REDES NEURONALES ARTIFICIALES PARA RESOLVER PROBLEMAS DE PROGRAMACIÓN LINEAL

Prof. Tutor: Dr. Eliezer Colina M.
Autor: Br. Argenis A. Moreno V.

Proyecto presentado ante la ilustre Universidad de los Andes como requisito final para optar al título de Ingeniero de Sistemas.

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE SISTEMAS

Junio - 2005.

A mi hermana Marggory, ejemplo
de esfuerzo, sacrificio y humildad...

AGRADECIMIENTOS

A Dios Todo Poderoso, por darme la salud, paciencia y fuerza necesaria para la realización de este trabajo.

A mi Tutor, Dr. Eliezer Colina, por prestarme en todo momento su invalorable ayuda y conocimientos para llevar a cabo este proyecto. Mil Gracias Profesor!!.

RESUMEN

El objetivo principal de este trabajo es mostrar de forma práctica el potencial que tienen las Redes Neuronales Artificiales (RNA) en la resolución problemas de programación lineal (P.L.). Para éste propósito se hace uso de dos modelos distintos de RNA, de acuerdo a su arquitectura de conectividad: el primer modelo es el de las Redes Perceptrónicas Multicapa con conexiones hacia adelante y el segundo modelo es el de las Redes Neuronales Recurrentes con conexiones hacia atrás. En el trabajo se comparan las bondades de ambos modelos respecto a la calidad de las soluciones generadas.

Para el modelo de Red Perceptrónica Multicapa (RPM) se diseñan 4 configuraciones distintas de red, variando el número de neuronas en la capa oculta y usando distintas funciones de activación para cada configuración. Se utilizan como datos de entrenamiento 120 problemas de P.L. de 2 variables y 3 restricciones, previamente resueltos mediante el método simplex con soluciones óptimas no vacías y acotadas. De éstos 120 problemas se utilizan 20 para la validación de la red.

En el modelo de Red Neuronal Recurrente (RNR), se transforma el problema de P.L. en un sistema de ecuaciones diferenciales no lineales que representan la RNR, cuyo estado estable representa la solución óptima del problema de P.L. Se resuelven ejemplos de 2 y 3 variables para comprobar el desempeño de la red en primera instancia. Luego se resuelven dos problemas reales de P.L. de 7 y 25 variables respectivamente.

Luego de evaluar los resultados, se encuentra que de los dos modelos expuestos, el de las RNR presenta un notable mejor desempeño en las soluciones obtenidas y se comprueba que el estado estable de la RNR representa la solución óptima a los problemas de P.L. Por otra parte, en el modelo de RPM ninguna de las configuraciones presentadas en este trabajo logra un desempeño aceptable y se obtienen errores no permisibles en las soluciones obtenidas durante la fase de validación.

TABLA DE CONTENIDO

INTRODUCCIÓN

El desarrollo de la programación lineal (P.L.) se sitúa entre los avances científicos más importantes del siglo XX [1], siendo en nuestros días una herramienta de uso normal y que ha permitido el ahorro de muchos recursos financieros a empresas e industrias alrededor de todo el mundo. Ahora bien, ¿cual es la esencia de esta herramienta?, se puede decir en forma breve que consiste en la asignación de recursos limitados entre actividades interdependientes y competitivas de la mejor forma posible, es decir, de forma óptima.

El método más usado a lo largo de los años para resolver los problemas de P.L. es el M*étodo Simplex* desarrollado por George B. Dantzig que ha demostrado ser extremadamente eficiente, no obstante sus dificultades de implementación a nivel computacional para resolver problemas de gran escala [2]. Este método es algebraico; sin embargo, sus conceptos fundamentales son geométricos, por eso es fácil de entender e implementar a nivel de software.

Sin embargo, debido a la NP-completitud de los problemas de P.L., durante los últimos años, varias técnicas "emergentes" han surgido para resolverlos de forma eficiente. Una de éstas técnicas está basada en el uso de ***Redes Neuronales Artificiales (RNA),*** las cuales constituyen un enfoque que intenta emular la funcionalidad inteligente de las redes neuronales en los sistemas biológicos y han sido utilizadas exitosamente en diversos problemas de optimización.

Existen varias definiciones de las RNA, sin embargo se puede decir que son modelos matemáticos conformados por un gran número de elementos de procesamiento (neuronas) que están interconectados masivamente y que tienen una organización jerárquica [3], los cuales intentan interactuar con los objetos del mundo real al igual que lo hace el cerebro humano para conseguir resolver problemas relacionados con el reconocimiento de patrones, predicción, codificación, control, optimización entre otros.

Así pues, éste trabajo se ha enfocado en la metodología de las RNA para resolver problemas de P.L., basados en su estructura topológica de conectividad: el primero de ellos basado en conexiones hacia adelante (*feedforward*) y el segundo basado en conexiones hacia atrás o recurrentes (*feedback*).

En el Capítulo 1 se da una breve introducción al marco general conceptual de las RNA. Seguidamente en el Capítulo 2 se describen las Redes Perceptronicas Multicapa con conexiones hacia adelante y el principal algoritmo utilizado para entrenar este tipo de redes, el de Retropropagación del Error. Más adelante se propone, se entrenan y se validan varias configuraciones de RPM para resolver problemas de P.L. Por último, en el Capítulo 3 se describen a las Redes Neuronales Recurrentes y los fundamentos teóricos para convertir un problema de P.L. en un sistema dinámico no lineal que representa la RNR y dar con la solución óptima del mismo.

●CAPITULO 1●

REDES NEURONALES ARTIFICIALES

1.1 INTRODUCCIÓN A LAS REDES NEURONALES ARTIFICIALES

Resulta contradictorio pensar que máquinas de cómputo, capaces de realizar 100 millones de operaciones por segundo, no sean capaces de entender el significado de las formas visuales o de distinguir entre distintas clases de objetos. Los sistemas de computación secuencial (computador Von Neumann) son exitosos en la resolución de problemas matemáticos, en la creación, manipulación y mantenimiento de bases de datos, en comunicaciones electrónicas, en el procesamiento de textos y gráficos, incluso en funciones de control de electrodomésticos, haciéndolos más eficientes y fáciles de usar, pero definitivamente tienen una gran incapacidad para interpretar el mundo.

Es por ello que muchos investigadores [4, 5, 6, 7] han volcado sus esfuerzos en los últimos 30 años al desarrollo de nuevos sistemas que permitan tratar ésta incapacidad que poseen las máquinas secuenciales, y qué mejor manera que fijar la vista en la super computadora que posee el ser humano: el cerebro.

El cerebro es un procesador de información con unas características muy notables: es capaz de procesar a gran velocidad grandes cantidades de información procedentes de los sentidos, combinarla o compararla con la información almacenada y dar respuestas adecuadas incluso en situaciones nuevas. Al mismo tiempo, el cerebro logra diferenciar un susurro en una sala ruidosa, distinguir una cara en una calle mal iluminada o leer entre líneas una declaración política; pero lo más impresionante de todo es su capacidad de aprender a representar la información necesaria para desarrollar tales habilidades sin instrucciones explícitas para ello.

Aunque todavía se ignora mucho sobre la forma en que el cerebro aprende a procesar la información, se han desarrollado modelos que tratan de emular tales habilidades denominados

Redes Neuronales Artificiales. La construcción de estos modelos supone, en primer lugar, la abstracción de las características esenciales de las neuronas biológicas y sus conexiones, y en segundo lugar, la implementación del modelo en una computadora de forma que se pueda simular. Las RNA se basan en el aprendizaje de estrategias de solución basadas en ejemplos de comportamiento típico de patrones [3]; estos sistemas no requieren que la tarea a ejecutar se programe, ellos generalizan y aprenden de la experiencia. Es evidente que estos modelos son idealizaciones simples de las redes neuronales biológicas, sin embargo resultan interesantes por sus capacidades de aprendizaje.

Los modelos RNA han brindado una alternativa emergente para aquellos problemas en los cuales los métodos clásicos no han entregado resultados muy convincentes, o poco eficientes. Las aplicaciones más exitosas de las RNA son:

1. Procesamiento de imágenes y de voz
2. Reconocimiento de patrones
3. Planeamiento
4. Predicción
5. Control y optimización
6. Filtrado de señales

Las RNA se respaldan sobre una teoría que aún está en proceso de desarrollo y cuyo verdadero potencial no se ha alcanzado todavía ya que las representaciones y procedimientos de que se sirve el cerebro son aún desconocidas. Tarde o temprano muchos datos empíricos concernientes al funcionamiento del cerebro comenzarán a adquirir sentido y se tornarán factibles muchas aplicaciones desconocidas de las redes neuronales.

1.2 UN POCO DE HISTORIA

En 1936, Alan Turing comenzó a estudiar el cerebro como una forma de ver el mundo de la computación; sin embargo los primeros teóricos que concibieron los fundamentos de la computación neuronal fueron Warren McCulloch, un neurofisiólogo, y Walter Pitts, un

matemático, quienes en 1943 lanzaron una teoría acerca de la forma de trabajar de las neuronas.

En 1956 se organizó en Dartmouth la primera conferencia sobre Inteligencia Artificial. Aquí se discutió el uso potencial de las computadoras para simular "todos los aspectos del aprendizaje o cualquier otra característica de la inteligencia" y se presentó la primera simulación de una red neuronal, aunque todavía no se sabían interpretar los datos resultantes.

En 1957, Frank Rosenblat comenzó el desarrollo del Perceptron. El Perceptron es la más antigua red neuronal artificial y se usa en la actualidad en varias aplicaciones como reconocedor de patrones. Este modelo es capaz de generalizar, es decir, después de haber aprendido una serie de patrones, puede identificar otros similares aunque no se le hayan presentado anteriormente.

En 1959, Bernard Widrow y Marcial Hoff desarrollaron el modelo ADALINE**. E**sta fue la primera red neuronal artificial aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas) y se ha usado comercialmente durante varias décadas.

En 1969 surgieron numerosas críticas que frenaron hasta 1982 el crecimiento que estaban experimentando las investigaciones sobre RNA. Marvin Minsky y Seymour Papert, del Instituto Tecnológico de Massachusetts (MIT), publicaron un libro, *Perceptrons*, que contenía un análisis matemático detallado del Perceptron.

Para 1982, coincidieron numerosos eventos que hicieron levantar el interés por las RNA. John Hopfield presentó su trabajo sobre redes neuronales en la Academia de Ciencias. En dicho trabajo se describe con claridad y rigor matemático una red a la que ha dado su nombre y mostró como ésta trabajaba y qué podía hacer.

En 1985, El Instituto Americano de Física comenzó lo que ha sido la reunión anual *Neural Networks for Computing*. En 1987, se formó la *International Neural Network Society*, y en

menos de dos años tenía más de 3000 socios. A partir de este momento, el interés por esta área se ha ido incrementando de forma notable como lo demuestran el número de congresos y reuniones científicas, la gran cantidad de revistas publicadas y el creciente número de empresas que día tras día incorporan la utilización de esta tecnología para el desarrollo de aplicaciones concretas.

1.3 FUNDAMENTOS DE LAS REDES NEURONALES ARTIFICIALES

La teoría y el modelado de las RNA están inspirados en la estructura y funcionamiento del cerebro humano, donde la neurona es el elemento principal. Existen neuronas de diferentes tamaños, estas características son importantes para determinar la función y utilidad de la neurona. La clasificación de las neuronas ha sido realizada por muchos neuroanatomistas.

1.3.1 DEFINICIÓN Y ESTUCTURA DE UNA NEURONA.

Una neurona es una célula viva y como tal contiene todos los elementos presentes en las células biológicas. Pero además de ello, contiene elementos que las caracterizan y las diferencian de las demás. Una neurona consta de un cuerpo celular de forma esférica de 5 a 10 micras de diámetro, del que salen una rama principal que se llama axón y varias ramas mas cortas llamadas dendritas.

La principal característica que posee la neurona es que se puede comunicar con sus similares. En forma resumida las dendritas reciben las señales de entrada, el cuerpo celular (núcleo) las combina e integra y emite señales de salida, el axón es el encargado de transportar estas señales de salida a un nuevo conjunto de neuronas. Por lo general una neurona recibe información de miles de neuronas y ésta a su vez la transmite a otras miles más, se calcula que el cerebro humano posee 10^{15} neuronas.

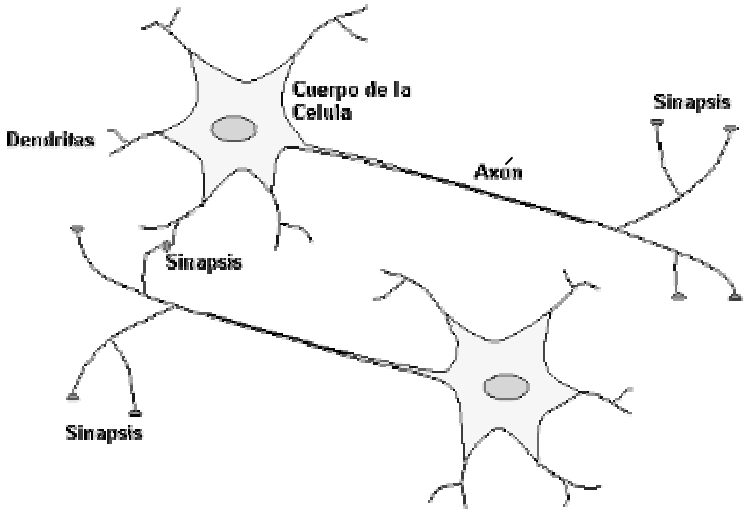


Fig. 1.1 Neurona Biológica

1.3.2 LA SINAPSIS.

Las señales a las que se refiere son de dos tipos: eléctricas y químicas. La señal generada por la neurona y transportada a lo largo del axón es un impulso eléctrico, mientras que la señal que se transmite entre los terminales axónicos de una neurona y las dendritas de las neuronas siguientes es de origen químico. Este proceso se llama Sinapsis, y es donde ocurre la transmisión de información entre las distintas neuronas. La longitud de la sinapsis viene dada por un complejo proceso químico.

Existen 2 tipos de sinapsis:

a) Sinapsis excitadoras: las cuales facilitan la generación de impulsos a mayor velocidad.
b) Sinapsis inhibidoras: las cuales dificultan la generación de impulsos.

Casi todas las neuronas reciben entradas procedentes de sinapsis excitadoras e inhibidoras, por tal motivo, en cada instante, algunas de ellas estarán activas y otras en reposo, la suma total de los efectos tanto excitadores como inhibidores determinará si la neurona es activada o no, es decir, si emitirá una respuesta.

1.4 REDES NEURONALES ARTIFICIALES.

1.4.1 CONCEPTO.

Existen numerosas formas de definir lo que son las RNA, sin embargo se puede decir que son modelos matemáticos conformados por un gran número de elementos de procesamiento que están interconectados masivamente y que tienen una organización jerárquica [3], los cuales intentan interactuar con los objetos del mundo real al igual que lo hace el cerebro humano para conseguir resolver problemas relacionados con el reconocimiento de patrones, predicción, codificación, control, optimización entre otros.

En las RNA, los *elementos de procesamiento* corresponden a las neuronas biológicas, las interconexiones se realizan por medio de las ramas de salida (axones) que producen un número variable de conexiones (sinapsis) con otras neuronas.

1.4.2 LA NEURONA ARTIFICIAL.

La neurona artificial es la que pretende emular las características principales de las neuronas biológicas. Haciendo uso de las figuras siguientes, analizaremos esas similitudes:

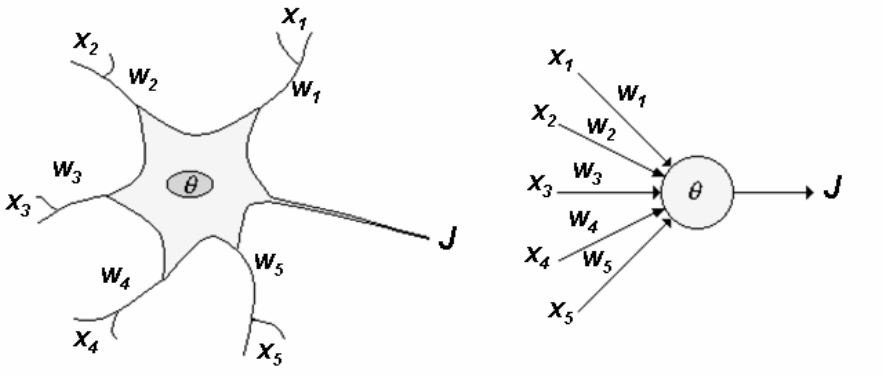


Fig. 1.2 De la Neurona biológica a la neurona artificial

a) Las entradas *Xi* representan todas las entradas que provienen de las demás neuronas y que son capturadas por la dendritas.

b) *Wi* son los pesos sinápticos, representan todas las conexiones entre todas las neuronas en la red.

c) θ es el umbral que debe sobrepasar toda neurona para poder activarse y generar una respuesta.

d) *J* es la salida que ofrece la neurona a través del axón.

De tal forma que el funcionamiento de la neurona artificial es simple, su función es recibir información del exterior o de otras neuronas vecinas y calcular un valor de salida el cual es transmitido a otras neuronas.

1.4.3 ESTADOS DE ACTIVACIÓN.

Todas las neuronas que componen la red se hallan en cierto estado; se puede decir, en forma simplificada que existen sólo dos estados: el de reposo y el de excitación, y cada uno de ellos tiene asociado un valor que puede ser discreto o continuo. Si los valores son discretos normalmente se toman valores binarios, entonces si la neurona está activada tomará el valor de 1 y si está en reposo tomará el valor de 0. Para el caso continuo se considera un conjunto continuo de estados de activación, normalmente en un intervalo entre [-1,1].

Ahora se debe conocer qué criterio siguen las neuronas para alcanzar tales estados de activación, principalmente va a depender de 2 factores, el primero de ellos es el peso o magnitud de la conexión entre las neuronas involucradas y el segundo, es que las señales que envía cada una de las neuronas a sus vecinas estarán influenciadas por su propio estado de activación.

1.4.4 CONEXIONES ENTRE NEURONAS.

Las conexiones que unen a las neuronas que forman una RNA tienen asociado un peso, el cual hace que la red adquiera conocimiento. Consideremos *Yi* como el valor de salida de la neurona *i* en un instante dado. Cada conexión (sinapsis) ente la neurona *i* y la neurona *j* está ponderada por un peso *Wij*. Normalmente se considera que el efecto de cada señal es aditivo, de tal forma que la entrada neta que recibe una neurona net_j es la suma del producto de cada señal individual por el valor de la sinapsis que conecta ambas neuronas [8].

$$net_j = \sum_{i}^{N} W_{ij} y_i \qquad (1.1)$$

Esta regla muestra el proceso a seguir para combinar los valores de entrada a una neurona con los pesos de las conexiones que llegan a esa neurona y es conocida como regla de propagación.

1.4.5 FUNCIÓN DE ACTIVACIÓN.

Es la regla que se requiere para combinar las entradas a la neurona y su estado actual para producir un nuevo estado de activación. En otras palabras esta función *F* produce un nuevo estado de activación en una neurona a partir del estado que existía y la combinación de las entradas con los pesos correspondientes a sus conexiones (net_j).

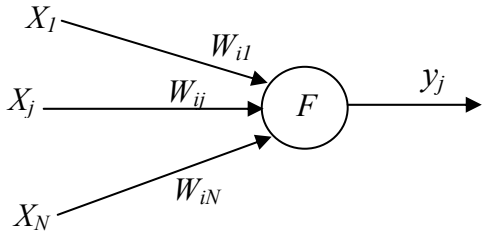


Figura 1.3 Combinación de entradas con los pesos sinápticos

Entre las funciones de activación más comunes tenemos:

- Función Escalón
- Función Lineal
- Función Sigmoidal
- Función Hiperbólica Tangencial
- Función Lineal Mixta

Función Escalón.

Es más frecuente utilizada que la función lineal. Con la función escalón la unidad envía señales de salida sólo cuando su activación es mayor o igual que cierto valor umbral. (Fig. 1.4) La respuesta de salida será binaria o discreta: sólo 1 ó 0 (o bien 1 y -1 si utilizamos otra notación), y dependerá de si el valor de activación (que en este caso es un valor continuo) supera cierto umbral.

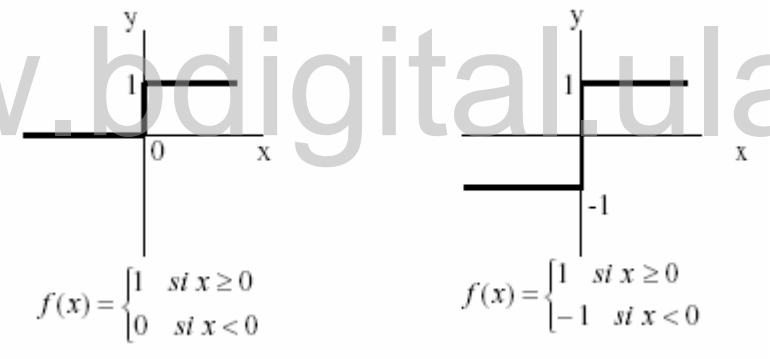
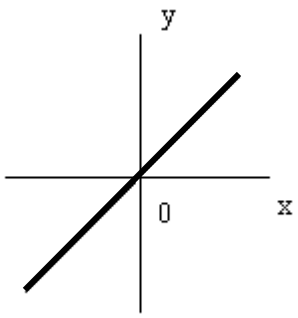


Figura 1.4 Función Escalón

- **Función Lineal.**

Es la más sencilla y la de menos uso. Se llama función de identidad porque la salida correspondiente de la neurona es igual a su estado de activación. En realidad esta función equivale a no aplicar función de salida. (Fig. 1.5)

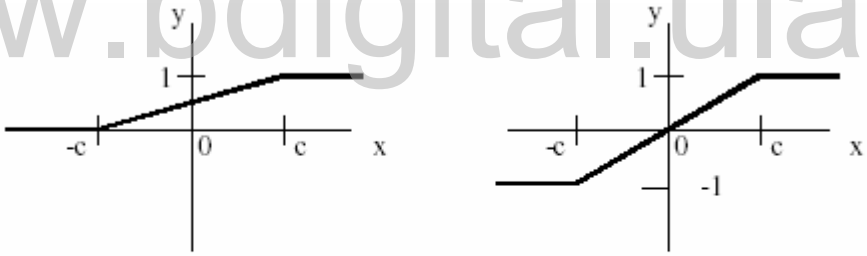


$f(x) = x$

Figura 1.5 Función Lineal

- **Función Mixta.**

Con esta función, si la activación es menor que un límite inferior, la salida es 0 (ó -1). Si la activación es mayor o igual que el límite superior, entonces la salida es 1. Si la activación está entre los límites, la salida es una función lineal de la activación. (Fig 1.6)



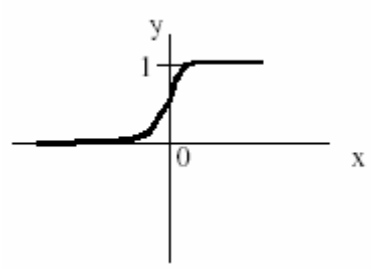
$$f(x) = \begin{cases} 0 & si\ x < -c \\ 1 & si\ x > c \\ x/(2c) + 1/2 & en\ otro\ caso \end{cases} \qquad f(x) = \begin{cases} -1 & si\ x < -c \\ 1 & si\ x > c \\ ax & en\ otro\ caso \end{cases}$$

Figura 1.6 Función Mixta.

- **Función Sigmoidal.**

Es una de las funciones de activación más utilizadas. Produce salidas continuas y proporcionales al nivel de activación de la neurona dentro del rango [0,1], por lo que su salida máxima será 1 y la mínima 0. Cuando el nivel de activación supere al umbral máximo la salida

seguirá siendo 1 y cuando el nivel de activación sea inferior al umbral mínimo la salida seguirá siendo 0. (Fig 1.7)

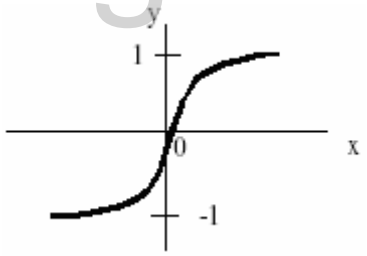


f(x)=1/1+exp(x)

Figura 1.7 Función Sigmoidal

- **Función Tangencial Hiperbólica.**

Es similar a la función sigmoidal pero en este caso la respuesta es 1 para el valor máximo y -1 para el mínimo. (Fig. 1.8)



f(x)=tanh(x
)

Figura 1.8 Función Tangencial Hiperbólica

1.4.6 REGLA DE APRENDIZAJE.

En el campo biológico se suele aceptar que la información memorizada en el cerebro está más asociada a los valores sinápticos de las conexiones entre las neuronas que con ellas mismas, es decir, el conocimiento y el aprendizaje se encuentran en la sinapsis [9]. Ahora bien, en el caso

de las RNA ocurre algo similar, el conocimiento se encuentra almacenado en los pesos de las conexiones entre las neuronas, por tanto, todo proceso de aprendizaje conduce cambios en estas conexiones, en otras palabras un RNA aprende modificando los valores de los pesos de ella misma.

El aprendizaje en las RNA se realiza mediante patrones de ejemplo, siendo dos los tipos de aprendizaje: supervisado y no supervisado.

APRENDIZAJE SUPERVISADO.

En este tipo de aprendizaje se le proporciona a la red parejas de datos entrada-salida y luego la red aprende a asociarlos. Existen 3 tipos de aprendizaje supervisado:

Aprendizaje por corrección de error

Consiste en ajustar los pesos de las conexiones de la red en función del error que se comete entre la salida deseada y la salida obtenida por la red.

Aprendizaje por refuerzo

Es un aprendizaje supervisado más lento que el anterior y consiste en indicar mediante una señal de refuerzo si la salida obtenida por la red se acerca o se ajusta a la salida deseada, y en función de ello se ajustan los pesos sinápticos.

Aprendizaje estocástico

Este tipo de aprendizaje consiste en realizar cambios aleatorios a los pesos de la red y medir el efecto a través de distribuciones de probabilidad.

APRENDIZAJE NO SUPERVISADO.

Cuando el aprendizaje es no supervisado, se suministra a la red únicamente patrones de datos de entrada, de tal forma que extraiga las características esenciales de los mismos. Las RNA

entrenadas por esta vía deben encontrar las correlaciones que se pueden establecer entre los datos de entrada. De esta forma, por un lado, las salidas de la red representan el grado de similitud entre las entradas y las informaciones que se les ha mostrado a la red hasta entonces. En otro caso establecen categorías, indicando a la salida a qué categoría pertenece la información presentada en la entrada, siendo la propia red quien debe encontrar las categorías apropiadas a partir de correlaciones entre las informaciones presentadas.

1.4.7 ESTRUCTURA DE UNA RED NEURONAL ARTIFICIAL.

La estructura de una red neuronal se fundamenta en función de:

A) Número de niveles o capas

B) Forma de conexión entre las neuronas

A) NIVELES O CAPAS.

De entrada: Es la capa que recibe toda la información proveniente de fuentes externas a la red.

Oculta: Es la capa que se encuentra interna en la red y no tiene contacto directo con el exterior. El número de capas en una RNA va desde cero hasta un número elevado, las neuronas en esta capa pueden estar conectadas de distintas manera lo que da como resultado las distintas topologías de conexión que tienen las RNA.

De Salida: Es la capa que transfiere la información hacia el exterior de la red.

B) FORMAS DE CONEXIÓN ENTRE NEURONAS.

Existen dos formas de conexión entre neuronas: cuando ninguna salida de las neuronas es entrada de otras neuronas del mismo nivel o de ellas mismas, hablamos de conexión hacia delante (*feedforward*). Cuando la salidas son conectadas como entradas hacia otras neuronas de niveles previos o del mismo nivel entonces hablamos de conexión hacia atrás o recurrente (*feedback*).

●CAPÍTULO 2●

REDES PERCEPTRÓNICAS MULTICAPA

2.1 REDES PERCEPTRÓNICAS MULTICAPA CON CONEXIONES HACIA ADELANTE.

En ésta sección se estudiará una clase importante de RNA, las llamadas Redes Neuronales con conexiones hacia adelante o como se les conoce comúnmente: **Redes Perceptrónicas Multicapa (RPM).**

Básicamente, éstas redes están formadas por un conjunto de neuronas sensoras que constituyen lo que se conoce como capa de entrada, luego una o más capas ocultas de neuronas de *procesamiento* y una capa de neuronas de salida que son las que ofrecen la información resultante del comportamiento de la red. Los patrones de entrada se *propagan* a través de la red en dirección hacia adelante y capa por capa.

Las RPM han sido utilizadas para resolver una gran diversidad de problemas del mundo real, siguiendo un entrenamiento supervisado a través del muy conocido algoritmo de Retropropagación del Error o Regla Delta Generalizada.

El proceso del Algoritmo de Retropropagación del Error consiste en 2 fases que se llevan a cabo en todas las capas de la red: una primera fase llamada *hacia adelante* y la segunda llamada *hacia atrás*. En la fase hacia adelante un vector de patrones de entrada es aplicado a las neuronas sensoras (capa de entrada) y su efecto se propaga a través de la red capa por capa, produciendo un conjunto de salidas que indican la respuesta actual de la red. Durante la primera fase todos los pesos de las conexiones de la red están fijos, por el contrario en la segunda fase los pesos son ajustados de acuerdo a la regla de corrección del error. Para ser más claros, la respuesta actual de la red (salida obtenida) es restada de la respuesta o salida deseada, produciéndose una señal de error. La señal de error es *propagada hacia atrás* a través

de la red, en dirección contraria a los pesos sinápticos. De esta forma los pesos son ajustados para hacer que la salida actual de la red se parezca cada vez más a la salida deseada.

Una RPM posee tres características importantes: [8]

1. El modelo de cada neurona en la red incluye una forma de no linealidad en la salida de la neurona. El punto a resaltar es que la forma no lineal es suave, lo que indica que es derivable en cualquier parte de la función. Una forma comúnmente usada que satisface estas condiciones es la función sigmoidal $y_j = \frac{1}{1+e^{-net_j}}$; donde net_j corresponde a la suma ponderada de las entradas a la neurona j y y_j es la salida de la neurona. La presencia de la no linealidad es importante ya que de otra forma la relación entrada-salida de la red podría reducirse a una transformación lineal entre la entrada y salida. Además el uso de la función sigmoidal se debe a las propiedades intrínsecas de las neuronas biológicas [10].

2. La red contiene una o más neuronas ocultas que no son parte de las capas de entrada o de salida de la red. Las neuronas ocultas le permiten a la red aprender tareas más complicadas extrayendo de forma progresiva más características importantes y de mayor significado de los patrones o datos de entrada.

3. La red muestra un alto grado de conectividad, determinado por los pesos de las conexiones. Un cambio en la conectividad de la red requiere un cambio en el conjunto de dichos pesos.

Es por esta combinación de características, junto con la habilidad de aprender de la experiencia a través del entrenamiento, que las RPM derivan su gran poder para resolver diversos problemas de clasificación, predicción, entre otros. Sin embargo estas mismas características son las responsables de las deficiencias en los actuales momentos del conocimiento sobre el comportamiento de las RNA. Primero la presencia de formas no

lineales junto con la alta conectividad de la red hace que un análisis teórico profundo sobre las RPM sea dificultoso. Por otro lado el uso de neuronas ocultas hace que visualizar el proceso de aprendizaje sea complicado. En forma implícita el proceso de aprendizaje debe decir qué características o información de los datos de entrada deben ser representadas por las neuronas ocultas.

El desarrollo del algoritmo de retropropagación representa, si se puede decir, una conquista en el campo de las redes perceptrónicas ya que el algoritmo provee un método computacionalmente eficiente para el entrenamiento de las redes perceptrónicas. De todas maneras, no es válido decir que el algoritmo de retropropagación puede arrojar una solución para todos los problemas; sin embargo, si es válido decir que gracias a su uso se colocó a un lado el pesimismo que se dió en una época sobre el aprendizaje de las RPM.

2.2 ARQUITECTURA DE LAS REDES PERCEPTRONICAS.

En la figura 2.1 se muestra la forma de una RPM con 2 capas ocultas, que posee conexión completa; lo que quiere decir que una neurona en cualquier capa de la red esta conectada a todas las neuronas de la capa anterior. El flujo de información en la red se produce en dirección hacia adelante desde la izquierda hacia la derecha y capa por capa.

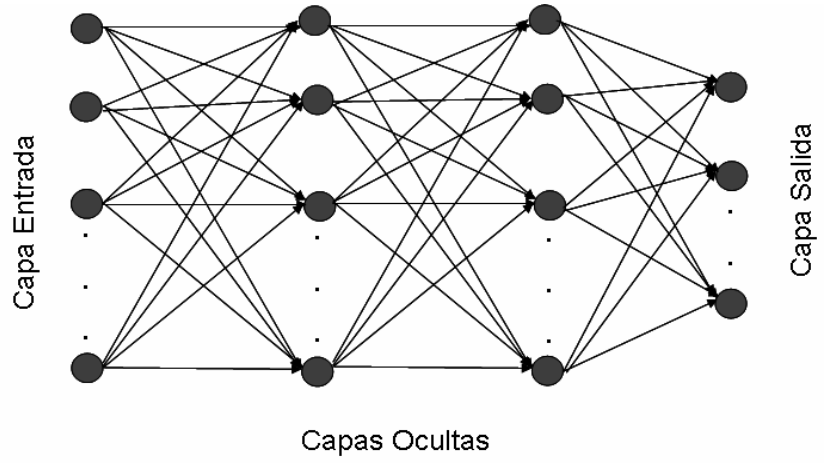


Figura 2.1 Red Perceptrónica con 2 capas

En la figura 2.2 se muestra una porción de una RPM. En ésta figura se identifica lo siguiente[8]:

1. Señales de entrada: Son aquellas que vienen desde la capa de entrada de la red y se propagan hacia adelante neurona por neurona a través de la red y emergen en la salida como la respuesta obtenida por la red.

2. Señales de error: Una señal de error se origina en las neuronas de la capa salida y se propagan hacia atrás capa por capa a través de la red.

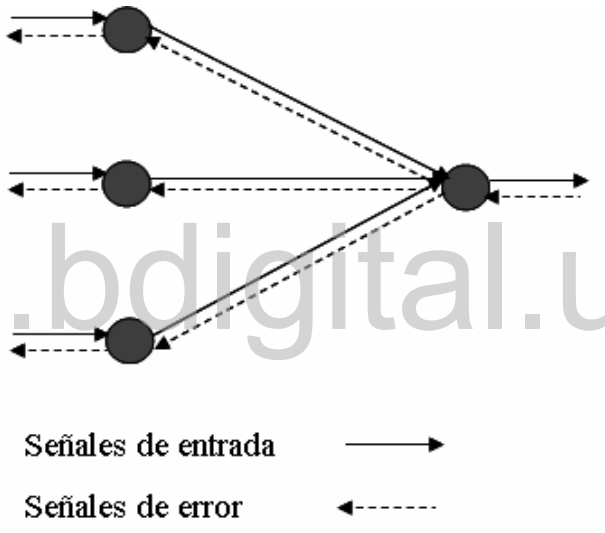


Figura 2.2 Pequeña porción de una Red Perceptrónica

Por otro lado cada neurona oculta o de salida de una RPM está diseñada para realizar lo siguiente:

1. El cálculo de las respuestas de las neuronas, el cual viene expresado como una función no lineal continua de las señales de entrada y los pesos de las conexiones asociados a esa neurona.

2. El cálculo del gradiente del error el cual se necesita para la fase de propagación hacia atrás a través de la red.

2.3 ALGORITMO DE RETROPROPAGACIÓN DEL ERROR O REGLA DELTA GENERALIZADA.

La regla delta (para redes monocapa), propuesta por Widrow en 1960, ha sido extendida para redes con capas ocultas y conexiones hacia adelante dando lugar al conocido algoritmo de retropropagación.

Las redes con capas ocultas presentan funciones de activación continuas bien sea lineales o no lineales (sigmoidales) y además son funciones decrecientes y derivables.

El algoritmo utiliza una función de error que está asociada a la red, y se encarga de buscar el estado de mínima energía o mínimo error a través del camino descendente de la superficie (cálculo del gradiente) que forma la función de error. Para ello, se retroalimenta el error que se produce entre la salida actual de la red y la salida deseada, con el fin de realizar los ajustes a los pesos en un valor proporcional al gradiente decreciente de la función de error.

De esta forma se define:

Señal de error: $e(p) = d_j(p) - y_j(p)$ *(2.1)*

donde: $d_j(p)$: salida deseada de la neurona *j* para *p-ésimo* patrón de entrenamiento
$y_j(p)$:salida actual de la neurona *j para p-ésimo* patrón de entrenamiento
La neurona *j* es una neurona de la capa de salida

Ahora se define el valor del error cuadrático para la neurona *j* como $\frac{1}{2}e_j(p)^2$. De esta forma, el valor del error *E* que corresponde a la suma de todos los errores cuadráticos $\frac{1}{2}e_j(p)^2$ de todas las neuronas de salida (las únicas visibles y de las cuales se puede obtener el error) es:

$$E(p) = \frac{1}{2}\sum_{j \in C} e_j(p)^2 \qquad (2.2)$$

Donde *C* corresponde a todas las neuronas en la capa de salida.

Por último se define el *Error Cuadrático Medio*, el cual se obtiene sumando el error cuadrático *E* para todo el conjunto de patrones de entrenamiento.

$$E_{medio} = \frac{1}{P}\sum_{p=1}^{P} E(p) \qquad (2.3)$$

FUNCIONAMIENTO DEL ALGORITMO.

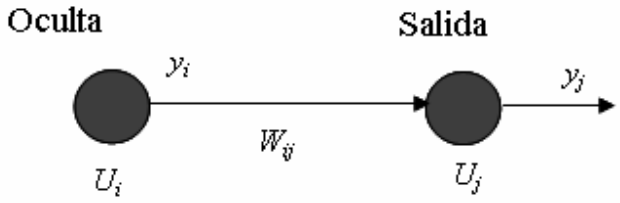


Figura 2.3 Neurona oculta y neurona de salida

El método que sigue la regla delta generalizada para ajustar los pesos consiste en actualizarlos de manera proporcional a la diferencia entre la salida deseada y la salida obtenida por la red. Esta diferencia queda definida por la ecuación 2.1.

De acuerdo a la Fig. 2.3 la entrada neta net_j que recibe la neurona U_j es:

$$net_j(p) = \sum_i^N W_{ij}(p) y_i(p) \qquad (2.4)$$

Donde N es el número total de entradas que se aplican a la neurona U_j.

Y la salida y_j correspondiente a la neurona U_j viene dada por:

$$y_j(p) = f(net_j(p)) \qquad (2.5)$$

De esta forma el algoritmo de retropropagación aplica un ajuste denotado por $\Delta W_{ji}(p)$ a los pesos de las conexiones $W_{ji}(p)$, el cual es proporcional al gradiente del error, denotado por $\partial E(p)/\partial W_{ji}(p)$. Usando la regla de la cadena, el gradiente queda:

$$\frac{\partial E(p)}{\partial W_{ji}(p)} = \frac{\partial E(p)}{\partial e_j(p)} \frac{\partial e_j(p)}{\partial y_j(p)} \frac{\partial y_j(p)}{\partial net_j(p)} \frac{\partial net_j(p)}{\partial W_{ji}(p)} \qquad (2.6)$$

El gradiente determina la dirección de búsqueda en el espacio de pesos, para el ajuste del peso de conexión W_{ji}.

Derivando ambos lados de la Ec. (2.2) con respecto a $e_j(p)$ tenemos:

$$\frac{\partial E(p)}{\partial e_j(p)} = e_j(p) \qquad (2.7)$$

Derivando ambos lados de la Ec. (2.1) con respecto a $y_j(p)$ tenemos:

$$\frac{\partial e_j(p)}{\partial y_j(p)} = -1 \qquad (2.8)$$

Derivando ambos lados de la Ec. (2.5) con respecto a net$_j(p)$ tenemos:

$$\frac{\partial y_j(p)}{\partial net_j(p)} = f'(net_j(p)) \qquad (2.9)$$

Finalmente derivamos ambos lados de la Ec. (2.4) con respecto a $W_{ji}(p)$ tenemos:

$$\frac{\partial net_j(p)}{\partial W_{ji}(p)} = y_i(p) \qquad (2.10)$$

Ahora sustituimos las Ec. (2.7), Ec.(2.8), Ec. (2.9) y Ec. (2.10) en la Ec. (2.6) y tenemos:

$$\frac{\partial E(p)}{\partial W_{ji}(p)} = -e_j(p) f'(net_j(p)) y_i(p) \qquad (2.11)$$

El ajuste $\Delta W_{ji}(p)$ aplicado a $W_{ji}(p)$ esta definido por la Regla Delta que es:

$$\Delta W_{ji}(p) = -\eta \frac{\partial E(p)}{\partial W_{ji}(p)} \qquad (2.12)$$

Donde η es la tasa de aprendizaje de la red y es una constante. El signo menos en la Ec.(2.11) y Ec. (2.12) significan el descenso del gradiente en el espacio de los pesos de las conexiones.

Así, de acuerdo a las Fig. 2.1, y sustituyendo la Ec. (2.11) en la Ec. (2.12), dada una neurona U_i y la salida que produce y_i, el cambio que se produce en el peso sináptico que conecta la salida de ésta neurona con la neurona U_j, para el p-ésimo patrón de entrenamiento, es:

$$\begin{aligned} \Delta W_{ji}(p) &= \eta . e_j(p) . f'(net_j(p)) y_i(p) \\ \Delta W_{ji}(p) &= \eta . \delta_j(p) y_i(p) \end{aligned} \qquad (2.13)$$

El punto en que difiere la Regla Delta con la Regla Delta Generalizada es en el cálculo de $\delta_j(p)$. Ésto se debe a que en las redes multicapa no se puede conocer de primera mano el valor deseado de las neuronas correspondientes a las capas ocultas, sin embargo si se conoce las salidas deseadas y salidas obtenidas de las neuronas de la capa de salida. De tal forma que si U_j es neurona de salida $\delta_j(p)$ viene dado simplemente por:

$$\begin{aligned}\delta_j(p) &= e_j(p).f'(net_j(p)) \\ \delta_j(p) &= (d_j(p) - y_j(p)).f'(net_j(p))\end{aligned} \qquad (2.14)$$

El término derivativo de la función de activación *f'(net$_j$(p))* representa la modificación que hay que realizar en la entrada que recibe la neurona *j*.

Ahora bien, en caso de que la neurona no sea de la capa de salida, el error que se produce estará en función del error que se cometa en las neuronas que reciban como entrada la salida de dicha neurona, resultando lo que se conoce como propagación del error hacia atrás (Retropropagación).

Tomando en cuenta esta afirmación, en caso de que U_j no sea una neurona de salida, el error que produce estará en función de la suma de los errores que se cometan en las neuronas que reciben como entrada la salida de U_j, entonces tenemos que $\delta_j(p)$ es:

$$\delta_j(p) = (\sum_k \delta_k(p) W_{kj}(p)).f'(net_j) \qquad (2.15)$$

Donde el rango *k* cubre todas las neuronas a las que esta conectada la salida de U_j. Entonces el error que se produce en la neurona "oculta" U_j es la suma de los errores que se producen en las neuronas a las que está conectada la salida de U_j multiplicado por el peso de conexión correspondiente.

ADICIÓN DE MOMENTO.

El Algoritmo de Retropropagación requiere un número importante de cálculos y posteriores iteraciones para lograr el ajuste de los pesos de la red.

En la implementación del algoritmo, se toma una amplitud de paso que viene dado por la tasa de aprendizaje η. A mayor tasa de aprendizaje, mayor modificación de los pesos en cada iteración y por consecuencia es más rápida la fase de entrenamiento, pero puede suceder que la red no llegue a converger al mínimo de la función de error.

Por esta razón, se añade a la expresión general de $\Delta W_{ji}(p)$ un término constante β denominado *momento* [11], de forma tal que se incremente la tasa de aprendizaje sin el riesgo de la no convergencia del algoritmo, al final la expresión para $\Delta W_{ji}(p)$ queda:

$$\Delta W_{ji}(p) = \eta \delta_j(p) y_i(p) + \beta \Delta W_{ji}(p) \qquad (2.16)$$

En Resumen los cambios de los pesos W_{ji} vienen dados por:

$$\Delta W_{ji}(p) = \eta \delta_j(p) y_i(p) + \beta \Delta W_{ji}(p)$$

$$\delta_j(p) = (d_j(p) - y_j(p)).f'(net_j(p)) \quad \text{Si j es neurona de salida}$$

$$\delta_j(p) = (\sum_k \delta_k(p) W_{kj}(p)).f'(net_j) \quad \text{Si j es neurona oculta}$$

2.4 REDES NEURONALES PERCEPTRÓNICAS PARA RESOLVER PROBLEMAS DE PROGRAMACIÓN LINEAL.

Luego de haber estudiado los fundamentos teóricos de las RPM, en esta sección se procederá a demostrar en forma práctica el potencial de las mismas a través de la resolución de problemas de programación lineal (P.L..), tomando como base para ello, problemas de dos variables y tres restricciones.

Una vez seleccionados los problemas, estos fueron resueltos por el Método Simplex [1], para de esta manera conformar el conjunto de datos de entrenamiento de la red (patrones entrada-salida). En total se resolvieron 120 problemas de P.L.. de 2 variables con 3 restricciones, 100 de los cuales se usaron para entrenamiento y los 20 restantes para las pruebas de la red.

Entonces, basados en los conceptos teóricos sobre redes perceptrónicas, acerca de su arquitectura, tipos de aprendizaje y algoritmos de entrenamiento, se ha planteado una hipótesis tomando en cuenta estas potencialidades, sobre todo en la capacidad que tienen de generalización.

Dicha hipótesis se puede resumir en la siguiente proposición:

"*Apoyados en las capacidades de aprendizaje y generalización de las redes perceptrónicas multicapa, problemas P.L.. pueden ser resueltos por esta vía, es decir, las salidas de la red representarán la solución óptima a un problema de P.L..*"

FUNDAMENTOS PARA LA DEMOSTRACION.

En forma general un problema de P.L.. de *n* variables y *m* restricciones se puede expresar de la siguiente forma:

$$Minimizar \quad Z = c^T x \qquad (2.17)$$
$$Sujeto\ a \quad Ax = b \qquad (2.18)$$
$$LI \le x \le LS \qquad (2.19)$$

Donde:

$c \in \Re^n$ Vector columna correspondiente a los coeficientes de costos de la Función Objetivo *(f)*

$A \in \Re^m x \Re^n$ Matriz de coeficientes de las restricciones

$x \in \Re^{n+m}$ Vector columna correspondientes a las variables de decisión y holgura

$b \in \Re^m$ Vector correspondiente a la cantidad de recursos disponibles

$LI \in \Re^n$, $LS \in \Re^n$ Vectores columna correspondientes a los límites superior e inferior que puede tomar las variables de decisión.

Además $-\infty \le LI \le LS \le +\infty \quad para\ i = 1,2,...n$

Desplegando las Ec. (2.17), Ec. (2.18) y Ec. (2.19), resulta la siguiente forma:

$$Minimizar\ Z = c_1x_1 + c_2x_2 + ...c_nx_n$$
Sujeto a:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ...a_{1n}x_n + x_{n+1} &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ...a_{2n}x_n + x_{n+2} &= b_2 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots & \\ a_{m1}x_1 + a_{m2}x_2 + ...a_{mn}x_n + x_{n+m} &= b_m \end{aligned}$$
$$x_i \ge 0; \quad i = 1,..n$$

Ahora bien, llevemos el modelo a una forma matricial para visualizar algunos aspectos importantes:

$$\begin{bmatrix} a_{11} & a_{12}\ldots & a_{1n} & 1 & 0\ldots. & 0 & b_1 \\ a_{21} & a_{22}\ldots & a_{2n} & 0 & 1\ldots. & 0 & b_2 \\ & & & & & & \vdots \\ a_{m1} & a_{m2}\ldots & a_{mn} & 0 & 0\ldots. & 1 & b_m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+m} \\ -1 \end{bmatrix} \overset{\text{A}}{\leftarrow} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \qquad (2.20)$$

$$\overset{\text{B}}{\rightarrow} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+m} \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} c_1 & c_2.....c_n & 0 & 0.....0 & Z \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix} \qquad (2.21)$$

De esta forma se observa que los vectores (A) y (B) son iguales, entonces se puede incluir la Ec. (2.21) en la Ec. (2.20), con el fin de obtener una forma general matricial del problema.

Reescribiendo tenemos:

$$\begin{bmatrix} a_{11} & a_{12}\ldots & a_{1n} & 1 & 0\cdot & \cdot 0 & b_1 \\ a_{21} & a_{22}\ldots & a_{2n} & 0 & 1\cdot & \cdot 0 & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2}\ldots & a_{mn} & 0 & 1\cdot & \cdot 0 & b_m \\ c_1 & c_2\ldots & c_n & 0 & 0\cdot & \cdot 0 & Z \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+m} \\ -1 \end{bmatrix} \overset{\text{C}}{} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \qquad (2.22)$$

Ahora, es de interés tener la variable Z dentro del vector de variables de decisión (C), de tal forma que podemos escribir la siguiente estructura:

$$\begin{bmatrix} a_{11} & a_{12}\dots & a_{1n} & 1 & 0 & \cdots & 0 & b_1 & 0 \\ a_{21} & a_{22}\dots & a_{2n} & 0 & 1 & & 0 & b_2 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2}\dots & a_{mn} & 0 & 0 & \cdots & 1 & b_m & 0 \\ c_1 & c_2 & c_n & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+m} \\ -1 \\ -Z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \qquad (2.23)$$

(D) (E)

Esta estructura matricial es la que necesitamos para visualizar la posible configuración de la red, es decir, cuántas neuronas en la capa de entrada vamos a utilizar y qué salidas son las que necesitamos conocer.

El objetivo de la red, luego de ser satisfactoriamente entrenada, es conseguir los valores de las variables presentes en el vector (E), es decir, las variables de decisión. Por lo tanto ellas serán las "salidas" de la red. La solución al problema vendrá asociada por medio de una transformación lineal o no lineal de la matriz (D), es decir, $\vec{X} = \Gamma(D)$

Es por ello, que los patrones de entrada de la red estarán conformados por los elementos de la matriz (D), es decir, los parámetros propios del problema P.L. (a_{ij}, c_i, b_i), formándose una especie de "mega-vector" que contiene las entradas de la red.

De esta forma la topología de la RPM propuesta es la siguiente:

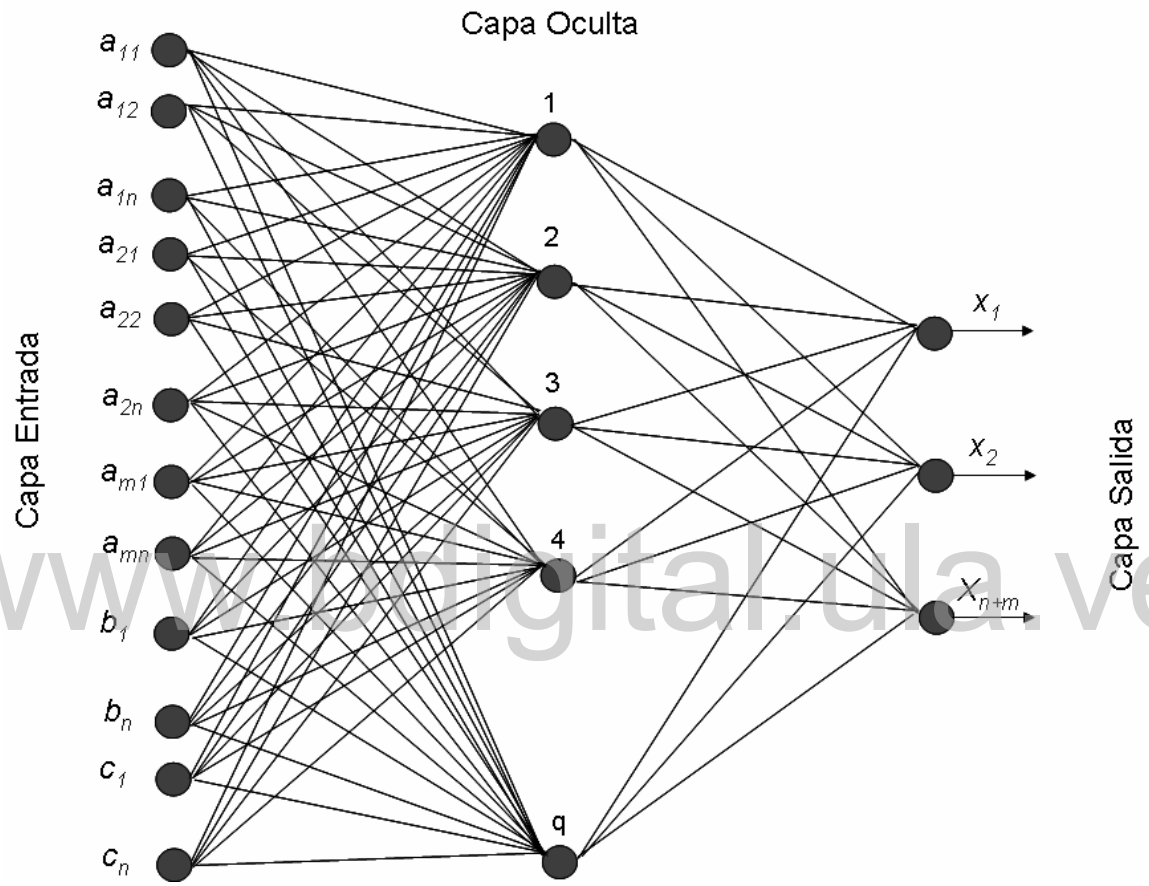


Fig. 2.4 Topología de la Red Perceptrónica propuesta

Ahora bien, el algoritmo de entrenamiento a utilizar es el Algoritmo de Retropropagación (con momento), de tal forma que el modelo general resultante es:

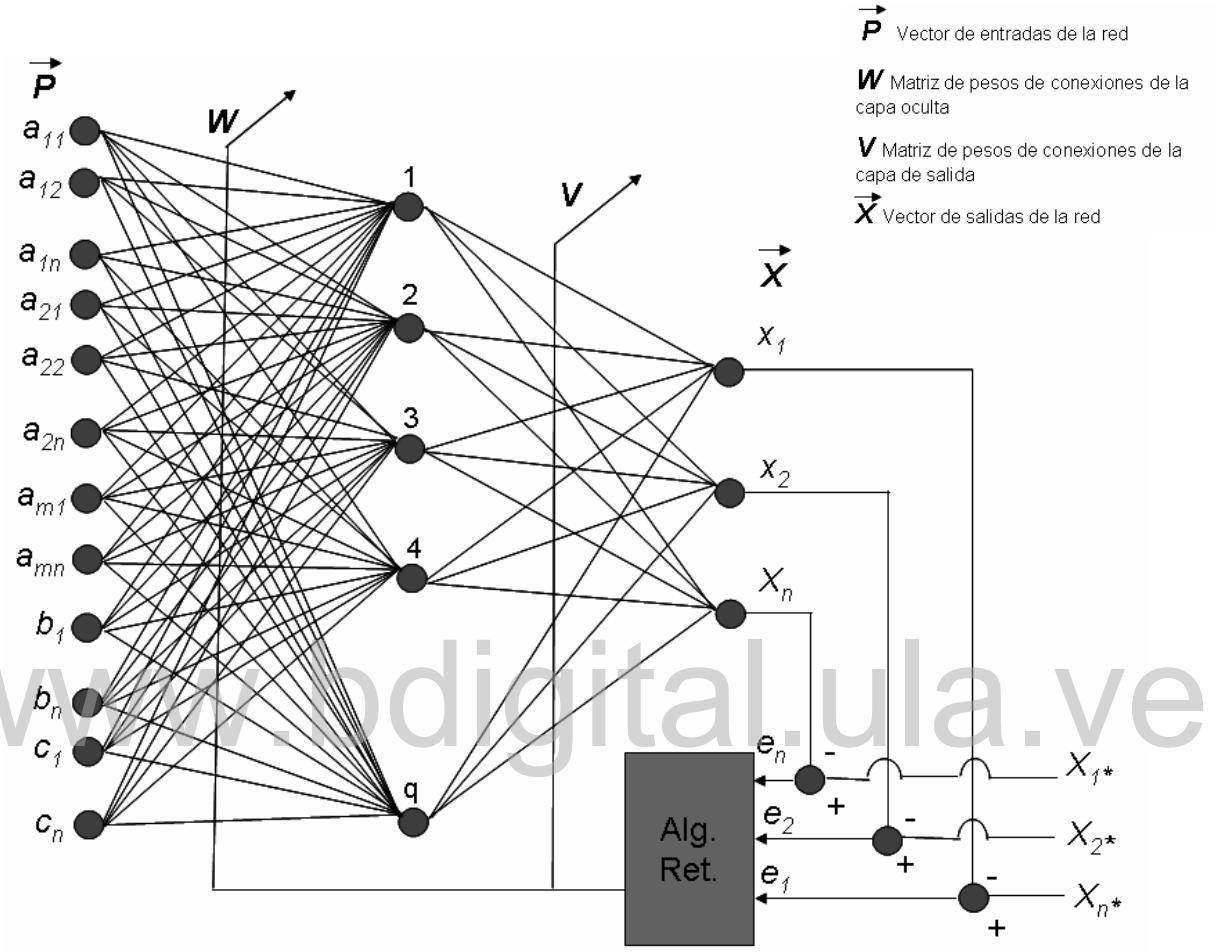


Fig. 2.5 Modelo general de la Red Perceptrónica propuesta

El algoritmo de retropropagación, se encargará de minimizar el error cuadrático medio (MSE) producido entre ***X y X****, ajustando las matrices de pesos *W* y *V*.

Entonces, la solución al problema de P.L., luego de haber entrenado satisfactoriamente la red y obtener las matrices (*W* y *V*) de los pesos de las conexiones, vendrá dada por:

$$\vec{X} = \Gamma(D)$$

$$\vec{X} = \Gamma(V(\Gamma(WP)))$$

2.4.1 ENTRENAMIENTO DE LA RED PERCEPTRÓNICA MULTICAPA PROPUESTA.

Para llevar a cabo el entrenamiento de la red, se generaron 120 problemas de P.L. de 2 variables con 3 restricciones, completamente operacionales, es decir, que poseen solución (no tienen solución vacía) y la función objetivo está acotada (no hay solución infinita).

Se trabajaron con problemas pequeños debido a que se quiere primero determinar si se puede obtener un buen entrenamiento, y segundo si se pueden obtener buenos resultados en la fase de validación. Los 120 problemas se resolvieron con el método simplex obteniendo de esta forma la matriz completa de datos para el entrenamiento, 100 casos serán para entrenar a la red y los 20 restantes para realizar las pruebas correspondientes.

Además se seleccionaron 4 configuraciones distintas de RPM para llevar a cabo el entrenamiento, principalmente se varía la función de activación tanto de la capa oculta como de salida, así como también, el número de neuronas en la capa oculta. El software utilizado para entrenar y validar las RPM, fue el *Neuro Solutions* 4.0 de la empresa *Neuro Dimension*, software desarrollado en el laboratorio de computación neuronal de la Universidad de la Florida.

Es importante resaltar que los parámetros de los problemas de P.L. escogidos para entrenar a la red (a_{ij}, c_i, b_i), están acotados por una cota superior e inferior seleccionada de forma arbitraria, esto se hace ya que se tiene una sospecha de que se puede obtener un mejor desempeño en el entrenamiento. Las cotas vienen dadas por:

$$1 \leq a_{ij} \leq 5$$
$$1 \leq c_i \leq 5$$
$$5 \leq b_i \leq 10$$

CONFIGURACIONES.

➢ CONFIGURACIÓN # 1.

- 11 Neuronas en capa de entrada
- 10 Neuronas en capa oculta
- 5 Neuronas en capa de salida
- Función de activación hiperbólica tangencial en capa oculta
- Función de activación lineal en capa de salida
- Algoritmo de entrenamiento: Retropropagación del error, con momento de 0.7 en capa oculta y en capa de salida
- Tasa de aprendizaje de 0.01 en capa oculta y en capa de salida.
- 100 casos de entrenamiento
- 10.000 iteraciones

Resultados del Entrenamiento: Configuración # 1.

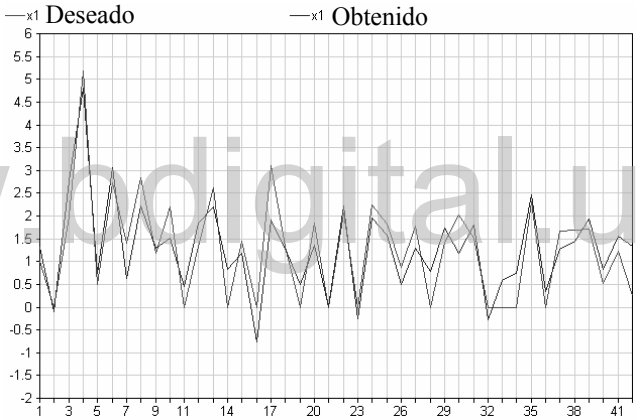


Figura 2.6 Comparación de respuestas para la variable de decisión x_1

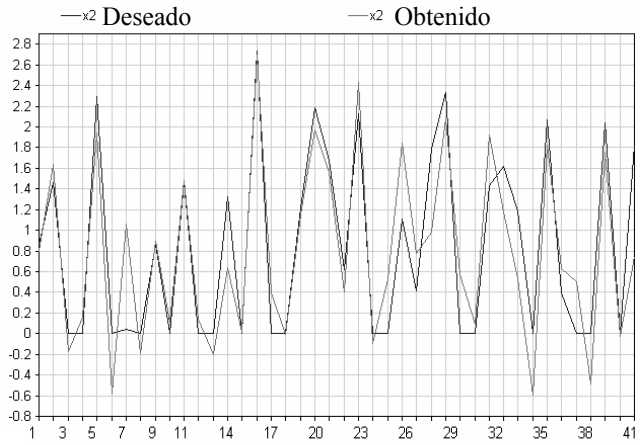


Figura 2.7 Comparación respuestas variable de decisión x_2

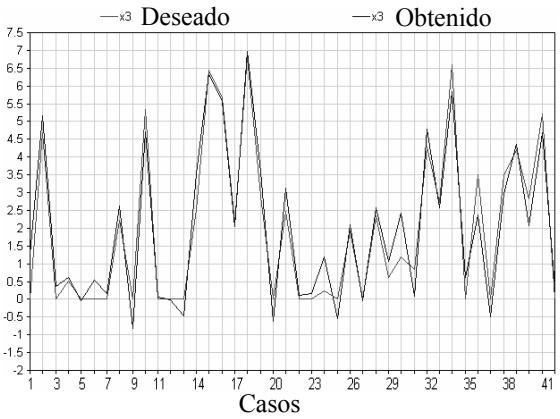


Figura 2.8 Comparación de respuestas para la variable de decisión x_3

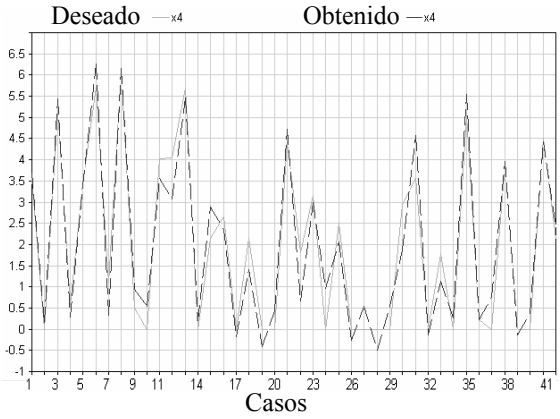


Figura 2.9 Comparación de respuestas para la variable de decisión x_4

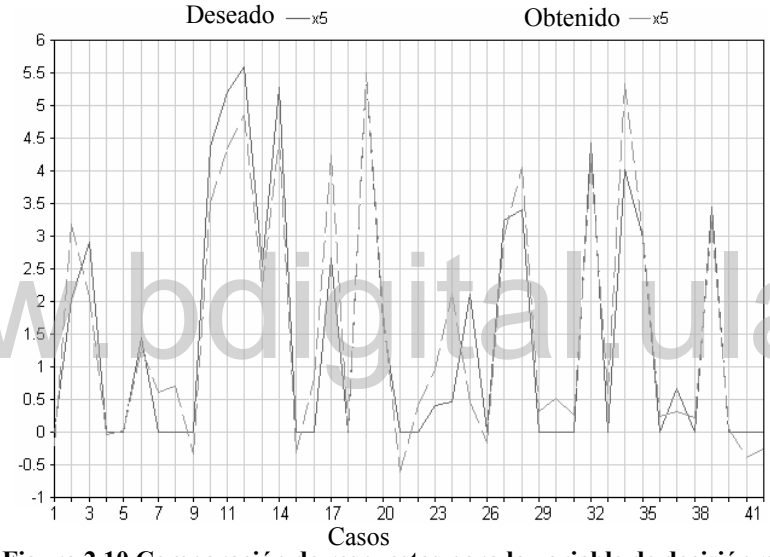


Figura 2.10 Comparación de respuestas para la variable de decisión x_5

En las Fig. 2.6 hasta la Fig. 2.10 se observa el comportamiento del entrenamiento de la red y se compara la salida actual de la red con la salida obtenida por la misma, para las variables de decisión x_1, x_2 y para las variables de holgura x_3 , x_4, x_5 . Se puede decir a la luz de las gráficas que se produjo un entrenamiento no muy aceptable, ciertamente se muestran discrepancias sobre todo en los valles que forman las líneas en el gráfico.

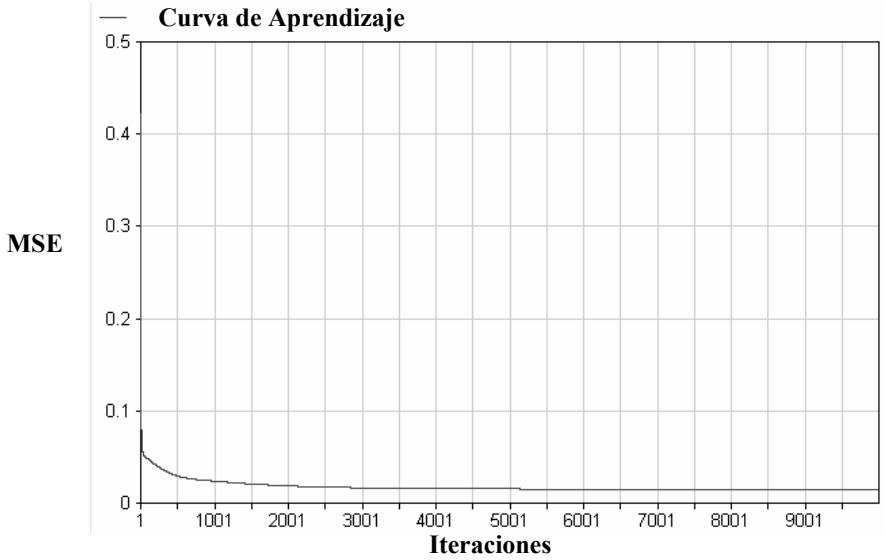


Figura 2.11 Curva de aprendizaje

Error cuadrático medio **(MSE)**	0.0288
Coeficiente de correlación **(r)**	0.9335

Tabla 2.1 Medidas de Desempeño de la Red.

En la Fig. 2.11 se observa la curva de aprendizaje, donde se visualiza la evolución del error cuadrático medio (MSE) a medida que transcurren las iteraciones. Se aprecia una lenta evolución en el aprendizaje de la red. En la Tabla 2.1 se muestran las medidas de desempeño, donde se precisa que el valor del error cuadrático medio (MSE) en la fase de entrenamiento resultó ser 0.0288. De igual forma, se muestra el coeficiente de correlación lineal *r,* que resultó igual a 0.933. Estos datos demuestran lo explicado anteriormente sobre el entrenamiento de la red, ya que el error antes mostrado no es aceptable para el propósito planteado sobre la resolución de problemas de P.L.

➢ CONFIGURACIÓN # 2.

- 11 Neuronas en capa de entrada
- 10 Neuronas en capa oculta
- 5 Neuronas en capa de salida
- Función de activación lineal en capa oculta
- Función de activación lineal en capa de salida
- Algoritmo de entrenamiento: Retropropagación del error, con momento de 0.7 en capa oculta y en capa de salida
- Tasa de aprendizaje de 0.1 en capa oculta y en capa de salida.
- 100 casos de entrenamiento
- 5.000 iteraciones

Resultados del Entrenamiento: Configuración # 2.

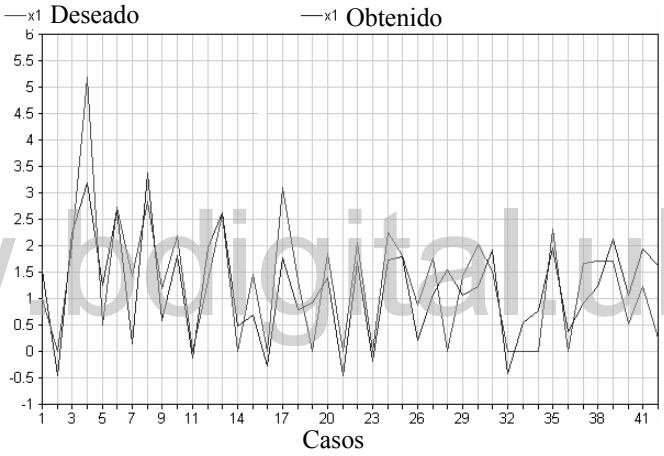


Figura 2.12 Comparación de respuestas para la variable de decisión x_1

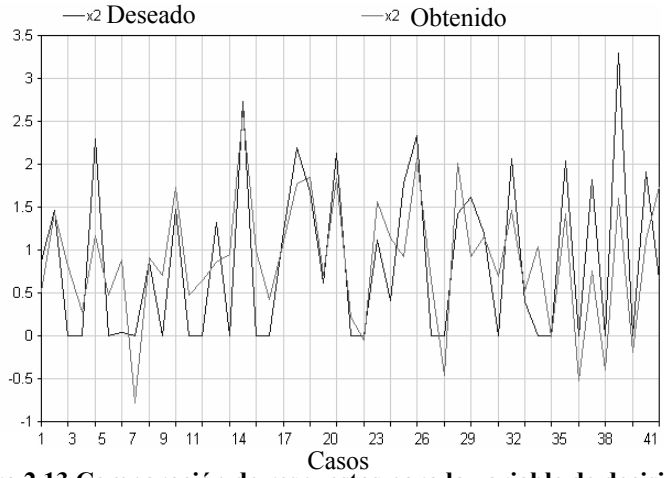


Figura 2.13 Comparación de respuestas para la variable de decisión x_2

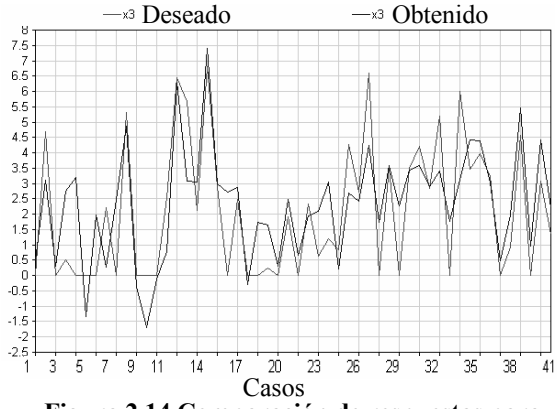


Figura 2.14 Comparación de respuestas para la variable de decisión x_3

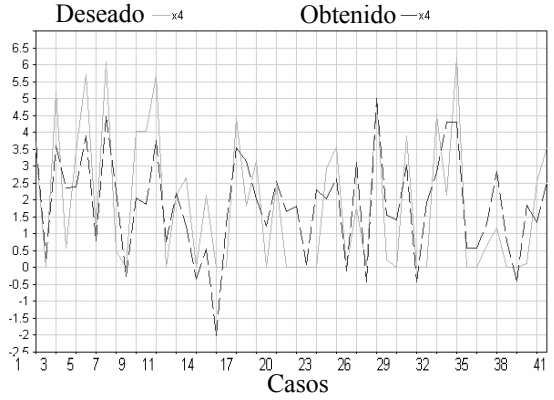


Figura 2.15 Comparación de respuestas para la variable de decisión x_4

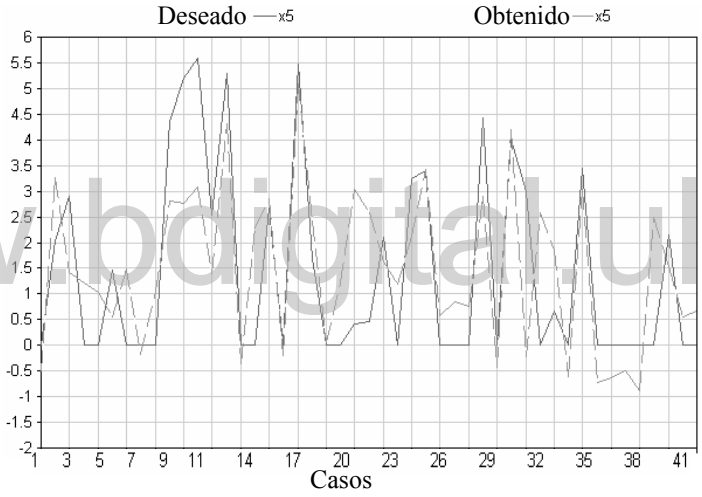


Figura 2.16 Comparación de respuestas para la variable de decisión x_5

En las Fig. 2.12 hasta la Fig. 2.16 se observa el comportamiento del entrenamiento de la red donde se compara la salida deseada de la red con la salida obtenida por la misma, para las variables de decisión x_1 y x_2 y las variables de holgura x_3, x_4, y x_5. Se aprecia diferencias bastantes notables entre las salidas de la red y se acentúan una vez más en los valles de los gráficos, por lo que se induce que no se obtuvo un buen entrenamiento.

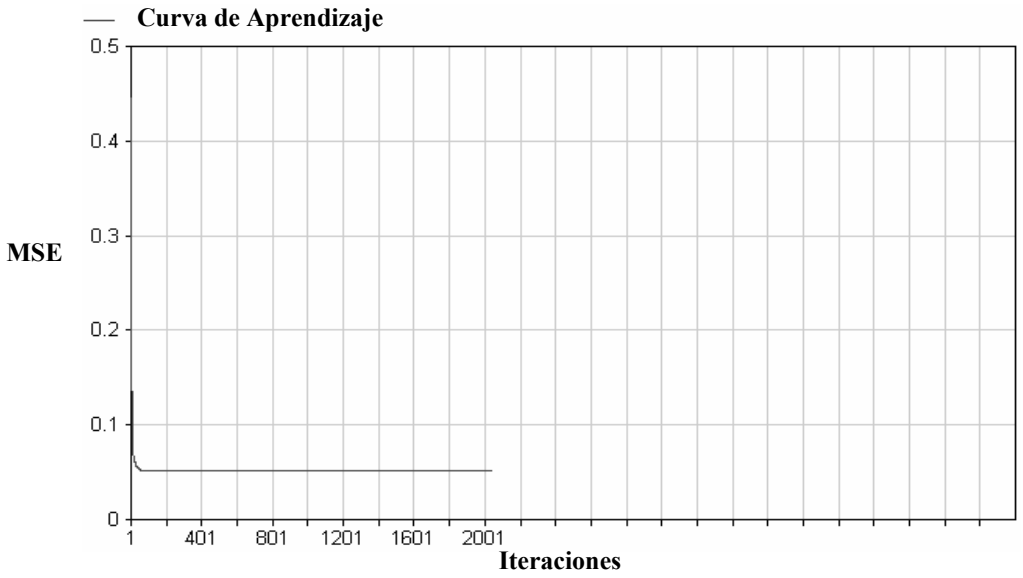


Fig. 2.17 Curva de aprendizaje

MSE	0.0712
r	0.7671

Tabla 2.2 Medidas de Desempeño de la Red

La Fig. 2.17 y Tabla 2.2 ayudan a reforzar lo explicado anteriormente sobre el pobre entrenamiento conseguido para esta configuración. En la Fig. 2.17 se observa la curva de aprendizaje, donde se visualiza la evolución del error cuadrático medio (MSE) a medida que transcurren las iteraciones. Se aprecia que la red llega a un instante en que no sigue evolucionando en su aprendizaje, entonces se optó por detener el entrenamiento en la iteración 2000, ya que era inútil seguir entrenando. En la Tabla 2.2 se muestran las medidas de desempeño, donde se precisa el valor del error cuadrático medio (MSE) en la fase de entrenamiento que resultó ser 0.0712, que es bastante elevado. De igual forma se muestra el coeficiente de correlación lineal *r,* que resultó igual a 0.76, bastante bajo en este caso.

➢ CONFIGURACIÓN # 3.

- 11 Neuronas en capa de entrada
- 20 Neuronas en capa oculta
- 5 Neuronas en capa de salida
- Función de activación tangencial hiperbólica en capa oculta
- Función de activación tangencial hiperbólica en capa de salida
- Algoritmo de entrenamiento: Retropropagación del error, con momento de 0.7 en capa oculta y en capa de salida
- Tasa de aprendizaje de 0.1 en capa oculta y en capa de salida
- 100 casos de entrenamiento
- 5.000 iteraciones

Resultados del Entrenamiento: Configuración # 3.

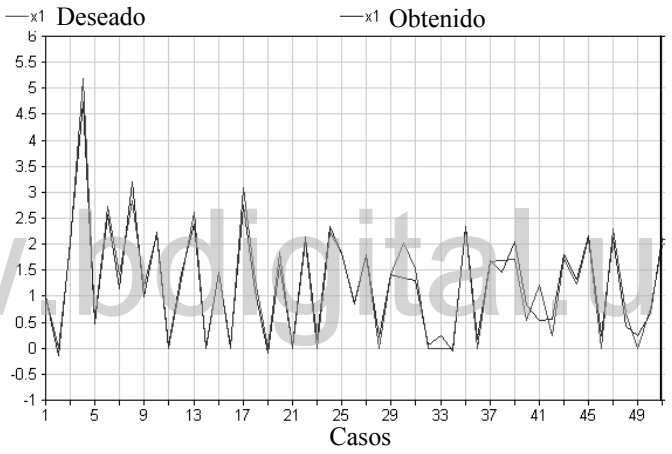


Figura 2.18 Comparación de respuestas para la variable de decisión x_1

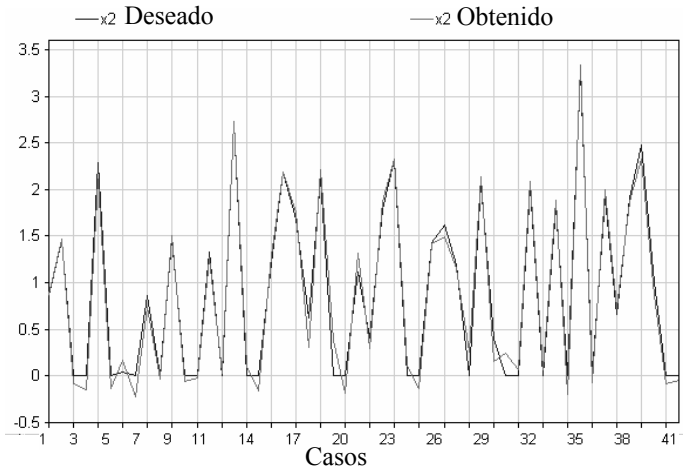


Figura 2.19 Comparación de respuestas para la variable de decisión x_2

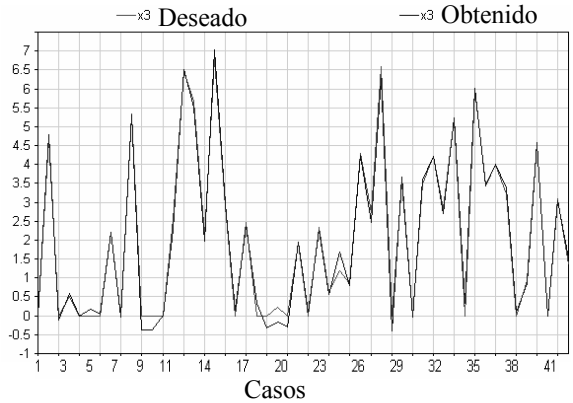


Figura 2.20 Comparación de respuestas para la variable de holgura x_3

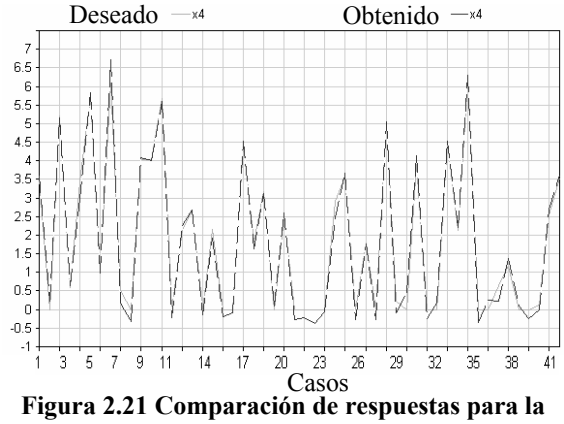


Figura 2.21 Comparación de respuestas para la variable de holgura x_4

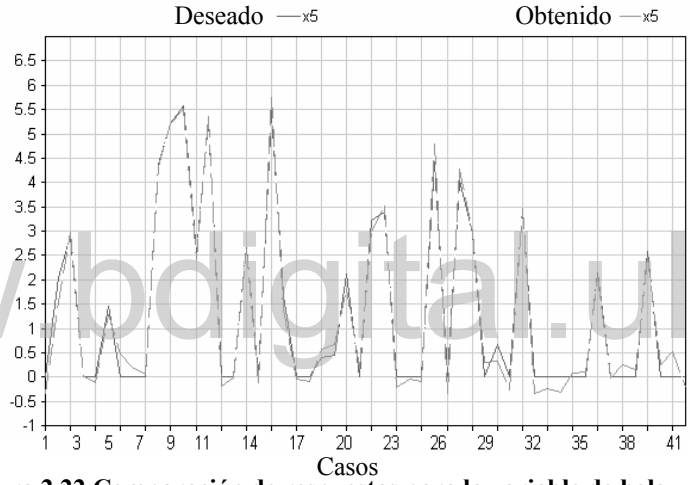


Figura 2.22 Comparación de respuestas para la variable de holgura x_5

En las Fig. 2.18 hasta la Fig. 2.22 se observa el comportamiento del entrenamiento de la red donde se compara la salida deseada de la red con la salida obtenida por la misma, para las variables de decisión x_1 y x_2 y las variables de holgura x_3, x_4, y x_5. Se aprecian pocas diferencias entre las salidas de la red, sobre todo en los valles de los gráficos. En líneas generales se puede decir que se obtuvo un entrenamiento bastante satisfactorio para esta configuración de red.

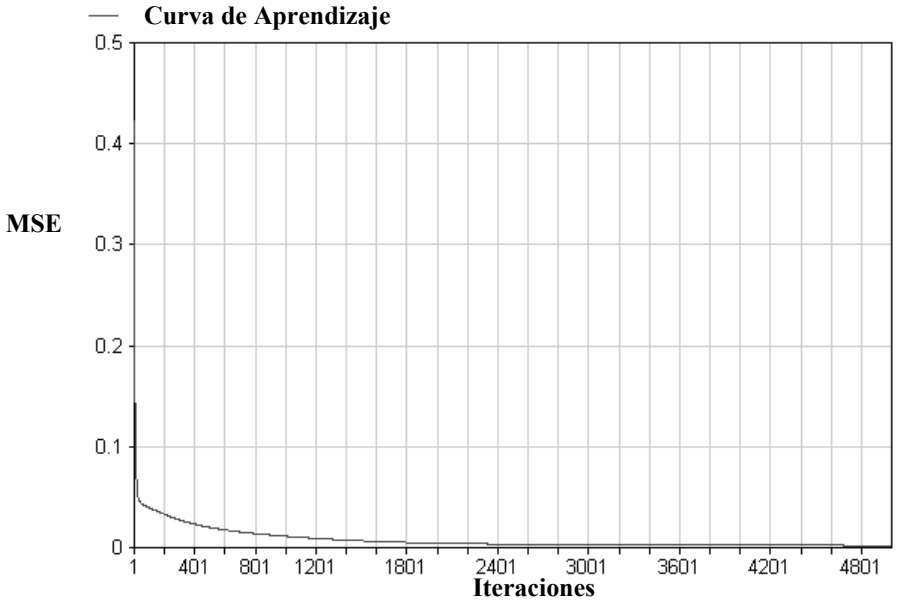


Fig. 2.23 Curva de aprendizaje

MSE	0.0042
r	0.9907

Fig. 2.3 Medidas de Desempeño de la Red

Las Fig. 2.23 y Tabla 2.3 ayudan a reforzar lo explicado anteriormente sobre el buen entrenamiento conseguido para esta configuración. En la Fig. 2.23 se observa la curva de aprendizaje, donde se visualiza la evolución del error cuadrático medio (MSE). Se aprecia que la red a lo largo de las iteraciones evoluciona en su aprendizaje y cada vez va disminuyendo el error. En la Tabla 2.3 se muestran las medidas de desempeño, donde se precisa el valor del error cuadrático medio (MSE) en la fase de entrenamiento que resultó ser bastante bajo 0.0042. De igual forma se muestra el coeficiente de correlación lineal *r,* que resultó igual a 0.9907, casi igual a 1, lo que significa una correlación casi perfecta.

➢ CONFIGURACIÓN # 4.

- 11 Neuronas en capa de entrada
- 20 Neuronas en capa oculta
- 5 Neuronas en capa de salida
- Función de activación lineal mixta en capa oculta
- Función de activación lineal mixta en capa de salida
- Algoritmo de entrenamiento: Retropropagación del error, con momento de 0.7 en capa oculta y en capa de salida
- Tasa de aprendizaje de 0.1 en capa oculta y en capa de salida
- 100 casos de entrenamiento
- 5.000 iteraciones

Resultados del Entrenamiento: Configuración # 4.

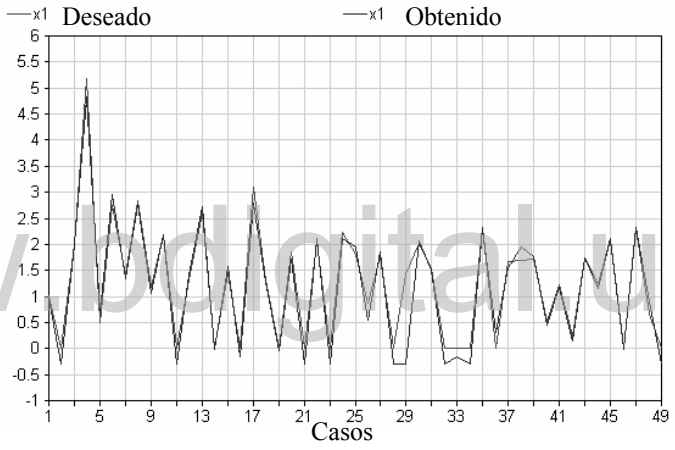


Figura 2.24 Comparación de respuestas para la variable de decisión x_1

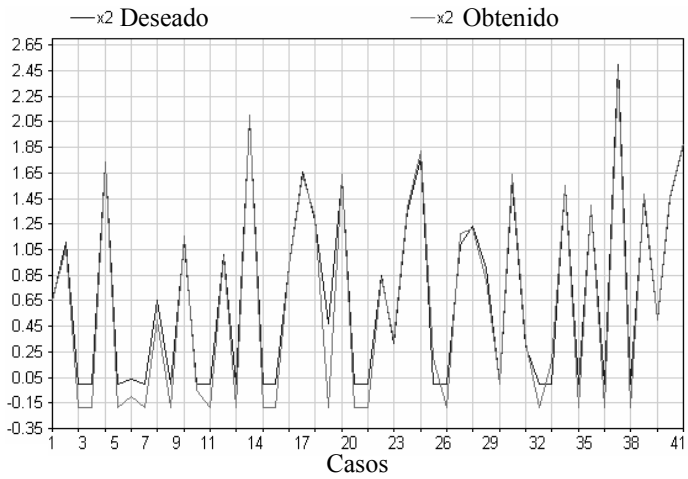


Figura 2.25Comparación de respuestas para la variable de decisión x_2

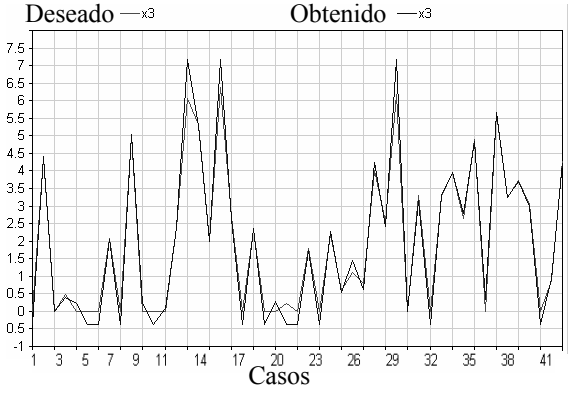


Figura 2.26 Comparación de respuestas para la variable de holgura x_3

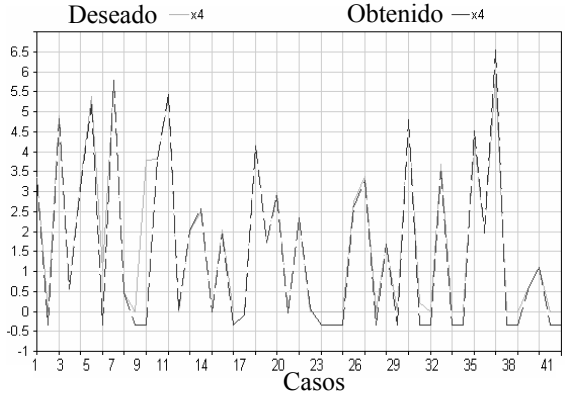


Figura 2.27 Comparación de respuestas para la variable de holgura x_4

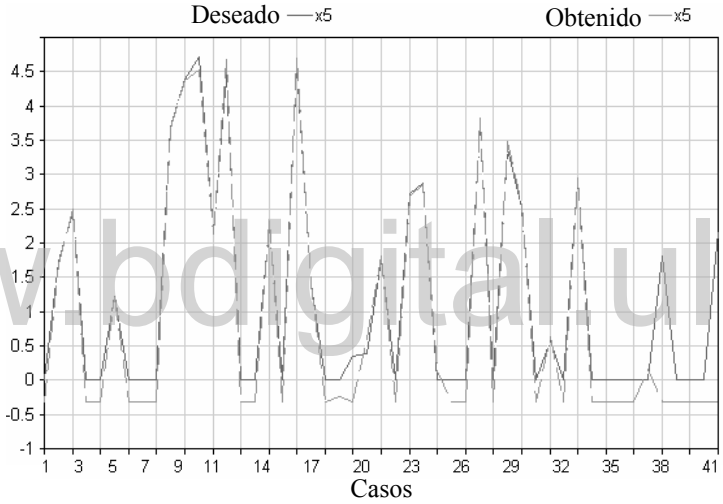


Figura 2.28 Comparación de respuestas para la variable de holgura x_5

En las Fig. 2.24 hasta la Fig. 2.28 se observa el comportamiento del entrenamiento de la red donde se compara la salida actual de la red con la salida obtenida por la misma, para las variables de decisión x_1 y x_2 y las variables de holgura x_3, x_4, y x_5. Se aprecian, de manera similar a la configuración anterior, pocas diferencias (aunque notables en los valles) entre las salidas de la red. Se puede decir que se obtuvo un entrenamiento aceptable.

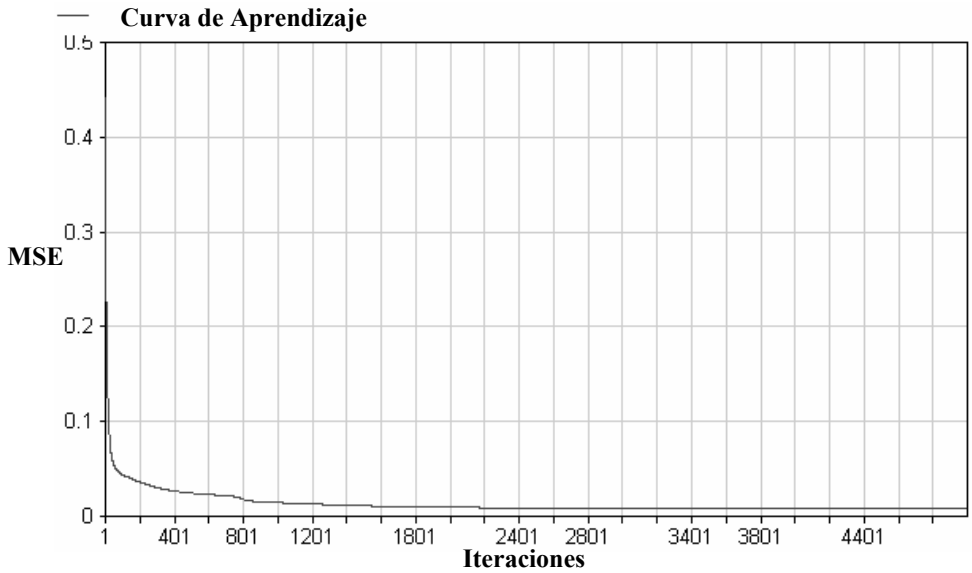


Fig. 2.29 Curva de aprendizaje

MSE	0.0153
r	0.9796

Fig. 2.4 Medidas de desempeño de la red.

En la Fig. 2.29 se observa la curva de aprendizaje, donde se visualiza la evolución del error cuadrático medio (MSE). Se aprecia que la red a lo largo de las iteraciones evoluciona en su aprendizaje y cada vez va disminuyendo el error. En la Tabla 2.4 se muestran las medidas de desempeño, el error cuadrático medio (MSE) en la fase de entrenamiento que resultó ser 0.0153. De igual forma se muestra el coeficiente de correlación lineal *r,* que resultó igual a 0.9796, medidas que en general muestran un entrenamiento aceptable, aunque hay que decir que éste error para los propósitos planteados, sigue siendo elevado.

2.4.2 VALIDACIÓN DE LA RED PERCEPTRÓNICA MULTICAPA.

➢ VALIDACIÓN ASOCIADA A LA CONFIGURACIÓN # 1.

- 20 casos de validación.

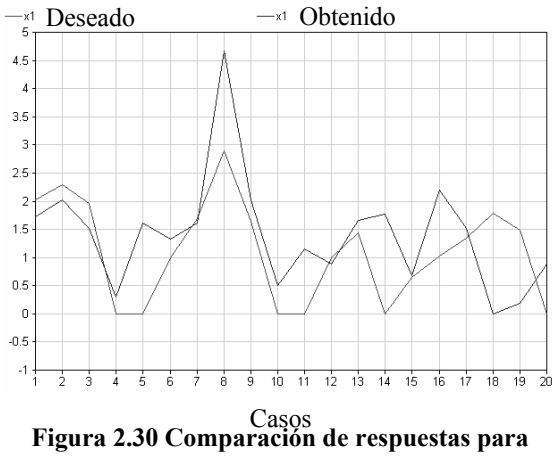


Figura 2.30 Comparación de respuestas para la variable de decisión x_1

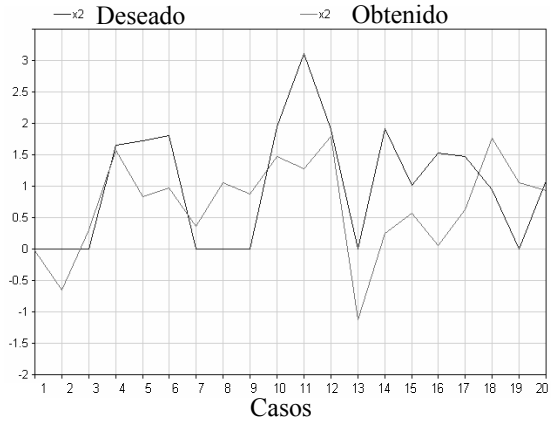


Figura 2.31 Comparación de respuestas para la variable de decisión x_2

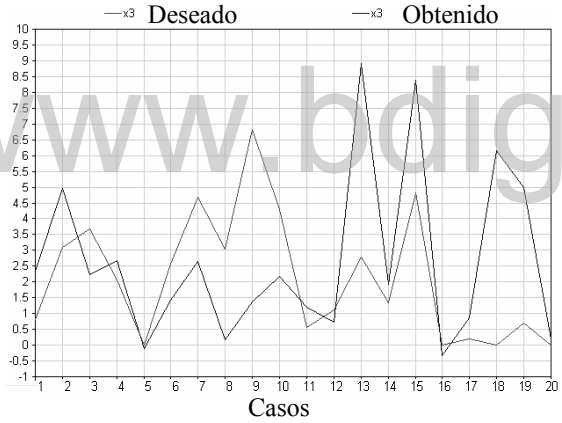


Figura 2.32 Comparación de respuestas para la variable de holgura x_3

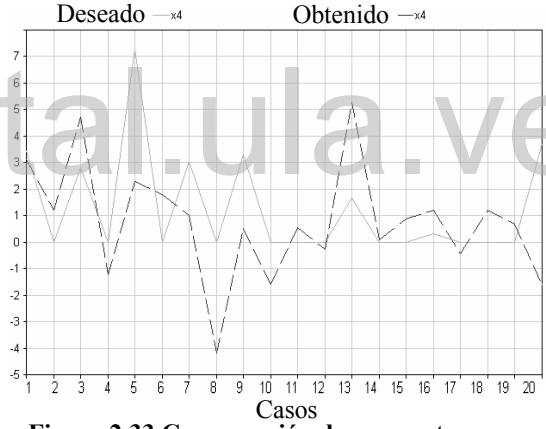


Figura 2.33 Comparación de respuestas para la variable de holgura x_4

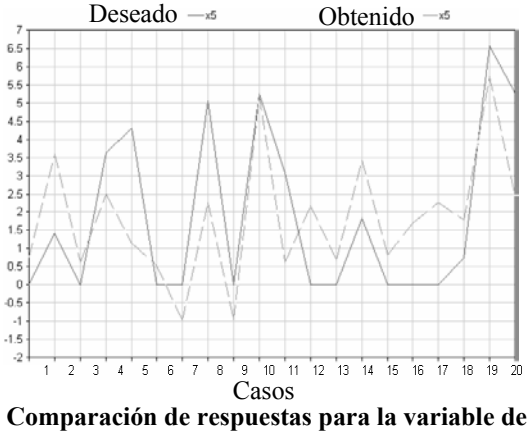


Figura 2.34 Comparación de respuestas para la variable de holgura x_5

MSE	0.2794
r	0.4562

Tabla 2.5 Medidas de Desempeño de la Red.

En las Fig. 2.30 hasta la Fig. 2.34 se compara la salida deseada con la salida obtenida por la red en la fase de validación, para las variables de decisión x_1 y x_2 y las variables de holgura x_3, x_4, y x_5. Se muestran claras diferencias no aceptables en las distintas comparaciones, lo que indica junto con la Tabla 2.5, donde se visualiza un error de 0.2794 y un coeficiente *r* bastante bajo, que esta configuración de red, produce un entrenamiento poco aceptable, y no arroja buenos resultados en la fase de validación.

En la tabla que sigue a continuación (Tabla 2.6) se observan los valores de las salidas deseadas y las salidas obtenidas en la fase de validación para la configuración # 1 de la RPM.

Des x_1	Des x_2	Salida x_1	Salida x_2	Des x_3	Des x_4	Des x_5	Salida x_3	Salida x_4	Salida x_5
2.025	0.000	1.388	-0.317	0.797	3.453	0.000	2.738	6.233	1.997
2.293	0.000	2.352	-0.122	3.098	0.000	1.415	5.598	-0.227	2.816
1.967	0.000	1.486	0.407	3.687	2.820	0.000	1.806	4.874	0.126
0.000	1.656	0.628	2.002	2.053	0.000	3.631	-1.257	-1.721	1.634
0.000	1.719	1.623	0.981	0.000	7.294	4.303	0.224	6.057	1.638
0.994	1.802	1.456	1.804	2.561	0.000	0.000	3.662	0.538	1.476
1.697	0.000	1.425	0.342	4.685	3.058	0.000	2.782	1.896	-0.547
2.893	0.000	2.149	-0.039	3.039	0.000	5.061	7.629	0.794	7.266
1.646	0.000	3.309	-0.311	6.825	3.290	0.000	2.565	3.203	0.468
0.000	1.946	-0.628	1.583	4.281	0.000	5.254	9.145	-0.066	7.520
0.000	3.105	0.555	1.514	0.558	0.000	3.084	3.894	0.853	1.961
0.993	1.897	1.446	0.709	1.128	0.000	0.000	2.885	0.184	3.180
1.436	0.000	1.217	-0.423	2.800	1.651	0.000	5.163	4.537	1.591
0.000	1.921	0.907	1.828	1.329	0.000	1.821	2.004	-0.191	1.669
0.642	1.020	0.263	0.902	4.815	0.000	0.000	6.525	2.979	1.279
1.033	1.533	1.236	2.387	0.000	0.303	0.000	2.673	1.290	1.193
1.342	1.479	0.343	2.048	0.199	0.000	0.000	4.354	1.095	1.127
1.780	0.941	1.012	1.244	0.000	0.000	0.738	3.026	1.436	0.889
1.488	0.000	0.283	0.896	0.710	0.000	6.573	4.729	1.575	5.499
0.000	1.082	1.437	0.375	0.000	3.700	5.273	0.993	0.473	4.210

Tabla. 2.6 Salidas deseadas y salidas obtenidas por la red.

➢ VALIDACIÓN ASOCIADA A LA CONFIGURACIÓN # 2.

- 20 casos de prueba

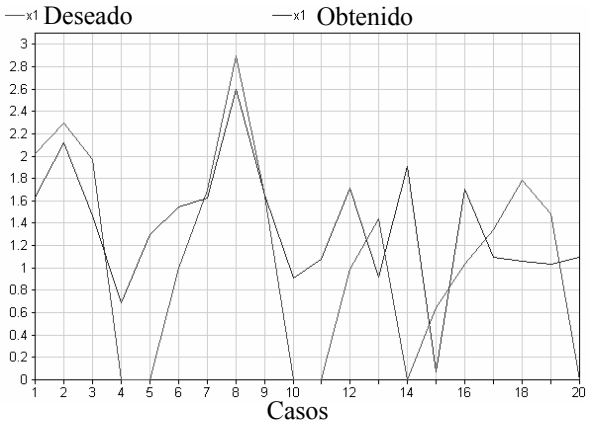


Figura 2.35 Comparación de respuestas para la variable de decisión x_1

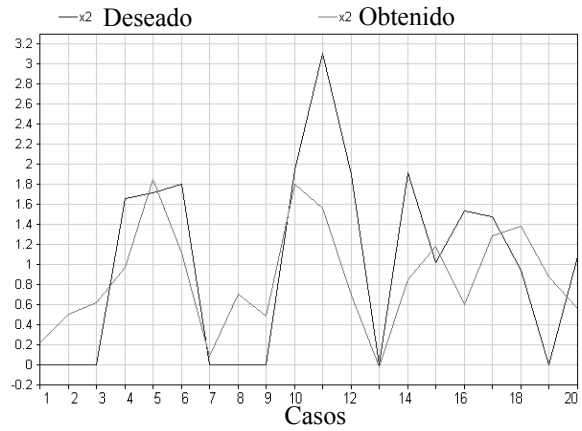


Figura 2.36 Comparación de respuestas para la variable de decisión x_2

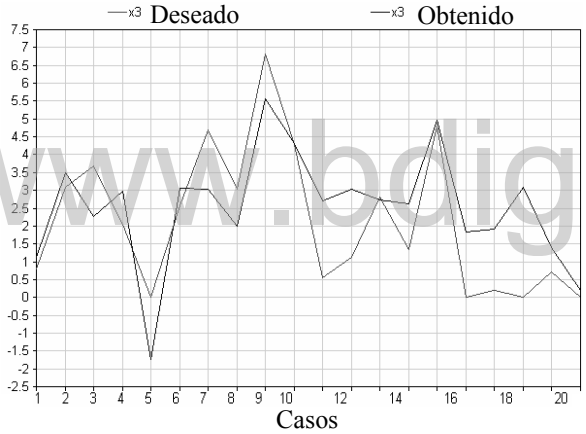


Figura 2.37 Comparación de respuestas para la variable de holgura x_3

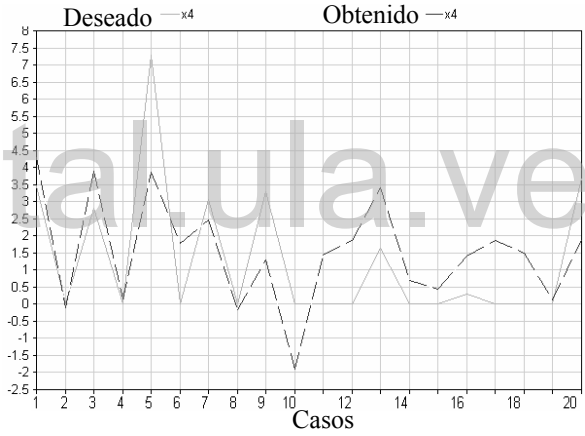


Figura 2.38 Comparación de respuestas para la variable de holgura x_4

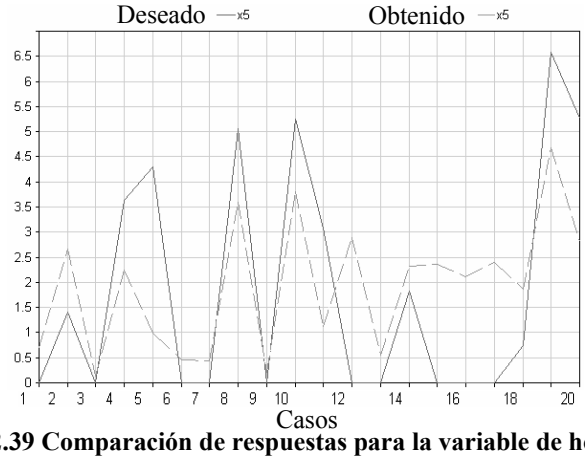


Figura 2.39 Comparación de respuestas para la variable de holgura x_5

MSE	0.1313
r	0.6587

Tabla 2.7 Medidas de Desempeño de la Red.

En las Fig. 2.35 hasta la Fig. 2.39 se compara la salida deseada con la salida obtenida por la red en la fase de validación, para las variables de decisión x_1 y x_2 y las variables de holgura x_3, x_4, y x_5. De forma similar que en la fase de entrenamiento se observan claras diferencias en las distintas comparaciones, lo que junto con la Tabla 2.7, donde se visualiza un error de 0.1313 y un coeficiente *r* de 0.65, indica que esta configuración de red, aunque produce un error menor que la configuración # 1 en la fase de validación, sigue sin arrojar resultados aceptables.

En la tabla que sigue a continuación (Tabla 2.8) se observan los valores de las salidas deseadas y las salidas obtenidas en la fase de validación para la configuración # 2 de la RPM.

Des x_1	Des x_2	Salida x_1	Salida x_2	Des x_3	Des x_4	Des x_5	Salida x_3	Salida x_4	Salida x_5
2.025	0.000	1.630	0.217	0.797	3.453	0.000	1.127	4.292	0.710
2.293	0.000	2.123	0.510	3.098	0.000	1.415	3.503	-0.095	2.671
1.967	0.000	1.465	0.618	3.687	2.820	0.000	2.262	3.885	0.135
0.000	1.656	0.693	0.971	2.053	0.000	3.631	2.974	0.152	2.253
0.000	1.719	1.299	1.854	0.000	7.294	4.303	-1.731	3.864	0.980
0.994	1.802	1.545	1.117	2.561	0.000	0.000	3.059	1.767	0.459
1.697	0.000	1.625	0.084	4.685	3.058	0.000	3.040	2.462	0.447
2.893	0.000	2.597	0.707	3.039	0.000	5.061	2.002	-0.167	3.598
1.646	0.000	1.648	0.491	6.825	3.290	0.000	5.557	1.306	0.218
0.000	1.946	0.912	1.801	4.281	0.000	5.254	4.318	-1.903	3.797
0.000	3.105	1.075	1.564	0.558	0.000	3.084	2.716	1.438	1.113
0.993	1.897	1.713	0.692	1.128	0.000	0.000	3.025	1.872	2.899
1.436	0.000	0.915	-0.016	2.800	1.651	0.000	2.744	3.400	0.557
0.000	1.921	1.904	0.849	1.329	0.000	1.821	2.631	0.694	2.327
0.642	1.020	0.069	1.184	4.815	0.000	0.000	4.962	0.430	2.350
1.033	1.533	1.706	0.604	0.000	0.303	0.000	1.826	1.396	2.107
1.342	1.479	1.097	1.283	0.199	0.000	0.000	1.913	1.879	2.395
1.780	0.941	1.058	1.387	0.000	0.000	0.738	3.079	1.480	1.872
1.488	0.000	1.030	0.875	0.710	0.000	6.573	1.396	0.136	4.674
0.000	1.082	1.092	0.563	0.000	3.700	5.273	0.184	1.903	2.812

Tabla. 2.8 Salida deseada y Salidas Obtenidas por la red.

➢ VALIDACIÓN ASOCIADA A LA CONFIGURACIÓN # 3.

- 20 casos de prueba.

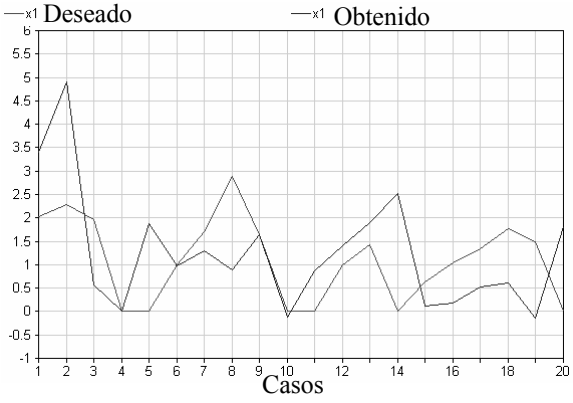


Figura 2.40 Comparación de respuestas para la variable de decisión x_1

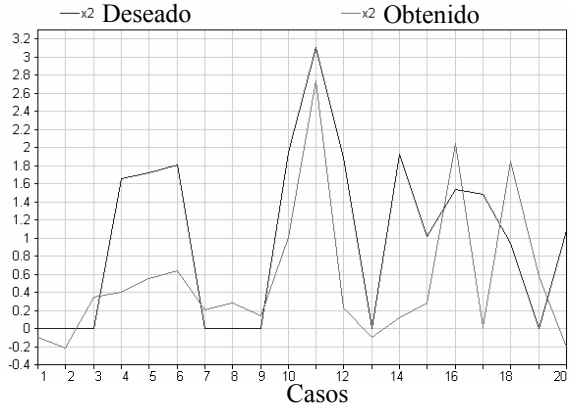


Figura 2.41 Comparación de respuestas para la variable de decisión x_2

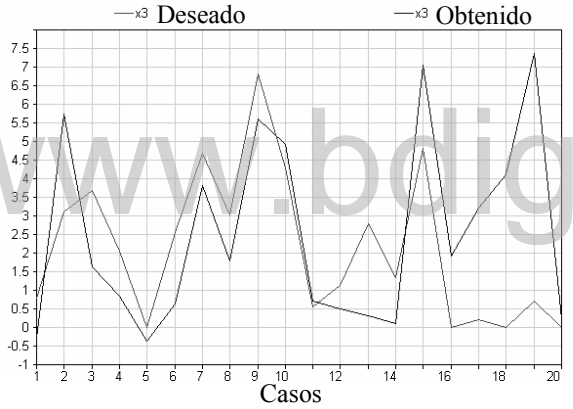


Figura 2.42 Comparación de respuestas para la variable de holgura x_3

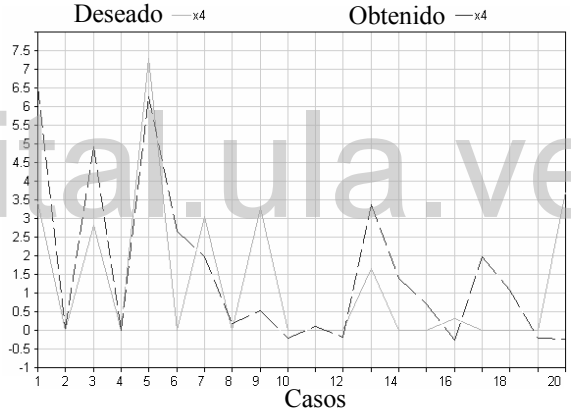


Figura 2.43 Comparación de respuestas para la variable de holgura x_4

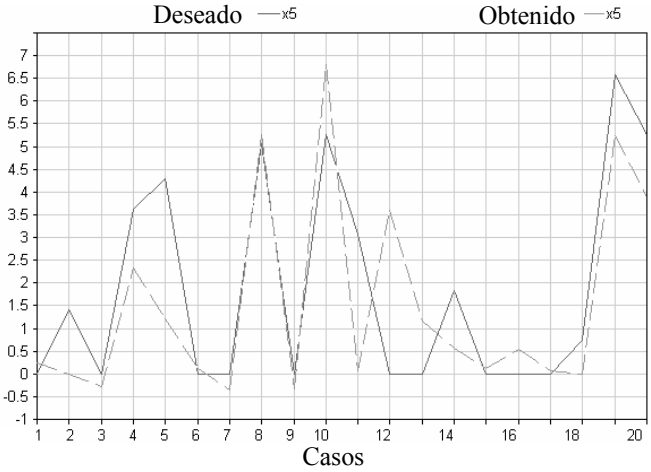


Figura 2.44Comparación de respuestas para la variable de holgura x_5

MSE	0.2154
r	0.5477

Tabla 2.9 Medidas de Desempeño de la Red.

En las Fig. 2.40 hasta la Fig. 2.44 se compara la salida deseada con la salida obtenida por la red en la fase de validación, para las variables de decisión x_1 y x_2 y las variables de holgura x_3, x_4, y x_5. Para sorpresa (debido a que se obtuvo un buen entrenamiento), se observan claras diferencias en las distintas comparaciones, además de acuerdo a la Tabla 2.9, se visualiza un error elevadísimo de 0.2154 y un coeficiente r bastante pobre de 0.54. Todos estos elementos conducen a decir que esta configuración no arroja resultados para nada importantes en la fase de validación, poniendo en evidencia una mala capacidad de generalización por parte de la RPM.

En la tabla que sigue a continuación (Tabla 2.10) se observan los valores de las salidas deseadas y las salidas obtenidas en la fase de validación para la configuración # 3 de la RPM.

Des x_1	Des x_2	Salida x_1	Salida x_2	Des x_3	Des x_4	Des x_5	Salida x_3	Salida x_4	Salida x_5
2.025	0.000	3.399	-0.096	0.797	3.453	0.000	-0.290	6.527	0.236
2.293	0.000	4.899	-0.215	3.098	0.000	1.415	5.736	0.054	0.004
1.967	0.000	0.568	0.342	3.687	2.820	0.000	1.629	4.911	-0.282
0.000	1.656	0.012	0.401	2.053	0.000	3.631	0.830	-0.001	2.333
0.000	1.719	1.869	0.550	0.000	7.294	4.303	-0.370	6.253	1.203
0.994	1.802	0.980	0.635	2.561	0.000	0.000	0.631	2.648	0.113
1.697	0.000	1.305	0.207	4.685	3.058	0.000	3.814	1.986	-0.350
2.893	0.000	0.896	0.280	3.039	0.000	5.061	1.790	0.182	5.253
1.646	0.000	1.650	0.136	6.825	3.290	0.000	5.608	0.516	-0.335
0.000	1.946	-0.126	1.007	4.281	0.000	5.254	4.933	-0.204	6.798
0.000	3.105	0.873	2.739	0.558	0.000	3.084	0.705	0.107	0.080
0.993	1.897	1.413	0.229	1.128	0.000	0.000	0.489	-0.185	3.600
1.436	0.000	1.891	-0.095	2.800	1.651	0.000	0.324	3.362	1.161
0.000	1.921	2.518	0.119	1.329	0.000	1.821	0.096	1.401	0.575
0.642	1.020	0.119	0.282	4.815	0.000	0.000	7.046	0.708	0.122
1.033	1.533	0.176	2.045	0.000	0.303	0.000	1.908	-0.265	0.531
1.342	1.479	0.518	0.006	0.199	0.000	0.000	3.211	1.982	0.071
1.780	0.941	0.613	1.846	0.000	0.000	0.738	4.106	1.067	-0.017
1.488	0.000	-0.149	0.572	0.710	0.000	6.573	7.375	-0.206	5.242
0.000	1.082	1.820	-0.200	0.000	3.700	5.273	0.236	-0.240	3.907

Tabla. 2.10 Salida deseada y Salidas Obtenidas por la red.

➢ VALIDACIÓN ASOCIADA A LA CONFIGURACIÓN # 4.

- 20 casos de prueba.

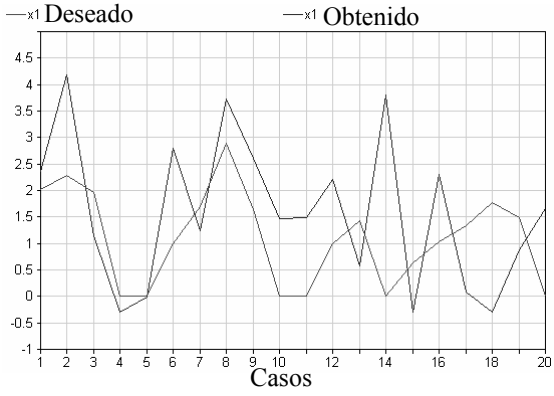


Figura 2.45 Comparación de respuestas para la variable de decisión x_1

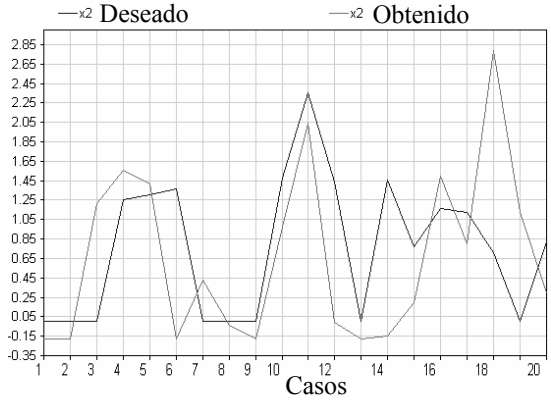


Figura 2.46 Comparación de respuestas para la variable de decisión x_2

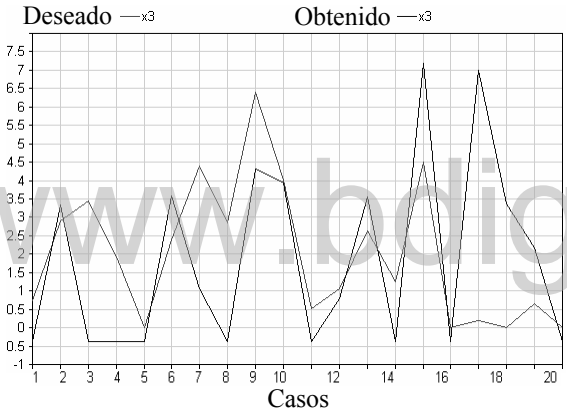


Figura 2.47 Comparación de respuestas para la variable de holgura x_3

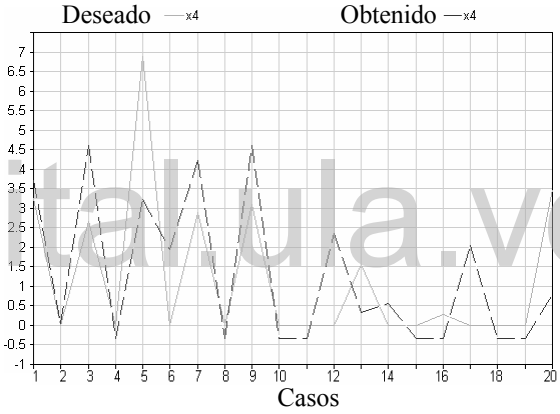


Figura 2.48 Comparación de respuestas para la variable de holgura x_4

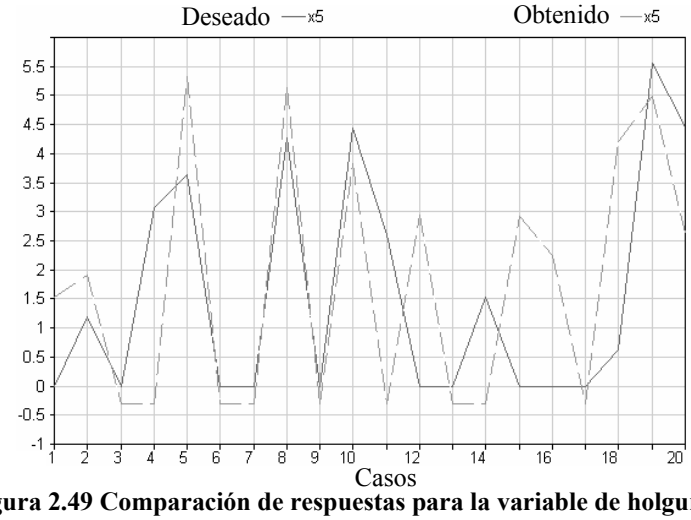


Figura 2.49 Comparación de respuestas para la variable de holgura x_5

MSE	0.2922
r	0.4620

Tabla 2.11 Medidas de Desempeño de la Red.

En las Fig. 2.45 hasta la Fig. 2.49 se compara la salida deseada con la salida obtenida por la red en la fase de validación, para las variables de decisión x_1 y x_2 y las variables de holgura x_3, x_4, y x_5. Nuevamente se observan claras y notables diferencias en las distintas comparaciones, sumando a esto que en la Tabla 2.11 se visualiza un error elevadísimo del 0.2922 y un coeficiente *r* bastante bajo de 0.462; por lo que se puede afirmar que esta configuración arroja los resultados más pobres en la fase de validación de las 4 configuraciones analizadas.

En las tablas que siguen a continuación se observan los valores de las salidas deseadas y las salidas obtenidas en la fase de validación para la configuración # 4 de la RPM (Tabla 2.12), además se muestra un resumen general sobre los parámetros escogidos para cada configuración de red (número de neuronas en la capa oculta, funciones de activación, tasa de aprendizaje, entre otros) y los errores obtenidos en la fase de entrenamiento y en la fase de validación para cada una de las mismas. (Tabla 2.13)

Des x_1	Des x_2	Salida x_1	Salida x_2	Des x_3	Des x_4	Des x_5	Salida x_3	Salida x_4	Salida x_5
2.025	0.000	2.367	-0.241	0.797	3.453	0.000	-0.403	3.950	1.805
2.293	0.000	4.190	-0.241	3.098	0.000	1.415	3.545	0.062	2.268
1.967	0.000	1.141	1.603	3.687	2.820	0.000	-0.403	4.901	-0.368
0.000	1.656	-0.288	2.050	2.053	0.000	3.631	-0.403	-0.366	-0.368
0.000	1.719	-0.014	1.867	0.000	7.294	4.303	-0.403	3.425	6.317
0.994	1.802	2.806	-0.241	2.561	0.000	0.000	3.831	2.058	-0.368
1.697	0.000	1.243	0.557	4.685	3.058	0.000	1.144	4.477	-0.368
2.893	0.000	3.728	-0.048	3.039	0.000	5.061	-0.403	-0.366	6.073
1.646	0.000	2.599	-0.241	6.825	3.290	0.000	4.590	4.901	-0.368
0.000	1.946	1.474	1.275	4.281	0.000	5.254	4.206	-0.366	4.545
0.000	3.105	1.482	2.689	0.558	0.000	3.084	-0.403	-0.366	-0.368
0.993	1.897	2.215	-0.003	1.128	0.000	0.000	0.837	2.490	3.482
1.436	0.000	0.584	-0.241	2.800	1.651	0.000	3.793	0.352	-0.368
0.000	1.921	3.813	-0.201	1.329	0.000	1.821	-0.403	0.598	-0.368
0.642	1.020	-0.288	0.263	4.815	0.000	0.000	7.664	-0.366	3.462
1.033	1.533	2.296	1.972	0.000	0.303	0.000	-0.403	-0.366	2.655
1.342	1.479	0.084	1.053	0.199	0.000	0.000	7.473	2.164	-0.368
1.780	0.941	-0.288	3.672	0.000	0.000	0.738	3.590	-0.366	4.992
1.488	0.000	0.884	1.477	0.710	0.000	6.573	2.308	-0.366	5.918
0.000	1.082	1.663	0.392	0.000	3.700	5.273	-0.403	0.850	3.111

Tabla. 2.12 Salida deseada y Salidas Obtenidas por la red

RESUMEN DE ENTRENAMIENTO Y VALIDACIÓN DE LA RED.

	Capas ocultas	Neuronas en capa oculta	Función activación capa oculta	Función activación capa de salida	Tasa de aprendizaje	Casos para entrenamiento	Casos para validac	Ciclos	Error Entrenamiento	Error de Validac.
Config # 1	1	10	Hiper. Tan	Lineal	0.01	100	20	10000	0.0288	0.2794
Config # 2	1	10	Lineal.	Lineal	0.01	100	20	2000	0.0712	0.1313
Config # 3	1	20	Hiper. Tan	Hiper. Tan.	0.01	100	20	5000	0.0042	0.2454
Config # 4	1	20	Lineal Tan	Lineal Tan	0.01	100	20	5000	0.0153	0.2922

Tabla. 2.13 Resumen del entrenamiento y pruebas obtenidas para las 4 configuraciones.

●CAPÍTULO 3●

REDES NEURONALES RECURRENTES

Las redes neuronales que se presentan en este capítulo representan sistemas dinámicos que evolucionan en el tiempo, bien sea en un espacio discreto o continuo en el tiempo. Las redes con conexión hacia atrás también se le conocen como Redes Retroalimentadas o como Redes Neuronales Recurrentes.(RNR)

La evolución de este tipo de red se considera disipativa ya que evoluciona en dirección del mínimo de una función de energía asociada al sistema. Esta función se le conoce como función de energía computacional. En otras palabras, la transición de una red neuronal dinámica es en dirección de una solución asintótica estable, la cual representa el mínimo de una función de energía disipativa.

Por otra parte las redes neuronales con conexiones hacia adelante no presentan ningún tipo de retroalimentación durante la fase de propagación de la información en la red, sólo se observa este tipo de actividad en la fase de aprendizaje, cuando el error producido es retroalimentado para hacer los ajustes en los pesos y tratar de minimizarlo. Sin embargo, en las redes con conexión hacia atrás, la retroalimentación es inherente al sistema, es decir, es espontánea.

Se debe hacer énfasis en el hecho de que las RNR son por su naturaleza sistemas dinámicos no lineales [12]. Estos sistemas son conocidos por tener dinámicas muy complejas, sin embargo las RNR permiten una gran reducción de esta complejidad.

El modelo de RNR discutido en este trabajo está basado en las investigaciones hechas por J. Hopfield (precursor de las RNR). Este tipo de redes pueden ser usadas en muchas áreas de aplicación como: clasificación, solución de problemas de optimización, restauración de patrones, y también pueden ser vistas como redes de mapeo entrada-salida. Más allá de algunas limitaciones, su impresionante desempeño ha sido documentado en una extensa

literatura. Su versatilidad para ser implementada tanto en hardware como por simulaciones numéricas indica que las RNR ofrecen una alternativa para los enfoques tradicionales en reconocimiento de patrones y problemas de optimización.

3.1 ARQUITECTURA DE LAS RNR.

Siguiendo los postulados de J. Hopfield, una RNR consiste en un modelo monocapa con *n* neuronas cuyos valores de salida pueden ser discretos o continuos. Para el modelo discreto las funciones de activación de las neuronas son de tipo umbral lógico, mientras que en el caso continuo la función de activación es de tipo sigmoidal.

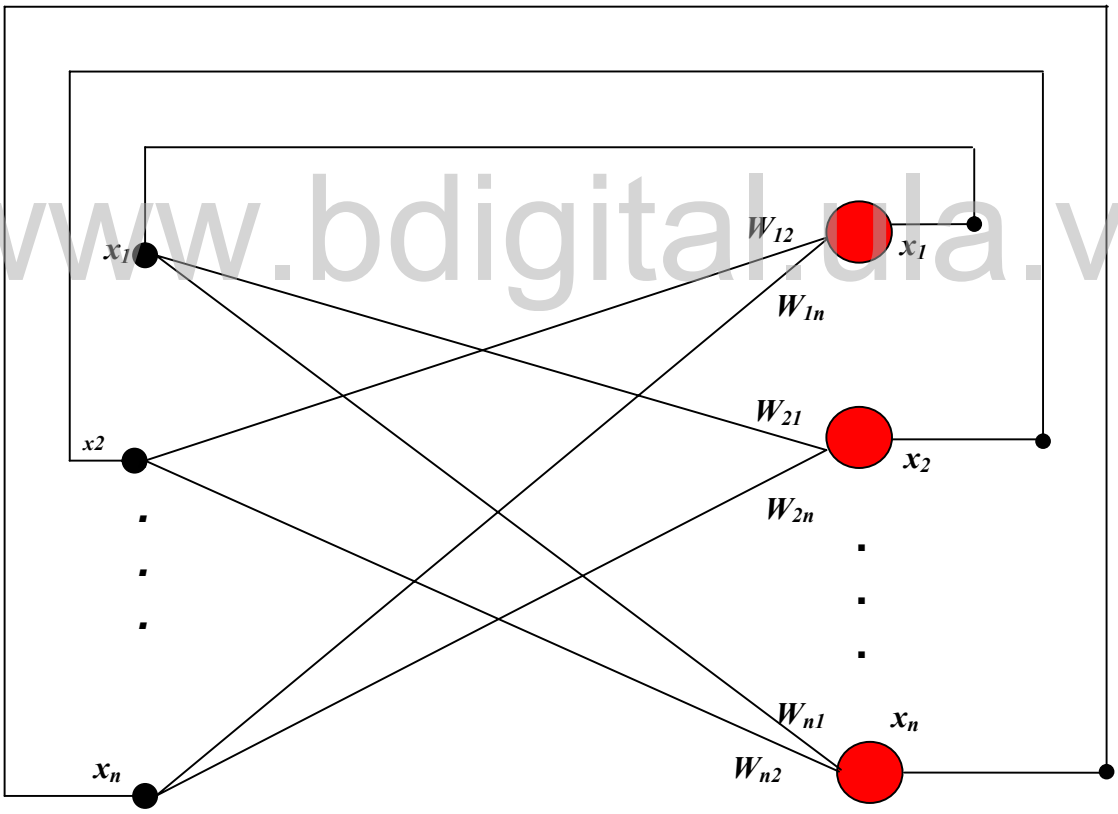


Figura 3.1 Arquitectura de una Red Neuronal Recurrente

Cada neurona de la red se encuentra conectada a todas las demás, pero no consigo misma (es decir, no existen conexiones autorecurrentes). Además, los pesos asociados a las conexiones

entre pares de neuronas son simétricos. Esto significa que el peso de la conexión de una neurona *i* con otra neurona *j* es igual valor que el de la conexión de la neurona *j* con la *i* ($W_{ij}=W_{ji}$).

3.2 FUNCIONAMIENTO DE LAS RNR.

Una de las características de las RNR es que se trata de una red autoasociativa [12]. De esta forma, informaciones diferentes pueden ser almacenadas en la red, como si fuera una memoria, durante la etapa de aprendizaje.

Luego, para recordar la información almacenada en la red, un patrón de entrada es aplicado y la salida de la red es inicializada. Seguidamente, el patrón de inicialización es eliminado y la salida inicial produce una nueva entrada actualizada a través de conexiones de retroalimentación (conexiones hacia atrás). La primera entrada actualizada produce la primera salida actualizada, donde de nuevo por retroalimentación se produce la segunda entrada actualizada. Este proceso de transición continúa hasta que la salida de las neuronas deja de cambiar lo que indica que la red ha alcanzado su equilibrio, es decir, se ha estabilizado.

3.3 LA FUNCIÓN DE ENERGÍA.

Muchas de las investigaciones acerca de la estabilidad de las RNR se basan en el establecimiento de una función, llamada función de energía computacional, la cual representa los posibles estados (punto de equilibrio) de la red.

La función de energía de J. Hopfield de una RNR discreta tiene la siguiente forma:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ i\neq i}}^{n} W_{ij}x_i x_j - \sum_{i=1}^{n}\theta_i x_i + \sum_{i=1}^{n}\tau_i x_i \quad (3.1a)$$

Donde:

W_{ij}: Peso de la conexión entre las neuronas *i* y *j*

x_i: Valor de la salida de la neurona *i*

x_j: Valor de la salida de la neurona *j*

θ_i: Entrada adicional a la neurona *i*

τ: Umbral de la función de activación de la neurona *i*

En cuanto a las RNR continuas, Hopfield considera como función de energía la siguiente:

$$E = -\sum_{i=1}^{n}\sum_{i\neq i} W_{ij}x_i(t)x_j(t)\Big/_2 - \sum_{i=1}^{n}\theta_i x_i(t) + \sum_{i=1}^{n}\int_0^{X_i} F^{-1}[x_i(t)]dx_i \qquad (3.1b)$$

Donde F^{-1} es la inversa de la función de activación sigmoidal (F) de la neurona *i*.

Estas expresiones guardan una profunda similitud con la energía mecánica clásica. Se trata de representar la evolución de la red, considerando cada salida de las neuronas como puntos en un espacio de dimensión *n* y relacionando el estado de la red en cada momento con un punto en ese espacio. La función de energía se puede imaginar como una superficie que presenta determinados valores mínimos. Cuando en la red se han almacenado *m* patrones, los posibles estados de la red serán también *m*. Estos *m* estados corresponden precisamente a los mismos de la función de energía. Cuando se presenta a la entrada de la red una nueva información, ésta evoluciona hasta alcanzar un mínimo de la función de energía y por lo tanto genera una salida estable.

3.4 REDES NEURONALES RECURRENTES PARA RESOLVER PROBLEMAS DE PROGRAMACIÓN LINEAL.

El principal interés en este apartado es la exploración de las características asintóticas de las soluciones generadas por una RNR para resolver problemas de P.L., así como también, desarrollar una RNR que sea capaz de generar soluciones óptimas a dichos problemas.

Para comenzar, se considera la forma general de un problema de P.L., el cual se describe como:

$$\begin{aligned} &\textit{Minimizar} && f = c^T x && (3.2)\\ &\textit{Sujeto a} && Ax = b && (3.3)\\ & && LI \le x \le LS && (3.4) \end{aligned}$$

donde:

$c \in \Re^n$ Vector columna correspondiente a los coeficientes de costos de la Función Objetivo *(f)*

$A \in \Re^m x \Re^n$ Matriz de coeficientes de las restricciones

$x \in \Re^{n+m}$ Vector columna correspondientes a las variables de decisión

$b \in \Re^m$ Vector correspondiente a la cantidad de recursos disponibles

$LI \in \Re^n$, $LS \in \Re^n$ Vectores columna correspondientes límite superior e inferior que puede tomar las variables de decisión.

Además $-\infty \le LI \le LS \le +\infty \quad para\ i = 1,2,...n$

Por otra parte por razones prácticas, se consideran sólo problemas de P.L. que sean operacionales, es decir, se asume que las soluciones admisibles (región factible) denotada por $\hat{X}$ es no vacía y que la función objetivo es acotada por debajo por la región factible $\hat{X}$. Todo esto implica que existe al menos una solución óptima a cada problema.

3.4.1 PROPIEDADES ASINTÓTICAS DE LAS RNR.

En este punto se comenzará con una representación formal de la RNR y se analizarán sus propiedades asintóticas en términos de la admisibilidad y optimalidad de las soluciones generadas a los problemas de programación lineal.

La representación formal de las RNR para optimización se describe como [13]:

$$v(t) = E[x(t), \lambda(t)] \quad (3.5)$$
$$x(t) = F[u(t)] \quad (3.6)$$
$$u(t) = G[t, x(t), \lambda(t)] \quad (3.7)$$
$$\lambda(t) = H[t, x(t)] \quad (3.8)$$

donde:

$v \in \Re$, es el índice de desempeño de la RNR

$x \in \Re^n$, corresponde a los estados de activación de la red (salidas)

$u \in \Re^n$, corresponde a las entradas de agregación a las neuronas

$\lambda \in \Re$, es un parámetro o variable de penalización

E: Regla de evaluación

F: Regla de activación

G: Regla de agregación

H: Regla de penalización

n es el número de neuronas artificiales.

Las condiciones iniciales $u(0)$, $\lambda(0)$ y las reglas $E,F,G,H,$ definen el comportamiento dinámico de las RNR.

Como se indicó en secciones anteriores, J. Hopfield propuso un modelo de RNR continua. Esta representación utiliza como función de evaluación (función de energía) una forma cuadrática, idéntica a la descrita en la Ec. (3.1b).

Dicha función de evaluación es una función de Liapunov (función de energía), que sirve como medida de abstracción de la energía para un sistema dinámico no lineal y es acotada y monótona decreciente con respecto del tiempo. Sin embargo, muchas veces una función de evaluación no es siempre una función de Liapunov, ya que una función de evaluación puede no ser monótona decreciente con respecto del tiempo.

La versión continua de la RNR de J. Hopfield usa una función sigmoidal como regla de activación, es decir, $F_i[u_i(t)] = \frac{1}{1+e^{(-u_i(t)/T_i)}}$, donde T_i es una constante positiva de escalamiento para la neurona *i*.

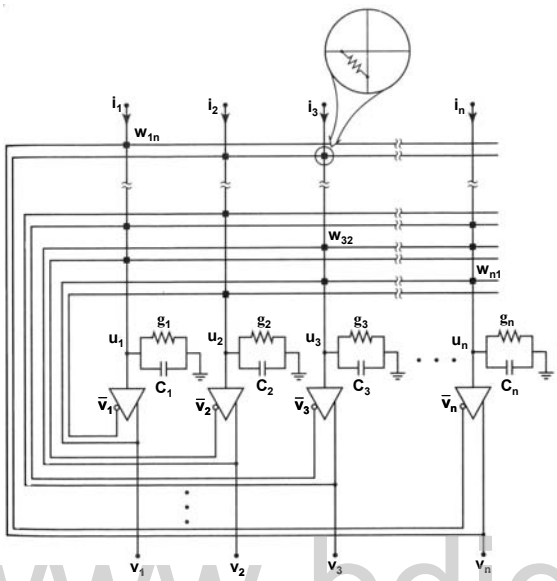


Figura 3.1a Modelo físico de una RNR

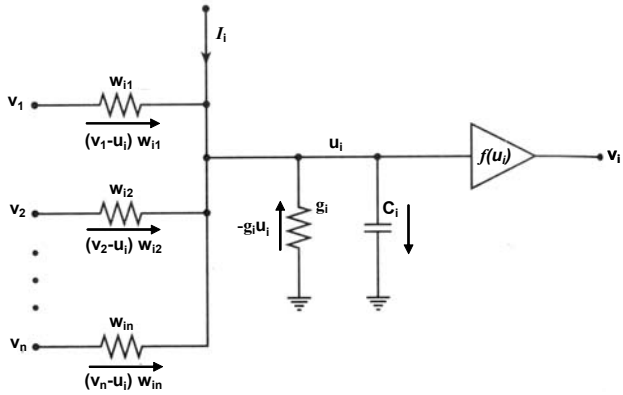


Figura 3.1b Fracción del modelo físico de una RNR: entrada de la i-ésima neurona

Por otro lado, de acuerdo a las Fig. 3.1a y Fig. 3.1b donde se observa el modelo físico de una RNR usando componentes electrónicos [12], se define la entrada a la i-ésima neurona, la cual siguiendo las leyes de Kirchoff viene dada por:

$$i_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} W_{ij} v_j(t) - u_i(t) \sum_{\substack{j=1 \\ j\neq i}}^{n} W_{ij} v_j(t) - u_i(t) g_i = C_i \dot{u}_i(t) \quad (3.9)$$

Reagrupando tenemos:

$$i_i + \sum_{\substack{j=1 \\ j\neq i}}^{n} W_{ij} v_j(t) - u_i(t) \left(\sum_{\substack{j=1 \\ j\neq i}}^{n} W_{ij} v_j(t) + g_i \right) = C_i \dot{u}_i(t) \quad (3.10)$$

Donde C_i y g_i son parámetros capacitivos y de conductancia para la neurona *i*, los cuales son una propiedad intrínseca de las neuronas biológicas y de la implementación física de la

neurona artificial [12]. u_i corresponde al voltaje de entrada mientras que v_j corresponde al voltaje de salida a través de la función de activación *f(u)*. El lado izquierdo de la Ec. (3.10) corresponde a todas las corrientes que entran en el capacitor C_i.

Ya que los pesos W_{ij} transforman el voltaje v_j-u_j en corrientes, ellos son considerados de igual forma conductancias, por tal motivo la conductancia total conectada a la neurona *i* es:

$$G_i = \sum_{j=1}^{n} W_{ij} + g_i$$

De esta forma la Ec.(3.10) puede ser simplificada como sigue:

$$i_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} W_{ij} v_j(t) - u_i(t) G_i = C_i \dot{u}_i(t) \qquad (3.11)$$

En el contexto de optimización, las RNR de Hopfield utilizan parámetros de penalización constantes para penalizar la violación de las restricciones [4]; esto implica que la regla de penalización en las RNR de Hopfield son invariantes en el tiempo, es decir H = constante.

El principal propósito de este enfoque neuronal es desarrollar una RNR cuyos estados estacionarios representan la solución al problema de P.L. Para esto, el proceso de solución del problema consiste en 2 fases:

FASE 1.

En la primera fase se realiza la reformulación del problema. Esta reformulación busca convertir el problema de P.L. en un problema irrestricto, incorporando las restricciones en la función de evaluación como un término a ser penalizado, es decir:

$$\textit{Minimizar} \qquad \min_{x \in X} E(x, \lambda) = c^T x + \lambda p(x) \qquad (3.12)$$

donde:

$E(x,\lambda)$ es la función de evaluación

$p(x)$ es la función de penalización

$\boldsymbol{X}$ es el espacio de estado definido por los *LI* y *LS* de la región factible $\widehat{X}$ del problema de programación lineal

$\boldsymbol{\lambda}$ es el parámetro de penalización.

FASE 2.

Esta fase tiene que ver con la estabilización de la RNR. Supongamos que una función de Liapunov existe para una RNR, y además que sus estados de activación son asintóticamente estables, es decir, al menos un estado estacionario existe. Esto indica que en esta fase, el estado de la RNR se mueve desde un estado inicial hacia un estado de equilibrio de acuerdo a una dinámica específica. El mayor objetivo del diseño de una RNR es asegurar que el estado estacionario represente la solución óptima al problema de P.L. La estabilización del estado es básicamente una búsqueda de la solución óptima de la función objetivo dentro de la región factible.

Para investigar el comportamiento asintótico de la RNR, es necesario conocer sobre la estabilidad asintótica de los estados de activación de la RNR.

La estabilidad de las RNR ha sido tratada por muchos investigadores. Hopfield demostró que su modelo de RNR continuo es asintóticamente estable si sus funciones de activación son continuas y monótonas crecientes, además de que las neuronas no tengan autoconexión y que los pesos sean simétricos ($W_{ii}=0$, $W_{ij}=W_{ji}$ para $i = 1,2,...n$). También ha sido justificado que una RNR en donde el parámetro de penalización sea monótono creciente con respecto al tiempo, es asintóticamente estable.

Para que una RNR lleve a cabo el procedimiento de solución de un problema de programación lineal, el estado estable de esta debe representar una solución admisible en la región factible, ahora bien, la estabilidad asintótica no garantiza que la solución generada por la red sea admisible u óptima. La admisibilidad y optimalidad vienen dadas por dos importantes teoremas [13].

TEOREMA 1. (Admisibilidad)

Sea la función de penalización:

$$p(x) = \frac{1}{2}(Ax-b)^T(Ax-b) = \frac{1}{2}(x^T A^T Ax - 2x^T A^T b + b^T b) \qquad (3.13)$$

y sean la función de evaluación y la función de agregación respectivamente:

$$E[x(t),\lambda(t)] = c^T x(t) + \lambda(t)p(x(t)) \qquad (3.14)$$

$$u(t) = G[x(t),\lambda(t)] = -\frac{1}{C_u}\int_0^t [c + \lambda(\tau)(A^T Ax(\tau) - A^T b)]p(x(\tau))d\tau \qquad (3.15)$$

donde:

C_u es un parámetro capacitivo positivo.

Si:

a) La función de activación es diferenciable y monótona creciente ($dF_i/du_i > 0$ para $-\infty < u_i < +\infty$, $i=1,2,..n$)

b) $\forall\, t \geq 0$, $\lambda(t) > 0$, $\lambda(t)$ es diferenciable

c) $p(x(t)) > 0$ implica que $\dot{\lambda}(t) \geq 0$

Entonces el estado estacionario de la RNR representa una solución admisible al problema de P.L.

TEOREMA 2. (Optimalidad)

Sea la función de penalización definida por la Ec. (3.13) y sean las funciones de evaluación y función de agregación definidas por las Ec. (3.14) y Ec. (3.15) respectivamente.

Si:

a) El estado inicial es una solución inadmisible $(x(0) \notin \hat{X})$

b) La función de activación es diferenciable y monótona creciente

c) $\forall\, t \geq 0$, $\lambda(t) > 0$, $\lambda(t)$ es diferenciable

d) $\dot{x}(t) \neq 0$ implica $\dot{\lambda}(t) = 0$, y $\dot{x}(t) = 0$ y $p(x(t)) > 0$ implica que $\dot{\lambda}(t) > 0$

Entonces el estado estacionario de la RNR representa la solución óptima al problema de programación lineal.

De esta forma el teorema 1 asegura que comenzando con una solución inicial inadmisible, el estado estacionario siempre será una solución admisible basada en las leyes propias de los estados transitorios. El teorema 2 conforma una serie de condiciones suficientes para lograr el óptimo global de las soluciones generadas por una RNR, lo que quiere decir, que si estas condiciones son satisfechas, cualquier estado estacionario de la RNR representa la solución óptima al problema. Para el caso de soluciones óptimas alternativas, la RNR sólo puede generar una de ellas la cual dependerá de las condiciones iniciales establecidas.

3.4.2 CONFIGURACIÓN DE LA RNR.

Para resolver un problema de optimización vía RNR la clave está en convertir el problema en una RNR cuyos estados estacionarios representan la solución ha dicho problema. En este apartado mostraremos la configuración de la RNR derivada de la representación formal de la sección anterior.

De acuerdo con los teoremas 1 y 2 la función de evaluación *E* y la función de agregación *G* en la RNR son dadas en las Ec. (3.14) y Ec. (3.15) respectivamente. La función de agregación es definida por el gradiente de la función de evaluación *E*.

Así mismo, de acuerdo a los teoremas 1 y 2 la función de activación $Fi[u_i(t)]$ debe ser diferenciable y monótona creciente con respecto a $u_i(t)$ para $-\infty < u_i(t) < +\infty$. La selección de esta función de activación también depende del conjunto de soluciones admisible del problema bajo estudio. Los límites inferior (LI) y superior (LS) explícitos tales como $x \geq 0$ pueden ser representados seleccionando funciones de activación en un rango apropiado. La siguiente tabla muestra posible funciones de activación para diferentes rangos de soluciones admisibles [13], donde F_i es el i-ésimo elemento de la función de activación para las neurona *i*, M_i y η_i son constantes positivas para $i = 1,2,..,n$.

X_i	$F_i(u_i)$	X_i	$F_i(u_i)$
$[-\infty,\infty,]$	$\eta_i u_i$	$[-M_i, M_i]$	$M_i(1-e^{-\eta_i u_i}) \big/ 1+e^{-\eta_i u_i}$
$[-\infty,0]$	$-e^{\eta_i u}$	$[0, \infty]$	$e^{\eta_i u}$
$[-M_i,0]$	$-M_i e^{-\eta_i u_i} \big/ 1+e^{-\eta_i u_i}$	$[0, M_i]$	$M_i \big/ 1+e^{-\eta_i u_i}$

Tabla 3.1 Funciones de Activación

Ya que $\lambda(t)$ puede cambiar el panorama de la función de energía $E[(x(t), \lambda(t)]$ en el espacio estado de *X*, la selección de la regla de penalización *H*, es un elemento importante. Nuevamente, los teoremas 1 y 2 ofrecen suficientes condiciones para seleccionar la variable de penalización $\lambda(t)$, requiriendo que esta sea monótona creciente con respecto al tiempo.

Tomando en cuenta esto, tres posibles funciones para la variable de penalización [13] son presentadas en las siguientes ecuaciones:

$$\lambda(t) = \int_0^t \frac{\mu}{p(x(\tau))} \frac{d\tau}{C_\lambda} \qquad (3.16)$$

$$\lambda(t) = \int_0^t \mu p(x(\tau)) \frac{d\tau}{C_\lambda} \qquad (3.17)$$

$$\lambda(t) = \frac{\alpha^{\mu t}}{C_\lambda} \qquad (3.18)$$

Donde $C_\lambda > 0$, $\mu > 0$ y $\alpha > 1$

Luego de haber establecido las condiciones previas, ahora se procederá a describir con detalle la representación de la RNR presentada en este trabajo para resolver problemas de P.L. Las ecuaciones de estado de la RNR basándose en las Ec. (3.15) y Ec. (3.16) son presentadas a continuación:

$$C_u \dot{u}(t) = -(c + \lambda(t)(A^T Ax(t) - A^T b)) \frac{1}{2}(x^T A^T Ax - 2x^T A^T b + b^T b) \qquad (3.19)$$

$$C_\lambda \dot{\lambda}(t) = \frac{2}{(x^T A^T Ax - 2x^T A^T b + b^T b)} \qquad (3.20)$$

Donde $C_\mu << C_\lambda$ son parámetros capacitivos escalares.

Sustituyendo la Ec. (3.20) en (3.19) podemos reescribir las ecuaciones de la siguiente forma:

$$C_u \dot{u}(t) = \frac{-c - \lambda(t)(A^T Ax(t) - A^T b)}{C_\lambda \dot{\lambda}(t)} \qquad (3.21)$$

$$C_\lambda \dot{\lambda}(t) = \frac{2}{(x^T A^T Ax - 2x^T A^T b + b^T b)} \qquad (3.22)$$

Los pesos de conexión de la RNR están determinados por la Ec. (3.21) y se observa que dependen de $\lambda(t)$, $\dot{\lambda}(t)$, A, b y c. La variable de penalización $\lambda(t)$ es parte esencial de los pesos $W_{ij}(t)$ entre las neuronas i y j, los cuales son variantes en el tiempo.

La RNR de la Fig.3.2 que se usará para resolver problemas de P.L. consiste en 1 sola capa, $n + 2$ neuronas masivamente conectadas, donde n neuronas corresponden a n elementos de los estados de activación y otras 2 neuronas ficticias que corresponden al $\lambda(t)$ y $\dot{\lambda}(t)$.

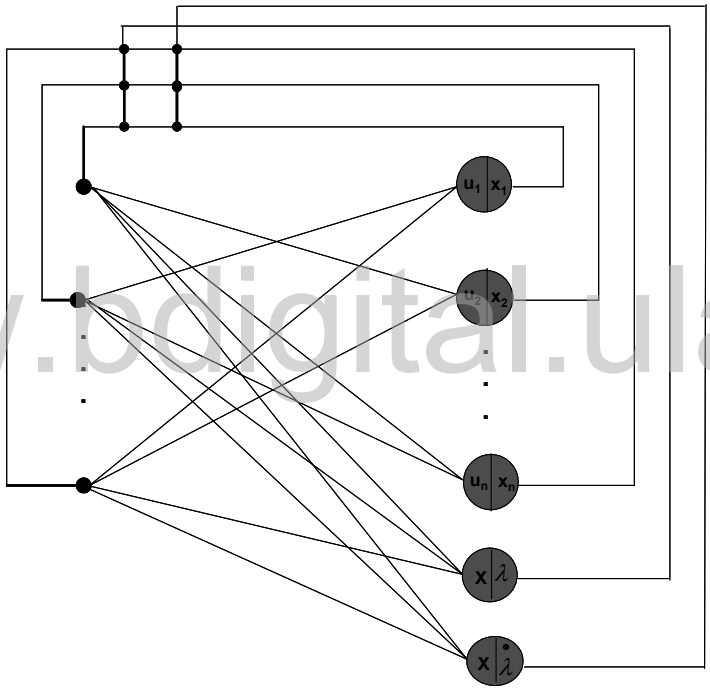


Figura 3.2 Red Neuronal Recurrente para resolver problemas de P.L.

Los parámetros capacitivos C_u y C_λ son usados para escalar la RNR. Debido a que la constante de tiempo del sistema dinámico es monótonamente creciente con respecto a los parámetros capacitivos C_u y C_λ, escoger C_u y C_λ suficientemente pequeños puede hacer que la RNR estabilice rápidamente.

La diferencia notable entre las RNR de J. Hopfield para optimización y la RNR estudiada en este trabajo radica en el tratamiento de la regla de penalización. En las RNR de J.Hopfield la penalización es implementada por un parámetro de penalización constante, trayendo como consecuencia que en muchos casos la RNR tenga problemas de estabilidad. En la RNR tratada en este trabajo, la penalización es conseguida mediante una variable de penalización monótona creciente.

3.4.3 RESOLUCIÓN DE PROBLEMAS DE PROGRAMACIÓN LINEAL.

En esta sección, se demostrará el desempeño de las RNR a través de la resolución de problemas de P.L. con diferentes configuraciones de variables de decisión y restricciones.

3.4.3.1 PROBLEMA CON 2 VARIABLES Y 3 RESTRICCIONES.

$$\begin{aligned} &\textit{Minimizar} \quad -3x_1 - 2x_2 \\ &\textit{Sujeto a}: \\ &\qquad x_1 + 3x_2 \le 5 \\ &\qquad 3x_1 + 2x_2 \le 8 \\ &\qquad 2x_1 + x_2 \le 3 \\ &\qquad x_1 \ge 0,\ x_2 \ge 0 \end{aligned}$$

Primero debemos colocar el problema en su forma estándar, es decir, llevando las restricciones a la forma $Ax - b=0$, agregando para ello, las variables de holgura necesarias.

De esta forma las restricciones quedan así:

$$\begin{aligned} x_1 + 3x_2 + x_3 - 5 &= 0 \\ 3x_1 + 2x_2 + x_4 - 8 &= 0 \\ 2x_1 + x_2 + x_5 - 3 &= 0 \end{aligned}$$

La solución optima a este problema mediante el método simplex es $x_1=0.8$, $x_2=1.4$, $x_3=0$, $x_4=0$, $x_5=0.5$ y $f(x)=c^Tx=-5.2$

Luego reformulamos el problema de programación lineal introduciendo las restricciones como una función de penalización en la función objetivo del problema, convirtiéndola en la función de evaluación E de la RNR.

De acuerdo a la Ec.(3.13) la función de penalización es:

$$p(x) = \frac{1}{2}[(x_1 + 3x_2 + x_3 - 5)^2 + (3x_1 + 2x_2 + x_4 - 8)^2 + (2x_1 + x_2 + x_5 - 3)^2]$$

Por lo tanto de acuerdo a la Ec.(3.14) la función de evaluación queda:

$$E(x,\lambda) = -3x_1 - 2x_2 + \frac{\lambda}{2}[(x_1 + 3x_2 + x_3 - 5)^2 + (3x_1 + 2x_2 + x_4 - 8)^2 + (2x_1 + x_2 + x_5 - 3)^2]$$

De acuerdo a la Tabla 3.1 se selecciona como función de activación:

$$x(t) = F(u) = e^{u(t)} \text{ para } x(t)=[0,\infty]$$

Se fijan los parámetros $C_u=0.01$, $C_\lambda = 1$, y de acuerdo con las Ec.(3.21) y Ec. (3.22) tenemos el siguiente sistema no lineal de ecuaciones diferenciales que representan la RNR para resolver este problema.

$$x_1(t) = e^{u_1(t)}$$

$$x_2(t) = e^{u_2(t)}$$

$$x_3(t) = e^{u_3(t)}$$

$$x_4(t) = e^{u_4(t)}$$

$$x_5(t) = e^{u_5(t)}$$

$$\dot{u}_1(t) = \frac{100[3 - \lambda(t)(14x_1(t) + 11x_2(t) + x_3(t) + 3x_4(t) + 2x_5(t) - 35)]}{\dot{\lambda}(t)}$$

$$\dot{u}_2(t) = \frac{100[2 - \lambda(t)(11x_1(t) + 14x_2(t) + 3x_3(t) + 2x_4(t) + x_5(t) - 34)]}{\dot{\lambda}(t)}$$

$$\dot{u}_3(t) = \frac{-100\lambda(t)(x_1(t) + 3x_2(t) + x_3(t) - 5)}{\dot{\lambda}(t)}$$

$$\dot{u}_4(t) = \frac{-100\lambda(t)(3x_1(t) + 2x_2(t) + x_4(t) - 8)}{\dot{\lambda}(t)}$$

$$\dot{u}_5(t) = \frac{-100\lambda(t)(2x_1(t) + x_2(t) + x_5(t) - 3)}{\dot{\lambda}(t)}$$

$$\dot{\lambda}(t) = \frac{2}{(x_1 + 3x_2 + x_3 - 5)^2 + (3x_1 + 2x_2 + x_4 - 8)^2 + (2x_1 + x_2 + x_5 - 3)^2}$$

$$E(x(t), \lambda(t)) = -3x_1(t) - 2x_2(t) + \frac{\lambda(t)}{2}[(x_1(t) + 3x_2(t) + x_3(t) - 5)^2 + (3x_1(t) + 2x_2(t) + x_4(t) - 8)^2 + (2x_1(t) + x_2(t) + x_5(t) - 3)^2]$$

Para resolver este sistema de ecuaciones diferenciales se hace uso del software MatLab 7.0, se realizan cinco corridas cambiando las condiciones iniciales de las $x_i(0)$. Cabe señalar que las condiciones iniciales de las 3 variables de holgura $x_3(0)$, $x_4(0)$, $x_5(0)$ se generan de forma aleatoria para cumplir con la condición a) del Teorema 1 referente a la optimalidad de la solución, que se refiere a que el estado inicial debe ser una solución inadmisible del problema.

De esta forma tenemos:

CORRIDA # 1.

Condiciones iniciales: $x_1(0)=0.1589$, $x_2(0)=0.3351$, $x_3(0)=0.3062$, $x_4(0)=1.2840$, $x_5(0)= 0.1231$, $\lambda(0)=0.2$

Solución vía simplex*:* $x_1=0.8$, $x_2=1.4$, $x_3=0$, $x_4=0$, $x_5=0.5$ *y* $f(x)=-5.2$
Solución vía RNR*:* ***$x_1=0.8$, $x_2=1.4$, $x_3=0$, $x_4=0$, $x_5=0$ $E=-5.1156$, $f(x)=-5.2$***

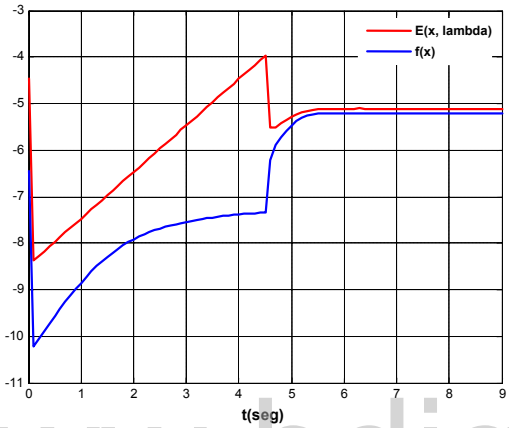


Figura 3.3 Función de energía (en rojo) y Función objetivo (en azul)

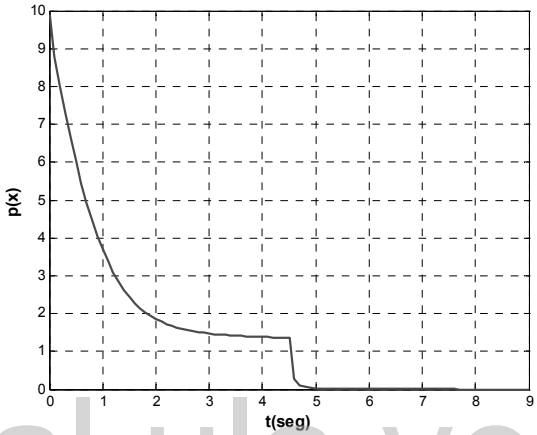


Figura 3.4 Función de penalización

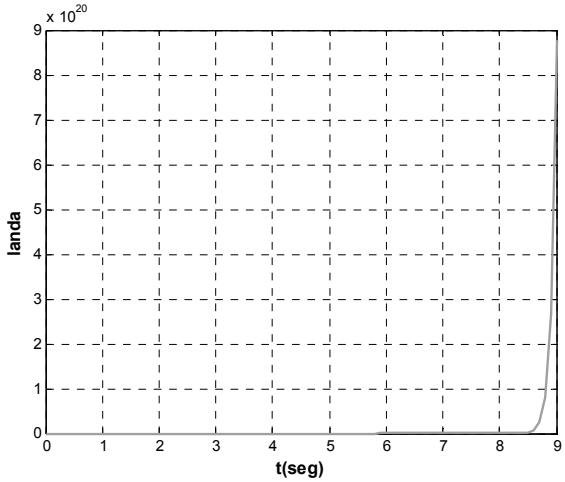


Figura 3.5 Variable de penalización

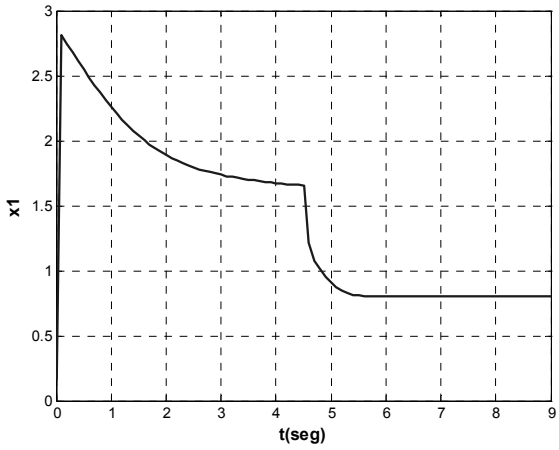


Figura 3.6 Respuesta de variable de decisión x_1

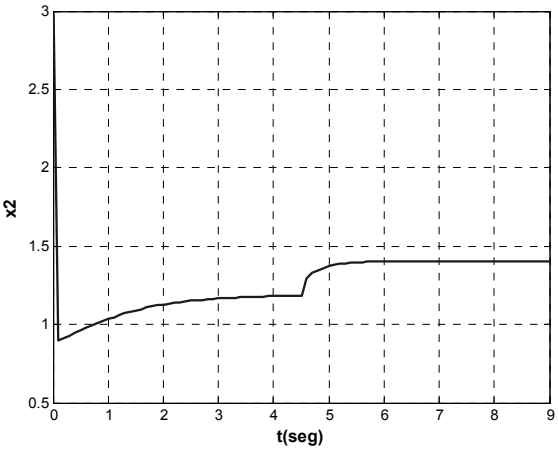


Figura 3.7 Respuesta de variable de decisión x_2

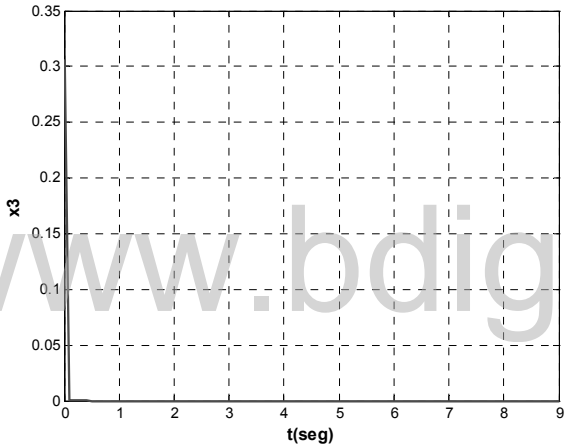


Figura 3.8 Respuesta de variable de holgura x_3

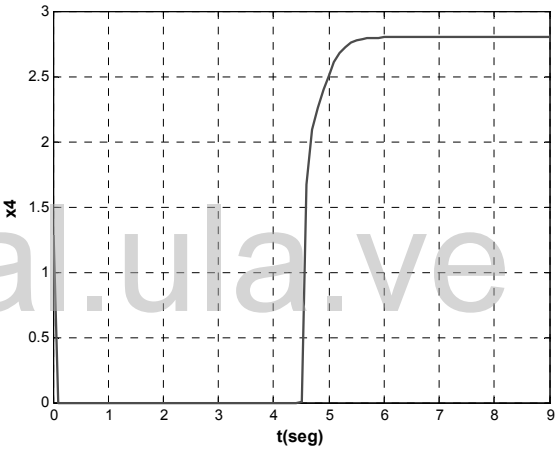


Figura 3.9 Respuesta de variable de holgura x_4

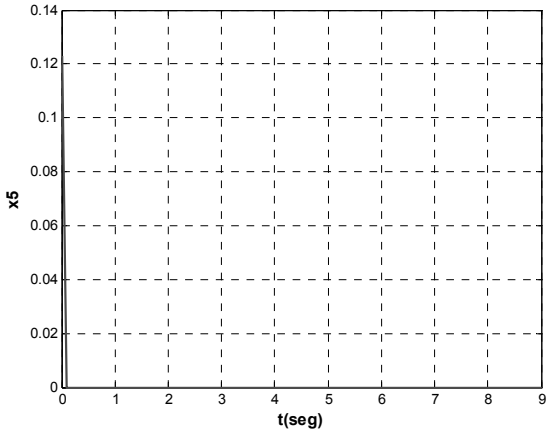


Figura 3.10 Respuesta variable de holgura x_5

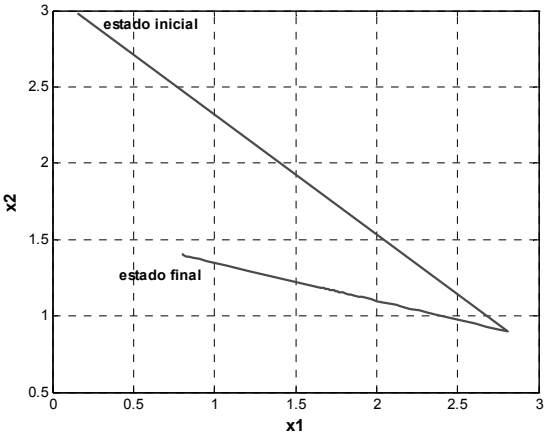


Figura 3.11 Diagrama de estado x_1-x_2

CORRIDA # 2.

Condiciones iniciales: $x_1(0)=0.2165$, $x_2(0)= 4.4817$, $x_3(0)=0.2725$, $x_4(0)=1.4918$, $x_5(0)= 0.1237$, $\lambda(0)=0.2$

Solución vía simplex: $x_1=0.8$, $x_2=1.4$, $x_3=0$, $x_4=0$, $x_5=0.5$ y $f(x)=-5.2$

Solución vía RNR: $x_1=0.8$, $x_2=1.4$, $x_3=0$, $x_4=0$, $x_5=0$ $E=-5.118$, $f(x)=-5.2$

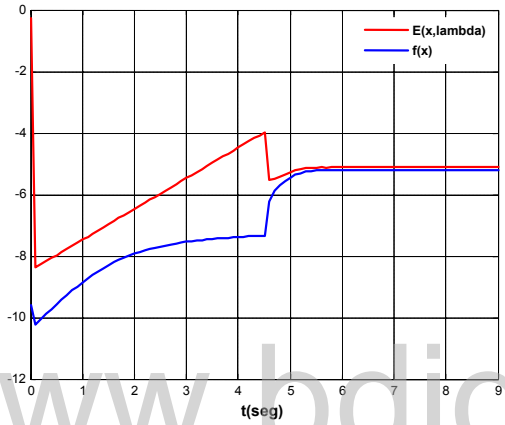


Figura 3.12 Función de energía (en rojo) y Función objetivo (en azul)

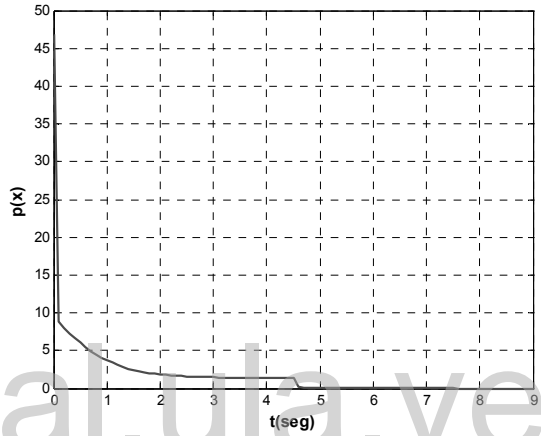


Figura 3.13 Función de penalización

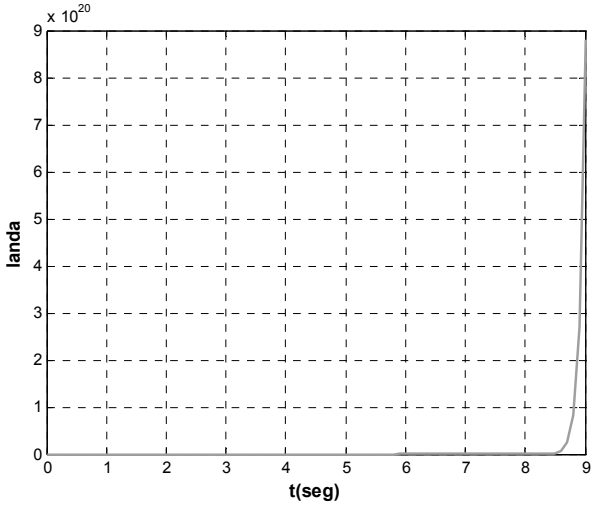


Figura 3.14 Variable de penalización

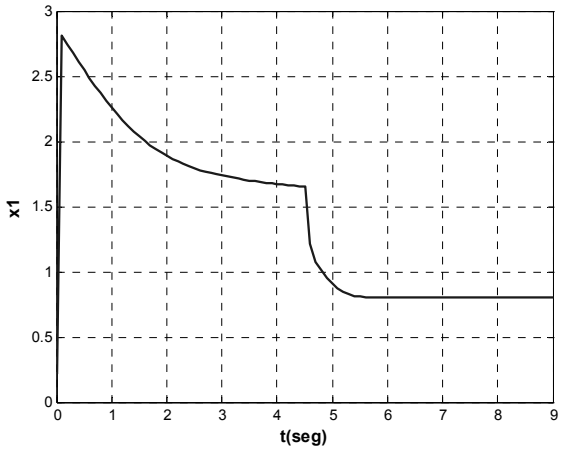


Figura 3.15 Respuesta de variable de decisión x_1

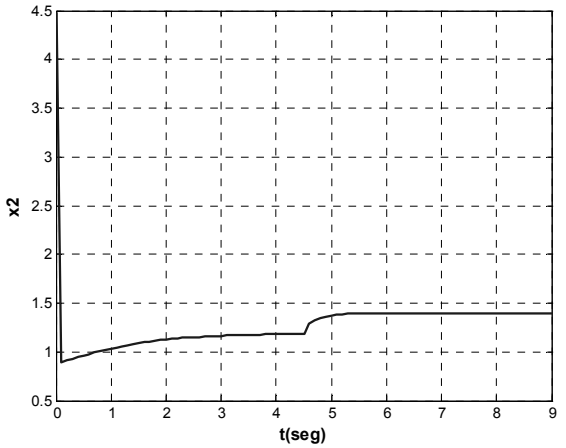


Figura 3.16 Respuesta de variable de decisión x_2

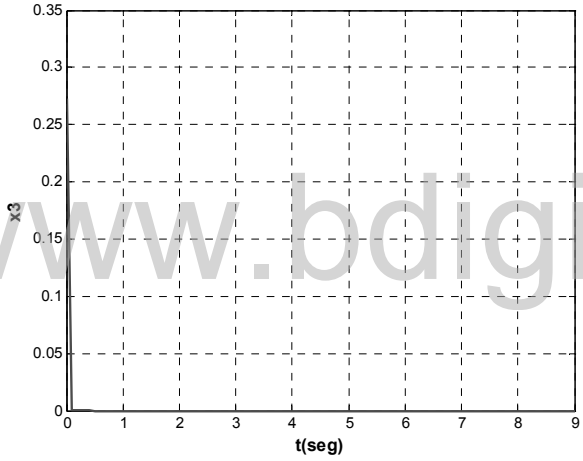


Figura 3.17 Respuesta de variable de holgura x_3

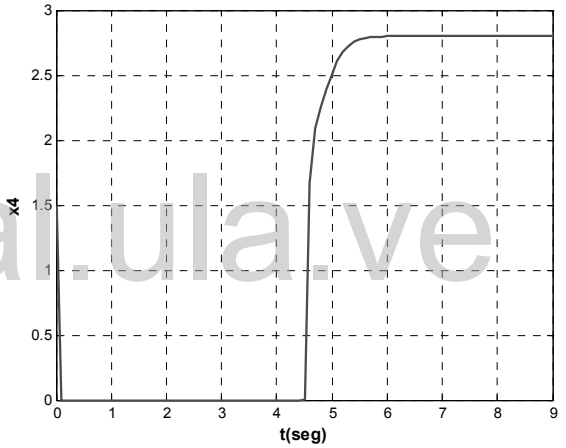


Figura 3.18 Respuesta de variable de holgura x_4

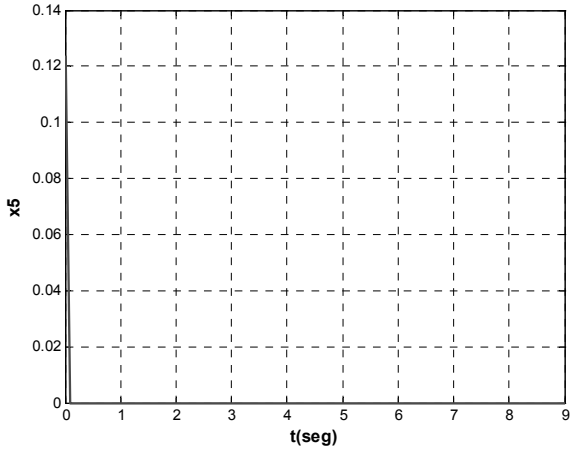


Figura 3.19 Respuesta de variable de holgura x_5

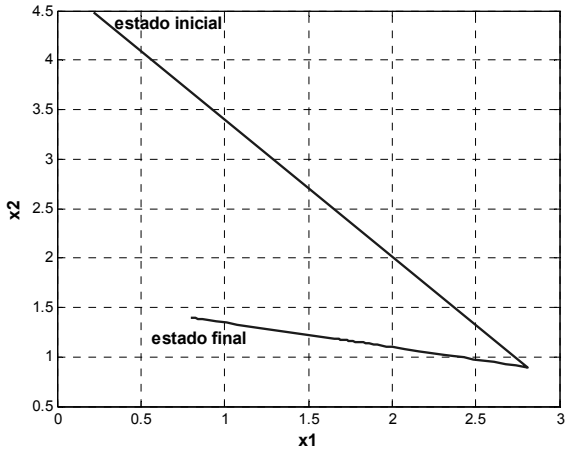


Figura 3.20 Diagrama de estado x_1-x_2

Los resultados de las simulaciones demuestran que el estado estacionario de la RNR converge hacia la solución óptima del problema para distintas condiciones iniciales de las variables de decisión. En las Fig. 3.3 y 3.12, se observa que la función de evaluación *E,* así como la función objetivo *f,* converge hacia la solución óptima. La pequeña discrepancia se debe al valor de la función de penalización *p(x)* multiplicado por la variable de penalización λ(t) (que es monótona creciente) en la función de evaluación *E*. Así mismo en las Fig. 3.4 y 3.13, como era de esperarse, *p(x)* es casi despreciable (pero no llega a 0). Además se demuestra que las RNR en su evolución minimizan una función de energía computacional.

En las Fig. 3.6 hasta la Fig. 3.10 y en las Fig. 3.15 hasta la Fig. 3.19 se observa que, partiendo de distintas condiciones iniciales, el estado estacionario es la solución óptima de las variables de decisión y variables de holgura del problema. De igual forma en las Fig. 3.11 y Fig. 3.20 tenemos el gráfico de trayectorias donde se observa que ambas convergen en su estado final hacia la solución óptima de dichas variables, a pesar de que cierto comportamiento caótico puede ser observado en ellas.

La siguiente tabla muestra un resumen de todas las corridas donde se evidencia la convergencia de la RNR hacia la solución óptima para distintas condiciones iniciales, con el fin de reforzar lo anteriormente explicado mediante las gráficas de las simulaciones. Sin embargo, se observa la importancia que tiene la condición inicial de la variable de penalización *λ(t)*, ya que para valores < 0.2 la RNR no estabiliza en la solución óptima del problema.

λ(0)	$X_1(0)$	$X_2(0)$	$X_3(0)$	$X_4(0)$	$X_5(0)$	X_1	X_2	X_3	X_4	X_5	E(x)	f(x)
0.2	0.1589	0.3351	0.3062	1.2840	0.1231	0.8	1.4	0	2.8	0	-5.1156	-5.20
0.2	0.2165	4.4817	0.2725	1.4918	0.1237	0.8	1.4	0	2.8	0	-5.1168	-5.20
0.3	0.9795	11.007	6.8881	2.3857	6.7382	0.8	1.4	0	2.8	0	-5.1167	-5.20
0.1	**2.6154**	**0.3874**	**0.2832**	**0.3857**	**1.2269**	**1.618**	**1.195**	**0**	**0**	**0**	**-2.2699**	**-7.245**
0.08	**5.864**	**15.504**	**1.1451**	**9.7855**	**0.1405**	**1.616**	**1.196**	**0**	**0**	**0**	**-2.1295**	**-7.240**

Tabla 3.2 Resumen de las 5 corridas realizadas (2 variables)

3.4.3.2 PROBLEMA CON 3 VARIABLES Y 3 RESTRICCIONES.

$$\begin{aligned} &\textit{Minimizar} \quad -3x_1 - x_2 - 3x_3 \\ &\textit{Sujeto a}: \\ &\qquad 2x_1 + x_2 + x_3 \le 2 \\ &\qquad x_1 + 2x_2 + 3x_3 \le 5 \\ &\qquad 2x_1 + 2x_2 + x_3 \le 6 \\ &\qquad x_1 \ge 0,\ x_2 \ge 0, x_3 \ge 0 \end{aligned}$$

Se coloca el problema en su forma estándar, agregando para ello, las variables de holgura necesarias.

De esta forma las restricciones quedan así:

$$\begin{aligned} 2x_1 + x_2 + x_3 + x_4 - 2 = 0 \\ x_1 + 2x_2 + 3x_3 + x_5 - 5 = 0 \\ 2x_1 + 2x_2 + x_3 + x_6 - 6 = 0 \end{aligned}$$

La solución óptima a este problema mediante el método simplex es $x_1=0.2,\ x_2=0.0,\ x_3=1.6,\ x_4=0,\ x_5=0.0,\ x_6=4.0$ y $f(x)=c^Tx= -5.4$

Se reformula el problema de programación lineal introduciendo las restricciones como una función de penalización en la función objetivo del problema, convirtiéndola en la función de evaluación *E* de la RNR.

De acuerdo a la Ec.(3.13) la función de penalización es:

$$p(x) = \frac{1}{2}[(2x_1 + x_2 + x_3 + x_4 - 2)^2 + (x_1 + 2x_2 + 3x_3 + x_5 - 5)^2 + (2x_1 + 2x_2 + x_3 + x_6 - 6)^2]$$

Por lo tanto de acuerdo a la Ec.(3.14) la función de evaluación queda:

$$E(x,\lambda) = -3x_1 - x_2 - 3x_3 + \frac{\lambda}{2}[(2x_1 + x_2 + x_3 + x_4 - 2)^2 + (x_1 + 2x_2 + 3x_3 + x_5 - 5)^2 + (2x_1 + 2x_2 + x_3 + x_6 - 6)^2]$$

De acuerdo a la Tabla 3.1 se selecciona como función de activación:

$$x(t)=F(u) = e^{u(t)} \text{ para } x(t) =[0 ,\infty]$$

Se fijan los parámetros $C_u=0.01$, $C_\lambda = 1$, y de acuerdo con la Ec.(3.21) y Ec.(3.22), tenemos el siguiente sistema no lineal de ecuaciones diferenciales que representan la RNR para resolver este problema.

$$x_1(t) = e^{u_1(t)}$$

$$x_2(t) = e^{u_2(t)}$$

$$x_3(t) = e^{u_3(t)}$$

$$x_4(t) = e^{u_4(t)}$$

$$x_5(t) = e^{u_5(t)}$$

$$x_6(t) = e^{u_6(t)}$$

$$\dot{u_1}(t) = \frac{100[3 - \lambda(t)(9x_1(t) + 8x_2(t) + 7x_3(t) + 2x_4(t) + x_5(t) + 2x_6(t) - 21)]}{\dot{\lambda}(t)}$$

$$\dot{u_2}(t) = \frac{100[1 - \lambda(t)(8x_1(t) + 9x_2(t) + 9x_3(t) + x_4(t) + 2x_5(t) + 2x_6(t) - 24)]}{\dot{\lambda}(t)}$$

$$\dot{u_3}(t) = \frac{100[3 - \lambda(t)(7x_1(t) + 9x_2(t) + 11x_3(t) + x_4(t) + 3x_5(t) + x_6(t) - 23)]}{\dot{\lambda}(t)}$$

$$\dot{u}_4(t) = \frac{-100\lambda(t)(2x_1(t) + x_2(t) + x_3(t) + x_4(t) - 2)}{\dot{\lambda}(t)}$$

$$\dot{u}_5(t) = \frac{-100\lambda(t)(x_1(t) + 2x_2(t) + 3x_3(t) + x_5(t) - 5)}{\dot{\lambda}(t)}$$

$$\dot{u}_6(t) = \frac{-100\lambda(t)(2x_1(t) + 2x_2(t) + x_3(t) + x_6(t) - 6)}{\dot{\lambda}(t)}$$

$$\dot{\lambda}(t) = \frac{2}{(2x_1 + x_2 + x_3 + x_4 - 2)^2 + (x_1 + 2x_2 + 3x_3 + x_5 - 5)^2 + (2x_1 + 2x_2 + x_3 + x_6 - 6)^2}$$

$$E(x(t), \lambda(t)) = -3x_1(t) - x_2(t) - 3x_3(t) + \frac{\lambda}{2}[(2x_1(t) + x_2(t) + x_3(t) + x_4(t) - 2)^2 + (x_1(t) + 2x_2(t) + 3x_3(t) + x_5(t) - 5)^2 + (2x_1(t) + 2x_2(t) + x_3(t) + x_6(t) - 6)^2]$$

Para resolver este sistema de ecuaciones diferenciales se hace uso nuevamente del software MatLab 7.0. Se realizan cinco corridas cambiando las condiciones iniciales de las $x_i(0)$ y de $\lambda(0)$. Cabe señalar que las condiciones iniciales de las 3 variables de holgura $x_3(0)$, $x_4(0)$, $x_5(0)$, $x_6(0)$ se generan de forma aleatoria para cumplir con la condición a) del Teorema 1 referente a la optimalidad de la solución, que se refiere a que el estado inicial debe ser una solución inadmisible del problema.

De esta forma tenemos:

CORRIDA # 1.

Condiciones iniciales: $x_1(0)=0$, $x_2(0)=0$, $x_3(0)=0.05$, $x_4(0)=0.0463$, $x_5(0)=0.0001$, $x6(0)=0.0043$, $\lambda(0)=0.2$

Solución vía simplex*:* $x_1=0.2$, $x_2=0.0$, $x_3=1.6$, $x_4=0$, $x_5=0.0$, $x_6=4.0$ y $f(x)= -5.4$

Solución vía RNR*:* ***x1=0.2, x_2=0.0, x_3=1.6, x_4=0, x_5=0.0, x_6=4.0*** **y** ***f(x)= -5.4 E=-5.2847***

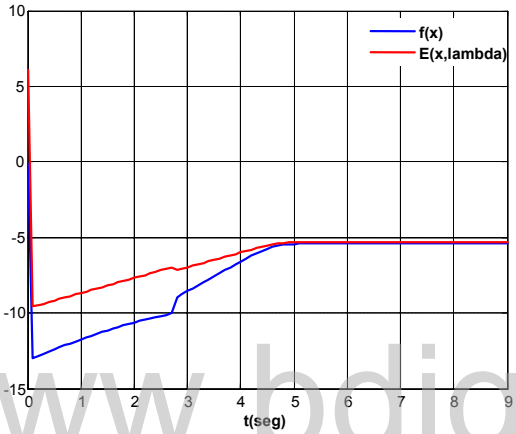


Figura 3.21 Función de energía (en rojo) y Función objetivo (en azul)

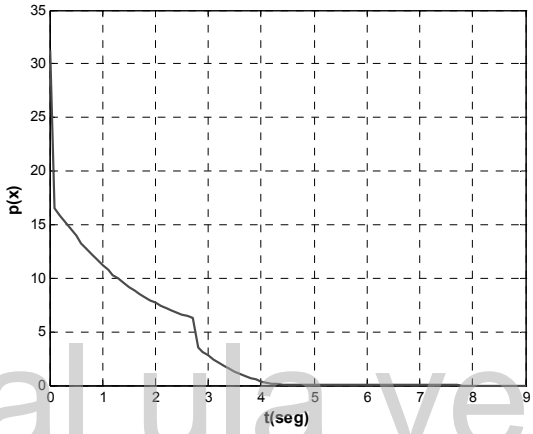


Figura 3.22 Función de penalización

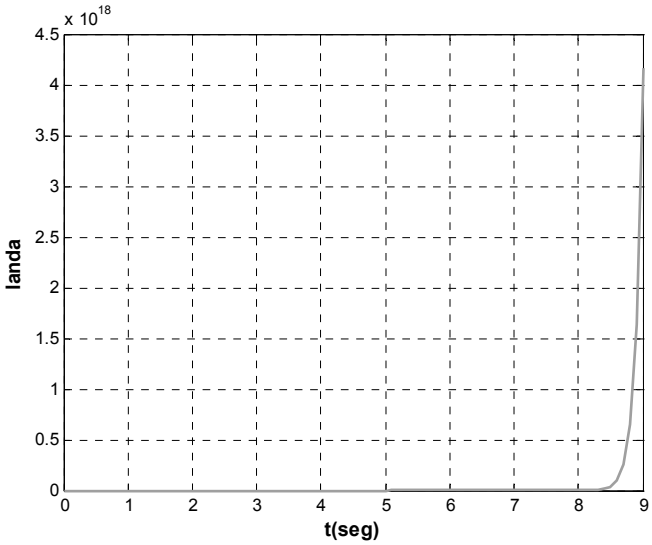


Figura 3.23 Variable de penalización

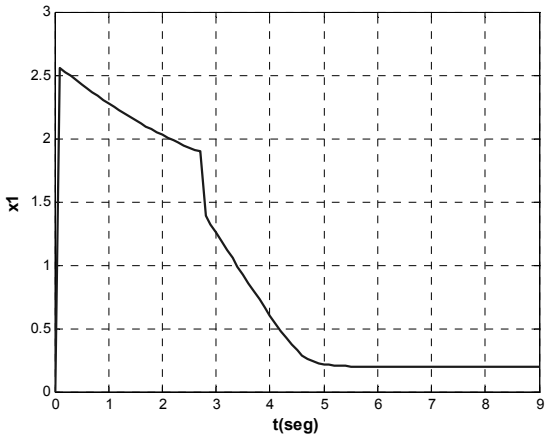


Figura 3.24 Respuesta de variable de decisión x_1

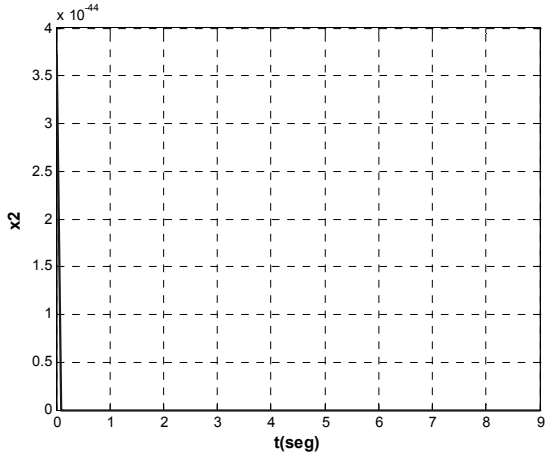


Figura 3.25 Respuesta de variable de decisión x_2

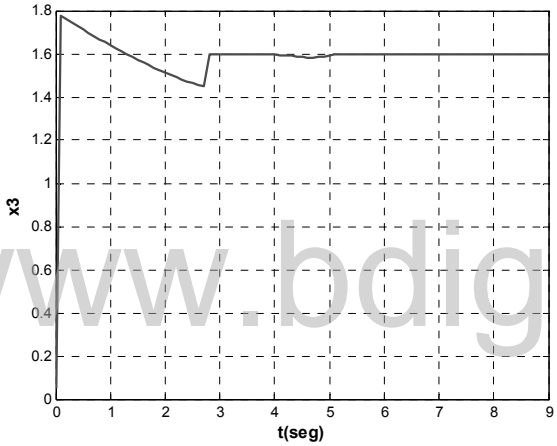


Figura 3.26 Respuesta de variable de decisión x_3

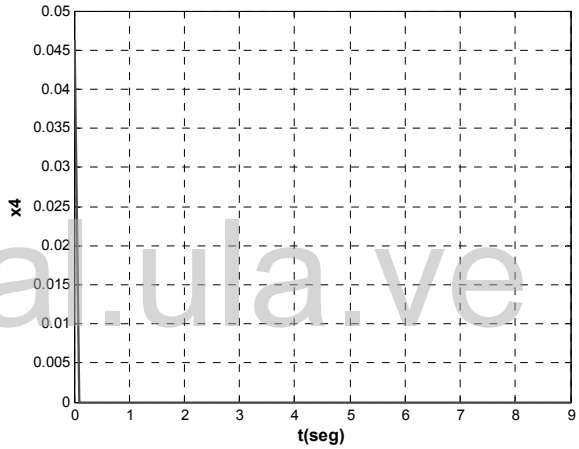


Figura 3.27 Respuesta de variable de holgura x_4

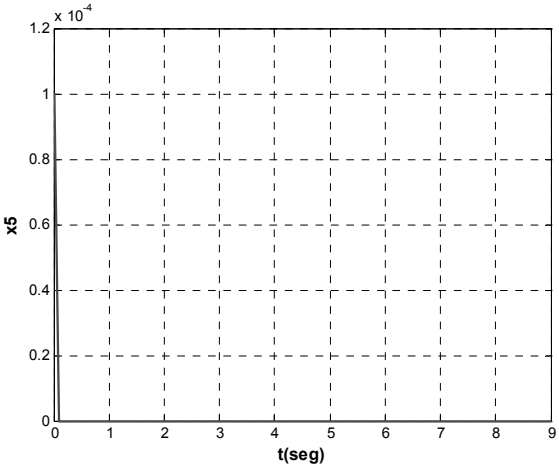


Figura 3.28 Respuesta de variable de holgura x_5

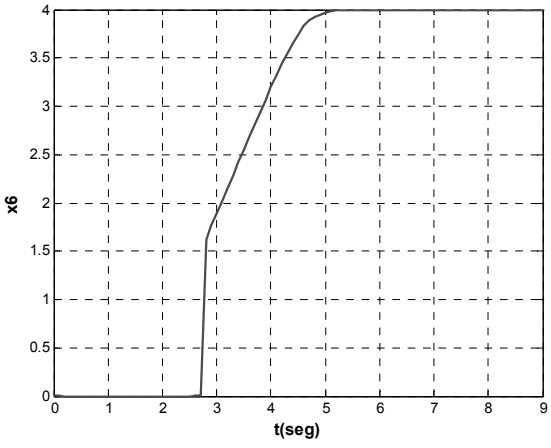


Figura 3.29 Respuesta de variable de holgura x_6

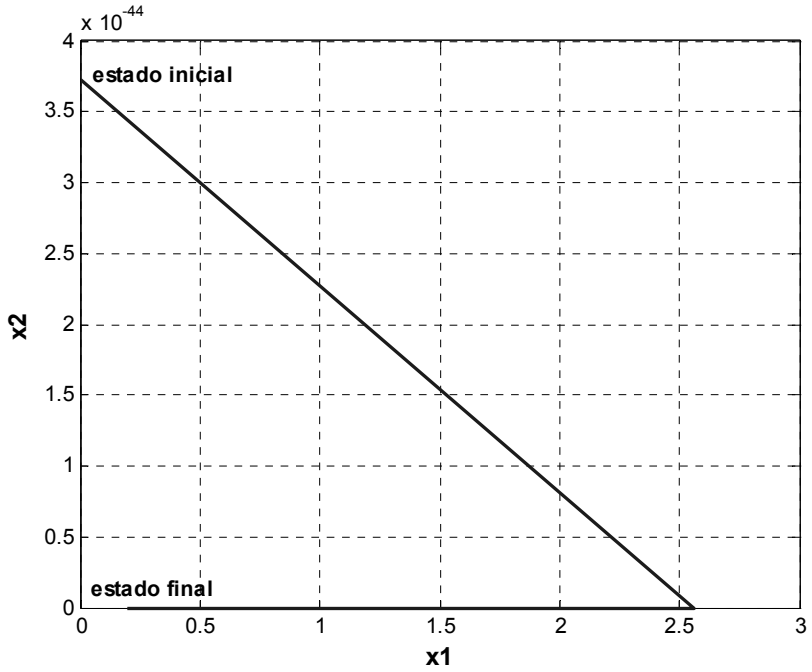


Figura 3.30 Diagrama de estado x_1-x_2

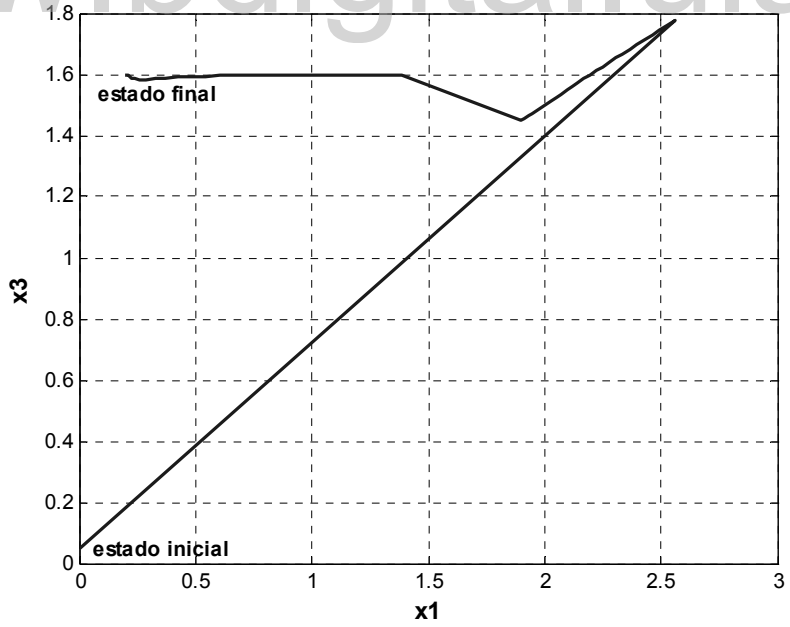


Figura 3.31 Diagrama de estado x_1-x_3

CORRIDA # 2.

Condiciones iniciales: $x_1(0)=1.5$, $x_2(0)=1$, $x_3(0)=0.5$, $x_4(0)=0.0390$, $x_5(0)=0.0494$, $x6(0)=0.0656$, $\lambda(0)=0.2$

Solución vía simplex**:** $x_1=0.2$, $x_2=0.0$, $x_3=1.6$, $x_4=0$, $x_5=0.0$, $x_6=4.0$ y $f(x)= -5.4$

Solución vía RNR: ***$x_1=0.2$, $x_2=0.0$, $x_3=1.6$, $x_4=0$, $x_5=0.0$, $x_6=4.0$ y $f(x)= -5.4$ $E=-5.2940$***

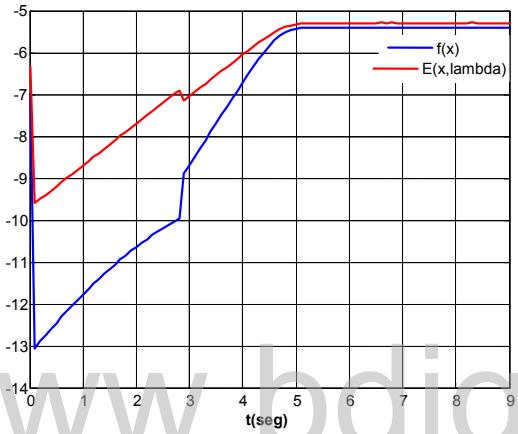


Figura 3.32 Función de energía (en rojo) y Función objetivo (en azul)

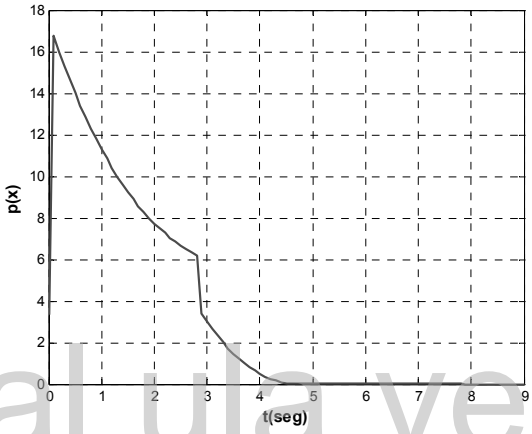


Figura 3.33 Función de penalización

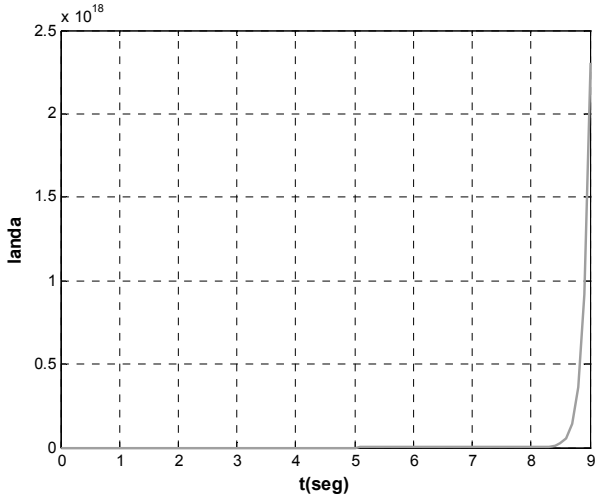


Figura 3.34 Variable de penalización

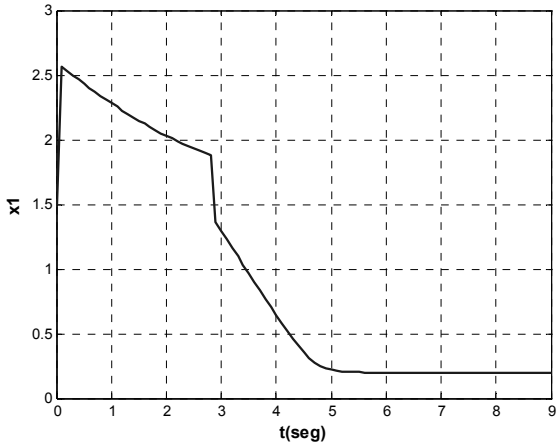


Figura 3.35 Respuesta de variable de decisión x_1

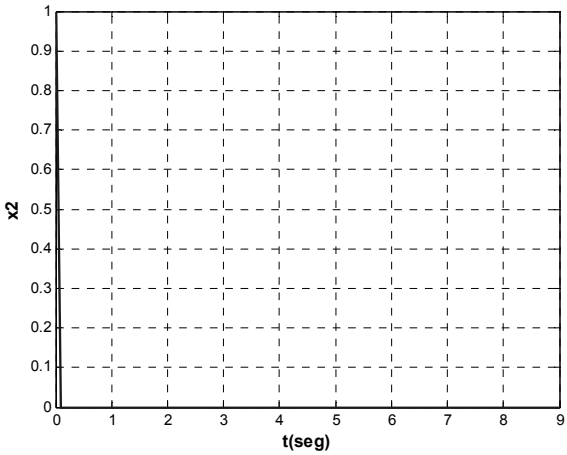


Figura 3.36 Respuesta de variable de decisión x_2

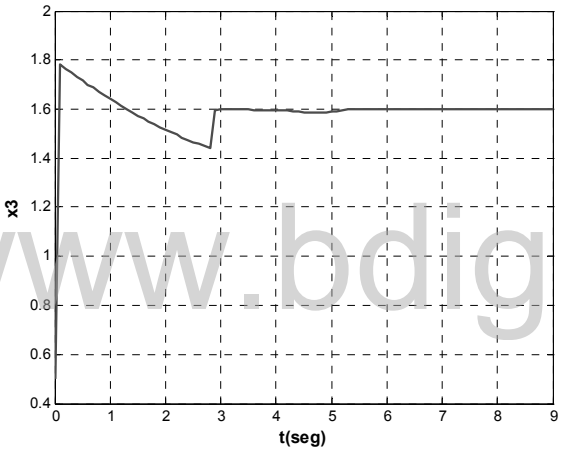


Figura 3.37 Respuesta de variable de decisión x_3

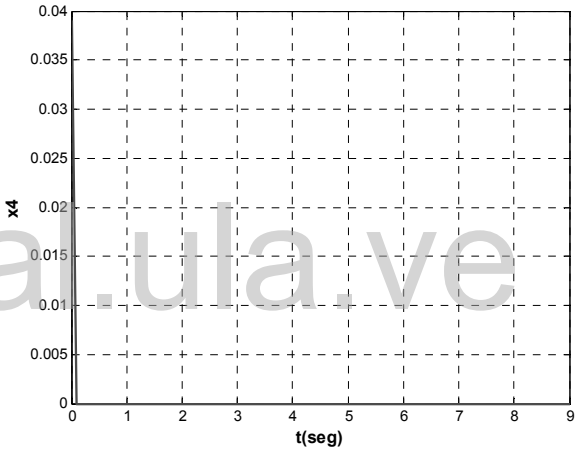


Figura 3.38 Respuesta de variable de holgura x_4

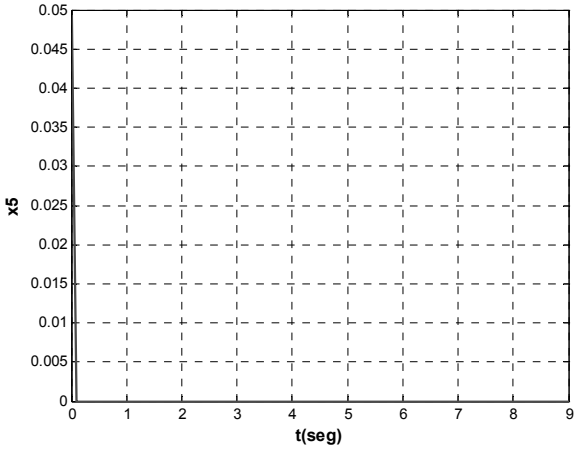


Figura 3.39 Respuesta de variable de holgura x_5

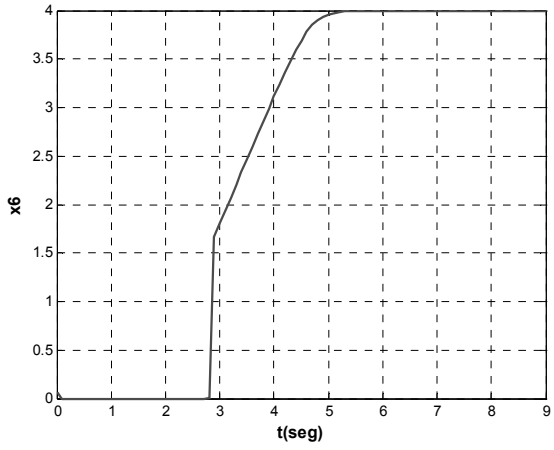


Figura 3.40 Respuesta de variable de holgura x_6

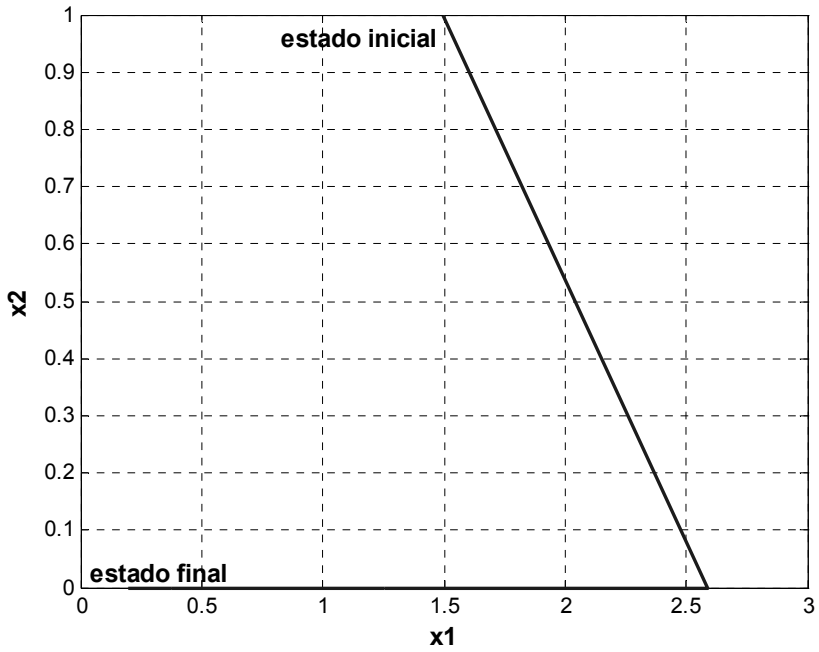


Figura 3.41 Diagrama de estado x_1-x_2

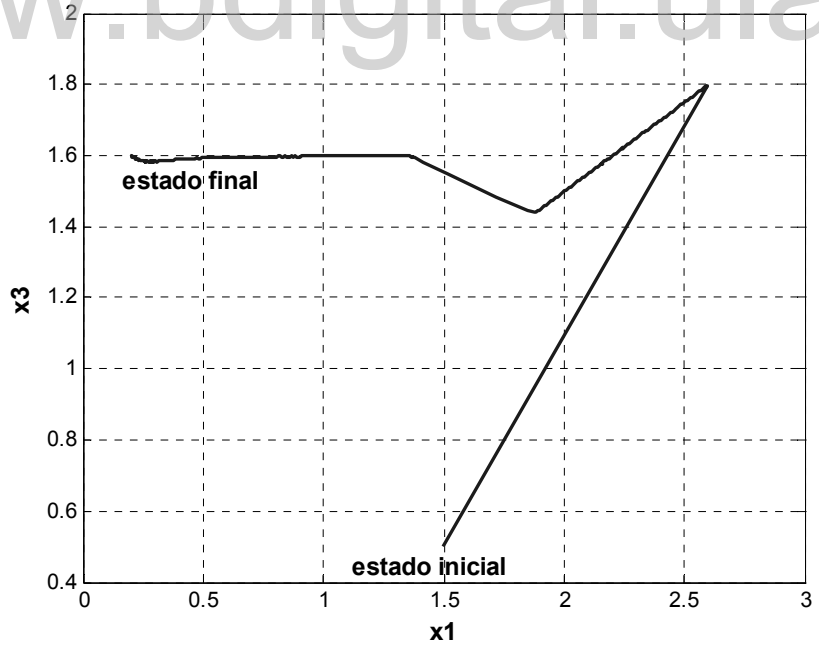


Figura 3.42 Diagrama de estado x_1-x_3

Los resultados de las simulaciones demuestran, de la misma forma que en el problema anterior, que el estado estacionario de la RNR converge hacia la solución óptima para distintas condiciones iniciales de las variables de decisión. En las Fig.3.21 y Fig. 3.32, se observa que la función de evaluación *E,* así como la función objetivo *f*, convergen hacia la solución óptima. De igual manera que en el problema de 2 variables se observa una pequeña discrepancia que se debe al valor de la función de penalización *p(x)* (que es casi igual a cero*)* multiplicado por la variable de penalización λ(t) (que es monótona creciente) en la función de evaluación .,

En las Fig. 3.24 hasta la Fig. 3.29 y en las Fig. 3.35 hasta la Fig. 3.40, se observa que partiendo de distintas condiciones iniciales, el estado estacionario de la RNR es la solución óptima de las variables de decisión y de las variables de holgura del problema. Además en las Fig.3.30, 3.31, 3.41 y 3.42 tenemos los gráficos de trayectorias donde se observa que todas convergen en su estado final a la solución óptima de dichas variables.

La siguiente tabla muestra un resumen de todas las corridas donde se evidencia la convergencia de la RNR hacia la solución óptima para distintas condiciones iniciales. Similar al problema de 2 variables, se observa la importancia que tiene la condición inicial del parámetro de penalización *λ(t)*, ya que para valores < 0.2 la RNR no estabiliza en la solución óptima.

λ(0)	**X_1(0)**	**X_2(0)**	**X_3(0)**	**X_4(0)**	**X_5(0)**	**X_6(0)**	**X_1**	**X_2**	**X_3**	**X_4**	**X_5**	**X_6**	**E(x)**	**f(x)**
0.2	0	0	0.05	0.0463	1e-4	0.0043	0.2	0	1.6	0	0	4	-5.29	-5.4
0.2	1.5	1	0.5	0.0390	0.0494	0.0656	0.2	0	1.6	0	0	4	-5.293	-5.4
0.4	1	1	0.154	5e-5	1.8e-4	3.4e-4	0.2	0	1.6	0	0	4	-5.29	-5.4
0.1	**2.5**	**0.5**	**6e-4**	**3.3e-4**	**5.2e-5**	**0.079**	**1.66**	**0**	**1.3**	**0**	**0**	**0**	**-4.79**	**-9.0**
0.07	**1**	**0.5**	**0.004**	**0.003**	**0.2149**	**0.0087**	**1.93**	**0**	**1.4**	**0**	**0**	**0**	**-7.22**	**-10.2**

Tabla 3.3 Resumen de las 5 corridas realizadas (3 variables)

3.5. RESOLUCIÓN DE PROBLEMAS REALES DE PROGRAMACIÓN LINEAL MEDIANTE RNR.

A continuación se presentan dos casos reales de problemas de P.L. que serán resueltos utilizando RNR. El primer caso se refiere al problema de producción óptima de productos en una fábrica, tomando en cuenta disponibilidad de maquinarias y tiempos de usos de las mismas por producto producido. Este tipo de situaciones se presentan con frecuencia tanto en empresas de producción continua como en las de manufactura.

El segundo problema corresponde a uno de asignación óptima de tareas y ha sido seleccionado para poner en relieve la versatilidad del enfoque basado en RNR, no obstante el carácter de programación entera del problema.

3.5.1 PROBLEMA # 1: Producción Óptima.

Una fábrica requiere determinar la mezcla de producción de sus 7 productos para la próxima semana. Para cada producto, cada unidad producida requiere una cantidad conocida de tiempo de producción en cada una de las 3 máquinas de fabricación. Cada máquina tiene cierto número de unidades de tiempo disponibles por semana. Cada producto proporciona cierta ganancia por unidad producida. La siguiente tabla muestra los datos relacionados con el tiempo de producción de las máquinas, de la igual forma muestra el tiempo del que se dispone en la semana y por último la ganancia por cada unidad producida. El objetivo es determinar la política de producción óptima de tal forma que se maximice la ganancia total sin exceder la capacidad de producción de cada máquina.

	Unidades de tiempo de producción							Tiempo disponible por semana
	Producto							
Máquina	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	
M1	1	1	3	1	1	2	4	15
M2	3	2	1	2	2	2	3	20
M3	2	2	1	3	1	3	1	25
Ganancia	5	6	5	9	7	6	8	

Tabla 3.4 Parámetros del problema

SOLUCIÓN.

La decisión a tomar es cuántas unidades de cada producto fabricar durante la semana, entonces, se define:

x_j = niveles de producción para el producto P_j
c_j = ganancia unitaria por producto P_j
a_{ij} = tiempo de producción en la máquina Mi por unidad de producto P_j
b_i = tiempo de producción disponible por semana en la máquina M_i

Modelo.

$$Maximizar \quad \sum_{j=1}^{7} c_j x_j$$

Sujeto a:

$$\sum_{j=1}^{7} a_{ij} x_j \le b_i \quad para\ i = 1,2,3.$$
$$x_j \ge 0 \ \ para\ j = 1,2,3 \ldots 7$$

Ahora, de acuerdo a la Tabla 3.4 desplegamos el modelo y queda:

$$Maximizar \quad Z = 5x_1 + 6x_2 + 5x_3 + 9x_4 + 7x_5 + 6x_6 + 8x_7$$

Sujeto a:

$$\begin{aligned} &x_1 + x_2 + 3x_3 + x_4 + x_5 + 2x_6 + 4x_7 \le 15 \\ &3x_1 + 2x_2 + x_3 + 2x_4 + 2x_5 + 2x_6 + 3x_7 \le 20 \\ &2x_1 + 2x_2 + x_3 + 3x_4 + x_5 + 3x_6 + x_7 \le 25 \\ &x_1, x_2, x_3, x_4, x_5, x_6, x_7 \ge 0 \end{aligned}$$

Llevamos a la forma estándar *Ax-b= 0, resultando*

$$\begin{aligned} Z &= 5x_1 + 6x_2 + 5x_3 + 9x_4 + 7x_5 + 6x_6 + 8x_7 \\ x_1 + x_2 + 3x_3 + x_4 + x_5 + 2x_6 + 4x_7 + x_8 - 15 &= 0 \\ 3x_1 + 2x_2 + x_3 + 2x_4 + 2x_5 + 2x_6 + 3x_7 + x_9 - 20 &= 0 \\ 2x_1 + 2x_2 + x_3 + 3x_4 + x_5 + 3x_6 + x_7 + x_{10} - 25 &= 0 \end{aligned}$$

De igual forma que en los ejemplos de la sección anterior, haciendo usa de la Ec. (3.13) y Ec. (3.14) construimos la función de penalización y la función de evaluación respectivamente.

$$p(x) = \frac{1}{2}[(\sum_{j=1}^{10} a_{1j}x_j - b_1)^2 + (\sum_{j=1}^{10} a_{2j}x_j - b_2)^2 + (\sum_{j=1}^{10} a_{3j}x_j - b_3)^2$$

Y la función de evaluación es:

$$E(x,\lambda) = \sum_{j=1}^{7} -c_j x_j + \frac{\lambda}{2}[(\sum_{j=1}^{10} a_{1j}x_j - b_1)^2 + (\sum_{j=1}^{10} a_{2j}x_j - b_2)^2 + (\sum_{j=1}^{10} a_{3j}x_j - b_3)^2$$

De acuerdo a la Tabla 3.1 se selecciona como función de activación:

$$x(t) = F(u) = e^{u(t)} \text{ para } x(t)=[0,\infty]$$

Se fijan los parámetros $C_u=0.01$, $C_\lambda = 1$, y de acuerdo con la Ec. (3.21) y Ec. (3.22) tenemos el siguiente sistema no lineal de ecuaciones diferenciales que representan la RNR para resolver este problema.

$$x_j(t) = e^{u_j(t)} \quad para\ j = 1,2,3,4\ldots10$$

$$c_u \dot{u_j}(t) = \frac{-c_j - \lambda(t)(\sum_{p=1}^{10}\sum_{i=1}^{3} a_{ip}a_{ij}x_p - \sum_{i=1}^{3} a_{ij}b_i)}{\dot{\lambda}(t)} \quad para\ j = 1,2,3,4....10$$

$$c_\lambda \dot{\lambda}(t) = \frac{2}{[(\sum_{j=1}^{10} a_{1j}x_j(t) - b_1)^2 + (\sum_{j=1}^{10} a_{2j}x_j(t) - b_2)^2 + (\sum_{j=1}^{10} a_{3j}x_j(t) - b_3)^2]}$$

$$E(x(t),\lambda(t))\sum_{j=1}^{7} -c_j x_j(t) + \frac{\lambda}{2}[(\sum_{j=1}^{10} a_{1j}x_j(t) - b_1)^2 + (\sum_{j=1}^{10} a_{2j}x_j(t) - b_2)^2 + (\sum_{j=1}^{10} a_{3j}x_j(t) - b_3)^2]$$

RESULTADOS.

- Solución vía Simplex: $x_1=0.0,\ x_2=0.0,\ x_3=2.0,\ x_4=7.0,\ x_5=2.0, x_6=0.0, x_7=0.0, x_8=0.0, x_9=0.0, x_{10}=0.0, ,\ f= -87.0$

- Solución vía RNR: $\boldsymbol{x_1=0.0,\ x_2=0.0,\ x_3=2.0,\ x_4=7.0,\ x_5=2.0, x_6=0.0, x_7=0.0, x_8=0.0, x_9=0.0, x_{10}=0.0, ,\ f= -87.0, E=- 87.0}$

Condiciones iniciales:

$x_1(0)=1.0, x_2(0)=1.0, x_3(0)=1.0, x_4(0)=1.0, x_5(0)=1.0, x_6(0)=1.0, x_7(0)=1.0, x_8(0)=0.3793, x_9(0)= 1.6382, x_{10}(0)= 0.8960 , \lambda(0)=0.5$

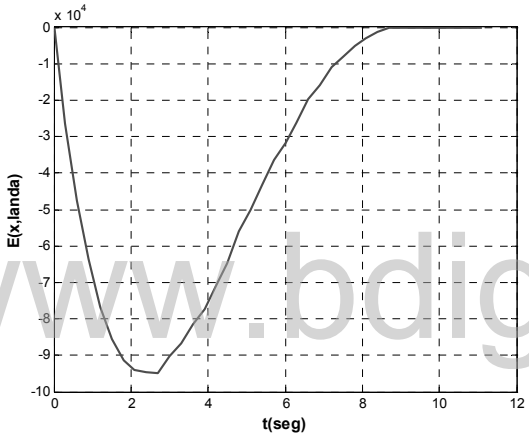


Figura 3.43 Función de Energía

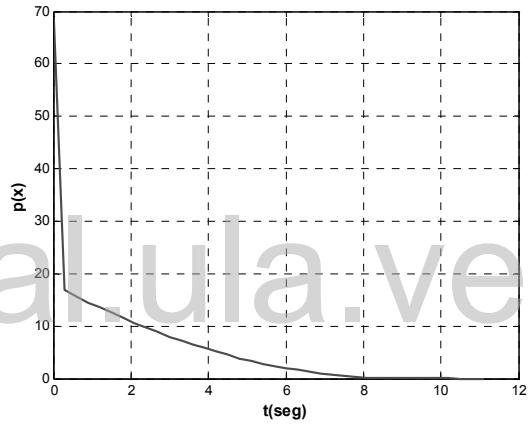


Figura 3.44 Función de penalización

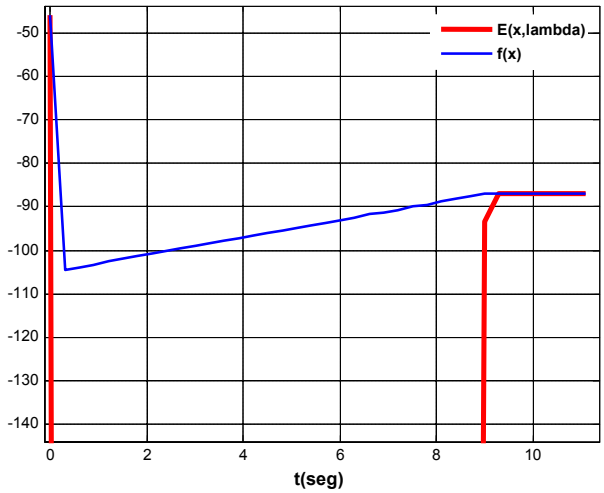


Figura 3.45 Función de energía (en rojo) y Función objetivo (en azul)

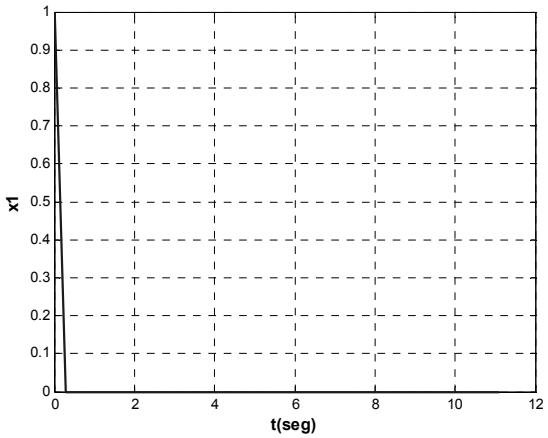


Figura 3.46 Respuesta de variable de decisión x_1

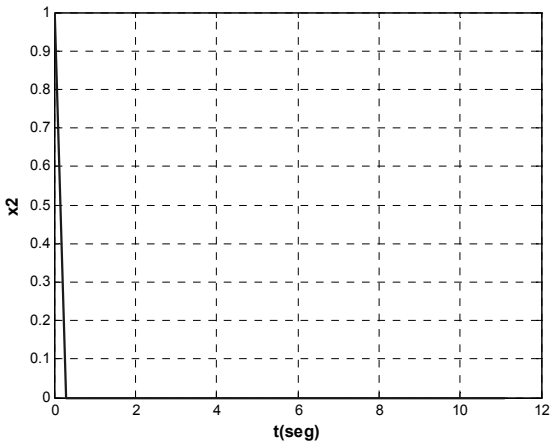


Figura 3.47 Respuesta de variable de decisión x_2

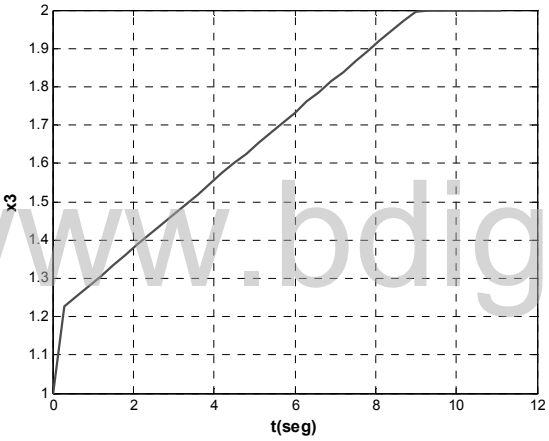


Figura 3.48 Respuesta de variable de decisión x_3

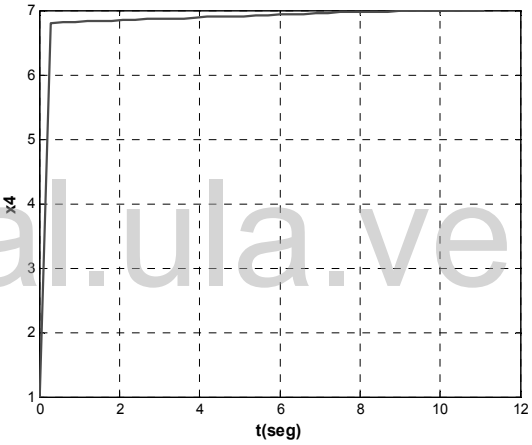


Figura 3.49 Respuesta de variable de decisión x_4

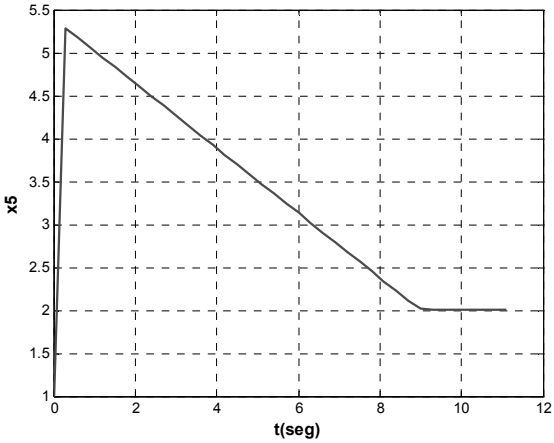


Figura 3.50 Respuesta de variable de decisión x_5

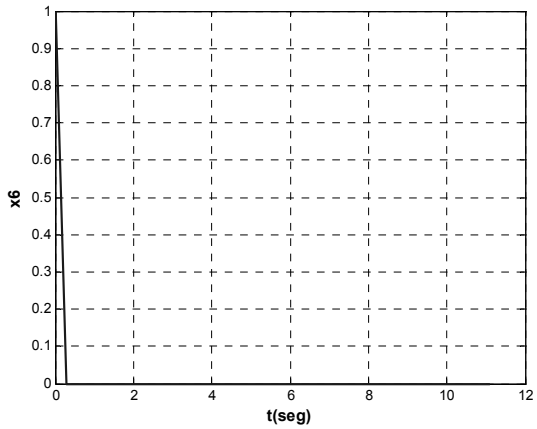


Figura 3.51 Respuesta de variable de decisión x_6

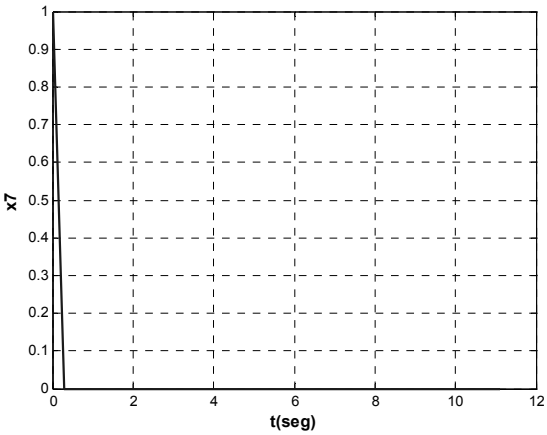


Figura 3.52 Respuesta de variable de decisión x_7

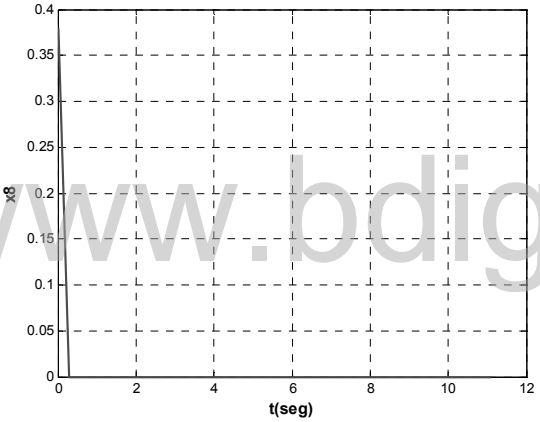


Figura 3.53 Respuesta de variable de holgura x_8

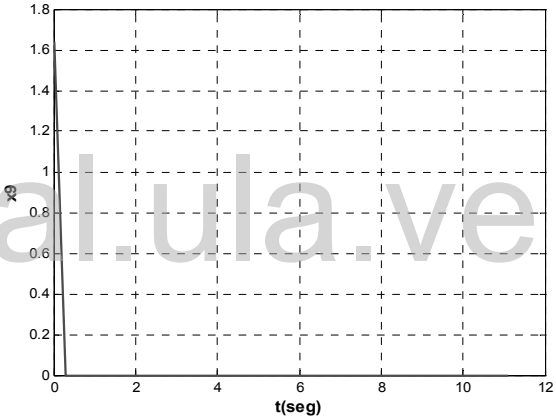


Figura 3.54 Respuesta de variable de holgura x_9

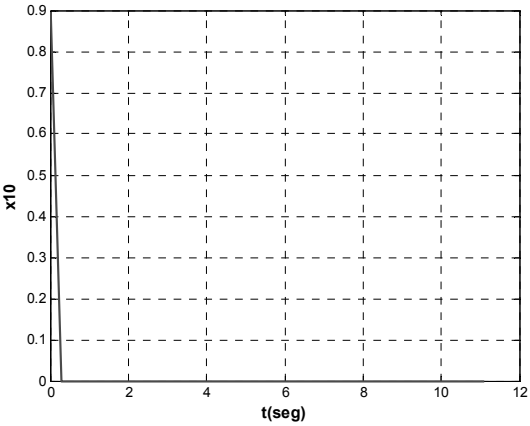


Figura 3.55 Respuesta de variable de holgura x_{10}

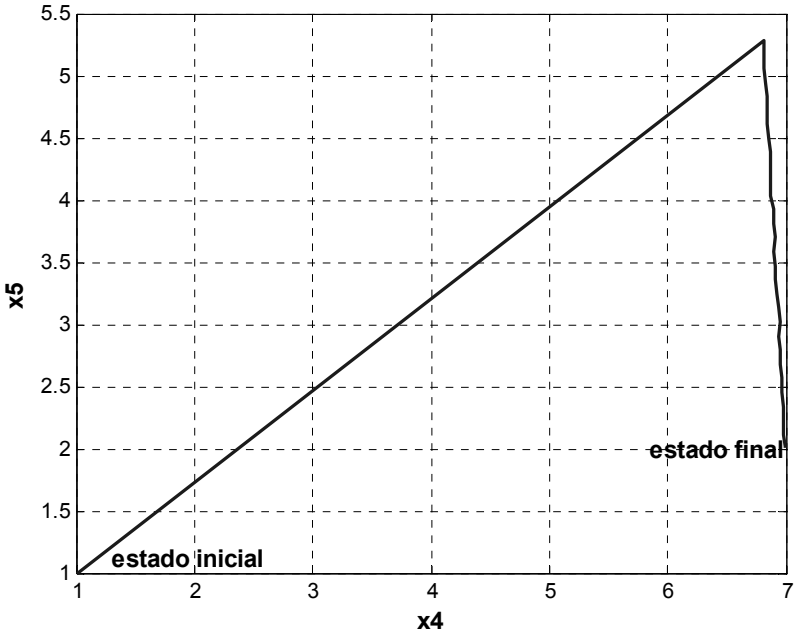


Figura 3.56 Diagrama de estado x_4-x_5

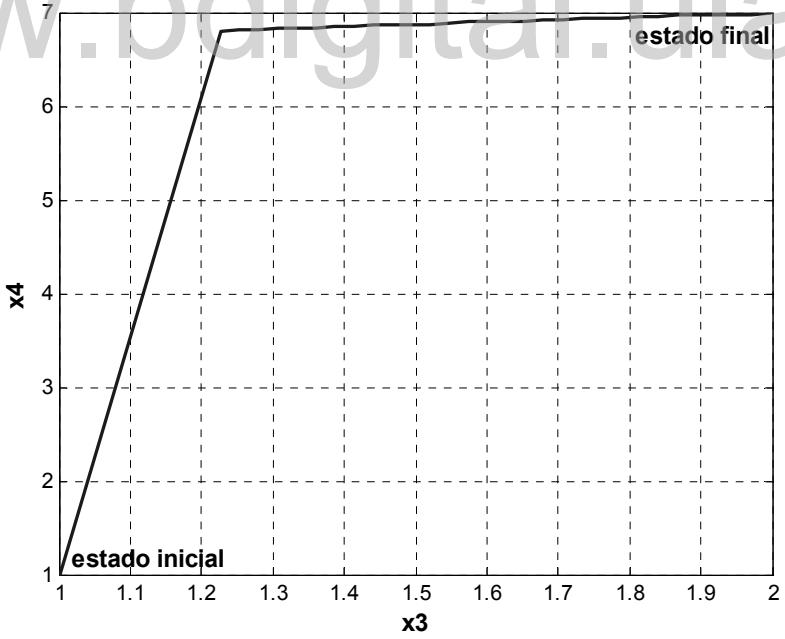


Figura 3.57 Diagrama de estado x_3-x_4

A la luz de los resultados anteriores, se observa en la Fig. 3.43 y Fig. 3.45 la evolución de la función de energía *E*, la cual estabiliza en la solución de la función objetivo del problema resuelto por el método simplex. No se observa ningún tipo de error entre los valores, ya para los 9 seg. aproximadamente los valores de la función de energía *E* y la función objetivo son exactamente iguales.

Luego en las Fig. 3.46 hasta la Fig. 3.52 se aprecia la transición y estabilización de las variables de decisión $x_1, x_2, x_3, x_4, x_5, x_6$ y x_7. Se observa que sus estados estables: $x_1=0, x_2=0, x_3=2, x_4=7, x_5=2, x_6=0$ y $x_7=0$ representan la solución óptima del problema, conseguida por el método simplex.

En las Fig. 3.53, 3.54 y 3.55 se muestran la evolución de las variables de holgura x_8, x_9, y x_{10}, que en sus estados estables son iguales a cero, coincidiendo de igual forma con la solución obtenida por el método simplex.

En las Fig. 3.44, se observa el gráfico para el parámetro de penalización $p(x)$, donde su estado estable es igual a cero, como debía de esperarse.

Por último se tienen en las Fig. 3.56 y Fig. 3.57, los diagramas de estado para algunas parejas de variables de decisión, se aprecian en ambas figuras que las trayectorias convergen hacia el estado final que no es otro que la solución óptima de las variables de decisión del problema.

3.5.2 PROBLEMA # 2: Asignación Óptima de Tareas.

Una compañía de software tiene a cargo un proyecto a desarrollar compuesto de 5 grandes módulos para ser trabajados por 5 programadores diferentes. Se desea que cada módulo sea desarrollado por un solo programador y que cada programador desarrolle un solo módulo del software.

Debido a los diferentes grados de dificultad de los módulos y a las diferencias individuales de las capacidades y aptitudes de los programadores, el tiempo (en días) que ellos emplean es diferente y se da en la siguiente tabla:

	Prog. 1	**Prog. 2**	**Prog. 3**	**Prog. 4**	**Prog. 5**
Módulo 1	2	4	4	3	6
Módulo 2	2	6	5	4	6
Módulo 3	5	6	5	3	7
Módulo 4	3	5	7	2	4
Módulo 5	8	5	6	2	1

Tabla 3.5 Parámetros del problema

La Gerencia de la compañía desea tomar la decisión adecuada referente a cuales módulos asignar a cada uno de los programadores de tal forma que se logre minimizar el tiempo total de desarrollo del software.

SOLUCIÓN.

La decisión a tomar es asignar los módulos a cada uno de los programadores, teniendo en cuenta que cada módulo debe ser desarrollado por un solo programador y que cada programador debe desarrollar un solo módulo del software, entonces se define:

x_{ij}: Asignación del módulo *i* al programador *j*
c_{ij} = tiempo que tarda el programador *j* en desarrollar el módulo *i*

El problema a simple vista se puede tomar como un problema de asignación entera donde las variables toman valores de 0 ó 1 solamente, sin embargo se puede reformular como un problema de programación lineal de la siguiente manera [1,14]:

MODELO.

$$Minimizar \quad \sum_{i=1}^{5}\sum_{j=1}^{5} c_{ij}x_{ij}$$

Sujeto a:

$$\sum_{j=1}^{5} x_{ij} = 1 \quad j = 1,2,3,4,5$$

$$\sum_{i=1}^{5} x_{ij} = 1 \quad i = 1,2,3,4,5$$

$$x_{ij} \geq 0 \qquad i, j = 1,2,3,4,5$$

De acuerdo a la Tabla 3.5 desplegamos el modelo y queda:

$$Minimizar\ Z = 2x_{11}+4x_{12}+4x_{13}+3x_{14}+6x_{15}+2x_{21}+6x_{22}+5x_{23}+4x_{24}+6x_{25}+x_{31}+6x_{32}+5x_{33}+3x_{34}+7x_{35}+3x_{41}+5x_{42}+7x_{43}+2x_{44}+4x_{45}+8x_{51}+5x_{52}+6x_{53}+2x_{54}+x_{55}$$

Sujeto a:

$$x_{11}+x_{12}+x_{13}+x_{14}+x_{15} = 1$$
$$x_{21}+x_{22}+x_{23}+x_{24}+x_{25} = 1$$
$$x_{31}+x_{32}+x_{33}+x_{34}+x_{35} = 1$$
$$x_{41}+x_{42}+x_{43}+x_{44}+x_{45} = 1$$
$$x_{51}+x_{52}+x_{53}+x_{54}+x_{55} = 1$$
$$x_{11}+x_{21}+x_{31}+x_{41}+x_{51} = 1$$
$$x_{12}+x_{22}+x_{32}+x_{42}+x_{52} = 1$$
$$x_{13}+x_{23}+x_{33}+x_{43}+x_{53} = 1$$
$$x_{14}+x_{24}+x_{34}+x_{44}+x_{54} = 1$$
$$x_{15}+x_{25}+x_{35}+x_{45}+x_{55} = 1$$
$$x_{ij} \geq 0$$

El modelo ya se encuentra en la forma estándar $AX = b$, la matriz A en este caso esta compuesta de 0 y 1 solamente y el vector de recursos b está compuesto de 1 solamente. De esta forma pasamos a construir la función de penalización y la función de evaluación, respectivamente, de acuerdo a las Ec. (3.13) y Ec. (3.14).

$$p(x) = \frac{1}{2}[\sum_{i=1}^{5} (\sum_{j=1}^{5} x_{ij} - 1)^2 + [\sum_{j=1}^{5} (\sum_{i=1}^{5} x_{ij} - 1)^2; \quad i, j = 1,2,3,4,5$$

Y la función de evaluación es:

$$E(x, \lambda) = \sum_{i=1}^{5}\sum_{j=1}^{5} - c_{ij} x_{ij} + \frac{\lambda}{2}[\sum_{i=1}^{5} (\sum_{j=1}^{5} x_{ij} - 1)^2 + [\sum_{j=1}^{5} (\sum_{i=1}^{5} x_{ij} - 1)^2]; \quad i, j = 1,2,3,4,5$$

De acuerdo a la Tabla 3.1 se selecciona como función de activación:

$$x(t) = F(u) = 1 / 1 + e^{-\eta_i u_i} \text{ para } x(t)=[0,1]$$

Se fijan los parámetros $C_u=0.01$, $C_\lambda = 1$, y de acuerdo con la Ec. (3.21) y Ec. (3.22) tenemos el siguiente sistema no lineal de ecuaciones diferenciales que representan la RNR para resolver este problema.

$$x_{ij}(t) = 1 / 1 + e^{u_{ij}} \text{ para } i,j = 1,2,3,4,5$$

$$c_u \dot{u}_{ij}(t) = \frac{-c_{ij} - \lambda(t)(\sum_{j=1}^{5} x_{ij}(t) - 1 + \sum_{i=1}^{5} x_{ij}(t) - 1)}{\dot{\lambda}(t)} \quad \text{para } i, j = 1,2,3,4,5$$

$$c_\lambda \dot{\lambda}(t) = \frac{2}{[\sum_{i=1}^{5} (\sum_{j=1}^{5} x_{ij}(t) - 1)^2 + [\sum_{j=1}^{5} (\sum_{i=1}^{5} x_{ij}(t) - 1)^2]}$$

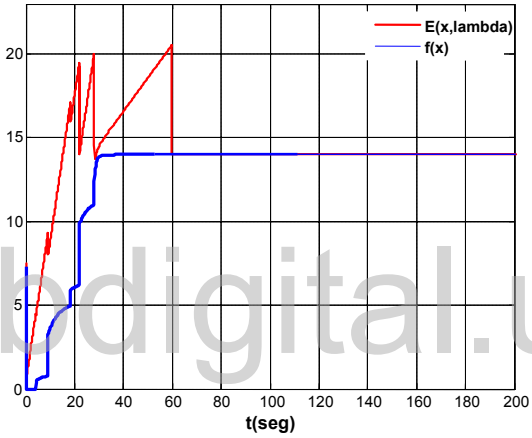
$$E(x(t), \lambda(t)) = \sum_{i=1}^{5}\sum_{j=1}^{5} - c_{ij} x_{ij}(t) + \frac{\lambda(t)}{2}[\sum_{i=1}^{5} (\sum_{j=1}^{5} x_{ij}(t) - 1)^2 + [\sum_{j=1}^{5} (\sum_{i=1}^{5} x_{ij}(t) - 1)^2]$$

RESULTADOS.

Condiciones iniciales.

$x_{11}(0) = 0.9444$; $x_{12}(0) = 0.1120$; $x_{13}(0) = 0.2272$; $x_{14}(0) = 0.0336$, $x_{15}(0) = 0.0007$
$x_{21}(0) = 0.0005$; $x_{22}(0) = 0.0032$; $x_{23}(0) = 0.0179$; $x_{24}(0) = 0.0030$, $x_{25}(0) = 0.0978$
$x_{31}(0) = 0.0001$; $x_{32}(0) = 0.0463$; $x_{33}(0) = 0.0850$; $x_{34}(0) = 0.6285$, $x_{35}(0) = 0.0022$
$x_{41}(0) = 0.0001$; $x_{42}(0) = 0.0019$; $x_{43}(0) = 0.0326$; $x_{44}(0) = 0.0507$, $x_{45}(0) = 0.0001$
$x_{51}(0) = 0.5945$; $x_{52}(0) = 0.0095$; $x_{53}(0) = 0.4075$; $x_{54}(0) = 0.0010$, $x_{55}(0) = 0.0639$

Solución vía simplex: ***f(x)* =14**
Solución vía RNR: ***f(x)* =14; *E(x, λ)* = 14**



Función de energía (en rojo)
y Función objetivo (en azul)

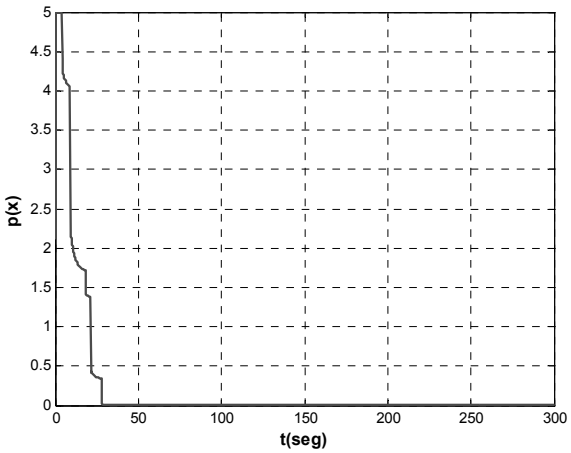


Figura 3.59 Función de penalización

- Solución vía simplex: x_{11}=0; x_{12}=1; x_{13}=0; x_{14}=0, x_{15}=0
- Solución vía RNR: $\boldsymbol{x_{11}}$**=0;** $\boldsymbol{x_{12}}$**=1;** $\boldsymbol{x_{13}}$**=0.000005 ;** $\boldsymbol{x_{14}}$**=0,** $\boldsymbol{x_{15}}$**=0**

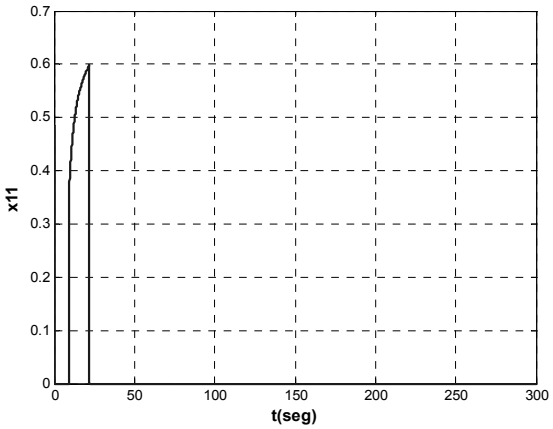


Figura 3.60 Respuesta de variable de decisión x_{11}

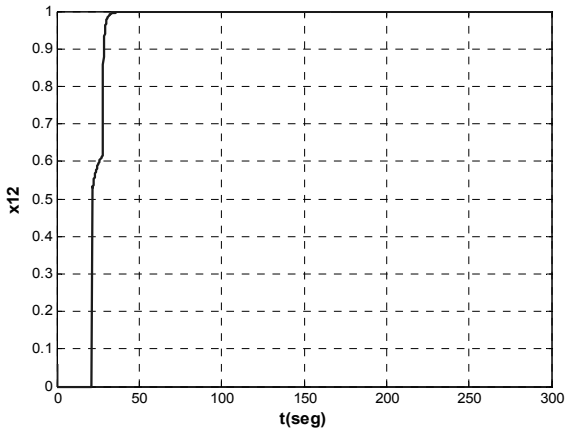


Figura 3.61 Respuesta de variable de decisión x_{12}

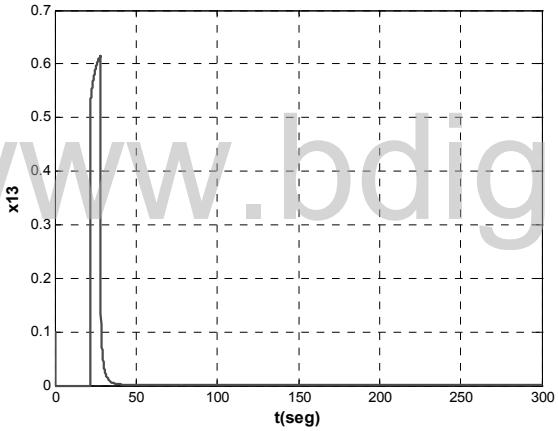


Figura 3.62 Respuesta de variable de decisión x_{13}

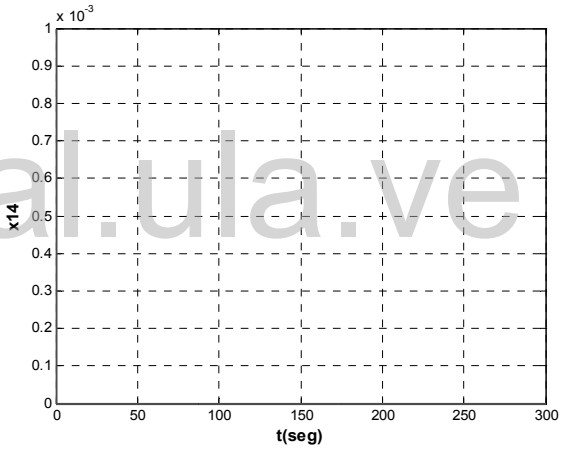


Figura 3.63 Respuesta de variable de decisión x_{14}

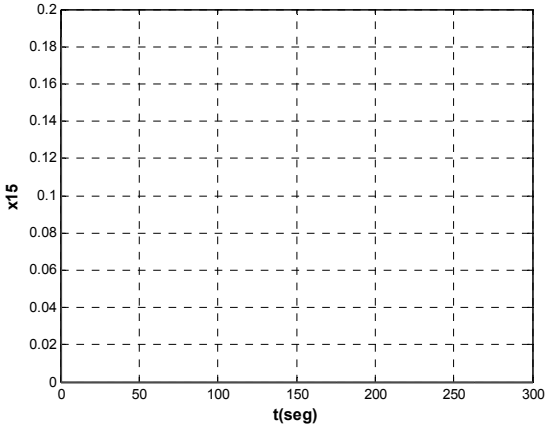


Figura 3.64 Respuesta de variable de decisión x_{15}

- Solución vía simplex: x_{21}=1; x_{22}=0; x_{23}=0; x_{24}=0, x_{25}=0
- Solución vía RNR: $\boldsymbol{x_{21}=1; x_{22}=0; x_{23}=0; x_{24}=0, x_{25}=0}$

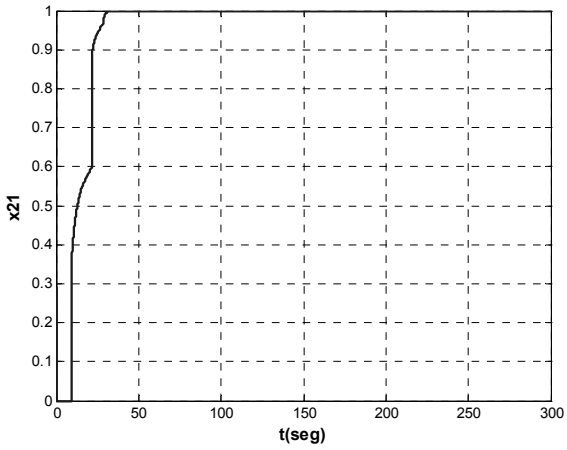


Figura 3.65 Respuesta de variable de decisión x_{21}

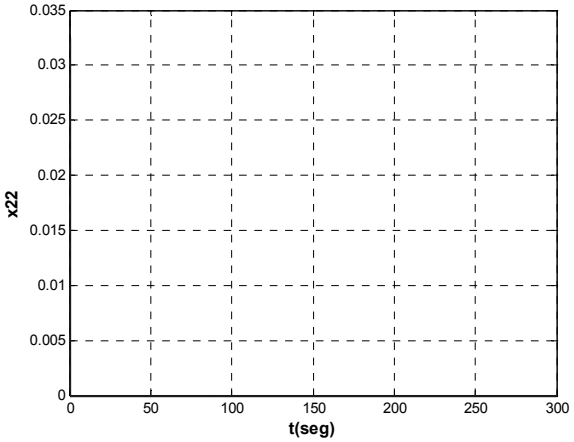


Figura 3.66 Respuesta de variable de decisión x_{22}

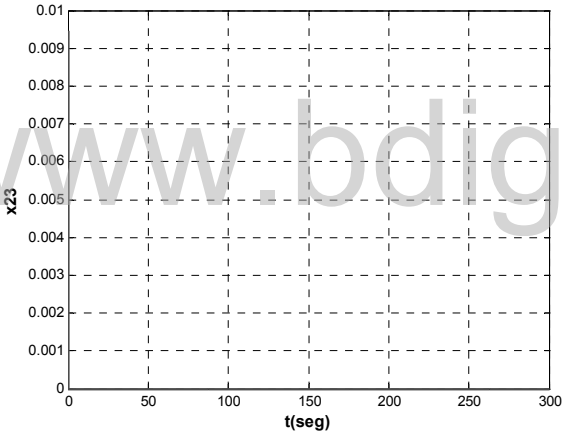


Figura 3.67 Respuesta de variable de decisión x_{23}

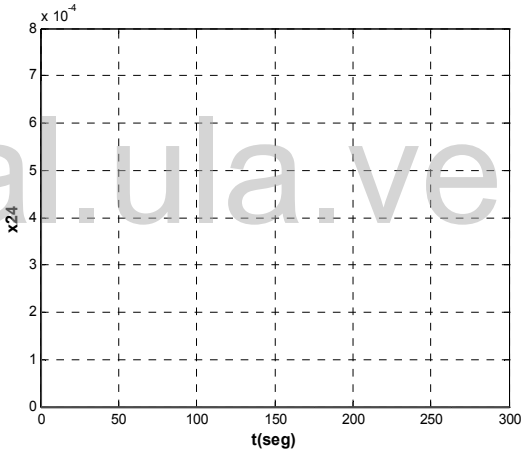


Figura 3.68 Respuesta de variable de decisión x_{24}

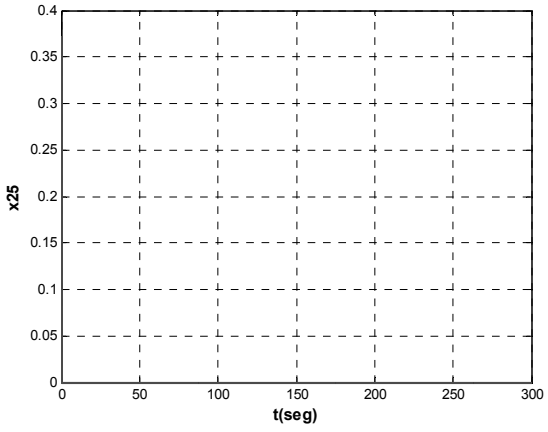


Figura 3.69 Respuesta de variable de decisión x_{25}

- Solución vía simplex: x_{31}=0; x_{32}=0; x_{33}=1; x_{34}=0, x_{35}=0
- Solución vía RNR: ***x*$_{31}$=0; *x*$_{32}$=0; *x*$_{33}$=1; *x*$_{34}$=0.000005, *x*$_{35}$=0**

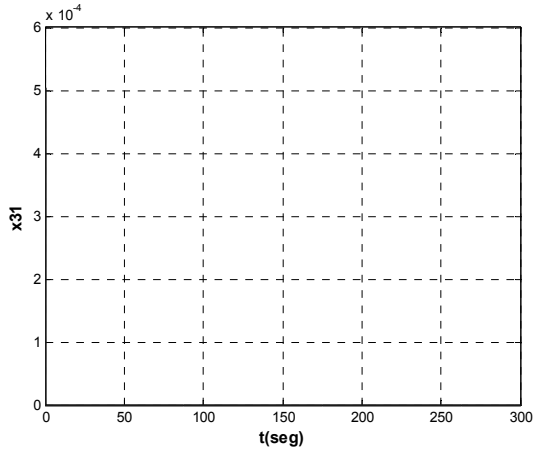


Figura 370 Respuesta de variable de decisión x_{31}

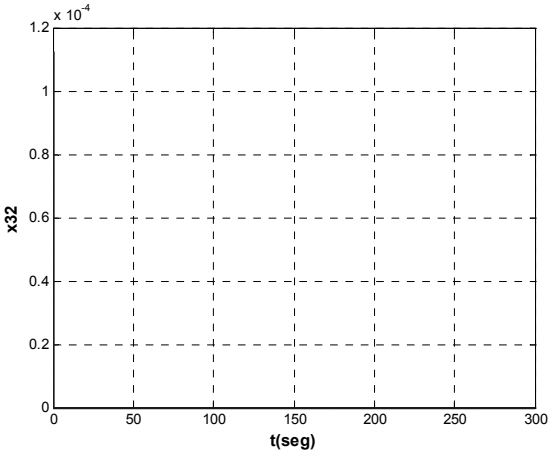


Figura 3.71 Respuesta de variable de decisión x_{32}

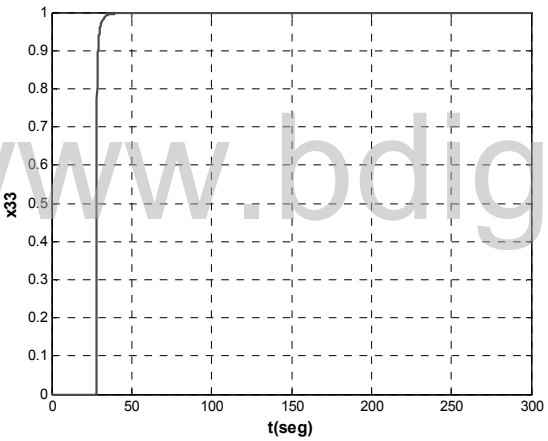


Figura 3.72 Respuesta de variable de decisión x_{33}

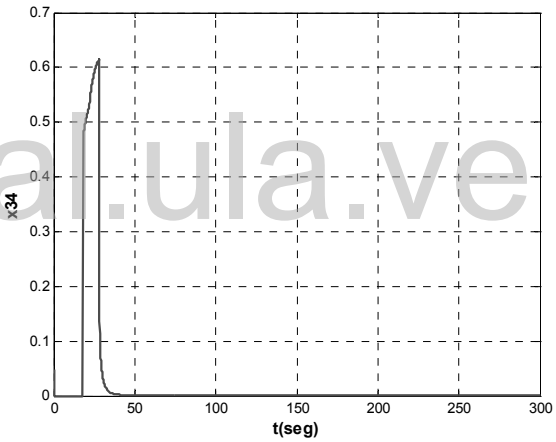


Figura 3.73 Respuesta de variable de decisión x_{34}

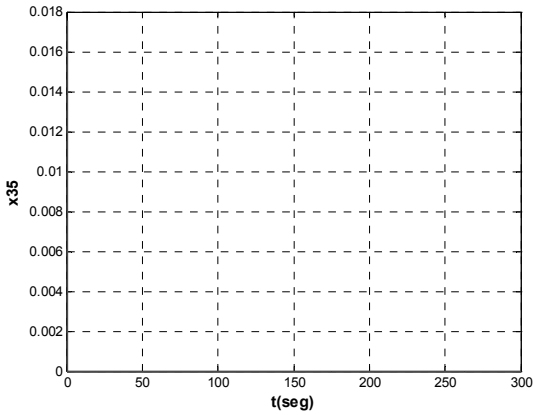


Figura 3.74 Respuesta de variable de decisión x_{35}

- Solución vía simplex: x_{41}=0; x_{42}=0; x_{43}=0; x_{44}=1, x_{45}=0
- Solución vía RNR: $\boldsymbol{x_{41}}$**=0;** $\boldsymbol{x_{42}}$**=0;** $\boldsymbol{x_{43}}$**=0;** $\boldsymbol{x_{44}}$**=1,** $\boldsymbol{x_{45}}$**=0**

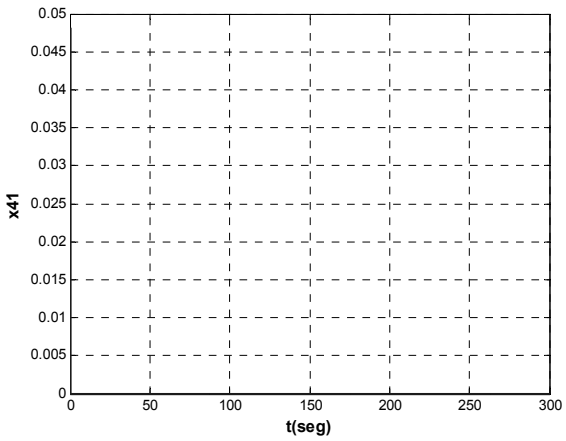


Figura 3.75 Respuesta de variable de decisión x_{41}

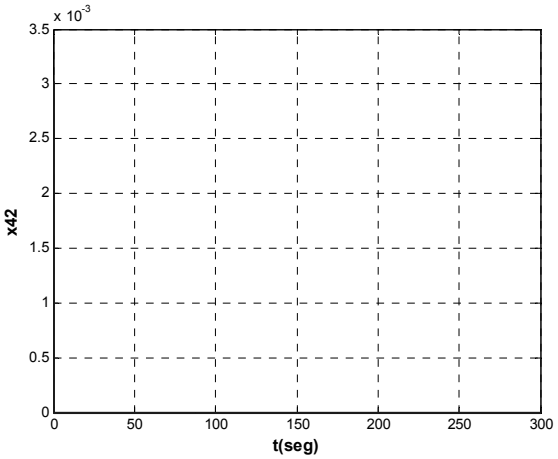


Figura 3.76 Respuesta de variable de decisión x_{42}

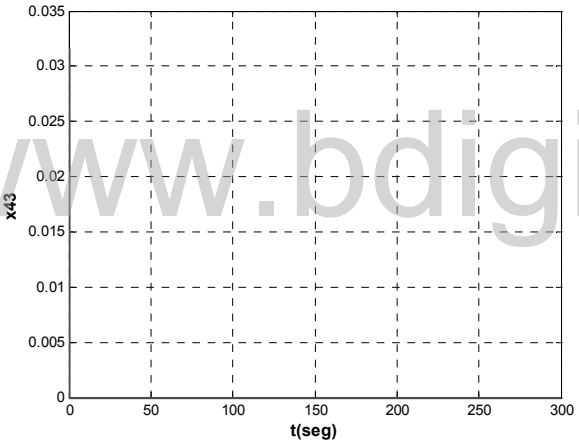


Figura 3.77 Respuesta de variable de decisión x_{43}

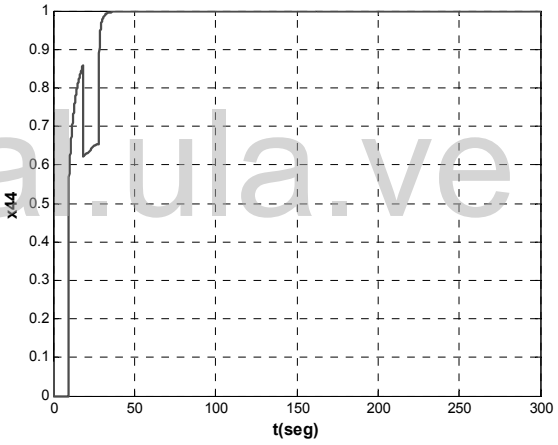


Figura 3.78 Respuesta de variable de decisión x_{44}

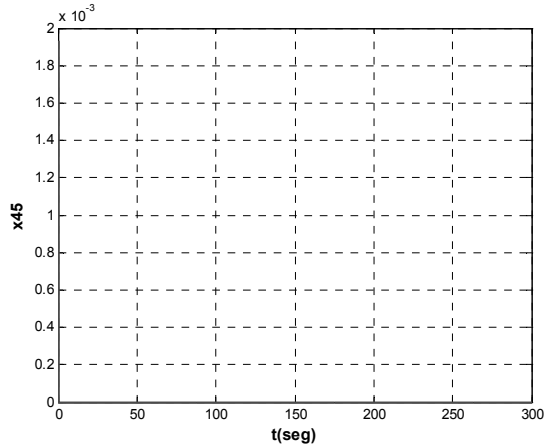


Figura 3.79 Respuesta de variable de decisión x_{45}

- Solución vía simplex: x_{51}=0; x_{52}=0; x_{53}=0; x_{54}=0, x_{55}=1
- Solución vía RNR: $\boldsymbol{x_{51}=0; x_{52}=0; x_{53}=0; x_{54}=0, x_{55}=1}$

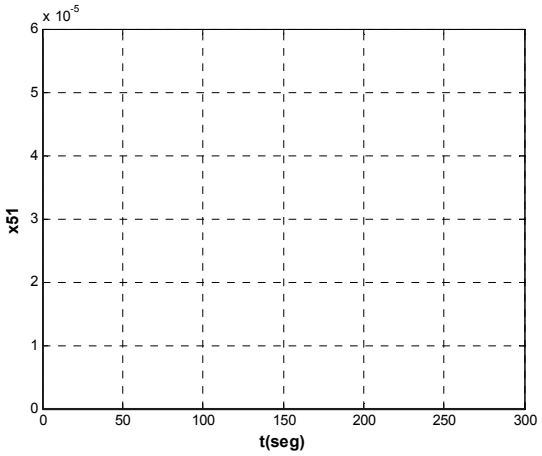


Figura 3.80 Respuesta de variable de decisión x_{51}

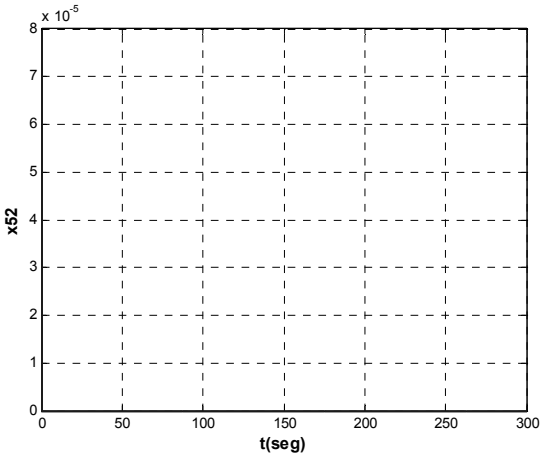


Figura 3.81 Respuesta de variable de decisión x_{52}

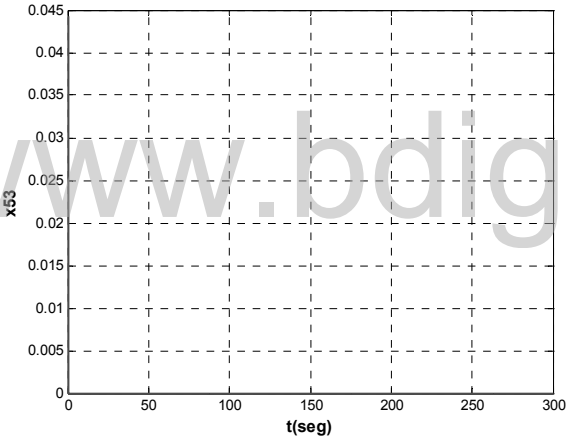


Figura 3.82 Respuesta de variable de decisión x_{53}

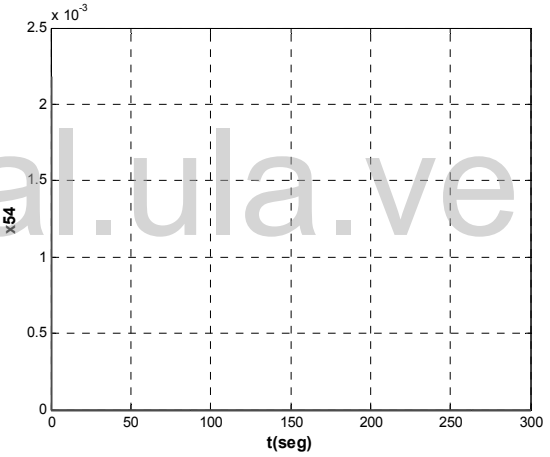


Figura 3.83 Respuesta de variable de decisión x_{54}

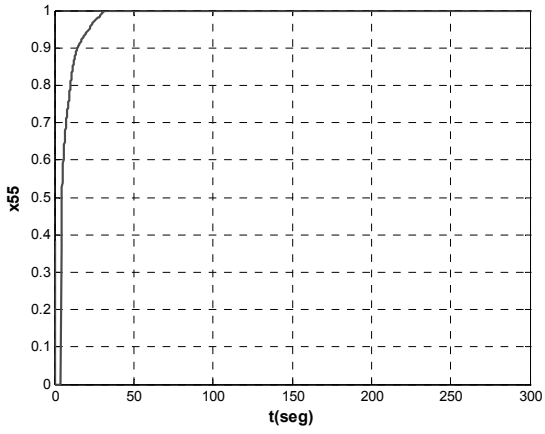


Figura 3.84 Respuesta de variable de decisión x_{55}

Los resultados son evidentes, se observa en la Fig. 3.58 la evolución de la función de energía *E*, la cual converge en su estado estable a la solución de la función objetivo del problema resuelto por el método simplex. No se observan discrepancias ya que el valor de la función de penalización (Fig. 3.59) es igual a cero.

Luego en las Fig. 3.60 hasta la Fig. 3.84 se aprecia la transición de todas las variables de decisión involucradas en el problema. Se observa de manera resaltante la estabilización de las variables x_{12}, x_{21}, x_{33}, x_{44} y x_{55}, cuyo valor es exactamente igual a 1, coincidiendo con la solución obtenida por el método simplex. Para las demás variables que resultaron igual a 0, los resultados coinciden de la misma manera con la solución por método simplex, obteniendo sólo una pequeñísima diferencia con las variables x_{13} y x_{34} que resultaron igual a 0.000005.

Es importante recalcar que a pesar de que los valores de las variables son enteros, la RNR es robusta a este tipo de soluciones, evoluciona de manera eficiente convergiendo en su estado estable casi exactamente (excepto por las variables x_{13} y x_{34}) a las soluciones óptimas del problema de P.L.

CONCLUSIONES Y RECOMENDACIONES

A la vista de los resultados ha sido demostrado lo siguiente:

1. Ninguna de las 4 configuraciones propuestas en el trabajo para el modelo de RPM han obtenido resultados satisfactorios. Si bien es cierto, la red se desempeña bastante bien en la fase de entrenamiento para la configuración # 3 donde se obtuvo un error de 0.0042, en la fase de validación las soluciones que genera para los problemas de P.L. son bastante pobres, el mínimo error que se obtiene en esta fase es de 0.1313, demasiado elevado y por tanto no aceptable sobre todo teniendo en cuenta que en la mayoría de las aplicaciones de P.L. un cambio en las variables de decisión implica grandes consecuencias en la función objetivo y en la solución en general. La razón de este desempeño se debe a que no se logró conseguir para ninguna configuración una generalización por parte de la RPM, es decir, para datos de entrenamiento ella se comporta bastante bien pero cuando se le presentan problemas ciegos (que no se usaron en la fase de entrenamiento) la respuesta no es satisfactoria. Se tiene la certeza de que esto sucede debido a que los parámetros a_{ij}, b_i, c_j son únicos para un solo problema y generan una única región factible de soluciones, al cambiar en pequeña medida estos parámetros estamos en presencia de un nuevo problema y de una nueva región factible de soluciones que no pueden ser aprendidas en su totalidad por una sola RPM. Esto es, existen infinitos problemas con sus correspondientes soluciones que tendrían que ser aprendidos por la RPM seleccionada.

2. Por otra parte, las simulaciones de la RNR han demostrado que este modelo es capaz de generar soluciones óptimas a los problemas de P.L., ya que el estado estable de la red representó la solución de los mismos y se logró minimizar la función de energía. En los 2 ejemplos analizados y en las 2 aplicaciones reales consideradas, no se han detectado diferencias entre los resultados obtenidos por el método simplex y las soluciones generadas por la RNR. A pesar de que en las simulaciones (por medio de software) no se ha logrado apreciar el potencial de las RNR en su procesamiento paralelo y distribuido, si se ha conseguido apreciar su gran potencial para generar de forma rápida resultados alentadores. Sin

embargo, la implementación a nivel de software de las RNR no es de gran utilidad, tomando en cuenta la gran cantidad de herramientas de software comerciales que resuelven problemas de P.L. con una gran cantidad de variables de una manera muy eficiente haciendo uso del método simplex. Ahora bien, para contrarrestar esta desventaja uno de los puntos fuertes de las RNR es que se pueden implementar a nivel de hardware (por medio de dispositivos electrónicos), por lo cual serían capaces de resolver "en línea" problemas de P.L. de gran escala y de forma mucho más eficiente que el método simplex, ésto debido a que el procedimiento de solución de las RNR no está basado en una búsqueda y evaluación secuencial de puntos extremos, sino que se trata de un procedimiento dinámico de búsqueda paralela y distribuida.

Recomendaciones:

1. Dado que los parámetros de los problemas P.L. son estáticos, y ya que las RNR son sistemas dinámicos por naturaleza, se insta a investigar sobre el desempeño de este tipo de redes para tratar de resolver problemas de programación dinámica, donde hay variaciones parámetricas del problema.

2. Debido a que uno de los fuertes de la RNR radica en su procesamiento paralelo, se insta a demostrar su desempeño para resolver problemas de P.L. a través de la implementación física de la red por medio de dispositivos electrónicos, de esta forma se podría lograr la optimización de procesos de gran escala y de forma casi instantánea.

REFERENCIAS BIBLIOGRÁFICAS

1. F. Hillier, G. Lieberman. "Investigación de Operaciones". Ed. McGraw Hill. 2001.
2. N. Loomba. "Linear Programming. An Introductory Analysis." Ed. McGraw Hill. 1983.
3. J. Hilera, V. Martinez. "Redes Neuronales Artificiales". Ed. Alfaomega. 2000.
4. J. Hopfield, D. Tank. "Neural Computations of Decision in Optimization Problems". Biolog Cybern. Vol 52, pp. 141-152. 1985.
5. C. Dagli, S. Kumara, Y. Shin. "Intelligent Engineering Systems Through Artificial Neural Networks". Ed. ASME Press. 1991.
6. C. Lau. "Neural Networks. Theoretical Foundation and Analysis". Ed. IEEE Press. 1991.
7. T. Kohonen, K. Maikisara, O. Simula, J. Kangas. "Artificial Neural Networks". Ed. North-Holland. 1991.
8. S. Haykin. "Neural Networks". Ed. Macmillan College Publishing Company. 1994
9. H. Ritter, T. Martinez, K. Schutten. "Neural Computation". Ed. Addison Wesley. 1992.
10. G.J. Pineda "Generalization of Backpropagation to Recurrent and Higher Order Neural Networks". Neural Information Processing Systems. pp. 602-611. Ed. D.Z Anderson. 1988.
11. D.E Rumelhart, G.E Hinton, R.J Williams. "Learning Representations by Back-Propagating errors". Nature. Vol 323, pp. 533-536. 1986.
12. J. Zurada. Artificial Neural Systems. Ed. West Publishing Company. 1992
13. J. Wang, V. Chankong. "Recurrent Neural Networks for linear programming: Analysis and design principles". Computers and Operations Research, Vol 19, pp. 297-311. 1992.
14. S. Gass. "Programación Lineal, Métodos y Aplicaciones". Ed. Continental. 1980.