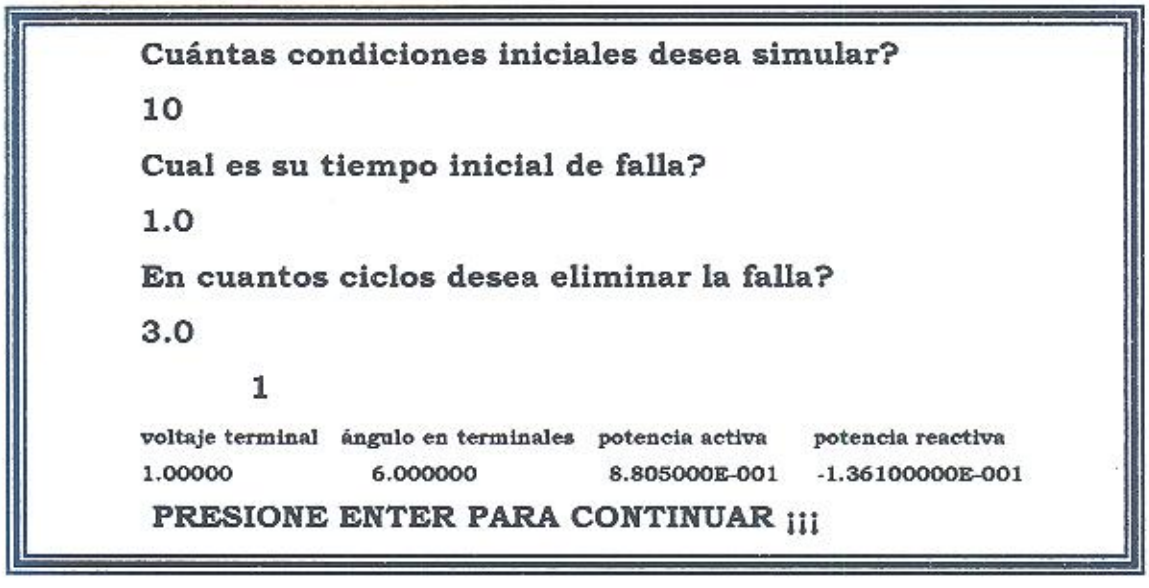


Figura A. 2 Pantalla inicial con una condición inicial simulada.

En la pantalla se indicará el número de condición inicial simulada, así como también el valor inicial de las variables: voltaje terminal, ángulo terminal, potencia activa y potencia reactiva. Al presionar ENTER, el programa continuará con la siguiente condición inicial. En caso de no haber una nueva condición inicial para simular, se desplegará la pregunta siguiente:

DESEA REPETIR LA PRUEBA NUEVAMENTE SI[1] NO [2] ?

Si la respuesta del usuario a esta pregunta es sí, el programa simulará una cantidad de condiciones iniciales idéntica a la solicitada anteriormente por el usuario. Una vez finalizada tal simulación, el programa volverá a desplegar la pregunta anterior, hasta obtener una respuesta negativa por parte del usuario. Si es el caso, el programa finalizará y todas las pantallas serán cerradas.

A. 4 Ejemplo.

Se desea observar el comportamiento de 1 condición inicial, con una falla aplicada en 0.2 seg. y despejarla en 4.0 ciclos.

Se inicia el programa y la pantalla inicial será la siguiente:

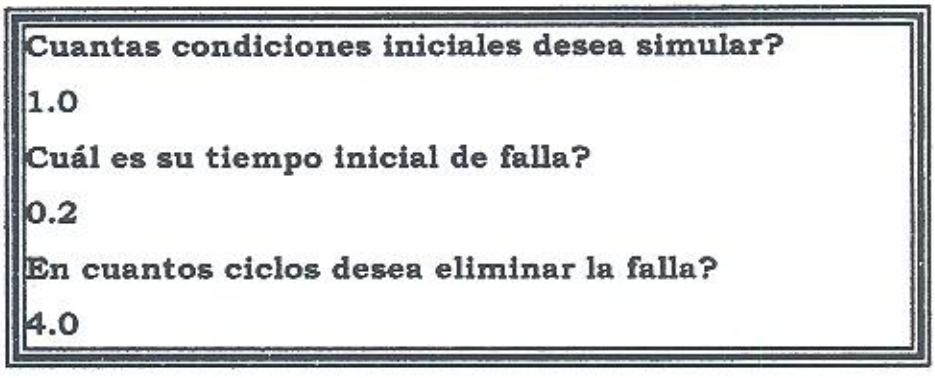


Figura A. 3 Pantalla inicial con una condición simulada, con una falla en 0.2 seg. y con una duración de 4 ciclos.

Se introducen los datos anteriores y se obtendrán las pantallas siguientes:

En la figura A.4 se grafica la potencia activa del generador sincrono con el sistema de regulación ST1 y con el sistema FALCON-ART.

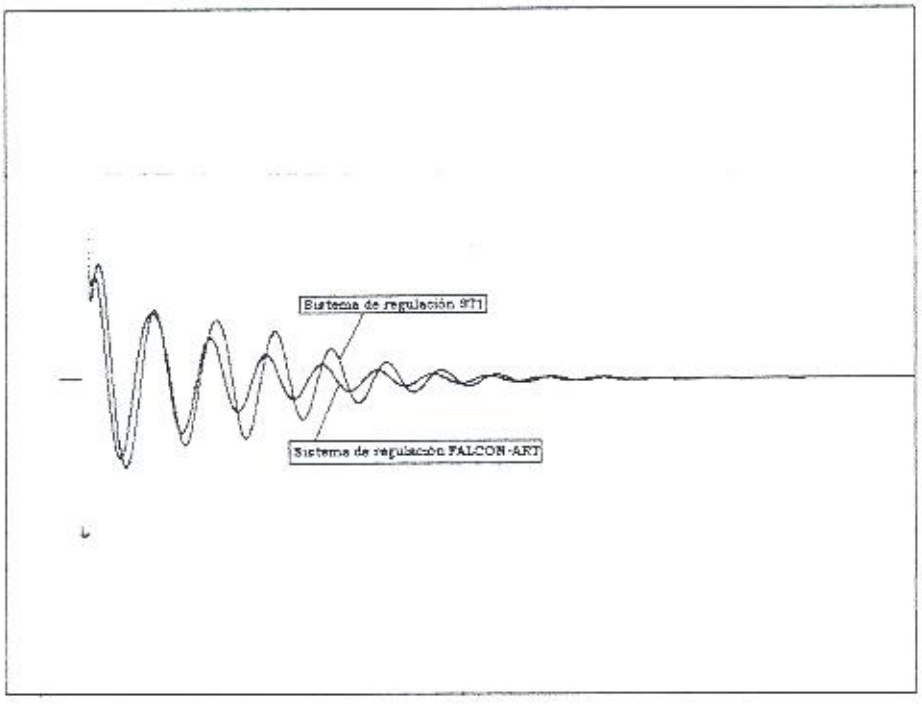


Figura A. 4 Pantalla COMPORTAMIENTO DE LA POTENCIA ACTIVA.

En la pantalla de la figura A.5 se grafica la potencia reactiva del generador síncrono con el sistema de regulación ST1 y con el sistema FALCON-ART.

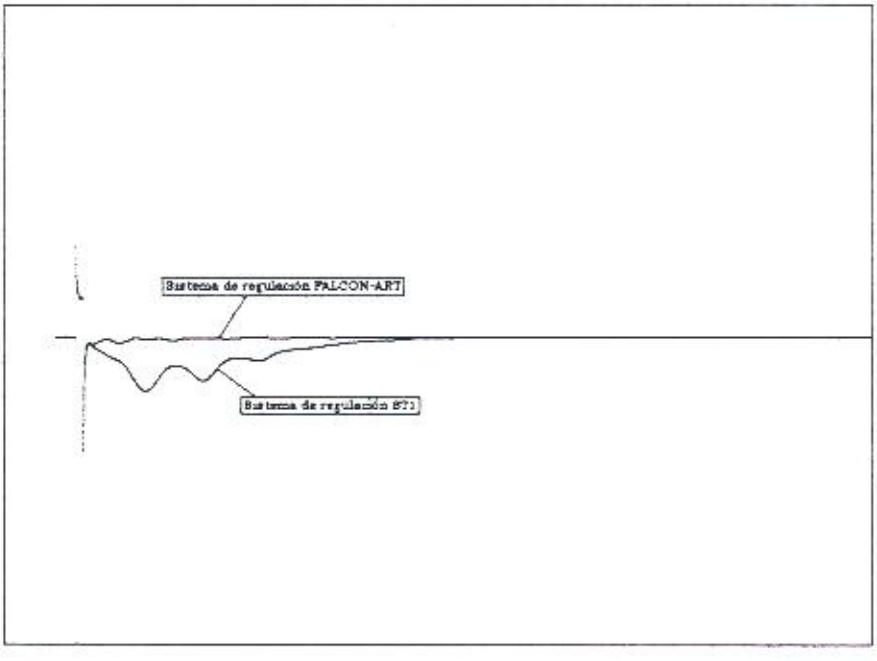


Figura A. 5 Pantalla COMPORTAMIENTO DE LA POTENCIA REACTIVA.

En la pantalla de la figura A.6 se grafica el voltaje de campo del generador síncrono con el sistema de regulación ST1 y con el sistema FALCON-ART.

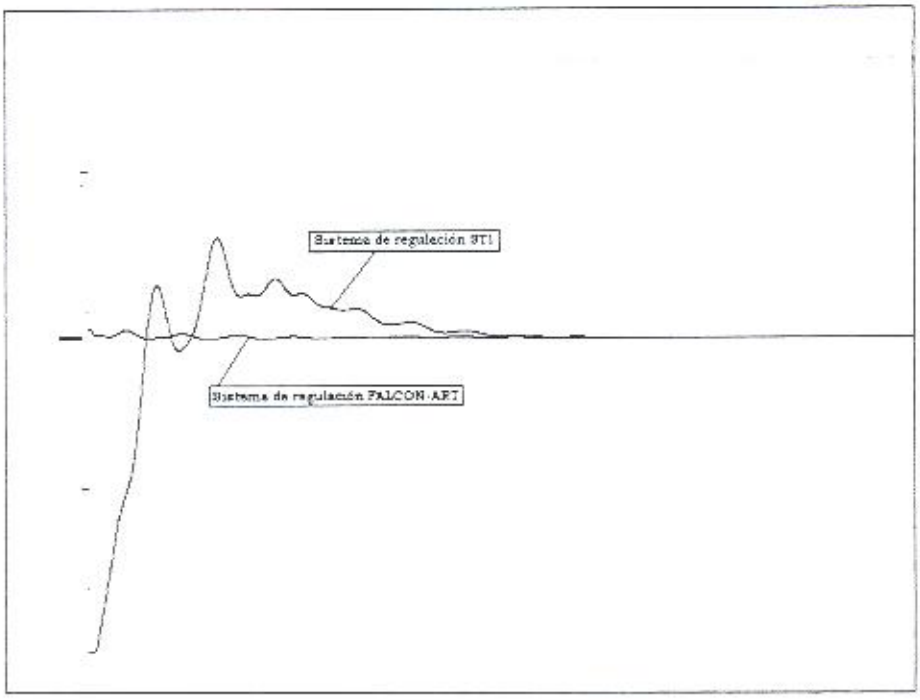


Figura A. 6 Pantalla COMPORTAMIENTO DEL VOLTAJE DE CAMPO.

En la figura de la figura A.7 se grafica el voltaje terminal del generador síncrono con el sistema de regulación ST1 y con el sistema FALCON-ART.

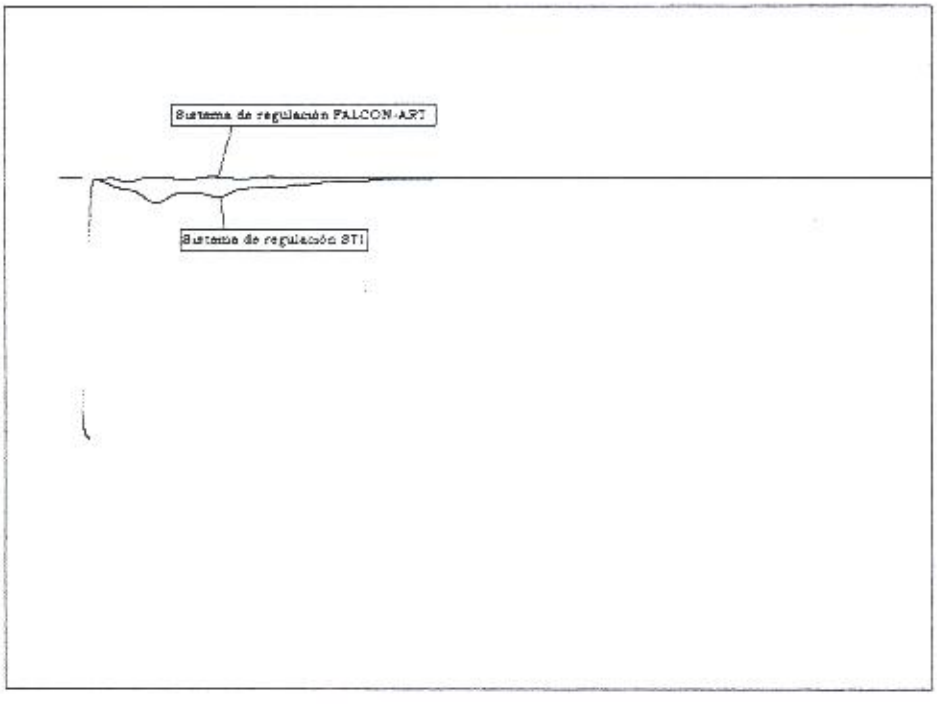


Figura A. 7 Pantalla COMPORTAMIENTO DEL VOLTAJE TERMINAL.

El software diseñado permitió la observación de una cantidad considerable de pruebas. Una vez finalizado en entrenamiento del sistema FALCON-ART, el software permitía observar rápidamente las 200 condiciones iniciales consideradas para prueba. El software no es un graficador muy exacto; sin embargo, permitia observar en forma bosquejada, si el entrenamiento habia logrado buenos resultados. Una vez corroborado el buen funcionamiento del sistema, se registraban los parámetros que habían sido escogidos para el entrenamiento.

APÉNDICE B

SISTEMA DE PRUEBA

B. 1 Modelo Matemático[39].

B.1. 1 Modelo de la máquina síncrona.

Las ecuaciones diferenciales del modelo de 5° orden, usadas en esta tesis son:

$$p\delta = w_0 s \qquad \textbf{B.1}$$

$$Mps = -Kds + T_m - T_e \qquad \textbf{B.2}$$

$$T'dope'q = V_f - (Xd - X'd)id - e'q \qquad \textbf{B.3}$$

$$T''dope''q = e'q - (X'd - X''d)id - e''q \qquad \textbf{B.4}$$

$$T''qope''d = (Xq - X''q)iq - e''d \qquad \textbf{B.5}$$

$$\textbf{B.6}$$

$$e''d = Vd + raid - X''qiq \qquad \textbf{B.7}$$

$$e''q = Vq + raiq + X''did$$

$$Te = e''did + e''qiq - (X''d - X''q)idiq \qquad \textbf{B.8}$$

[39] Las ecuaciones referenciadas en este apéndice, así como los datos del sistema simulado fueron obtenidos de la Ref. 16

B.1. 2 Modelo de la Red.

Sin pérdida de generalidad, la línea de transmisión se describe por una impedancia equivalente de Thevenin. Por lo tanto, el voltaje terminal y sus componentes en eje directo y eje en cuadratura, son las siguientes:

$$Vd = Vb\,\mathrm{sen}\,\delta + reid - Xeiq \qquad \textbf{B.9}$$

$$Vq = Vb\cos\delta + reiq + Xeid \qquad \textbf{B.10}$$

$$V_T^2 = Vd^2 + Vq^2 \qquad \textbf{B.11}$$

B.1. 3 Modelo de la turbina de vapor y del gobernador de velocidad.

El gobernador de velocidad que aquí se utiliza es del tipo mecánico-hidraúlico, el modelo está idealizado, puesto que no se está considerando el limitador de posición de la válvula, así como tampoco la función de transferencia que describe el relevador de velocidad, quedando únicamente representado por el bloque (b) de la figura B.1.

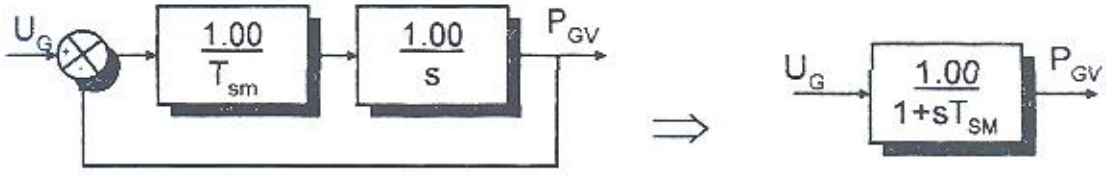


Fig. B.1 Diagrama de bloques del gobernador de velocidad mecánico-hidraúlico

En el caso de la turbina, se utilizó una turbina térmica de simple recalentamiento modificada. Se usa únicamente el modelo gobernador-turbina con los elementos básicos necesarios, para realizar las funciones de incremento y decremento de potencia activa, mediante la manipulación de la entrada al gobernador Ug sin hacer control sobre el lazo primario. Por lo tanto no se necesitan modelos más detallados en esta parte del sistema. Así, el conjunto de ecuaciones diferenciales que describen este sistema son las siguientes:

$$T_{SM}\,pP_{GV} = Ug - P_{GV} \qquad \textbf{B.12}$$

$$T_{CH}\,pP_{HP} = P_{GV} - P_{HP} \qquad \textbf{B.13}$$

$$T_{RH}\,pT_m = F_{HP}\,pP_{HP} + P_{HP} - T_m \qquad \textbf{B.14}$$

B.1. 4 Modelo del sistema de excitación y regulador de voltaje.

El sistema de excitación empleado es del tipo rotatorio, este sistema queda representado por la ecuación diferencial siguiente:

$$T_x pV_f = u - V_f \qquad \textbf{B.15}$$

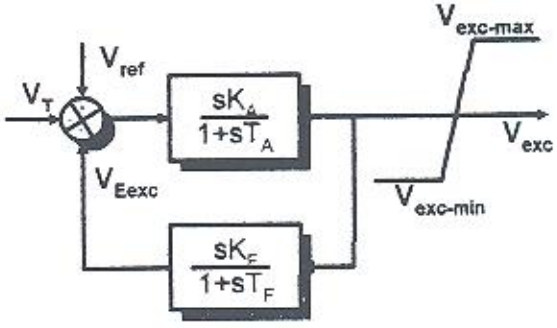


Fig. B.2 Diagrama de bloques del regulador tipo ST1

El sistema de regulación de voltaje que se usa en este trabajo, es el regulador tipo ST1 (excitatriz con fuente de potencial a través de rectificadores controlados), propuesto por el comité sobre sistemas de excitación de la IEEE para la mayoría de los sistemas de excitación modernos como el considerado aquí. Tomando en cuenta que las constantes de tiempo de los transformadores de potencial para acondicionar la señal de voltaje terminal a niveles apropiados, son del orden de 0.0001 segundos, en este trabajo se desprecia. En este tipo de sistema el voltaje límite es proporcional al voltaje generado, por ésta razón no se considera el efecto de la saturación de la excitatriz. Para obtener las ecuaciones de estado del sistema excitación-regulación, se parte del diagrama de bloques de la figura B.2, que representa este regulador.

Por lo tanto las ecuaciones son:

$$T_A pV_f = K_A(V_{ref} - V_T - V_{Eexc}) - V_f \qquad \textbf{B.16}$$

$$T_f pV_{Eexc} = K_f pV_f - V_{Eexc} \qquad \textbf{B.17}$$

Para todos los casos:

$$p = \frac{d}{dt}$$

B.1. 5 Datos del sistema bajo prueba.

Los datos del sistema máquina-bus infinito se muestran en la tabla B.1. corresponden a los datos de un sistema[40] de 330 kV y 400MW conectados por una línea corta con una impedancia Ze= 0.115 + j0.024

Variable	Valor	Variable	Valor
W_O	377.0	Xe	0.115
M	5.5294	Re	0.024
Kd	3.0	T_{SM}	0.1
T'do	5.66	T_{CH}	0.15
T"do	0.041	T_{RH}	0.15
T"qo	0.065	F_{HP}	0.33
Xd	1.904	K_A	400.0
X'd	0.312	T_A	0.02
X"d	0.266	K_F	0.008
Xq	1.881	T_F	1.0
X"q	0.260	Tx	0.025
Ra	0.0		

Tabla B.1. Datos del sistema bajo prueba

B. 2 Determinación de las condiciones iniciales y de las ecuaciones dinámicas para el sistema máquina bus-infinito.

Para determinar las condiciones iniciales, se parte del sistema máquina-bus infinito que se muestra en la Fig. B.3, y del diagrama vectorial de la máquina síncrona en estado subtransitorio Fig. B.4 y de las ecuaciones para el modelo subtransitorio.

[40] Sistema tonado de la Ref. 16

B.2. 1 Red.

Se considera que la máquina síncrona está conectada a un bus infinito mediante una impedancia externa Zl=re+jXe representada por una línea de transmisión, como se muestra en la Fig. B.3.

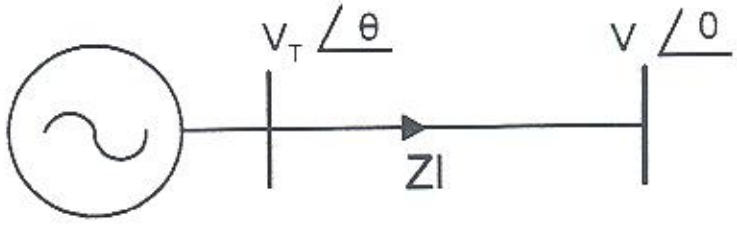


Figura B.3 Diagrama del sistema máquina-bus infinito utilizado en este trabajo

Voltaje terminal. El voltaje terminal (V_T) y su ángulo (θ),son datos conocidos previamente. Para determinar las componentes real (e_Tr) e imaginaria (e_Ti), se trabaja el voltaje terminal en forma compleja, quedando de la siguiente forma:

$$V_T = e_T r + j e_T i \qquad \textbf{B.18}$$

Donde:

$$e_T r = VT\cos\theta$$
$$e_T i = VT\,\mathrm{sen}\,\theta$$

Voltaje de Bus. El voltaje del bus, también es un dato conocido con anterioridad. Este se trabaja en forma compleja y queda de la siguiente forma $Vb = Vb\angle 0$.

Potencia Eléctrica. Se sabe que, S=V*I=P-jQ, por lo tanto:

$$S = V^* \frac{V_T - V_b}{Zl} \qquad \textbf{B.19}$$

$$P = \mathrm{Re}(S) \qquad \textbf{B.20}$$

$$Q = -\mathrm{Im}(S) \qquad \textbf{B.21}$$

Corriente de Armadura: Partiendo del triángulo de potencias se tiene que:

$$S = V_T Ia \quad , \quad Ia = \frac{S}{V_T} \qquad \textbf{B.22}$$

donde $S = \sqrt{P^2 + Q^2}$

así $$I_a = \frac{\sqrt{P^2 + Q^2}}{V_T} \qquad \textbf{B.23}$$

Y el ángulo es: $$\phi = arctan \frac{Q}{P} \qquad \textbf{B.24}$$

B.2.2 Generador

Ángulo de carga δr. Para determinar el ángulo de carga se parte del diagrama vectorial de la máquina síncrona para el caso subtransitorio, correspondiente a la Fig. B.4. Los trazos auxiliares son mostrados para un mejor entendimiento en la obtención de δr.

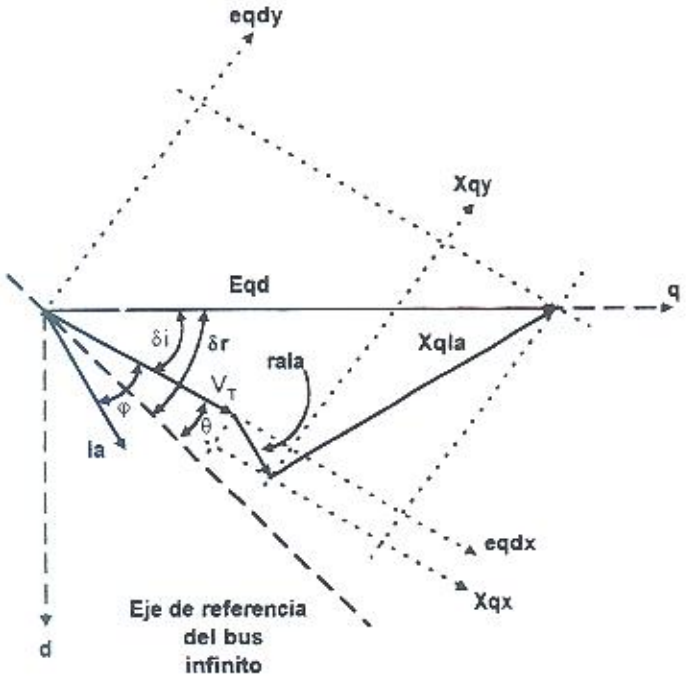


Fig. B.4 Diagrama Vectorial

Del diagrama vectorial se determina que:

$$eqdx = V_T + raia\cos\phi + xqia\,\mathrm{sen}\,\phi \qquad \textbf{B.25}$$

$$eqdy = -raia\,\mathrm{sen}\,\phi + Xqia\cos\phi \qquad \textbf{B.26}$$

Donde:

$$\delta i = arctan\frac{eqdy}{eqdx} \qquad \textbf{B.27}$$

$$\therefore \delta r = \theta + \delta i \qquad \textbf{B.28}$$

Deslizamiento s: Haciendo cero la derivada de la ecuación B.1 se obtiene:

$$0 = w_0 s \quad \therefore \quad s = 0 \qquad \textbf{B.29}$$

Voltaje transitorio en eje en cuadratura e'q: Haciendo cero la derivada B.4 y despejando para e'q, se tiene:

$$0 = e'q - (X'd - X''d)id - e''q \quad \therefore \quad e'q = (X'd - X''d)id + e''q \qquad \textbf{B.30}$$

De las ecuaciones el sistema máquina-bus infinito, para un modelo subtransitorio se tiene:

Voltaje subtransitorio en eje en cuadratura e''q:

$$e''q = V_T \cos\delta + raiq + X''did \qquad \textbf{B.31}$$

Voltaje subtransitorio en eje directo e''d:

$$e''d = V_T \operatorname{sen}\delta + raiq + X''qiq \qquad \textbf{B.32}$$

Par mecánico Tm:

$$Tm = e''did + e''qiq \qquad \textbf{B.33}$$

Par eléctrico Te:

$$Te = Tm \qquad \textbf{B.34}$$

B.2. 3 Sistema de Excitación-regulador de voltaje

Para obtener las condiciones iniciales, tanto de la excitatriz como del regulador de voltaje tipo ST1, se hacen cero las derivadas de las ecuaciones B.15, B.16 y B.17.

Cuando se utiliza el regulador adaptable, el voltaje de campo es:

$$0 = u - V_f \quad \therefore \quad V_f = u \qquad \textbf{B.35}$$

Y cuando se utiliza el regulador ST1 las condiciones iniciales son:

$$0 = K_A(V_{ref} - V_T - V_{Eexc}) - V_f \quad \therefore \quad V_f = K_A(V_{ref} - V_T - V_{Eexc}) \qquad \textbf{B.36}$$

$$0 = -V_{Eexc} \quad \therefore \quad V_{Eexc} = 0 \qquad \textbf{B.37}$$

$$V_{ref} = \frac{V_f}{K_A} + V_T \qquad \textbf{B:38}$$

Voltaje máximo y mínimo de campo en el limitador de voltaje:

$$V_{exc_{MIN}} = -6.0\,p.u.$$
$$V_{exc_{MAX}} = 6.0\,p.u. \qquad \textbf{B.39}$$

B.2. 4 Turbina de Vapor y gobernador de velocidad

Haciendo cero las derivadas en las ecuaciones B.12, B.13 y B.14, se tiene que:

$$0 = Ug - P_{GV} \quad \therefore \quad P_{GV} = Ug \qquad \textbf{B.40}$$

$$0 = P_{GV} - P_{HP} \quad \therefore \quad P_{HP} = P_{GV} \qquad \textbf{B.41}$$

$$0 = P_{HP} - Tm \quad \therefore \quad Tm = P_{HP} \qquad \textbf{B.42}$$

B.2. 5 Ecuaciones dinámicas

Cuando se simula el sistema, la primera iteración se realiza usando las condiciones iniciales; sin embargo, para las iteraciones posteriores el programa debe calcular el punto de operación para ese instante de tiempo. Para realizar esto, se deben proporcionar las ecuaciones que realicen esta función, basándose en las ecuaciones B.6, B.7, B.9 y B.10, se obtienen las ecuaciones siguientes:

Sustituyendo B.9 en B.6

$$e''d = Vb\,\mathrm{sen}\,\delta + reid - Xeiq + raid - X''qiq$$
$$e''d = Vb\,\mathrm{sen}\,\delta + (re + ra)id - (Xe + X''q)iq \qquad \textbf{B.43}$$

Sustituyendo B.10 en B.7

$$e''q = Vb\cos\delta + reiq + Xeid + raiq + X''did$$
$$e''q = Vb\cos\delta + (re + ra)iq + (Xe + X''d)id \qquad \textbf{B.44}$$

Resolviendo el sistema de dos ecuaciones simultáneas B.43 y B.44 para id e iq, se obtiene:

$$id = \frac{(re + ra)(e''d - Vb\,\mathrm{sen}\,\delta) + (Xe + X''q)(e''q - Vb\cos\delta)}{(re + ra)2 + (Xe + X''d)(Xe + X''q)} \quad \textbf{B.45}$$

$$iq = \frac{(re + ra)(e''q - Vb\cos\delta) \overline{+} (Xe + X''d)(e''d - Vb\,\mathrm{sen}\,\delta)}{(re + ra)^2 + (Xe + X''d)(Xe + X''q)} \quad \textbf{B.46}$$

$$\therefore \qquad I = (iq + jid) \quad \textbf{B.47}$$

Los voltajes Vq y Vd se obtienen se B.6 y B.7:

$$Vd = e''d - raid - X''qiq \quad \textbf{B.48}$$

$$Vq = e''q - raiq + X''did \quad \textbf{B.49}$$

$$\therefore V = (vq + jvd) \quad \textbf{B.50}$$

Las potencias son:

$$S = VI^* \quad \textbf{B.51}$$

$$P = \mathrm{Re}(S) \quad \textbf{B.52}$$

$$Q = -\mathrm{Im}(S) \quad \textbf{B.53}$$

Finalmente el par eléctrico es:

$$Te = e''did + +e''qiq \quad \textbf{B.54}$$

Por lo tanto las llamadas ecuaciones dinámicas son de la ecuación B.45 a la ecuación B.54.

APÉNDICE C

CÓDIGO DE PROGRAMAS UTILIZADOS EN LA APLICACIÓN DE LA RED FALCON-ART COMO REGULADOR DE VOLTAJE EN LA MÁQUINA SÍNCRONA

C. 1 Introducción.

En esta sección se presenta en forma total, el programa principal y las subrutinas diseñadas para obtener los resultados de este trabajo. El código en su totalidad, se conforma de 5 partes que se denominan al igual que en el programa.

- El programa principal *PROGRAM MAQUINA_CON_REGULADOR*, contiene las llamadas a subrutinas, y en general todas las estructuras que pueden llegar a sufrir un cambio en su dimensionamiento durante la ejecución del programa.
- La subrutina *SUBROUTINE CONTROL_FALCON_ART*, que representa en sí, el regulador aplicado al modelo de máquina síncrona; y de esta manera, es la estructura que se desarrolló en este trabajo en este trabajo.
- La subrutina *SUBROUTINE REGULADOR_FALCON*, incluye el regulador FALCON-ART, pero contiene únicamente la propagación de señales hacia delante, debido a que esta subrutina se aplica una vez que el entrenamiento ha sido realizado y no se necesita realizar la corrección del error.
- El módulo *MODULE DECLARACIONES*, contiene declaraciones de variables que en general, no necesitaban ser modificadas una vez que sus características habían sido determinadas, y no necesitan más, la atención del programador.

- El módulo *MODULE MAQUINA_SINCRONA*, contiene declaraciones de variables y subrutinas que se utilizan únicamente para el modelo de máquina síncrona, excepto la subrutina *SOLICITUD*, que solicita datos que determinan el comportamiento de ambos sistemas: modelo de la máquina y entrenamiento.
- El módulo *MODULE PARAMETROS_FUNDAMENTALES*, contiene como su nombre lo indica, los parámetros que determinan el comportamiento de la estructura FALCON-ART durante el entrenamiento. Estos parámetros son: El tamaño de las clases en el espacio de entrada ρ_a, el parámetro que determina el tamaño de las clases en el espacio de salida ρ_b, elparámetro que determina la velocidad con la cual la estructura FALCON-ART corregirá los pesos contenidos en los nodos, y finalmente los parámetros ENT y ENT_B, que permiten dimensionar aquellas variables que pueden variar su dimensión de acuerdo al número de entradas en el espacio de entrada y salida, respectivamente.

A continuación se documentan estos segmentos de programa mencionados anteriormente, su descripción se realiza de acuerdo al orden con el cual fueron descritos en los párrafos anteriores. En la documentación, se intenta proporcionar la mayor información posible para que el lector pueda percibir la operación realizada en cada una de las partes del programa. Los líneas con **negritas** son únicamente comentarios y pueden omitirse si se desea correr el programa.

C. 2 PROGRAMA PRINCIPAL

```
!*************************************************************
*****************
!********************      PROGRAMA MAQUINA_CON_REGULADOR
*****************

!*************          PROGRAMADOR: JACOB EFRAÍN DIAZ
LAVARIEGA ************

!************      FECHA DE ELABORACIÓN: SEMESTRE ENE/JUN
2000 **********
!
!*******     OBJETIVO: DISEÑARSE LA RED FALCON-ART Y
APLICARSE COMO *****

!** UN REGULADOR DE VOLTAJE DE CAMPO EN UN MODELO DE MÁQUINA
SÍNCRONA**
!*************************************************************
************
!*****************                    INICIO DEL PROGRAMA
************************
program Maquina_con_regulador
!*************************************************************
************
!SE DECLARAN LOS DIFERENTES MÓDULOS QUE SON REQUERIDOS EN
!EL PROGRAMA.ESTO ES IMPRECINDIBLE CUANDO SE UTILIZAN
MÓDULOS
!EN CUALQUIER TIPO DE PROGRAMA EN EL COMPILADOR FORTRAN
!POWER STATION. LA UTILIDAD DE LOS MÓDULOS PORTLIB Y MSLIB
SE
!DESCRIBE EN LA AYUDA DEL COMPILADOR MENCIONADO.
```

```
!*************************************************************
*****************
USE PORTLIB
USE MAQUINA_SINCRONA
USE MSFLIB
USE DECLARACIONES
!*************************************************************
*****************
!SE DIMENSIONAN LAS VARIABLES: ENTRADA Y META.
!DICHAS VARIABLES SON LA VARIABLE QUE RECIBE LA
ESTRUCTURA
!FALCON-ART PARA LLEVAR A CABO SUS OPERACIONES Y LA
VARIABLE QUE ESTA
!ESTRUCTURA ENTREGA AL SISTEMA COMO RESULTADO DE SUS
OPERACIONES,
!RESPECTIVAMENTE
!*************************************************************
*****************
real*8,dimension(ent+ent_b)   ::  entrada
real*8,dimension(ent_b)       ::  meta
!LAS SIGUIENTES VARIABLES SÓLO SON UTILIZADAS PARA
!FINES DE GRAFICACIÓN.
real*8 help, help1, suma1, suma2, suma3, suma4
!LAS 4 DECLARACIONES SIGUIENTES REPRESENTAN
!MAX Y MIN DE UNA FUNCIÓN, LAS PRIMERAS DOS SON MAX Y
MIN
!LOCALES Y LAS SIGUIENTES SON MAX Y MIN GLOBALES.
real*8 va_aux1_max,      va_aux2_max
real*8 va_aux1_min, va_aux2_min
real*8 va_aux1_MaxGral, va_aux2_MaxGral
real*8 va_aux1_MinGral,  va_aux2_MinGral
!ESTA VARIABLE SE UTILIZA PARA
!DETERMINAR EL ARCHIVO AL QUE SE DIRIGIRÁN LOS
!RESULTADOS DE LAS CORRIDAS Y PODER GRAFICARSE.
character(1) archivo
!ms$debug
!*************************************************************
*************
!**************                    SUBRUTINA     SOLICITUD
*****************************
!*************************************************************
*************
!VARIABLES DE SALIDA:
!    TINIC_FALLA = MOMENTO EN EL TIEMPO EN QUE
INICIAMOS LA
!                  FALLA EN TERMINALES DE LA MÁQUINA.
!    TDURA= MOMENTO EN EL CUAL DETENEMOS LA FALLA
INICIADA POR TINIC_FALLA
!    RESPUESTA= ES LA VARIABLE QUE NOS INDICARÁ SI SE
ESTA SIMULANDO UNA
!                  FALLA O UN COMPORTAMIENTO EN ESTADO
ESTACIONARIO DE LA
!           MÁQUINA.
!    EPOCAS= ES LA VARIABLE QUE INDICA AL PROGRAMA EL
NÚMERO DE
!                  VECES QUE SE RECORRERA EL NÚMERO DE
CONDICIONES INICIALES
!           CON LAS QUE SE ENTRENARÁ LA ESTRUCTURA
FALCON-ART.
!    N_DE_CONDICIONES= REPRESENTA EL NUMERO DE
CONDICIONES INICIALES
!                  QUE SE TOMARÁN EN CUENTA PARA EL
ENTRENAMIENTO DE LA
!           ESTRUCTURA FALCON-ART.
call
solicitud(tinic_falla,tdura,respuesta,epocas,N_de_condicione
s)
!LA SIGUIENTE DECLARACIÓN ASIGNA UNA LOCALIDAD DE MEMORIA A
LOS
!ARREGLOS QUE SE UTILIZAN EN EL MODELO DE MÁQUINA SÍNCRONA.
allocate(a(na),a_aux(na),bet(nm))
!LA VARIABLE INICIO INDICA A LA SUBRUTINA QUE SE ESTA
REALIZANDO LA
!ITERACIÓN No. UNO, INMEDIATAMENTE DESPUES DE ESTA ITERACION
!ESTA VARIABLE SE HACE CERO.
inicio = 1
!LA VARIABLE IENTRE=1 INDICA QUE SE ESTA LLEVANDO A CABO EL
ENTRENAMIENTO.
!LA VARIABLE IENTRE=0 INDICA QUE NO SE ESTA LLEVANDO A CABO
!EL ENTRENAMIENTO.
ientre = 1
!LA VARIABLE ITERMINA=0 INDICA QUE SE ESTA INICIANDO
!EL PROCESO DE SIMULACIÓN.
itermina = 0
!LA VARIABLE ICONTADOR PERMITE MUESTREAR LOS PATRONES
!CADA DETERMINADA CANTIDAD DE PUNTOS.
icontador = 1
!LA VARIABLE vbus QUE INDICA EL VALOR DEL VOLTAJE EN EL BUS,
!LA AUSENCIA DEL VALOR DE ESTA VARIABLE CAUSA
!PROBLEMAS EN EL COMPORTAMIENTO DEL ANGULO DE LA MÁQUINA.
vbus = 1.0
!SE PIDE LA CANTIDAD DE EPOCAS O VECES QUE SE RECORERÁ LA
!CANTIDAD TOTAL DE CONDICIONES INICIALES.
print*, ' CUAL ES SU NUMERO DE EPOCAS GLOBALES ? '
READ*, epocas_global

! SE NECESITA DEFINIR SI SE DESEA ENTRENAR A LA RED O NO
print*, ' DESEA REALIZAR ENTRENAMIENTO CON LA RED ?  si [1]
no[2] '
read*, ientrenar
!EL SIGUIENTE  PROCESO SE INICIA EN CASO DE
!REALIZARSE EL ENTRENAMIENTO DE LA RED.
!*************************************************************
************
```

C.2. 1 Bloque de entrenamiento.

```
!*************************************************************
************
!************ B L O Q U E   D E   E N T R E N A M I E N T
O  *******
!*************************************************************
************
if ( ientrenar == 1) then
!SE IMPRIME EL MOMENTO INICIAL DEL ENTRENAMIENTO
!ESTO SE HACE CON EL OBJETIVO DE CONOCER
!EL LAPSO DE TIEMPO QUE DURA EL ENTRENAMIENTO.
!ES LA VARIABLE TIEMPO_EFECTIVO LA CUAL ALMACENA
!ESTE DATO.
TIEMPO_EFECTIVO = CLOCK ()
PRINT *, '     TIEMPO INICIAL ',TIEMPO_EFECTIVO
!SE INICIA EL PROCESO DE ENTRENAMIENTO.
!EL SIGUIENTE DO SE LLEVARÁ A CABO SEGUN EL NÚMERO DE
!ÉPOCAS QUE INDIQUE LA VARIABLE ÉPOCAS_GLOBAL.
```

```
108. DO KK = 1, epocas_global
109. !EL ARCHIVO MAXI_MIN ALMACENARÁ LOS VALORES MÁXIMOS Y
110. !MINIMOS DE LAS VARIABLES QUE SON ENTRADAS A LA
111. !ESTRUCTURA FALCON-ART.A ESTE ARCHIVO SE PUEDEN ENVIAR
112. !VALORES MAX Y MIN TANTO GLOBALES COMO LOCALES.
113. !EL ARCHIVO RESULTADOS2 FUE UTILIZADO PARA UNA FORMA
114. !DIFERENTE DE ENTRENAMIENTO Y ES SÓLO UN ARCHIVO
     AUXILIAR.
115. open(60,file= ' maxi_mini.dat ')
116. open(150,file= ' resultados2.dat ' )
117. !EL DO SIGUIENTE SE LLEVA A CABO EL NUMERO DE
     CONDICIONES
118. !INCIALES CON LAS QUE SE DESEA ENTRENAR LA RED FALCON-
     ART.
119. !DICHO DO SE REALIZA EL NÚMERO DE ÉPOCAS GLOBALES QUE
120. !SE HAYAN ESCOGIDO.
121. do k = 1,N_de_condiciones
122. !*****************************************************
     ***************
123. !****************          subrutina     datos_iniciales
     ************************
124. !*****************************************************
     ***************
125. !SE LLAMA LA SUBRUTINA DATOS_INICIALES
126. !ESTA SUBRUTINA ARROJA LAS VARIABLES SIGUIENTES:
127. !    TINIC= QUE REPRESENTA EL MOMENTO INICIAL DE
     SIMULACION.
128. !  TFIN = REPRESENTA EL MOMENTO FINAL DE LA SIMULACIÓN.
129. !  H = REPRESENTA EL INCREMENTO DELTA
130. !  N = REPRESENTA EL NUMERO DE PASOS DE SIMULACION.
131. !*****************************************************
     ***************
132. call datos_iniciales(tinic,tfin,h,n)
133. !*****************************************************
     ***************
134. !****************          subrutina  condicion_inicial
     ****************
135. !*****************************************************
     ***************

136. !A CONTINUACIÓN SE LLEVA A CABO LA OBTENCIÓN DE
          MEDIANTE LA
137. !SUBRUTINA CONDICIONES INICIALES PARA LA SIMULACIÓN
138. !DE LA MÁQUINA SÍNCRONA. SUS VARIABLES DE SALIDA SON:
139. !  VA = VOLTAJE EN TERMINALES DE LA MÁQUINA.
140. !  VBUS = VOLTAJE EN EL BUS Y QUE
141. !                PERMANECE CONSTANTE DURANTE TODA LA
     SIMULACIÓN, LA
142. !          AUSENCIA DEL VALOR DE ESTA VARIABLE DENTRO DE
     LA SUB
143. !          RUTINA, ES DEBIDO A QUE SE MENCIONA AL INICIO
     DE ESTE
144. !          PROGRAMA.
145. !  THET = ÁNGULO DE LA MÁQUINA.
146. !  TE = PAR ELÉCTRICO DE LA MÁQUINA.
147. !  ID = COMPONENTE DE CORRIENTE EN EJE DIRECTO.
148. !  IQ = COMPONENTE DE CORRIENTE EN EJE EN CUADRATURA.
149. !  VREF = VOLTAJE DE REFERENCIA.
150. !  P = POTENCIA REAL GENERADA.
151. !  Q = POTENCIA REACTIVA GENERADA.
152. !  UG = PAR DE CARGA.
153. !  AI = REPRESENTA EL VECTOR QUE ALMACENA LOS VALORES
     DE LAS
154. !       VARIABLES DE ESTADO EN UN INSTANTE DETERMINADO.
155. call condicion_inicial(va,vbus,thet,te,id,iq,vref,p,q,ug,ai)
156. ! A CONTINUACIÓN DE UTILIZAN LAS VARIABLES AUXILIARES:
157. ! *_AUX QUE SE IGUALAN A SUS HOMÓNIMOS EN EL INSTANTE
158. ! T(0). ESTO CON LA FINALIDAD DE NO PERDER ESTA INFORMACIÓN
159. ! YA QUE LAS VARIABLES ORIGINALES POSIBLEMENTE CAMBIARAN
160. ! SU VALOR AL FINAL DE LA PRIMERA SIMULACIÓN DE UNA
161. ! DETERMINADA CONDICIÓN INICIAL, Y SE DESEA REALIZAR
162. ! UNA SEGUNDA SIMULACIÓN CON ESTA MISMA CONDICIÓN INICIAL
163. ! MAS ADELANTE EN ESTE SEGMENTO DEL PROGRAMA.
164. va_aux = va; vbus_aux = vbus; thet_aux = thet; te_aux = te;
     id_aux = id
165. iq_aux = iq; p_aux = p; q_aux = q; ug_aux = ug; ai_aux = ai
166. ! AMBOS VALORES, MAXIMO Y MINIMO LOCALE, SE IGUALAN AL VALOR
167. ! QUE REPRESENTAN DURANTE LA SIMULACIÓN.
168. va_aux1_max = va                    !  renglones  a
     permanecer
169. va_aux1_min = va                    !  renglones  a
     permanecer
170. va_aux2_max = vref - va             ! renglones a permanecer
171. va_aux2_min = vref - va             ! renglones a permanecer
172. !SE INICIAL LA SIMULACIÓN DEL MODELO
173.       do i = 1,n
174. ! EN ESTE SEGMENTO DE PROGRAMA SI EL MODELO SE CORRERÁ
175. ! CON LA FINALIDAD DE SIMULAR CAMBIOS DE CARGA O
176. ! ÚNICAMENTE UNA FALLA EN TERMINALES DE LA MÁQUINA.
177. ! ESTA INFORMACIÓN SE ALMACENA EN LA VARIABLE RESPUESTA,Y
178. ! SU VALOR SE OBTIENE EN LA SUBRUTINA SOLICITUD.
179. !*****************************************************
     ************
180. !*****************                    subrutina    falla
     ******************************
181. !*****************************************************
     ************
182. ! LA SUBRUTINA FALLA CONVIERTE EL VOLTAJE
183. ! EN EL BUS CON VALOR MUY APROXIMADO A CERO, EN
184. ! EN MOMENTO INDICADO PARA LLEVAR A CABO LA FALLA.
185. ! SUS VALORES DE ENTRADA SON:
186. !    TINIC =  TIEMPO INICIAL DE SIMULACIÓN
187. !   TINIC_FALLA= TIEMPO INICIA DE FALLA
188. !   TDURA = TIEMPO DE DURACIÓN DE LA FALLA.
189. ! SU VALOR DE SALIDA:
190. !   VBUS = REPRESENTA EL VOLTAJE EN EL BUS.
191. !*****************************************************
     ************
192. !*****************                    subrutina    cambio
     ***************************
193. !*****************************************************
     ************
194. ! SI EL VALOR DE LA VARIABLE RESPUESTA ES 2,
195. ! ENTONCES SOLO LA SUBRUTINA CAMBIO
196. ! REALIZARÁ LOS CAMBIOS DE CARGA
197. ! SIN QUE SE REALICE NINGÚN TIPO DE FALLA.
198. ! LAS VARIABLES DE ENTRADA DE ESTA SUBRUTINA SON:
199. ! TINIC = TIEMPO INICIAL DE SIMULACIÓN
200. ! VARIABLE SE SALIDA:
201. ! UG = PAR DE CARGA.
202.       if( respuesta == 1)then
203.         call falla(tinic,tinic_falla,tdura,vbus)
204.       else
205.         call cambio(tinic,ug)
206.       endif
207. ! SE REALIZA EL CÁLCULO DE LOS VALORES
```

```
208. ! DE LAS VARIABLES DE ESTADO EN EL TIEMPO t+1
209.       call kutta(ai,id,iq,te,va,ug,vref,h)
210.       call dinamica(ai,id,iq,te,va,p,q,vbus)
211. ! SE ACTUALIZAN LOS VALORES MAXIMOS Y MINIMOS LOCALES
212. ! CON LA FINALIDAD DE QUE AL FINAL DE LA SIMULACIÓN
213. ! SE OBTENGAN ESTOS VALORES Y UTILIZARLOS
214. ! EN LA SIMULACIÓN SIGUIENTE.
215. !ESTOS VALORES MAX Y MIN LOCALES,SE OBTIENE UNICAMENTE
216. !PARA EL VOLTAJE TERMINAL Y EL ERROR DEL VOLTAJE
     TERMINAL
217. va_aux1_max = max(va_aux1_max,va)
     renglones a permanecer
218. va_aux1_min = min(va_aux1_min,va)
     renglones a permanecer
219. va_aux2_max = max(va_aux2_max,vref - va)
     renglones a permanecer
220. va_aux2_min = min(va_aux2_min,vref - va)
     renglones a permanecer
221. ! SE INCREMENTA EL TIEMPO.
222. tinic = tinic + h
223. enddo
224. !UNA VEZ FINALIZADA LA PRIMERA SIMULACIÓN DONDE SE
     OBTIENE
225. !EL MAX Y MIN LOCAL, SE PROCEDE ENTRENAR LA ESTRUCTURA
     FALCON
226. !ART CON LOS PATRONES DE LA MISMA CONDICIÓN INICIAL
227. !UTILIZADA ANTERIORMENTE.
228. !LA VARIABLE GLOBAL FUE UTILIZADA CUANDO SE BOSQUEJO
229. !UN ENTRENAMIENTO EN BASE AL ERROR, CON LA FINALIDAD
230. !DE LLEVAR A CABO COMPARACIONES DE EXACTITUD.
231. global = 1.0
232. do while ( global >= 0.1 )
233. global = 0.09
234. !*****************************************************
     ******************
235. !*****************************************************
     ******************
236. !*****************************************************
     ******************
237. ! SE REALIZA UN PROCEDIMIENTO SIMILAR AL REALIZADO EN
238. ! EL SEGMENTO ANTERIOR
239. call datos_iniciales(tinic,tfin,h,n)
240. ! SE RECUPERAN VALORES DE LAS VARIABLES ORIGINALES
241. ! MENCIONADAS ANTERIORMENTE, PARA LLEVAR A CABO
242. ! UNA SIMULACIÓN ADICIONAL CON LA MISMA CONDICIÓN
     INICIAL
243. va = va_aux; vbus = vbus_aux; thet = thet_aux; te =
     te_aux; id = id_aux
244. iq = iq_aux; p = p_aux; q = q_aux; ug = ug_aux; ai =
     ai_aux
245. do i = 1,n
246. if( respuesta == 1)then
247. call falla(tinic,tinic_falla,tdura,vbus)
248. else
249. call cambio(tinic,ug)
250. endif
251. !*****************************************************
     ****************
252. !************     SE OBTIENEN LOS VALORES DE ENTRADA AL
     *****************
253. !************     ESPACIO DE ENTRADA
     *****************
254. ! SE OBTIENE EN VECTOR DE ENTRADA A LA ESTRUCTURA
255. ! FALCON-ART. ESTE VECTOR CONTIENE EN SUS LOCALIDADES
256. ! ENTRADA(1),ENTRADA(2),...,ENTRADA(ENT)
257. ! (DONDE ENT = No. DE ENTRADAS AL ESPACIO DE ENTRADA, ESTA
258. ! VARIABLE SE ENCUENTRA EN EL MODULE PARAMETROS_IMPORTANTES)
259. ! LOS VALORES QUE SERAN CLASIFICADOS EN EL ESPACIO DE
260. ! ENTRADA.
261. !*********************************************!
262. !*** ESTOS VALORES DE ENTRADA SE OBTIENEN ****!
263. !****     EN EL INSTANTE DE TIEMPO t     ********!
264. !*********************************************!
265. !VOLTAJE TERMINAL
266. entrada(1) = (va - va_aux1_min)/(va_aux1_max  - va_aux1_min)
267. !ERROR
268. entrada(2)     =     ((vref-va)-va_aux2_min)/(va_aux2_max     -
     va_aux2_min)
269. !VOLTAJE DE EXCITACIÓN
270. entrada(3) = (ai(6)-VMIN)/(VMAX-VMIN)
271. !REFERENCIA tomada en t0
272. if( i == 1 ) entrada(4) = entrada(3)
273. !SE CALCULAN LOS VALORES DE LAS VARIABLES DE ESTADO
274. !EN t+1
275. call kutta(ai,id,iq,te,va,ug,vref,h)
276. ! LAS COMPONENTES DE ESTE VECTOR:
277. ! ENTRADA(ENT+1),ENTRADA(ENT+2),...,ENTRADA(ENT+ENT_B)
278. !(DONDE ENT_B = No. DE ENTRADAS AL ESPACIO DE SALIDA, ESTA
279. ! VARIABLE SE ENCUENTRA EN EL MODULE PARAMETROS_IMPORTANTES)
280. ! SON LOS VALORES QUE SERAN CLASIFICADOS EN EL ESPACIO
281. ! DE SALIDA.
282. !*********************************************!
283. !*** ESTOS VALORES DE ENTRADA SE OBTIENEN ****!
284. !****     EN EL INSTANTE DE TIEMPO t+1  ********!
285. !*********************************************!
286. !VOLTAJE DE EXCITACIÓN
287. entrada(ent+1) = ( ai(6)-VMIN)/(VMAX-VMIN)
288. !REFERENCIA  tomada en t0
289. if( i == 1 ) entrada(ent+2) = entrada(ent+1)
290. !*****************************************************
     **********
291. !*****************************************************
     **********
292. !*****************************************************
     **********
293. call
     control_falcon_art(entrada,inicio,ientre,itermina,global,N,N
     _de_condiciones)
294. call dinamica(ai,id,iq,te,va,p,q,vbus)
295. tinic = tinic + h
296. enddo
297. enddo
298. if ( k == 1) then
299. va_aux1_MaxGral = va_aux1_max      ! renglones a permanecer
300. va_aux1_MinGral = va_aux1_min   ! renglones a permanecer
301. va_aux2_MaxGral = va_aux2_max      ! renglones a permanecer
302. va_aux2_MinGral = va_aux2_min      ! renglones a permanecer
303. endif
304. !SE VAN OBTENIENDO LOS VALORES MAXIMOS Y MINIMOS
305. !GENERALES PARA CUANDO SE CULMINEN EL ENTRENAMIENTO,
306. !PODER REALIZAR UNA SIMULACIÓN SIN TENER QUE PASAR
307. !POR EL ENTRENAMIENTO NUEVAMENTE.
308. !ESTOS VALORES MAX Y MIN GENERALES SE OBTIENE UNICAMENTE
     PARA
309. !EL VOLTAJE TERMINAL Y EL ERROR DEL VOLTAJE TERMINAL, ESTO
310. !SE DEBE A QUE, ES DE ESTAS VARIABLES DE LAS QUE SE
```

```
!DESCONOCE LOS VALORES MAX Y MIN QUE PUEDEN ALCANZAR
!DURANTE LA SIMULACIÓN. LO QUE NO SUCEDE CON LA REFERENCIA
!NI CON EL VOLTAJE DE EXCITACIÓN DE LOS CUALES SE CONOCE
!SUS LIMITES CON ANTERIORIDAD. ESTE CRITERIO TAMBIEN SE
!APLICA A LOS VALORES MAX Y MIN LOCALES.
va_aux1_MaxGral = max(va_aux1_MaxGral,va_aux1 max
      !renglones a permanecer
va_aux1_MinGral = min(va_aux1_MinGral,va_aux1 min
renglones a permanecer
va_aux2_MaxGral = max(va_aux2_MaxGral,va_aux2 max
renglones a permanecer
va_aux2_MinGral = min(va_aux2_MinGral,va_aux2 min
renglones a permanecer
print*,k
enddo
close(150)
!UNA VEZ REALIZADAS TODAS LA EPOCAS GLOBALES
!SE IMPRIME EL TIEMPO FINAL
!CON EL OBJETIVO DE CONOCER EL TIEMPO DE ENTRENAMIENTO.

if ( KK == epocas_global ) then
TIEMPO_EFECTIVO = CLOCK ()
PRINT *, '     TIEMPO FINAL ',TIEMPO_EFECTIV
!EN ESTE BLOQUE DE INSTRUCCIONES SE INDICA QUE SE HA
TERMINADO
!EL ENTRENAMIENTO ( COMO LO INDICA ITERMINA=1).
MEDIANTE ESTO,
!SE LOGRA LIBERAR LAS LOCALIDADES DE MEMORIA DINAMICA
!QUE SE HA UTLIZADO EN EL ENTRENAMIENTO.
if(ientrenar == 1) then
itermina = 1
call
control_falcon_art(entrada,inicio,centro,itermina,globa
l,N,N_de_condiones)
endif
endif
print*,KK
enddo
!SE ENVÍA AL ARCHIVO max_min LOS VALORES DE LOS
!VALORES MAXIMOS Y MINIMOS.
write(60,*)va_aux1_MaxGral          !valores      maximos
generales
write(60,*)va_aux1_MinGral  !valores minimos generales
write(60,*)va_aux2_MaxGral          !valores      maximos
generales
write(60,*)va_aux2_MinGral          !valores      minimos
generales
!SE CIERRA EL ARCHIVO max_min
CLOSE(60)
ENDIF
!EN ESTE BLOQUE DE INSTRUCCIONES, SI SE HA ELEGIDO
!NO REALIZAR EL ENTRENAMIENTO, ES DECIR, SE DESEA
APRECIAR
!LOS RESULTADOS DEL ENTRENAMIENTO, SE LEEN LOS VALORES
!MAXIMOS Y MINIMOS GENERALES DEL ARCHIVO max_min
IF ( IENTRENAR == 2) THEN
open(60,file= ' maxi_mini.dat ')
READ(60,*)va_aux1_MaxGral           !valores      maximos
generales
READ(60,*)va_aux1_MinGral           !valores      minimos
generales
READ(60,*)va_aux2_MaxGral           !valores maximos generales
READ(60,*)va_aux2_MinGral           !valores minimos generales
CLOSE(60)
ENDIF
```

C.2. 2 Bloque de prueba

```
!*************************************************************
***********
!*************************************************************
***********
!************          B L O Q U E     D E     P R U E B A
*********
!*************************************************************
***********
!*************************************************************
***********
IREPETICION = 1
!ESTE CICLO SE REALIZA DEPENDIENDO SI SE DESEA VER EL
COMPORTAMIENTO
!DEL SISTEMA CON OTRAS CONDICIONES INICIALES LAS CUALES NO
FUERON
!INCLUIDAS EN EL ENTRENAMIENTO.
DO WHILE ( IREPETICION == 1)

!SE INDICA NUEVAMENTE QUE SE INICIALIZA LA SIMULACIÓN
!MEDIANTE INICIO = 1 E ITERMINA = 0
inicio = 1
itermina = 0
!EN ESTE BLOQUE SE INDICA AL PROGRAMA SI SE DESEA
!REALIZAR UN SIMULACION DE FALLA, O UNA SIMULACIÓN
!CON CAMBIOS DE CARGA.
if(respuesta==2)then
ienviar = 2
else
ienviar = 1
endif
!DEPENDIENDO DEL RESULTADO DEL BLOQUE ANTERIOR
!EL SIGUIENTE CICLO RECORRERA EL NÚMERO DE CONDICIONES
!INICIALES A OBSERVAR, YA SEA SOLO EN CONDICIONES DE FALLA
!((IENVIAR=1) O EN CONDICIONES DE CAMBIO DE CARGA (IENVIAR=2)
do kk = 1,ienviar
!SE ABRE EL ARCEIVO SIGUIENTE PARA
!ALMACENAR LOS DATOS DE LA VARIABLES A OBSERVAR
open(150,file = ' resultados2.dat ') !cond_prueba.dat
!EL SIGUIENTE CICLO SE LLEVA A CABO EL No. DE
!CONDICIONES INICIALES QUE SE DESEAN OBSERVAR
do kk = 1,N_de_condiciones
!LAS VARIABLES SIGUIENTES SE GENERAN PARA OBTENER
!LOS INDICES DE COMPORTAMIENTO DEL VOLTAJE TERMINAL
!DEL ERROR Y DEL ANGULO DE LA MAQUINA.
suma1 = 0.0
suma2 = 0.0
suma3 = 0.0
suma4 = 0.0
!                        archivo = char(64+kk)
!                        OPEN(200,FILE=
'C:\AXUM\DATA\DAT'//archivo//'.dat' )
OPEN(200,FILE= 'C:\AXUM\DATA\DATA.DAT' )
!SE LLAMA LA SUBRUTINA DATOS_INICIALES
!REALIZANDO LA MISMA FUNCION QUE EN LOS SEGMENTOS
!ANTERIORES.
call datos_iniciales(tinic,tfin,h,n)
```

```
!EN ESTE BLOQUE SE REALIZA LO SIGUIENTE:
! SI LM = 1, SE LLEVA A CABO UNA SIMULACIÓN DEL
! SISTEMA EN CONDICIONES DE FALLA
! SI LM = 2, SE LLEVA A CABO UNA SIMULACIÓN CON VARIACIONES
! DE CARGA, Y SE OBTIENE TAMBIEN, EL NUEVO VALOR DE
! PASOS DE INTEGRACIÓN.
IF( LM == 2 ) THEN
TFIN = 400.0
RESPUESTA = 2
N = TFIN / H
else
respuesta = 1
ENDIF
!SE LLAMA LA SUBRUTINA CONDICION_INICIAL QUE CUMPLE LA
!MISMA FUNCIÓN QUE EN LOS SEGMENTOS ANTERIORES.
call condicion_inicial(va,vbus,thet,te,id,iq,vref,p,q,uq,ai)
!ESTE CONJUNTO DE VARIABLES SE Bª, SE UTILIZARON PARA
!SIMPLEMENTE DIFERENCIAR LAS VARIABLES A LAS QUE SE IGUALAN
!DE LA ETAPA ANTERIOR.
bai = ai;bid = id;biq=iq;bte=te;bva=va;[illegible]
!LAS INSTRUCCIONES SIGUIENTES PERMITEN CONFIGURAR
!LAS PANTALLAS DE GRAFICACIÓN DE ACUERDO A LAS NECESIDADES
!DEL PROGRAMADOR.
open(1,file = 'user',title = 'COMPORTAMIENTO DEL VOLTAJE DE CAMPO' ,iofocus = .true.)
status = setwindow(invert, -.5,7.0, ffin, -1.0)
open(2,file = 'user',title = 'COMPORTAMIENTO DEL VOLTAJE TERMINAL' ,iofocus = .true.)
status = setwindow(invert, -.5,1.5, ffin, -1.0
!SE INICIA EL CICLO DE SIMULACIÓN EN DONDE YA SE SUSTITUYE
!LA SEÑAL DEL REGULADOR ST1 POR LA SEÑAL GENERADA POR LA
!ESTRUCTURA FALCON-ART.

do i = 1,n
if( respuesta == 1)then
call falla(tinic,tinic_falla,tdura,vbus)
else
call cambio(tinic,uq)
endif
!SE OBTIENEN LAS ENTRADAS AL SISTEMA FALCON-ART
!EN UN INSTANTE t0, YA QUE CON ESTAS ENTRADAS FUE ENTRENADA
!LA RED. DICHO VECTOR DE ENTRADA SE DESCRIBIO ANTERIORMENTE,
entrada(1) = (va - va_aux1_MinGral)/(va_aux1_MaxGral - va_aux1_MinGral)
entrada(2) = (vref-va)-va_aux2_MinGral)/(va_aux2_MaxGral - va_aux1_MinGral)
entrada(3) = (ai(6)-VMIN)/(VMAX-VMIN)
if( i == 1 ) entrada(4) = entrada(3)
call kutta(bai,bid,biq,bte,bva,buq,vref,h)
call dinamica(bai,bid,biq,bte,bva,bp,bq,vbus)
call kutta(ai,id,iq,te,va,uq,vref,h)
!*****************************************************************
!*****************************************************************
!*****************************************************************
!*****************************************************************
!**************                    SUBRUTINA  REGULADOR_FALCON **************
!*****************************************************************
!*****************************************************************
!SE UTILIZA LA SUBRUTINA REGULADOR_FALCON_ART
!QUE CONTIENE UNICAMENTE LA PROPAGACIÓN DE SEÑALES
!HACIA ADELANTE.
!SUS VARIABLES DE ENTRADA SON:
!  ENTRADA = VECTOR DEL ESPACIO DE ENTRADA
!  INICIO = VARIABLE QUE PERMITE ASIGNAR
!           LOCALIDADES DE MEMORIA DINAMICA.
!  ITERMINA = PERMITE ELIMINAR TODA LA MEMORIA DINAMICA
!          UTILIZADA DURANTE LA SIMULACION
!SUS VARIABLES DE SALIDA ES:
!  NETA = VECTOR QUE CONTIENE EL VALOR GENERADO
!          POR LA RED FALCON-ART DE ACUERDO A VECTOR
!          ENTRADA CORRESPONDIENTE
call REGULADOR_FALCON(ENTRADA,INICIO,NETA,ITERMINA)
!*****************************************************************
!*****************************************************************
!*****************************************************************
!*****************************************************************
!EL SIGUIENTE BLOQUE DE INSTRUCCIONES PERMITE
!LIMITAR LA SEÑAL ENVIADA POR LA ESTRUCTURA FALCON_ART
if( neta(1) >= 1.0e-06 .AND. neta(1) <= 1.0) then
neta(1) = neta(1)
else if( neta(1) > 1.0 ) then
neta(1) = 1.0
else if( neta(1) < 1.0e-06 ) then
neta(1) = 0.0
endif
!EN ESTA INSTRUCCION SE LLEVA A CABO LA IGUALACIÓN
!DE LA SEÑAL DEL REGULADOR ST1 CON LA SEÑAL DE LA
!ESTRUCTURA FALCON-ART AMPLIFICADA.
ai(6) = neta(1) * (VMAX-VMIN) ) + VMIN
!             neta(1)=( neta(1) * (VMAX-VMIN) ) + VMIN
!             neta(2)=( neta(2) * (VMAX-VMIN) ) + VMIN
call dinamica(ai,id,iq,te,va,p,q,vbus)
!*****************************************************************
!*****************************************************************
!*****************************************************************
!*****************************************************************
!          SE LLEVA A CABO LA GRAFICACIÓN DE LAS VARIABLES
!             DE LAS CUALES SE DESEA OBTENER INFORMACION
!*****************************************************************
!*****************************************************************
!*****************************************************************
!*****************************************************************
if( i == 1 ) then
help = va
help1 = ai(1)
endif
IRESULT = SETACTIVEQQ(1)
oldcolor = SETCOLORRGB(#0000FF)   ! red
dummy = setpixel_w(tinic,ai(6))
oldcolor = SETCOLORRGB(#00FF00)   ! VERDE
dummy = setpixel_w(tinic,neta(1))
!                                 oldcolor =
SETCOLORRGB(#FF0000)
!                                 dummy =
setpixel_w(tinic,neta(2))
```

```
IRESULT = SETACTIVEQQ(2)
oldcolor = SETCOLORRGB(#FFFFFF)
dummy = setpixel_w(tinic,va)
dummy = setpixel_w(tinic,vref-va)
suma1 = suma1 + (va-help)**2
suma2 = suma2 + (bva-help)**2
suma3 = suma3 + (ai(1)-help1)**2
suma4 = suma4 + (bai(1)-help1)**2
write(200,200)tinic,va,bva,ai(6),bai(6),p,bp,q,bq,ai(1),bai(1), suma1, suma2, suma3, suma4
200 format(15f20.15)
tinic = tinic + h
enddo
CLOSE(200)
PAUSE
IRESULT = SETACTIVEQQ(1)
CALL CLEARSCREEN($GVIEWPORT)
IRESULT = SETACTIVEQQ(2)
CALL CLEARSCREEN($GVIEWPORT)
enddo
CLOSE(150)
enddo
!    CLOSE(150)
!*****************************************************************
!*****************************************************************
!*****************************************************************
! SE INFORMA AL PROGRAMA MEDIANTE ITERMINA = 1
! QUE SE DESEA LIBERAR LA MEMORIA DINAMICA UTILIZADA
! EN LA SUBRUTINA REGULADOR_FALCON
itermina = 1
call REGULADOR_FALCON(IENT,INICIO,NETA,ITERMINA)
!MEDIANTE ESTA PREGUNTA, SE PERMITE AL
!PROGRAMA REALIZAR LA SIMULACIÓN DE OTRAS CONDICIONES INICIALES.
!QUE NO FUERON INCLUIDAS EN EL ENTRENAMIENTO.
PRINT*, ' DESEA REPETIR LA PRUEBA NUEVAMENTE SI[1]  NO[2] ? '
READ(*,*) IREPETICION
ENDDO
! SE LIBERA LA MEMORIA DINAMICA OCUPADA POR
! AI Y AI_AUX
deallocate(ai,ai_aux)
end program
```

C. 3 SUBRUTINA CONTROL_FALCON_ART

```
!*****************************************************************
!*****************************************************************
!*****************************************************************
!              subrutina control_falcon_art
!Sus variables de entrada son:
!         Entrada = vector que contiene las entradas tanto del espacio
!                   de entrada como de salida a la estructura FALCON-ART.
!         Inicio = variable que indica a la subrutina que es el momento de
!                  realizar el dimensionamiento de las variables que asi lo
!                  necesiten, ya que es el inicio del entrenamiento.
!         itrain = variable que indica que es la etapa de entrenamiento.
!         final  = variable que indica a la subrutina que se ha terminado
!                  el entrenamiento y se desea desalojar memoria dinámica.
!       global,nnn,ikj = son variables que se ocuparon en un momento dado
!                  para monitorear el error.
!*****************************************************************
!*****************************************************************
!*****************************************************************
SUBROUTINE CONTROL_FALCON_ART(entrada,inicio,itrain,final,global,NNN,IJK)
!Se indica al compilador cuáles son los módulos que son necesarios dentro
!de la subrutina: es decir , que contienen datos imprecindibles para el
!funcionamiento de la subrutina.
USE PARAMETROS_FUNDAMENTALES
USE DECLARACIONES
USE MSFLIB
!Se declaran las caracteristicas de los argumentos de la subrutina.
!La caracteristica IN indica que es una variable que únicamente ingresa
!a la subrutina. La caracteristica INOUT indica que es una variable
!cuyo valor original será cambiado dentro de la subrutina y será regresado
!al programa principal.
real*8,intent(in)     :: entrada(ent+ent_b)
integer,intent(inout) :: inicio
integer,intent(in)        :: final,itrain
real*8,intent(out)    :: global
!Se declaran las variables siguientes:
!    NA = representa el número de nodos en la capa del espacio de entrada
!    es decir FA.
! NB = representa el número de nodos en la capa del espacio de salida
!    es decir FB.
! XAB = es el vector resultante de la operación 'AND' entre el vector
!    de activación YB y el vector de activación FAB.
! YB = representa el vector de activación de la capa FB. Este vector
!    contendrá un UNO en la localidad la cual corresponda al nodo activado
!    en una determinada búsqueda. Las demás entradas de este vector serán
```

```
45. !     CERO. Este vector es una estructura de dimensión variable.
46. ! FAB = representa un arreglo de dos dimensiones que contiene la información
47. !     que indica que nodos de la capa FA predicen a sus respectivos nodos
48. !       de la capa FB, dicha predicción concuerda de acuerdo a los
49. !     métodos de búsqueda y asignación de la estrctutra FALCON_ART.
50. ! FFAB = es un arreglo de dos dimensiones que es utilizado cuando el
51. !     arreglo FAB debe cambiar de número de elementos en una dimensión
52. !     determinada.
53. ! FA = representa el conjunto de nodos que se ubican en la capa FA. Es
54. !     un arreglo de una dimensión.
55. ! FFA = es una estructura auxiliar que permite almacenar la información
56. !         contenida en la estructura FA (mencionada anteriormente), mientras
57. !     esta (FA) es aumentada en el número de nodos.
58. ! FB = representa el conjunto de nodos que se ubican en la capa FB. Es
59. !     un arreglo de una dimensión.
60. ! FFB = es una estructura auxiliar que permite almacenar la información
61. !         contenida en la estructura FB (mencionada anteriormente), mientras
62. !     esta (FB) es aumentada en el número de nodos.
63. ! VAR = Es un arreglo de una dimensión que permite almacenar los valores
64. !     de las funciones de activación de la capa FA. De donde, el nodo que
65. !     contenga la función de activación que contenga el valor más alto será
66. !     el nodo ganador.
67. ! VAR_B = Es un arreglo de una dimensión que permite almacenar los valores
68. !     de las funciones de activación de la capa FB. De donde, el nodo que
69. !     contenga la función de activación que contenga el valor más alto será
70. !     el nodo ganador.
71. ! IENT = Es un arreglo de una dimensión y almacena los valores, que
72. !         introduce el vector ENTRADA a la estrucutra FALCON-ART,
73. !     que corresponden al espacio de ENTRADA: es decir, ENTRADA(1),
74. !     ENTRADA(2),...,ENTRADA(ENT).
75. ! IENT_B = Es un arreglo de una dimensión y almacena los valores, que
76. !         introduce el vector ENTRADA a la estrucutra FALCON-ART,
77. !     que corresponden al espacio de SALIDA; es decir, ENTRADA(ENT+1),
78. !     ENTRADA(ENT+2),...,ENTRADA(ENT+ENT_B).
79. ! ERRORES = Es un arreglo de una dimensión y que permite almacenar los
80. !     errores generados por la red FALCON-ART durante el entrenamiento.
81. ! FNETA = Es un arreglo de una dimensión y que corresponde a la salida
82. !     proporcionada por la red FALCON-ART.
83. ! VALORMEDIO_POR_SALIDA = Es un arreglo de una dimensión y almacena
84. !         información relativa a la dedifuzificación de las señales
85. !     provenientes de capas inferiores.
86. INTEGER,SAVE                                                    ::
    NA,NB,IVAR
87. INTEGER,ALLOCATABLE,DIMENSION(:)                               :: XAR,YB
88. INTEGER,ALLOCATABLE,DIMENSION(:,:),SAVE                        :: FAB
89. INTEGER,ALLOCATABLE,DIMENSION(:,:)                            :: FFAB
90. TYPE(ELEMENTOS_FA),DIMENSION(:),ALLOCATABLE,SAVE              :: FA
91. TYPE(ELEMENTOS_FA),DIMENSION(:),ALLOCATABLE                   :: FFA
92. TYPE(ELEMENTOS_FB),DIMENSION(:),ALLOCATABLE,SAVE              :: FB
93. TYPE(ELEMENTOS_FB),DIMENSION(:),ALLOCATABLE                   :: FFB
94. REAL*8,DIMENSION(:),ALLOCATABLE,SAVE                          :: VAR
95. REAL*8,DIMENSION(:),ALLOCATABLE                               :: IENT
96. REAL*8,DIMENSION(:),ALLOCATABLE,SAVE                          :: VAR_B
97. REAL*8,DIMENSION(:),ALLOCATABLE                               :: IENT_B
98. REAL*8,DIMENSION(:),ALLOCATABLE                                ::
    ERRORES,FNETA
99. REAL*8,DIMENSION(:),ALLOCATABLE,SAVE                          ::
    VALORMEDIO_POR_SALIDA
100. !esta variable fue utilizada durante otras pruebas del programa.
101. TYPE  (WXYCOORD)  XY
102. !*************************************************************
     *************
103. !*************************************************************
     *************
104. !*************************************************************
     *************
105. !         SE DIMENSIONAN LAS ESTRUCTURAS NECESARIAS EN EL PROGRAMA
106. !       ESTA ACCIÓN SE EJECUTA UNA SOLA VEZ DURANTE TODA LA SIMULACION
107. !       DEL PROGRAMA MAQUINA_CON_REGULADOR.
108. !*************************************************************
     *************
109. !*************************************************************
     *************
110. !*************************************************************
     *************
111. if(inicio == 1) then
112. ! Se desactiva la señal de inicio INICIO y se envia al programa
113. ! principal.
114. inicio = 0
115. ! Inicialmante se supone que tendremos un solo nodo en cada una de las
116. ! capa FA y FB.
117. NA = 1
118. NB = 1
119. !    IVAR = 0
120. !    ERROR_GLOBAL_AUX = 0.0
121. !    GLOBAL = 0.0
122. ! Se asignan localidades de memoria a las variables que tienen la
123. ! propiedad ALLOCATE de acuerdo al número de entradas que
124. ! corresponda a su espacio ( ENTRADA o SALIDA, según sea el
```

```
! caso) respectivo o de acuerdo a la función que vayan a realizar.
ALLOCATE(IENT(2*ENT))
ALLOCATE(IENT_B(2*ENT_B))
ALLOCATE(VALORMEDIO_POR_SALIDA(ENT_B))
ALLOCATE(FAB(NA,NB))
ALLOCATE(ERRORES(ENT_B))
ALLOCATE(FNETA(ENT_B))
ALLOCATE(XAB(NB))
ALLOCATE(YB(NB))
ALLOCATE(FA(NA))
ALLOCATE(FB(NB))
ALLOCATE(VAR(NA))
ALLOCATE(VAR_B(NB))
! Se supone que el primer nodo contenido
! en la capa FA predice al primer nodo de la capa FB,
! por lo tanto indicamos la siguiente asignación:
FAB = 1
!Se asignan valores a XAB y YB
XAB = 1
YB = 0
! Se asignan valores iniciales a los componentes de la estructura
! FA y FB. El significado de cada componente se informa en el
! módulo DECLARACIONES2.
FA(NA)%TJ = 1.0
FA(NA)%RJ = 0.0
FA(NA)%WJ = 1.0
FA(NA)%WJ1 = 0.0
FA(NA)%CLASE = 1
FA(NA)%NETA = 0.0
FA(NA)%PRODUCTO = 0.0
FA(NA)%ERROR  = 0.0

FB(NB)%TJ = 1.0
FB(NB)%RJ = 0.0
FB(NB)%WJ = 1.0
FB(NB)%WJ1 = 0.0
FB(NB)%CLASE = 1
FB(NB)%ERROR = 0.0
FB(NB)%CENTRO = 0.0
FB(NB)%NETA = 0.0
!Se finaliza la secuencia de inicialización del sistema FALCON-ART
endif
!*****************************************************************
!*****************************************************************
!       SE INICIA EL ENTRENAMIENTO, ASIGNANDO LOS
!       VALORES DE ENTRADA AL SISTEMA EN LAS VARIABLES CORRESPONDIENTES.
!
!*****************************************************************
!*****************************************************************
ient(1:ent) = entrada(1:ent)
ient_b(1:ent_b) = entrada(ent+1:ent+ent_b)

if(itrain == 1)then
! Se obtienen los complementos de las entradas; es decir,
! Por ejemplo, si entrada = 0.7 su complemento = 0.3
! de manera tal que su suma siempres sea una constante.
IENT_B(ENT_B+1:) = 1. - IENT_B(:)
IENT(ENT+1:) = 1. - IENT(:)
!Mediante el ciclo siguiente se obtienen todas las variables
!contenidas en cada nodo de la capa FA; por ejemplo, TJ que es la
!función de activación de cada nodo de la capa FA, RJ que es el tamaño
!de la clase contenida en dicho nodo.
DO I = 1,NA
FA(I)%WJ1 = MIN(FA(I)%WJ,IENT)
RJJ          =          MAX(IENT,(1.-FA(I)%WJ1(ENT+1:)))-MIN(IENT,FA(I)%WJ1(:ENT))
FA(I)%TJ=SUM(FA(I)%WJ1)/(ALFA+SUM(FA(I)%WJ))
FA(1)%RJ= SUM(RJJ)
ENDDO
!Mediante el ciclo siguiente se obtienen todas las variables
!contenidas en cada nodo de la capa FB; por ejemplo, TJ que es la
!función de activación de cada nodo de la capa FB, RJ que es el tamaño
!de la clase contenida en dicho nodo.
DO I = 1,NB
FB(I)%WJ1 = MIN(FB(I)%WJ,IENT_B)
RJJ_B          =          MAX(IENT_B,(1.-FB(I)%WJ1(ENT_B+1:)))-MIN(IENT_B,FB(I)%WJ1(:ENT_B))
FB(I)%TJ=SUM(FB(I)%WJ1)/(ALFA+SUM(FB(I)%WJ))
FB(I)%RJ= SUM(RJJ_B)
ENDDO
!Se obtienen las funciones de activación de las capas FA y FB, en los
!arreglos VAR y VAR_B,respectivamente.
!se obtiene también el subindice del nodo ganador y que se almacena
!en IA para la capa FA y en IA_B para la capa FB.
VAR_B(:) = FB(:)%TJ
AR_B = MAXLOC(VAR_B)
IA_B = AR_B(1)

VAR(:) = FA(:)%TJ
AR = MAXLOC(VAR)
IA = AR(1)
!*****************************************************************
!*****************************************************************
! Una ver localizadas las unidades ganadoras en ambas capas, FA y FB, se
! evaluan los criterios de aprendizaje, primero en FB, y en seguida en la
! capa FA.
!*****************************************************************
!*****************************************************************
!Se inicia la búsqueda en la capa FB
IVAR2 = 1
MOUNT = 0
```

```
222. DO WHILE ( IVAR2 /= 0 )
223. ! Se evalúa si el nodo cumple con el criterio de igualación
224. ! (MATCHING) y con el tamaño de la clase, es decir, que esta
225. ! no exceda un tamaño predeterminado.
226. ! El tamaño de la clase lo determina la variable
227. ! RESTRICCIÓN_ y que se obtiene en el módulo DECLARACIONES.
228. IF ((SUM(FB(IA_B)%WJ1)/ENT_B)>= RO_B .AND. FB(IA_B)%RJ<=RESTRICCION_B) THEN
229. ! IF (FB(IA_B)%RJ<=RESTRICCION_B)
230. ! Si se cumple con los criterios anteriores se modifican los pesos
231. ! en el nodo ganador de la capa FB,
232. FB(IA_B)%WJ = FB(IA_B)%WJ1
233. ! Se envia una señal a la estructura FAB que se ha activado
234. ! un nodo en la capa FB
235. ISENAL_BFAB = 1
236. ! Se indica que se debe detener la búsqueda de nodos, ya que se
237. ! se ha encontrado el ganador.
238. IVAR2 = 0
239. ! El vector de activación de la capa FB se hace CERO,
240. YB = 0
241. ! la localidad del nodo ganador en FB se hace iguala a UNO.
242. YB(IA_B) = 1
243. !**********************************************************************
244. !**********************************************************************
245. ! En caso de no cumplirse ambos criterios anteriores se inicia la
246. ! búsqueda de un nuevo nodo ganador.
247. !**********************************************************************
248. !**********************************************************************
249. ELSE
250. !la función de activación TJ, del nodo ganador se hace cero
251. !para evitar que sea seleccionada nuevamente durante la búsqueda
252. VAR_B(IA_B) = 0.0
253. !se localiza el subindice del nuevo nodo ganador
254. AR_B = MAXLOC(VAR_B)
255. IA_B = AR_B(1)
256. !se aumenta un contador que indica si aún existen o no nodos
257. !por seleccionar en la capa FB.
258. MOUNT = MOUNT + 1
259. !en caso de no haberse hallado un nodo ganador durante la búsqueda
260. !se indica dentro del IF precedente lo siguiente:
261. IF ( MOUNT == NB )THEN
262. ! Se indica que la capa FB no se activo con ISENAL_BFAB = 0
263. ISENAL_BFAB = 0
264. ! el vector de activación YB se hace CERO.
265. YB = 0
266. ! se hace IA_B = 0
267. IA_B = 0
268. ! y se detiene la búsqueda con IVAR2 = 0

269. IVAR2 = 0
270. ENDIF
271. ENDIF
272. ENDDO
273. !**********************************************************************
274. !**********************************************************************
275. !**********************************************************************
276. !!!!!!!!!!!!!!!!!!!! Se inicia la búsqueda en la capa FA!!!!!!!!!!!!!!!!!!
277. !**********************************************************************
278. !**********************************************************************
279. !**********************************************************************
280. IVAR2 = 1
281. MOUNT = 0

282. DO WHILE ( IVAR2 /= 0 )
283. ! Se lleva a cabo la operación "AND" entre los
284. ! vectores de activación de la capa FB y la componente
285. ! de FAB que representa la predicción del nodo ganador en FA
286. ! y que su subindice esta memorizado en IA (nodo ganador)
287. !Si la capa FB se activó se lleva la operación "AND" entre los
288. !vectores mencionados; en caso contrario, el vector XAB
289. !se iguala a la componente FAB del nodo ganador en FA.
290. IF (ISENAL_BFAB == 1 ) THEN
291. XAB(1:NB) = MIN(YB(1:NB),FAB(IA,1:NB))
292. ELSE
293. XAB(1:NB-1) = FAB(IA,1:NB-1)
294. ENDIF
295. !Se evalúa lo siguiente:
296. !El nodo ganador en FA debe predecir el nodo ganador en la capa
297. !FB,
298. !El nodo ganador de FA debe cumplir con el criterio de
299. !igualdad (MATCHING), y
300. !su clase no debe exceder el tamaño determinado por
301. !variable RESTRICCION ( que se obtiene en el módulo
302. !DECLARACIONES)
303. IF ( SUM(XAB) >= RO_AB * SUM(YB) .AND. (SUM(FA(IA)%WJ1)/ENT)>= RO_A &
304. & .AND. FA(IA)%RJ<=RESTRICCION ) THEN
305. !En caso de cumplirse lo anterior se lleva a cabo el aprendizaje
306. !se indica que se activó la capa FA con ISENAL_AFAB = 1
307. !y se detiene la búsqueda.
308. ! IF (FA(IA)%RJ<=RESTRICCION)
309. FA(IA)%WJ = FA(IA)%WJ1
310. ISENAL_AFAB = 1
311. IVAR2 = 0
312. !**********************************************************************
```

```
!*************************************************************
*******************
!*************************************************************
*******************
!  De no cumplirse lo anterior se inicia la búsqueda de
un nuevo nodo
!  ganador en FA.
!*************************************************************
*******************
!*************************************************************
*******************
ELSE
!se llevan a cabo las mismas operaciones que en la capa
FB
VAR(IA) = 0.0
AR = MAXLOC(VAR)
IA = AR(1)
MOUNT = MOUNT + 1
IF ( MOUNT == NA )THEN
ISENAL_AFAB = 0
IVAR2 = 0
IA = 0
ENDIF
ENDIF
ENDDO
!*************************************************************
*******************
!*************************************************************
*******************
!       SE INICIA EL PROCESO DE MAPEO, EN DONDE SE
DETERMINA,
!       DE ACUERDO A LOS RESULTADOS ANTERIORES, SI SE
MODIFICAN
!    LAS ESTRUCTURAS FA O FB.
!*************************************************************
*******************
!*************************************************************
*******************
!*************************************************************
*******************
!  PRIMER CASO: AMBAS CAPAS, FA Y FB, NO SE ACTIVARON
!*************************************************************
*******************
! Es necesario entonces crear nodos nuevos en ambas
!    capas y se realiza lo siguiente:
IF ( ISENAL_AFAB  == 0 .AND. ISENAL_BFAB == 0 ) THEN
!Se utilizan las variables auxiliares mencionadas
!al inicio de este programa, primero se alojan en
!FFB el número de nodos existentes en la capa FB.
ALLOCATE(FFB(SIZE(FB)))
FFB(:) = FB(:)
!Se elimina el arreglo FB
DEALLOCATE(FB)
!Se incrementa el arreglo en una unidad
NB = NB + 1
!Se asigna  una nueva localidad de memoria para FB
!ya con un nuevo número de nodos.
ALLOCATE(FB(NB))
!Se recupera la información previa almacenada en FFB
FB(:NB-1) = FFB(:NB-1)
!Se inicializa el nuevo nodo recién creado, y se
almacena
!la entrada a la estructura FALCON-ART en este nodo, la
!cual no había sido clasificada.
FB(NB)%TJ = 1.
FB(NB)%RJ = 0.
!*************************************************************
*************
!!!!  se almacena la entrada en este nodo de nueva
creacion!!!!!!!
FB(NB)%WJ = IENT
!*************************************************************
*************
FB(NB)%WJ1 = 0.0
FB(NB)%CLASE = 0
FB(NB)%CENTRO = 0.0
FB(NB)%ERROR  = 0.0
FB(NB)%META = 0.0
!Se libera la memoria de FFB
DEALLOCATE(FFB)
!Se ajusta las estructura VAR_B ya que ahora habrá
!un nodo adicional en cual generará una nueva función
!de activación.
DEALLOCATE(VAR_B)
ALLOCATE(VAR_B(NB))
!Las operaciones realizadas anteriormente se repiten para
!la estructura FA, obviamente utilizando las variables
!correspondientes a esta capa.
ALLOCATE(FFA(SIZE(FA)))
FFA(:) = FA(:)
DEALLOCATE(FA)
NA = NA + 1
ALLOCATE(FA(NA))
FA(:NA-1) = FFA(:NA-1)
FA(NA)%TJ = 1.
FA(NA)%RJ = 0.
FA(NA)%WJ = IENT
FA(NA)%WJ1 = 0.0
FA(NA)%CLASE = 0
FA(NA)%META = 0.0
FA(NA)%PRODUCTO = 0.0
FA(NA)%ERROR = 0.0
DEALLOCATE(FFA)
DEALLOCATE(VAR)
ALLOCATE(VAR(NA))
!Ahora se lleva a cabo la modificación de la estructura
!FAB que contiene la información de las predicciones de FA
hacia
!FB.
!Inicialmente se ajusta la variable auxiliar de acuerdo
!al número de elementos en ambas dimensiones de FAB.
ALLOCATE(FFAB(SIZE(FAB,dim=1),SIZE(FAB,d[m=2)))
!Se almacena la información de FAB en FFAB
DO I = 1,NA-1
FFAB(I,:NB-1) = FAB(I,:NB-1)
ENDDO
!Se elimina temporalmente FAB
DEALLOCATE(FAB)
!Con los valores actualizados de NA y NB se ajusta el
!tamaño de FAB.
ALLOCATE(FAB(NA,NB))
!Se recupera la información de la variable auxiliar FFAB
DO I = 1,NA-1
FAB(I,:NB-1) = FFAB(I,:NB-1)
ENDDO
```

```
!Ahora todo la fila NA (de nueva creación) se hace cero
!igualmentee toda la columna, y se hace igual a
!UNO la entrada en la cual coincide NA y NB
FAB(NA,:) = 0
FAB(:,NB) = 0
FAB(NA,NB) = 1
!Se adjudica al nuevo nodo en FA  la clase que predecirá
!a partir de este momento, que será la clase NB
FA(NA)%CLASE = NB
!Se aumenta el numero de veces que ha sido predicho el
!nodo de recien creacion en FB (obviamente en este momento
! es uno), esta acción se lleva a cabo para verificaciones
!posteriores por el programador.
FB(NB)%CLASE = FB(NB)%CLASE + 1
! Se actualizan las variables que dependen del valor de NB
! como es XAB, YB
DEALLOCATE(XAB,YB)
ALLOCATE(XAB(NB),YB(NB))
!Se libera la memoria de la variable auxiliar FFAB.
DEALLOCATE(FFAB)

ENDIF

!*************************************************************
!  SEGUNDO CASO:  CAPA FA NO ACTIVADA, CAPA FB ACTIVADA
!*************************************************************
!En este caso se actualiza únicamente la capa FA y se sigue
!un procedimiento similar al anterior.
IF( ISENAL_AFAB  == 0 .AND. ISENAL_BFAB == 1) THEN
ALLOCATE(FFA(SIZE(FA)))
FFA(:) = FA(:)
DEALLOCATE(FA)
NA = NA + 1
ALLOCATE(FA(NA))
FA(:NA-1) = FFA(:NA-1)
FA(NA)%TJ = 1.
FA(NA)%RJ = 0.
!Se almacena el patrón no memorizado
!en el nuevo nodo de recién creación.
FA(NA)%WJ = IENT
FA(NA)%WJ1 = 0.0
FA(NA)%CLASE = 0
FA(NA)%NETA = 0.0
FA(NA)%PRODUCTO = 0.0
FA(NA)%ERROR = 0.0
DEALLOCATE(FFA)
DEALLOCATE(VAR)
ALLOCATE(VAR(NA))
!Se lleva a cabo el mapeo, y se ejecutan las mismas
!operaciones del caso anterior
ALLOCATE(FFAB(SIZE(FAB,DIM=1),SIZE(FAB,DIM=2)))

DO I = 1,NA-1
FFAB(I,1:NB) = FAB(I,1:NB)
ENDDO
DEALLOCATE(FAB)

ALLOCATE(FAB(NA,NB))

DO I = 1,NA-1
FAB(I,1:NB) = FFAB(I,1:NB)
ENDDO
!En este caso la componente de FAB que corresponde
!al nodo de nueva creación NA, se hace cero excepto
!en la localidad del nodo ganador en FB. Permitiendo así
!que el nuevo nodo de FA prediga el nodo FB correspondiente.
FAB(NA,:) = 0
FAB(NA,IA_B) = 1
!Se aumenta el número de veces que ha sido predicho
!el nodo ganador en FB (cuyo índice esta representado por IA_B)
FB(IA_B)%CLASE = FB(IA_B)%CLASE + 1
!La componente clase del nodo recién creado en FA se hace
!igual al índice IA_B, que será el nodo que predecirá en FB.
FA(NA)%CLASE = IA_B
!Se libera la memoria de la variable auxiliar FFAB
DEALLOCATE(FFAB)
ENDIF
ENDIF
!*************************************************************
!*************************************************************
!*************************************************************
!       SE INICIA AL PROCESO DE PROPAGACIÓN DE SEÑALES HACIA ADELANTE
!*************************************************************
!*************************************************************
!*************************************************************
!Este proceso se lleva a cabo basado en la ec. 3.3 y que representa la
!de difusificación, y se utiliza también la ecuación 3.4

!NA = número de nodos existentes en la capa FA.
!ENT = No. de entradas a la capa FA
!FA(.)%WJ(.) = representa la entrada del vector de pesos en el nodo
!             este vector tiene ENT*2 elementos
DO K = 1, NA
DO I = 1, ENT

S(1) = IENT(I) - (1. - FA(K)%WJ(I+ENT))
S(2) = FA(K)%WJ(I) - IENT(I)

!este ciclo se basa en la ec. 3.4

DO J = 1,2

IF  (S(J)*GAMMA > 1.0) THEN

S(J) = 1.0

ELSE IF( (S(J)*GAMMA)>=0.0  .AND.  (S(J)*GAMMA)<=1.0 )THEN
```

```
514. S(J) = S(J)*GAMMA

515. ELSE IF( (S(J)*GAMMA)<0.0 ) THEN

516. S(J) = 0.0

517. ENDIF

518. ENDDO

519. !Se obtiene el valor de membresia segun la ec. 3.3

520. FA(K)%NETA(I) = ( 1.0 - S(1) - S(2) ) * (1.0/ENT)

521. ENDDO

522. !Se obtiene lo indicado por la ec. 3.5
523. FA(K)%PRODUCTO = PRODUCT(FA(K)%NETA)

524. ENDDO
525. !El siguiente ciclo se utiliza para determinar lo
     requerido por la ecuación
526. !3.6, es decir, obtener el valor máximo de los nodos de
     la capa FA que predicen
527. !determinado nodo en FB.
528. !El ciclo más externo recorrerá todos los nodos de la
     capa FB, haciendo inicialmente
529. !la variable MM = 0 que es tipo integer y que permite
     contabilizar los nodos que
530. !predicen al nodo en turno mediante  ciclo precedente a
     la inicialización de MM.
531. !Si la variable MM es diferente de cero, se hace lo
     siguiente:
532. !se inicializa una variable K = 0 y se le asigna una
     memoria dinámica a la variable
533. !F de acuerdo al número de nodos que predicen al nodo
     FB en turno.
534. !En seguida se recorrerán los nodos FA para registrar
     su variable PRODUCTO
535. !en una localidad de F.
536. !Una vez terminado este proceso se determina el valor
     máximo del arreglo F que
537. !será el valor de la variable NETA del nodo FB en
     turno.
538. DO I = 1,NB
539. MM = 0
540. DO J = 1,NA
541. IF(FA(J)%CLASE == I ) MM = MM + 1
542. ENDDO
543. IF ( MM /= 0 ) THEN
544. K = 0
545. ALLOCATE( F ( MM ))
546. DO J = 1,NA
547. IF(FA(J)%CLASE == I ) THEN
548. K = K + 1
549. F(K) = FA(J)%PRODUCTO
550. ENDIF
551. ENDDO
552. R = MAXLOC(F)
553. IR = R(1)
554. FB(I)%NETA = F(IR)
555. DEALLOCATE(F)
556. ENDIF
557. ENDDO

558. !La variable suma se iguala a cero y en seguida se halla la
     suma de todas
559. !las NETA de los nodos FB, que representa el denominador de
     la ecuación
560. !3.9
561. SUMA = 0.0
562. DO I = 1,NB
563. SUMA=SUMA+FB(I)%NETA
564. ENDDO
565. !En los siguientes ciclos se obtienen el valor central de
     las funciones
566. !de membresia que se encuentran en todos los nodos FB
     correspondientes a las
567. !entradas 1,2,_,ENT_B. Este centro se representa mediante la
     ec. 3.10
568. DO I = 1, ENT_B
569. DO J = 1,NB
570. FB(J)%CENTRO(I)  =  0.5  *  (  1.0  +  FB(J)%WJ(I)  -
     FB(J)%WJ(I+ENT_B) )
571. ENDDO
572. ENDDO
573. !En los siguientes ciclos se desarrollan las ecuaciones 3.8
     y 3.9
574. !Inicialmente la variable VALORMEDIO_POR_SALIDA = 0.0, es
     decir, se inicializa
575. !una suma que es la ecuación 3.8. Una vez completada esa
     suma
576. !se obtiene el resultado final contenido por la red FALCON-
     ART, que es
577. !registrado en el arreglo FNETA
578. DO I = 1,ENT_B
579. VALORMEDIO_POR_SALIDA(I) = 0.0
580. DO J = 1,NB
581. VALORMEDIO_POR_SALIDA(I)=          VALORMEDIO_POR_SALIDA(I)+
     FB(J)%CENTRO(I) * FB(J)%NETA
582. ENDDO
583. FNETA(I) = VALORMEDIO_POR_SALIDA(I)/SUMA
584. ENDDO
585. !Mediante la instrucción siguiente se obtienen los errores
     en las salidas
586. !esta operación es correspondiente a la ec. 3.22, que será
     el error de la capa 5
587. !y que será propagado hacia las capas inferiores.
588. ERRORES(:) = IENT_B(:) - FNETA(:)
589. !Mediante los ciclos siguientes se modifican los límites
     indicados en la fig. 3.2
590. !es decir, los valores u_ij y v_ij en cada uno de los nodos de
     la capa FB
591. !Inicialmente se calcula la variable INCREMENTO representa
     el segundo sumando
592. !del lado derecho de la ecuación 3.19.
593. !En seguida se realiza la modificación de los pesos
     anteriores.
594. DO J = 1,ENT_B
595. DO I = 1,NB

596. INCREMENTO =    ETA * ERRORES(J) * (FB(I)%NETA/(2*SUMA))

597. FB(I)%WJ(J) = FB(I)%WJ(J) + INCREMENTO

598. FB(I)%WJ(J+ENT_B) =FB(I)%WJ(J+ENT_B) - INCREMENTO
```

```
599. ENDDO

600. ENDDO
601. !En los siguientes ciclos se calcula el error con el
     que contribuyen los nodos
602. !de la capa 4, y que esta dado por la ecuación 3.23
603. !los términos de esta operación son:
604. !FB()%CENTRO() = que representa el valor central de la
     función de membresia    !correspondiente.
605. !SUMA = que representa la sumatoria de las netas de los
     nodos de la capa FB.
606. !ERRORES = Es la magnitud del error generado en la capa
     5 de la estructura Falcon-!Art
607. !VALORMEDIO_POR_SALIDA el producto efectuado entre el
     valor medio de la función
608. !de membresia correspondiente y la salida del nodo
     respectivo.

609. DO I = 1,ENT_B
610. DO J = 1,NB
611. FB(J)%ERROR(I)=    ((FB(J)%CENTRO(I)    *    SUMA    -
     VALORMEDIO_POR_SALIDA(I))/SUMA**2) * ERRORES(I)
612. ENDDO
613. ENDDO
614. !Se calcula el error generado por la capa 3 de la
     estructura FALCON-ART
615. !Se determina inicialmente el valor máximo de las
     salidas proporcionadas
616. !por los nodos de la capa FB y que se da en el
     siguiente bloque de
617. !instrucciones:
618. ALLOCATE(AUXILIAR(NB))
619. AUXILIAR(:) = FB(:)%NETA
620. RR = MAXLOC(AUXILIAR)
621. K = RR(1)
622. !En seguida se recorren todos los nodos de FA con un
     valor de error igual con cero
623. !y se hace la variable L igual a la clase que predice
     el nodo FA correspondiente.
624. !La variable normalización se iguala con el producto
     indicado y que corresponde a la
625. !ec. 3.25.
626. !En seguida, se realiza una sumatoria donde interviene
     el error generado por la !clase predicha por el nodo FA
     y la variable NORMALIZACIÓN, esta operación
     !corresponde a la ec. 3.26
627. DO I = 1,NA
628. FA(I)%ERROR = 0.0
629. L = FA(I)%CLASE
630. NORMALIZACION = FA(I)%PRODUCTO/AUXILIAR(K)
631. DO J = 1,ENT_B
632. FA(I)%ERROR    =    FA(I)%ERROR    +    NORMALIZACION    *
     FB(L)%ERROR(J)
633. ENDDO
634. ENDDO
635. DEALLOCATE(AUXILIAR)
636. !Enseguida se modifican los limites Uij y Vij de las
     funciones de membresia de las
637. !entradas al espacio de entrada.
638. !El ciclo recorrerá cada uno de los nodos en FA;
     inicialmente nos auxiliamos de una
639. !variable XVEC donde registramos los valores de las NETAS de
     todos los nodos FA.
640. !Se efectúa un ciclo de acuerdo al número de entradas ENT, y
     también utilizamos las !variables auxiliares AUX1 y AUX2 con
     valores cero. Esto se hace de acuerdo a la ec.
641. !3.28

642. DO I = 1, NA
643. XVEC = FA(I)%NETA
644. DO J = 1,ENT
645. AUX1 = 0.0
646. AUX2 = 0.0
647. FA(I)%NETA(J) = 1.0

648. IF(0.0 <= IENT(J)-(1.-FA(I)%WJ(J+ENT))  .AND.   IENT(J)-(1.-
     FA(I)%WJ(J+ENT))<=1.0)AUX1=ETA/ENT
649. IF(0.0 <= FA(I)%WJ(J)-IENT(J)  .AND.  FA(I)%WJ(J)-IENT(J)<=1.0
     ) AUX2 = -ETA/ENT

650. PRODUCTO =     PRODUCT(FA(I)%NETA)

651. INCREMENTO = ETA * FA(I)%ERROR * PRODUCTO
652. FA(I)%WJ(J) = FA(I)%WJ(J) + AUX2 * INCREMENTO
653. FA(I)%WJ(J+ENT) = FA(I)%WJ(J+ENT) - AUX1 * INCREMENTO
654. FA(I)%NETA = XVEC
655. ENDDO
656. ENDDO

657. !Si el proceso de entranamiento ha llegado a su fin, se
     registran los pesos en los
658. !nodos de la capa FA y Fb en PESOS_FA.DAT y PESOS_FB.DAT,
     respectivamente. Además se !registran también las clases
     predichas por los nodos FA en CLASES_FA.DAT,
659. !así como también el número de clases en cada una de las
     capas en NO_CLASES.dat.

660. if(final == 1 )then

661. OPEN(8,FILE='nodos_tomados.dat')
662. OPEN(90,FILE=' PESOS_FA.DAT ')
663. OPEN(91,FILE=' CLASES_FA.DAT')
664. OPEN(92,FILE=' PESOS_FB.DAT ')
665. OPEN(93,FILE=' NO_CLASES.dat ' )

666. WRITE(8,*)'SE FORMARON', NA,' CLASES EN FA'
667. WRITE(8,*)'SE FORMARON', NB,' CLASES EN FB'

668. WRITE(93,*)NA
669. WRITE(93,*)NB

670. DO I = 1,NA
671. WRITE(8,*)'LA  CLASE',I,'  EN  FA  PREDICE  EN  FB  LA
     CLASE',FA(I)%CLASE
672. WRITE(91,*)FA(I)%CLASE
673. ENDDO

674. DO I = 1,NB
675. WRITE(8,*)'LA  CLASE',I,'  EN  FB  ES  PREDICHA',FB(I)%CLASE,'
     VECES'
676. ENDDO

677. DO I = 1,NA
```

```
670. IF(I == 1)WRITE(8,*)'   LOS PESOS EN LAS UNIDADES DE FA
     Y LA CLASE     '
679. WRITE(8,*)FA(I)%WJ
680. WRITE(90,*)FA(I)%WJ
681. !    100 FORMAT(4E12.6)
682. ENDDO

683. DO I = 1,NB
684. IF(I == 1)WRITE(8,*)'   LOS PESOS EN LAS UNIDADES DE FB
     Y LA CLASE     '
685. WRITE(8,*)FB(I)%WJ,I
686. WRITE(92,*)FB(I)%WJ
687. !    200 FORMAT(2E12.6,5X,I4,I4)
688. ENDDO

689. IF ( ENT_B == 1 ) THEN

690. OPEN(22,FILE = 'USER',TITLE = 'OTROS_GRAFICOS' ,IOFOCUS
     = .TRUE.)
691. STATUS_2 = SETWINDOW(INVERT, -1.5, 1.05, 2.5, 0.0)
692. AUX3 = .96
693. DUMMY = RECTANGLE_W($GBORDER,1.0,1.0,0.0,0.0)
694. DO I = 1,NB
695. CALL MOVETO_W(FB(I)%WJ(1),AUX3,XY)
696. SJACOB = LINETO_W(FB(I)%WJ(1), AUX3)
697. SJACOB = LINETO_W(1.-FB(I)%WJ(2),AUX3)
698. AUX3 = AUX3 - 0.009
699. ENDDO

700. PAUSE
701. CLOSE(22)
702. ELSE IF ( ENT_B == 2 ) THEN
703. OPEN(22,FILE = 'USER',TITLE = 'GRAFICOS' ,IOFOCUS =
     .TRUE.)
704. STATUS = SETWINDOW(INVERT, UPX, UPY, DOWNX, DOWNY)
705. OLDCOLOR = SETCOLORRGB(#FFFFFF)  ! COLOR BLANCO
706. DUMMY = RECTANGLE_W($GFILLINTERIOR,DOWNX,DOWNY,UPX,UPY)
707. OLDCOLOR = SETCOLORRGB(#FF0000)  ! COLOR AZUL
708. DUMMY = RECTANGLE_W($GBORDER,0.0,0.0,1.0,1.0)
709. OLDCOLOR = SETCOLORRGB(#000000)    ! COLOR NEGRO

710. DO I = 1,NB
711. IF ( FB(I)%CLASE/=0)THEN
712. XX1 = FB(I)%WJ(1)
713. XY1 = FB(I)%WJ(2)
714. XX2 = 1.0 - FB(I)%WJ(3)
715. XY2 = 1.0 - FB(I)%WJ(4)
716. DUMMY = RECTANGLE_W($GBORDER,XX2,XY2,XX1,XY1)
717. ENDIF
718. ENDDO

719. PAUSE
720. CLOSE(22)

721. ENDIF

722. CLOSE(90)
723. CLOSE(91)
724. CLOSE(92)
725. CLOSE(93)

726. DEALLOCATE(XAB,YB)
727. DEALLOCATE(IENT,IENT_B)
728. DEALLOCATE(FA,FB,FAB,VAR,VAR_B,FNETA,ERRORES)
729. DEALLOCATE(VALORMEDIO_POR_SALIDA)

730. endif

731. END SUBROUTINE
```

C. 4 SUBROUTINE REGULADOR_FALCON

```
1.  !*************************************************************
    ************************
2.  !*************************************************************
    ************************
3.  !*************************************************************
    ************************
4.  !La subrutina siguiente realiza únicamente la propagación de
    señales hacia delante
5.  !y tiene como entradas las señales siguientes:
6.  !IENT = que es es vector de entrada al espacio de entrada.
7.  !INICIO = es la variable que indica  el dimensionamiento de
    las variables que
8.  !modifican su estructura durante el programa
9.  !NETA = variable que indica la señal envia al programa
    principal durante la etapa de producción.
10. !ITERMINA= señal que indica el lieberamiento de memoria
    dinámica una vez concluido lel programa.
11. !*************************************************************
    ************************
12. !*************************************************************
    ************************
13. !*************************************************************
    ************************


14. SUBROUTINE REGULADOR_FALCON(IENT,INICIO,NETA,ITERMINA)

15. use msflib

16. INTEGER,PARAMETER                          :: ENT   = 4
17. INTEGER,PARAMETER                          :: ENT_B = 2
18. INTEGER,PARAMETER                        :: GAMMA = 4
19. REAL*8,INTENT(OUT)                     :: NETA(ENT_B)
20. REAL*8                                   SUMA
21. INTEGER                                  R(1)
22. REAL*8,DIMENSION(2)                    :: S
23. REAL*8,DIMENSION(ENT_B)                :: VALORMEDIO
24. REAL*8,DIMENSION(ENT_B)                :: FNETA
25. REAL*8,DIMENSION(:),ALLOCATABLE        :: F
26. REAL*8,DIMENSION(ENT),INTENT(IN)       :: IENT


27. TYPE ELEMENTOS_FA
28. REAL*8,DIMENSION(2*ENT) ::  WJ
29. INTEGER                 ::  CLASE
30. REAL*8,DIMENSION(ENT)   ::  NETA
31. REAL*8                  ::  PRODUCTO
32. END TYPE ELEMENTOS_FA

33. TYPE ELEMENTOS_FB
34. REAL*8,DIMENSION(2*ENT_B) ::  WJ
35. REAL*8,DIMENSION(ENT_B)   ::  CENTRO
```

```
36.  REAL*8                    ::  NETA
37.  END TYPE ELEMENTOS_FB

38.  TYPE(ELEMENTOS_FA),DIMENSION(:),ALLOCATABLE   :: FA
39.  TYPE(ELEMENTOS_FB),DIMENSION(:),ALLOCATABLE   :: FB
40.  !MS$DEBUG

41.  IF ( INICIO == 1 ) THEN

42.  OPEN(90,FILE='PESOS_FA.DAT')
43.  OPEN(92,FILE='PESOS_FB.DAT')
44.  OPEN(91,FILE='CLASES_FA.DAT')
45.  OPEN(93,FILE='NO_CLASES.DAT ' )

46.  INICIO = 0
47.  READ(93,*)NA
48.  READ(93,*)NB
49.  ALLOCATE(FA(NA),FB(NB))

50.  DO I = 1,NA
51.  READ(90,*) FA(I)%WJ
52.  READ(91,*)  FA(I)%CLASE
53.  ENDDO

54.  DO I = 1, NB
55.  READ(92,*) FB(I)%WJ
56.  ENDDO
57.  CLOSE(90)
58.  CLOSE(91)
59.  CLOSE(92)
60.  CLOSE(93)

61.  ENDIF
62.  !Se llevan a cabo los mismos segmentos de la Subrutina
     Control_Falcon_art que realizan la propagación de señales
     hacia delante.
63.  DO K = 1, NA

64.  DO I = 1, ENT

65.  S(1) = IENT(I) + (1. - FA(K)%WJ(I+ENT))
66.  S(2) = FA(K)%WJ(I) - IENT(I)


67.  DO J = 1,2

68.  IF  (S(J)*GAMMA > 1.0) THEN

69.  S(J) = 1.0

70.  ELSE IF( (S(J)*GAMMA)>=0.0   .AND.   (S(J)*GAMMA)<=1.0
     )THEN

71.  S(J) = S(J)*GAMMA

72.  ELSE IF( (S(J)*GAMMA)<0.0 ) THEN

73.  S(J) = 0.0

74.  ENDIF

75.  ENDDO

76.  FA(K)%NETA(I) = ( 1.0 - S(1) - S(2) ) * (1.0/ENT)

77.  ENDDO

78.  FA(K)%PRODUCTO = PRODUCT(FA(K)%NETA)

79.  ENDDO

80.  DO I = 1,NB

81.  MM = 0

82.  DO J = 1,NA
83.  IF(FA(J)%CLASE == I ) MM = MM + 1
84.  ENDDO

85.  IF ( MM /= 0 ) THEN

86.  K = 0
87.  ALLOCATE( F ( MM ))

88.  DO J = 1,NA
89.  IF(FA(J)%CLASE == I ) THEN
90.  K = K + 1
91.  F(K) = FA(J)%PRODUCTO
92.  ENDIF
93.  ENDDO

94.  R = MAXLOC(F)
95.  IR = R(1)
96.  FB(I)%NETA = F(IR)
97.  DEALLOCATE(F)

98.  ENDIF

99.  ENDDO

100. SUMA = 0.0

101. DO I = 1,NB
102. SUMA=SUMA+FB(I)%NETA
103. ENDDO

104. DO I = 1, ENT_B
105. DO J = 1,NB
106. FB(J)%CENTRO(I)   =   0.5   *   (   1.0   +   FB(J)%WJ(I)   -
     FB(J)%WJ(I+ENT_B) )
107. ENDDO
108. ENDDO

109. IF ( SUMA >= 1.0E-06 ) THEN
110. DO I = 1,ENT_B
111. VALORMEDIO(I) = 0.0
112. DO J = 1,NB
113. VALORMEDIO(I)= VALORMEDIO(I)+ FB(J)%CENTRO(I) * FB(J)%NETA
114. ENDDO
115. FNETA(I) = VALORMEDIO(I)/SUMA
116. ENDDO
117. ELSE
118. FNETA = 0.0000000001
119. ENDIF
120. neta = fneta
121. IF ( ITERMINA == 1)DEALLOCATE(FA,FB)
```

```
122. END SUBROUTINE
```

C. 5 MÓDULO PARÁMETROS_IMPORTANTES

```
1.  module parametros_fundamentales

2.  INTEGER,PARAMETER                :: ENT   = 4

3.  INTEGER,PARAMETER                :: ENT_B = 2

4.  REAL*8,PARAMETER                 :: RO_B  = 0.4

5.  REAL*8,PARAMETER                 :: RO_A  = 0.5

6.  REAL*8,PARAMETER                 :: ETA   = 0.001


7.  end module parametros_fundamentales
```

C. 6 MÓDULO MÁQUINA SÍNCRONA

```
1.  module Maquina_Sincrona

2.  REAL,PARAMETER :: VMAX = 6.0,VMIN = -6.0
3.  REAL,PARAMETER ::  CAM1  = 0.2,CAM2  = 0.1,CAM3  = -
    0.2,CAM4 = 0.1
4.  REAL,PARAMETER :: T1I = 20.0,T1F = 40.0
5.  REAL,PARAMETER :: T2I = 120.0,T2F = 200.0
6.  REAL,PARAMETER :: T3I = 220.0,T3F = 270.0
7.  REAL,PARAMETER :: T4I = 300.0,T4F = 350.0
8.  REAL,PARAMETER :: T5I = 380.0,T5F = 390.0

9.  real,parameter :: RA = 0.0
10. real,parameter :: re = 0.024
11. real,parameter :: xe = 0.115
12. real,parameter :: kd = 3.0
13. real,parameter :: w0 = 377.0
14. real,parameter :: M = 5.5294
15. real,parameter :: tsm = 0.1
16. real,parameter :: tch = 0.15
17. real,parameter :: trh = 5.0
18. real,parameter :: fhp = 0.33
19. real,parameter :: tx = 0.025
20. real,parameter :: tq02p = 0.065
21. real,parameter :: td0p = 5.66
22. real,parameter :: td02p = 0.041
23. real,parameter :: xd = 1.904
24. real,parameter :: xdp = 0.312
25. real,parameter :: xd2p = 0.266
26. real,parameter :: xq = 1.881
27. real,parameter :: xq2p = 0.266

28. REAL,PARAMETER :: KA = 400.0
29. real,parameter :: ta = 0.02
30. real,parameter :: kf = 0.008
31. real,parameter :: tf = 1.0

32. real vbus,vbus_aux,tinic,tfin,h,tinic_falla,tdura
33. integer n,respuesta,epocas,N_de_condiciones,epocas_global
34. integer,parameter :: na = 10
35. real*8 va,thet,p,q,ug,id,iq,te,vref
36. real*8 bva,bthet,bp,bq,bug,bid,biq,bte
37. real*8  va_aux, thet_aux,  te_aux, id_aux
38. real*8  iq_aux, p_aux, q_aux, ug_aux
39. real*8 global
40. integer ll
41. real*8,dimension(:),allocatable :: ai,ai_aux,bai

42. CHARACTER(8) TIEMPO_EFECTIVO

43. contains

44. subroutine
    condicion_inicial(va,vbus,thet,te,id,iq,vref,p,q,ug,ai)

45. real*8,intent(inout)             :: thet
46. real*8,intent(inout)              :: va
47. real,intent(in)                  :: vbus
48. real*8,intent(out)               :: p,q,te,id,iq,vref,ug
49. real*8,dimension(:),intent(out) :: ai
50. complex ea,eb,pow,zpq
51. real*8 phi,del,delr,eqx,eqy,edppr,eqppr,eqpr,esr,esi,ia
52. real*8 aux(4)

53. read(150,*)aux
54. print*,aux

55. va = aux(1)
56. thet = aux(2)
57. p = aux(3)
58. q = aux(4)

59. thet = thet*(3.1415927/180.0)
60. ia = dsqrt(p*p+q*q)/va
61. if (ia > 1.0e-06) then
62. phi = atan2(q,p)
63. else
64. phi = 0.0
65. endif

66. eqx = va + ra*ia*dcos(phi) + xq*ia*dsin(phi)
67. eqy = -ra*ia*dsin(phi) + xq*ia*dcos(phi)
68. del = atan2(eqy,eqx)

69. id = ia*dsin(del+phi)
70. iq = ia*dcos(del+phi)

71. edppr = va*dsin(del) + ra*id - xq2p*iq
72. eqppr = va*dcos(del) + ra*iq + xd2p*id

73. eqpr = eqppr + (xdp-xd2p)*id

74. vf = eqpr + (xd-xdp)*id
75. te   = id*edppr + iq*eqppr

76. delr = thet+del
```

```
77.  vref = (vf/KA) + va
78.  ug = te

79.  ai(1) = delr
80.  ai(2) = 0.0
81.  ai(3) = eqpr
82.  ai(4) = eqppr
83.  ai(5) = edppr
84.  ai(6) = ke*(vref-va)
85.  ai(7) = te
86.  ai(8) = te
87.  ai(9) = te
88.  ai(10) = 0.0

89.  end subroutine

90.  ! rungo kutta de 4o. orden
91.  subroutine kutta(ainic,id,iq,te,va,ug,vr=f,h)

92.  real*8,intent(in) :: id,iq,te,va,ug,vref
93.  real,intent(in)    :: h
94.  real*8,dimension(:),allocatable :: temp,dk1,dk2,dk3,dk4
95.  real*8,dimension(:),intent(inout)  :: ainic

96.  allocate(temp(na),dk1(na),dk2(na),dk3(na),dk4(na))

97.  call ecdif(ainic,dk1,iq,id,te,va,ug,vref)
98.  do i = 1,nA
99.  temp(i) = ainic(i) + dk1(i) * (h*0.5)
100. enddo
101. call ecdif(temp,dk2,iq,id,te,va,ug,vref)
102. do i = 1,na
103. temp(i) = ainic(i) + dk2(i) * (h*0.5)
104. enddo
105. call ecdif(temp,dk3,iq,id,te,va,ug,vref)
106. do i = 1,na
107. temp(i) = ainic(i) + dk3(i) * h
108. enddo
109. call ecdif(temp,dk4,iq,id,te,va,ug,vref)

110. do i = 1,na
111. ainic(i) = ainic(i) + (( dk1(i) + dk4(i) +
     2*(dk2(i)+dk3(i)))*h)/6.0
112. enddo

113. if( ainic(6) >= VMIN .AND. ainic(6) <= VMAX) then
114. ainic(6) = ainic(6)
115. else if( ainic(6) > VMAX ) then
116. ainic(6) = VMAX
117. else if( ainic(6) < VMIN ) then
118. ainic(6) = VMIN
119. endif

120. deallocate(temp,dk1,dk2,dk3,dk4)


121. end subroutine

122. subroutine ecdif(a,d,iq,id,te,va,ug,vref)

123. real*8,intent(in) :: id,iq,te,va,ug,vref
124. real*8,dimension(:),intent(in)  :: a
125. real*8,dimension(:),intent(out)  :: d

126. d(1) =  wo * a(2)
127. d(2) = ( -kd * a(2) + a(9) - te )/M
128. d(3) = (-a(3) + a(6) - (xd-xdp)*id)/td0p
129. d(4) = ( a(3) - a(4) - (xdp-xd2p)*id ) / td02p
130. d(5) = ( -a(5) - (xq-xq2p)*iq )/ tq02p
131. d(6) = (ka*( vref - va + a(10)) - a(6))/ ta
132. d(7) = ( -a(7) + ug )/ tsm
133. d(8) = ( a(7) - a(8) )/ tch
134. d(9) = ( fhp * d(8) + a(8) - a(9) ) / trh
135. d(10)= (kf*d(6) - a(10))/tf

136. end subroutine


137. subroutine dinamica( ai,id,iq,te,va,p,q,vbus)

138. real*8,dimension(:),intent(in)  :: ai

139. !ms$debug

140. real*8 b1,b4,b5
141. real*8 b2,b3

142. real,intent(in)     :: vbus
143. real*8,intent(out) :: id,iq,te,va,p,q
144. real*8 vd1,vq1
145. complex s,vet,tti

146. b1 = ro + ra
147. b2 = ai(5) - vbus*dsin(ai(1))
148. b3 = ai(4) - vbus*dcos(ai(1))
149. b4 = xe + xq2p
150. b5 = xe + xd2p

151. id = ( b1*b2 + b4*b3 )/ (b1**2 + b5*b4)
152. iq = ( b1*b3 - b5*b2 ) / ( b1**2 + b5*b4 )

153. te = ai(5)*id + ai(4)*iq
154. vd1 = ai(5) - ra*id + xq2p*iq
155. vq1 = ai(4) - ra*iq - xd2p*id
156. va = dsqrt(vd1*vd1 + vq1*vq1 )
157. vet = cmplx(vq1,vd1)
158. tti = cmplx(iq,-id)
159. s = vet * tti
160. p = real(s)
161. q = -aimag(s)

162. end subroutine

163. SUBROUTINE CAMBIO(TINIC,UG)

164. REAL,INTENT(IN)    :: TINIC
165. REAL*8,INTENT(INOUT) :: UG


166. IF(TINIC>=T1I .AND. TINIC<= T1F)THEN
167. UG1 = UG
168. UG2 = UG1+CAM;
169. ELSE IF(TINIC>T1F .AND. TINIC< T2I)THEN
170. EM = (UG2-UG1)/(T2I-T1F)
171. UG = (( TINIC-T1F)*EM)+UG1
172. ELSE IF(TINIC>=T2I .AND. TINIC<= T2F)THEN
```

```
173. UG = UG2
174. UG3 = UG2+CAM2
175. ELSE IF(TINIC>T2F .AND. TINIC< T3I)THEN
176. EM =(UG3-UG2)/(T3I-T2F)
177. UG = ((TINIC-T2F)*EM)+UG2
178. ELSE IF(TINIC>=T3I .AND. TINIC<= T3F)THEN
179. UG = UG3
180. UG4 = UG3+CAM3
181. ELSE IF(TINIC>T3F .AND. TINIC< T4I)THEN
182. EM = (UG4-UG3)/(T4I-T3F)
183. UG =((TINIC-T3F)*EM)+UG3
184. ELSE IF(TINIC>=T4I .AND. TINIC<= T4F)THEN
185. UG = UG4
186. UG5 = UG4+CAM4
187. ELSE IF(TINIC>T4F .AND. TINIC<T5I)THEN
188. EM = (UG5-UG4)/(T5I-T4F)
189. UG = ((TINIC-T4F)*EM) + UG4
190. ELSE IF(TINIC>=T5I .AND. TINIC<= T5F)THEN
191. UG = UG5
192. END IF

193. END SUBROUTINE

194. !LA SUBRUTINA DATOS INICIALES REQUIERE AL USUARIO
195. !TINIC(T1): EL TIEMPO INICIAL DE SIMULACION
196. !TFIN(T2) : EL TIEMPO FINAL DE SIMULACION
197. !H(H)     : EL PASO DE INTEGRACION
198. !VA(V1)   : EL VOLTAJE EN TERMINALES DE LA MAQUINA
199. !VBUS(V2) : EL VOLTAJE EN EL VBUS
200. !THET(ANG): EL ANGULO DEL VOLTAJE

201. subroutine datos_iniciales(t1,t2,hh,nn)

202. !LA INSTRUCCION INTENT(OUT) INDICA AL PROGRAMA QUE SON
     VARIABLES
203. !QUE SERAN PROPORCIONADAS POR LA SUBRUTINA
204. real,intent(out)    :: t1,t2,hh
205. integer,intent(out) :: nn

206. open(14,file=' entrada2.dat')

207. !            print*,' CUAL ES SU TIEMPO INICIAL DE
     SIMULACION?'
208. read(14,*)t1
209. !            print*,t1
210. !            read*,t1
211. !            print*,' CUAL ES SU TIEMPO FINAL DE
     SIMULACION?'
212. read(14,*)t2
213. !            print*,t2
214. !            read*,t2
215. !            print*,'   CUAL   ES   SU   PASO   DE
     INTEGRACION?'
216. read(14,*)hh
217. !            print*,hh
218. !            read*,h
219. !            call condiciones_aleatorias(v1,pp)
220. !            print*,v1,pp
221. !    pause
222. !    print*,' CUAL ES EL VOLTAJE EN TERMINALES DE LA
     MAQUINA?'
223. !            read(14,*)v1
224. !            print*,v1
225. !            read*,v1
226. !            print*,' CUAL ES EL ANGULO DEL VOLTAJE?'
227. !            read(14,*)ang
228. !            print*,ang
229. !            read*,ang
230. !            print*,' CUAL ES EL VOLTAJE EN EL BUS?'
231. !            read(14,*)v2
232. !            print*,v2
233. !            read*,v2

234. !SE OBTIENE EL NUMERO DE PASOS PARA REALIZAR LA INTEGRACION

235. nn = t2/hh

236. !            print*, ' el numero de pasos es ', nn

237. close(14)

238. end subroutine

239. subroutine solicitud(t,tt,r,epocas,N_de_condiciones)

240. integer,intent(out) :: r,epocas,N_de_condiciones
241. real,intent(out)    :: t,tt
242. real ciclos

243. print*, ' sobre que numero de condiciones desea realizar el
     entrenamiento ? '
244. read*, N_de_condiciones

245. print*, ' cuantas epocas desea en el entrenamiento ? '
246. read*, epocas

247. print*, ' desea realizar la simulacion de una falla? '
248. print*, '           si[1]          no[2]                 '

249. read*, r

250. if ( r == 1) then

251. print*, ' cual es su tiempo inicial de falla? '
252. read*,t

253. print*, ' en cuantos ciclos desea eliminar la falla? '
254. read*,ciclos

255. tt = (ciclos*(1.0/60.0)) + t
256. else
257. t = 0.0
258. tt = 0.0
259. endif


260. end subroutine

261. subroutine falla(t,tt,ttt,vbus)

262. real,intent(in)  :: tt,ttt,t
263. real,intent(out) :: vbus

264. if(t >= tt .and. t <= ttt ) then
265. vbus = 0.1
266. else
```

```
267.  vbus = 1.0
268.  end if

269.  end subroutine

270.  end module Maquina_sincrona
```

D. 7 MÓDULO DECLARACIONES

```
1.  !*****************************************************************
    ***************
2.  !*****************************************************************
    ***************
    !                MODULO DECLARACIONES
    !           Contiene las variables requeridas para el
    funcionamiento
3.  !        del sistema FALCON-ART
4.  !        Fecha de documentación: SEP/2000
5.  !*****************************************************************
    ***************
6.  !*****************************************************************
    ***************
7.  MODULE DECLARACIONES
8.  !Se indica el module adicional que se requiere.
9.  USE PARAMETROS_FUNDAMENTALES

10. !Este bloque de declaraciones se utiliza
11. !para la graficación de las variables a observar.
12. INTEGER(2)  DUMMY,STATUS
13. REAL(8) XX1,XY1,XX2,XY2
14. LOGICAL(2)  INVERT /.TRUE./
15. REAL(8) UPX /1.5/, UPY /1.5/
16. REAL(8) DOWNX /-0.5/, DOWNY /-0.5/
17. !Se declaran las variables siguientes:
18. ! GAMMA = variable que define la difusividad de los
    conjuntos
19. !     sobre espacio de entrada.
20. ! ALFA = variable que indica al programa que indica
21. !      al sistema FALCON-ART que trabaje en el limite
    conservativo.
22. ! RO_AB = variable que es el parámetro de aprendijaze
23. !     para el campo de mapeo FAB y 0<RO_AB<1.0
24. ! RESTRICCION = Determina el tamaño de las clases en
25. !     el campo de entrada FA.
26. ! RESTRICCION_B = determina el tamaño de las clases en
    el
27. !     campo de salida FB.
28. ! DI = variable que no es utilizada en este programa,
    solo se
29. !     mantiene ahi debido a la utilización del sistema
30. !     FALCON-ART en otras pruebas.
31. INTEGER,PARAMETER                    :: GAMMA = 4
32. REAL*8,PARAMETER                     :: ALFA  = 0.001
33. REAL*8,PARAMETER                     :: RO_AB = 0.4
34. REAL*8,PARAMETER                     ::  RESTRICCION_B  =
    ENT_B * ( 1. - RO_B )
35. REAL*8,PARAMETER                     :: RESTRICCION
    = ENT * ( 1. - RO_A )
36. REAL*8,PARAMETER                     ::        PI       =
    3.14159265359
37. ! variables auxiliares ( sin relevancia )
38. REAL*8  SUMA,AUX3,Y1,H
39. INTEGER ENTRENAMIENTO,EXPERIMENTO,AR(1),AR_B(1)
40. !Las estructuras que conforman los nodos FA están formados
    por los
41. !siguientes elementos:
42. !   TJ = que representa el valor de la función de activación
43. !     del nodo al cual pertenece.
44. !   RJ = representa el tamaño de la clase contenida en el
    nodo
45. !     al cual pertenece.
46. !     WJ1 = es el vector de pesos utilizado mientras se
    ejecutan
47. !      operaciones y no afectar el vector de pesos WJ, una
    vez
48. !      terminado el proceso este vector se iguala al vector
    WJ.
49. !   WJ = es el vector almacenado en el nodo correspondiente
    y
50. !     que representa la clase codificada en este nodo.
51. !      CLASE = Es la clase FB predicha por el nodo
    correspondiente.
52. !     NETA = variable que es utilizada en el proceso de
    fuzificación
53. !     PRODUCTO = Variable que permite obtener el resultado
    total que
54. !       se obtiene a partir del producto del vector neta,
    realizado entre
55. !     sus entradas.
56. !   ERROR = es el valor del error con el que contribuye el
    nodo con el
57. !   error global.

    TYPE ELEMENTOS_FA
58. REAL*8                  ::  TJ
        REAL*8                  ::  RJ
        REAL*8,DIMENSION(2*ENT) ::  WJ1
        REAL*8,DIMENSION(2*ENT) ::  WJ
59. INTEGER                 ::  CLASE
60. REAL*8,DIMENSION(ENT)   ::  NETA
61. REAL*8                  ::  PRODUCTO
62. REAL*8                  ::  ERROR
63. END TYPE ELEMENTOS_FA
64. !Las estructuras que conforman los nodos FB están formados
    por los
    !siguientes elementos:
65. !TJ, RJ, WJ1, WJ, ERROR, NETA corresponden a sus
66. !contrapartes en la capa FA, los nuevos elementos son:
67. !CENTRO = representa el centro del área en el proceso de
    dedifusificación.
68. !CLASE = representa el número de veces que el nodo
    correspondiente
69. !       es predicho por los nodos de FA.
70. TYPE ELEMENTOS_FB
        REAL*8                    ::  TJ
        REAL*8                    ::  RJ
        REAL*8,DIMENSION(2*ENT_B) ::  WJ1
        REAL*8,DIMENSION(2*ENT_B) ::  WJ
        REAL*8,DIMENSION(ENT_B)   ::  CENTRO
        REAL*8,DIMENSION(ENT_B)   ::  ERROR
71. INTEGER                 ::  CLASE
72. REAL*8                  ::  NETA
73. END TYPE ELEMENTOS_FB
```

```
74.  REAL*8,DIMENSION(ENT)                                    ::
     RJJ
75.  REAL*8,DIMENSION(ENT_B)                                  ::
     RJJ_B
76.  REAL*8,DIMENSION(2)                                      :: B
77.  REAL*8,DIMENSION(:),ALLOCATABLE                          :: P
78.  INTEGER  R(1)
79.  REAL*8  INCREMENTO
80.  INTEGER RR(1)
81.  REAL*8 NORMALIZACION
82.  REAL*8,DIMENSION(:),ALLOCATABLE                          ::
     AUXILIAR

83.  REAL*8,DIMENSION(ENT)
     :: XVEC
84.  REAL*8  AUX1,AUX2,PRODUCTO
85.  REAL*8,SAVE ::  ERROR_GLOBAL_AUX
86.  END MODULE DECLARACIONES
```

APÉNDICE D

CONDICIONES INICIALES PARA EL ENTRENAMIENTO DE LA ESTRUCTURA FALCON-ART

D. 1 Introducción

En este segmento se presentan 200 condiciones aleatorias con las cuales se probó la red FALCON-ART, las que aparecen en cursivas negritas, pertenecen al grupo con el cual se entrenó, las restantes son condiciones de prueba.

D. 2 Condiciones de entrenamiento

	Voltaje terminal	ángulo terminal	potencia activa	potencia reactiva
1.	*.1000000e+01*	*.6000000e+01*	*.8805000e+00*	*-.1361000e+00*
2.	*.1000000e+01*	*.6807718e+01*	*.1000000e+01*	*-.1473878e+00*
3.	*.1042691E+01*	*.5361553E+01*	*.8971906E+00*	*.2394987E+00*
4.	*.1029103E+01*	*.3981104E+01*	*.6517569E+00*	*.1460104E+00*
5.	*.1015208E+01*	*.4251334E+01*	*.6588190E+00*	*.2105461E-01*
6.	*.1045324E+01*	*.4056943E+01*	*.7031847E+00*	*.2880037E+00*
7.	*.1033225E+01*	*.4214542E+01*	*.6972846E+00*	*.1772876E+00*
8.	*.9998438E+00*	*.5021966E+01*	*.7357140E+00*	*-.1215235E+00*
9.	*.9813626E+00*	*.4173386E+01*	*.5678285E+00*	*-.2549196E+00*
10.	*.1056394E+01*	*.3734606E+01*	*.6808600E+00*	*.3954524E+00*
11.	*.1017962E+01*	*.5398565E+01*	*.8376989E+00*	*.2343407E-01*
12.	*.1027015E+01*	*.4260949E+01*	*.6890222E+00*	*.1221480E+00*
13.	*.1038928E+01*	*.3536333E+01*	*.6077517E+00*	*.2420443E+00*
14.	*.1018066E+01*	*.5753865E+01*	*.8913938E+00*	*.1850880E-01*
15.	*.9896917E+00*	*.6534585E+01*	*.9319504E+00*	*-.2272967E+00*
16.	*.9919246E+00*	*.4030202E+01*	*.5712479E+00*	*-.1675409E+00*
17.	*.1049300E+01*	*.5777179E+01*	*.9793495E+00*	*.2917894E+00*
18.	*.9919636E+00*	*.5799645E+01*	*.8302205E+00*	*-.1984313E+00*
19.	*.1012302E+01*	*.5270969E+01*	*.8040097E+00*	*-.2228127E-01*
20.	*.9904293E+00*	*.6707360E+01*	*.9592349E+00*	*-.2236688E+00*
21.	*.1048568E+01*	*.5494096E+01*	*.9334847E+00*	*.2899110E+00*
22.	*.1016150E+01*	*.4143004E+01*	*.6448828E+00*	*.3120595E-01*
23.	*.1001640E+01*	*.3579380E+01*	*.5273286E+00*	*-.7877937E-01*
24.	*.1042796E+01*	*.3127172E+01*	*.5543289E+00*	*.2858777E+00*
25.	*.1047059E+01*	*.3745733E+01*	*.6595582E+00*	*.3102633E+00*

90.	.9967496E+00	.5206982E+01	.7552863E+00	-.1500301E+00
91.	.1001523E+01	.4809214E+01	.7084445E+00	-.1039298E+00
92.	.1047644E+01	.5231193E+01	.8903183E+00	.2861753E+00
93.	.1002534E+01	.5848431E+01	.8647229E+00	-.1129971E+00
94.	.1059497E+01	.3903792E+01	.7149498E+00	.4203133E+00
95.	.1045869E+01	.4344368E+01	.7488131E+00	.2870107E+00
96.	.1045402E+01	.2758820E+01	.5039248E+00	.3180970E+00
97.	.1029238E+01	.4237147E+01	.6908831E+00	.1419513E+00
98.	.9900431E+00	.6445171E+01	.9197907E+00	-.2232645E+00
99.	.1005101E+01	.5318445E+01	.7927478E+00	-.8323474E-01
100.	.1008715E+01	.3446585E+01	.5237703E+00	-.1700383E-01
101.	.1034632E+01	.4635397E+01	.7649232E+00	.1813718E+00
102.	.9859474E+00	.5532755E+01	.7759993E+00	-.2424856E+00
103.	.9872275E+00	.6945496E+01	.9854334E+00	-.2523055E+00
104.	.9932953E+00	.4323248E+01	.6172689E+00	-.1621557E+00
105.	.1030138E+01	.4222670E+01	.6909029E+00	.1500913E+00
106.	.9815740E+00	.6102023E+01	.8476570E+00	-.2858159E+00
107.	.1007704E+01	.4186299E+01	.6311469E+00	-.4083129E-01
108.	.1001657E+01	.5439570E+01	.8019449E+00	-.1137064E+00
109.	.1059741E+01	.4712473E+01	.8418001E+00	.4059913E+00
110.	.1040833E+01	.4745278E+01	.7975901E+00	.2341321E+00
111.	.1016893E+01	.6128510E+01	.9445974E+00	.2781002E-02
112.	.1016554E+01	.3178414E+01	.5016401E+00	.5523482E-01
113.	.9945771E+00	.5557774E+01	.8013920E+00	-.1734910E+00
114.	.1020008E+01	.3312173E+01	.5295156E+00	.8176686E-01
115.	.9826534E+00	.3805492E+01	.5175695E+00	-.2373971E+00
116.	.1040227E+01	.3449100E+01	.5975219E+00	.2555542E+00
117.	.1010000E+01	.4392074E+01	.6672293E+00	-.2563500E-01
118.	.1025520E+01	.3916970E+01	.6334176E+00	.1162159E+00
119.	.9919895E+00	.4500812E+01	.6401569E+00	-.1760957E+00
120.	.1036535E+01	.4229181E+01	.7077187E+00	.2061444E+00
121.	.1009519E+01	.5172045E+01	.7821750E+00	-.4393208E-01
122.	.1055159E+01	.3480723E+01	.6384043E+00	.3897963E+00
123.	.1031796E+01	.3697488E+01	.6152368E+00	.1755578E+00
124.	.1010181E+01	.6167799E+01	.9324389E+00	-.5431757E-01
125.	.1021495E+01	.4943614E+01	.7782996E+00	.6154536E-01
126.	.1036354E+01	.3279940E+01	.5625525E+00	.2249705E+00
127.	.1043622E+01	.4793619E+01	.8122288E+00	.2580995E+00
128.	.1015682E+01	.4309330E+01	.6686403E+00	.2393359E-01
129.	.1009882E+01	.5209434E+01	.7886664E+00	-.4153687E-01
130.	.1053862E+01	.3658240E+01	.6627494E+00	.3739496E+00
131.	.1013478E+01	.4311619E+01	.6636463E+00	.5222655E-02
132.	.1054888E+01	.4932549E+01	.8632797E+00	.3572954E+00
133.	.1023122E+01	.4255363E+01	.6786420E+00	.8860616E-01
134.	.1048811E+01	.4020564E+01	.7062737E+00	.3202035E+00
135.	.9984919E+00	.6067038E+01	.8864787E+00	-.1494667E+00
136.	.1003281E+01	.4526588E+01	.6709546E+00	-.8419066E-01
137.	.9821888E+00	.6017122E+01	.8369111E+00	-.2797265E+00
138.	.1007283E+01	.6662190E+01	.9983460E+00	-.8541706E-01
139.	.1011003E+01	.5764726E+01	.8744129E+00	-.4129621E-01
140.	.1017475E+01	.3329480E+01	.5263079E+00	.5970871E-01
141.	.1036303E+01	.4983426E+01	.8223545E+00	.1895811E+00
142.	.1000361E+01	.5855302E+01	.8600848E+00	-.1309692E+00
143.	.1057958E+01	.4651132E+01	.8275352E+00	.3907790E+00
144.	.1001754E+01	.6535971E+01	.9645268E+00	-.1294009E+00
145.	.9823618E+00	.3885064E+01	.5284195E+00	-.2413188E+00
146.	.9977456E+00	.4982905E+01	.7247801E+00	-.1380285E+00
147.	.1050359E+01	.5355812E+01	.9169059E+00	.3084779E+00
148.	.1027229E+01	.3493279E+01	.5735072E+00	.1401254E+00
149.	.1045544E+01	.3450289E+01	.6104254E+00	.3031537E+00
150.	.1024903E+01	.6298077E+01	.9920111E+00	.6870057E-01
151.	.1025135E+01	.5972935E+01	.9433704E+00	.7557393E-01
152.	.1029647E+01	.4012742E+01	.6578698E+00	.1500964E+00
153.	.1010436E+01	.6050450E+01	.9155934E+00	-.5043828E-01
154.	.1025894E+01	.5068645E+01	.8084213E+00	.9716153E-01
155.	.9913858E+00	.3581222E+01	.5045209E+00	-.1627183E+00
156.	.1044782E+01	.4812428E+01	.8181372E+00	.2681311E+00
157.	.1033511E+01	.5392993E+01	.8775901E+00	.1577950E+00
158.	.1018389E+01	.4412035E+01	.6906239E+00	.4495545E-01
159.	.1007198E+01	.5812269E+01	.8715321E+00	-.7382089E-01
160.	.1005837E+01	.4910416E+01	.7340549E+00	-.7004346E-01

APÉNDICE E

PROGRAMA DE GENERACIÓN DE CONDICIONES INICIALES PARA EL ENTRENAMIENTO DEL SISTEMA FALCON-ART

E. 1 Introducción.

El estudio de flujos de carga en un sistema eléctrico permite determinar las condiciones de operación en las que se encuentra dicho sistema, a partir del conocimiento de los voltajes en los nodos. A partir de estos voltajes se pueden determinar otras cantidades como flujos en las líneas, corrientes y pérdidas.

En un estudio de flujos de carga existen tres tipos de nodos:

- Nodo flotante o compensador.
- Nodo de generación.
- Nodo de carga.

Nodo flotante o compensador. Es un nodo en el que se especifica la magnitud del voltaje (V) y su ángulo de fase (δ) y se desconocen la potencia activa (P) y reactiva (Q).

Nodo de generación. En el nodo de generación se especifica la magnitud del voltaje de operación $|V|$ y la potencia activa P ya que éstas cantidades son fisicamente controlables. Se desconocen la potencia reactiva Q y el ángulo δ.

Nodo de carga. El nodo de carga es aquel en el cual hay demanda de energía y en el que se conocen las potencias activa (P) y reactiva (Q) y se desconocen la magnitud del voltaje $|V|$ y el ángulo de fase δ del nodo que se trate. En suma se tiene:

NODO o BUS	Cantidades conocidas	Cantidades desconocidas
Flotante	V y δ	P y Q
Generación	V y P	Q y δ
Carga	P y Q	V y δ

Se tiene, por lo tanto, 4 variables por nodo, dos de las cuales son conocidas, es decir, que en un sistema de N nodos se tienen 2N ecuaciones.

Para el cálculo de los voltajes en los diferentes nodos se debe tomar un nodo de referencia que usualmente es el neutro del sistema al que se fija un valor de cero. Para los ángulos de voltaje la referencia es el ángulo del nodo compensador que se toma como cero (δ=0).

Debido a que no se tiene un número suficiente de ecuaciones en base de nodos o en base de mallas que permita definir el sistema, se tiene un problema de carácter no lineal. Para resolver el problema se recurre a la adición de ecuaciones que tengan cantidades conocidas como P y Q.

E. 2 Desarrollo de ecuaciones para un estudio de flujos de carga

A continuación se desarrolla un ejemplo generalizado para mayor comprensión del problema de flujos de potencia. Se tiene el diagrama de un sistema de 2 nodos siguiente:

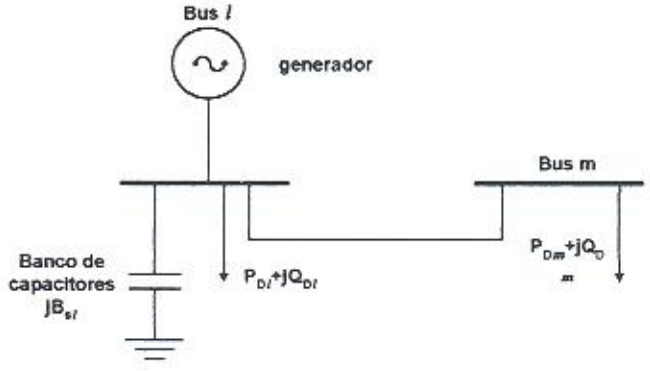


Fig. E.1 Sistema generalizado de flujo de carga

Se tiene un generador y una carga en el bus *l*, y que se encuentra conectada a un bus *m* por medio de una línea de transmisión.

S_{Gl} = potencia compleja en el bus l (generación)

S_{Dl} = potencia compleja para la carga en el bus l

$S_l = S_{Gl} - S_{Dl} = (P_{Gl} - P_{Dl}) + j(Q_{Gl} - Q_{Dl})$

V_l = voltaje complejo en el bus $l = V_l e^{j\theta_l}$

I_l = corriente neta injectada al nodo l

= corriente generada – corriente de carga

B_{Sl} = susceptancia en paralelo en el bus l

Dadas las definiciones anteriores, se expresa a continuación la potencia compleja S_l en términos de los voltajes complejos de bus:

$$
\begin{aligned}
S_l &= V_l I_l^* \\
&= V_l[V_l(jB_{Sl} + G_{Sl} + jB_{Slm}) + (V_l - V_m)(G_{lm} + jB_{lm})]^* \\
&= V_l e^{j\delta_l}\{V_l e^{-j\delta_l}[G_{Slm} - j(B_{Sl} + B_{Slm})] + (V_l e^{-j\delta_l} - V_l e^{-j\delta_m})(G_{lm} - jB_{lm})\} \\
&= V_l^2(G_{Slm} + G_{lm}) - V_l V_m[G_{lm}\cos(\delta_l - \delta_m) + B_{lm}\sin(\delta_l - \delta_m)] \\
&\quad + j[-V_l^2(B_{Sl} + B_{Slm} + B_{lm}) - V_l V_m(G_{lm}\sin(\delta_l - \delta_m) - B_{lm}\cos(\delta_l - \delta_m)]
\end{aligned}
$$

E. 3 Programa computacional de flujo de carga para dos nodos (generacion y carga)

```
1.  Program Generador_de_Cond_Inic

2.  ! Definición de Variables
3.  ! parametros

4.  real, parameter  :: PaConductanciaPropiaDeLinea =  1.739
5.  real, parameter  :: PaSusceptanciaPropiaDeLinea = -6.9327
6.  real, parameter  :: PaConductanciaMutuaDeLinea  = -1.739
7.  real, parameter  :: PaSusceptanciaMutuaDeLinea  =  6.9327
8.  real, parameter  :: PaVoltNodoCompensador     = 1.0
9.  real, parameter  :: PaAlfaNodoCompensador = 0.0
10. real, parameter  :: PaEpsilon = 0.001

11. ! VARIABLES

12. real*8 ReVoltNodoGenerador
        !Voltage del nodo generador
13. real*8 ReAlfaNodoGenerador
        !Angulo del voltaje del nodo generador
14. real*8 RePotenciaReactivaGenerada
        !potencia reactiva generada
15. real*8 RePotenciaConRespectoAlfaNodoUno
        !Derivada de la potencia en el nodo
16. !generador con respecto al angulo del mismo nodo.
17. real*8 ReIncrementoDePotencia
        !Incremento en la potencia del nodo generador

18. real*8 RePotenciaNodoUno                    !Potencia en
    el Nodo Uno
19. real*8 RePotenciaDeCarga                    !Potencia de
    Carga          en el nodo Generador
20. real*8 RePotenciaRealGenerada               !potencia Real
    a generar en el nodo generador
21. real*8 ReIncrementoAlfaUno
    !Incremento del angulo del nodo generador



22. open(7, file = ' resultados2.DAT ' )

23. ! se inicia un ciclo en el cual deseamos generar un cierto
    número de
24. ! condiciones iniciales.

25. RePotenciaDeCarga = 0.0

26. do k = 1, 1

27. !Condiciones iniciales para el angulo del nodo generador, y para
    la
28. !Potencia reactiva generada.

29. ReIncrementoDePotencia = 1.0
30. ReAlfaNodoGenerador = 0.0
31. RePotenciaReactivaGenerada = 0.0

32. call
        condiciones_aleatorias(ReVoltNodoGenerador,RePotenciaNodoUno
    )

33.     ReVoltNodoGenerador = 1.0
34.     RePotenciaNodoUno = 1.0

35. ReA = ReVoltNodoGenerador * ReVoltNodoGenerador
36. ReB = PaVoltNodoCompensador * ReVoltNodoGenerador

37.     print*, ReVoltNodoGenerador,RePotenciaNodoUno

38. n = 0

39. do while ( abs(ReIncrementoDePotencia) > PaEpsilon )

40. RePotenciaReactivaGenerada = ReA * ( PaConductanciaPropiaDeLinea
    * &
41. & sin (         ReAlfaNodoGenerador - ReAlfaNodoGenerador          )
        - &
42. & PaSusceptanciaPropiaDeLinea * cos ( ReAlfaNodoGenerador &
43. &   - ReAlfaNodoGenerador )) + &
44. &   ReB * ( PaConductanciaMutuaDeLinea * sin (
    ReAlfaNodoGenerador &
45. & - PaAlfaNodoCompensador ) - PaSusceptanciaMutuaDeLinea * &
46. & cos (         ReAlfaNodoGenerador - PaAlfaNodoCompensador       )

47.             print*, RePotenciaReactivaGenerada

48. RePotenciaRealGenerada = ReA * (  PaConductanciaPropiaDeLinea +
    &
49. & cos (         ReAlfaNodoGenerador - ReAlfaNodoGenerador          )
        + &
```

```
50.  & PaSusceptanciaPropiaDeLinea * sin ( ReAlfaNodoGenerador &
51.  & - ReAlfaNodoGenerador )          )        + &
52.  & ReB * ( PaConductanciaMutuaDeLinea * cos (
     ReAlfaNodoGenerador &
53.  & - PaAlfaNodoCompensador ) + PaSusceptanciaMutuaDeLinea *
     &
54.  & sin (        ReAlfaNodoGenerador - PaAlfaNodoCompensador
          )          )

55.  !              print*, RePotenciaRealGenerada

56.  ReIncrementoDePotencia = RePotenciaNodoUno -
     RePotenciaDeCarga - RePotenciaRealGenerada

57.  !              print*, ReIncrementoDePotencia

58.  RePotenciaConRespectoAlfaNodoUno = -
     RePotenciaReactivaGenerada - &
59.  & ( PaSusceptanciaPropiaDeLinea *  (ReVoltNodoGenerador**2)
     )

60.  !              print*, RePotenciaConRespectoAlfaNodoUno

61.  ReIncrementoAlfaUno =  ReIncrementoDePotencia /
     RePotenciaConRespectoAlfaNodoUno

62.  !              print*, ReIncrementoAlfaUno

63.  ReAlfaNodoGenerador      =  ReAlfaNodoGenerador +
     ReIncrementoAlfaUno

64.  !              print*,   ReAlfaNodoGenerador

65.  !              pause

66.  RePotenciaReactivaGenerada = ReA * ( PaConductanciaPropiaDeLinea
     * &
67.  & sin (        ReAlfaNodoGenerador - ReAlfaNodoGenerador        )
          - &
68.  & PaSusceptanciaPropiaDeLinea * cos ( ReAlfaNodoGenerador &
69.  & -   ReAlfaNodoGenerador )) + &
70.  &    ReB * ( PaConductanciaMutuaDeLinea * sin (
     ReAlfaNodoGenerador &
71.  & - PaAlfaNodoCompensador ) - PaSusceptanciaMutuaDeLinea * &
72.  & cos (        ReAlfaNodoGenerador - PaAlfaNodoCompensador       )
          )


73.  n = n + 1

74.  enddo

75.  ReAlfaNodoGenerador =  (ReAlfaNodoGenerador * 180.0)/3.14159265

76.  write(7,5)ReVoltNodoGenerador,ReAlfaNodoGenerador,RePotenciaNodo
     Uno,RePotenciaReactivaGenerada
77.  5 format(4E18.7)

78.  enddo

79.  Contains

80.  Subroutine Condiciones_Aleatorias(volt_term,potencia)

81.  real*8,intent(out)  :: volt_term,potencia
82.  real*8 r(2)

83.  do i = 1, 10
84.  call random_number(r)
85.  enddo

86.  volt_term = 0.98 + ( r(1) * 0.08)
87.  potencia  = 0.5 + ( r(2) * 0.5 )

88.  end subroutine


89.  End Program
```

APÉNDICE F

FUNDAMENTOS DE REDES NEURONALES ARTIFICIALES

F. 1 Introducción.

Por medio de la computadora los científicos han obtenido logros importantes y rebasado expectativas en diferentes áreas de investigación, por ejemplo: Aeronáutica, Medicina, Climatología, Comunicaciones, etc., y muchos otros avances que han permitido que los seres humanos tengamos hoy en día una vida más cómoda y placentera. Sin embargo, a pesar de que la computadora desempeña funciones importantes en estos avances, los científicos han intentado aún que ha esta máquina se le sea añadida una característica más importante, *INTELIGENCIA*.

Esta cualidad es característica sólo en los seres humanos y en algunos otros seres vivos (delfines y primates por ejemplo, y aunque en ocasiones, lamentablemente, en el hombre exista una gran ausencia de dicha cualidad). El órgano encargado de proporcionar esta característica es el *cerebro*, que está constituido, de células llamadas neuronas y que realizan sus funciones de una manera muy eficiente para que podamos desenvolvernos en nuestra vida diaria.

Las neuronas forman nuestro cerebro, y es en ellas donde se encuentra almacenado el conocimiento adquirido en nuestra vida; existe también otra parte muy importante del pensamiento humano y que se llama *RACIOCINIO*, y que es la forma en que ordenamos nuestras ideas para formar juicios y conclusiones en cada una de las actividades en que nos encontramos diariamente.

Podemos percibir que nuestras neuronas realizan la función de recipientes de información y además nos permiten "pensar", tales actividades han intentado ser emuladas por las RNA que han sido diseñadas por los científicos para su aplicación en diferentes áreas de investigación, en este apéndice se presentan brevemente

conceptos que pueden ser útiles para comprender las RNA, y que ayudan a entender las operaciones que se presentan en este trabajo.

F. 2 Definición.

Las Redes neuronales artificiales son sistemas que emulan el funcionamiento del cerebro, y aunque su semejanza con este órgano es limitada, su aplicación se lleva a cabo en muchas áreas de investigación. Algunas de las definiciones para estos sistemas son:

Definición Uno. "Una red neuronal es un procesador distribuido en paralelo, hecho de unidades simples de procesamiento, el cual tiene una propensión natural de almacenar conocimiento y volverlo útil en aplicaciones, se parece al cerebro humano en dos aspectos: (a)El conocimiento es adquirido por las neuronas del medio ambiente por medio de entrenamiento y, (b) Los pesos de interconexión, conocidos como pesos sinápticos, son utilizados para almacenar el conocimiento."[41]

Definición dos. "Una red neuronal artificial es un sistema que está construido deliberadamente para hacer uso de algunos principios organizacionales que se encuentran en el cerebro".[42]

Los sistemas neuronales artificiales están formados por elementos de procesamiento conectados en paralelo, esta característica contribuye a su buen desempeño en reconocimiento de imágenes, controles, identificadores de voz, planeación de la producción, etc.

F. 3 Estructura básica de un nodo o elemento de procesamiento.

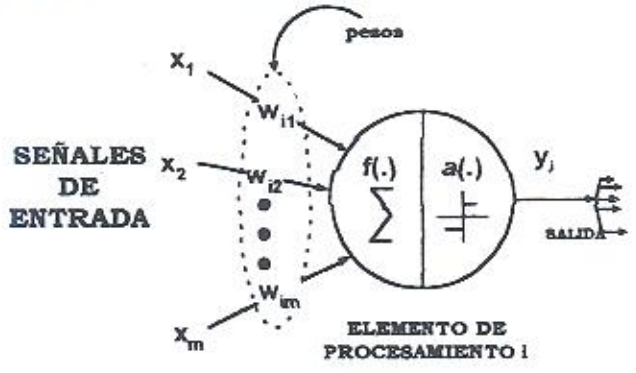


Figura F. 1 Modelo de una neurona artificial.

[41] Ref. 11, pág. 2
[42] Ref. 12, pág. 230

Un sistema neuronal artificial está conformado por elementos simples de procesamiento llamados nodos, unidades o elementos de procesamiento, estas "unidades de procesamiento...son fundamentales para la red neuronal"[43] y su estructura básica se presenta en la Fig. F.1.

Donde f(.) se define comúnmente como la neta de la neurona y que "...es usualmente una función lineal de las entradas x_i al elemento de procesamiento..."[44]

Esta neta puede caracterizarse por la función:

$$fi \equiv neta_i = \sum_{j=1}^{m} w_{ij} x_j - \theta_i$$

donde θ_i es la función escalón del i-ésimo elemento de procesamiento. Existen otras funciones más complejas como: "...funciones cuadráticas, funciones esféricas y funciones polinómicas..." [45].

F. 4 Funciones de activación.

El elemento a(.) de la Fig. F.1 se define como la "...función de activación o transferencia..."[46] y puede existir en diferentes modalidades.

- Función escalón $$a(f) = \begin{cases} 1 & \text{si } f \geq 0 \\ 0 & \text{de otra forma} \end{cases}$$
- Función Umbral $$a(f) = \begin{cases} 1 & \text{si } f \geq 0 \\ -1 & \text{si } f < 0 \end{cases}$$
- Función rampa $$a(f) = \begin{cases} 1 & \text{si } f \geq 1 \\ f & \text{si } 0 \leq f \leq 1 \\ 0 & \text{si } f < 0 \end{cases}$$
- Función sigmoidal unipolar $$a(f) = \frac{1}{1+e^{-\lambda f}}$$
- Función sigmoidal bipolar $$a(f) = \frac{2}{1+e^{-\lambda f}} - 1$$

En la Fig. F.2 se grafican cada una de estas funciones de activación.

[43] Ref. 11, pág. 10
[44] Ref. 11, pág. 232
[45] Ref. 12, pág. 208
[46] Ref. 12, pág. 210

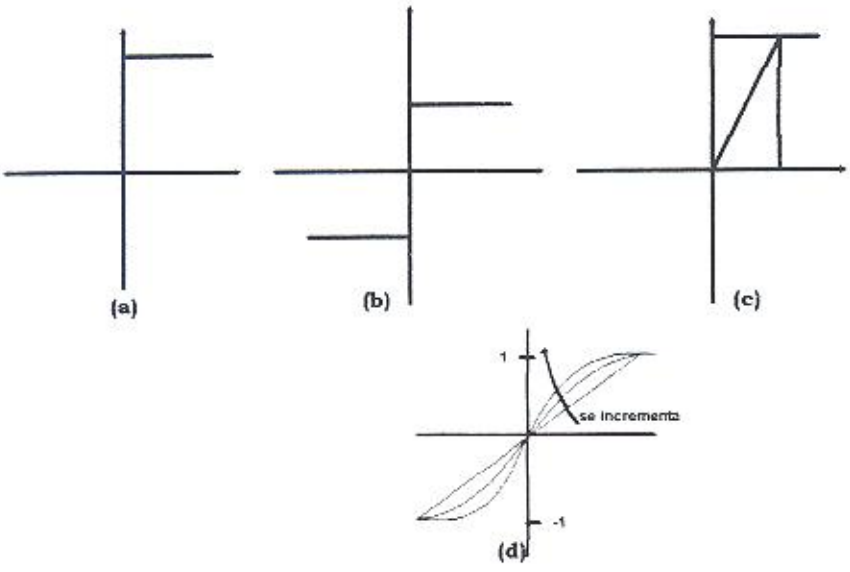


Figura F. 2 Funciones de activación. (a) Función escalón; (b)Función umbral; (c) función rampa; (d) función sigmoidal bipolar.

F. 5Aprendizaje en los sistemas de redes neuronales.

El aprendizaje que adquieren estos sistemas está regido por reglas que son "...un elemento importante de las redes neuronales artificiales..."[47]. Las arquitecturas de redes neuronales artificiales requieren de aprendizaje en dos aspectos:

- El aprendizaje de parámetros, y
- El aprendizaje de la estructura de la red.

El primero se refiere a la actualización de los pesos y el segundo se enfoca en los cambios de la estructura de la red, incluyendo el número de elementos de procesamiento y las conexiones. Estos dos tipos de aprendizaje pueden llevarse a cabo simultáneamente o por separado.

F.5. 1 Aprendizaje de parámetros.

Se pueden clasificar principalmente los siguientes aprendizajes:

- Aprendizaje supervisado.
- Aprendizaje reforzado.
- Aprendizaje no supervisado.

[47] Ref. 12, pág. 212

Aprendizaje supervisado. En cada entrada de la red, se da la salida deseada correspondiente. Es decir, existe un conjunto de pares que se alimentan a la red en cada instante de tiempo.

Aprendizaje reforzado. Este es un tipo de aprendizaje supervisado como se ve en la Fig. F.3, porque recibe la señal de retroalimentación exterior. Pero esta señal, es "... una señal de análisis más que instructora...También puede llamarse aprendizaje con un crítico...". [48]

Aprendizaje no supervisado. En este aprendizaje no existe retroalimentación exterior que dirija al sistema por la senda correcta. Este tipo de sistemas se llama auto-organizables.

En la Fig. F.3 se muestran los diagramas de bloques para cada uno de estos aprendizajes.

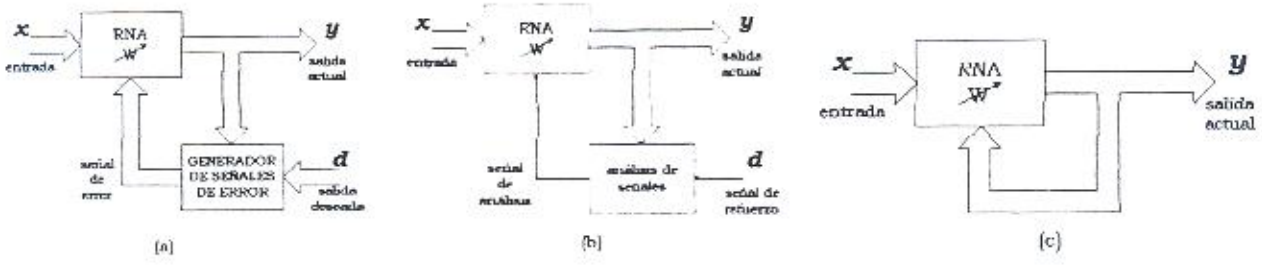


Figura F. 3 (a) Aprendizaje Supervisado; (b) Aprendizaje reforzado; (c) Aprendizaje no supervisado.

F.5. 2 Aprendizaje estructural.

Existen modelos de sistemas de redes neuronales que "...necesitan del ensayo y error para determinar el No. de elementos de procesamiento que se necesitan en el sistema..."[49], y existen modelos como el que se presenta en este trabajo que pueden auto-organizarse. La aplicación de algoritmos genéticos pueden "...contribuir también con resultados satisfactorios para el aprendizaje de la estructura del sistema".[50]

[48] Ref. 12, pág. 214
[49] Ref. 13, pág. 91
[50] Ref. 12, Cap. 15

F. 6 Antecedentes Históricos[51].

El trabajo inicial fue realizado por McCuloch y Pitts[1943]. McCulloch era un neuroanatomista y psiquiatra y Pitts era un prodigio matemático. Su estudio clásico de las neuronas todo-o-nada, determinó la lógica de cálculo de las redes neuronales.

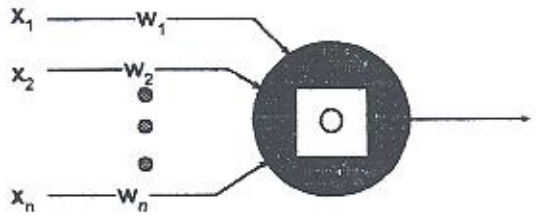


Figura F. 4 Modelo de McCulloch-Pitts

La Fig. F.4 muestra un modelo McCuloch-Pitts con entradas x_i, para i= 1,2,...3, W_i denota los pesos multiplicativos (restricciones sinápticas) que conectan la entrada i-ésima a la neurona. ***O*** es la función escalón, el cual debe ser excedido por la sumatoria de entradas por los pesos correspondientes para activar la neurona. El peso W_i, es positivo sí la conexión (sinapsis) es positiva y negativo si la conexión es inhibitoria. Las x_i son binarias (0 ó 1) y pueden provenir directamente de sensores o de otras neuronas.

La relación siguiente define la regla de activación para la neurona:

$$O = g(\sum_{i=1}^{N} W_i x_i)$$

Donde g(x) es la función de activación definida como:

$$g(x) = \begin{cases} 1 & \text{si} \quad x \geq \theta \\ 0 & \text{si} \quad x < \theta \end{cases}$$

Este modelo tan simple pudo demostrar un poder computacional potencial, a partir de la selección apropiada de los pesos podía realizar operaciones lógicas como AND, OR, NOT, etc. La Fig. F.5 muestra como se seleccionan los pesos para llevar a cabo estas operaciones:

[51] Ref. 8, Cap. 1

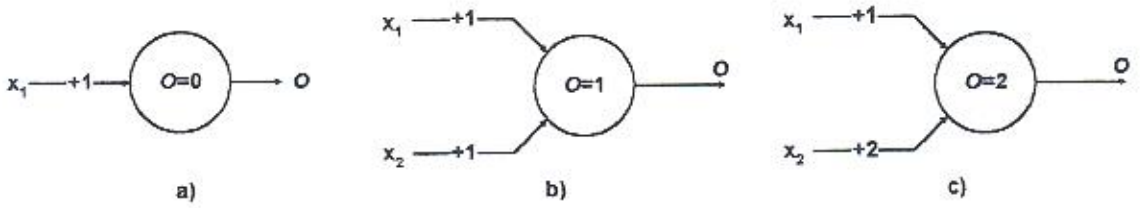


Figura F. 5 a) implementación de una compuerta NOT; b) implementación de una compuerta OR; c) implementación de una compuerta AND.

Wiener[1948] Discutió problemas importantes de control, comunicación y procesamiento de señales basado en su concepción de similitud entre las computadoras y el cerebro.

Hoppfield[1982,1984] Estableció la relación real entre la mecánica estadistica y los ensamblajes neuronales.

Von Neumann[1958] Sugirió en utilizar el lenguaje cerebral para diseñar máquinas con procesamiento semejante al cerebro.

Hebb[1949] Propuso un esquema de aprendizaje mediante la actualización de los pesos sinápticos existentes entre las neuronas. También fue el primero en proponer que el aprendizaje neuronal es debido a estos cambios. Su famoso *postulado de aprendizaje*, al cual hoy se hace referencia como *Regla de aprendizaje Hebbian*, estableció que la información puede ser almacenada en las conexiones sinápticas, y la fuerza de éstas conexiones se incrementa con la activación repetitiva de una neurona a través de las otras neuronas que se encuentran conectadas a la sinápsis.

Rochester et al.[1956] Realizaron simulaciones computacionales para probar la teoría de Hebb de aprendizaje en un ensamble de neuronas artificiales. Demostraron que era esencial adicionar inhibiciones a la teoría para que el ensamble realmente trabajara.

Frank Rosenblatt[1958] Propuso un elemento llamado Perceptrón. Su intención, como menciona en su libro, era [Rosenblatt 1958, p. 387]:

"... Ilustrar algunas de las propiedades fundamentales de los sistemas inteligentes, sin profundizar en las a veces desconocidas, condiciones de funcionamiento que son intrínsecas de los sistemas biológicos."

Widrow y Hoff[1960, 1962] propusieron un mecanismo de aprendizaje donde el error cuadrado sumarizado en la salida de la red fuera minimizado. Ellos introdujeron su diseño ADALINE (Adaptative linear combiner) basado en esta poderosa regla de aprendizaje (también llamada Regla de aprendizaje Widrow-Hoff).

Minsky y Pappert[1969] Utilizaron matemáticas muy detalladas para demostrar las limitaciones computacionales de una sola capa de perceptrones. Los autores hacían conjeturas en su libro (que más tarde fueron injustificadas) a partir de las limitaciones encontradas para una sola capa de perceptrones, y que sugerían podrían también cumplirse para las variantes de este modelo o específicamente para una red multicapa.

Nilson[1965] Demostró que una multicapa de perceptrones podía ser utilizada para separar patrones no lineales (como el caso del problema XOR) en un hiperespacio.

Paul Werbos[1974] Propuso un nuevo esquema matemático de entrenamiento para redes multicapas de perceptrones. Amari[1972, 1977] Desarrolló un modelo adaptable de una neurona con función escalón, y lo utilizó para estudiar el comportamiento de redes aleatorias. Kohonen[1972,1977,1980] Desarrolló su trabajo sobre memorias asociativas. Fukushima[1975] Propuso su modelo llamado cognitrón y sus variaciones [1980] llamado neocognitrón para el reconocimeinto de patrones. Grossberg[1967, 1968] Desarrolló teorías y arquitecturas que incluyen un modelo de neurona adaptable y demostró su utilización como una memoria de corto plazo. Posteriormente, desarrolló su filtro adaptable abajo-arriba y su sistema de igualación arriba-abajo. Esto permitió el reconocimiento de patrones con la resonancia adaptable. Carpenter y Grossberg[1987] diseñaron sus redes basadas en este fenómeno, la Teoría de Resonancia Adaptable.

Todo el trabajo en conjunto mencionado anteriormente, contribuyó a derribar las barreras para el entrenamiento de sistemas neuronales, que habían anclado los esfuerzos iniciados a mitad de los años 60's.

APÉNDICE G

GENERALIDADES DE SISTEMAS DIFUSOS[52]

G. 1 Introducción.

La razón principal de utilizar los sistemas difusos es la dificultad existente para el modelado y simulación de sistemas reales muy complejos para el desarrollo de sistemas de control, especialmente cuando se considera el aspecto de la implementación, *"lo anterior está plenamente comprobado"*[53]. Aún cuando se pueda desarrollar un modelo relativamente exacto de un sistema dinámico, es muy frecuente la dificultad para desarrollar un controlador, especialmente para el diseño de procedimientos de control convencional en donde se requiere suposiciones restrictivas de la planta.

Los sistemas difusos han sido usados en una gran variedad de aplicaciones de ingeniería, ciencia, negocios, medicina, psicología y otros campos. Dentro de la ingeniería podemos encontrar campos como:

- Navegación aérea y espacial: Controles de vuelo, control de motores, detección de fallas, navegación y control de altitud de satélites.
- Automóviles: Frenos, transmisión, suspensión y control de inyección de combustible.
- Vehículos Autónomos: Por tierra y por aire.
- Sistemas de manufactura: Planeación y control de inventarios.

---

[52] Este capítulo esta tomado en su totalidad de la Ref. 10

[53] Ref. 10, pág. 9

- Control de procesos: Temperatura, presión y control de nivel. Diagnóstico de fallas, destilación, control de columnas y procesos de desalinización.

G. 2 Variables, valores y reglas difusos.

G.2. 1Variables Lingüísticas.

Un sistema difuso en un mapeo estático no lineal (i.e.: no es un sistema dinámico). En un sistema difuso se tienen entradas $u_i \in U_i$ donde i.=1,2,...,n y salidas $y_i \in Y_i$ donde i.=1,2,...,m difusas como en la Fig. G.1.

Las entradas y las salidas son números reales y no conjuntos difusos. El bloque de difusificación convierte los números reales en entradas difusas, el mecanismo de inferencia utiliza la base de reglas para generar conclusiones difusas y el bloque de dedifusificación convierte las conclusiones difusas en salidas reales. Los espacios reales U_i y Y_i en donde varían u_i y y_i, se llaman universos de discurso.

G.2. 2 Valores Lingüísticos.

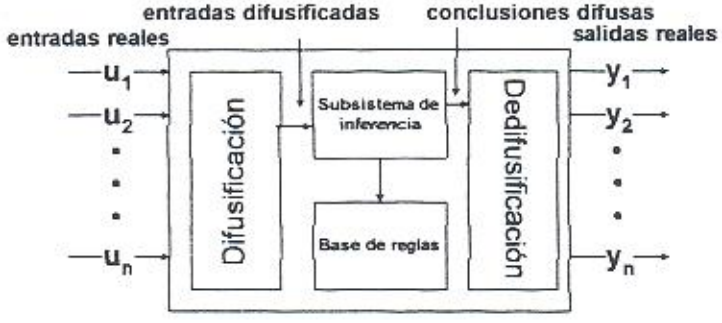


Fig. G.1 Sistema Difuso.

Para lograr especificar las reglas dentro de la *base de reglas* es necesario utilizar una descripción lingüística que permita identificar a las entradas y las salidas (así como sus características) al sistema difuso. Las entradas y salidas lingüísticas u_i y y_i, respectivarnente, pueden tener a su vez valores lingüísticos sobre los mismos universos de discurso y que pueden representarse por: $\tilde{A}_i^j$ que representa el *j*-ésimo valor de la variable lingüística u_i y $\tilde{B}_i^j$ que representa el *j*-

ésimo valor de la variable lingüística $\tilde{y}_i$. Los grupos de estos valores lingüísticos pueden representarse como:

$$\tilde{A}_i = \left\{ \tilde{A}_i^j : j = 1,2,\ldots, N_i \right\} \quad \text{y} \quad \tilde{B}_i = \left\{ \tilde{B}_i^p : p = 1,2,\ldots, M_i \right\}$$

Los valores lingüísticos pueden tener términos descriptivos como: "positivo grande", "cero", ó "negativo grande" (que son adjetivos). Por ejemplo, se puede tener una variable lingüística $\tilde{u}_1$="velocidad" y asignarle los siguientes valores lingüísticos $\tilde{A}_1^1$= "lenta", $\tilde{A}_1^2$= "media" y $\tilde{A}_1^3$= "rápida"; y que pueden formar el conjunto $\tilde{A}_1 = \left\{ \tilde{A}_1^1, \tilde{A}_1^2, \tilde{A}_1^3 \right\}$.

G.2. 3 Reglas Lingüísticas.

El mapeo de las entradas hacia las salidas en un sistema difuso se caracteriza en parte por un conjunto de reglas condición→acción, o en su forma *modus ponens* (Si-Entonces):

Si premisa **entonces** consecuencia

Las entradas al sistema difuso son asociadas frecuentemente con las premisas, y las salidas con las consecuencias. Estas reglas pueden ser representadas de diversas formas, dos de las formas estandar son: MIMO y MISO (multi-input multi-output, multi-inputs single-output por sus siglas en inglés, respectivamente). Una regla MISO puede ser así:

Si $\tilde{u}_1$ es $\tilde{A}_1^j$ **y** $\tilde{u}_2$ es $\tilde{A}_2^k$ **y** ,..., **y** $\tilde{u}_n$ es $\tilde{A}_n^l$ **entonces** $\tilde{y}_q$ es $\tilde{B}_q^p$

Y pueden conformar una base de reglas de esta misma forma que un experto puede especificar para poder controlar un sistema. Finalmente, si se utilizan todas las premisas en cada regla y una regla está formada por cada posible combinación de elementos premisa, entonces habrá:

$$\prod_{i=1}^{n} N_i = N_1 \cdot N_2 \cdot \ldots \cdot N_n$$

Reglas en la *base de reglas*. Por ejemplo, si n = 2 entonces se tiene N_i=11 funciones de membresía en cada universo de discurso, podría haber entonces

11*11=121 reglas posibles. En este caso, el número de reglas se incrementa exponencialmente con incremento en el número de entradas al sistema difuso.

G. 3 Conjuntos difusos, lógica difusa y la base de reglas.

Los conjuntos difusos y la Lógica Difusa son utilizados para cuantificar heurísticamente el significado de las variables, valores y reglas lingüísticos que son especificados por un experto. El concepto de conjunto difuso mediante la definición de *función de membresía.*

G.3. 1Funciones de membresía.

U_i representa un universo de discurso y $\tilde{A}_i^j \in \tilde{A}_i$ representa un valor específico lingüístico para una determinada variable lingüística $\tilde{u}_i$, la función $\mu(u_i)$ que se asocia con $\tilde{A}_i^j$ que mapea U_i hacia [0,1] se conoce como *función de membresía.* Esta función de membresía describe la certidumbre que un elemento de U_i representado por u_i, con una descripción lingüística, podría ser clasificada lingüísticamente como $\tilde{A}_i^j$. Las funciones de membresía se especifican subjetivamente de una manera heurística según la experiencia o la intuición.

Existen muchas formas de funciones de membresía (trapezoidales, rectangulares, etc.) y cada una de ellas puede tener diferente significado para las variables lingüísticas que representan.

G.3. 2Conjuntos difusos.

Si se tiene una variable lingüística $\tilde{u}_i$ con un valor lingüístico $\tilde{A}_i^j$ definido sobre el universo de discurso U_i, y con una función de membresía $\mu_{\tilde{A}_i^j}(u_i)$ (que es la función de membresía que se asocia al conjunto difuso $\tilde{A}_i^j$) que mapea U_i hacia [0,1], un conjunto difuso representado por A_i^j se define como:

$$A_i^j = \left\{(u_i, \mu_{\tilde{A}_i^j}(u_i)) : u_i \in U_i\right\}$$

Un conjunto difuso es un conjunto de pares ordenados de números reales que pertenecen a un universo de discurso, acoplado a sus valores de membresía.

Algunos conceptos adicionales que relacionan las funciones de membresía y conjuntos difusos son los siguientes:

- "*Soporte de un conjunto difuso*": Es el conjunto de puntos que están sobre el universo de discurso donde la función de membresía es mayor a cero.
- "*α - cut*": Es el conjunto de puntos sobre el universo de discurso donde la función de membresía es mayor a un valor α.
- *"Altura" de un conjunto difuso*: Es el punto más algo alcanzado por la función de membresía.
- *"Conjuntos difusos normales"*: Conjuntos con función de membresía que alcanzan un punto o al menos uno sobre el universo de discurso.
- *"Principio de extensión"*: Si se es dada una función que mapea algún dominio en algún rango y se tienen funciones de membresía sobre el dominio, el principio de extensión muestra como mapear las funciones de membresía del dominio hacia el rango.

G.3. 3Operaciones en la lógica difusa.

Subconjunto difuso: Dados los conjuntos difusos A_i^1 y A_i^2 asociados en el universo de discurso U_i (N_i=2), con sus funciones de membresía representadas por $\mu_{A_i^1}(u_i)$, y $\mu_{A_i^2}(u_i)$ respectivamente. Entonces A_i^1 es definida como un *subconjunto difuso* de A_i^2 representado por $A_i^1 \subset A_i^2$, si $\mu_{A_i^1}(u_i) \leq \mu_{A_i^2}(u_i)$ para toda $u_i \in U_i$.

Complemento difuso: El complemento ("no") de un conjunto difuso A_i^1 con una función de membresía $\mu_{A_i^1}(u_i)$ tiene una función de membresía dada por 1- $\mu_{A_i^1}(u_i)$.

Intersección Difusa ("y"): La intersección de los conjuntos difusos A_i^1 y A_i^2, que se encuentran definidos sobre el universo de discurso U_i, es un conjunto representado por $A_i^1 \cap A_i^2$, con una función de membresía que puede definirse por medio de los dos métodos más comunes siguientes:

1. *Mínimo*: Podemos encontrar el mínimo de los valores de membresía mediante: $\mu_{A_i^1 \cap A_i^2} = min\{\mu_{A_i^1}(u_i), \mu_{A_i^2}(u_i) : u_i \in U_i\}$
2. *Producto Algebraico*. En este podemos encontrar el producto de los valores de membresía con: $\mu_{A_i^1 \cap A_i^2} = \{\mu_{A_i^1}(u_i), \mu_{A_i^2}(u_i) : u_i \in U_i\}$

Unión Difusa("or"): La unión de dos conjuntos difusos A_i^1 y A_i^2, que se encuentran definidos sobre el universo de discurso U_i, es un conjunto representado

por $A_i^1 \cup A_i^2$, con una función de membresía que puede definirse por medio de los dos métodos más comunes siguientes:

1. *Máximo*: Se pueden encontrar el valor máximo de los valores de membresía mediante: $\mu_{A_i^1 \cup A_i^2}(u_i) = max\{\mu_{A_i^1}(u_i), \mu_{A_i^2}(u_i) : u_i \in U_i\}$
2. *Suma Algebraica*: Se puede encontrar la suma algebraica de los valores de membresía con: $\mu_{A_i^1 \cap A_i^2}(u_i) = \{\mu_{A_i^1}(u_i) + \mu_{A_i^2}(u_i) - \mu_{A_i^1}(u_i)\mu_{A_i^2}(u_i) : u_i \in U_i\}$

Producto cartesiano. La operación intersección y unión se llevan a cabo en los mismos universos de discurso donde se encuentran los conjuntos difusos. Si existen los conjuntos difusos $A_1^j, A_2^k, ..., A_n^l$ sobre los universos $U_1, U_2, ..., U_n$, respectivamente. El producto cartesiano puede denominarse una relación difusa caracterizada por:

$$A_i^j \times A_2^k \times ... \times A_n^l$$

Con una función de membresía:

$$\mu_{A_1^j \times A_2^k \times ... \times A_n^l}(u_1, u_2, ..., u_n) = \mu_{A_1^j}(u_1) * \mu_{A_2^k}(u_2) * ... * \mu_{A_n^l}(u_n)$$

G. 4 Difusificación.

Este proceso consiste en especificar las entradas $u_i \in U_i$ dentro de un conjunto difuso. Suponiendo que U_1^* caracteriza todos los conjuntos difusos posibles sobre U_1, entonces una entrada $u_i \in U_i$ puede representarse mediante la difusificación en un conjunto difuso A_i^{fuz} que se encuentre en el universo del discurso U_1^*.

Esta transformación se produce mediante un operador definido como:

$$F : U_i \rightarrow U_1^*$$

Donde:

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

G. 5 Mecanismo de inferencia.

Este mecanismo cumple con dos tareas:

Determinar la regla más relevante en la presentación de las entradas $u_i, i = 1,2,...n$ .A este paso se le puede llamar "igualación".

Realizar conclusiones usando las entradas u_i y la información en la base de reglas. Este paso se puede denominar también "paso de inferencia".

G.5. 1Igualación.

Si se tienen entradas $u_i, i = 1,2,...n$, y su difusificación produce $A_1^{fuz}, A_2^{fuz},..., A_n^{fuz}$ que son los conjuntos que representan a cada una de las entradas, existen dos pasos básicos para llevar a cabo una "igualación":

1.- Combinación de las entradas con las reglas difusas.

Este paso involucra encontrar conjuntos difusos $A_1^j, A_2^k,..., A_n^l$ que sus funciones de membresía son:

$$\mu_{A_1^j}(u_1) = \mu_{A_1^j}(u_1) * \mu_{A_1^{fuz}}(u_1)$$

$$\mu_{A_2^k}(u_1) = \mu_{A_2^k}(u_2) * \mu_{A_2^{fuz}}(u_2)$$

$$\vdots$$

$$\mu_{A_n^l}(u_n) = \mu_{A_n^l}(u_n) * \mu_{A_n^{fuz}}(u_n)$$

(Para toda $j,k,...,l$) y que combinan los conjuntos difusos de la difusificación con los conjuntos utilizados en cada uno de los términos en las premisas de las reglas.

2.- Determinar que reglas se activan. En este paso se hallan valores de membresía $\mu_i(u_1,u_2,...u_n)$ para la premisa de la i-ésima regla que representan la certeza que cada premisa de regla es correcta para las entradas dadas.

G.5. 2 Paso de inferencia.

En este paso se determina el conjunto implicado $\hat{B}_q^i$ que corresponde a la i-ésima regla $(j,k,\ldots,l;p,q)_i$ y que su función de membresía es:

$$\mu_{\hat{B}_q^i}(y_q) = \mu_i(u_1,u_2,\ldots,u_n) * \mu_{B_q^p}(y_q)$$

Este conjunto especifica el nivel de certeza que la salida $y_q \in Y_q$ debe tomar, tomando sólo en cuenta la regla i.

G. 6 Dedifuzificación.

Existen varios métodos de dedifuzificación, a continuación se muestran dos de ellos que toman en cuenta el conjunto difuso $\hat{B}_q^i$:

- *Centro de gravedad (COG).*

La salida y_q se escoge utilizando el centro del área y el área de cada uno de los conjuntos implicados y se da por medio de:

$$y_q = \frac{\sum_{i=1}^{R} b_i^q \int_{Y_q} \mu_{B_q^i}(y_q)dy_q}{\sum_{i=1}^{R} \int_{Y_q} \mu_{B_q^i}(y_q)dy_q}$$

Donde R es el número de reglas, b_i^q es el centro del área de la función de membresía de B_q^i asociado con el conjunto difuso implicado $\hat{B}_q^i$ para la regla i-ésima $(j,k,\ldots,l;p,q)_i$, y

$$\int_{Y_q} \mu_{B_q^i}(y_q)dy_q$$

Denota el área bajo: $\mu_{B_q^i}(y_q)$

- *Centro-promedio.*

La salida se escoge utilizando los centros de cada una de las salidas de las funciones de membresía y la máxima certeza de cada una de las conclusiones representadas con los conjuntos difusos implicados, y se da por medio de:

$$y_q = \frac{\sum_{i=1}^{R} b_i^q \sup_{y_q} \left\{ \mu_{B_q^i}(y_q) \right\}}{\sum_{i=1}^{R} \sup_{y_q} \left\{ \mu_{B_q^i}(y_q) \right\}}$$

Donde "sup" denota el valor máximo del límite más alto. De esta forma $\sup_x \{\mu(x)\}$ es simplemente como el valor más alto de u(x).

También b_i^q es el centro del área de la función de membresía de B_q^i asociado con el conjunto difuso implicado B_q^i para la regla i-ésima *(j,k,...,l;p,q)*.