Sistema de Limitación de Central Nuclear Atucha II con Redes Neuronales Artificiales

PROYECTO FINAL INTEGRADOR

Ingeniería Nuclear con Orientación en Aplicaciones

Alumna: Rosana Alelí Gómez de la Peña

Tutores :

Ing. Luis Javier Lencina

Ing. Martín Sebastián Silva

UNSAM
UNIVERSIDAD NACIONAL DE SAN MARTÍN

Comisión Nacional de Energía Atómica

UNIVERSIDAD NACIONAL DE SAN MARTIN

12 de julio de 2019

Ing. Luis Javier Lencina e Ing Martín Sebastían Silva

DECLARAN

que Rosana Alelí Gómez de la Peña, , ha realizado bajo nuestra supervisión el Proyecto Final Integrador de la Carrera de Ingeniería Nuclear con Orientación en Aplicaciones titulado:

Sistema de Limitación de Central Nuclear Atucha II con Redes Neuronales Artificiales,

Una vez revisado, autorizamos el comienzo de los trámites para su presentación.

.....

.....

Yo, Rosana Alelí Gómez de la Peña,

DECLARO

Mi autoría del trabajo que se presenta en la memoria de este Proyecto Final Integrador de la Carrera de Ingeniería Nuclear con Orientación en Aplicaciones que tiene por título:

Sistema de Limitación de Central Nuclear Atucha II con Redes Neuronales Artificiales,

Lo cual firmo,

.....

en la UNIVERSIDAD NACIONAL DE SAN MARTIN

12/07/2019

A Sofía, el motor de mi existencia.

Resumen

En el marco del cambio del elemento combustible de uranio natural (UN) a uranio levemente enriquecido (ULE) de la Unidad II de la Central Nuclear Atucha (CNA-UII), se planteó la necesidad de mejorar el método de predicción del margen de potencia con que cuentan los elementos combustibles.

En este proyecto se realizó el estudio de la utilización de Redes Neuronales Artificiales (ANN del inglés *Artificial Neural Network*) para el Sistema de Limitación de la planta, ya que el algoritmo actual es poco flexible a las variaciones provocadas por el ULE.

Se evaluó las técnicas de Retropropagación del Error y Recocido Simulado para el entrenamiento de distintas estructuras de ANN, con el fin de predecir el margen de operación al fenómeno de apartamiento de ebullición nucleada en el Sistema de Limitación a partir de las señales de los detectores internos en el núcleo, tanto para estados de planta con UN como con ULE. Se realizaron comparaciones de ganancia de margen relativo en los casos estudiados

Asímismo se estudió la robustez de estos métodos al presentarse fallas en los detectores *in-core*.

Por último se desarrolló un programa para evaluar en tiempo real el comportamiento de la mejor ANN obtenida con las señales de planta.

Abstract

Within the framework of the change of the nuclear fuel from natural uranium (UN after "Uranio Natural") to low enriched uranium (ULE after "Uranio Levemente Enriquecido") in Unit II of the Atucha Nuclear Power Plant (CNA-UII), the need to improve the power margin prediction method arose.

In the present project the use of Artificial Neural Networks (ANN) for the Limitation System of the plant was studied, since the current algorithm may not be flexible enough to cope with the variations caused by the ULE.

The techniques of Error Backpropagation and Simulated Annealing were evaluated as training methods for different ANN structures, in order to predict the operation margin to the phenomenon of Departure from Nucleated Boiling (DNB) in the Limitation System with the signals of the in-core detectors, both for plant states with UN and with ULE. Relative margin gains of the studied cases were compared.

The robustness of these methods was studied by simulating several failures in the in-core detectors signals.

Finally a program was developed to evaluate the real-time performance of the optimum ANN fed with actual in-core signals.

Agradecimientos

En primer lugar quiero agradecer a mis viejos, ya que nada de esto hubiera sido posible sin su apoyo y comprensión durante todos estos años. No me alcanza la vida para devolver todo el esfuerzo que hicieron para ayudarme con Sofía en esta etapa de mi vida.

A mi compañero de vida Dani, por su paciencia y calidez durante todos estos años. A mi hija, por adaptarse a todos los cambios, a pesar de su corta edad.

A mis amigos, por estar siempre (y tolerarme). A Anna por ser mi cable a tierra todo este tiempo.

Al grupo de Termohidráulica de Na-Sa, en especial a Luis Lencina por su paciencia y dedicación. Al grupo de Neutrónica de Na-Sa, especialmente a Martín Silva, por la oportunidad de hacer este trabajo y por las enseñanzas adquiridas en este tiempo.

A mis compañeros de cohorte, que me acompañaron en el camino. Gracias por la buena onda, voy a extrañar sus ocurrecias.

A Ro y a Pichu, gracias por tanto y perdón por tan poco.

Al personal de Instituto Dan Beninson, a Valeria Kaplan, a Natalia Bidart. Gracias por la paciencia. A Vero, muchas gracias por el apoyo brindado en estos años. A los Javis y a Adriel por su buena predisposición siempre.

A las Autoridades y profesores del Instituto Dan Beninson, en especial al Dr. Ricardo Ramos por esta gran oportunidad.

Gracias a Pablo Matias Degimertjis, te fuiste de este mundo antes de poder leer esto. Gracias por el impulso, donde quiera que estés. Nunca me hubiera animado de no ser por vos.

Prefacio

El Proyecto Final Integrador tiene como finalidad integrar los conocimientos adquiridos a lo largo de la carrera de Ingeniería Nuclear con Orientación en Aplicaciones dictada en el Instituto de Tecnología Nuclear Dan Beninson.

El Proyecto elegido tiene asimismo un desarrollo de temas que son extra a los contenidos de la carrera. En el transcurso del mismo se han adquirido conocimientos de Inteligencia Artificial, existiendo la posibilidad de poder aplicarlos como una potencial solución en la digitalización del sistema de limitación de una central nuclear. Esto es para quien escribe, un tema de sumo interés por lo novedoso de la metodología.

En la siguientes páginas se podrá encontrar la información ordenada en partes, las cuales a su vez tienen capítulos. La primera parte es la Introducción, donde se explica a grandes rasgos la funcionalidad de la centrales nucleares de potencia en el mundo y en el país. También se pone en conocimiento, a partir de antecedentes en otras partes del mundo, cómo una herramienta de la Inteligencia Artificial como las Redes Neuronales Artificiales se aplican para distintos sistemas en reactores nucleares. Y por último se describe el objetivo del Proyecto.

En la segunda parte se profundiza en el detalle de la central. Aquí se realiza una descripción de la Central Nuclear Atucha II en cuanto a su estructura y funcionamiento normal, y de los sistemas de vigilancia. Se hace hincapié en el Sistema de Limitación existente, el diseño termohidráulico del mismo y los fenómenos involucrados. Luego, se describe brevemente la función de la computadora de procesos de la planta (READAT) y de los programas que contiene, profundizando en el funcionamiento de los que fueron utilizados durante el desarrollo del proyecto.

La tercera parte, se enfoca en las Redes Neuronales Artificiales, que son la herramienta de inteligencia artificial utilizada en el proyecto, su estructura, su funcionamiento, y las distintas metodologías empleadas para su entrenamiento.

En la cuarta parte, se describen los procedimientos utilizados en el desarrollo de los algoritmos, y en el tratamiento de la información de la planta.

En la quinta parte se presentan los resultados obtenidos y sus discusiones y en la sexta y última parte se dan a conocer las conclusiones el proyecto.

Índice general

ÍNDICE DE FIGURAS

Índice de Tablas

Parte I

Introducción

Centrales Nucleares de Potencia

La tecnología por si sola no basta. También tenemos que poner el corazón.

Jane Goodall

Desde sus orígenes, la humanidad ha buscado aumentar su eficiencia en el uso de los recursos para su beneficio. Hoy en día gracias a todos los conocimientos adquiridos a lo largo de la historia, y con un avance creciente de la ciencia y tecnología a nivel mundial es posible hacer uso de la energía proveniente del interior de las reacciones nucleares en diversas aplicaciones tecnológicas. Esto abarca campos tan diversos como la salud, la industria, la agronomía y la investigación, entre muchos otros. Uno de los usos más importantes de la energía nuclear que existen en la actualidad es la generación de energía eléctrica en las centrales nucleares de potencia.

1.1 Principio de funcionamiento de las Centrales Nucleares

La función de una central nuclear de potencia es, principalmente, la generación de energía eléctrica. Se emplean generadores que proporcionan este tipo de energía a partir de movimientos rotativos. El generador está conectado a una turbina, cuyos álabes entran en movimiento por el vapor proveniente del calentamiento de agua. Es aquí donde entra en juego la energía nuclear, ya que la fuente térmica utilizada para el calentamiento del agua es justamente un combustible nuclear, que en su interior contiene átomos pesados que liberan energía por procesos de fisión nuclear.

1.1.1 Principios físicos

Los núcleos de los átomos pesados son inestables, y es por ello que se escinden en dos o más partes para adquirir un estado de mayor estabilidad o menor energía interna. Esto ocurre de manera espontánea en ciertos átomos, o siendo inducido en otros como ocurre con el uranio-235 (^{235}U).

El (^{235}U) se denomina físil, ya que puede inducirse la fisión mediante la colisión de un neutrón con el núcleo, como se esquematiza en la Figura §1.1. Al fisionarse, parte del exceso de energía contenida en el estado anterior es liberada como radiación. Esto incluye emisión de fotones y de distintas partículas con o sin carga, entre ellos los neutrones de alta energía.

Estos neutrones pueden ser absorbidos o capturados por otros núcleos pesados, dando pie a otras reacciones nucleares de fisión, y así sucesivamente. Esto se conoce como reacción en cadena.

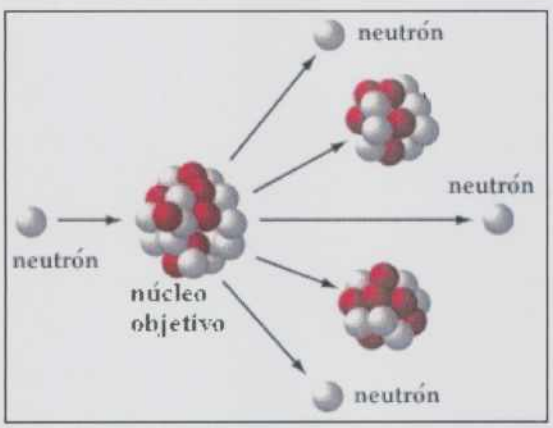

Figura 1.1: Esquema del proceso de fisión del núcleo inducida por la colisión de un neutrón.

En el caso del uranio-238 (^{238}U) que es fértil, cuando el neutrón es capturado por el núcleo, este se transforma en un núcleo inestable que decae hasta generar un núcleo físil, el plutonio-239 (^{239}Pu), el cual se escinde de la forma y con los efectos descriptos anteriormente, lo que conlleva a una nueva liberación de neutrones.

La probabilidad de ocurrencia de que un neutrón sea capturado por el núcleo a partir de su colisión está relacionada con la sección eficaz microscópica de captura σ_c del núcleo, la cual a su vez depende, entre otras variables, de la energía del neutrón. Los neutrones pueden clasificarse según su energía en fríos, térmicos, epitérmicos, y rápidos. Si la energía del neutrón es menor que 0.025 eV, se denominan fríos; en cambio si su energía es mayor a 1 MeV se denominan rápidos, entre estas energías se encuentran los neutrones térmicos y epitérmicos.

Si a partir de un proceso de fisión se genera una mayor cantidad de fisiones de otros núcleos en la generación siguiente, entonces estamos ante un estado hipercrítico.
En oposición, si a partir de una cierta cantidad de procesos de fisión de una generación se producen tal cantidad de neutrones que provoquen una cantidad menor de fisiones en la generación siguiente, entonces estamos en un estado subcrítico, que tiende a cesar con el tiempo.

El estado que se busca en un reactor nuclear es el estado crítico, el cual es un proceso autosostenido, donde el número de fisiones se mantienen constantes en el tiempo.

En los combustibles hechos de dióxido de uranio, como se ha mencionado anteriormente, se encuentran los isótopos ^{235}U y ^{238}U. La relación de la concentración de estos isótopos es de 0,0711% si se trata de uranio natural. Si esta relación es mayor pero menor que el 1% (generalmente entre 0,8 y 0,9%) se lo considera uranio levemente enriquecido (ULE). Los neutrones producidos en los procesos de fisión se encuentran en el rango de neutrones rápidos.

El valor de σ_c del ^{235}U es mayor cuando el neutrón incidente se encuentra en el rango térmico, por debajo de los 1 eV. Por ello es necesario que el combustible se encuentre inmerso en un medio moderador que termalice estos neutrones, para así sostener la reacción en cadena.

En otras palabras es necesario contar con una sustancia cuyas propiedades sean tales, que al colisionar el neutrón, no sea absorbido por la misma, sino que se disperse, disminuyendo su energía en cada colisión producida con este medio.

A partir de la ecuación de momento se puede hallar la masa óptima para que la termalización sea lo más eficiente posible, la cual es coincidente con la masa del neutrón.

A nivel mundial, en los reactores convencionales construidos hasta el momento se encuentra muy difundido el uso de agua liviana como moderador que al contener átomos de hidrógeno en su interior, cumplen con esta condición. Sin embargo también tiene la capacidad de absorber los neutrones lo cual es contraproducente con los efectos deseados.

El agua pesada, que en su fórmula molecular contiene dos átomos de deuterio y uno de oxígeno, tiene la ventaja de ser un menor absorbente que el agua liviana. En contraparte, tiene la desventaja de que su capacidad de termalizar los neutrones es menor y necesita mayores distancias para disminuir la energía hasta el valor deseado.

Se mantiene una circulación de este medio en el circuito primario para refrigerar los combustibles y poder extraer la energía térmica que será utilizada para generar el vapor que dará movimiento a los álabes de la turbina, generando energía eléctrica.

Dados todos estos aspectos, en el núcleo de un reactor nuclear, el combustible debe estar inmerso en un medio moderador, y a su vez debe ser refrigerado. En el diseño del reactor se deben tener en cuenta los materiales absorbentes y el tipo de combustible de tal forma que la reacción en cadena se mantenga en un estado de criticidad durante su operación.

1.1.2 Seguridad Nuclear

La seguridad de las instalaciones nucleares juega un rol esencial en el uso de la energía nuclear, por ello se han desarrollado diversos sistemas de control y regulación para que las distintas centrales operen de forma segura.

Generalmente en las centrales existe un sistema de regulación y uno de protección, que tienen como principal objetivo controlar la operación del reactor y llevarlo a un estado seguro en caso de producirse alguna desviación en las condiciones normales de operación.

El sistema de regulación es el que se encarga de controlar continuamente las propiedades del reactor mientras se encuentre en Operación Normal. Cuando existan desviaciones que lleven al reactor a un estado fuera de lo normal y que no puedan ser revertidas por el sistema de regulación, el sistema de protección actúa apagando el reactor. Estos sistemas son esenciales en la vida útil de un reactor nuclear, otorgando la seguridad en operación que tanto se busca.

1.2 Argentina, un país nuclear

En nuestro país existen centrales nucleares de potencia desde el año 1974, cuando se conectó a la matriz energética a la Central Nuclear Atucha I (CNA-I). La misma se encuentra ubicada en Buenos Aires, a poco más de 100 km de la Ciudad Autónoma de Buenos Aires, capital del país. Esta central contiene un reactor del tipo PHWR del inglés *Pressurized Heavy Water Reactor* cuya traducción al español es reactor de agua pesada presurizada. Esto significa que contiene agua pesada como moderador y refrigerante.

Originalmente utilizaba combustible de dióxido de uranio natural, pero a fines de la década de los '90 se realizó la transición a dióxido de uranio levemente enriquecido (ULE). La tecnología empleada en esta central es de diseño alemán desarrollado por la empresa Siemmens.

Mas adelante, en el año 1984 se puso en funcionamiento la Central Nuclear Embalse (CNE) emplazada en la costa sur del embalse del Río Tercero en la provincia de Córdoba, a 665 metros sobre el nivel del mar. Esta central posee un reactor del tipo CANDU (CANada Deuterio Uranio), que es un reactor de agua pesada presurizada con dióxido de uranio natural como combustible. La tecnología fue desarrollada por la empresa canadiense Atomic Energy of Canada Limited. La CNE además cuenta con

una extensión de vida, habiendo sido reconectada nuevamente en febrero del corriente año.

Por último se puso en funcionamiento la Central Nuclear Atucha II (CNA-II), la cual se encuentra localizada en el mismo Complejo que su homónima. Esta última es una central de potencia cuya tecnología fue desarrollada por Siemmens (Alemania) en el año 1984, posee también un reactor PHWR, pero a diferencia de la Central Nuclear Atucha I, utiliza dióxido de uranio natural como combustible y fue conectada a la matriz energética en el año 2014.

Actualmente estas tres centrales son operadas por la empresa Nucleoeléctrica Argentina (NA-SA) y tienen una capacidad de 1790 MWe nucleares (362 MWe de CNA-I, 745 MWe de CNA-II y 683 MWe de CNE) representan el 4,8 % de la potencia instalada en el Sistema Argentino de Interconexión Eléctrica (SADI)[1].

1.2.1 Proyecto de cambio de combustible de Atucha II

NA-SA tiene como proyecto realizar un cambio en el tipo de combustible utilizado en la Central Nuclear Atucha II, modificándolo de uranio natural, con una proporción de 0,0711 % de U^{235}, a uranio levemente enriquecido (ULE) con una composición de U^{235} al 0,85 %, como el realizado en Atucha I a fines de la década de 1990.

Con esto se busca reducir los costos de generación, ya que el ULE entrega mas energía al permanecer más tiempo en el reactor. Además, se reduce el uso de la maquina de recambio.

Para poder llevar esto a cabo el Sistema de Limitación de la planta debe ser mejorado, debido a que los valores de la predicción de márgenes de operación obtenidos con el algoritmo actual del sistema son demasiado conservadores, debido a las perturbaciones locales que se dan por el incremento en la reactividad.

Los valores conservadores otorgados se deben a que el algoritmo actual utilizado es muy simple y poco flexible.

La mejora consiste en modificar el sistema analógico actual por uno digital programable, lo que permitiría desarrollar distintos algoritmos, entre ellos algoritmos de inteligencia artificial, como Redes Neuronales Artificiales (ANN por el inglés *Artificial Neural Network*).

2

Inteligencia Artificial

La Inteligencia Artificial es una rama de las Ciencias de la Computación de naturaleza multidisciplinaria que combina conceptos de lógica, computación y filosofía. Su objetivo es la creación de entidades artificiales por medio de algoritmos basándose en modelos del comportamiento humano, para que éstas sean capaces de la realización de tareas o de la resolución de problemas.

2.1 Antecedentes de aplicación en la industria nuclear

En la actualidad se están utilizando técnicas de inteligencia artificial en distintos campos de la industria, medicina e investigación con fines diversos. Entre las herramientas proporcionadas por esta disciplina, se encuentran las Redes Neuronales Artificiales, las cuales tienen como objetivo emular una salida a partir del procesamiento de entradas determinadas.

Esta tecnología también se ve implementada en distintas problemáticas, en lo que a reactores nucleares se refiere: un claro ejemplo son los antecedentes de la implementación de estas redes para la estimación del desempeño del DNB en núcleo de PWR que datan del año 1993 [2]. En este estudio se concluye que las ANN son una herramienta viable que pueden desarrolarse para ser utilizadas en el sistema de monitoreo *on-line* de los margenes termohidaúlicos en este tipo de centrales.

En los últimos años se hicieron varios estudios de implementación de Redes Neuronales Artificiales en combinación con Algoritmos Genéticos para la Predicción de Valores del Flujo de Calor Crítico (CHF inglés *Critical Heat Flux*) en un amplio rango de países del mundo, siendo más importantes los aportes de Corea del Sur y China. [3]. Particularmente en Corea del Sur se han obtenido muchos logros en cuanto a la predicción de DNBr.

En esta recopilación de trabajos se concluye que son métodos muy útiles para este tipo de predicciones, siendo más precisos que otros sistemas que se utilizan convencionalmente para la predicción de CHF como las tablas de valores empíricas.

3

Objetivos

3.1 Objetivo del Trabajo

El objetivo del presente Trabajo Final Integrador es estudiar la factibilidad de la implementación de Redes Neuronales Artificiales como alternativa en el dispositivo de calculo del Sistema de Limitación de la Central Nuclear Atucha II, tanto en su utilización con combustible de uranio natural, así como con la utilizaciones de combustible de uranio levemente enriquecido.

Parte II

Descripción de la Planta

Central Nuclear Atucha II

Nothing in life is to be feared, it is only to be understood.
Now is the time to understand more, so that we may fear less.

Marie Curie

Cómo se introdujo en el Capítulo 1 un reactor nuclear es una máquina térmica cuya particularidad reside en el combustible de material físil que utiliza para la obtención de la energía térmica. En el caso de la Central Nuclear Atucha II, el origen del diseño del reactor de agua pesada presurizada (PHWR) con dos circuitos o *loops* es alemán. Fue realizado por compañía Siemens-KWU y es único en el mundo.

4.1 Estructura

El núcleo del reactor se encuentra contenido en un recipiente de presión de gran tamaño. El material de base de este recipiente es el 20MnMoNi55 revestido en acero austenítico. Sus dimensiones son aproximadamente de 14,3 metros de alto y tiene un diámetro de 7,4 metros, como se presenta en la Figura §4.1.

Fue diseñado para operar con una presión P = 115 bar y con una Temperatura nominal T_N = 313 ^oC en la rama caliente y a una P = 126 bar y T_N = 278 ^oC en la fría.

El plenúm superior y el plenúm inferior de este recipiente están ocupados en gran medida por cuerpos de relleno de acero, diseñados para disminuir el inventario de agua pesada necesaria, así como resguardo de una excesiva radiación del recipiente.

En su interior se alberga el tanque moderador, el cual es atravesado por 451 canales refrigerantes donde se disponen los combustibles y por donde circula el agua pesada a alta presión del sistema refrigerante o Circuito Primario, que transfiere la potencia térmica extraída al Circuito Secundario por medio de generadores de vapor ubicados fuera del recipiente. En estos generadores se calienta el agua liviana desmineralizada del circuito secundario hasta obtener vapor a alta presión, que es utilizado para producir el movimiento de los álabes de la Turbina Principal. Esta turbina gira unida al

Generador Eléctrico Principal, produciendo con el movimiento una potencia eléctrica de 745 MWe en condiciones nominales [4].

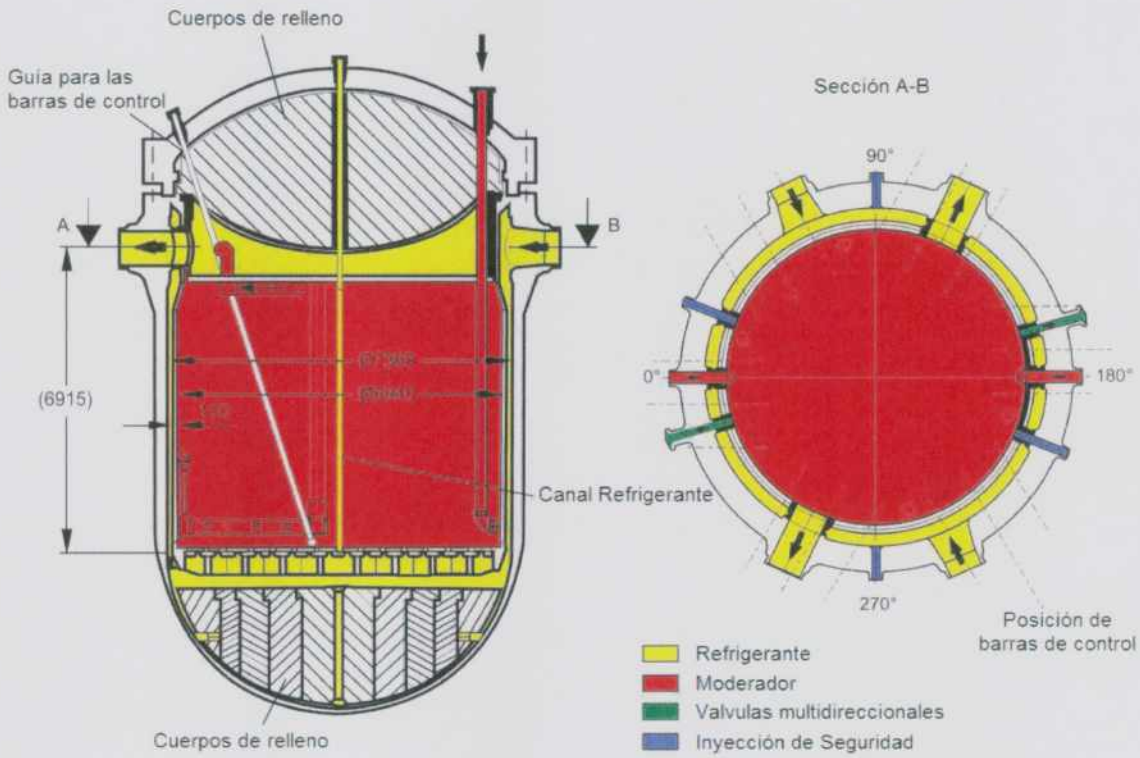

Figura 4.1: Esquema del recipiente de presión de la central Nuclear Atucha II y sus principales sistemas.

Los canales refrigerantes a su vez transfieren un remanente de calor al agua pesada del sistema del moderador contenida en el tanque moderador, por lo cual también se encuentra en circulación durante la operación. Ambos sistemas se encuentran separados físicamente, aunque se encuentran conectados por orificios ubicados en la tapa del tanque moderador para ecualizar la presión de ambos sistemas. De esta manera sólo el refrigerante circula a alta velocidad, y se evita poner en movimiento a esas velocidades a toda la masa de agua del moderador.

4.1.1 Sistema de Refrigeración Principal - Circuito Primario

El Sistema de Refrigeración Principal consiste de dos *loops* idénticos, cada uno formado por un Generador de vapor, una bomba principal y las cañerías que los interconectan, además de un presurizador conectado a uno de los *loops*, como se esquematiza

en la Figura §4.2.

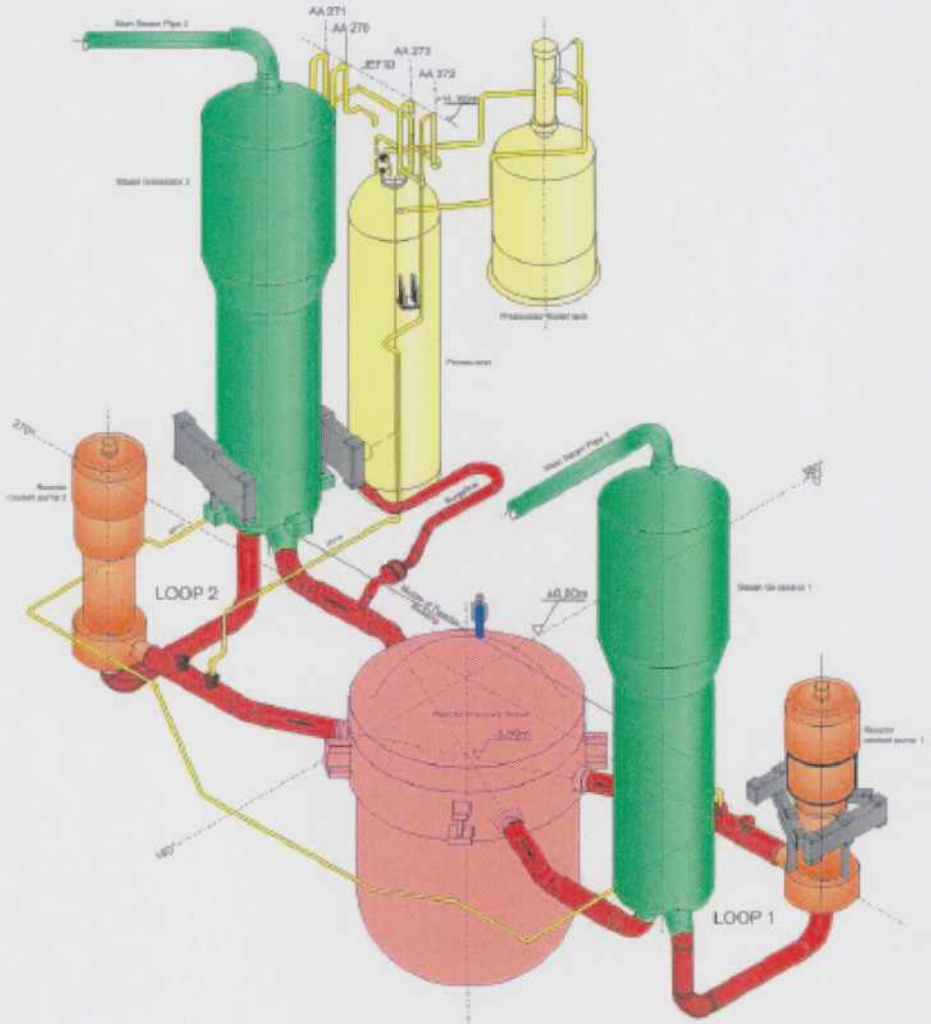

Figura 4.2: Disposición del Sistema de Refrigeración Principal de la Central Nuclear Atucha II

4.1.2 Sistema Moderador

El agua pesada del Sistema Moderador tiene transferido un 10% aproximado de la potencia generada en el núcleo. Como se encuentra a una temperatura promedio de 180^{o}C, menor que la temperatura promedio del sistema refrigerante a 290^{o}C, existe una transferencia de calor desde los canales de debido a la conducción térmica, que se traduce en un 5% aproximado de la potencia generada en el núcleo. El otro 5% le es transferido debido a la interacción de neutrones y la radiación gamma en el agua

pesada del tanque. El Sistema Moderador consiste de cuatro *loops* idénticos que operan en paralelo. Cada *loop* contiene un intercambiador de calor, una bomba y válvulas y cañerías que los interconectan, como se esquematiza en la Figura §4.3.

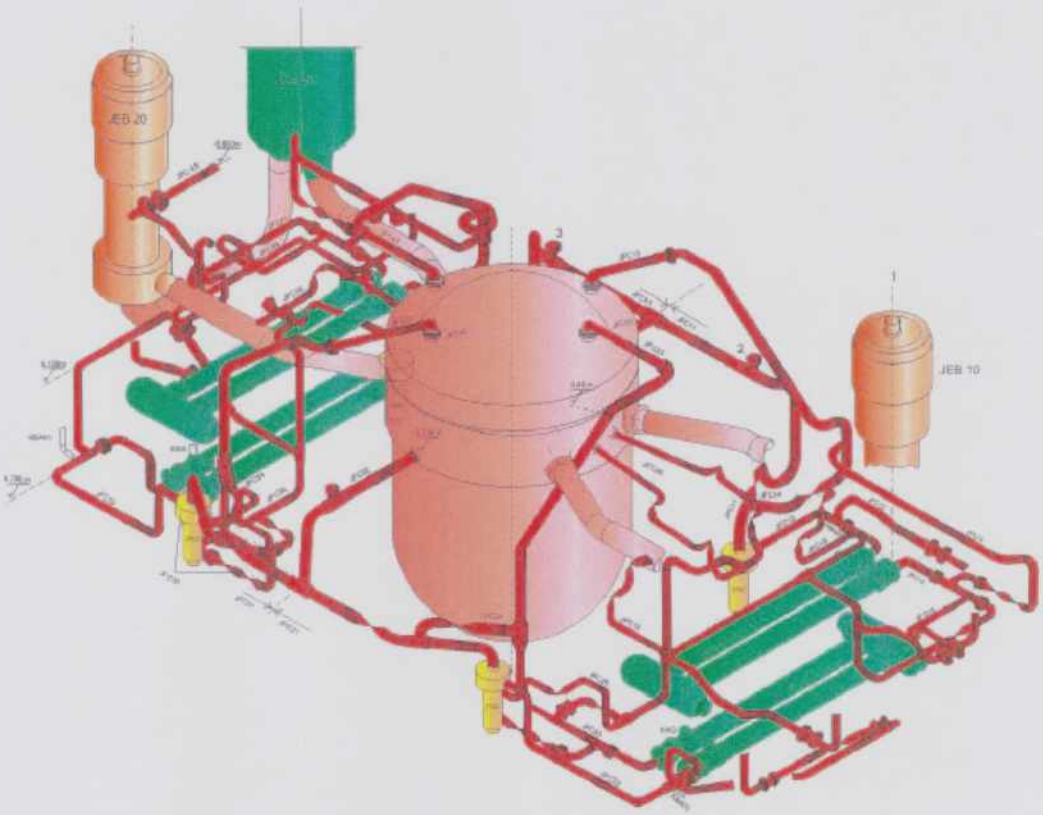

Figura 4.3: Disposición del Sistema Moderador de la Central Nuclear Atucha II. (Cañerias en rojo pertenecen al Sistema Moderador, las color pastel pertenecen al Sistema Refrigerante y sólo se toman como referencia)

Este sistema posee varias funciones dependiendo del modo de operación del reactor. Durante la operación normal, el sistema del moderador mantiene la diferencia de temperaturas descripta anteriormente con el sistema de refrigeración, asimismo está diseñado para operar como sistema de remoción del calor residual en parada y como sistema de enfriamiento en caso de emergencia.

4.1.3 Núcleo

El núcleo del reactor consiste en 451 elementos combustibles de uranio natural dispuestos en sus respectivos canales refrigerantes.

Cada elemento combustible tiene en su interior 37 barras de zircaloy-4, ubicadas formando 3 anillos concéntricos alrededor de una barra central. Estas barras se encuentran sostenidas por trece separadores dispuestos espaciadamente a lo largo del elemento combustible y placas de soporte en los extremos, como se esquematiza en la Figura §4.4. En el interior de cada barra se encuentran pastillas de dióxido de uranio natural sinterizado apiladas hasta completar una longitud efectiva de 5.3 m.

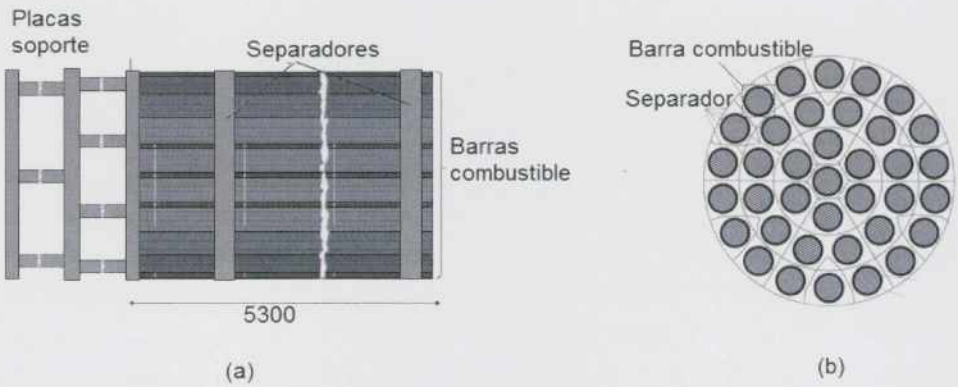

Figura 4.4: Elemento Combustible.(a) Vista longitudinal del elemento combustible (b) corte axial del elemento combustible a la altura de un separador.

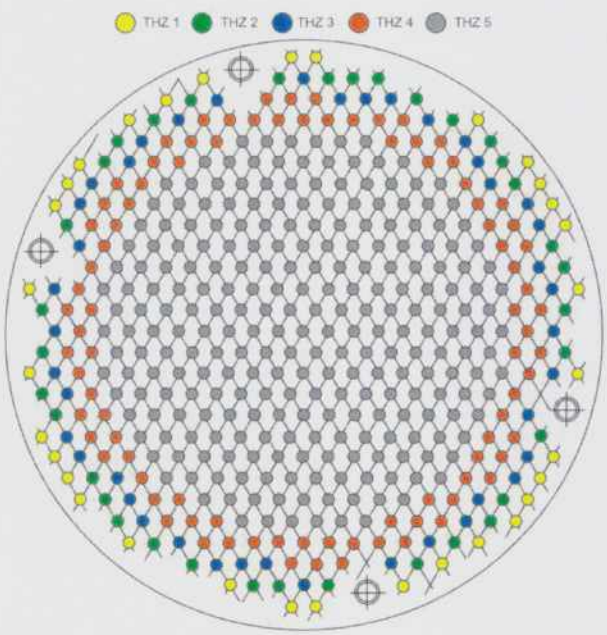

Figura 4.5: Zonas termohidraúlicas de la Central Nuclear Atucha II

No todos los canales producen la misma potencia, a medida que uno se aleja del centro, la potencia generada va disminuyendo. Por ello, los canales de refrigeración se dividen en cinco zonas termohidráulicas dentro del núcleo, las cuales son representadas en la Figura §4.5. Para obtener una temperatura uniforme en las distintas zonas, el caudal se regula en los canales de la zona exterior mediante restrictores de flujo o toberas, para que este sea proporcional con la potencia térmica generada.

El núcleo tiene la capacidad de generar 2160 MW térmicos cuando opera a plena potencia. Aproximadamente el 95% de esta potencia se genera en el combustible, y el otro 5% por termalización

de neutrones en el tanque moderador. De la potencia generada en el combustible, el 90 % es extraída por el sistema de refrigeración, lo remanente es transferido al sistema moderador.

El agua pesada del sistema de refrigeración ingresa por las cañerías laterales en la parte superior del recipiente de presión, desciende a través del huelgo (downcomer) que existe entre el tanque del moderador y el recipiente de presión hasta el plenúm inferior, luego circula a través de los canales de manera ascendente hasta el plenúm superior por donde sale radialmente a través de unas ranuras ubicadas en la pared superior lateral del recipiente.

4.1.4 Máquina de Carga de Combustible

Durante la operación los elementos combustibles (EECC) sufren una reducción en la cantidad de material físil por su consumo o quemado en función del tiempo. Esta reducción en el material físil afecta la distribución neutrónica en el núcleo, pero es compensada con el movimiento de las barras de control hasta cierto límite en el cual se debe extraer el elemento combustible e insertar otro con material fresco.

Cada elemento combustible permanece en el canal un determinado tiempo, relacionado con el nivel de potencia al que es irradiado, hasta que alcanza cierto nivel de quemado predefinido. En general, ese nivel de quemado está relacionado con el rendimiento en la generación de energía. Una vez alcanzado el nivel de quemado deseado, el combustible puede ser trasladado a una posición en la que su eficiencia energética mejore o, si su quemado es muy alto, extraído del núcleo. La estrategia de recambio es definida por la Gestión de Combustibles.

El proceso de recambio de combustibles es realizado por la máquina de carga, que es el componente que realiza la introducción y descarga de los elementos combustibles en los respectivos canales. Esta máquina se desplaza por encima de la tapa del recipiente de presión y se posiciona sobre cada canal. En su interior puede alojar dos EECC y diversas herramientas para extraer el tapón de cierre. En el proceso de recambio se acopla al sistema primario y pasa a formar parte de él, manteniendo el combustible refrigerados todo el tiempo.

En la CNA-II se recambian dos combustibles cada tres días, durante la operación de la central, lo que se conoce como operación *on – line*. Esto genera perturbaciones locales de la población neutrónica debido al cambio abrupto de los materiales, las cuales se verían incrementadas al utilizar ULE, y es uno de los motivos principales para la

realización de este proyecto, ya que como se describirá posteriormente, esto afecta la operación del Sistema de Limitación de la central.

4.1.5 Control de reactividad

En la Central Nuclear Atucha II se emplean distintos métodos para el control de reactividad. Principalmente se utilizan las barras de control del reactor y una dilución de ácido bórico en el circuito primario.

El reactor posee 18 barras de control construidas con material absorbente, las cuales son posicionadas de forma oblicua desde la periferia de la tapa del recipiente de presión. Esta disposición particular de las barras se debe a que sobre la tapa del recipiente de presión se encuentra la máquina de recambio, la cual requiere que esta parte se encuentre despejada para su funcionamiento.

Del total de barras de control, 9 son llamadas "grises-de acero inoxidable y las 9 restantes son llamadas "negras-de Hafnio. Se agrupan en bancos de tres barras del mismo material, los cuales se encuentran a 120º entre sí. Seis de las barras negras se encuentran extraídas al 100% durante la Operación Normal, y tienen la capacidad de absorción de neutrones para el apagado del reactor en caso de ser necesario. El resto de las barras son utilizadas para la regulación de la reacción en cadena y la distribución de potencia del núcleo. Los cambios de reactividad debidos al quemado, a los recambios u otros efectos son compensados por el movimiento de estas barras de regulación para mantener el reactor crítico.

4.1.6 Detectores dentro y fuera del núcleo

In-Core

Para obtener información del flujo neutrónico dentro del núcleo se emplean detectores de neutrones autoenergizados de Vanadio. El núcleo cuenta con 90 detectores de este tipo distribuidos en 15 lanzas. Cada lanza cuenta con 6 detectores ubicados en posiciones axiales fijas para todas las lanzas. En la Figura §4.6 se presentan la disposición de los detectores en cada lanza (a), y las posiciones de las lanzas en el núcleo (b).

Ex-core

Fuera del recipiente de presión se utilizan doce cámaras de ionización, en combinación con los detectores *in-core* para medición y control de potencia del reactor.

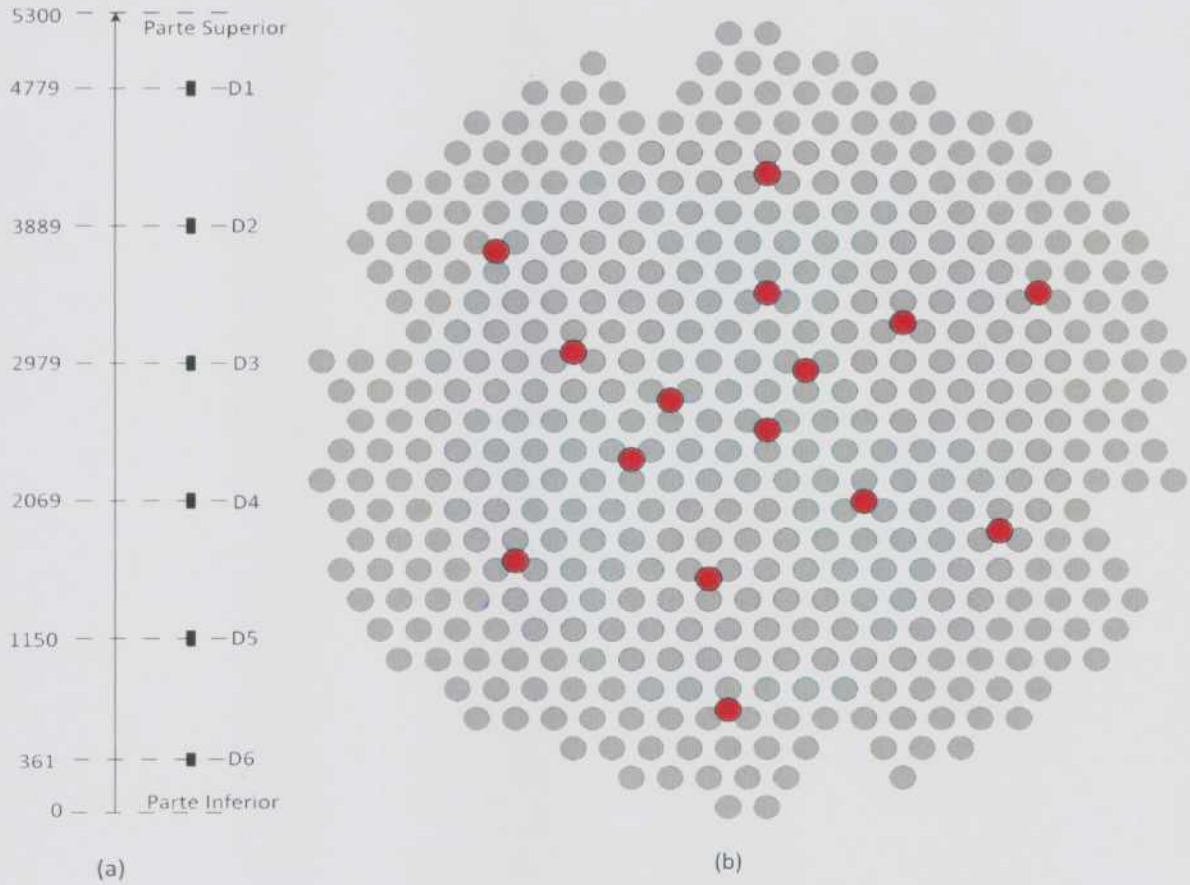

Figura 4.6: Detectores *in – core*: (a)disposición de los detectores en las lanzas, (b) posición de las lanzas (rojo) y de los canales (gris) en el núcleo

4.2 Sistemas de Vigilancia del Reactor

En la planta, existen tres niveles que aseguran el correcto funcionamiento de la planta, los cuales se encuentran organizados según su función y nivel de seguridad.

4.2.1 Sistema de Regulación

El Sistema de Regulación se encarga de controlar el nivel de potencia global y amortiguar las oscilaciones del xenón, compensando las perturbaciones de poca magnitud que se producen mediante movimientos de las barras de control. Se controla la distribución global a partir de las mediciones de los detectores dentro y fuera del núcleo, bajo la suposición que esta medición es representativa de los canales de la zona, sin tener en cuenta la posibilidad de máximos locales.

Para la distribución axial se promedian las señales de los detectores superiores y se

comparan con los inferiores. Para la distribución azimutal en cambio se hace una suma pesada de todas las mediciones de las lanzas que corresponden a un mismo sexto del núcleo.

Para compensación de las distribuciones global y axial se actúa con movimientos de bancos de barras de control. Para la distribución axial se actúa con movimientos de barras independientes.

4.2.2 Sistema de Limitación

El Sistema de Limitación se encuentra por encima del Sistema de Regulación, y toma acciones en la posición de las barras de control en caso que se alcancen estados no permitidos de potencia como superar los límites de DNB, PCI o LOCA. Su finalidad es mitigar las causas y mantener al reactor en condiciones normales de operación. Las acciones de mitigación con las barras de control involucran una serie de pasos escalonados, en principio el bloqueo de la extracción de barras, luego la inserción de barras a baja velocidad y posteriormente la inserción de barras a alta velocidad.

4.2.3 Sistema de Protección

El Sistema de Protección se encuentra por encima de los otros niveles de seguridad y ante las desviaciones que no puedan ser mitigadas por estos se encarga de llevar a la planta a un estado seguro conservando la integridad de los combustibles. Su actuación es mediante la caída total de las barras para un apagado rápido (SCRAM) o el segundo sistema de parada de inyección de ácido bórico en el sistema del moderador.

Sistema de Limitación (RELEB)

El Sistema de Limitación de la Central Nuclear Atucha II, RELEB de el alemán *REaktor LEistungs Begrenzung* tiene como objetivo que los EECC operen con un margen de seguridad adecuado respecto del límite por apartamiento de la ebullición nucleada (DNB), límite por interacción pastilla-vaina (PCI) y límite por perdida de refrigerante. Por ello, actúa limitando la potencia lineal máxima que presentan los elementos combustibles dentro del núcleo.

5.1 Diseño Termohidraúlico

Las vainas del combustible son la primera barrera contra la liberación de productos de fisión que integra el primer nivel de seguridad dentro del concepto de defensa en profundidad. El objetivo del diseño termohidraúlico es la preservación de la integridad de las vainas durante condiciones normales y en caso de transitorios operacionales anticipados (AOO), siendo el más limitante la parada de bombas del Sistema Primario. Existen limitaciones térmicas del combustible, de la vaina y del refrigerante: se debe asegurar la integridad de las vainas en caso de producirse fusión de pastilla, PCI y DNB.

5.1.1 Limitaciones térmicas del combustible: Fusión de pastilla

Una de las limitaciones principales consiste en prevenir que la temperatura del combustible exceda en cualquier punto del núcleo su temperatura de fusión. La temperatura máxima alcanzada se encuentra en el centro de la pastilla y depende principalmente de la conductividad térmica del combustible y de la densidad de potencia lineal del núcleo. La fusión del centro de pastilla de combustible no ocurre si la potencia lineal tiene un valor menor que $q' = 680$ W/cm. Se toma como base de diseño un valor de potencia lineal de hasta $q' = 600$ W/cm.

5.1.2 Limitaciones térmicas de la vaina: Interacción Pastilla-Vaina

Ante la presencia de variaciones de potencia lineal se manifiesta el fenómeno de la interacción pastilla-vaina o *pellet clading interaction* (PCI). Por esta interacción se produce una deformación localizada de la vaina debida al contacto preferencial de la pastilla en sus bordes superior e inferior, como se esquematiza en la Figura §5.1. Esto a su vez puede llevar a deformaciones importantes que degeneren en roturas de la vaina.

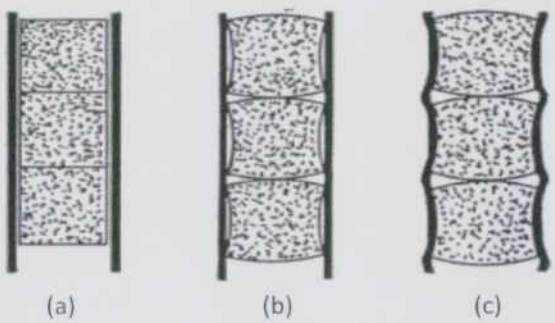

Figura 5.1: Combustible: (a) Fábrica; (b) Antes de PCI; (c) Después de PCI

El PCI es un fenómeno que se manifiesta ante la presencia de variaciones rápidas de la potencia lineal. Esto quiere decir, que el fenómeno de PCI no depende de un límite fijo de potencia lineal, sino que sus límites son dinámicos o deslizantes. El sistema de limitación ajusta el valor de los límites por PCI en función de la evolución de las mediciones de los detectores $in-core$, pero limita las máximas variaciones con pendientes límites establecidas.

El límite por PCI es un valor de potencia lineal que se ajusta entre una serie de bandas de potencias lineales que siguen a la potencia medida en los detectores $in-core$ con pendientes acotadas, evitando así los cambios bruscos de potencia.

5.1.3 Limitaciones del refrigerante: Apartamiento de la Ebullición Nucleada

Cuando el agua pesada del refrigerante comienza a circular por el canal de refrigeración atravesando las barras del EECC, el mismo transmite el calor por convección a la fase líquida, ya que la temperatura de la pared se encuentra por debajo del valor de saturación. Conforme aumenta la temperatura, la pared alcanza la temperatura de saturación, comenzando la ebullición nucleada del fluido (*Onset of Nucleate Boiling* ONB). A medida que sigue aumentando, se llega a un estado donde se produce el apartamiento de la ebullición nucleada (*Departure from Nucleate Boiling* DNB), con lo cual se produce una capa o film de vapor en la pared, predominando la transmisión de calor por radiación, impidiendo una adecuada refrigeración de la zona calentada. El incremento de la temperatura de la superficie del EECC a partir de este punto es tal que se puede ocasionar la fusión del centro de la pastilla.

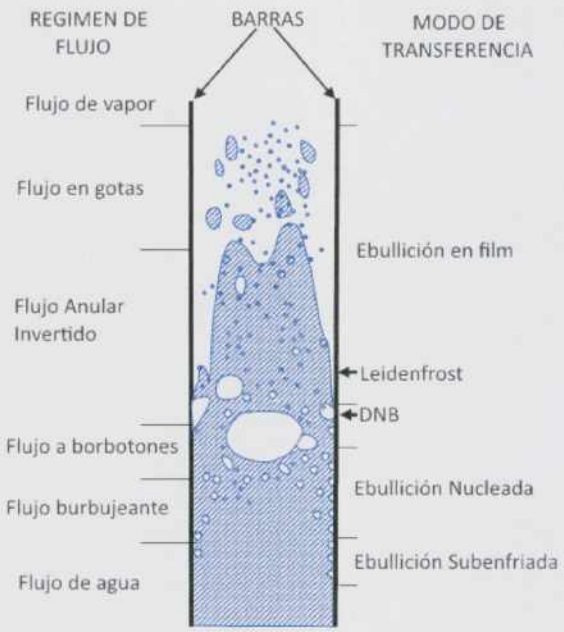

Figura 5.2: Patrones de flujo del refrigerante en un canal caliente con alto flujo de calor y regiones de transferencia de calor.

El límite de la potencia lineal por DNB debe asegurar que no ocurra daño en los EECC por DNB durante transitorios que se espera ocurran durante la vida útil de la central (como la parada de bombas principales), aun considerando las incertezas de los parámetros termohidráulicos y de fabricación de los elementos combustibles. Este sistema limita la potencia lineal máxima que miden los detectores *in–core* contra una serie de constantes de potencia lineal, las cuales son corregidas por la deformación del perfil de potencia del reactor y las condiciones operativas como las revoluciones de las bombas, la presión y la temperatura de entrada al reactor.

Se asegura la no ocurrencia de daño a elementos combustibles por DNB en operación normal y en AOO mediante límites de potencia de canal para la gestión de combustible. Además, se requiere una vigilancia continua de las potencias lineales de los elementos combustibles de modo de asegurar que se encuentran a un margen seguro del fenómeno de DNB. Esto se lleva a cabo por medio del Sistema de Limitación que establece límites a la potencia lineal de los elementos combustibles.

5.2 Descripción del Dispositivo Actual

En la CNA-II existe un dispositivo de cálculo (LVÜ: *Leistungs Verteilungs Überwachung*) dentro del sistema de limitación. Este dispositivo toma las señales de las 15 lanzas ubicadas en el interior del reactor[5].

Estas lanzas se agrupan en 3 circuitos de vigilancia, de una manera tal que se disminuya el número de operaciones del sistema de limitación. Algunas de las lanzas son exclusivas de un circuito, mientras que otras son compartidas por dos circuitos. Los circuitos de vigilancia que agrupan lanzas de las zonas internas, medias y externas del núcleo se presentan en la Figura §5.3. En cada lanza, los detectores miden corriente eléctrica proporcional al flujo neutrónico en el núcleo.

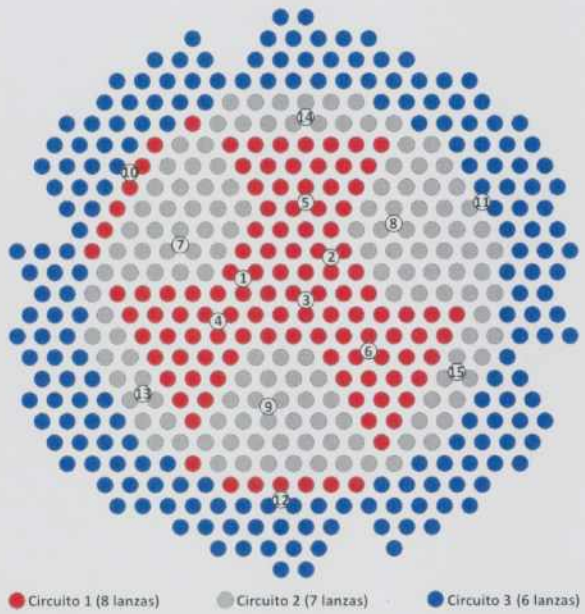

Figura 5.3: Circuitos de Vigilancia del RELEB

Esta corriente medida se encuentra en un rango entre 4 y 20 mA y se transforma a un rango entre 0 250% de plena potencia, donde el 100% indica el flujo medio del núcleo a plena potencia. La potencia térmica se transforma en potencia lineal mediante factores de transmisión. Luego se multiplica por los factores de sobrecalibración k_u, los cuales están calculados para cada una de las lanzas en función del circuito en que se encuentre. Esta potencia lineal ficticia (ya que los detectores no contienen material fisil) se corresponde con la correlación del flujo neutrónico a la potencia lineal máxima de los elementos combustibles en la zona de vigilancia del detector. Comprende valores en un rango entre 0 y 600 W/cm.

Las potencias lineales obtenidas de las tres posiciones superiores de cada lanza son comparadas con valores límites generados por el Sistema de Limitación denominados Potencias Lineales Admisibles q_{zul}.

Determinación de q_{zul}

Para determinar estas potencias admisibles se generan en cada circuito de vigilancia una "lanza virtual"que contiene las potencias lineales máximas medidas para cada una de las seis posiciones en cada circuito. Con estas mediciones y mediante las ecuaciones 5.1, 5.2 y 5.3 se calcula el valor de q_{zul} para las tres posiciones superiores de cada circuito.

$$q'_{zul,1} = CKM + A_1 + C_1 \cdot (q'_{max,2} - q'_{max,5}) \tag{5.1}$$

$$q'_{zul,2} = CKM + A_2 + C_2 \cdot (q'_{max,2} - q'_{max,5}) \tag{5.2}$$

$$q'_{zul,3} = CKM + A_1 \tag{5.3}$$

Donde:

A_1, A_2, A_3 son constantes de ajuste (W/cm) para los detectores 1, 2 y 3;

C_1, C_2 son constantes de ajuste para la deformación del perfil de potencia;
max $q'_{max,2}$ Máxima potencia lineal medida en los detectores 2 de todas las lanzas del circuito;
max $q'_{max,5}$ Máxima potencia lineal medida en los detectores 5 de todas las lanzas del circuito;
$CKM = C_t(T_c - T_0) + C_P(P - P_0) + C_n(\Sigma n - \Sigma_0)$ es un factor de corrección que tiene en cuenta los apartamientos del punto nominal de operación;
aquí T_e es la temperatura de entrada al núcleo en ^oC
T_0 es la temperatura de entrada nominal (278^oC),
P es la presión a la salida del núcleo en bar
P_0 es la presión a la salida nominal (115 bar)
Σn son las revoluciones de las Bombas Principales en %
y Σn_0 son las revoluciones de las Bombas Principales nominal 100 %;
C_T, C_P, C_n son constantes de ajuste para la temperatura, presión y revoluciones.

5.2.1 Obtención de la distancia mínima al DNB

El Margen Relativo de Potencia Extrapolada *LEX* (del alemán Leistungs-Extrapolation) es la relación entre la diferencia entre la potencia lineal admisible y la potencia lineal actual medida por el detector con respecto a la potencia lineal medida por el detector. El valor *LEX* es una indicación del margen relativo de potencia global del reactor disponible extrapolado sobre las condiciones actuales del reactor.

El margen de potencia extrapolado en el cual se utiliza la potencia lineal admisible determinada por el algoritmo del sistema de limitación se denomina LEX_{RELEB}. Se obtienen nueve valores de LEX_{RELEB} para cada una de las tres posiciones superiores de detectores dentro de las lanzas virtuales de los tres circuitos mediante la ecuación 5.4.

$$(LEX_{RELEB})_i = \frac{(q'_{zul,i} - q'_i)}{q'_i - C_i(q'_{max,2} - q'_{max,5})} \cdot 100 \tag{5.4}$$

Donde $q'_{zul,i}$ son los valores otorgados por el Sistema de Limitación en la posición i; q'_i son los valores de potencia lineal de la medición del detector en la posición i, $q'_i(1 + C_i(q'_{max,2} - q'_{max,5})$ son los valores de potencia lineal que mediría el detector en el caso de encontrarse en el limite del apartamiento del punto de ebullición. y LEX_{RELEB} es el margen de potencia extrapolado [%].

Una vez obtenidos los nueve valores de LEX_{RELEB} para cada una de las tres posiciones superiores de la lanza virtual de los tres circuitos de vigilancia, se toma el minimo valor como la distancia mínima al punto de apartamiento de la ebullición nucleada.

Computadora de procesos de la planta (READAT)

La sala de control de la central dispone de una computadora (READAT) que procesa en forma periódica una cierta cantidad de variables de estado de la planta. Esta computadora se encarga de brindar información adicional sobre el comportamiento de la planta al personal de operación y apoyo, a partir de un conjunto de variables de estado tales como:

- Flujo neutrónico de los 90 detectores in-core.
- Flujo neutrónico de las cámaras ex-core.
- Posición de las barras de control.
- Velocidad de las bombas principales.
- Temperaturas y presiones del circuito primario.
- Temperaturas y presiones del circuito moderador.
- Estado de la máquina de recambio de elementos combustibles.
- Datos del circuito secundario o de vapor.

En la computadora se ejecutan diversos programas que procesan los distintos datos de entradas y con sus salidas permiten verificar el estado del sistema de regulación, limitación y protección.

Estos programas se encargan de distintos aspectos funcionales del reactor:

- **AXCNA2** Cálculo del margen al DNB en cada elemento combustible.
- **PODESY** Cálculo de la distribución de potencia instantánea del núcleo a partir de las lecturas de los detectores in-core.
- **BURNUP** Cálculo del quemado en los Elementos Combustibles.

- **REALEI** Cálculo de la potencia térmica instantánea del reactor.
- **XENON** Cálculo de la concentración instantánea de xenón en el reactor.
- **LVDKAL** Cálculo de la pérdida de sensibilidad de los detectores in-core debido al quemado.

Se describirán a grandes rasgos los programas que fueron empleados en alguna instancia en el desarrollo del PFI.

6.1 PODESY

La función del programa es el cálculo de la distribución de potencia instantánea del núcleo a partir de la lectura de los detectores $in-core$, la posición de las barras de control, la distribución de quemados instantáneos proporcionado por el programa BURNUP y el nivel de potencia del reactor proporcionado por REALEI.

Se fundamenta en el hecho de que esta distribución puede aproximarse por una combinación lineal de las autofunciones de la ecuación de difusión en estado estacionario.

6.2 AXCNA2

La función de este programa es el cálculo de los márgenes del apartamiento de la ebullición nucleada en todos los elementos combustibles de la central, a partir de los datos de la distribución de potencia calculada por el programa PODESY y otros parámetros de la planta.[6]

Utiliza un modelo de canales paralelos en una dimensión para simular la termohidráulica en el núcleo del reactor.

En este modelo cada canal refrigerante se divide en una cantidad n de elementos finitos en donde se realiza el balance térmico entre la cantidad de calor transferida desde las barras combustible y la cantidad de calor transferida al moderador. La perdida de carga se calcula en cada uno de esos elementos finitos y tiene en cuenta la variación de la misma en presencia de doble fase de agua a través del factor de multiplicación.

El cálculo de CHF se realiza utilizando las tablas de Groeneveld "1995 CHF Look-Up Table"[7], donde se representan valores de CHF en agua liviana (H_2O) para un

tubo de 8 mm de diámetro a partir de valores discretos de presión (P), flujo másico (G) y título termodinámico (X). Internamente la función corrige el valor de CHF del tubo de 8mm al Elemento Combustible de la CNA-II a través factores de corrección. A su vez se emplean factores de escala fluido-fluido de Ahmad [8] para obtener los valores en agua pesada.

Los resultados del programa fueron contrastados contra casos de prueba utilizando el código COBRA3-CP que es de uso internacional en el cálculo de CHF, a fin de validar el mismo.

Parte III

Inteligencia Artificial

7 Redes Neuronales Artificiales

Como se mencionó previamente en la Introducción, dentro de la Inteligencia Artificial existen distintas herramientas para abordar un problema. En la parte de Sistemas inteligentes existe un campo definido como Redes Neuronales Artificiales. Estas Redes Neuronales Artificiales o *Artificial Neural Networks* (*ANN* por sus siglas en inglés) están inspiradas en el funcionamiento biológico de las neuronas del sistema nervioso de los animales.

A grandes rasgos las ANN son un sistema de neuronas (perceptrones) interconectadas, dispuestas en una o varias capas, que dada una entrada interactúan para producir una salida. El perceptrón multicapa es utilizado como interpolador universal. Existen una infinidad de aplicaciones de estas redes desde reconocimiento de caracteres hasta predicciones meteorológicas.

7.1 Estructura

Las ANN poseen una estructura tal que es posible obtener sistemas sencillos como el perceptrón simple o más complejos como el perceptrón multicapa según vaya aumentando la dificultad del problema que se quiera abordar. En todas ellas las unidades de proceso funcionan de la misma forma.

7.1.1 Neuronas

Las unidades básicas de proceso de una ANN se denominan neuronas. Su creador Frank Rosenblatt se inspiró en el modelo de McCulloch y Pitts y una regla de aprendizaje para desarrollar este modelo que denominó con el nombre de perceptrón. En estas unidades de proceso se reciben las señales de entrada desde otras neuronas o fuentes externas y se procesa la salida.

La estructura más sencilla que puede tener una red neuronal es el perceptrón simple, que esta compuesto de una sola neurona, donde las señales de entrada son procesadas para dar una señal de salida. [9]

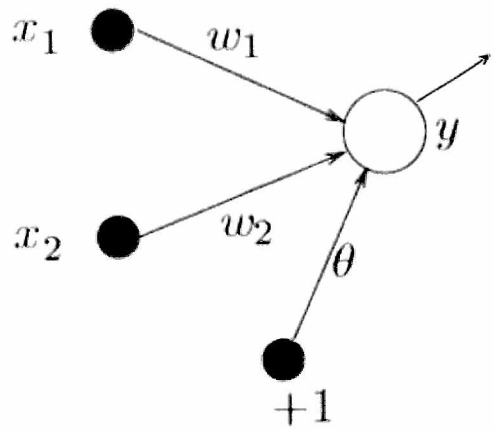

Figura 7.1: Esquema del perceptrón simple.

En el esquema de la Figura §7.1 se busca ilustrar cómo es el funcionamiento de la misma. Se basa en la operación matemática descripta en la ecuación 7.1. Donde las distintas entradas x_i (dos para el caso del ejemplo) son valoradas con un peso sináptico w_i, y la sumatoria del producto escalar de cada caso más un valor fijo θ son procesadas en la neurona a través de una función de activación f_a para obtener la salida de la ANN y.

$$y = f(\sum_{i=1}^{i} W_i \cdot x_i + \theta) \tag{7.1}$$

El perceptrón simple puede ser utilizado como clasificador lineal, al implementar una condición que separa por un hiperplano dos regiones en el espacio, para distinguir dos clases de patrones diferentes.

Con lo cual sólo sirve para clasificar problemas linealmente separables.

En el caso de utilizarlo como interpolador, éste es eficaz en sistemas lineales. Si el sistema se aleja de linealidad, es necesario incrementar la complejidad de la red y el número de neuronas para el procesamiento de las entradas.

Cuando se tiene más de una neurona, las mismas pueden agruparse de forma variada, conformando lo que se conoce como Perceptrón Multicapa.

En este tipo de estructura las neuronas se agrupan en capas, en donde:

- La capa de entrada contiene a las neuronas que reciben señales desde el exterior.
- La capa de salida contiene a las neuronas que emiten señales hacia el exterior.
- Las capas intermedias se denominan ocultas.

7.1.2 Pesos sinápticos

Cada neurona existente en una capa se encuentra conectada con todas las neuronas pertenecientes a la capa predecesora y posterior.

Análogamente al esquema de perceptrón simple, cada conexión (o camino) entre una neurona de la capa j y una de la capa k tiene un peso sináptico asociado w_{jk}.

Cada una de las neuronas de la capa k tiene como entrada las señales de salida de las neuronas de la capa j multiplicadas por su peso correspondiente w_{jk}. Cuando los pesos sinápticos tienen valor positivo $w_{jk} > 0$, las contribuciones se denominan excitaciones, por el contrario, cuando los pesos sinápticos tienen un valor negativo $w_{jk} < 0$, estas contribuciones son inhibiciones.

7.1.3 Desfasaje

Como se ha mencionado anteriormente en cada neurona se procesan las señales con una regla simple de propagación, realizando una sumatoria de cada entrada multiplicadas por su peso. A este procedimiento se le suele agregar un valor θ de desfasaje, o en inglés *offset* o *bias*, extendiendo la ecuación 7.1 al perceptrón multicapa mediante la ecuación 7.2.

$$s_k(t) = \sum_j w_{jk}(t) y_j(t) + \theta_k(t) \tag{7.2}$$

7.1.4 Función activación

Estas sumas son activadas por funciones de activación, que pueden ser la función escalón 7.3, la función rampa 7.4 , o la función sigmoide 7.5. Las cuales se presentan en la Figura §7.2.

$$F(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases} \tag{7.3}$$

$$F(x) = \begin{cases} 0 & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases} \tag{7.4}$$

$$F(s_k) = \frac{1}{1 + e^{-s_k}} \tag{7.5}$$

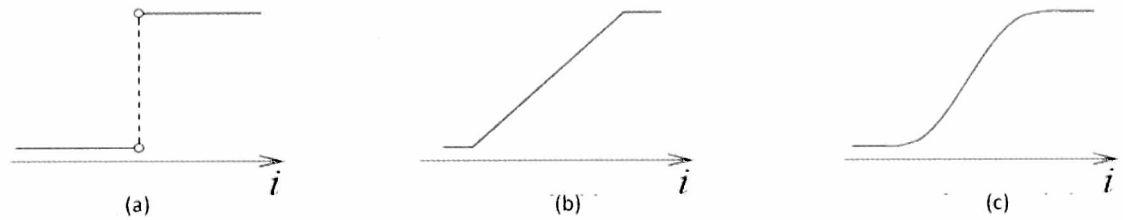

Figura 7.2: Funciones de Activación:(a)Función escalón; (b)Función Rampa; (c)Función sigmoide.

La función activación se utiliza en vez de las sumas ponderadas, ya que esta última es lineal con respecto a sus entradas, no permitiendo modelar correctamente funciones curvas o no triviales.

Para el objetivo del trabajo se utilizará la Función Sigmoide 7.5, la cual tiene la propiedad de ser continuamente diferenciable.

7.1.5 Salidas

Tal como es presentado en la ecuación 7.6, luego de que la sumatoria de las entradas multiplicadas sus respectivos pesos sinápticos más el θ $(s_k(t))$ se active mediante la función activación F_k, se obtiene la señal de salida $y_k(t)$.

$$y_k(t) = F_k(s_k(t)) \tag{7.6}$$

Una vez obtenida la salida se da por terminado el proceso dentro de la neurona.

7.2 Interconexiones

Según su topología, la red puede tener o no influencia de la salida dada. Es decir, puede tener conexiones hacia adelante (*Feed Forward Conections*), sin ninguna conexión entre la entrada y la salida. O puede ser recurrente, existiendo conexiones entre la salida y la entrada de la misma.

Para el tema de estudio se utilizaran las redes neuronales con conexiones hacia adelante o *Feed' Forward Conections*.

En este caso para definir el número de capas se tienen en cuenta las capas ocultas y la capa de salida, sin tener en cuenta la capa de entrada.

Las redes neuronales pueden ser evaluadas en cuanto a su capacidad de representación, ya que incluso con una serie de pesos óptimos, siempre existirá una diferencia con los valores deseados para la salida. También se evalúa el algoritmo de entrenamiento, con el cual se hayan los pesos óptimos para la red.

7.3 Entrenamiento

Se llama entrenamiento al proceso por el cual los pesos de la ANN se ajustan a una serie de patrones o casos que deseamos que la ANN represente. Normalmente, una ANN nueva es inicializada con pesos aleatorios, por lo cual no representa ningún caso en particular.

Cuando se conoce externamente el valor deseado que se quiere obtener y en función de ello se corrige la Red Neuronal Artificial, se denomina aprendizaje *supervisado*. Existen otras clases de ANN en los cuales el aprendizaje es *no supervisado* donde las unidades de salida son entrenadas a responder a ciertos patrones de entrada. Esto se aplica a otra clase de problemas que no se abordarán en el transcurso de este proyecto.

Existen distintas técnicas de entrenamiento de Redes Neuronales Artificiales, en este proyecto se trabajó con dos modelos de entrenamiento:

- Error Backpropagation Algorithm o Algoritmo propagación hacia atrás del error
- Simulated Annealing o Recocido Simulado

7.3.1 Error Backpropagation Algorithm

Para las redes multicapas, se aplica la regla delta generalizada, en la cual se emplea la función Error $e = (d_o^P - y_o^P)$, donde e es la diferencia actual entre el valor obtenido por la red y_o^P y el valor deseado de la misma d_o^P, siendo P el patrón introducido. Asimismo la función e es la derivada de la función E presentada en la ecuación 7.7, la cual siempre toma valores positivos o cero.

$$E = \sum_P E^P = \frac{1}{2} \sum_P (d_o^P - y_o^P)^2. \tag{7.7}$$

Se busca minimizar el error variando los pesos sinápticos. Para ello se utiliza el método de descenso por el gradiente, para lo cual se calcula la derivada parcial del error

cuadrático con respecto a los pesos sinápticos. El cambio en los pesos se realiza en forma proporcional al opuesto de la derivada para cada patrón P, según la ecuación 7.8.

$$\Delta_p w_{jk} = -\gamma \frac{\partial E^P}{\partial w_{jk}} \tag{7.8}$$

Donde γ es la tasa de aprendizaje, que es una constante. A su vez podemos calcular $\delta = -\frac{\partial E^P}{\partial s_k^P}$ aplicando la regla de la cadena a la ecuación 7.8.

$$\frac{\partial E^P}{\partial w_{jk}} = \frac{\partial E^P}{\partial s_k^P} \frac{\partial s_k^P}{\partial w_{jk}} = -\delta_k^P y_j^P \tag{7.9}$$

Con lo cual la ecuación 7.8 se convierte en la ecuación 7.10.

$$\Delta_p w_{jk} = \gamma \delta_k^P y_j^P \tag{7.10}$$

Para las capa de salida (k = o) el valor de δ esta dado por la ecuación 7.11 mientras que para las capas ocultas esta dado por la expresión recursiva 7.12.

$$\delta_o = -\frac{\partial E^P}{\partial y_o^P} \frac{\partial y_o^P}{\partial s_o^P} = (d_o^P - y_o^P) F'(s_o^P) \tag{7.11}$$

$$\delta_h = -\frac{\partial E^P}{\partial y_h^P} \frac{\partial y_h^P}{\partial s_h^P} = \sum_{k=1}^{N_k} \frac{\partial E^P}{\partial s_k^P} \frac{\partial s_k^P}{\partial y_h^P} \frac{\partial y_h^P}{\partial s_h^P} = \sum_{k=1}^{N_k} \delta_k^P w_{hk} F'(s_h^P) \tag{7.12}$$

Para el caso de estudio donde $F(s_k)$ viene dada por la ecuación sigmoide 7.5, $F'(s_k)$ esta dada por la ecuación 7.13.

$$F'(s_k) = F(s_k)(1 - F(s_k)) \tag{7.13}$$

De manera análoga se procede con los valores de θ, en donde $\Delta_p \theta_k$ esta dada por la ecuación (7.14).

$$\Delta_p \theta_k = -\gamma \frac{\partial E^P}{\partial \theta_k} = \gamma \delta_k^P \tag{7.14}$$

En este procedimiento el cambio es proporcional $\frac{\partial E^P}{\partial w}$. La tasa de aprendizaje γ se elige de tal forma que sea tan grande como sea posible para lograr una rápida convergencia pero que no deje lugar a oscilaciones en el proceso.

Una alternativa para utilizar una tasa de aprendizaje grande y evitar las oscilaciones es agregar en la expresión de Δw un término de momento que tenga en cuenta el efecto del Δw previo, entonces la expresión 7.10 quedaría 7.15

$$\Delta w_{jk}(t+1) = \gamma \delta_k^P y_j^P + \alpha \Delta w_{jk}(t) \tag{7.15}$$

Donde t representa el número de época, y α es una constante que determina el efecto del cambio previo.

Con este término de momento se evitan las oscilaciones y se logra la convergencia en menos iteraciones.

7.3.2 Simulated Annealing

Este tipo de algoritmo se encuentra inspirado en el proceso de recocido de un material metalúrgico o cerámico. En este proceso se calienta un material y se enfría lentamente en condiciones controladas con el fin aumentar el tamaño de los cristales en el material y reducir los defectos en la matriz del mismo. Esto tiene como efecto mejorar la resistencia y durabilidad del material.

El principio físico detrás de este proceso es que, al ser calentado el material se aumenta la energía de los átomos permitiéndoles moverse libremente, y el lento programa de enfriamiento permite una nueva configuración de bajas energías internas.

El proceso de enfriamiento se modeló simulando los cambios de energía de un sistema de partículas a medida que se disminuye la temperatura, hasta su convergencia en un estado estable o congelado.

Según la termodinámica, la ecuación de Boltzmann expresa que a una temperatura T, la probabilidad de un incremento energético ΔE se puede apróximar por la ecuación 7.16.

$$p[\Delta E] = e^{-\frac{\Delta E}{kT}} \tag{7.16}$$

Siendo k la constante Boltzmann.

En el algoritmo Metrópolis [10], se genera una perturbación en el sistema de naturaleza aleatoria y se calcula ΔE. Si este valor es negativo, el cambio se acepta automáticamente; por el contrario, si el valor es positivo, el cambio será aceptado con una probabilidad indicada por la anterior expresión.

Lo que busca el Recocido Simulado mediante esta estrategia es permitir al algoritmo el corrimiento a soluciones peores que la anterior, para evitar que la búsqueda finalice en un mínimo local. Conforme se avanza con la búsqueda por medio de la función de probabilidad, se irá disminuyendo la probabilidad de escape, diversificando al principio e intensificando al final.

En el proyecto se utilizará el método de Recocido Simulado o sus siglas SA del inglés *Simulated Annealing* como una alternativa al método de EBPA para entrenar las ANN.

Parte IV

Implementación de ANN

en los casos de estudio

8

Primera Fase : Evaluación de Algoritmos

Si no puede volar entonces corre, si no puedes correr entonces camina, si no puedes caminar entonces arrástrate, pero sea lo que sea que hagas, sigue moviéndote hacia adelante.

Martin Luther King

Para el desarrollo de los algoritmos se utilizó el entorno de desarrollo integrado NetBeans, y se emplearon los lenguajes Octave y C++.

Para estudiar el comportamiento del algoritmo y de las Redes Neuronales Artificiales desarrolladas se realizaron en principio ensayos de problemas sencillos donde se aplicaron las metodologías de Retropropagacion de Error (EBPA por el inglés *Error Back Propagation Algorithm*) y de Recocido Simulado (SA por el inglés *Simulated Annealing*) descriptas en el Capítulo 7 para su entrenamiento.

8.1 Retropropagación del Error

Se desarrolló el algoritmo que construye la estructura de la Red Neuronal Artificial definida por el usuario. Este algoritmo toma los números enteros otorgados por el usuario como argumentos y define el número de capas y la cantidad de neuronas en cada una de las capas. A partir de estos datos este algoritmo construye tanto los vectores entradas, sumatoria, salidas de cada capa, así como las matrices que contienen a los pesos sinápticos de las distintas capas con el tamaño respectivo. Luego asigna un valor aleatorio entre 0 y 1 a los pesos w_{ij} de estas matrices. También crea en cada capa una variable θ con un valor aleatorio entre 0 y 1 para cada neurona en las capas, exceptuando las capas de entrada.

Se crearon las siguientes funciones principales para realizar las tareas especificas:

- Evalua_Red: permite la obtención de una salida a partir del procesamiento de los valores de entrada por la ANN como fue explicado en el Capítulo 7.

- Corrección_BackPropagation: permite la obtención de nuevos pesos sinápticos, a partir del entrenamiento con el método EBPA descripto en el Capítulo 7 en la sección Entrenamiento.

Para el caso de entrenamiento de las Redes Neuronales Artificiales con el uso del algoritmo EBPA se modeló una superficie dada por la función vectorial de la ecuación 8.1.

$$z(x,y) = \frac{(sen(2\pi x)cos(2\pi y)+1)}{2} \tag{8.1}$$

Donde las entradas de la ANN toman los valores de las variables x e y, del tipo *double* entre 0 y 1; y el valor deseado d_O, también del tipo *double*, toma el valor de la función $z(x,y)$ evaluada en x e y.

8.1.1 Perceptrón Simple: Estudio del algoritmo con un solo patrón

Como primer paso se evaluaron los algoritmos de creación de la estructura de la ANN, su función Evalua_Red, y su entrenamiento mediante la función Corrección_BackPropagation para un único patrón.

Se tomó un valor de x entre 0 y 1, y un valor de y entre 0 y 1, como entradas, y d_o es el valor de salida deseada $z(x,y)$ obtenida de la ecuación 8.1 evaluada en x e y.

En principio se utilizó la estructura de un perceptrón simple: una capa de entrada con dos neuronas y otra capa de salida con una sola neurona. En cada caso de evaluación se iniciaron los valores de los pesos sinápticos con valores aleatorios entre 0 y 1.

En cada iteración (o época), se evaluó las entradas con la función Evalua_Red y a partir de la salida obtenida se corrigieron los pesos a partir de la función Correccion_Backpropagation. Una vez corregidos los pesos sinápticos, se evalúa nuevamente las entradas con ellos. A la salida (Out) obtenida, se le restó el valor deseado, como lo expresa la ecuación 8.2.

$$e(t) = |Out(t) - d_o| \tag{8.2}$$

El valor absoluto de esta diferencia e fue almacenado en cada iteración identificándolo como un elemento del vector V_e, con longitud igual al número de épocas.

Se realizaron una serie de entrenamientos del mismo patrón antes mencionado, modificando para cada caso el valor del Coeficiente de Aprendizaje γ.

Posteriormente se realizó otra serie de entrenamientos con un parámetro γ fijo, y con distintos valores de Términos de Momento α para cada caso.

8.1.2 Perceptrón Simple: Varios patrones

Se creó un set de 441 patrones, variando los valores de x e y entre 0 y 1 con pasos de 0.05. Los valores d_o son los obtenidos mediante la evaluación de estos valores por la función $z(x, y)$ de la ecuación 8.1.

Se dice que se ha completado una época cuando se ha aplicado un conjunto completo de patrones de entrenamiento a la ANN.

Se aplicó el algoritmo de entrenamiento de retropropagación del error o EBPA, el cual fue repetido iterativamente sobre el conjunto de los patrones en una cantidad t de épocas, aplicando la función de corrección de los pesos una cantidad de veces $k = 1$ en cada uno de los patrones por época.

Se obtuvo el valor absoluto de la diferencia entre el valor deseado y la salida obtenida para cada patrón (ecuación 8.2), para luego obtener el promedio de ese valor en la época 8.3.

$$e(t) = \frac{1}{N} \sum_{i=1}^{N} |(d_o - Out)| \tag{8.3}$$

8.1.3 Perceptrón Multicapa: Varios patrones

Se crearon distintas estructuras de Perceptrones Multicapa. En todos los casos se estableció la capa de entrada con dos neuronas y la capa de salida con una neurona, y se introdujo una o dos capas ocultas, variando en cada caso la cantidad de neuronas que constituyen estas capas. Se entrenaron las ANN de estas estructuras con los 441 patrones utilizados para el perceptrón simple, con el mismo procedimiento descripto para éste.

Se evaluó la convergencia del algoritmo EBPA para resolver el sistema en las distintas estructuras. Se seleccionó aquella que presentaba la mejor convergencia y sobre esta se estudió la sensibilidad al modificar el coeficiente de momento α, así como distintos

valores de k, que determina la cantidad de veces que se emplea el uso de la función de corrección de los pesos en un mismo patrón dentro de la época.

8.2 Algoritmo de Recocido Simulado

Se desarrolló el algoritmo para el empleo de la técnica de recocido simulado para resolver el problema del viajante: En este problema, se parte de la hipótesis de que un viajante debe, partiendo de una ciudad determinada, recorrer N ciudades regresando a la ciudad de partida al finalizar el recorrido. Se quiere obtener el orden de las ciudades donde el camino recorrido tenga el valor mínimo posible.

Se definió una matriz de $Nx2$, donde las columnas de esta matriz contienen las coordenadas de longitud x y latitud y de las ciudades, en un plano que contiene valores entre 0 y 1; Las cantidad de filas N representa el número de ciudades que se deben recorrer. Esta matriz representa el recorrido del viajero, desde la ciudad de partida (primer fila), hasta la última ciudad (fila N) antes de retornar a la ciudad de partida.

La longitud del recorrido L se determina con la expresión de la ecuación 8.4:

$$L = \left(\sum_{i=0}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \right) + \sqrt{(x_N - x_0)^2 + (y_N - y_0)^2} \tag{8.4}$$

Donde x_i e y_i son la latitud y longitud de la ciudad i y N es la cantidad de ciudades por recorrer.

Luego se procede a modificar la posición de dos ciudades que son seleccionadas de forma aleatoria en el recorrido, intercambiando una con otra. Con este nuevo itinerario se vuelve a calcular la longitud del recorrido L con la ecuación 8.4.

Con esta nueva longitud L y la longitud L_0 anterior se calcula la diferencia entre ambas según la ecuación 8.5.

$$\Delta L = L - L_0 \tag{8.5}$$

Este ΔL toma la posición del ΔE de la ecuación 7.16 en la página 45, donde se describe la técnica de SA. Por medio de esta expresión se determina la probabilidad p.

Se obtuvieron una cantidad de valores M para poder realizar un estudio estadístico de los valores de ΔL con los que se trabajarán. A partir del análisis de los valores obte-

nidos se ajustó valor de k y del parámetro Temperatura Inicial T_i, de la ecuación 7.16. Cabe aclarar que el nombre del parámetro se inspira en la magnitud, por la ecuación utilizada en el método, pero no se refiere a una temperatura real como medición de la energía interna del sistema, sino a un parámetro utilizado en el código.

Una vez optimizados estos valores se procedió a hacer un lazo o *loop* donde se varíe el itinerario de la forma descripta anteriormente en cada uno de las iteraciones, y se calcule la longitud de este recorrido. Una vez hecho este cálculo se obtiene el δL y se calcula p para cada caso.

Se define una variable p_0 que en cada paso toma un valor aleatorio entre 0 y 1. Si el p calculado con este cambio es mayor o igual que p_0 entonces se acepta el cambio, en caso contrario se lo rechaza. Cada vez que se acepta un cambio se cuenta el número de éxitos con el contador *iexitos*.

Este proceso se repite a una temperatura T constante, hasta que se cumpla que el número de éxitos $iexitos = 10 \cdot N$ o haya realizado un total de $i = 100 \cdot N$ iteraciones.

Una vez concluido este *loop* , la temperatura desciende en una proporción J de la temperatura actual $T = T \cdot (1 - J)$, y se inicia nuevamente el *loop* de cálculo anterior. Esto se repite hasta un número de iteraciones tal que las Temperatura llega a un valor mínimo.

8.2.1 Evaluación de la eficacia del algoritmo

Se realizó una variación del algoritmo anterior, para que sólo acepte casos donde la longitud nueva calculada sea menor que la anterior, lo que equivale a aceptar valores únicamente de p mayores que 1.

Se evaluó el mismo set de N ciudades con esta variación y con el algoritmo original.

Para finalizar se calculó la longitud de los recorridos obtenidos en todas las combinaciones posibles de ciudades dejando fija la ciudad de inicio, y se obtuvo el menor de los recorridos de esta forma. Esta estrategia es conocida como Fuerza Bruta.

9

Segunda Fase: Sistema de Limitación con Uranio Natural

Una vez desarrollado, implementado y evaluado el algoritmo de entrenamiento con las técnicas presentadas en el Capítulo 8, se procedió a dimensionar distintos conjuntos de patrones con las entradas y las salidas deseadas como opciones a utilizar en el Sistema de Limitación de la CNA-II.

Se realizaron evaluaciones en cuatro etapas.

9.1 Etapa 1: Evaluación con datos históricos

En una primera instancia se evaluaron distintas estructuras de Redes Neuronales Artificiales con datos históricos de la planta desde enero de 2017 hasta agosto de 2017.

9.1.1 Creación de patrones

Se procedió a crear dos conjuntos de datos de patrones de entrenamiento, cuyos valores de entradas sean útiles para ser usados por un Sistema de Limitación.

Para el primer conjuntos se obtuvieron patrones con 18 entradas correspondientes a los datos de las lanzas virtuales de los tres circuitos de vigilancia (Capitulo 5), datos que utiliza el Sistema de Limitación Actual en su algoritmo, y una salida deseada d_o que se corresponde con la distancia mínima al DNB obtenido por la READAT (margen porcentual).

Para la generación de este conjunto se obtuvieron datos de la potencia lineal máxima para cada una de las seis posiciones axiales en los tres circuitos de vigilancia. Los datos de entrada fueron extraídos de los archivos históricos de salida del programa PODESY de la planta. Una vez extraídos, sus valores numéricos fueron divididos por 600, ya que el valor máximo de potencia lineal establecido para los detectores es de 600 (W/cm)[5].Los valores obtenidos del cociente anterior fueron almacenados en forma matricial, en un nuevo archivo, donde cada fila contiene información de las entradas. A esta matriz se le agregó una posición nueva de identificación temporal, conteniendo

la información de la fecha del archivo de PODESY de donde fueron extraídos los datos en formato UNIX, el cual se define como la cantidad de segundos transcurridos desde la medianoche UTC (Tiempo Universal Coordinado) del 1 de enero de 1970.

Luego, desde otro archivo con datos históricos de la planta, se obtuvieron los datos de salida. En este archivo se encuentran, ordenados por fecha, el margen porcentual al DNB calculado tanto por el sistema de limitación actual RELEB, la READAT (a través del programa AXCNA2), y el sistema de Gestión de Combustibles. Este último campo no se utilizó en el PFI.

Manipulando estos datos con una rutina escrita en lenguaje PERL, se sincronizó la información de las salidas con las entradas temporalmente, por medio de una interpolación lineal simple. De esta forma se agregó en la matriz de almacenamiento previamente mencionada, por cada fila de datos tres posiciones más, pertenecientes a los márgenes porcentuales correspondientes a cada conjunto de entradas.

Con un procedimiento análogo se creó otro conjunto de patrones que contuviesen como entradas las 90 señales de los detectores *in–core*. Para ello se extrajeron los datos de las señales de los detectores *in–core* de los archivos de salida PODESY históricos de la planta, se los identificó agregando una celda con la información temporal en formato UNIX y, con la misma rutina utilizada con el otro conjunto de entradas, se sincronizó la información y se agregaron las tres columnas de información de márgenes al DNB, del RELEB, READAT, y Gestión de Combustible a cada fila de señales de detectores.

Posteriormente a ambos conjuntos de patrones se les aplicó un filtro, es decir otra rutina escrita en PERL para descartar los valores de margen de READAT mayores al 50%, ya que se consideró que estos valores no corresponden a estados de planta de interés para el análisis.

9.1.2 Entrenamiento de patrones

Para cada conjunto de patrones, tanto para el sistema que tiene como entradas los datos de las lanzas virtuales, como para el sistema que tiene como entradas las señales de los 90 detectores se realizó un estudio de distintas estructuras de Redes Neuronales Artificiales con una y dos capas ocultas, modificando el número de neuronas en cada capa. Se adaptaron los algoritmos desarrollados en la Primera Fase a los dos sistemas de patrones creados con los datos de la planta. En todos los casos se entrenó la red neuronal artificial con el 70% de los patrones, haciendo uso tanto del algoritmo de retropropagacion del error (EBPA) como del algoritmo de recocido simulado (SA).

9.2 Etapa 2: Entrenamiento de estados Virtuales

9.2.1 Creación de patrones

Para la segunda instancia se generaron otros nuevos conjuntos de patrones de ambos sistemas que contienen además de los datos obtenidos en la etapa anterior, datos de distintas potencias de operación, al 80%, 90% y 110% de la potencia obtenida en los estados de planta de la Etapa 1. Para la obtención de estos datos se generaron nuevos estados virtuales a partir de las señales obtenidas en la instancia anterior.

Para poder realizar este cometido se utilizó una versión portable del programa AXCNA2 de la READAT que se puede ejecutar de manera *off-line*, reconfigurándolo para que tome sus entradas desde un archivo determinado del input en vez de tomar las señales de planta. A su vez se modificó el factor de potencia para el incremento/decremento de la potencia total (q_{fak}) con los valores 0.8, 0.9 o 1.1 según correspondiese.

Se creó una rutina en PERL que tomaba los datos necesarios para el uso del AXCNA2 de cada archivo de salida de PODESY, los multiplicaba por un coeficiente, el cual tomaba el valor de 0.8, 0.9 y 1.1 según el caso y modificaba el archivo de entrada ubicado en el input del programa, imprimiendo los datos obtenidos.

En cada lectura de los archivos de salida de PODESY, se actualizaba el archivo del input de esta manera y luego se iniciaba el programa AXCNA2. Una vez terminada la operación, en cada caso se extrajo del archivo de salida los márgenes de potencia por DNB de la READAT y de Gestión de Combustibles, guardándolos en un archivo de salida que se identificaba con formato temporal UNIX.

En cuanto al margen de potencia obtenido por el RELEB para estos casos, se realizó otra rutina en PERL para el cálculo de los mismos suponiendo que la planta se encontraba con parámetros como la presión P, temperatura de entrada T y revolución de bombas Σn_0 en valor nominal. Esto genera un valor de constante $CKM = 0$ en las ecuaciones 5.1, 5.2 y 5.3 para todos los casos.

De esta forma se obtuvieron los datos para patrones al 90%, 80% y 110% y se generó un archivo similar al producido en la etapa anterior tanto para los valores de las lanzas virtuales como para las señales de los detectores, donde las entradas solo se modificaron multiplicando por un factor de conversión de potencia correspondiente a los datos calculados con el AXCNA2.

9.2.2 Entrenamiento

En esta instancia se reentrenaron las estructuras cuyo desempeño fue el mejor en la etapa anterior tanto para el sistema de lanzas virtuales como entradas, como para el sistema que tiene las señales de los noventa detectores *in – core* como entradas. Este reentrenamiento comenzó a partir de los pesos sinápticos obtenidos luego del entrenamiento de la Etapa 1.

9.3 Etapa 3: Comportamiento ante fallas

En esta instancia se generaron conjuntos de patrones con fallas para evaluar el sistema que tiene como entradas las señales de los noventa detectores y los pesos sinápticos adquiridos en la última época del entrenamiento de la Etapa 2. Se utilizó el conjunto de patrones de entrenamiento de la Etapa 1 como base.

9.3.1 Desconexión de un único detector

Se simuló una desconexión de uno de los 90 detectores sistemáticamente, en todo el conjunto de patrones. Es decir, se estableció el valor en 0 para una de las 90 entradas en todo el conjunto de patrones. Se evaluaron las salidas con el mejor resultado de la ANN obtenido en el entrenamiento de la Etapa 1. Se promediaron los errores de todos los patrones y se repitieron estos pasos para todos los detectores.

9.3.2 Desconexión de una lanza

Se simuló la desconexión de una lanza de las quince sistemáticamente, en todo el conjunto de patrones. Es decir se estableció el valor en 0 para seis de las 90 entradas correspondientes a los detectores de cada lanza en todo el conjunto de patrones. Se evaluaron las salidas con el mejor resultado de la ANN obtenido en el entrenamiento. Se promediaron los errores de todos los patrones y se repitieron estos pasos para todas las lanzas.

9.3.3 Amplificación de la señal en un solo detector

Se simuló la amplificación de la señal en cada uno de los 90 detectores sistemáticamente, en todo el conjunto de patrones. Es decir se estableció el valor en 1 para una de las 90 entradas en todo el conjunto de patrones. Se evaluaron las salidas con el

mejor resultado de la ANN obtenido en el entrenamiento. Se promediaron los errores de todos los patrones y se repitieron estos pasos para todos los detectores.

9.3.4 Amplificación de la señal de una lanza

Se simuló la amplifación de una lanza de las quince sistemáticamente, en todo el conjunto de patrones. Es decir se estableció el valor en 1 para seis de las 90 entradas correspondientes a los detectores de cada lanza en todo el conjunto de patrones. Se evaluaron las salidas con el mejor resultado de la ANN obtenido en el entrenamiento. Se promediaron los errores de todos los patrones y se repitieron estos pasos para todas las lanzas.

9.4 Etapa 4: Implementación del SLN para Funcionamiento *on-line*

Se implementó un programa que funciona tomando las señales de planta en forma periódica y automática cada un minuto y sea capaz de precedir el margen del punto de apartamiento de la ebullición nucleada con el Sistema de Limitación por ANN. Esta ANN contiene la mejor configuración de pesos sinápticos obtenida sobre los casos de entrenamiento de la Etapa 2 para el Sistema de las señales de los 90 detectores-

Se debe tener en cuenta que los casos en los cuales se evaluará la ANN no han sido previamente casos de entrenamiento de la misma.

10

Tercera Fase: Sistema de Limitación para la planta utilizando combustible ULE

En la tercera y última parte se trabajó con datos neutrónicos de la planta simulando el uso de combustible con Uranio Levemente Enriquecido ULE que fueron otorgados por el grupo de neutrónica de NA-SA..

10.1 Creación de patrones

El grupo de neutrónica de NA-SA realizó la simulación de una Gestión de Combustible con el programa PUMA utilizando ULE y restringida a los límites de potencias impuestos para EECC con Uranio Natural. La simulación considera que el núcleo se encuentra en equilibrio de quemado con EECC de ULE en todos los canales.

Estas simulaciones no consideran variaciones en las condiciones de la planta, como así tampoco oscilaciones de xenón o cambio en las posiciones de las barras de control. Las señales de los detectores fueron determinadas para cada estado de la simulación interpolando en la malla de calculo de PUMA el flujo térmico de los volúmenes cercanos a cada uno de los 90 detectores.

Se otorgaron 1278 set de datos neutrónicos obtenidos a partir de esta simulación, los cuales cuentan con el formato de los archivos de salida de PODESY.

10.1.1 Obtención de los datos de Margen al DNB

A partir del conjunto de datos neutrónicos otorgados se utilizó el programa AXCNA2 para determinar el margen de potencia al DNB para cada una de las 1278 configuraciones.

Se utilizó un procedimiento análogo al utilizado en la Etapa 2 de la instancia anterior, generando para cada caso un archivo de entrada para el input de AXCNA2 desde un archivo con formato de salida del programa PODESY.

A partir de estas configuraciones se generaron nuevas configuraciones a distintas potencias de operación, al 75%, 80%, 85%, 90%, 95%, 105% y 110% de la potencia

obtenida en los estados simulados anteriores. De esta manera se obtuvieron 10224 patrones.

A este conjunto de patrones se agregaron otros 2148 patrones generados a partir de configuraciones al 100% . En donde a las simulaciones que no consideran variaciones en las condiciones de la planta, como así tampoco oscilaciones de xenón o cambio en las posiciones de las barras, se le superpusieron las perturbaciones típicas de la distribución de flujo estacionaria, observadas en planta, traducidas en amplitudes de los modos armónicos del mapeo de flujo.

10.1.2 Obtención de los datos RELEB actual

Con el conjunto de 1278 datos otorgados por el grupo de neutrónica se calcularon las constantes para el Sistema de Limitación actual de la planta para las condiciones en que fueron realizadas las simulaciones. Para ello se tuvo como primer supuesto que los datos fueron obtenidos en condiciones nominales de Presión, Temperatura y Revoluciones de la Bomba, con lo cual el factor CKM de las tres ecuaciones 5.1,5.2 y 5.3 (pág 30) resulta nulo. Luego, con un criterio aún más conservativo, se estableció que la constantes C_1 de la ecuación 5.1 y C_2 de la ecuación 5.2 también toman valor nulo. De esta forma las ecuaciones 5.1, 5.2 y 5.3 se transforman en las ecuaciones 10.1, 10.2 y 10.3 para cada circuito de vigilancia i.

$$q'_{zul,1,i} = A_{1,i} \tag{10.1}$$

$$q'_{zul,2,i} = A_{2,i} \tag{10.2}$$

$$q'_{zul,3,i} = A_{3,i} \tag{10.3}$$

Para la obtención de las constantes A_i correspondientes a las posiciones i superiores de los detectores de cada circuito, se obtuvieron para cada uno de ellos los valores de potencia lineal que tendrían los detectores de estas posiciones si se alcanzara la mínima distancia al margen al DNB otorgado por el AXCNA2 en cada estado analizado. Al conjunto de potencias lineales en el margen obtenidas, se calculó la media μ_i y el desvío estandard σ_i. Se determinaron las constantes A_i, mediante la ecuación 10.4, donde se obtiene un valor el cual es superado por el 99,75% de los casos de potencias lineales en el margen obtenidas.

$$A_i = \mu_i - 3\sigma_i \tag{10.4}$$

Con estas constantes obtenidos para cada circuito, se conocen por medio de las ecuaciones 10.1, 10.2 y 10.3 los nueve valores de q'_{zul}. Estos valores se comparan con los obtenidos en las respectivas posiciones de las lanzas virtuales de cada circuito mediante la ecuación 10.5

$$(LEX_{RELEB})_i = \frac{(q'_{zul,i} - q'_i)}{q'_i} \cdot 100 \tag{10.5}$$

Con el valor mínimo de estos 9 valores se calculó el valor que otorgaría el RELEB actual para los 12408 estados simulados.

10.2 Entrenamiento

Se entrenaron ambos sistemas, tanto los patrones del sistema de Lanzas Virtuales como el sistema de señales de los 90 detectores con ANN de estructura análoga a las que obtuvieron mejor desempeño con Uranio Natural.

En una primera instancia se entrenó con el método de EBPA hasta alcanzar un valor de e tan bajo como el método lo permita.

Una vez que se hubo alcanzado un valor límite, sobre el cual el método no fue capaz de encontrar una configuración óptima tal que este error disminuya, se procedió a reentrenar el sistema con el método de SA a partir de este punto.

10.3 Sistema de Limitación

En cada finalización de una instancia de entrenamiento, tanto para el sistema de Lanzas Virtuales como para el sistema de señales de los noventa detectores se aplicaron los resultados obtenidos para creación de un Sistema de Limitación en cada caso, y se lo comparó con los valores que se obtendrían con el algoritmo de cálculo del sistema de limitación actual en los mismos casos.

Parte V

Resultados y Discusiones

Primera Fase : Evaluación de Algoritmos

Atómico, se dice atómico

HJS

11.1 Perceptrón simple: patrón único

Se evaluaron los algoritmos de creación de estructura de la ANN, la función que evalúa la ANN, y su entrenamiento mediante la función que corrige los pesos sinápticos para un único patrón.

Se tomaron los valores de x, y y d_o presentados en la Tabla §11.1, donde las entradas son los valores de x e y, y d_o es el valor de salida deseada, obtenido de evaluar la función $z(x,y)$ en x y en y mediante la ecuación 8.1 presentada en el Capítulo 8.

x	y	d_o
0,6	0,1	0,26244

Tabla 11.1: Valores del patrón utilizado para la evaluación del algoritmo.

11.1.1 Variación del coeficiente de aprendizaje

Se realizaron entrenamientos del mismo patrón antes mencionado, modificando para cada caso el valor del Coeficiente de Aprendizaje γ de la Tabla §11.2.

Caso	γ
1	0,99
2	0,01
3	0,5

Tabla 11.2: Valores del coeficiente de aprendizaje γ para los distintos casos de estudio.

En la Figura §11.1 se muestra la evolución del valor absoluto de la diferencia entre

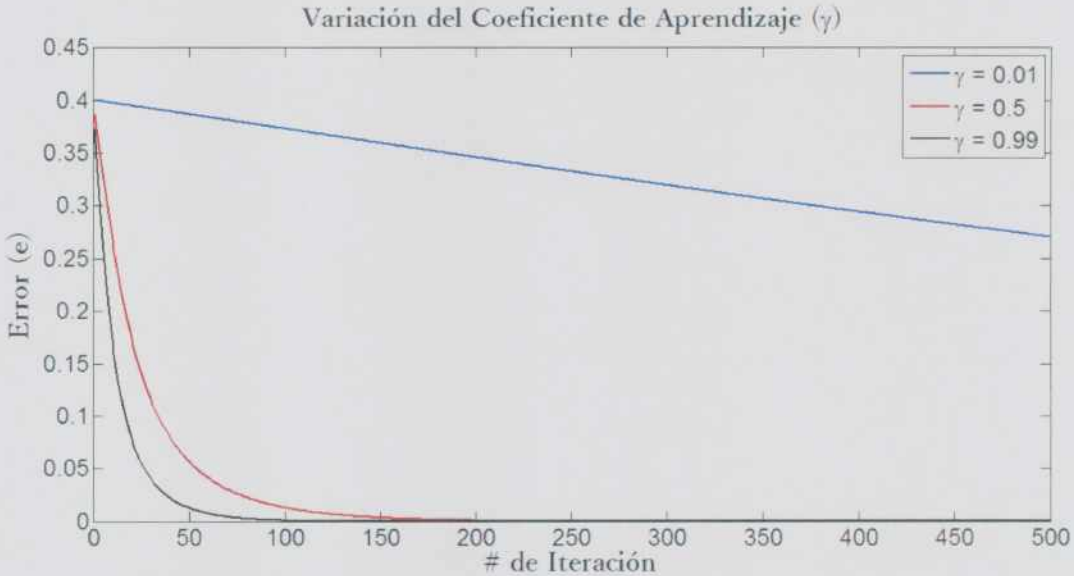

Figura 11.1: Evolución del error en función del número de iteración para los distintos casos de estudio.

el valor deseado d_o y el valor otorgado por la ANN, denominado error e, en función del número de iteración obtenidos para cada valor de γ.

Se puede observar que a medida que se aumenta el coeficiente de aprendizaje γ, el valor e disminuye en menos iteraciones, convergiendo más rápidamente.

11.1.2 Introducción del término de momento

Se realizó otra serie de entrenamientos con el parámetro de $\gamma = 0{,}99$ fijo, que es el valor con el cual se obtuvo la mejor evolución en el estudio anterior. En esta serie de entrenamientos se introdujo un término de momento α en cada caso, cuyos valores son presentados en la Tabla §11.3.

Caso	γ	α
4	0,99	0,1
5	0,99	0,3
6	0,99	0,5
7	0,99	0,7
8	0,99	0,9

Tabla 11.3: Valores del coeficiente de aprendizaje y de momento para los distintos casos de estudio.

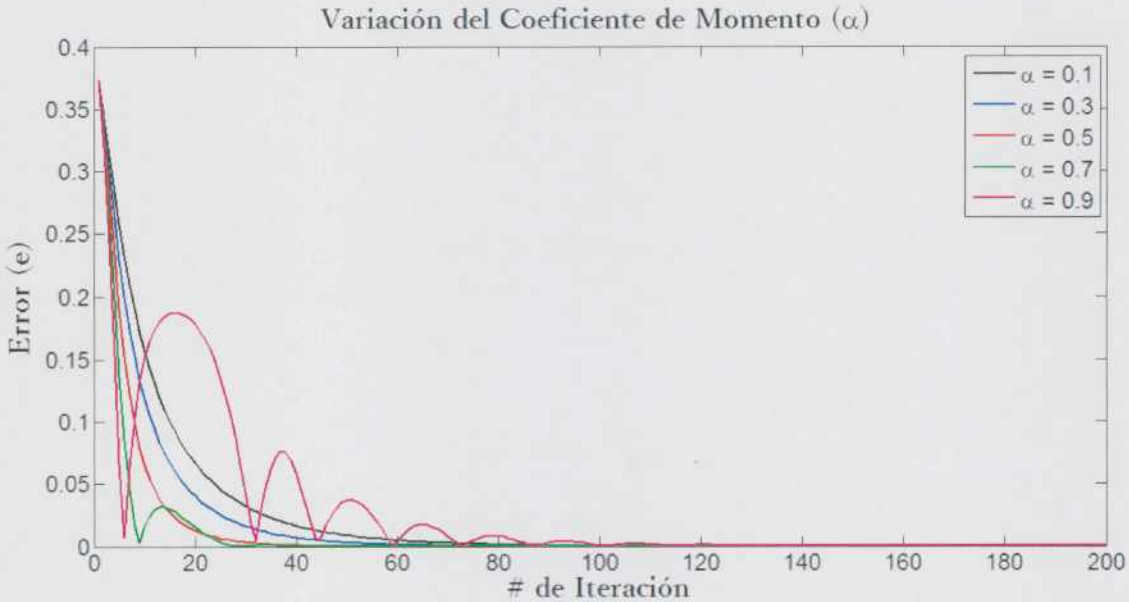

Figura 11.2: Evolución del error en función del número de iteración para los distintos casos de estudio

En el gráfico de la Figura §11.2 se presenta la evolución del error e para distintos casos en función de las iteraciones realizadas. Se puede ver que hasta el valor de $\alpha = 0{,}5$ el valor de e tiene una tasa de convergencia más elevada monótonamente decreciente.

Cuando $\alpha = 0{,}7$ inicialmente la tasa de convergencia es más elevada que el caso anterior. No obstante al alcanzar la iteración 9 el error e aumenta y comienza un proceso oscilatorio amortiguado convergente.

Si el valor del coeficiente de momento es mayor que éste, por ejemplo $\alpha = 0{,}9$ lo que sucede es que la amplitud de las oscilaciones se amplifica.

A partir de la ecuación 7.15 del Capítulo 7 se puede ver que el término de momento se encuentra sumando al término obtenido por la regla de Retropropagación o *BackPropagation*.

Como se mencionó en la descripción del método (pág 45), el término de momento tiene la finalidad de reducir las posibles oscilaciones que se durante el entrenamiento. Este término no debería ser mucho mayor al término de corrección calculado por retropropagación del error ya que el algoritmo no incorporaría las correcciones del entrenamiento y se alejaría del mínimo error, que es el objetivo de este método por descenso del gradiente.

11.2 Perceptrón simple: varios patrones

Una vez que se determinó que el algoritmo funciona de manera correcta otorgando resultados positivos para un solo patrón, se procedió a analizar el comportamiento del perceptrón simple con múltiples patrones. Para ello se entrenó la estructura con 441 patrones, los cuales se obtienen variando los valores de entrada x e y entre 0 y 1 con pasos de 0,05. Los valores d_o son los obtenidos mediante la evaluación de estos valores por la ecuación 8.1 Los valores de $\gamma = 0{,}99$ y $\alpha = 0{,}5$ son los óptimos obtenidos según en estudio con un solo patrón.

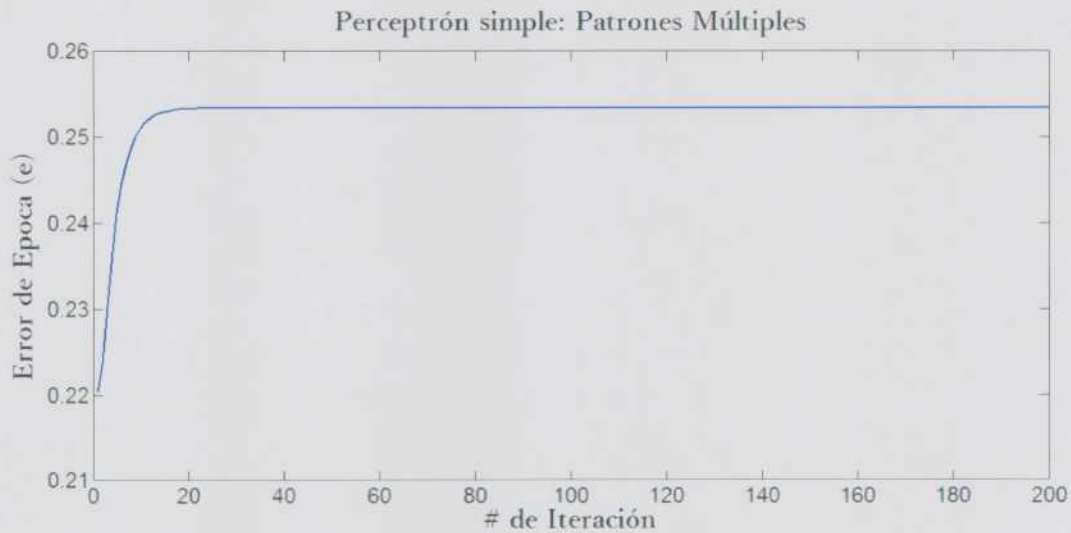

Figura 11.3: Evolución del error de la época en función del número de época para el caso de estudio

En la Figura §11.3 se presenta el gráfico de la evolución de e en función del número de iteraciones para este sistema. Se puede ver que e se incrementa hasta un valor límite por encima de $e = 0{,}25$ y se paraliza en ese valor. Debido a la simpleza de la estructura, se encuentra limitado para poder resolver el sistema.

Las salidas obtenidas en la época número 200 se presentan en la Figura §11.4. Se puede observar que las distintas salidas de la ANN no se adaptan a la salida deseada sino que se se encuentran en las inmediaciones de un plano medio a la superficie.

Esto es así porque el perceptrón simple no tiene la capacidad de resolver este sistema de forma adecuada, ya que los valores no pueden separarse por un hiperplano. De esta forma se demuestra empíricamente las limitaciones de este sistema simple, y la necesidad de implementar una estructura más compleja como el perceptrón multicapa para poder resolver este sistema eficazmente.

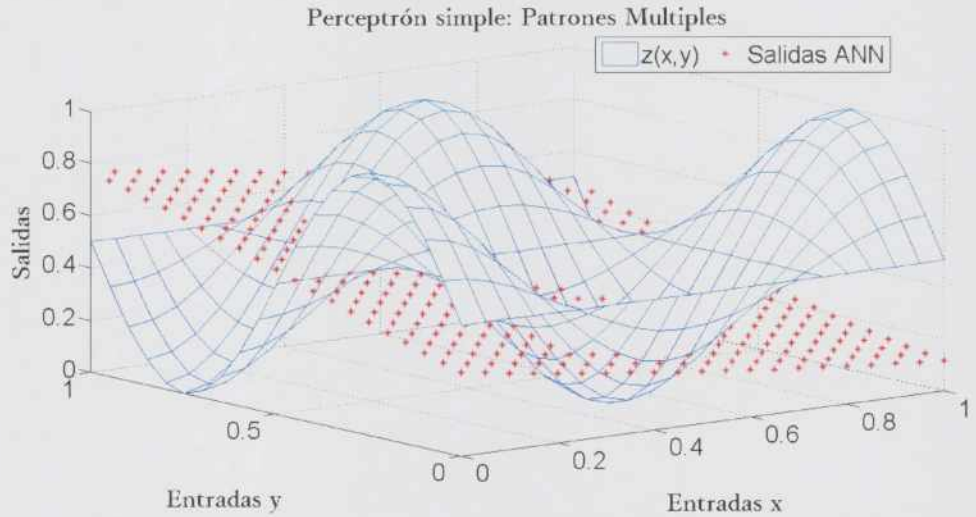

Figura 11.4: Salidas obtenidas con el perceptrón simple, luego del entrenamiento de 200 épocas con 441 patrones

11.2.1 Perceptrón Multicapa

En esta instancia se entrenaron distintas estructuras de Redes Neuronales Artificiales con 441 patrones, variando los valores de x e y entre 0 y 1 con pasos de 0,05. Los valores d_o son los obtenidos mediante la evaluación de estos valores por la función 8.1 Las estructuras de las distintas redes son presentadas en la Tabla §11.4.

Caso	# de capas	Estructura
1	3	[2 1 1]
2	3	[2 2 1]
3	3	[2 6 1]
4	3	[2 10 1]
5	4	[2 1 1 1]
6	4	[2 1 2 1]
7	4	[2 2 1 1]
8	4	[2 4 2 1]
9	4	[2 2 4 1]
10	4	[2 2 10 1]
11	4	[2 10 2 1]

Tabla 11.4: Distintas estructuras de perceptrón multicapa entrenadas con 441 patrones.

Los valores de aprendizaje utilizados fueron $\gamma = 0{,}99$ y $\alpha = 0{,}1$. En el gráfico presentado en la Figura §11.5 se puede ver la evolución de e en función del número de iteración para cada caso. Del análisis de este gráfico se puede ver que de todas las configuraciones, la que mejor resuelve el sistema es la configuración número 11, la cual tiene 2 capas ocultas, de 10 y 2 neuronas respectivamente.

Luego, con esta estructura ([2 10 2 1]) se realizó el entrenamiento variando el número de veces k que se reentrena cada patrón por época, tomando los valores 1, 3, 5 y

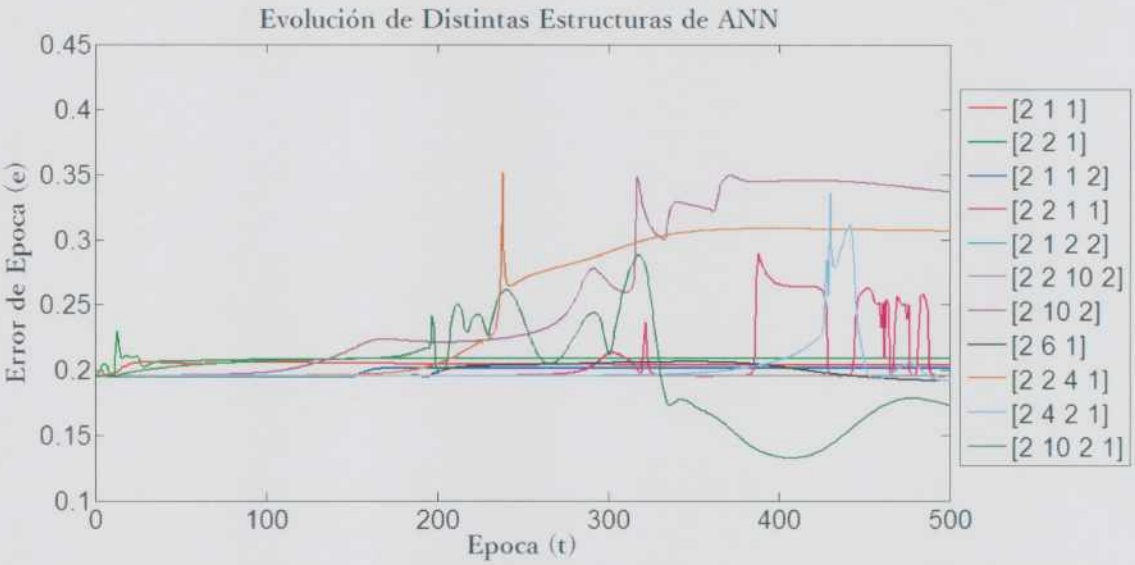

Figura 11.5: Evolución del error de época, obtenido en el entrenamiento de distintas estructuras de Perceptrones multicapa con 441 patrones

10.

Se obtuvieron las evoluciones presentadas en la Figura §11.5.

A partir de lo observado en la Figura §11.5 el valor $k = 3$ presenta la óptima convergencia.

Posteriormente se entrenaron los patrones dejando fijo el valor de $\gamma = 0{,}99$ y $k = 3$ y variando el valor del coeficiente de término de momento α, con lo cual se obtienen las siguientes evoluciones presentadas en el gráfico de la Figura §11.7.

Analizando el gráfico de la Figura §11.7 se puede ver que a diferencia de lo obtenido para un sólo patrón en este caso el término de momento con un coeficiente $\alpha = 0{,}5$ tiene efectos contraproducentes en el entrenamiento, siendo el valor óptimo para este tipo de estructuras el valor de $\alpha = 0{,}1$, con el cual se converge en una menor cantidad de iteraciones en este sistema.

Las salidas y el error obtenido para la época numero 100000 para la estructura de perceptrón multicapa [2 10 2 1], con un $\gamma = 0{,}99$ y $\alpha = 0{,}1$, se grafican en la Figura §11.8.

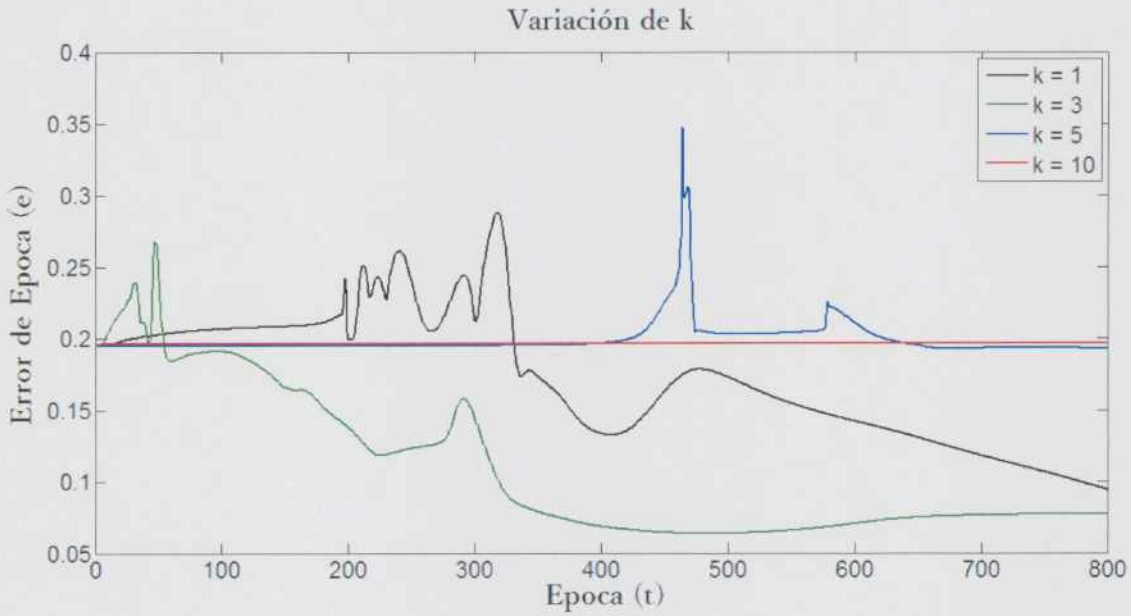

Figura 11.6: Evolución del error de época, obtenido en el entrenamiento de la estructura de perceptrón multicapa [2 10 2 1] con 441 patrones para distintos valores de k.

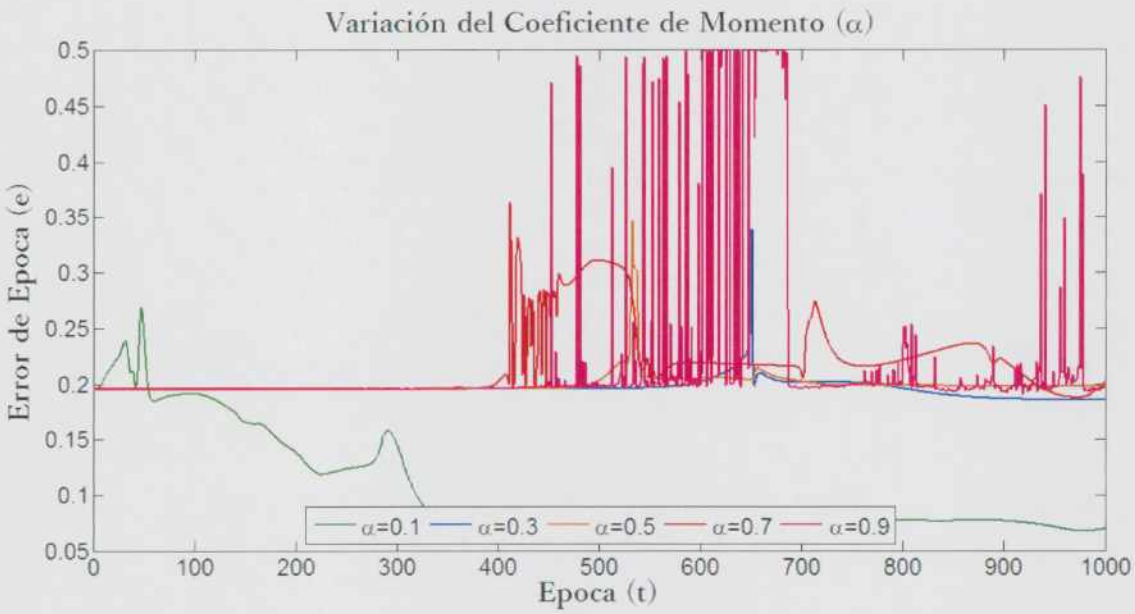

Figura 11.7: Evolución del error de época, obtenido en el entrenamiento de la estructura de perceptrón multicapa [2 10 2 1] con 441 patrones para distintos valores del término α.

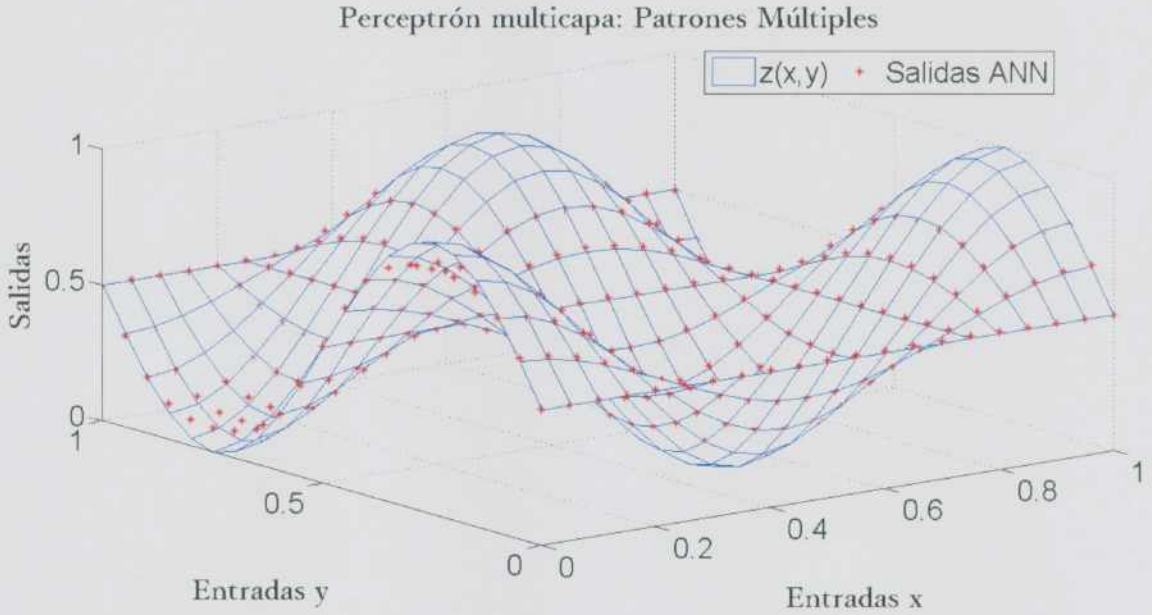

Figura 11.8: Salidas obtenidas con la estructura [2 10 2 1], en la época número 100000

11.3 Recocido Simulado

11.3.1 Determinación de constantes para el problema del Viajante

La constante k (análoga a la Constante de Boltzmann de la ecuación 7.16 del Capítulo 7) y la variable T_i Temperatura de Inicio, fueron determinadas a partir del estudio estadístico de $M = 100$ casos para distintos valores N de cantidad de ciudades en el itinerario. Esta técnica se basa en el método de recocido de los materiales, pero en el caso de estudio, no se está evaluando un cambio de energía interno, como al que hace referencia dicha ecuación, sino un valor obtenido para una diferencia de distancias producidas por el cambio en el orden de la ciudades en el itinerario del viajante. Por ello se debe hallar un valor k que se adecúe a esta variación, ya que si utilizamos la Constante de Boltzmann ($K = 1,380 \times 10^{-23}$ J/K), la cual se utiliza para la conversión de temperatura a energía, al ser un rango de valores completamente distinto al utilizado, no se obtendrán los valores de probabilidad útiles para el objetivo que se quiere cumplir.

Al realizar la diferencia de longitudes entre el recorrido anterior y el actual (ΔL) para los distintos valores N de ciudades se obtienen los parámetros de media μ y desvío del promedio σ_p presentados en la Tabla §11.5.

N	μ	σ
20	0,03	0,54
50	0,09	0,53
70	0,14	0,56
100	0,30	0,61

Tabla 11.5: Estudio estadístico del ΔL para distintos valores N de ciudades

Con estos valores se determinaron las probabilidades obtenidas mediante la Ecuación 7.16 del Capitulo 7 (pág 45) para distintos valores de k para una $T_i = 1000$. Estas probabilidades son presentadas en la Tabla §11.6.

ΔL	k	Probabilidad
0,03	$1,38 \times 10^{-7}$	$3,8\times 10^{-95}$
0,03	$1,38 \times 10^{-6}$	$3,6\times 10^{-10}$
0,03	$1,38 \times 10^{-5}$	$1,1\times 10^{-1}$
0,03	$1,38 \times 10^{-4}$	$8,1\times 10^{-1}$
0,03	$1,38 \times 10^{-3}$	$9,7\times 10^{-1}$
0,03	$1,38 \times 10^{-2}$	$9,9\times 10^{-1}$
0,30	$1,38 \times 10^{-6}$	$3,8\times 10^{-95}$
0,30	$1,38 \times 10^{-5}$	$3,6\times 10^{-10}$
0,30	$1,38 \times 10^{-4}$	$1,1\times 10^{-1}$
0,30	$1,38 \times 10^{-3}$	$8,1\times 10^{-1}$
0,30	$1,38 \times 10^{-2}$	$9,8\times 10^{-1}$
0,30	$1,38 \times 10^{-1}$	$9,9\times 10^{-1}$

Tabla 11.6: Probabilidades Obtenidas para distintos valores de k, con una T = 1000.

En la tabla §11.6 se tomaron en cuenta los valores de μ obtenidos para $N = 20$ ciudades, y un valor μ obtenido del estudio de $N = 100$ ciudades. Con esto se determinó el valor de k a utilizar, ya que se necesita un valor tal que otorgue una resolución óptima del algoritmo.

En base a lo obtenido, se estableció como constante $k = 1,38 \times 10^{-4}$, ya que con ese valor las probabilidades obtenidas se encuentran en el rango para aceptar entre el 10% y el 80% de los recorridos resultantes más largos.

11.3.2 Evaluación del Algoritmo para distintos valores N

Se evaluó el algoritmo de recocido simulado (también conocido como el algoritmo de Metrópolis)[10] para el problema del viajante, con una cantidad N de 5, 13 y 20 ciudades, modificando el código para que acepte o no los valores negativos de diferencias de longitudes.

Partiendo de un mismo estado inicial, las longitudes de recorrido obtenidas por las variaciones del algoritmo y por fuerza bruta, se presentan en la Tabla §11.7;

N	Método utilizado	L_f	t [s]
5	Recocido Simulado	2,44	2,32
5	Recocido Simulado aceptando solo p>1	2,44	2,12
5	Fuerza Bruta	2,44	0,6
13	Recocido Simulado	3,27	29,53
13	Recocido Simulado aceptando solo p>1	3,77	27,94
13	Fuerza Bruta	3,27	105,78
20	Recocido Simulado	5,44	58,28
20	Recocido Simulado aceptando p>1	5,98	32,88
20	Fuerza Bruta	5,44	(>1 día)

Tabla 11.7: Longitudes obtenidas con distintos métodos para distintas cantidades N de ciudades, con una $T_i = 1000$.

La ventaja más significativa es el tiempo de cálculo utilizado en el algoritmo de recocido simulado con respecto al utilizado por fuerza bruta. Entre los casos analizados, si se tienen 5 ciudades existe una cantidad de $4! = 24$ combinaciones de ciudades posibles en el itinerario, mientras que para 20 ciudades se tienen $19! = 1{,}2 \times 10^{17}$, con lo cual se puede ver que al aumentar el número de ciudades el número de combinaciones aumenta sustancialmente, y con ello el tiempo de cálculo empleado por el algoritmo, ya que debe evaluar cada una de ellas.

Si se compara el tiempo de cálculo utilizado por el algoritmo de recocido simulado con el tiempo empleado por un algoritmo de fuerza bruta, éste es similar para un número de ciudades pequeño ($N = 5$), pero la relación entre ellos aumenta sustancialmente a medida que se aumenta el número de ciudades. Como se ha comprobado, se llega al mismo resultado mediante ambas técnicas, pero en un tiempo mucho menor con SA.

Al comparar las evoluciones del algoritmo para el método de recocido simulado que acepta los casos con probabilidades mayores a un número aleatorio, este converge en un valor de recorrido menor que el algoritmo únicamente acepta las probabilidades mayores a uno.

Esto se debe a que al aceptar estas probabilidades, se está aceptando una longitud de recorrido mayor a la que se tiene como mejor opción, lo cual le permite al algoritmo evitar encerrarse en un mínimo local, y avanzar hacia el mínimo global del problema. En el caso del método que no le permite esta opción, otorga como mejor recorrido un itinerario de mayor longitud, ya que queda estancado en un mínimo local.

12

Segunda Fase: Sistema de Limitación con Uranio Natural

12.1 Etapa 1: Datos de Planta

12.1.1 Sistema de Lanzas Virtuales

Retropropagación del Error

A partir de los datos de los patrones que contienen como entradas los datos de las lanzas virtuales, se procedió a entrenar con ellos distintas estructuras de Redes Neuronales Artificiales.

Del total de 5260 patrones existentes en este sistema, se creó una rutina que separaba aleatoriamente patrones en dos archivos. Esta rutina tomaba el 70% (3682) de los patrones para el entrenamiento de la ANN y los almacenaba en un archivo. Con el 30% restante (1578) procedía análogamente, almacenándolos en otro archivo.

Se procesaban los datos almacenados en ambos archivos mencionados anteriormente. En cada época t de la rutina principal se realizaba el entrenamiento de la ANN con los patrones de entrenamiento. Para poder utilizar la ANN los datos contenidos en cada uno de ellos debieron ser convertidos a valores entre cero y uno. En el caso de las entradas, como la potencia lineal máxima (q'_M) de cada posición de las lanzas virtuales tiene un límite en 600 W/cm [5], los valores contenidos en los patrones fueron divididos por 600. Las salidas esperadas o deseadas fueron calculadas a partir del margen al punto de ebullición nucleada en porcentaje obtenido por la READAT, el cual fue dividido en 50, ya que este es el valor máximo de porcentaje que contienen los patrones.

Una vez concluido el entrenamiento en todos los patrones, en cada época se evaluaban tanto los patrones del entrenamiento como los de los testigos con los pesos obtenidos, calculando el promedio del error obtenido en cada uno de ellos.

Caso	# de capas	estructura	ECM
1	3	[18 6 1]	$3{,}5\times10^{-3}$
2	3	[18 12 1]	$2{,}4\times10^{-3}$
3	3	[18 18 1]	$1{,}6\times10^{-3}$
4	3	[18 36 1]	$8{,}6\times10^{-2}$
5	4	[18 6 3 1]	$1{,}6\times10^{-3}$
6	4	[18 12 6 1]	$1{,}5\times10^{-3}$
7	4	[18 12 12 1]	$6{,}1\times10^{-4}$
8	4	[18 18 12 1]	$4{,}6\times10^{-4}$
9	4	[18 36 18 1]	$2{,}2\times10^{-2}$

Tabla 12.1: Distintas estructuras entrenadas con 3682 patrones.

Se entrenaron las estructuras presentadas en la Tabla §12.1 con el algoritmo de retropropagación, en todos los casos se utilizaron los valores de coeficiente de aprendizaje $\gamma = 0{,}99$ y coeficiente de término de momento $\alpha = 0{,}1$, que fueron los valores óptimos de entrenamiento en la primera parte del PFI.

Se evaluaron todos los patrones, de entrenamiento y testigos con los pesos obtenidos en la época de entrenamiento número $t = 10^6$ de todas las estructuras y se cálculo el Error Cuadrático Medio (ECM) a partir de la ecuación 12.1.

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(d_{oi} - out_i)^2 \tag{12.1}$$

Donde $n = 5260$ es el número de patrones totales por época, d_o la salidas esperadas para cada patrón i y out las salidas obtenidas para cada patrón i.

Se obtuvieron las evoluciones del error promedio por época para el conjunto de entrenamiento con las distintas estructuras, las cuales son presentadas en el gráfico de la Figura §12.1.

De las distintas estructuras, la número 8 con una estructura de cuatro capas y con 18, 18, 12 y 1 neuronas respectivamente en cada una de ellas, fue una de las que mejor evolución del error tuvo según lo observado en la Figura §12.1 y fundamentalmente la que otorgó un menor valor de ECM en la época $t = 10^6$, con lo cual es la que mejor se adapta a las salidas que se desean obtener.

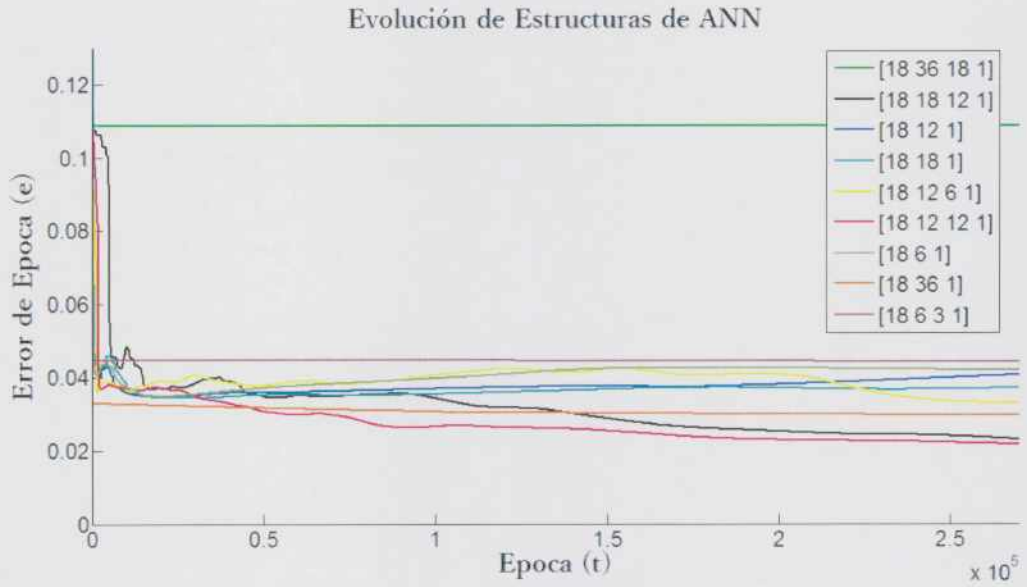

Figura 12.1: Evolución del error de época, obtenido en el entrenamiento de distintas estructuras de perceptrón multicapa con 3682 patrones

Se grafican los promedios del error obtenido por época, tanto para los patrones de entrenamiento como los patrones testigos, en función de la época en la Figura §12.2

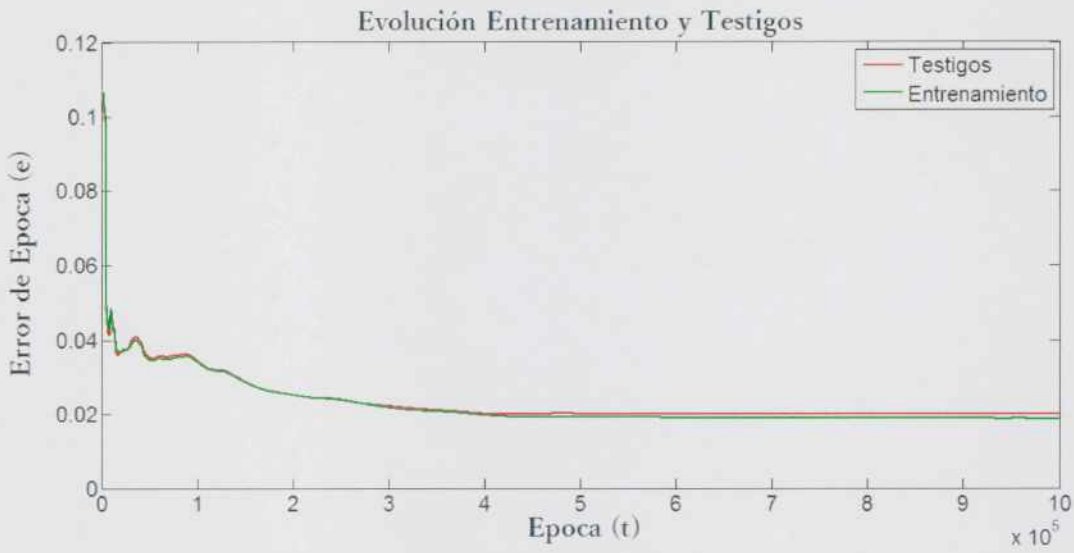

Figura 12.2: Evolución del error de época, para el conjunto de entrenamiento con 3682 patrones (verde), y testigos con 1578 patrones (rojo).

Las salidas para la época $t = 10^6$ se presentan en la Figura §12.3.

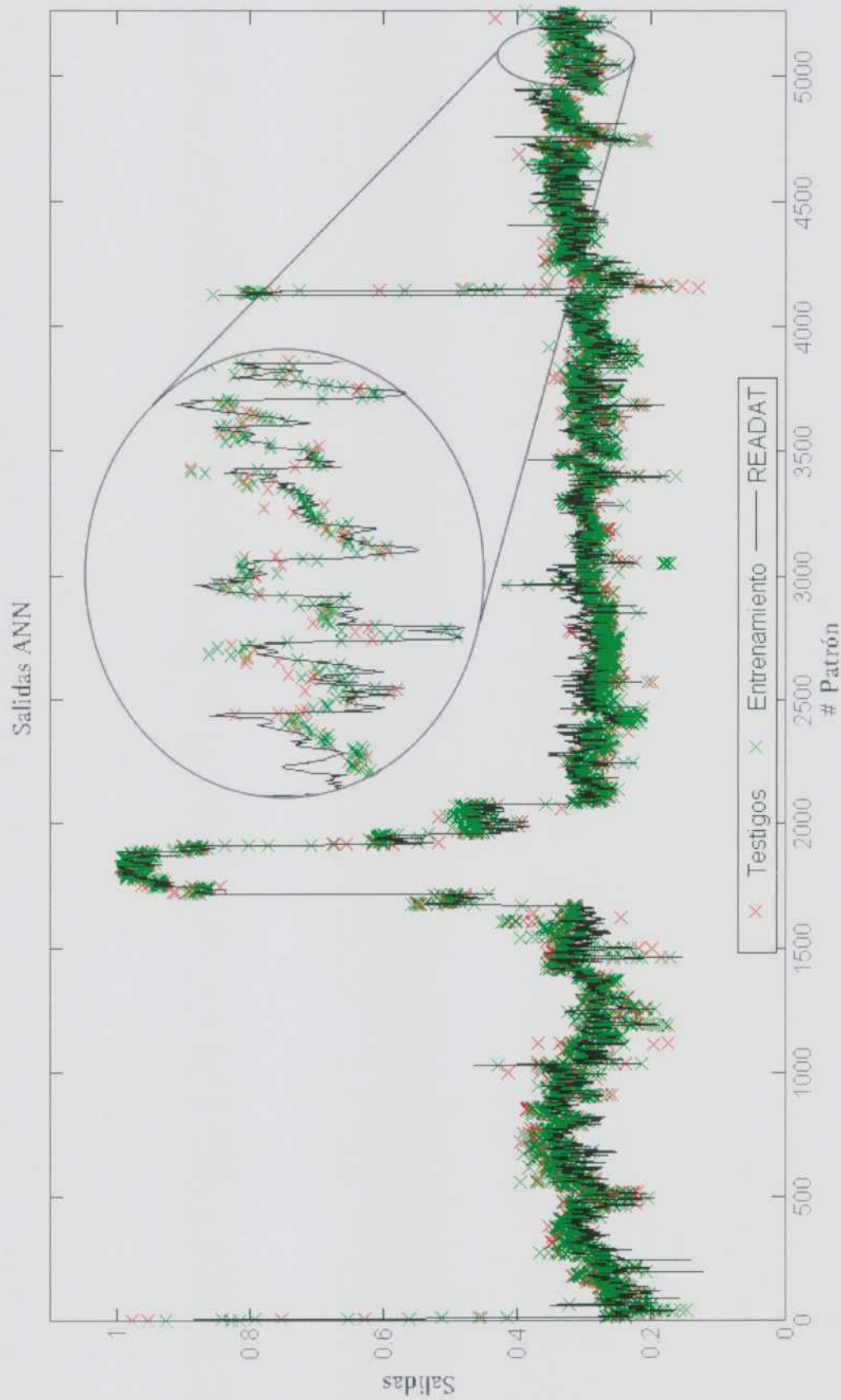

Figura 12.3: Salidas esperadas, Margen al DNB obtenido por la READAT dividido en 50 (negro) y salidas obtenidas por la Red Neuronal Artificial (Salidas ANN), para testigos (rojo) y para entrenamiento (verde)

En la Figura §12.3 se muestra el total de los 5260 patrones, de los cuales los primeros 1717 se corresponden a los datos obtenidos desde el 03 de enero de 2017 hasta el 17 de marzo de ese mismo año. Entre patrón y patrón hay una diferencia de aproximadamente una hora en la adquisición de los datos.

El periodo transcurrido entre el 17 de marzo y el 05 de agosto la planta estuvo fuera de servicio por reparaciones. A partir del patrón 1718 hasta el 4125 se corresponde a otro periodo de tiempo entre el 05 de agosto hasta el 17 de noviembre de 2017, donde la lanza número se encontraba desconectada. Por último los patrones 4126 hasta el final van desde el 26 de noviembre hasta principios de enero de 2018, periodo en el cual la central vuelve a estar operativa luego de encontrarse 9 días fuera de servicio.

En el gráfico la línea negra indica los valores de salida deseada para la ANN, los cuales fueron obtenidos a partir del margen al DNB obtenido por la READAT. Este margen que está en porcentaje, fue dividido en 50, ya que sólo se contemplan los valores hasta 50% de margen.

Al observar esta línea negra se puede ver el comportamiento del margen al DNB obtenido por la READAT en los distintos periodos y estados operativos. Sufre un incremento abrupto antes de salir de servicio (cada vez que se baja potencia) y puede verse una disminución pronunciada cada vez que vuelve al mismo.

En el Capítulo 8 se describió como se crearon los patrones, y se explicó que se descartaron los datos en los cuales el margen al DNB eran mayores al 50%, esto es debido a que se corresponden con estados operativos de poco interés (cuando se está saliendo de servicio u otros estados de baja exigencia para la integridad de los EECC).

También es de interés destacar que existen pruebas repetitivas en el reactor que son obligatorias para saber si los sistemas funcionan correctamente, entre ellas conmutar las bombas del moderador por lo que aumenta la temperatura en el tanque Esto genera una variación abrupta en el margen obtenido, que se refleja en el gráfico como picos que se ven con cierta frecuencia a lo largo de los patrones.

La ANN brinda salidas que se corresponden con lo esperado en cuanto a las salidas deseadas. Se tiene también una vista amplificada de los patrones obtenidos con los datos de la primera quincena de enero. Aquí se puede apreciar más detalladamente cómo las salidas otorgadas por la ANN al evaluar los patrones que fueron entrenados y aquellos que quedaron como patrones testigos, se adaptan en gran parte a los valores deseados con desviaciones que se dan tanto por defecto como por exceso, aunque tienden al exceso, otorgando una salida mayor a la que se espera.

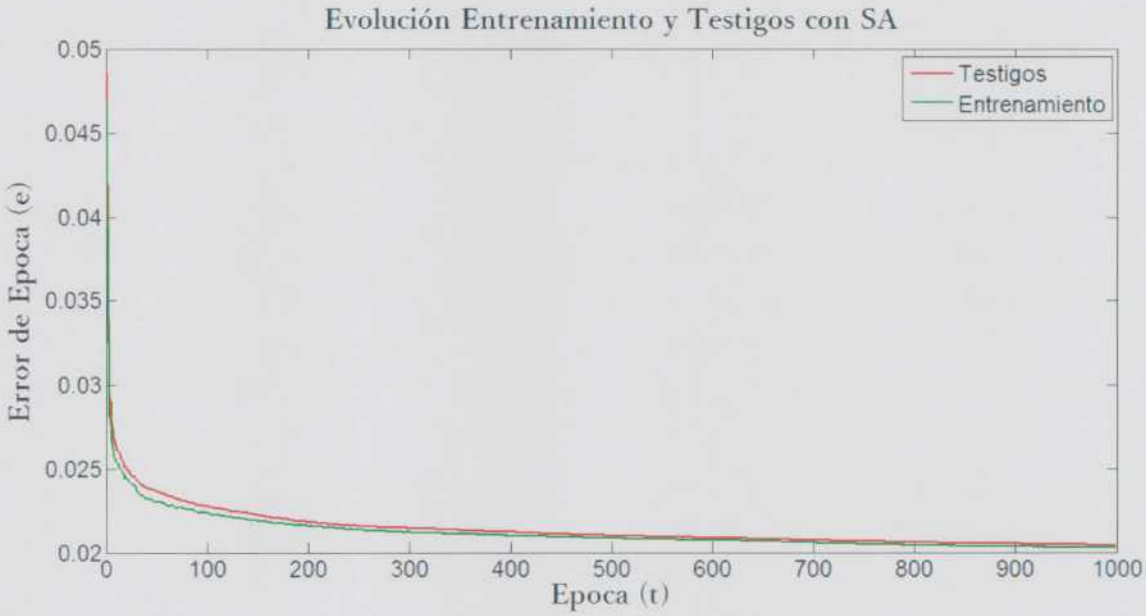

Figura 12.4: Evolución del error de época, para el conjunto de entrenamiento con 3682 patrones (verde), y testigos con 1578 patrones (rojo).

Recocido Simulado

Con la estructura numero 8 de ANN de 4 capas con 18, 18 12 y 1 neuronas respectivamente en cada una de ellas, se entrenaron los patrones de entrenamiento con el algoritmo de Recocido Simulado a partir de un estado inicial con pesos aleatorios.

Se entrenaron los 3682 patrones del archivo de entrenamiento del sistema de lanzas virtuales con el algoritmo de recocido simulado, con un valor de constante $k = 1{,}38 \times 10^{-4}$, y comenzando con una temperatura de T =1000, la cual fue descendiendo un 10% del estado anterior en cada época. Definiéndose a la época como el lazo o *loop* que se completa cada vez que se obtienen una cantidad de éxitos $iexitos = 52600$ o un número de iteraciones $i = 526000$.

En la Figura §12.4 se muestra la evolución del error en función de las épocas. A diferencia con la evolución del error para el mismo sistema entrenado con el algoritmo de Algoritmo de Retropropagación del Error (Figura§12.2), en esta evolución se puede ver una convergencia a valores similares en una menor cantidad de épocas.

De esta forma, se puede ver como un método pseudo heurístico como el de Recocido Simulado puede en ciertos casos ser más eficiente temporalmente en comparación con un método de descenso de gradiente como lo es el Algoritmo de Retropropagación del Error.

Las salidas obtenidas del entrenamiento se presentan en la Figura §12.5.

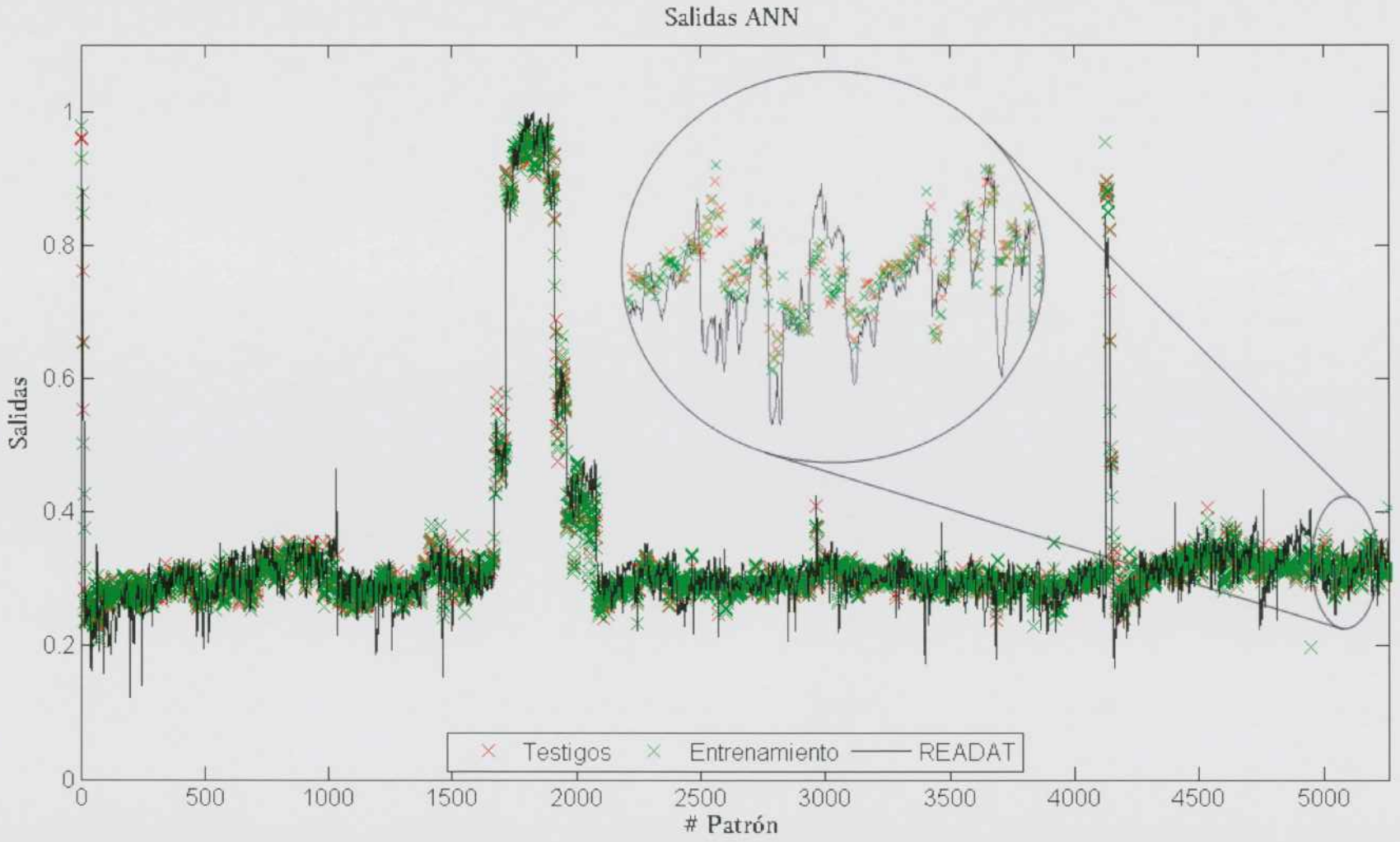

Figura 12.5: Salidas esperadas, Margen al DNB obtenido por la READAT dividido en 50 (negro) y salidas obtenidas por la Red Neuronal Artificial (Salidas ANN), para testigos (rojo) y para entrenamiento (verde)

Técnica	ECM	μ	σ
Algoritmo de Retropropagación del Error	$4{,}8\times10^{-4}$	$-8{,}8\times10^{-4}$	0,022
Recocido Simulado	$7{,}6\times10^{-4}$	$-3{,}73\times10^{-4}$	0,028

Tabla 12.2: Parámetros obtenidos del análisis de las diferencias entre las salidas deseadas y las salidas otorgadas por la ANN luego de ser entrenadas por EBPA y SA.

Se realizó la diferencia entre las salidas deseadas y las salidas obtenidas con ambas técnicas y se obtuvieron los errores presentados en la Figura §12.6.

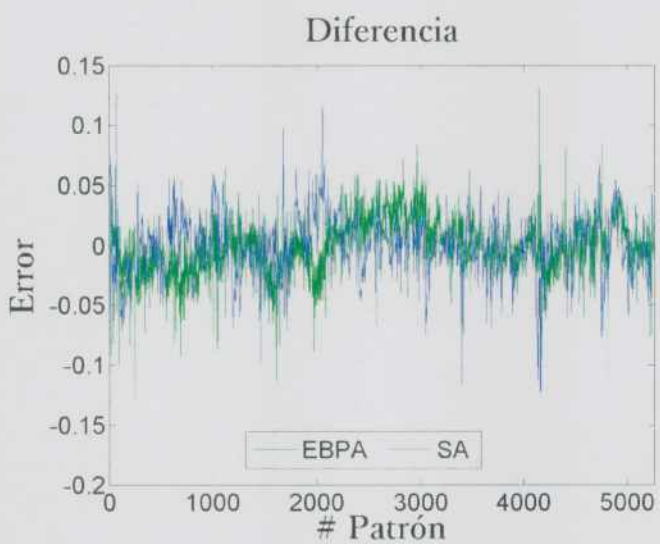

Figura 12.6: Diferencias entre las salidas esperadas y las salidas obtenidas con la técnica EBPA (verde), y las salidas esperadas y las salidas obtenidas con la técnica de SA (azul)

Se calculó el Error Cuadrático Medio (ECM) de las salidas obtenidas con los pesos sinápticos obtenidos en la última época de entrenamiento de cada técnica mediante la ecuación 12.1 y se obtuvo la media μ y la desviación estandard σ de las diferencias.

Los valores obtenidos se presentan en la Tabla §12.2.

Como se puede ver en la Figura §12.6 de los errores a lo largo de todos los patrones, si bien la media μ y el desvío σ se encuentran en el mismo orden de magnitud, puede observarse que con el método SA se obtienen diferencias mucho más variadas de un patrón a otro, y tiene un ECM mayor con lo cual estaría más alejada de los valores que se esperan como respuesta de la ANN.

Sistema de Limitación por ANN

Teniendo en cuenta los parámetros obtenidos, se calculan los valores del Sistema de Limitación por ANN mediante la ecuación 12.2.

$$SL = (Out + \mu - 2\sigma) \cdot 50 \tag{12.2}$$

Siendo SL los valores del Sistema de Limitación en porcentaje de distancia mínima al punto de ebullición nucleada, Out las salidas de la ANN, μ la media obtenida de las diferencias entre las salidas deseadas y las obtenidas por la ANN, y σ la raíz del ECM obtenido. Todo se encuentra multiplicado por 50 para obtener valores equivalentes en porcentaje de margen al DNB.

De esta forma al sumar el μ de las diferencias se "trasladan"las salidas obtenidas a un valor mas cercano al valor deseado, luego se resta dos veces σ, para que el Sistema de Limitación obtenido sea conservador y otorgue una distancia mínima al punto de ebullición nucleada que esté siempre por debajo del 95% de los casos contemplados.

En el gráfico de de la Figura §12.7, se presentan superpuestos los gráficos del margen al DNB en porcentaje obtenidos con los Nuevos Sistemas de Limitación por ANN con ambas técnicas, el Sistema de Limitación actual RELEB, y el margen al DNB Calculado por la READAT.

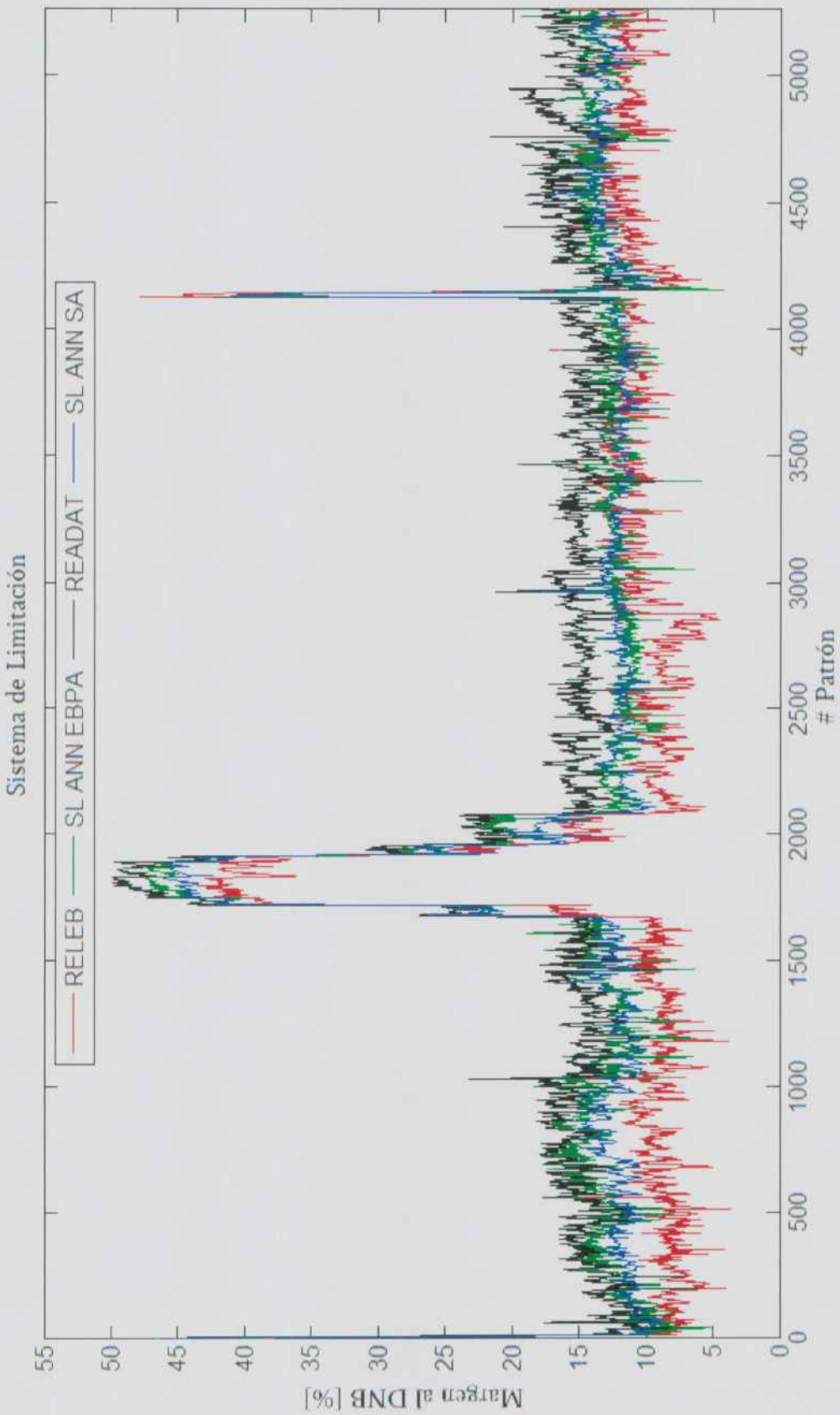

Figura 12.7: Margen al DNB obtenido con el Sistema de Limitación por ANN utilizando la técnica de EBPA (verde) y la técnica SA (azul). Sistema de Limitación actual RELEB(rojo), y READAT (Negro)

A partir de los datos obtenidos se calcularon la ganancia promedio 12.3 y la ganancia relativa G_r de cada sistema mediante la ecuación 12.4.

$$G_m = \frac{1}{n}\sum_{i=1}^{n}(SL_i - RELEB_i) \tag{12.3}$$

Donde SL_i son los margenes al DNB otorgados por cada Sistema de Limitación para el estado de planta i obtenido con cada método y $RELEB$ son los brindados por el sistema de limitación actual de la planta para el estado i y n es el total de estados de planta.

$$G_{ri} = \left(\frac{\frac{1}{n}\sum_{i=1}^{n}(SL_i - RELEB_i)}{\frac{1}{n}\sum_{i=1}^{n}(READAT_i - RELEB_i)} \right) \cdot 100 \tag{12.4}$$

Donde $READAT$ son los margenes al DNB obtenidos por la READAT para cada estado i mediante el programa AXCNA2.

Con este parámetro se puede analizar qué tan efectivo es cada Sistema de Limitación obtenido con las distintas técnicas con respecto a las ganancias en margen que se pueden obtener, teniendo como límite superior el Margen al DNB obtenido con la READAT. En el caso ideal donde el Sistema de Limitación otorgue los mismos valores que la READAT, este parámetro sería del 100%, caso contrario, si otorgara valores iguales al RELEB, este parámetro seria 0%, y se tornaría negativo en el caso en que los valores fuesen menores a los otorgados por el RELEB.

Sistema de Limitación	Ganancia [Margen]	Ganancia Relativa
Algoritmo de Retropropagación del Error	5,6%	56%
Recocido Simulado	4,4%	44%

Tabla 12.3: Ganancias obtenidas a partir del análisis de los distintos Sistemas de Limitación obtenidos a partir de la ANN y su diferencia con el Sistema de Limitación actual RELEB.

Para este sistema, la técnica de EBPA es la que mejor cumple con el objetivo, ya que el Sistema de Limitación obtenido es más preciso, su σ es menor, y otorga una ganancia relativa del 56%.

12.1.2 Sistema de señales de 90 detectores

Para el sistema de patrones que contiene las 90 señales de los detectores, se realizó un estudio análogo al realizado para el sistema de Lanzas Virtuales.

Algoritmo de Retropropagación del Error

De lo estudiado con el sistema de Lanzas Virtuales, se puede concluir que las redes que resuelven con un nivel de precisión adecuado este sistema son aquellas que poseen una estructura de 4 capas.

A partir de estos antecedentes se realizaron evaluaciones para estructuras con esta forma, las cuales son presentadas en la Tabla §12.4.

Estas estructuras se entrenaron con 3682 patrones seleccionados aleatoriamente de un total de 5260 patrones con el método de EBPA. El resto de los patrones se establecieron como testigos, y no fueron incluidos durante el entrenamiento.

Se evaluaron todos los patrones, tanto de entrenamiento como testigos con los pesos obtenidos en la época de entrenamiento número 100000 de todas las estructuras y se calculó el Error Cuadrático Medio (ECM) a partir de la ecuación 12.1.

Caso	# de capas	estructura	ECM
1	4	[90 18 3 1]	$7{,}8\times10^{-4}$
2	4	[90 15 15 1]	$4{,}12\times10^{-4}$
3	4	[90 30 15 1]	$2{,}06\times10^{-4}$
4	4	[90 90 30 1]	$2{,}6\times10^{-4}$
5	4	[90 90 45 1]	$1{,}6\times10^{-4}$

Tabla 12.4: Distintas estructuras entrenadas con 3682 patrones y sus parámetros.

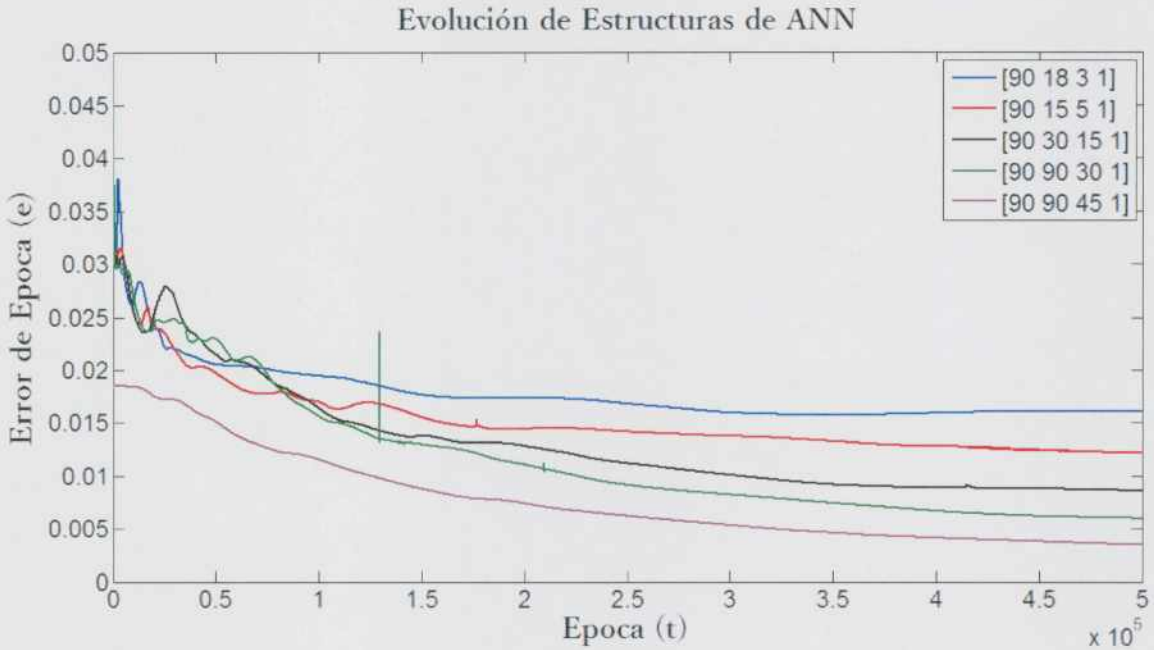

Figura 12.8: Evolución del error de época, obtenido en el entrenamiento de distintas estructuras de perceptrón multicapa con 3682 patrones

En la Figura §12.8 se grafican las evoluciones del error promedio de época *e* del conjunto de entrenamiento en función de la época para las distintas estructuras evaluadas.

A partir de estos datos se puede ver que la estructura que mejor ha evolucionado y que tiene un menor valor de ECM es la número 5, que cuenta con una estructura de 90, 90, 45 y 1 neuronas en cada una de las 4 capas.

En la Figura §12.9 se presenta la evolución del Error de Época (*e*) tanto para los patrones que fueron parte del Entrenamiento como de aquellos que no lo fueron. Se puede ver que en el caso de los testigos el *e* llega a un límite en el cual ya no disminuye su valor, difiriendo del comportamiento de los patrones de entrenamiento los cuales a medida que se incrementa (t), disminuye progresivamente su valor.

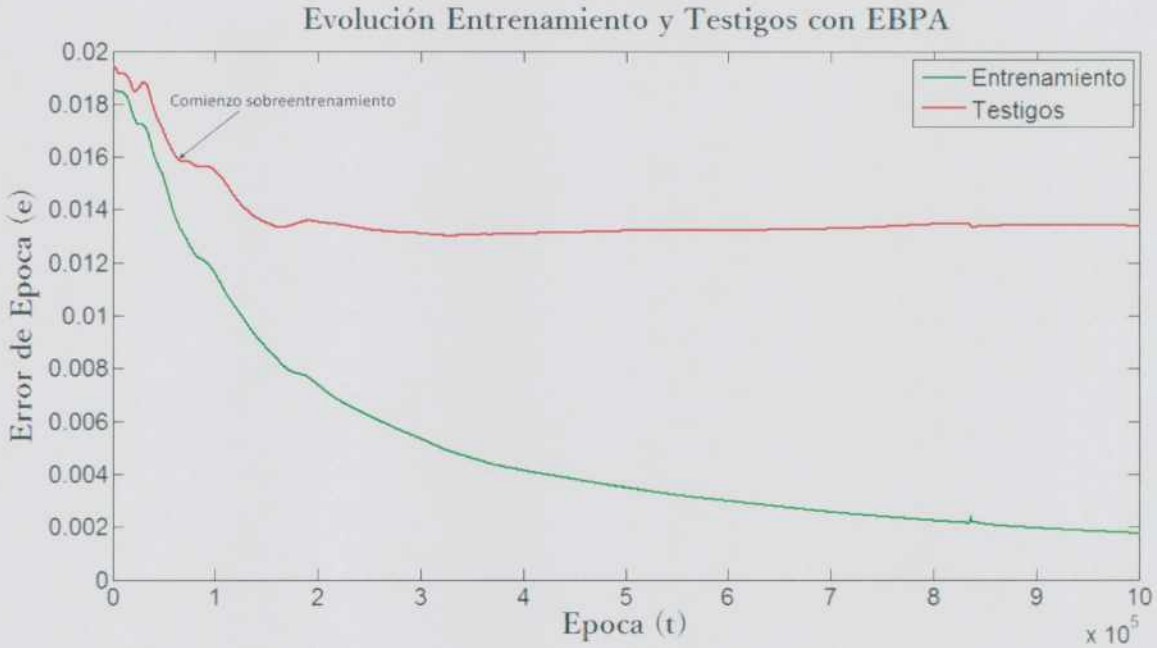

Figura 12.9: Evolución del error de época para la estructura [90 90 45 1], para el conjunto de entrenamiento con 3682 patrones (verde), y testigos con 1578 patrones (rojo).

Esto se debe a que la ANN al poseer tantas neuronas tiene una enorme capacidad de adaptarse a los patrones, y termina sobreentrenándose a los que se le presentaron durante su entrenamiento. Esto tiene como desventaja que a partir de cierto límite de entrenamiento pierde su capacidad predictiva, otorgando un valor alejado del verdadero al evaluar un conjunto de entradas no entrenado.

Para obtener un Sistema de Limitación confiable en sus predicciones, se utilizaron los pesos sinápticos obtenidos del entrenamiento de esta estructura de ANN que es la más capaz de resolver el sistema, pero en una instancia antes del punto desde el cual comienza el sobreentrenamiento sobre los patrones de entrenamiento.

Dicho en otras palabras se tomaron los pesos sinápticos obtenidos para $t = 60000$, donde se ve el cambio de pendiente en la curva de testigos, lo cual es el primer indicio de sobreentrenamiento de la ANN. Si bien luego de este cambio de pendiente, la curva sigue disminuyendo en épocas ulteriores, se toma el primer límite porque los valores de e son mayores, y otorgarán mayores valores de ECM y por ende de σ, lo cual es coherente con el enfoque conservador que se pretende para la obtención de un Sistema de Limitación dada por la ecuación 12.2, y con lo cual sus predicciones serán más confiables.

Las salidas obtenidas con los pesos sinápticos de $t = 60000$ se presentan en la Figura §12.10.

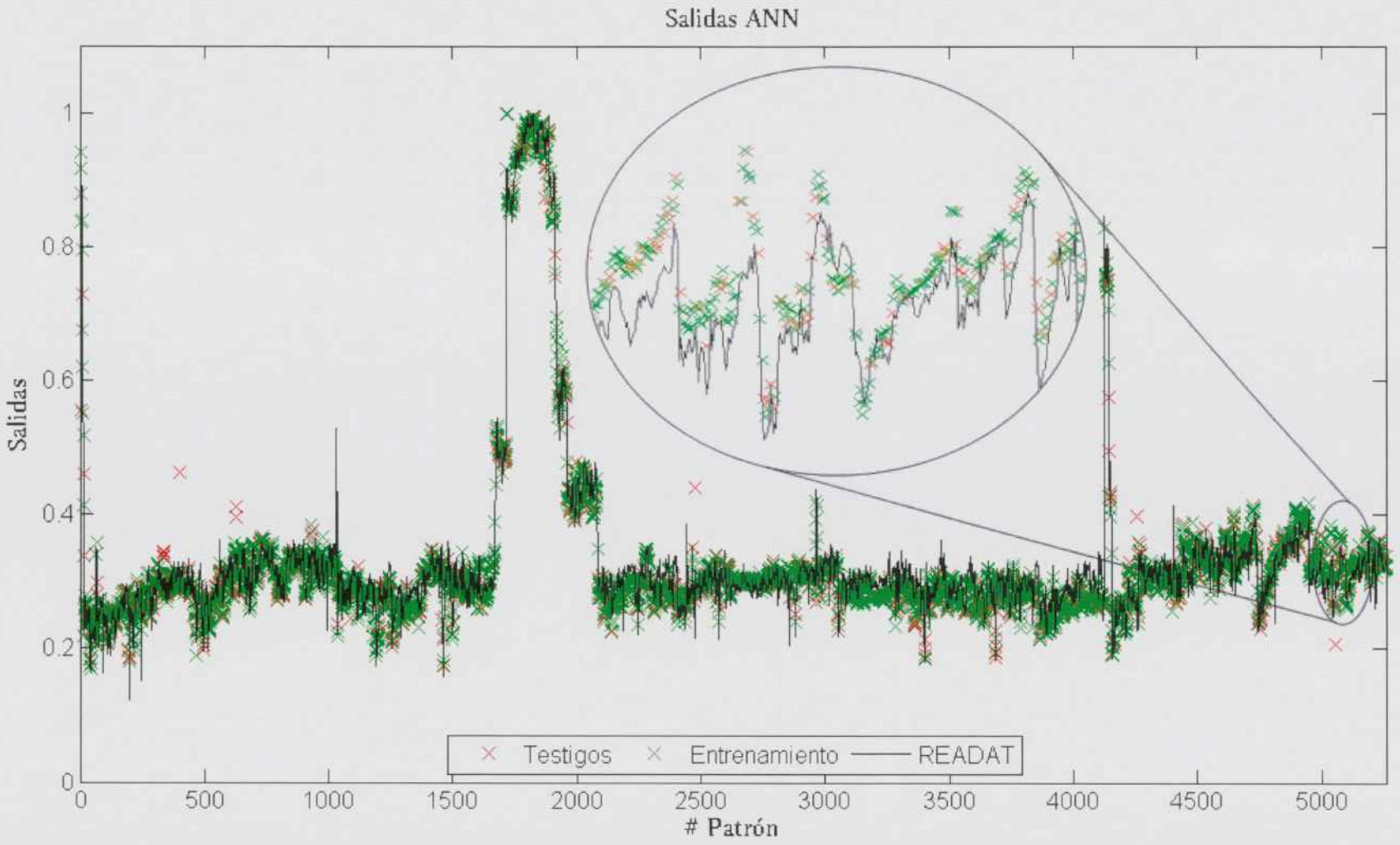

Figura 12.10: Salidas esperadas, Margen al DNB obtenido por la READAT dividido en 50 (negro) y salidas obtenidas por la Red Neuronal Artificial (Salidas ANN), para testigos (rojo) y para entrenamiento (verde)

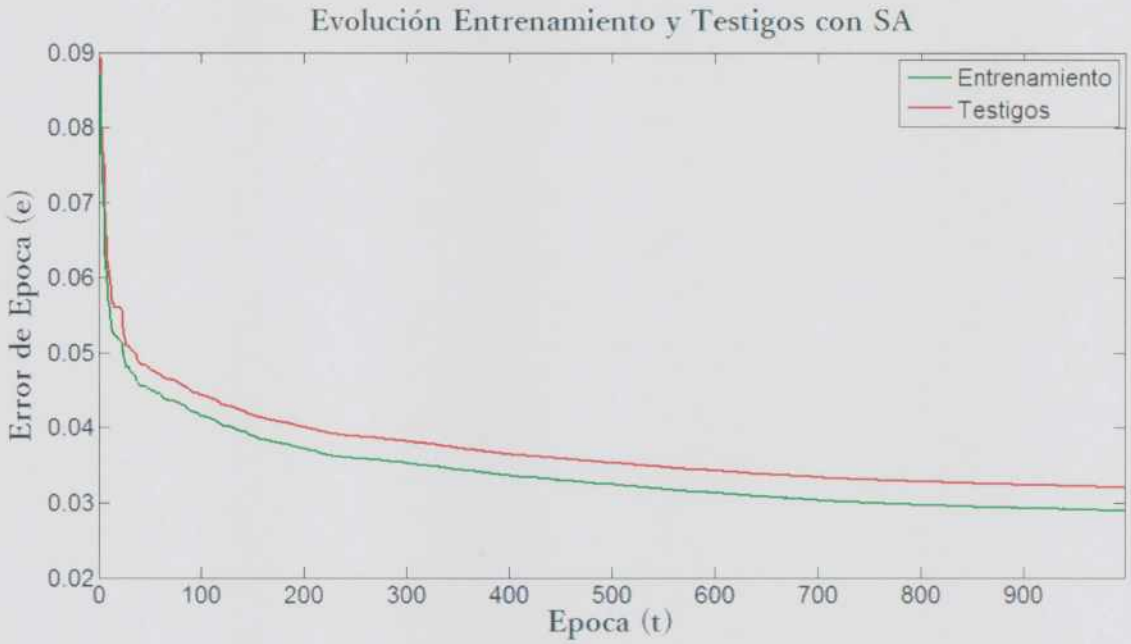

Figura 12.11: Evolución del error de época, para el conjunto de entrenamiento con 3682 patrones (verde), y testigos con 1578 patrones (rojo).

Recocido Simulado

Con la estructura 5 de la Tabla §12.4, que demostró tener un mejor desempeño con el algoritmo EBPA, se entrenaron los mismos 3682 patrones respectivos con el método de SA a partir de un estado inicial de la red con pesos aleatorios.

Se obtuvo la evolución del error promedio en función de las épocas presentada en la Figura §12.11. Se puede ver que para la cantidad de épocas seleccionadas $t = 1000$, los valores de e no llegan a disminuir tanto en comparación con los valores obtenidos con el método EBPA. Hasta este punto se puede ver que ambos disminuyen con el mismo ritmo tanto para los patrones del entrenamiento como con los patrones de testigos.

Las salidas obtenidas para la época $t = 1000$ se presentan en la Figura §12.12

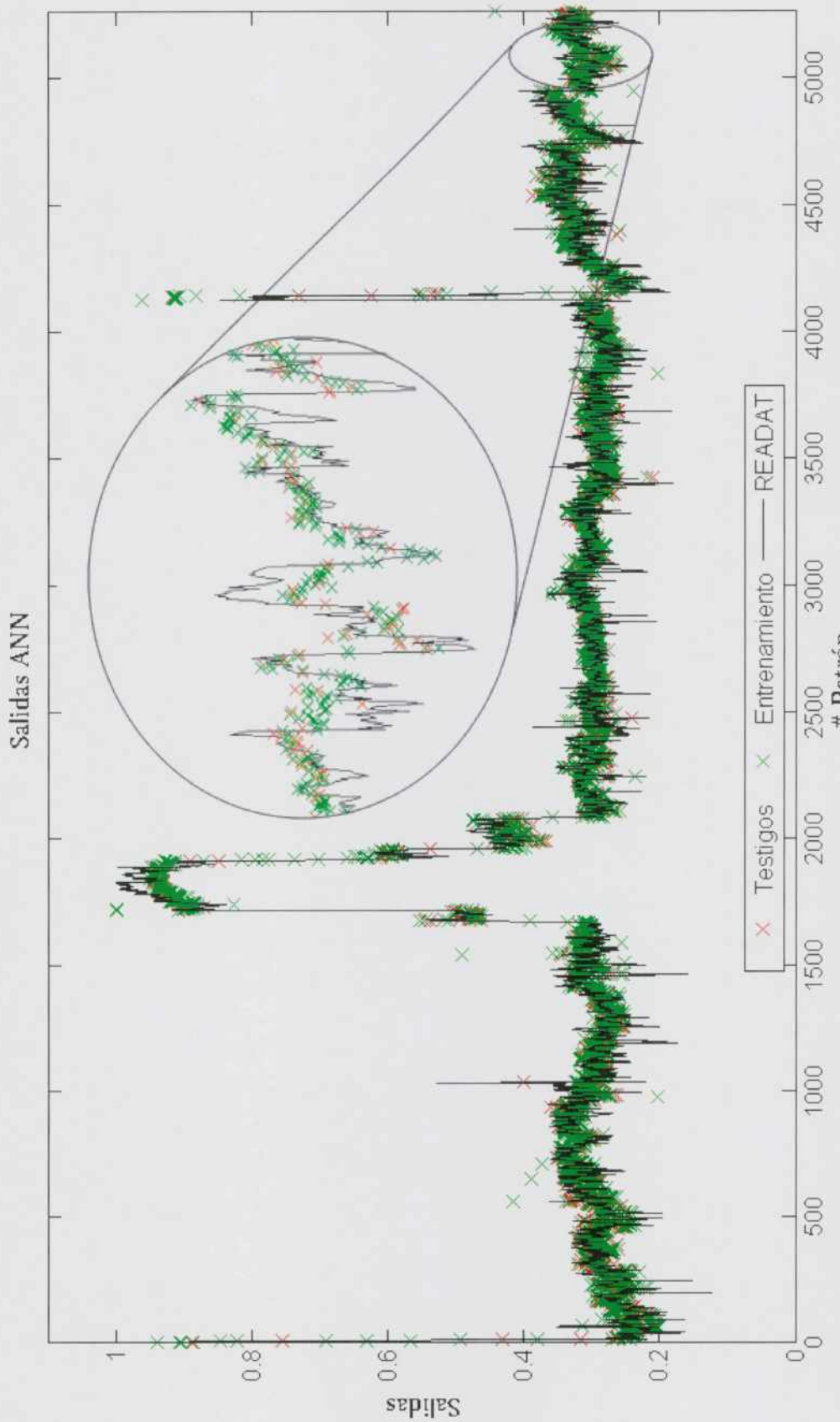

Figura 12.12: Salidas esperadas, Margen al DNB obtenido por la READAT dividido en 50 (negro) y salidas obtenidas por la Red Neuronal Artificial (Salidas ANN) utilizando el método SA, para testigos (rojo) y para entrenamiento (verde)

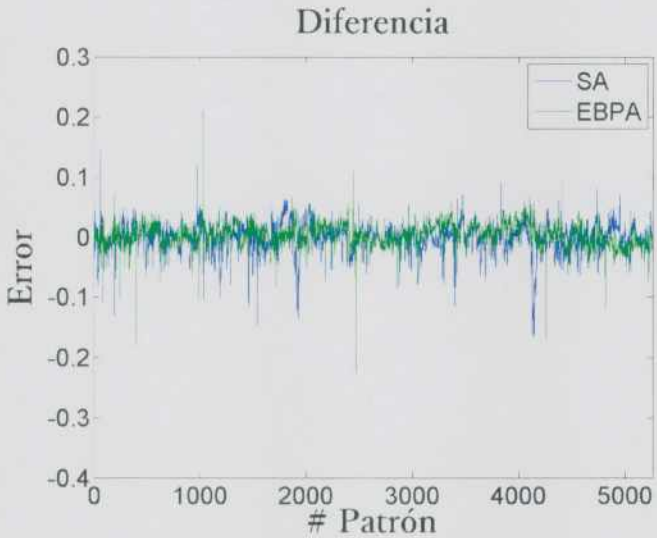

Figura 12.13: Diferencias entre las salidas esperadas y las salidas obtenidas con la técnica EBPA (verde), y las Salidas esperadas y las salidas obtenidas con la técnica de SA (azul).

Se calcularon las diferencias entre la salida deseada y la obtenida por cada técnica en la época $t = 60000$ para EBPA y $t = 1000$ para SA, las cuales se presentan en la Figura §12.13 . Se obtuvo el ECM, μ y σ, cuyos valores se presentan en la Tabla §12.5 De los valores obtenidos, se puede ver que la ANN que mejor se adapta sigue siendo la obtenida por EBPA, de todas maneras, los parámetros obtenidos con la técnica de SA, se encuentran en el mismo orden de magnitud, siendo mayores en menos del 50% respecto de los anteriores.

Técnica	ECM	μ	σ
Algoritmo de Retropropagación del Error	$3{,}8\times10^{-4}$	0,004	0,019
Recocido Simulado	$6{,}4\times10^{-4}$	-0,002	0,025

Tabla 12.5: Parámetros obtenidos del análisis de las diferencias entre las salidas deseadas y las salidas otorgadas por la ANN luego de ser entrenadas por EBPA y SA.

Sistema de Limitación

A partir de la ecuación 12.2 se calcularon los valores del Sistema de Limitación para las salidas obtenidas en ambas técnicas utilizando el mismo criterio que se utilizó con el conjunto de patrones de Lanzas Virtuales. Se graficaron los Sistemas de Limitación obtenidos con ambas técnicas en la Figura §12.14, junto con el Sistema de Limitación RELEB actual, y los valores de READAT.

A partir de los datos de los Sistemas de Limitación y de la READAT se obtuvieron las ganancias promedio y relativas mediante las ecuaciones 12.3 y 12.4, que se presentan en la Tabla §12.6

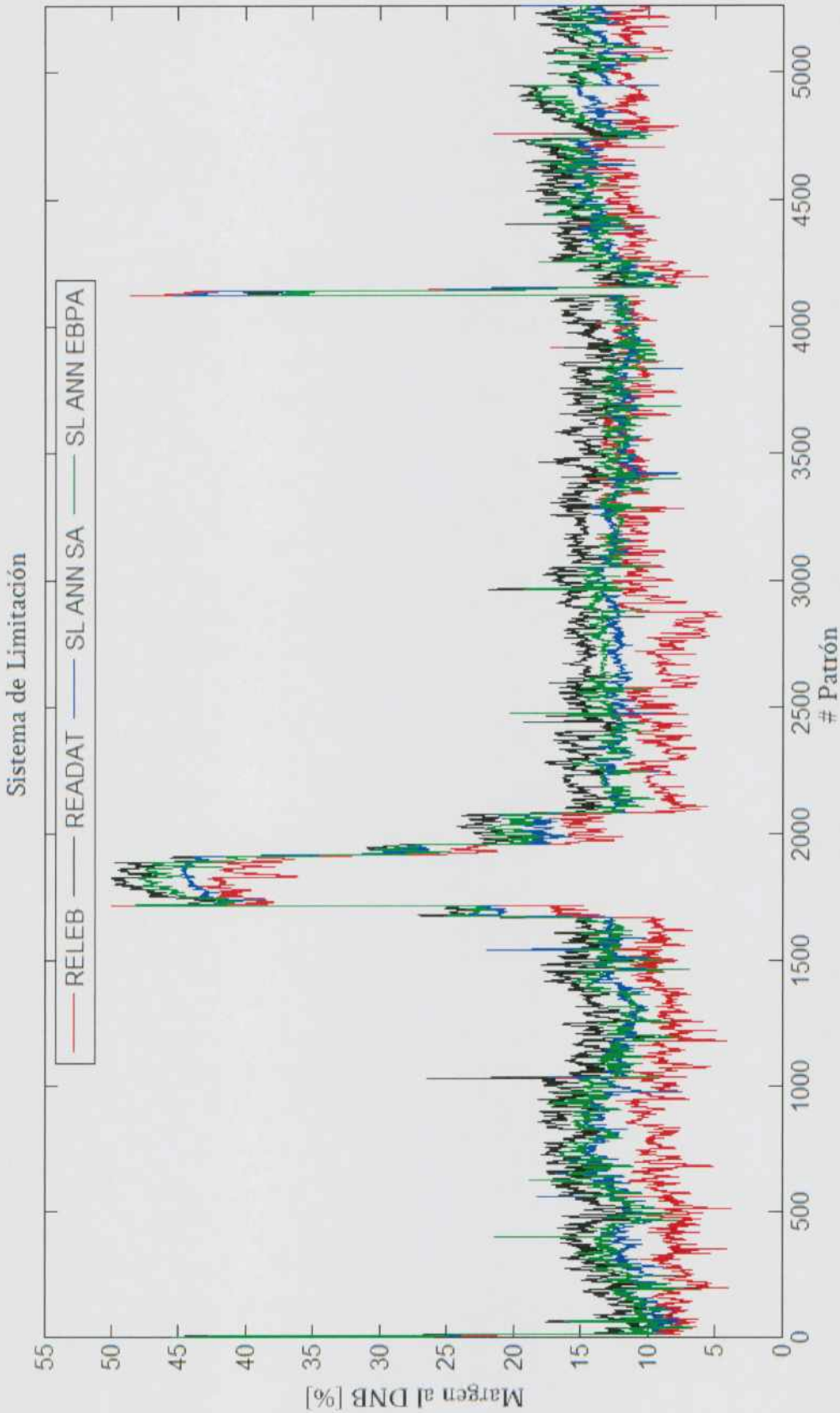

Figura 12.14: Margen al DNB obtenido con el Sistema de Limitación por ANN utilizando la técnica de EBPA (verde) y la técnica SA (azul). Sistema de Limitación actual RELEB (rojo), y READAT (Negro)

Sistema de Limitación	Ganancia [Margen]	Ganancia Relativa
Algoritmo de Retropropagación del Error	3,05%	30,5%
Recocido Simulado	2,46%	24,6%

Tabla 12.6: Ganancias obtenidas a partir del análisis de los distintos Sistemas de Limitación obtenidos a partir de la ANN y su diferencia con el Sistema de Limitación actual RELEB.

De los parámetros obtenidos en la Tabla §12.6 se puede concluir que con los valores de épocas configurados para ambas técnicas, el Sistema de Limitación por ANN obtenido con EBPA es el que mejor cumple con los objetivos propuestos en este sistema.

12.2 Etapa 2: Estados Simulados de Planta

Para los dos sistemas estudiados, Lanzas Virtuales y 90 detectores se agregaron patrones, simulados al 80, 90 y 110% de la potencia de los estados de planta utilizados como patrones en la etapa anterior. Se entrenaron las ANN, cuya estructura fue la que mejor resolvió el sistema en la Etapa anterior de estudio, teniendo como pesos iniciales, los óptimos obtenidos en el entrenamiento de la Etapa anterior en cada caso.

RELEB simulado

Se realizó el cálculo de los valores obtenidos con el RELEB en estos estados simulados, con un criterio conservativo. Esto implica considerar que los parámetros como Presión, Temperatura y Revolución de la Bomba se encuentran en el valor nominal. De las ecuaciones 5.1 5.2 y 5.3 se puede ver que al tener en cuenta estas consideraciones el término CKM toma un valor nulo. De esta forma se consideraron sólo las demás constantes, cuyos valores fueron tomados de los datos históricos coincidiendo en las fechas de las cuales fueron extraidos los patrones de la Etapa 1. Con estas constantes, y la información de las lanzas virtuales se pudo calcular los valores de RELEB para los estados simulados de planta.

12.2.1 Sistema de lanzas Virtuales

El número total de patrones es de 20540 ya que se descartaron aquellos en los cuales el margen al DNB obtenido por la READAT es menor al 50%, con el mismo criterio utilizado en la Etapa anterior de descartar los estados de planta que son de poco interés para la finalidad del análisis. Del total, los primeros 5260 patrones se

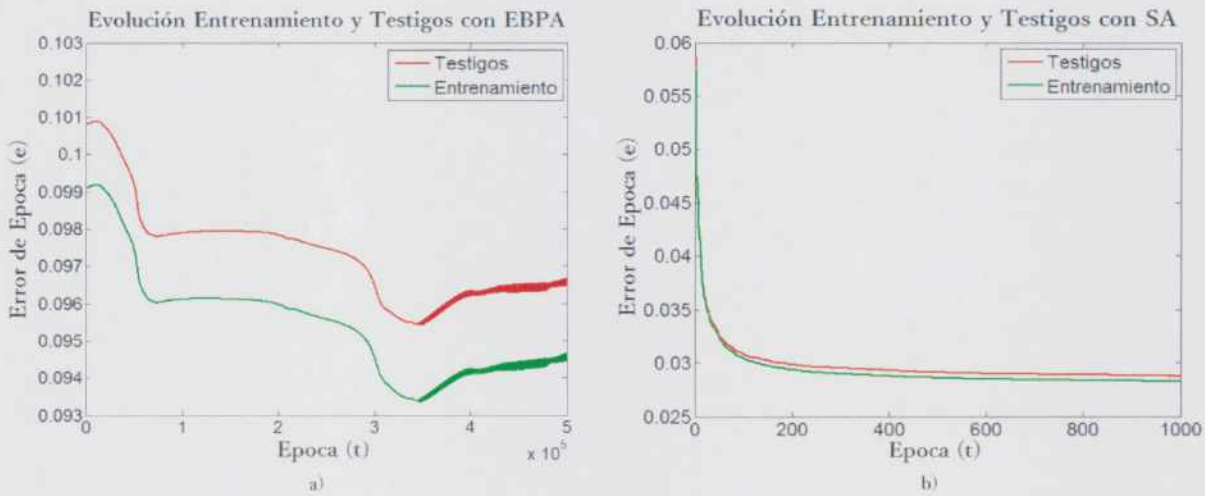

Figura 12.15: Evolución del Error de Época en los patrones de Entrenamiento (Verde) y Testigos (rojo) con los métodos: a) Recocido Simulado y b) Algoritmo de Retropropagación

corresponden a los de la Etapa 1. Se entrenaron el 70% de los casos. Se partió los pesos sinápticos obtenidos en la época número diez millones del entrenamiento con la técnica EBPA (la evolución que otorgó mejores parámetros de salida) de la etapa anterior para la estructura 8 de 4 capas con [18 18 12 1].

Se implementaron las dos técnicas y la evolución del error promedio e en función de las épocas se presenta en los gráficos de la Figura §12.15, en los cuales se puede ver que, en éste caso, el método SA obtiene un valor de e considerablemente menor que el método de EBPA, en valores de t de casi dos ordenes de magnitud menores. Cabe destacar que a pesar de la gran cantidad de épocas de entrenamiento utilizada con EBPA, se mejora muy poco el valor de e.

Con los pesos sinápticos de la época del entrenamiento en donde el valor de e es el mínimo alcanzado, se evaluaron tanto los patrones del conjunto de entrenamiento como para los patrones que quedaron como testigos y no fueron entrenados y se obtuvieron las salidas que se presentan en las Figuras §12.16 y §12.17. En estos gráficos se puede ver que la ANN utilizada no puede resolver efectivamente el sistema con la técnica de EBPA, es decir con esta técnica no se llega a efectivamente encontrar los pesos sinápticos de la ANN para que la misma otorgue las salidas que uno esperaría.

Esto puede ser posible debido a que al comenzar con los pesos obtenidos en la Etapa 1, la ANN estuviese sobreadaptada a este tipo de patrones, y se haya estancado en los alrededores de un mínimo local de e. Con el método de SA es posible sortear esta dificultad ya que el algoritmo es capaz de escapar de las inmediaciones de un mínimo

local por permitirle aceptar los casos donde las variaciones empeoren el *e* obtenido.

De todas maneras, para poder concluir que esta es la causa, habría que entrenar a la ANN con EBPA desde pesos sinápticos aleatorios para evaluar si la evolución efectivamente es diferente de la obtenida con EBPA a partir de los pesos sinápticos otorgados en la Fase 1, y analizar los resultados, lo cual no fue realizado en este PFI.

Con la técnica de SA se puede ver un mejor rendimiento de la ANN al evaluar los patrones del conjunto de entrenamiento, sin embargo es posible notar que las salidas obtenidas al evaluar los testigos se alejan de lo esperado marcadamente a lo largo de todo el conjunto de patrones.

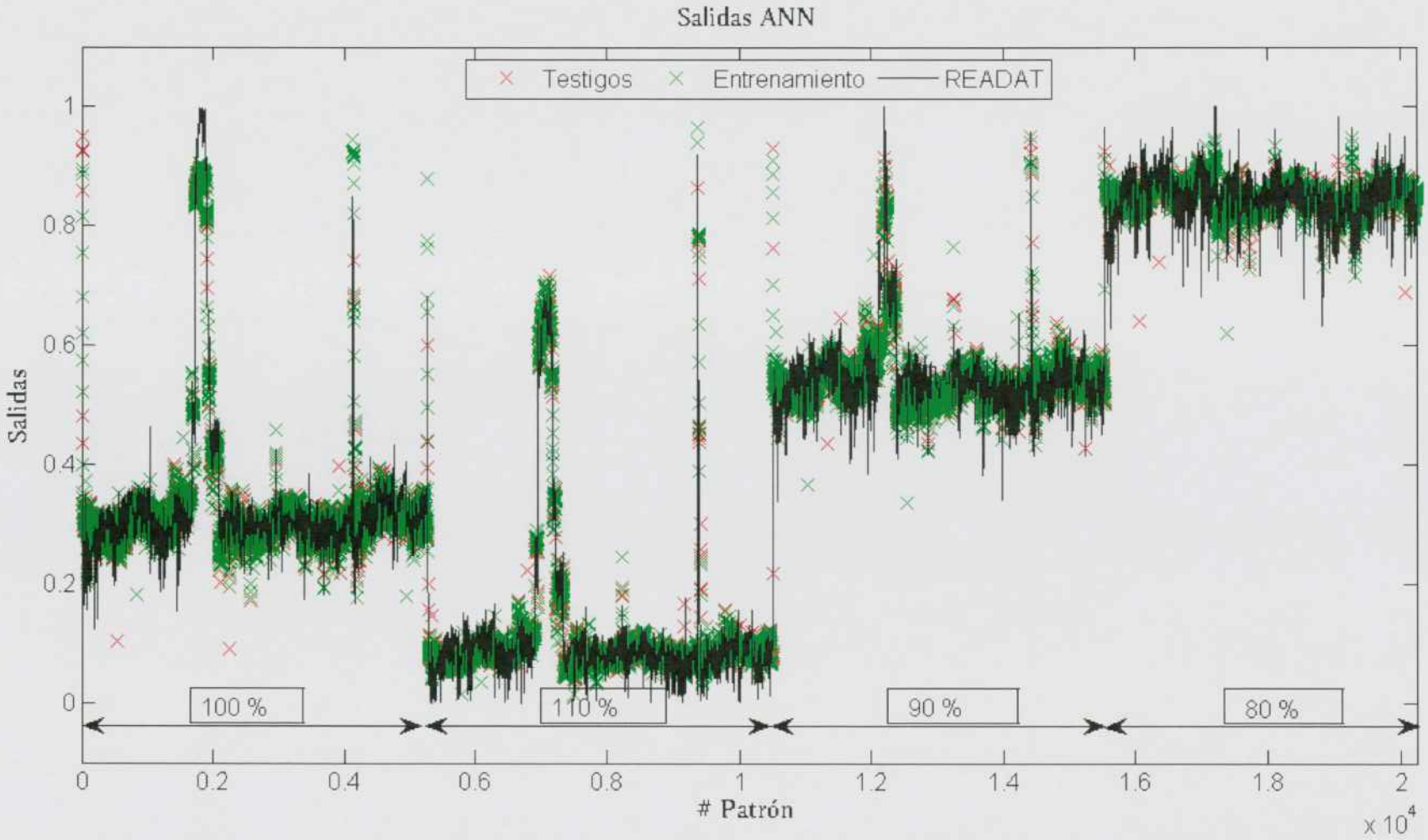

Figura 12.16: Salidas deseadas, Margen al DNB obtenido por la READAT dividido en 50 (negro) y salidas obtenidas por la Red Neuronal Artificial con el método de SA en la época $t = 1000$ (Salidas ANN), para testigos (rojo) y para entrenamiento (verde).

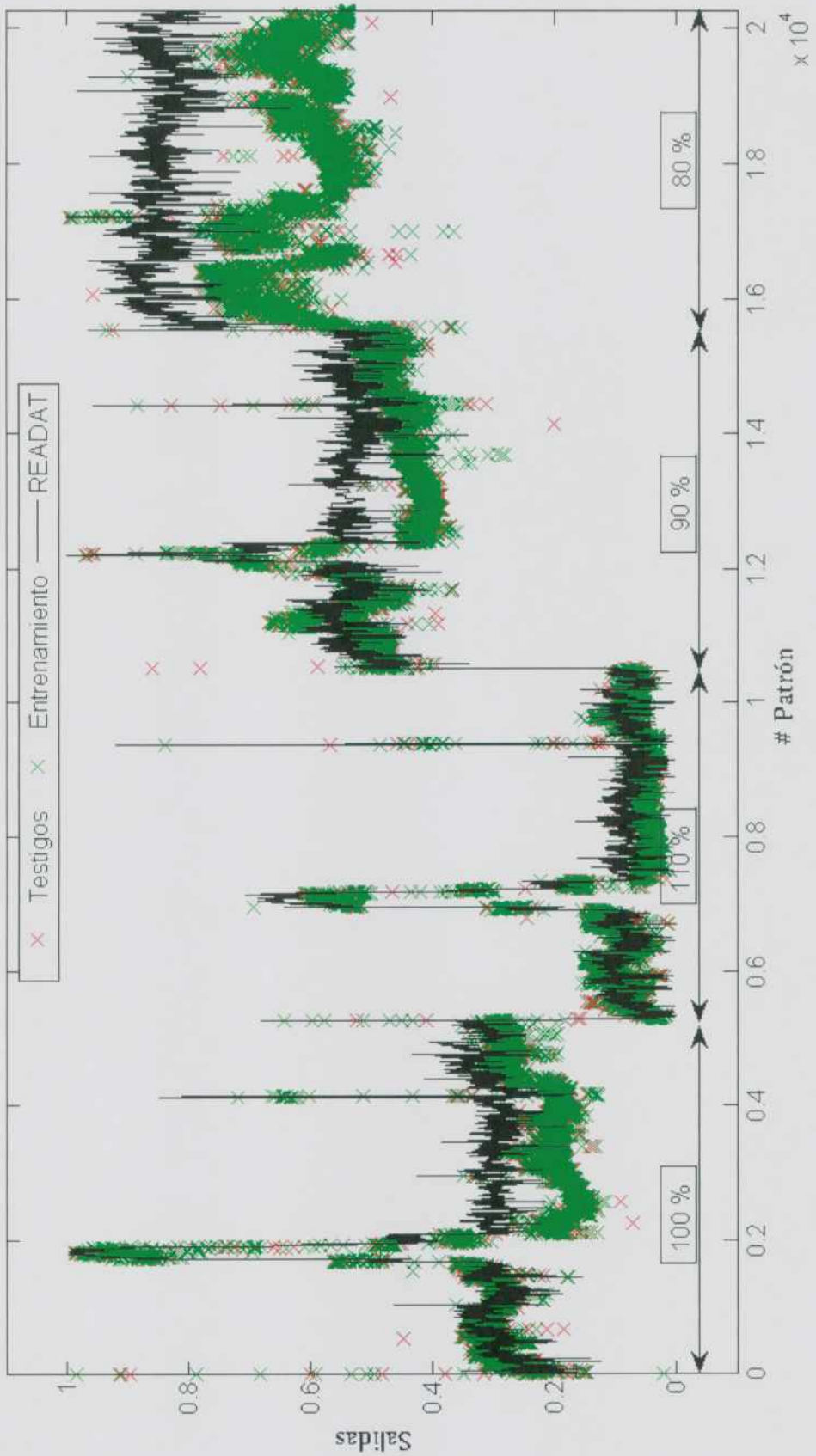

Figura 12.17: Salidas deseadas, Margen al DNB obtenido por la READAT dividido en 50 (negro) y salidas obtenidas por la Red Neuronal Artificial con el método de EBPA en la época $t = 340000$ (Salidas ANN), para testigos (rojo) y para entrenamiento (verde).

Luego, las diferencias obtenidas entre las salidas deseadas y las obtenidas con ambos métodos son presentadas en la Figura §12.18. Con estos datos se calcularon los parámetros presentados en la Tabla §12.7.

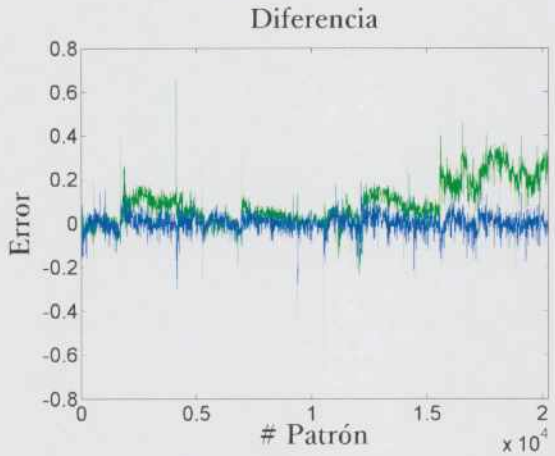

Figura 12.18: Diferencias entre las salidas esperadas y las salidas obtenidas con la técnica EBPA (verde), y las salidas esperadas y las salidas obtenidas con la técnica de SA (azul)

A partir de estos parámetros y de la Figura §12.17 se puede ver que la ANN obtenida con EBPA no logra adaptarse a las salidas cómo lo ha venido haciendo hasta ahora con los sistemas de la Etapa anterior. Aunque esta estructura ha demostrado su capacidad de adaptarse a los valores esperados con las 18 entradas otorgadas, en este caso sufre ciertas limitaciones con este método de entrenamiento por lo que no resuelve los pesos óptimos para que el e disminuya de la forma que el método de SA si lo puede realizar, llegando a un valor de ECM casi diez veces menor que el método EBPA.

Técnica	ECM	μ	σ
Algoritmo de Retropropagación del Error	0,016	0,09	0,13
Recocido Simulado	0,002	0,004	0,045

Tabla 12.7: Parámetros obtenidos del análisis de las diferencias entre las salidas deseadas y las salidas otorgadas por la ANN luego de ser entrenadas por EBPA y SA.

Sistema de Limitación por ANN

Nuevamente se calculó el Sistema de Limitación para ambos casos, utilizando la ecuación 12.2, y se presentan en la Figura §12.19 junto con el margen al DNB calculado por la READAT y los valores obtenidos por RELEB para los estados simulados.

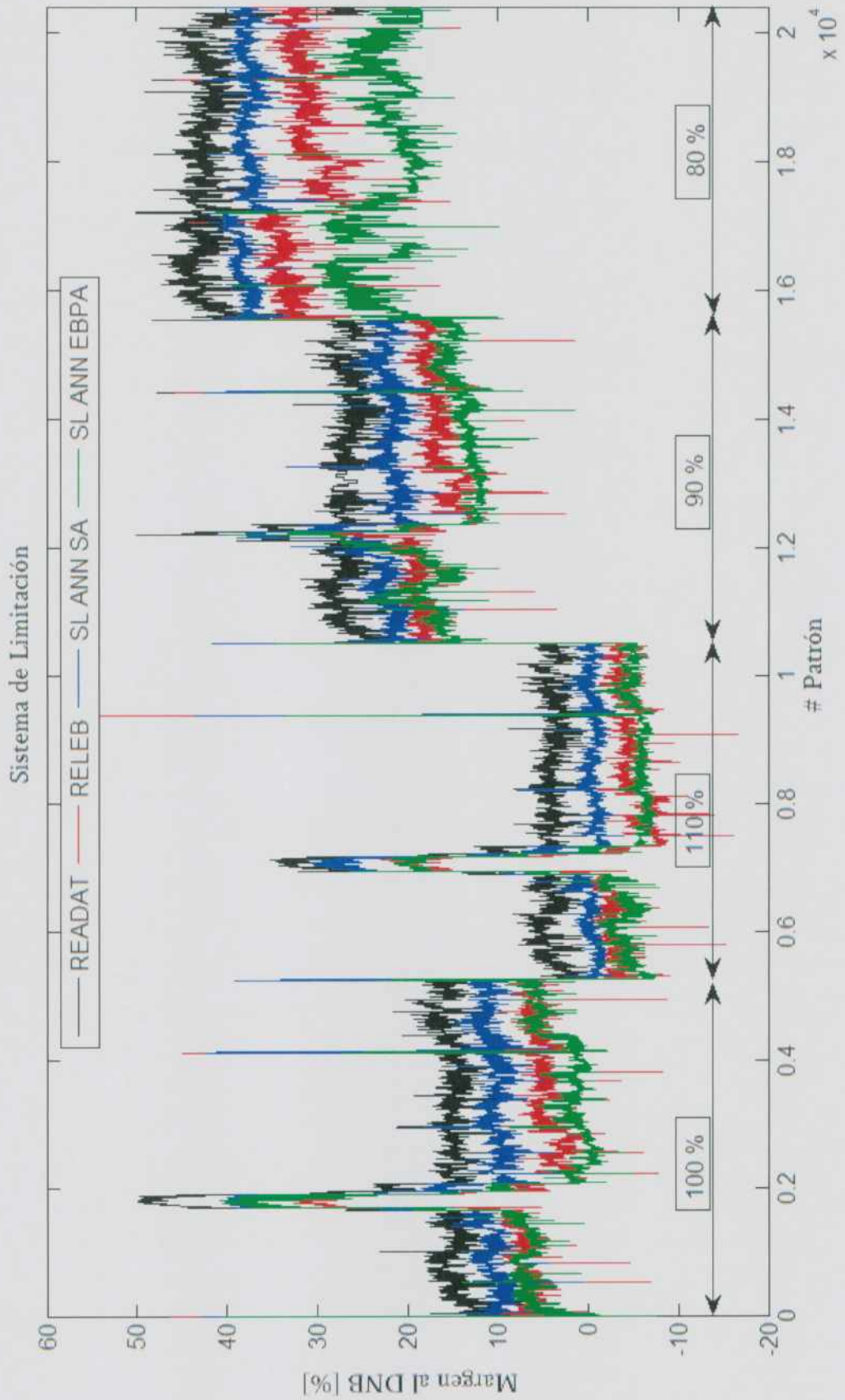

Figura 12.19: Margen al DNB obtenido con READAT (negro), Sistema de Limitación por ANN con SA (azul), Sistema de Limitación por ANN con EBPA (verde), RELEB (rojo), en función del número de patrón.

Las ganancias promedio G_m y ganancia relativa G_r obtenidas con ambos sistemas de limitación se calcularon utilizando las ecuaciones 12.3 y 12.4 y se presentan en la Tabla §12.8.

Sistema de Limitación	Ganancia [Margen]	Ganancia Relativa
Algoritmo de Retropropagación del Error	-3,07%	-31,7%
Recocido Simulado	5,17%	53,5%

Tabla 12.8: Ganancias obtenidas a partir del análisis de los distintos Sistemas de Limitación obtenidos a partir de la ANN y su diferencia con el Sistema de Limitación actual RELEB.

Los valores obtenidos con la técnica de EBPA son negativos, es decir, los valores de margen al DNB otorgados a partir de las entradas son en promedio menores a los otorgados por el Sistema de Limitación actual. Esto es debido a las limitaciones que tiene este sistema en cuanto al entrenamiento de EBPA para llegar a los pesos que minimizan el valor de e. Una mala adaptación a las salidas, sobre todo con desvíos por defecto, dando un valor de μ negativo, más el efecto de tener un valor de σ grande, otorgan un Sistema de Limitación por ANN peor que el Sistema de Limitación actual en cuanto a la predicción de margen al DNB a partir de las entradas dadas

En este caso el Sistema de Limitación que mejor cumple con el objetivo es el obtenido con la técnica de SA, el cual es capaz de encontrar una configuración de ellos que hacen que la ANN se adapte a las salidas con mucha mayor precisión y otorgando un Sistema de Limitación más eficiente con una ganancia relativa del 53,5%.

12.2.2 Sistema con las señales de 90 detectores

Análogamente a lo realizado con el sistema de Lanzas Virtuales se utilizó la ANN de estructura de 90, 90, 45 y 1 neuronas en cada capa, ya que ésta fue la que tuvo mejor desempeño en la etapa anterior. Se realizó el entrenamiento a partir de los pesos sinápticos obtenidos en la época $t = 60000$ de la Etapa 1, mediante ambas técnicas con el 70% de los 20540 patrones, elegidos aleatoriamente. De manera similar a la etapa anterior, el 30% de los patrones remanentes quedaron como testigos y no fueron parte del entrenamiento.

La evolución del error promedio e de las épocas mediante la utilización de ambas técnicas en función de las épocas se presentan en la Figura §12.20. Se puede ver que la técnica de SA nuevamente es más efectiva que la de EBPA en cuanto a proporcionar

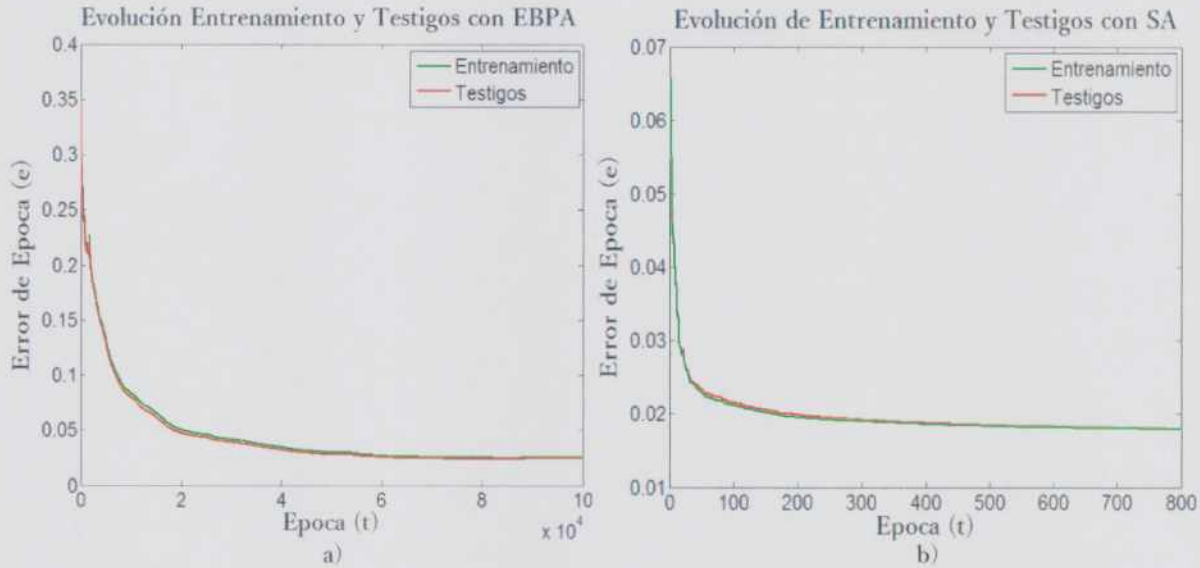

Figura 12.20: Evolución del Error de Epoca en los patrones de Entrenamiento (Verde) y Testigos (rojo) con los métodos: a) Algoritmo de Retropropagación del Error y b) Recocido Simulado

valores menores de *e* en una menor cantidad de épocas. De todas maneras esta evolución logra disminuir los valores de *e* con la técnica de EBPA de una forma mucho más efectiva que en el sistema de Lanzas Virtuales.

Las salidas obtenidas al evaluar los patrones de entrenamiento y testigos con los pesos sinápticos de la época *t* donde el valor de *e* es mínimo mediante la utilización de cada técnica se presentan en las Figura §12.21 y §12.22.

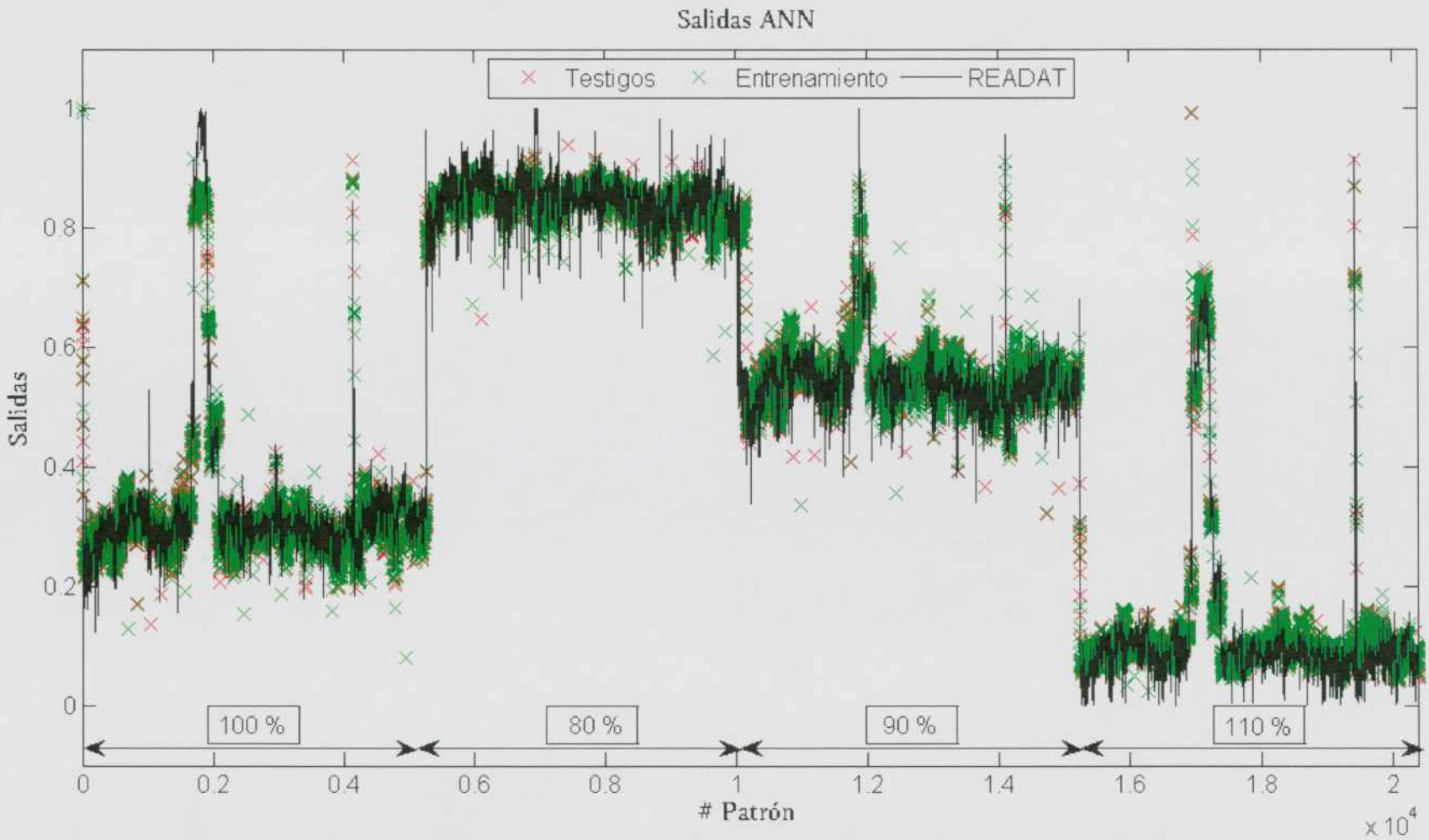

Figura 12.21: Salidas deseadas, Margen al DNB obtenido por la READAT dividido en 50 (negro) y salidas obtenidas por la Red Neuronal Artificial (Salidas ANN por SA), para testigos (rojo) y para entrenamiento (verde).

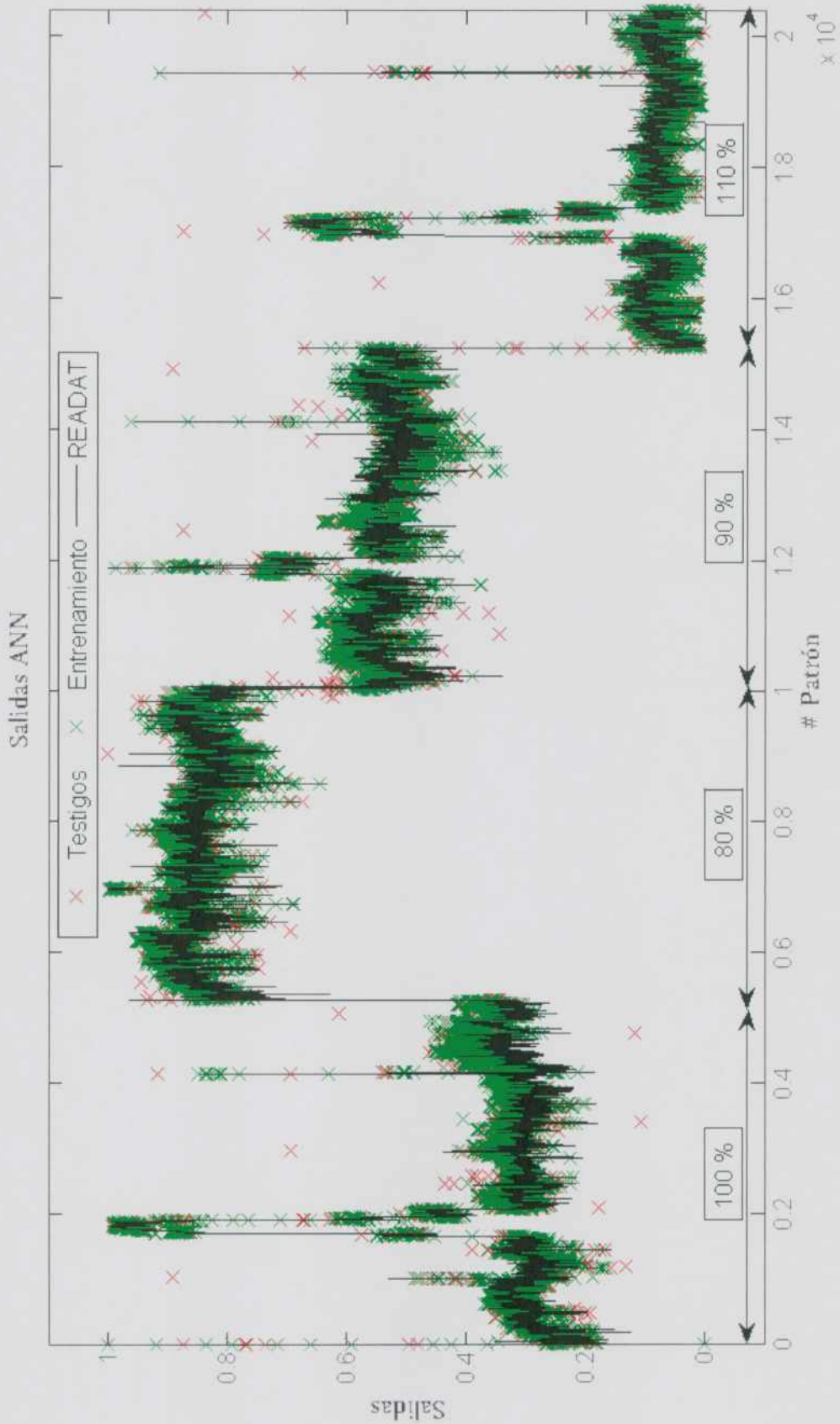

Figura 12.22: Salidas deseadas, Margen al DNB obtenido por la READAT dividido en 50 (negro) y salidas obtenidas por la Red Neuronal Artificial (Salidas ANN por EBPA), para testigos (rojo) y para entrenamiento (verde).

De las Figuras §12.21 y §12.22 puede verse que la ANN puede resolver efectivamente el sistema a distintas potencias con la información de las 90 entradas con ambas técnicas de entrenamiento.

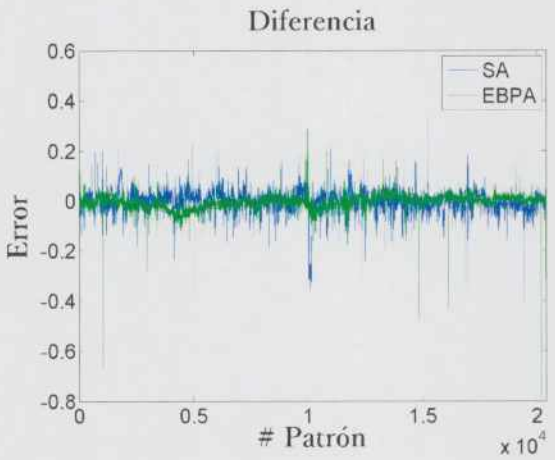

Figura 12.23: Diferencias entre las salidas esperadas y Las salidas obtenidas con la técnica EBPA (verde), y las salidas esperadas y Las salidas obtenidas con la técnica de SA (azul)

Las diferencias entre las salidas deseadas y las obtenidas por cada técnica se presentan en la Figura §12.23. Se calcularon a partir de estas diferencias el μ y el σ, así como el ECM con la ecuación 12.1 para cada técnica, parámetros que son presentados en la Tabla §12.9. A partir del análisis de estos datos se puede concluir que para un sistema que tome como entradas las 90 señales de los detectores, las salidas de la ANN se adaptan a lo esperado, por su valor de ECM bajo y su desviación también baja, alcanzando mejores resultados con la técnica de EBPA (con las épocas configuradas).

Técnica	ECM	μ	σ
Algoritmo de Retropropagación del Error	$8,1 \times 10^{-4}$	-0,005	0,02
Recocido Simulado	0,002	-0,003	0,044

Tabla 12.9: Parámetros obtenidos del análisis de las diferencias entre las salidas deseadas y las salidas otorgadas por la ANN luego de ser entrenadas por EBPA y SA.

Sistema de Limitación por ANN

A partir de la ecuación 12.2 se calculó el Sistema de Limitación por ANN (SLN) para los dos métodos y se presentan en la Figura §12.24 superpuestos a los datos de la READAT y los valores obtenidos por el Sistema de Limitación actual para estos estados.

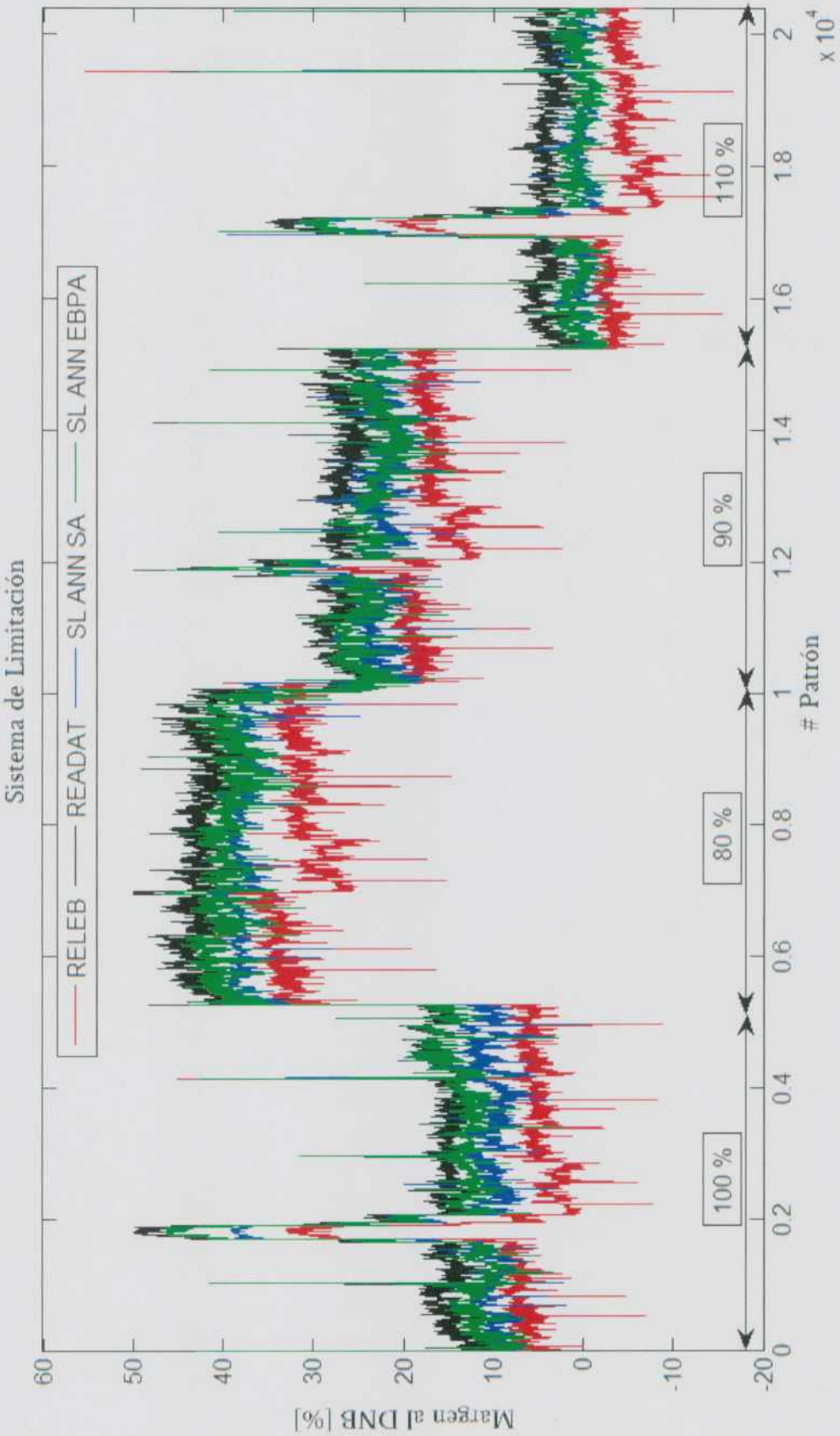

Figura 12.24: Margen al DNB obtenido con READAT (negro), Sistema de Limitación por ANN con SA (azul), Sistema de Limitación por ANN con EBPA (verde), RELEB (rojo), en función del número de patrón.

Se calcularon las ganancias promedio y ganancias relativas mediante las ecuaciones 12.3 y 12.4, cuyos valores se presentan en la Tabla §12.10.

Sistema de Limitación	Ganancia [Margen]	Ganancia Relativa
Algoritmo de Retropropagación del Error	6,83%	70,56%
Recocido Simulado	5,27%	54,4%

Tabla 12.10: Ganancias obtenidas a partir del análisis de los distintos Sistemas de Limitación obtenidos a partir de la ANN y su diferencia con el Sistema de Limitación actual RELEB.

El SLN obtenido mediante la técnica de entrenamiento de EBPA otorga una ganancia relativa del 70,56% de predicción del margen respecto del Sistema de Limitación actual de la planta. Esta es la mejor ganancia relativa obtenida hasta el momento.

Combinación de métodos

A partir de los pesos obtenidos del entrenamiento de EBPA en la época $t = 100000$ de la Etapa 2 para este sistema, se reentrenó la ANN con el método de SA en 1000 épocas adicionales para analizar si los resultados obtenidos otorgan un sistema aún más eficaz en cuanto a la ganancia de predicción en el margen al DNB respecto del sistema actual.

Se obtuvieron los parámetros presentados en la Tabla §12.11.

ECM	μ	σ	G_m (margen)	G_r
$7{,}5\times10^{-}4$	0,0003	0,0275	6,93%	71,61%

Tabla 12.11: Parámetros obtenidos del análisis de las diferencias entre las salidas deseadas y las salidas obtenidas luego de utilizar la combinación de las dos técnicas de entrenamiento, y las ganancias promedio y relativa otorgadas por el Sistema de Limitación creado a partir de las salidas obtenidas

Con lo cual se concluye que es posible encontrar una configuración de pesos sinápticos, tales que se disminuya aún más el valor de ECM obtenido con el entrenamiento con el método de EBPA, si se emplea SA a partir del punto en que el primero ya no sea capaz de hacerlo. De esta forma, se obtiene un Sistema de Limitación por ANN que supera los valores de ganancias obtenidos hasta el momento.

12.3 Etapa 3: Comportamiento ante Fallas

En esta Etapa se evaluó el comportamiento del sistema de 90 detectores con la ANN de estructura [90 90 45 1] ante desconexiones o amplificaciones de detectores. Los valores de los pesos sinápticos utilizados por la Red Neuronal Artificial son aquellos obtenidos en la Etapa 2 luego de utilizar la combinación de las dos técnicas de entrenamiento, EBPA y SA.

Este estudio no se realiza en el sistema de lanzas virtuales ya que, por cómo esta configurado el sistema, las entradas toman los valores máximos de potencia lineal del circuito en cada posición, reemplazando las desconexiones si es que hubiera. Sin embargo, ante una falla por amplificación subestiman el márgen.

Para el Sistema de Limitación existente en la planta esto se resuelve tomando el segundo máximo cuando la señal del máximo difiere más de un 15% del promedio de los detectores aledaños.

12.3.1 Desconexión de un único detector

Para estudiar la desconexión de un único detector se evaluó con la ANN especificada el conjunto de 5260 patrones utilizados en la Etapa 1, estableciendo en cada serie el valor de cada detector a cero. Se obtuvo el promedio en valor absoluto de las diferencias entre las salidas obtenidas por la ANN al evaluar los conjuntos de patrones sin y con la falla antes descripta. Los promedios se presentan en la Tabla §12.12.

De los datos presentados en la Tabla §12.12 se puede ver que el mayor promedio del error fue obtenido en el detector 5 de la lanza 13. Las salidas obtenidas por la ANN para ese detector se presentan en la Figura §12.25.

Se puede ver la sensibilidad de la ANN respecto de la variación de sus entradas. En la mayoría de los patrones otorga una salida con un valor que es mayor que el valor de salida esperada (Margen al DNB calculado por la READAT dividido en 50). Lo que demuestra la falta de robustez del sistema en este caso.

Lanza	Det	E_e	Lanza	Det	E_e	Lanza	Det	E_e
1	1	0,4236	6	1	0,2305	11	1	0,1914
1	2	0,216	6	2	0,3218	11	2	0,3009
1	3	0,2406	6	3	0,472	11	3	0,1423
1	4	0,1472	6	4	0,2558	11	4	0,1464
1	5	0,1207	6	5	0,2655	11	5	0,217
1	6	0,4205	6	6	0,1039	11	6	0,1065
2	1	0,1195	7	1	0,1796	12	1	0,1922
2	2	0,1501	7	2	0,1764	12	2	0,3904
2	3	0,1061	7	3	0,1807	12	3	0,1868
2	4	0,0888	7	4	0,2213	12	4	0,2057
2	5	0,085	7	5	0,199	12	5	0,3056
2	6	0,0765	7	6	0,1881	12	6	0,0926
3	1	0,3892	8	1	0,1631	13	1	0,1039
3	2	0,1795	8	2	0,1868	13	2	0,2925
3	3	0,285	8	3	0,3499	13	3	0,3412
3	4	0,2054	8	4	0,2399	13	4	0,4585
3	5	0,2364	8	5	0,2633	13	5	0,5056
3	6	0,2293	8	6	0,2335	13	6	0,4204
4	1	0,138	9	1	0,1765	14	1	0,244
4	2	0,143	9	2	0,2498	14	2	0,2727
4	3	0,2764	9	3	0,4226	14	3	0,2413
4	4	0,3678	9	4	0,4291	14	4	0,282
4	5	0,319	9	5	0,4327	14	5	0,3915
4	6	0,1708	9	6	0,1707	14	6	0,1771
5	1	0,2244	10	1	0,1083	15	1	0,1652
5	2	0,2849	10	2	0,1557	15	2	0,2996
5	3	0,2396	10	3	0,265	15	3	0,4557
5	4	0,2501	10	4	0,4167	15	4	0,2223
5	5	0,3322	10	5	0,1376	15	5	0,1703
5	6	0,1386	10	6	0,1153	15	6	0,2947

Tabla 12.12: Promedios de la diferencia entre la salidas obtenidas mediante la ANN evaluando los patrones sin fallas y las salidas obtenidas con un detector desconectado.

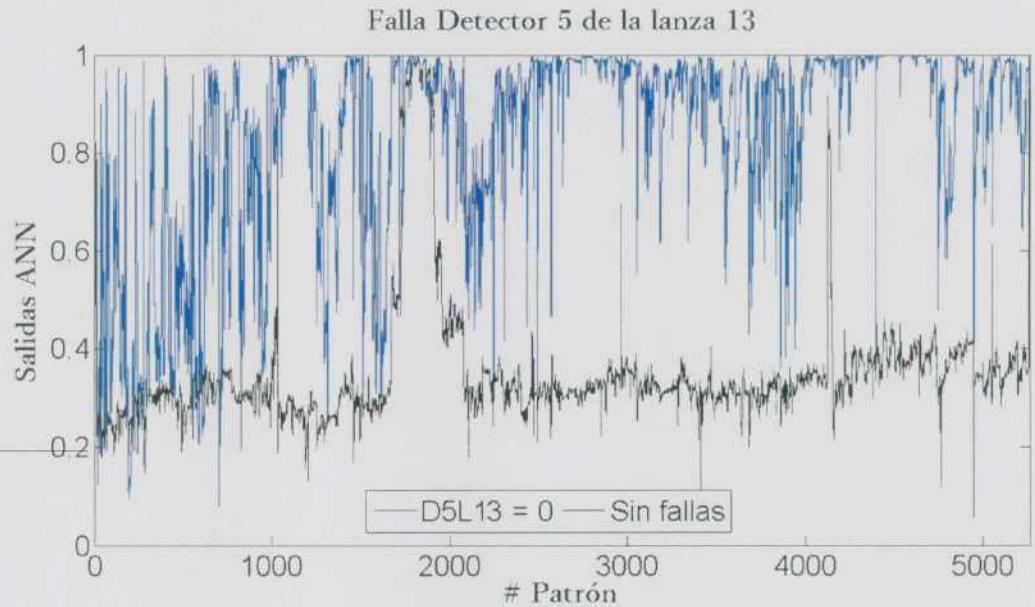

Figura 12.25: Salidas obtenidas con ANN al evaluar 5260 patrones con el detector 5 de la lanza 11 desconectado en todos los patrones.

12.3.2 Desconexión de una lanza

EL mismo conjunto fue evaluado con la ANN especificada estableciendo en 0 las seis entradas pertenecientes a cada lanza por serie. Los promedios de las diferencias entre las salidas obtenidas con la ANN al evaluar los patrones con y sin fallas se presentan en la Tabla §12.13.

Lanza	E_e	Lanza	E_e	Lanza	E_e	Lanza	E_e	Lanza	E_e
1	0,3505	4	0,2517	7	0,3113	10	0,4002	13	0,446
2	0,0827	5	0,2707	8	0,2775	11	0,3246	14	0,284
3	0,3717	6	0,2363	9	0,654	12	0,2565	15	0,23

Tabla 12.13: Promedios de la diferencia entre la salidas obtenidas mediante la ANN evaluando los patrones sin fallas y las salidas obtenidas con una lanza desconectada.

De la Tabla §12.13 se puede ver que la lanza con el promedio más alto en errores por época fue la número 9. Las salidas obtenidas para la ANN con esa lanza desconectada se presenta en la Figura §12.26.

Siguiendo en línea con lo obtenido para la desconexión de un sólo detector, la evaluación del sistema con una lanza desconectada nos otorga una salida mayor que las

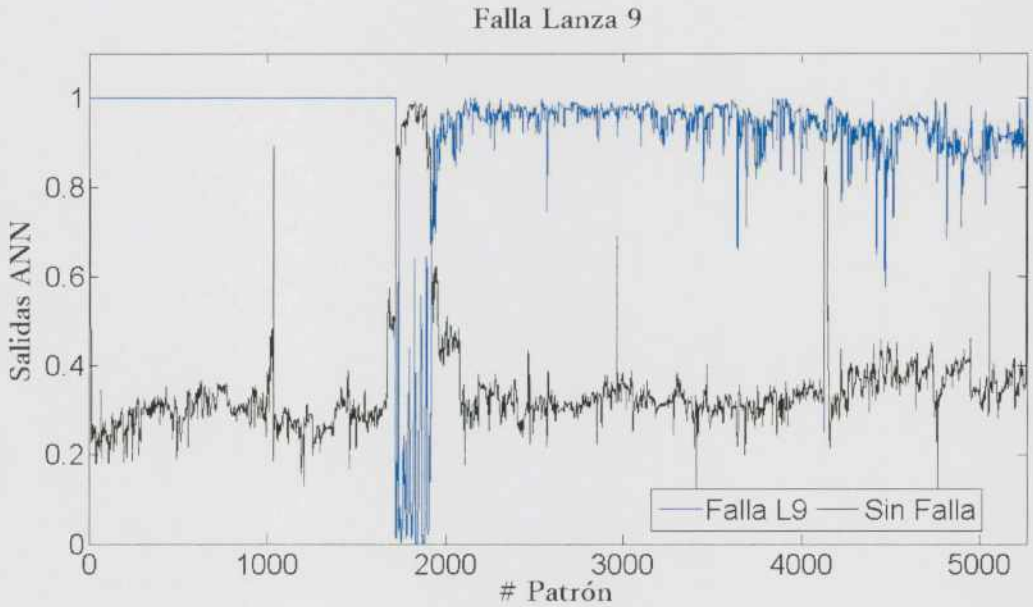

Figura 12.26: Salidas obtenidas con ANN al evaluar 5260 patrones con el detector 5 de la lanza 13 desconectado en todos los patrones

salidas deseadas en la mayoría de los casos. Esto representa una gran debilidad de esta metodología por no ser robusta ante este tipo de fallas.

De esta forma se llega a la conclusión de que el Sistema de Limitación obtenido a partir de la ANN no puede implementarse sin haber aplicado alguna estrategia de análisis y tratamiento de datos previa.

Una alternativa viable sería depurar las señales mediante otro sistema que detecte previamente las desconexiones de las lanzas o detectores y los reemplace por el promedio de los aledaños, antes de que los datos sean procesados por el Sistema de Limitación.

12.3.3 Amplificación de la señal en un solo detector

Se realizó el mismo procedimiento que con las desconexiones de cada detector, pero con el caso contrario. Al conjunto de 5260 patrones se le estableció el valor de la entrada correspondiente en 1, máxima entrada posible. Los promedios de los valores absolutos de las diferencias obtenidas entre las salidas obtenidas por la ANN y las obtenidas para la evalucaión sin fallas se presentan en la Tabla §12.14.

Lanza	Det	E_e	Lanza	Det	E_e	Lanza	Det	E_e
1	1	0,3307	6	1	0,3337	11	1	0,2022
1	2	0,6661	6	2	0,302	11	2	0,6658
1	3	0,6585	6	3	0,2365	11	3	0,2852
1	4	0,6655	6	4	0,0594	11	4	0,1968
1	5	0,2874	6	5	0,5649	11	5	0,3281
1	6	0,2903	6	6	0,6617	11	6	0,3257
2	1	0,1136	7	1	0,2205	12	1	0,2385
2	2	0,2282	7	2	0,6611	12	2	0,3326
2	3	0,2762	7	3	0,6642	12	3	0,5665
2	4	0,6117	7	4	0,6421	12	4	0,6644
2	5	0,5277	7	5	0,0637	12	5	0,6648
2	6	0,463	7	6	0,3338	12	6	0,6661
3	1	0,4698	8	1	0,6661	13	1	0,648
3	2	0,5399	8	2	0,6661	13	2	0,6639
3	3	0,666	8	3	0,2279	13	3	0,1058
3	4	0,6106	8	4	0,3004	13	4	0,301
3	5	0,2498	8	5	0,0602	13	5	0,457
3	6	0,1451	8	6	0,4565	13	6	0,1248
4	1	0,1084	9	1	0,0654	14	1	0,2415
4	2	0,6658	9	2	0,2628	14	2	0,3339
4	3	0,3133	9	3	0,3157	14	3	0,6372
4	4	0,3054	9	4	0,2966	14	4	0,6575
4	5	0,3259	9	5	0,2763	14	5	0,6661
4	6	0,1616	9	6	0,6633	14	6	0,6223
5	1	0,448	10	1	0,3129	15	1	0,3329
5	2	0,5062	10	2	0,3613	15	2	0,1406
5	3	0,6661	10	3	0,1009	15	3	0,1613
5	4	0,6601	10	4	0,3335	15	4	0,5737
5	5	0,3261	10	5	0,319	15	5	0,6656
5	6	0,2489	10	6	0,3324	15	6	0,4147

Tabla 12.14: Promedios de la diferencia entre la salidas obtenidas mediante la ANN evaluando los patrones sin fallas y las salidas obtenidas con un detector amplificado.

De la Tabla §12.14 se puede ver que el peor de los promedios resultó ser el obtenido con el detector 6 de la lanza 9 amplificado.

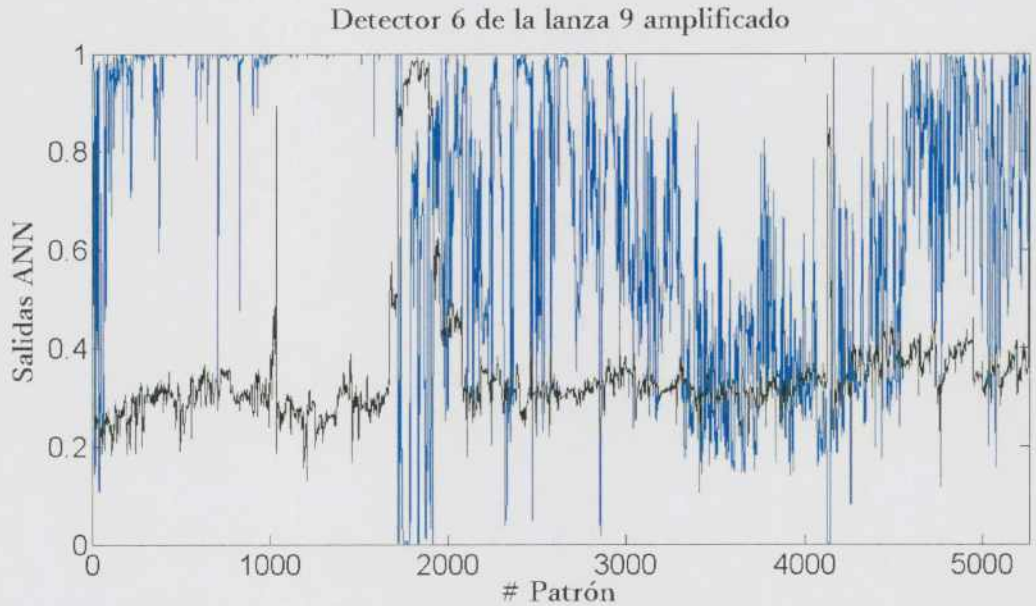

Figura 12.27: Salidas obtenidas con ANN al evaluar 5260 patrones con el detector 6 de la lanza 9 amplificado en todos los patrones.

Las salidas obtenidas por la ANN son presentadas en la Figura §12.27.

En este caso, se puede ver que existen discrepancias entre la salida obtenida y la salida esperada tanto por defecto como por exceso. Aunque se esperaba que se subestime el margen ante este tipo de fenómeno de amplificación, la ANN muestra un comportamiento poco predecible ante el ingreso de una señal totalmente amplificada. Lo cual denota su falta de robustez no sólo ante las desconexiones sino también ante las amplificaciones en las señales de los detectores.

12.3.4 Amplificación de la señal de una lanza

Se realizó el mismo procedimiento que con las desconexiones de cada lanza al conjunto de 5260 patrones se modificaron los valores a 1, representando el valor máximo de entrada. Los promedios se presentan en la Tabla §12.15.

Lanza	E_e	Lanza	E_e	Lanza	E_e	Lanza	E_e	Lanza	E_e
1	0,6628	4	0,4037	7	0,1891	10	0,325	13	0,1181
2	0,631	5	0,6659	8	0,2813	11	0,317	14	0,1088
3	0,1602	6	0,312	9	0,2907	12	0,6573	15	0,6275

Tabla 12.15: Promedios obtenidos de la diferencia entre las salidas de la ANN al evaluar los patrones sin fallas y al evaluar los patrones con una lanza amplificada.

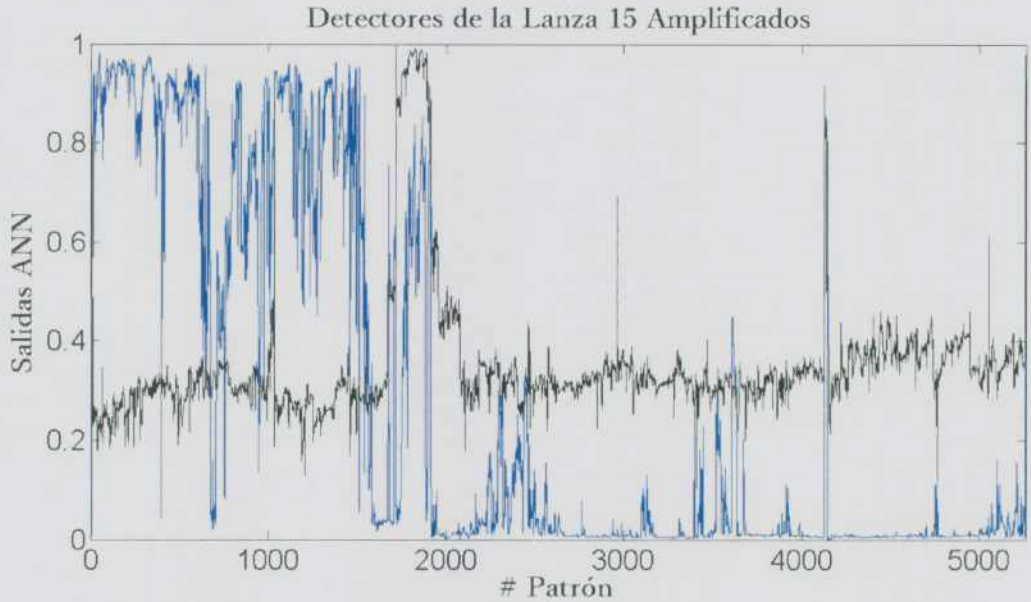

Figura 12.28: Salidas obtenidas con ANN al evaluar 5260 patrones con los detectores de la lanza 15 amplificados en todos los patrones.

De la Tabla §12.15 es posible ver que el promedio más alto que se corresponde con el peor de los resultados se da con la lanza 15 amplificada. Las salidas obtenidas por la ANN en este caso se presentan en la Figura §12.28.

En el caso de una amplificación de toda la lanza 15 existen dos comportamientos a lo largo de todo el conjunto de patrones.

Como se dijo en la Etapa 1 (pág 82) la primera serie de patrones que abarcan desde el comienzo hasta el primer incremento en las salidas esperadas, corresponden a los datos de la planta que contienen las mediciones de los detectores de la lanza número 2. Luego, cuando las salidas esperadas descienden, los patrones ya no cuentan con esas mediciones, y tienen entre sus valores, los correspondientes a la lanza número 2 desconectada.
A partir de esta disrupción se puede ver un comportamiento alejado del esperado, pero intuitivo en cuanto a las desviaciones, ya que en la mayoría de los patrones a partir de este punto se obtiene un margen inferior, o más conservativo que el que se hubiera obtenido si las entradas no hubiesen estado amplificadas.

Debido a que en el resto de los patrones no ocurre lo mismo, ni tampoco se tiene ese comportamiento ante la amplificación de un único detector, no se puede decir que el Sistema de Limitación sea confiable si es que, como se concluyó con las desconexiones, no se complementa con una estrategia de tratamiento de datos previo al procesamiento

de las entradas por la ANN.

Ya que este comportamiento de la ANN se debe a que se están evaluando patrones que son estados muy distintos a los estados con los que fue entrenada la ANN, otra solución alternativa sería el entrenamiento de patrones extraídos de estados de planta actuales y sus distintas variaciones de fallas, tanto por desconexiones como por amplificaciones, lo que llevaría un tiempo mucho más prolongado de entrenamiento. Por ello esta alternativa resulta ser mucho menos práctica que la anterior.

12.4 Etapa 4: Implementación del SLN para Funcionamiento *on-line*

En esta Etapa se evaluó el comportamiento del sistema de 90 detectores con la ANN de estructura [90 90 45 1], cuyos pesos sinápticos obtuvieron la mejor convergencia en la Etapa 2 de esta Fase.

Se obtuvo el margen al DNB de la planta con el SLN durante el mes de junio del año 2019 y se comparó con los margenes obtenidos por la READAT y el RELEB. Los resultados se presentan en la Figura §12.29.

Haciendo un análisis cualitativo de los resultados se puede observar en la Figura §12.29 que pese a que los casos evaluados no se encuentran en el rango de potencia en que fue entrenada la ANN, los margenes obtenidos se encuentran entre el valor calculado por el sistema de Limitación actual y la mejor estimación de la READAT.

El único periodo donde esto no se cumple es a partir del caso 15540, que se corresponde temporalmente con el día 21 de junio, fecha en que se realizó un recambio en las inmediaciones de una lanza que fue desconectada para READAT y RELEB, pero cuyas señales no se filtraron para el SLN.

Esto evidencia que el SLN puede obtener predicciones que son mejores a las obtenidas por el RELEB actual, incluso en estados de potencia distintos de utilizados en el entrenamiento.

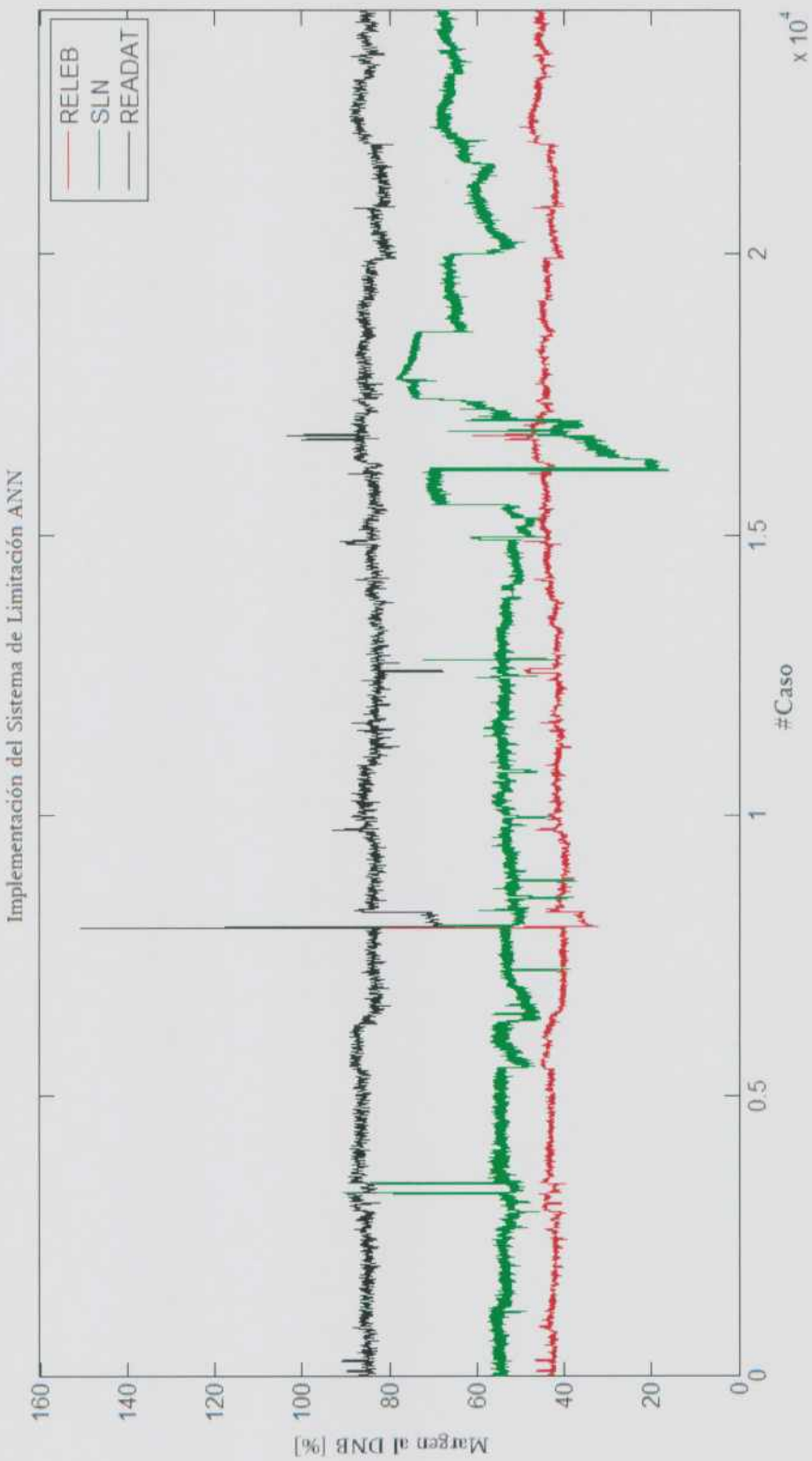

Figura 12.29: Margen al DNB obtenido con READAT (negro), Sistema de Limitación por ANN combinado (verde), RELEB (rojo), en función del número de caso.

Tercera Fase: Sistema de Limitación para la planta utilizando combustible ULE

13.1 Cálculos de Constantes de RELEB para ULE

A partir de los 1278 estados de planta simulados y del margen al DNB obtenido por AXCNA2 para cada uno, se obtuvieron las potencias lineales máximas en el DNB (q'_L) para las tres posiciones superiores de los detectores dentro de las lanzas en cada uno de los tres circuitos de vigilancia. (Figura §13.1).

Circuito	Detector	μ	σ	A_{ci}
1	1	127,6	3,1	118,4
1	2	189,6	4,8	175,3
1	3	197,6	4,9	182,7
2	1	130,9	3,0	121,7
2	2	196,5	4,8	181,9
2	3	203,3	5,3	187,5
3	1	94,7	2,7	86,5
3	2	158,2	4,7	143,9
3	3	170,9	4,8	156,5

Tabla 13.1: Parámetros obtenidos para la determinación de las constantes A_{ci} para los casos simulados de ULE.

Suponiendo que los valores siguen una distribución normal se obtuvo la media μ y el desvío σ para cada una de las tres posiciones superiores de los detectores dentro de las lanzas en los tres circuitos de vigilancia. Con estos valores se calcularon los valores de la constante A_{ci} para cada caso mediante la ecuación 10.4 presentada en la página 63 del Capítulo 10. Las constantes obtenidas se presentan en la Tabla §13.1, y corresponden a los valores de $q_{zul,i}$, en cada circuito según lo expresado en las ecuaciones 10.1, 10.2 y 10.3.

Con estos valores se calculó la distancia mínima al DNB según lo explicado en el Capítulo 5 (pág 31).

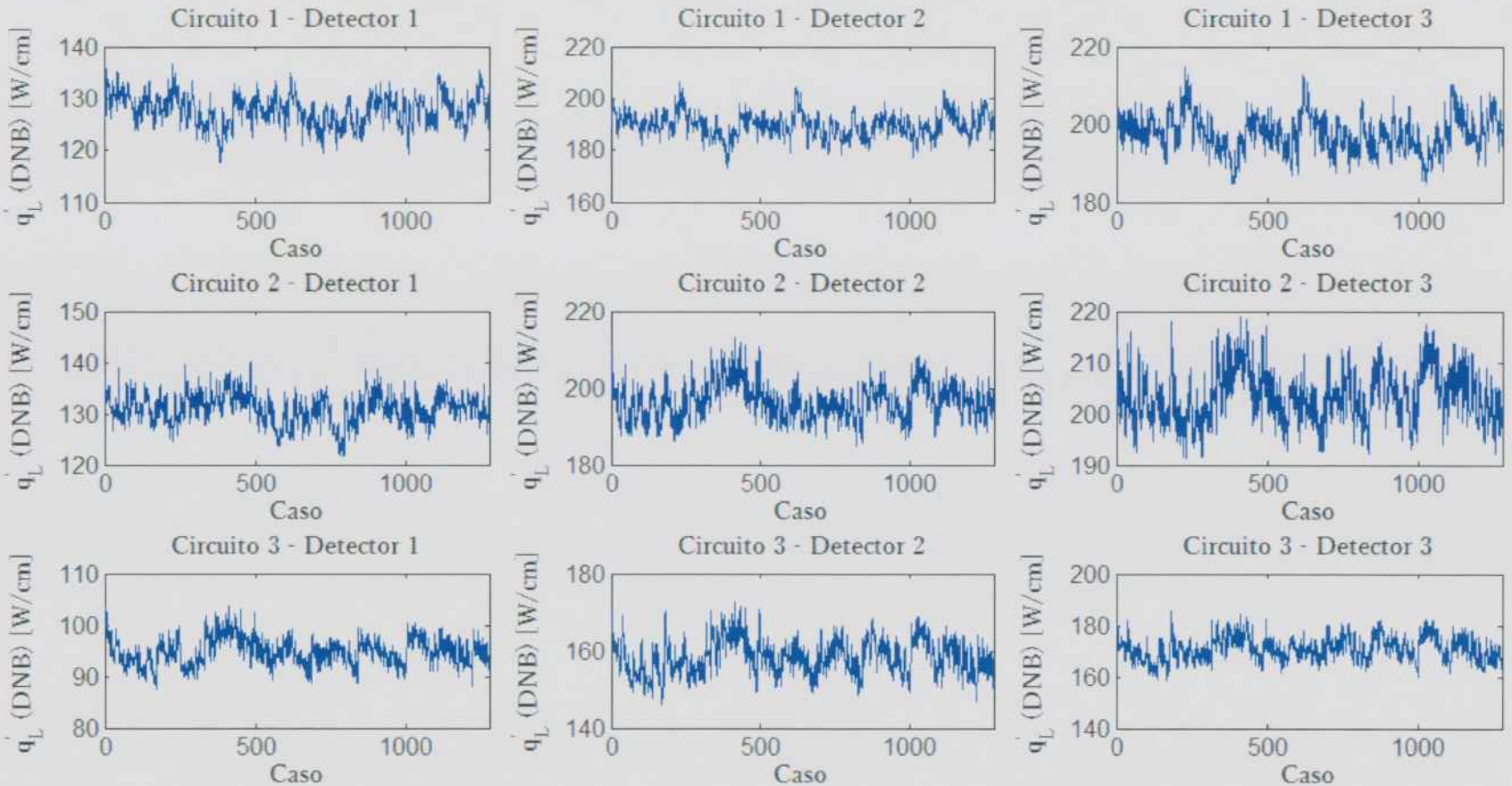

Figura 13.1: Valores de q'_L obtenidos para todos los patrones en las tres posiciones superiores de cada circuito

13.2 Sistema de Lanzas Virtuales

El sistema de Lanzas Virtuales cuenta con 12408 patrones en total, en los cuales se tienen los datos de las simulaciones del grupo de neutrónica con ULE a distintas potencias. Se seleccionó aleatoriamente el 90% de los patrones para ser utilizados en el entrenamiento, dejando el resto de los patrones como testigos sin participar del entrenamiento.

La estructura de ANN utilizada es la que cuenta con una estructura de 4 capas, con 18, 18, 12 y 1 neuronas en cada una de ellas, ya que tuvo el mejor desempeño en la evaluación de los datos de planta con uranio natural realizada en la Segunda Fase. Se partió de un estado inicial con una configuración de pesos sinápticos con números aleatorios.

Cabe destacar que al obtener los patrones en las potencias 105% y 110%, la predicción de margen al DNB obtenida por el AXCNA resultó menor que 0, con lo cual para poder utilizar valores que estén dentro del rango que la ANN puede utilizar ([0 1]), los valores de Margen al DNB de la READAT se corrigieron por medio de la expresión $d_o = (READAT/100 + 0{,}1) \cdot 1{,}6$.

De esta forma los valores deseados abarcan la mayor cantidad de valores posibles dentro del rango de valores posibles otorgados por la función de activación elegida. La funcionalidad del sistema no se ve afectada por ser sólo una modificación de escala, la cual es revertida a partir de la salida obtenida.

Este procedimiento no se realizó en ningún caso de la Segunda Fase de la Segunda Etapa, ya que entre los estados que se utilizaron como patrones no había valores de margenes obtenidos por la READAT menores a cero. Con los estados de ULE no se pueden descartar estos casos, ya que de lo contrario no se estaría contemplando estos valores de potencias factibles en las distintas condiciones de operación.

13.2.1 Entrenamiento

Se entrenaron a partir de una configuración de pesos aleatorios los patrones de entrenamiento con el método de EBPA. En la época número $t = 10^6$ se obtuvieron las salidas de la ANN, tanto de los patrones del entrenamiento como de los patrones testigos presentadas en la Figura §13.2. En ella se puede observar que en ciertas regiones la ANN otorga salidas que se corresponden con las deseadas, mientras que en determinadas potencias, las salidas difieren notablemente con las salidas deseadas, en su mayoría por defecto. Una vez alcanzado el límite en el cual el valor e no disminuye al

transcurrir las etapas, se continuó reentrenando la ANN con los patrones de entrenamiento utilizando el método de SA. Se a partió de la configuración de pesos alcanzada en la época $t = 1000000$ del entrenamiento con EBPA. Las salidas obtenidas al finalizar el entrenamiento con SA se presentan en la Figura §13.3. Se observa una clara mejoría en cuanto a las discrepancias obtenidas entre las salidas deseadas y las obtenidas.

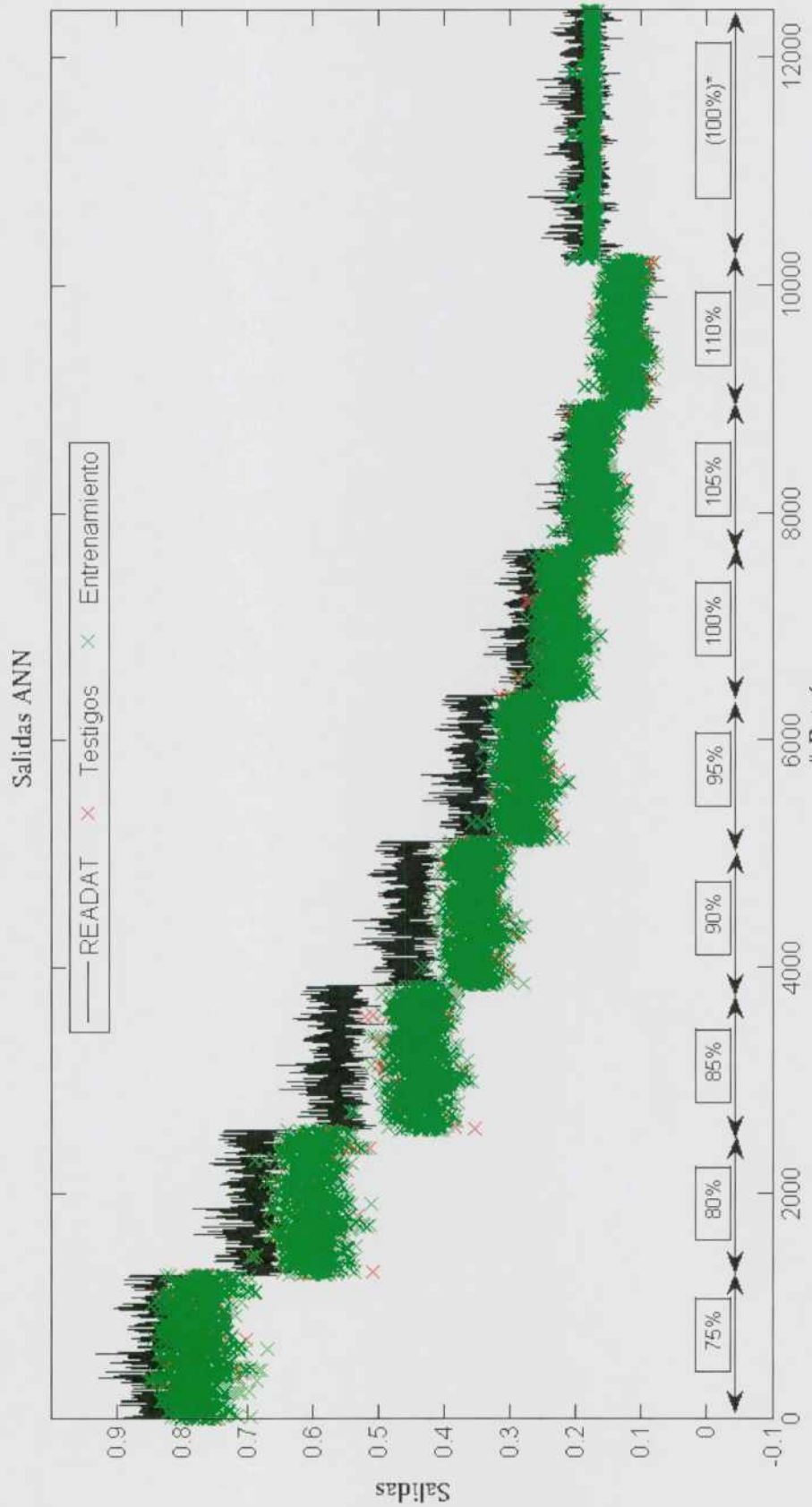

Figura 13.2: Salidas deseadas (negro): Margen al DNB obtenido por la READAT corregida por $d_o = (READAT/100+0,1)\cdot 1,6$; salidas obtenidas por la Red Neuronal Artificial (Salidas ANN por EBPA), para testigos (rojo) y para entrenamiento (verde).

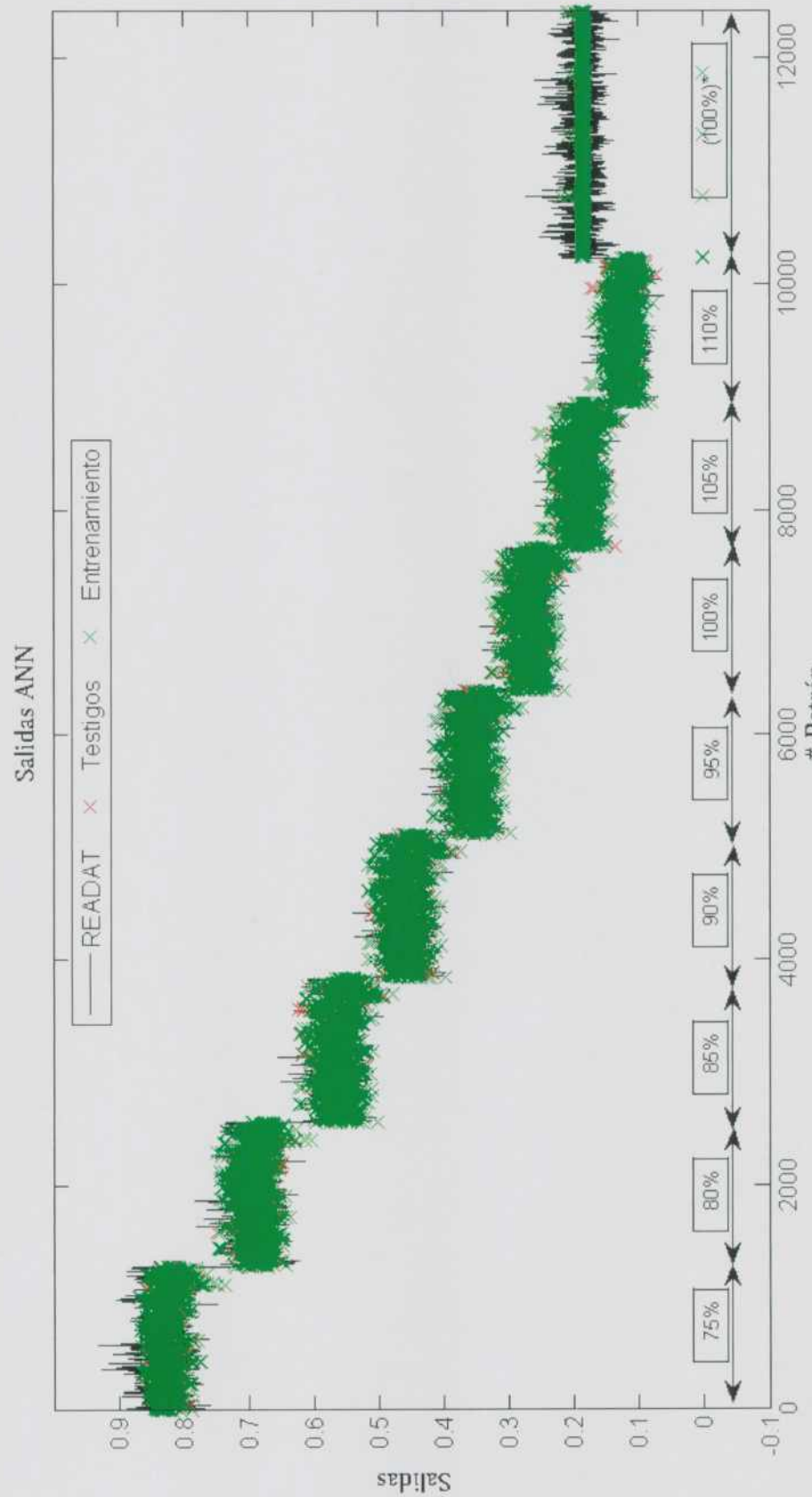

Figura 13.3: Salidas deseadas (negro): Margen al DNB obtenido por la READAT corregida por $d_o = (READAT/100+0,1)\cdot 1,6$; salidas obtenidas por la Red Neuronal Artificial (Salidas ANN por EBPA + SA), para testigos (rojo) y para entrenamiento (verde).

Técnica	ECM	μ	σ
Algoritmo de Retropropagación	0,005	0,000	0,070
Combinado	$5{,}5\times10^{-4}$	0,053	0,023

Tabla 13.2: Parámetros obtenidos del análisis de las diferencias entre las salidas deseadas y las salidas otorgadas por la ANN luego de ser entrenadas por el método EBPA y el método EBPA combinado con SA.

Sistema de Limitación	Ganancia [Margen]	Ganancia Relativa
Algoritmo de Retropropagación	9,3%	61,04%
Combinado	12,15%	82,68%

Tabla 13.3: Ganancias obtenidas a partir del análisis de los distintos Sistema de Limitación obtenidos a partir de la ANN y su diferencia con el Sistema de Limitación actual RELEB.

De las diferencias entre las salidas deseadas y las obtenidas en ambos casos de entrenamiento se obtuvieron el ECM, la media μ y la desviación estándar σ, cuyos valores se presentan en la Tabla §13.2.

A partir de los parámetros obtenidos se generó un Sistema de Limitación para cada uno de los casos con el mismo procedimiento empleado en la Segunda Fase, empleando la ecuación 12.2 (pág 87). Estos Sistemas de Limitación obtenidos se presentan en la Figura §13.4. Con los valores obtenidos del Sistema de limitación se calcularon las ganancias promedio y las ganancias relativas de ambos sistemas de forma análoga a lo realizado en la Segunda Fase, mediante las ecuaciones 12.3 y 12.4 (pág 89). Los resultados obtenidos se presentan en la Tabla §13.3.

Primero, es destacable notar que el Sistema de Limitación actual (rojo) otorga valores negativos a partir de los patrones al 90% de la potencia, de esta manera se puede corroborar lo poco eficaz que resultaría su implementación con ULE, ya que con esos valores la planta no podría operar normalmente. Aquí se hace evidente la necesidad de modificar el sistema por uno más flexible, lo cuál es el motivo fundamental de la realización de este proyecto.

Asimismo se puede ver que a pesar de que el entrenamiento con EBPA arrojó salidas con un desvío importante con respecto a las salidas deseadas, el Sistema de Limitación obtenido a partir de estos parámetros otorga un valor de ganancias relativa positivo, con lo cual brindaría valores de predicción al margen al DNB más precisos que los valores que proporcionaría el Sistema de Limitación actual.

Al reentrenar esta ANN con el método de SA, se obtiene un Sistema de Limitación aún más preciso, aumentando el valor de G_r hasta un valor de 82,68%.

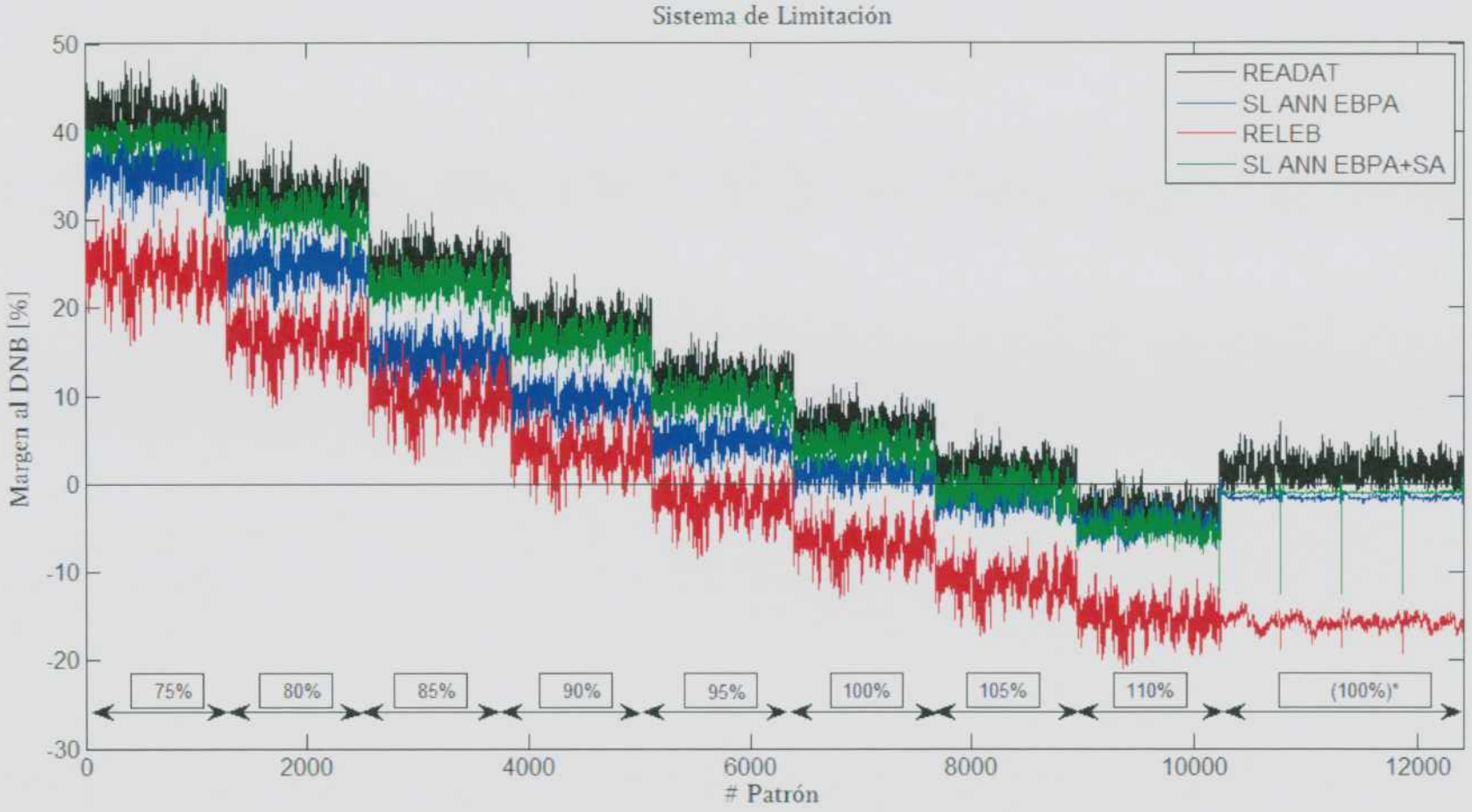

Figura 13.4: Margen al DNB porcentual obtenido por la READAT (negro), predicción al margen al DNB obtenido con el Sistema de Limitación actual RELEB, y los Sistemas de Limitación generados a partir de las salidas de ANN entrenada con EBPA (azul) y con los métodos combinados EBPA + SA (verde).

13.3 Sistema de señales de los 90 detectores

Al igual que el Sistema de Lanzas Virtuales, este sistema cuenta con 12408 patrones en total. Se procedió análogamente al Sistema de Lanzas Virtuales, utilizando la estructura de 4 capas con 90, 90, 45 y 1 neuronas en cada una de ellas, ya que es la que mejor desempeño obtuvo en la Fase Anterior.

13.3.1 Entrenamiento

Se seleccionó el 90% de los patrones para ser utilizados como datos de entrenamiento, dejando el 10% restante como testigos, los cuales no participaron del mismo. Se entrenó la ANN con los patrones utilizando el método de EBPA, y los resultados obtenidos al evaluar tanto los patrones testigos como los de entrenamiento con los pesos sinápticos otorgados en la época $t=10^6$ se presentan en la Figura §13.5.

Luego se reentrenó la ANN con los patrones de entrenamiento utilizando el método de SA, tomando como configuración inicial los pesos sinápticos otorgados con el entrenamiento de EBPA en la epoca $t=10^6$.

Las salidas obtenidas tanto para el entrenamiento como para los testigos se presentan en la Figura §13.6. A partir de las diferencias entre las salidas deseadas y las obtenidas en cada sistema se obtienen los parámetros presentados en la Tabla §13.4.

Del análisis de estos parámetros se puede ver que nuevamente y coincidentemente con lo obtenido para estas estructuras en la Segunda Fase, los valores de ECM obtenidos son menores que los obtenidos para el Sistema de Lanzas Virtuales, dado que las desviaciones entre las salidas deseadas y las obtenidas son menores que en el último.

De todas maneras puede verse que para la última fracción de patrones, que se corresponden con los estados simulados a los que se superponen los estados con perturbaciones, las ANNs tienen dificultades para otorgar las salidas en forma precisa, lo cual no ocurre para los estados anteriores donde no se contemplan estas variaciones.

A partir de estos parámetros se obtuvieron los Sistemas de Limitación presentados en la Figura §13.4.

Técnica	ECM	μ	σ
Algoritmo de Retropropagación	$1,6\times10^{-4}$	0,013	0,006
Combinado	$1,3\times10^{-4}$	0,012	0,003

Tabla 13.4: Parámetros obtenidos del análisis de las diferencias entre las salidas deseadas y las salidas otorgadas por la ANN luego de ser entrenadas por EBPA y EBPA combinado con SA.

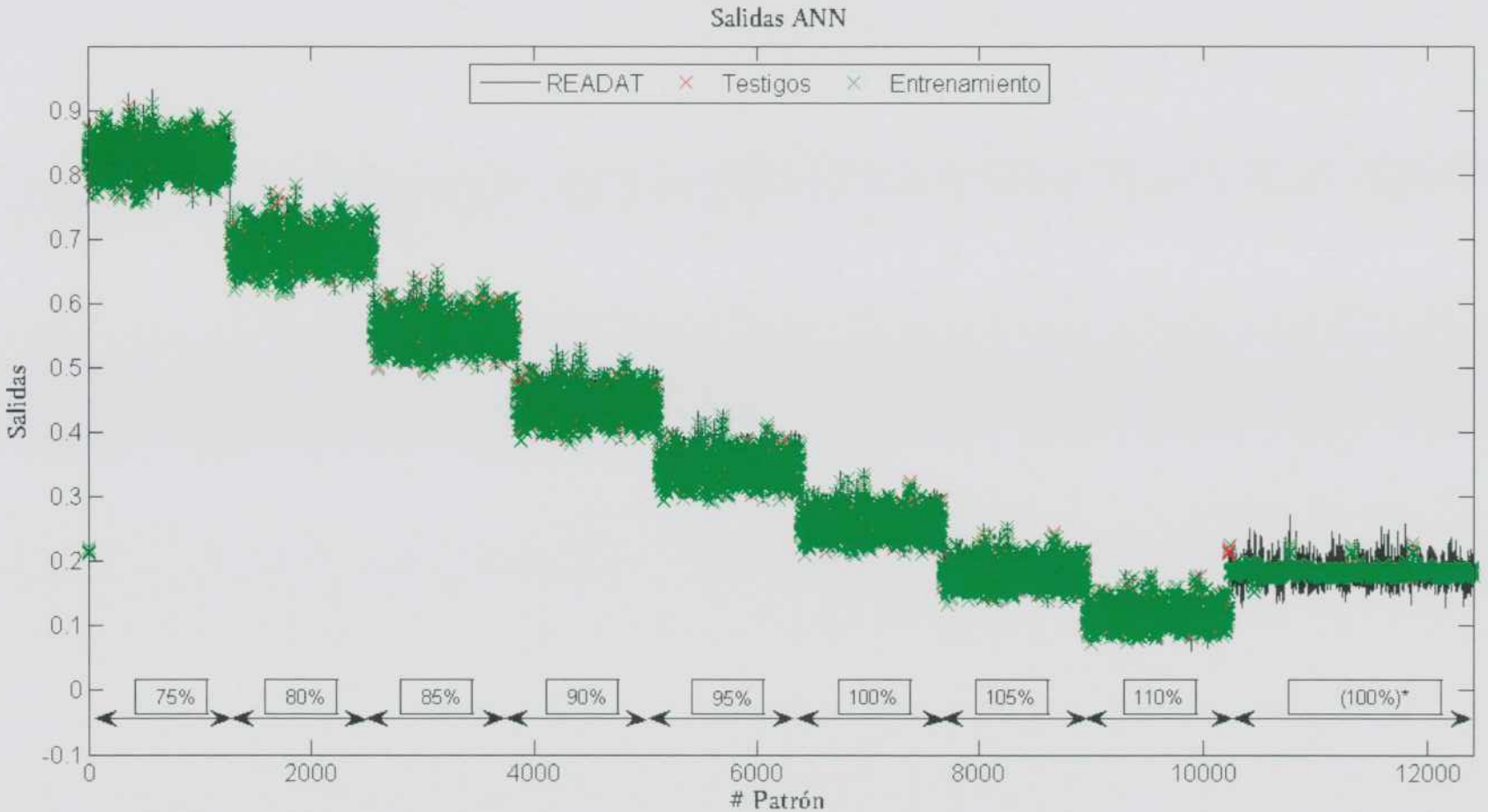

Figura 13.5: Salidas deseadas (negro): Margen al DNB obtenido por la READAT corregida por $d_o = (READAT/100+0,1)\cdot 1,6$; salidas obtenidas por la Red Neuronal Artificial (Salidas ANN por EBPA), para testigos (rojo) y para entrenamiento (verde).

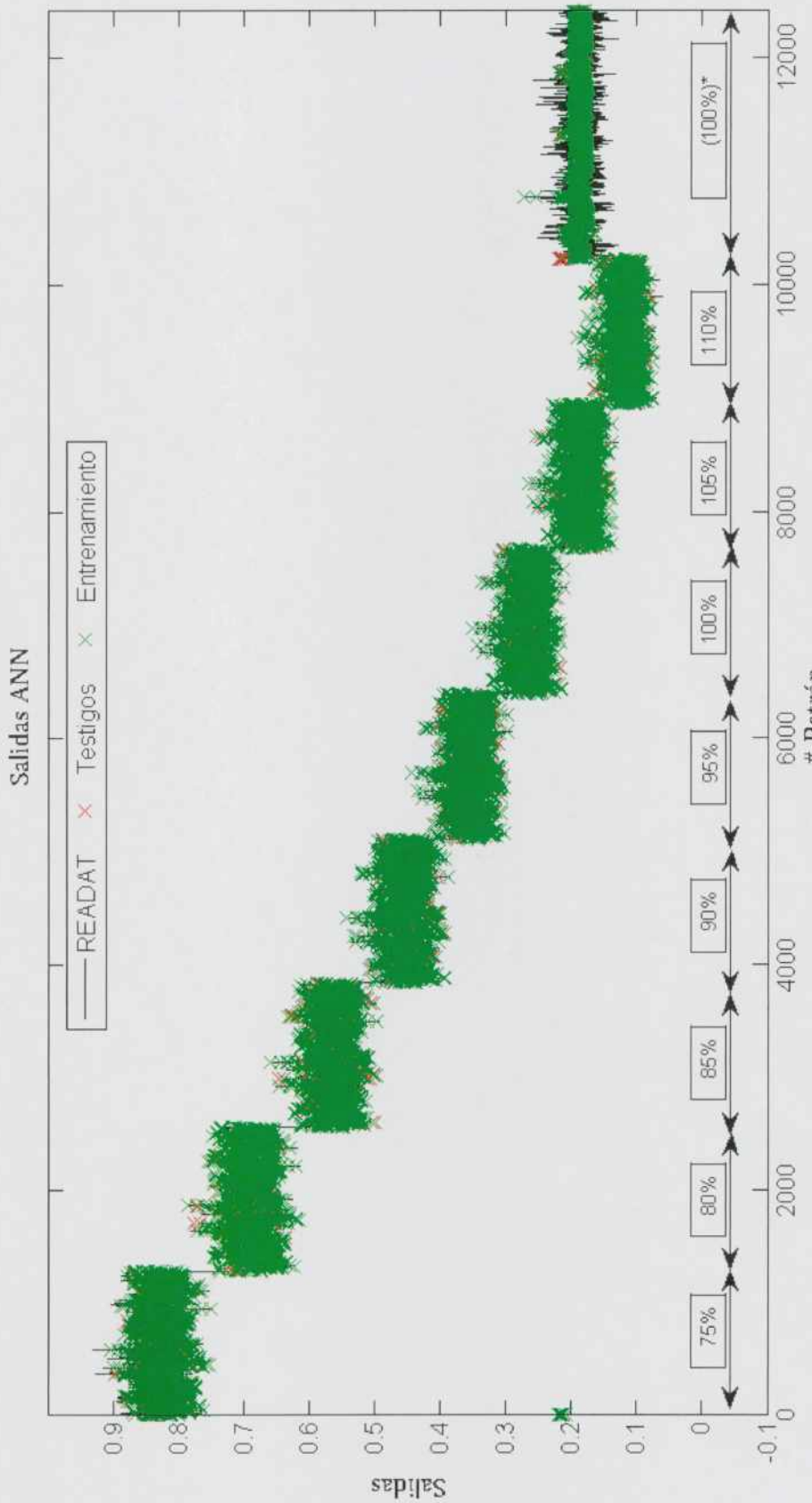

Figura 13.6: Salidas deseadas (negro): Margen al DNB obtenido por la READAT corregida por $d_o = (READAT/100+0,1)\cdot 1,6$; salidas obtenidas por la Red Neuronal Artificial (Salidas ANN por EBPA), para testigos (rojo) y para entrenamiento (verde).

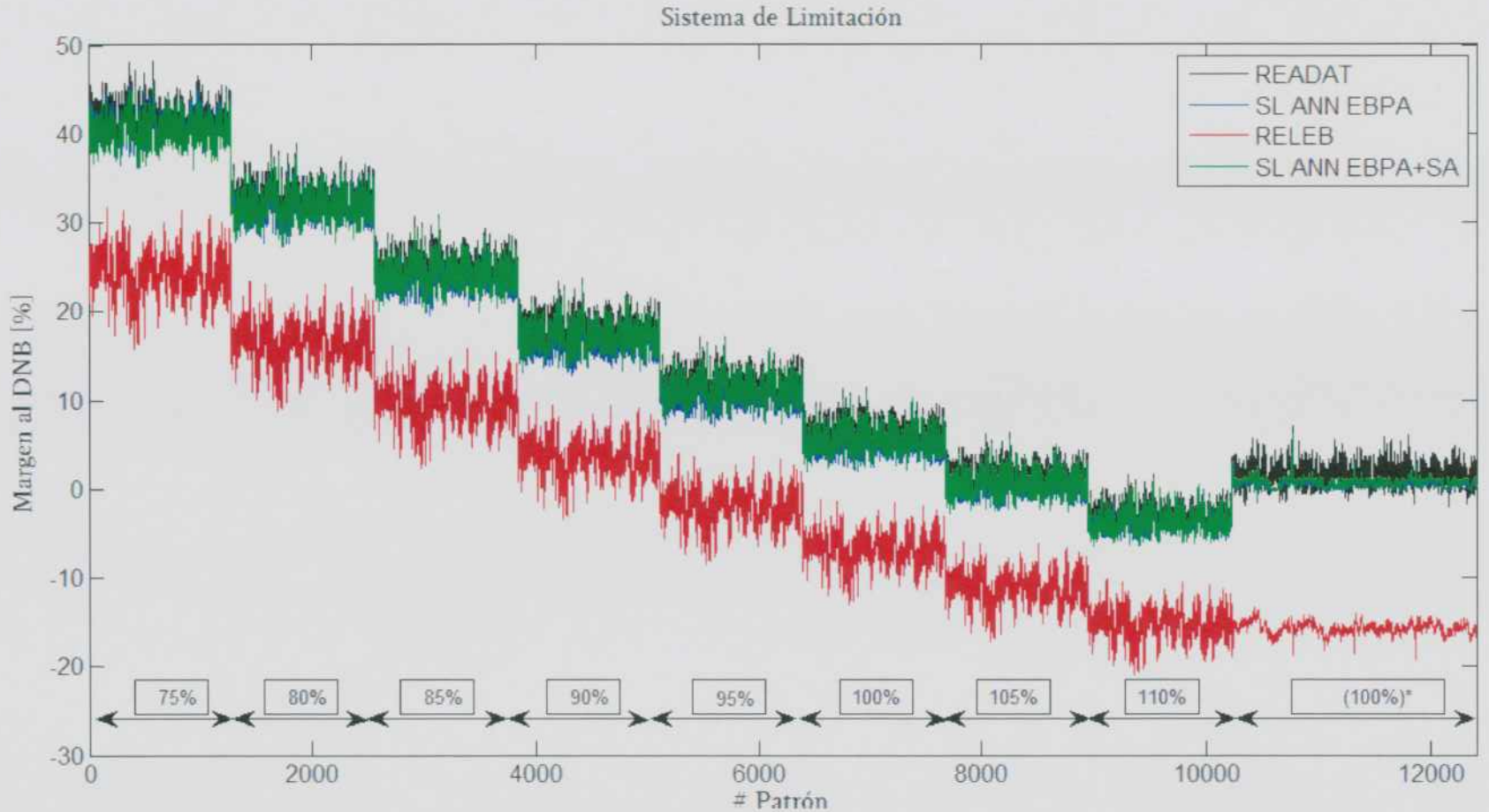

Figura 13.7: Margen al DNB porcentual obtenido por la READAT (negro), predicción al margen al DNB obtenido con el Sistema de Limitación actual RELEB, y los Sistemas de Limitación generados a partir de las salidas de ANN entrenada con EBPA (azul) y con los métodos combinados EBPA + SA (verde)

A partir de los valores de los Sistema de Limitación obtenidos, se calcularon las ganancias promedio y las ganancias relativas de cada SLN, los cuales se presentan en la Tabla §13.5.

Sistema de Limitación	Ganancia [Margen]	Ganancia Relativa
Algoritmo de Retropropagación	13,78%	90,09%
Combinado	14,05%	92,25%

Tabla 13.5: Ganancias obtenidas a partir del análisis de los distintos Sistema de Limitación obtenidos a partir de la ANN y su diferencia con el Sistema de Limitación actual RELEB.

Los valores de ganancias relativas nos indican una notable mejora con respecto a las obtenidas con el sistema de Lanzas Virtuales, ya que se ha llegado a obtener valores por encima del 90% siendo que con el sistema anterior se llegaba a un 82,68%.

La desventaja que se tiene es que, cómo se puede apreciar en la Figura §13.7, en la fracción correspondiente a los estados obtenidos por superposición de perturbaciones, tanto el Sistema de Limitación por ANN obtenido con el método EBPA, como el obtenido con los métodos combinados de EBPA + SA otorgan valores por encima de los valores otorgados por la READAT.

Con esto se excede una de las limitaciones: El margen predicho por el Sistema de Limitación debe proporcionar valores mas conservativos que los proporcionados por la Computadora de Procesos.

Este fenómeno se debe a que esta región es donde la ANN mostró limitaciones al adaptar sus salidas a las salidas deseadas, y por ende tiene los desvíos más pronunciados. Al crear el Sistema de Limitación a partir de la ecuación 12.2 (pág 12.2 del Capítulo 12), se contempla que esté por debajo del 95% de los valores de las salidas deseadas. Con lo cual un 5% de los casos pueden quedar excluidos del criterio.
Para este sistema un 5% de casos es a lo sumo 621 casos, los cuales se dan en su mayoría en la región antes mencionada.

Para poder subsanar este efecto, sería conveniente realizar un entrenamiento con un mayor número de épocas para que la ANN pueda alcanzar un nivel más elevado de desempeño en cuanto a la adaptación de estas salidas.

Otra alternativa es crear el Sistema de Limitación con un criterio mas conservativo, contemplando que este brinde valores que estén por encima del 99,5% o 99,9% de los casos, lo que corresponde a que en la ecuación 12.2 se resten 3 o 4 valores de σ respectivamente, en lugar de 2.

Parte VI

Conclusiones

Conclusiones del Proyecto Final Integrador

El que quiere algo conseguirá un medio, el que no, una excusa

Stephen Dolley

14.1 Sistemas de Limitación por ANN obtenidos

A partir del estudio del comportamiento de distintas estructuras de Redes Neuronales Artificiales, se llegó a las estructuras óptimas para la resolución de dos sistemas construidos a partir de datos de la planta. Para el Sistema de Lanzas Virtuales, la estructura de 4 capas con 18, 18, 12 y 1 neurona respectivamente y para el Sistema de 90 señales de Detectores *in-core*. la estructura de 4 capas con 90, 90, 45 y 1 respectivamente.

En cuanto al entrenamiento de estás estructuras, el método de Recocido Simulado ha demostrado obtener mejores tasas de convergencia para el entrenamiento en todas las instancias, independientemente del tamaño de la estructura de la ANN y de la cantidad de entradas. Esto se ve reflejado en tiempos menores de resolución en la ANN.

De todas maneras, se han alcanzado valores de e menores con el método de EBPA, ya que el mismo fue entrenado en períodos de épocas t tres ordenes de magnitud mayor. Para que ambos métodos sean comparables, se deberían realizar más pruebas con una mayor cantidad de épocas en SA para poder determinar su comportamiento en esas condiciones.

La combinación de los dos métodos, ha proporcionado predicciones que otorgan una ganancia relativa del margen calculado por la READAT (AXCNA2) respecto al margen del Sistema de Limitación actual (RELEB) mayor al obtenido por EBPA. Para la Segunda Fase - UN (71,61 % vs 70,56 %) como para la Tercera Fase - ULE (92,25 % vs 90,09 %).

Del estudio del Sistema de Lanzas Virtuales y del Sistema con las señales de los 90 detectores $in-core$, se puede concluir que en el segundo sistema proporciona predic-

ciones más precisas y que otorgan una ganancia relativa mayor:
71,61 % vs 53,3 % con combustible de uranio natural y 92,25 % vs 82,68 % con ULE.

Cabe destacar que el método de Retropropagación del Error debería ser evaluado a partir de configuraciones aleatorias para discernir si es posible obtener mejores resultados para el Sistema de Lanzas Virtuales en la Etapa 2 de la Segunda Fase.

En la implementación del SLN en la Etapa 4 de la Segunda Fase se obtuvieron resultados cualitativos positivos, a pesar de que se realizó la predicción del margen con estados de potencia menores que los estados utilizados en el entrenamiento de la ANN.

La desventaja de los SLN para el sistema de 90 señales de detectores *in-core* es que ante fallas no se obtienen resultados predecibles. Se comprobó que no son robustas ante desconexiones o ante señales amplificadas.

Una solución viable es la aplicación de un sistema de filtrado de señales previo al procesamiento de la ANN, análogo al existente en la planta. Otra alternativa es un entrenamiento con todas las combinaciones posibles de errores (90!) pero resulta poco práctico para su implementación, debido al elevado número de combinaciones que presenta. Con lo cual el costo computacional de entrenamiento se elevaría en más de 100 ordenes de magnitud por patrón.

Los SLN obtenidos en la Tercera Fase para estados simulados con combustible ULE presentan una clara mejora respecto a la utilización del Sistema de Limitación actual. Esto otorga evidencia que la implementación de Redes Neuronales Artificiales es una alternativa viable para la construcción de un nuevo algoritmo de cálculo para el Sistema de Limitación.

No obstante, debe ser estudiado con mucho más estados simulados y con una mayor cantidad de épocas t de entrenamiento para poder estudiar la confiabilidad del sistema.

Bibliografía

[1] Informe trimestral de coyuntura energética - primer trimestre 2018. *Dirección Nacional de Información Energética*, 2018. (page 9).

[2] Hyun-Koon Kim, Seung-Hyuk Lee, and Soon-Heung Chang. Neural network model for estimating departure from nucleate boiling performance of a pressurized water reactor core. *Nuclear technology*, 101(2):111–122, 1993. (page 11).

[3] BT Jiang and YN Liu. A brief review of computational intelligence techniques for critical heat flux prediction. In *2018 26th International Conference on Nuclear Engineering*, pages V06BT08A050–V06BT08A050. American Society of Mechanical Engineers, 2018. (page 11).

[4] MS Silva. *Desarrollo de un programa para calcular la distribución de flujo neutrónico por un método de síntesis modal basado en las lecturas de los detectores de la Central Nuclear Atucha-II*. PhD thesis, Tesis de Maestría, Facultad de Ingeniería, Universidad de Buenos Aires, 2012. (page 18).

[5] L. Lencina. Cálculo de los coeficientes del algoritmo de limitación por dnb del sistema de limitación de potencia del reactor (releb) para la cna2 considerando la transferencia de calor al moderador. *LN-BN-000061 (Informe interno NA-SA)*. (pages 29, 55 y 79).

[6] H. Ballesteros. Descripción y teoría del programa axcna2. *LN-BN-000133 (Informe interno NA-SA)*. (page 34).

[7] DC Groeneveld, LKH Leung, PL Kirillov, VP Bobkov, IP Smogalev, VN Vinogradov, XC Huang, and E Royer. The 1995 look-up table for critical heat flux in tubes. *Nuclear Engineering and design*, 163(1-2):1–23, 1996. (page 34).

[8] SY Ahmad. Fluid to fluid modeling of critical heat flux: a compensated distortion model. *International Journal of Heat and Mass Transfer*, 16(3):641–662, 1973. (page 35).

[9] Ben Krose and PV Smagt. An introduction to neural network . the university of amsterdam. *Amesterdam, Netherland*, 1996. (page 39).

[10] Brian P Flannery, William H Press, Saul A Teukolsky, and William Vetterling. Numerical recipes in c. *Press Syndicate of the University of Cambridge, New York*, 24:78, 1992. (pages 46 y 76).

Parte

ANEXOS

ANEXO A

Código desarrollado para la creación de la estructura de ANN y Entrenamiento con EBPA

```
#include <iostream>
#include <octave/oct.h>
#include <vector>
#include <fstream>
#include <iomanip>
#include <stdio.h>
using namespace std;
RowVector activacion(RowVector s);//declaro las funciones de
   activacion y su derivada
RowVector der_activacion(RowVector s);
double der_activacion(double s);
class mlp { // definicion clase mlp
public:
    int ncapas; //numero de capas
    double gamma; //coeficiente de aprendizaje
    double alpha; //coeficiente de entrenamiento
    ColumnVector tamanio; //numero de neuronas en cada capa
    std::vector <Matrix> pesos; // especie de vector que contiene
       matrices
    std::vector <Matrix> bias; //
    std::vector <Matrix> DeltaAnterior; //
    std::vector <RowVector> delta_chico; //
    std::vector <Matrix> Delta; //
    std::vector <Matrix> DeltaBias; //
    std::vector <Matrix> DeltaAnteriorBias; //
    std::vector <RowVector> in; //
    std::vector <RowVector> s; //
    std::vector <ColumnVector> sumaaux; //
```

```
27      mlp(ColumnVector &v) : ncapas(v.dim1()), tamanio(v) { //
            construccion mlp
28          gamma = 0.99;
29           alpha = 0.1;
30       for (int i = 0; i < (ncapas - 1); i++) { //este for contruye las
             matrices con el tama o dado en el vector "v"
31              pesos.push_back(Matrix(tamanio(i), tamanio(i + 1)));
32              bias.push_back(Matrix(1, tamanio(i + 1)));
33              DeltaAnterior.push_back(Matrix(tamanio(i), tamanio(i + 1)
                    ));
34              Delta.push_back(Matrix(tamanio(i), tamanio(i + 1)));
35              DeltaBias.push_back(Matrix(1, tamanio(i + 1)));
36              DeltaAnteriorBias.push_back(Matrix(1, tamanio(i + 1)));
37              delta_chico.push_back(RowVector(tamanio(i + 1)));
38              s.push_back(RowVector(tamanio(i + 1))); //las sumatorias
39       }
40          for (int i = 0; i < (ncapas); i++) {//este for es para hacer
               las entradas y las sumas auxiliares
41              in.push_back(RowVector(tamanio(i))); //entradas
42              sumaaux.push_back(ColumnVector(tamanio(i)));
43          }
44          for (int i = 0; i < (ncapas - 1); i++) {//para usar desde
               epoca=0
45              for (octave_idx_type j = 0; j < (tamanio(i)); j++) {
46                  for (octave_idx_type k = 0; k < tamanio(i + 1); k++) {
47                       pesos[i](j, k) = double (rand() % 100) / 100*0.1;
                             //defino los pesos aleatorios
48                       DeltaAnterior[i](j, k) = 0; //inicio los delta
                            anteriores en cero
49                   }
50              }
51              for (octave_idx_type j = 0; j < (tamanio(i + 1)); j++) {
                   //defino los bias para los lugares posteriores
52                   bias[i](0, j) = double (rand() % 100) / 100*0.1;
53                   DeltaAnteriorBias[i](0, j) = 0;
54              }
55          }
56      }
```

```
57  //-------------------------------------------------------------------------

58    RowVector evalua_red(RowVector in) { //defino un vector que de las
        salidas para las entradas establecidas
59        this->in[0] = in;
60        for (int i = 0; i < ncapas - 1; i++) {
61            this->s[i] = this->in[i] * this->pesos[i] + this->bias[i
                ]; //todos los elementos deben ser congruentes en
                dimensiones
62            this->in[i + 1] = activacion(this->s[i]);
63        }
64        return this->in[ncapas - 1];
65    }
66  //-------------------------------------------------------------------------

67    int correccion_BackPropagation(RowVector& out, RowVector & d_o) {
        //corrige estructuras a partir de los valores deseados
68      this->sumaaux[ncapas - 1] = d_o - out; //defino la suma
          auxiliar del la ultima capa como la diferencia entre lo
          obtenido con lo deseado
69        for (int i = ncapas - 2; i >= 0; i--) { //la suma
            auxiliares para las capas ocultas necesitan del calculo
             del delta chico
70          for (int k = 0; k<this->s[i].dim2(); k++) {
71            this->delta_chico[i](k) = (der_activacion(this->s[i](k)
                )) * this->sumaaux[i + 1](k);
72           }
73           this->sumaaux[i] = this->pesos[i] * this->delta_chico[i].
               transpose();
74           this->Delta[i] = this->gamma * this->in[i].transpose() *
               this->delta_chico[i] + this->alpha * this->
               DeltaAnterior[i];
75           this->DeltaBias[i] = this->gamma * this->delta_chico[i] +
                this->alpha * this->DeltaAnteriorBias[i];
76        }
77        for (int i = ncapas - 2; i >= 0; i--) { //con esta funcion
            redefino los pesos y los bias y redefino los pesos y bias
            "anteriores"
78           this->pesos[i] += this-> Delta[i];
```

```
79              this->bias[i] += this-> DeltaBias[i];
80              this->DeltaAnterior[i] = this->Delta[i];
81              this->DeltaAnteriorBias[i] = this->DeltaBias[i];
82          }
83          return 0;
84      }
85      int guarda_pesos_bias(int epoca, int codigo) {
86          char filename[100];
87          snprintf(filename, 100, "pesos_y_bias_epoca_%d_%d.txt", epoca
                , codigo);
88          fstream out_pesos_bias(filename, fstream::out);
89          for (int i = ncapas - 2; i >= 0; i--) {
90              out_pesos_bias << "Pesos " << i << endl;
91              out_pesos_bias << this->pesos[i] << endl;
92              out_pesos_bias << "Bias " << i << endl;
93              out_pesos_bias << this->bias[i] << endl;
94          }
95          out_pesos_bias.close();
96          return 0;
97      }
98      int guarda_out(int epoca, int codigo, Matrix Matriz_salidas) {
99          char filename_out[100];
100         snprintf(filename_out, 100, "salidas_%d_%d.txt", codigo,
                epoca);
101         fstream out_s(filename_out, fstream::out);
102         out_s << Matriz_salidas << endl;
103         out_s.close();
104         return 0;
105     }
106     int guarda_out_entrenamiento(int epoca, int codigo, Matrix
            Matriz_salidas_entrenamiento) {
107         char filename_out[100];
108         snprintf(filename_out, 100, "salidas_entrenamiento_%d_%d.txt
                ", codigo, epoca);
109         fstream out_s(filename_out, fstream::out);
110         out_s << Matriz_salidas_entrenamiento << endl;
111         out_s.close();
112         return 0;
113     }
```

```
114     int guarda_out_testigos(int epoca, int codigo, Matrix
           Matriz_salidas_testigos) {
115         char filename_out[100];
116         snprintf(filename_out, 100, "salidas_testigos_%d_%d.txt",
               codigo, epoca);
117         fstream out_s(filename_out, fstream::out);
118         out_s <<Matriz_salidas_testigos << endl;
119         out_s.close();
120         return 0;
121     }
122 };
123 //---------------------------------------------------------------------------

124 //       defino las funciones previamente declaradas
125 //---------------------------------------------------------------------------

126 RowVector activacion(RowVector s) {
127     octave_idx_type ns = s.dim2();
128     RowVector y(ns);
129     for (int i = 0; i < ns; i++) {
130         y(i) = float ( 1 / (1 + exp(-s(i))));
131     }
132     return y;
133 }
134 //---------------------------------------------------------------------------

135 double der_activacion(double s) {
136     return ( 1.0 / (1.0 + exp(-s)))*(1.0 - (1.0 / (1.0 + exp(-s))));
137 }
138 //---------------------------------------------------------------------------

139 int main(int argc, char *argv[]) {
140     ColumnVector data(argc - 1);
141     int val;
142     cout << "configuracion:" << endl;
143     for (int i = 1; i < argc; i++) {
144         istringstream ss(argv[i]);
145         ss>>val;
146         data(i - 1) = val;
```

```
147     }
148     istringstream ss(argv[argc - 1]);
149     //ss>>filename_pesos;
150     //cout  << filename_pesos << endl;
151     int codigo;
152     cout << "Ingrese codigo de ID del proceso";
153     cin>>codigo;
154     //codigo = data(0);
155     cout << "Identificacion del procesamiento " << codigo << "\n" <<
           endl;
156     //creo los vectores para los patrones de entrenamiento
157     std::vector <RowVector> Vector_entrenamiento_in;
158     std::vector <RowVector> Vector_entrenamiento_d_o;
159     std::vector <RowVector> Vector_diferencia;
160     std::vector <RowVector> Vector_error_epoca;
161     //para los testigos
162     std::vector <RowVector> Vector_testigos_in;
163     std::vector <RowVector> Vector_testigos_d_o;
164     std::vector <RowVector> Vector_diferencia_t;
165     std::vector <RowVector> Vector_error_epoca_t;
166     //para todos los patrones
167     std::vector <RowVector> Vector_data_in;
168     std::vector <RowVector> Vector_data_d_o;
169     std::vector <RowVector> Vector_data_out;
170     std::vector <RowVector> Vector_diferencia_data;
171     std::vector <RowVector> Vector_error_epoca_data;
172     //*****************************************************************

173     //TRATAMIENTO DE DATOS DE ENTRADA
174     //*****************************************************************

175     //leo los valores del entrenamiento desde un archivo determinado
176     //en caso de querer que el programa sea interactivo se pueden
           comentar las lineas correspondientes
177     char filename_ent[132] = "data_entrenamiento_e2.txt";
178     //cout << "\n Ingrese nombre del archivo de entrenamiento\n";
179     //cin >> filename_ent;
180     char filename_t[132] = "data_testigos_e2.txt";
181     //cout << "\n Ingrese nombre del archivo de testigos\n";
```

```
182     //cin >> filename_t;
183     fstream in(filename_ent, fstream::in); //abro ese archivo
184     int dim1, dim2; //declaro filas (dim1) y columnas (dim2)
185     in >> dim1>>dim2;
186     Matrix x_m(dim1, dim2); //creo una matriz con esas dimensiones
187     Matrix Matriz_salidas_entrenamiento(dim1,2);//creo la matriz para
            guardar las salidas
188     int entradas, salidas;
189     cout<<"\n Ingrese numero de entradas\n";
190     cin>>entradas;
191     cout<<"\n Ingrese numero de salidas\n";
192     cin>>salidas;
193     //entradas = data(4);
194     //salidas = data(5);
195     cout << "entradas: " << entradas << " salidas: " << salidas <<
           endl;
196     for (int i = 0; i < dim1; i++) {//este pasa por todas las filas (
           cada fila es un patron))
197      Vector_entrenamiento_in.push_back(RowVector(entradas)); //
            crea un vector de longitud de "entradas")
198      Vector_entrenamiento_d_o.push_back(RowVector(salidas)); //
            crea un vector de longitud de "salidas"))
199      Vector_diferencia.push_back(RowVector(salidas));
200     }
201     in >> x_m; //los valores luego de los primeros dos que son las
           dimensiones
202     for (int p = 0; p < dim1; p++) {//p de patron
203      for (int i = 0; i < entradas; i++) {
204          Vector_entrenamiento_in[p](i) = x_m(p, i + 1) / 250; //
                normalizo las entradas
205      }
206      Vector_entrenamiento_d_o[p](0) = (x_m(p, 92) / 100+0.1)*1.6;
            // normalizo las salidas
207      Matriz_salidas_entrenamiento(p,0)=x_m(p, 0);
208     }
209     in.close();
210     //leo los testigos desde otro archivo determinado procedo igual
           que en entrenamiento
211     fstream in_t(filename_t, fstream::in); //
```

```
212     int dim1t, dim2t;
213     in_t >> dim1t>>dim2t;
214     Matrix x_m_t(dim1t, dim2t);
215     Matrix Matriz_salidas_testigos(dim1t,2);
216     for (int i = 0; i < dim1t; i++) {
217         Vector_testigos_in.push_back(RowVector(entradas));
218         Vector_testigos_d_o.push_back(RowVector(salidas));
219         Vector_diferencia_t.push_back(RowVector(salidas));
220     }
221     in_t >> x_m_t;
222     for (int p = 0; p < dim1t; p++) {
223         for (int i = 0; i < entradas; i++) {
224             Vector_testigos_in[p](i) = x_m_t(p, i + 1) / 250;
225         }
226         Vector_testigos_d_o[p](0) = (x_m_t(p, 92) / 100+0.1)*1.6;
227         Matriz_salidas_testigos(p,0)=x_m_t(p, 0);
228     }
229     in_t.close();
230     //patrones totales
231     char filename_data[132] = "Data_red_ULE_90d_e2_bis.txt";
232     //cout << "\n Ingrese nombre del archivo de patrones\n";
233     //cin >> filename_ent;
234     fstream in_d(filename_data, fstream::in); //abro ese archivo
235     int dim1_data, dim2_data;
236     in_d >> dim1_data>>dim2_data;
237     Matrix x_m_data(dim1_data, dim2_data);
238     for (int i = 0; i < dim1_data; i++) {
239         Vector_data_in.push_back(RowVector(entradas));
240         Vector_data_d_o.push_back(RowVector(salidas));
241         Vector_data_out.push_back(RowVector(salidas));
242     }
243     in_d >> x_m_data;
244     for (int p = 0; p < dim1_data; p++) {
245         for (int i = 0; i < entradas; i++) {
246             Vector_data_in[p](i) = x_m_data(p, i + 1) / 250;
247         }
248         //cout<<Vector_data_in[p]<<endl;
249         Vector_data_d_o[p](0) = (x_m_data(p, 92) / 100+0.1)*1.6;
250     }
```

```
251     in_d.close();
252     //********    **********    ***********    *************
           ***********
253     //                      CREACION DE LA RED NEURONAL
254     //********    **********    ***********    *************
           ***********
255     //en principio creo un vector con la cantidad de capas que deseo
256     int ne = entradas; //numero de entradas
257     cout<< "\nIngrese numero de capas que tendra la red\n";
258     int ncapas; //numero de capas
259     cin>>ncapas;
260     //ncapas = data(3);
261     ColumnVector a(ncapas); //creo vector que va a definir la red
           neuronal
262     RowVector s;
263     //se coloca en cada capa la cantidad de neuronas que deseo que
           tenga;
264     a(0) = entradas;
265     a(ncapas - 1) = salidas;
266     for (int i = 0; i < ncapas - 2; i++) {
267         cout <<"\n Ingrese el numero de neuronas de la capa:"<<i+2<<
               endl;
268         cin>>a(i+1);
269         //a(i + 1) = data(i + 6);
270         cout << " neuronas de la capa " << i + 2 << ":" << a(i + 1)
               << endl;
271     }
272     // creo la red neuronal con la clase mlp previamente definida
273     mlp Red_Neuronal(a);
274     //*************************************************************************

275     // Modifico los pesos y los bias en la estructura
276     //*************************************************************************

277     char filename_pesos[100]="pesos_y_bias_epoca_8923_1.txt";
278     //cout<<"Ingrese el nombre del archivo que contiene los pesos y
           bias deseados para continuar el entrenamiento";
279     //cin>>filename_pesos;
280     fstream in_p(filename_pesos, fstream::in); //abro ese archivo
```

```
281     for (int i = ncapas - 2; i >= 0; i--) {
282         int b;
283         char pesos_char[7];
284         char bias_char [7];
285         Matrix pesos1(a(i), a(i + 1));
286         Matrix bias2(1, a(i + 1));
287         in_p>> pesos_char; //extraccion de datos del archivo con el
                formato indicado
288         in_p>> b;
289         in_p>>pesos1;
290         in_p>>bias_char;
291         in_p>> b;
292         in_p>>bias2;
293         for (octave_idx_type j = 0; j < (a(i)); j++) {//redefino los
                pesos a partir de los datos
294             for (octave_idx_type k = 0; k < a(i + 1); k++) {
295                 Red_Neuronal.pesos[i](j, k) = pesos1(j, k); //defino
                        los pesos
296             }
297         }
298         for (octave_idx_type j = 0; j < (a(i + 1)); j++) { //defino
                los bias para los lugares posteriores, si bien es una
                matriz solo uso una fila
299             Red_Neuronal.bias[i](0, j) = bias2(0, j);
300         }
301     }
302     //-------------------------------------------------------------------------

303     //        Entrenamiento
304     //-------------------------------------------------------------------------

305     int epoca, feedback;
306     cout<< "\n Ingrese el numero total de epocas que desea entrenar\n
            ";
307     cin>> epoca; //defino la cantidad de epocas
308     cout<< "\n Ingrese el numero de epocas cada cuanto desea obtener
            un feedback\n";
309     cin>> feedback;//defino la cantidad de epocas en que quiero
            obtener un feedback
```

```
310     //epoca = data(1);
311     //feedback = data(2);
312     cout << "epoca " << epoca << " fb " << feedback << endl;
313     Matrix Matriz_promedio(epoca, 3); //matriz de salida con el
           promedio del arror por epoca
314     //inicializo la matriz en cero
315     for (int f = 0; f < epoca; f++) {
316         for (int c = 0; c < 3; c++) {
317             Matriz_promedio(f, c) = 0;
318         }
319     }
320     for (int t = 0; t < epoca; t++) { //t epoca
321         Vector_error_epoca.push_back(RowVector(1));
322         Vector_error_epoca_t.push_back(RowVector(1));
323         double aux, aux_t;
324         aux = 0; //defino nuevos auxiliares para los proximos
               calculos
325         aux_t = 0;
326         for (int p = 0; p < dim1; p++) {
327             RowVector in = Vector_entrenamiento_in[p];
328             RowVector d_o(1);
329             RowVector out = Red_Neuronal.evalua_red(in);
330             d_o(0) = Vector_entrenamiento_d_o[p](0);
331             for (int k = 0; k < 3; k++) { //k=iteraciones internas
332                 Red_Neuronal.correccion_BackPropagation(out, d_o);
333                 out = Red_Neuronal.evalua_red(in);
334             }
335         }       //una vez que pase por todos los patrones vuelvo a
               evaluar en cada epoca a los patrones del entrenamiento y
               saco el vector diferencias
336         #pragma omp parallel for firstprivate(Red_Neuronal)
337         for (int p = 0; p < dim1; p++) {
338             RowVector in = Vector_entrenamiento_in[p];
339             RowVector d_o(1);
340             RowVector out = Red_Neuronal.evalua_red(in);
341             d_o(0) = Vector_entrenamiento_d_o[p](0);
342             Vector_diferencia[p](0) = abs(out(0) - d_o(0));
343             Matriz_salidas_entrenamiento(p,1)=out(0);
344             aux += Vector_diferencia[p](0);
```

```
345         }
346         //con esos pesos corregidos hago el mismo procedimiento con
               los testigos
347         #pragma omp parallel for firstprivate(Red_Neuronal)
348         for (int p = 0; p < dim1t; p++) {
349             RowVector in_t = Vector_testigos_in[p];
350             RowVector d_o_t(1);
351             RowVector out_t = Red_Neuronal.evalua_red(in_t);
352             d_o_t(0) = Vector_testigos_d_o[p](0);
353             Vector_diferencia_t[p](0) = abs(out_t(0) - d_o_t(0));
354             Matriz_salidas_testigos(p,1) = out_t(0);
355             aux_t += Vector_diferencia_t[p](0);
356         }
357         Vector_error_epoca[t](0) = aux / dim1; //saco el promedio de
               los errores de cada epoca tano para el entrenamiento como
               para los testigos
358         Vector_error_epoca_t[t](0) = aux_t / dim1t;
359         //ingreso los valores obtenidos en una matriz
360         Matriz_promedio(t, 0) = t + 1; //porque t empieza en 0
361         Matriz_promedio(t, 1) = Vector_error_epoca[t](0);
362         Matriz_promedio(t, 2) = Vector_error_epoca_t[t](0);
363       if (t % feedback == 0) {//guardo la matriz en un txt si epoca
            es multiplo de feedback
364           char filename_promedios[100];
365           snprintf(filename_promedios, 100, "Promedios_%d.txt",
                 codigo);
366           fstream out(filename_promedios, fstream::out);
367           out << Matriz_promedio << endl;
368           cout.precision(10);
369           Matrix Matriz_salidas(dim1_data, 1);//Evaluacion de todos
                 los patrones con la red neuronal
370           for (int p = 0; p < dim1_data; p++) {
371               RowVector in = Vector_data_in[p];
372               RowVector d_o_d(1);
373               RowVector out_d = Red_Neuronal.evalua_red(in);
374               Vector_data_out[p] = out_d;
375               Matriz_salidas(p, 0) = Vector_data_out[p](0);
376           }
377           out.close();
```

```
378                 Red_Neuronal.guarda_pesos_bias(codigo, t);//guardo los
                        pesos obtenidos
379                 Red_Neuronal.guarda_out(t, codigo, Matriz_salidas);
380                 Red_Neuronal.guarda_out_entrenamiento(t, codigo,
                        Matriz_salidas_entrenamiento);
381                 Red_Neuronal.guarda_out_testigos(t, codigo,
                        Matriz_salidas_testigos);
382             }
383         }
384         return 0;
385     }
```

ANEXO B

Código desarrollado para el Entrenamiento con SA

```
1     //Algoritmo Metropolis adaptado;
2      int epoca, temperatura, casos;
3      //epoca = data(1);
4      //casos = data(2);
5      epoca=1000;
6      casos=10;
7      temperatura = 1000;
8      cout << "epoca " << epoca << " temperatura " << temperatura <<
          endl;
9      Matrix Matriz_promedio(epoca,4);
10     //evaluo red inicial:
11     double aux = 0.00;
12     for (int p = 0; p < dim1; p++) {
13              RowVector in = Vector_entrenamiento_in[p];
14              RowVector d_o_d(1);
15              d_o_d(0)=Vector_entrenamiento_d_o[p](0);
16              RowVector out_d = Red_Neuronal.evalua_red(in);
17              Vector_data_out[p] = out_d;
18              RowVector diferencia(1);
19              diferencia(0) = abs(out_d(0) - d_o_d(0));
20              aux = aux+ diferencia(0);
21         }
22     double promedio;
23     promedio= aux/dim1;
24  //-------------------------------------------------------------------------

25         for (int t=1;t<epoca;t++) {
26          Matrix Matriz_salidas2(dim1,3);
27          Matrix Matriz_salidas3(dim1t,3);
```

```
28          Matrix Matriz_salidas4(dim1,3);
29          temperatura =temperatura *0.99;
30          double k=0.0000138;
31          int iexitos=0;
32          int icasos=0;
33          while (icasos<100*casos && iexitos<10*casos){
34          icasos++;
35          int eleccion;
36              for (int i = ncapas - 2; i >= 0; i--) {
37                  for (octave_idx_type j = 0; j < (a(i)); j++) {
38                          for (octave_idx_type k = 0; k < a(i + 1); k
                               ++) {
39                          Red_Neuronal.pesos_actual[i](j, k) =
                               Red_Neuronal.pesos[i](j, k);
40                          eleccion= rand()%100;
41                          if (eleccion>50){
42                          Red_Neuronal.pesos[i](j, k) = Red_Neuronal.
                               pesos[i](j, k)+double (rand()%100) / (100*
                               t); //defino los pesos
43                          }
44                          else{
45                          Red_Neuronal.pesos[i](j, k) = Red_Neuronal.
                               pesos[i](j, k)-double (rand()%100) / (100*
                               t); //defino los pesos
46                          }
47                      }
48                  }
49                  for (octave_idx_type j = 0; j < (a(i + 1)); j++) { //
                       defino los bias para los lugares posteriores
50                   Red_Neuronal.bias_actual[i](0, j) = Red_Neuronal.
                        bias[i](0, j);
51                   eleccion= rand()%100;
52                   if (eleccion>50){
53                   Red_Neuronal.bias[i](0, j) = Red_Neuronal.bias[i
                        ](0, j)+double (rand()%100) / (100*t);
54                   }
55                   else{
56                       Red_Neuronal.bias[i](0, j) = Red_Neuronal.
                            bias[i](0, j)-double (rand()%100) / (100*t
```

```
                        );
57                  }
58              }
59          }
60      //Evaluacion de los patrones con la red neuronal;
61       double aux2 = 0.00;
62       double aux3 = 0.00;
63       double aux4 = 0.00;
64      #pragma omp parallel for firstprivate(Red_Neuronal)
65              for (int p = 0; p < dim1; p++) {
66              RowVector in = Vector_entrenamiento_in[p];
67              RowVector d_o_d(1);
68              RowVector diferencia_entrenamiento(1);
69              d_o_d(0)=Vector_entrenamiento_d_o[p](0);
70              RowVector out_d2 = Red_Neuronal.evalua_red(in);
71              Vector_entrenamiento_out[p] = out_d2;
72              diferencia_entrenamiento(0) = abs(out_d2(0) - d_o_d
                    (0));
73              Matriz_salidas2(p, 0) = out_d2(0);
74              Matriz_salidas2(p, 1)= d_o_d(0);
75              Matriz_salidas2(p, 2)= diferencia_entrenamiento(0);
76              aux2 += diferencia_entrenamiento(0);
77          }
78          double promedio_actual, delta_e, probabilidad;
79          promedio_actual = aux2 / dim1;
80          delta_e = (promedio_actual-promedio);
81          probabilidad = exp(-delta_e/(temperatura*k));
82          double var_probabilidad;
83          var_probabilidad=(rand()%100)/100;
84          if(probabilidad>var_probabilidad){
85              promedio = promedio_actual;
86              iexitos=iexitos+1;
87                              #pragma omp parallel for firstprivate
                                    (Red_Neuronal)
88                                  for (int p = 0; p < dim1_data
                                        ; p++) {
89                                  RowVector in = Vector_data_in
                                        [p];
90                                  RowVector d_o_d(1);
```

```
91                                      RowVector diferencia_data(1);
92                                      d_o_d(0)=Vector_data_d_o[p
                                           ](0);
93                                      RowVector out_d3 =
                                           Red_Neuronal.evalua_red(in
                                           );
94                                           Vector_data_out[p] =
                                                out_d3;
95                                      diferencia_data(0) = abs(
                                           out_d3(0) - d_o_d(0));
96                                      Matriz_salidas3(p, 0) =
                                           out_d3(0);
97                                      Matriz_salidas3(p, 1)= d_o_d
                                           (0);
98                                      Matriz_salidas3(p, 2)=
                                           diferencia_data(0);
99                                      aux3 += diferencia_data(0);
100         }
101                              for (int p = 0; p < dim1t; p++) {
102           RowVector in = Vector_testigos_in[p];
103            RowVector d_o_t(1);
104            RowVector diferencia_testigos(1);
105            d_o_t(0)=Vector_testigos_d_o[p](0);
106            RowVector out_d4 = Red_Neuronal.evalua_red(in);
107         Vector_testigos_out[p] = out_d4;
108            diferencia_testigos(0) = abs(out_d4(0) - d_o_t(0));
109            Matriz_salidas4(p, 0) = out_d4(0);
110            Matriz_salidas4(p, 1)= d_o_t(0);
111            Matriz_salidas4(p, 2)= diferencia_testigos(0);
112            aux4 += diferencia_testigos(0);
113                            }
114                            double promediototal=aux3/dim1_data;
115                            cout<<"promedio tot"<<promediototal<<
                                 endl;
116                            double promedio_entrenamiento=
                                 promedio_actual;
117                            double promedio_testigos=aux4/dim1t;
118                            Matriz_promedio(t-1,0)=t;
119                            Matriz_promedio(t-1,1)=promediototal;
```

```
120                                 Matriz_promedio(t-1,2)=
                                        promedio_entrenamiento;
121                                 Matriz_promedio(t-1,3)=
                                        promedio_testigos;
122                                 cout<<promediototal<<
                                        promedio_entrenamiento<<
                                        promedio_testigos;
123                             }
124                             else{
125                 for (int i = ncapas - 2; i >= 0; i--) {
126                     for (octave_idx_type j = 0; j < (a(i)); j++) {
127                         for (octave_idx_type k = 0; k < a(i + 1); k
                                ++) {
128                             Red_Neuronal.pesos[i](j, k) =
                                    Red_Neuronal.pesos_actual[i](j, k);
129                         }
130                     }
131                     for (octave_idx_type j = 0; j < (a(i + 1)); j++)
                            {
132                         Red_Neuronal.bias[i](0, j) = Red_Neuronal.
                                bias_actual[i](0, j);
133                     }
134                 }
135             }
136         }//guardo los pesos obtenidos
137         char filename_promedios[100];
138         snprintf(filename_promedios,100,"promedios_%d.txt",codigo);
139         fstream out_promedios(filename_promedios,fstream::out);
140         out_promedios<<Matriz_promedio<<endl;
141         out_promedios.close();
142         Red_Neuronal.guarda_out(t, codigo, Matriz_salidas3);
143         Red_Neuronal.guarda_pesos_bias(codigo, t);
144         Red_Neuronal.guarda_out_entrenamiento(t, codigo,
                Matriz_salidas2);
145   }
146 return 0;
147 }
```