



**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE INGENIERÍA  
MECÁNICA Y ELÉCTRICA**

**SECCIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN  
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA**

***Identificación de la Estabilidad a Pequeños  
Disturbios de la Máquina Síncrona Bus-Infinito  
por Redes Neuronales***

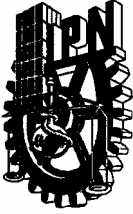
**T E S I S**

QUE PARA OBTENER EL GRADO DE  
MAESTRO EN CIENCIAS CON ESPECIALIDAD  
EN INGENIERÍA ELÉCTRICA  
P R E S E N T A :

*OSCAR MORENO REYES*



México, D. F., Diciembre 2005



# INSTITUTO POLITECNICO NACIONAL

## SECRETARIA DE INVESTIGACION Y POSGRADO

### ACTA DE REVISION DE TESIS

En la Ciudad de México, D.F. siendo las 13:00 horas del día 21 del mes de Noviembre del 2005 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de La ESIME ZAC. para examinar la tesis de grado titulada:

**“ IDENTIFICACIÓN DE LA ESTABILIDAD A PEQUEÑOS DISTURBIOS DE LA MAQUINA SINCRONÍA BUS-INFINITO POR REDES NEURONALES”**

Presentada por el alumno:

**MORENO**

Apellido paterno

**REYES**

materno

**OSCAR**

nombre(s)

Con registro: 

A	0	4	0	5	0	4
---	---	---	---	---	---	---

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. DANIEL RUIZ VEGA

M. EN.C. TOMAS ASIAIN OLIVARES

DR. JAIME ROBLES GARCIA

EL PRESIDENTE DEL COLEGIO

DR. JAIME ROBLES GARCIA



E. P. N.  
SECCION DE ESTUDIOS DE POSGRADO E INVESTIGACION




INSTITUTO POLITECNICO NACIONAL  
SECRETARÍA DE INVESTIGACIÓN DE INVESTIGACIÓN

**CARTA CESION DE DERECHOS**

En la Ciudad de México, Distrito Federal, el día 15 del mes diciembre del año 2005, el que suscribe Oscar Moreno Reyes alumno del Programa de Maestría en Ciencias en Ingeniería Eléctrica con número de registro A040504, adscrito a la Sección de Estudios de Posgrado e Investigación de la ESIME Unidad Zacatenco, manifiesta que es autor intelectual del presente Trabajo de Tesis bajo la dirección del Dr. David Romero Romero y cede los derechos del trabajo intitulado: Identificación de la Estabilidad a Pequeños Disturbios de la Máquina Síncrona Bus-Infinito por Redes Neuronales, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, graficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección: [oscar.moreno@ieee.org](mailto:oscar.moreno@ieee.org), [moreno.oscar@gmail.com](mailto:moreno.oscar@gmail.com).

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

  
Oscar Moreno Reyes

## Resumen

Se presenta una alternativa para analizar la estabilidad del sistema eléctrico de potencia ante disturbios pequeños, el sistema de prueba es el sistema máquina síncrona bus-infinito y su estabilidad se determina utilizando una RN tipo retro-propagación.

Usualmente, la estabilidad a pequeños disturbios se determina linealizando el modelo del sistema alrededor del punto de operación y después se analizan sus eigenvalores, este proceso necesita gran cantidad de recursos computacionales y las condiciones de operación del sistema cambian constantemente, por lo que es necesario determinar la estabilidad de manera eficiente en el menor tiempo posible, con el fin de poder responder oportunamente ante posibles problemas causados por pequeños disturbios, que pudieran inestabilizar al sistema.

En este trabajo la determinación de la estabilidad se realiza como un reconocimiento de patrones, en donde los patrones son los objetos definidos por las condiciones de operación del sistema eléctrico de potencia en un momento establecido. De esta manera la identificación de la estabilidad se puede realizar más fácil y eficientemente que mediante el cálculo y análisis de eigenvalores.

Debido a las características inherentes de las redes neuronales, éstas son aptas para el reconocimiento de patrones, como se ha comprobado a lo largo del tiempo; existen varios tipos de redes neuronales, cada una ha sido desarrollada para resolver problemas determinados y tienen características propias, pero en un balance general, las que ofrecen buena relación desempeño-simplicidad son las del tipo retro-propagación, por lo que se recurre a ellas continuamente y son utilizadas de manera exitosa en diversos campos.

## Abstract

An alternative approach to estimate the power system small-disturbance stability is presented, the tested system is the synchronous-machine infinite-bus power system and its stability is determined by a feed-forward back-propagation neural network.

Usually, the small-disturbance stability is determined by linearizing the system model around an operating point, and then, the eigenvalues are analyzed, this is not an easy task and huge computational resources are required; in addition, there are continuous changes in the system conditions, therefore, it is important to determine the system stability efficiently as fast as possible, in order to give an adequate response before probable problems occur due to small disturbances, which could turn the system into unstable.

In this work the stability analysis is posed as a pattern recognition problem, in which the patterns are the objects described by the operating conditions of the electric power system in a specific moment. Thus, the stability identification by pattern recognition results easier and more efficient than doing it by analyzing the eigenvalues.

Because of the intrinsic characteristics of the artificial neural networks, these are suitable for performing pattern recognition, as it has been proved along the time; there are many varieties of artificial neural networks, each one has been developed for a specific purpose and possesses its own distinctiveness, the feed-forward back-propagation neural network has a good enough simplicity-performance ratio, hence, this kind of ANN is currently very accepted and it is successfully used in many fields.

# Contenido

<b>Resumen</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>Índice de Tablas y Figuras</b>	<b>ix</b>
<b>Nomenclatura</b>	<b>xiii</b>
<hr/>	
<b>1. Introducción</b>	<b>1</b>
1.1. Generalidades	1
1.2. Objetivos	3
1.3. Justificación	3
1.4. Descripción del Problema	3
1.5. Estado del Arte	4
1.6. Aportaciones	4
1.7. Estructura de la Tesis	5
<hr/>	
<b>2. Estabilidad a Pequeños Disturbios en los Sistemas Eléctricos de Potencia</b>	<b>6</b>
2.1. Introducción	6
2.2. Definición y Clasificación de la Estabilidad del Sistema Eléctrico de Potencia	6
2.3. Modelo Linealizado del Sistema Eléctrico de Potencia	8
2.4. Estabilidad en los Sistemas Lineales	10
2.5. Regiones y límite de estabilidad	
<hr/>	
<b>3. Redes Neuronales para Reconocimiento de Patrones</b>	<b>13</b>
3.1. Introducción	13
3.2. Reconocimiento de Patrones	13
3.2.1. Proceso General	14
3.2.2. Diseño de Sistemas	15
3.3. Redes Neuronales	16
3.3.1. Modelo de una Neurona	16
3.3.2. Una Capa de Neuronas	17
3.3.3. Redes Multicapa	19

3.3.4. Función de Activación	20
3.3.5. Diseño	22
3.3.5.1. Pre-procesamiento	23
3.3.5.2. Determinación de la Topología	23
3.3.5.3. Entrenamiento	25
3.3.6. Entrenamiento Supervisado	26
3.3.6.1. Gradiente Descendente	26
3.3.6.2. Variantes del Algoritmo de Aprendizaje	27
3.3.6.3. Retro-propagación	30

---

## **4. Pruebas y Resultados** **31**

---

4.1. Introducción	31
4.2. Descripción del Sistema	32
4.2.1. Límite de estabilidad para línea corta	32
4.2.2. Límite de estabilidad para línea larga	34
4.3. Descripción de la RN	35
4.4. Entrenamiento de la RN con datos obtenidos de la simulación	37
4.4.1. Determinación de la topología	37
4.4.1.1. Línea corta	37
4.4.1.2. Línea larga	41
4.4.1.3. Análisis de resultados	47
4.4.2. Pruebas de identificación	47
4.4.2.1. Línea corta	47
4.4.2.2. Línea larga	53
4.4.2.3. Análisis de resultados	57
4.5. Entrenamiento de la RN con datos contaminados con ruido	58
4.5.1. Contaminación de ruido a la base de datos	58
4.5.2. Entrenamiento con topología determinada con datos de la simulación	59
4.5.2.1. Línea corta	59
4.5.2.2. Línea larga	63
4.5.2.3. Análisis de resultados	66
4.5.3. Entrenamiento con topologías alternativas	66
4.5.3.1. Línea corta – Reduciendo unidades en la capa oculta	67
4.5.3.2. Línea larga	73
4.5.3.2.1. Reduciendo unidades en la capa oculta	73
4.5.3.2.2. Aumentando unidades en la capa oculta	77
4.5.3.3. Análisis de resultados	80

---

## **5. Conclusiones y Recomendaciones para Trabajos Futuros** **81**

---

---



---

**Referencias** **83**


---



---



---



---

**Apéndices** **86**


---



---

A. Parámetros de la Máquina Síncrona	86
B. Corrientes, Voltajes y Ángulo de Carga en el Sistema Máquina Síncrona Bus-Infinito	86
C. Constantes $K_1, K_2, \dots, K_6$ del modelo linealizado	87
D. Algoritmo de Levenberg-Marquardt	88
E. Algoritmo de la Retro-propagación	90
F. Códigos de Programa	95
F. 1 Generación de base de datos para entrenamiento de la RN (FORTRAN 90)	95
F. 2 Determinación de Límites de Estabilidad de la Máquina Síncrona Bus-Infinito a Pequeños Disturbios por Análisis de Eigenvalores (FORTRAN 90)	98
F. 3 Determinación de Límites de Estabilidad de la Máquina Síncrona Bus-Infinito a Pequeños Disturbios por RN (FORTRAN 90)	102
F. 4 Búsqueda de Topología para RN de 3 Capas con n-Unidades en la Capa Oculta y 1 Unidad en la Capa de Salida (MATLAB®)	104
F. 5 Generación de archivo de parámetros de RN (MATLAB®)	106



## Índice de Tablas y Figuras

---

### Tablas

---

2. 1	<i>Parte real de eigenvalores</i>	12
4. 1	<i>Rango de valores de las entradas de la RN</i>	36
4. 2	<i>Parámetros para la búsqueda de la topología (LC)</i>	39
4. 3	<i>Parámetros para la búsqueda de la topología (LL)</i>	41
A. 1	<i>Parámetros de la máquina síncrona</i>	86

---

### Figuras

---

2. 1	<i>Clasificación de la estabilidad en el sistema de potencia.</i>	8
2. 2	<i>Diagrama a bloques de función de transferencia para estudios de oscilaciones de baja frecuencia.</i>	9
2. 3	<i>Límite y regiones de estabilidad</i>	12
3. 1	<i>Etapas en el reconocimiento de patrones.</i>	14
3. 2	<i>Diseño de sistemas de reconocimiento de patrones.</i>	15
3. 3	<i>Una neurona.</i>	16
3. 4	<i>Nomenclatura simplificada para una neurona.</i>	17
3. 5	<i>Una capa de neuronas.</i>	18
3. 6	<i>Nomenclatura simplificada para una capa de neuronas.</i>	18
3. 7	<i>Una RN multicapa.</i>	19
3. 8	<i>Nomenclatura simplificada para una RN multicapa.</i>	19
3. 9	<i>Función rígida.</i>	20
3. 10	<i>Función lineal.</i>	21
3. 11	<i>Función sigmoideal.</i>	21
3. 12	<i>Familia de curvas de la función sigmoideal.</i>	22
3. 13	<i>Metodología de diseño de RN.</i>	22
3. 14	<i>Diagrama de flujo de búsqueda exhaustiva de la topología de la RN.</i>	25
3. 15	<i>Actualización de los pesos de la RN por retro-propagación.</i>	30
4. 1	<i>Sistema máquina bus infinito</i>	28
4. 2a	<i>Límite determinado por análisis de eigenvalores (LC - 1L)</i>	33
4. 2b	<i>Límite determinado por análisis de eigenvalores (LC - 2L)</i>	33
4. 3a	<i>Límite determinado por análisis de eigenvalores (LL - 1L)</i>	34

<b>4. 3b</b>	<i>Límite determinado por análisis de eigenvalores (LL - 2L)</i>	35
<b>4. 4a</b>	<i>Parámetros de la primer capa oculta en la primer prueba para determinar topología (LC)</i>	38
<b>4. 4b</b>	<i>Parámetros de la segunda capa oculta en la primer prueba para determinar topología (LC)</i>	38
<b>4. 4c</b>	<i>Parámetros de la capa de salida en la primer prueba para determinar topología (LC)</i>	39
<b>4. 5</b>	<i>Resultado de la búsqueda de la topología (LC)</i>	40
<b>4. 6a</b>	<i>Pesos finales para la capa oculta (LC - entrenamiento sin ruido)</i>	40
<b>4. 6b</b>	<i>Pesos finales para la capa de salida (LC - entrenamiento sin ruido)</i>	41
<b>4. 7</b>	<i>Resultado de la búsqueda de la topología (LL)</i>	42
<b>4. 8a</b>	<i>Pesos de la capa oculta en la primer prueba para determinar la topología (LL)</i>	42
<b>4. 8b</b>	<i>Pesos de la capa de salida en la primer prueba para determinar la topología (LL)</i>	43
<b>4. 9a</b>	<i>Pesos de la capa oculta en la segunda prueba para determinar la topología (LL)</i>	44
<b>4. 9b</b>	<i>Pesos de la capa de salida en la segunda prueba para determinar la topología (LL)</i>	44
<b>4. 10a</b>	<i>Pesos de la capa oculta en la tercer prueba para determinar la topología (LL)</i>	45
<b>4. 10b</b>	<i>Pesos de la capa de salida en la tercer prueba para determinar la topología (LL)</i>	45
<b>4. 11a</b>	<i>Pesos finales de la capa oculta (LL - entrenamiento sin ruido)</i>	46
<b>4. 11b</b>	<i>Pesos finales de la capa de salida (LL - entrenamiento sin ruido)</i>	46
<b>4. 12a</b>	<i>Límites por análisis de eigenvalores y por RN (LC - 1L)</i>	48
<b>4. 12b</b>	<i>Límites por análisis de eigenvalores y por RN (LC - 2L)</i>	48
<b>4. 13a</b>	<i>Porcentaje de patrones estables mal identificados por la RN (LC)</i>	49
<b>4. 13b</b>	<i>Porcentaje de patrones inestables mal identificados por la RN (LC)</i>	49
<b>4. 13c</b>	<i>Porcentaje de patrones estables e inestables mal identificados por la RN (LC)</i>	50
<b>4. 14a</b>	<i>Prueba 1 de identificación y límites por RN y por análisis de eigenvalores (LC - 2L)</i>	51
<b>4. 14b</b>	<i>Prueba 2 de identificación y límites por RN y por análisis de eigenvalores (LC - 2L)</i>	52
<b>4. 14c</b>	<i>Prueba 3 de identificación y límites por RN y por análisis de eigenvalores (LC - 1L)</i>	52
<b>4. 14d</b>	<i>Prueba 4 de identificación y límites por RN y por análisis de eigenvalores (LC - 1L)</i>	53
<b>4. 15a</b>	<i>Límites por análisis de eigenvalores y por RN (LL - 1L)</i>	54
<b>4. 15b</b>	<i>Límites por análisis de eigenvalores y por RN (LL - 2L)</i>	55
<b>4. 16a</b>	<i>Porcentaje de patrones estables mal identificados por la RN (LL)</i>	56
<b>4. 16b</b>	<i>Porcentaje de patrones inestables mal identificados por la RN (LL)</i>	56
<b>4. 16c</b>	<i>Porcentaje de patrones estables e inestables mal identificados por la RN (LL)</i>	57
<b>4. 17</b>	<i>Contaminación con ruido a los datos de la simulación</i>	59
<b>4. 18a</b>	<i>Límites por análisis de eigenvalores y por RN1 (LC - 1L)</i>	60

<b>4. 18b</b>	<i>Límites por análisis de eigenvalores y por RN1 (LC - 2L)</i>	60
<b>4. 19a</b>	<i>Límites por análisis de eigenvalores y por RN2 (LC - 1L)</i>	61
<b>4. 19b</b>	<i>Límites por análisis de eigenvalores y por RN2 (LC - 2L)</i>	61
<b>4. 20a</b>	<i>Límites por análisis de eigenvalores y por RN3 (LC - 1L)</i>	62
<b>4. 20b</b>	<i>Límites por análisis de eigenvalores y por RN3 (LC - 2L)</i>	62
<b>4. 21a</b>	<i>Límites por análisis de eigenvalores y por RN1 (LL - 1L)</i>	64
<b>4. 21b</b>	<i>Límites por análisis de eigenvalores y por RN1 (LL - 2L)</i>	64
<b>4. 22a</b>	<i>Límites por análisis de eigenvalores y por RN2 (LL - 1L)</i>	65
<b>4. 22b</b>	<i>Límites por análisis de eigenvalores y por RN2 (LL - 2L)</i>	65
<b>4. 23a</b>	<i>Límites por análisis de eigenvalores y por RN3 (LL - 1L)</i>	66
<b>4. 23b</b>	<i>Límites por análisis de eigenvalores y por RN3 (LL - 2L)</i>	66
<b>4. 24a</b>	<i>Pesos finales de la capa oculta de la RN1 con 4 neuronas en la capa oculta (LC)</i>	68
<b>4. 24b</b>	<i>Pesos finales de la capa de salida de la RN1 con 4 neuronas en la capa oculta (LC)</i>	68
<b>4. 25a</b>	<i>Pesos finales de la capa oculta de la RN2 con 4 neuronas en la capa oculta (LC)</i>	69
<b>4. 25b</b>	<i>Pesos finales de la capa de salida de la RN2 con 4 neuronas en la capa oculta (LC)</i>	69
<b>4. 26a</b>	<i>Pesos finales de la capa oculta de la RN3 con 4 neuronas en la capa oculta (LC)</i>	70
<b>4. 26b</b>	<i>Pesos finales de la capa de salida de la RN3 con 4 neuronas en la capa oculta (LC)</i>	70
<b>4. 27a</b>	<i>Límites por análisis de eigenvalores y por RN1 con 4 unidades en la capa oculta (LC - 1L)</i>	71
<b>4. 27b</b>	<i>Límites por análisis de eigenvalores y por RN1 con 4 unidades en la capa oculta (LC - 2L)</i>	71
<b>4. 28a</b>	<i>Límites por análisis de eigenvalores y por RN2 con 4 unidades en la capa oculta (LC - 1L)</i>	72
<b>4. 28b</b>	<i>Límites por análisis de eigenvalores y por RN2 con 4 unidades en la capa oculta (LC - 2L)</i>	72
<b>4. 29a</b>	<i>Límites por análisis de eigenvalores y por RN3 con 4 unidades en la capa oculta (LC - 1L)</i>	73
<b>4. 29b</b>	<i>Límites por análisis de eigenvalores y por RN3 con 4 unidades en la capa oculta (LC - 2L)</i>	73
<b>4. 30a</b>	<i>Límites por análisis de eigenvalores y por RN1 con 2 unidades en la capa oculta (LL - 1L)</i>	74
<b>4. 30b</b>	<i>Límites por análisis de eigenvalores y por RN1 con 2 unidades en la capa oculta (LL - 2L)</i>	75
<b>4. 31a</b>	<i>Límites por análisis de eigenvalores y por RN2 con 2 unidades en la capa oculta (LL - 1L)</i>	75
<b>4. 31b</b>	<i>Límites por análisis de eigenvalores y por RN2 con 2 unidades en la capa oculta (LL - 2L)</i>	76
<b>4. 32a</b>	<i>Límites por análisis de eigenvalores y por RN3 con 2 unidades en la capa oculta (LL - 1L)</i>	76
<b>4. 32b</b>	<i>Límites por análisis de eigenvalores y por RN3 con 2 unidades en la capa oculta (LL - 2L)</i>	77

<b>4. 33a</b>	<i>Límites por análisis de eigenvalores y por RN1 con 4 unidades en la capa oculta (LL - 1L)</i>	77
<b>4. 33b</b>	<i>Límites por análisis de eigenvalores y por RN1 con 4 unidades en la capa oculta (LL - 2L)</i>	78
<b>4. 34a</b>	<i>Límites por análisis de eigenvalores y por RN2 con 4 unidades en la capa oculta (LL - 1L)</i>	78
<b>4. 34b</b>	<i>Límites por análisis de eigenvalores y por RN2 con 4 unidades en la capa oculta (LL - 2L)</i>	79
<b>4. 35a</b>	<i>Límites por análisis de eigenvalores y por RN3 con 4 unidades en la capa oculta (LL - 1L)</i>	79
<b>4. 35b</b>	<i>Límites por análisis de eigenvalores y por RN3 con 4 unidades en la capa oculta (LL - 2L)</i>	80

## Nomenclatura

$w_i$	Peso sináptico de la neurona $i$ para una RN de una capa.
$w_{ij}$	Peso sináptico de la neurona $i$ proveniente de la entrada $j$ para una RN de una capa.
$w_{ij}^k$	Peso sináptico de la neurona $i$ de la capa $k$ proveniente de la entrada $j$ para una RN multicapa.
$\mathbf{w}$	Matriz de pesos sinápticos para una RN de una capa.
$\mathbf{w}^{11}$	Matriz de pesos sinápticos de la primer capa de una RN multicapa.
$\mathbf{w}^{ij}$	Matriz de pesos sinápticos de la capa $i$ con entradas provenientes de la capa $j$ de una RN multicapa.
$b_i$	Sesgo de la neurona $i$ para una RN de una capa.
$b_i^k$	Sesgo de la neurona $i$ de la capa $k$ de una RN multicapa.
$\mathbf{b}$	Vector de sesgo para una RN de una capa
$\mathbf{b}^k$	Vector de sesgo de la capa $k$ de una RN multicapa.
$n_i$	Combinación lineal de las entradas y los pesos de la neurona $i$ de una RN de una capa
$n_i^k$	Combinación lineal de las entradas y los pesos de la neurona $i$ de la capa $k$ de una RN multicapa.
$a_i$	Salida de la neurona $i$ de una RN de una capa
$a_i^k$	Salida de la neurona $i$ de la capa $k$ de una RN multicapa (también entrada $i$ de la capa $k+1$ ).
$\mathbf{a}$	Salida de una RN de una capa.
$\mathbf{a}^k$	Salida de la capa $k$ de una RN multicapa.
$R$	Número de entradas de la RN.
$S$	Número de neuronas para una RN de una capa.
$S^k$	Número de neuronas de la capa $k$ de una RN multicapa.
$p_i$	Entrada $i$ de la RN.
$\mathbf{p}$	Vector de entradas a la RN.

# 1 | Introducción

## 1.1. Generalidades

La estabilidad de los sistemas eléctricos de potencia (SEP), ha sido objeto de estudio por un largo tiempo; en un principio, los SEP tenían capacidad de sobra en general para satisfacer la demanda de energía eléctrica, pero con el paso de los años, la demanda creció notablemente y así también lo tuvieron que hacer los SEP.

Actualmente, es difícil que los principales centros de consumo estén localizados cerca de los puntos donde se encuentra concentrada la generación, y es necesario transmitir grandes cantidades de energía a lo largo de grandes distancias; es por eso que grandes subsistemas tienen que ser interconectados, y así es como los SEP son analizados.

Con la interconexión en los grandes sistemas de potencia, se ha presentado el fenómeno de oscilaciones a muy bajas frecuencias; estas oscilaciones son generalmente debidas a la insuficiencia de par de amortiguamiento, por lo que no se atenúan, provocando el incremento en el ángulo de carga de las máquinas, y como consecuencia, la desincronización de ellas.

Bajo las condiciones en que la amplitud de dichas oscilaciones crece constantemente, se dice que el sistema es inestable, y las máquinas que pierden sincronismo tienen que ser desconectadas para preservar la integridad del sistema; por eso es importante detectar oportunamente estas condiciones de inestabilidad y evitar que se causen problemas mayores.

Tradicionalmente, la evaluación de la estabilidad se hace a través de modelos matemáticos, los cuales tienen que ser precisos para que el análisis sea confiable [6]; debido a que las oscilaciones provocadas por los pequeños disturbios son de amplitud pequeña, es válido utilizar el modelo del sistema linealizado alrededor del punto de operación, que se considera estacionario aunque en realidad no lo es.

Al trabajarse entonces con un modelo lineal, la estabilidad es determinada a partir de los eigenvalores del sistema de ecuaciones diferenciales que constituyen al modelo; el criterio para determinar la estabilidad es, que si todos los eigenvalores tienen parte real negativa, entonces el sistema es estable, porque la solución al sistema de ecuaciones converge exponencialmente.

Desde hace algunos años, ha habido un importante crecimiento en el desarrollo y aplicación de técnicas de control inteligente (que incluyen el reconocimiento de patrones) [8]-[14], estas técnicas ofrecen alternativas a la solución de varios tipos de problemas como el aquí presentado; dichas técnicas están basadas en general en la experiencia o en observaciones, a diferencia de las técnicas de control tradicional que dependen de un buen modelado para dar buenos resultados.

El reconocimiento de patrones puede realizarse de distintas maneras, la diferencia está en el tipo de información con base a la cual se hace el proceso, los sistemas de reconocimiento de patrones pueden ser [17]-[24]:

- Probabilísticos
- Basados en métricas
- Basados en árbol de decisiones
- Difusos
- Por redes neuronales.

En este trabajo se desarrolla el reconocimiento de patrones por redes neuronales; los estudios en redes neuronales (RN) no son nuevos, ya que inician en los años 30s; pero la formalización de los trabajos se hace en los años 50s, con la presentación del perceptrón en 1957 por Frank Rosenbalt; el perceptrón es la primer RN artificial, tenía varias limitaciones, la mas importante era la incapacidad de clasificar objetos no separados linealmente.

Bernard Widrow y Marcial Hoff desarrollaron el modelo ADALINE (Adaptive Linear Elements) y MADALINE (Multiple Adaptive Linear Elements), que fueron las primeras RN aplicadas exitosamente en un problema real. En 1967, Stephen Grossenberg realizó la red Avalancha, que era una RN con actividad variante en el tiempo.

Por un largo periodo de tiempo las RN no fueron estudiadas de manera importante, hasta principios de los 80s que fueron presentados los modelos SOM (Self Organizing Map) por Teuvo Kohonen y Neocognitrón por Kunihiko Fukushima; otros trabajos incluyen el de John Hopfield (1982) y la teoría de resonancia adaptiva (ART por sus siglas en inglés) de Gail Carpenter y Stephen Grossberg en 1986.

Todos los modelos desarrollados tienen fuertes limitaciones y son aplicables para un problema en específico, pero un tipo de red que causa especial interés es la red tipo retro-propagación, desarrollada inicialmente por Paul Werbos en 1974 generalizando la regla delta de aprendizaje del perceptrón, y redescubierta por D. Rumelhart, G. Hinton y R. Williams en 1986, quienes formalizaron el algoritmo de aprendizaje por retro-propagación.

La red retro-propagación es la más utilizada, debido a la facilidad de aprendizaje y a las numerosas aplicaciones exitosas, pero la limitación principal es la

lentitud del aprendizaje y la necesidad de gran cantidad de ejemplos para obtener buenos resultados [28][30]; para resolver el problema del lento aprendizaje, se han desarrollado distintos algoritmos, todos basados en el del gradiente descendente [26].

## **1.2. Objetivos**

- Proponer un método sencillo para la evaluación de la estabilidad ante disturbios pequeños en el SEP, tomando como base el reconocimiento de patrones con redes neuronales (RN).
- Evaluar el desempeño del sistema de identificación de la estabilidad con RN para diferentes condiciones.
- Investigar los efectos de la incertumbre en el desempeño del sistema de identificación.
- Investigar y proponer la mejor estructura de la RN conforme a su desempeño.

## **1.3. Justificación**

El proceso de linealización y cálculo de eigenvalores no es fácil, además, tratándose de aplicaciones de tiempo real, el tiempo que toma realizarlo puede ser insuficiente para responder adecuadamente ante la presencia de la inestabilidad en el sistema, mas aún porque las condiciones del sistema cambian constantemente.

Las técnicas de control clásico son dependientes completamente de un buen modelado, el cual no siempre es fácil de desarrollar, aunque para sistemas eléctricos de potencia existen modelos prácticos que han sido desarrollados y utilizados satisfactoriamente, por lo que no es realmente necesario investigar en nuevos modelos. Los resultados de la aplicación de estas técnicas tradicionales de control se ven afectados por otro tipo de causas, como pueden ser malas interpretaciones de datos, errores en mediciones, estimaciones equivocadas, etc., las cuales son parte de la realidad; por lo que es necesario lidiar con estas incertidumbres y entender sus efectos para que los resultados no se vean afectados de manera importante.

## **1.4. Descripción del Problema**

La estabilidad ante disturbios pequeños depende del punto de operación [7]; si se tiene un conjunto de mediciones, que corresponden a una condición de operación del sistema, se puede determinar si el sistema es estable o inestable en esa condición; de tal manera que si se tienen varios conjuntos de mediciones, para cada



conjunto corresponde una condición estable o inestable; es decir, cada conjunto de mediciones pertenece a una de dos condiciones.

Visto de esa manera, la determinación de la estabilidad es una clasificación de objetos, cada objeto es un punto de operación, y cada uno se puede clasificar como “estable” o “inestable”; así la identificación de la estabilidad puede verse como un problema de reconocimiento de patrones.

### **1.5. Estado del Arte**

Para las computadoras actuales no es problema realizar el cálculo de los eigenvalores, sin embargo no deja de ser un procedimiento para el que se requiere tiempo y capacidad de cálculo importante; mas que nada para aplicaciones de análisis en línea o en tiempo real, las cuales, por costo, sugieren el uso de microcontroladores, que no tienen la capacidad que tiene una PC.

El campo del análisis de la estabilidad a pequeños disturbios esta ampliamente desarrollado con base en el control clásico. Existe paquetería que tiene capacidad y ha sido desarrollada específicamente para resolver problemas de estabilidad en SEP's, como es el caso del *Small Signal Stability Program (SSSP)* del *Electric Power Research Institute (EPRI)*, que en su versión 6.1 contiene herramientas diversas, incluyendo el *Multi-Area Small Signal Stability Program (MASS)* con capacidad de análisis de hasta 800 estados dinámicos y el *Program for Eigenvalue Analysis of Large Systems (PEALS)* con capacidad de hasta 4500 buses.

La incorporación de las RN's para resolver problemas en los SEP's no es nueva, y como muestra están los trabajos realizados en la Sección de Estudios de Posgrado e Investigación del Instituto Politécnico Nacional en los últimos 10 años, que incluyen áreas como el control de la máquina síncrona, detección y clasificación de fallas en sistemas de transmisión, sintonización de estabilizadores de potencia y estimación de parámetros.

### **1.6. Aportaciones**

En la bibliografía actual referente al reconocimiento de patrones hay bastante información acerca del entrenamiento, conceptos, aplicaciones, etc., pero realmente no queda claro el comportamiento de los sistemas al incorporar incertidumbre en los datos; en este trabajo se presentan de manera concreta los efectos del ruido en los sistemas de identificación de patrones con RN's, específicamente con una RN tipo retro-propagación, la cual tiene un amplio uso debido a sus ventajas sobre otras, específicamente la sencillez y simplicidad de la arquitectura.

Contrario a lo que se podría pensar sin conocimiento previo, cuanto mas compleja es una RN ésta es mas susceptible al ruido en los datos de entrada, dicho

de otra manera, el desempeño ante la incertidumbre es mejor para topologías simples. Lo cual es completamente comprensible una vez si se tiene conocimiento del tema, ya que una RN con topología compleja es capaz de identificar objetos con características mas complejas.

### **1.7. Estructura de la Tesis**

- ✂ En el capítulo 2 de este trabajo, se describe el problema de la estabilidad en los sistemas eléctricos de potencia ante disturbios pequeños, se da la definición y su clasificación [7]; después se presenta un modelo linealizado [1] para el sistema de potencia y el criterio para evaluar la estabilidad en los sistemas lineales, utilizado tradicionalmente para determinar la estabilidad a pequeños disturbios.
- ✂ En el capítulo 3 se dan los conceptos básicos del reconocimiento de patrones y el proceso general que se debe seguir en el diseño de sistemas para tal fin; también se dan conceptos básicos de las redes neuronales y del entrenamiento supervisado para redes multicapa por retro-propagación.
- ✂ En el capítulo 4 se presenta la metodología del entrenamiento de la RN para identificar la estabilidad a pequeños disturbios del sistema de prueba, y los resultados obtenidos en dos casos con características completamente diferentes, los dos casos son generados a partir del cambio de valor de impedancia de la línea de enlace (línea corta y línea larga)
- ✂ Finalmente en el capítulo 5 se concentran las conclusiones obtenidas con base en los resultados que se alcanzan de las pruebas y se resaltan los aspectos mas importantes que se deben tomar en consideración para el desarrollo de sistemas de reconocimiento de patrones basado en redes neuronales; de la misma manera se dan sugerencias para trabajo futuro.

# 2

## Estabilidad a Pequeños Disturbios en los Sistemas Eléctricos de Potencia

### **2.1. Introducción**

El objetivo de un SEP es proporcionar servicio confiable, eficiente y de buena calidad a los usuarios, por lo que es diseñado con un margen de confiabilidad y seguridad para diferentes condiciones de operación. Se considera un sistema dinámico por que constantemente se presentan cambios en la configuración de la red, los cambios pueden ser normales (debidos a cambios de carga o generación) o anormales (causados por fallas), pero son considerados en general como cambios de un estado de equilibrio a otro.

Bajo condiciones normales de operación, la potencia mecánica en las turbinas de los generadores se puede considerar la misma que la consumida en la red a cada instante, sin embargo, cualquier cambio en la turbina o en la carga ocasiona desbalanceo de energía, que se refleja en la aceleración o desaceleración de los rotores de los generadores, estos cambios pueden ocasionar pérdida de sincronismo de las máquinas u oscilaciones no amortiguadas que pueden provocar la desconexión de elementos de la red.

### **2.2. Definición y Clasificación de la Estabilidad del Sistema Eléctrico de Potencia[7]**

La estabilidad del SEP es la capacidad que tiene, dada una condición inicial de operación, de recuperar un estado de equilibrio después de estar sujeto a una perturbación física. Si durante el periodo de transición después de presentarse una perturbación se tiene una respuesta oscilatoria amortiguada y el sistema se restablece en un tiempo finito se dice que es estable.

Los SEP están sujetos a una gran cantidad de perturbaciones, los disturbios pequeños son, en general, los debidos a los cambios en la carga, por otro lado los disturbios grandes son los debidos a fallas en las líneas o en otros elementos importantes como generadores de gran capacidad. Para que el sistema continúe operando apropiadamente, debe ser capaz de adaptarse a estos cambios y evitar que este tipo de disturbios lo lleve a la propia desintegración.

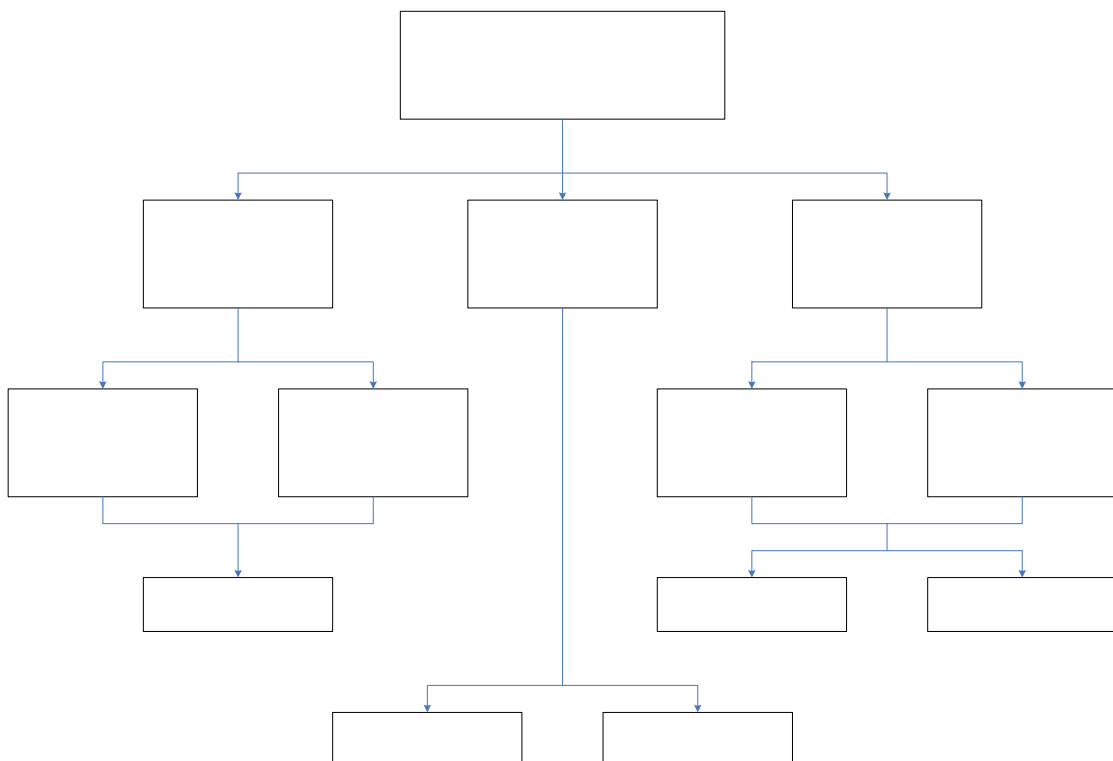
El SEP es altamente no lineal, debido a las características propias de las cargas, los generadores y los demás elementos de la red, por lo que su estabilidad depende tanto de las condiciones de operación como de la naturaleza del disturbio presentado, es decir, la estabilidad es también una propiedad del sistema alrededor de un punto de equilibrio (condición inicial).

La estabilidad depende en gran medida de algunos elementos que lo forman, al presentarse los disturbios, el sistema puede responder desconectando componentes propios del sistema, es decir, cambiando la topología, lo cual puede debilitarlo y ocasionar reacciones en cadena, terminando en problemas mayores como la desintegración del sistema en islas.

De cualquier manera, la configuración de la red cambia constantemente debido a maniobras operativas y/o de protección; aunque en algunos casos es inevitable, se debe asegurar que al llevarse a cabo dichas maniobras, la estabilidad no se vea amenazada, para así conservar la operación adecuada del SEP.

Debido a la gran variedad y la complejidad del problema de la estabilidad, es necesario clasificarlo para que pueda ser analizado de mejor manera (figura 2. 1), en general los criterios para clasificar la estabilidad están dados en función de:

- La naturaleza física de la variable observada desde la cual se analiza la estabilidad del objeto.
- El tamaño del disturbio analizado.
- El rango de tiempo que se considera aceptable para recobrar la estabilidad



*Figura 2. 1 – Clasificación de la estabilidad en el sistema de potencia*

Para el análisis de la estabilidad es válido considerar condiciones de operación en estado estacionario, aunque en realidad no es así, ya que siempre se observan pequeñas fluctuaciones en las variables del sistema.

La estabilidad angular es la capacidad de los generadores de permanecer en sincronismo con el sistema después de presentarse un disturbio; depende de la capacidad de mantener o restablecer el equilibrio entre el par mecánico y el par electromagnético para cada máquina conectada al sistema.

La estabilidad de voltaje es la capacidad del sistema de mantener el voltaje en los nodos del sistema dentro de sus límites; depende de la capacidad de conservar o restablecer el equilibrio entre la carga demandada y la generación en cada instante de tiempo, la inestabilidad de voltaje puede manifestarse como caída de voltaje en uno o mas nodos del sistema, debida a la insuficiente generación para satisfacer la carga, lo que provoca la pérdida de carga del sistema.

La estabilidad de frecuencia depende también del equilibrio entre la carga demandada y la generación, los cambios en el sistema provocan oscilaciones en la frecuencia por la aceleración o desaceleración de las máquinas, el sistema debe ser capaz de mantener la frecuencia del sistema en un rango predeterminado de frecuencia que normalmente es muy restrictivo.

La inestabilidad ante disturbios pequeños puede ser causada por un incremento en el ángulo de carga debido a la carencia de par de sincronización, pero en los sistemas actuales, la inestabilidad ante disturbios pequeños es originada principalmente por insuficiencia en el par de amortiguamiento ocasionando oscilaciones con amplitud en aumento.

Para el análisis adecuado de la estabilidad del SEP es necesario contar con modelos apropiados de acuerdo al tipo de fenómeno presentado, por lo que el sistema se modela dependiendo del tipo de estabilidad que se esta analizando. Para los estudios de estabilidad ante disturbios grandes se debe trabajar con un modelo no lineal, pero en pequeños disturbios se utilizan modelos linealizados del sistema alrededor del punto de operación.

### **2.3. Modelo Linealizado del Sistema Eléctrico de Potencia**

Mientras los SEP crecen, ha sido mas común observar la aparición de oscilaciones a bajas frecuencias, producidas esencialmente por la interconexión de áreas o subsistemas grandes que conforman al sistema completo, estas oscilaciones son causadas principalmente por la falta de par mecánico de amortiguamiento y pueden: desaparecer luego de un tiempo, o crecer, provocando la desintegración del

sistema, como estas oscilaciones son de amplitud pequeña, para el análisis de estabilidad ante la presencia de ellas, se usan modelos linealizados del SEP.

Los análisis de las máquinas síncronas pueden ser simplificados enormemente si las corrientes del rotor y del estator son transformadas en un marco de referencia rotatorio siguiendo al rotor que consta de dos ejes, un eje está alineado al eje del campo conocido como el eje directo (eje  $d$ ) y otro eje ortogonal al primero llamado eje de cuadratura (eje  $q$ ).

Las características principales para este análisis (transformación  $d-q-0$  o transformación de Park) son que bajo operación constante de la máquina las corrientes del rotor y las corrientes transformadas de armadura tienen valores constantes de cd; y eligiendo los dos ejes separados  $90^\circ$  se logra que no haya enlace de flujo, por lo que se pueden considerar dos redes independientes, una para el eje  $d$  y otra para el eje de cuadratura.

Durante oscilaciones de baja frecuencia, las corrientes inducidas al devanado amortiguador son tan pequeñas que pueden ser despreciadas, por lo que estos devanados también pueden ser ignorados en el modelado del sistema bajo dichas condiciones.

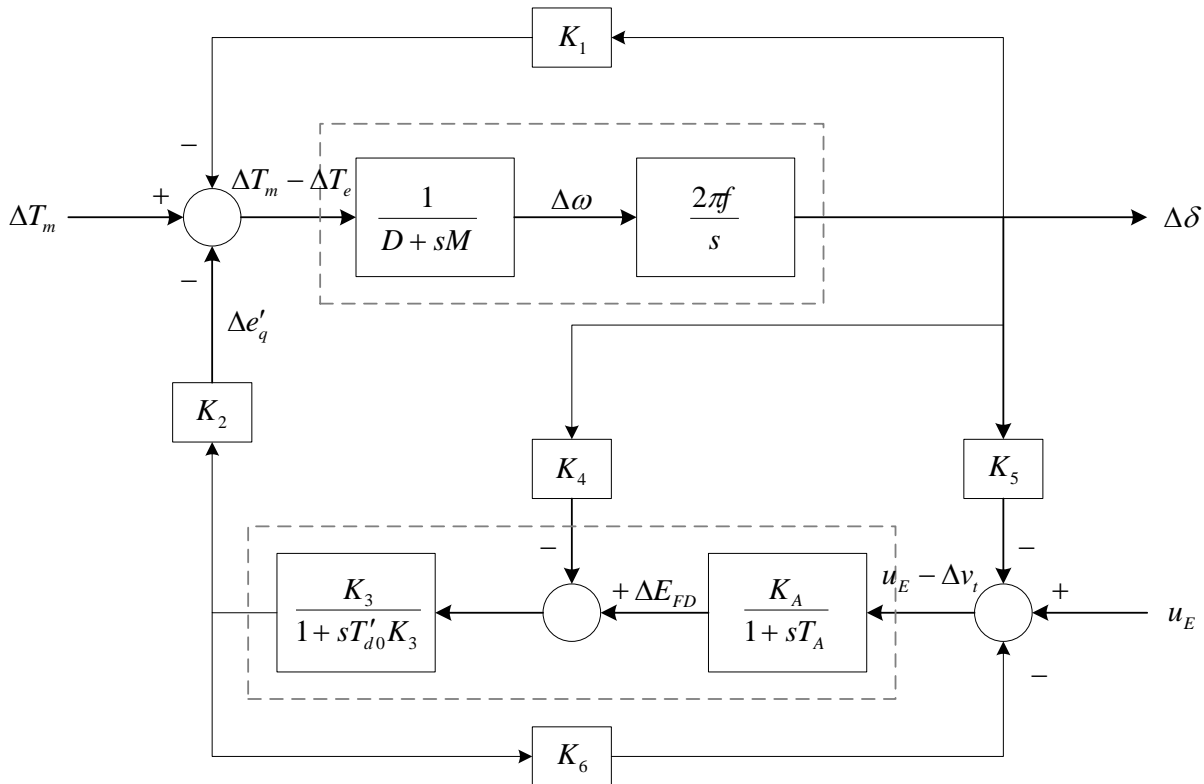


Figura 2. 2 – Diagrama a bloques de función de transferencia para estudios de oscilaciones de baja frecuencia [1]

En la figura 2. 2 se muestra el modelo linealizado de tercer orden de un sistema completo para estudios de oscilaciones de baja frecuencia, tiene dos lazos principales, uno para el sistema mecánico y otro para el sistema eléctrico. El sistema tiene dos entradas, el lazo mecánico tiene como entrada el cambio en el par, y como salida el cambio del ángulo de carga, que es también la salida del sistema, el lazo del sistema eléctrico tiene como entrada la diferencia de una señal de control menos el cambio del voltaje en terminales, y como salida al cambio del voltaje interno de la máquina. El modelo mostrado en la figura contiene también un bloque que representa un sistema excitador y regulador de voltaje de respuesta rápida, el otro bloque del sistema eléctrico representa la función de transferencia del circuito de campo afectado por la reacción de armadura.

Para poder hacer un análisis de estabilidad basado en los eigenvalores del sistema, se requiere escribir el sistema en la forma estándar de variables de estado en ecuaciones de primer grado; de la figura 2. 2, se tiene en forma matricial:

$$\begin{bmatrix} \dot{\Delta\delta} \\ \dot{\Delta\omega} \\ \dot{\Delta e'_q} \\ \dot{\Delta E_{FD}} \end{bmatrix} = \begin{bmatrix} 0 & \omega_s & 0 & 0 \\ -\frac{K_1}{M} & -\frac{D}{M} & -\frac{K_2}{M} & 0 \\ -\frac{K_4}{T'_{d0}} & 0 & \frac{1}{T'_{d0}K_3} & \frac{1}{T'_{d0}} \\ -\frac{K_A K_5}{T_A} & 0 & -\frac{K_A K_6}{T_A} & -\frac{1}{T_A} \end{bmatrix} \begin{bmatrix} \Delta\delta \\ \Delta\omega \\ \Delta e'_q \\ \Delta E_{FD} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{M} & 0 \\ 0 & 0 \\ 0 & \frac{K_A}{T_A} \end{bmatrix} \begin{bmatrix} \Delta T_m \\ u_E \end{bmatrix} \quad (2. 1)$$

que corresponde a la forma  $\dot{x} = Ax + Bu$ .

Para calcular las constantes  $K_1, K_2, \dots, K_6$  del sistema (2. 1) se deben conocer las condiciones de operación de la máquina, las cuales se obtienen usualmente de un estudio de flujos de carga.

## 2.4. Estabilidad en los Sistemas Lineales

La idea de verificar estabilidad del sistema no lineal desde el modelo lineal del sistema se conoce como el 1er. Método de Lyapunov o método indirecto de Lyapunov (1892). Sea el sistema lineal autónomo:

$$\dot{x} = Ax \quad (2. 2)$$

El cual tiene un punto crítico (PC) o punto de equilibrio (PE)  $\bar{x}$  en el origen, que es aislado sí  $\det A \neq 0$ . Si  $\det A = 0$ , todo punto en el subespacio nulo de  $A$  es un PE. Un sistema lineal no puede tener múltiples PE aislados, porque si  $\bar{x}$  y  $\bar{z}$  son dos PE de (2. 2), entonces, por linealidad, todo punto en la recta que conecta a  $\bar{x}$  y  $\bar{z}$  es un PE.

Se dice que el PE  $\bar{x}$  del sistema (2. 2) es Estable si siempre que el estado del sistema se inicia cerca de ese punto, permanece cerca de él, incluso puede hasta tender a él conforme pasa el tiempo, entonces se dice que el PE es Asintóticamente Estable (AE). Si  $s(\bar{x}, \varepsilon)$  denota una región esférica en el espacio de estado con centro en  $\bar{x}$  y radio  $\varepsilon$ :

- i. El PE  $\bar{x}$  es Estable si existe un  $\delta > 0$  para el cual se cumple:  
Para toda  $\varepsilon < \delta$ , existe un  $r$ ,  $0 < r < \varepsilon$  tal que si  $x(0)$  esta dentro de  $s(\bar{x}, \varepsilon)$  entonces  $x(t)$  esta dentro de  $s(\bar{x}, r)$  para todo  $t > 0$ .
- ii. Un PE  $\bar{x}$  es AE si es Estable y además existe un  $\delta^* > 0$  tal que siempre que el estado se inicie dentro de  $s(\bar{x}, \delta^*)$  tiende a  $\bar{x}$  cuando  $t \rightarrow \infty$ .
- iii. Un PE es Marginalmente Estable si es estable pero no asintóticamente.
- iv. Un PE es Inestable si no es Estable.

Las propiedades de estabilidad del origen pueden caracterizarse mediante la ubicación de los eigenvalores de  $A$ . Cuando todos los eigenvalores de  $A$  tienen parte real negativa, se dice que  $A$  es una matriz de estabilidad o *matriz Hurwitz*; la estabilidad del sistema se determina con el método indirecto de Lyapunov con base en las siguientes pruebas:

1. El origen es AE si todos los eigenvalores de  $A$  tienen parte real negativa.
2. El origen es inestable si uno o más eigenvalores de  $A$  tiene parte real positiva.

## 2.5. Regiones y límite de estabilidad

La estabilidad del SEP ante disturbios pequeños depende, como ya se mencionó anteriormente, del punto de operación; analizado desde el punto de vista de los eigenvalores con base en el método indirecto de Lyapunov, si dada una condición inicial o punto de operación, se tiene que todos los eigenvalores tienen parte real negativa, entonces el sistema es AE; y si dada otra condición inicial, se tiene que al menos un eigenvalor tiene parte real positiva, entonces el sistema será inestable.

Pero puede presentarse también el caso en el que para cierto punto de operación, algún eigenvalor tenga parte real nula, es decir, igual a cero; en esta condición el sistema lineal es marginalmente estable, ya que no tiende al origen cuando  $t \rightarrow \infty$ , y no es posible dar conclusión acerca de la estabilidad del sistema no lineal.

Cuando un eigenvalor del sistema tiene parte real nula, entonces se está en el límite de estabilidad (figura 2. 3); porque si se mantienen constantes todas menos una variable del sistema, los eigenvalores dependen solamente de esa variable, y así



también la estabilidad del sistema; entonces en la cercanía de dicho punto de operación se puede tener una condición de estabilidad asintótica, pero con un cambio pequeño en la magnitud de dicha variable se podrá pasar a una condición de inestabilidad.

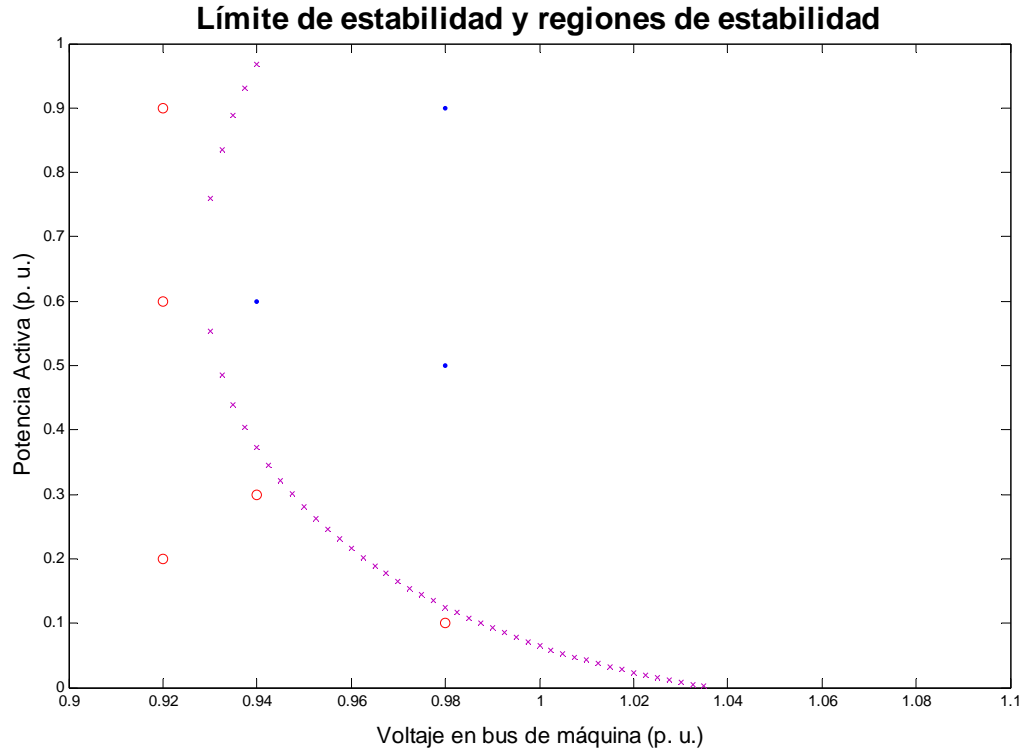


Figura 2. 3 – Límite y regiones de estabilidad

Si todos los puntos vecinos a una condición estable son también estables, ese conjunto de puntos de operación estables forman una región estable (figura 2. 3), la cual esta delimitada por el límite de estabilidad, que es donde al menos un eigenvalor tiene parte real nula; y mas allá de ese límite se encontrará la región de inestabilidad.

Tabla 1. 1 – Parte real de eigenvalores

ID	Vt (p. u.)	P (p. u.)	Parte Real	
			EV 1, 2	EV 3, 4
1	0.92	0.2	-25.438	0.0206
2	0.92	0.6	-25.432	0.0145
3	0.92	0.9	-25.456	0.0388
4	0.94	0.3	-25.424	0.006
5	0.94	0.6	-25.398	-0.0192
6	0.98	0.1	-25.42	0.0023
7	0.98	0.5	-25.331	-0.0863
8	0.98	0.9	-25.268	-0.1493

# 3

## Redes Neuronales para Reconocimiento de Patrones

### 3.1. Introducción

Los estudios en el reconocimiento de patrones comienzan formalmente a principios de los cincuentas, para finales de la misma década es cuando Rosenbalt presenta el perceptrón, el cual fue desarrollado tratando de imitar la forma en la que el cerebro organiza la información, convirtiéndose en la primer RN artificial (RN) de la historia, pero no es hasta mediados de los ochentas, que las RN adquieren importancia y son realmente aplicadas, aunque han sido estudiadas desde los años treinta.

### 3.2. Reconocimiento de Patrones

El desarrollo de la teoría para el reconocimiento de patrones tuvo una gran importancia debido a la necesidad de manejar grandes volúmenes de información, la cual requiere no solo almacenarse, sino también organizarse apropiadamente para su adecuado manejo, es evidente que si se tiene una gran cantidad de información, es más fácil manejarla al agruparla, por lo que el reconocimiento de patrones puede verse como un problema de clasificación y manejo de información.

Un patrón, identificado por una etiqueta propia, es la descripción general de un objeto y cualquier otro objeto que posea las mismas características que este patrón, se puede identificar por la misma etiqueta, este proceso de identificación de los objetos con base a características o rasgos comunes, es el reconocimiento de patrones [16]; si un objeto es descrito por  $n$  características  $x_i$  con  $i=1,2,\dots,n$ , de manera que para cada objeto se puede formar el vector  $\mathbf{x}=[x_1 \ x_2 \ \dots \ x_n]^T$  (llamado vector de características), y si se tienen  $m$  clases de objetos distintos  $c_j$  con  $j=1,2,\dots,m$ , entonces se puede decir que el reconocimiento de patrones, es un mapeo de información del espacio de características de los objetos al espacio de etiquetas que los identifican (el reconocimiento de patrones es un mapeo  $\mathbb{R}^n \rightarrow \mathbb{R}$ ) [18].

El reconocimiento de patrones se hace con base en información previa de los objetos, con mayor información descriptiva que se tenga de los patrones, la clasificación puede hacerse de mejor manera, aunque para un reconocimiento

apropiado es importante tomar en cuenta solo los rasgos que realmente describen a los objetos [17].

El tipo de información que describe un objeto y que puede ser usada para clasificarlo es diverso, en ocasiones la distinción de los objetos se hace con un enfoque cuantitativo y cualitativo, en donde además de diferenciar por el tipo de características que poseen los objetos, se toma en cuenta el grado de similitud.

### 3.2.1. Proceso General

En general, los sistemas de reconocimiento de patrones se realizan en cuatro etapas (figura 3. 1), pudiendo cambiar para cada caso en particular; normalmente el proceso sigue un orden y un sentido, pero puede desarrollarse de distintas maneras.



Figura 3. 1 – Etapas en el reconocimiento de patrones

El sensado depende del tipo de aplicación, y por consiguiente del tipo de datos que se estén procesando, normalmente en esta etapa se utilizan transductores, los problemas que aquí se pueden presentar dependen completamente de las características del transductor.

La extracción de rasgos es realmente la parte fundamental y está relacionada con la etapa de la clasificación, la complejidad de esta etapa depende del tipo de aplicación, se refiere a la determinación de las características con base a las cuales se discriminan los objetos, y es donde se lleva a cabo la interpretación numérica de los rasgos para su manejo por el clasificador en sí.

La etapa de la clasificación utiliza el vector de rasgos, proveniente del extractor de características, para asignar el objeto a su categoría correspondiente. Existen distintos tipos de clasificadores, la diferencia entre ellos radica principalmente en el tipo de información que procesan y la manera en la que las decisiones son tomadas, principalmente pueden ser [17]-[24]:

- ⇒ Probabilísticos
- ⇒ Basados en métricas
- ⇒ Basados en árbol de decisiones
- ⇒ Difusos
- ⇒ Redes neuronales.

El post-procesamiento implica una evaluación del resultado de la clasificación; los sistemas de reconocimiento de patrones normalmente son solo parte de una aplicación, y debido a diversas circunstancias, el clasificador puede arrojar resultados

erróneos, con el consecuente costo por una decisión tomada inadecuadamente, en esta última etapa de sistema es en la que se pueden hacer ajustes para minimizar el costo total.

### 3.2.2. Diseño de Sistemas

El diseño de un sistema de reconocimiento de patrones implica varias etapas fundamentales, no existe una metodología general, pero la manera usual de hacerlo es como se indica en la figura 3. 2.

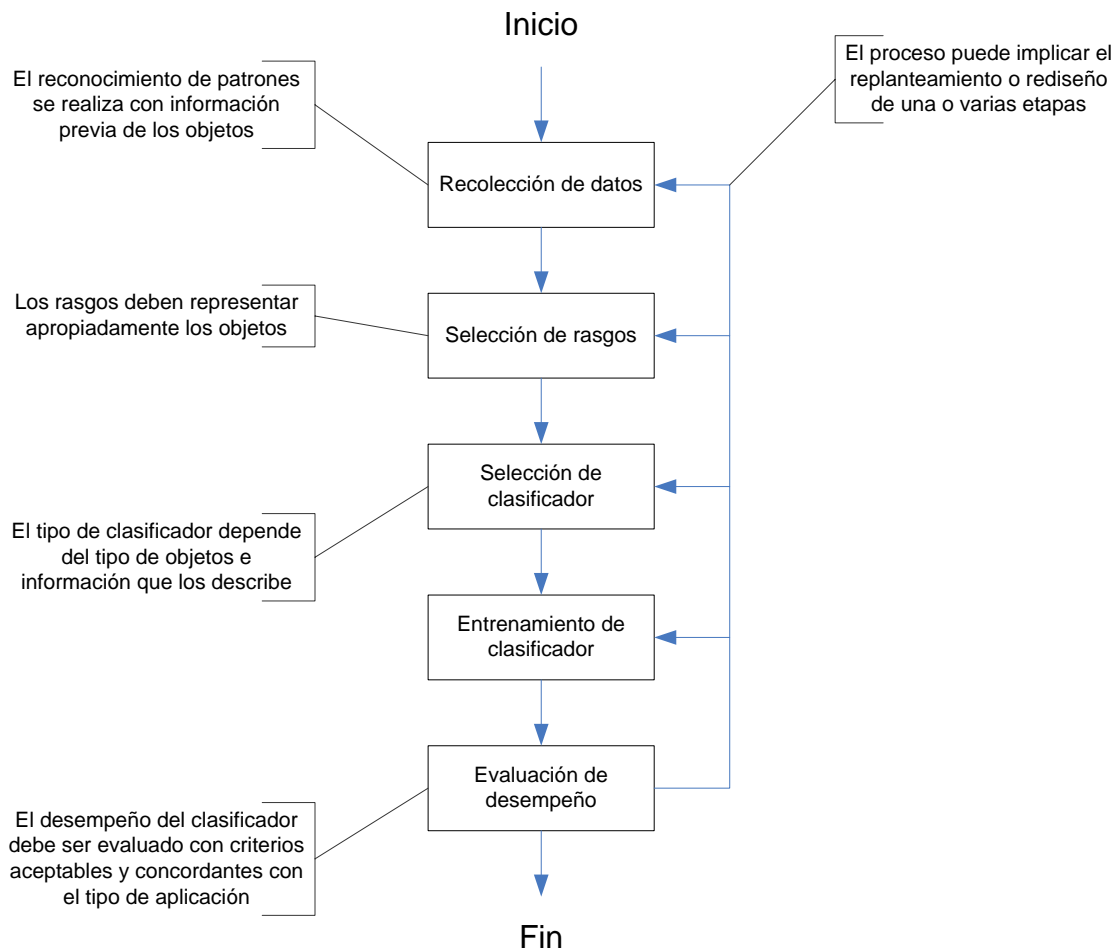


Figura 3. 2 – Diseño de sistemas de reconocimiento de patrones [17]

El entrenamiento está ligado directamente con el tipo de clasificador, el cual depende del tipo de información que se va a procesar, que es la que define a los objetos y la base para la clasificación; la información que se requiere para realizar el reconocimiento de patrones está concentrada en los datos recolectados, sin embargo, otro tipo de información basada en observaciones y experiencia puede ser útil para otras etapas, como para la selección de los rasgos característicos.

Los rasgos deben ser suficientes para describir adecuadamente los objetos, pero no deben ser redundantes ya que esto puede afectar al desempeño del clasificador. Para poder seleccionar adecuadamente los rasgos y el tipo de clasificador, habitualmente es necesario el previo conocimiento de la aplicación y el proceso al cual se integra el reconocimiento de patrones

### 3.3. Redes Neuronales [26][27][28]

El principio del trabajo en RN parte del reconocimiento de que el cerebro humano funciona de manera totalmente diferente a la computadora digital convencional. El cerebro es un sistema de procesamiento de información altamente complejo, no lineal y paralelo, tiene la capacidad de organizar sus componentes estructurales, conocidos como neuronas, con el fin de realizar ciertas funciones mucho más rápido que la computadora digital más veloz. Desde el nacimiento del ser, el cerebro desarrolla su estructura basándose en lo que conocemos como la experiencia, la cual se va formando a través del tiempo mediante la recolección de datos y el análisis de ellos.

En su forma más general, una RN es una máquina que está diseñada para imitar la manera en la que el cerebro realiza una tarea en particular o una función de interés; para un buen funcionamiento, las RN, usualmente implementadas en computadoras digitales, emplean la interconexión masiva de sencillas celdas de cálculo o unidades de procesamiento haciendo referencia a las neuronas.

#### 3.3.1. Modelo de una Neurona

Una neurona es una unidad de cálculo, en la que básicamente se suman las  $R$  entradas  $p_i$  multiplicadas cada una por un peso  $w_j$  más un sesgo (figura 3. 3), el sesgo puede ser visto también como una entrada (de valor unitario) a través de un peso  $b$ . Luego de sumarse las entradas ponderadas, la salida de la neurona es una función (lineal o no) de dicha sumatoria.

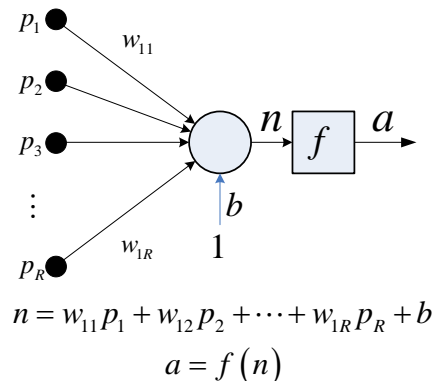


Figura 3. 3 – Una neurona

$\mathbf{w}$  es un vector llamado vector de pesos, y  $\mathbf{p}$  es el vector de entrada, se puede observar claramente que  $n = \mathbf{w}\mathbf{p} + b$ , y la salida de la neurona es una  $f(n)$ ; la función de activación, como se denomina, puede ser cualquiera, pero principalmente se utilizan las de la forma lineal y la sigmoideal, en parte porque para el entrenamiento se realiza un proceso de optimización, donde se requieren las derivadas de dichas funciones.

Una nomenclatura más simple para la neurona se muestra en la figura 3. 4, (debajo de cada elemento se indican sus dimensiones).

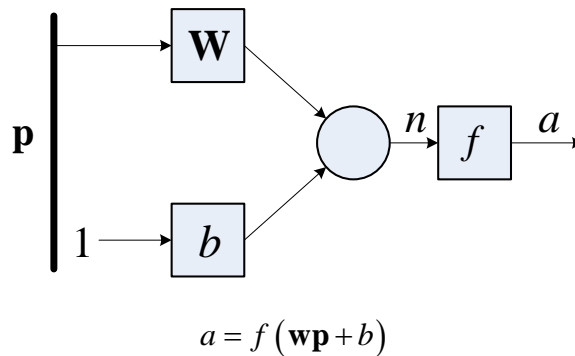


Figura 3. 4 – Nomenclatura simplificada para una neurona[26]

### 3.3.2. Una Capa de Neuronas

Una capa de neuronas es simplemente el conjunto de varias unidades operando simultáneamente (figura 3. 5), una capa de  $R$  entradas y  $S$  neuronas sería:

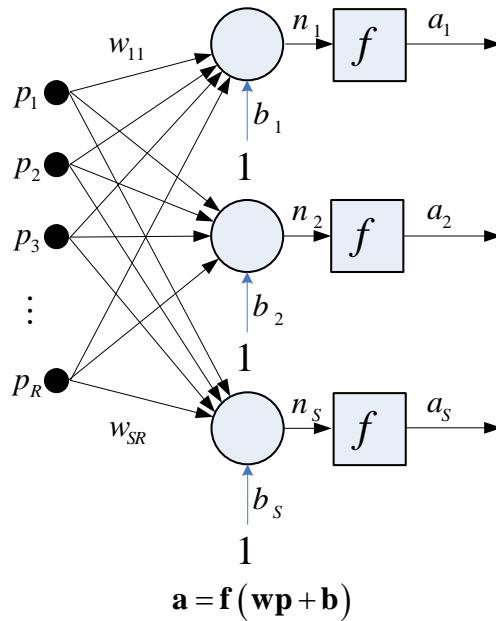


Figura 3. 5 – Una capa de neuronas

En esta red, cada entrada tiene conexión con cada neurona, sin embargo esto no es necesario en todos los casos, dependiendo de la estructura, puede haber o no conexión entre una entrada y una neurona específicas, para una capa de neuronas  $\mathbf{w}$  es:

$$\mathbf{w} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1R} \\ w_{21} & w_{22} & \cdots & w_{2R} \\ \vdots & \vdots & & \vdots \\ w_{S1} & w_{S2} & \cdots & w_{SR} \end{bmatrix} \quad (3.1)$$

Los índices del renglón de la matriz indican la neurona destino del peso, y los índices de columna indican el origen de la entrada a la que pertenecen. De manera abreviada la misma red mostrada puede dibujarse como en la figura 3. 6.

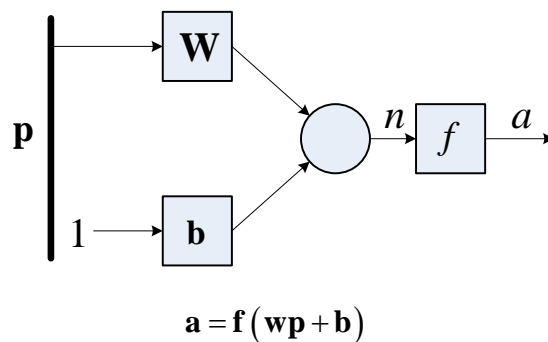


Figura 3. 6 – Nomenclatura simplificada para una capa de neuronas[26]

### 3.3.3. Redes Multicapa

Una red multicapa de neuronas es una red compuesta por varias capas de neuronas como las descritas anteriormente, la salida de una capa es la entrada de la siguiente y así sucesivamente.

Una red de dos capas, con la primer capa con función sigmoideal y la segunda con función lineal, puede ser entrenada para aproximar bien cualquier función con cualquier precisión, esta clase de redes es conocida como red tipo retro-propagación, debido al algoritmo con el cual se entrenan, en las figuras 3. 7 y 3. 8 se muestra una red general de tres capas.

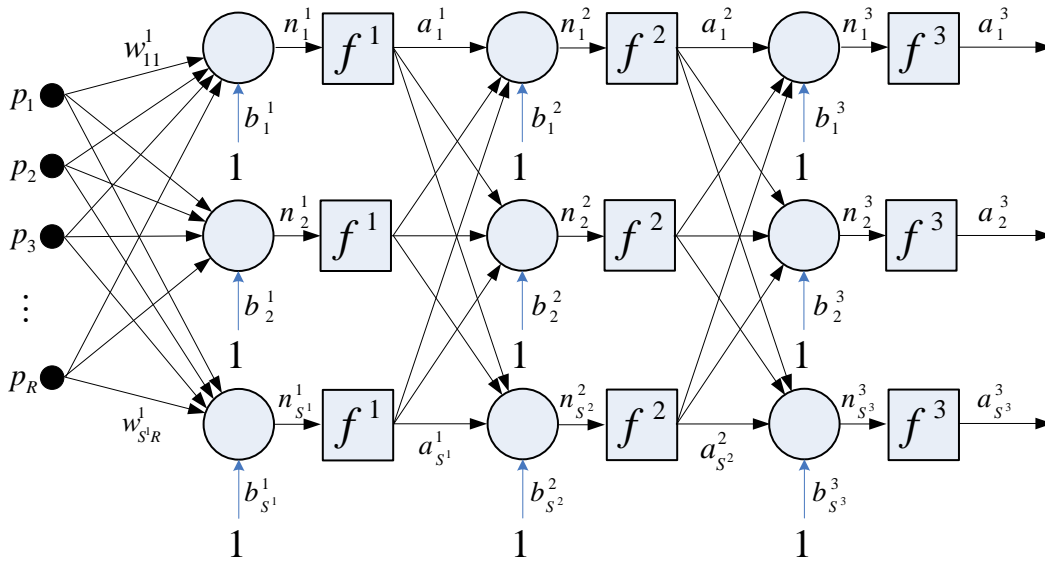
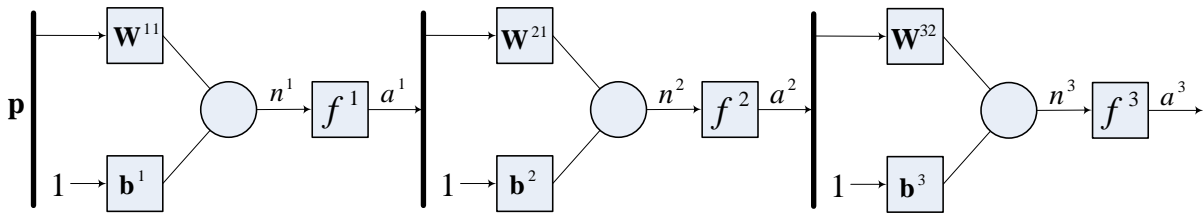


Figura 3. 7 – Una RN multicapa



$$a^1 = f^1(w^{11}p + b^1) \quad a^2 = f^2(w^{21}a^1 + b^2) \quad a^3 = f^3(w^{32}a^2 + b^3)$$

$$a^3 = f^3(w^{32}f^2(w^{21}f^1(w^{11}p + b^1) + b^2) + b^3) = y$$

Figura 3. 8 – Nomenclatura simplificada para una RN multicapa[26]



La modificación de los pesos sinápticos es el método tradicional para el diseño de redes neuronales, es posible que la red modifique su propia topología, lo cual es motivado por el hecho de que las neuronas en el cerebro humano pueden morir y entonces nuevas conexiones sinápticas pueden madurar.

Las redes neuronales deben su poder a su estructura distribuida paralela y a su habilidad de aprendizaje y por lo tanto de generalizar. La generalización en las redes neuronales se refiere a la producción de salidas congruentes para entradas no presentadas durante el entrenamiento o aprendizaje. Estas dos capacidades de procesamiento de la información hacen posible para las redes neuronales la solución de problemas complejos, sin embargo, en algunos casos, las redes neuronales no son capaces de resolver los problemas por si mismas, mas bien, son integradas a un sistema consistente; un problema complejo se descompone en tareas relativamente sencillas y las redes neuronales se asignan a la solución de problemas que encajan dentro de sus capacidades.

### 3.3.4. Función de Activación

Existe una variedad de funciones que han sido satisfactoriamente utilizadas como funciones de activación, éstas pueden ser lineales o no lineales y la elección depende del tipo de problema específico que se trata de resolver, las no lineales son útiles, especialmente para problemas en los que las redes neuronales se utilizan en clasificación de patrones. El entrenamiento es parte importante en el diseño de redes neuronales, las herramientas más poderosas para entrenar redes neuronales, son procesos numéricos de optimización basados en derivadas, por lo que se hace necesario utilizar funciones de activación diferenciables.

La función rígida (figura 3. 9) fue la utilizada en las primeras redes neuronales (perceptrón); es una función no lineal que comprime el rango infinito de entradas a un rango finito de salidas, específicamente con dos valores posibles (0/1 ó  $-1/1$ ), debido a que se trata de una función discontinua, su utilización es limitada principalmente debido a la dificultad del entrenamiento de perceptrones multicapa.

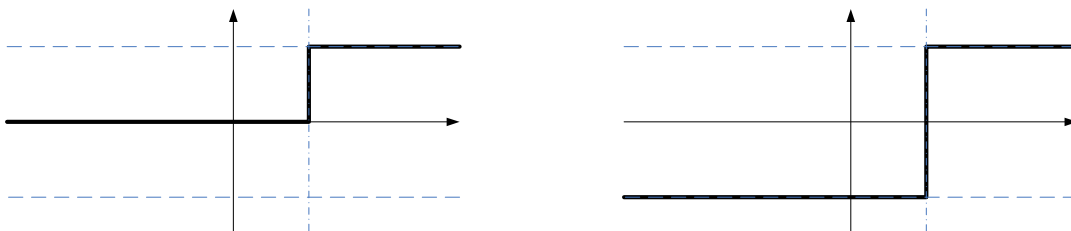


Figura 3. 9 – Función rígida

La función lineal (figura 3. 10) utilizada en las redes neuronales es  $f(x) = x$ , normalmente el objetivo de utilizarla es realizar la superposición de las salidas de las

neuronas de una capa anterior en redes multicapa, aproximando curvas de manera equivalente a la unión de varios segmentos individuales.

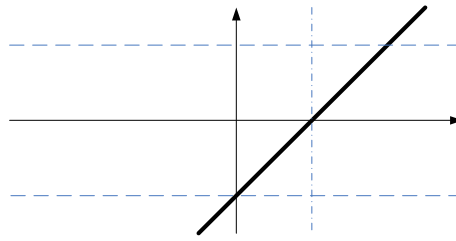


Figura 3. 10 – Función lineal

La función sigmoideal (figura 3. 11) puede tener un comportamiento muy similar a la función rígida como puede observarse en la figura 3. 12, ahí también se puede observar que incluso puede tener un comportamiento casi lineal.

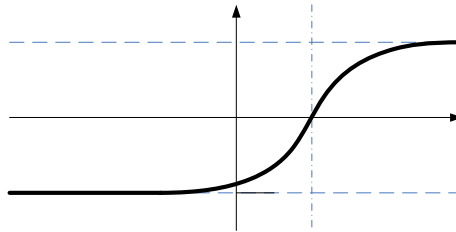
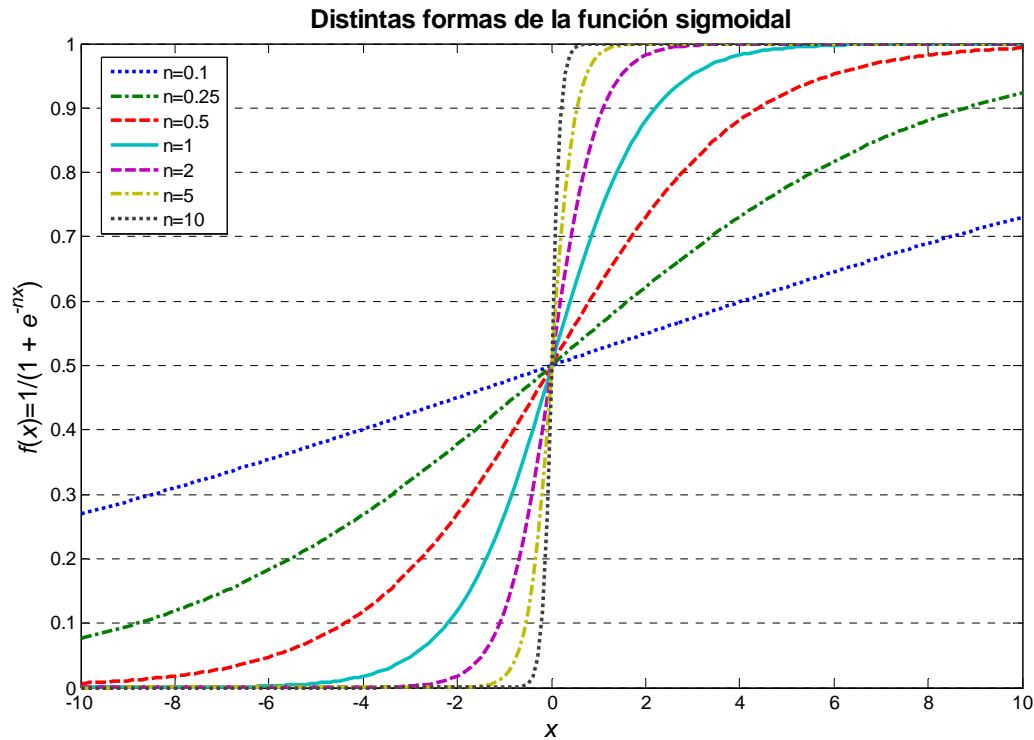


Figura 3. 11 – Función lineal

$$f(x) = \frac{1}{1 + e^{-x}}$$

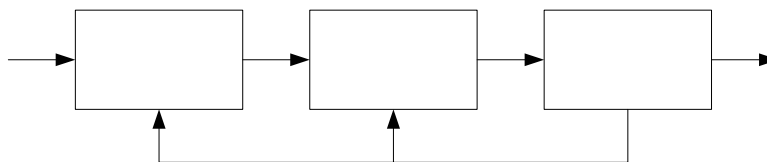


Figuras 3. 12 – Familia de curvas de la función sigmoial

En una neurona, se puede decir que una  $n$  grande equivale a pesos  $w$  de magnitud grande, y una  $n$  pequeña equivale a pesos  $w$  de magnitud pequeña, entonces si una neurona tiene pesos pequeños la función de activación se aproxima a una función lineal, y si tiene pesos grandes a una función rígida.

### 3.3.5. Diseño

La metodología típica para diseñar redes neuronales se muestra en la figura 3. 13, las características de cada etapa en el proceso están definidas por la aplicación del sistema para la cual se está diseñando; debido a diversas circunstancias, la etapa crítica del proceso es la determinación de la topología, en gran parte por la diversidad de aplicaciones y por lo tanto la imposibilidad de globalizarlas o generalizarlas, el pre-procesamiento y el entrenamiento en cambio, son etapas en las que el conocimiento está más acotado.



Figuras 3. 13 – Metodología de diseño de RN

### 3.3.5.1. Pre-procesamiento

En el pre-procesamiento de los datos se llevan a cabo dos acciones, la normalización de los datos y el análisis de ellos. El fin de la normalización es el de hacer que todos los datos tengan magnitudes dentro de un rango típico de  $[0,1]$  o de  $[-1,1]$  para que el entrenamiento sea homogéneo, de lo contrario es posible que durante el entrenamiento, algunos parámetros encuentren su valor final rápidamente mientras que otros tarden demasiado, se puede intuir que sería un proceso ineficiente.

El análisis de los datos se hace para eliminar todos aquellos datos no útiles que contengan información poco precisa y provoquen mal entrenamiento de la RN, normalmente se separan dos juegos de datos, uno para entrenamiento y otro para prueba, y en ocasiones se considera un tercer grupo de datos de validación que se utiliza para monitorear el entrenamiento de la RN, ya que es conocido que se puede presentar el fenómeno de sobreentrenamiento o pérdida de generalización.

### 3.3.5.2. Determinación de la Topología

En los sistemas de identificación de patrones se puede conocer el número de rasgos que determinan las clases y el número de clases que existen, para una RN como sistema de identificación el número de rasgos corresponde al número de unidades en la capa de entrada y la cantidad de clases esta relacionada directamente con el número de unidades en la capa de salida, y se ha comprobado que una RN de tres capas con función de activación sigmoideal en la capa oculta y lineal en la capa de salida es suficiente para aproximar cualquier función con cualquier grado de precisión.

En el diseño de sistemas de reconocimiento de patrones por RN, el problema es determinar el número de unidades en la capa oculta, y en ocasiones el número de capas ocultas necesarias para obtener buenos resultados.

Se han propuesto algunos métodos para determinar la topología de las RN, sin embargo no hay método alguno que de buenos resultados en general, el número de parámetros de una RN, que esta determinado por el número de unidades, determina la capacidad de identificación del sistema, cuanto mayor sea el número de unidades la RN es capaz de identificar patrones separados por límites de decisión complejos, pero si los patrones son linealmente separados o por límites uniformes y bien definidos entonces un número pequeño de unidades es suficiente.

Los métodos que se han desarrollado para determinar el número de unidades en las capas ocultas son variados, algunos se basan en aumentar o disminuir el

número de unidades durante el entrenamiento con base al error de entrenamiento, o con base a la sensibilidad del error de cada parámetro [31], en otro se elige basado en un análisis previo de los datos de entrenamiento donde el número de unidades corresponde al número de hiperplanos que separan las distintas clases, otros métodos se basan en reglas empíricas o técnicas heurísticas y algunos otros más se basan en técnicas estadísticas.

Algunos criterios para la selección de la topología de la RN son basados en la cantidad de datos de entrenamiento, lo cual es válido cuando se tienen pocos datos, pero cuando son suficientes, entonces los criterios cambian y se basan en la complejidad de los límites que separan las clases (desde el punto de vista del reconocimiento de patrones); invariablemente siempre se debe buscar la topología menos compleja y con menos parámetros, esto resulta en sistemas más rápidos y eficientes.

La búsqueda exhaustiva (figura 3. 14) consiste en entrenar varias veces la RN cambiando el número de unidades en la capa oculta y de ser necesario el número de capas ocultas, haciéndolo con distintos valores iniciales de los pesos, en este trabajo el criterio para determinar la mejor topología de la RN es buscar la RN para la cual el número de épocas en el que se llegue a cierto error de entrenamiento sea el menor.

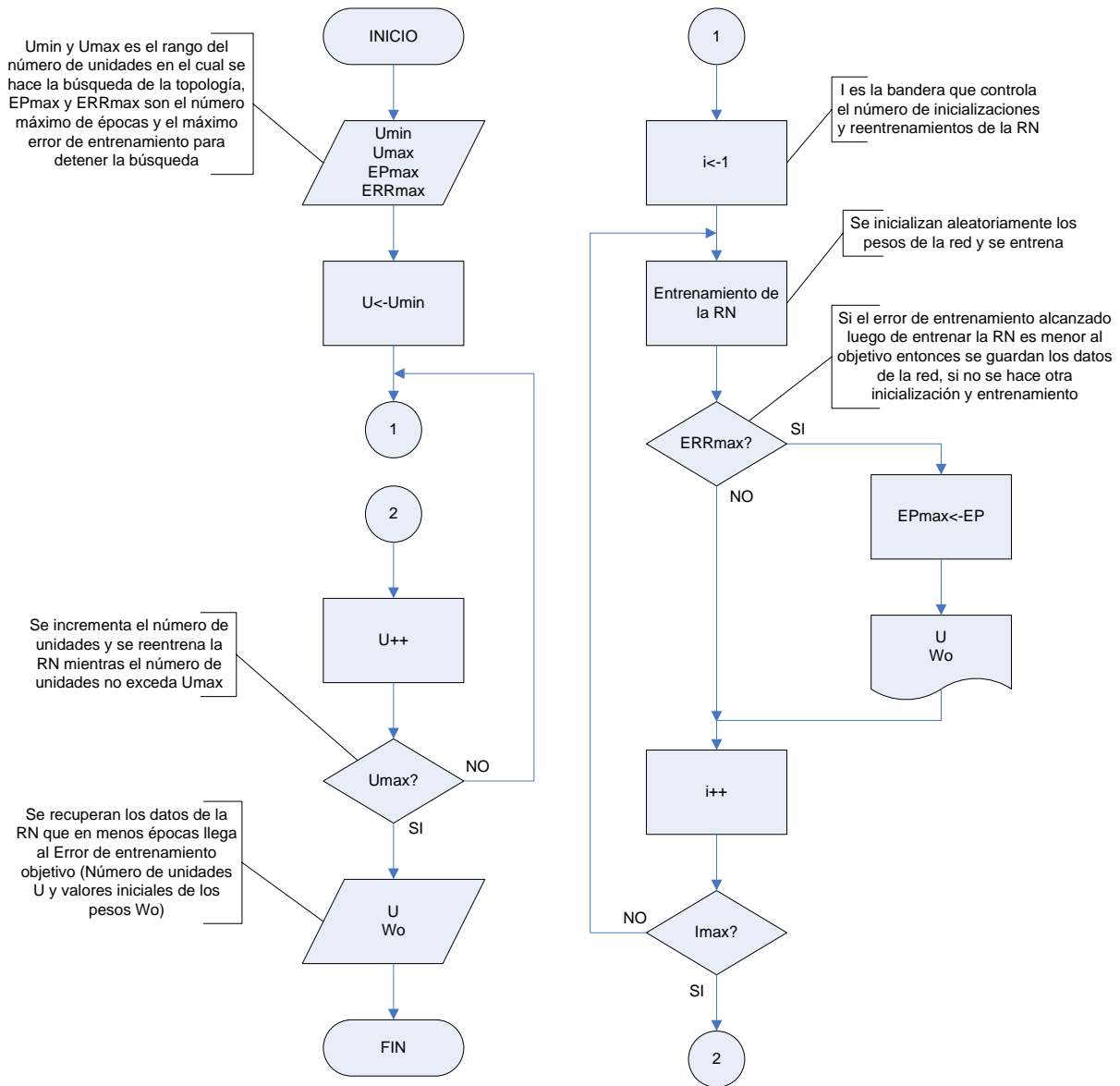


Figura 3. 14 – Diagrama de flujo de búsqueda exhaustiva de la topología de la RN

### 3.3.5.3. Entrenamiento

El procedimiento mediante el cual la RN se adapta de para que responda de manera deseada se llama algoritmo de aprendizaje, cuya función es la de modificar los parámetros de la RN metodológicamente para alcanzar el objetivo de diseño deseado.

Un aspecto importante a considerar es la inicialización de los parámetros de la RN que serán ajustados durante el entrenamiento, de estas condiciones iniciales depende completamente el entrenamiento, las RN se inicializan aleatoriamente para

su entrenamiento cuando no se tiene idea de los valores óptimos y es recomendable realizar varios entrenamientos con valores iniciales distintos cada vez.

El método de entrenamiento es importante durante el proceso mismo, aunque es de esperarse que si el entrenamiento converge, no importa cual sea el método, la RN entrenada tendrá un buen desempeño, en este trabajo se eligió el método de optimización de Levenberg-Marquardt que generalmente resulta en entrenamientos más rápidos y estables.

### 3.3.6. Entrenamiento Supervisado

Hay dos tipos en entrenamiento para las RN, el supervisado y el no supervisado, el entrenamiento supervisado es aquel en el que se cuentan con datos entrada-salida y estos se utilizan para entrenar a la RN. Otra clasificación de los tipos de entrenamiento es entrenamiento incremental y entrenamiento en conjunto, la diferencia es que en el entrenamiento incremental los pesos se actualizan cada vez que se presenta un juego de datos (entrada-salida), y en el entrenamiento en conjunto, los pesos se actualizan hasta que se han presentado todos los juegos de datos. El uso del entrenamiento incremental es cuando los juegos de datos se tienen de uno a la vez, y el entrenamiento en conjunto es cuando se cuenta con todos los juegos de datos reunidos en una matriz.

#### 3.3.6.1. Gradiente Descendente

Éste, es un algoritmo iterativo de optimización; la idea en la cual esta fundamentada es en la minimización del error como una función de los parámetros de la red. Para un punto determinado, el gradiente de una función indica la dirección en la cual la función crece mas rápidamente.

Si se parte de un punto  $\mathbf{x}_0$ , y después se actualizan los valores con la ecuación:

$$\Delta \mathbf{x}_k = \alpha_k \mathbf{p}_k \quad (3.2)$$

donde  $\mathbf{p}_k$  es un vector de búsqueda del valor óptimo para cada iteración, y  $\alpha_k$  es la constante de entrenamiento, la cual determina el tamaño del paso en la búsqueda; cuanto mas grande es el valor  $\alpha_k$  menos tiempo toma encontrar el óptimo, pero el proceso puede volverse inestable.

Lo que se busca es que  $\mathbf{F}(\mathbf{x}_{k+1}) < \mathbf{F}(\mathbf{x}_k)$ , aproximando  $\mathbf{F}(\mathbf{x}_{k+1})$  con serie de Taylor con los términos de 1er orden se tiene:

$$\mathbf{F}(\mathbf{x}_{k+1}) = \mathbf{F}(\mathbf{x}_k + \Delta \mathbf{x}_k) = \mathbf{F}(\mathbf{x}_k) + \nabla \mathbf{F}(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \Delta \mathbf{x}_k \quad (3.3)$$

Para que  $F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$ , el segundo término de la parte derecha de la ecuación debe ser negativa, esto es:

$$\nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \Delta \mathbf{x}_k = \alpha_k \nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k < \mathbf{0} \quad (3.4)$$

y como  $\alpha_k > 0$ :

$$\nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \mathbf{p}_k < \mathbf{0} \quad (3.5)$$

cualquier vector  $\mathbf{p}_k$  que satisfaga la ecuación anterior es llamada *dirección descendente*, la función decrece si se da un paso en esa dirección, si se supone que la magnitud de  $\mathbf{p}_k$  es unitaria, esto corresponde al calculo de la derivada de la función  $F(\mathbf{x})$  en dirección de  $\mathbf{p}_k$ , la dirección en la cual la función decrece mas rápidamente es en la dirección opuesta a  $\nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k}$ , entonces:

$$\mathbf{p}_k = -\nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k} \quad (3.6)$$

De donde se deduce el método del *gradiente descendente*:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k \quad (3.7)$$

donde  $\mathbf{g}_k^T = \nabla F(\mathbf{x}) \Big|_{\mathbf{x}=\mathbf{x}_k}$  es el gradiente evaluado en el punto anterior, normalmente se utiliza  $\alpha_k$  constante para todo el entrenamiento, pero el proceso resulta ser lento.

### 3.3.6.2. Variantes del Algoritmo de Aprendizaje

La implementación más simple del algoritmo es determinar el gradiente de la función de desempeño (error como función de los parámetros de la red) y actualizar los pesos con la ecuación  $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k$  pero en la práctica el proceso es lento e ineficiente, por lo que se han desarrollado diversas técnicas para entrenamiento más rápido.

La variante más común del algoritmo es conocida como gradiente descendente con momento, en donde los pesos son actualizados con base en la información del gradiente además de la tendencia del error, esto evita el estancamiento en un mínimo local. La implementación de esta variante se logra tomando en cuenta una fracción del gradiente anterior y otra del gradiente actual.



En la implementación básica del gradiente descendente, la constante de entrenamiento determina el tamaño del cambio para la actualización de los parámetros, sin embargo si el paso es muy pequeño se tiene un entrenamiento lento, pero si el paso es muy grande se puede alcanzar inestabilidad en el aprendizaje.

Existe un valor óptimo de dicha constante, pero realmente no es práctico determinarla, de hecho, el valor óptimo de la constante de aprendizaje cambia durante todo el proceso, por lo que se puede intuir que el entrenamiento puede mejorarse adaptando la constante de aprendizaje al proceso, esto es haciendo que la constante se incremente mientras se encuentre en una región estable, y se disminuya si se encuentra en una región inestable o si se esta cercano al punto mínimo.

La función sigmoideal es usualmente utilizada en las redes neuronales, el problema con ellas es que cuando las entradas crecen, la pendiente de la función sigmoideal tiende a cero, como el proceso de optimización depende de la magnitud del gradiente, el aprendizaje no es uniforme, y si los pesos tienen magnitud grande, el proceso puede volverse inestable, por lo que otra alternativa es tomar en cuenta solo la dirección del gradiente y no la magnitud, así se tienen actualizaciones uniformes en los pesos.

Aunque el gradiente indica hacia donde decrece más rápidamente el error, esto no significa que sea la dirección en la que converge mas rápidamente, los algoritmos de gradiente conjugado calculan el gradiente y luego realizan una búsqueda a partir de esa dirección y así determinar la dirección óptima para actualizar los pesos.

$$\mathbf{p}_k = -\mathbf{g}_k + \beta_k \mathbf{p}_{k-1} \quad (3.8)$$

Los algoritmos de gradiente conjugado se distinguen por la manera en que calculan  $\beta_k$ , para el algoritmo de Fletcher-Reeves es el cociente del cuadrado de la norma del gradiente actual y el cuadrado de la norma del gradiente anterior:

$$\beta_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \quad (3.9)$$

En la actualización de Polak-Ribière la constante es calculada con el producto interno del cambio del gradiente anterior con el gradiente actual dividido por el cuadrado de la norma del gradiente anterior:

$$\beta_k = \frac{\Delta \mathbf{g}_{k-1}^T \mathbf{g}_k}{\mathbf{g}_{k-1}^T \mathbf{g}_{k-1}} \quad (3.10)$$

Para todos los algoritmos de gradiente conjugado, la dirección de búsqueda se inicializa periódicamente, el punto de reinicio se da comúnmente cuando el número de iteraciones es igual al número de parámetros de la red, pero hay otras maneras más eficientes de establecer cuando reiniciar la dirección de búsqueda para mejorar el entrenamiento, Powell propuso un método basado en una versión propuesta de Beale, que establece el punto de reinicio cuando hay una muy pequeña ortogonalidad entre el gradiente actual y el anterior:

$$|\mathbf{g}_{k-1}^T \mathbf{g}_k| \geq 0.2 \|\mathbf{g}_k\|^2 \quad (3.11)$$

El método de Newton es una opción para hacer más eficiente los algoritmos del gradiente conjugado, el método básico de Newton es

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1} \mathbf{g}_k \quad (3.12)$$

Donde  $\mathbf{A}_k$  es la matriz Hessiana de la ecuación de desempeño, de todos los valores actuales de pesos, normalmente, el método de Newton converge más rápidamente que los métodos de gradiente conjugado; el cálculo de  $\mathbf{A}_k$  es complejo, pero existe una clase de métodos desarrollados y publicados satisfactoriamente por Broyden, Fletcher, Goldfarb y Shanno (BFGS).

Como los métodos cuasinewtonianos, el de Levenberg-Marquardt trata de calcular la matriz Hessiana si calcular dobles derivadas, cuando la función de desempeño tiene una forma cuadrática, la matriz Hessiana puede aproximarse como:

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} \quad (3.13)$$

Y el gradiente como:

$$\mathbf{g} = \mathbf{J}^T \mathbf{e} \quad (3.14)$$

$\mathbf{J}$  es el Jacobiano que contiene las primeras derivadas de los errores de la red con respecto a los pesos, y  $\mathbf{e}$  es el vector de errores; el método de Levenberg-Marquardt usa este Jacobiano para actualizar los pesos de acuerdo a la ecuación cuasi-newtoniana:

$$\Delta \mathbf{w}_{k+1} = -\left(\mathbf{H}(\mathbf{w}_k) + \mu \text{diag}(\mathbf{H}(\mathbf{w}_k))\right)^{-1} \mathbf{J}^T(\mathbf{w}_k) \mathbf{e}(\mathbf{w}_k) \quad (3.15)$$

Cuando el escalar  $\mu$  es cero, entonces la ecuación es justo la ecuación del método de Newton (con la aproximación de la matriz Hessiana), cuando  $\mu$  toma un valor muy grande, la ecuación se aproxima a la ecuación del gradiente descendente con constante de aprendizaje pequeña. El método de Newton es más rápido y más preciso en un mínimo del error, así que el objetivo es llevarlo hacia el método de

Newton lo antes posible, entonces  $\mu$  es reducido para cada paso correcto (función de desempeño), y solo se incrementa cuando un paso tentativo incrementa el valor en la función de desempeño (error como función de los parámetros de la RN); de esta manera la función de desempeño se reduce siempre a cada iteración.

### 3.3.6.3. Retro-propagación

La retro-propagación, es un método eficiente y exacto para calcular las derivadas de una función de un elevado número de variables, la primer aplicación práctica fue presentada por Paul Werbos en 1974 en la estimación de un modelo dinámico el nacionalismo y las comunicaciones sociales; sin embargo, hoy en día las aplicaciones se extienden a diversos campos como el reconocimiento de patrones y el diagnostico de fallas.

El término retro-propagación se refiere a la manera en la que el gradiente es calculado para redes multicapa no lineales; este método se basa en la generalización de la regla delta (presentada por Widrow y Hoff en 1960 mejorando el algoritmo del perceptrón) y permite el entrenamiento de redes neuronales multicapa con funciones continuas y diferenciables.

La retro-propagación parte del algoritmo del gradiente descendente, el problema es que al desarrollar éste último para redes multicapa, el error en una capa no es una función explícita de los parámetros propios de la capa, ya que no se puede conocer la salida deseada de las neuronas de las capas ocultas. Únicamente es posible determinar el error en la capa de salida, y por la regla de la cadena se puede definir la sensibilidad del cambio del error en cada capa al cambio en la salida de la capa anterior

La sensibilidad de la capa  $k$  es calculada a partir de la sensibilidad de la capa  $k+1$ , nuevamente por la regla de la cadena, por lo que el error es propagado desde la capa de salida (hacia atrás) hasta la primer capa, y los pesos de cada capa se actualizan con base a la regla de aprendizaje, el proceso es resumido en la figura 3.15.

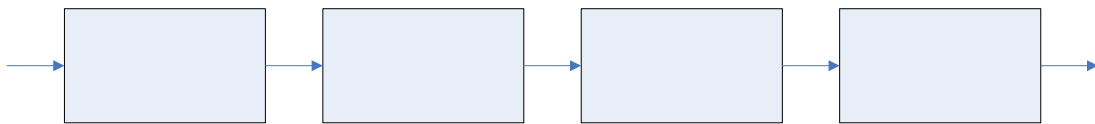


Figura 3.15 – Actualización de los pesos de la RN por retro-propagación

# 4 | Pruebas y Resultados

## 4.1. Introducción

Anticipar problemas rápida y oportunamente en el SEP, permite realizar acciones preventivas para que el sistema continúe operando adecuadamente; el linealizar el modelo alrededor de un punto en particular y calcular los eigenvalores, no es un procedimiento sencillo, y por supuesto, la dificultad aumenta con la complejidad de la configuración de la red; para éste, como para muchos otros problemas en la industria actual, se buscan soluciones simples y prácticas, que ayuden a hacer más eficientes y seguros los procesos.

En la teoría de control clásico, se necesitan modelos que representen adecuadamente los sistemas reales, desafortunadamente, los sistemas de control basados en modelos son muy rígidos, y para que se desempeñen de la manera en que han sido diseñados, se requieren modelos muy exactos; además, un problema común de implementación, es que las condiciones reales de operación, debido a varias causas, no son ideales como se plantean en las simulaciones, por lo que siempre hay errores que en general, afectan los resultados de manera importante.

Una propiedad de los sistemas de control inteligente, es que éstos no necesitan modelos para obtener buenos resultados, en cambio se basan en reglas u observaciones. En un tipo de reconocimiento de patrones se puede identificar o clasificar objetos, sin tener una descripción explícita de las clases a las que pertenecen, solo con base en observaciones de rasgos comunes, ésta característica permite integrar el reconocimiento de patrones a los sistemas de control inteligente.

Entre otras técnicas para el reconocimiento de patrones, las RN poseen la capacidad de generalizar objetos no observados, además, las RN son no lineales inherentemente, por lo que son utilizadas de manera exitosa en la solución de problemas altamente no lineales; la estabilidad del SEP tiene comportamiento no lineal, por lo que el reconocimiento de patrones con RN es una buena alternativa para la determinación de la misma.

## 4.2. Descripción del Sistema

El sistema de prueba es el sistema máquina síncrona bus-infinito (figura 4. 1). En este trabajo en particular, se considera que la máquina síncrona tiene un enlace de 2 líneas en paralelo al bus infinito, las cuales pueden o no, estar conectadas al mismo tiempo en el sistema, lo que significa que la impedancia de la línea puede tomar valores  $Z$  (1L) o  $Z/2$  (2L).

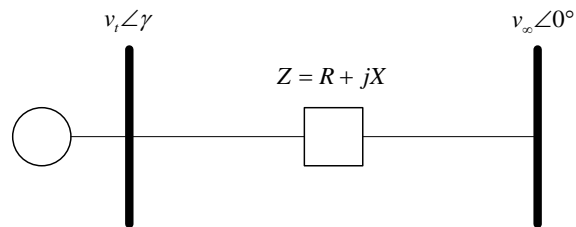


Figura 4. 1 – Sistema máquina bus infinito

Se analizan dos casos con valores distintos de impedancia de línea, identificados como *línea corta (LC)* y *línea larga (LL)*, donde se tienen impedancias equivalentes  $Z = 0.024 + j0.115$  y  $Z = 0.12 + j1.1$  respectivamente. Para la máquina síncrona se utiliza el modelo linealizado de 4<sup>o</sup> orden (figura 2. 2 y ecuación 2. 1) y los parámetros de la máquina se dan en el apéndice A.

### 4.2.1. Límite de estabilidad para línea corta

En las figuras 4. 2a y 4. 2b se muestran las gráficas de los límites de estabilidad a pequeños disturbios del sistema, obtenidos por análisis de eigenvalores; la manera en la que se obtienen las curvas mostradas es cambiando el punto de operación del sistema (variando la potencia, la magnitud en el voltaje del bus de la máquina y la magnitud del voltaje en el bus infinito).

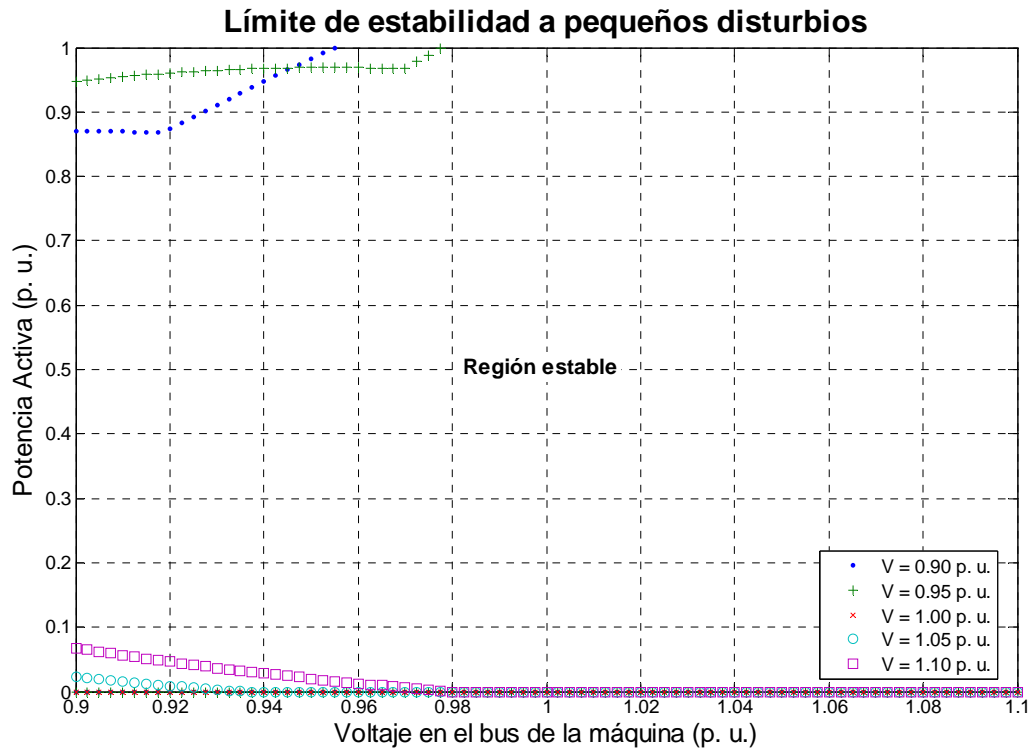


Figura 4. 2a – Límite de estabilidad determinado por análisis de eigenvalores (LC - 1L)

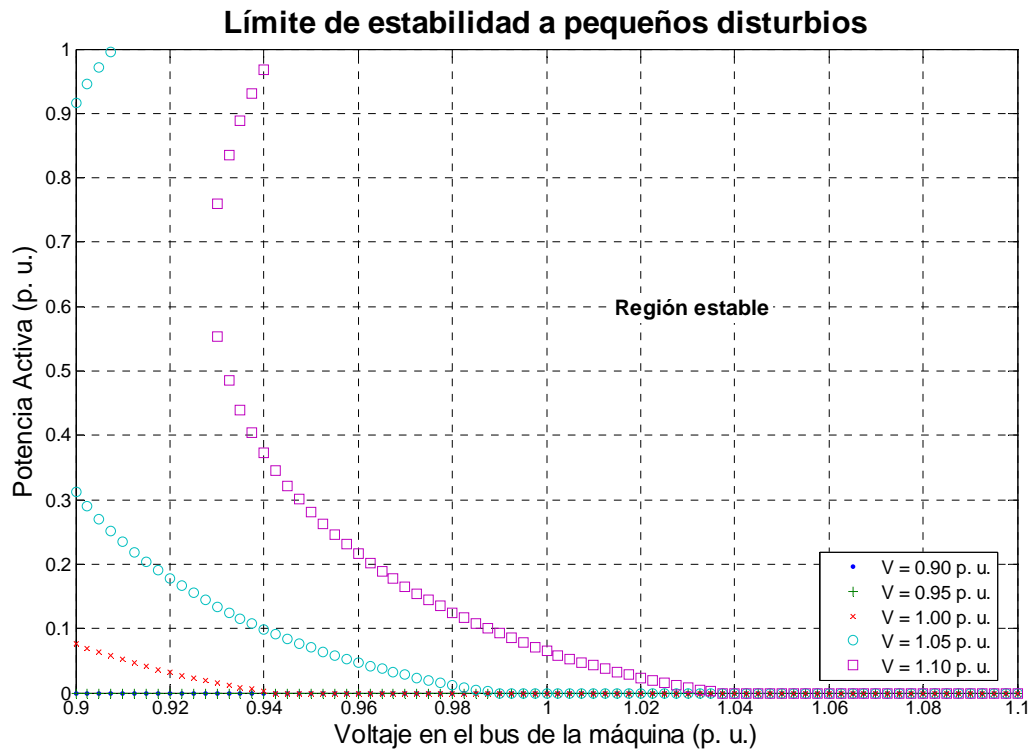


Figura 4. 2b – Límite de estabilidad determinado por análisis de eigenvalores (LC - 2L)

Como se puede observar, la región de estabilidad con una línea de enlace conectada, o con dos en paralelo, es completamente diferente.

#### 4.2.2. Límite de estabilidad para línea larga

Éste caso es el de un sistema con operación mas restringida; debido a que la impedancia de la línea es mayor, la estabilidad del sistema se ve comprometida; para obtener las curvas mostradas en las figuras 4. 3a y 4. 3b, se sigue el mismo procedimiento que para las figuras 4. 2a y 4. 2b.

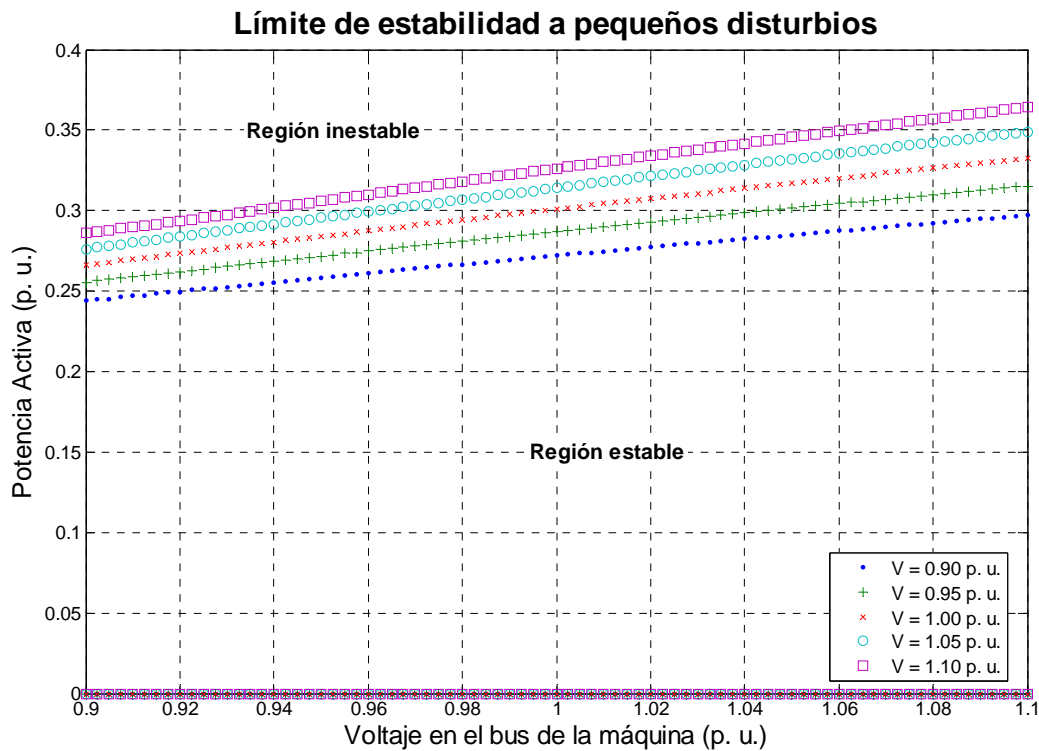


Figura 4. 3a – Límite de estabilidad determinado por análisis de eigenvalores (LL - 1L)

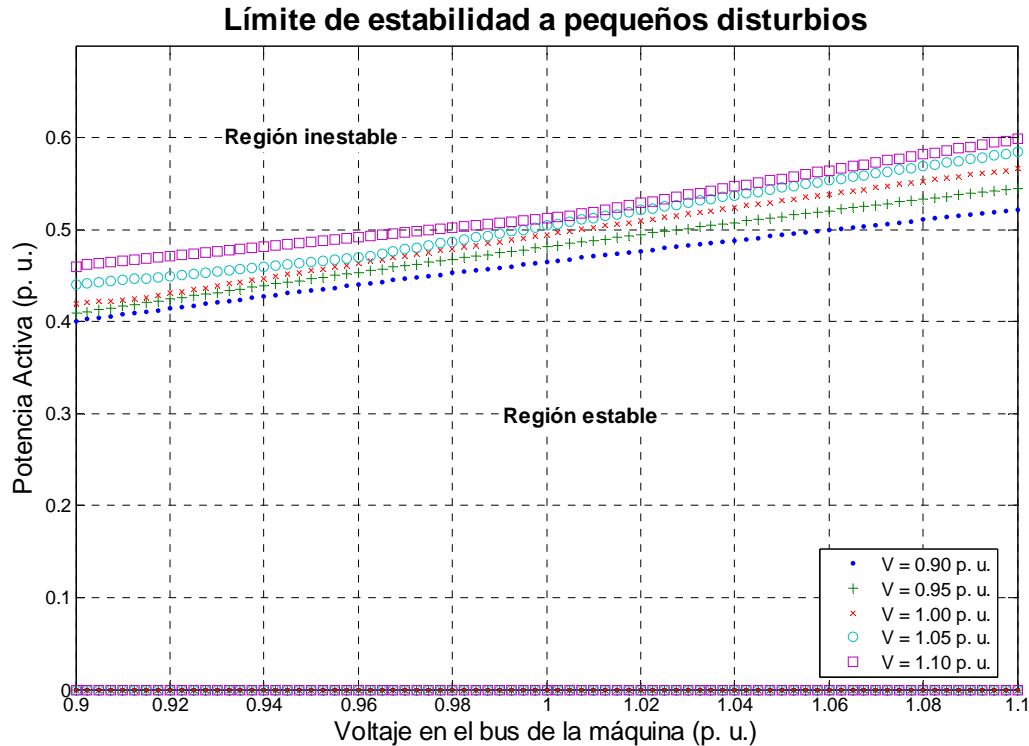


Figura 4. 3b – Límite de estabilidad determinado por análisis de eigenvalores (LL - 2L)

Como se puede verificar de las figuras 4. 3a y 4. 3b, las curvas de los límites de estabilidad no son tan complejas como para el caso de la línea corta, y tienen un comportamiento casi lineal.

### 4.3. Descripción de la RN

El objetivo es entrenar una RN, para que con ésta, se determine si el sistema es estable o no ante disturbios pequeños, sin tener que calcular los eigenvalores cada vez; las entradas a la RN son las condiciones de operación del sistema en un momento determinado:

- ➔ Potencia activa de la máquina  $P$
- ➔ Magnitud del voltaje en el bus de la máquina  $|V_i|$
- ➔ Magnitud del voltaje en el bus infinito  $|V_\infty|$

Como las curvas que delimitan las regiones de estabilidad son muy distintas para una o dos líneas de enlace, se agrega una cuarta entrada a la RN, con la información de si es una o son las dos líneas que están conectadas en el momento del análisis de estabilidad, con esto se le da más robustez a la identificación de la estabilidad.



La salida de la RN será un indicador binario de estabilidad (“1” si estable, “0” si inestable). Se espera que una vez entrenada, la RN sea capaz de identificar los mismos límites de estabilidad mostrados en las figuras 4. 2a a 4. 3b.

Tabla 4. 1 – Rango de valores de las entradas de la RN

<i>Entrada</i>	<i>Tipo</i>	<i>Rango</i>
Magnitud de voltaje en bus infinito	Numérico	0.9 - 1.1 p. u.
Magnitud de voltaje en bus de máquina	Numérico	0.9 - 1.1 p. u.
Potencia activa de la máquina	Numérico	0 - 1.0 p. u.
Número de líneas en paralelo	Binario	0, 1

Los datos de entrenamiento para la RN se obtienen mediante la simulación del modelo linealizado para estudios de oscilaciones de baja frecuencia, variando el punto de operación y determinando la estabilidad en ese punto, la obtención de estos datos de entrenamiento se hace con un programa en FORTRAN 90 y el entrenamiento de la RN se hace en MATLAB® R14.

Como se trabaja en un sistema p. u., la normalización de los datos no es necesaria, pues los datos ya están normalizados y tienen magnitudes similares; se hacen dos entrenamientos para cada caso, en uno de ellos los datos obtenidos de la simulación se utilizan como datos de entrenamiento, como provienen de simulaciones, entonces no es necesario realizar un análisis previo al entrenamiento. En otro entrenamiento, los datos de la simulación se contaminarán con distintos niveles de ruido (con una secuencia de números aleatorios de distribución normal), para el entrenamiento con estos datos si es necesario realizar un análisis previo de los datos, con el fin de detectar anomalías en ellos.

La determinación de la topología de la RN se hace con los datos obtenidos de la simulación, con lo que es de esperar que se tenga un buen resultado y un buen punto de partida para encontrar la estructura de la RN que de mejores resultados cuando se entrene la RN con datos con ruido.

El sistema debe ser sencillo y los requerimientos para la RN son:

- ✕ Capa de entrada con 4 entradas analógicas
- ✕ Una unidad para la capa de salida binaria
- ✕ Entrenamiento supervisado

La sugerencia es utilizar una topología sencilla como inicio en la búsqueda del sistema, la topología básica que satisface las condiciones, sería una RN de una capa, que constaría de una neurona con función de activación escalón. Pero para una RN con estas características las limitaciones propias no le permiten resolver un

problema como éste, ya que solo es capaz de clasificar objetos separados linealmente (perceptrón).

La alternativa es entonces agregar al menos una capa, pero el entrenamiento de los perceptrones multicapa es complejo y el sistema se vuelve menos sencillo, es por eso que la opción sencilla más viable es utilizar un RN Retro-propagación multicapa, la capa oculta tendría función sigmoïdal, ya que si se utilizara función lineal, sería el equivalente a una sola unidad (por linealidad), y la capa de salida puede tener función lineal o sigmoïdal de activación, lo mas sencillo es utilizar la función lineal, pero no hay restricción al respecto.

#### **4.4. Entrenamiento de la RN con datos obtenidos de la simulación**

En esta sección se describe el entrenamiento con los datos provenientes de la simulación tal cual, por esto, no es necesario hacer un análisis previo de ellos; el entrenamiento se realiza por separado para los casos LC y LL.

##### **4.4.1. Determinación de la topología**

La RN retro-propagación, que se recomienda en la bibliografía como primera opción, es con función sigmoïdal en las capas ocultas y función lineal en la de salida, en la sección 3.2.5 se describieron las etapas del diseño de RN, el principal problema es la determinación del número de neuronas en las capas ocultas e incluso el número de de capas ocultas.

###### **4.4.1.1. Línea corta**

Haciendo pruebas se encontró que la RN con función de activación lineal en la capa de salida no da buenos resultados, de hecho, en los resultados se observó que una RN de 2 capas con función sigmoïdal en la capa oculta y función lineal en la de salida no es capaz de identificar los datos presentados, se decidió aumentar una capa oculta con función sigmoïdal y el resultado fue que solo una unidad de la segunda capa oculta es trascendental (figuras 4. 4a, 4. 4b y 4. 4c).

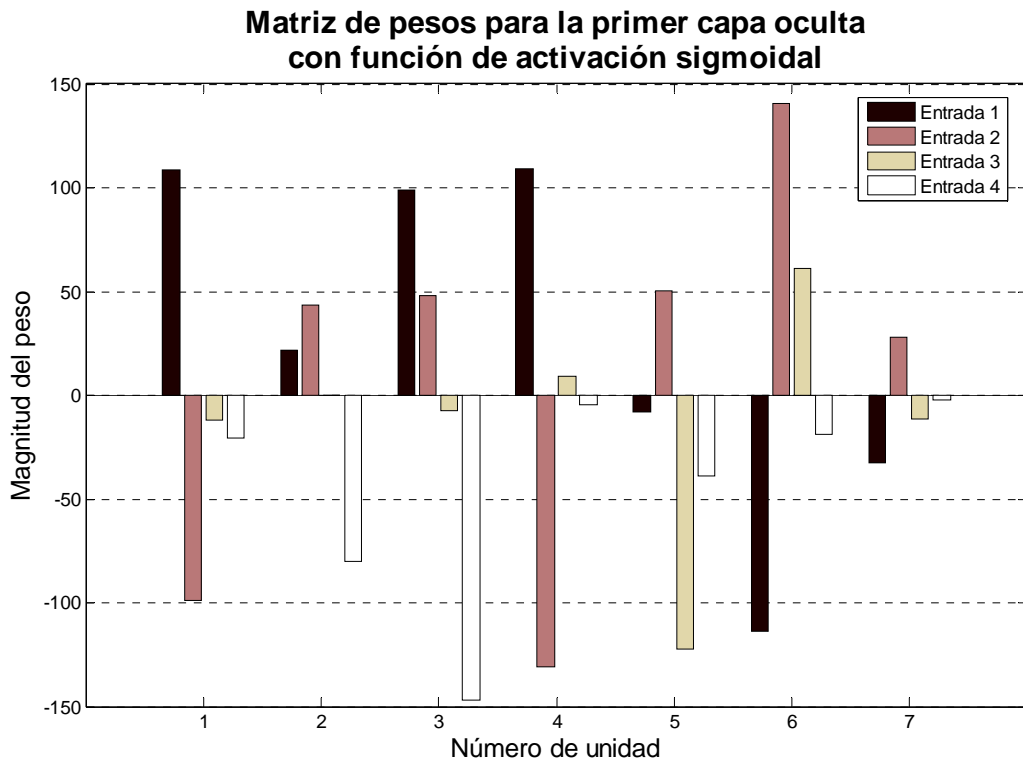


Fig. 4. 4a – Parámetros de la primer capa oculta en la primer prueba para determinar topología (LC)

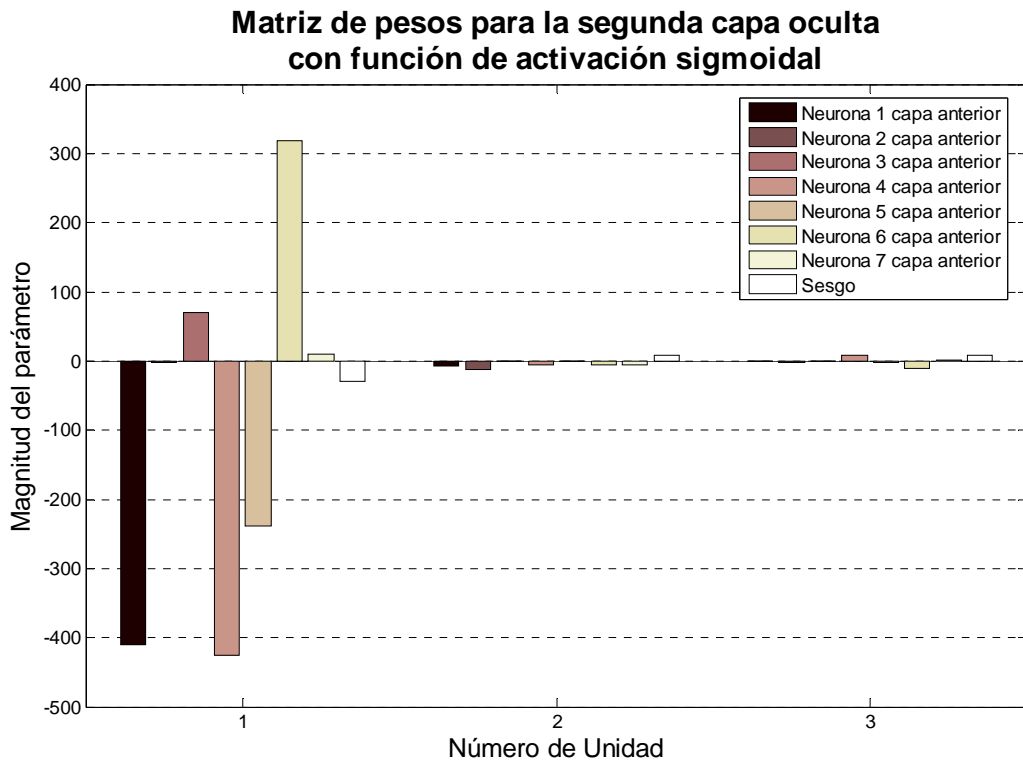


Fig. 4. 4b – Parámetros de la segunda capa oculta en la primer prueba para determinar topología (LC)

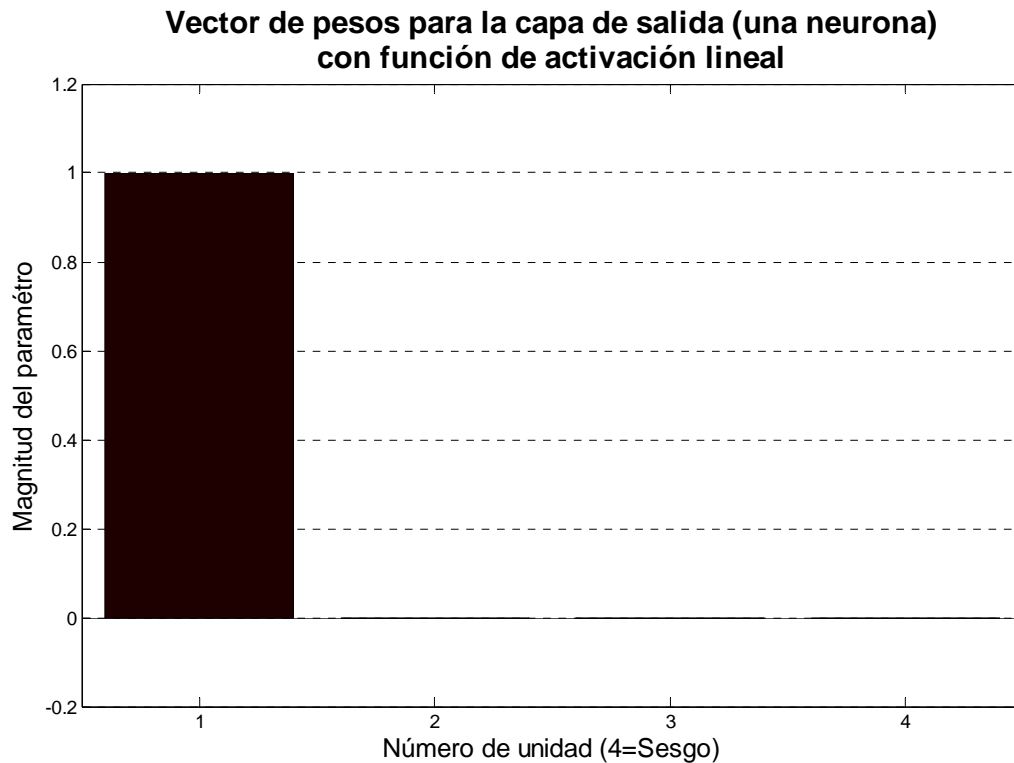


Fig. 4. 4c – Parámetros de la capa de salida en la primer prueba para determinar topología (LC)

En la figura 4. 4c se muestra que el peso que conecta la 1ª neurona de la 2ª capa con la única neurona en la capa de salida es de magnitud 1, y los otras 2 así como el sesgo de la neurona de salida son de magnitud 0; entonces se eliminó la capa de salida con función lineal y se hizo la búsqueda del número de unidades para la capa oculta de la RN con función de activación sigmoideal para ambas capas, se hicieron varias pruebas, la más representativa se resume en la tabla 4. 2 y las figuras 4. 5, 4. 6a y 4. 6b.

Tabla 4. 2 – Parámetros para la búsqueda de la topología (LC)

Parámetro	Valor
Error de entrenamiento objetivo	1e-6
Rango de búsqueda de neuronas en capa oculta	6 → 12
Número de reentrenamientos	19

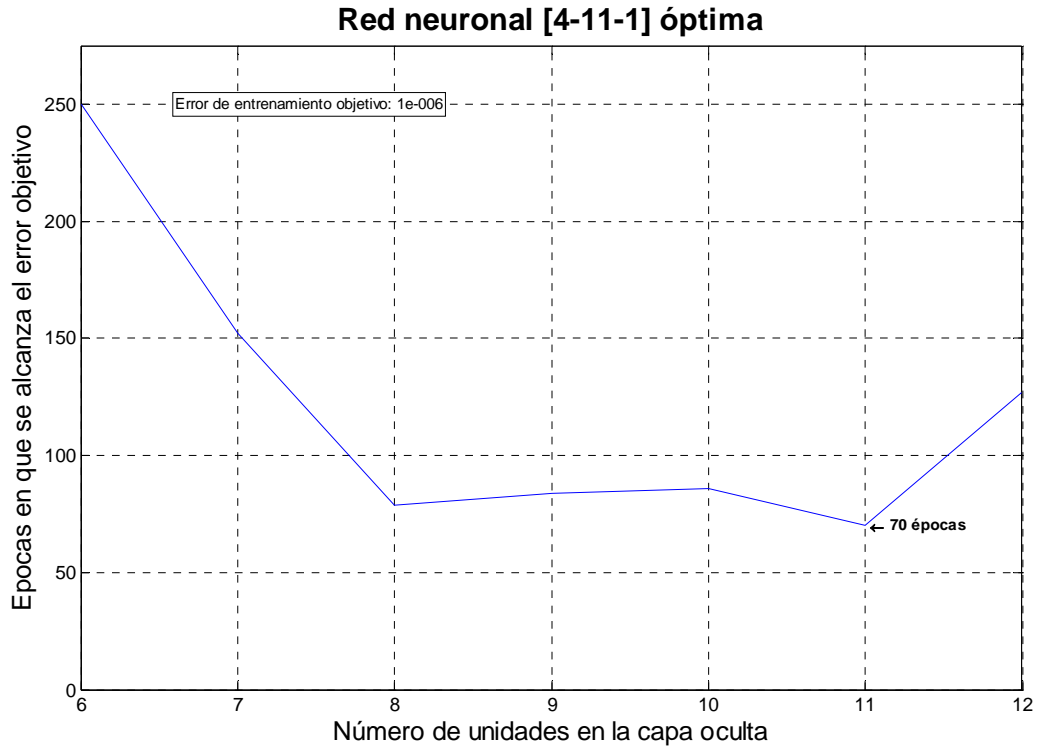


Figura 4. 5 – Resultado de la búsqueda de la topología (LC)

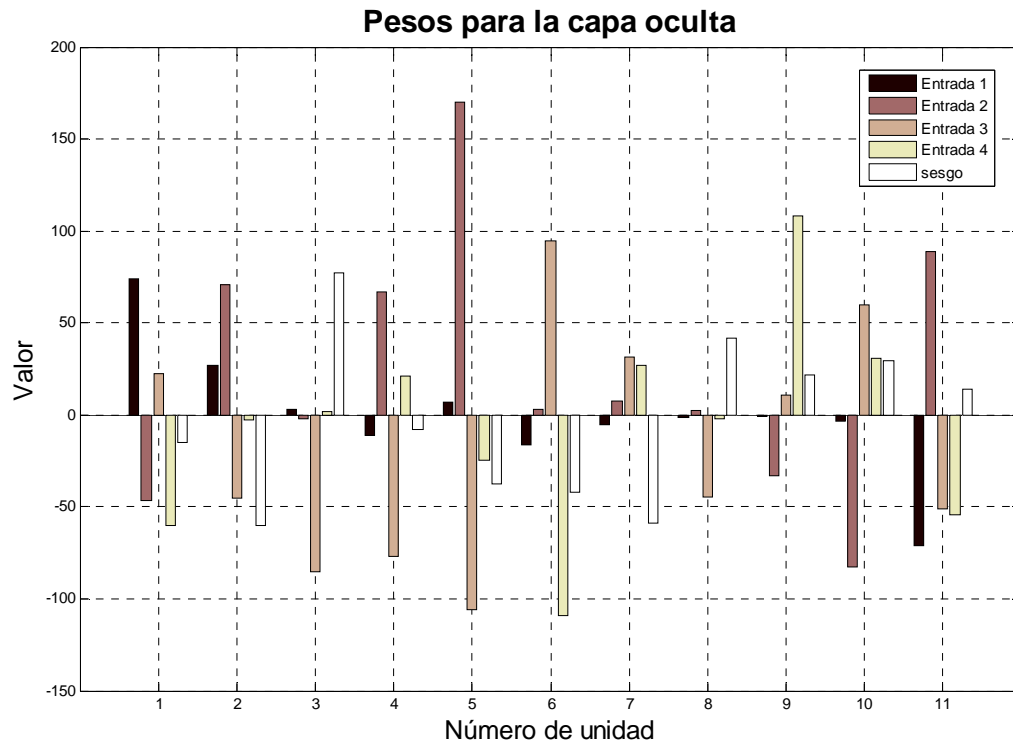


Figura 4. 6a – Pesos finales para la capa oculta (LC - entrenamiento sin ruido)

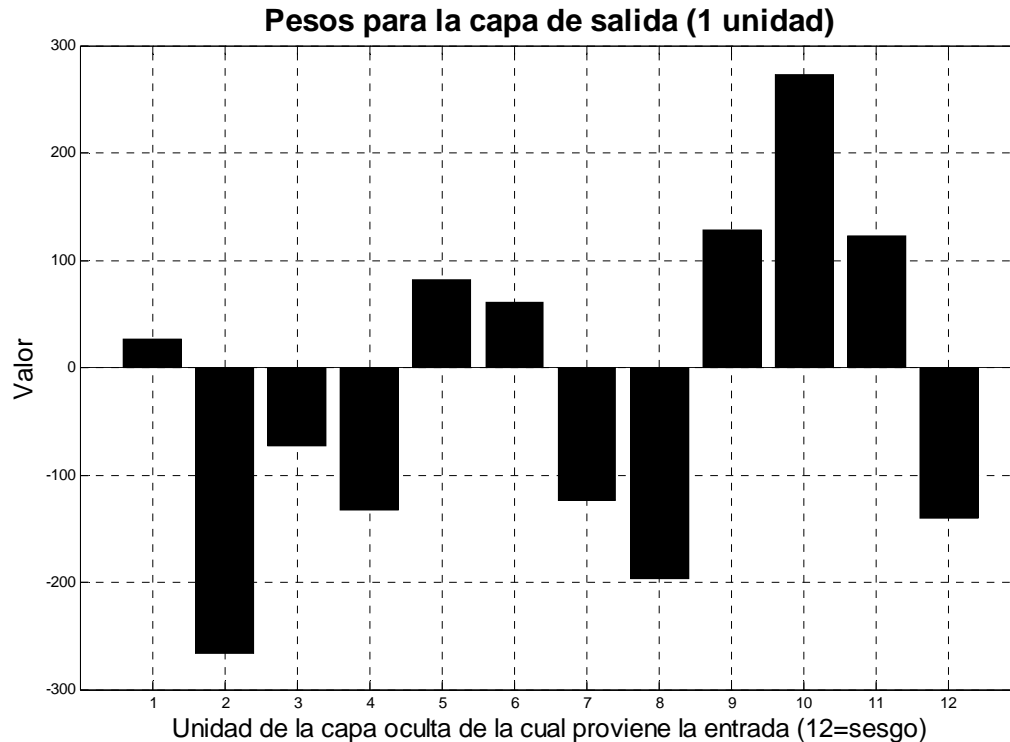


Figura 4. 6b – Pesos finales para la capa de salida (LC – entrenamiento sin ruido)

Luego de varias pruebas de búsqueda de topología se determinó que la mejor topología es con 11 unidades en la capa oculta, con este resultado se espera que la RN sea capaz de identificar adecuadamente los límites de estabilidad obtenidos del análisis de eigenvalores del modelo linealizado.

#### 4.4.1.2. Línea larga

Observando las gráficas de los límites de estabilidad para este caso (figuras 4. 3a y 4. 3b) y de los resultados previos, se puede intuir que una RN con menos unidades sería capaz de identificar adecuadamente los límites, se comienza la búsqueda con una RN de tres capas con función sigmoial en la capa oculta y lineal en la de salida con los parámetros de búsqueda mostrados en la tabla 4. 3.

Tabla 4. 3 – Parámetros para la búsqueda de la topología (LL)

Parámetro	Valor
Épocas	250
Rango de búsqueda de neuronas en capa oculta	2 → 8
Número de reentrenamientos	19

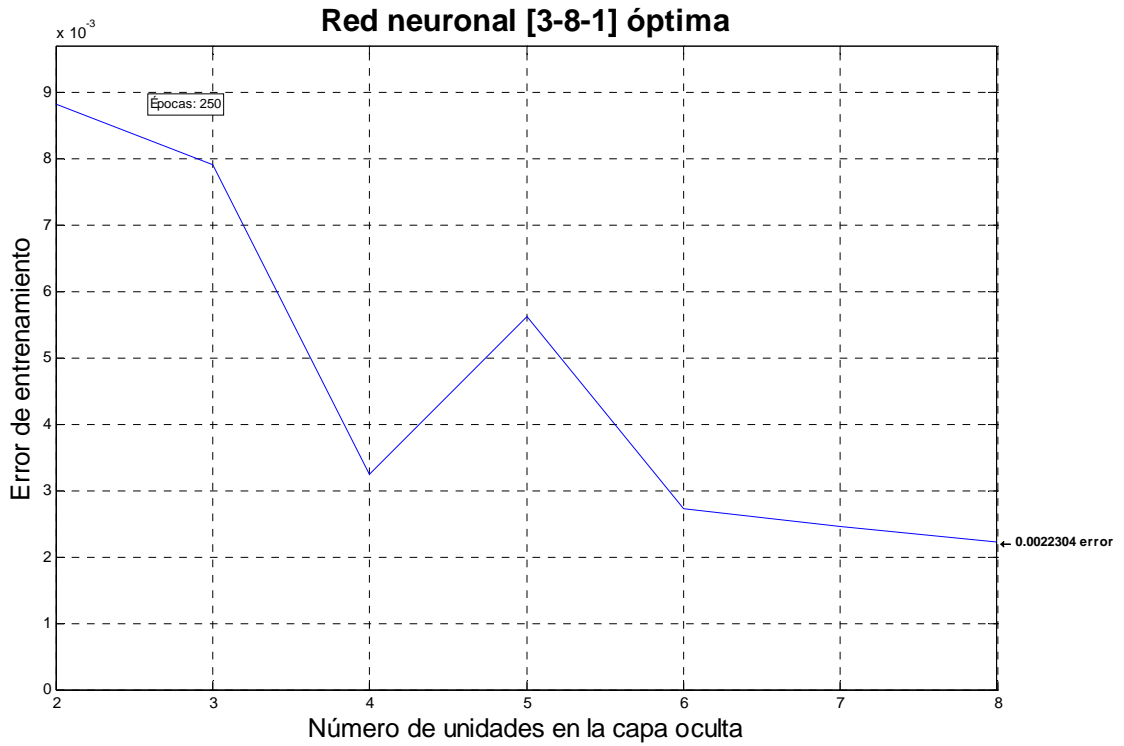


Figura 4. 7 – Resultado de la búsqueda de la topología (LL)

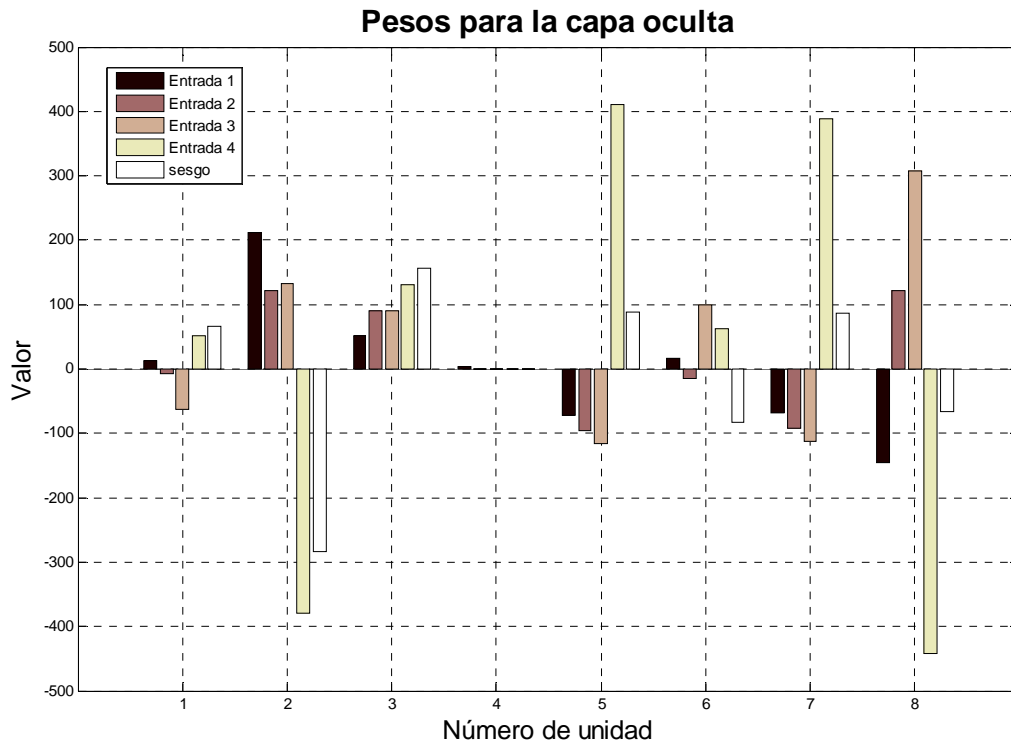


Figura 4. 8a – Pesos de la capa oculta en la primer prueba para determinar la topología (LL)

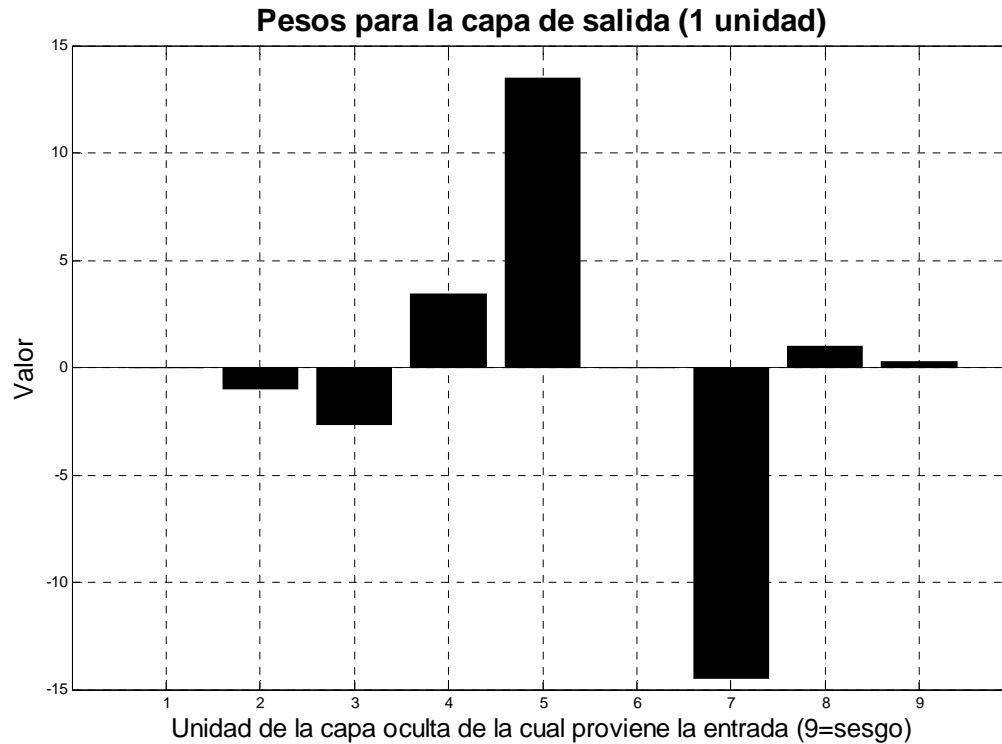


Figura 4. 8b – Pesos de la capa de salida en la primer prueba para determinar la topología (LL)

Luego de realizar la búsqueda se encontró que con 8 unidades en la capa oculta se alcanza el menor error de entrenamiento en 250 épocas, pero al analizar los resultados, se observa que 2 neuronas de la capa oculta son poco trascendentales, ya que el peso mediante el cual se conectan a la capa de salida es muy cercano a cero, por lo que se comienza un proceso de *poda* o reducción de unidades; mediante dicho proceso se concluye que la RN con mejores resultados es la de 3 neuronas en la capa oculta (figuras 4. 9a a 4. 11b).



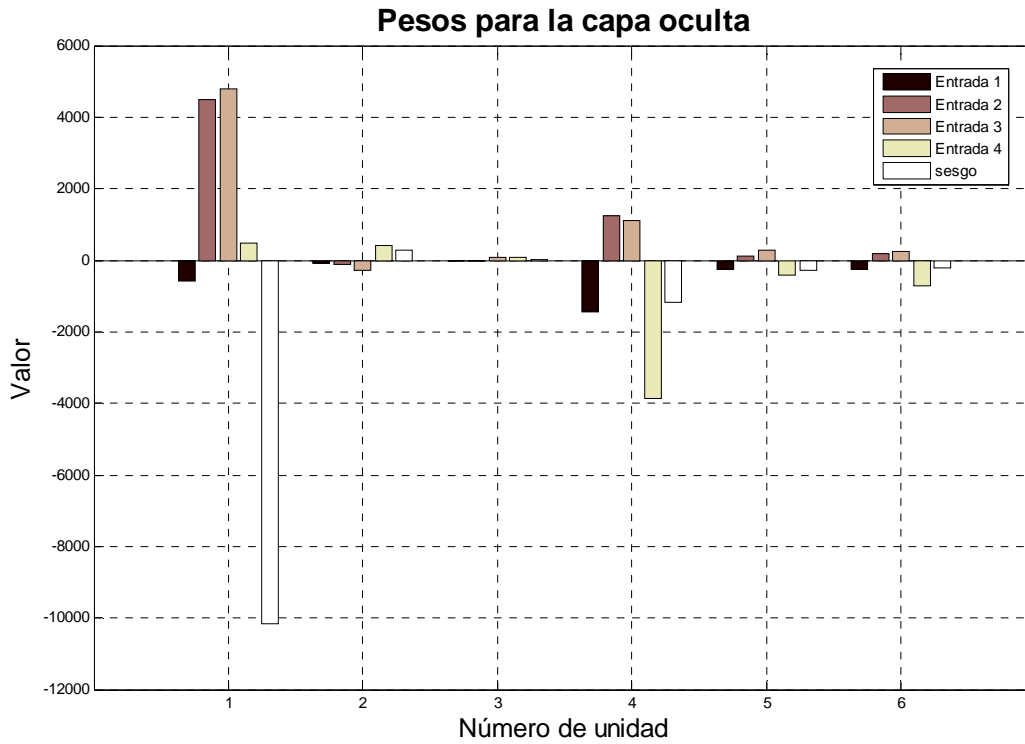


Figura 4. 9a – Pesos de la capa oculta en la segunda prueba para determinar la topología (LL)

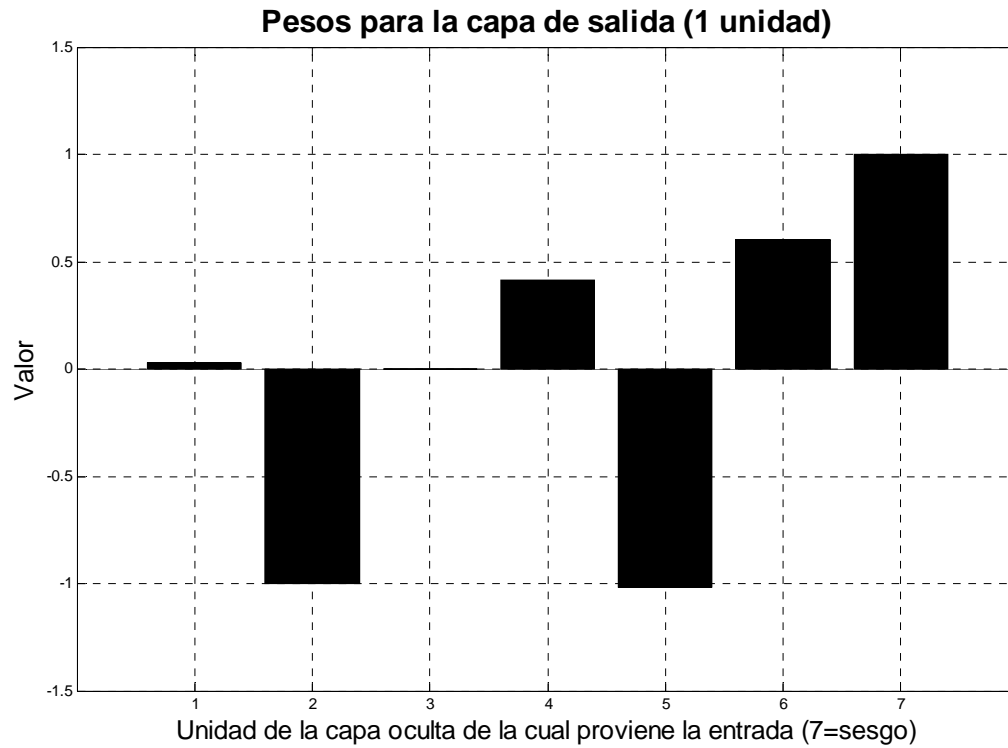


Figura 4. 9b – Pesos de la capa de salida en la segunda prueba para determinar la topología (LL)

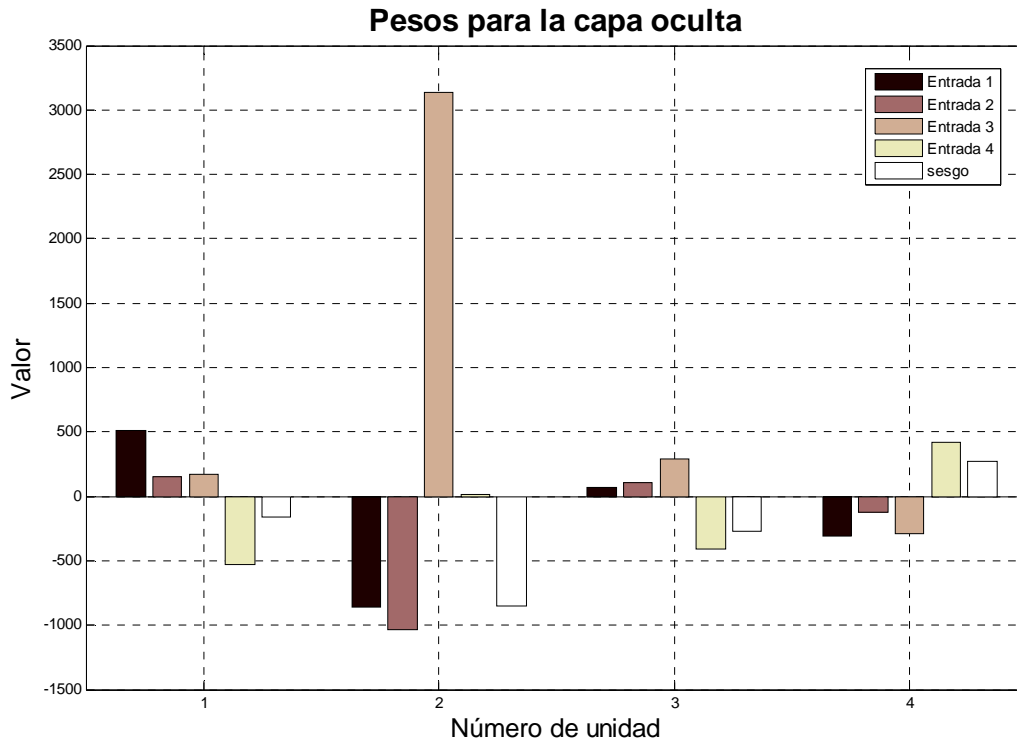


Figura 4. 10a – Pesos de la capa oculta en la tercer prueba para determinar la topología (LL)

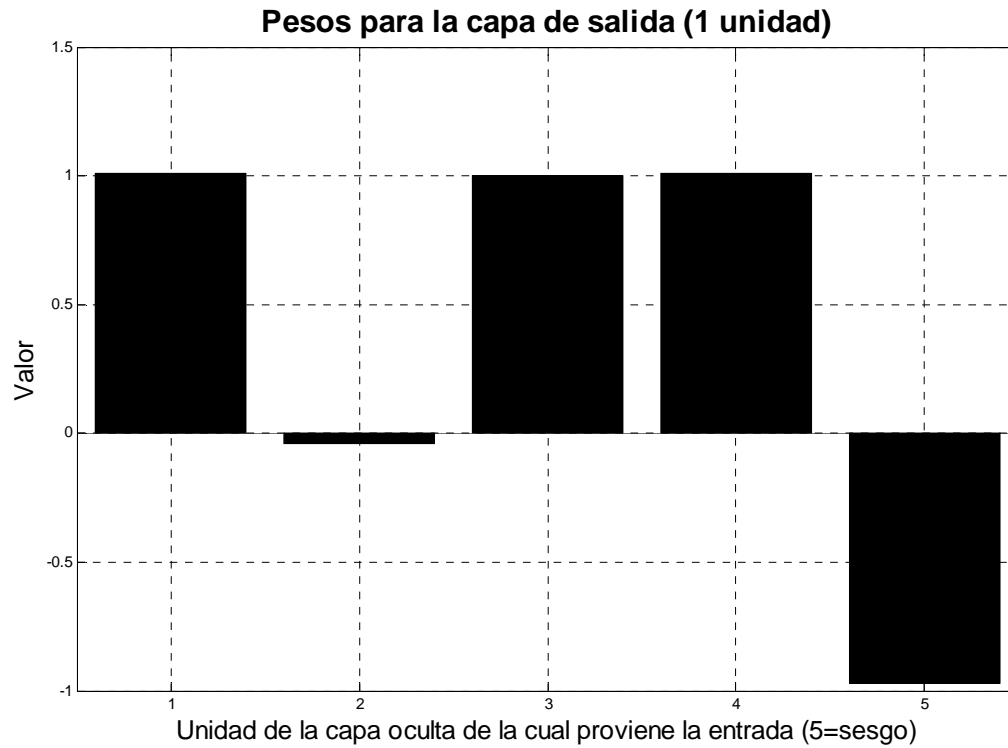


Figura 4. 10b – Pesos de la capa de salida en la tercer prueba para determinar la topología (LL)

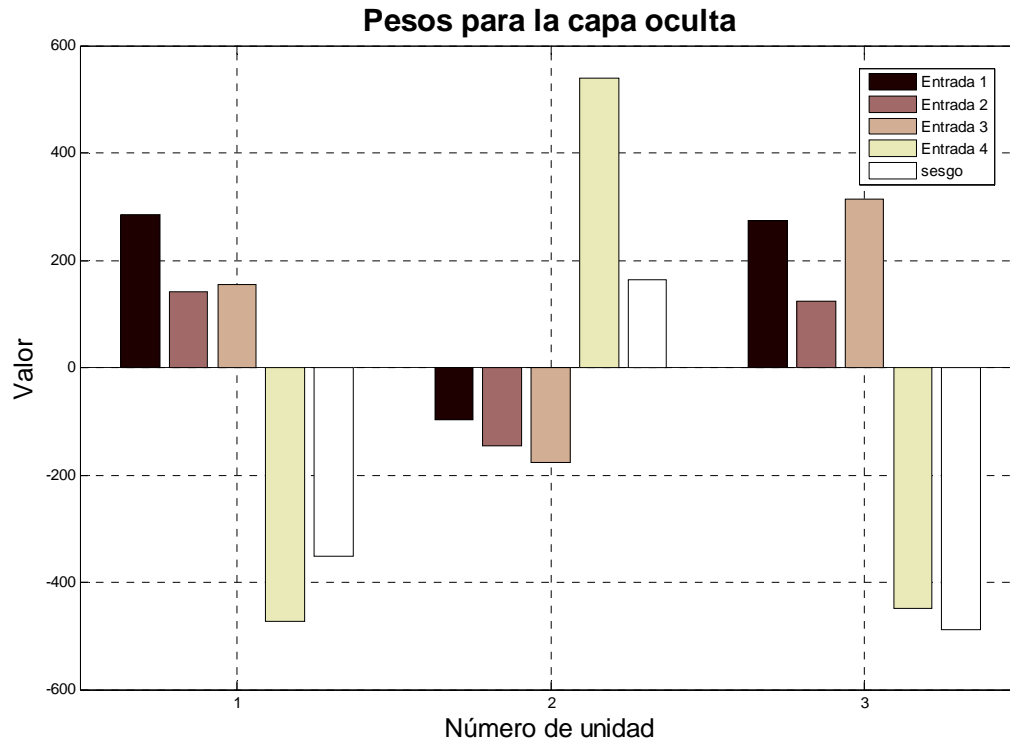


Figura 4. 11a – Pesos finales de la capa oculta (LL – entrenamiento sin ruido)

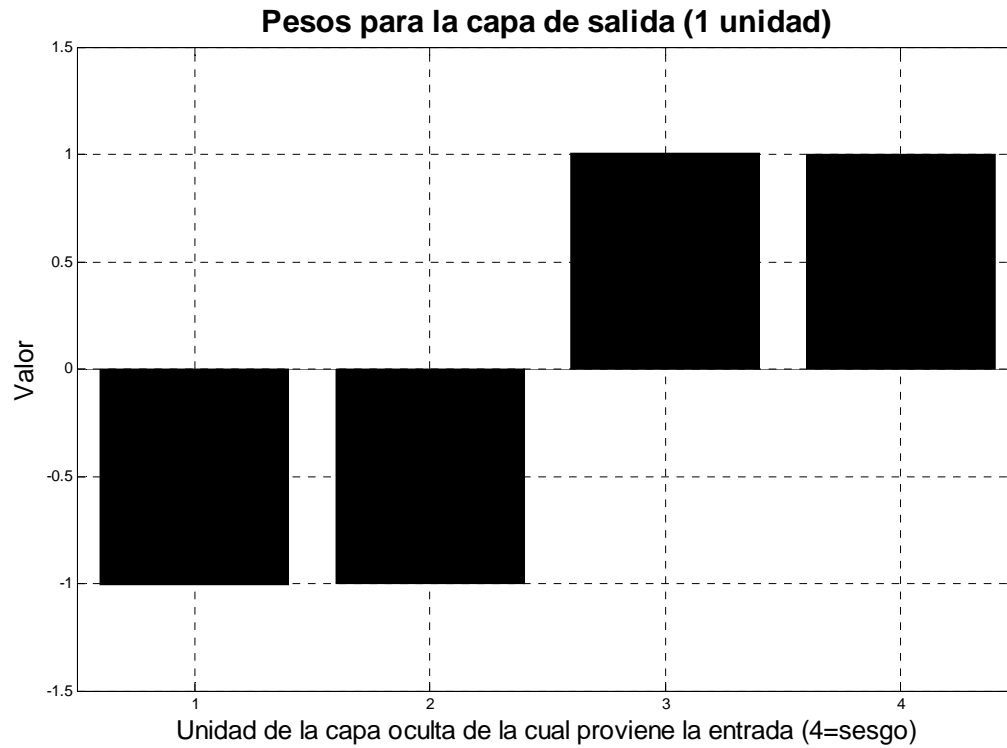


Figura 4. 11b – Pesos finales de la capa de salida (LL – entrenamiento sin ruido)

Los parámetros finales no están demasiado dispersos y en varias pruebas se obtienen resultados similares, la validación de los resultados se hace con la simulación del sistema y la obtención de su estabilidad mediante el análisis de eigenvalores.

#### *4.4.1.3. Análisis de resultados*

Para el caso de la LC, la figura 4. 5 muestra que con el criterio propuesto se encuentra un rango de 8 a 11 neuronas en la capa oculta que parecería ser buena elección, después de varias pruebas el rango cambia (se debe recordar que los resultados dependen de los valores aleatorios iniciales de los pesos), generalmente aumentando el valor de unidades a más de 11, por lo que ese fue el número que se eligió

Para el caso de la LL en varias pruebas se obtuvieron distintos valores, lo importante de este caso fue la detección de las unidades no trascendentales, y así se logró simplificar la topología de la RN, lo cual es en cierta manera congruente dada menor complejidad de las curvas de los límites respecto a las curvas para el caso LC.

### **4.4.2. Pruebas de identificación**

Como el sistema se diseña a partir de un modelo, entonces la evaluación del desempeño del reconocimiento de patrones se hará comparándolo directamente con resultados de la simulación, en problemas prácticos en los que no se tenga el modelo u algún sistema de referencia, la evaluación se hace con base en la experiencia y observaciones previas.

#### *4.4.2.1. Línea corta*

En las figuras 4. 12a y 4. 12b se observa que la RN ha aprendido correctamente los límites a partir de los datos de entrenamiento, este resultado era de esperarse pues la distribución de los datos de entrenamiento es uniforme y son suficientes, la siguiente prueba es ver que tan robusto es el sistema como para identificar los mismos datos pero contaminados con ruido aleatorio de distribución normal.

Se hacen varias pruebas las cuales consisten en contaminar una variable a la vez con distintos niveles de ruido para comparar los resultados y determinar cual es la variable que mas afecta al resultado y después contaminar las tres variables simultáneamente y comparar las diferencias, con esto se puede determinar cual es la variable que mas afecta a la identificación al ser contaminada con ruido.

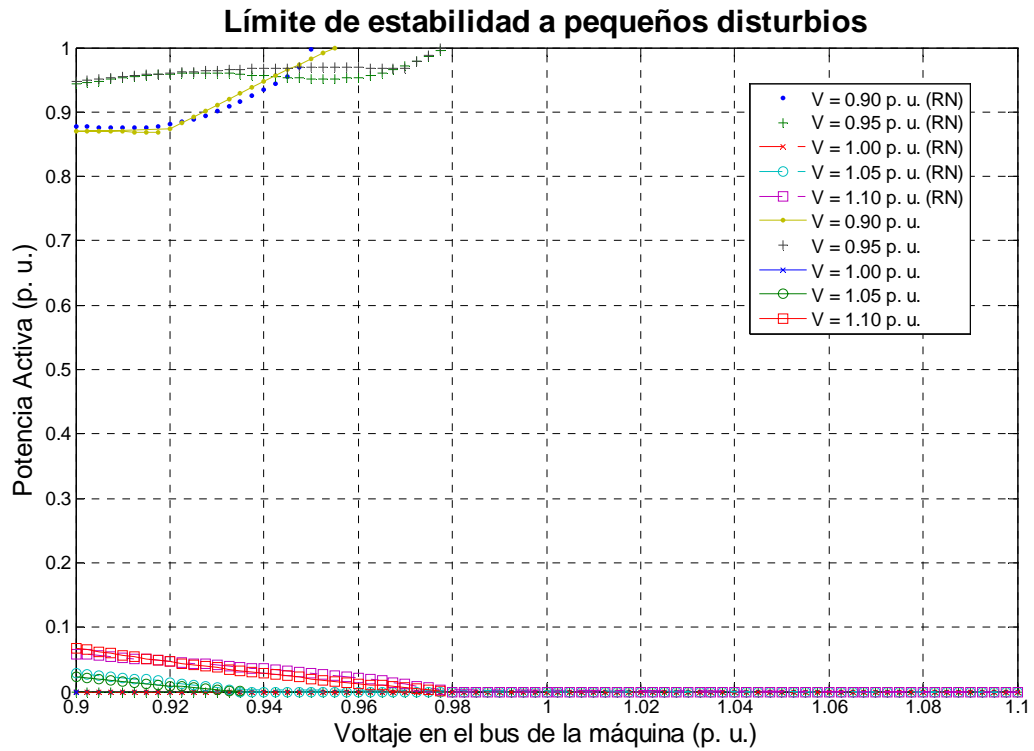


Figura 4. 12a – Límites de estabilidad por análisis de eigenvalores y por RN (LC – 1L)

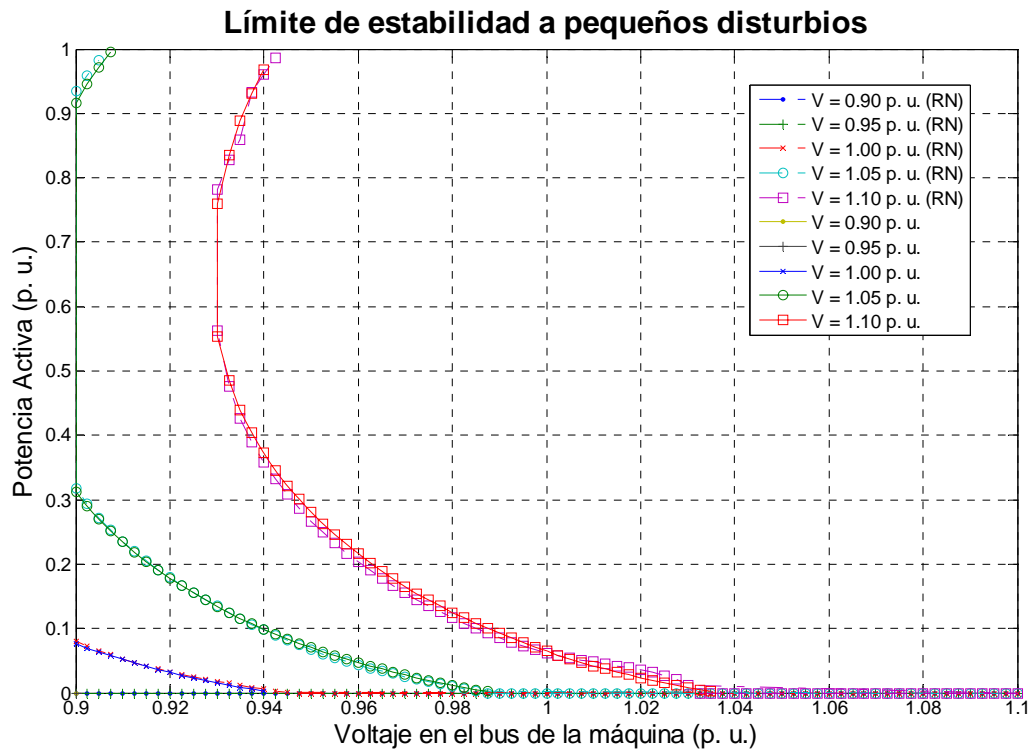


Figura 4. 12b – Límites de estabilidad por análisis de eigenvalores y por RN (LC – 2L)

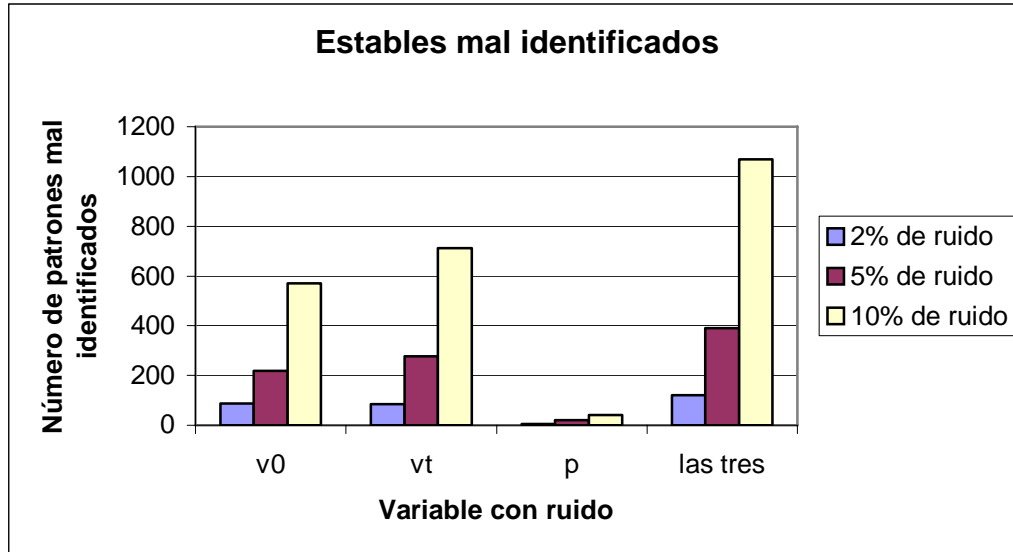


Figura 4. 13a – Porcentaje de patrones estables mal identificados por la RN (LC)

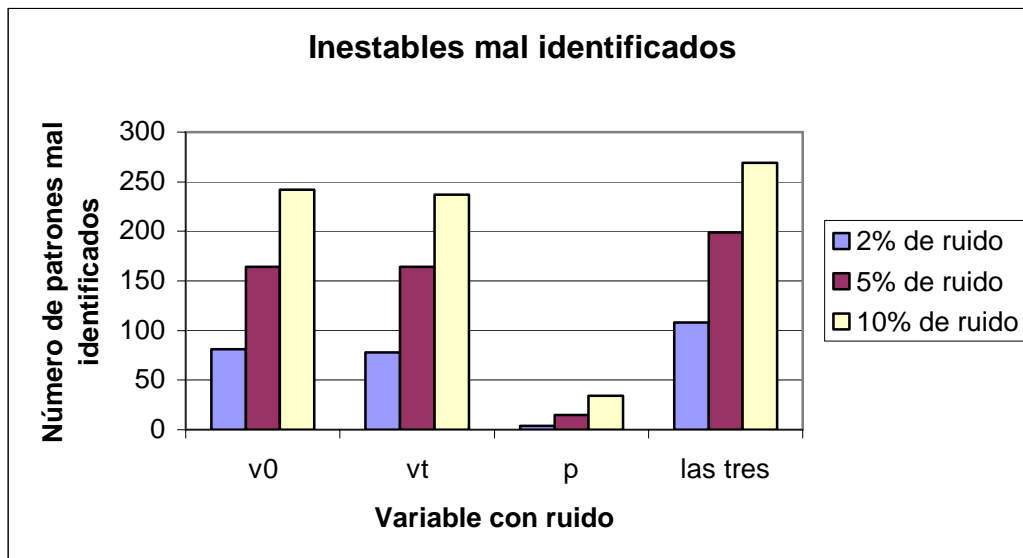


Figura 4. 13b – Porcentaje de patrones inestables mal identificados por la RN (LC)

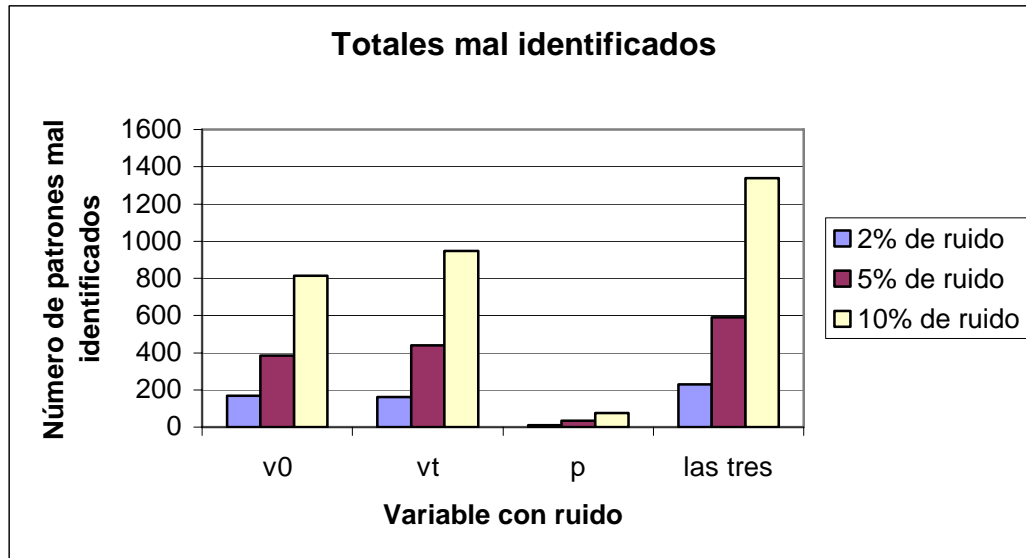


Figura 4. 13c – Porcentaje de patrones estables e inestables mal identificados por la RN (LC)

En las figuras 4. 13a, 4. 13b y 4. 13c se observa claramente que el ruido en la Potencia de la máquina afecta menos que en cualquiera de los dos voltajes, otro punto a resaltar es que son mas los patrones estables que identifica como inestables que los patrones inestables que identifica como estables, se puede decir que hacer una mala identificación de un patrón inestable es mas grave que la de un patrón estable, porque representaría un problema no detectado oportunamente.

Al aumentar el nivel de ruido también aumenta el número de patrones mal identificados, lo cual es razonable, así como también que al contaminar con ruido las tres variables simultáneamente el número de patrones mal identificados también aumente.

Se hace otra prueba comparando la identificación de la estabilidad con los datos de prueba con la RN y mediante el análisis de los eigenvalores para ciertos niveles de ruido en los datos (2.50% para  $|v_0|$  y  $|v_i|$ , y 5.00% para  $P$ ), los resultados de estas pruebas cuantitativas:

Prueba de determinación de la estabilidad por RN:

```
18505 PATRONES PROBADOS
  182 PATRONES ESTABLES IDENTIFICADOS EQUIVOCADAMENTE ( 0.98%)
  131 PATRONES INESTABLES IDENTIFICADOS EQUIVOCADAMENTE ( 0.71%)
  313 PATRONES TOTALES IDENTIFICADOS EQUIVOCADAMENTE ( 1.69%)
```

Prueba de determinación de la estabilidad por análisis de eigenvalores:

```
18505 PATRONES PROBADOS
  165 PATRONES ESTABLES IDENTIFICADOS EQUIVOCADAMENTE ( 0.89%)
  129 PATRONES INESTABLES IDENTIFICADOS EQUIVOCADAMENTE ( 0.70%)
  294 PATRONES TOTALES IDENTIFICADOS EQUIVOCADAMENTE ( 1.59%)
```

Tratando de mostrar gráficamente los resultados anteriores se analizan algunos ejemplos que resuman el comportamiento general de la prueba (figuras 4.

14a a 4. 14d), en ellas los círculos rojos muestran los patrones inestables obtenidos de la simulación y contaminados con ruido, los puntos azules muestran los patrones estables obtenidos de la simulación y contaminados con ruido, los asteriscos rojos son los patrones inestables que la RN identifica como estables, y los asteriscos azules son los patrones estables que la RN identifica como inestables, en la misma figura se muestran los límites de la estabilidad obtenidos mediante en análisis de eigenvalores y con la RN.

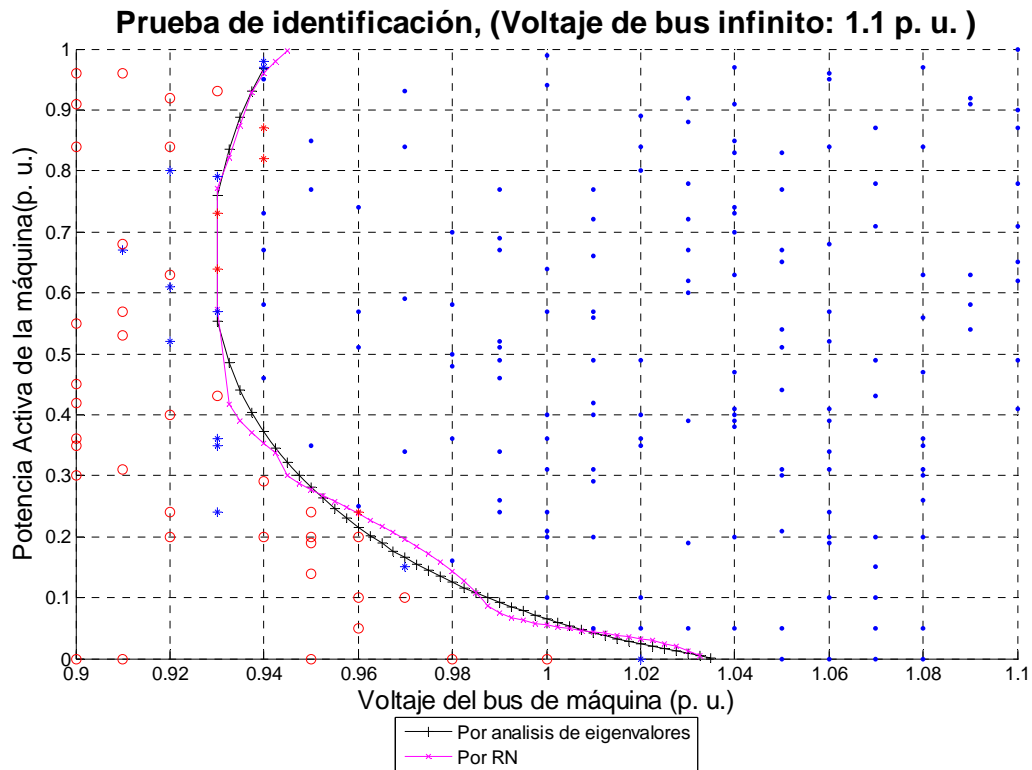


Figura 4. 14a – Prueba 1 de identificación y límites de estabilidad por RN y por análisis de eigenvalores (LC – 2L)



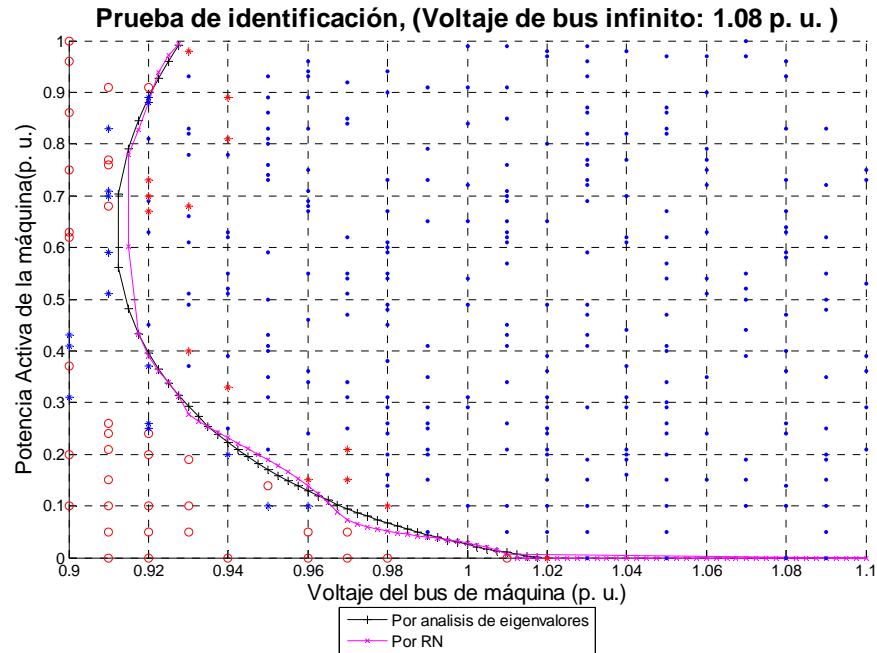


Figura 4. 14b – Prueba 2 de identificación y límites de estabilidad por RN y por análisis de eigenvalores (LC – 2L)

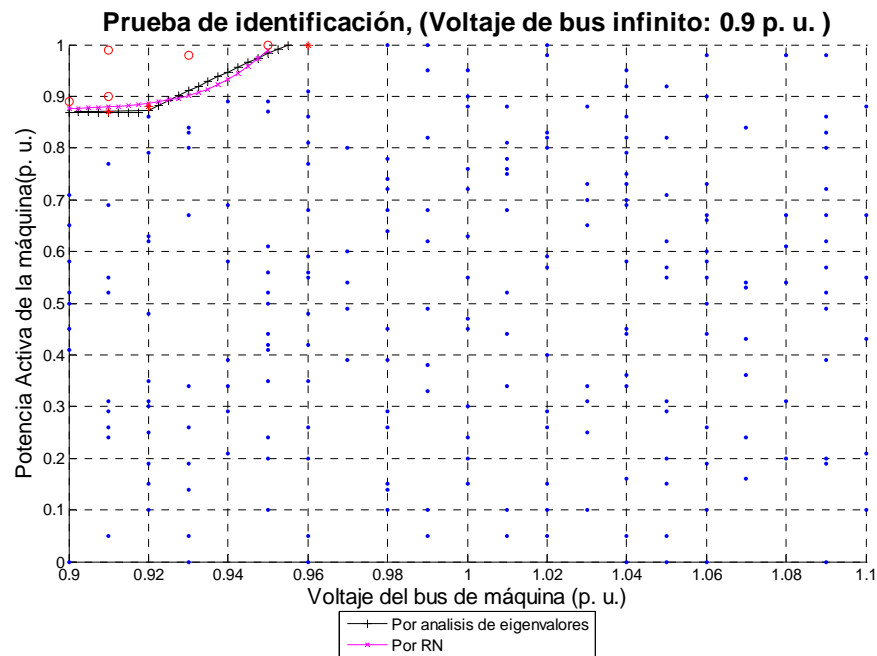


Figura 4. 14c – Prueba 3 de identificación y límites de estabilidad por RN y por análisis de eigenvalores (LC – 1L)

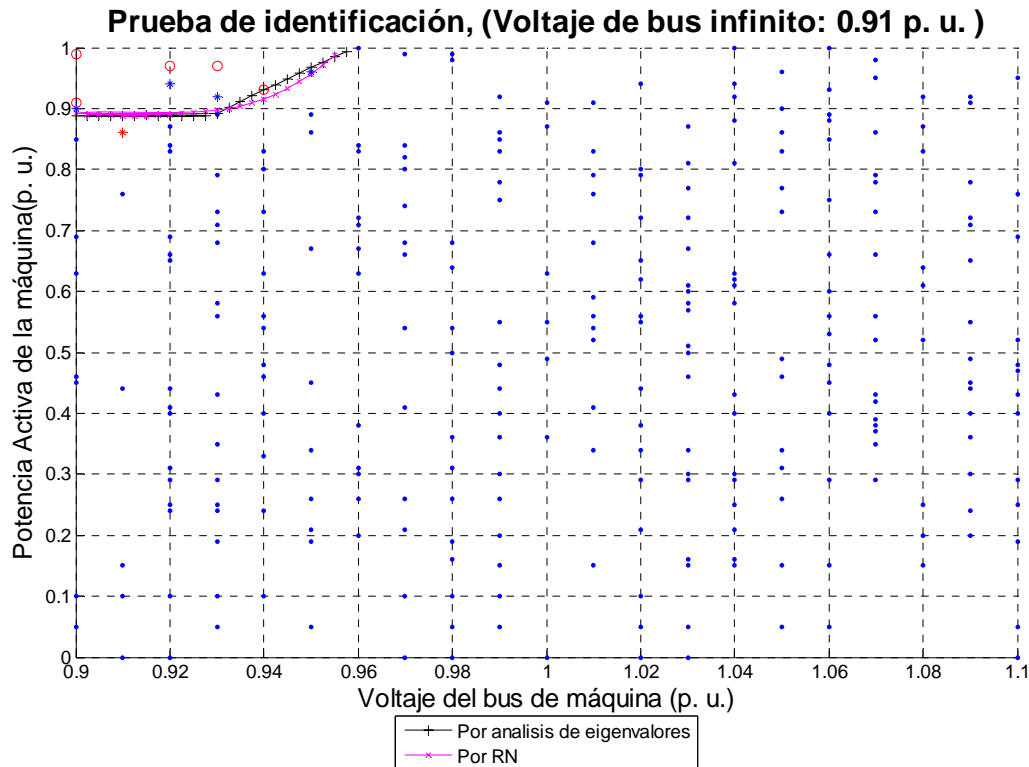


Figura 4. 14d – Prueba 4 de identificación y límites de estabilidad por RN y por análisis de eigenvalores (LC – 1L)

Los resultados de la mala identificación de algunos patrones no depende de las características propias de la RN sino del sistema en sí, porque los mismos patrones son mal identificados tanto por la RN como haciendo el análisis de eigenvalores, o bien los que no son igualmente identificados son los que están muy cerca de los límites.

El modelo linealizado se utiliza para validar los resultados del entrenamiento de la RN, y en los resultados anteriores se ha mostrado que las pruebas que validan los resultados de mejor manera son las pruebas de identificación de límites de estabilidad ya que la mala identificación debida al ruido en los datos de prueba no depende del entrenamiento, además probar un juego de datos puede no englobar todos los resultados como lo haría en cierta manera la identificación de los límites de estabilidad.

#### 4.4.2.2. Línea larga

En las pruebas de identificación anteriores se concluyó que las pruebas mas demostrativas son las de identificación de los limites de estabilidad más que las de cuantificación de patrones mal identificados de un conjunto de datos de prueba, por lo que en las pruebas posteriores los resultados se validarán mediante la

comparación de los límites identificados por la RN y los obtenidos por el análisis de los eigenvalores (figuras 4.15a y 4.15b).

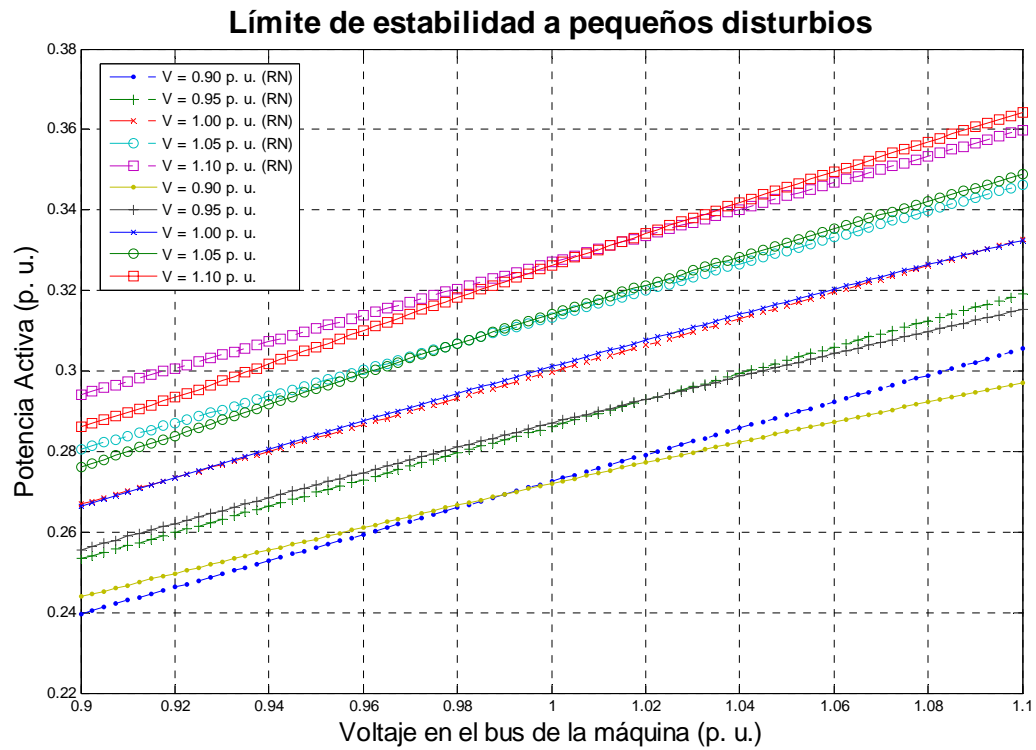


Figura 4.15a – Límites de estabilidad por análisis de eigenvalores y por RN (LL – 1L)

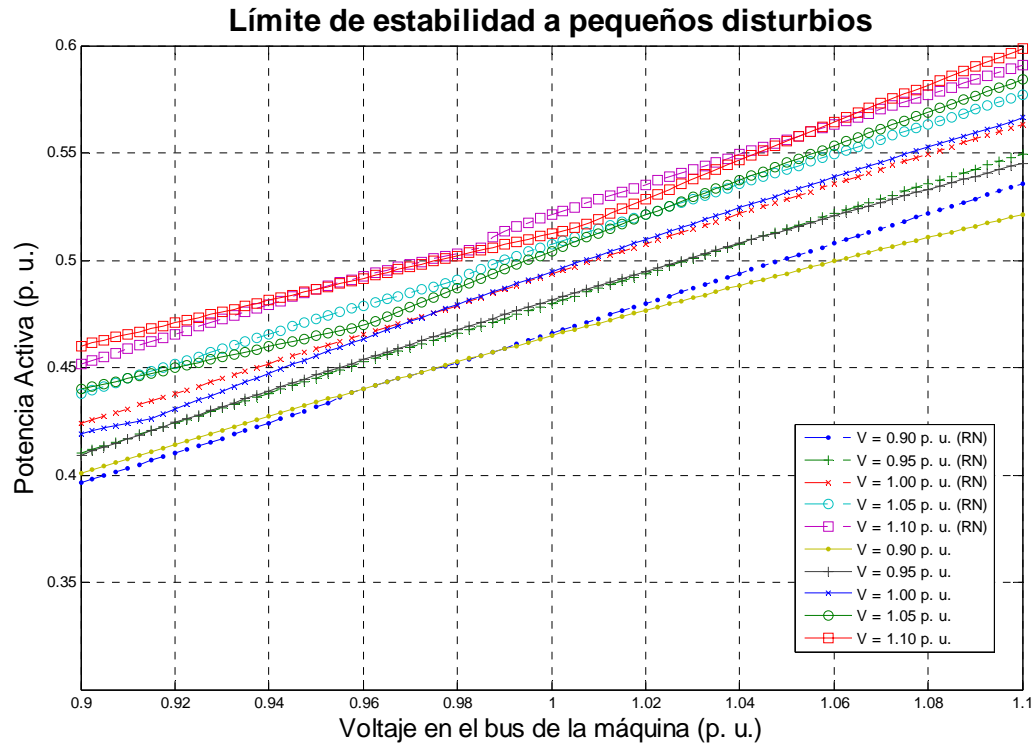


Figura 4. 15b – Límites de estabilidad por análisis de eigenvalores y por RN (LL – 2L)

En las figuras 4. 15a y 4. 15b se puede observar que la RN ha aprendido aceptablemente los límites, con una variación pequeña, pero no significativa; la dificultad en este caso es que es un sistema más restrictivo. La siguiente prueba consiste en contaminar con ruido los datos de prueba y ver que tantos patrones son mal identificados, el ruido se agrega a una variable cada vez para determinar cual es la variable mas sensible al ruido en cuanto a estabilidad, también se prueba con ruido en las tres variables al mismo tiempo y con ruido a distintos niveles en cada prueba.

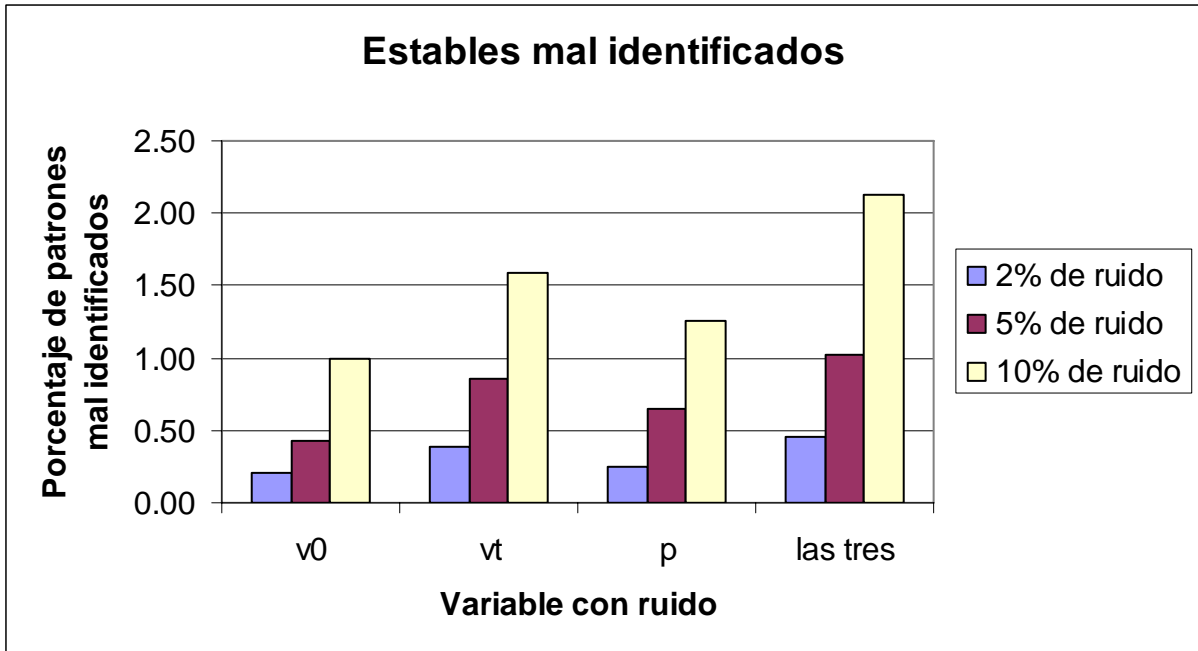


Figura 4. 16a – Porcentaje de patrones estables mal identificados por la RN (LL)

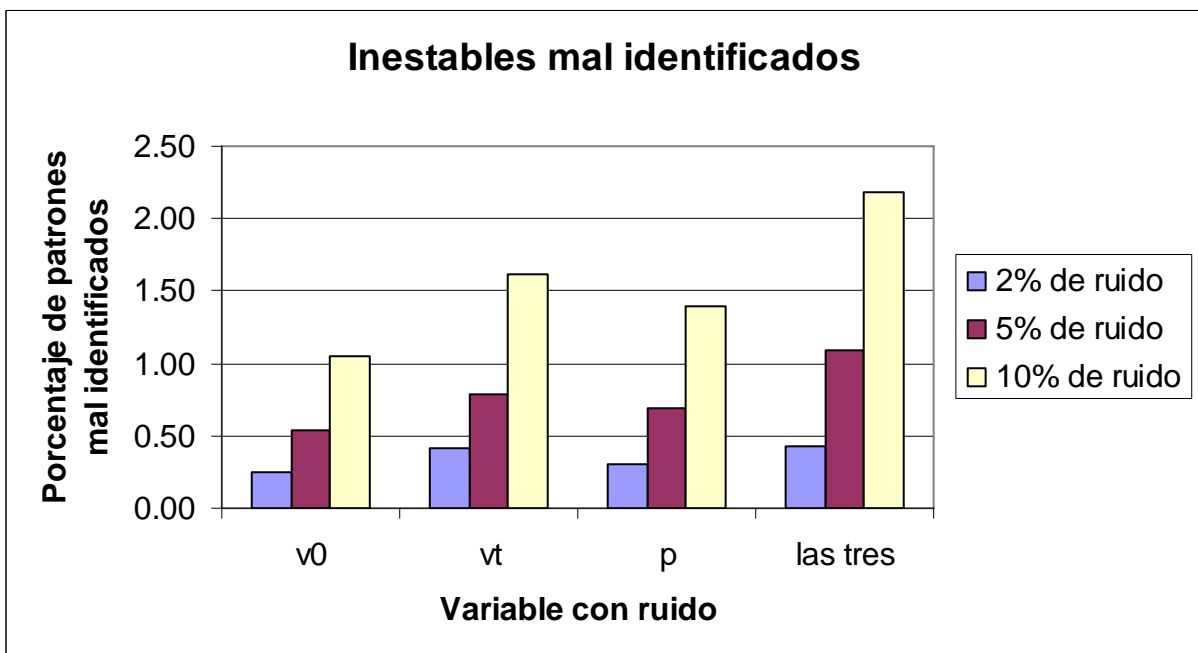


Figura 4. 16b – Porcentaje de patrones inestables mal identificados por la RN (LL)

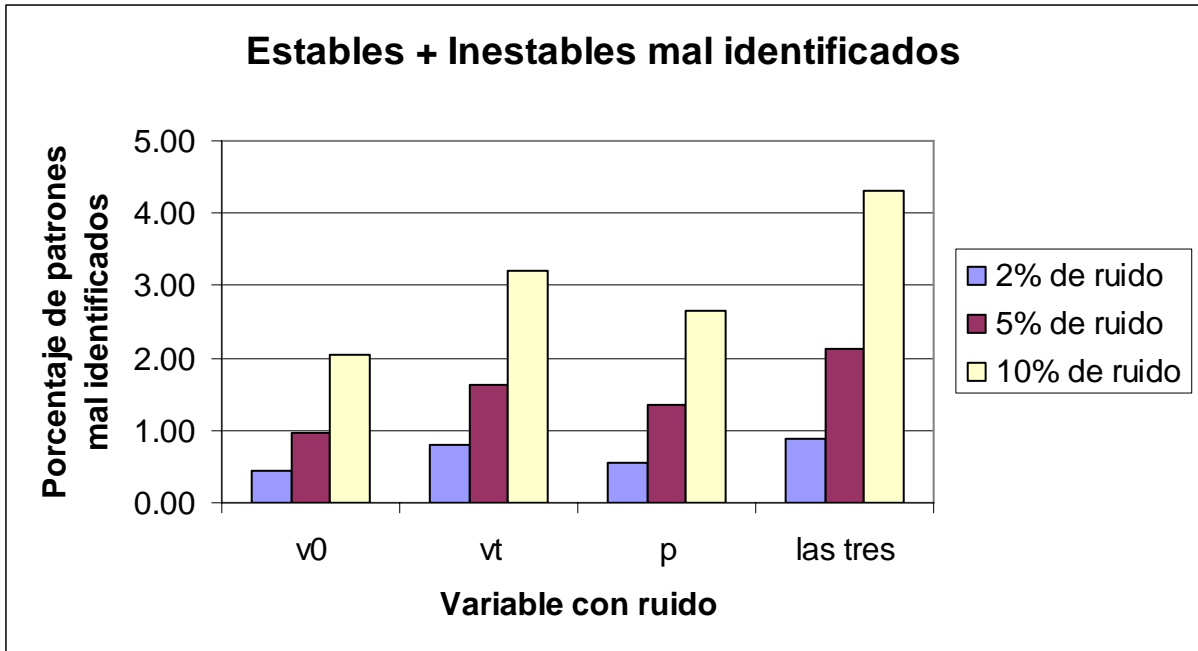


Figura 4. 16c – Porcentaje de patrones estables e inestables mal identificados por la RN (LL)

En las figuras 4. 16a, 4. 16b y 4. 16c se observa que el comportamiento de la mala identificación de patrones estables es muy similar al de la mala identificación de los patrones inestables; también se puede observar que el ruido en el voltaje del bus de la máquina afecta más que el ruido en las otras variables, aunque no muy significativamente, por supuesto a mayores niveles de ruido se incrementa también el porcentaje de patrones mal identificados.

#### 4.4.2.3. Análisis de resultados

Las pruebas de identificación de los límites de estabilidad son los más representativos; pero el hecho de que dichos límites sean identificados adecuadamente, no garantiza que al presentarse un nuevo objeto, éste no sea mal clasificado, ya que si contiene ruido, entonces puede ubicarse mal, pero esa es una característica del sistema analizado y no de la RN.

Con las pruebas de sensibilidad al ruido se puede ver que al aumentar la contaminación, el porcentaje de condiciones de operación mal identificadas también aumenta, como es de esperarse, lo interesante es ver que para el caso LC la potencia parece no contribuir significativamente a la mala identificación, mientras que en el caso LL las tres variables contribuyen aproximadamente en la misma proporción, es decir, cada caso tiene características propias de comportamiento.

En general entonces, el desempeño final de la RN depende de la calidad de las mediciones o estimaciones de las variables del sistema, y no de su topología o del caso en particular.

## **4.5. Entrenamiento de la RN con datos contaminados con ruido**

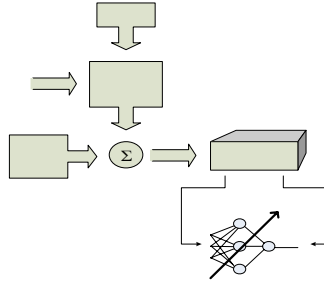
Hasta este momento los resultados han sido los esperados, pero no es posible hacer una evaluación del desempeño real del sistema de identificación ya que en un problema real los datos con los cuales se entrena la RN difícilmente provienen de simulaciones de modelos precisos, en cambio los datos se toman de mediciones o estimaciones; las cuales pueden no reflejar la realidad debido a distintas situaciones como malos métodos de medición o estimación, errores debidos a la instrumentación, etc.

Una ventaja muy atractiva de usar una RN para identificar la estabilidad es que así no es necesario contar con el modelo del sistema. Para tratar de recrear una situación real, en la que existen errores de medición, de procedimiento, etc., los datos obtenidos de la simulación se contaminan con ruido, y la RN se entrena con estos datos contaminados.

### **4.5.1. Contaminación de ruido a la base de datos**

El ruido se genera con números aleatorios de distribución normal en un rango controlado, se prueba con ruido a distintos niveles para experimentar con distintas condiciones, la identificación de los distintos entrenamientos se hará refiriéndose a ellos como RN1, RN2 y RN3 según lo siguiente:

- RN1. Nivel bajo de ruido (1.0% de ruido para voltajes y 2.0% para potencia)
- RN2. Nivel moderado de ruido (2.5% de ruido para voltajes y 5.0% para potencia)
- RN3. Nivel alto de ruido (5.0% de ruido para voltajes y 10.0% para potencia)



#### 4. 17 – Contaminación con ruido a los datos de la simulación

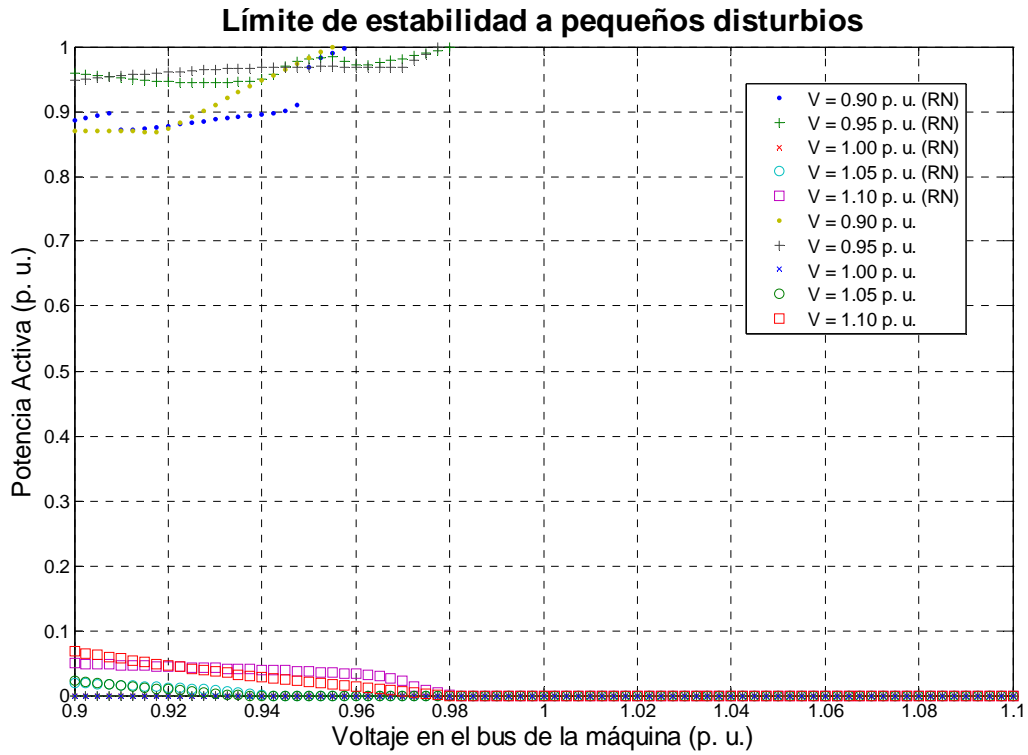
### 4.5.2. Entrenamiento con topología determinada con datos de la simulación

Como se ha mencionado, no existen métodos generales para la determinación de la topología de una RN; en el entrenamiento de la RN con datos provenientes de la simulación (sin contaminación de ruido) se hizo la búsqueda con dos métodos (búsqueda exhaustiva y poda), de ahí se encontró una topología para la RN con la que se obtuvieron buenos resultados, los cuales sirven como base para el entrenamiento con datos contaminados con ruido, aunque los resultados no deben ser necesariamente los mismos.

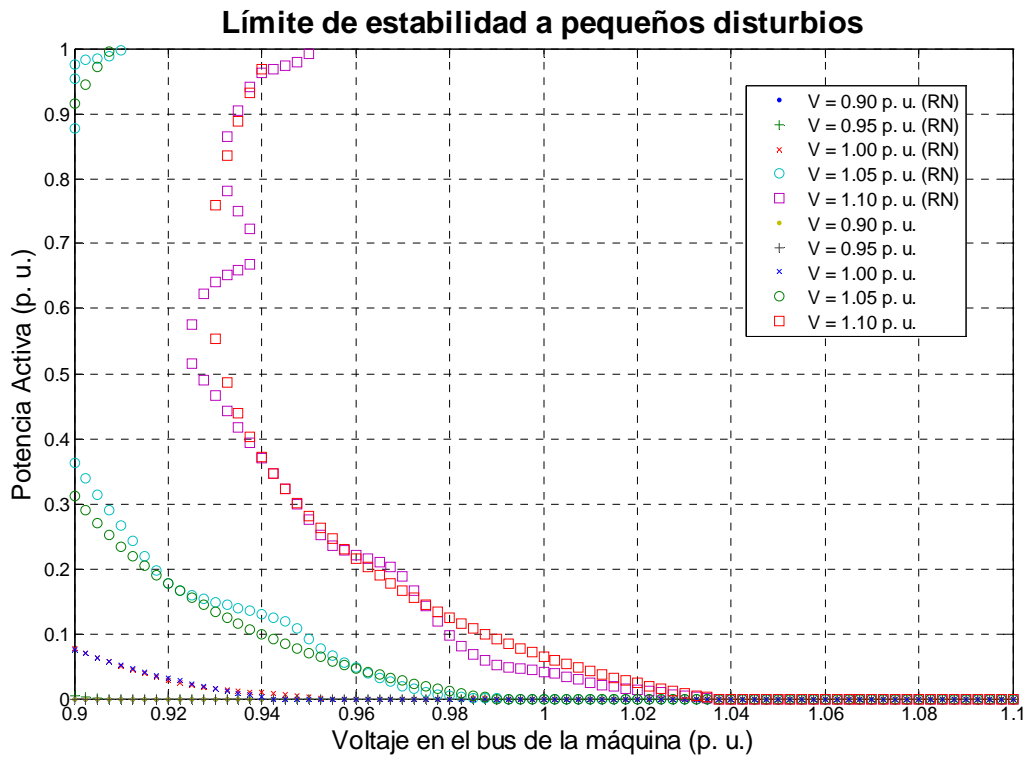
#### 4.5.2.1. Línea corta

Como los datos de entrenamiento son contaminados con ruido es necesario realizar un análisis previo a entrenar la RN, el análisis consiste en eliminar todos aquellos patrones que se repiten y en caso de haber patrones diferentes con características idénticas (salida diferente para un mismo juego de entradas), se eliminan los patrones estables y se conservan los inestables. Los resultados de la identificación de los límites se muestran en las figuras 4. 18a a 4. 20b.

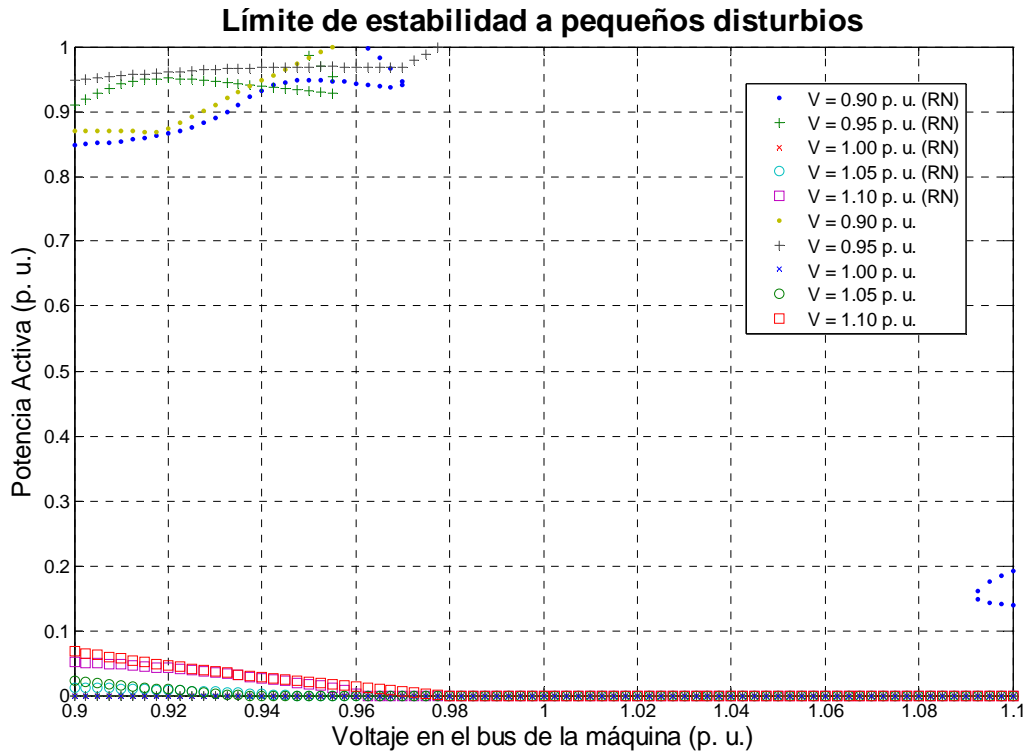




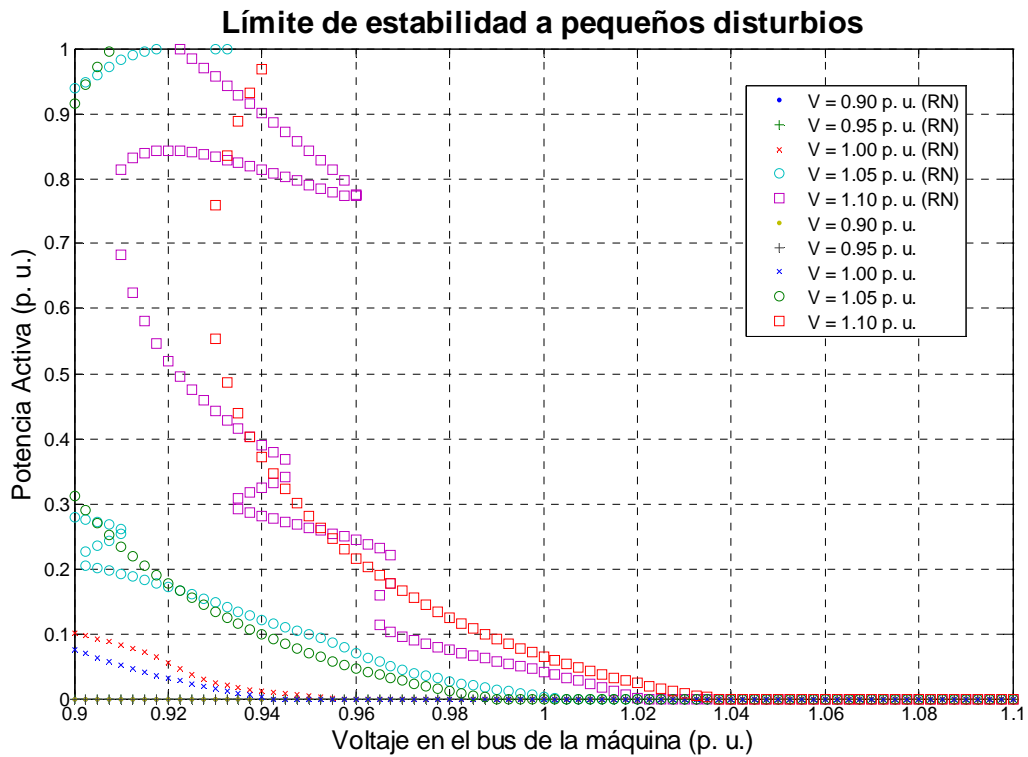
4. 18a – Límites de estabilidad por análisis de eigenvalores y por RN1 (LC – 1L)



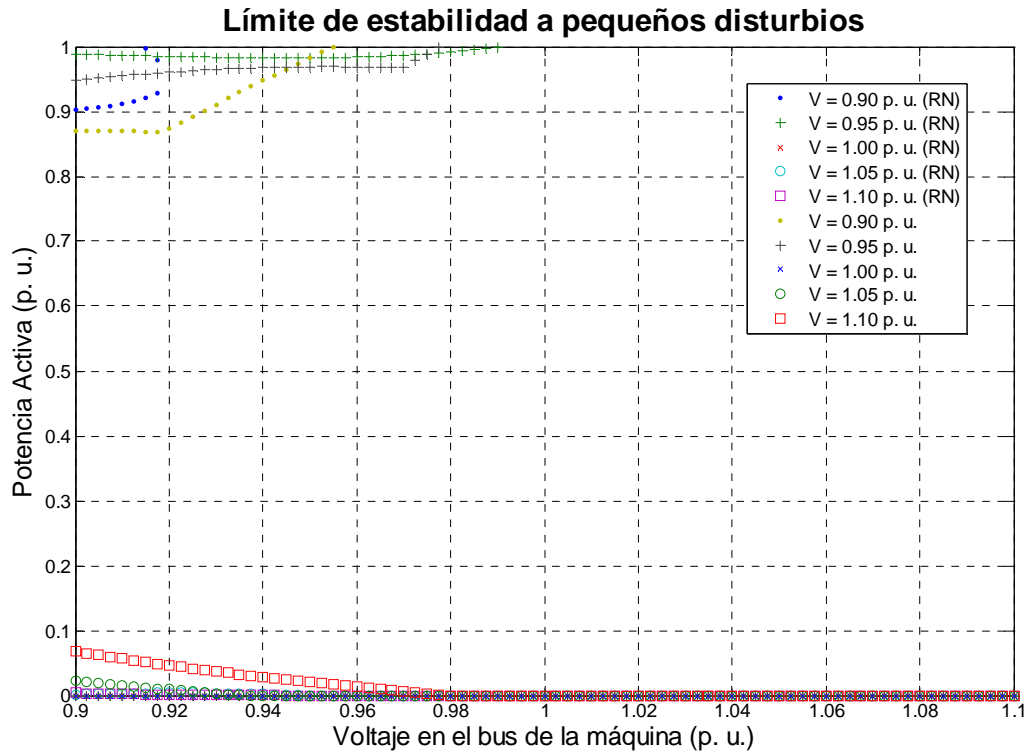
4. 18b – Límites de estabilidad por análisis de eigenvalores y por RN1 (LC – 2L)



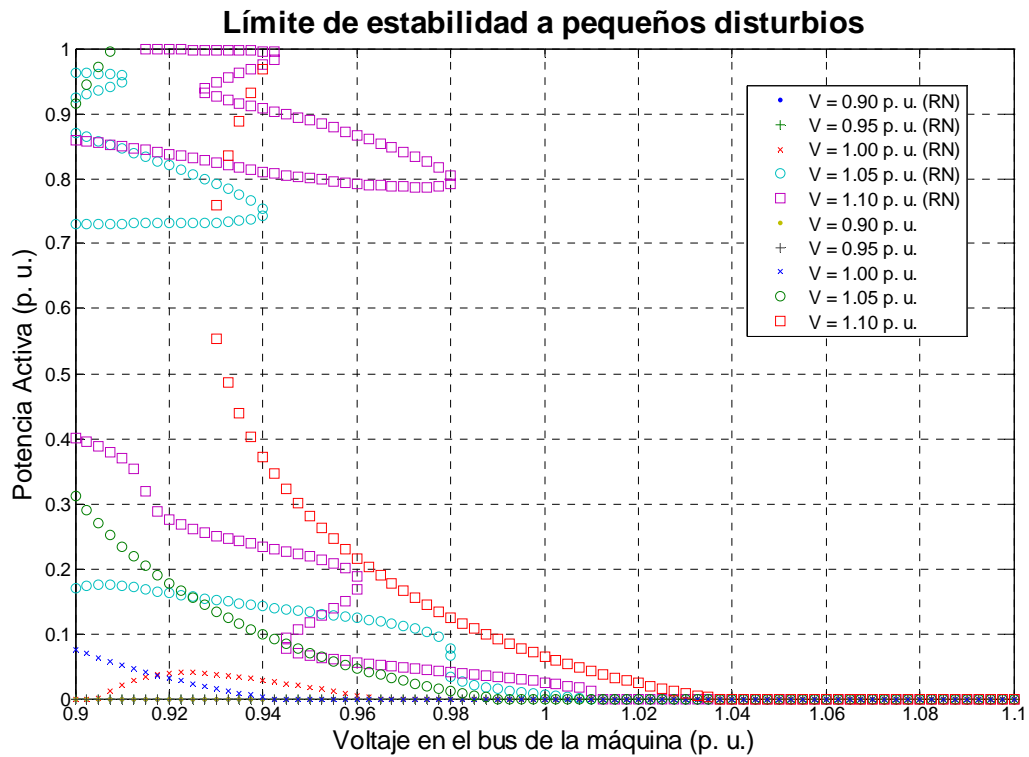
4. 19a – Límites de estabilidad por análisis de eigenvalores y por RN2 (LC – 1L)



4. 19b – Límites de estabilidad por análisis de eigenvalores y por RN2 (LC – 2L)



4. 20a – Límites de estabilidad por análisis de eigenvalores y por RN3 (LC – 1L)

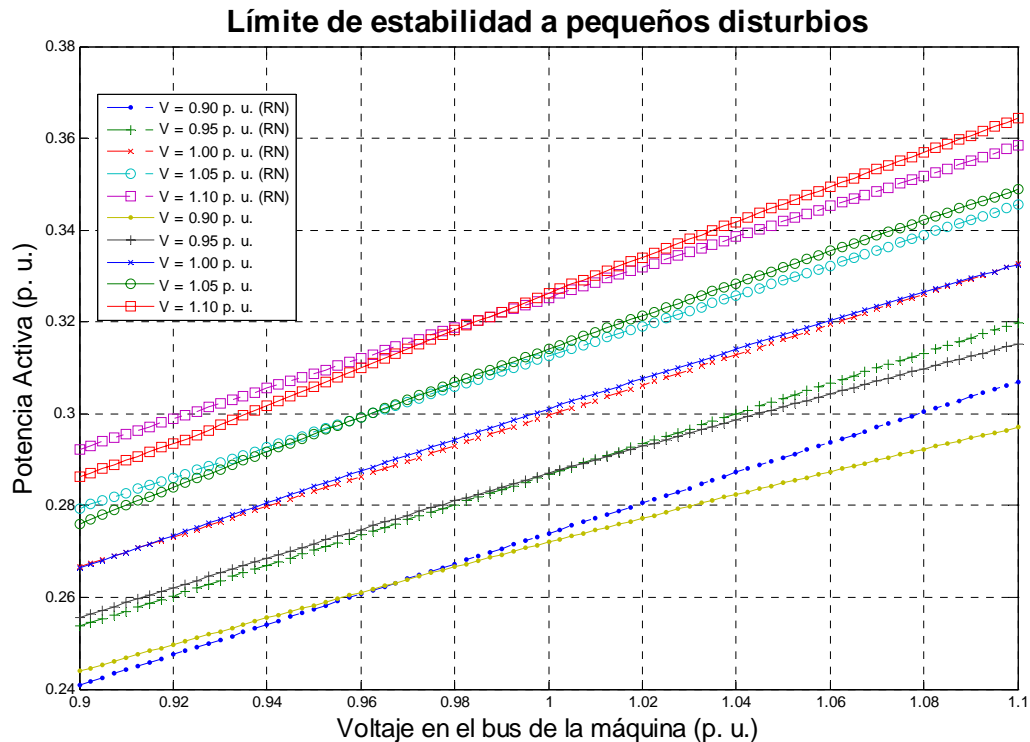


4. 20b – Límites de estabilidad por análisis de eigenvalores y por RN3 (LC – 2L)

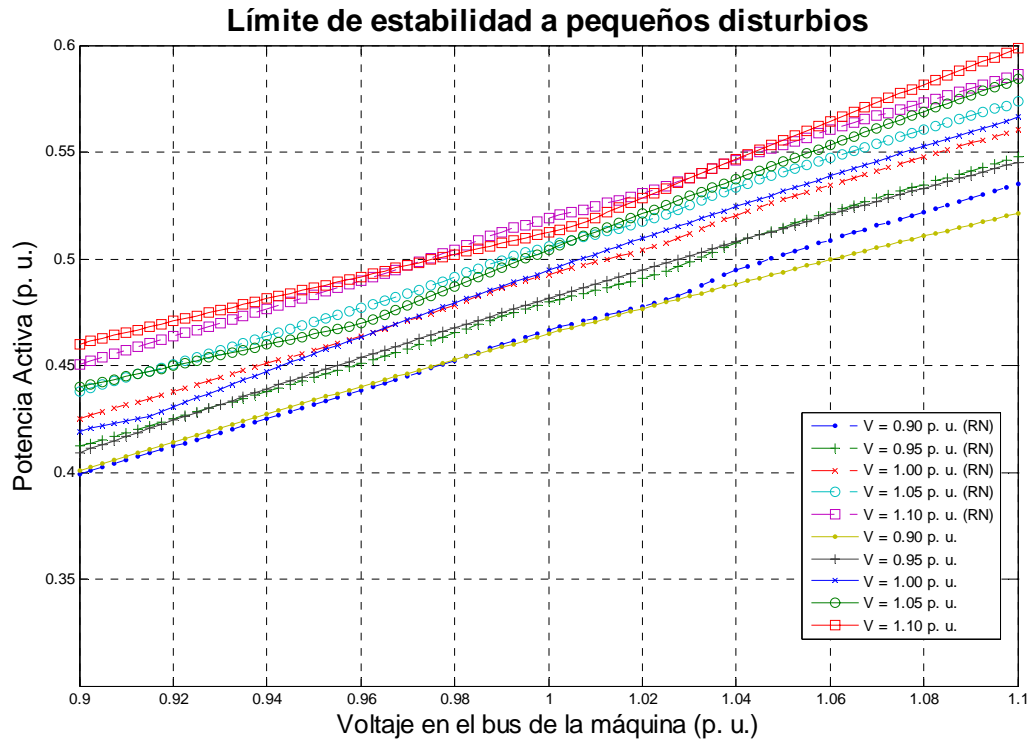
Los resultados son malos en general, aunque para bajos niveles de ruido se puede decir que son aceptables, pero no así para niveles de ruido moderado y alto.

#### 4.5.2.2. Línea larga

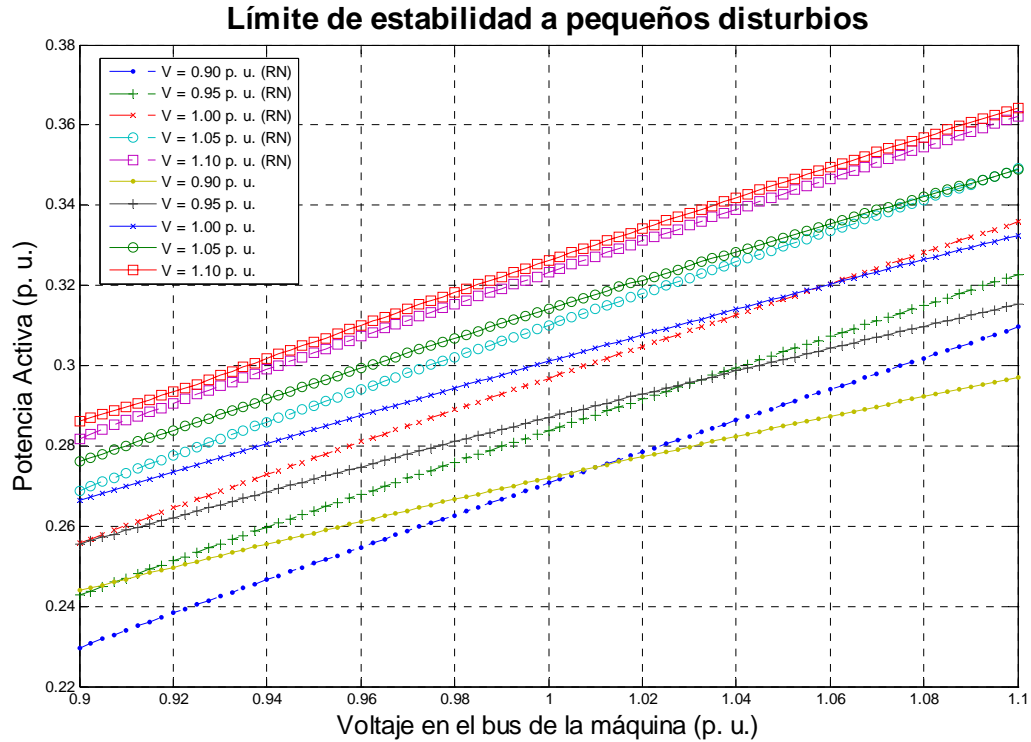
Al igual que para el caso LC, los datos de entrenamiento se contaminan con ruido, por lo que es necesario hacer un análisis previo de los datos; la intención es que la RN aprenda solo las características más importantes de los patrones, en ese sentido, el número de parámetros de la RN debe ser suficiente para aprender dichas características, pero no debe ser excesivo por que entonces aprende un caso particular. Al contaminar con ruido los datos de entrenamiento, se pueden llegar a tener varios casos particulares, dependiendo de la distribución del ruido, pero como es aleatorio, ninguno de estos casos refleja necesariamente el comportamiento general del problema. Los resultados de la identificación de los límites de estabilidad para este caso se muestran en las figuras 4. 21a a 4. 23b.



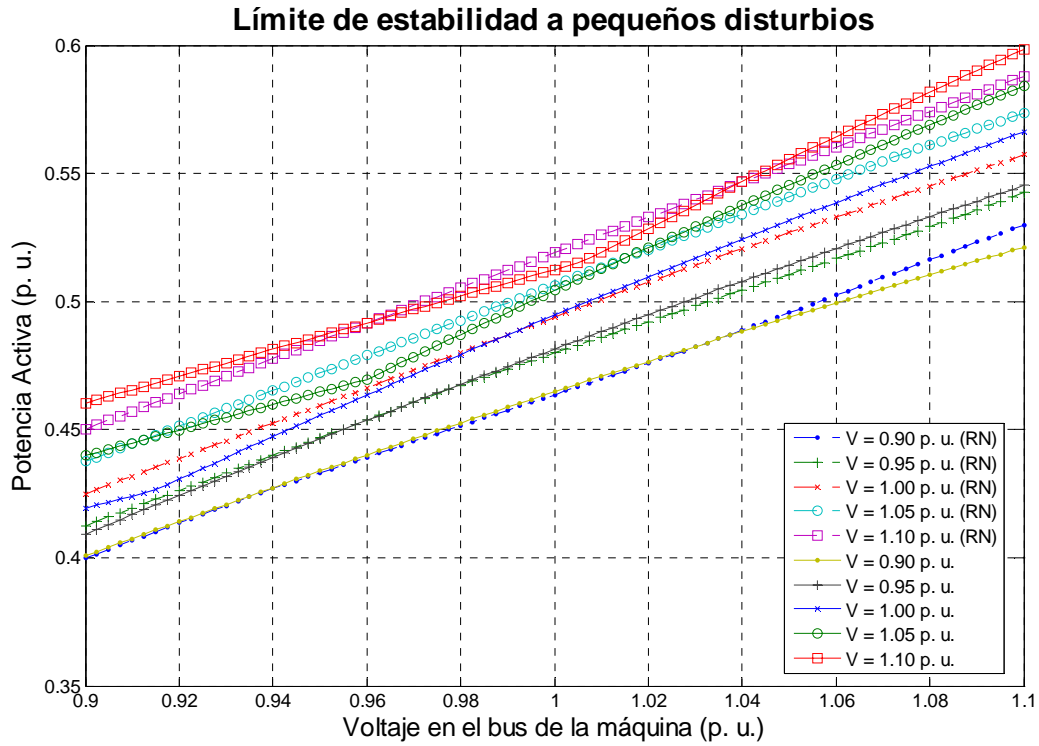
4.21a – Límites de estabilidad por análisis de eigenvalores y por RN1 (LL – 1L)



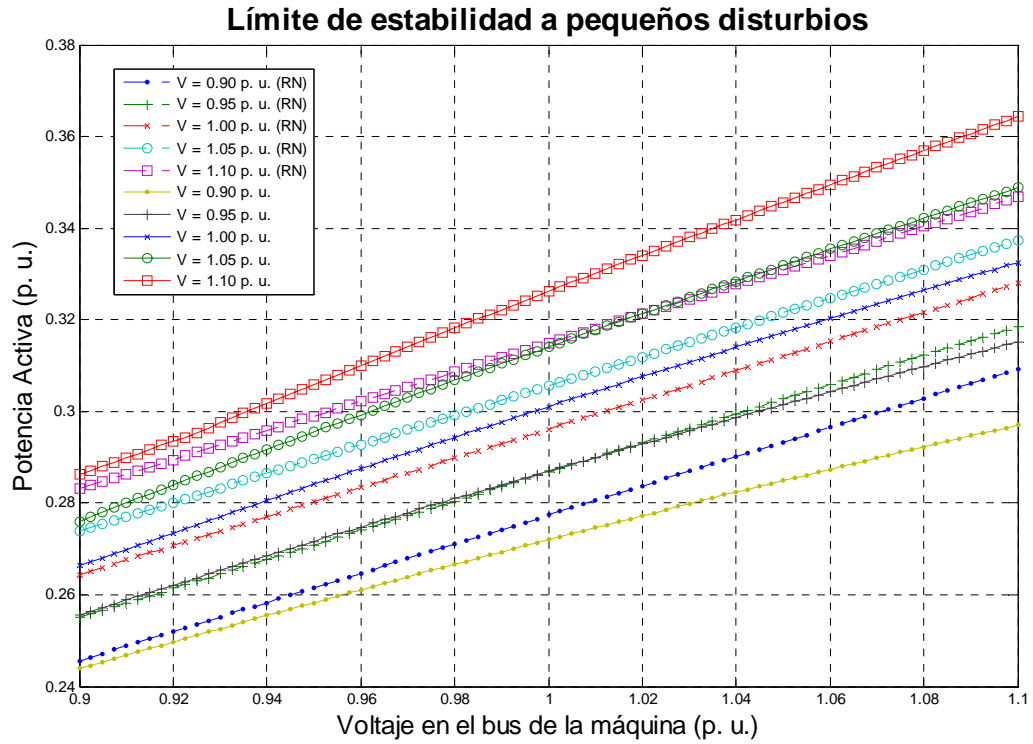
4.21b – Límites de estabilidad por análisis de eigenvalores y por RN1 (LL – 2L)



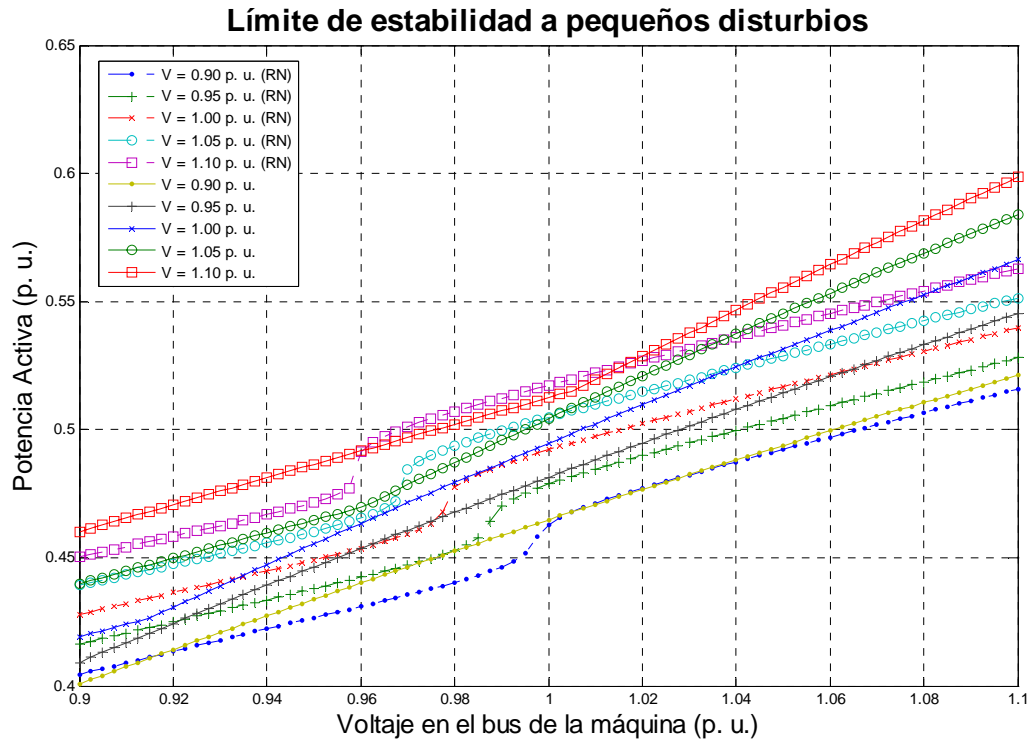
4.22a – Límites de estabilidad por análisis de eigenvalores y por RN2 (LL – 1L)



4. 22b – Límites de estabilidad por análisis de eigenvalores y por RN2 (LL – 2L)



4.23a – Límites de estabilidad por análisis de eigenvalores y por RN3 (LL – 1L)



Los resultados son en general buenos, aunque a medida que el nivel de ruido aumenta, los límites identificados por la RN se desvían un poco lo cual es natural; luego de varios entrenamientos, se observa que los parámetros de la capa oculta de la RN2 están muy dispersos, lo que afecta a su desempeño.

#### 4.5.2.3. Análisis de resultados

En el caso LL se tienen menos parámetros de la RN que para el caso LC, es posible que el mal desempeño de la RN para al caso LC se deba a un excesivo número de parámetros en la RN. Como el resultado final para ese caso (LC) es malo, se deben probar alternativas en la topología de la RN, y de persistir el mal desempeño, se debe buscar otro tipo de RN que de mejores resultados; como alternativas de topología de la RN se tienen el cambio de unidades en las capas ocultas, el cambio de funciones de activación, y el aumento de capas ocultas, lo mas lógico sería buscar una alternativa mas sencilla, es decir, reducir el número de neuronas para reducir el número de parámetros.

#### 4.5.3. Entrenamiento con topologías alternativas

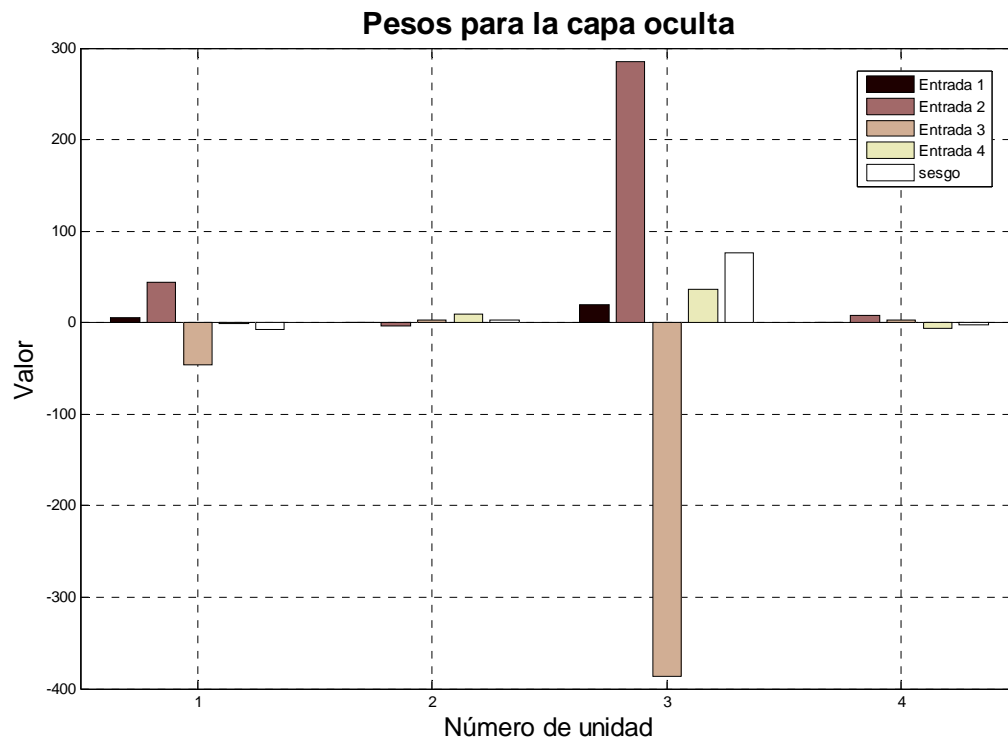
En está sección se buscaran topologías alternativas con el fin de ratificar o reconsiderar los resultados de la sección anterior; en el caso LC, los resultados no

son buenos, por lo que se buscará una mejor solución, y para el caso LL se demostrará que los resultados encontrados son confiables, o en su caso, se buscará también una mejor solución. Las topologías alternativas que se presentan aquí son en función del número de unidades en la capa oculta, otra opción es cambiar el número de capas ocultas y/o el tipo de función de activación, lo cual se intentó ya para el caso línea corta en la sección 4.4.1.1.

#### 4.5.3.1. Línea corta – Reduciendo unidades en la capa oculta

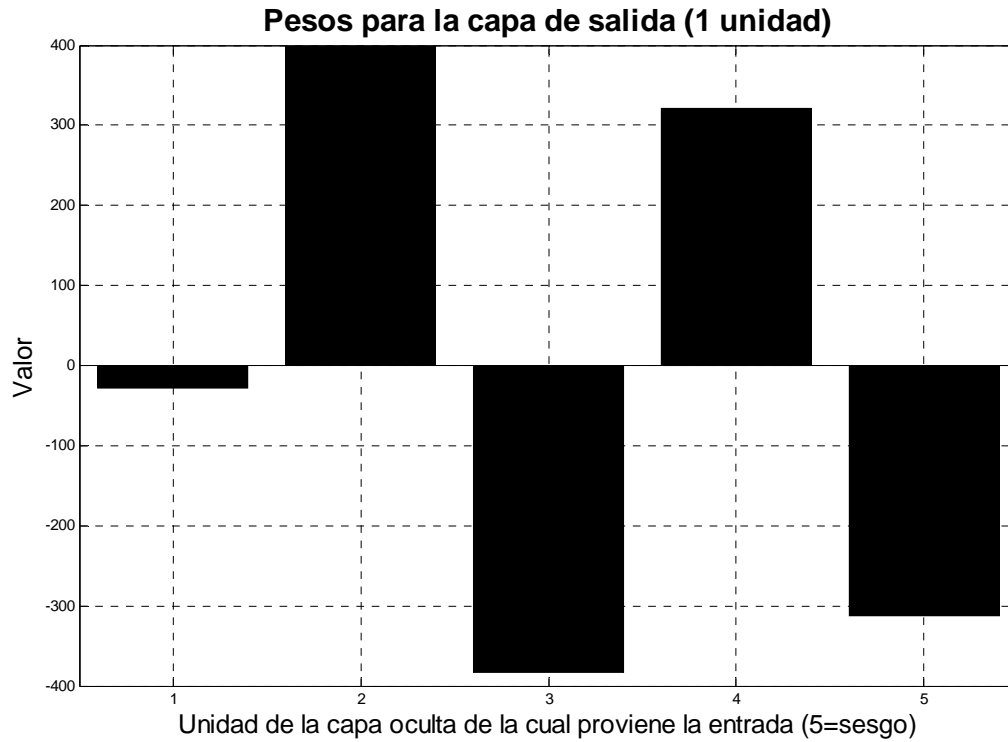
En los resultados anteriores, algunas unidades de la capa oculta no son trascendentales pues sus pesos, a través de los cuales están conectadas a la capa de salida, son pequeños en comparación a los de las demás; se realiza una *poda*, es decir, se eliminan aquellas unidades poco trascendentales y se re-entrena la RN, con esto además, se reduce su capacidad de identificación, es decir, la RN tiende a no identificar todos los patrones, sobre todo aquellos donde hay una dispersión considerable y no se tienen bien definidos los límites.

Al final del proceso se encuentra que la RN debe tener 4 unidades en la capa oculta para identificar aceptablemente los límites de estabilidad. Los parámetros de las RNs se muestran en las figuras 4. 24a a 4. 26b.

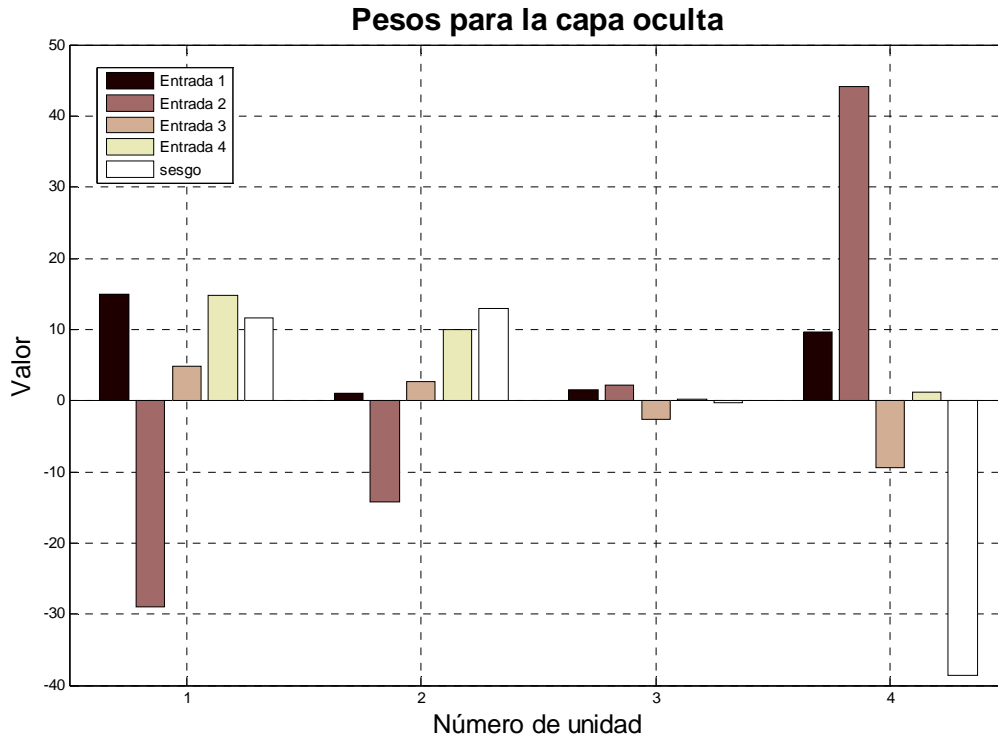


4. 24a – Pesos finales de la capa oculta de la RN1 con 4 neuronas en la capa oculta (LC)

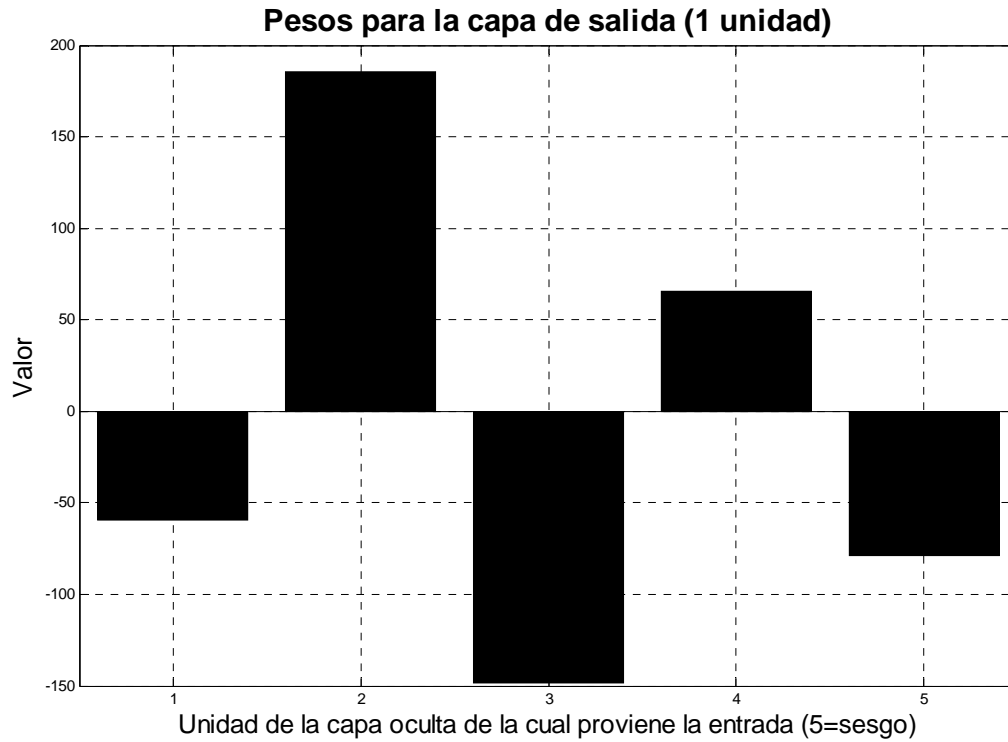




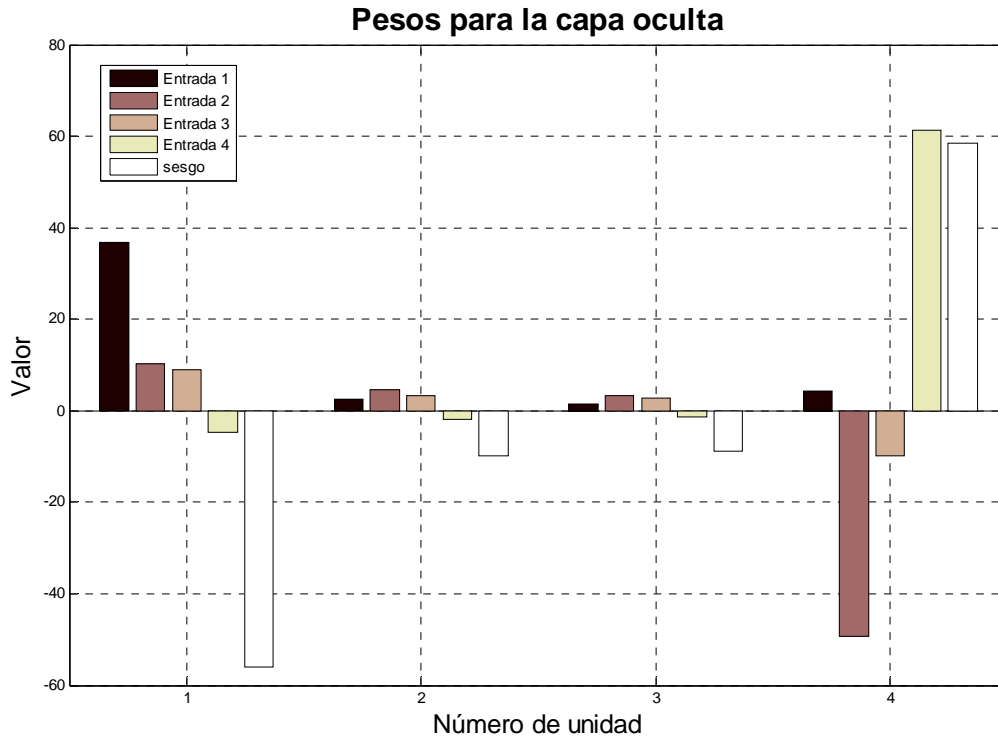
4. 24b – Pesos finales de la capa de salida de la RN1 con 4 neuronas en la capa oculta (LC)



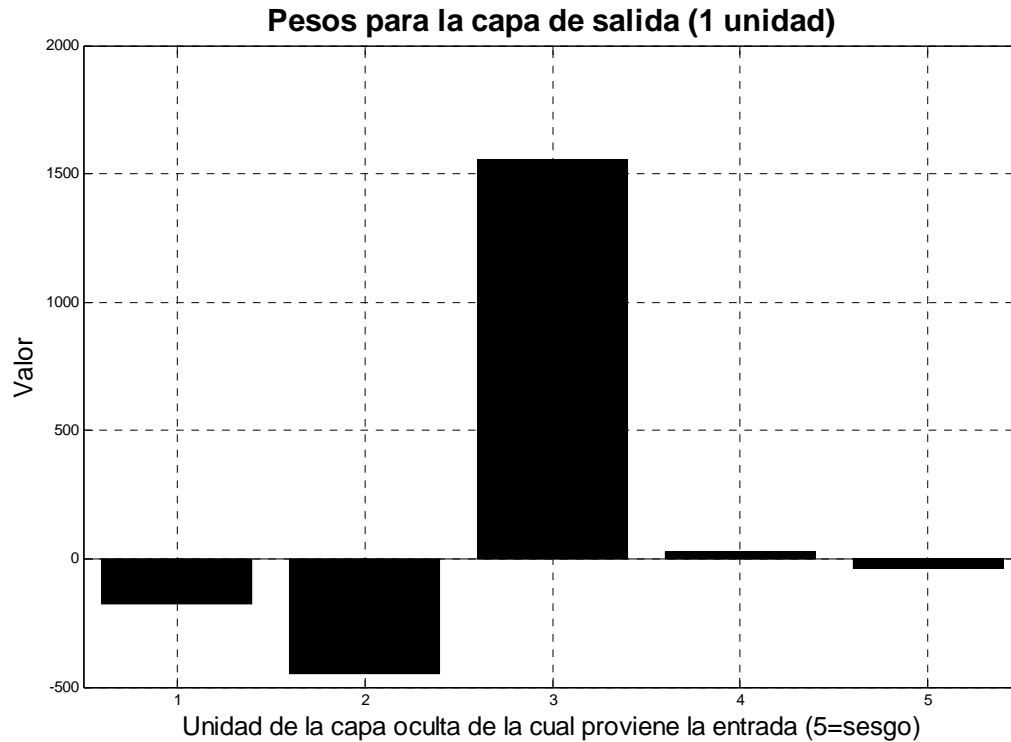
4. 25a – Pesos finales de la capa oculta de la RN2 con 4 neuronas en la capa oculta (LC)



4. 25b – Pesos finales de la capa de salida de la RN2 con 4 neuronas en la capa oculta (LC)

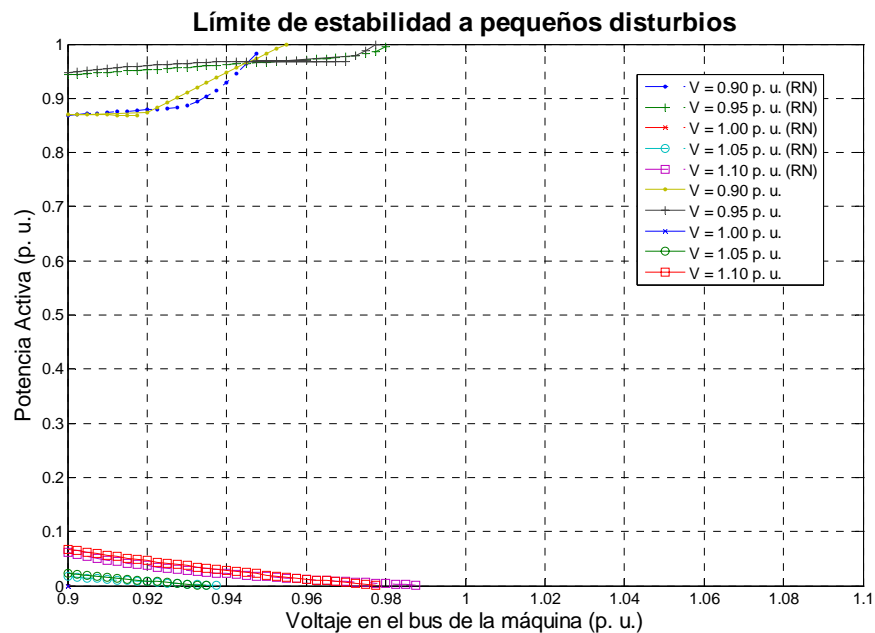


4. 26a – Pesos finales de la capa oculta de la RN3 con 4 neuronas en la capa oculta (LC)

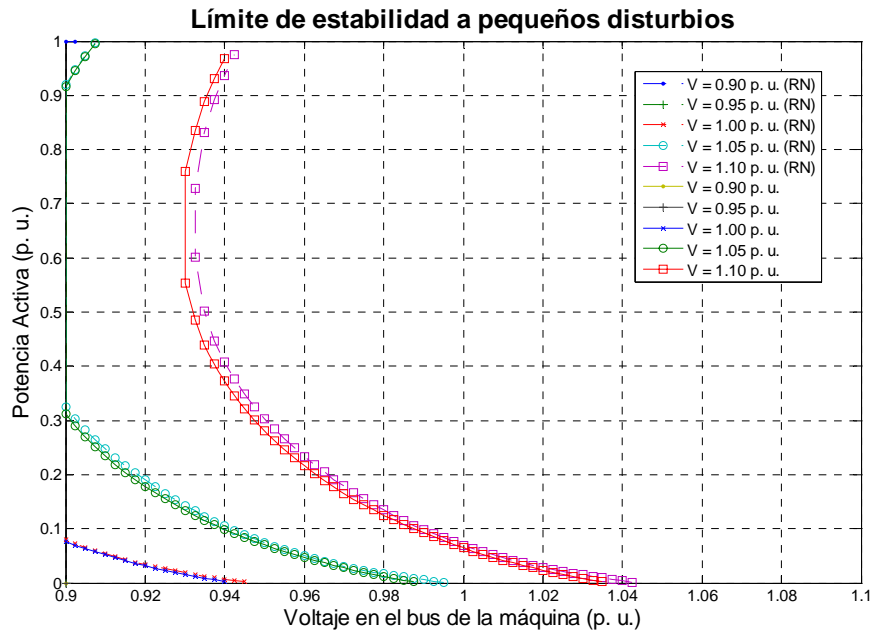


4. 26b – Pesos finales de la capa de salida de la RN3 con 4 neuronas en la capa oculta (LC)

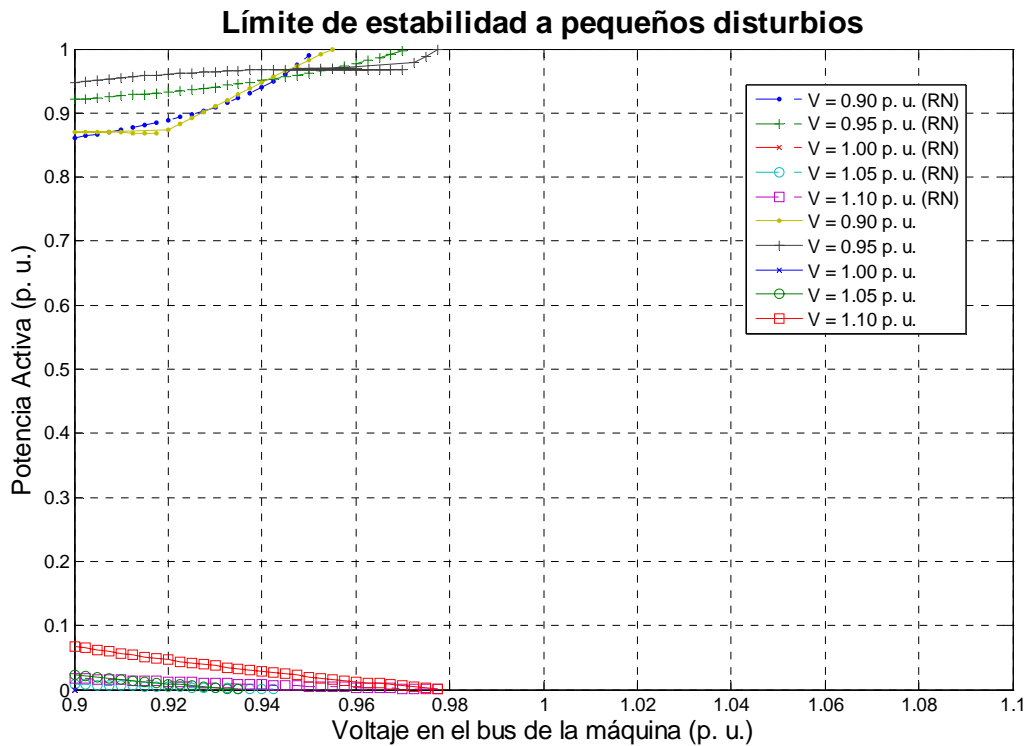
Los parámetros de las RNs están menos dispersos que en las pruebas anteriores. Los resultados de la identificación de los límites con dichos parámetros se muestran en las figuras 4. 27a a 4. 29b:



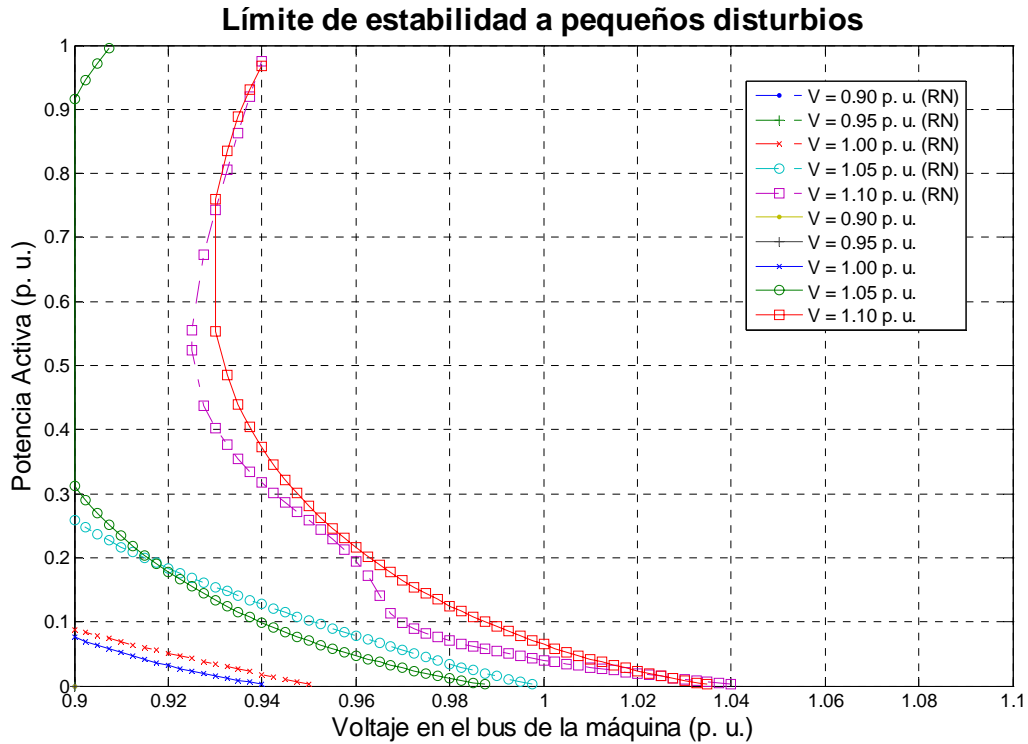
4. 27a – Límites de estabilidad por análisis de eigenvalores y por RN1 con 4 unidades en la capa oculta (LC – 1L)



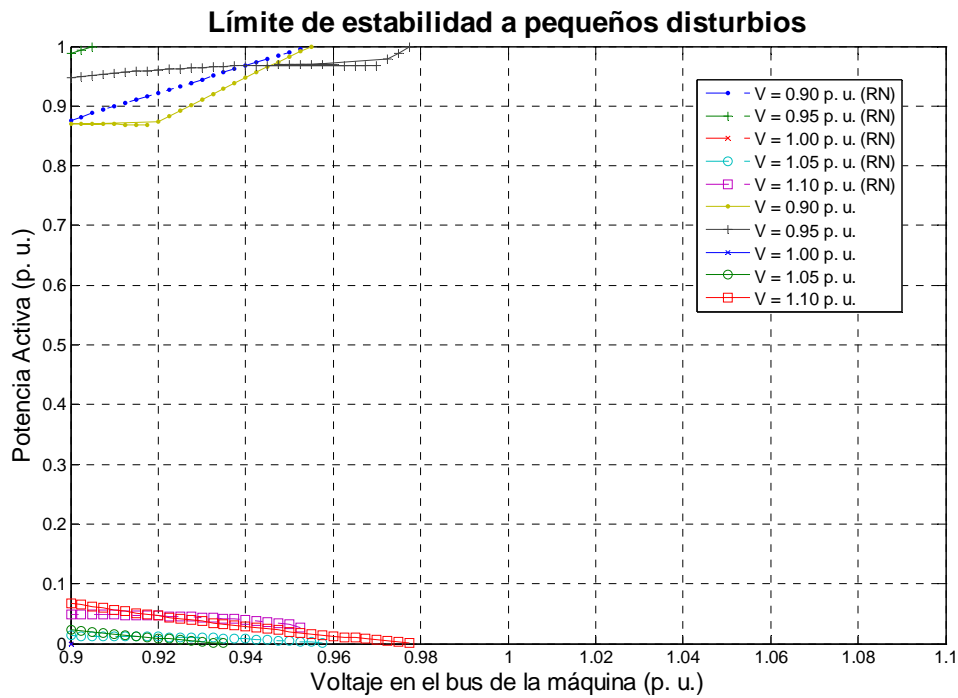
4. 27b – Límites de estabilidad por análisis de eigenvalores y por RN1 con 4 unidades en la capa oculta (LC – 2L)



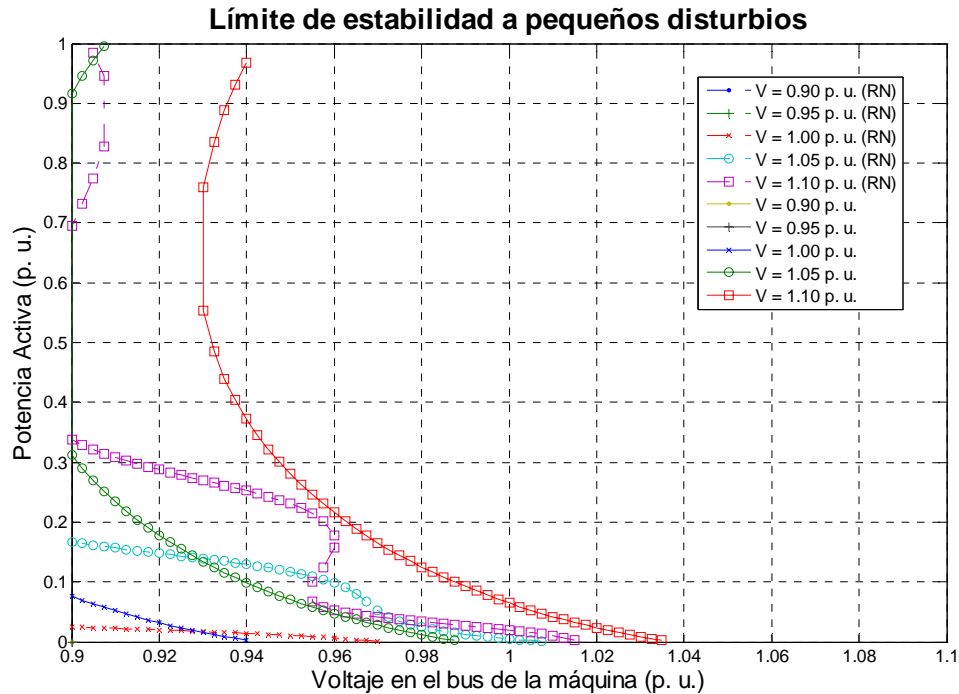
4. 28a – Límites de estabilidad por análisis de eigenvalores y por RN2 con 4 unidades en la capa oculta (LC – 1L)



4. 28b – Límites de estabilidad por análisis de eigenvalores y por RN2 con 4 unidades en la capa oculta (LC – 2L)



4. 29a – Límites de estabilidad por análisis de eigenvalores y por RN3 con 4 unidades en la capa oculta (LC – 1L)



4. 29b – Límites de estabilidad por análisis de eigenvalores y por RN3 con 4 unidades en la capa oculta (LC – 2L)

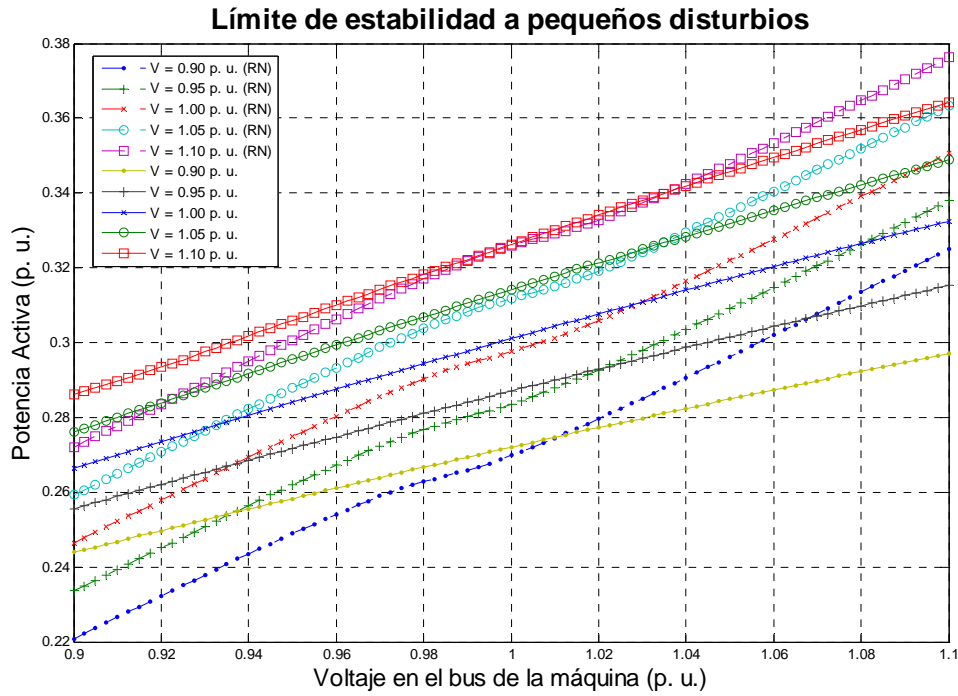
El desempeño de la RN mejora notablemente con bajos niveles de ruido pero parece ser incapaz de dar buenos resultados para mayores niveles de ruido. Entonces la topología de la RN que da mejores resultados es de 4 unidades en la capa oculta.

#### 4.5.3.2. Línea larga

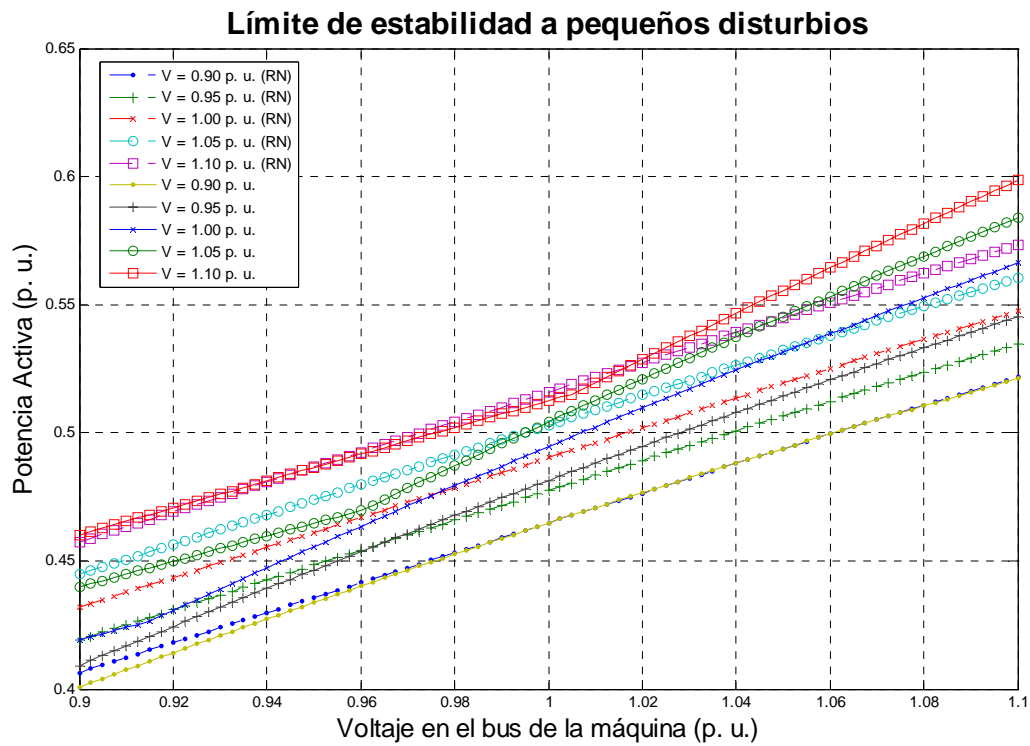
El número de unidades en la capa oculta que se determinó parece dar buenos resultados aún al contaminarse los datos de entrenamiento con ruido, pero para corroborar que los resultados son los mejores probará aumentando y reduciendo el número de unidades en la capa oculta, primero se reducirá a 2 el número de neuronas, y después se aumentará a 4, al final se analizarán los resultados para determinar la confiabilidad de los resultados.

##### 4.5.3.2.1. Reduciendo unidades en la capa oculta

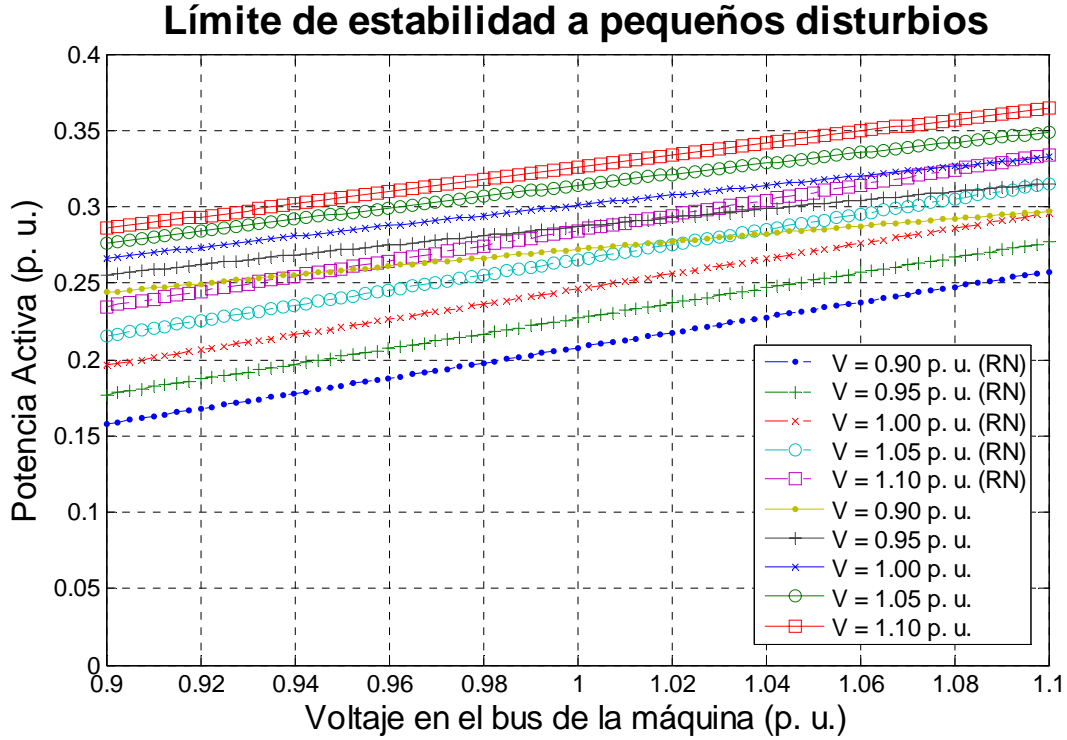
Por la experiencia adquirida del caso LC, se experimenta reduciendo el número de neuronas en la capa oculta. Luego de varias pruebas se obtiene cierta similitud en los parámetros resultantes del entrenamiento, lo que indica que son resultados confiables. Los resultados de la identificación de los límites de estabilidad con 2 unidades en la capa oculta se muestran en las figuras 4. 30a a 4. 32b.



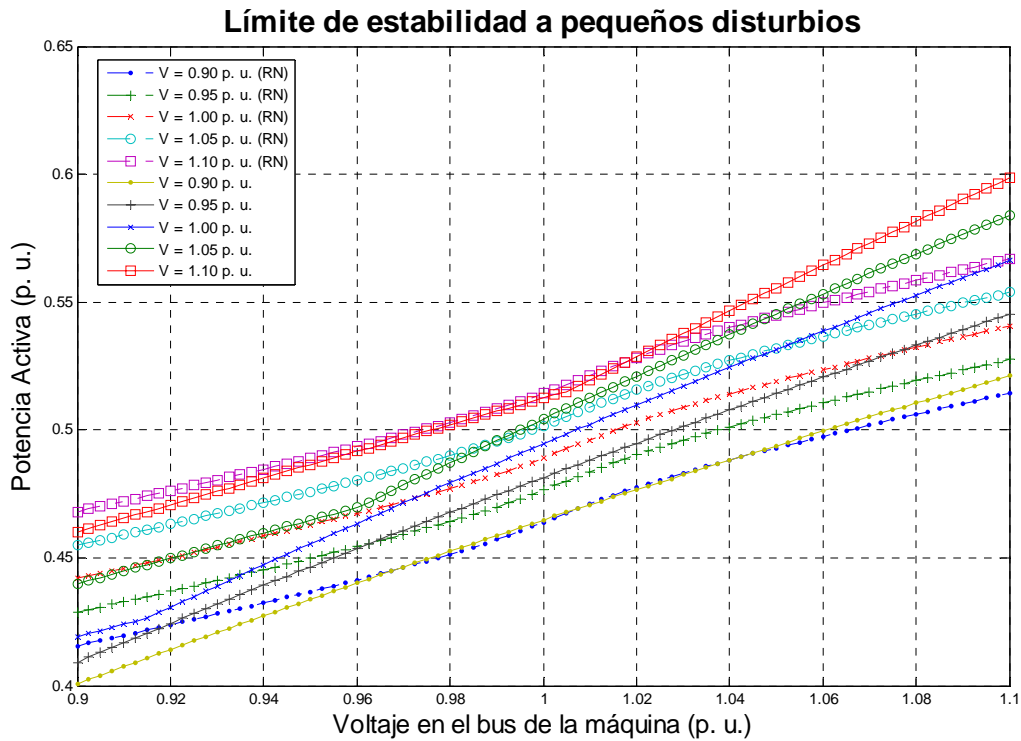
4. 30a – Límites de estabilidad por análisis de eigenvalores y por RN1 con 2 unidades en la capa oculta (LL – 1L)



4. 30b – Límites de estabilidad por análisis de eigenvalores y por RN1 con 2 unidades en la capa oculta (LL – 2L)

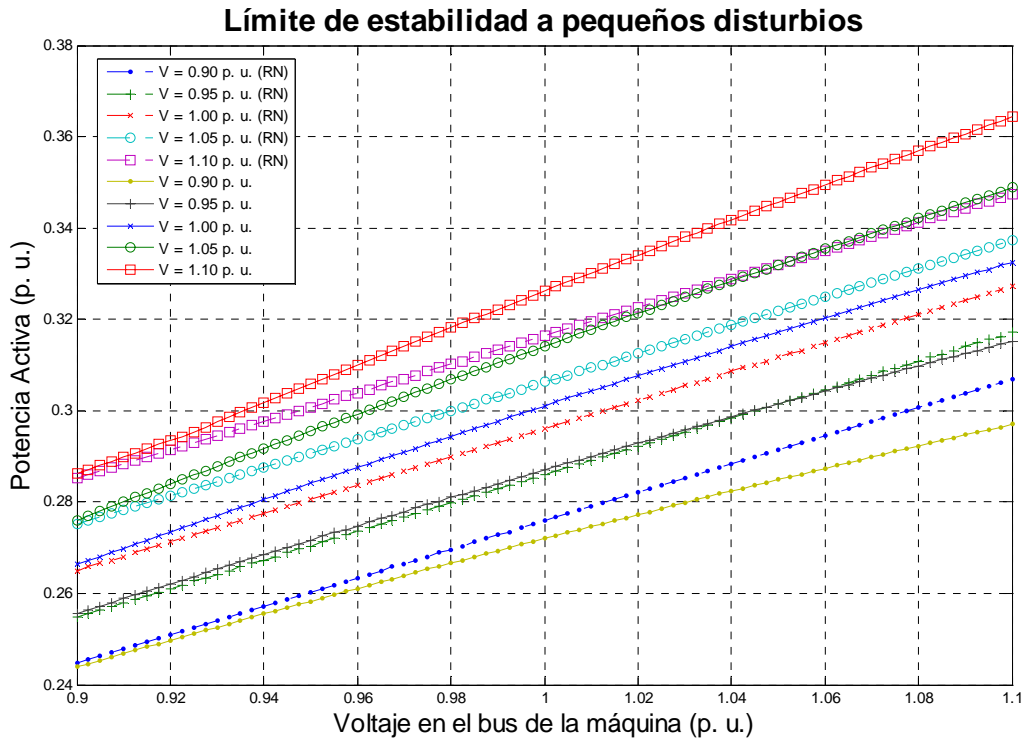


4. 31a – Límites de estabilidad por análisis de eigenvalores y por RN2 con 2 unidades en la capa oculta (LL – 1L)

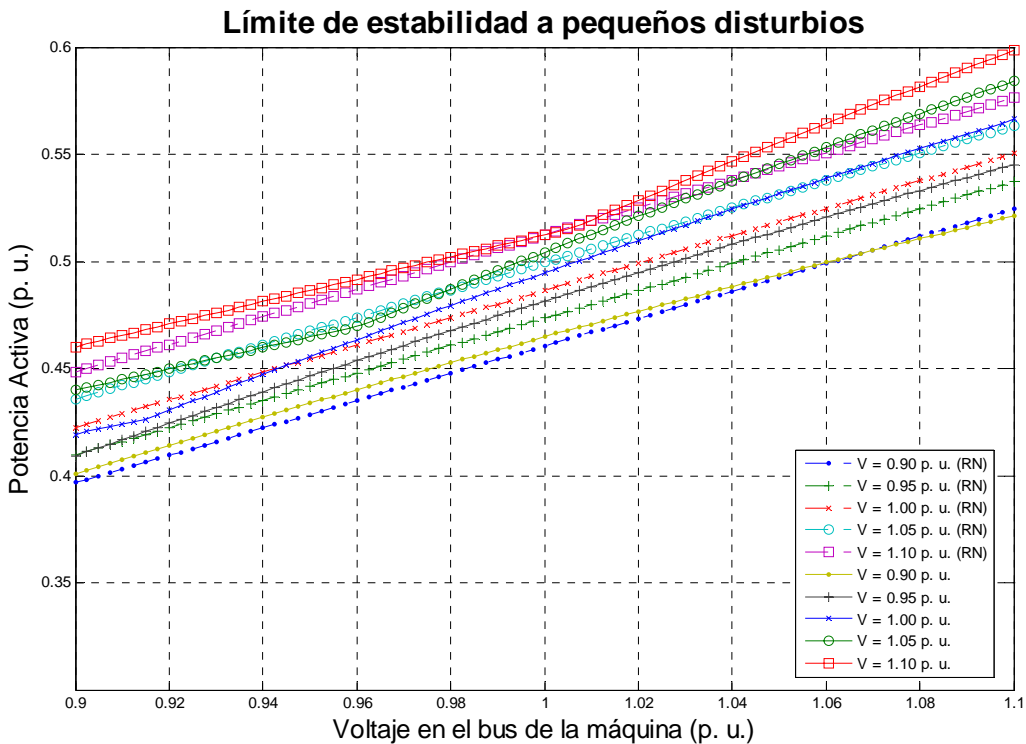


4. 31b – Límites de estabilidad por análisis de eigenvalores y por RN2 con 2 unidades en la capa oculta (LL – 2L)





4. 32a – Límites de estabilidad por análisis de eigenvalores y por RN3 con 2 unidades en la capa oculta (LL – 1L)

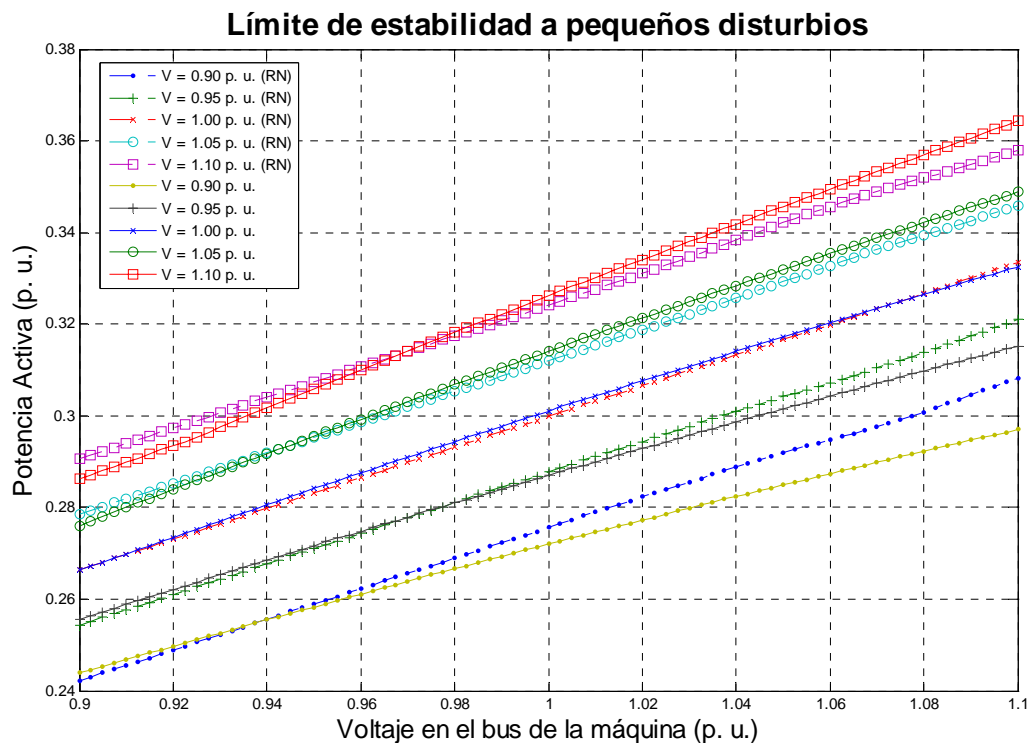


4. 32b – Límites de estabilidad por análisis de eigenvalores y por RN3 con 2 unidades en la capa oculta (LL – 2L)

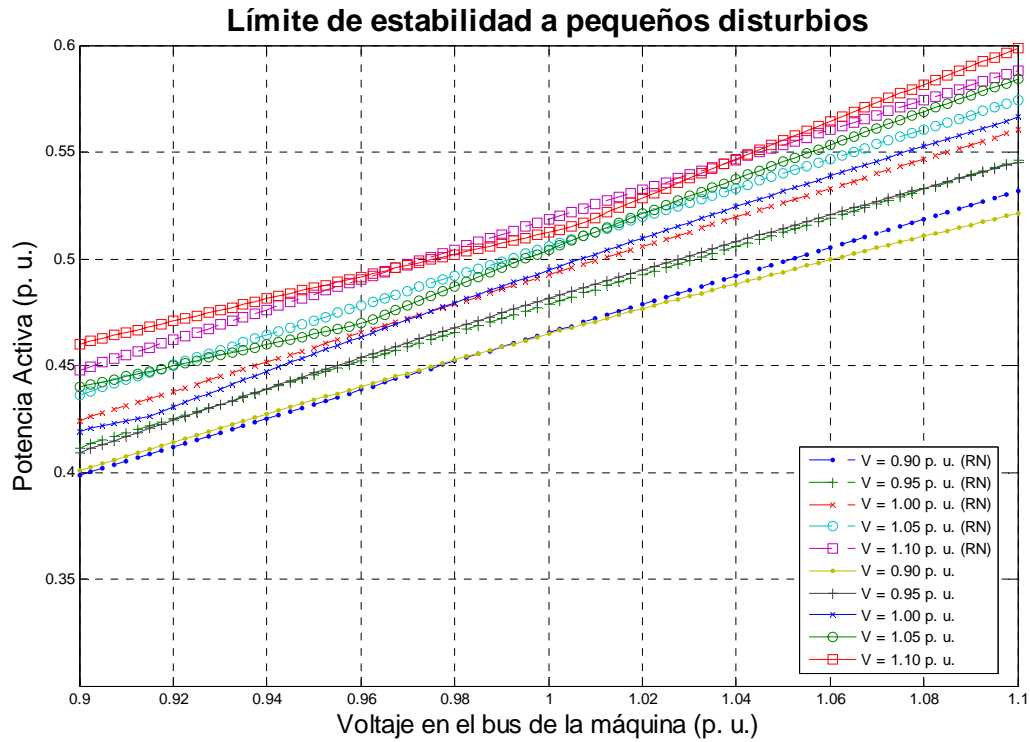
Los límites identificados por la RN están desviados de los obtenidos por análisis de eigenvalores, en general no se puede decir que se tengan mejores resultados de identificación.

#### 4.5.3.2.2. Aumentando unidades en la capa oculta

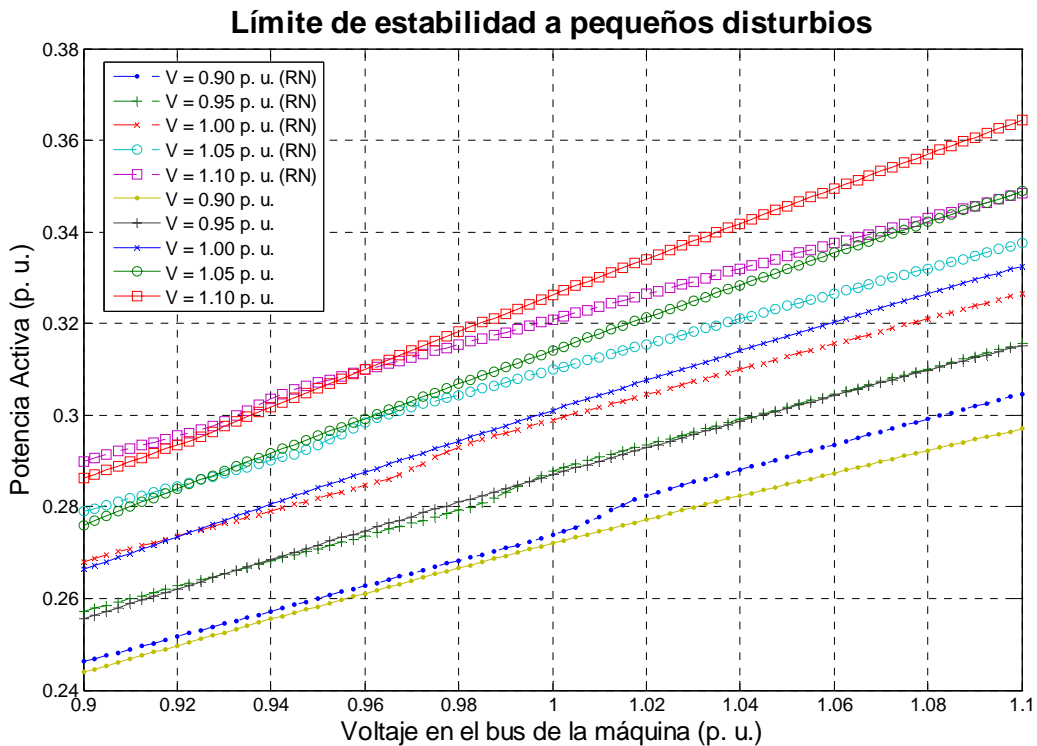
Ahora se experimentara aumentando el número de neuronas con el fin de corroborar si los resultados previos son congruentes o no. Los resultados de la identificación de los límites de estabilidad con 4 unidades en la capa oculta se muestran en las figuras 4. 33a a 4. 35b.



4. 33a – Límites de estabilidad por análisis de eigenvalores y por RN1 con 4 unidades en la capa oculta (LL – 1L)

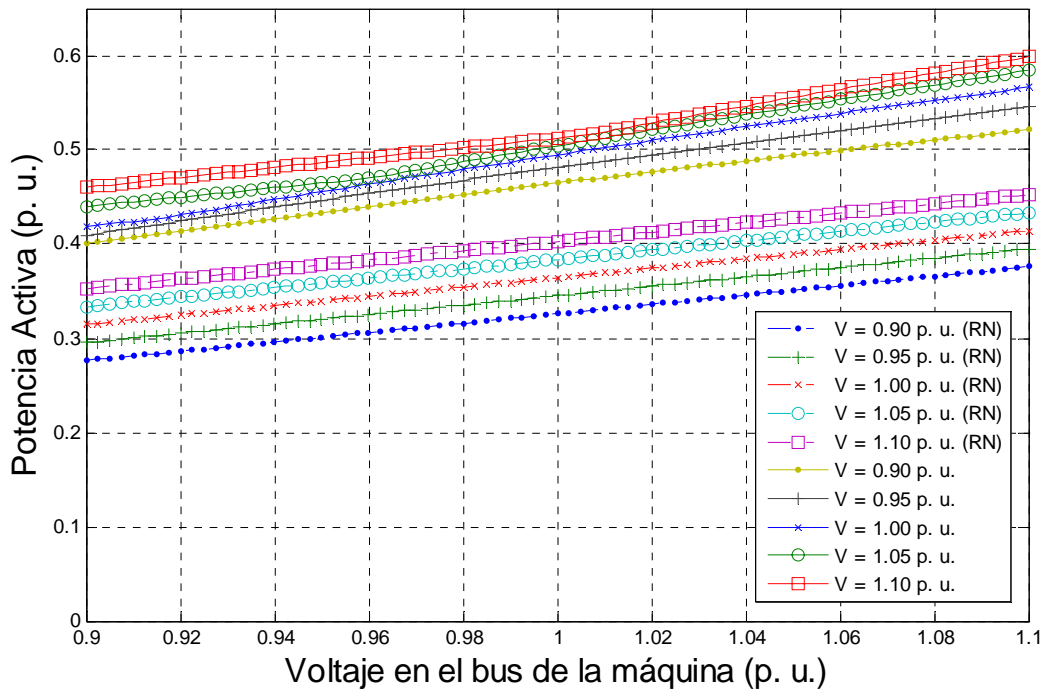


4. 33b – Límites de estabilidad por análisis de eigenvalores y por RN1 con 4 unidades en la capa oculta (LL – 2L)



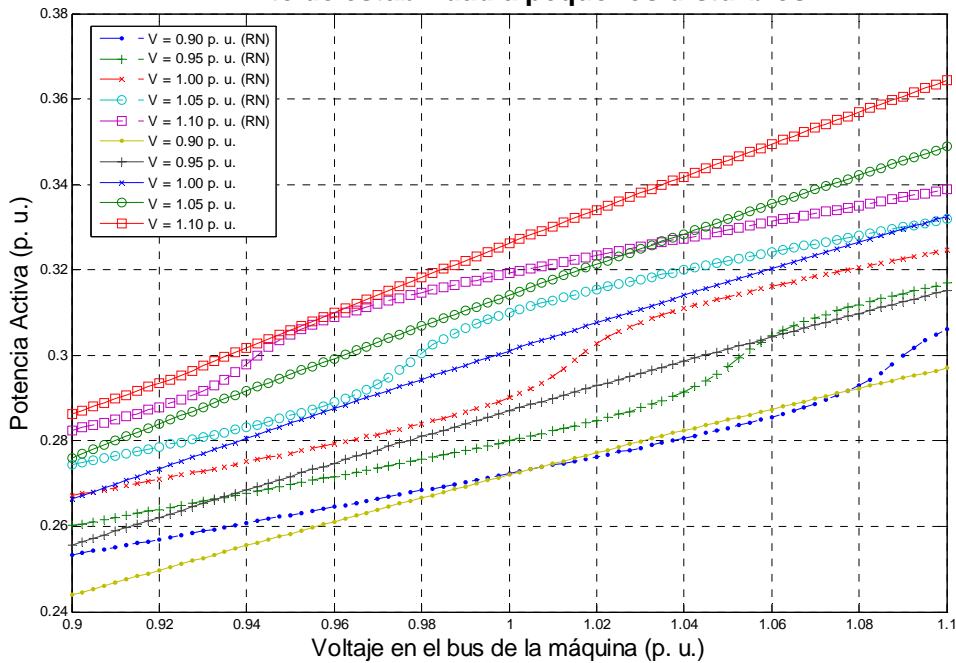
4. 34a – Límites de estabilidad por análisis de eigenvalores y por RN2 con 4 unidades en la capa oculta (LL – 1L)

### Límite de estabilidad a pequeños disturbios

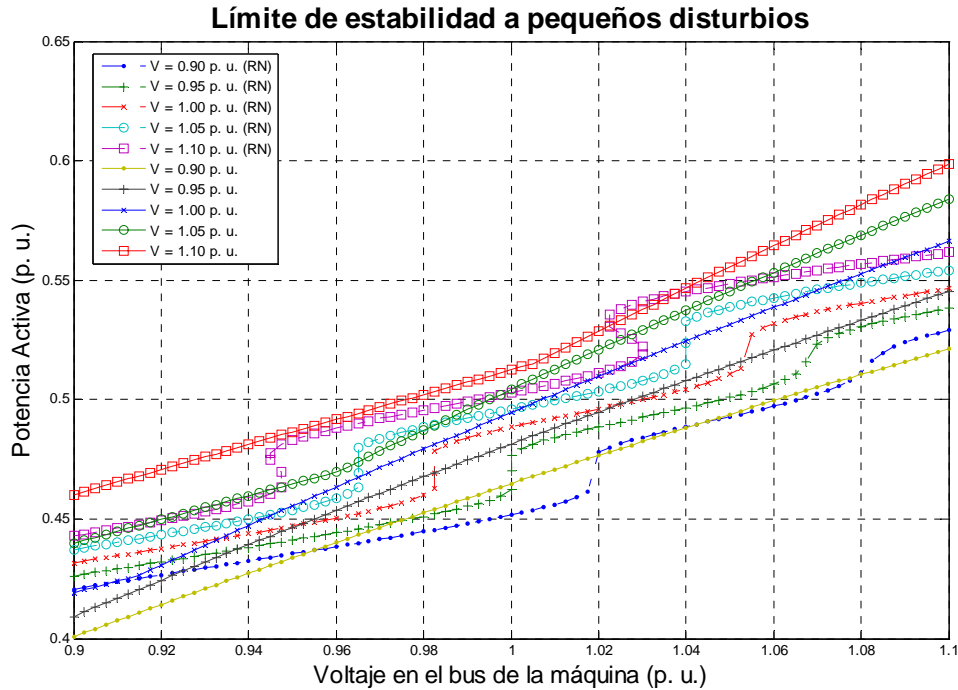


4. 34b – Límites de estabilidad por análisis de eigenvalores y por RN2 con 4 unidades en la capa oculta (LL – 2L)

### Límite de estabilidad a pequeños disturbios



4. 35a – Límites de estabilidad por análisis de eigenvalores y por RN3 con 4 unidades en la capa oculta (LL – 1L)



4. 35b – Límites de estabilidad por análisis de eigenvalores y por RN3 con 4 unidades en la capa oculta (LL – 2L)

Para altos niveles de ruido, los límites identificados por la RN se distorsionan notablemente, pero en general se obtienen buenos resultados; se detecta que si se aumenta el número de parámetros de la RN, los límites tienden a volverse más complejos y desviarse de los reales, sobre todo con mayor ruido.

#### 4.5.3.3. Análisis de resultados

De todos los resultados de esta sección se puede decir que, al reducir el número de unidades de la capa oculta, se reduce también la capacidad de la RN de reconocer los límites que separan los objetos; y también es posible observar que al aumentar el número de neuronas, la RN tiende a clasificar todos los objetos que se le presentan, pero como están contaminados con ruido, lo que aprende no son los rasgos característicos del problema, es decir, la RN aprende un caso específico y no el comportamiento general del problema.

## 5 Conclusiones y Recomendaciones para Trabajos Futuros

La metodología tradicional, para determinar la estabilidad del SEP ante disturbios pequeños, ha sido el análisis basado en los valores propios del modelo linealizado del sistema, pero el cálculo de los eigenvalores de cualquier sistema real no es una tarea fácil; aunque no es problema para cualquier computadora contemporánea, el tiempo que se lleva puede ser esencial para realizar otras acciones de control, sobre todo en sistemas dinámicos como el SEP.

La determinación de la estabilidad ante disturbios pequeños del SEP se ha planteado como una identificación de patrones, donde los objetos a clasificar son las condiciones de operación del sistema, y las clases a las cuales puede pertenecer cada objeto son: “estable” e “inestable”. Visto de esta manera, se permite proponer soluciones alternativas al análisis de eigenvalores, ampliando los recursos disponibles el análisis de la estabilidad.

La identificación de los patrones de estabilidad se realizó con Redes Neuronales, y se logró obtener un sistema de identificación más rápido que el análisis de eigenvalores y con buena precisión.

La importancia que tiene la selección de la topología adecuada para la RN es innegable; un problema en el diseño de sistemas con RN es la escasez de metodologías eficientes para encontrar la mejor estructura, en la bibliografía, solamente se proporcionan ideas generales, que en el mejor de los casos solo sirven para encontrar un punto de partida en la búsqueda de la solución satisfactoria.

Actualmente existe una gran variedad de software disponible para el entrenamiento de las RN, pero no todos comparten las mismas características y están diseñados para trabajar con problemas muy específicos, algunos se han desarrollado para dar solución a problemas financieros, otros a para realizar diagnósticos médicos, etc., por eso mismo no son muy útiles para estudiar el problema de este trabajo. MATLAB® en cambio, con su grupo de herramientas para el manejo de redes neuronales (*Neural Networks toolbox*), provee una buena opción para el desarrollo y análisis del entrenamiento de las RN en general.

Es de esperarse que si la RN se entrena con datos obtenidos de simulación ésta se adapte bien, sin embargo, para una aplicación real se debe considerar que en muchos casos no es posible disponer de buenos modelos, y los datos para entrenar la RN no provienen de simulaciones, sino de mediciones directas o

estimaciones, que son susceptibles a errores o ruido, esta incertidumbre puede causar, como se comprobó en este trabajo, que los resultados cambien completamente.

La adición de ruido en los datos provoca que los patrones se dispersen, y entonces los límites que separan las dos clases se distorsionan y se vuelven más complejos; es difícil realmente visualizar estas distorsiones, ya que se trabaja en un espacio de 4 dimensiones que describen a los objetos: *potencia*, *voltaje en terminales*, *voltaje en bus infinito* y *número de líneas de enlace*, e incluso el *número de líneas de enlace* no es continuo.

Lo que se pretendía era que el sistema fuera sencillo, e identificará los nuevos objetos presentados con base en los límites “reales” y no los distorsionados por el ruido, entonces era deseable que, al entrenar con los datos contaminados con ruido, los parámetros de la RN se ajustarán para identificar los límites “reales”, es decir, que la RN no aprendiera del ruido de los datos. Para resolver este tipo de problemas hay algunas opciones, un ejemplo es el tipo de RN ARTMAP, pero el sistema se vuelve complejo, por lo que se optó por reducir la capacidad de aprendizaje de la RN reduciendo el número de neuronas, así se puede decir que la RN tiende a aprender solamente los rasgos importantes que distinguen las clases.

### **Recomendaciones para trabajo futuro**

En este trabajo se utilizaron las redes neuronales como método de reconocimiento de patrones, el modelo utilizado es de tercer orden y es de esperar que el uso de modelos mas detallados de la máquina, o la utilización de un sistema multimáquina no afecte en los resultados aquí obtenidos; las sugerencias para realizar trabajo futuro son:

- Utilizar modelos mas detallados del sistema
- Utilizar un sistema multimáquina como sistema de prueba
- Utilizar otro método de reconocimiento de patrones

## Referencias

- [1] Yao–Nan Yu, *Electric Power System Dynamics*, Academic Press 1983.
- [2] P. Kundur, *Power System Stability and Control*, McGrawHill, New York, 1994.
- [3] F. P. deMello, C. Concordia, *Concepts of synchronous machine stability as affected by excitation control*. IEEE Transactions on Power Apparatus and Systems, abril 1969.
- [4] Marija Ilić, John Zaborszky, *Dynamics and Control of Large Electric Power Systems*, Wiley-IEEE Press, 2000.
- [5] Chee-Mun Ong, *Dynamic Simulations of Electric Machinery: Using MATLAB®/SIMULINK*, Prentice Hall PTR, 1997
- [6] IEEE Std 1110-2002 (Revision of IEEE Std 1110-1991), *IEEE Guide for Synchronous Generator Modeling Practices and Applications in Power System Stability Analyses*.
- [7] IEEE/CIGRE Joint Task Force on Stability Terms and Definitions, *Definition and Classification of Power System Stability*, IEEE TRANSACTIONS ON POWER SYSTEMS, Vol. 19, No. 2, mayo 2004.
- [8] A. M. Sharaf, T. T. Lie, *ANN Based Classification of synchronous generator stability and loss of excitation*, IEEE Transactions on Energy Conversion, Vol. 9, No. 4, Diciembre 1994.
- [9] D. J. Sobajic, Yoh-Han Pao, *Artificial Neural-Net Based Dynamic Security Assesment for Electric Power Systems*, IEEE Transactions on Power Systems, Vol. 4, No. 1, Febrero 1989.
- [10] R. Fischl, M. Kam, J-C Chow, S. Ricciardi, *Sceening Power System Contingencies Using a Back-Propagation Trained Multiperceptron*, IEEE International Symposium on Circuits and Systems, 1989.
- [11] M. E. Aggouns, M. A. El-Sharkawi, D. C. Park, M. J. Damborg, R. J. Marks II, *Preliminary Results on Using Artificial Neural Networks for Security*



- 
- Assesment, IEEE Transactions on Power Systems, Vol. 6, No. 2, Mayo 1991.
- [12] M. Z. Youssef, P. K. Jain, E. A. Mohamed, M. Orabi, *A Neuro-optimal Control Power System Stabilizer: A Comparative Study*, IEEE CCECE 2004-CCGEI 2004, Niagara, Mayo 2004.
- [13] H. Mori, Y. Tamaru, S. Tsuzuki, *An Artificial Neural-Net Based Technique for Power System Dynamic Stability with the Kohonen Model*, IEEE Transactions on Power Systems, Vol. 7, No. 2, Mayo, 1991.
- [14] C. R. Minussi, M. do C. G. Silvera, *Electric Power Systems Transient Stability Analisis by Neural Networks*, UNESP, Brasil, 1996.
- [15] M. C. G. Silvera, A. D. P. Lotufo, C. R. Minussi, *Transient Stability Analysis of Electrical Power Systems Using a Neural Network Based on Fuzzy ARTMAP*, IEEE Bologna Power Conference, Junio 23-26 2003.
- [16] J. T. Tou, R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley Publishing Company, Inc., 1981.
- [17] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, 2a. edición, Wiley-Interscience, 2001.
- [18] M. I. Torres, A. Sanfeliu, *Pattern Recognition and Applications*, IOS Press Ohmsha, 2000.
- [19] L. I. Kuncheva, *Combining Pattern Classifiers*, Wiley, 2004.
- [20] Shingeo Abe, *Pattern Classification, Neuro-Fuzzy Methods and their comparision*, Springer, 2000.
- [21] S. R. Kulkarni, G. Lugosi, S. S. Venkatesh, *Learning Pattern Classification – A Survey*, IEEE Transactions on Information Theory, Vol. 4, No 6, Octubre 1998.
- [22] Yoh-Han Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley Publishing Company Inc, 1989.
- [23] Carl G. Looney, *Pattern Recognition Using Neural Networks: Theory and Algorithms for Engineers and Scientists*, Oxford University Press Inc, 1997.
- [24] G. P. Zhang, *Neural Networks for Classification: A Survey*, IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and reviews, Vol. 30, No. 4, Noviembre 2000.
- [25] P. J. Werbos, *Backpropagation Trough Time: What It Does and How to Do It*, Proceedings of the IEEE, Vol. 78, No. 10, octubre 1990.

- [26] M. T. Hagan, H. B. Demuth, M. Beale, *Neural Network Design*, PWS Publishing Company – International Thomson Publishing Inc., 1996.
- [27] S. Haykin, *Neural Networks*, New York: Macmillan, 1994.
- [28] J. R. Hiler, V. J. Martínez, *Redes Neuronales Artificiales*, Alfaomega, 2000.
- [29] G. A. Carpenter, S. Grossberg, *A self Organizing Neural Network For Supervised Learning, Recognition, and Prediction*, IEEE Communications Magazine, Septiembre 1992.
- [30] R. M. Gulden, *Mathematical Methods for Neural Networks Analysis and Design*, MIT Press, 1996.
- [31] R. Reed, *Pruning Algorithms – A Survey*, IEEE Transactions on Neural Networks, Vol. 4, No. 5, Septiembre 1993.
- [32] P. F. Baldi, K. Hornik, *Learning in Neural Networks: A Survey*, IEEE Transactions on Neural Networks, Vol. 6, No. 4 julio 1995.

## Apéndices

### A. Parámetros de la Máquina Síncrona

En la tabla A. 1 se muestran los parámetros utilizados en la simulación para la máquina síncrona, todas las cantidades están en p. u. excepto M y las constantes de tiempo que están dadas en segundos.

Tabla A. 1 – Parámetros de la máquina síncrona

Parámetro	Valor
$X_q$	1.881
$X_d$	1.904
$X'_d$	0.312
$M$	5.5294
$T'_{do}$	5.66
$T_A$	0.02
$K_A$	400

### B. Corrientes, Voltajes y Ángulo de Carga en el Sistema Máquina Síncrona Bus-Infinito [1]

Dadas la Potencia Eléctrica  $P$ , la magnitud del Voltaje en Terminales  $|v_t|$  y la magnitud del voltaje del bus infinito  $|v_0|$ :

$$v_0 = |v_0| \angle 0^\circ \quad (\text{B. 1})$$

$$|i| = Y|v_t| + Z^{-1}(|v_t| - |v_0|) \quad (\text{B. 2})$$

$$\varphi = \arccos \frac{P}{|v_t||i|} \quad (\text{B. 3})$$

$$\beta = \arctan \frac{x_q |i| \cos \varphi}{|v_t| + x_q |i| \sin \varphi} \quad (\text{B. 4})$$

$$\alpha = \arctan \left( \frac{X}{R} \right) \quad (\text{B. 5})$$

$$\gamma = \arccos \frac{|v_t|^2 \left( G + \frac{R}{|Z|^2} \right) - P}{v_0 |v_t| |Z|} - \alpha \quad (\text{B. 6})$$

$$\delta = \beta + \gamma \quad (\text{B. 7})$$

$$v_d = |v_t| \sin \beta \quad (\text{B. 8})$$

$$i_d = |i| \sin(\beta + \varphi) \quad (\text{B. 9})$$

$$v_q = |v_t| \cos \beta \quad (\text{B. 10})$$

$$i_q = |i| \cos(\beta + \varphi) \quad (\text{B. 11})$$

$$e'_q = v_q + x'_d i_d \quad (\text{B. 12})$$

### C. Constantes $K_1, K_2, \dots, K_6$ del modelo linealizado [1]

Las constantes  $K_1$  y  $K_2$  se obtiene a partir del Par Eléctrico con la relación:

$$\begin{bmatrix} K_1 \\ K_2 \end{bmatrix} = \begin{bmatrix} 0 \\ i_{q0} \end{bmatrix} + \begin{bmatrix} F_d & F_q \\ Y_d & Y_q \end{bmatrix} \begin{bmatrix} (x_q - x'_d) i_{q0} \\ e'_{q0} + (x_q - x'_d) i_{d0} \end{bmatrix} \quad (\text{C. 1})$$

Las constantes  $K_3$  y  $K_4$  a partir de la ecuación de voltaje de campo eléctrico:

$$K_3 = \frac{1}{1 + (x_d - x'_d) Y_d} \quad (\text{C. 2})$$

$$K_4 = (x_d - x'_d) F_d \quad (\text{C. 3})$$

Las constantes  $K_5$  y  $K_6$  de la magnitud de voltaje terminal:

$$\begin{bmatrix} K_5 \\ K_6 \end{bmatrix} = \begin{bmatrix} 0 \\ v_{q0}/v_{t0} \end{bmatrix} + \begin{bmatrix} F_d & F_q \\ Y_d & Y_q \end{bmatrix} \begin{bmatrix} -x'_d v_{q0}/v_{t0} \\ x_q v_{q0}/v_{t0} \end{bmatrix} \quad (\text{C. 4})$$

Donde:

$$C_1 = \text{Re}(1 + ZY) = 1 + RG - XB \quad (\text{C. 5})$$

$$C_2 = \text{Im}(1 + ZY) = RB + XG \quad (\text{C. 6})$$

$$R_1 = R - C_2 x'_d \quad (\text{C. 7})$$

$$R_2 = R - C_2 x_q \quad (\text{C. 8})$$

$$X_1 = X + C_1 x_q \quad (\text{C. 9})$$

$$X_2 = X + C_1 x'_d \quad (\text{C. 10})$$

$$Z_e^2 = R_1 R_2 + X_1 X_2 \quad (\text{C. 11})$$

$$\begin{bmatrix} Y_d \\ Y_q \end{bmatrix} = \frac{1}{Z_e^2} \begin{bmatrix} X_1 & -R_2 \\ R_1 & X_2 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \quad (\text{C. 12})$$

$$\begin{bmatrix} F_d \\ F_q \end{bmatrix} = \frac{v_0}{Z_e^2} \begin{bmatrix} -R_2 & X_1 \\ X_2 & R_1 \end{bmatrix} \begin{bmatrix} \cos \delta_0 \\ \sin \delta_0 \end{bmatrix} \quad (\text{C. 13})$$

#### **D. Algoritmo de Levenberg-Marquardt**

El algoritmo de Levenberg-Marquardt fue diseñado para dar solución al problema de minimización de la suma de cuadrados de funciones no lineales de la forma:

$$\mathbf{f}(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^m e_j^2(\mathbf{w}) = \frac{1}{2} \mathbf{e}^T(\mathbf{w}) \mathbf{e}(\mathbf{w}) \quad (\text{D. 1})$$

donde  $\mathbf{w} = (w_1, w_2, \dots, w_n)$  y  $\mathbf{e}(\mathbf{w}) = (e_1(\mathbf{w}), e_2(\mathbf{w}), \dots, e_m(\mathbf{w}))$ , además se tiene:

$$\nabla \mathbf{f}(\mathbf{w}) = \mathbf{J}^T(\mathbf{w}) \mathbf{e}(\mathbf{w}) \quad (\text{D. 2})$$

$$\nabla^2 \mathbf{f}(\mathbf{w}) = \mathbf{H}(\mathbf{w}_k) = \mathbf{J}^T(\mathbf{w})\mathbf{J}(\mathbf{w}) + \sum_{j=1}^m e_j(\mathbf{w}) \nabla^2 e_j(\mathbf{w}) \quad (\text{D. 3})$$

La característica de los problemas de los mínimos cuadrados es que el Hessiano puede ser calculado directamente a partir del Jacobiano (si es posible aproximar  $e_j(\mathbf{w})$  por una función lineal) mediante:

$$\mathbf{H}(\mathbf{w}_k) \approx \mathbf{J}^T(\mathbf{w})\mathbf{J}(\mathbf{w}) \quad (\text{D. 4})$$

Minimizando  $\mathbf{e}(\mathbf{w})$  con el método de Newton, se obtiene el método de Gauss-Newton:

$$\Delta \mathbf{w}_{k+1} = -(\mathbf{H}(\mathbf{w}_k))^{-1} \mathbf{J}^T(\mathbf{w}_k) \mathbf{e}(\mathbf{w}_k) \quad (\text{D. 5})$$

Con la ventaja de la no necesidad de calcular segundas derivadas, pero puede presentarse el caso en que la matriz  $\mathbf{H}(\mathbf{w}_k) = \mathbf{J}^T(\mathbf{w}_k)\mathbf{J}(\mathbf{w}_k)$  puede no ser invertible, lo cual se resuelve aproximando el Hessiano con:

$$\mathbf{H}(\mathbf{w}_k) \approx \mathbf{J}^T(\mathbf{w})\mathbf{J}(\mathbf{w}) + \mu \mathbf{I} \quad (\text{D. 6})$$

Con lo cual se obtiene la ecuación del algoritmo de Levenberg:

$$\Delta \mathbf{w}_{k+1} = -(\mathbf{H}(\mathbf{w}_k) + \mu \mathbf{I})^{-1} \mathbf{J}^T(\mathbf{w}_k) \mathbf{e}(\mathbf{w}_k) \quad (\text{D. 7})$$

Si  $\mu$  es cero, se convierte en el método de Gauss-Newton, y si  $\mu$  es muy grande entonces la ecuación se aproxima a la del gradiente descendente, por lo que este algoritmo es robusto (figura D.1).

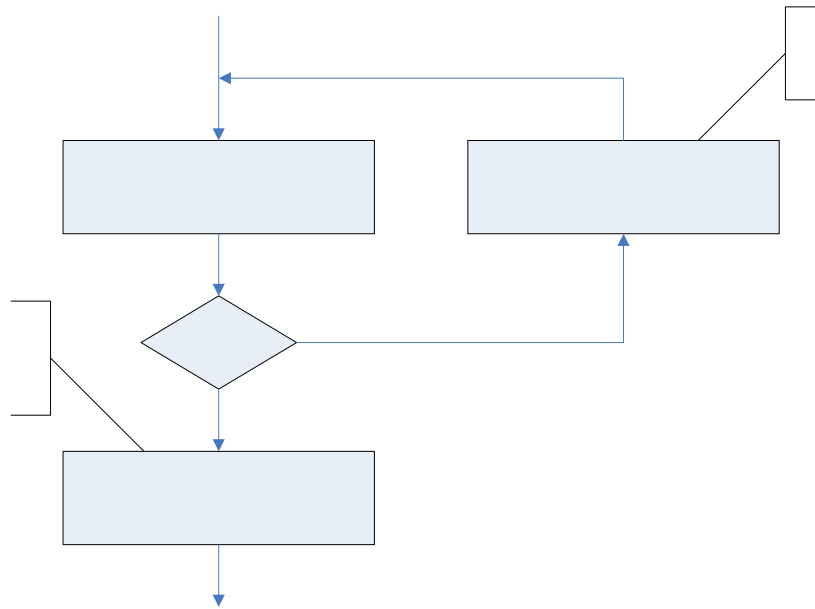


Figura D. 1 – Algoritmo de Levenberg

Marquardt propuso mejorar el algoritmo, incorporando información de la curvatura; de manera que cuando la ecuación se aproxima a la del gradiente descendente, el incremento en los pesos se hace mayor en la dirección en la que el gradiente decrece, con la ecuación:

$$\Delta \mathbf{w}_{k+1} = -(\mathbf{H}(\mathbf{w}_k) + \mu \text{diag}(\mathbf{H}(\mathbf{w}_k)))^{-1} \mathbf{J}^T(\mathbf{w}_k) \mathbf{e}(\mathbf{w}_k) \tag{D. 8}$$

Actualiz  
evaluar

El algoritmo de Levenberg-Marquardt es heurístico, y funciona bastante bien en la práctica; tiene la ventaja de no necesitar del cálculo de las dobles derivadas, por lo que es más rápido que el método de Gauss-Jordan, pero la inversión de la matriz puede requerir demasiados recursos, lo que puede resultar no tan eficiente en ciertas condiciones (comparado con el método del gradiente descendente) especialmente con un elevado número de parámetros.

**E. Algoritmo de la Retro-propagación [26]**

Se ha tenido una

Retomando los conceptos de la RVP, la salida de una capa es la entrada de la siguiente capa, lo cual puede ser visto por la ecuación siguiente:

actualización exitosa y se modifica la ecuación para

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{w}^{m+1} \mathbf{a}^m + \mathbf{b}^{m+1})$$

aproximarla al método

Gauss-Newton

Acepta  
y diam

donde  $M$  es el número de capas en la red, las neuronas en la primer capa reciben entradas externas  $\mathbf{a}^0 = \mathbf{p}$  y las salidas de las neuronas en la ultima capa son consideradas como la salida de la red  $\mathbf{a} = \mathbf{a}^M$ .

El entrenamiento de la red se hace optimizando el índice de desempeño de la red, que normalmente es el *error medio cuadrático*, para el entrenamiento se necesitan juegos de datos entrada-salida conocidos

$$[\mathbf{p}_1, \mathbf{t}_1], [\mathbf{p}_2, \mathbf{t}_2], \dots, [\mathbf{p}_Q, \mathbf{t}_Q]$$

donde  $\mathbf{p}_q$  es una entrada del sistema y  $\mathbf{t}_q$  es el objetivo de la salida de la red, cada entrada es aplicada a la red y se obtiene una salida  $\mathbf{a}$  que se compara con los datos objetivos, el error entre el objetivo y la salida de la red es  $\mathbf{e} = \mathbf{t} - \mathbf{a}$  y el índice de desempeño de la red

$$\mathbf{F}(\mathbf{x}) = \mathbf{E}[\mathbf{e}^T \mathbf{e}] = \mathbf{E}[(\mathbf{t} - \mathbf{a})^T (\mathbf{t} - \mathbf{a})] \quad (\text{D. 9})$$

que se aproxima por medio de la ecuación

$$\hat{\mathbf{F}}(\mathbf{x}) = \mathbf{e}^T(k) \mathbf{e}(k) = (\mathbf{t}(k) - \mathbf{a}(k))^T (\mathbf{t}(k) - \mathbf{a}(k)) \quad (\text{D. 10})$$

donde se calcula el cuadrado del error para cada iteración  $k$ . El gradiente de la función es

$$\hat{\nabla} \mathbf{F}(\mathbf{x}) = \hat{\nabla} \mathbf{e}^T(k) \mathbf{e}(k) \quad (\text{D. 11})$$

$$[\hat{\nabla} \mathbf{e}^T(k) \mathbf{e}(k)]_j = \frac{\partial \mathbf{e}^T(k) \mathbf{e}(k)}{\partial w_{i,j}} \quad (\text{D. 12})$$

El *algoritmo del gradiente descendente* para actualizar los parámetros de la red queda como

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha \frac{\partial \hat{\mathbf{F}}}{\partial w_{i,j}^m} \quad (\text{D. 13})$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial \hat{\mathbf{F}}}{\partial b_i^m} \quad (\text{D. 14})$$



Debido a que para las redes multicapa el error no es una función explícita de los parámetros de la red, las derivadas no pueden ser calculadas fácilmente, como el error es una función indirecta de los parámetros de la red, se usa la regla de la cadena para calcular las derivadas

$$\frac{\partial \hat{\mathbf{F}}}{\partial w_{i,j}^m} = \frac{\partial \hat{\mathbf{F}}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m} \quad (\text{D. 15})$$

$$\frac{\partial \hat{\mathbf{F}}}{\partial b_i^m} = \frac{\partial \hat{\mathbf{F}}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m} \quad (\text{D. 16})$$

el segundo término de la ecuación puede ser calculado fácilmente, porque la entrada de la capa  $m$  es una función explícita de los parámetros de la red

$$n_i^m = \sum_{j=1}^{s^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m \quad (\text{D. 17})$$

por lo tanto

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1} \quad \text{y} \quad \frac{\partial n_i^m}{\partial b_i^m} = 1$$

Definiendo la sensibilidad de  $\hat{\mathbf{F}}$  como  $s_i^m = \frac{\partial \hat{\mathbf{F}}}{\partial n_i^m}$  se obtiene

$$\frac{\partial \hat{\mathbf{F}}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \quad (\text{D. 18})$$

$$\frac{\partial \hat{\mathbf{F}}}{\partial b_i^m} = s_i^m \quad (\text{D. 19})$$

Y el algoritmo del gradiente descendente se puede expresar como

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1} \quad (\text{D. 20})$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m \quad (\text{D. 21})$$

O en forma matricial

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \quad (\text{D. 22})$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m \quad (\text{D. 23})$$

donde

$$\mathbf{s}^m = \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial \hat{\mathbf{F}}}{\partial n_1^m} \\ \frac{\partial \hat{\mathbf{F}}}{\partial n_2^m} \\ \vdots \\ \frac{\partial \hat{\mathbf{F}}}{\partial n_{s^m}^m} \end{bmatrix} \quad (\text{D. 24})$$

Para el cálculo de la sensibilidad se necesita recurrir nuevamente a la regla de la cadena, quedando la sensibilidad como

$$\mathbf{s}^m = \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{n}^m} = \left( \frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \right)^T \frac{\partial \hat{\mathbf{F}}}{\partial \mathbf{n}^{m+1}} \quad (\text{D. 25})$$

donde  $\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}$  es el Jacobiano definido como

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} \equiv \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_1^{m+1}}{\partial n_{s^m}^m} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_2^{m+1}}{\partial n_{s^m}^m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_1^m} & \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_{s^{m+1}}^{m+1}}{\partial n_{s^m}^m} \end{bmatrix} \quad (\text{D. 26})$$

que también es  $\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1} \mathbf{F}^m(\mathbf{n}^m)$

donde

$$\mathbf{W}^{m+1}\mathbf{F}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(n_{s^m}^m) \end{bmatrix} \quad (\text{D. 27})$$

El método de la retro-propagación lleva ese nombre porque la sensibilidad es propagada desde la última capa a lo largo de toda la red:

$$\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \cdots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1$$

con

$$\mathbf{s}^m = \mathbf{F}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1} \text{ para } m = M-1, \dots, 2, 1$$

donde el punto inicial para el cálculo de la sensibilidad  $\mathbf{s}^M$  es calculado con

$$\mathbf{s}^M = -2\mathbf{F}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) \quad (\text{D. 28})$$

El primer paso en el algoritmo de la retro-propagación es propagar la entrada a través de la red:

$$\mathbf{a}^0 = \mathbf{p} \quad (\text{D. 29})$$

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \text{ para } m = 0, 1, \dots, M-1 \quad (\text{D. 30})$$

$$\mathbf{a} = \mathbf{a}^M \quad (\text{D. 31})$$

Después se propaga hacia atrás la sensibilidad de la red:

$$\mathbf{s}^M = -2\mathbf{F}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) \quad (\text{D. 32})$$

$$\mathbf{s}^M \rightarrow \mathbf{s}^{M-1} \rightarrow \cdots \rightarrow \mathbf{s}^2 \rightarrow \mathbf{s}^1$$

Finalmente se actualizan los parámetros de la red:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m (\mathbf{a}^{m-1})^T \quad (\text{D. 33})$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \alpha \mathbf{s}^m \quad (\text{D. 34})$$

## F. Códigos de Programa

En esta sección se presenta el código de los programas utilizados para la determinación de la estabilidad mediante el análisis de eigenvalores, para la obtención de la estabilidad mediante la RN entrenada y para la búsqueda de la topología con base a lo descrito en la sección 3.3.5.2. El entrenamiento de la RN se realizó en MATLAB®, pero la simulación del sistema, la generación de la base de datos y la construcción de la RN se realizó en FORTRAN 90; la interacción entre los distintos programas se realiza mediante archivos de texto para entrada/salida de datos.

### F. 1. Generación de base de datos para entrenamiento de la RN (FORTRAN 90)

Este programa contiene el modelo linealizado del sistema, con él se determina la estabilidad del mismo cambiando el punto de operación y evaluando los eigenvalores, si alguno tiene parte real no negativa se escribe en el archivo de salida (base de datos para entrenamiento) una condición inestable; si todos los eigenvalores tienen parte real negativa entonces se escribe una condición estable en el archivo de salida. El archivo de salida es utilizado después por un programa de MATLAB® entrenar la RN y determinar cual es la que ofrece mejor desempeño.

#### data.f90

```

MODULE PAR_SISTEMA
  COMPLEX Z, Y
  REAL XQ, XD, XDP, M, TDOP, TA, KA
  REAL KC, T, T1, T2
END MODULE PAR_SISTEMA

MODULE EST_SISTEMA
  REAL Q, VD, VQ, ID, IQ, EQP, BET, DELTA, GAMA, PHI, MI
END MODULE EST_SISTEMA

PROGRAM FEATURE

  USE NUMERICAL_LIBRARIES
  USE DFPORT

  USE EST_SISTEMA
  USE PAR_SISTEMA

```

```

CHARACTER (12) AR_SAL
INTEGER, PARAMETER :: N=4
INTEGER I, EST, LIN
REAL P, V0, VT
REAL RV0, RVT, RP
REAL A(N,N)
COMPLEX EIGEN(N)

AR_SAL='fe2linar.dat'

CALL LEER

OPEN(UNIT=7, FILE=AR_SAL)
WRITE (UNIT=6, FMT='(A)')'ARCHIVO DE SALIDA [ '//AR_SAL// ]'

WRITE (UNIT=6, ADVANCE='NO', FMT='(A)')'% DE NIVEL DE RUIDO PARA LAS
ENTRADAS (Vo, Vt, P): '
READ (UNIT=5, FMT='(F5.2, F5.2, F5.2)')RV0, RVT, RP

WRITE(UNIT=7, FMT='(A)')FDATE()
WRITE(UNIT=7, FMT='(A, F5.2, A)')'RUIDO PARA Vo: ', RV0, '%'
WRITE(UNIT=7, FMT='(A, F5.2, A)')'RUIDO PARA Vt: ', RVT, '%'
WRITE(UNIT=7, FMT='(A, F5.2, A)')'RUIDO PARA P: ', RP, '%'
WRITE(UNIT=7, FMT='(A6, A3, A3, A2, A12)')'LINEAS', 'V0', 'VT', 'P',
'ESTABILIDAD'
DO LIN=2,1,-1
  IF(LIN==1) THEN
    Z=Z*2
  END IF
  DO V0=0.9,1.1,0.01
    DO VT=0.9,1.1,0.01
      DO P=0,1.04,0.05
        IF(MATA(P, V0, VT, A, N)) THEN
          CALL EVLRG (N, A, N, EIGEN)
          EST=1
          DO I=1,N
            IF (REAL(EIGEN(I))>=0) EST=0
          END DO
          WRITE(UNIT=7, FMT='(I2, F5.2, F5.2,
F5.2, I2)')LIN-1, RUIDO(V0, RV0), RUIDO(VT, RVT), RUIDO(P, RP), EST
        END IF
      END DO
    END DO
  END DO
END DO
END PROGRAM FEATURE

SUBROUTINE LEER
  USE PAR_SISTEMA
  REAL R, X, G, B
  CHARACTER (12) AR_PAR
  WRITE (UNIT=6, ADVANCE='NO', FMT='(A)')'ARCHIVO DE PARAMETROS
(param.dat): '
  READ (UNIT=5, FMT='(A12)')AR_PAR
  OPEN(UNIT=8, STATUS='OLD', ERR=100, FILE=AR_PAR)
  WRITE (UNIT=6, FMT='(A)')AR_PAR//'...OK'
  GOTO 102

```

```

100  AR_PAR='param.dat'
      OPEN(UNIT=8, FILE=AR_PAR)
102  WRITE  (UNIT=6,   FMT='(A)')'PARAMETROS   TOMADOS   DEL   ARCHIVO
      ['//AR_PAR//']'
      READ (UNIT=8, FMT='(F7.4, F7.4, F7.4, F7.4)')R, X, G, B
      READ (UNIT=8, FMT='(F7.4, F7.4, F7.4, F7.4, F7.4, F7.4, F7.2)')XQ, XD,
XDP, M, TDOP, TA, KA
      READ (UNIT=8, FMT='(F7.4, F7.3, F7.3, F7.3)')KC, T, T1, T2
      CLOSE(UNIT=8)
      Z=CMPLX(R,X)
      Y=CMPLX(G,B)
END SUBROUTINE LEER

```

```

LOGICAL FUNCTION MATA(P, Vo, V, A, N)
  USE PAR_SISTEMA
  USE EST_SISTEMA
  INTEGER, INTENT(IN)      :: N
  REAL, INTENT(IN)        :: P, Vo, V
  REAL, INTENT(OUT), DIMENSION(N,N)  :: A
  REAL R, X, G, B, R1, R2, X1, X2, Z2, YD, YQ, FD, FQ
  REAL K(6)
  COMPLEX II, VT, V0
!  CALCULO DE CONDICIONES INICIALES
  R=REAL(Z)
  X=AIMAG(Z)
  G=REAL(Y)
  B=AIMAG(Y)
  V0=CMPLX(Vo,0)
  ALFA=ATAN(X/R)
  IF (ABS(SQRT(R*R+X*X))*(V**2*(G+R/(R*R+X*X))-P)/Vo/V)>1) THEN
    MATA=.FALSE.
    RETURN
  END IF
  GAMA=ACOS(SQRT(R*R+X*X)*(V*V*(G+R/(R*R+X*X))-P)/Vo/V)-ALFA
  VT=CMPLX(V*COS(GAMA),V*SIN(GAMA))
  II=Y*VT+(VT-V0)/Z
  MI=SQRT(REAL(II)**2+AIMAG(II)**2)
  IF (P/V>MI) THEN
    MATA=.FALSE.
    RETURN
  END IF
  PHI=ACOS(P/V/MI)
  Q=-V*MI*SIN(PHI)
  BET=ATAN(XQ*MI*COS(PHI)/(V+XQ*MI*SIN(PHI)))
  DELTA=BET+GAMA
  VD=V*SIN(BET)
  VQ=V*COS(BET)
  ID=MI*SIN(BET+PHI)
  IQ=MI*COS(BET+PHI)
  EQP=VQ+XDP*ID
!  CALCULO DE CONSTANTES
  C1=1.0+R*G-X*B
  C2=X*G+R*B
  R1=R-C2*XDP
  R2=R-C2*XQ
  X1=X+C1*XQ

```

```

X2=X+C1*XDP
Z2=R1*R2+X1*X2
FD=V0*( -R2*COS(DELTA)+X1*SIN(DELTA) )/Z2
FQ=V0*(X2*COS(DELTA)+R1*SIN(DELTA) )/Z2
YD=(C1*X1-C2*R2)/Z2
YQ=(C1*R1+C2*X2)/Z2
K(1)=FD*(XQ-XDP)*IQ+FQ*(EQP+(XQ-XDP)*ID)
K(2)=IQ+YD*(XQ-XDP)*IQ+YQ*(EQP+(XQ-XDP)*ID)
K(3)=1.0/(1.0+(XD-XDP)*YD)
K(4)=(XD-XDP)*FD
K(5)=(-FD*XDP*VQ+FQ*XQ*VD)/V
K(6)=(VQ-YD*XDP*VQ+YQ*XQ*VD)/V
! FORMANDO LA MATRIZ A
A(1:N,1:N)=0.
A(1,2)=-K(1)/M
A(1,3)=-K(2)/M
A(2,1)=377.0
A(3,2)=-K(4)/TDOP
A(3,3)=-1.0/(TDOP*K(3))
A(3,4)=1.0/TDOP
A(4,2)=-KA*K(5)/TA
A(4,3)=-KA*K(6)/TA
A(4,4)=-1.0/TA
IF (N==6) THEN
    A(4,6)=KA/TA
    A(5,2)=-K(1)/M
    A(5,3)=-K(2)/M
    A(5,5)=-1/T
    A(6,2)=-KC*K(1)*T1/M/T2
    A(6,3)=-KC*K(2)*T1/M/T2
    A(6,5)=KC*(1-T1/T)/T2
    A(6,6)=-1/T2
END IF
MATA=.TRUE.
END FUNCTION MATA

REAL FUNCTION RUIDO(DATO, NIVEL)
REAL, INTENT(IN) :: DATO, NIVEL
REAL DX, RND
DX=DATO*NIVEL/100
CALL RANDOM_NUMBER(RND)
RUIDO=DATO+DX*(2*RND-1)
END FUNCTION RUIDO

```

## F. 2. Determinación de Límites de Estabilidad de la Máquina Síncrona Bus-Infinito a Pequeños Disturbios por Análisis de Eigenvalores (FORTRAN 90)

Con este programa se realiza la obtención de los eigenvalores del modelo linealizado y se determina la estabilidad para distintos puntos de operación, al encontrarse un cambio de condición de estabilidad (de estable a inestable o viceversa) se encuentra el límite de estabilidad. Los parámetros del sistema se leen

de un archivo de entrada (de nombre param.dat predeterminadamente) y se genera el archivo de salida limitee.dat.

### limev.f90

```

MODULE PAR_SISTEMA
  COMPLEX Z, Y
  REAL XQ, XD, XDP, M, TDOP, TA, KA
  REAL KC, T, T1, T2
END MODULE PAR_SISTEMA

MODULE EST_SISTEMA
  REAL Q, VD, VQ, ID, IQ, EQP, BET, DELTA, GAMA, PHI, MI
END MODULE EST_SISTEMA

PROGRAM LIM_EST
  CALL LEER
  CALL LIMITE
END PROGRAM LIM_EST

SUBROUTINE LEER
  USE PAR_SISTEMA
  REAL R, X, G, B
  CHARACTER (12) AR_PAR
  WRITE (UNIT=6, ADVANCE='NO', FMT='(A)') 'ARCHIVO DE PARAMETROS
(param.dat): '
  READ (UNIT=5, FMT='(A12)') AR_PAR
  OPEN(UNIT=8, STATUS='OLD', ERR=100, FILE=AR_PAR)
  WRITE (UNIT=6, FMT='(A)') AR_PAR// '...OK'
  GOTO 102
100  AR_PAR='param.dat'
  OPEN(UNIT=8, FILE=AR_PAR)
102  WRITE (UNIT=6, FMT='(A)') 'PARAMETROS TOMADOS DEL ARCHIVO
[ '//AR_PAR// ']'
  READ (UNIT=8, FMT='(F7.4, F7.4, F7.4, F7.4)') R, X, G, B
  READ (UNIT=8, FMT='(F7.4, F7.4, F7.4, F7.4, F7.4, F7.4, F7.2)') XQ, XD,
XDP, M, TDOP, TA, KA
  READ (UNIT=8, FMT='(F7.4, F7.3, F7.3, F7.3)') KC, T, T1, T2
  CLOSE(UNIT=8)
  Z=CMPLX(R,X)
  Y=CMPLX(G,B)
END SUBROUTINE LEER

SUBROUTINE LIMITE
  USE NUMERICAL_LIBRARIES
  USE DFPORT
  USE PAR_SISTEMA
  INTEGER, PARAMETER :: N=4

  REAL A(N,N)
  COMPLEX EIGEN(N)
  INTEGER I

  INTEGER LIN, EST, ESIS
  REAL P, V0, VT
  CHARACTER(12) AR_SAL

```



```

REAL(8) SEGI, SEGF

AR_SAL='limitee.dat'
WRITE (UNIT=6, FMT='(A)') 'ARCHIVO DE SALIDA [ '//AR_SAL// ]'
OPEN(UNIT=7, FILE=AR_SAL)
WRITE(UNIT=7, FMT='(A6, A3, A3, A2)') 'LINEAS', 'V0', 'VT', 'P'
SEGI=RTC()
DO LIN=1, 0, -1
  IF(LIN.EQ.0) THEN
    Z=Z*2
  END IF
  DO V0=0.9,1.1,0.01
    DO VT=0.9,1.1,0.0025
      EST=0
      DO P=0.0,1.0,0.0001
        IF(MATA(P, V0, VT, A, N)) THEN
          ESIS=1
          CALL EVLRG (N, A, N, EIGEN)
          DO I=1,N
            IF (REAL(EIGEN(I))>=0) THEN
              ESIS=0
            END IF
          END DO
          IF(EST.NE.ESIS) THEN
            IF (P>0) WRITE(UNIT=7, FMT='(I2,
F5.2, F7.4, F7.4)')LIN, V0, VT, P
            EST=ESIS
          END IF
        END IF
      END DO
    END DO
  END DO
  SEGF=RTC()
  WRITE(UNIT=7, FMT='(A, F9.4)') 'TIEMPO: ', SEGF-SEGI
  WRITE(UNIT=7, FMT='(A)') FDATE()
END SUBROUTINE LIMITE

LOGICAL FUNCTION MATA(P, Vo, V, A, N)
  USE PAR_SISTEMA
  USE EST_SISTEMA
  INTEGER, INTENT(IN)      :: N
  REAL, INTENT(IN)        :: P, Vo, V
  REAL, INTENT(OUT), DIMENSION(N,N)  :: A
  REAL R, X, G, B, R1, R2, X1, X2, Z2, YD, YQ, FD, FQ
  REAL K(6)
  COMPLEX II, VT, V0
  ! CALCULO DE CONDICIONES INICIALES
  R=REAL(Z)
  X=AIMAG(Z)
  G=REAL(Y)
  B=AIMAG(Y)
  V0=CMPLX(Vo,0)
  ALFA=ATAN(X/R)
  IF (ABS(SQRT(R*R+X*X))*(V**2*(G+R/(R*R+X*X))-P)/Vo/V)>1) THEN
    MATA=.FALSE.
  RETURN

```

```

END IF
GAMA=ACOS(SQRT(R*R+X*X)*(V*V*(G+R/(R*R+X*X))-P)/Vo/V)-ALFA
VT=CMPLX(V*COS(GAMA),V*SIN(GAMA))
II=Y*VT+(VT-V0)/Z
MI=SQRT(REAL(II)**2+AIMAG(II)**2)
IF (P/V>MI) THEN
    MATA=.FALSE.
    RETURN
END IF
PHI=ACOS(P/V/MI)
Q=-V*MI*SIN(PHI)
BET=ATAN(XQ*MI*COS(PHI)/(V+XQ*MI*SIN(PHI)))
DELTA=BET+GAMA
VD=V*SIN(BET)
VQ=V*COS(BET)
ID=MI*SIN(BET+PHI)
IQ=MI*COS(BET+PHI)
EQP=VQ+XDP*ID
! CALCULO DE CONSTANTES
C1=1.0+R*G-X*B
C2=X*G+R*B
R1=R-C2*XDP
R2=R-C2*XQ
X1=X+C1*XQ
X2=X+C1*XDP
Z2=R1*R2+X1*X2
FD=V0*(-R2*COS(DELTA)+X1*SIN(DELTA))/Z2
FQ=V0*(X2*COS(DELTA)+R1*SIN(DELTA))/Z2
YD=(C1*X1-C2*R2)/Z2
YQ=(C1*R1+C2*X2)/Z2
K(1)=FD*(XQ-XDP)*IQ+FQ*(EQP+(XQ-XDP)*ID)
K(2)=IQ+YD*(XQ-XDP)*IQ+YQ*(EQP+(XQ-XDP)*ID)
K(3)=1.0/(1.0+(XD-XDP)*YD)
K(4)=(XD-XDP)*FD
K(5)=(-FD*XDP*VQ+FQ*XQ*VD)/V
K(6)=(VQ-YD*XDP*VQ+YQ*XQ*VD)/V
! FORMANDO LA MATRIZ A
A(1:N,1:N)=0.
A(1,2)=-K(1)/M
A(1,3)=-K(2)/M
A(2,1)=377.0
A(3,2)=-K(4)/TDOP
A(3,3)=-1.0/(TDOP*K(3))
A(3,4)=1.0/TDOP
A(4,2)=-KA*K(5)/TA
A(4,3)=-KA*K(6)/TA
A(4,4)=-1.0/TA
IF (N==6) THEN
    A(4,6)=KA/TA
    A(5,2)=-K(1)/M
    A(5,3)=-K(2)/M
    A(5,5)=-1/T
    A(6,2)=-KC*K(1)*T1/M/T2
    A(6,3)=-KC*K(2)*T1/M/T2
    A(6,5)=KC*(1-T1/T)/T2
    A(6,6)=-1/T2
END IF

```

```

      MATA=.TRUE.
!      WRITE(UNIT=6, FMT='(A)')'MATA'
END FUNCTION MATA

```

### F. 3. Determinación de Límites de Estabilidad de la Máquina Síncrona Bus-Infinito a Pequeños Disturbios por RN (FORTRAN 90)

Con este programa se realiza la determinación de los límites de estabilidad del sistema de prueba al igual que en el programa anterior, pero la determinación de la estabilidad del punto de operación se realiza mediante la RN entrenada, cuyos parámetros están contenidos en el archivo de entrada (parrn.dat es el nombre predeterminado).

#### limrn.f90

```

MODULE PAR_RN
  INTEGER      UO
  INTEGER, DIMENSION(2) :: F_ACT
  DOUBLE PRECISION, DIMENSION(:, :), ALLOCATABLE :: LW1
  DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: LW2
  DOUBLE PRECISION, DIMENSION(:), ALLOCATABLE :: B1
  DOUBLE PRECISION B2, UMBRAL
END MODULE PAR_RN

PROGRAM LIM_EST
  USE PAR_RN
  CALL LEER
  CALL LIMITE
END PROGRAM LIM_EST

SUBROUTINE LEER
  USE PAR_RN
  CHARACTER (12) AR_PAR
  WRITE (UNIT=6, ADVANCE='NO', FMT='(A)')'ARCHIVO DE PARAMETROS DE LA
RN(parrn.dat): '
  READ (UNIT=5, FMT='(A12)')AR_PAR
  OPEN(UNIT=8, STATUS='OLD', ERR=100, FILE=AR_PAR)
  WRITE (UNIT=6, FMT='(A)')AR_PAR//'...OK'
  GOTO 102
100  AR_PAR='parrn.dat'
     OPEN(UNIT=8, FILE=AR_PAR)
102  WRITE (UNIT=6, FMT='(A)')'PARAMETROS TOMADOS DEL ARCHIVO
[ '//AR_PAR// ]'
     READ (UNIT=8, FMT='(I2, I1, I1)')UO, F_ACT(1), F_ACT(2)
     ALLOCATE(LW1(UO,4))
     ALLOCATE(LW2(UO))
     ALLOCATE(B1(UO))
     DO I=1,UO
       READ (UNIT=8, FMT='(F14.8, F14.8, F14.8, F14.8)')LW1(I,1),
LW1(I,2), LW1(I,3), LW1(I,4)
     END DO
     DO I=1,UO
       READ (UNIT=8, FMT='(F14.8)')LW2(I)

```

```

        END DO
        DO I=1,UO
            READ (UNIT=8, FMT='(F14.8)')B1(I)
        END DO
        READ (UNIT=8, FMT='(F14.8)')B2
        CLOSE(UNIT=8)
    END SUBROUTINE LEER

SUBROUTINE LIMITE
    USE DFPORT
    USE PAR_RN
    INTEGER RED
    DOUBLE PRECISION, DIMENSION(4) :: V_ENT
    INTEGER LIN, EST
    REAL P, V0, VT
    CHARACTER(12) AR_SAL
    REAL(8) SEGI, SEGF
    AR_SAL='limiter.dat'
    WRITE(UNIT=6, ADVANCE='NO', FMT='(A)')'UMBRAL DE SALIDA (0->1.0000): '
    READ (UNIT=5, FMT='(F6.4)')UMBRAL
    WRITE (UNIT=6, FMT='(A)')'ARCHIVO DE SALIDA [ '//AR_SAL// ]'
    OPEN(UNIT=7, FILE=AR_SAL)
    WRITE(UNIT=7, FMT='(A6, A3, A3, A2)')'LINEAS', 'V0', 'VT', 'P'
    SEGI=RTC()
    DO LIN=1,0,-1
        DO V0=0.9,1.1,0.01
            DO VT=0.9,1.1,0.0025
                EST=0
                DO P=0,1.0,0.0001
                    V_ENT(1)=REAL(LIN)
                    V_ENT(2)=V0
                    V_ENT(3)=VT
                    V_ENT(4)=P
                    IF(EST.NE.RED(V_ENT)) THEN
                        IF (P>0) WRITE(UNIT=7, FMT='(I2, F5.2,
F7.4, F7.4)')LIN, V0, VT, P
                            EST=RED(V_ENT)
                        END IF
                    END DO
                END DO
            END DO
        END DO
    END DO
    SEGF=RTC()
    WRITE(UNIT=7, FMT='(A, F9.4)')'TIEMPO: ', SEGF-SEGI
    WRITE(UNIT=7, FMT='(A)')FDATE()
END SUBROUTINE LIMITE

INTEGER FUNCTION RED(ENT)
    USE PAR_RN
    DOUBLE PRECISION, DIMENSION(4), INTENT(IN) :: ENT
    DOUBLE PRECISION, DIMENSION(UO) :: A1
    DOUBLE PRECISION A2
    INTEGER I
    A1 = 0
    DO J = 1, 4
        A1 = A1 + LW1(1:UO, J) * ENT(J)
    END DO

```

```

A1 = A1 + B1
!FUNCIÓN DE ACTIVACIÓN CAPA OCULTA
IF (F_ACT(1).EQ.1) THEN
    DO J = 1, UO
        IF (A1(J)>80) THEN
            A1(J)=1
        ELSE IF (A1(J)<-80) THEN
            A1(J)=0
        ELSE
            A1(J) = 1 / (1 + EXP(-A1(J)))
        END IF
    END DO
END IF
A1 = A1 * LW2
A2 = 0
DO J = 1, UO
    A2 = A2 + A1(J)
END DO
A2 = A2 + B2
! FUNCIÓN DE ACTIVACIÓN CAPA DE SALIDA
IF (F_ACT(2).EQ.1) THEN
! SI -80<A2<80 HACER A2 = 1 / (1 + EXP(-A2)) SI NO ENTONCES TOMA 0 O +1
    IF (A2>80) THEN
        A2=1
    ELSE IF (A2<-80) THEN
        A2=0
    ELSE
        A2 = 1 / (1 + EXP(-A2))
    END IF
END IF
IF(A2>UMBRAL) THEN
    RED=1
ELSE
    RED=0
END IF
END FUNCTION RED

```

#### F. 4. Búsqueda de Topología para RN de 3 Capas con n-Unidades en la Capa Oculta y 1 Unidad en la Capa de Salida (MATLAB®)

Con este programa se realiza la búsqueda sistematizada de la topología para la RN, con base en los criterios previamente establecidos; se requiere una base de datos para entrenamiento (feature.dat) y se guarda el espacio de trabajo de MATLAB® (RN.mat) con la información del proceso y los parámetros de la RN encontrada.

##### busqueda.m

```

clear
clc
data=load('feature.dat');
ENT=data(:,1:4)';
SAL=data(:,5)';

```

```

clear data;
perf=1e-6;
inic=13;
umin=6;
umax=18;
du=2;
epmaxg=300;
epmax=epmaxg.*ones((umax-umin)/du+1,1);
grad=1e-16;
for u=umin:du:umax
    rn=newff(minmax(ENT),[u,1],{'logsig' 'purelim'},'trainlm');
    rn.trainParam.min_grad=grad;
    rn.trainParam.goal=perf;
    rn.trainParam.epochs=epmax((u-umin)/du+1);
    rn.trainParam.show=round(epmax((u-umin)/du+1)/4);
    for i=1:inic
        rn=init(rn);
        iw11t=rn.IW{1,1};
        lw21t=rn.LW{2,1};
        b1t=rn.b{1,1};
        b2t=rn.b{2,1};
        fprintf('Inicializacion %i/%i para topología [4-%i-1]\n',i,inic,u);
        [rn,tr]=train(rn,ENT,SAL);
        error=tr.perf;
        size(error);
        ept=ans(1,2);
        if ((ept-1)<epmax((u-umin)/du+1))&(error(ept)<perf)
            epmax((u-umin)/du+1)=ept-1;
            rn.trainParam.epochs=epmax((u-umin)/du+1);
            rn.trainParam.show=round(epmax((u-umin)/du+1)/4);
        end
        if ((ept-1)<=epmaxg)&(error(ept)<perf)
            epmaxg=ept-1;
            uo=u;
            iw11i=iw11t;
            lw21i=lw21t;
            bli=b1t;
            b2i=b2t;
            iw11f=rn.IW{1,1};
            lw21f=rn.LW{2,1};
            b1f=rn.b{1,1};
            b2f=rn.b{2,1};
        end
    end
    save RN.mat;
end
fprintf('Topología [4-%i-1] óptima\n',uo);
rn=newff(minmax(ENT),[uo,1],{'logsig' 'purelim'},'trainlm');
rn.trainParam.min_grad=grad;
rn.trainParam.epochs=epmax((uo-umin)/du+1);
rn.trainParam.show=round(epmax((uo-umin)/du+1)/7);
rn.trainParam.goal=perf;
rn.IW{1,1}=iw11i;
rn.LW{2,1}=lw21i;
rn.b{1,1}=bli;
rn.b{2,1}=b2i;
[rn,tr]=train(rn,ENT,SAL);

```

```
save RN.mat;
```

## F. 5. Generación de archivo de parámetros de RN (MATLAB®)

A partir de la información generada por el programa de búsqueda de topología, se cargan los datos y se genera un archivo de salida conteniendo los parámetros de la RN que se encuentra con mejor desempeño, ese archivo es el que se utiliza como archivo de entrada para el programa de determinación de los límites de estabilidad por RN, presentado en esta misma sección de apéndices.

### paramrn.m

```
%Guarda los parametros de la red neuronal en un archivo
clc
fprintf(1,'RED NEURONAL [4-%i-1] ÓPTIMA\n',uo);
fprintf('Función de activación de la capa oculta:
%s\n',rn.layers{1}.transferFcn);
fprintf('Función de activación de la capa de salida:
%s\n',rn.layers{2}.transferFcn);
fprintf(1,'Matriz de pesos de la capa oculta (%i unidades):\n',uo);
switch size(ENT,1)
    case 3
        fprintf('%8.2f%8.2f%8.2f\n',iw11f');
    case 4
        fprintf('%8.2f%8.2f%8.2f%8.2f\n',iw11f');
end
fprintf(1,'Vector de pesos de la capa de salida (1 unidad):\n');
fprintf(1,'%8.2f',lw21f);
fprintf(1,'\n');
fprintf(1,'Vector de sesgo de la capa oculta (%i unidades):\n',uo);
fprintf(1,'%8.2f\n',blf');
fprintf(1,'Sesgo de la capa de salida (1 unidad):\n');
fprintf(1,'%8.2f',b2f);
fprintf(1,'\n');
fid=fopen('parrn.dat','wt');
fprintf(fid,'%s\n',rn.layers{1}.transferFcn);
fprintf(fid,'%s\n',rn.layers{2}.transferFcn);
fprintf(fid,'%i\n',uo);
switch size(ENT,1)
    case 3
        fprintf(fid,'%14.8f%14.8f%14.8f\n',iw11f');
    case 4
        fprintf(fid,'%14.8f%14.8f%14.8f%14.8f\n',iw11f');
end
fprintf(fid,'%14.8f\n',lw21f);
fprintf(fid,'%14.8f\n',blf');
fprintf(fid,'%14.8f',b2f);
fprintf(fid,'\n');
fclose(fid);
```