Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco
Departamento de Control Automático

CONTROL NEURONAL CON TÉRMINO INTEGRAL PARA SISTEMAS NO LINEALES CON OSCILACIONES

Tesis que presenta

Jacob Raúl Moreno Cruz

Para Obtener el Grado de

Maestro en Ciencias

En la Especialidad de

Control Automático

Director de Tesis:
Dr. Ieroham Solomon Barouh

México, Distrito Federal **Febrero** **2012.**

RESUMEN

El objetivo de este trabajo es comparar el desempeño de los distintos algoritmos para el problema de identificación y control de plantas con modelos no lineales con oscilaciones. La identificación se realiza utilizando una red neuronal recurrente entrenable modular compuesta de una primer capa dinámica y una segunda estática, en donde la matriz de pesos dinámicos posee una forma canónica de Jordan. Algunos bloques de esta matriz tienen dimensión (2x2) que servirán para identificar la parte oscilatoria de la planta.

El algoritmo de entrenamiento para la red neuronal entrenable modular se obtiene utilizando el método grafico de adjunta, el cual consta de realizar una red simétrica en donde la información fluye en sentido opuesto de la red original. Este método presenta grandes ventajas comparado con los métodos tradicionales de cálculo de gradientes.

En el capítulo 6 se realiza la identificación de tres plantas mediante una red neuronal recurrente entrenable utilizando dos algoritmos de entrenamientos diferentes, retro-propagación del error (backpropagation) y el de Levenberg-Marquardt. Las plantas a identificar son el modelo de un vehículo espacial, el modelo de un motor de combustión interna y el modelo del sistema masa resorte amortiguador cuyos resortes presentan comportamiento no lineal. Al final del capítulo se comparan ambos algoritmos utilizando como parámetro al error medio cuadrático de aproximación.

Una vez realizada la identificación se procedió a realizar un control neuronal utilizando el esquema directo e indirecto, con y sin término integral y utilizando dos algoritmos para el entrenamiento de la red neuronal, back propagation y el Levenberg-Marquardt. Se compararon los resultados obtenidos utilizando el error medio cuadrático.

En el control indirecto se tomo como controlador una variante del control por modos deslizantes, donde se acotaba la salida del controlador para obtener una salida de control más suave. Por otro lado, el control directo consta de dos redes neuronales recurrentes entrenables, una para resolver el problema de la identificación y la segunda para generar la señal de control. El término integral fue agregado a ambos controladores, con objetivo de eliminar el error en estado estacionario.

ABSTRACT

The objective of this work is to compare the performance of algorithms for the problem of identifying and controlling nonlinear plants with oscillations. The identification was performed using a modular trainable recurrent neural network which first layer consists of a dynamic and its second layer is static. The dynamic weighting matrix has a Jordan canonical form. Some blocks of this matrix have dimensions (4x4) and are used to identify the oscillatory part of the plant.

The training algorithm for modular trainable neural network is obtained using a diagrammatic method, which consists of making a symmetrical network of the original, where information flows in the opposite direction of the original network. This method has great advantages compared with traditional methods of calculating gradients.

In Chapter 6 is performed the identification of each plant with a trainable recurrent neural network and using two different training algorithms, back propagation and Levenberg Marquardt. The plants to be identified are the model of a spacecraft, a model of an internal combustion engine and model of the spring mass damper system whose springs have nonlinear behavior. At the end of the chapter there is a comparison between the two algorithms using as measure parameter the mean square error of each approximation.

Once the identification was performed, we developed different control schemes. We used two neural control schemes; direct and indirect, and for each schemes we analyzed the advantage of having an integral term. We also compared the performance of two algorithms for training the neural network; back propagation and Levenberg Marquardt. We compared the results obtained using the mean square error as our measure parameter.

The indirect control scheme that we propose is one variant of sliding mode control with saturation of the controller variable so to obtain a smoother control output. On the other hand, direct control consists of two recurrent trainable neural networks, one for solving the problem of identification and the second to generate the control signal. The integral term was added to both controllers, in order to eliminate steady state errors.

AGRADECIMIENTOS

A mi esposa Dulce por compartir esta etapa de nuestras vidas, por su cariño, dedicación y paciencia

A mis Padres a quienes todo les debo, quienes fomentaron en sus hijos el gusto por el estudio y la lectura, apoyándonos y exigiéndonos cuando así se requería.

A mis amigos y compañeros de la maestría, con quienes ha sido un placer compartir estos dos años.

A mis compañeros de FoM, Artemio y Omar, por sus enseñanzas, paciencia y amistad.

A mi asesor, Dr. Ieroham S. Barouh por su constante apoyo y paciencia.

Al jurado examinatorio por sus comentarios al revisar este trabajo.

Al CINVESTAV por abrirme sus puertas y brindarme todas las herramientas necesarias para la realización de esta tesis.

Al CONACYT por el apoyo económico que me brindo para la realización de esta tesis.

Índice General

Índice de figuras

Capítulo 1

Introducción.

El control automático ha venido desempeñando un papel cada vez más importante en la ingeniería, aportando algoritmos de control que permiten tener tecnologías más eficientes, limpias y confiables que en el pasado. Actualmente existen muchas técnicas de control, una que ha venido ganando interés dentro de la comunidad de control son las Redes Neuronales Artificiales que han demostrado proveer eficientes y a la vez elegantes soluciones a problemas de ingeniería, tales como el reconocimiento de patrones, bioingeniería, desarrollo de pruebas no destructivas, diagnósticos médicos, modelado y control de procesos industriales, robótica, etc. [1].

Las redes neuronales son capaces de aproximar funciones no lineales lo que las hace una gran herramienta para la identificación de sistemas con dinámicas no lineales, siempre y cuando, suministremos a red del suficiente número de pares entrada-salida para su entrenamiento. Una vez que tenemos identificada a la planta podemos implementar un control adaptable ya sea directo o indirecto.

En este trabajo de tesis se propone un tipo de red neuronal recurrente llamada red neuronal recurrente entrenable, la cual será utilizada para identificar y controlar plantas mecánicas no lineales con oscilaciones.

Las redes neuronales recurrentes son un modelo computacional más potente que las clásicas redes neuronales hacia adelante. Esta mayor potencia proviene de que las redes recurrentes son capaces de procesar secuencias temporales gracias a la posibilidad de recordar la historia relevante de la secuencia por medio de una representación en forma de estado. Esta memoria no existe de manera natural en las redes no recurrentes, utilizadas principalmente para el procesamiento de datos estáticos, sin embargo como se ha mostrado en la literatura [16], agregando retrasos de las señales de entrada y salida podemos corregir este problema.

El rasgo que diferencia las redes neuronales recurrentes de las que no lo son es la existencia de una conexión recurrente entre las neuronas que las configuran. Esta diferencia tiene profundas implicaciones en la capacidad de computación del modelo y en los algoritmos de entrenamiento necesarios para conseguirla.

Las oscilaciones son fenómenos extremadamente importantes en diversas actividades humanas, como en la ciencia, la tecnología y la industria. Cualquier

movimiento esta estrechamente conectado con una de las más importantes propiedades de la naturaleza, su habilidad para reaccionar con oscilaciones a cualquier influencia interna o externa, [21]. A veces estas oscilaciones son dañinas o incluso pueden llegar a ser peligrosas, en otras ocasiones las percibimos solo como ruido ó vibración.

Si las oscilaciones son suficientemente pequeñas y la dinámica del sistema es suave, podemos linealizar en alguna vecindad y analizar a la planta. Sin embargo en muchas aplicaciones las oscilaciones ni son pequeñas ni la dinámica del sistema es suave, en tales casos es preferible el uso de herramientas como las RN para tratar con el sistema.

Las RN han sido utilizadas en diversos trabajos para lidiar con problemas de plantas mecánicas con vibraciones, por ejemplo en [22] se utiliza una red multicapa para estimar el par aplicado a la planta y se compara con un método basado en un observador, dando mejores resultados la RN por sus propiedades de adaptabilidad; en [23] se propone una RN recurrente para suprimir las vibraciones mecánicas en un motor lineal síncrono de imán permanente, aquí la red neuronal trabaja en conjunto con un controlador PI para incrementar el amortiguamiento del sistema; en [24] se utilizan dos redes "*feedforward*", una para identificación y otra para control de sonido y vibración; en [25] se utiliza un RN para modelar la inversa de la planta para después aplicarlo como un control *feedforward* y así compensar la dinámica de la planta.

1.1 Planteamiento del problema

En muchos problemas de control suelen presentarse plantas no lineales con comportamiento oscilatorio, en los cuales se desea suprimir la dinámica oscilatoria. Este tipo de problemas suelen ser difíciles de tratar con métodos de control convencionales. Una opción para solucionar este problema es la implementación de redes neuronales artificiales para la identificación y control de la planta.

El objetivo de este trabajo es el desarrollo e implementación de sistemas neuronales modulares para identificación y control adaptable (directo e indirecto) con término integral de plantas no lineales mecánicas con oscilaciones. Para la identificación utilizaremos una RNRE con una estructura diagonal por bloques modular en su matriz de pesos de la capa oculta. Esta matriz a bloques es quien identifica la parte oscilatoria de la planta. En el control se utilizaran dos esquemas de control adaptable, el directo e indirecto; en el primero se utilizaran dos redes neuronales modulares, una de identificación y la otra de control, esta ultima toma

como entrada los estados de la red de identificación, además del error, y las señales de salida de la planta. El control neuronal indirecto consta de una red de identificación que provee el modelo de la planta a un controlador por modos deslizantes.

1.2 Estructura del trabajo

En el capítulo 2 se presentan las plantas con las que se trabajará a lo largo de la tesis, se presentarán sus modelos matemáticos y sus respuestas a diversas entradas para observar el tipo de comportamiento oscilatorio que poseen.

El capitulo 3 aborda los fundamentos de las redes neuronales artificiales, el modelo matemático de una neurona artificial, topología de las RN, el aprendizaje. Además de la teoría de las redes neuronales recurrentes, red de Hopfield, la RNRE y otras dos arquitecturas de redes recurrentes encontradas en la literatura. La RNRE modular es la utilizada en este trabajo de tesis. Además se presenta el análisis de Observabilidad y Controlabilidad para la RNRE.

En el capítulo 4 se abordan los distintos algoritmos de aprendizaje como el backpropagation, Levemberg-Marquardt aplicados en la RNRE. Se presenta el método diagramático como una herramienta para obtener las leyes de adaptación de los pesos sinápticos de una manera más simple si se comparada con el método tradicional de derivar iterativamente hasta obtener las ecuaciones de aprendizaje, este método grafico se utiliza para deducir el algoritmo backpropagation y Levenberg-Marquardt recursivo.

El capitulo 5 trata sobre el problema de identificación de sistemas no lineales utilizando las Redes Neuronales recurrentes Modulares. Se presentan los resultados de simulación usando la RNRE modulares con un algoritmo de aprendizaje L-M recursivo.

El capitulo 6 presenta una introducción al control adaptable y sus diversos tipos de arreglos. También se presentan los resultados de simulación del control neuronal adaptable, usando esquemas de control adaptable directo e indirecto.

Finalmente en el capítulo 7 se presentas las conclusiones generales y el trabajo futuro.

Capitulo 2

Modelos Matemáticos

2.1 Introducción

En este capítulo se presentan las plantas con las que se trabajaron en la tesis. La primera es un modelo matemático simplificado de un vehículo espacial, el objetivo de control es controlar el ángulo de dirección del vehículo. La segunda es un motor de combustión interna operando en "*idle speed*". La tercera es un sistema de dos masas con amortiguadores lineales y resortes no lineales.

En numerosas aplicaciones ingenieriles es común que los sistemas estén afectados por ruido o vibraciones indeseables, estás en muchas ocasiones genera efectos indeseables como fracturas por sobrecarga o fallas por fatiga en las estructuras. Cuando la frecuencia de las perturbaciones coinciden con alguna frecuencia natural del sistema se alcanza el fenómeno de resonancia que puede llegar a colapsar el sistema por completo [29], [30], [31] y [32].

El problema de controlar vibraciones es de enorme interés práctico, ya que por medio de los métodos de control se pueden cancelar o al menos atenuar las vibraciones indeseables, [34]. Existen 3 métodos para el control de vibraciones: control pasivo, control semi-activo y control activo. El control pasivo de vibraciones se basa en la incorporación de resortes y amortiguamiento al sistema, con el objeto de reducir la amplitud de la respuesta del sistema primario para una frecuencia de excitación específica; este enfoque solo será útil para la frecuencia de diseño y bajo condiciones de operaciones estables, por lo cual no es recomendable para casos de frecuencias de excitación variables o parámetros desconocidos en el sistema.

El control semi-activo de vibraciones se basa en la aplicación de resortes o amortiguadores que pueden adaptarse o ajustarse, manual o semi-automáticamente, para las condiciones de operación del sistema. El control activo de vibraciones incorpora grados de libertad actuados al sistema siendo generalmente la acción de control una fuerza suministrada por los actuadores y controlada a partir de retroalimentación o prealimentación de la información de los estados del sistema obtenida mediante sensores. En general, la incorporación de grados de libertad al sistema ocasiona dinámicas complejas y fuertes acoplamientos que pueden ser lineales o no lineales. El control activo en los

últimos años ha adquirido un importante impulso, por ejemplo, en la industria automotriz mediante la aplicación de la suspensión inteligente o suspensión activa [33].

2.2 Vehículo espacial.

Esta planta fue tomada de [3]. Supóngase que se desea controlar la posición angular Θ de un vehículo espacial. Para controlar este aparato se dispone de una tobera que puede girar alrededor de su base sobre un pivote especial. El ángulo de orientación de la tobera respecto al eje principal del cuerpo de la astronave es β. La tasa de variación del ángulo de la tobera es directamente proporcional a *u*. L es la distancia desde el punto de apoyo de la tobera en el cuerpo del artefacto hasta el centro de gravedad de la nave (cg). Se supone que la fuerza f de reacción, debida a la expulsión de los gases de la combustión del motor del artefacto, esta aplicada sobre el punto de apoyo de la tobera. Como consecuencia de la fuerza F el artefacto gira alrededor de su centro de gravedad en uno u otro sentido. El problema de control consiste en variar el ángulo Θ en función de una señal de referencia dada, usando como control la velocidad de variación *u* del ángulo β de la tobera.

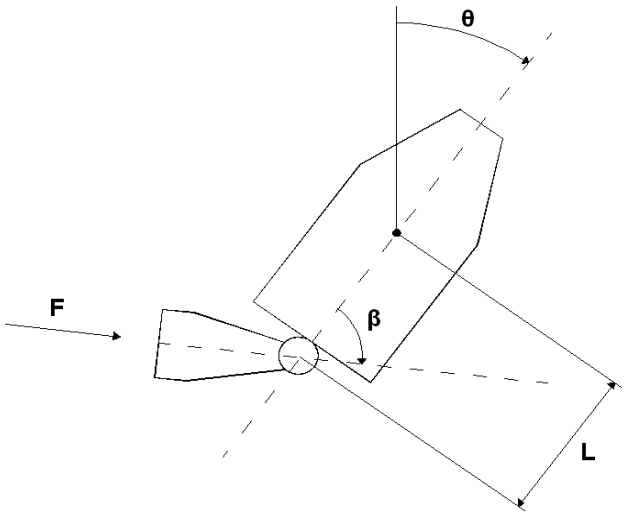


Figura 2.1 Representación grafica de la planta donde se muestran; θ ángulo de orientación del vehículo, F es la fuerza, β es el ángulo de orientación de la tobera y L es la distancia del pivote al centro de masa.

2.2.1 Modelado Matemático de la planta.

Las ecuaciones que rigen la dinámica del sistema son:

$$J\frac{d^2\theta}{dt^2} = FL\sin(\beta) \tag{2.1}$$

$$\frac{d\beta}{dt} = Ru \tag{2.2}$$

Donde *R* es una ganancia estática conocida del actuador que convierte el comando u en velocidad de variación del ángulo β. Supondremos que existe cierta limitación en los valores de u los cuales estarán en el intervalo [-1, 1].
Las variables de estado son:

$$\begin{aligned} x_1 &= \theta \\ x_2 &= \frac{d\theta}{dt} = w \\ x_3 &= \beta \end{aligned} \tag{2.3}$$

Por lo que el sistema representado en variables de estado queda como sigue.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{FL}{J}\sin(x_3) \\ \dot{x}_3 &= Ru \end{aligned} \tag{2.3a}$$

Parámetro	Valor	Parámetro	Valor
F	200 N	R	20
L	3 m	J	50

Tabla 2.1 Valores de los parámetros de la planta

2.2.2 Simulación de la planta.

La simulación se realizó en Matlab-Simulink y en código m de Matlab. Los valores para las constantes están resumidos en la tabla 2.1. El comportamiento (fig. 2.2) de esta planta es que para una entrada acotada (escalón de magnitud 0.5 y que comienza en el primer segundo de la simulación) la salida no lo es, por lo que no tenemos estabilidad BIBO (*Bounded Input-Bounded Output*).

De la simulación vemos que la planta presenta un comportamiento oscilatorio aunque el periodo de oscilación depende inversamente de la magnitud del escalón unitario de entrada, además observamos que nuestra planta presenta oscilación tanto en la posición como en la velocidad angular, mientras que el estado x_3 es creciente debido al tipo de entrada que se utilizó, un escalón.

2.3 Motor de Combustión Interna.

La principal función de un motor es asegurar la movilidad de un vehículo al proveer a la transmisión la potencia suficiente, sin embargo también debe proveer

suficiente par para diversos componentes tales como el aire acondicionado o la dirección.

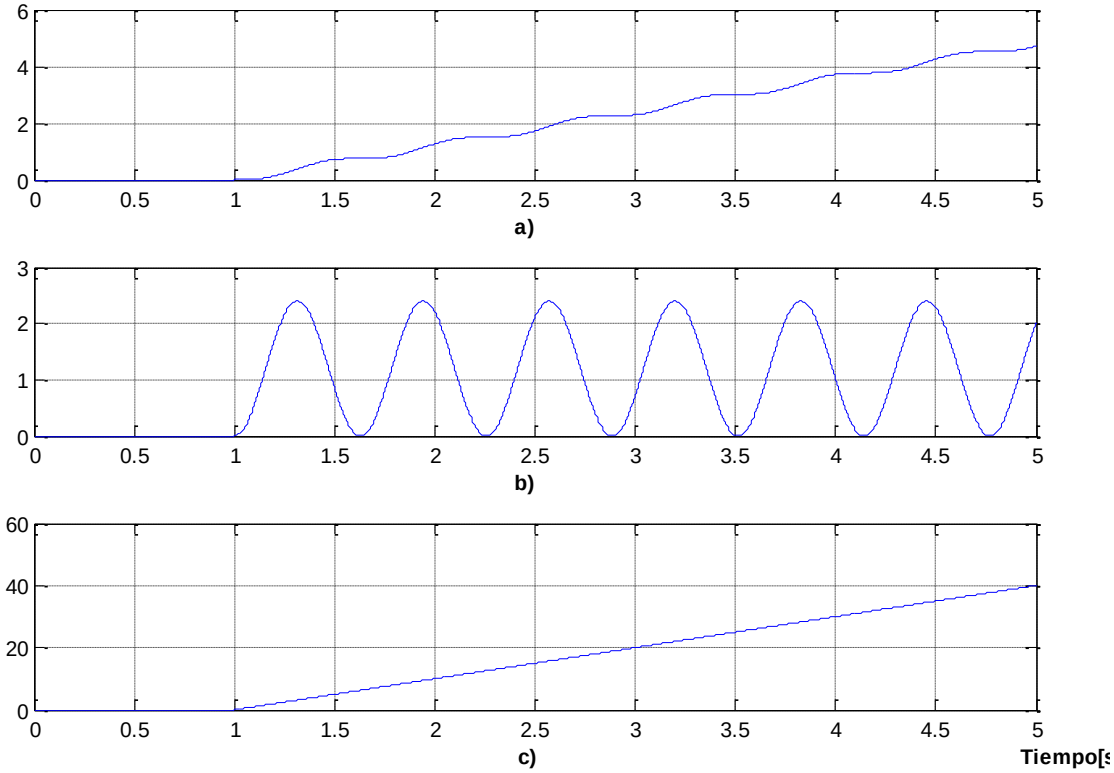


Figura 2.2. Respuesta de la planta a una entrada escalón de magnitud 0.5: a) Posición angular(x_1) b) velocidad angular (x_2) c) ángulo de la tobera (x_3).

Los motores de los automóviles modernos están controlados por una unidad electrónica de control (ECU, *Electronic Control Unit*), la cual ejecuta algoritmos de control muy complejos. La simulación del motor en una computadora *(Hardware in the loop,HIL*), es una herramienta muy importante durante la fase de desarrollo de un motor. El modelo tiene que ser preciso y reflejar el comportamiento dinámico del motor en todo su espectro de operación, [11].

El desempeño de un motor usualmente se refiere al par y potencia de salida, siendo ambos afectados por los parámetros de control dentro del ECU. Actualmente, los datos del desempeño de un motor son obtenidos a través de diversas pruebas en un dinamómetro, [6]. Un ejemplo de esto está dado en la Figura 2.3.

La práctica común para ajustar los parámetros del controlador es un proceso llamado calibración, que depende mucho de la experiencia del ingeniero, quien tiene que manejar una gran cantidad de combinaciones de los parámetros. El ingeniero primero determina una serie de parámetros basados en su experiencia, información de modelos pasados y algunas ecuaciones matemáticas para

después probar al motor en un dinamómetro para obtener su desempeño. El ingeniero repite este proceso ajustando los valores del controlador hasta obtener un desempeño satisfactorio..

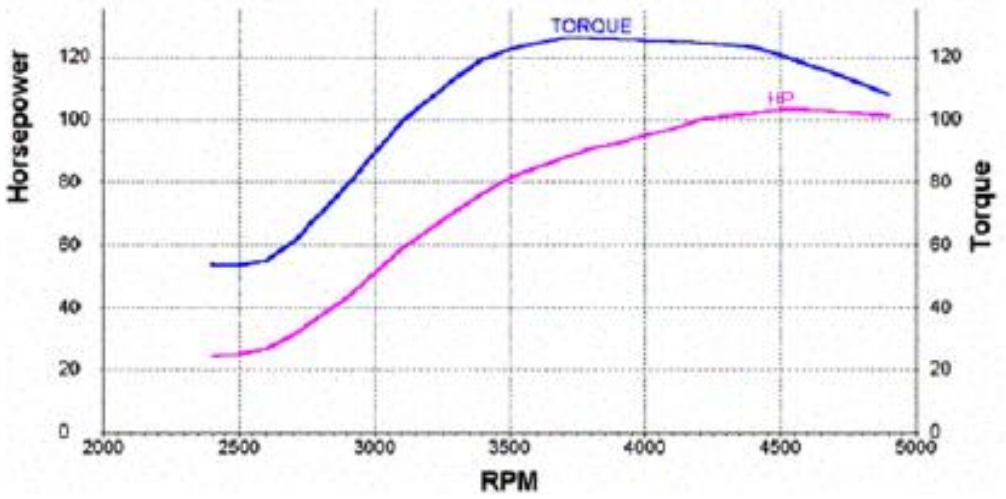


Figura 2.3. Ejemplo de la salida de un motor en Potencia y Par respecto a las RPM.

La primera vez que se uso una red neuronal en una aplicación automotriz fue a inicios de la década de los noventa. En 1991, Marko probó varios clasificadores neuronales para el diagnostico en línea de defectos en el control del motor (*misfires*) y propuso un control directo vía un modelo neuronal inverso de un sistema activo de suspensión, [7]. En [8], Puskorius y Feldkamp, resumen una década de investigación, proponen la utilización de redes neuronales para controlar: la razón aire combustible en los cilindros, el control de la velocidad en *ralenti*, la detección de "*misfires*" (eventos en los que la mezcla aire-combustible no detona), el monitoreo del convertidor catalítico, la predicción de emisiones contaminantes. De hecho desde el inicio de los noventa diversos autores han propuesto las redes neuronales en la industria automotriz, ejemplo [9]:

Control del vehículo. Sistema anti-bloqueo de frenos (ABS), control activo de la suspensión, dirección, control de velocidad del vehículo.

Modelado del motor. Presión en el múltiple de admisión, flujo másico del aire de entrada, eficiencia volumétrica, estimación de la presión en los cilindros, razón aire-combustible [11], torque y potencia, consumo de combustible [28], etc.

Control de motor: *ralenti*, compensación de los transientes de combustible(TFC), control de la carga de aire dentro de los cilindros con válvulas de tiempo de apertura variable(VVT), control del tiempo de ignición, turbocargadores, válvulas EGR, reducción de contaminantes.

Diagnostico. Detección de cascabeleo y *misfires* [26], voltaje de las bujías.

En este ejemplo trataremos con un modelo matemático obtenido de [10] que representa a un motor de gasolina operando en *ralenti*, el cual es un proceso altamente no lineal. El modelo incluye retrasos en el tiempo que varían inversamente con la velocidad del motor, medida en rpm. Además el control de un motor real debe lidiar con procesos variantes en el tiempo tales como el envejecimiento de los componentes, cambios en el medio ambiente y condiciones de operación. La ocurrencia de perturbaciones en la planta tales como la entrada en funcionamiento del compresor del sistema de aire acondicionado, el cambio de neutral a “drive” en una transmisión automática, la aplicación de cargas eléctricas como el alternador son ejemplos de perturbaciones que el sistema de control debe compensar.

El objetivo del control de velocidad en ralenti (ISC *Idle Speed Control*) es regular la velocidad del motor a determinadas rpm a pesar de las perturbaciones que puedan afectar al sistema.

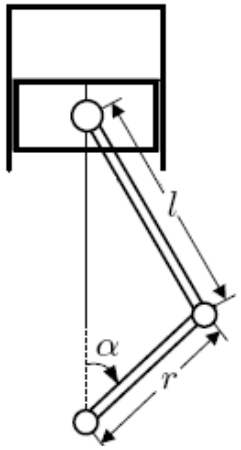


Figura 2.4 Diagrama esquemático de un pistón de un motor. En donde se aprecia el ángulo α con respecto al cuál se mide el ángulo de chispa

Una de las variables de control es el ángulo de chispa. En un motor de gasolina de 4 tiempos, este se mide como se puede observar en la figura 2.4, en donde el ángulo α, es el ángulo del cigüeñal, cuando α la cabeza del cilindro se encuentra en su punto más alto, mientras que si se encuentra a 180 º la cabeza está en su punto más bajo.

La variable de control es el ángulo de atraso o adelanto de la chispa δ con respecto al punto más alto del pistón, se mide en función de α, por ejemplo si α es cero en el momento de comandar la chispa, el ángulo de atraso será 0, sí α es menor que 0 se está adelantando δ.

2.3.1 Modelo Matemático

El modelo dinámico del motor a utilizar corresponde a un motor de 1.6 litros con 4 cilindros y esta dado por las siguientes ecuaciones diferenciales:

$$\dot{P} = k_P\left(\dot{m}_{ai} - \dot{m}_{ao}\right) \tag{2.4}$$

$$\dot{N} = k_N\left(T_i - T_L\right) \tag{2.5}$$

$$\dot{m}_{ai} = \left(1 + 0.907\theta + 0.0998\theta^2\right)g(P) \tag{2.6}$$

$$g(P) = \begin{cases} 1 & P < 50.6625 \\ 0.0197\left(101.325P - P^2\right)^{1/2} & P >= 50.6625 \end{cases} \tag{2.7}$$

$$\dot{m}_{ao} = -0.0005968N - 0.1336P + 0.0005341NP + 0.000001757NP^2 \tag{2.8}$$

$$m_{ao} = \dot{m}_{ao}(t - \tau)/(120N) \tag{2.9}$$

$$\tau = \frac{45}{N} \tag{2.10}$$

$$T_i = -39.22 + 325024m_{ao} - 0.0112\delta^2 + 0.635\delta + \left(0.0216 + 0.000675\delta\right) N(2\pi/60) - 0.000102N^2(2\pi/60)^2 \tag{2.11}$$

$$T_L = \left(N/263.17\right)^2 + T_d \tag{2.12}$$

En donde P es la presión del múltiple medida en kPa, N es la velocidad del motor medida en rpm, θ y δ son las señales de control, θ representa el ángulo de la mariposa que controla limita el flujo de aire de entrada, medido en el intervalo [5,25] y δ el ángulo de avance de la chispa medido en grados en el intervalo [10,45]. T_d representa a las perturbaciones que pueden provenir de los diversos accesorios del automóvil, tales como el alternador ó la puesta en marcha del compresor del aire acondicionado, con un rango de [0, 61] N-m.

La figura 2.5 muestra una representación gráfica de la planta, en ella se pueden apreciar las señales medidas, rpm y la Presión, así como las dos señales de control. La señal de perturbación se representa como un par aplicado al eje del motor. Las variables m_{ai} y m_{ao} son la masas de aire de entrada y salida respectivamente, m_{ao} es la masa en el cilindro, τ es el retraso en el tiempo debido a la dinámica de llenado del cilindro, $g(P)$ es una función de influencia de la presión en el múltiple, T_i es el par generado por el motor y T_L es el par de la carga.

2.3.2 Simulación de la Planta.

Para la simulación utilizamos MATLAB/Simulink con entradas escalón con magnitudes 10 y 30 para θ y δ respectivamente. La figura 2.6 muestra los

resultados de la simulación para un tiempo de 7 segundos. Los resultados obtenidos son los siguientes, encontramos que antes de la estabilización oscila nuestra planta, estas oscilaciones se presentan en los primeros dos segundos, además la figura 2.6 muestra la salida de la velocidad de motor, la *b* es la presión del múltiple, mientras que *c* y *d* son el ángulo de la mariposa y de la chispa respectivamente.

Podemos observar que al modificar el ángulo de la mariposa y dejar fija el de la chispa, las rpms se incrementan de 1000 rpms con un ángulo de 10º a 1800 para un ángulo de 20º, comparando la figura 2.6 a) y figura 2.7 a). Lo que significa en términos reales es que cuando uno aumenta la flujo de aire hacia al motor, es decir pisa el acelerador, la velocidad aumenta. También podemos apreciar que durante los primeros segundos, nuestra planta oscila, mientras que se estabiliza.

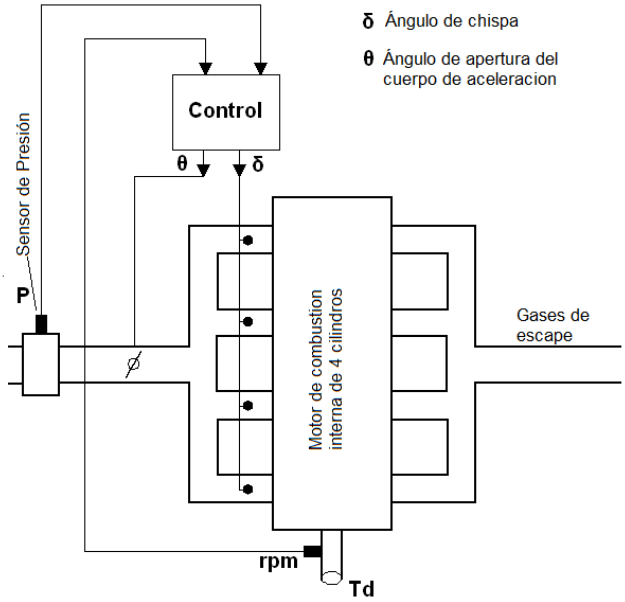


Figura 2.5 Representación gráfica del modelo de un motor de gasolina de cuatro cilindros

2.4 Sistema masa resorte con amortiguadores no lineales.

La dinámica no lineal de esta planta es de tipo cúbica, la cual dependiendo de su signo se clasifica en suave o dura [3]. Una no linealidad de este tipo se considera suave si el factor que multiplica al término cúbico es menor a cero y es considerada dura cuando es mayor a cero. Un resorte suave produce fuerzas cada vez más pequeñas conforme la deformación se hace más grande, en el caso de un resorte duro, por el contrario, deformaciones pequeñas producirán grandes fuerzas.

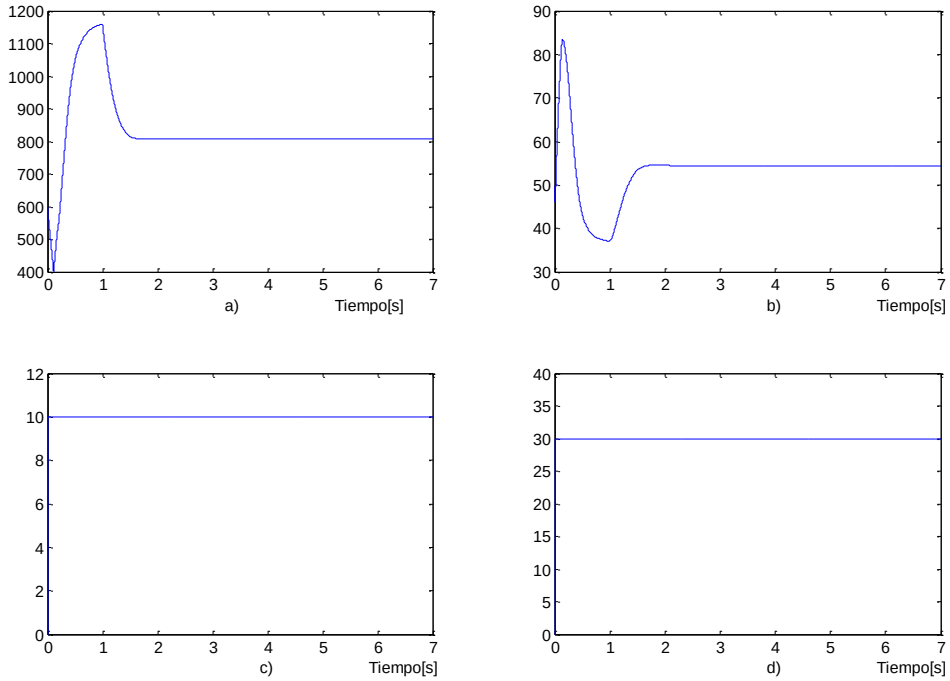


Figura 2.6 Resultados de Simulación, a) Revoluciones por minuto medidas en rpm, b) Presión medida en kPa, c) Señal de entrada θ y d) Señal de entrada δ.

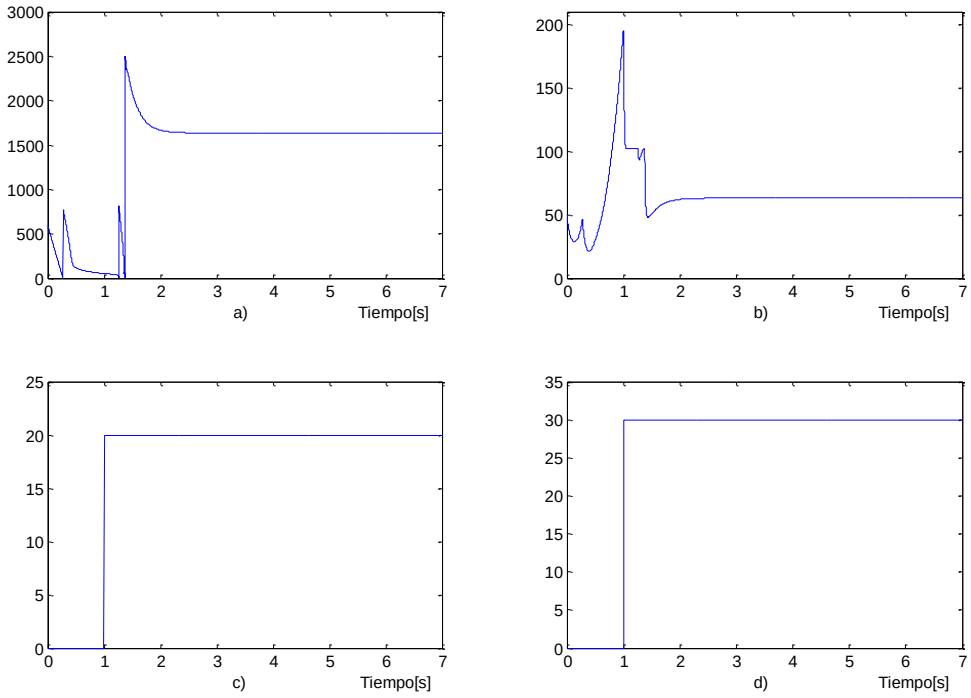


Figura 2.7 Resultados de Simulación, a) Revoluciones por minuto medidas en rpm, b) Presión medida en kPa, c) Señal de entrada θ y d) Señal de entrada δ.

2.4.1 Modelo Matemático

La planta está definida por las siguientes ecuaciones en espacio de estados:

$$\dot{x}_1 = x_2 \tag{2.13}$$

$$\dot{x}_2 = -\frac{1}{m}\left[a_1 x_1 + b_1 x_1^3 + k_1 x_2 + a_2(x_1 - x_3) - b_2(x_1 - x_3)^3 + k_2(x_2 - x_4)\right] + \frac{u_1}{m} \tag{2.14}$$

$$\dot{x}_2 = -\frac{1}{m}\left[g_1(x_1) + k_1 x_2 + g_2(x_1 - x_3) + k_2(x_2 - x_4)\right] + \frac{u_1}{m}$$

$$\dot{x}_3 = x_4 \tag{2.15}$$

$$\dot{x}_4 = -\frac{1}{M}\left[a_2(x_3 - x_1) - b_2(x_3 - x_1)^3 + k_2(x_4 - x_2)\right] + \frac{u_2}{M} \tag{2.16}$$

$$\dot{x}_4 = -\frac{1}{M}\left[g_2(x_3 - x_1) + k_2(x_4 - x_2)\right] + \frac{u_2}{M}$$

Donde x_1 y son x_3 son las posiciones de las masas a lo largo del eje x, x_2 y son x_4 son las velocidades de las masas m y M respectivamente, k_1 y k_2 son constantes del amortiguador y las fuerzas ejercidas por los resortes vienen dadas por las siguientes expresiones.

$$\begin{aligned} g_1(\varepsilon) &= a_1\varepsilon + b_1\varepsilon^3 \\ g_2(\varepsilon) &= a_2\varepsilon - b_2\varepsilon^3 \end{aligned} \tag{2.17}$$

Donde a_1, a_2 > 0, el signo (+) corresponde a resortes duros y el (–) a resortes suaves o blandos; la variable ε se corresponde a la elongación del resorte, [3].

2.4.2 Simulación de la Planta

La tabla 2.2, contiene los valores de los parámetros de la planta a usar en las simulaciones de todo este trabajo. Como se aprecia al observar los valores que definen a los resortes no lineales, tenemos una no linealidad suave y una fuerte.

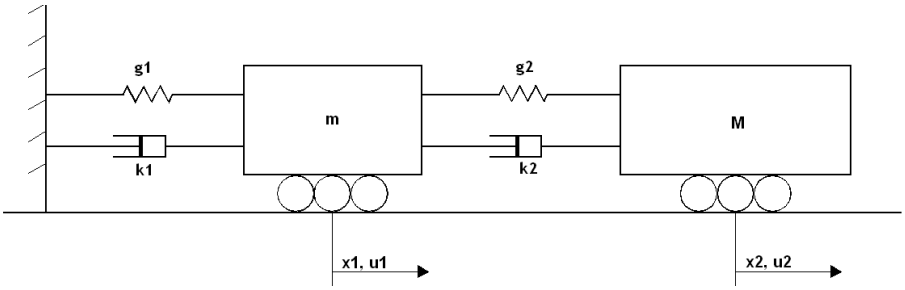


Figura 2.8 Diagrama del sistema de dos masas

Parámetro	Valor	Parámetro	Valor
k_1	5	a_1	3
K_2	3	a_2	2
M	10	b_1	1.5
m	20	b_2	-2.0

Tabla 2.2 Parámetros de la Planta

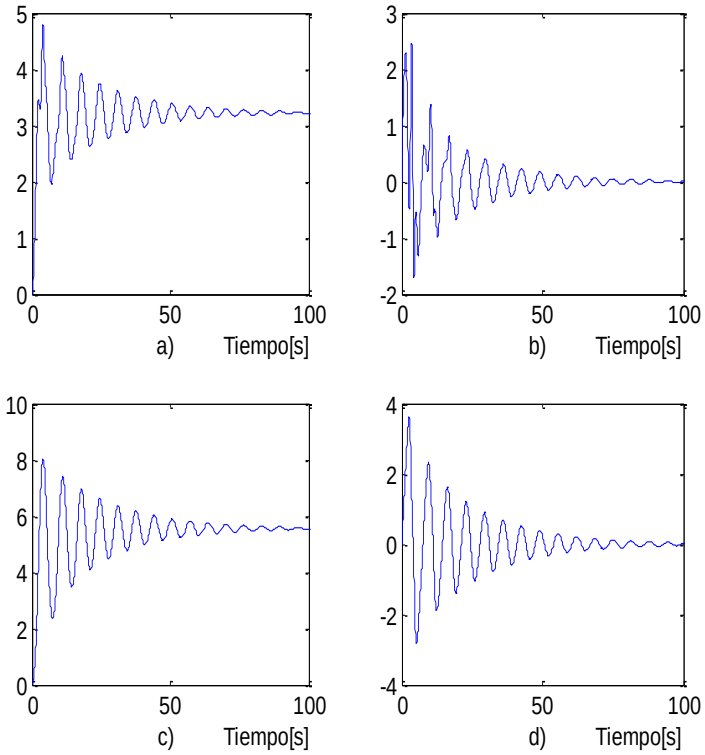


Figura 2.9 Resultados de Simulación, a) posición de la masa m, $x_1(t)$; b) velocidad lineal de la masa m ($x_2(t)$), c) posición de la masa M ($x_3(t)$), d) velocidad lineal de la masa M ($x_4(t)$).

2.5 Conclusiones

En este capítulo se presentó en un inicio la importancia de las oscilaciones en la dinámica de plantas mecánicas. Normalmente percibimos estas oscilaciones en forma de vibración o ruido, siendo la presencia de vibraciones un problema para estructuras o maquinaria, en tanto que el ruido puede ser causa de insatisfacción como es el caso del ruido producido por los frenos de un automóvil. Se presentaron los modelos matemáticos con los que se probará la efectividad de nuestra RNRE modular para atenuar las oscilaciones. Los modelos de las plantas escogidas son tanto SISO como MIMO.

Capitulo 3

Redes Neuronales Artificiales.

A pesar del hecho de que el término neuronal nos refiere rápidamente a un contexto biológico, este capítulo trata de presentar los fundamentos matemáticos y algoritmos que definen a las redes neuronales artificiales y las redes neuronales recurrentes.

3.1 Redes Neuronales Artifíciales

Una neurona es una función no lineal acotada $y = f(x_1, x_2, \cdots x_n; w_1, w_2, \cdots w_p)$ donde $\{x_i\}$ son las variables y $\{w_i\}$ son los parámetros o pesos de la neurona. Las variables de la neurona son usualmente conocidas como entradas. La representación grafica está en la Figura 3.1. Una neurona artificial está compuesta por un vector de entrada, pesos sinápticos, un valor de umbral, un operador matemático de suma y una función de activación. El primer modelo matemático de una neurona biológica fue desarrollado por McCulloch y Pitt [5], el cual es el siguiente:

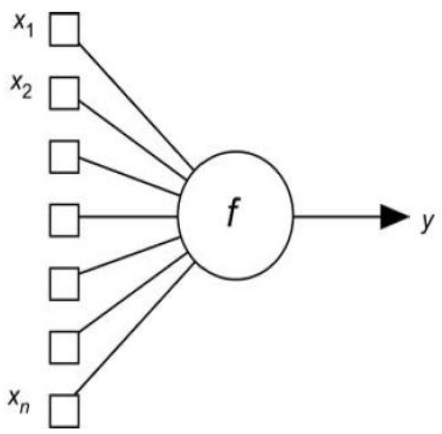


Figura 3.1 Representación gráfica de una Neuronal

$$y_j = \varphi(\nu_j + \theta_j) \tag{3.1}$$

$$\nu_j = \sum_{i=1}^{m} w_{ji} x_i \tag{3.2}$$

Donde:

- $X = (x_1, x_2 \cdots x_m)$ son las señales de entrada. Los datos pueden representar información proveniente del medio ambiente o bien, ser salidas de neuronas anteriores.
- $w_{ji} \in \Re^{m+1}$ son los pesos sinápticos de la j-ésima neurona. Estos son propagados a través de la red cada componente donde X es multiplicado por un peso w_{ji}, la cual aumenta o atenúa la señal de entrada. Durante el proceso de aprendizaje los pesos cambian de valor hasta que convergen o tienen una variación acotada.

- θ_j es el umbral cuya entrada siempre es constante de valor -1, la cual pondera al peso w_{j0}. Sirve para trasladar el punto de equilibrio de la red neuronal del origen a un punto deseado.
- y_j salida se obtiene sumando las entradas ponderadas de la neurona y evaluando el resultado de esta suma en la función de activación no lineal $\varphi(\cdot)$. [2]

3.1.1 La función de activación

La función de activación es para proveer un comportamiento no lineal a la red y así poder aproximar funciones de varios tipos y las más utilizadas son:

- **Identidad**. Tiene la forma $f(v) = v$ y se desea cuando no se desea acotar la salida de la neurona.
- **Función escalón**. Está definida por la siguiente expresión:

$$\varphi(v) = \begin{cases} 1 & si\ v \geq 0 \\ -1 & si\ v < 0 \end{cases} \tag{3.4}$$

- **Función saturación**. Matemáticamente se expresa de la siguiente forma:

$$\varphi(v) = \begin{cases} -1 & si \quad v < -c \\ 1 & si \quad v > c \\ av & \text{en otro caso} \end{cases} \tag{3.5}$$

  Donde *a* es la pendiente de la recta y *c* es el punto de saturación
- **Función sigmoide**. Son un conjunto de funciones crecientes, monótonas y acotadas que provocan una transformación no lineal de su argumento. Una de las funciones más utilizadas es la siguiente, en donde *p* es la pendiente de la función de activación. Esta función se encuentra acotada entre los valores 0 y -1.

$$\varphi(v) = \frac{1}{1 + \exp(-pv)} \tag{3.6}$$

- **Función Tangente hiperbólica**. Esta es un tipo especial de función sigmoide la cual esta acotada entre los valores 1 y -1.

$$\varphi(v) = \frac{1 - \exp(-pv)}{1 + \exp(-pv)} \tag{3.7}$$

- **Función de base radial**. Las más comunes son funciones gaussianas no monótonas, como:

$$\varphi(v) = \exp\left(-\frac{v^2}{2\sigma^2}\right) \tag{3.8}$$

3.1.2 Topología de las Redes Neuronales

Basados en su arquitectura, las redes neuronales pueden ser agrupadas dentro de tres categorías:

- **Redes con conexión hacia delante ó *Feedforward* (Monocapa)** figura 3.2: su nombre hace referencia a que están constituidas por una sola capa, la capa de salida. Las neuronas de entrada no se toman en cuenta ya que en ellas no se realiza ningún cálculo, [3]. Estas redes son usualmente usadas en tareas de asociación., esta red es estrictamente *feedforward* o acíclica.

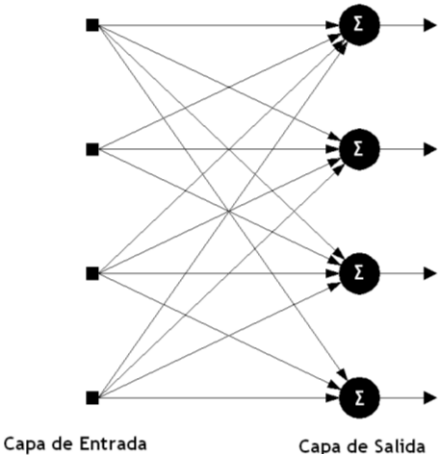


Figura 3.2 Red Monocapa

- **Redes (*feedforward)* Multicapa ó redes estáticas**: consta de una o más capas ocultas, cuyos elementos son llamados neuronas ocultas. La función de las neuronas ocultas es intervenir de una manera útil entre la señal de entrada y la salida de la red. Añadiendo mas capas ocultas, la neurona es capaz de extraer mayor información de su entorno. Todas las señales neuronales se propagan hacia delante a través de las capas que conforman la red, no existen conexiones hacia atrás y normalmente tampoco autorrecurentes, ni laterales. Son muy útiles en aplicaciones como el reconocimiento de patrones o clasificación de patrones. Las más conocidas son: el Perceptrón Multicapa, *Adaline, Madaline, Drive-Reinforcement,* etc. La figura 3.3 nos muestra la representación gráfica de un perceptrón multicapa con una capa oculta conectado completamente.

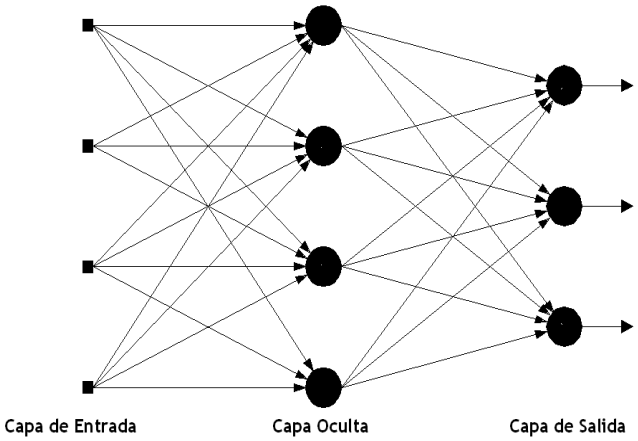


Figura 3.3 Red Multicapa

- **Redes Neuronales Recurrentes (RNR)** en las cuales tenemos lazos de retroalimentación: se distingue de una red *feedforward* porque tiene al menos un lazo de retroalimentación. Por ejemplo, una red recurrente puede consistir de una sola capa de neuronas con cada neurona alimentando con su señal de salida la entrada de las demás neuronas. Se extenderá el tema en la sección 3.2.

- **Perceptrón multicapa:** ésta red neuronal consiste en un conjunto de neuronas que constituyen la capa de entrada, una o más capas ocultas y una capa de salida. La señal de entrada se propaga capa por capa a través de la red en dirección de entrada a salida. Esta red ha sido utilizada exitosamente para resolver diversos problemas en conjunto con el algoritmo de aprendizaje *backpropagation*. El perceptrón multicapa posee tres características distintivas: el modelo de cada neurona posee una función de activación no lineal y suave, la red contiene una o más capas ocultas de neuronas que no forman parte ni de la entrada o salida de la red, son estas neuronas de la capa oculta las que permiten a la red aprender tareas complejas. La ultima característica es que exhibe un alto grado de conectividad, determinado por el numero de sinapsis dentro de la red.

3.1.3 Proceso de Aprendizaje.

La propiedad más importante de una red neuronal es su capacidad de aprender sobre su medio ambiente y mejorar su desempeño a través del aprendizaje. Una red neuronal aprende de su medio ambiente y mejora su desempeño a través del aprendizaje. Una red neuronal aprende de su medio a través de un proceso de iterativo de corrección, aplicado a los pesos sinápticos. Idealmente, la red requiere más conocimiento sobre su medio ambiente después de cada iteración del proceso de aprendizaje.

Entonces se puede definir el proceso de aprendizaje como: “Procesamiento por el cual los parámetros libres de una red neuronal son ajustados a través de un proceso de simulación del entorno en el cual la red es sometida. El tipo de aprendizaje es determinado por la manera en la cual los cambios de los parámetros toman lugar " [2].

Los algoritmos difieren unos de otros por el método en que se corrigen los pesos sinápticos, así por ejemplo tenemos las siguientes reglas de aprendizaje:

- **Aprendizaje por corrección por error: l**os pesos sinápticos de la red se ajustan en función de la diferencia entre los valores deseados y los

obtenidos de la salida de la red, es decir, se ajustan en base al error cometido por la red durante el aprendizaje.

Un algoritmo simple de este tipo puede estar definido por la siguiente expresión:

$$\Delta w_{ji} = \alpha(y_{di} - y_i)u_i \quad (3.9)$$

En donde Δw_{ji} es la variación del peso sináptico que está entre la neurona i y la j; y_{di} es el valor deseado para la j-ésima neurona y α es la constante de aprendizaje con valor menor a la unidad.

Existen diversas versiones mejoradas de este método, tales como la regla Delta, el Algoritmo Backpropagation (Retropropagacion del error), algoritmo Levenberg-Marquardt, [4].

- **Aprendizaje competitivo y cooperativo:** en las redes con aprendizaje se dice que las neuronas compiten o cooperan entre ellas con el fin de lograr la tarea deseada. Con este aprendizaje se pretende que cuando se presente a la red cierto tipo de información de entrada, solo una de las neuronas de la salida de la red, o un cierto grupo de neuronas se active (alcance su valor de respuesta máximo). Por lo tanto, las neuronas compiten por activarse, quedando finalmente una, o un grupo de ellas como vencedoras. El resto de las neuronas quedan anuladas al ser forzados sus valores de respuesta al mínimo.

La competencia entre neuronas se realiza en todas las capas de la red, existiendo en estas neuronas conexiones recurrentes de auto-excitación y conexiones de inhibición (signo negativo) por parte de neuronas vecinas. Si el aprendizaje de la red es cooperativo, estas conexiones con las vecinas serán de excitación (signo positivo).

El objetivo de este aprendizaje es categorizar los datos que se introducen en la red. De esta forma, las informaciones similares son clasificadas formando parte de la misma categoría, y por tanto deben activar la misma neurona de salida. Las clases o categorías deben ser las correcciones entre los datos de entrada, [4].

- **Aprendizaje Hebbiano:** se basa en el postulado formulado por Danald O. Hebb en 1949: Cuando un axón (conexión entre dos neuronas) de una celda A esta suficientemente cerca como para conseguir excitar a una celda B y repetida o persistentemente toma parte en su activación, algún proceso de crecimiento o cambio metabólico tiene lugar en una o en ambas celdas, de tal forma que la eficiencia de A, cuando la celda es activar B, aumenta. Por celda, se define a un conjunto de neuronas fuertemente conectadas a

través de una estructura compleja. La eficiencia podría identificarse con la intensidad o magnitud de la conexión; es decir, con el peso.

Se puede decir, por tanto que el aprendizaje Hebbiano consiste básicamente en el ajuste de los pesos de las conexiones de acuerdo con la corrección de los valores de activación o salida de las neuronas conectadas; es decir:

$$\Delta w_{ji} = y_i y_j \tag{3.10}$$

Donde y_i es la salida de la neurona *i* y y_j es la salida de la neurona *j*.

Esta expresión responde a la idea de Hebb puesto que si las dos unidades son activas (positivas), se produce un reforzamiento de la conexión. Por el contrario, cuando una es activa y la otra pasiva (negativa), se produce un debilitamiento de la conexión. Se trata de una regla de aprendizaje no supervisado, pues la modificación de los pesos se realiza en función de los estados (salidas) de las neuronas obtenidas tras la presentación de cierto estimulo (entrada), sin tener en cuenta si se deseaba o no obtener esos estados de activación, [4].

- **Aprendizaje de Boltzmann:** la regla de aprendizaje de Boltzmann es un algoritmo estocástico derivado de la mecánica estadística. Una red neuronal diseñada en base a esta regla de aprendizaje es llamada una maquina de Boltzmann.

En un maquina de Boltzmann, las neuronas constituyen una estructura recurrente y ellas operan en una manera binaria ya que, por ejemplo, pueden estar en un estado "*on"* encendido denotado por +1 o en un estado "*off"* apagado denotado por -1. La máquina está caracterizada por una función de energía, E, el valor de la cual está determinado por los estados particulares de las neuronas individuales de la maquina, como se muestra en la siguiente ecuación:

$$E = -\frac{1}{2}\sum_{\substack{j \\ j \neq k}}\sum_{k} w_{kj} x_k k_j \tag{3.11}$$

Donde x_k es el estado de la neurona *j*, y w_{kj} es el peso sináptico que conecta a la neurona *j* con la *k*. el hecho de que $j \neq k$ significa solamente que ninguna de las neuronas de la maquina tiene retroalimentación. La maquina funciona escogiendo una neurona aleatoriamente (por ejemplo la neurona *k*) en algún paso del proceso de aprendizaje, entonces invierte el estado de la neurona *k*, de estado x_k al estado - x_k a cierta temperatura *T* con probabilidad:

$$P(x_k \longrightarrow -x_k) = \frac{1}{1+\exp\left(-\Delta E_k / T\right)} \tag{3.12}$$

Donde ΔE_k es el cambio en la energía. Si esta regla es aplicada repetidamente la maquina alcanzaría el equilibrio térmico, [2].

- **Aprendizaje supervisado:** este aprendizaje se realiza se realiza por medio de un entrenamiento controlado mediante un agente externo o supervisor, el cual determina la respuesta que debe generar la red a partir de una entrada determinada.

El supervisor compara la salida de la red con la salida deseada y en caso de existir una diferencia procede a modificar los pesos de las conexiones con el propósito de que la salida de que la salida de la red se aproxime a la salida deseada [4].

- **Aprendizaje no supervisado:** este aprendizaje carece de un agente externo para su aprendizaje por lo que se dice que es un aprendizaje auto organizado. Sin embargo, existe una medida independiente de la salida de aprendizaje y entonces los pesos son optimizados con respecto a esta medida. El aprendizaje no supervisado puede usar una regla de aprendizaje competitivo.

- **Aprendizaje reforzado:** es un aprendizaje en línea de un mapeo entrada-salida a través de un proceso de prueba diseñado para maximizar un índice de desempeño escalar llamado señal de reforzamiento. El aprendizaje reforzado puede ser de dos tipos, asociativo y no asociativo.

3.2 Redes Neuronales Recurrentes

Las redes neuronales recurrentes son también conocidas como redes dinámicas dado que cuentan con conexiones de retroalimentación que pueden tanto locales o globales. Las ecuaciones que definen a las RNR (Redes Neuronales Recurrentes) en tiempo continuo son las siguientes:

$$\begin{aligned} \dot{x} &= f\left(x, u, w_1\right) \\ y &= g\left(x, w_2\right) \end{aligned} \tag{3.14}$$

En donde x es el estado de la red neuronal, u es la señal de entrada, w_1 y w_2 son los pesos sinápticos f y g son funciones de activación suave y acotada.

La presencia de lazos de retroalimentación tiene un gran impacto en la capacidad de aprendizaje de la red y en su desempeño. Cuando la red es de más de una capa tenemos que la información puede circular tanto hacia delante como hacia atrás. En general en este tipo de conexiones existen dos conjuntos de pesos: los correspondientes a las conexiones *feedforward* provenientes de la capa de entrada, las cuales se propagan a través de la red y las de las conexiones *feedback* [35]. Los valores de los pesos de estos tipos de redes son diferentes en la mayor parte de los casos.

Algunas redes de este tipo tienen un funcionamiento basado en lo que se llama resonancia, de tal forma que la información entre la primera y segunda capas interactúa entre si hasta alcanzar un estado estable. Este funcionamiento permite un mejor acceso a la información almacenada en la red. Las conexiones laterales entre neuronas de una misma capa se diseñan de tal forma que existen conexiones excitadoras e inhibidoras. Una conexión excitadora posee pesos sinápticos positivos que permiten la cooperación entre las neuronas que conectan, mientras que una conexión inhibidora posee pesos sinápticos negativos los cuales inducen la competencia entre las neuronas correspondientes. La figura 3.4 muestra un ejemplo de red recurrente conectada completamente.

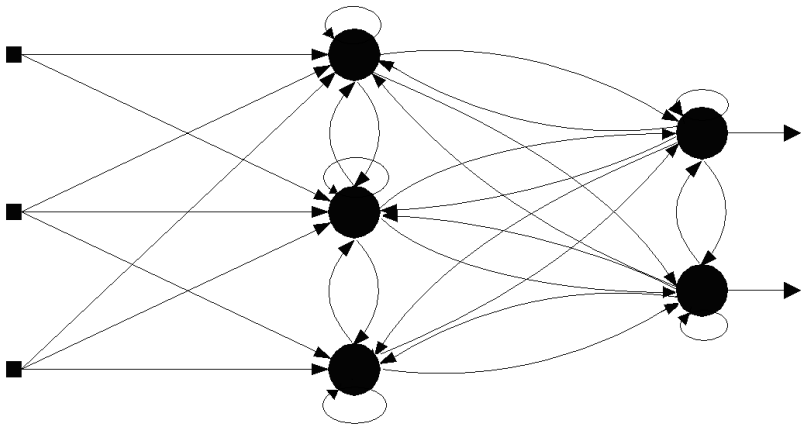


Figura 3.4 Red Recurrente conectada completamente.

3.2.1 Red de Hopfield.

La red de Hopfield es un tipo de red recurrente en donde el número de lazos de retroalimentación es igual al número de neuronas, está formada por un conjunto de neuronas y bloques de retardos, los cuales forman un sistema retroalimentado tal como se muestra en la figura 3.5.

La ecuación que describe a la red de Hopfield es la siguiente:

$$x_j(k+1) = f(v_j(k))$$
$$v_j(k) = \sum_{i=1}^{n} w_{ij} x_i(k) + \theta_j \qquad (3.15)$$

Donde f es una función de activación de tipo escalón, x_i es el estado de la red, w_{ij} son los pesos sinápticos y θ_j es el umbral.

La salida de cada neurona se pondera y retroalimenta a través de un elemento de retardo a cada una de las neuronas de la red, la presencia de retroalimentación en esta red la hace interesante para aplicaciones de modelado y control de sistemas dinámicos, [5].

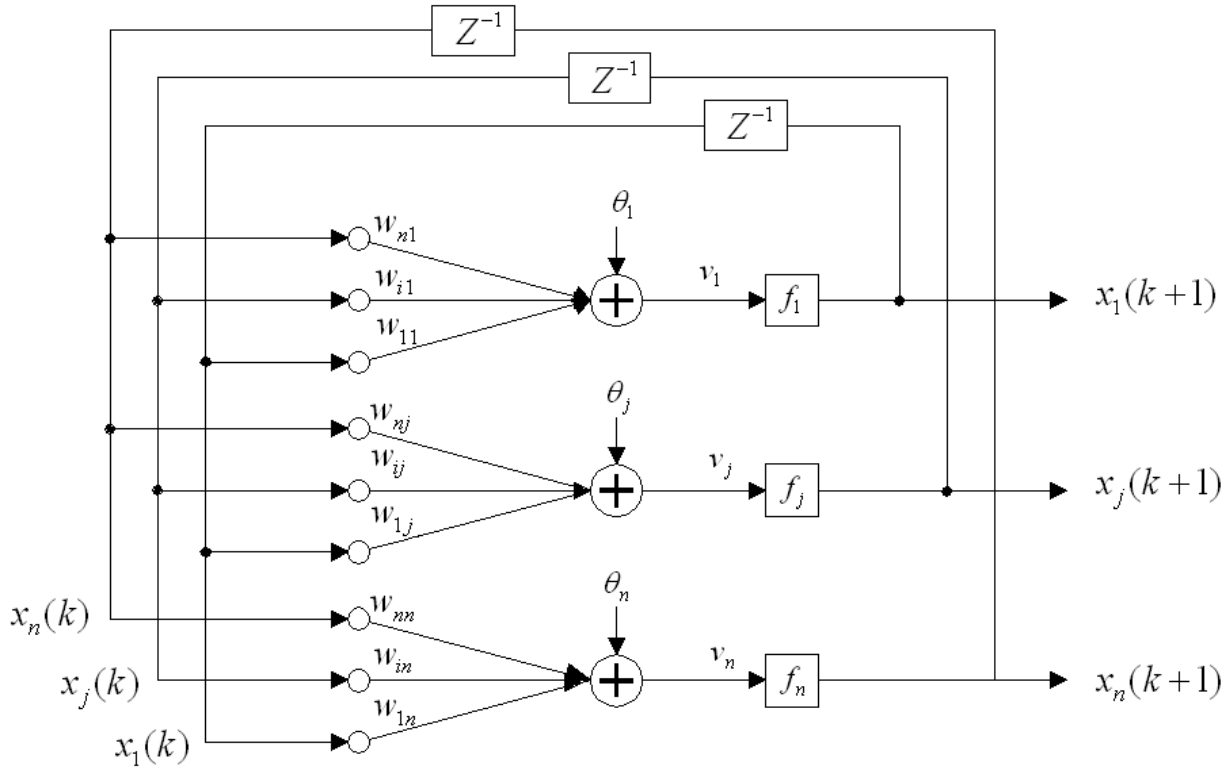


Figura 3.5 Red de Hopfield

Aprendizaje de la red de Hopfield

La red de Hopfield tiene un mecanismo de aprendizaje fuera de línea. Por lo tanto, existe una etapa de entrenamiento y otra de funcionamiento de la red. En la etapa de entrenamiento se fijan los valores de los pesos en función de la información que se pretende almacenar en la red, una vez establecidos, la red entra en funcionamiento.

Esta red utiliza un aprendizaje no supervisado de tipo Hebbiano, de tal forma que el peso de la conexión entre una neurona *i* y otra *j* se obtiene del producto de los componentes *i*-ésimo y *j*-ésimo del vector que representa la información o patrón que debe almacenar.

Muchas de las investigaciones acerca de la estabilidad de las redes se basan en el establecimiento de una función, denominada función de energía de la red, para representar los posibles estados de la red.

La función de energía de una red de Hopfield discreta tiene la siguiente forma:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\left(\sum_{\substack{j=1 \\ i\neq j}}^{N} w_{ij}x_i x_j + \theta_i x_0\right) \tag{3.16}$$

Donde w_{ij} es el peso de la conexión entre la neurona *i* y la *j*, x_i es la entrada de la neurona i, θ_i es el umbral. Cuando se presenta a la entrada de la red una nueva información, esta evoluciona hasta alcanzar un mínimo en la función de energía, generando una salida estable.

3.3 Red Neuronal Recurrente Entrenable Modular

En los últimos años investigadores como Baruch et. al. [12], [13], han trabajado con una nueva topología de red recurrente, llamada Red Neuronal Recurrente Entrenable (RNRE). En [12], se expone una RNRE con una estructura canónica de Jordan, y en [13] se da su extensión para el caso multivariable. En [14] se presenta un modelo de la RNRE con restricciones en los pesos de retroalimentación para garantizar la estabilidad de la red. En este trabajo se utilizó una topología modular la cual se explica en la siguiente sección.

La principal ventaja de la RNRE modular es que si la dinámica no lineal de la planta puede ser descompuesta en diversas dinámicas de menor orden, entonces es posible modelar dichas dinámicas con los bloques de la estructura de la RNRE modular.

3.3.1 Topología de la Red Neuronal Recurrente Entrenable Modular.

La RNRE modular esta descrita por el siguiente conjunto de ecuaciones [15]:

$$x_D(k+1) = A_D x_D(k) + B_D u(k);$$
$$x_C(k+1) = A_C x_C(k) + B_C u(k); \quad (3.17)$$

$$z_D(k) = G[x_D(k)];$$
$$z_C(k) = G[x_C(k)]; \quad (3\text{-}18)$$

$$y(k) = F[C_D z_D(k); \quad C_C z_C(k)]; \quad (3.19)$$

$$A_D = block - diag(A_{Di}); \quad |A_D| < 1$$
$$A_C = block - diag(A_{Ci}); \quad |A_C| < 1 \quad (3.20)$$

Donde $y(k)$ es el vector de salida con dimensión l, $x(k)$ es el vector de estado con dimensión n, $u(k)$ es el vector de entrada con dimensión m; A es la matriz de pesos de la capa oculta con una estructura diagonal a bloques y dimensión $n \times n$, B es la matriz de pesos de la capa de entrada con dimensión $n \times m$, C es la matriz de pesos de la capa de salida con dimensión $l \times m$; $G(\cdot)$ y $F(\cdot)$ son vectores de funciones de activación, con elementos funcionales tales como saturación, sigmoidea o tangente hiperbólica.

La figura 3.6 muestra un ejemplo del tipo de red a utilizar, en este caso se presenta una topología 2-4-2, en donde se aprecia que las neuronas 3 y 4 de la capa oculta se retroalimentan mutuamente, lo que matemáticamente se expresa como un bloque de tamaño dos por dos en la diagonal de la matriz A. En la figura 3.7 se muestra la representación simplificada de la red en un diagrama a bloques.

Debido a que la red debe ser entrenable, los valores propios del modelo deben permanecer dentro del círculo unitario, esta es la razón de la restricción en la ecuación 3.20. También dado que una red neuronal recurrente modular tiene una estructura más simple que una totalmente retroalimentada, su implementación en tiempo real se vuelve más viable. Por otra parte la mayoría de las redes neuronales recurrentes utilizada en la literatura poseen un solo peso de retroalimentación en la capa oculta, mientras que nuestra red tiene la ventaja de tener uno o más bloques capaces de identificar la parte oscilatoria.

La arquitectura descrita de la RNRE modular puede ser usada como un estimador de estados e identificador de sistemas. Otra propiedad de este modelo es que es globalmente no lineal pero localmente lineal. Esto es por lo que las matrices generadas durante el aprendizaje pueden ser usadas para diseñar una ley de control. Más aún, el modelo de la RNRE es robusto, debido a la ley de adaptación dinámica de los pesos, la cual está basada en el modelo de sensibilidad de la red.

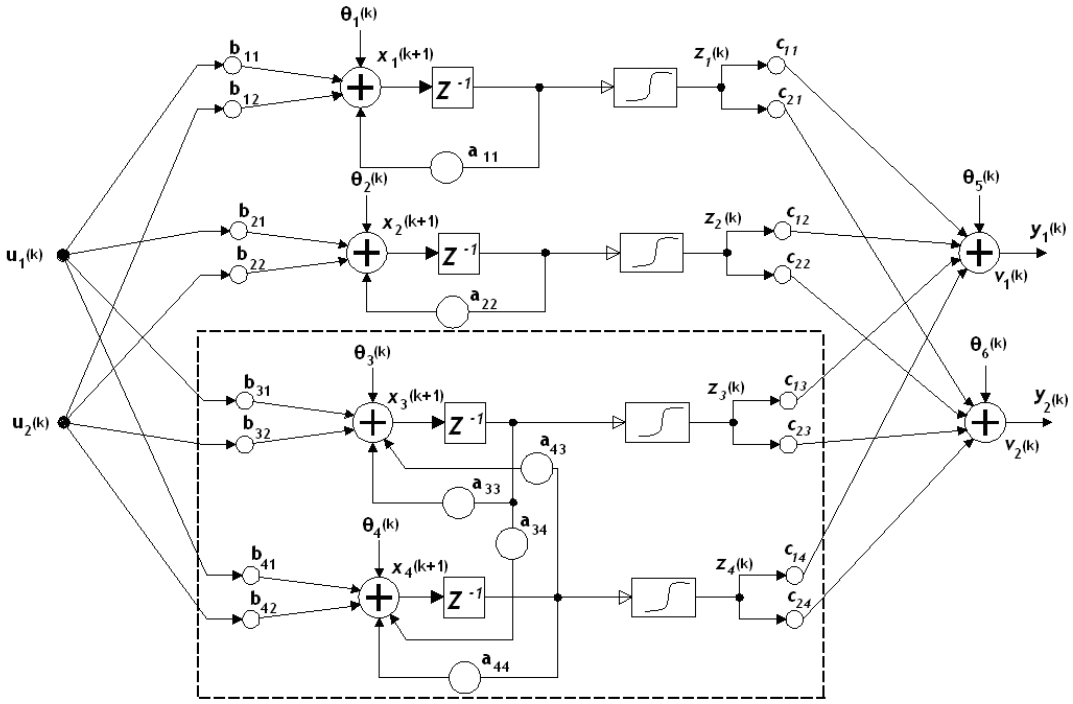


Figura 3.6 Ejemplo de una red Neuronal Recurrente Entrenable Modular de dos módulos

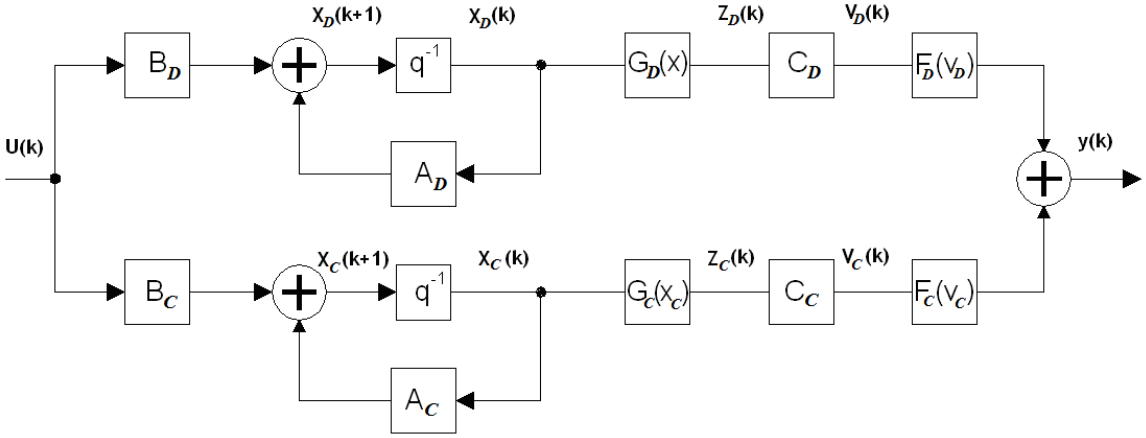


Figura 3.7 Esquema de la red Neuronal Recurrente Entrenable Modular

3.3.2 Observabilidad de la RNRE modular.

Una RN se dice observable si el estado de la red puede ser determinado para un conjunto finito de pares entrada-salida medible [35], [36]. Reescribiendo las ecuaciones 3.17 a 3.20 de la siguiente forma:

$$x(k+1) = Ax(k) + Bu(k); \tag{3.21}$$

$$y(k) = f[Cf(x(k))]; \tag{3.22}$$

Dado que f es una función diferenciable, podemos reescribir la ecuación 3.21 como:

$$y(k) = C_1 x(k); \tag{3.23}$$

Si q es la dimensión del espacio de estados, podemos calcular *q* salidas de la red, como sigue:

$$y(k+1) = C_1 x(k+1) = C_1 Ax(k) + C_1 Bu(k); \tag{3.24}$$

$$y(k+2) = C_1 A^2 x(k) + C_1 ABu(k) + C_1 Bu(k+1);$$

$$\vdots \tag{3.25}$$

$$y(k+q-1) = C_1 A^{q-1} x(k) + C_1 A^{q-2} Bu(k) + \cdots + C_1 ABu(k+q-3) + C_1 Bu(k+q-2);$$

Entonces podemos decir que el sistema dado por (3.21) y (3.22) es observable sí

$$M_O = [C_1, C_1 A, \cdots, C_1 A^{q-1}] \tag{3.26}$$

Es de rango pleno. M_O es la matriz de Observabilidad.

Sea $u_{q-1}(k) = [u(k), u(k+1), \cdots, u(k+q-2)]^T$ una secuencia de entradas de la RNRE modular, entonces dado un estado inicial $x(k)$, $u_{q-1}(k)$ genera la secuencia de salidas $y_p(k) = [y(k), y(k+1), \cdots, y(k+q-1)]^T$. Considerando la función $F : \Re^{2q-1} \to \Re^{2q-1}$, $(u_{q-1}(k), x(k)) \mapsto (u_{q-1}(k), y_q(k))$.

Por tanto el Jacobiano de F respecto a $u_q(k)$ y $x(k)$ evaluado en el origen es:

$$J^{obs}_{(0,0)} = \begin{bmatrix} I & S \\ 0 & M_O \end{bmatrix} \tag{3.27}$$

Donde nuevamente I es la matriz identidad, 0 es la matriz nula y S no tiene ningún interés.

Dado que $Det(J^{obs}_{(0,0)}) = Det(M_O)$ tenemos que si M_O es de rango pleno $J^{obs}_{(0,0)}$ también lo tendrá.

Otra vez aplicando el teorema de la función inversa tenemos que existe una función E tal que $E[F(x)] = x, \ \forall x \in \Re^{2q-1}$ entonces $(u_{q-1}(k), x(k)) = E(u_{q-1}(k), y_q(k))$ esto quiere decir que en una vecindad del origen, $x(k)$ es una función lineal de $u_{q-1}(k)$ y $y_q(k)$, y que esa función no

lineal es un observador de la RNRE. Formalmente queda dicho en el siguiente teorema cuya demostración hemos presentado.

Teorema 3.1 Sea la RNRE definida por (3.9) y (3.10). Entonces, la RNRE es observable alrededor del origen.

3.3.3 Controlabilidad de la RNRE

Un sistema se dice controlable si un estado inicial es dirigido a un estado deseado en un número finito de pasos. Aquí nos referimos a controlabilidad local en una vecindad del punto de equilibrio, supongamos sin pérdida de la generalidad que el origen es un estado de equilibrio.

Para demostrar que una RNRE es controlable supongamos sin pérdida de generalidad que la red posee una entrada y un salida. Desarrollando la ecuación (3.21) tenemos:

$$x(k+1) = Ax(k) + Bu(k); \tag{3.21}$$

$$\begin{aligned} x(k+2) &= A^2x(k) + ABu(k) + Bu(k+1); \\ &\vdots \\ x(k+q) &= A^qx(k) + A^{q-1}Bu(k) + \cdots + ABu(k+q-2) + Bu(k+q-1); \end{aligned} \tag{3.22}$$

Donde q es la dimensión del espacio de estados.

Entonces podemos afirmar que el sistema a (3.21) es controlable si

$$M_C = [A^{q-1}B, \cdots, AB, B] \tag{3.23}$$

Es de rango pleno, ya que entonces (3.21) tiene solución única. M_C es la matriz de Controlabilidad del sistema. Sea $u_q(k) = [u(k), u(k+1), \cdots, u(k+q-1)]^T$ una secuencia de entradas de la RNRE. Considerando la función $G : \Re^{2q} \to \Re^{2q}$, $(x(k), u_q(k)) \mapsto (x(k), x(k+q))$. Entonces el Jacobiano de G con respecto a $x(k)$ y $u_q(k)$ evaluando el origen es:

$$J^{ctrl}_{(0,0)} = \begin{bmatrix} I & S \\ 0 & M_C \end{bmatrix} \tag{3.24}$$

Donde nuevamente I es la matriz identidad, 0 es la matriz nula y S no tiene ningún interés. Dado que $Det\left(J^{ctrl}_{(0,0)}\right) = Det(M_C)$ tenemos que si M_C es de rango pleno $J^{ctrl}_{(0,0)}$ también lo tendrá.

Teorema 3.2 (De la función inversa), [17]; supongamos que $f=(f_1, f_2,\cdots, f_n)\in C^1$ en un conjunto abierto $q\subseteq \Re^n$, y sea $T=f(Q)$. Si el determinante jacobiano $J(a)\neq 0$ en un punto $a\in Q$, entonces existen dos conjuntos abiertos $X\subseteq Q$ y $Y\subseteq T$ y una función g unívocamente determinada tal que, [18];

a) $a\in Q$ y $f(a)\in Y$
b) $Y=f(X)$
c) f es uno a uno en X
d) g está definida en Y, $g(Y)=X$ y $g[f(x)]=x,\ \forall x\in X$
e) $g\in C^1$ en Y

Ahora supongamos que nuestra G es f en el teorema anterior y puesto que G cumple con la hipótesis entonces tenemos que $\exists H$ tal que $H[G(x)]=x,\ \forall x\in \Re^{2q}$ entonces $(x(k), x(k+q))=H(x(k), u_q(k))$. Esto quiere decir que existe una secuencia $\{u_q(k)\}$ tal que localmente puede llevar a la RNRE del estado $x(k)$ al estado $x(k+q)$ en q pasos.

Teorema 3.3. Sea la RNRE definida por (3.8) y (3.9). Entonces la RNRE es localmente controlable alrededor del origen.

De los teoremas anteriores podemos afirmar que dado que la RNRE es observable y controlable, entonces es posible obtener una ley de aprendizaje para la RNRE.

3.4 Redes Neuronales Recurrentes de Segundo Orden

En esta sección y la siguiente se presentan dos de ejemplos de redes neuronales recurrentes existentes en la literatura con las cuales comparamos a nuestra RNRE modular.

En el articulo [37] fig. 3.8, se propone una red diagonal recurrente que contiene dos pesos recurrentes en su capa oculta. Esta red se utiliza para la identificación del modelo matemático de una planta no lineal, así como para la identificación y control de una planta real la cual es un periscopio.

El modelo matemático de esta red es:

$$y(k) = O(k) = \sum W_j^0 Z_j(k) \tag{3.25}$$

$$Z_j = \rho(H_j(k)) \tag{3.26}$$

$$H_j = \sum W_{ij}^I u_j + W_j^{D1} Z_j(k-1) + W_j^{D2} Z_j(k-2) \tag{3.27}$$

Donde la función de activación denotada por $\rho(*)$ es una sigmoide, W_j^0 es la matriz de pesos de la salida, W_j^{D1} y W_j^{D2} son las matrices de los pesos de la capa oculta, u_j es el vector de entrada e $y(k)$ es el vector de salida. La función de costo es el error medio cuadrático, dado por:

$$E(k) = \frac{1}{2}(e(k))^2 = \frac{1}{2}(y_d(k) - y(k))^2 \tag{3.28}$$

En donde y_d es la salida deseada de la red.

Los pesos son ajustados por la siguiente ecuación:

$$W(k+1) = W(k) + \eta(-\frac{\partial E}{\partial W}) \tag{3.28}$$

$$\frac{\partial E}{\partial W} - e(k) \cdot \frac{\partial y(k)}{\partial W} = -e(k) \cdot \frac{\partial O(k)}{\partial W} \tag{3.29}$$

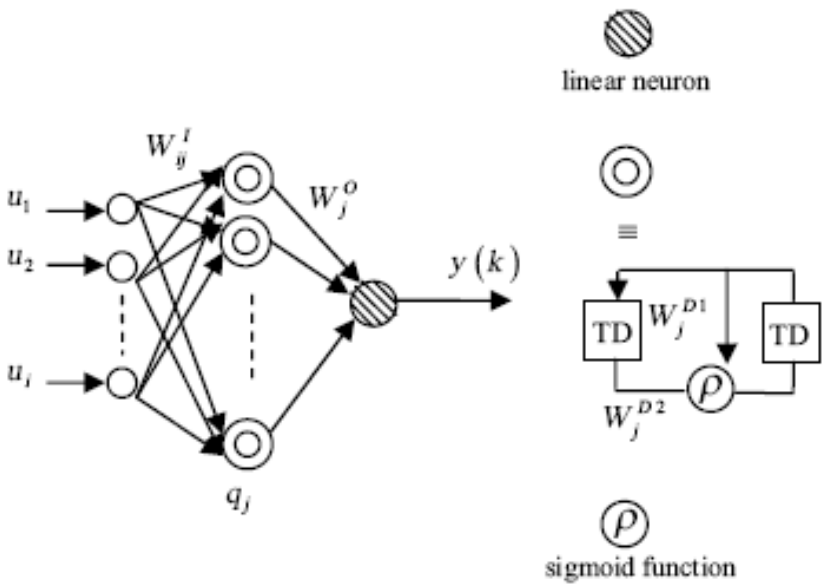


Figura 3.8 Estructura de la red neuronal recurrente de segundo orden

3.5 Redes Neuronal Recurrentes con problema de Optimización con Restricciones (RENNCOM).

El algoritmo llamado Redes Neuronal Recurrentes con problema de Optimización con Restricciones RENNCOM, formula la tarea de entrenamiento como un problema de optimización, donde la estabilidad es tratada como una funcional para ser optimizada.

La tarea de entrenamiento es formulada como un problema de optimización con restricciones cuyos objetivos son los siguientes: a) Minimizar la medida del error, obteniendo una aproximación buena de la entrada/salida. b) Optimización de una funcional adicional que ayuda asegurar la estabilidad de la red a través del proceso de aprendizaje.

3.5.1 Topología de la red

La topología de la red es Red Neuronal Recurrente Diagonal a Bloques BDRNN por sus siglas en inglés (Block-Diagonal *Recument* Neural Network) figura 3.9 .

La operación de la BDRNN con ***m*** entradas, ***r*** salidas y *N* neuronas en la capa oculta es descrita mediante las siguientes ecuaciones:

$$\begin{aligned} x(k) &= f_a(W \cdot x(k+1) + B \cdot u(k)) \\ y(k) &= f_b(C \cdot x(k)) \end{aligned} \tag{3.30}$$

Donde f_a , f_b son las funciones de activación de la neurona de las capas oculta y de salida respectivamente. En el siguiente las funciones de activación son escogidas como la sigmoide.

$u(k) = \left[u_i(k)\right]$ es vector de m elementos que son las entradas de la red al tiempo k

$x(k) = \left[x_i(k)\right]$ es vector de N elementos que son las salidas de la capa oculta.

$y(k) = \left[y_i(k)\right]$ es vector de r elementos que son la salida de la red en el tiempo k.

$B = \lfloor b_{i,j} \rfloor$ y $C = \lfloor c_{i,j} \rfloor$ son *N x m* y *r x N* pesos matriciales de las entradas y salidas respectivamente.

$W = \lfloor w_{i,j} \rfloor$ es la matriz de retroalimentación diagonal a bloques de bloque diagonal N x N. En particular

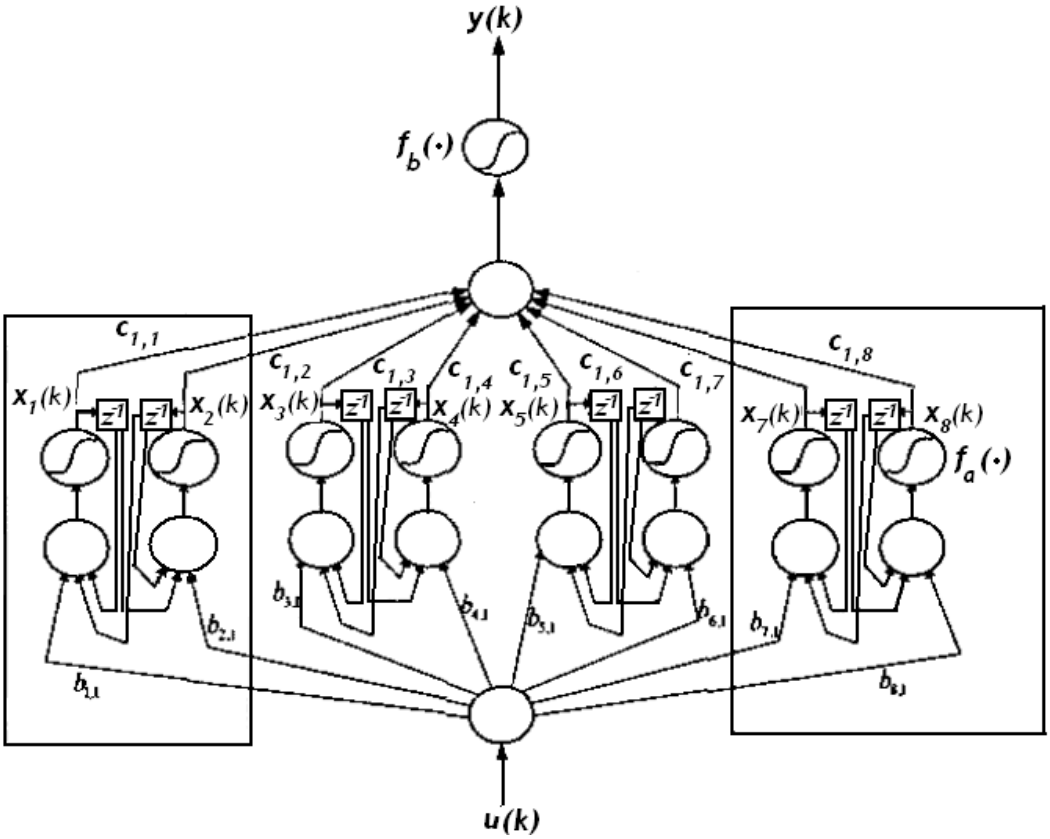


Figura 3.9 Configuración de BDRNN

$$w_{i,j} = \begin{cases} \neq 0 & si \text{ i} = \text{j} \\ \neq 0 & si \text{ i} \neq \text{j y i} = \text{j - 1 an i es impar} \\ \neq 0 & \text{si i} \neq \text{j y i} = \text{j} + 1 \text{ an i es par} \\ 0 & \text{de otra manera} \end{cases} \tag{3.31}$$

La matriz de retroalimentación W es diagonal a bloques: $W = diag\left[W^{(1)},...W^{(\frac{N}{2})} \right]$;

cada elemento de la diagonal, correspondiente al bloque de la red neuronal, tiene un submatriz de bloque de la forma:

$$W^{(i)} = \begin{bmatrix} w_{2i,2i} & w_{2i,2i+1} \\ -w_{2i,2i+1} & w_{2i,2i} \end{bmatrix} \; i = 1,2,..., \frac{N}{2} \tag{3.32}$$

La ecuación anterior describe el caso general de la topología BDRNN, que es llamada BDRNN de la forma libre de submatrices. Un caso especial de BDRNN consiste en las submatrices ortogonales escaladas de la forma:

$$W^{(i)} = \begin{bmatrix} w_{2i,2i} & w_{2i,2i+1} \\ -w_{2i,2i+1} & w_{2i,2i} \end{bmatrix} = \begin{bmatrix} w_i^{(1)} & w_i^{(2)} \\ -w_i^{(2)} & w_i^{(1)} \end{bmatrix} i = 1,2,..., \frac{N}{2} \tag{3.33}$$

De las ecuaciones (3.32) y (3.33), se vuelve evidente que la Forma-Libre de BDRNN consiste en las sub-matrices de retroalimentación con cuatro elementos distintos y da un gran grado de libertad comparado con la BDRNN ortogonal escalada, la cual tiene dos peso en cada una de las sub-matrices de retroalimentación. De las ecuaciones (3.25), BDRNN puede tomar la forma:

$$x_{2i-1}(k) = f_a(\sum_{j=1}^{m} b_{2i-1,j} \cdot u_j(k) + w_i^{(1)} \cdot x_{2i-1}(k+1) + w_i^{(2)} \cdot x_{2i}(k+1)) \tag{3.34}$$

$$x_{2i}(k) = f_a(\sum_{j=1}^{m} b_{2i,j} \cdot u_j(k) - w_i^{(2)} \cdot x_{2i-1}(k+1) + w_i^{(1)} \cdot x_{2i}(k+1) \tag{3.35}$$

$$y_l(k) = f_b(\sum_{j=1}^{m} c_{l,j} \cdot x_j(k)) \; l = 1,2,...r \tag{3.36}$$

Donde $w_i^{(1)}$, $w_i^{(2)}$ son los pesos de retroalimentación de la capa oculta.

Como se puede ver ambos ejemplos de redes neuronales recurrentes comparten algunas características con nuestra RNRE modular. Ambos ejemplos se aplican como identificadores de sistemas oscilatorios. En el capítulo 5 de esta tesis se presentaran los resultados de simulación comparando sus resultados con nuestra RNRE modular.

3.6 Teorema Universal de Aproximación.

Teorema 3.4 Sea $\varphi(\cdot)$ una función continua, no constante, acotada y monótona creciente. Sea I_{mo} un hipercubo unitario de dimensión m_0. El espacio de funciones continuas en I_{mo} es denotado por $C(I_{mo})$. Entonces dada cualquier función $f \in C(I_{mo})$ y $\varepsilon > 0$, existe un entero M y conjuntos de contantes reales α_i, β_i y w_{ij} donde $i = 1,\ldots,m_1$ y $j = 1,\ldots,m_0$ tal que definimos:

$$F(x_1,\ldots,x_{mo}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{i=1}^{m_0} w_{ij} x_j + \beta_i \right) \tag{3.37}$$

Con una realización aproximada de la función $f(\cdot)$; esto es:

$$\left| F(x_1,\ldots,x_{mo}) - f(x_1,\ldots,x_{mo}) \right| < \varepsilon \tag{3.36}$$

Para todo $x_1, x_2, \ldots, x_{mo}$ que yace en el espacio de entradas.

3.7 Conclusiones

En este capítulo se abordaron los conceptos básicos de las redes neuronales artificiales. Comenzamos con una descripción del modela matemático de la neurona artificial y la definición de sus elementos principales como son: el vector de entrada, los pesos sinápticos, el umbral, etc. También incluimos los distintos tipos de función de activación más comunes en la práctica. Se presentaron los elementos básicos que conforman a una red, las topologías existentes, así como las diferentes reglas y paradigmas de aprendizaje.

Se hablo de la capacidad de las redes neuronales para aprender de su entorno y mejora de su desempeño a través del proceso de aprendizaje del cual se hablará más a fondo en el capítulo 5.

En este capítulo se ha presentado una introducción a las redes neuronales recurrentes. Comenzamos hablando de la red de Hopfield, su topología y su mecanismo de aprendizaje par depuse citar al Teorema universal de aproximación. Se hablo de las ventajas que poseen las RNR sobre las redes feedforward cuando se trata de modelar la dinámica de una planta no lineal.

También se presento la topología de la RNRE modular la cual es una red hibrida compuesta de una capa oculta de neuronas dinámicas y una capa de salida de neuronas estáticas. Una de las principales propiedades de esta red es su estructura diagonal por bloques en la capa oculta. En la parte final del capítulo se presentaron dos topologías de redes neuronales recurrentes encontradas en la literatura que nos sirven de punto de comparación para evaluar a nuestra red.
.

Capitulo 4

Algoritmos de Aprendizaje.

4.1 Algoritmo Backpropagation

El algoritmo Backpropagation está basado en la regla delta, la cual será descrita más adelante en este capítulo. Su funcionamiento consiste en un aprendizaje de un conjunto predefinido de entradas salidas dados como ejemplo, empleando un ciclo de propagación – adaptación dividida en dos fases, [19]:

Primero se aplica un patrón de entrada a la primera capa de la red el cual va a ser propagado a través de todas las capas que conforman a la red hasta llagar a la capa de salida, donde se compara el resultado obtenido con el deseado, generándose una señal de error para cada neurona de salida.

Se transmite la señal de error hacia atrás, comenzando con la capa de salida, hacia todas las neuronas de las capas intermedias que contribuyen directamente con la salida. Recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original. Este proceso se repite capa por capa hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Usando este error relativo se ajustan los pesos sinápticos de cada neurona de tal manera que el error disminuya en el siguiente patrón de entrada.

4.1.1 Regla Delta Generalizada.

La regla delta se aplica a redes con capas ocultas con conexiones *feedforward* y cuyas neuronas tienen funciones de activación continuas, [4].

Supongamos una red de una sola capa con valor de salida:

$$y = \sum_{j=1}^{m} w_j x_j + b \tag{4.1}$$

Donde m es el número de neuronas en la capa de entrada. Sin pérdida de generalidad podemos suponer al umbral como un peso extra y reescribir la ecuación anterior como:

$$y = w^T z \tag{4.2}$$

Donde $w = [w_1, w_2, \cdots, w_m, b]^T$ y $z = [x_1, x_2, \cdots, x_m, 1]^T$.

El ajuste del vector de pesos se obtiene a través de minimizar el error medio cuadrático:

$$\min J = \frac{1}{2}\sum_{r=1}^{M}\left(y_r - d_r\right) \tag{4.3}$$

La regla delta utiliza la superficie de error asociada a la red, buscando en el estado estable de mínima ganancia o de mínimo error a través del camino descendente de la superficie del error. Realimentando el error del sistema para así realizar las modificaciones de los pesos en un valor proporcional al gradiente decreciente de la función de error. Además ajusta los pesos de forma proporcional a la delta, o diferencia entre la salida deseada y la obtenida.

$$\delta_r = d_r - y_r \tag{4.4}$$

Así dada una neurona i, y la salida que produce y_i; el cambio que se produce en el peso w_{ji} de la conexión que une a la salida de esa neurona con otra j para un patrón de aprendizaje p determinado es:

$$\Delta w_{ji}(k+1) = \alpha\delta_{pi}y_{pi} \tag{4.5}$$

Donde el subíndice p indica el patrón de aprendizaje y α es la tasa de aprendizaje. Sin embargo en las redes multicapas no se puede conocer la salida deseada de las neuronas de las capas ocultas para poder determinar los pesos en función del error cometido. Pero si podemos conocer la salida de las neuronas de salida. Podemos definir

$$\delta_{pj} = \left(d_{pj} - y_{pj}\right)f'\left(net_j\right) \tag{4.6}$$

Donde d_{pj} es la salida deseada de la neurona j para el patrón p y net_j es la entrada neta que recibe la neurona j.

Este término presenta la modificación que hay que realizar en la entrada que recibe la neurona j. En el caso de que dicha neurona no sea de salida, el error que se produce esta en función del error que se cometa en las neuronas que reciban como entrada la salida de dicha neurona; esto en si es lo que se conoce como procedimiento backpropagation.

En caso de que la neurona j no sea una neurona de salida, el error que se produce en las neuronas que reciben la salida de la neurona j es:

$$\delta_{pj} = \left(\sum_{k=1}^{n}\delta_{pk}w_{pk}\right)f'\left(net_j\right) \tag{4.7}$$

De esta manera, el error que se produce en una neurona oculta es la suma de los errores que se producen en las neuronas que la tienen de entrada, multiplicando cada uno de ellos por el peso de la conexión.

Al implementar el algoritmo anterior se toma una amplitud de paso que está definida por la tasa de aprendizaje α, entre mayor sea esta mayor es la modificación de los pesos y el aprendizaje es más rápido, pero esto puede dar lugar a oscilaciones [2], por esto se sugiere el uso de un término momento β el cual filtra dichas oscilaciones, de manera que el algoritmo queda:

$$w_{ji}(k+1) = w_{ij}(k) + \alpha\delta_{pi}y_{pj} + \beta\left(w_{ji}(k) - w_{ji}(k-1)\right) \quad (4.8a)$$

$$\Delta w_{ji}(k+1) = \alpha\delta_{pi}y_{pj} + \beta\Delta w_{ji}(k+1) \quad (4.8b)$$

En donde β es el término momento que determina el efecto que determina el cambio en k + 1 del cambio en el instante k. Con este término se consigue la convergencia de la red en menor número de iteraciones, ya que si en k el incremento de un peso era positivo y en k + 1 también, entonces el descenso por la superficie de error es mayor, sin embargo si en k el incremento es positivo y en k+1 es negativo, el paso será más pequeño, lo que significa que se ha pasado por un mínimo y que los pasos deben ser menores para poder alcanzarlo.

4.1.2 Computo del Algoritmo Backpropagation

Los pasos para aplicar el algoritmo de entrenamiento *backpropagation* son los siguientes:

- Inicializar los pesos de la red con valores pequeños aleatorios.
- Presentar un patrón de entrada, $x_{p1}, x_{p2}, \cdots, x_{pN}$ y especificar la salida deseada $d_1, d_2, \cdots, d_M$.
- Calcular la salida actual de la red, propagando la entrada por la red y calculando la salida en cada capa hasta llegar a la capa de salida, obteniendo la salida de la red $y_1, y_2, \cdots, y_M$.
- Se calculan las entradas netas de las neuronas ocultas procedentes de las neuronas de entrada:

$$net_{pi}^{h} = \sum_{j=1}^{N} w_{ji}x_{pj} + b_i^h \quad (4.9)$$

- En donde h se refiere a la capa oculta; el subíndice p, es el p-ésimo vector de entrenamiento, e i es la i-ésima neurona oculta. El término b es el umbral de la neurona i-ésima.
- Se calculan las salidas de las neuronas ocultas:
  - $y_{pi} = f_i^h\left(net_{pi}^h\right)$ (4.10)
- Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida.

- $$net_{pk}^{o} = \sum_{j=1}^{L} w_{kj} x_{pj} + b_k^o \quad (4.11)$$
- $$y_{pk} = f_k^o\left(net_{pk}^{o}\right) \quad (4.12)$$

- Calcular los términos de error para todas las neuronas
- Si la neurona es de la capa de salida:
  - $$\delta_{pk}^{o} = \left(d_{pk} - y_{pk}\right) f_k^{o\,\prime}\left(net_{pk}^{o}\right) \quad (4.13)$$
- La función f debe ser derivable. La selección de esta función depende de la forma en que se desea presentar los datos de salida.
- Si la neurona j no es de salida, entonces la derivada parcial del error no puede ser evaluada directamente, la expresión para este caso es:
  - $$\delta_{pi}^{h} = f_i^{h\,\prime}\left(net_{pi}^{h}\right) \sum_{k=1}^{N} \delta_k^o w_{kj}^o \quad (4.14)$$
- Donde el error en las capas ocultas depende de todos los términos de error de la capa de salida.
- Actualización de los pesos. Se utiliza un algoritmo recursivo comenzando por las ecuaciones de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la siguiente forma:
  - $$\Delta w_{ij}^{o}\left(k+1\right) = \alpha \delta_{pj}^{o} x_{pj} + \beta \Delta w_j^o\left(k+1\right) \quad (4.15)$$
  - $$\Delta w_{ji}^{h}\left(k+1\right) = \alpha \delta_{pj}^{h} x_{pj} + \beta \Delta w_{ji}^{h}\left(k+1\right) \quad (4.16)$$
- El proceso se repite hasta que el error medio cuadrático sea aceptablemente pequeño para cada uno de los patrones aprendidos:
  - $$J_p = \frac{1}{2} \sum_{r=1}^{M} \left(\delta_{pr}\right)^2 \quad (4.17)$$

4.2 Método Diagramático

Existe un método para poder deducir fácilmente los gradientes en redes neuronales muy complejas el cual es conocido como método diagramático. En este método se hace uso de la teoría de graficas de flujo para construir representaciones de las redes Neuronales. Con esto se pretende construir una red adjunta la cual directamente especifica la ecuación para poder deducir el gradiente, [20].

En el algoritmo Backpropagation se busca encontrar un conjunto de pesos que minimicen a la función de costo $J(w)$, es decir, encontrar $\partial J/\partial w_{ji}$ si tomamos en cuenta que $v(k)=\sum_i w_{ji}u_i(k)$, entonces:

$$\frac{\partial J}{\partial w_{ji}}=\frac{\partial J}{\partial v_j}\frac{\partial v_j}{\partial w_{ji}} \tag{4.18}$$

Definiendo

$$\delta_j =: \frac{\partial J}{\partial v_j} \tag{4.19}$$

$$G^N_{wji} =: \frac{\partial v_j}{\partial w_{ij}} \tag{4.20}$$

Donde G^N_{wji}, es el Jacobiano del vector v; entonces:

$$\frac{\partial J}{\partial w_{ji}}=\delta_j G^N_{wji} \tag{4.21}$$

$$\Delta w_{ji}=\eta\delta_j G^N_{wji} \tag{4.22}$$

Una red neuronal puede ser representada como un diagrama de bloques básicos de construcción a través de un operador suma, puntos de bifurcación, funciones univaluadas y multivaluadas y operadores de retardo en tiempo.

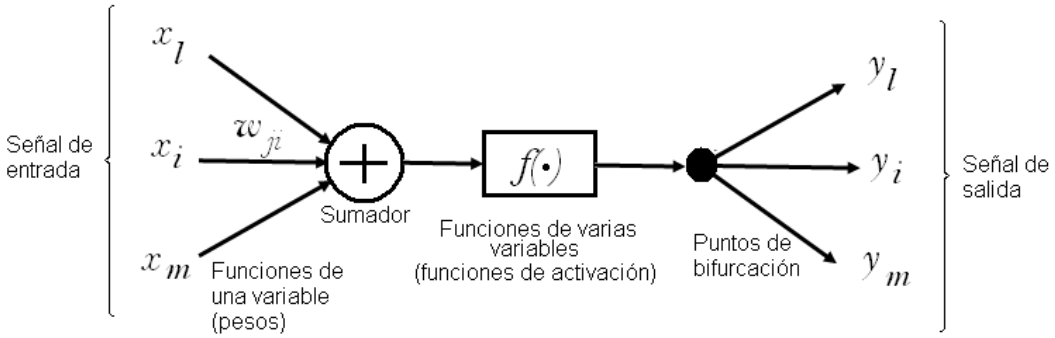


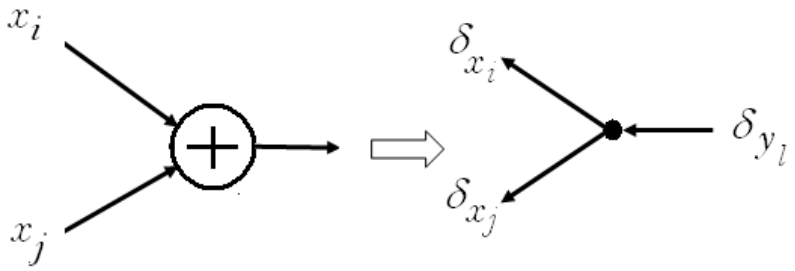
Figura 4.1 Representación en diagrama de bloques de una RN

4.2.1 Construcción de la Red Adjunta

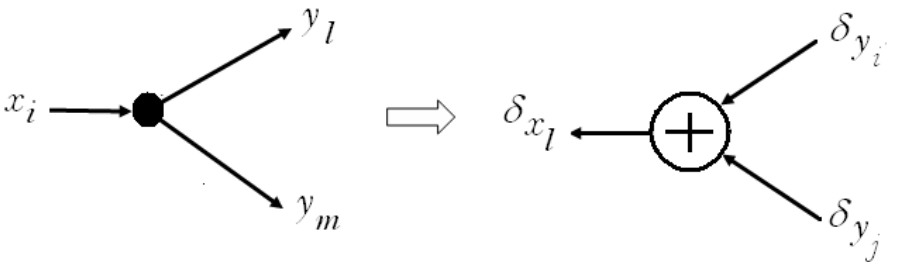
Teniendo una representación grafica de la Red Neuronal, se puede obtener su red adjunta aplicando las reglas que se enuncian a continuación y así poder determinar el gradiente local, ecuación 4.19.

Las reglas para la construcción de la red adjunta son las siguientes, [20]:

- Puntos suma son reemplazados por puntos de bifurcación.

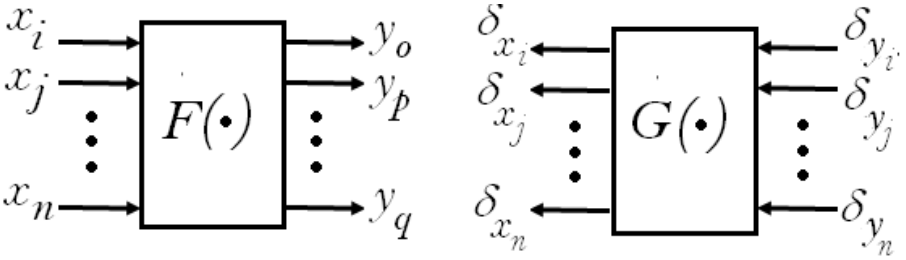


- Puntos de bifurcación son reemplazados por puntos suma.



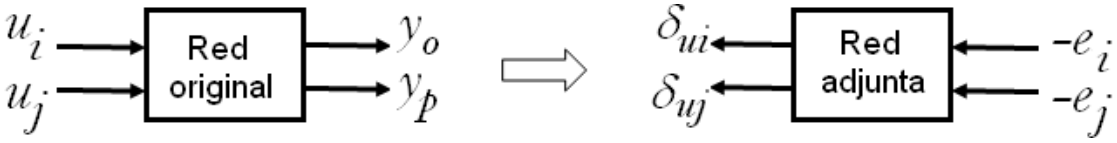
- Funciones de una variable son reemplazadas por sus derivadas.

- Funciones de varias variables son reemplazadas por sus Jacobianos.



- Operadores de retardo son reemplazados por operadores de avance.

- Las salidas de la red original se convierten en entradas de la red adjunta.



4.2.2 Deducción del algoritmo Backpropagation por el Método Diagramático.

En la figura 4.2 se muestra una neurona que pertenece a una capa oculta conectada a otras neuronas y una neurona de salida de una red multicapas. Sea v_i^l la activación sináptica en la i-ésima neurona de la l-ésima capa de la red, $1 \le l \le L$; L es el número total de capas. El término (k) se omite por simplicidad.

En la figura 4.2 Podemos ver la red adjunta construida a través de las reglas anteriores. Observando esta figura podemos ver rápidamente las ecuaciones para el cálculo del gradiente local δ_i^l , que son:

$$\delta_i^l = \begin{cases} -ef'\left(v_i^l\right) & l = L \\ f'\left(v_i^l\right)\sum_j \delta_j^{l+1} w_{ij}^{l+1} & 1 \le l \le L-1 \end{cases} \tag{4.23}$$

El cual corresponde con el término deducido matemáticamente para el algoritmo backpropagation.

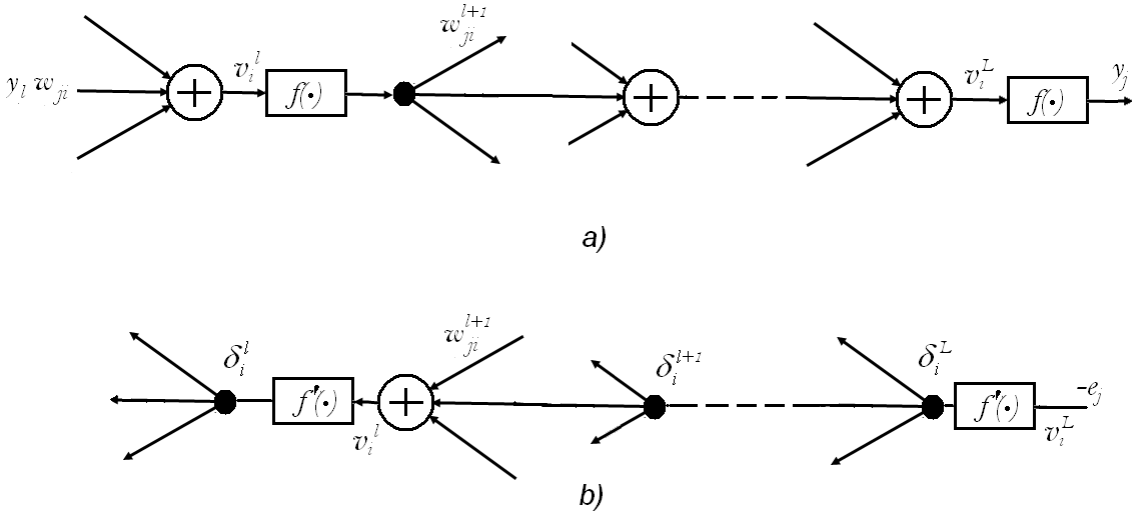


Figura 4.2. Diagrama representativo de un RN y el esquema de su red adjunta.

4.3 Aprendizaje supervisado enfocado al problema de optimización

Se entiende por optimización al proceso de encontrar los valores mínimos de una función $J(w)$ dada conocida como función de costo. Si expandimos a dicha función en series de Taylor alrededor del punto actual sobre la superficie del error $w(k)$ tenemos lo siguiente:

$$J\left(w(k)+\Delta w(k)\right) = J\left(w(k)\right) + \Delta w^T(k)\nabla J(w(k)) + \frac{1}{2}\Delta w^T(k)\nabla^2 J\left(\overline{w}\right)\Delta w(k) + (\text{terminos de orden superior}) \tag{4.24}$$

Donde $\nabla J(w(k))$ es el vector gradiente local definido como:

$$\nabla J(w(k)) = \left.\frac{\partial J(w)}{\partial w}\right|_{w=w(k)} \tag{4.25}$$

Además $\nabla^2 J(\overline{w})$ es la matriz Hessiana local, definida por:

$$\nabla^2 J(\overline{w}) = \left.\frac{\partial^2 J(w)}{\partial w^2}\right|_{w=w(k)} \tag{4.26}$$

El uso de un conjunto promedio de la función de costo se asume que es un aprendizaje fuera de línea.

En el método del gradiente descendente, el ajuste $\Delta w(k)$ es el siguiente

$$\Delta w(k) = -\eta \nabla J(w(k)) \tag{4.27}$$

Donde η es el parámetro de aprendizaje. El método del gradiente descendiente opera en base a una aproximación lineal de la función de costo en una vecindad del punto de operación $w(k)$. Entonces, el vector gradiente depende de $\nabla J(w(k))$ cómo la única fuente de información alrededor de la superficie de error. Esta restricción tiene como punto a favor que su implementación es muy simple, desafortunadamente, posee una tasa de convergencia lenta. Cuando se agrega un término momento en la actualización para el vector de pesos es un intento de usar información de segundo orden alrededor de la superficie de error, el cual ayuda pero el proceso de entrenamiento se vuelve más delicado teniendo que ajustar mas parámetros.

Para producir una mejora significante en el desempeño de la convergencia, se tiene que usar información de orden más alto en el proceso de entrenamiento. Se puede lograr usando una aproximación cuadrática de la superficie del error alrededor del punto $w(k)$. De la ecuación 4.24 vemos que el valor óptimo de ajuste $\Delta w(k)$ aplicado al vector de actualización de pesos es

$$\Delta W^*(k) = \left[\nabla^2 J\left(\overline{w}\right)\right]^{-1} \nabla J(w(k)) \tag{4.28}$$

Donde $\left[\nabla^2 J(\overline{w})\right]^{-1}$ es la inversa de la matriz Hessiana $\nabla^2 J(\overline{w})$ asumiendo que existe. La ecuación 5.28 es la esencia del método de Newton. Si la función de costo es cuadrática, el método de Newton converge a la solución optima en una iteración, [2]. Sin embargo, la aplicación del método de Newton al entrenamiento supervisado es impedido por:

Requiere el cálculo de la matriz Hessiana inversa la cual puede llegar a ser computacionalmente costosa.

Para que $\left[\nabla^2 J(\overline{w})\right]^{-1}$ sea calculable, la matriz $\nabla^2 J(\overline{w})$ debe ser no singular.

Cuando la función de costo $J(w)$ es no cuadrático, no hay garantía de convergencia del método de Newton el cual la vuelve inadecuado para entrenamiento del perceptrón multicapas.

Para sobrellevar algunas de estas dificultades, se puede recurrir al uso del método de cuasi-Newton el cual solo requiere de un estimado del vector gradiente $\nabla J(w(k))$.

Esta modificación del método de cuasi-Newton mantiene positiva definida la estimada de la matriz inversa $\left[\nabla^2 J(\overline{w})\right]^{-1}$ sin tener que invertir la matriz. Usando la estimada, el método de cuasi-Newton asegura el decremento sobre la superficie del error.

De cualquier forma, se tiene un problema computacional el cual es $O(W^2)$, en donde W es el tamaño del vector de pesos. Los métodos de cuasi-Newton son sin embargo computacionalmente imprácticos, excepto para el entrenamiento de pequeñas redes neuronales.

4.3.1 Métodos cuasi-Newton.

Los métodos cuasi-Newton son básicamente métodos de gradiente descritos como sigue:

$$w(k+1) = w(k) + \eta(k)s(k) \tag{4.29}$$

Donde el vector de dirección $s(k)$ es definida positiva en términos del vector gradiente $\nabla J(w(k))$ por

$$s(k) = -S(k)\nabla J(w(k)) \tag{4.30}$$

La matriz $S(k)$ es una matriz definida positiva que es ajustada en cada iteración. Esto se hace para tener un vector de dirección $s(k)$ aproximado a la dirección de Newton.

$$s(k) = -\left(\frac{\partial^2 J}{\partial w^2}\right)^{-1}\left(\frac{\partial J}{\partial w}\right) \tag{4.31}$$

Los métodos de cuasi-Newton usan información de segundo orden (curvatura) alrededor de la superficie de error sin requerir conocimiento de la matriz Hessiana *H*. Eso se hace usando dos iteraciones sucesivas $w(k)$ y $w(k+1)$, juntas con su respectivo vector de gradientes $\nabla J(w(k))$ y $\nabla J(w(k+1))$. Sean

$$q(k) = \nabla J(w(k+1)) - \nabla J(w(k)) \tag{4.32}$$

$$\Delta w(k) = w(k+1) - w(k) \tag{4.33}$$

Podemos derivar la curvatura usando la formula.

$$q(k) \approx \left(\frac{\partial}{\partial w} \nabla J(w(k)) \right) \Delta w(k) \tag{4.34}$$

En particular, dados los incrementos de pesos W linealmente independientes $\Delta w(0),\ \Delta w(1),\ \ldots, \Delta w(W-1)$ y su respectivo incremento de gradientes $q(0),\ q(1)$ $,\ldots, q(W-1)$, se puede aproximar la matriz Hessiana $\nabla^2 J(\bar{w})$, [2] como

$$H \approx [q(0),\ \ q(1),\ \ \ldots\ \ q(W-1)][\Delta w(0),\ \ \Delta w(1),\ \ \ldots\ \ \Delta w(W-1)]^{-1} \tag{4.35}$$

También podemos aproximar la inversa de la matriz Hessiana como:

$$[\nabla^2 J(\bar{w})]^{-1} \approx [\Delta w(0),\ \ \Delta w(1),\ \ \ldots\ \ \Delta w(W-1)][\Delta q(0),\ \ \Delta q(1),\ \ \ldots\ \ \Delta q(W-1)]^{-1} \tag{4.36}$$

Cuando la función de costo es cuadrática, las dos ecuaciones anteriores son exactas.

4.4 Algoritmo Levenberg-Marquadt fuera de línea.

Es un método de segundo orden el cual es una variación del método de Newton que fue diseñado para minimizar funciones que son sumas de cuadrados de otras funciones no lineales. Este es un entrenamiento bien preparado donde el índice de desempeño es el error medio cuadrático, [46-48].

Mientras que el *Backpropagation* es un algoritmo de gradiente descendiente, el algoritmo Levenberg-Maquardt es una aproximación al método de Newton. Suponga que tenemos una función $J(\bar{w})$ tal que queremos minimizar con respecto al vector de parámetros. Entonces el método de Newton será:

$$\Delta \bar{w} = -[\nabla^2 J(\bar{w})]^{-1} \nabla J(\bar{w}) \tag{4.37}$$

Donde $\nabla^2 J(\bar{w})$ es la matriz Hessiana y $\nabla J(\bar{w})$ es el gradiente. Si asumimos que es una suma de funciones cuadráticas

$$J(\bar{w}) = \sum_{i=0}^{N} e_i^2(\bar{w}) \tag{4.38}$$

Se puede ver qué

$$\begin{aligned} \nabla J(\bar{w}) &= G^T(\bar{w}) e(\bar{w}) \\ \nabla^2 J(\bar{w}) &= G^T(\bar{w}) G(\bar{w}) + S(\bar{w}) \end{aligned} \tag{4.39}$$

Donde $G(\bar{w})$ es la matriz jacobiana

$$G(\bar{w})\begin{bmatrix} \frac{\partial e_{1(w)}}{\partial w_1} & \frac{\partial e_{1(w)}}{\partial w_2} & \cdots & \frac{\partial e_{1(w)}}{\partial w_n} \\ \frac{\partial e_{2(w)}}{\partial w_1} & \frac{\partial e_{2(w)}}{\partial w_2} & \cdots & \frac{\partial e_{2(w)}}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_{N(w)}}{\partial w_1} & \frac{\partial e_{N(w)}}{\partial w_2} & \cdots & \frac{\partial e_{N(w)}}{\partial w_n} \end{bmatrix} \tag{4.40}$$

$$S(\bar{w}) = \sum_{i=0}^{N} e_i\,(\bar{w})\nabla^2 e_i\,(\bar{w}) \tag{4.41}$$

Para el método Gauss-Newton se asume que $S(\bar{w}) \approx 0$ por tanto la actualización es

$$\Delta\bar{w} = \left[G^T(\bar{w})G(\bar{w})\right]^{-1} G^T(\bar{w})e(\bar{w}) \tag{4.42}$$

La modificación del método Gauss-Newton al método Levenberg-Marquardt es:

$$\Delta\bar{w} = \left[G^T(\bar{w})G(\bar{w}) + \mu I\right]^{-1} G^T(\bar{w})e(\bar{w}) \tag{4.43}$$

El parámetro μ es multiplicado por un factor (β) cuando el factor $J(\bar{w})$ se incrementa en el paso siguiente. Si en el siguiente paso $J(\bar{w})$ se reduce, entonces es dividido por β. Nótese que cuando μ es grande, el algoritmo llega a ser de paso descendiente, si μ es pequeño será Gauss-Newton.

La clave de este algoritmo radica en el cálculo de la matriz Jacobiana, En este caso la matriz Jacobiana puede ser calculada por una simple modificación del algoritmo *backpropagation*. El índice de desempeño está dada por la ecuación 4.38. El algoritmo de *backpropagation* estándar calcula términos como:

$$\frac{\partial \hat{J}}{\partial w_{ij}} = \frac{\partial \sum_{m=1} e_q^2(m)}{\partial w_{ij}} \tag{4.44}$$

Se necesitan calcular los elementos de la matriz Jacobiana para el algoritmo Levenberg-Maquardt cuyos términos son de la forma:

$$\frac{\partial e_q(m)}{\partial w_{ij}} \tag{4.45}$$

Estos términos pueden ser calculados usando el algoritmo *backpropagation* estándar con una modificación en la capa final.

$$\Delta^M = -\dot{F}^M(\bar{v}^M) \tag{4.46}$$

Se puede ver que cada columna de la matriz en la ecuación anterior es un vector gradiente el cual debe ser retropropagado a través de la red para producir una fila en el jacobiano.

Resumen del algoritmo

Para la aplicación del algoritmo Levenberg-Maquardt se procede como sigue:

Presentar el vector de entradas a la red y calcular su correspondiente salida usando $e_j(k) = d_j(k) - y_j(k)$, $J(\bar{w}) = \frac{1}{2}\sum_{j \in C} e_i^2(\bar{w})$ y el error $\bar{e}_k = \bar{y}d_k - \bar{y}_j{}^M$. Calcular la suma de los errores cuadráticos $J(\bar{w})$

Calcular la matriz Jacobiana usando 4.19, 4.46 y 4.40

Resolver 4.43 para obtener $\Delta\bar{w}$

Recalcular la suma de los errores cuadráticos usando $\bar{w} + \Delta\bar{w}$. Si la nueva suma de los errores cuadráticos es más pequeña que la calculada en el paso 1, entonces reducir μ por β, y dejar $\bar{w} = \bar{w} + \Delta\bar{w}$, e ir al paso 1. Si la suma no se reduce, entonces incrementar μ por β e ir al paso 3.

Se asume que el algoritmo ha convenido cuando la norma del gradiente 4.39 es menor que algún valor predeterminado o cuando la suma de los cuadrados ha sido reducido por algún error predeterminado.

4.5 Algoritmo de Entrenamiento Recursivo Levenberg-Maquardt

El entrenamiento de redes neuronales se define normalmente como una minimización de la función de costo del error cuadrático con respecto a los parámetros de la red. Esta minimización puede ser ejecutada en el modo fuera de línea o modo en línea (recursivo).

El modo fuera de línea, la función de costo se define explícitamente sobre un lote de datos de entrenamiento y es minimizando usando procedimientos de optimización del gradiente descendiente.

En el modo de entrenamiento recursivo, los pesos de la red son actualizados con cada iteración en base a una estimación de la función de costo instantáneo usando los datos de entrenamiento actúales, [39], [40].

En este trabajo se propone un algoritmo recursivo que está basado en el método Levenberg-Marquadt para entrenar redes neuronales no lineales.

4.5.1 Entrenamiento Recursivo

En el entrenamiento recursivo de redes neuronales [41] se intenta minimizar la función de costo instantáneo estimado como

$$E(w(k)) \approx e^2(w(k)) = (y_d(k) - (y(w, x(k)))^2 \tag{4.47}$$

Donde $w(k)$ es el vector de pesos de la red y $(x(k), y_d(k))$ es el dato puntual de entrenamiento en el *k*-ésimo instante. En el *backpropagation* o gradiente descendiente los pesos son actualizados de acuerdo con la regla:

$$w(k+1) = w(k) - \lambda(k) e^2(w(k)) \tag{4.48}$$

Donde $\lambda(k)$ es el parámetro de aprendizaje el cual puede ser constante o variable en el tiempo.

4.5.2 Error de Predicción Recursivo (EPR)

El algoritmo recursivo de mínimos cuadrados de segundo orden el cual ha sido extensamente estudiando para modelos de parámetros lineales se puede extender a modelos no lineales linealizándolos con respecto a los pesos estimados actuales $w(k)$. El algoritmo resultante puede ser visto como una equivalencia de la optimización instantánea Gauss- Newton con una aproximación de una matriz Hessina recursiva para cada iteración.

$$R(k) = \alpha R(k-1) + (1-\alpha)(\nabla y(w(k)) \nabla y^T(w(k)) \tag{4.49}$$

Con $0.95 \le \alpha \le 1$

$$\nabla y(w(k)) = \frac{\partial}{\partial w} y(w, x(k)) \Big|_{w=w_k} \tag{4.50}$$

Donde $y(w, x(k))$ es la salida de la red neuronal.

El vector gradiente, $\nabla y(w(k))$ es calculado de la siguiente forma. Primero definimos al término retropropagado

$$G_j^n = \frac{\partial y(.,.)}{\partial v_j^n} \tag{4.51}$$

Donde v_j está definida en (4.2). Haciendo uso de la regla de la cadena, el gradiente para los pesos es:

$$\frac{\partial y(.,.)}{\partial w_{ji}^n} = \frac{\partial y(.,.)}{\partial v_j^n}\frac{\partial v_j^n}{\partial w_{ji}^n} = G_j^n y_{in-1} \tag{4.52}$$

Los términos del umbral son

$$\frac{\partial y(.,.)}{\partial b_j^n} = \frac{\partial y(.,.)}{\partial v_j^n}\frac{\partial v_j^n}{\partial b_j^n} = G_j^m \tag{4.53}$$

Donde y_i^{n-1} es la relación entrada salida de la neurona (i) en la capa $(n-1)$ y es definida como:

$$y_j^n = f^n(v_j^n) \tag{4.54}$$

En la capa de salida M (4.52) y (4.53) son

$$\frac{\partial y(.,.)}{\partial w_{ji}^M} = G_j^M y_i^{M-1} \tag{4.55}$$

$$\frac{\partial y(.,.)}{\partial b_j^n} = G_j^M \tag{4.56}$$

Donde $i = 1$ en la capa de salida M representa un caso SISO.

En la capa oculta, es decir $1 \le n \le M-1$ (4.51) puede ser reescrita usando la regla de la cadena como

$$G_j^n = \left(\sum_{l=1}^{N_{n+1}} \frac{\partial y(.,.)}{\partial v_l^{n+1}}\frac{\partial v_l^{n+1}}{\partial y_l^n}\right)\frac{\partial y_j^n}{\partial v_l^{n+1}} \tag{4.57}$$

Se puede ver qué (4.57) es igual a

$$G_j^n = \left(\sum_{l=1}^{N_{n+1}} G_l^{n+1} w_{l,j}^{n+1}\right)\frac{\partial f^n\left(v_j^n\right)}{\partial v_l^{n+1}} \tag{4.58}$$

Ahora podemos calcular iterativamente el término G_j^n, empezando de G_j^M en la capa de salida. La actualización recursiva de los pesos de segundo orden es entonces:

$$w(k+1) = w(k) + R^{-1}\nabla y\left(w(k)\right)e\left(w(k)\right) \tag{4.59}$$

La inversa de *R* en cada iteración es computacionalmente costosa y normalmente el lema de inversión de matrices (4.60) se usa para obtener de forma recursiva la inversa de *R* directamente.

$$(A+B)^{-1} = A^{-1} - A^{-1}B(I + CA^{-1}B)^{-1}CA^{-1} \tag{4.60}$$

El resultado de la regla para la actualización de los pesos usualmente se refiere como un algoritmo de error de predicción recursivo (EPR).

$$P(k) = \frac{1}{\alpha}\left[P(k-1) - \frac{P(k-1)\nabla y(w(k))\nabla y^T(w(k))P(k-1)}{\alpha + \nabla y^T(w(k))P(k-1)\nabla y(w(k))} \right] \tag{4.61}$$

$$w(k+1) = w(k) + P_k \nabla y(w(k))e(w(k)) \tag{4.62}$$

La matriz $P(k) = R^{-1}(k)$ puede ser interpretada como la matriz de covarianza de los pesos estimados $w(k)$.

Computo del Algoritmo Recursivo L-M

El algoritmo recursivo Levenberg-Marquardt se deduce por la incorporación de un término de regularización en la ecuación (4.49) del Algoritmo EPR, [41-45].

$$R(k) = \alpha R(k-1) + (1-\alpha)(\nabla y(w(k))\nabla y^T(w(k) + \rho I_{N_w}) \tag{4.63}$$

Desafortunadamente, la ecuación (4.63) es incompleta para poder aplicar el lema de inversión de matrices, pero una solución efectiva es agregar una ρ a uno de los elementos de la diagonal de $\nabla y(w(k))\nabla y^T(w(k))$, la ecuación (4.63) puede ser expresada como:

$$R(k) = \alpha R(k-1) + (1-\alpha)(\nabla y(w(k))\nabla y^T(w(k) + \rho Z_{N_w}) \tag{4.64}$$

Donde Z_{N_w} es una matriz diagonal N_w x N_w con un elemento no cero en la diagonal, el cual cambia de posición de iteración en iteración como sigue:

$z_{ii} = 1$ cuando $i = k \bmod(N_w) + 1$ y $k > N_w$ (4.65)

$z_{ii} = 0$ de otra forma. (4.66)

Con esta modificación la expresión (4.63) puede rescribirse de la siguiente forma

$$R(k) = \alpha R(k-1) + (1-\alpha)(\Omega(w(k)))\Lambda^T(k)\Omega^T(w(k)) \tag{4.67}$$

Donde $\Omega(w(k))$ es una matriz de N_w x 2 con la primera columna correspondiente a $\nabla y(w(k))$ y la segunda columna consiste de un vector de N_w x 1 con un elemento igual a 1 de acuerdo con las ecuaciones (4.65) y (4.67)

$$\Omega^T(w(k)) = \begin{bmatrix} & \nabla y^T(w(k)) & \\ 0 \quad \cdots & 1 & \cdots \quad 0 \end{bmatrix} \tag{4.68}$$

$$\Lambda^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix} \tag{4.69}$$

El lema de inversión de matrices puede ahora ser aplicado a (4.67) llevando la siguiente formulación recursiva de Levenberg-Marquardt

$$S(w(k)) = \alpha\Lambda(k) + \Omega^T(w(k))P(k-1)\Omega(w(k)) \tag{4.70}$$

$$P = \frac{1}{\alpha}\left[P(k-1) - P(k-1)\Omega(w(k))S^{-1}(w(k))\Omega^T(w(k))P(k-1)\right] \tag{4.71}$$

$$w(k+1) = w(k) + P(k)\nabla y(w(k))e(w(k)) \tag{4.72}$$

Ahora la actualización de los pesos requiere solamente de la inversión de una matriz $S(w(k))$ de 2 x 2. Esto es mucho más efectivo que tener que invertir la matriz N_w x N_w.

4.6 Aprendizaje de la red RNRE modular

4.6.1 Algoritmo Backpropagation para la RNRE modular

El algoritmo de aprendizaje back-propagation esta dado por:

$$W(k+1) = W(k) + \eta\Delta W(k) + \alpha\Delta W(k-1) \tag{4.73}$$

En donde $W(k)$ es la matriz de pesos que será actualizada (A, B, C); $\Delta W(k)$ es la matriz de corrección de los pesos $(\Delta A, \Delta B, \Delta C)$; η es el factor de aprendizaje $(|\eta| < 1)$, y α es el término momento, el cual es usado para eliminar oscilaciones en el error durante el aprendizaje.

La representación grafica de la adjunta de la RNRE modular de la figura (3.3), se ve en la figura (4.3), en donde se representan como dos redes en paralelo. Las dos partes que conforman a una RNRE modular, siendo la matriz de pesos de la capa oculta de la primera parte una matriz diagonal, mientras que el segundo modulo posee una matriz de pesos completa de tamaño 2 x 2.

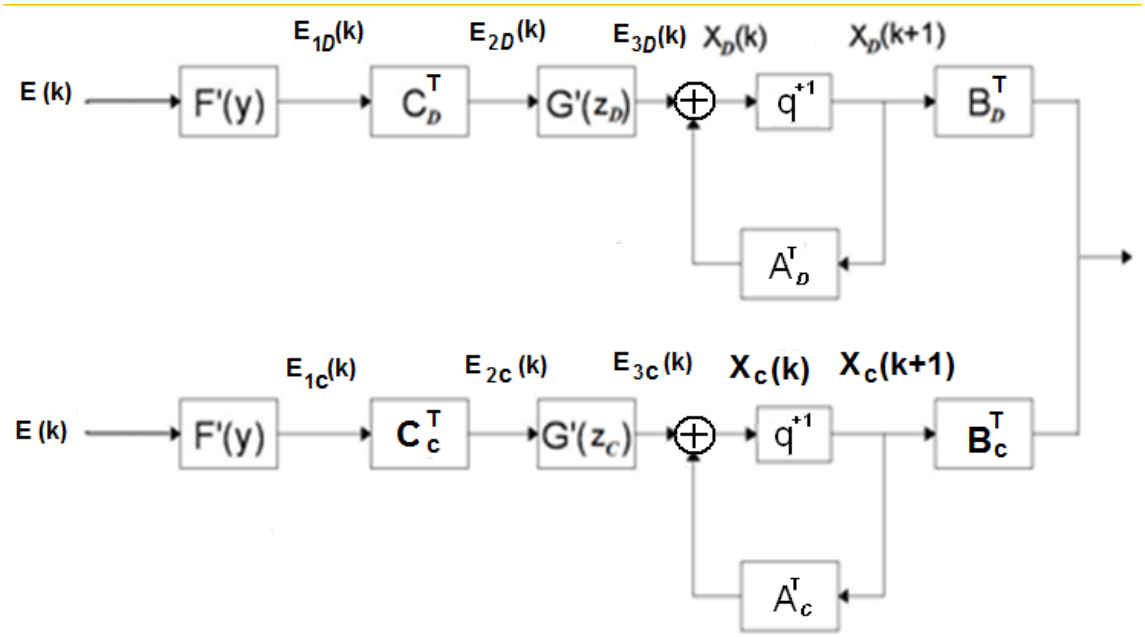


Figura 4.3 Red Adjunta de la RNRE modular

Una vez que hemos construido la red adjunta de la red, podemos obtener las ecuaciones de los pesos. La actualización de los elementos de la matriz de la capa de salida para el modelo en tiempo discreto de la RNRE modular viene dada por las siguientes ecuaciones

$$E(k) = y_p(k) - y(k); \tag{4.74}$$

$$\Delta C_D(k) = E_{1D}(k) Z^T(k); \tag{4.75}$$

$$\Delta C_C(k) = E_{1C}(k) Z_C{}^T(k); \tag{4.76}$$

$$E_{2D}(k) = C_D{}^T(k) E_{1D}(k); \tag{4.77}$$

$$E_{2C}(k) = C_C{}^T(k) E_{1C}(k); \tag{4.78}$$

$$E_{3D}(k) = G'[Z(k)] X_D{}^T(k); \tag{4.79}$$

$$E_{3C}(k) = G'[Z_C(k)] X_C{}^T(k); \tag{4.80}$$

$$\Delta B_D(k) = E_{3D}(k) U^T(k); \tag{4.81}$$

$$\Delta B_C(k) = E_{3C}(k)U^T(k); \tag{4.82}$$

$$\Delta A_D(k) = E_{3D}(k)X_D{}^T(k); \tag{4.83}$$

$$\Delta A_C(k) = E_{3C}(k)X_C{}^T(k); \tag{4.84}$$

En donde el vector $E(k)$ es el error entre la salida deseada y la salida de la red en el tiempo k, $\Delta C(k)$, $\Delta B(k)$ y $\Delta A(k)$ son las correcciones de las matrices de pesos del bloque diagonal de la RNRE modular. Para la ecuación (4.83) la cual es la parte diagonal de la red, $A = diag[A_{ii}]; \quad i = 1,\ldots,n; |A_{ii}| < 1$, (condición impuesta de estabilidad) y la diagonal de la matriz es un $(n\ x\ 1)$ vector, denotado por vA, entonces la actualización de pesos de este vector es la siguiente:

$$\Delta vA = E_3(k) \oplus x(k) \tag{4.85}$$

4.6.2 Algoritmo Levenberg-Marquardt para la RNRE modular

Otro algoritmo con el cual la RNRE modular puede ser entrenada es el algoritmo Levenberg-Marquardt dado por la siguiente ecuación en forma matricial:

$$W(k+1) = W(k) + P(k)\nabla y(w(k))E(w(k)) \tag{4.86}$$

Donde $W(k)$ es la matriz de pesos, que será modificada (A,B,C); $P(k)$ puede ser interpretada como la matriz de covarianza de los pesos estimados, $\nabla y(w(k))$ es un vector de (N_w x 1) donde $N_w = \dim(w)$, $E(w(k))$ es el vector del error.

Para la construcción del vector $\nabla y(w(k))$ se usan las mismas técnicas utilizadas para la retropropagación del error.

$$\nabla y(w(k)) = \frac{\partial y(w,x)}{\partial w} \tag{4.87}$$

Como en el caso del BP, los elementos del vector $\nabla y(w(k))$ pueden ser obtenidos siguiendo el diagrama a bloques de la red adjunta de la RNRE modular dada en la figura (4.4) construida utilizando el método diagramático. Aquí el vector error es nuevamente definido como en la ecuación (4.74).

Los elementos del vector $\nabla y(w(k))$ para la capa de salida para el modelo en tiempo discreto de la RNRE modular esta dado por la siguiente ecuación:

$$\frac{\partial}{\partial C_{i,j}} y(w,x) = D1_i(k) z(k) \tag{4.88}$$

Donde $$D1_i(k) = F_i'(y_i(k)) \tag{4.89}$$

$\frac{\partial}{\partial C_{ij}} y(w,x)$ son los elementos del vector $\nabla y((w(k))$ correspondiente al *ij*-ésimo elemento de la matriz de aprendizaje C; $F_i'(y_i(k))$ es el *i*-ésimo elemento del vector de la primera derivada de la función de activación, el cual es expresado como función del vector de salida $y(k)$; $z_j(k)$ es el *j*-ésimo elemento del vector de salida de la capa oculta; $D1_i(k)$ es el *i*-ésimo elemento del vector de salida $D1$.

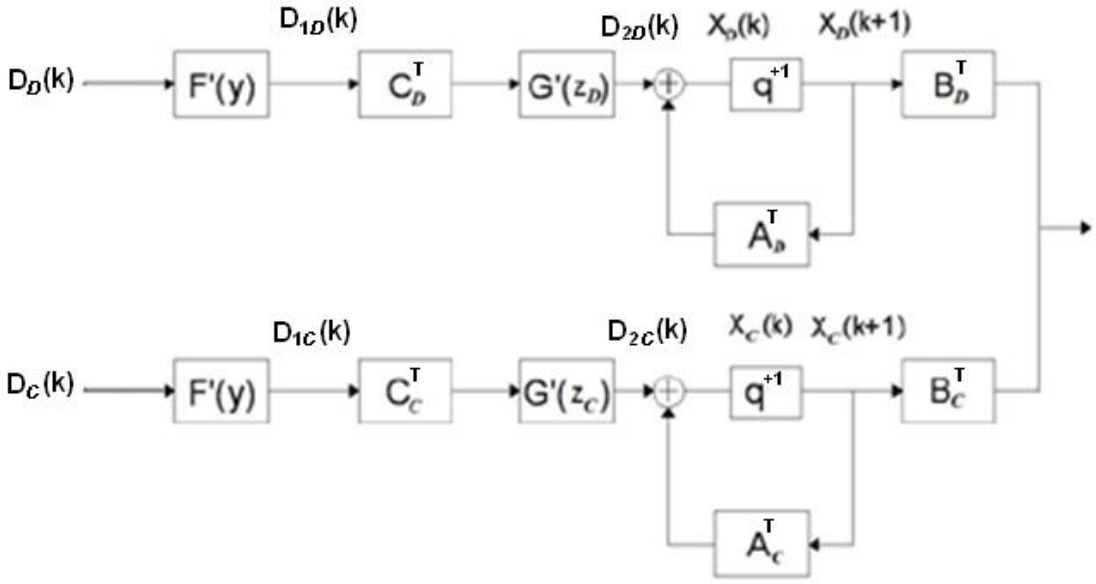


Figura 4.4 Red Adjunta de la RNRE modular para el Algoritmo L-M.

Los elementos del vector $\nabla y(w,x)$ para la parte de la capa oculta con estructura diagonal, es realizado de la siguiente manera:

$$\frac{\partial}{\partial A_{ij}} y(w,x) = D2_i(k) x_j(k) \tag{4.90}$$

$$\frac{\partial}{\partial B_{ij}} y(w,x) = D2_i(k) u_j(k) \tag{4.91}$$

Donde $$D2_i(k) = G'_i(z_i(k)) C_i D1_i(k) \tag{4.92}$$

Para las neuronas pertenecientes a la capa oculta que poseen retroalimentación entre ellas, y forman bloques de tamaño 2×2 en la matriz de pesos de la capa

oculta, los elementos del vector $\nabla y(w,x)$ vienen dados por las siguientes expresiones:

$$\frac{\partial}{\partial A_{Cij}} y(w,x) = D2_{Ci}(k) x_{Cj}(k) \tag{4.93}$$

$$\frac{\partial}{\partial B_{Cij}} y(w,x) = D2_{Ci}(k) u_j(k) \tag{4.94}$$

Donde $$D2_{Ci}(k) = G'_i\left(z_{Ci}(k)\right) C_{iC} D1_i(k) \tag{4.95}$$

El lado izquierdo de las (4.90) y (4.93) son los elementos del vector $\nabla y(w,x)$ correspondientes al *ij*-ésimo elemento de las matrices A y A_C. C_i y C_{Ci} son los *i*-ésimos componentes del vector renglón tomados de C^T; $D2_{Ci}(k)$ es el *i*-ésimo elemento del vector de salida $D2_C(k)$ y $D2_i(k)$ es el *i*-ésimo elemento del vector de salida $D2(k)$; $x_{Cj}(k)$ y $x_j(k)$ son los j-ésimos elementos de los vectores de estado $x_C(k)$ y $x(k)$.

Por otro lado, las ecuaciones (4.91) y (4.94) son los elementos del vector $\nabla y(w,x)$ correspondientes al *ij*-ésimo elemento de las matrices B y B_c; $u_j(k)$ es le *i*-ésimo elemento del vector de entrada $u(k)$.

El cálculo de la matriz $P(k)$ de (N_w x N_w) se realiza de la siguiente forma:

$$P(k) = \frac{1}{\alpha}\left[P(k-1) - P(k-1)\Omega(w(k))S^{-1}(w(k))\Omega^T(w(k))P(k-1)\right] \tag{4.96}$$

$$S(w(k)) = \alpha\Lambda(k) + \Omega^T(w(k))P(k-1)\Omega(w(k)) \tag{4.97}$$

Con 0.95 $\geq \alpha \geq$ 1, 1 x $10^3 \leq P(0) \leq$ 1 x 10^6, 1 x $10^{-4} \leq P(0) \leq$ 1 x 10^{-6} para las ecuaciones (4.68) y (4.69)

$$\Omega^T(w(k)) = \begin{bmatrix} & \nabla y^T(w(k)) & \\ 0 \quad \cdots & 1 & \cdots \quad 0 \end{bmatrix} \tag{4.68}$$

$$\Lambda^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix} \tag{4.69}$$

Donde $\Omega^T(w(k))$ es una matriz de tamaño (N_w x 2) con la primera columna correspondiente a $\nabla y(w,x)$ y la segunda columna consiste de un vector de (N_w x1) con un elemento igual a 1 cuya posición se define de acuerdo con las siguientes reglas

$z_{ii} = 1$ cuando $i = k$ mod (N_w) + 1 y $k > N_w$ (4.98)

$z_{ii} = 0$ de otra forma.

4.7 Conclusiones.

En este capítulo se trabajo sobre distintos tipos de algoritmos de aprendizaje, tales como el Backpropagation, el algoritmo Levenberg-Marquardt recursivo y fuera de línea. Se presentó el método diagramático y su importancia para construir la red adjunta, la cual nos permite obtener de manera rápida el algoritmo de aprendizaje. Se mostró la efectividad del método diagramático al deducir el algoritmo backpropagation a través de las reglas de construcción de la red adjunta.

Capitulo 5

Identificación

5.1 Introducción

La identificación de la planta es generalmente un proceso fundamental previo al control de la misma ya que permite obtener información sobre la dinámica de la planta. El objetivo de la identificación es conocer los estados y parámetros de la estructura de una planta, así como obtener un modelo que tenga un comportamiento aproximado de la misma.

El uso de las redes neuronales para la identificación de sistemas se ha extendido debido a que permiten aproximar una gran variedad de clases de funciones no lineales con suficiente precisión, haciendo de ellas las principales candidatas para utilizarse en modelos dinámicos para la representación de modelos no lineales.

La caracterización e identificación son problemas fundamentales en la teoría de sistemas. El problema de caracterización consiste en la representación matemática de un sistema; el modelo de un sistema es expresado como un operador *P* de un espacio de entrada *U* a un espacio de salida *Y*, y el objetivo es caracterizar la clase *P* a la cual *P* pertenece. Dada la clase *P* y el hecho de que *P* $\in$*P*, el problema de identificación es determinar la clase $\hat{\mathcal{P}} \subset \mathcal{P}$ tal que $\hat{\mathcal{P}}$ aproxime a *P* en algún sentido deseado.

El problema de identificación de patrones es un problema típico de identificación de sistemas estáticos. En estos problemas, conjuntos compactos $U_i \subset R^n$ son mapeados a elementos $y_i \subset R^m$; $(i = 1,2,...,)$ en el espacio de salida por una función de decisión *P*. los elementos de U_i denotan los vectores de patrones correspondientes a la clase y_i. En sistemas dinámicos, el operador P, que representa a una planta dada, es definido implícitamente por el par de entrada salida de las funciones del tiempo $u(t),\ y(t),\ t \in [0,T)$. En ambos casos el objetivo es determinar $\hat{\mathcal{P}}$ tal que:

$$\|y - \hat{y}\| = \left\|P(u) - \hat{P}(u)\right\| \leq \varepsilon, \quad u \in U \tag{5.1}$$

Para algún $\varepsilon > 0$ deseado y con una norma apropiadamente definida en el espacio de salida. En la ecuación 5.1 $\hat{P}(u) = \hat{y}$ denota la salida del modelo de identificación de ahí que $\hat{y} - y = e$, donde e es el error de salida generado por $\hat{P}$ y la salida observada y.

En el campo de los sistemas lineales invariantes en el tiempo con parámetros desconocidos la obtención del modelo de identificación ha sido bastante bien estudiada, por ejemplo, sea;

$$x(k+1) = A(k) + Bu(k); \tag{5.2a}$$

$$y(k) = Cx(k) \tag{5.2b}$$

Una planta SISO controlable y observable, la matriz *A* y los vectores *B* y *C* pueden elegidos de tal forma que la ecuación 6.2 puede ser rescrita como:

$$y_p(k+1) = \sum_{i=0}^{n-1} \alpha_i y(k-1) + \sum_{i=0}^{m-1} \beta_j u(k-j) \tag{5.3}$$

Donde α_i y β_i son parámetros desconocidos. Esto implica que la salida del tiempo $(k+1)$ es una combinación de sus entradas y salidas pasadas. La ecuación (5.3) motiva la construcción de los siguientes modelos de identificación [38].

5.2 Modelos de identificación.

Existen dos modelos de identificación: identificación en serie-paralelo e identificación en paralelo.

Modelo Paralelo

La salida del modelo de identificación en paralelo es una combinación lineal de las entradas y salidas pasadas, como se muestra en la siguiente expresión (5.4).

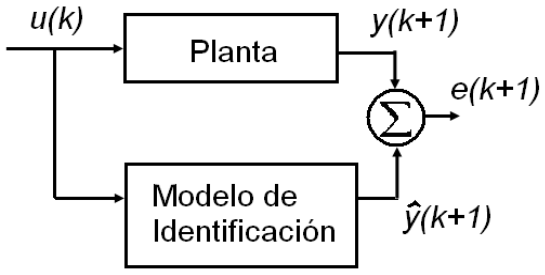


Figura 5.1 Modelo de Identificación en Paralelo

$$\hat{y}(k+1)=\sum_{i=0}^{n-1}\hat{\alpha}_i(k)\hat{y}(k-i)+\sum_{j=0}^{m-1}\hat{\beta}(k)u(k-j) \tag{5.4}$$

Donde α_i, $i=0,1,\ldots,n-1$ y β_j, $j=0,1,\ldots,m-1$ son parámetros ajustables.

Modelo Serie-Paralelo

En este caso la salida de la planta es conectada directamente al modelo durante el entrenamiento, el modelo esta en forma serie y en forma paralelo con respecto a la planta.

$$\hat{y}(k+1)=\sum_{i=0}^{n-1}\hat{\alpha}_i(k)y(k-i)+\sum_{j=0}^{m-1}\hat{\beta}(k)u(k-j) \tag{5.5}$$

Donde α_i, $i=0,1,\ldots,n-1$ y β_j, $j=0,1,\ldots,m-1$ son parámetros ajustables. La salida del modelo serie-paralelo de identificación en el instante $k+1$ es $\hat{y}(k+1)$ y es una combinación lineal de los valores pasados de la entrada y la salida de la planta.

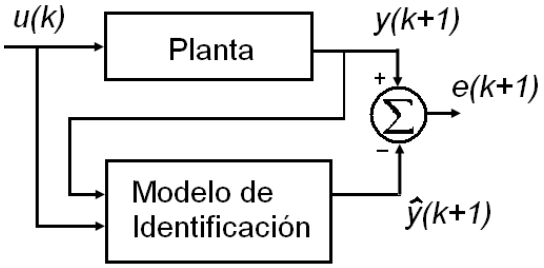


Figura. 5.2 Modelo de Identificación Serie-Paralelo

5.3 Identificación con redes neuronales

La identificación de sistemas no lineales desconocidos utilizando redes neuronales es una herramienta importante que no requiere de un modelo analítico, y se basa en las bondades de aproximación de las redes neuronales.

El empleo de redes neuronales es una herramienta muy efectiva para identificar sistemas no lineales muy complejos cuando no se tiene suficiente información del modelo de la planta, o cuando se considera a la planta como una caja negra. Los identificadores neuronales pueden clasificarse dependiendo del tipo de red neuronal que se utiliza para su construcción, de aquí que existan identificadores neuronales estáticos y dinámicos. Las redes neuronales dinámicas tienen la capacidad de representar sistemas no lineales y un buen comportamiento en

presencia de dinámicas no modeladas gracias a que su estructura incorpora retroalimentación [49].

El procedimiento para entrenar una red neuronal con la finalidad de representar la dinámica de la planta se le conoce como modelado directo [16]. La estructura para obtener este modelo se ilustra en la figura (5.3)

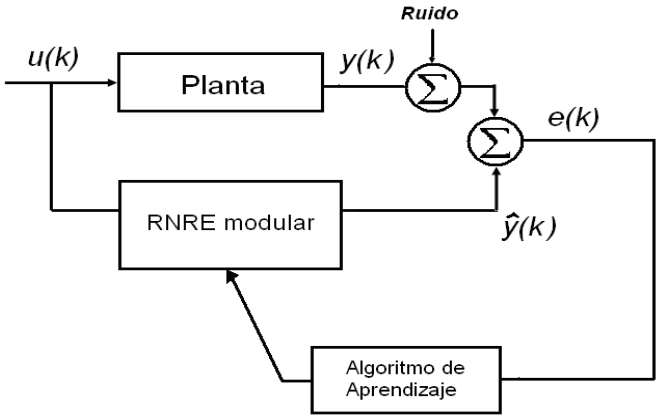


Figura 5.3. Esquema de Identificación con la RNRE modular.

En este esquema de identificación la red neuronal se conecta en paralelo con la planta, el error de identificación es la señal que se utiliza en el algoritmo de aprendizaje de la red para ajustar los pesos. Esto se hace con la finalidad de que la salida de la red aproxime a la salida de la planta, minimizando el error medio cuadrático.

Una de las ventajas del enfoque de identificación de sistemas desconocidos o “cajas negras”, es que las incertidumbres pueden considerarse dentro de la caja negra, de tal manera que el algoritmo sin modificación alguna puede tener propiedades de robustez ante variaciones en la dinámica de la planta e incertidumbres.

5.3.1 Dilema Bias-Varianza

La precisión de la aproximación de una función dada por una red neuronal se incrementa al aumentar el número de neuronas en la capa oculta, sin embargo, una metodología de diseño que consista de construir una red con tantas neuronas como sea posible es una idea poco sensata. En problemas reales de ingeniería, se requiere que la red para aproximar una función desconocida a partir de un número finito de puntos o conjunto de entrenamiento. La red no solo debe ajustarse a los puntos de entrenamiento sino que además debe ser capaz de

generalizar, es decir, dar una respuesta satisfactoria a situaciones que no se le presentaron durante el entrenamiento.

Si la red neuronal tiene muchos parámetros, será muy flexible, por lo que su salida ajustara con mucha precisión todos los puntos de entrenamiento incluyendo al ruido presente, pero en la generalización proveerá respuestas sin sentido en situaciones en que los datos de entrada no sean los presentes en la fase de entrenamiento, por el contrario, una red neuronal con pocos parámetros no será lo suficientemente compleja para aprender la complejidad de la función, por lo que no será capaz de de aprender de los datos de entrenamiento. A este dilema se le conoce como el dilema bias-varianza. En otras palabras, el modelo de red seleccionado debe considerar el mejor balance entre la capacidad de aprendizaje y la capacidad de generalización; si el modelo aprende muy bien, entonces su generalización será pobre [50].

Desde el punto de vista teórico, el modelo que buscamos es uno para el cual la función teórica de costo $\int (y_P(x) - g(x,w))^2 P_X(x) dx$ es mínima. Esta cantidad puede ser dividida en dos partes: el bias y la varianza. El bias expresa el promedio sobre todos los posibles conjuntos de entrenamiento de la diferencia al cuadrado entre las predicciones del modelo y la función de regresión. La varianza expresa la sensitividad del modelo con respecto al conjunto de entrenamiento. Debido a que la función de costo anterior no puede ser calculada, la función empírica de mínimos cuadrados es utilizada durante el entrenamiento.

5.4 Identificación con la RNRE modular.

La RNRE modular además de ser capaz de generar su propio estado, el cual es retroalimentado logra que la RNRE modular sea robusta a incertidumbres y/o dinámicas no modeladas, y algo más importante que no está sujeta a consideraciones del orden del sistema.

Sea un sistema SISO no lineal estable dado por:

$$x(k+1) = f(x(k), u(k)) \tag{5.6}$$

$$y(k) = g(x(k)) \tag{5.7}$$

Donde x, u y y son el estado, la entrada y salida del sistema respectivamente, $x \in \Re^n$, $f : \Re^{n+1} \to \Re^n$, $g : \Re^n \to \Re$.

Entonces un posible esquema de identificación del sistema (5.6 - 5.7) es el siguiente:

$$x_R(k+1) = Ax_R(k) + Bu(k)\ ;\ A = \begin{bmatrix} A_D & 0 \\ 0 & A_m \end{bmatrix} \tag{5.8}$$

$$z(k) = \Psi(x_R(k)) \tag{5.9}$$

$$y_R(k) = \Psi(Cz(k)) \tag{5.10}$$

Estas ecuaciones representan a una RNRE modular donde $x_R \in \Re^n$ es el estado de la RNRE modular, tal que $x_R(k)$ no es necesariamente una aproximación de $x(k)$, u y $y_R \in \Re$ son la entrada y salida de la red con $y_R(k) \cong y(k)$, $\Psi \in \Re^n$ es un vector de funciones de activación $\varphi(\cdot)$, $A \in \Re^{nxn}$, $B \in \Re^{nx1}$ y $C \in \Re^{1xn}$ son matrices de pesos de retroalimentación, entrada y salida respectivamente. A se divide en A_D que es de estructura diagonal y A_m que consta de 1 o más bloques de pares de neuronas con retroalimentación entre ellas, esto es bloques de dimensión dos con elementos distintos de cero.

5.5 Teorema de estabilidad de la RNRE con BP usada como identificador.

Sea la RNRE con estructura canónica de Jordan dada por las ecuaciones (3.17) (3.20) y el modelo de una planta no lineal como sigue:

$$X_p(k+1) = G[X_p(k), U(k)] \tag{5.11}$$

$$Y_p(k) = F[X_p(k)] \tag{5.12}$$

donde: $\{Y_p(\cdot),\ X_p(\cdot), U(\cdot)\}$ son las variables de salida, estado y entrada con dimensiones L, N_p, M, respectivamente; F(·), G(·) son funciones vectoriales no lineales con las dimensiones respectivas. Bajo la suposición de que ha sido identificada la RTNN, la aplicación del algoritmo BP para A(·), B(·), C(·) en general de la forma matricial, descrita en las ecuaciones (4.73)-(4.85), y las tazas de aprendizaje $\eta(k)$, α (k) (aquí son consideradas como dependientes del tiempo y normalizadas con respecto al error) son derivadas usando la siguiente función de Lyapunov.

$$L(k) = L_1(k) + L_2(k) \tag{5.13}$$

Donde: $L_1(k)$ y $L_2(k)$ están dadas por:

$$\mathrm{L_1(k)} = \frac{1}{2}\mathrm{e^2(k)}, \tag{5.14}$$

$$L_2(k) = \mathrm{tr}\left(W_A(k)W_A^T(k)\right) + \mathrm{tr}\left(W_B(k)W_B^T(k)\right) + \mathrm{tr}\left(W_C(k)W_C^T(k)\right); \tag{5.15}$$

Donde: $W_A(k) = \hat{A}(k) - A^*, \quad W_B(k) = \hat{B}(k) - B^*, \quad W_C(k) = \hat{C}(k) - C^*,$ son vectores del error estimado y (A^*, B^*, C^*), $(\hat{A}(k), \hat{B}(k), \hat{C}(k))$ denotan el peso ideal neuronal y el estimado de los pesos del k-ésima iteración respectivamente en cada caso. Después la identificación del error está delimitada por:

$$\begin{aligned} L(k+1) &= L_1(k+1) + L_2(k+1) < 0, \\ \Delta L(k+1) &= L(k+1) - (k); \end{aligned} \tag{5.16}$$

donde la condición para $L_1(k+1) < 0$ es la siguiente:

$$\frac{\left(1 - \frac{1}{\sqrt{2}}\right)}{\psi_{max}} < \eta_{max} < \frac{\left(1 + \frac{1}{\sqrt{2}}\right)}{\psi_{max}}; \tag{5.17}$$

y para $L_2(k+1) < 0$ tenemos:

$$\Delta L_2(k+1) < -\eta_{max} |e(k+1)|^2 \alpha_{max} |e(k)|^2 + d(k+1). \tag{5.18}$$

Note que η_{max} cambia adaptativamente durante el aprendizaje de RTNN y:

$$\eta_{max} = \max_{i=1}^{3} \{\eta_i\}; \tag{5.19}$$

donde todas las dinámicas no modeladas, las aproximaciones del error y las perturbaciones son representadas mediante el termino de la taza del Lema de Convergencia usado. La prueba completa de estabilidad está dada en [35]

5.6 Resultados de Simulación.

5.6.1 Identificación de la planta de un vehículo espacial

En esta sección presentaremos los resultados de la identificación utilizando una RNRE modular con topología, 1,5,1, es decir, una neurona en la capa de entrada, una de salida y cinco en la capa oculta, tres de ellas componen una matriz diagonal, las dos restantes forman un bloque con pesos de retroalimentación entre ambas neuronas. El algoritmo de aprendizaje empleado es el Levenberg-Marquardt recursivo.

La función de activación utilizada es de tipo tangente hiperbólico. Dado que deseamos que la función tangente hiperbólica opere en su región lineal y no en saturación, la pendiente debe ser modificada de tal forma que la parte lineal tenga una mayor extensión, en este caso utilizamos un coeficiente de ρ igual a 4 al emplear la función tangente hiperbólico como se definió en la ecuación (3.7).

La planta esta descrita por las ecuaciones (2.1-2.3) descritas en el capítulo 2. Se utilizo un control con retroalimentación estática de estado calculado a partir de la linealización de la planta alrededor de un punto de equilibrio del sistema.

El valor de la media aritmética (ε) y la desviación estándar (σ) del error medio cuadrático para las 20 corridas del programa de identificación para los algoritmos Backpropagation y Levenberg-Marquardt se presentan en la tabla 6.2. Estos valores fueron calculados utilizando las siguientes formulas:

$$\varepsilon = \frac{1}{n}\sum_{k=1}^{n} \xi_{avk} \tag{5.20}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \Delta_i^2} \tag{6.12}$$

$$\Delta = \xi_{av} - \varepsilon \tag{5.21}$$

La figura (5.4) muestra los resultados obtenidos en términos del error medio cuadrático al correr 20 veces la simulación, en cada una de estas simulaciones se agregó ruido de medición a la salida de la planta. La tabla (5.1) muestra el comparativo de la media y desviación estándar entre los algoritmos backpropagation y Levenberg-Marquardt. La tabla (5.2) muestra el error medio cuadrático para cada corrida.

Algoritmo BP	Algoritmo L-M
ε = 4.10E-04	ε = 2.60E-04
σ = 1.19072E-04	σ = 2.69076E-05

Tabla (5.1) Valores de la Media y desviación estándar para los algoritmos BP y LM

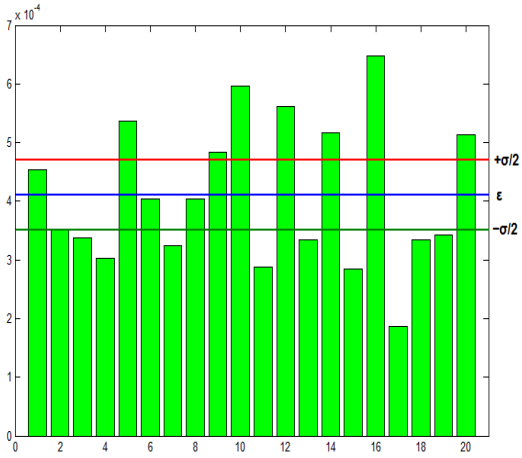


Figura 5.4 Comparación del Error medio cuadrático final después de 20 simulaciones

ALGORITMO BP					
No	1	2	3	4	5
EMC	4.53E-04	3.52E-04	3.37E-04	3.03E-04	5.36E-04
No	6	7	8	9	10
EMC	4.04E-04	3.25E-04	4.04E-04	4.84E-04	5.96E-04
No	11	12	13	14	15
EMC	2.88E-04	5.62E-04	3.34E-04	5.17E-04	2.84E-04
No	16	17	18	19	20
EMC	6.47E-04	1.86E-04	3.35E-04	3.42E-04	5.13E-04

Tabla (5.2). Valores del EMC para cada simulación, algoritmo Backpropagation.

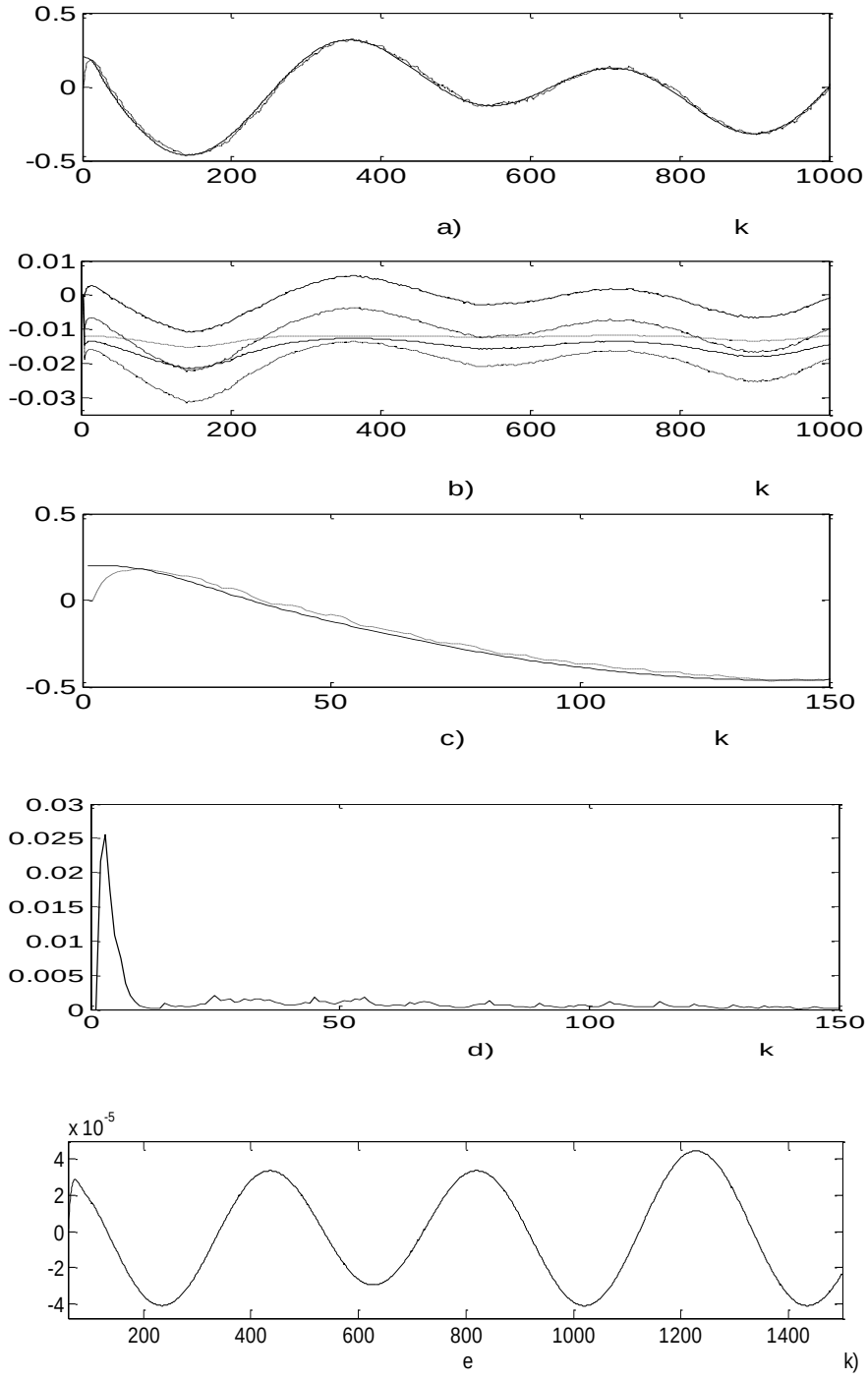


Figura 5.5 Resultados de Identificación utilizando BP para el modelo del vehículo espacial. a) Señal de salida de la planta (Línea continua) contra la señal de salida de la RNRE modular (línea punteada), b) Estados de la RNRE modular, c) Comparación entre la salida de la planta (continua) y la salida de la RNRE modular (punteada) durante los primeros instantes de la simulación, d) Error Medio Cuadrático e) Señal de entrada

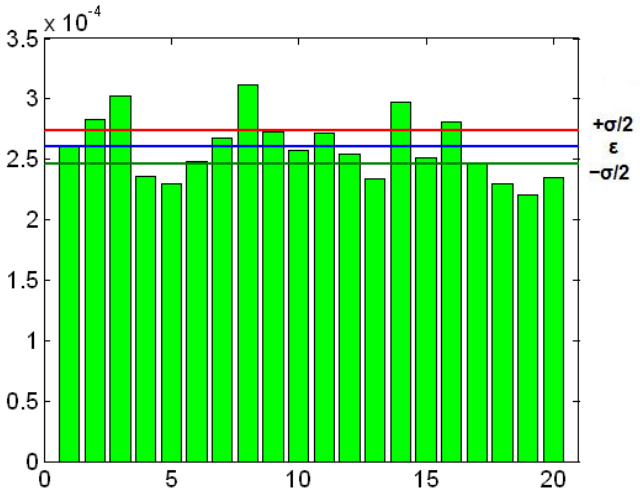


Figura 5.6 Comparación del Error medio cuadrático final después de 20 simulaciones del algoritmo de identificación utilizando L-M.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	2.61E-04	3.18E-04	2.83E-04	2.36E-04	2.30E-04
No	6	7	8	9	10
EMC	2.48E-04	2.68E-04	3.12E-04	2.73E-04	2.57E-04
No	11	12	13	14	15
EMC	2.72E-04	2.54E-04	2.34E-04	2.97E-04	2.51E-04
No	16	17	18	19	20
EMC	2.81E-04	2.46E-04	2.30E-04	2.21E-04	2.35E-04

Tabla (5.3). Valores del error medio cuadrático para cada simulación, algoritmo L-M

En la tabla 5.1 se muestra que mientras el algoritmo Backpropagation arroja una media aritmética del error bastante buena, el algoritmo Levenberg-Marquardt lo supera al obtener una media del error 36% menor que el primer algoritmo. Más aun las desviaciones de este error promedio de cada una de las simulaciones son menores para el algoritmo de Levenberg-Marquardt, como se puede ver en la figura 5.6, que para las respectivas simulaciones con el algoritmo Backpropagation, como se observa en la figura 5.4.

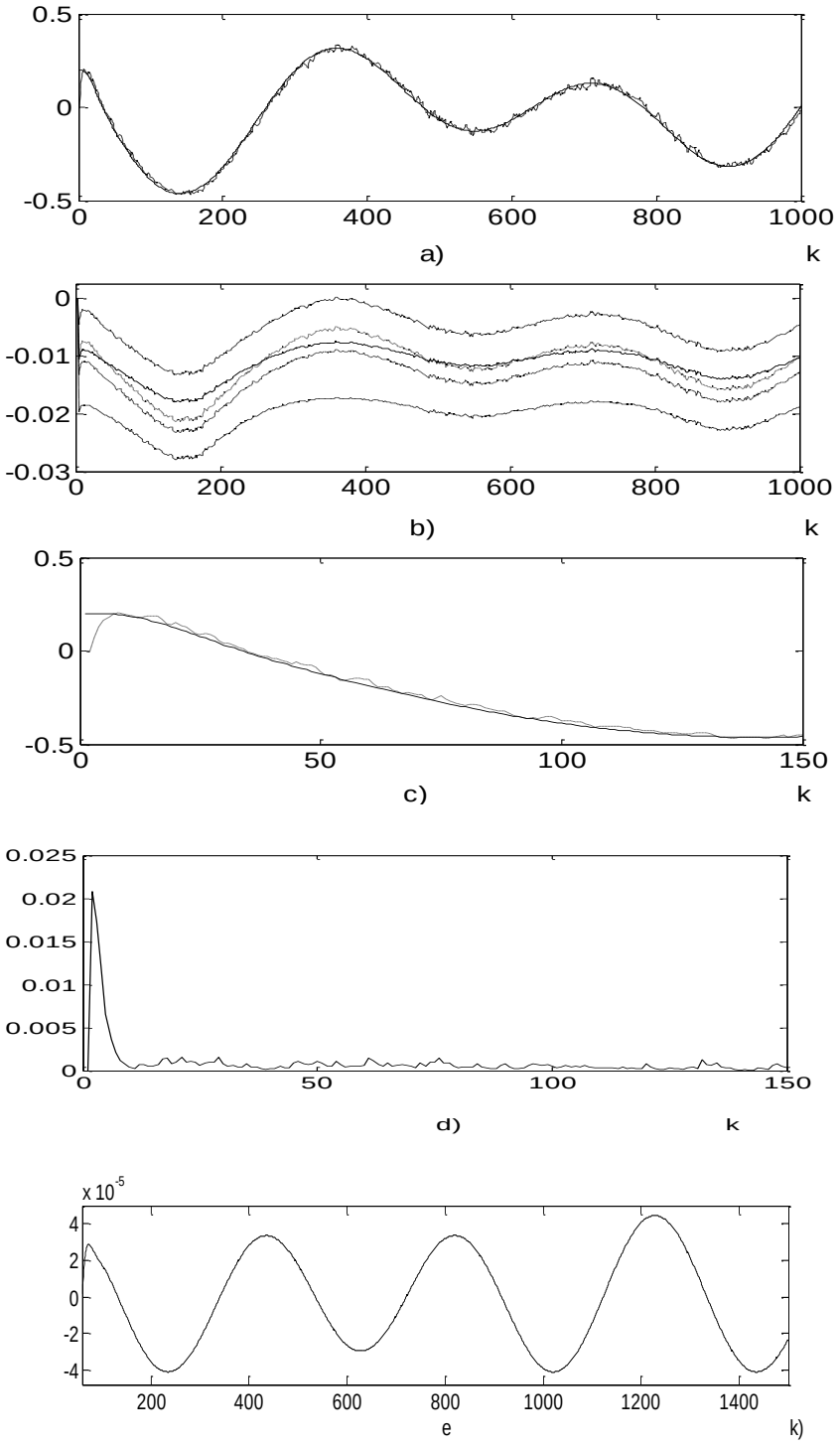


Figura 5.7 Resultados de Identificación utilizando L-M para el modelo del vehículo espacial. a) Señal de salida de la planta(Línea continua) contra la señal de salida de la RNRE modular (línea punteada), b) Estados de la RNRE modular, c) Comparación entre la salida de la planta(continua) y la salida de la RNRE modular (punteada) durante los primeros instantes de la simulación, d) Error Medio Cuadrático, e) Señal de entrada

5.6.2 Identificación de la planta de un motor de combustión interna.

La topología usada para la identificación de esta planta es una 2,5,2, es decir posee dos neuronas en la capa de entrada, cinco neuronas en la capa oculta y dos neuronas en la capa de salida. Nuevamente utilizamos la función de activación tangente hiperbólica y un valor de ρ igual a 4. La planta fue descrita por las ecuaciones (2.4-2.12) del capítulo 2. La figura (5.10) muestra los resultados obtenidos, al identificar a la planta

Algoritmo BP	Algoritmo L-M
ε = 3.5024e-005	ε = 1.0231e-006
σ = 2.2356e-006	σ = 3.0344e-008

Tabla (5.4) Valores de la media y desviación estándar para los algoritmos BP y LM

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.3652E-04	3,54E-05	3,46E-05	3,78E-05	3,54E-05
No	6	7	8	9	10
EMC	3,67E-05	3,22E-05	3,35E-05	3,63E-05	3,13E-05
No	11	12	13	14	15
EMC	3,26E-05	3,55E-05	3,50E-05	3,85E-05	3,58E-05
No	16	17	18	19	20
EMC	3,49E-05	3,08E-05	3,62E-05	3,91E-05	3,25E-05

Tabla (5.5). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo BP.

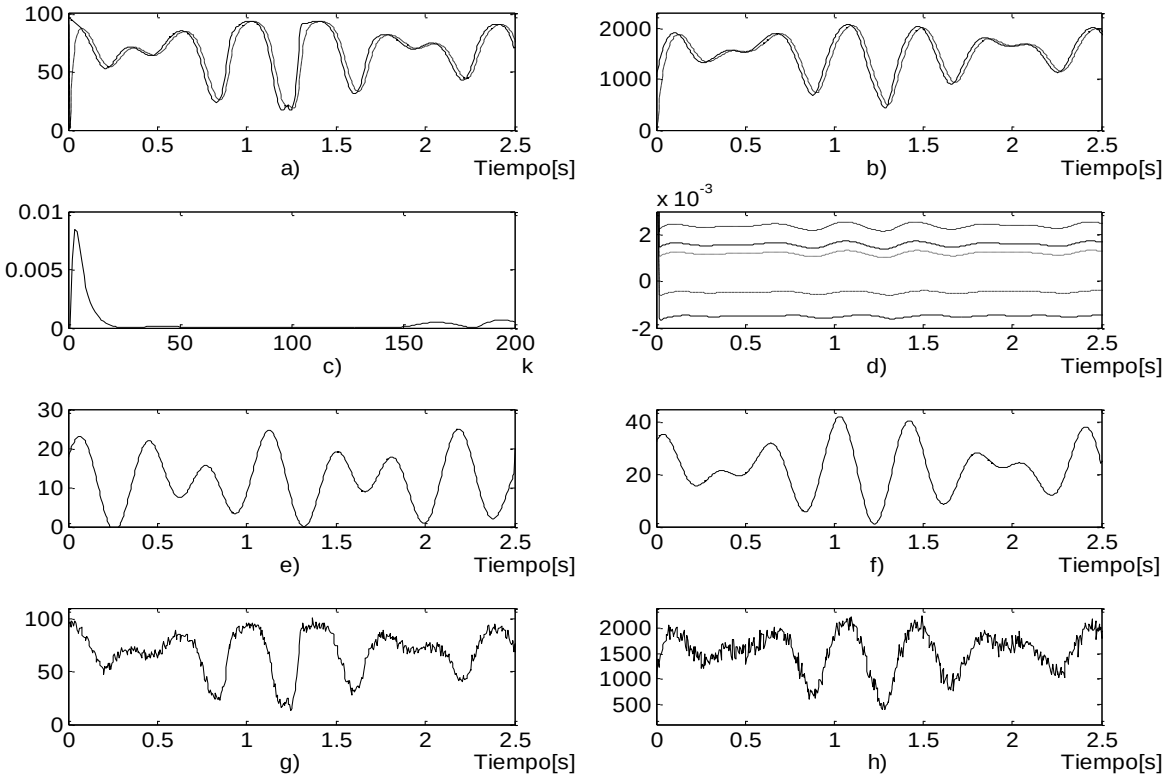


Figura 5.8. Resultados de Identificación utilizando el algoritmo BP a) Presión (línea continua) y señal de salida de la red neuronal (línea punteada), b) RPM (Continua) y señal de salida 2 de la red neuronal (Punteada), c) Error medio cuadrático, d) Estados de la red, e)Señal de entrada 1, f) Señal de entrada 2, g) y h) Señales de medición con ruido.

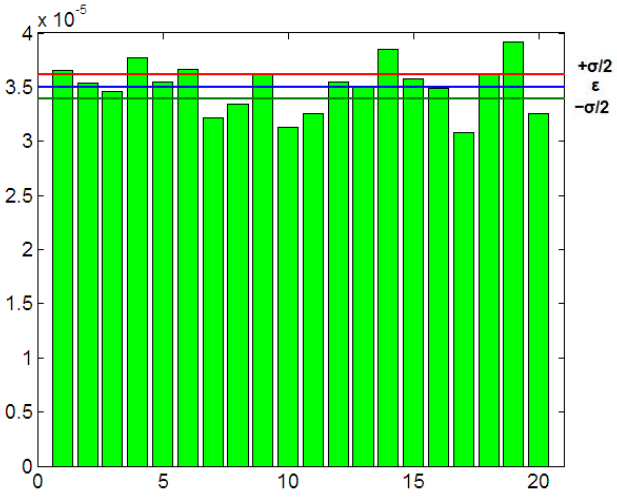


Figura 5.9 Comparación del Error medio cuadrático final después de 20 simulaciones del algoritmo de identificación utilizando BP.

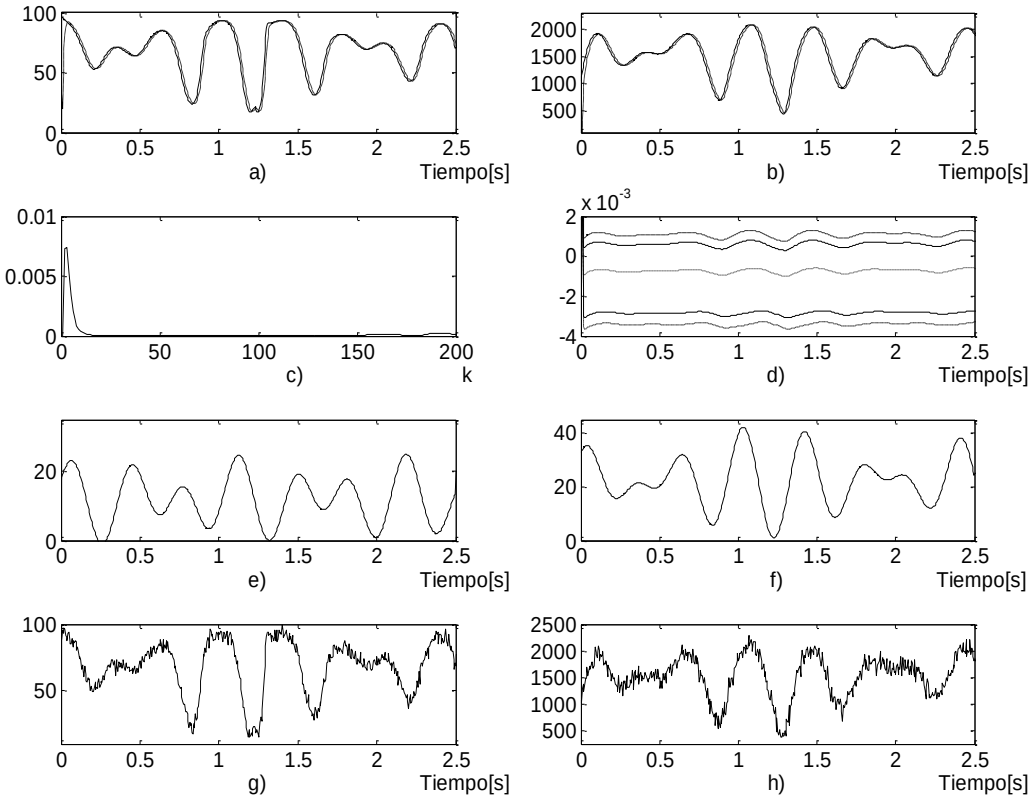


Figura 5.10. Resultados de Identificación utilizando el algoritmo L-M a) Presión (Continua) y señal de salida 1 de la red neuronal (Punteada), b) RPM (Continua) y señal de salida 2 de la red neuronal (Punteada), c) Error medio cuadrático, d) Estados de la red, e)Señal de entrada 1, f) Señal de entrada 2, g) y h) Señales de medición con ruido.

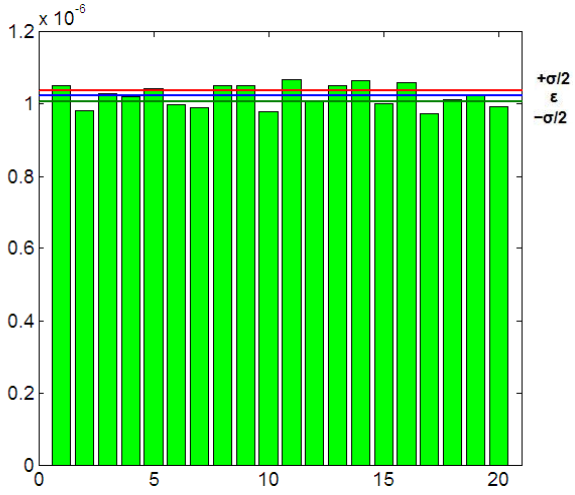


Figura 5.11 Comparación del Error medio cuadrático final después de 20 simulaciones del algoritmo de identificación utilizando L-M.

ALGORITMO LM					
No	1	2	3	4	5
EMC	1,05E-06	9,82E-07	1,03E-06	1,02E-06	1,04E-06
No	6	7	8	9	10
EMC	1,00E-06	9,90E-07	1,05E-06	1,05E-06	9,80E-07
No	11	12	13	14	15
EMC	1,07E-06	1,01E-06	1,05E-06	1,07E-06	1,00E-06
No	16	17	18	19	20
EMC	1,06E-06	9,74E-07	1,01E-06	1,02E-06	9,93E-07

Tabla (5-6). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo L-M.

Haciendo un análisis cuantitativo de las respuestas obtenidas, según la tabla 5.4, se tiene que el algoritmo de Levenberg-Marquardt presenta menor error promedio a lo largo de las 20 simulaciones corridas con respecto a la identificación con el algoritmo Backpropagation; además de una menor desviación estándar para cada una de las 20 simulaciones utilizando el primer algoritmo con respecto al segundo algoritmo.

Un análisis cualitativo de las respuestas de identificación, de acuerdo a las figuras 5.8, a) y b), y 5.9, a) y b), muestra que la respuesta del identificador neuronal entrenado con el algoritmo de Levenberg-Marquardt tiene una mejor aproximación a las salidas del sistema a identificar, que el identificador entrenado con el algoritmo Backpropagation.

5.5.3 Identificación de la planta del sistema de dos masas-resorte-amortiguador.

En este caso la topología usada para la identificación de esta planta fue 2,5,2, es decir posee dos neuronas en la capa de entrada, cinco neuronas en la capa oculta y dos neuronas en la capa de salida. En este caso utilizamos 2 bloques de dimensión dos en nuestra matriz de pesos de la RNRE modular.

Se utilizó la función de activación tangente hiperbólico y un valor de ρ igual a 4 para una mejor identificación. La planta esta descrita por las ecuaciones (2.13-2.17) del capítulo 2.

La figura (65.12) muestra los resultados obtenidos, al identificar a la planta y la tabla (5.8) muestra los valores propios de la matriz de pesos de la capa oculta A, obtenidos en dos simulaciones con valores iniciales de los pesos aleatorios.

Algoritmo BP	Algoritmo L-M
ε = 8.6712E-06	ε = 3.11E-06
σ = 3.0235E-08	σ = 1.38E-8

Tabla (5.7) Valores de la Media y desviación estándar para los algoritmos BP y LM

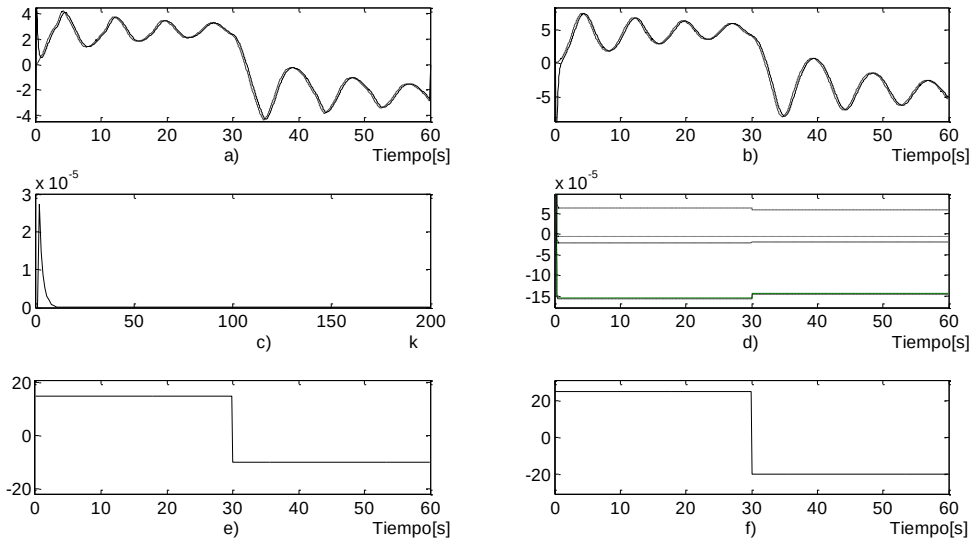


Figura 5.12. Resultados de Identificación utilizando el algoritmo BP para el modelo de dos masas con resorte y amortiguador a) Posición de la masa m en el eje horizontal (línea Continua) y señal de salida de la red neuronal (línea Punteada), b) Posición de la masa M en el eje horizontal (línea Continua) y señal de salida 2 de la red neuronal (línea punteada), c) Error medio cuadrático, d) Estados de la red2 e)Señal de entrada 1, f) Señal de entrada 2, g) y f) Señales de medición con ruido.

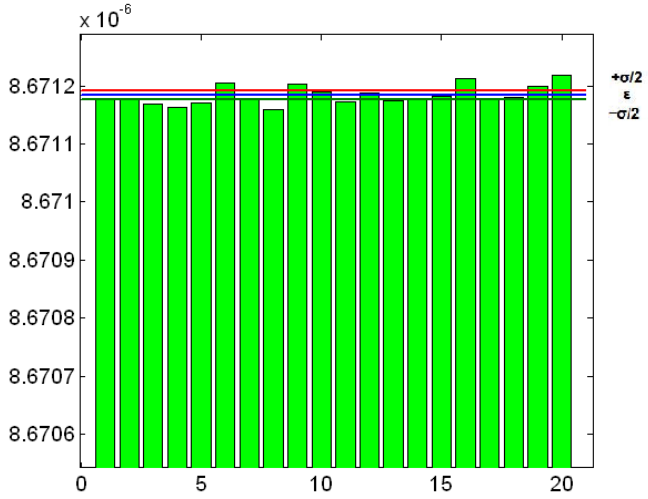


Figura 5.13 Comparación del Error medio cuadrático final después de 20 simulaciones del algoritmo de identificación utilizando BP.

ALGORITMO BP					
No	1	2	3	4	5
EMC	8.67E-06	8.67E-06	8.67E-06	8.67E-06	8.67E-06
No	6	7	8	9	10
EMC	8.67E-06	8.67E-06	8.67E-06	8.67E-06	8.67E-06
No	11	12	13	14	15
EMC	8.67E-06	8.67E-06	8.67E-06	8.67E-06	8.67E-06
No	16	17	18	19	20
EMC	8.67E-06	8.67E-06	8.67E-06	8.67E-06	8.67E-06

Tabla (5.9). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo BP.

ALGORITMO LM					
No	1	2	3	4	5
EMC	3.11E-06	3.11E-06	3.11E-06	3.11E-06	3.11E-06
No	6	7	8	9	10
EMC	3.11E-06	3.11E-06	3.11E-06	3.11E-06	3.11E-06
No	11	12	13	14	15
EMC	3.11E-06	3.11E-06	3.11E-06	3.11E-06	3.11E-06
No	16	17	18	19	20
EMC	3.11E-06	3.11E-06	3.11E-06	3.11E-06	3.11E-06

Tabla (5.10). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo L-M.

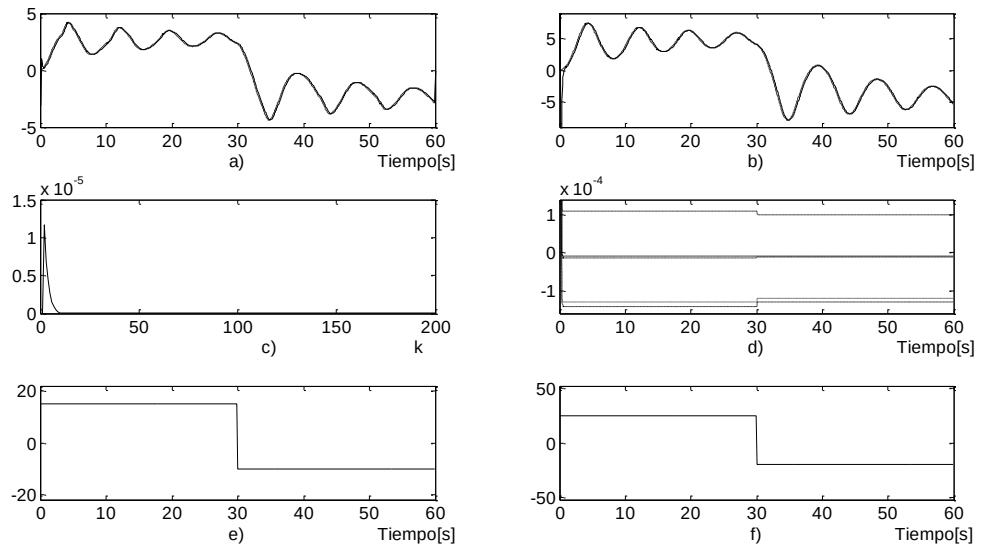


Figura 5.14. Resultados de Identificación utilizando el algoritmo LM para el modelo de dos masas con resorte y amortiguador, a) Posición de la masa m en el eje horizontal (azul) y señal de salida de la red neuronal (verde), b) Posición de la masa M en el eje horizontal (azul) y señal de salida 2 de la red neuronal (verde), c) Error medio cuadrático, d) Estados de la red2 e)Señal de entrada 1, f) Señal de entrada 2, g) y f) Señales de medición con ruido.

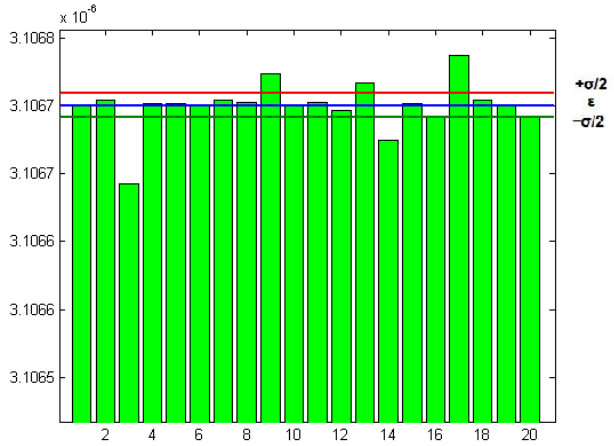


Figura 5.15 Comparación del Error medio cuadrático final después de 20 simulaciones del algoritmo de identificación utilizando L-M.

Analizando los resultados de la tabla 5.7 se observa que la menor media del error de aproximación se logra con el algoritmo de Levenberg-Marquardt, así como una menor desviación estándar de dicho error para cada una de las simulaciones, con respecto al algoritmo Backpropagation.

5.6 Conclusiones

En este capítulo se realizó la identificación de sistemas mecánicos por medio de RNRE, con este método no es necesario linealizar los modelos. En la identificación el objetivo es construir un modelo apropiado el cual cuando sea sometido a la misma entrada u(k) de la planta, produzca una salida $\hat{y}(k)$ tal que aproxime a la salida de la planta $y(k)$.

A partir de los resultados de este capítulo se puede concluir que la identificación neuronal con el algoritmo de Levenberg-Marquardt presenta un menor error medio de identificación y así como una menor desviación de cada una de las simulaciones de la media, con respecto al identificador entrenado con el algoritmo Backpropagation; todo esto se refleja en una mejor aproximación por parte del primer algoritmo.

Esta parte de identificación es de gran utilidad para aplicar el control en el siguiente capítulo por medio de RNRE's.

Capitulo 6

Control Adaptable Neuronal con Termino Integral

6.1 Sistemas de Control en el Espacio de Estados

Un sistema moderno puede tener varias entradas y salidas relacionadas entre sí, en una forma muy complicada. Para analizar un sistema con estas características, se requiere reducir la complejidad de las expresiones matemáticas, además de recurrir a computadoras para resolver cálculos tediosos necesarios en el análisis. Desde este punto de vista, el método más adecuado para el análisis de estos sistemas es el método en espacio de estados [51], [52].

En la teoría de control clásico, el análisis y diseño de los controladores se basa en la utilización de funciones de transferencia junto con una serie de técnicas gráficas (Bode, lugar geométrico de las raíces), en donde se concentran importantes las señales de entrada, salida y error [51].

La teoría de control moderna se basa en la descripción de las ecuaciones del sistema en términos de n ecuaciones diferenciales de primer orden. El uso de la notación matricial simplifica mucho la representación matemática de sistemas de ecuaciones. El aumento de la cantidad de variables de estado, de entradas o salidas, no incrementa la complejidad de las ecuaciones. De hecho es posible proseguir el análisis de sistemas más complicados con entradas y salidas múltiples con procedimientos ligeramente complicados que los requeridos por el análisis de sistemas de ecuaciones diferenciales escalares de primer orden REF.

6.2 Sistemas de Control Adaptable

Un sistema de control adaptable es aquel en el que en forma continua o discreta y automática mide características dinámicas de la planta, las compara con las características de la dinámica deseada y usa la diferencia para variar los parámetros ajustables del sistema o generar una señal actuante de modo que se mantenga un desempeño óptimo, independientemente de las condiciones ambientales [51].

6.3 Índice de desempeño

La base de control adaptable descansa en el fundamento de la existencia de una condición de operación de desempeño óptimo del sistema. Entonces es necesario definir qué constituye un desempeño óptimo. Un método para determinar este desempeño óptimo se basa en la teoría del gradiente descendiente, en donde se busca minimizar un índice de desempeño o error generado por la diferencia entre la señal de referencia $y_d(k)$ y la salida de la planta $y(k)$. Este error es definido como $e(k) = y_d(k) - y(k)$ y w son los parámetros del controlador. El índice de desempeño o función de costo es:

$$J(w(k)) = \frac{1}{2}e^2(k) \tag{6.1}$$

$$\hat{J}(w) = \frac{1}{N}\sum_{k=1}^{N} J(w(k)) \tag{6.2}$$

Para propósito de optimización se busca obtener los valores mínimos en este criterio y para esto se modifica el factor w en dirección negativa del gradiente de J i.e. de la siguiente forma:

$$\Delta w = -\eta \frac{\partial J}{\partial w} \tag{6.3}$$

Donde η es la constante de entrenamiento. Existen otros métodos más complicados de adaptación como el caso de la función de Lyapunov. [53]

6.4 Control Adaptable usando RNRE

Un esquema de control que use redes neuronales por definición es un control adaptable debido a que la RN siempre ajusta sus pesos tratando de minimizar el índice de desempeño.

Recientemente las redes neuronales se han dado a conocer como una poderosa herramienta en el aprendizaje de la dinámica de sistemas no lineales. Debido a su masivo paralelismo, rápida adaptación e inherente capacidad de aproximación [4].

El problema de control de seguimiento consiste en diseñar un controlador que reduce asintóticamente el error entre la salida de la planta y la señal de referencia, La RNRE ha probado ser un sistema de identificación el cual nos da un modelo de estado lineal estable y controlable de la planta, este modelo es mejorado constantemente durante el aprendizaje y puede ser usado para el diseño de

sistemas de control, aplicando la técnica de diseño ya bien conocidas para seguimiento de sistemas dados en [54].

Se han logrado resultados para el control de sistemas dinámicos no lineales [55], [56], [57] y [58]. Baruch et al en [14] implementa un control adaptable de tipo indirecto para una planta no lineal con una RNRE, en [12] son implementados 4 distinto tipos de control adaptable neuronal para plantas no lineales con una RNRE.

6.4.1 Teorema de estabilidad de la RNRE con algoritmo BP usada como controlador directo

Sea la RNRE modular con estructura canónica de Jordan dada por las ecuaciones (3.17)-(3.20) y el modelo de planta no lineal dado por:

$$X_p(k+1) = F[X_p(k), U(k), O(k)] \tag{6.4}$$

$$Y_p = \varphi(X_p(k)) \tag{6.5}$$

La aplicación de algoritmo BP para $A(\cdot), B(\cdot), C(\cdot)$ en general de forma matricial, descrita por las ecuaciones (4.86) - (4.98) sin termino momento y tasa de aprendizaje $\eta(k)$considerada como dependiente del tiempo y normalizada con respecto al error, son derivadas usando la siguiente función de Lyapunov.

$$L(k) = L_1(k) + L_2(k) \tag{6.6}$$

En la cual $L_1(k)\ y\ L_2(k)$ están dados por:

$$L_1(k) = \frac{1}{2}e^2(k) \tag{6.7}$$

$$L_2(k) = tr(\widetilde{W}_A(k)\widetilde{W}_{\overline{A}}^{T}(k)) + tr(\widetilde{W}_B(k)\widetilde{W}_{\overline{B}}^{T}(k)) + tr(\widetilde{W}_C(k)(\widetilde{W}_{\overline{C}}^{T}k)) \tag{6.8}$$

Donde $\widetilde{W}_A(k) = \hat{A}(k) - A^*, \widetilde{W}_B(k) = \hat{B}(k) - B^*, \widetilde{W}_C(k) = \hat{C}(k) - C^*$son vectores del error de estimación y además A^*, B^*, C^* y $\hat{A}(k), \hat{B}(k), \hat{C}(k)$ denotan los pesos neuronales ideales y el estimado de los pesos neuronales en el k-ésimo paso, respectivamente, para cada caso.

Definamos:

$$\psi_{max} = max\|\psi\| \tag{6-9}$$

$$\vartheta_{max} = max\|\vartheta\| \tag{6.10}$$

$$\psi(k) = \frac{\partial o(k)}{\partial W(k)} \tag{6.11}$$

$$\vartheta(k) = \frac{\partial y(k)}{\partial W(k)} \tag{6.12}$$

donde: W es el vector compuesto por todos los pesos de la RNRE usada como controlador y $\|.\|$ es la norma Euclideana en R^n.

Entonces el error de identificación esta acotado:

$$L(k+1) = L_1(k+1) + L_2(k+1) < 0 \tag{6.13}$$

$$\Delta L(k+1) = L(k+1) - L(k) \tag{6.14}$$

donde la condición para $L_1(k+1) < 0$ sea cierta es que la máxima tasa de aprendizaje queda dentro de los limites

$$0 < \eta_{max} < \frac{2}{\vartheta_{max}^2 \psi_{max}^2}, \tag{6.15}$$

y para $L_2(k+1) < 0$, se tiene:

$$\Delta L_2(k+1) < -\eta_{max} |e(k+1)|^2 + \beta(k+1). \tag{6.16}$$

Nótese que η_{max} cambia durante el proceso de aprendizaje de la red, donde:

$$\eta_{max} = \max_{i=1}^{3} \{\eta_i\}. \tag{6.17}$$

El termino β representa todas las dinámicas no modeladas, errores de aproximación y perturbaciones. La prueba completa del teorema puede encontrarse en [35]

6.5 Sistema de Control Adaptable Directo Neuronal con Termino Integral.

En el control adaptable directo los parámetros del controlador son ajustados directamente para minimizar el índice de desempeño. En muchos casos el desempeño deseado puede ser especificado en términos de las características de un sistema dinámico o señal de referencia, el cual es una realización de la conducta deseada del sistema en lazo cerrado. En estos casos el controlador es diseñado, tal que para un modelo de una planta dado, el sistema en lazo cerrado tiene las características de la dinámica deseada del sistema [12].

En la figura (6.1) se muestra el esquema de control adaptable directo neuronal con un término integral, este esquema de control posee un identificador compuesto de una RNRE y otra RNRE como controlador y un término integral. Los estados de la red de identificación son utilizados como entrada a la red de control, de esta forma el control estará formado por una parte *feedforward* dependiente de la trayectoria y otra parte *feedback* dependiente de los estados que genera el identificador.

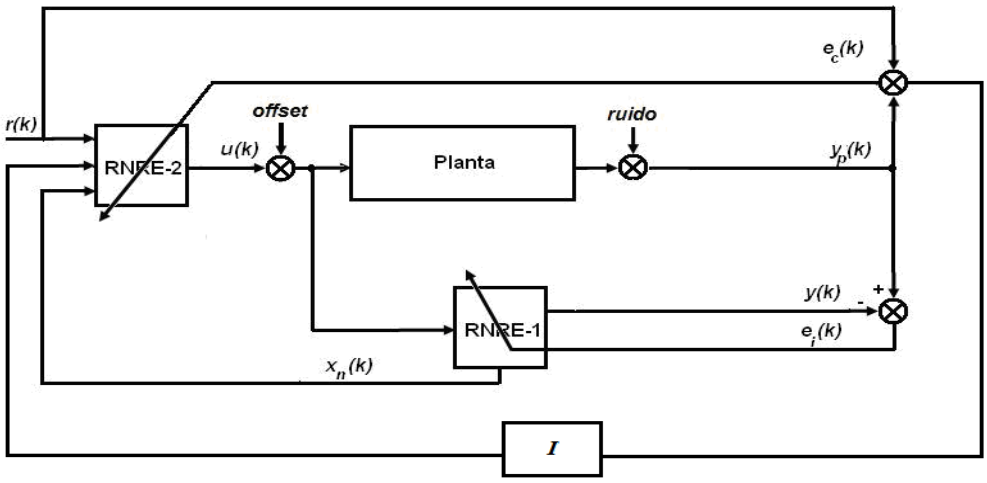


Figura 6.1 Diagrama de bloques de un sistema de control adaptable directo

Tomemos el modelo no lineal

$$X_p(k+1) = F\left[X_p(k), U(k), O(k)\right] \tag{6.18}$$

$$Y_p(k) = \varphi(X_p(k)) \tag{6.19}$$

Linealizando el modelo:

$$X_p(k+1) = AX_p(k) + B_pU(k) + B_pO(k) \tag{6.20}$$

$$Y_p(k) = C_pX_p(k) \tag{6.21}$$

donde $Ap \in R^{(Np\ x\ Np)}$ es la matriz de estado; $Bp \in R^{(Np\ x\ M)}$ es la matriz de entradas; $Cp \in R^{(L\ x\ Np)}$ es la matriz de salida y $L = M$ se supone cierta. Para poder estudiar el comportamiento del sistema en lazo cerrado, es conveniente definir las siguientes transformadas *z*

$$W^p(z) = C_p(zI - A_p)^{-1}B_p \tag{6.22}$$

$$Y_p = W^p(\text{U+O}) \tag{6.23}$$

$$P^i(z) = (zI - J^i)^{-1}B^i \tag{6.24}$$

$$X^i(z) = P^i(z)\text{U(z)} \tag{6.25}$$

Siguiendo el diagrama a bloques de la figura (6.1) podemos definir que la RNRE de control posee la misma topología y aprendizaje que la RNRE de identificación y que su modelo linealizado es:

$$X^C(k+1) = J^CX^C(k) - B_1^CV(k) - B_2^CX^i(k) - B_3^CR(k) \tag{6.26}$$

$$U(k) = C^CX^C(k) \tag{6.27}$$

donde $X^C(k)$ es el vector de estado de la RNRE de control de dimension N_C donde $N_C \leq L + M + N_i$; $J_C \in R^{(N_C\ x\ N_C)}$; $B_1^C \in R^{(N_C\ x\ L)}$; $B_2^C \in R^{(N_C\ x\ N_i)}$; $B_3^C \in R^{(N_C\ x\ L)}$; $C^C \in R^{(M\ x\ N_C)}$ son matrices constantes, el termino integral $V(k) \in R^{(L\ x\ 1)}$ se define como:

$$V(k+1) = V(k) + T_OY_p(k) \tag{6.28}$$

para derivar la dinámica en lazo cerrado necesitamos definir las siguientes transformadas z

$$I(z) = I(zI - I)^{-1} \quad (6.29)$$

$$V(z) = I(z)Y_p(z) \quad (6.30)$$

$$Q_1(z) = C^C(zI - J^C)^{-1}B_1^C \quad (6.31)$$

$$Q_2(z) = C^C(zI - J^C)^{-1}B_2^C \quad (6.32)$$

$$Q_3(z) = C^C(zI - J^C)^{-1}B_3^C \quad (6.33)$$

Como ya se ha visto, el aprendizaje es convergente y la RNRE es estable, controlable y observable, entonces los errores de identificación y control tienden a cero. Se asume que la planta posee estabilidad BIBO, entonces las funciones de transferencia (6.22), (6.23), (6.30)-(6.33) son estables. Expresando (6.26) y (6.27) utilizando las ecuaciones (6.22)-(6.25) y (6.29)-(6.33) y después de algunas manipulaciones algebraicas tenemos:

$$Y_p(z) = W^p(z)U(z) + W^p(z)O(z) \quad (6.34)$$

$$U(z) = -Q_1(z)I(z)Y_p(z) - Q_2(z)P_i(z)U(z) + Q_3(z)R(z)\{(z-1)I + W^p(z)[I +$$

$$Q2zPiz{-}1Q1(z)TOYp(z){=}Wp(z)I{+}Q2zPiz{-}1Q3z(z{-}1)R(z){+}Wp(z)(z{-}1)O(z)$$

(6.35)

La ecuación 18 muestra que el sistema en lazo cerrado es estable y que el termino integral reduce el error en estado estacionario cuando k tiende al infinito.

6.6 Control Adaptable Indirecto con Termino Integral Usando RNRE's

Las ecuaciones que describen a la RNRE son las siguientes:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ z(k) &= \varphi(x(k)) \\ y(k) &= \varphi[Cz(k)] \end{aligned} \quad (6.36)$$

Donde $x(k)$, $u(k)$ y $y(k)$ son los estados, la entrada y salida de la RNRE respectivamente. A, B y C son las matrices de los parámetros (pesos) de la RNRE, es la función de activación que puede ser tangente hiperbólica, sigmoide, saturación.

Supongamos que φ es la función identidad entonces las ecuaciones (6.36) se transformarán en:

$$x(k+1) = Ax(k) + Bu(k) \quad (6.37)$$

$$y(k) = Cx(k) \quad (6.38)$$

Reescribiendo la ecuación de la salida para *k+1* y sustituyendo en ésta la ecuación para el estado siguiente ecuación (6.37) se tiene:

$$y(k+1) = Cx(k+1) \tag{6.39}$$

$$y(k+1) = CAx(k) + CBu(k) \tag{6.40}$$

Sea $r(k)$ la señal de referencia, entonces se propone la siguiente ley de control R

$$u(k) = [CB]^{-1}\{-CAx(k) + r(k+1) + \gamma[r(k) - y(k)]\} + K_I e_C(k) \tag{6.41}$$

Donde A, B, C, D y $x(k)$ son generadas por la red. Al aplicar esta ley de control al modelo aproximado del sistema se tiene:

$$e_C(k+1) + \gamma e_C(k) = 0 \tag{6.42}$$

Donde $e_C(k) = r(k) - y_p(k)$

En la figura (6.2) se muestra el diagrama de bloques del control indirecto adaptable usando RNRE's. En éste se puede ver que los parámetros y estados generados por la red son proporcionados al controlador adaptable.

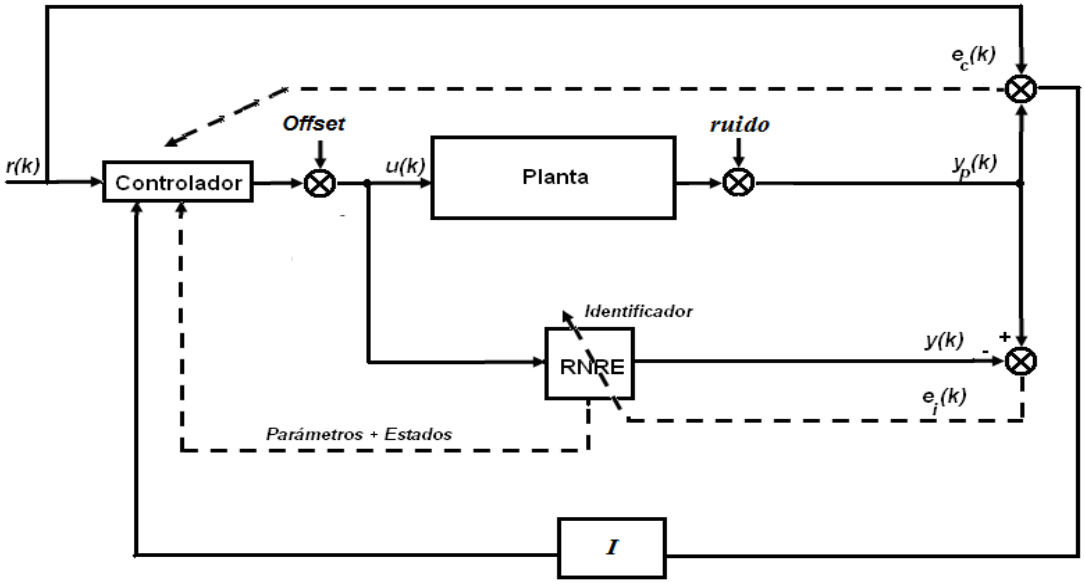


Figura 6.2 Diagrama de bloques de un sistema de control adaptable indirecto.

6.7 Resultados de Simulación.

Para todas las simulaciones de control se utilizó una señal de offset del 20%, se agregó ruido blanco a la señal de salida de la planta para poder verificar las capacidades de filtrado de la RNRE modular de identificación. Los parámetros específicos de la arquitectura de las redes se especifican en cada caso.

6.7.1 Vehículo Espacial

A continuación se presentan los resultados de simulación de los esquemas directo e indirecto de control utilizando los algoritmos de entrenamiento Back-propagation y L-M.

Control neuronal directo más término integral con algoritmo BP para el vehículo espacial

La red utilizada para la identificación presenta el esquema 1-5-1, es decir, una neurona en la primer capa, 5 en la capa oculta y una de salida. La red de control posee una estructura 8-11-1, ya que el vector de entrada se compone del error de control, los estados de la RNRE de identificación, la señal de referencia y el termino integral, la capa oculta posee 11 neuronas y la salida de la red corresponde a la única señal de control del sistema. Las tasas de aprendizaje para el algoritmo BP se tomaron constantes.

Algoritmo BP
ε = 0.0010
σ = 6.1324e-005

Tabla (6.1) Valores de la media y desviación estándar para el algoritmo BP

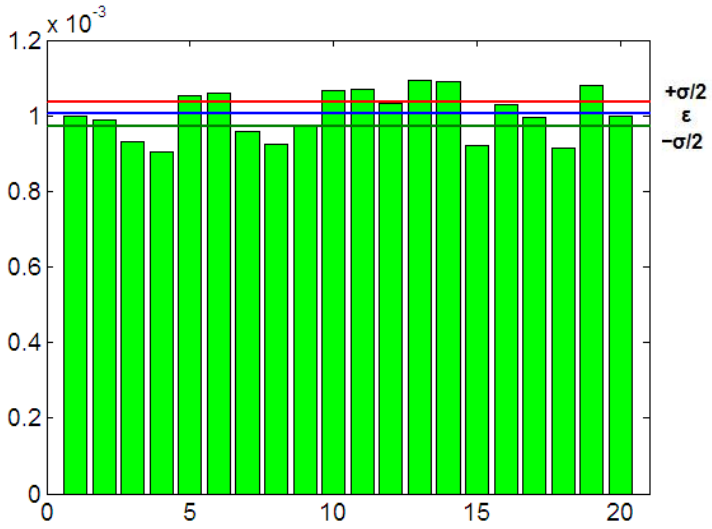


Figura 6.3 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP.

ALGORITMO BP					
No	**1**	**2**	**3**	**4**	**5**
EMC	0.0010	0.0010	0.0009	0.0009	0.0011
No	**6**	**7**	**8**	**9**	**10**
EMC	0.0011	0.0010	0.0009	0.0010	0.0011
No	**11**	**12**	**13**	**14**	**15**
EMC	0.0011	0.0010	0.0011	0.0011	0.0009
No	**16**	**17**	**18**	**19**	**20**
EMC	0.0010	0.0010	0.0009	0.0011	0.0010

Tabla (6.2). Valores del error medio cuadrático para cada simulación, algoritmo BP.

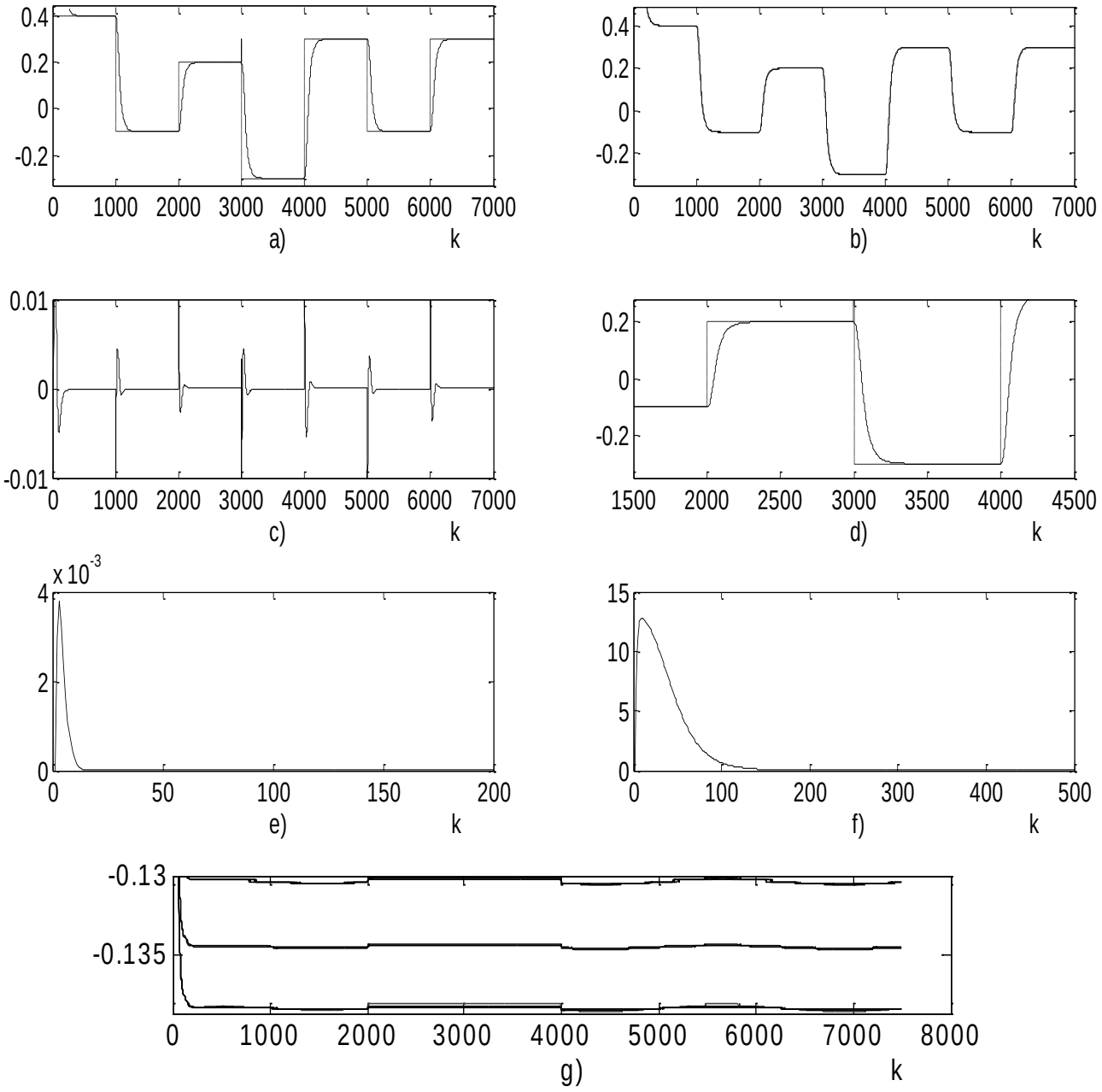


Figura 6.4 Control Neuronal directo mas término integral de un vehículo espacial, utilizando algoritmo BP a) Señal de salida de la planta (continua) y señal de referencia (punteada), b) Salida de la planta (continua) y salida del identificador neuronal, c) Señal de Control d) Zoom de a), e) error medio cuadrático de control, f) error medio cuadrático de identificación g) Estados de la red de identificación

Control neuronal directo sin término integral con algoritmo BP para el vehículo espacial

La red utilizada para la identificación presenta el esquema 1-5-1, es decir, una neurona en la primer capa, 5 en la capa oculta y una de salida. La red de control posee una estructura 7-11-1, ya que el vector de entrada se compone del error de control, los estados de la RNRE de identificación y la señal de referencia, la capa oculta posee 11 neuronas y la salida de la red corresponde a la única señal de control del sistema. Las tasas de aprendizaje para el algoritmo BP se tomaron constantes

Algoritmo BP
ε = 0.020
σ = 1.2758e-003

Tabla (6.3) Valores de la media y desviación estándar para el algoritmo BP

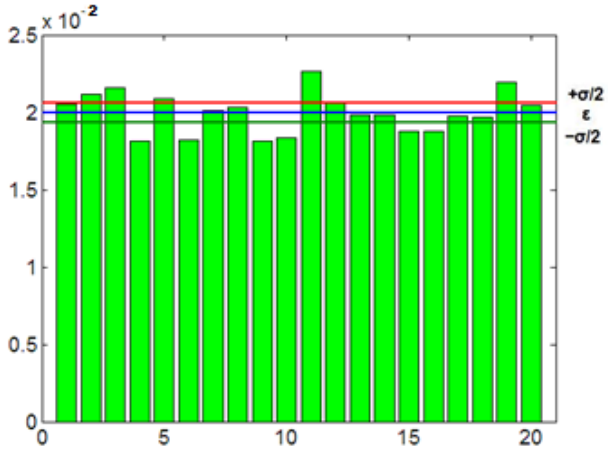


Figura .6.5 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.021	0.021	0.022	0.018	0.021
No	6	7	8	9	10
EMC	0.018	0.020	0.020	0.018	0.018
No	11	12	13	14	15
EMC	0.023	0.021	0.020	0.020	0.019
No	16	17	18	19	20
EMC	0.019	0.020	0.020	0.022	0.020

Tabla (6.4). Valores del error medio cuadrático para cada simulación, algoritmo BP.

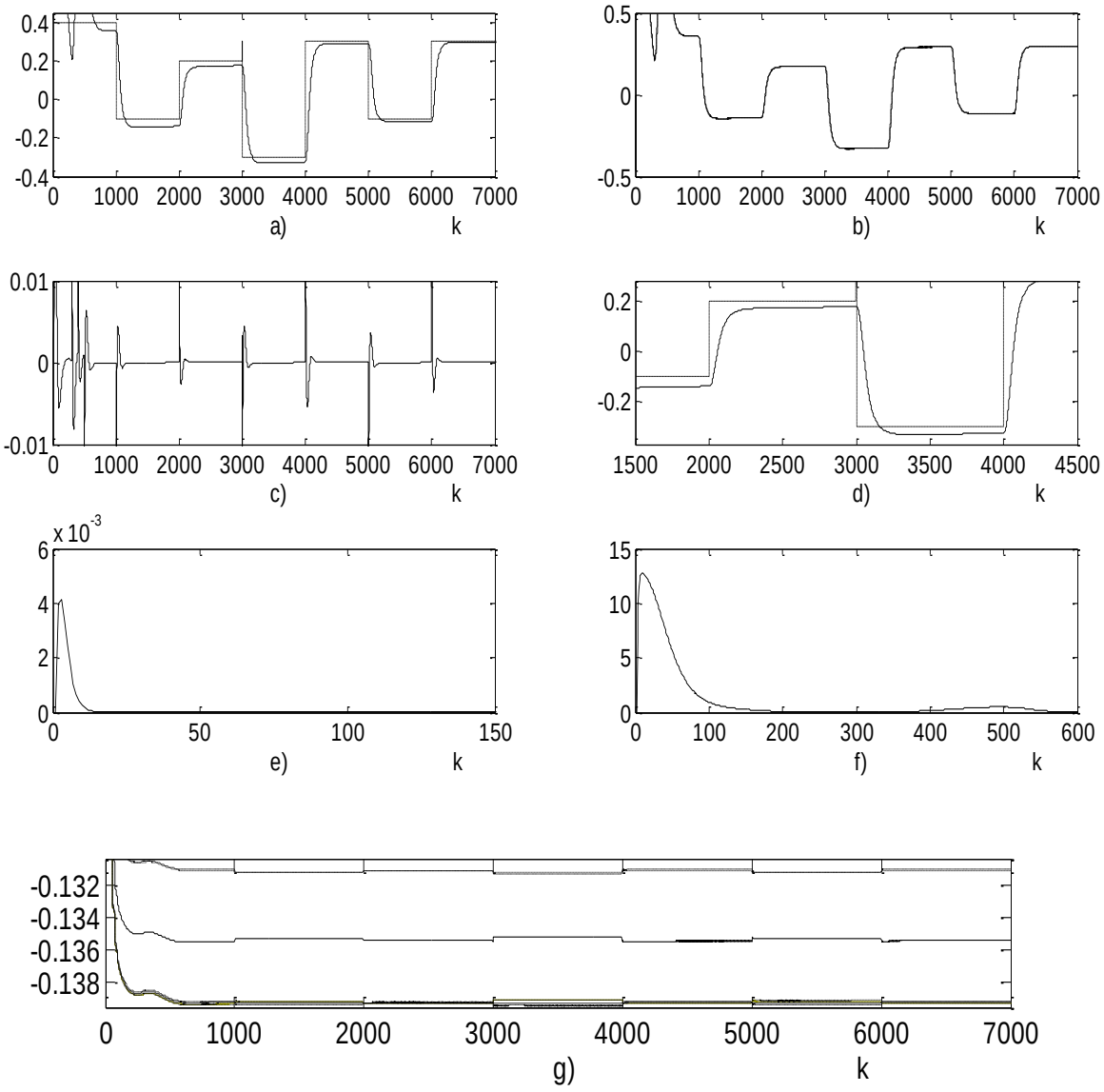


Figura 6.6 Control Neuronal directo sin término integral de un vehículo espacial, utilizando algoritmo BP a) Señal de salida de la planta (continua) y señal de referencia (punteada), b) Salida de la planta (continua) y salida del identificador neuronal, c) Señal de Control d) Zoom de a), e) error medio cuadrático de control, f) error medio cuadrático de identificación g) Estados de la red de identificación

Control directo neuronal sin término Integral con algoritmo L-M para el vehículo espacial

La red utilizada para la identificación presenta el esquema 1-5-1, es decir, una neurona en la primer capa, 5 en la capa oculta y una de salida. La red de control posee una estructura 8-11-1, ya que el vector de entrada se compone del error de control, los estados de la RNRE de identificación y la señal de referencia. La capa oculta posee 11 neuronas y la salida de la red corresponde a la única señal de control del sistema.

Algoritmo L-M
ε = 0.019
σ = 7.4839e-004

Tabla (6.5) Valores de la media y desviación estándar para el algoritmo L-M

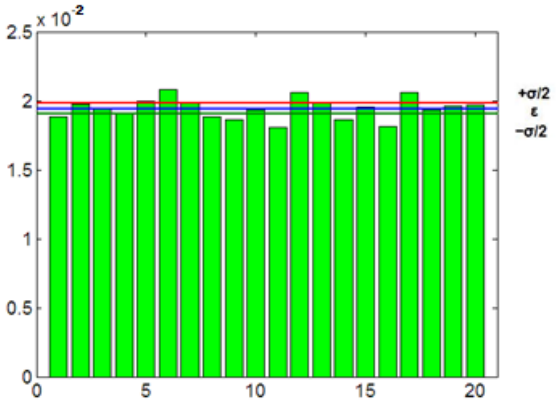


Figura 6.7 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	0.019	0.020	0.019	0.019	0.020
No	6	7	8	9	10
EMC	0.021	0.020	0.019	0.019	0.019
No	11	12	13	14	15
EMC	0.018	0.021	0.020	0.019	0.020
No	16	17	18	19	20
EMC	0.018	0.021	0.019	0.020	0.020

Tabla (6.6). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo L-M.

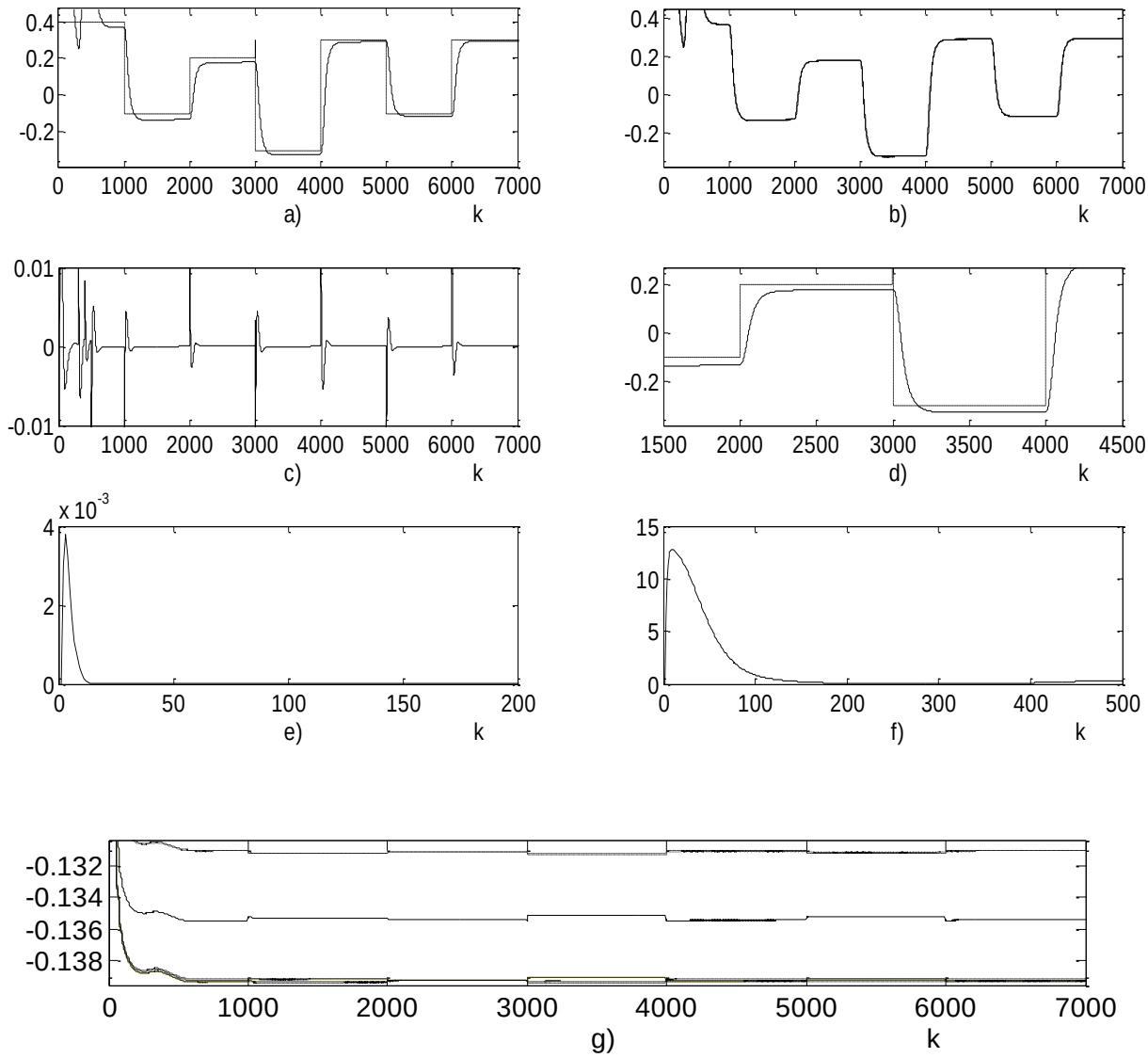


Figura 6.8 Control Neuronal directo sin término integral de un vehículo espacial, utilizando algoritmo L-M a) Señal de salida de la planta (continua) y señal de referencia (punteada), b) Salida de la planta (continua) y salida del identificador neuronal, c) Señal de Control d) Zoom de a), e) error medio cuadrático de control, f) error medio cuadrático de identificación g) Estados de la red.

Control neuronal directo mas término integral algoritmo L-M para el vehículo espacial

La red utilizada para la identificación presenta el esquema 1-5-1, es decir, una neurona en la primer capa, 5 en la capa oculta y una de salida. La red de control posee una estructura 8-11-1, ya que el vector de entrada se compone del error de control, los estados de la RNRE de identificación, termino integral y la señal de referencia. La capa oculta posee 11 neuronas y la salida de la red corresponde a la única señal de control del sistema.

Algoritmo L-M
ε = 7.4427e-004
σ = 2.9212e-005

Tabla (6.7) Valores de la media y desviación estándar para el algoritmo L-M

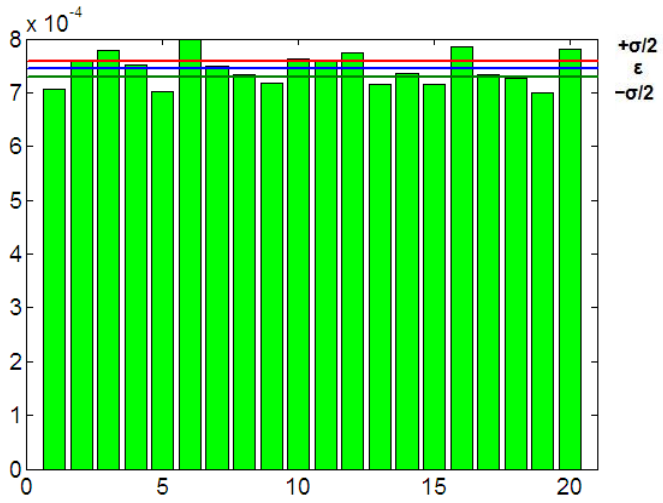


Figura 6.9 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	0.7054 E-03	0.7609 E-03	0.7777 E-03	0.7511 E-03	0.7028 E-03
No	6	7	8	9	10
EMC	0.7990 E-03	0.7501 E-03	0.7332 E-03	0.7174 E-03	0.7626 E-03
No	11	12	13	14	15
EMC	0.7575 E-03	0.7751 E-03	0.7154 E-03	0.7357 E-03	0.7144 E-03
No	16	17	18	19	20
EMC	0.7851 E-03	0.7338 E-03	0.7275 E-03	0.7006 E-03	0.7802 E-03

Tabla (7.8). Valores del error medio cuadrático para cada simulación, algoritmo L-M.

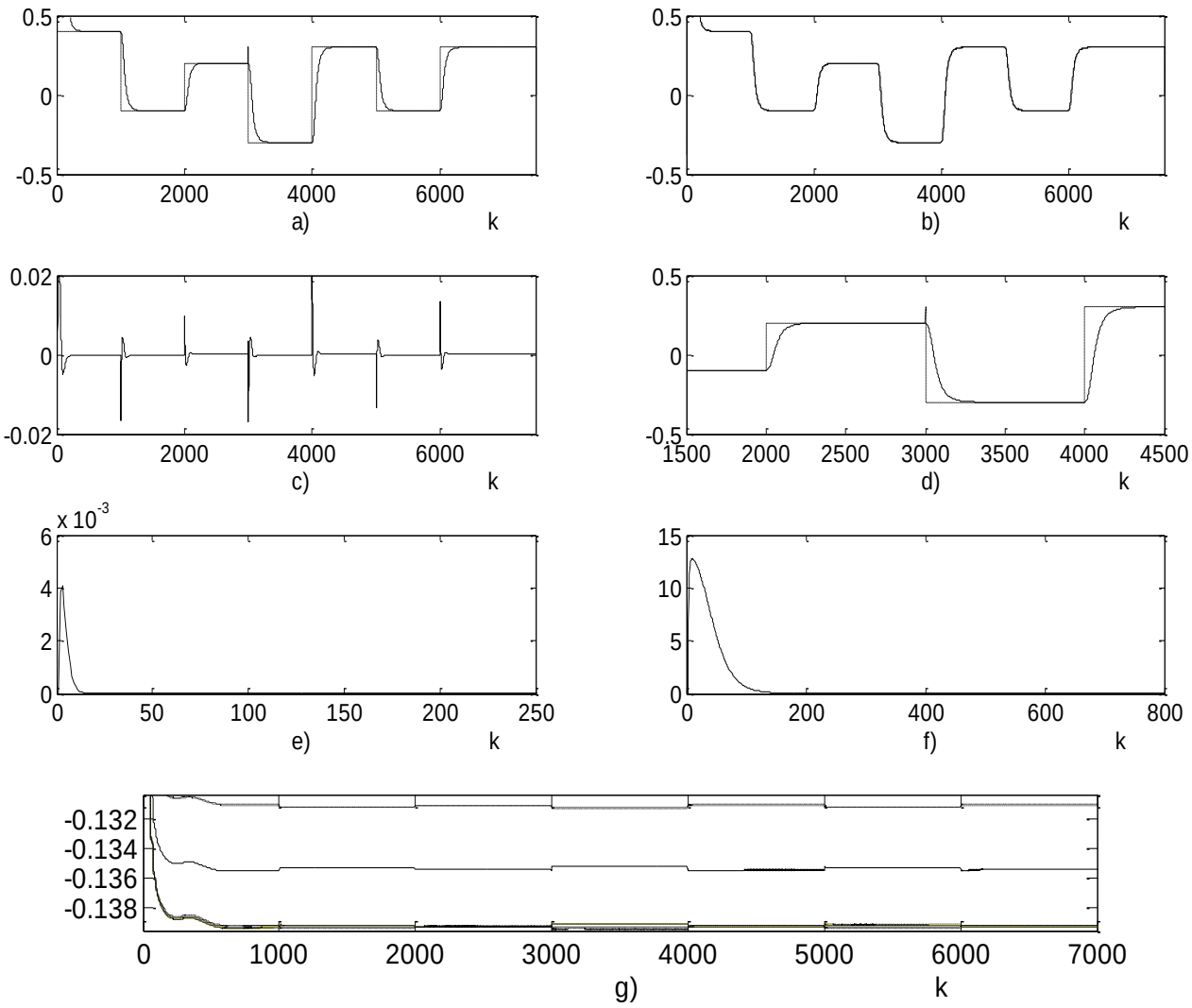


Figura 6.10 Control Neuronal directo mas término integral de un vehículo espacial, utilizando algoritmo L-M a) Señal de salida de la planta (continua) y señal de referencia (punteada), b) Salida de la planta (continua) y salida del identificador neuronal, c) Señal de Control d) Zoom de a), e) error medio cuadrático de control, f) error medio cuadrático de identificación g) Estados de la red.

Control neuronal indirecto con término integral algoritmo L-M para el vehículo espacial

La red utilizada para la identificación presenta el esquema 1-5-1, es decir, una neurona en la primer capa, 5 en la capa oculta y una de salida. EL control se realizo utilizando el método de modos deslizantes con termino integral descrito en este capítulo.

Algoritmo L-M
ε = 1.2828e-004
σ = 4.7086e-006

Tabla (6.9) Valores de la media y desviación estándar para el algoritmo L-M

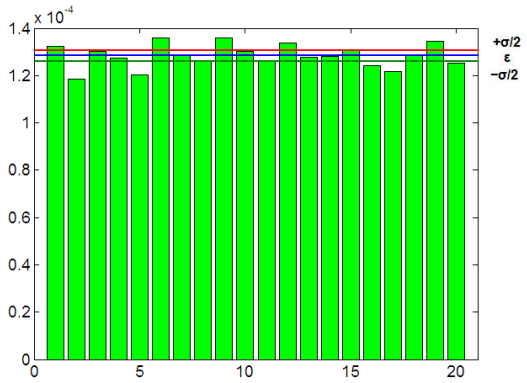


Figura 6.11 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	0.1323 E-03	0.1187 E-03	0.1302 E-03	0.1273 E-03	0.1204 E-03
No	6	7	8	9	10
EMC	0.1358 E-03	0.1288 E-03	0.1258 E-03	0.1358 E-03	0.1301 E-03
No	11	12	13	14	15
EMC	0.1260 E-03	0.1338 E-03	0.1277 E-03	0.1281 E-03	0.1306 E-03
No	16	17	18	19	20
EMC	0.1243 E-03	0.1218 E-03	0.1286 E-03	0.1343 E-03	0.1251 E-03

Tabla (6.10). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo L-M.

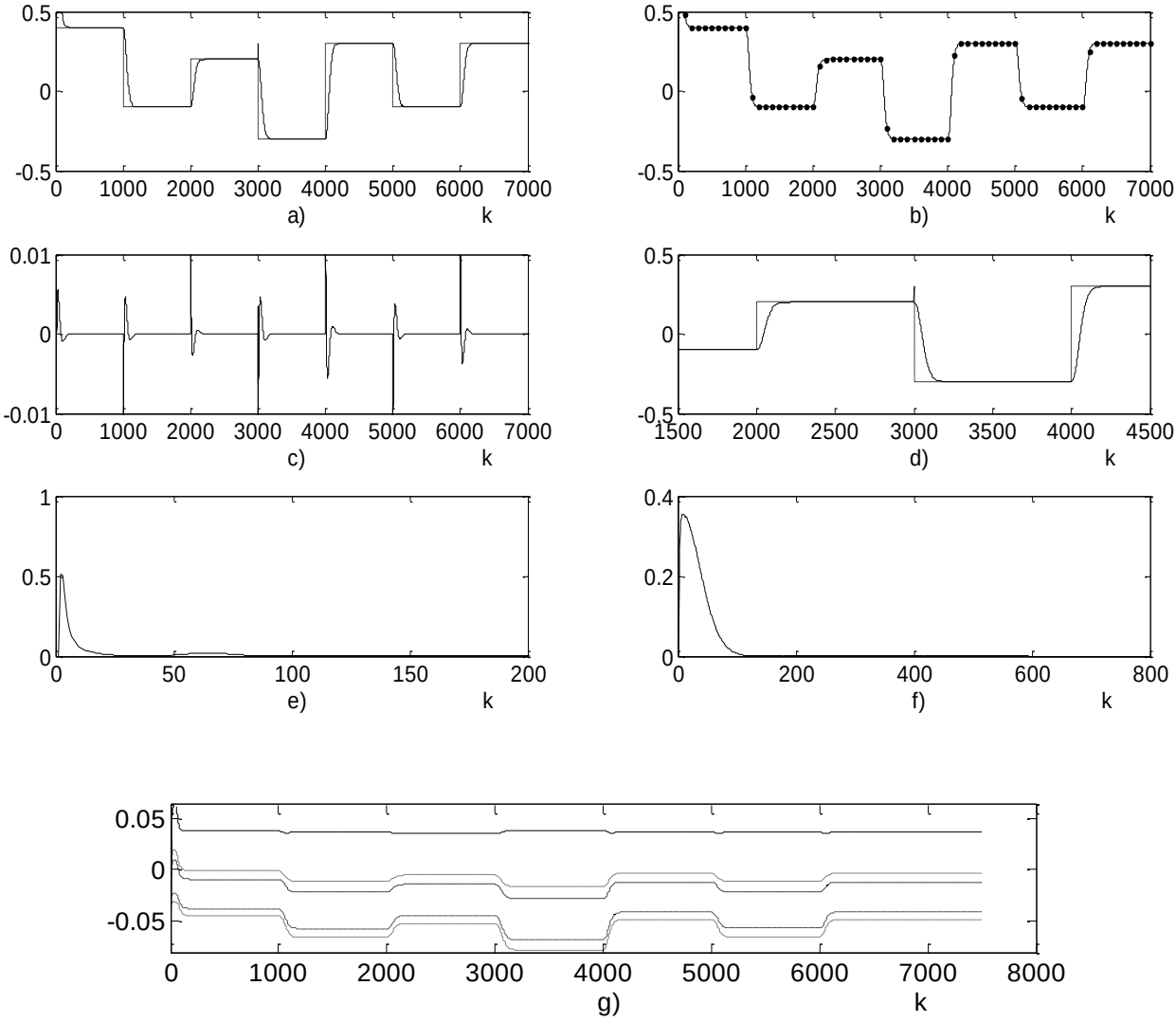


Figura 6.12 Control Neuronal indirecto mas término integral de un vehículo espacial, utilizando algoritmo L-M a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Salida de la planta (continua) y salida del identificador neuronal, c) Señal de Control d) Zoom de a), e) error medio cuadrático de control, f) error medio cuadrático de identificación g) Estados de la red.

Control neuronal indirecto sin término integral algoritmo L-M para el vehículo espacial

La red utilizada para la identificación presenta el esquema 1-5-1, es decir, una neurona en la primer capa, 5 en la capa oculta y una de salida. EL control se realizo utilizando el método de modos deslizantes.

Algoritmo L-M
ε = 0.030
σ = 1.0265e-003

Tabla (6.11) Valores de la media y desviación estándar para el algoritmo L-M

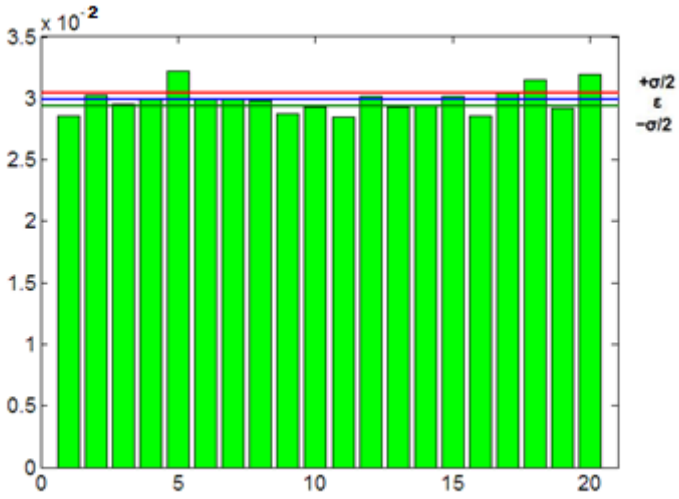


Figura 6.13 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	0.029	0.030	0.030	0.030	0.032
No	6	7	8	9	10
EMC	0.030	0.030	0.030	0.029	0.029
No	11	12	13	14	15
EMC	0.028	0.030	0.029	0.029	0.030
No	16	17	18	19	20
EMC	0.029	0.030	0.032	0.029	0.032

Tabla (6.12). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo L-M.

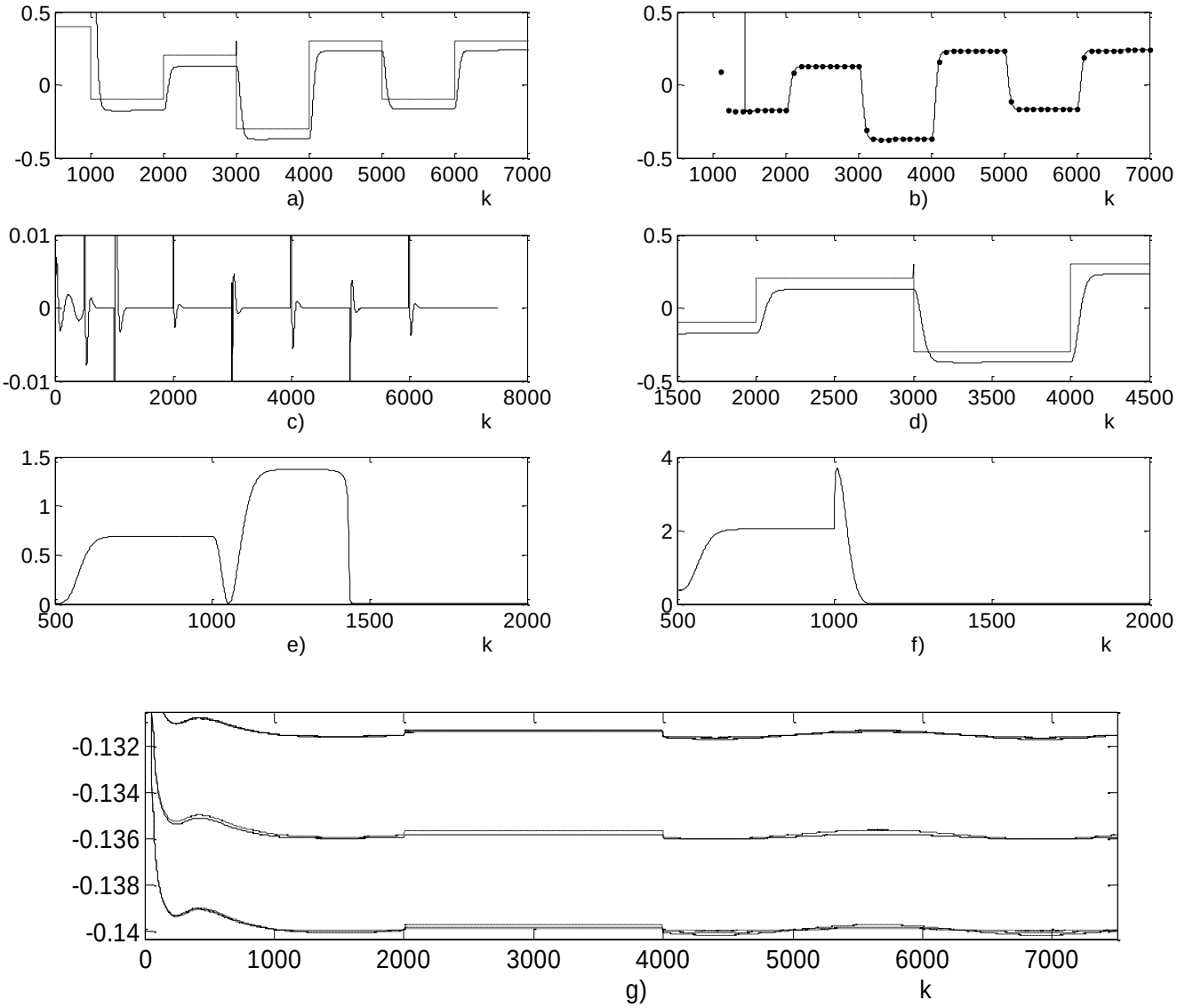


Figura 6.14 Control Neuronal indirecto sin término integral de un vehículo espacial, utilizando algoritmo L-M a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Salida de la planta (continua) y salida del identificador neuronal, c) Señal de Control d) Zoom de a), e) error medio cuadrático de control, f) error medio cuadrático de identificación g) Estados de la red.

Control neuronal indirecto mas término integral con algoritmo BP para el vehículo espacial

La red utilizada para la identificación presenta el esquema 1-5-1, es decir, una neurona en la primer capa, 5 en la capa oculta y una de salida. EL control se realizo utilizando el método de modos deslizantes con termino integral.

Algoritmo BP
ε = 1.8388e-004
σ = 1.3231e-005

Tabla (6.13) Valores de la media y desviación estándar para el algoritmo BP

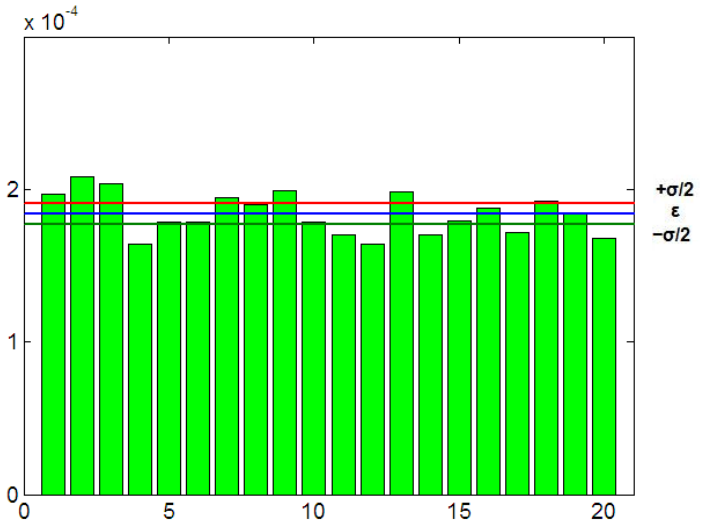


Figura 6.15 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP.

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.1967 E-03	0.2085 E-03	0.2033 E-03	0.1643 E-03	0.1783 E-03
No	6	7	8	9	10
EMC	0.1785 E-03	0.1943 E-03	0.1899 E-03	0.1995 E-03	0.1784 E-03
No	11	12	13	14	15
EMC	0.1703 E-03	0.1643 E-03	0.1986 E-03	0.1703 E-03	0.1794 E-03
No	16	17	18	19	20
EMC	0.1876 E-03	0.1714 E-03	0.1921 E-03	0.1842 E-03	0.1676 E-03

Tabla (6.14). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo BP

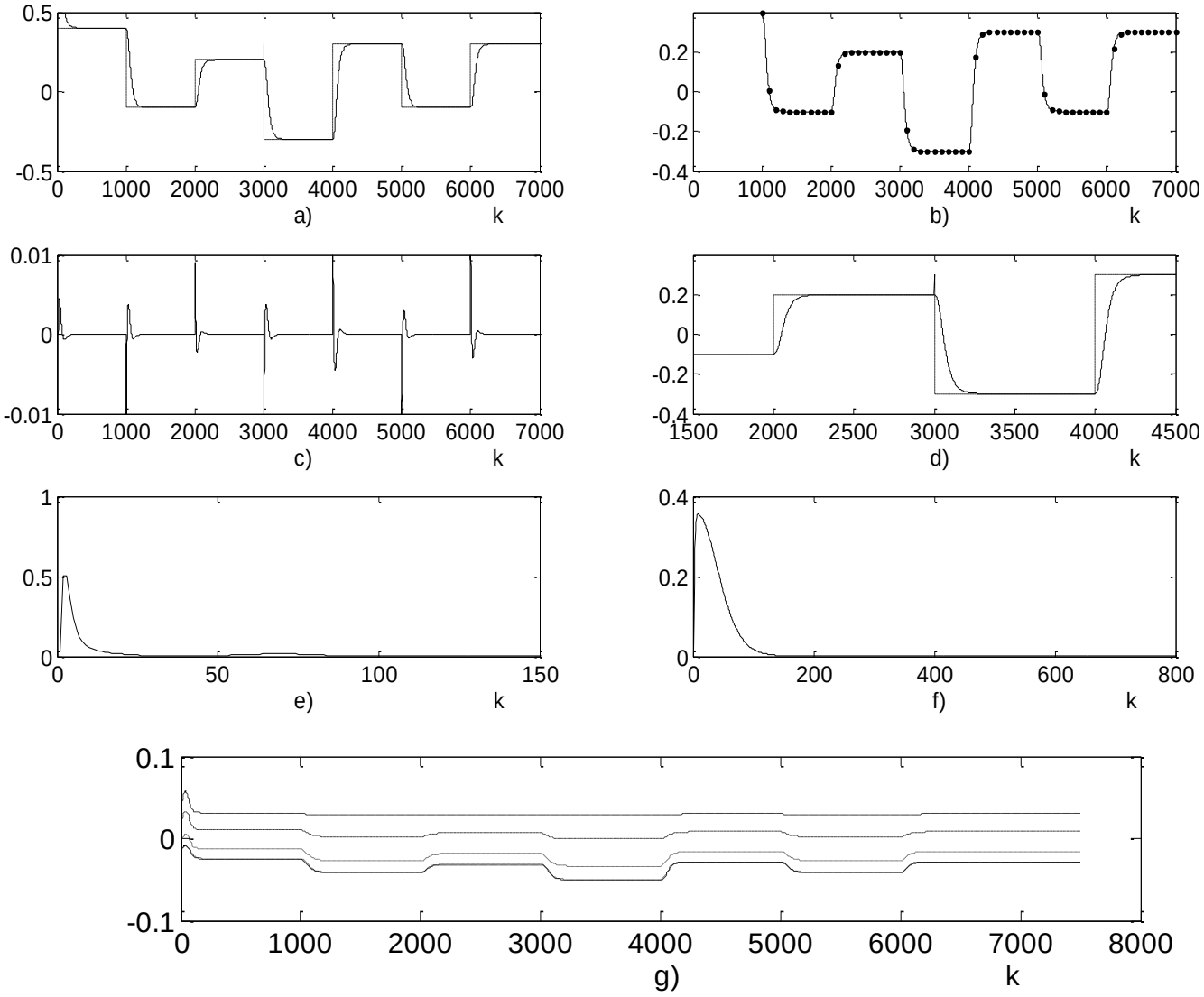


Figura 6.16 Control Neuronal indirecto mas término integral de un vehículo espacial, utilizando algoritmo BP a) Señal de salida de la planta (continua) y señal de referencia (punteada), b) Salida de la planta (continua) y salida del identificador neuronal, c) Señal de Control d) Zoom de a), e) error medio cuadrático de control, f) error medio cuadrático de identificación g) Estados de la red.

Control neuronal indirecto sin término integral algoritmo BP para el vehículo espacial

La red utilizada para la identificación presenta el esquema 1-5-1, es decir, una neurona en la primer capa, 5 en la capa oculta y una de salida. EL control se realizo utilizando el método de modos deslizantes con termino integral y usando como algoritmo de aprendizaje el Back-propagation.

Algoritmo BP
ε =0.040
Σ= 1.3599e-003

Tabla (6.15) Valores de la media y desviación estándar para el algoritmo BP

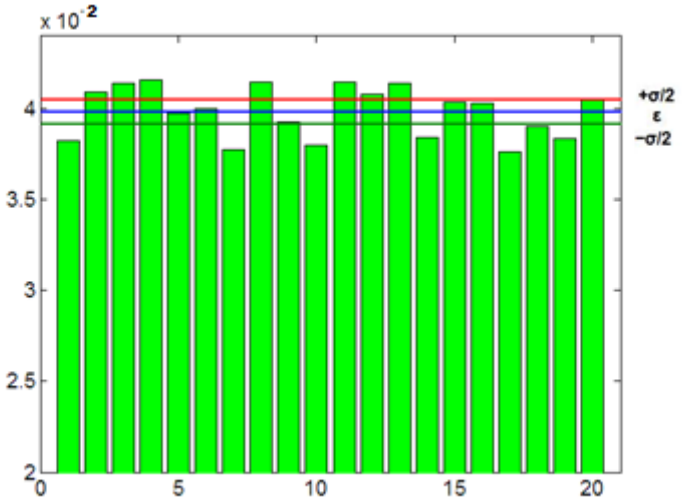


Figura 6.17 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP.

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.038	0.041	0.041	0.042	0.040
No	6	7	8	9	10
EMC	0.040	0.038	0.041	0.039	0.038
No	11	12	13	14	15
EMC	0.041	0.041	0.041	0.038	0.040
No	16	17	18	19	20
EMC	0.040	0.038	0.039	0.038	0.041

Tabla (6.16). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo BP

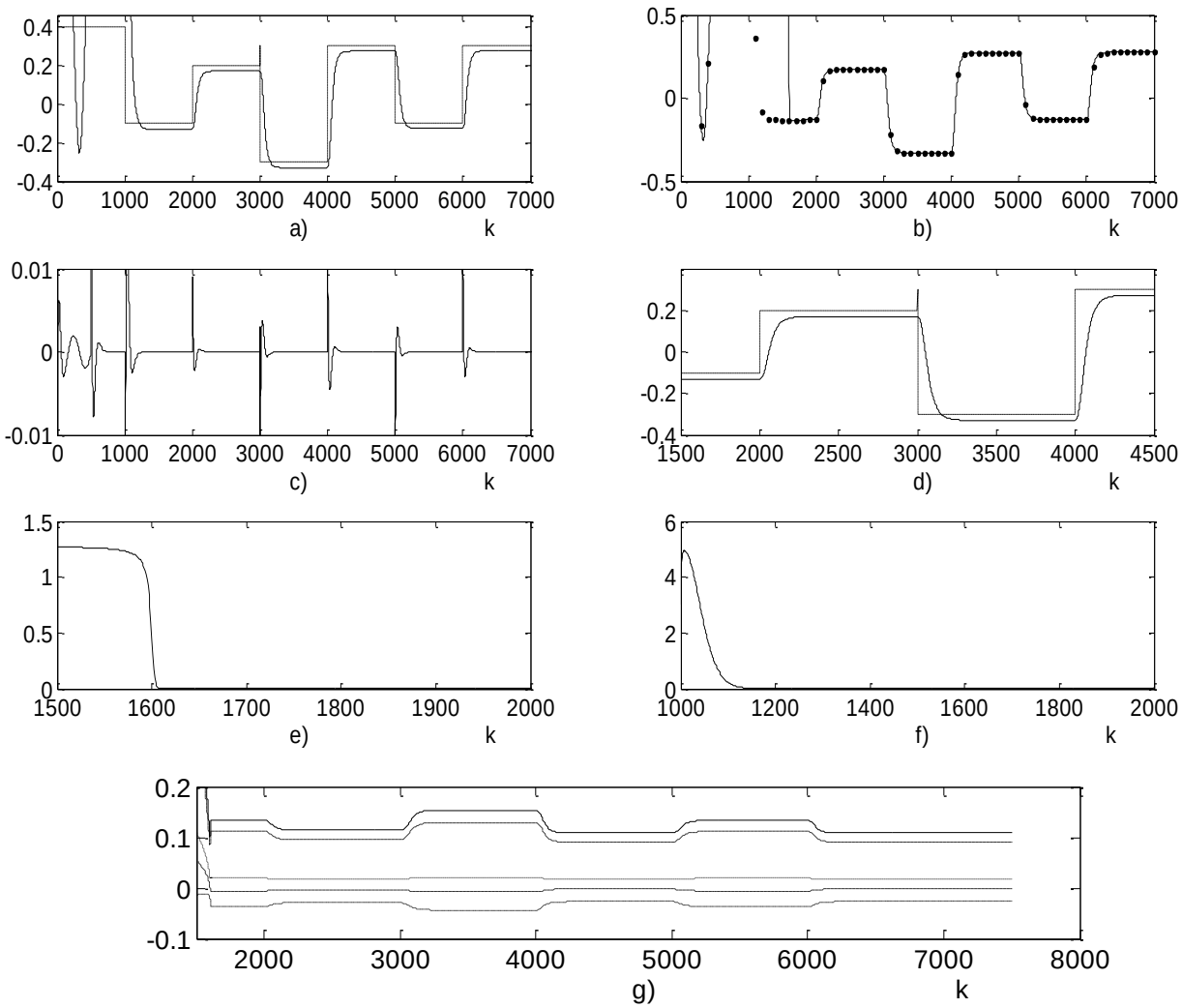


Figura 6.18 Control Neuronal indirecto sin término integral de un vehículo espacial, utilizando algoritmo BP a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Salida de la planta (continua) y salida del identificador neuronal, c) Señal de Control d) Zoom de a), e) error medio cuadrático de control, f) error medio cuadrático de identificación g) Estados de la red.

La tabla 6.17 muestra un resumen del desempeño de cada algoritmo y esquema de control con base en la media y varianza del error de control, donde el control neuronal indirecto presentó los mejores resultados.

CONTROL	Algoritmo	MEDIA	VARIANZA
Control neuronal directo con término integral	BP	0.0010	6.1324e-005
Control neuronal directo sin término integral	BP	0.020	1.2758e-003
Control neuronal indirecto con término integral	BP	1.8388e-004	1.3231e-005
Control neuronal indirecto sin termino integral	BP	0.040	1.3599e-003
Control neuronal directo con termino integral	L-M	7.4427e-004	2.9212e-005
Control neuronal directo sin termino integral	L-M	0.019	7.4839e-004
Control neuronal indirecto con termino integral	L-M	1.2828e-004	4.7086e-006
Control neuronal indirecto sin termino integral	L-M	0.030	1.0265e-003

Tabla (6.17) Comparación de las medias y varianzas de 20 corridas con distintos algoritmos de control

De acuerdo a la tabla 6.17 y centrándose en el control con algoritmo Backpropagation, se puede observar que con el control neuronal directo sin término integral se obtiene una media del error menor con respecto al control neuronal indirecto sin término integral. La incorporación del término integral mejora en un 95% el error medio para el control neuronal directo, así como una mejora de 99% del error medio para el control neuronal indirecto. Aunque si comparamos ambos controladores con término integral, el que mejor resultados obtuvo el control indirecto con término integral; la varianza mejora en la misma proporción que la media del error para todos los controladores. Para los mismos controladores pero entrenados con el algoritmo de Levenberg-Marquardt se tiene una mejora del 25% en promedio para cada uno de los controladores, tanto para la media como para la varianza.

6.7.2 Motor de Combustión interna

A continuación se presentan los resultados de control obtenidos por medio de simulación para la planta del motor de combustión interna funcionando en ralentí.

Control neuronal directo más término integral con algoritmo BP

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida. La red de control posee un esquema 11-13-2, con 11 neuronas de entrada compuestas por el error de control, los estados de la red de identificación, el vector de referencia y los términos integrales, 13 neuronas en la capa oculta y 2 neuronas que representan la salida de control. Ambas redes fueron entrenadas utilizando el algoritmo BP.

Algoritmo BP
ε = 4.3079e-005
σ = 1.5128e-006

Tabla (6.18) Valores de la media y desviación estándar para el algoritmo BP

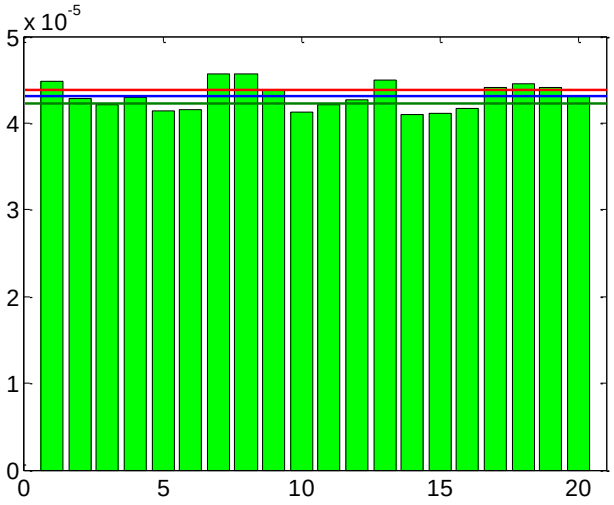


Figura 6.19 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP.

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.4481E-04	0.4285E-04	0.4211E-04	4.293-E05	4.139E-05
No	6	7	8	9	10
EMC	4.157E-05	4.562E-05	4.569E-05	4.378E-05	4.120E-05
No	11	12	13	14	15
EMC	4.208E-05	4.267E-05	4.501E-05	4.098E-05	4.112E-05
No	16	17	18	19	20
EMC	4.175E-05	4.415E-05	4.456-05	4.4414E-05	4.316E-05

Tabla (6.19). Valores del error medio cuadrático para cada simulación, algoritmo BP.

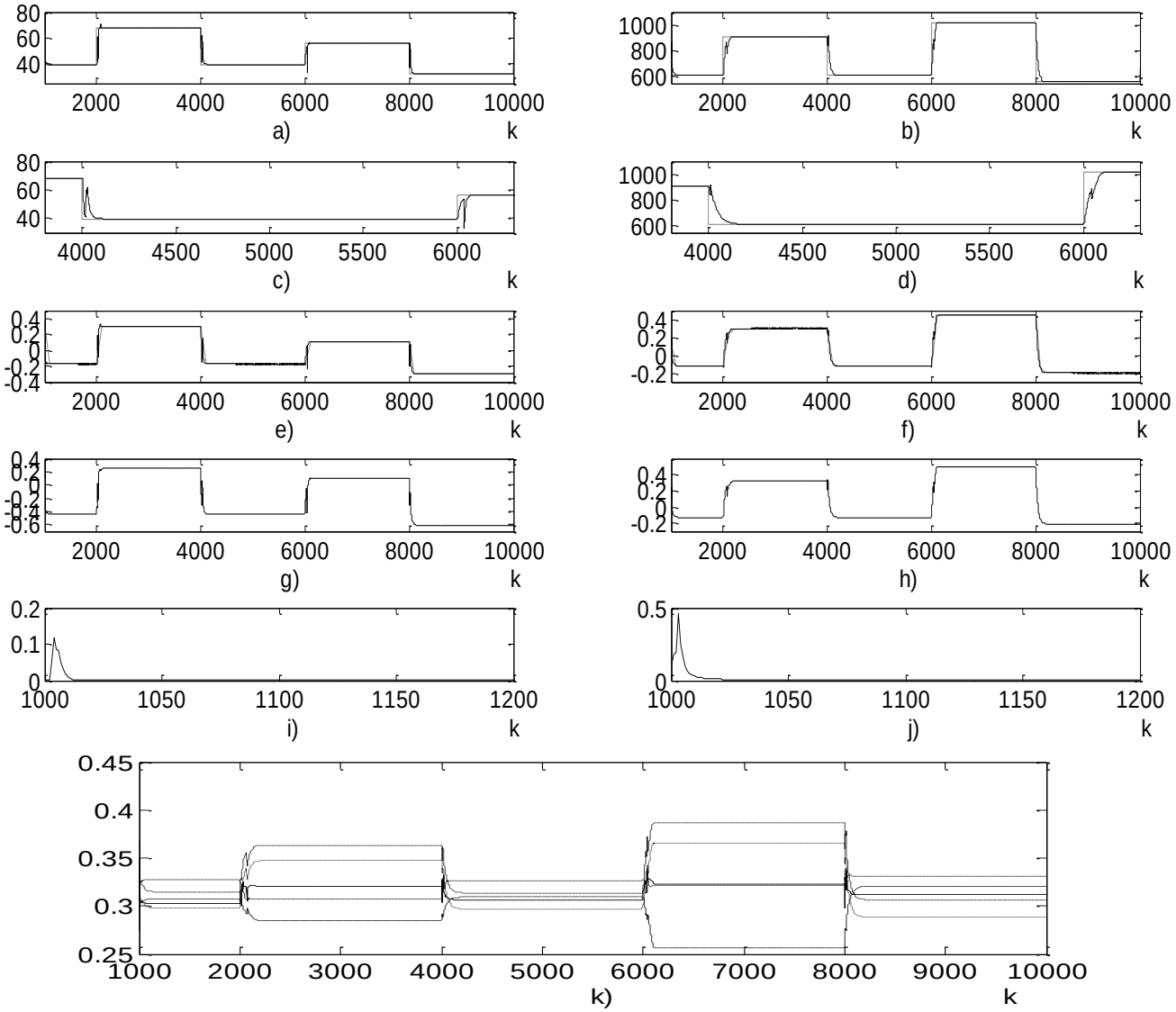
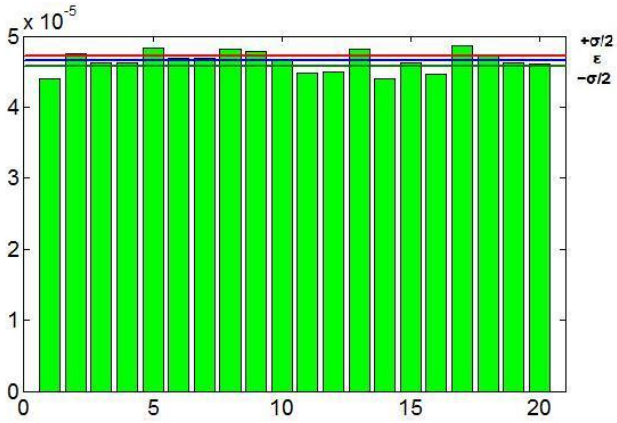


Figura 6.20 Control Neuronal directo mas término integral de un motor de combustión interna, utilizando algoritmo BP a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Segunda señal de salida de la planta(continua) y segunda señal de referencia (punteada), c) Acercamiento a la grafica a) d) acercamiento a una sección de b) e)Salida de la planta (continua) y salida del identificador neuronal, f) Salida de la planta (continua) y salida del identificador neuronal, g) y h) Señales de Control i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la red de identificación.

Control neuronal directo sin término integral con algoritmo BP

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida. La red de control posee un esquema 9-13-2, con 9 neuronas de entrada compuestas por el error de control, los estados de la red de identificación y, el vector de referencia, 13 neuronas en la capa oculta y 2 neuronas que representan la salida de control. Ambas redes fueron entrenadas utilizando el algoritmo BP.



Algoritmo BP
ε = 4.6495e-005
σ = 1.4110e-006

Tabla (6.20) Valores de la media y desviación estándar para el algoritmo BP

Figura 6.21 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP.

ALGORITMO BP					
No	1	2	3	4	5
EMC	4.395E-05	4.752E-05	4.630E-05	4.620-E05	4.832E-05
No	6	7	8	9	10
EMC	4.685E-05	4.689E-05	4.810E-05	4.783E-05	4.668E-05
No	11	12	13	14	15
EMC	4.471E-05	4.500E-05	4.823E-05	4.394E-05	4.625E-05
No	16	17	18	19	20
EMC	4.464E-05	4.869E-05	4.736-05	4.630E-05	4.615E-05

Tabla (6.21). Valores del error medio cuadrático para cada simulación, algoritmo BP.

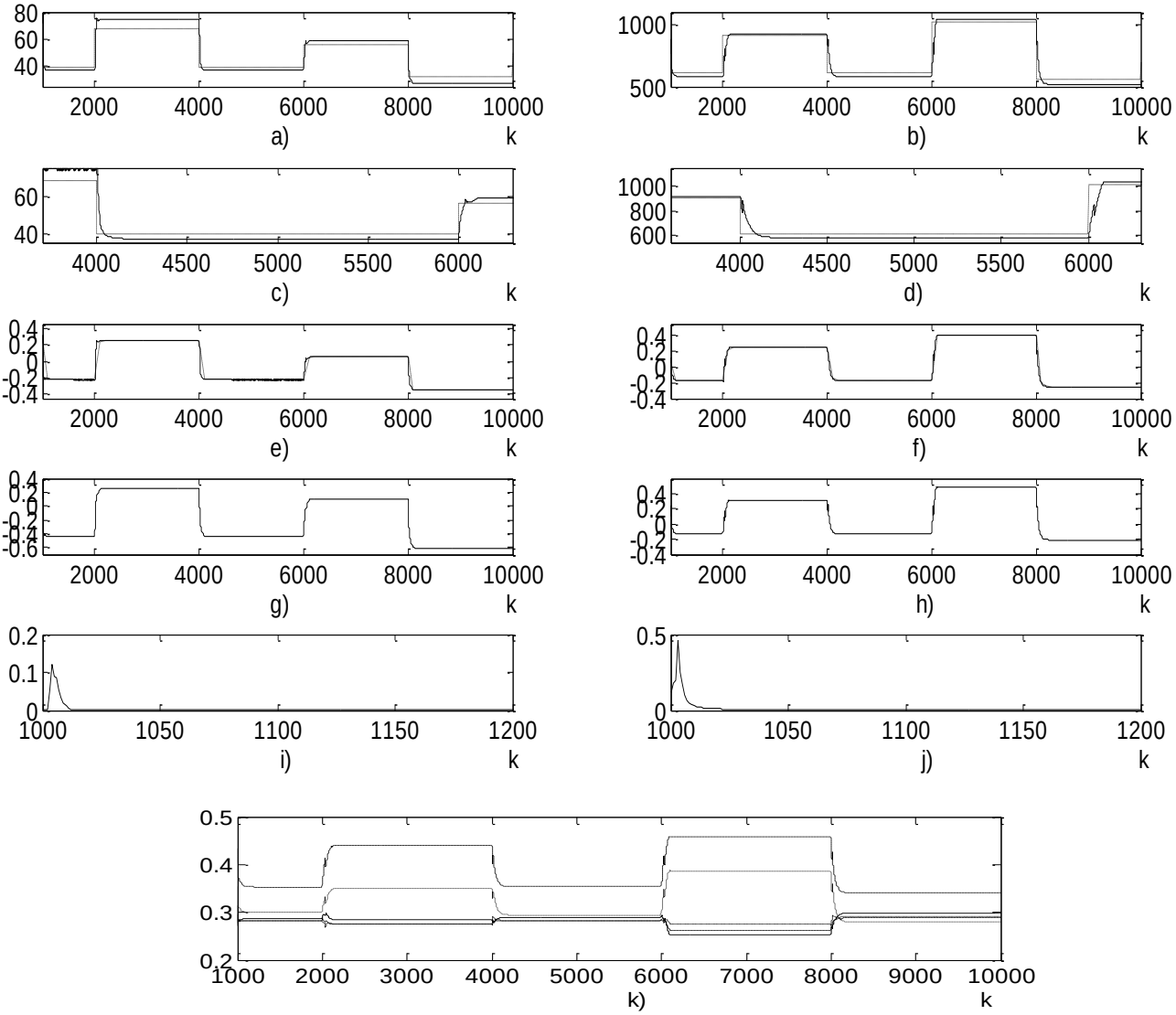


Figura 6.22 Control Neuronal directo sin término integral de un motor de combustión interna, utilizando algoritmo BP a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Segunda señal de salida de la planta(continua) y segunda señal de referencia (punteada), c) Acercamiento a la grafica a) d) acercamiento a una sección de b) e)Salida de la planta (continua) y salida del identificador neuronal, f) Salida de la planta (continua) y salida del identificador neuronal, g) y h) Señales de Control i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la red de identificación.

Control directo neuronal sin término Integral algoritmo L-M

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida. La red de control posee un esquema 9-13-2, con 9 neuronas de entrada compuestas por el error de control, los estados de la red de identificación y el vector de referencia , 13 neuronas en la capa oculta y 2 neuronas que representan la salida de control. Ambas redes fueron entrenadas utilizando el algoritmo LM.

Algoritmo L-M
ε = 4.6358e-005
σ = 1.1309e-006

Tabla (6.22) Valores de la media y desviación estándar para el algoritmo L-M

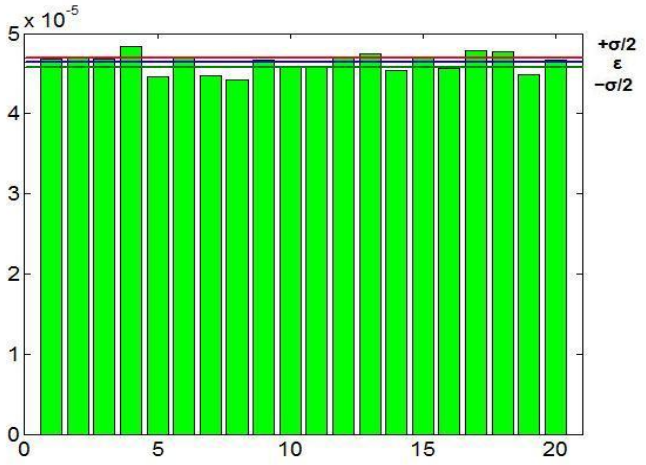


Figura 6.23 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	4.682E-05	4.698E-05	4.676E-05	4.831-E05	4.463E-05
No	6	7	8	9	10
EMC	4.713E-05	4.476E-05	4.418E-05	4.662E-05	4.583E-05
No	11	12	13	14	15
EMC	4.588E-05	4.689E-05	4.743E-05	4.533E-05	4.689E-05
No	16	17	18	19	20
EMC	4.566E-05	4.779E-05	4.775-05	4.486E-05	4.665E-05

Tabla (6.23). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo L-M.

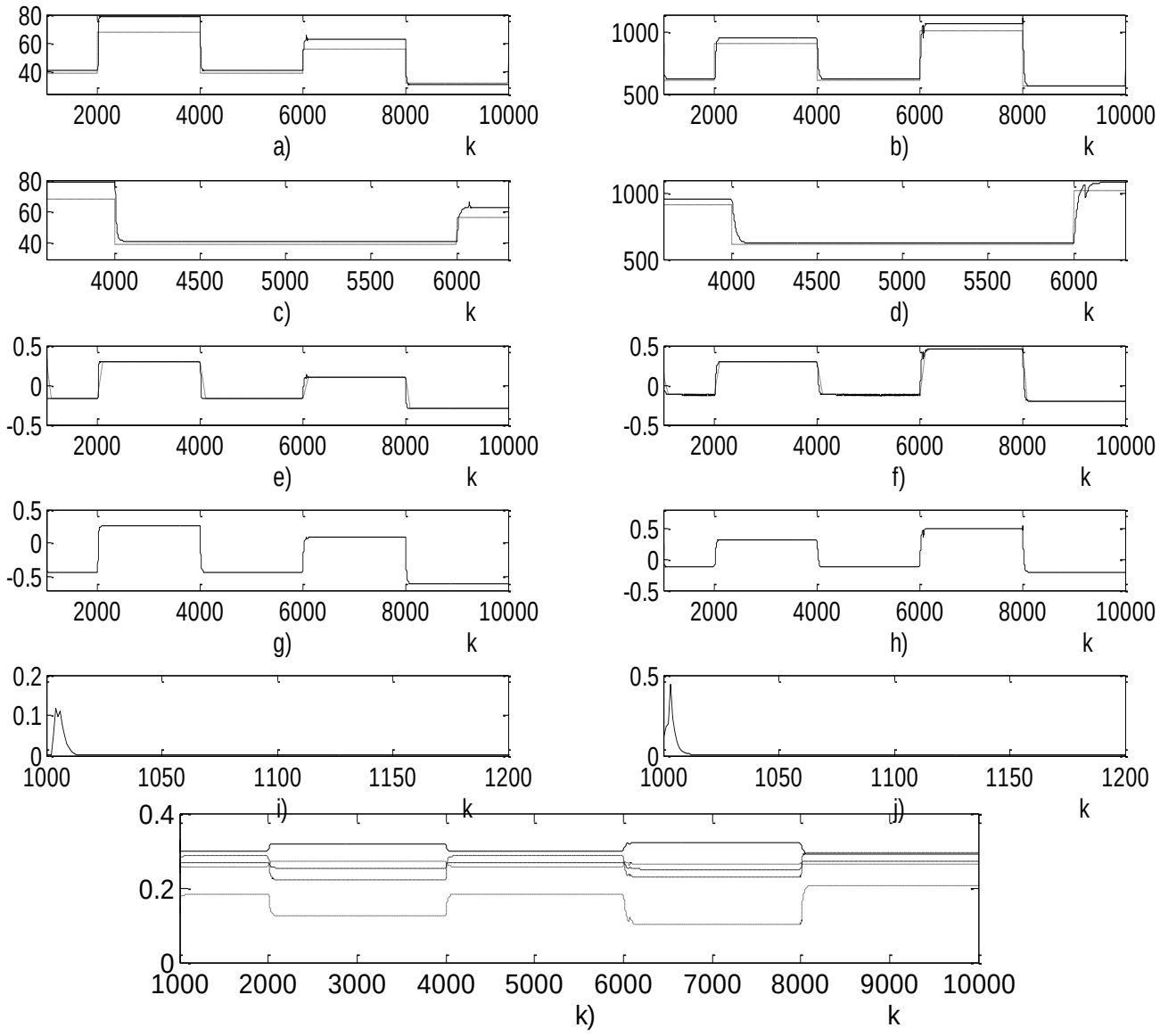


Figura 6.24 Control Neuronal directo sin término integral de un motor de combustión interna, utilizando algoritmo L-M a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Segunda señal de salida de la planta(continua) y segunda señal de referencia (punteada), c) Acercamiento a la grafica a) d) acercamiento a una sección de b) e)Salida de la planta (continua) y salida del identificador neuronal, f) Salida de la planta (continua) y salida del identificador neuronal, g) y h) Señales de Control i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la red de identificación.

Control neuronal directo mas término integral algoritmo L-M

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida. La red de control posee un esquema 11-13-2, con 11 neuronas de entrada compuestas por el error de control, los estados de la red de identificación, el vector de referencia y los términos integrales, 13 neuronas en la capa oculta y 2 neuronas que representan la salida de control. Ambas redes fueron entrenadas utilizando el algoritmo LM.

Algoritmo L-M
ε = 3.9337e-005
σ = 1.0644e-006

Tabla (6.24) Valores de la media y desviación estándar para el algoritmo L-M

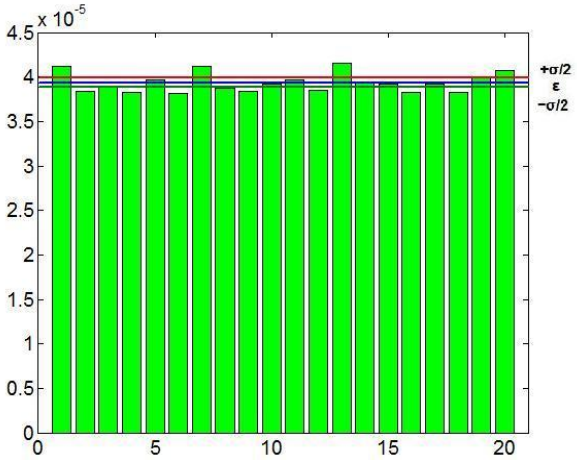


Figura 6.25 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	4.123E-05	3.835E-05	3.895E-05	3.829-E05	3.965E-05
No	6	7	8	9	10
EMC	3.815E-05	4.124E-05	3.879E-05	3.843E-05	3.918E-05
No	11	12	13	14	15
EMC	3.970E-05	3.384E-05	4.158E-05	3.926E-05	3.914E-05
No	16	17	18	19	20
EMC	3.832E-05	3.914E-05	3.826-05	3.987E-05	4.076E-05

Tabla (6.25). Valores del error medio cuadrático para cada simulación, algoritmo L-M.

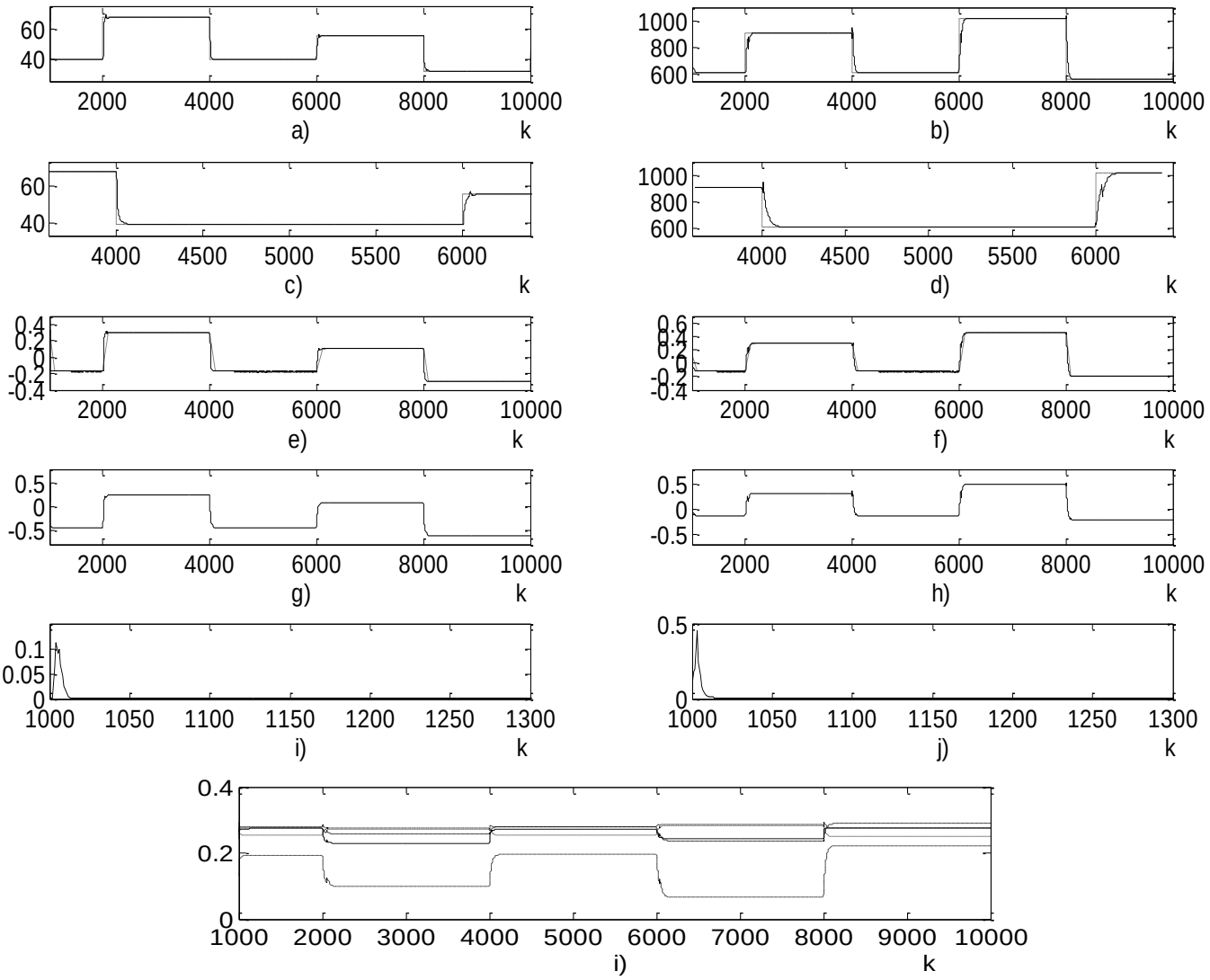


Figura 6.26 Control Neuronal directo mas término integral de un motor de combustión interna, utilizando algoritmo L-M a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Segunda señal de salida de la planta(continua) y segunda señal de referencia (punteada), c) Acercamiento a la grafica a) d) acercamiento a una sección de b) e)Salida de la planta (continua) y salida del identificador neuronal, f) Salida de la planta (continua) y salida del identificador neuronal, g) y h) Señales de Control i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la red de identificación.

Control neuronal indirecto con término integral algoritmo L-M

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida, utilizando el algoritmo L-M para el aprendizaje. El control se realiza utilizando un controlador por modos deslizantes con termino integral.

Algoritmo L-M
ε = 4.0768e-006
σ = 6.5208e-008

Tabla (6.26) Valores de la media y desviación estándar para el algoritmo L-M

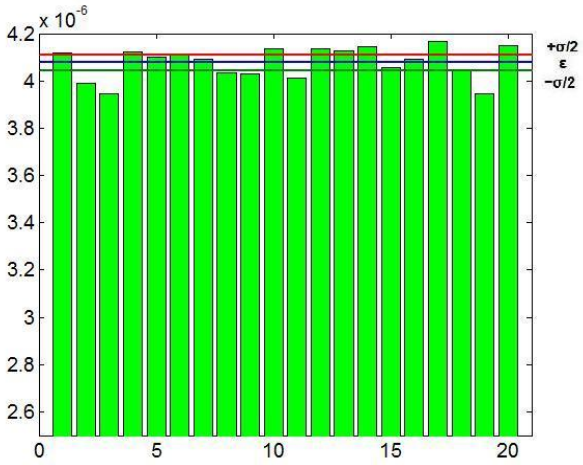


Figura 6.27 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	4.118E-6	3.988E-06	3.947E-06	4.122-E06	4.098E-06
No	6	7	8	9	10
EMC	4.109E-06	4.091E-06	4.035E-06	4.028E-06	4.135E-06
No	11	12	13	14	15
EMC	4.010E-06	4.134E-06	4.128E-06	4.144E-06	4.057E-06
No	16	17	18	19	20
EMC	4.089E-06	4.168E-06	4.041-06	3.946E-06	4.147E-06

Tabla (6.27). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo L-M.

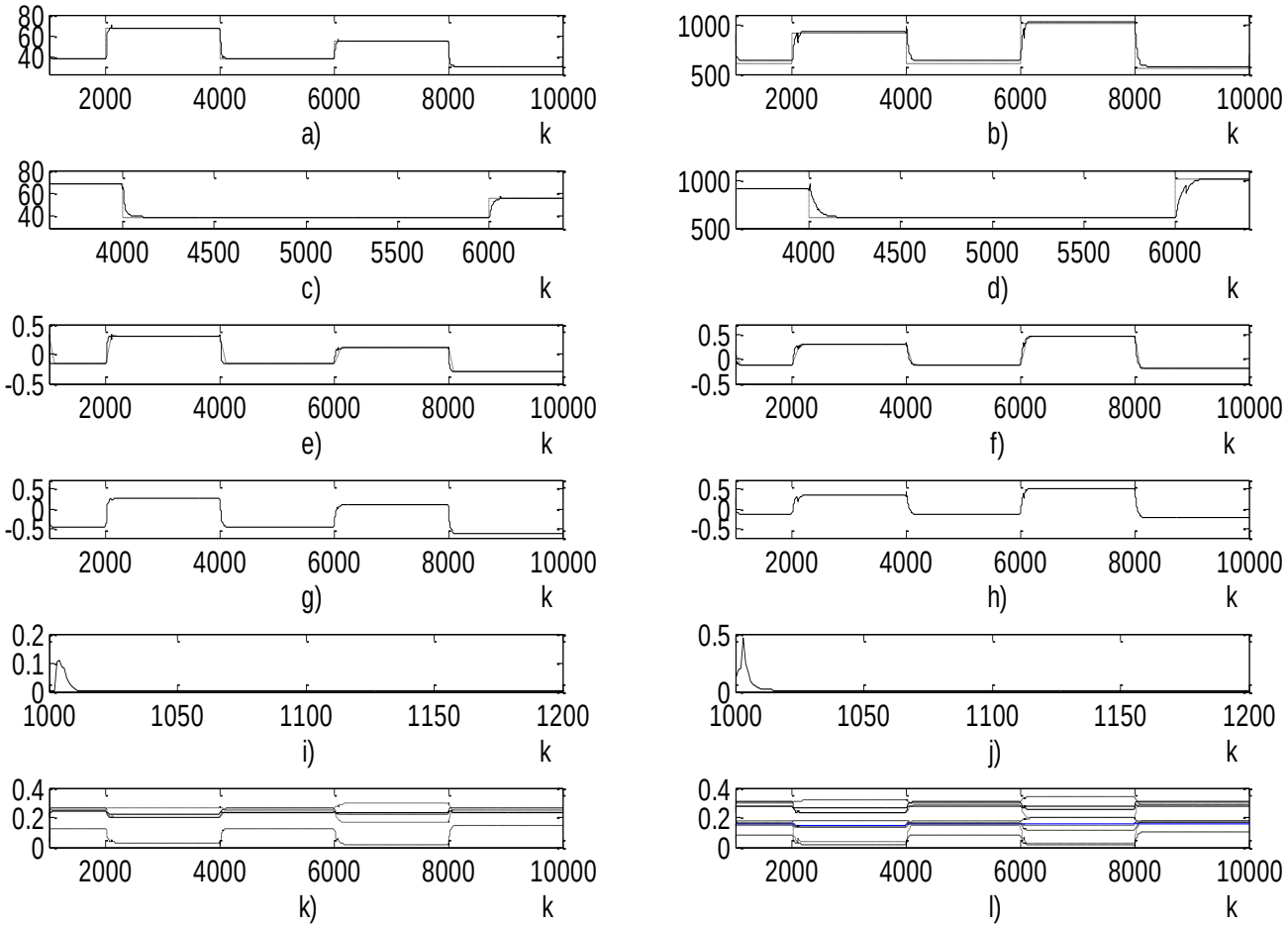


Figura 6.28 Control Neuronal indirecto mas término integral de un motor de combustión interna, utilizando algoritmo L-M a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Segunda señal de salida de la planta(continua) y segunda señal de referencia (punteada), c)acercamiento a la grafica a), d) acercamiento a la grafica b), e) Salida de la planta (continua) y salida del identificador neuronal, f) Salida de la planta (continua) y salida del identificador neuronal, g) y h) Señales de Control i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la red de identificación, l) estados de la red de control.

Control neuronal indirecto sin término integral algoritmo L-M

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida, utilizando el algoritmo L-M para el aprendizaje. El control se realiza utilizando un controlador por modos deslizantes.

Algoritmo L-M
ε = 7.9560e-007
σ = 1.4478e-008

Tabla (6.28) Valores de la media y desviación estándar para el algoritmo L-M

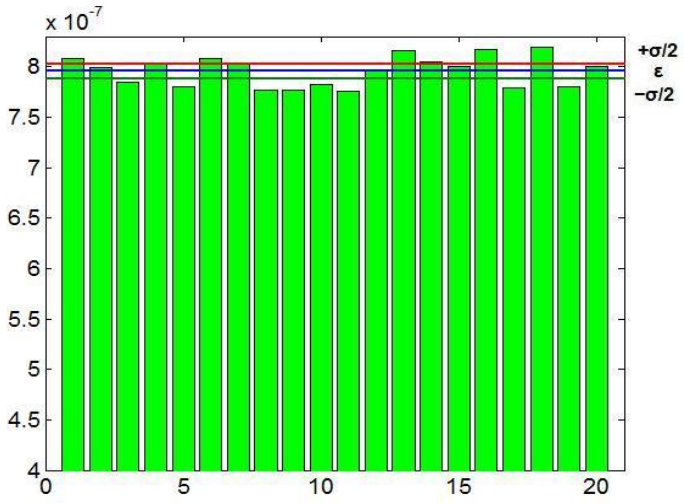


Figura 6.29 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	4.118E-6	3.988E-06	3.947E-06	4.122-E06	4.098E-06
No	6	7	8	9	10
EMC	4.109E-06	4.091E-06	4.035E-06	4.028E-06	4.135E-06
No	11	12	13	14	15
EMC	4.010E-06	4.134E-06	4.128E-06	4.144E-06	4.057E-06
No	16	17	18	19	20
EMC	4.089E-06	4.168E-06	4.041-06	3.946E-06	4.147E-06

Tabla (6.29). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo L-M.

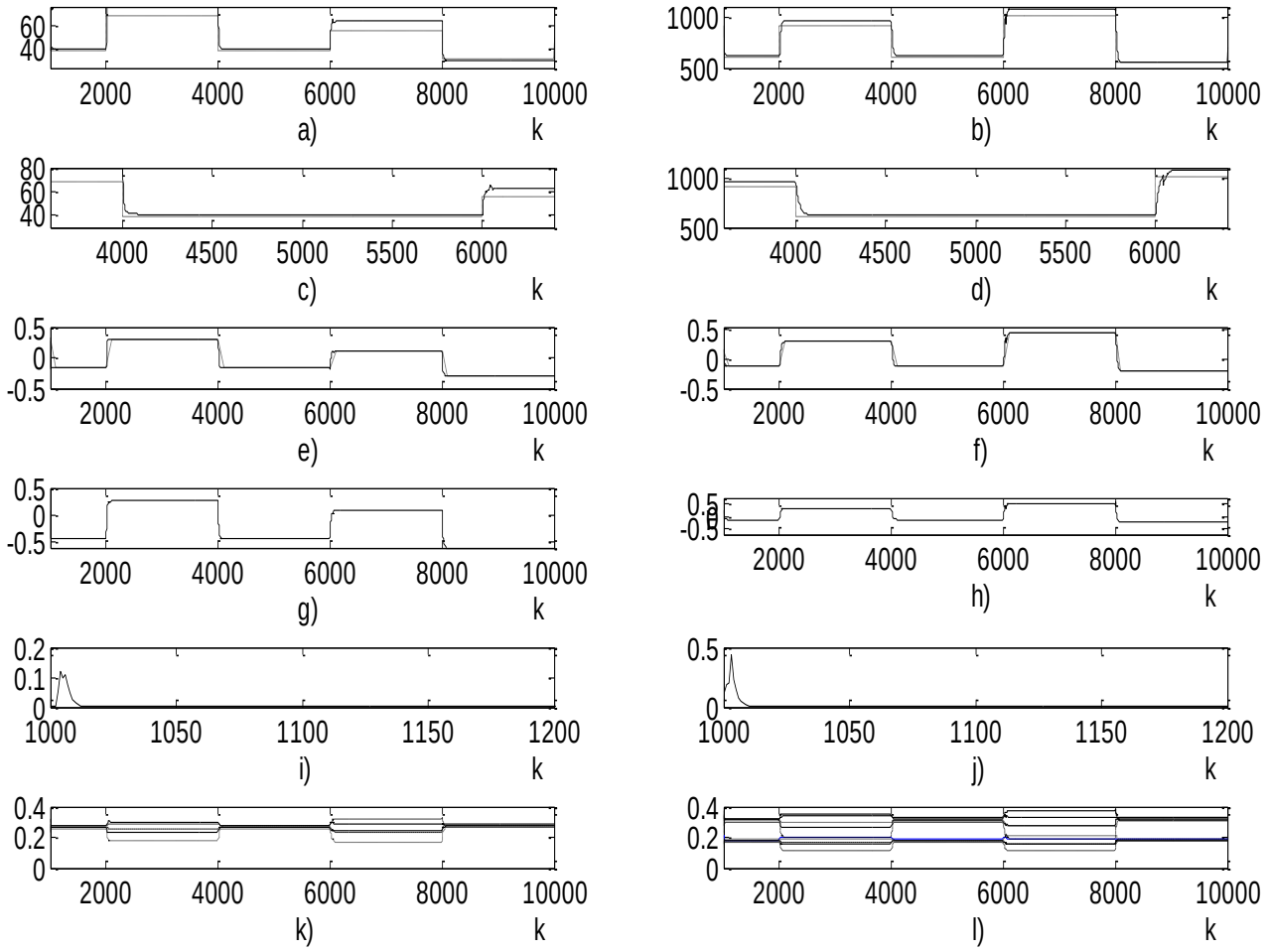


Figura 6.30 Control Neuronal indirecto sin término integral de un motor de combustión interna, utilizando algoritmo L-M a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Segunda señal de salida de la planta(continua) y segunda señal de referencia (punteada), c)acercamiento a la grafica a), d) acercamiento a la grafica b), e) Salida de la planta (continua) y salida del identificador neuronal, f) Salida de la planta (continua) y salida del identificador neuronal, g) y h) Señales de Control i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la red de identificación, l) estados de la red de control.

Control neuronal indirecto mas término integral algoritmo BP

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida, utilizando el algoritmo BP para el aprendizaje. El control se realiza utilizando un controlador por modos deslizantes con termino integral.

Algoritmo BP
ε = 5.6879e-006
σ = 2.5106e-007

Tabla (6.30) Valores de la media y desviación estándar para el algoritmo BP

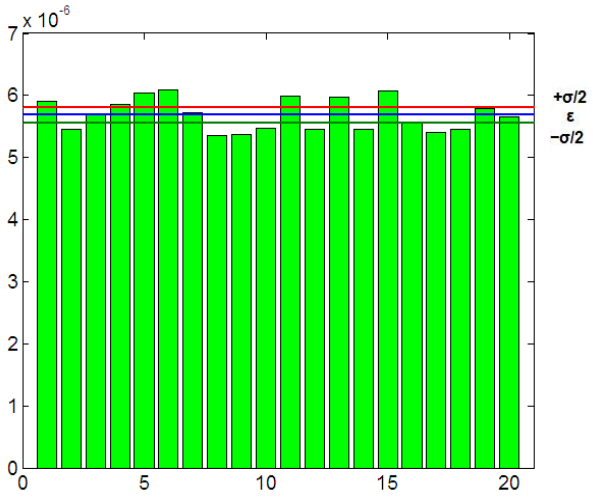


Figura 6.31 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP.

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.5909E-05	0.5462E-05	0.5688E-05	0.5862E-05	0.6034E-05
No	6	7	8	9	10
EMC	0.6096E-05	0.5725E-05	0.5357E-05	0.5367E-05	0.5464E-05
No	11	12	13	14	15
EMC	0.5989E-05	0.5461E-05	0.5965E-05	0.5452E-05	0.6069E-05
No	16	17	18	19	20
EMC	0.5547E-05	0.5409E-05	0.5458E-05	0.5787E-05	0.5658E-05

Tabla (6.31). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo BP

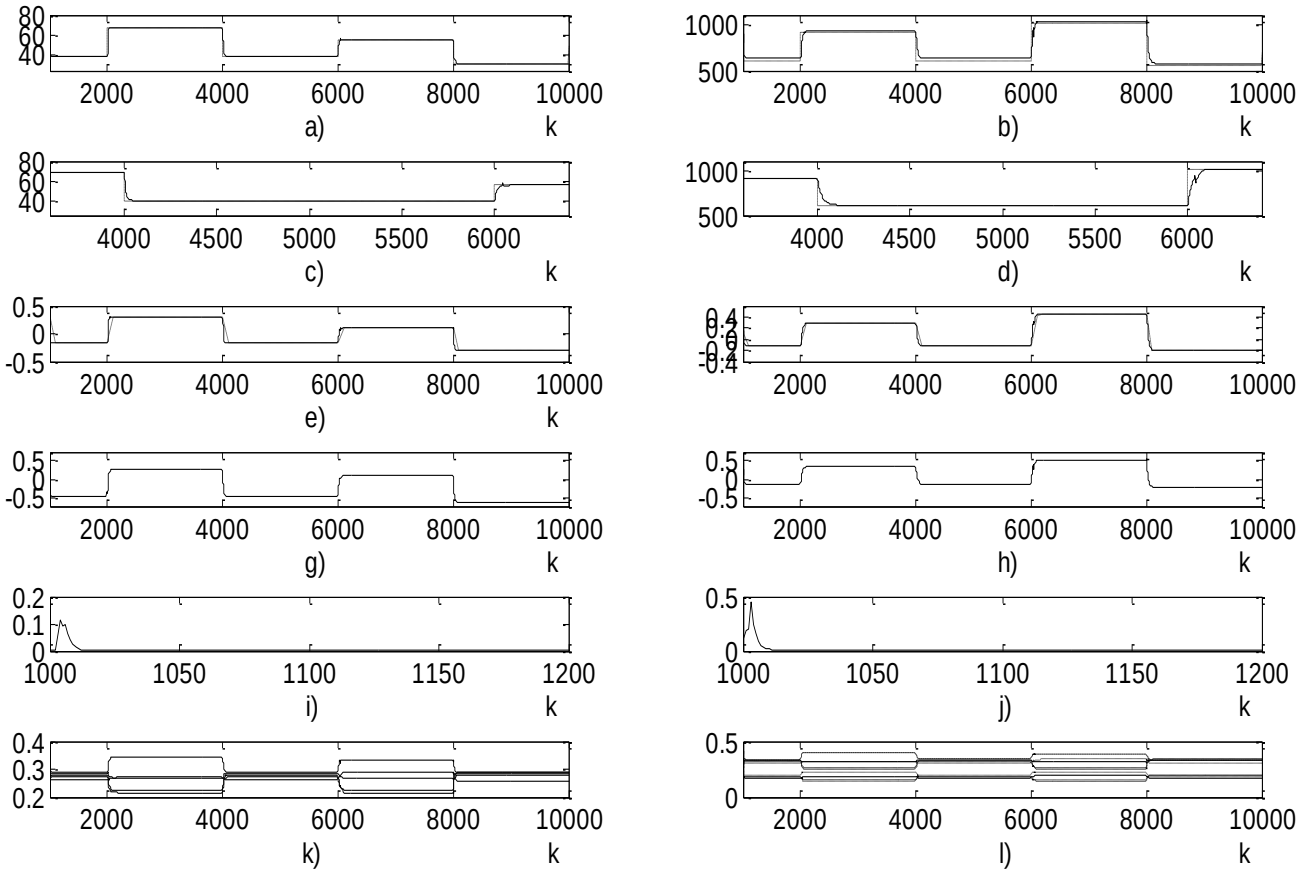


Figura 6.32 Control neuronal indirecto mas término integral de un motor de combustión interna, utilizando algoritmo BP a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Segunda señal de salida de la planta(continua) y segunda señal de referencia (punteada), c)acercamiento a la grafica a), d) acercamiento a la grafica b), e) Salida de la planta (continua) y salida del identificador neuronal, f) Salida de la planta (continua) y salida del identificador neuronal, g) y h) Señales de Control i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la red de identificación, l) estados de la red de control.

Control neuronal indirecto sin término integral algoritmo BP

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida, utilizando el algoritmo BP para el aprendizaje. El control se realiza utilizando un controlador por modos deslizantes.

Algoritmo BP
ε = 4.3460e-005
σ = 2.0357e-006

Tabla (6.32) Valores de la media y desviación estándar para el algoritmo BP

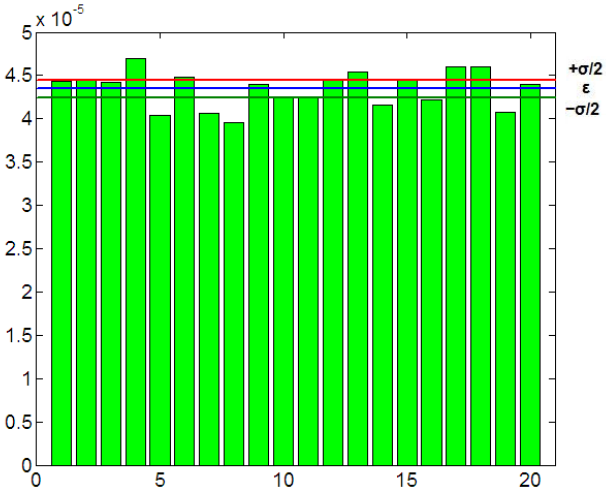


Figura 6.33 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP.

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.4429E-04	0.4458E-04	0.4419E-04	0.4697E-04	0.4034E-04
No	6	7	8	9	10
EMC	0.4485E-04	0.4059E-04	0.3954E-04	0.4393E-04	0.4252E-04
No	11	12	13	14	15
EMC	0.4259E-04	0.4442E-04	0.4540E-04	0.4162E-04	0.4442E-04
No	16	17	18	19	20
EMC	0.4221E-04	0.4604E-04	0.4596E-04	0.4077E-04	0.4399E-04

Tabla (6.33). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo BP

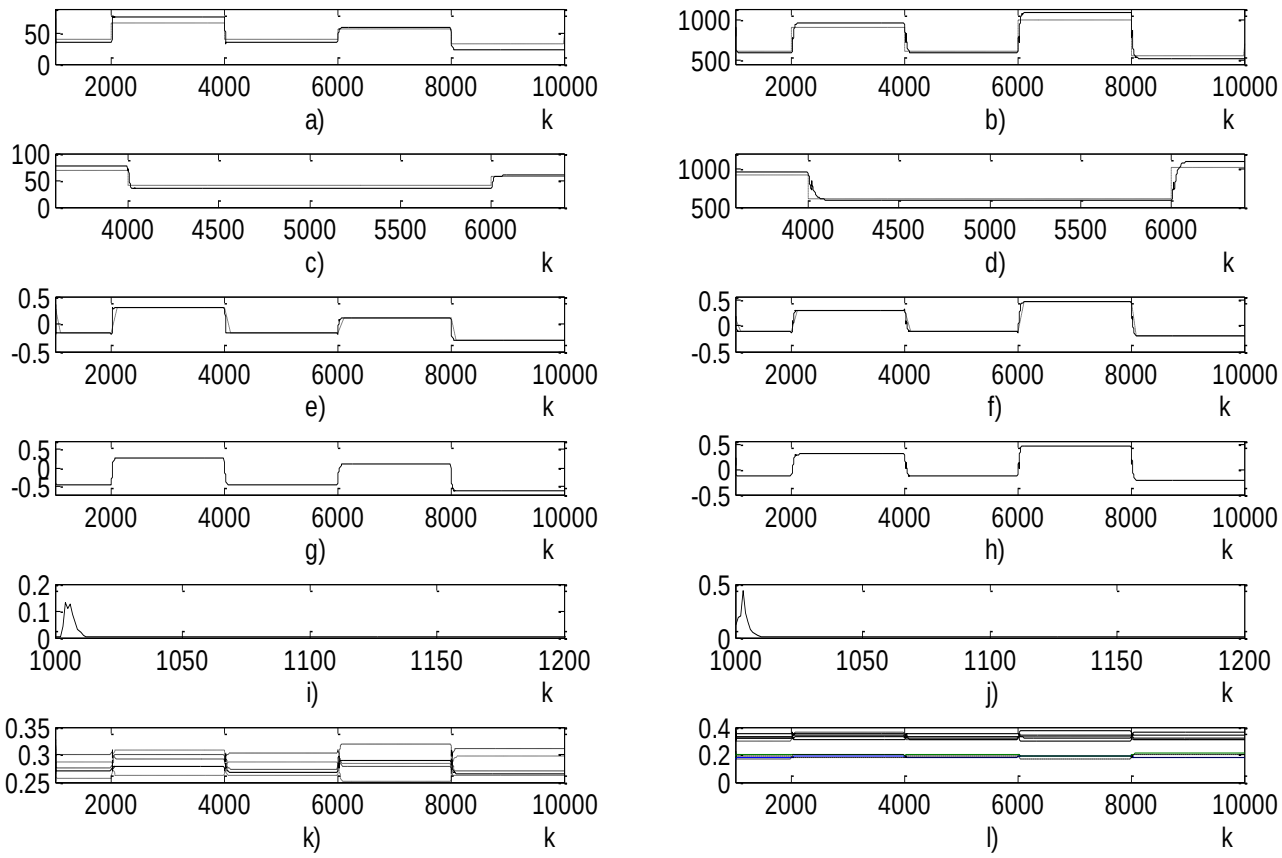


Figura 6.34 Control neuronal indirecto sin término integral de un motor de combustión interna, utilizando algoritmo BP a) Señal de salida de la planta(continua) y señal de referencia (punteada), b) Segunda señal de salida de la planta(continua) y segunda señal de referencia (punteada), c)acercamiento a la grafica a), d) acercamiento a la grafica b), e) Salida de la planta (continua) y salida del identificador neuronal, f) Salida de la planta (continua) y salida del identificador neuronal, g) y h) Señales de Control i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la red de identificación, l) estados de la red de control.

CONTROL	Algoritmo	MEDIA	VARIANZA
Control neuronal directo con término integral BP	BP	4.3079e-005	1.5128e-006
Control neuronal directo sin término integral	BP	4.6495e-005	1.4110e-006
Control neuronal indirecto con término integral	BP	5.6879e-006	2.5106e-007
Control neuronal indirecto sin termino integral	BP	4.3460e-005	2.0357e-006
Control neuronal directo con término integral.	L-M	3.9337e-005	1.0644e-006
Control neuronal directo sin termino integral	L-M	4.6358e-005	1.1309e-006
control neuronal indirecto con termino integral	L-M	4.0768e-006	6.5208e-008
control neuronal indirecto sin termino integral	L-M	7.9560e-007	1.4478e-008

Tabla (6.34) Comparación de las medias y varianzas de 20 corridas con distintos algoritmos de control

De acuerdo a la tabla 6.34, y tomando en cuenta únicamente los controladores entrenados con el algoritmo Backpropagation, se puede notar que tanto el control directo como el indirecto, ambos sin término integral, tienen el mismo desempeño (media del error); cuando se le añade el término integral a cada uno de estos controladores, la mejora es ligera para el control directo, mientras que para el control indirecto la mejora es muy buena. Cuando se utiliza el algoritmo de aprendizaje de Levenberg-Marquardt se logra tener una disminución en la media del error para cada uno de los controladores, así como de la varianza del error para cada una de las simulaciones.

6.7.3 Sistema de masas.

A continuación se presentan los resultados de control obtenidos por medio de simulación para la planta de dos masas con resorte y amortiguador.

Control neuronal directo más término integral con algoritmo BP

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida. La red de control posee un esquema 11-13-2, con 11 neuronas de entrada compuestas por el error de control, los estados de la red de identificación, el vector de referencia y los términos integrales, 13 neuronas en la capa oculta y 2 neuronas que representan la salida de control. Ambas redes entrenadas con algoritmo BP.

Control neuronal directo mas término integral algoritmo BP

Algoritmo BP	
ε =	1.3486e-005
σ =	6.8966e-007

Tabla (6.35) Valores de la media y desviación estándar para el algoritmo BP

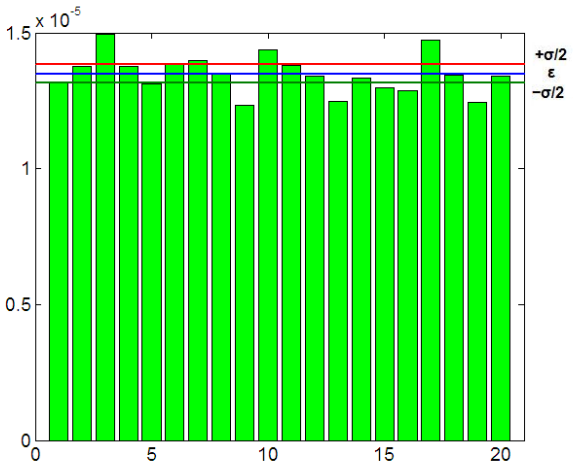


Figura 6.35 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP.

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.1317E-04	0.1376E-04	0.1494E-04	0.1378E-04	0.1310E-04
No	6	7	8	9	10
EMC	0.1387E-04	0.1400E-04	0.1351E-04	0.1232E-04	0.1438E-04
No	11	12	13	14	15
EMC	0.1379 E-04	0.1342E-04	0.1247E-04	0.1335E-04	0.1298E-04
No	16	17	18	19	20
EMC	0.1288E-04	0.1471E-04	0.1345E-04	0.1243E-04	0.1340E-04

Tabla (6.36). Valores del error medio cuadrático para cada simulación, utilizando el algoritmo BP

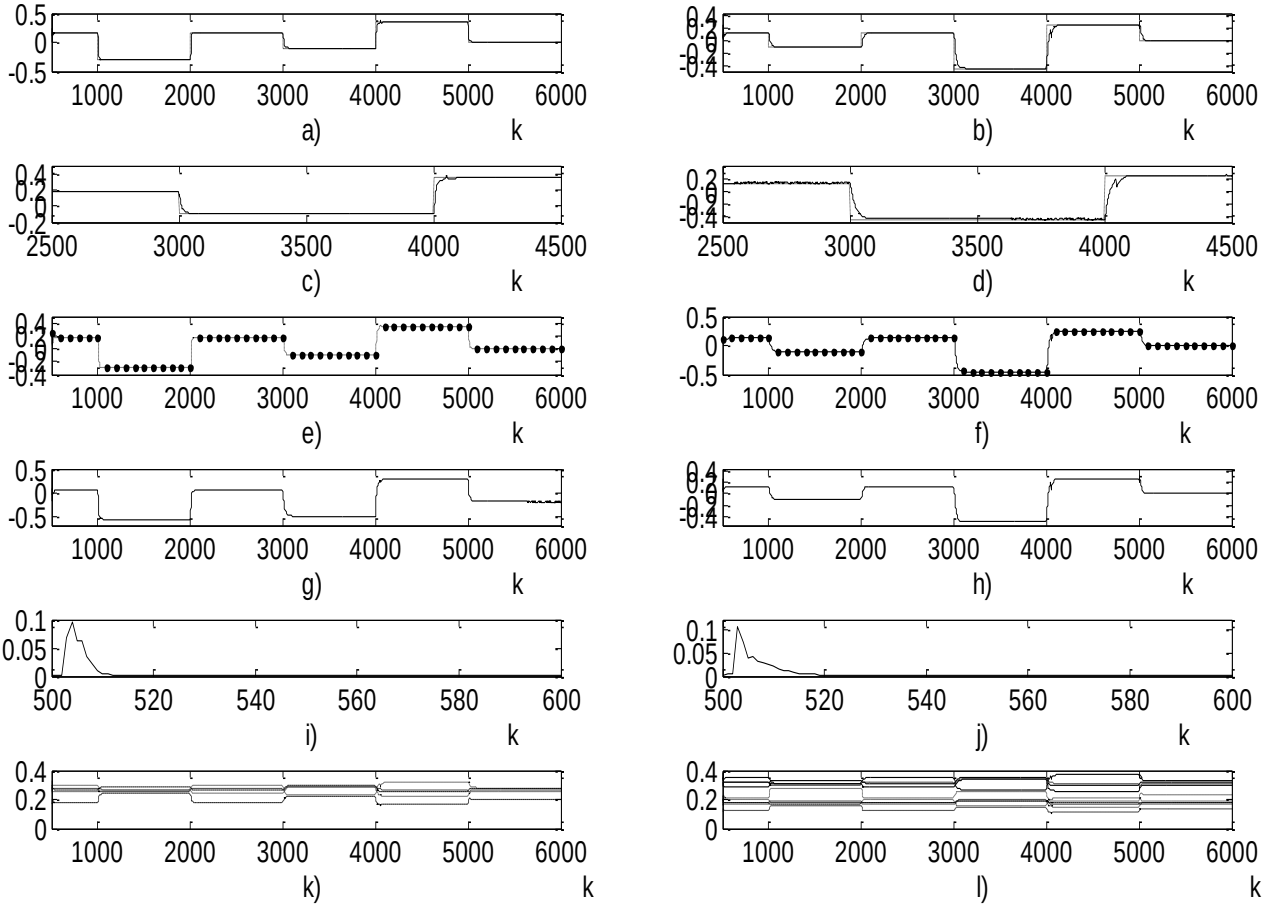


Figura 6.36 Resultados de simulación de un control neuronal directo con término integral y algoritmo L-M, a) Posición de la masa m (Línea Continua) y Señal de referencia (línea punteada), b) Posición de la masa M (Línea Continua) y Señal de referencia (línea punteada), c) acercamiento a una sección de la grafica mostrada en “a” , d) acercamiento a una sección de la grafica mostrada en “b”. e) Comparación entre la primer salida de la planta(continua) y la primer salida de la red de identificación(punteada), f) Comparación entre la segunda salida de la planta(continua) y la segunda salida de la red de identificación(punteada), g) y h) Señales de Control, i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la Red, l) estados de la red de control.

Control neuronal directo sin término integral utilizando el algoritmo BP

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida. La red de control posee un esquema 9-13-2, con 9 neuronas de entrada compuestas por el error de control, los estados de la red de identificación y el vector de referencia, 13 neuronas en la capa oculta y 2 neuronas que representan la salida de control. Ambas redes fueron entrenadas utilizando el algoritmo BP.

Algoritmo BP
ε = 2.2346e-005
σ = 1.5706e-006

Tabla (6.37) Valores de la media y desviación estándar para el algoritmo BP

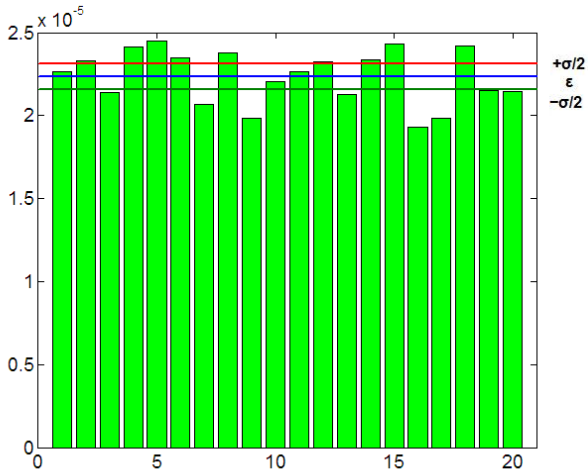


Figura 6.37 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP.

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.2263E-04	0.2328E-04	0.2140E-04	0.2414E-04	0.2451E-04
No	6	7	8	9	10
EMC	0.2350E-04	0.2070E-04	0.2377E-04	0.1985E-04	0.2202E-04
No	11	12	13	14	15
EMC	0.2265E-04	0.2321E-04	0.2129E-04	0.2336E-04	0.2431E-04
No	16	17	18	19	20
EMC	0.1933E-04	0.1982E-04	0.2417E-04	0.2152E-04	0.2146E-04

Tabla (6.38). Valores del error medio cuadrático para cada simulación, algoritmo BP

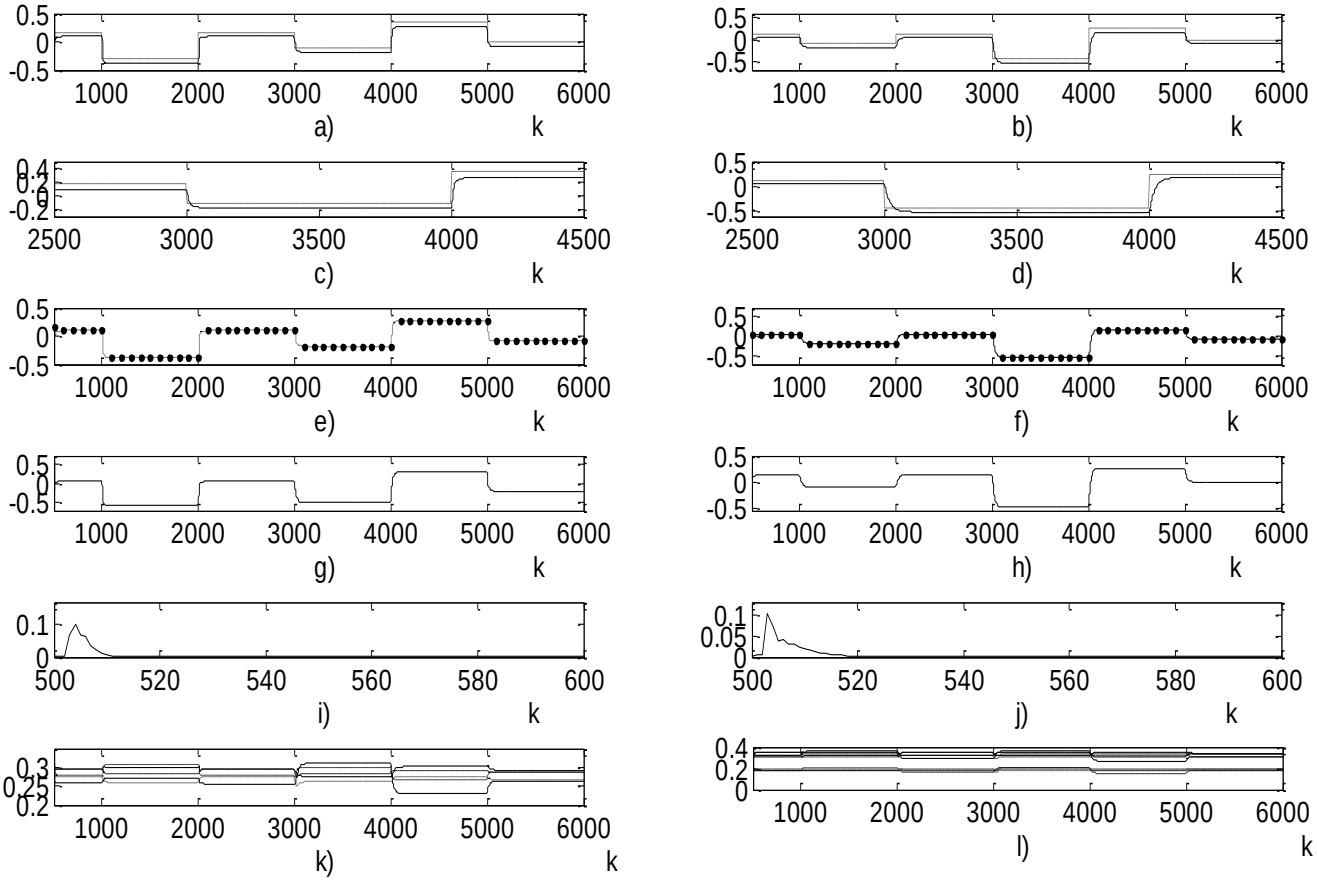


Figura 6.38 Resultados de simulación de un control neuronal directo sin término integral y algoritmo BP, a) Posición de la masa m (Línea Continua) y Señal de referencia (línea punteada), b) Posición de la masa M (Línea Continua) y Señal de referencia (línea punteada), c) Comparación entre la primer salida de la planta(continua) y la primer salida de la red de identificación(punteada), d) Comparación entre la segunda salida de la planta (continua) y la segunda salida de la red de identificación(punteada), e) Error Medio Cuadrático de Control, e) y f) Señales de Control, g) error medio cuadrático de identificación, h) error medio cuadrático de control, i) Estados estimados por la RN de control.

Control neuronal directo mas término integral utilizando el algoritmo L-M

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida. La red de control posee un esquema 11-13-2, con 11 neuronas de entrada compuestas por el error de control, los estados de la red de identificación, el vector de referencia y los términos integrales, 13 neuronas en la capa oculta y 2 neuronas que representan la salida de control. Ambas redes fueron entrenadas utilizando el algoritmo LM.

Algoritmo L-M	
ε =	1.3532e-005
σ =	7.6576e-007

Tabla (6.39) Valores de la media y desviación estándar para el algoritmo L-M

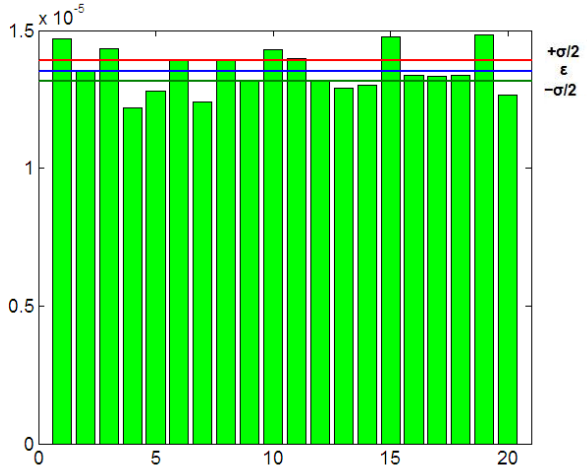


Figura 6.39 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M con término integral.

ALGORITMO L-M					
No	**1**	**2**	**3**	**4**	**5**
EMC	0.1470E-04	0.1350E-04	0.1432E-04	0.1219E-04	0.1279E-04
No	6	7	8	9	10
EMC	0.1396E-04	0.1241E-04	0.1392E-04	0.1316E-04	0.1430E-04
No	11	12	13	14	15
EMC	0.1396E-04	0.1315E-04	0.1291E-04	0.1303E-04	0.1476E-04
No	16	17	18	19	20
EMC	0.1337E-04	0.1333E-04	0.1337E-04	0.1484E-04	0.1266E-04

Tabla (6.40). Valores del error medio cuadrático para cada simulación, algoritmo L-M

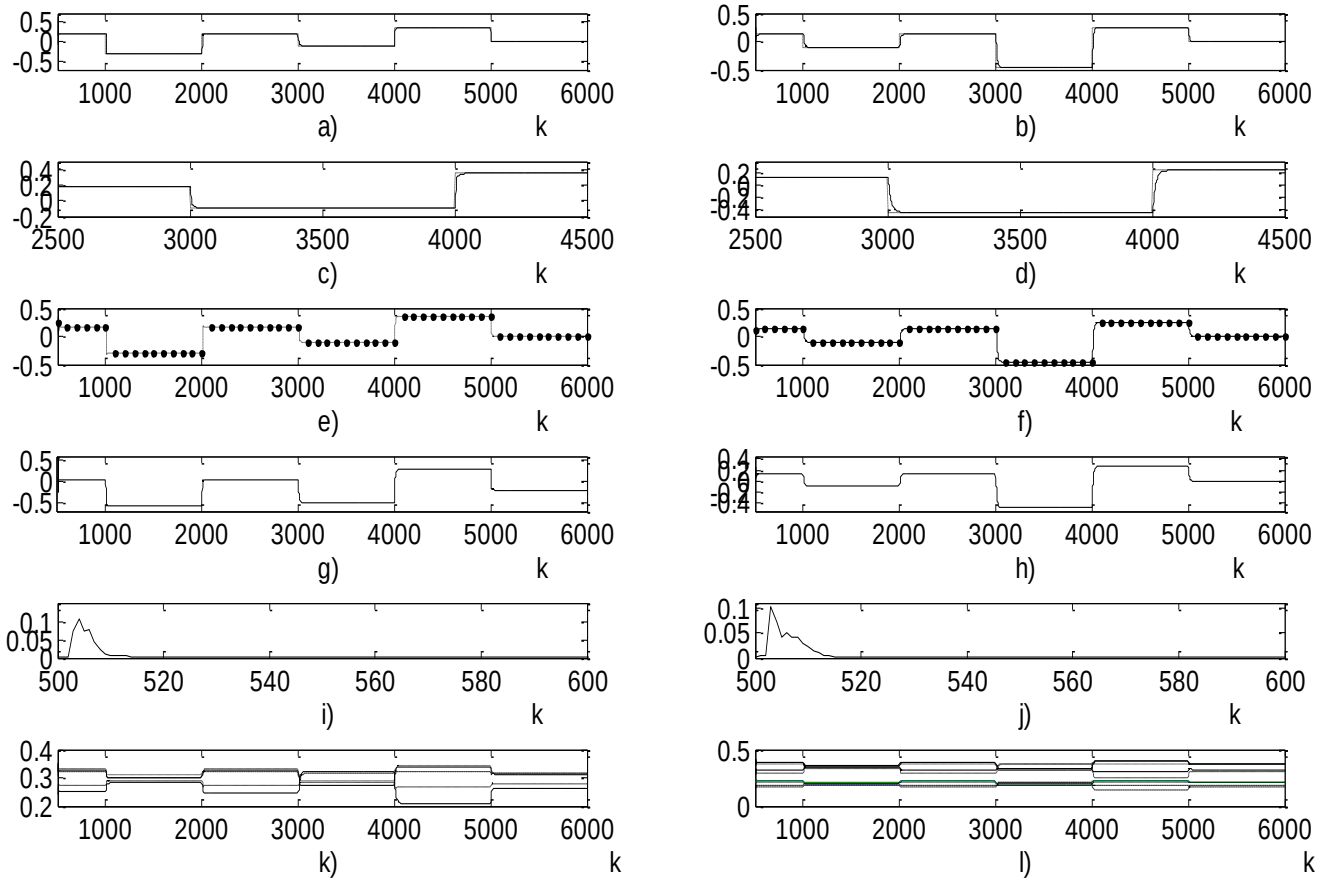


Figura 7.40 Resultados de simulación de un control neuronal directo con término integral y algoritmo L-M, a) Posición de la masa m (Línea Continua) y Señal de referencia (línea punteada), b) Posición de la masa M (Línea Continua) y Señal de referencia (línea punteada), c) acercamiento a una sección de la grafica mostrada en “a” , d) acercamiento a una sección de la grafica mostrada en “b”. e) Comparación entre la primer salida de la planta(continua) y la primer salida de la red de identificación(punteada), f) Comparación entre la segunda salida de la planta(continua) y la segunda salida de la red de identificación(punteada), g) y h) Señales de Control, i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la Red, l) estados de la red de control.

Control neuronal directo sin término integral utilizando algoritmo L-M

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida. La red de control posee un esquema 9-13-2, con 9 neuronas de entrada compuestas por el error de control, los estados de la red de identificación, el vector de referencia y los términos integrales, 13 neuronas en la capa oculta y 2 neuronas que representan la salida de control. Ambas redes fueron entrenadas utilizando el algoritmo LM.

Algoritmo L-M
ε = 1.9062e-005
σ = 1.0034e-006

Tabla (6.41) Valores de la media y desviación estándar para el algoritmo L-M

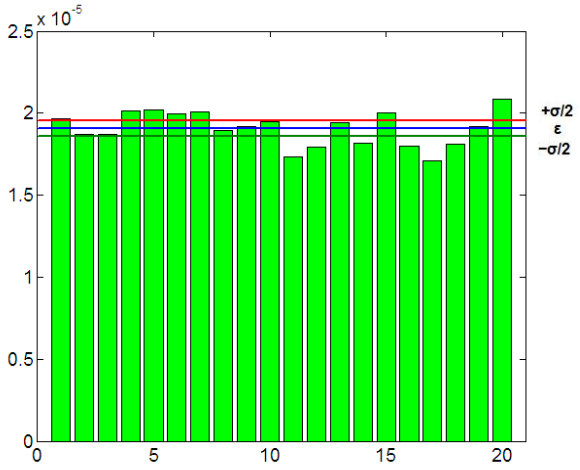


Figura 6.41 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M sin término integral.

ALGORITMO L-M					
No	1	2	3	4	5
EMC	0.1969E-04	0.1872E-04	0.1869E-04	0.2012E-04	0.2021E-04
No	6	7	8	9	10
EMC	0.1996E-04	0.2011E-04	0.1896E-04	0.1917E-04	0.1947E-04
No	11	12	13	14	15
EMC	0.1734E-04	0.1793E-04	0.1941E-04	0.1817E-04	0.2002E-04
No	16	17	18	19	20
EMC	0.1802E-04	0.1711E-04	0.1814E-04	0.1918E-04	0.2083E-04

Tabla (6.42). Valores del error medio cuadrático para cada simulación, algoritmo L-M

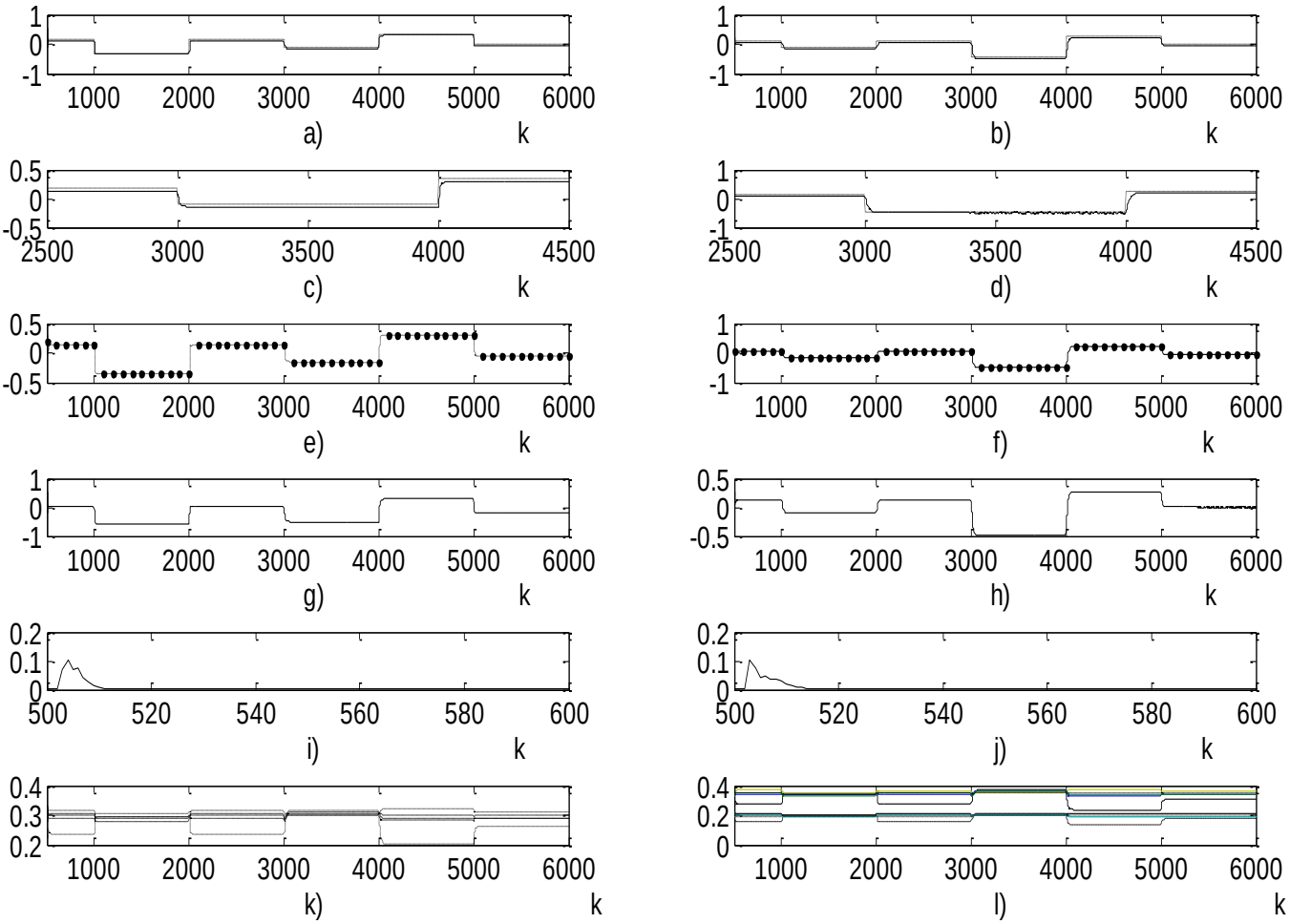


Figura 6.42 Resultados de simulación de un control neuronal directo sin término integral y algoritmo L-M, a) Posición de la masa m (Línea Continua) y Señal de referencia (línea punteada), b) Posición de la masa M (Línea Continua) y Señal de referencia (línea punteada), c) acercamiento a una sección de la grafica mostrada en “a” , d) acercamiento a una sección de la grafica mostrada en “b”. e) Comparación entre la primer salida de la planta(continua) y la primer salida de la red de identificación(punteada), f) Comparación entre la segunda salida de la planta(continua) y la segunda salida de la red de identificación(punteada), g) y h) Señales de Control, i) error medio cuadrático de identificación, j) error medio cuadrático de control, k) Estados de la Red de Identificación, l) estados de la red de control.

Control neuronal indirecto mas término integral utilizando algoritmo BP

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida, el algoritmo de aprendizaje es el BP. EL control se realiza mediante un controlador por modos deslizantes con termino integral.

Algoritmo BP	
ε =	2.6059e-005
σ =	1.1016e-006

Tabla (6.43) Valores de la media y desviación estándar para el algoritmo BP

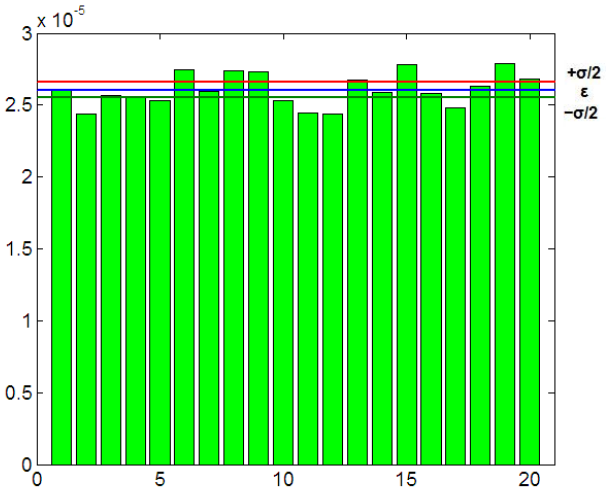


Figura 6.43 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP con término integral.

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.2600E-04	0.2435E-04	0.2570E-04	0.2561E-04	0.2531E-04
No	6	7	8	9	10
EMC	0.2743E-04	0.2596E-04	0.2739E-04	0.2732E-04	0.2529E-04
No	11	12	13	14	15
EMC	0.2443E-04	0.2438E-04	0.2671E-04	0.2586E-04	0.2778E-04
No	16	17	18	19	20
EMC	0.2583E-04	0.2478E-04	0.2632E-04	0.2791E-04	0.2680E-04

Tabla (6.44). Valores del error medio cuadrático para cada simulación, algoritmo BP

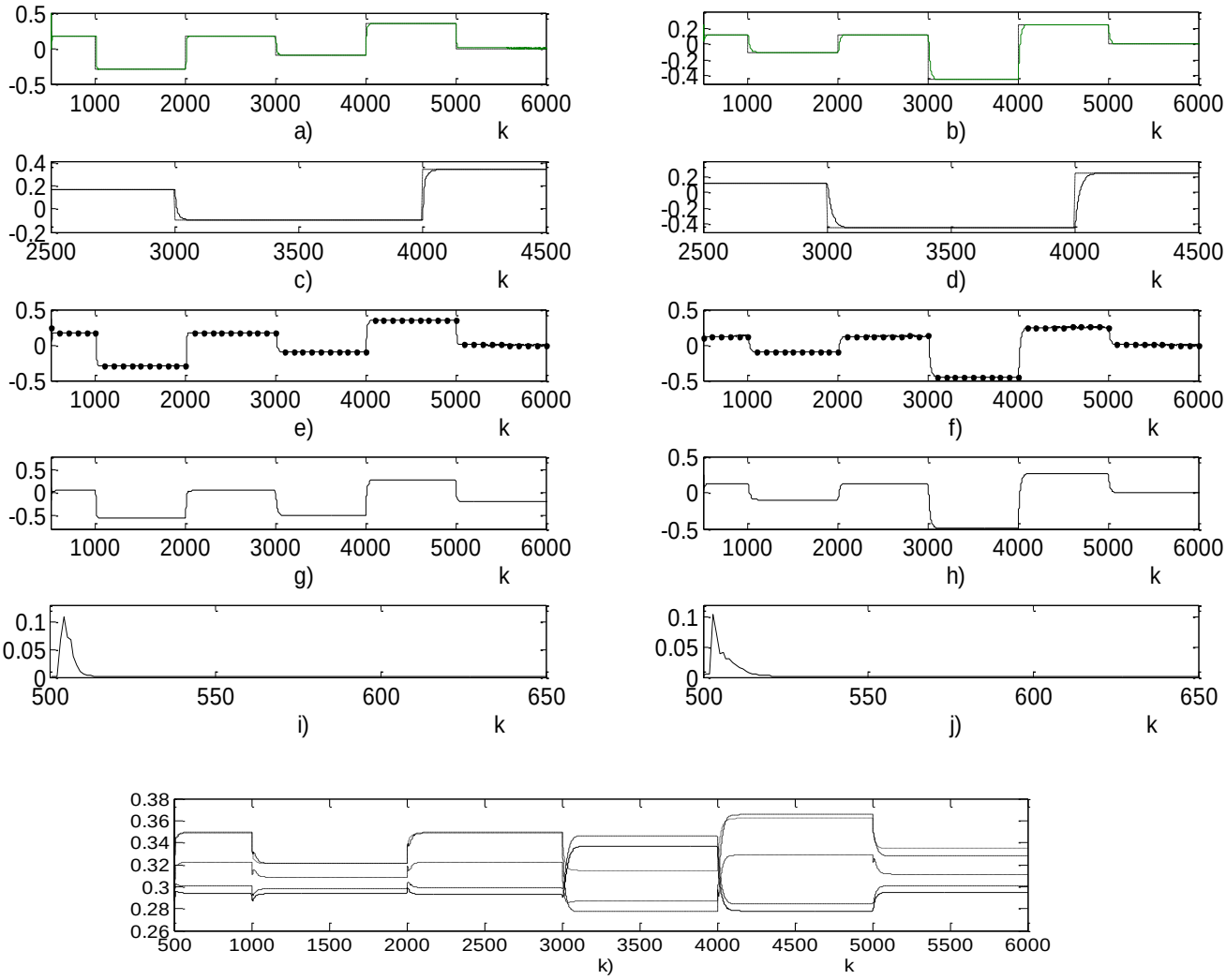


Fig. 6.44 Resultados de simulación de un control neuronal indirecto mas término integral y algoritmo BP, a) Posición de la masa m (Línea Continua) y Señal de referencia (línea punteada), b) Posición de la masa M (Línea Continua) y Señal de referencia (línea punteada), c) Comparación entre la primer salida de la planta(continua) y la primer salida de la red de identificación(punteada), d) Comparación entre la segunda salida de la planta (continua) y la segunda salida de la red de identificación(punteada), e) Error Medio Cuadrático de Control, e) y f) Señales de Control, g) error medio cuadrático de identificación, h) error medio cuadrático de control, i) Estados estimados por la RN de control.

Control neuronal indirecto sin término integral utilizando algoritmo BP

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida, el algoritmo de aprendizaje es el BP. EL control se realiza mediante un controlador por modos deslizantes.

Algoritmo BP
ε = 2.8203e-005
σ = 1.7736e-006

Tabla (6.45) Valores de la media y desviación estándar para el algoritmo BP

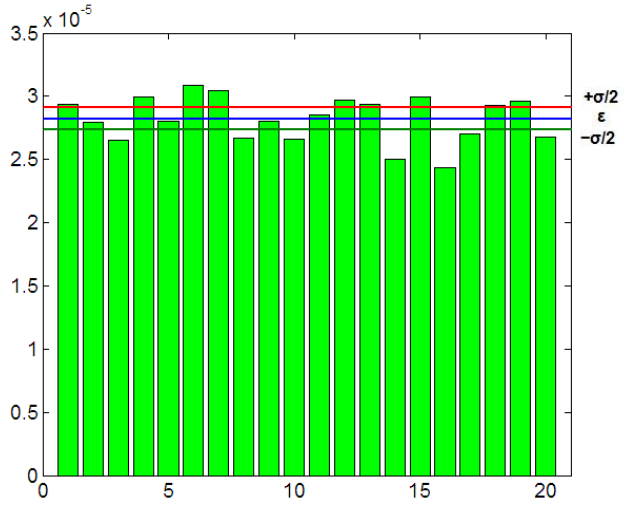


Figura 6.45 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP sin término integral.

ALGORITMO BP					
No	1	2	3	4	5
EMC	0.2940 E-4	0.2796 E-4	0.2653 E-4	0.2999 E-4	0.2803 E-4
No	6	7	8	9	10
EMC	0.3086 E-4	0.3041 E-4	0.2666 E-4	0.2799 E-4	0.2659 E-4
No	11	12	13	14	15
EMC	0.2852 E-4	0.2974 E-4	0.2938 E-4	0.2504 E-4	0.2992 E-4
No	16	17	18	19	20
EMC	0.2434 E-4	0.2706 E-4	0.2928 E-4	0.2963 E-4	0.2673 E-4

Tabla (6.46). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo BP

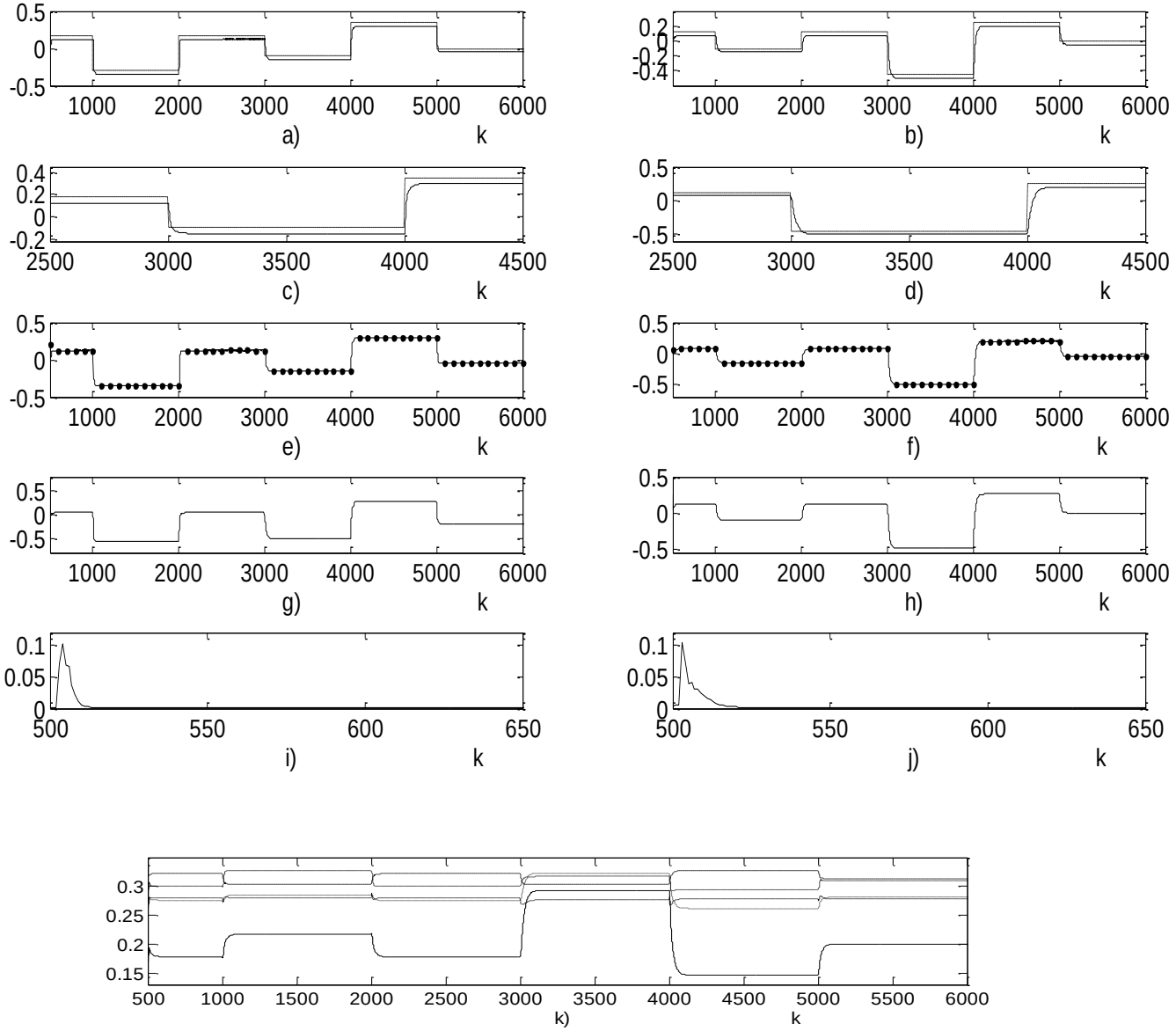


Figura 6.46 Resultados de simulación de un control neuronal indirecto sin término integral y utilizando algoritmo BP, a) Posición de la masa m (Línea Continua) y Señal de referencia (línea punteada), b) Posición de la masa M (Línea Continua) y Señal de referencia (línea punteada), c) Comparación entre la primer salida de la planta(continua) y la primer salida de la red de identificación(punteada), d) Comparación entre la segunda salida de la planta (continua) y la segunda salida de la red de identificación(punteada), e) Error Medio Cuadrático de Control, e) y f) Señales de Control, g) error medio cuadrático de identificación, h) error medio cuadrático de control, i) Estados estimados por la RN de control.

Control neuronal indirecto mas término integral utilizando algoritmo L-M

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida, el algoritmo de aprendizaje es el LM. EL control se realiza mediante un controlador por modos deslizantes con termino integral.

Algoritmo BP
ε = 1.4742e-005
σ = 1.3517e-006

Tabla (6.47) Valores de la media y desviación estándar para el algoritmo BP

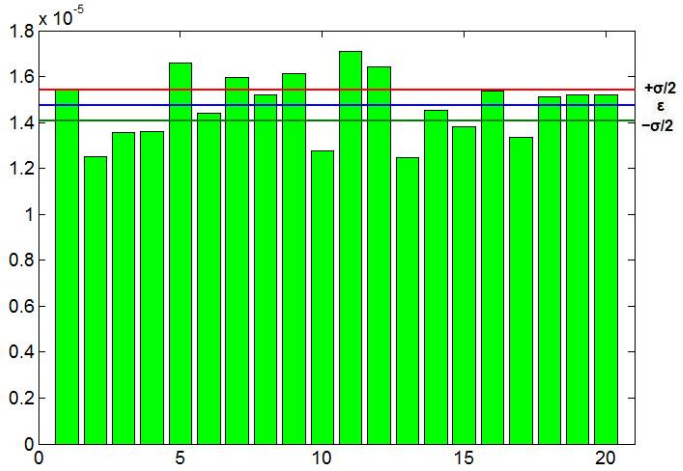


Figura 6.47 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo BP sin término integral.

ALGORITMO LM					
No	1	2	3	4	5
EMC	0.1546E-04	0.1358E-04	0.1253E-04	0.1361E-04	0.1660E-04
No	6	7	8	9	10
EMC	0.1442 E-04	0.1598 E-04	0.1522 E-04	0.1612 E-04	0.1277 E-04
No	11	12	13	14	15
EMC	0.1709 E-04	0.1644E-04	0.1245E-04	0.1453E-04	0.1383E-04
No	16	17	18	19	20
EMC	0.1535E-04	0.1335 E-04	0.1510E-04	0.1522 E-04	0.1520 E-04

Tabla (6.48). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo LM

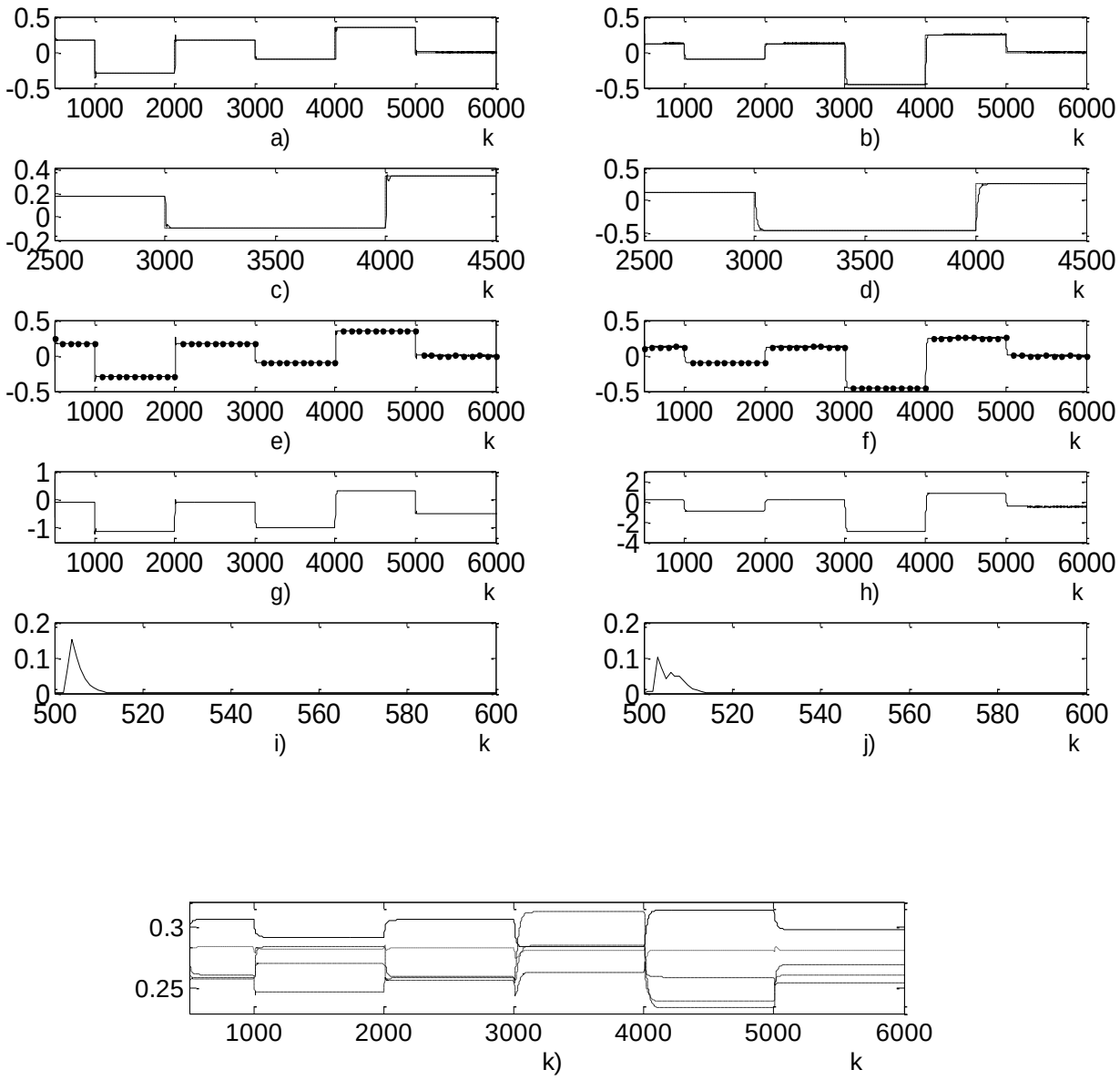


Figura 6.48 Resultados de simulación de un control neuronal indirecto mas término integral y utilizando algoritmo L-M, a) Posición de la masa m (Línea Continua) y Señal de referencia (línea punteada), b) Posición de la masa M (Línea Continua) y Señal de referencia (línea punteada), c) Comparación entre la primer salida de la planta(continua) y la primer salida de la red de identificación(punteada), d) Comparación entre la segunda salida de la planta (continua) y la segunda salida de la red de identificación(punteada), e) Error Medio Cuadrático de Control, e) y f) Señales de Control, g) error medio cuadrático de identificación, h) error medio cuadrático de control, i) Estados estimados por la RN de control.

Control neuronal indirecto sin término integral utilizando algoritmo L-M

La red utilizada para la identificación presenta el esquema 2-5-2, es decir, dos neuronas en la primer capa, 5 en la capa oculta y dos más en la de salida, el algoritmo de aprendizaje es el LM. EL control se realiza mediante un controlador por modos deslizantes.

Algoritmo BP
ε = 2.8235e-005
σ = 1.6079e-007

Tabla (6.48) Valores de la media y desviación estándar para el algoritmo L-M

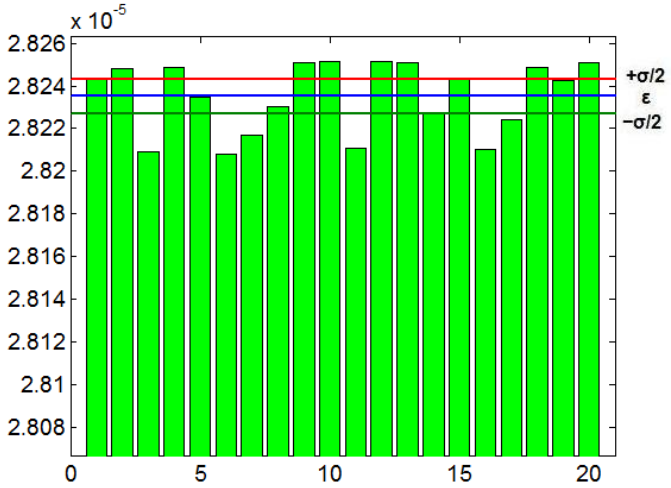


Figura 6.48 Comparación del error medio cuadrático final de control después de 20 simulaciones utilizando el algoritmo L-M sin término integral.

ALGORITMO LM					
No	1	2	3	4	5
EMC	0.1546E-04	0.1358E-04	0.1253E-04	0.1361E-04	0.1660E-04
No	6	7	8	9	10
EMC	0.1442 E-04	0.1598 E-04	0.1522 E-04	0.1612 E-04	0.1277 E-04
No	11	12	13	14	15
EMC	0.1709 E-04	0.1644E-04	0.1245E-04	0.1453E-04	0.1383E-04
No	16	17	18	19	20
EMC	0.1535E-04	0.1335 E-04	0.1510E-04	0.1522 E-04	0.1520 E-04

Tabla (6.50). Valores del error medio cuadrático para cada simulación, utilizando al algoritmo LM

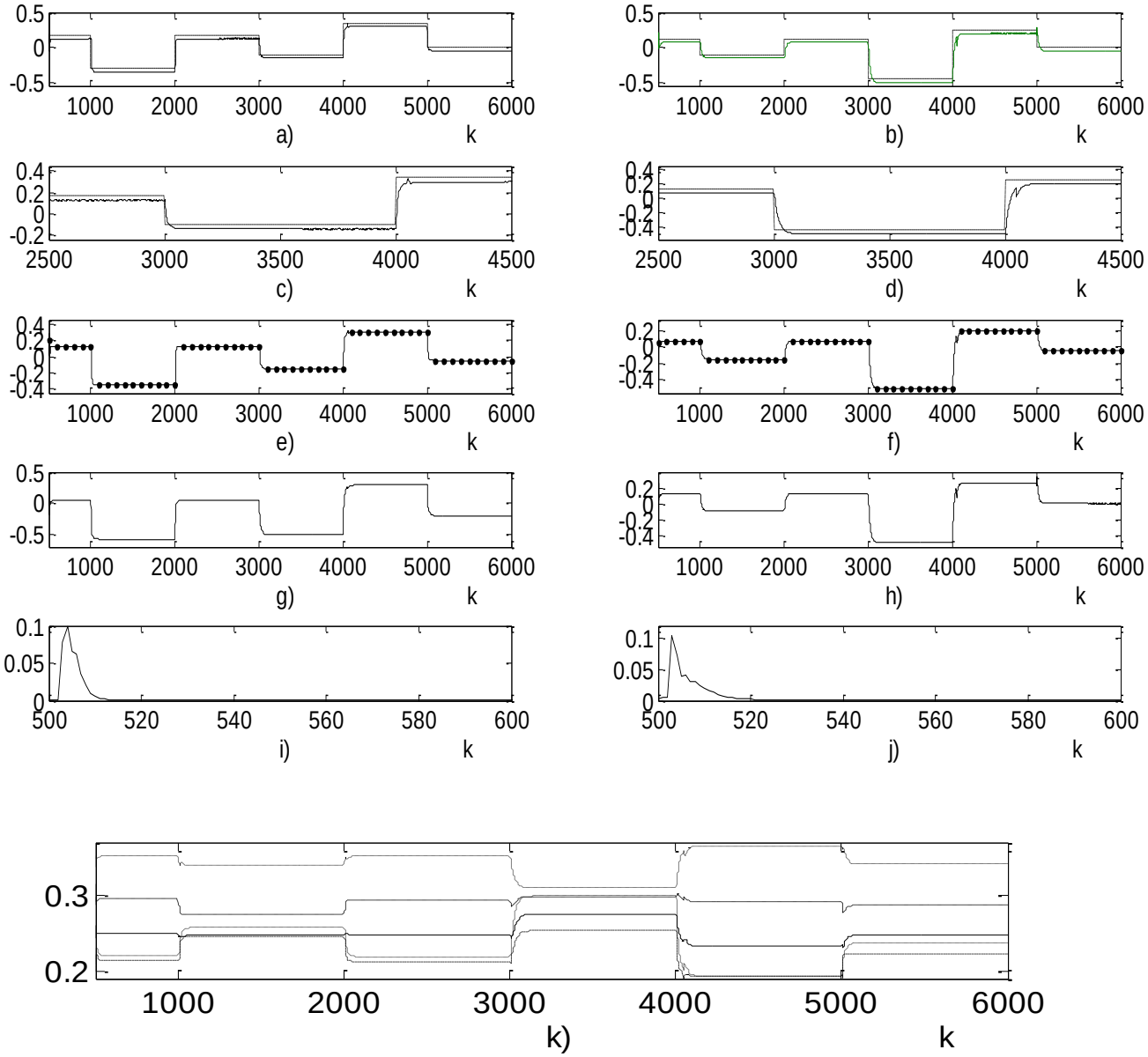


Figura 6.50 Resultados de simulación de un control neuronal indirecto sin término integral y utilizando algoritmo L-M, a) Posición de la masa m (Línea Continua) y Señal de referencia (línea punteada), b) Posición de la masa M (Línea Continua) y Señal de referencia (línea punteada), c) Comparación entre la primer salida de la planta(continua) y la primer salida de la red de identificación(punteada), d) Comparación entre la segunda salida de la planta (continua) y la segunda salida de la red de identificación(punteada), e) Error Medio Cuadrático de Control, e) y f) Señales de Control, g) error medio cuadrático de identificación, h) error medio cuadrático de control, i) Estados estimados por la RN de control.

CONTROL	Algoritmo	MEDIA	VARIANZA
Control neuronal directo con término integral BP	BP	1.3486e-005	6.8966e-007
Control neuronal directo sin término integral	BP	2.2346e-005	1.5706e-006
Control neuronal indirecto con término integral	BP	2.6059e-005	1.1016e-006
Control neuronal indirecto sin termino integral	BP	2.8203e-005	1.7736e-006
Control neuronal directo con termino integral usando algoritmo L-M	L-M	1.3532e-005	7.6576e-007
Control neuronal directo sin termino integral	L-M	1.9062e-005	1.0034e-006
Control neuronal indirecto con termino integral	L-M	1.4742e-005	1.3517e-006
Control neuronal indirecto sin termino integral	L-M	2.8235e-005	1.6079e-007

Tabla (6.51) Comparación de las medias y varianzas de 20 corridas con distintos algoritmos de control

En el control de este sistema de masas-resortes, se obtuvieron resultados muy similares a los obtenidos en las secciones anteriores, es decir, una mejora de los controladores cuando se tiene un término integral. Además se observa que los resultados se mejoran utilizando el algoritmo de aprendizaje de Levenberg-Marquardt.

Conclusiones Generales

En términos generales se puede concluir, con base en los resultados obtenidos, que el término integral ayuda a reducir el error en estado estacionario, que no es capaz de reducir el control neuronal puro por sí solo.

En esta tesis se realizó la simulación de tres plantas no lineales con oscilaciones. Posteriormente se realizaron las simulaciones de identificación, con un esquema de identificación paralelo, con dos algoritmos de aprendizaje, el Backpropagation y el de Levenberg-Marquardt. Se obtuvo un mejor resultado de aproximación, y por ende un menor error medio de aproximación, con el algoritmo de Levenberg-Marquardt.

En la parte de control se vieron dos enfoques, uno directo y otro indirecto, este último basado en un control en modos deslizantes. El primero muestra mejor adaptación, por parte de las RNRE, a los sistemas no lineales oscilantes en general. En el caso del control indirecto se tiene que los resultados son más pobres debido a que la aproximación de los parámetros y de los estados del sistema, proporcionados por la identificación neuronal, no son los estados reales de la planta, y esta discrepancia impide que los resultados sean mejores. En este último no existe una adaptación neuronal por parte del controlador, es decir, se hace la suposición de que los estados y parámetros identificados, sean los reales de la planta.

Siguiendo sobre la línea de control, se obtuvieron resultados que nos permiten afirmar que la inclusión de un término integral a un esquema de control neuronal permite mejorar los errores de control. Esto se hizo más notorio en el caso del control indirecto con término integral, en el cual el efecto del término integral permitió disminuir más la media del error a lo largo de las veinte simulaciones, debido a que el término integral se suma a la acción de control generada por el controlador neuronal. Por otro lado en el caso del control directo con término integral si se obtuvo una ligera mejora con respecto al control directo sin término integral, la cual estuvo limitada debido a que el término integral sólo servía de información al control neuronal y no como un complemento.

Adicionalmente se puede mencionar que el control neuronal directo con termino integral presenta un mejor resultado en términos del error medio cuadrático comparado con los demás métodos, sin embargo este método es también el más costoso computacionalmente dado la cantidad de operaciones matemáticas y el numero de variables que la memoria debe almacenar para realizar cada iteración. Aunque se debe destacar que al utilizar una forma canónica en la matriz dinámica de la red, no solo se reduce la complejidad de la red y su entrenamiento, sino el número de variables por calcular y por ende el tiempo de cómputo total.

Finalmente se puede concluir que el término integral mostró en cada esquema de control analizado, la ventaja de ser capaz de reducir el error en estado estacionario, lo cual se aprecia mejor gráficamente al incrementar la señal del offset aplicado a la entrada de cada señal de entrada de la planta.

Trabajo Futuro

Aplicar los algoritmos de control analizados en este trabajo para el control de una planta física no lineal con comportamiento oscilatorio,

Bibliografía

[1] Dreyfus, G., “Neural Networks: Methodology and Applications”, Ed. New York. Springer-Verlag Berlin Heidelberg, 2005, 2da ed., pp. 1-5. ISBN-10 3-540-22980-9

[2] Haykin, S., “Neural Networks, A comprehensive Foundation”. India. Ed. Prentice Hall US. 1999. 2da ed. pp. pp. 144-146.. ISBN 81-7808-300-0

[3] Sira-Ramírez, H., Márquez R., Rivas-Echeverria, F. y Llanes-Santiago, O., “Control de sistemas no lineales”. España. Ed. Prentice Hall. 2005. 1era ed. pp. 9-24. ISBN 84-205-4449-3

[4] Hilera, R., y Martínez V., “Redes Neuronales Artificiales. Fundamentos, Modelos y Aplicaciones”, Madrid, Ed. RA-MA / Addison Wesley Iberoamericana, 1995, pp. 4-96 ISBN: 84-7897-155-6

[5] Calderon, G., y Draye, J, “Nonlinear Dynamic System Identification with Dynamic Recurrent Neural Networks”, Proc. International Workshop on Neural Networks for Identification, Control, Robotics and Signal/Image Processing, pp. 49-54,1996. ISBN: 0-8186-7456-3

[6] Pak-Kin W., Chi-Man V., y Lap-Mou, “Data preprocessing and modeling of electronically-controlled automotive engine power performance using kernel principal components analysis and least squares support vector machines”, International Journal of Vehicle Systems Modelling and Testing, Vol. 3, No 4, pp 312-330, 2008.

[7] Marko, K., "Neural network application to diagnostics and control of vehicle control systems". In: Lippmann, R., Moody, J.E., Touretzky, D.S. (eds.) Advances in Neural Information Processing Systems, vol. 3, pp. 537–543. Morgan Kaufmann, San Francisco (1991)

[8] Puskorius, Gintaras V., Feldkamp, y Lee A., "Parameter-based Kalman filter training: theory and implementation". In: Haykin, S. (ed.) Kalman filtering and neural networks, ch. 2, pp. 23–67. Wiley, Chichester (2001). ISSN: 0018-9219

[9] Bloch, G., Lauer, F. y Colin, G.: “On learning Machines for Engine Control”, Studies in Computational Intelligence, pp. 165–189. Springer-Verlag Berlin Heidelberg 2008. ISBN 978-3-540-79256-7.

[10] Puskorius, G. y Feldkamp, L., “Neurocontrol of Nonlinear Dynamical Systems with Kalman Filter Trained Recurrent Networks”, IEEE Transactions on Neural Networks, Vol 5. No 2, Marzo 1994. pp 279-297, ISSN: 0018-9219

[11] Ayeb, M., Lichtenthaler, D., Winsel, T. and Theuerkauf, J: “SI Engine Modeling Using Neural Networks”. SAE technical paper series, Electronic Engine Controls 1998: Diganostics and Controls. DOI: 10.4271/980790

[12] Baruch, I., Gorcheva, E. y Garrido, R., “Redes Neuronales para la identificación de objetos No lineales”, Científica, No. 14, ESIME, pp. 34-35, Marzo 1990.

[13] Flores, J.M., Baruch, I. y Garrido, R., “Red Neuronal Recurrente para Identificación y Control de Sistemas No Lineales”, Científica, Vol.5, No 1, pp. 11-20, México 2001.

[14] Flores, J.M., Baruch, I., Thomas, F. y Garrido, R., “Adaptive Neural Control of Nonlinear Systems”, InDorfner, G., Bischof, H., Hornik, K., (eds): Artificial Neural Networks-ICANN 2001, Lectures Notes in Computer Science, Vol. 2130. Springer Verlag, Berlin Heidelberg New York, pp. 930-936, 2001.

[15] Baruch, I., Stoyanov I. and Gorcheva, E. “Topology and Learning of a class RNN”, ELEVTRIK, Vol. 4 supplement, pp. 35-42.

[16] K. S Narendra, and K. Pasthasarathy, “Identification and Control of Dynamic Systems using Neural Networks”. IEEE Transaction on NN`s, 1(1), 1990, 4-27. ISSN: 1083-4419

[17] Apostol, T. M., “Análisis Matemático”, Ed. Reverté, S.A., 2da Edición. ISBN: 8429150048

[18] Nava R. F., “Redes neuronales Recurrentes para la Identificación y Control de Sistemas No Lineales”, Tesis de maestría, CINVESTAV, IPN, México DF, 2001.

[19] Galván R., “Identificación y Control de un Sistema de Digestión Anaeróbica usando Redes Neuronales Recurrentes”, Tesis de maestría, CINVESTAV, IPN, México DF, 2007.

[20] Wan E., Beaufays F., “Diagramatic Methods for Deriving and Relating Temporal Neural Networks Algorithms”, Neural Computations, Vol. 8, No. 4, pp182-201. ISSN 1530-888X

[21] Fidlin A., “Nonlinear oscillations in Mechanical Engineering”, Springer-Verlag Berlin Heidelberg 2006, 2nd Edition. ISBN 3-642-06634-8

[22] Song, J., Lee, K., Choi, J., “Vibration control of 2-mass system using a neural network torsional torque estimator”., IECON ’98 Proceedings of the 24th Annula Conference of the IEEE, 1998, pp 1785-1788 vol 3. ISBN: 0-7803-4503-7

[23] Yousefi, H., Hirvonen, M., Handroos, H. and Soleymani, A., “Application of Neural Network in Suppressing Mechanical Vibration of a Permanent Magnet Linear Motor”. Elsevier, Control Engineering Practice No 16 pp 787-797, july 2008. doi:10.1016/j.conengprac.2007.08.003

[24] Bouchard, M., “New Recursive-Least-Square Algorithms for Nonlinear Active Control of Sound and Vibration Using Neural Networks”. IEEE Transactions on Neural Networks, Vol. 12, No. 1. January 2001.

[25] Luiz, G., Teixeira, R. and Ribeiro, J., “A Neural Network-Based Direct Inverse Control for Active Control of Vibrations of Mechanical Systems”. IEEE VI Brazilian Symposium on Neural Networks January 2000. ISBN: 0-7695-0856-1.

[26] Kirkham, C. y Cambio, R., “Misfire Detection Including Confidence Indicators Using a Hardware Neural Network” . SAE Technical papers, Electronic Engine Controls 2006. DOI: 10.4271/2006-01-1349

[28] N. Kara Togunm and S. Baysec, “Prediction of torque and specific fuel consumption of a gasoline engine by using artificial neural networks” Elsevier: Applied Energy, September 2009. doi:10.1016/j.apenergy.2009.08.016

[29] W.T Thomson, Theory of Vibration with Applications, George, Allen and Unwin, London, 1981. ISBN 0-04-. 620012-6

[30] D. Inman, “Engineering vibration”. Prentice Hall, NY 1994. ISBN: 0132281732

[31] S.S. Rao. “Mechanical vibrations” Tercera edición Adison Wesley Publishing Company. 1995. ISBN: 0130489875.

[32] C.R. Fuller, J. Eliott, P.A. Nelson “Active control of vibration” Second Edition. Academic Press. 1997. ISBN-10: 0122694406, ISBN-13: 9780122694400

[33] Cirera E. A. “Control por regimenes deslizantes muestreados para aislamiento activo y vibraciones”. International Conference Science and technology for development CIMAF, 1999. Memorias II Simposium de Control Automático. CIMAF 99 pp. 12-18

[34] Enríquez J., “Seguimiento de trayectorias y atenuación de vibraciones en sistemas tipo masa-resorte-amortiguador utilizando regímenes deslizantes y plenitud diferencial”, Tesis de maestría, CINVESTAV, IPN, México DF, 2002.

[35] Mariaca, C., “Topologías, aprendizaje y estabilidad de Redes Neuronales Hibridas con aplicaciones en procesos biotecnológicos no lineales”, Tesis de doctorado, CINVESTAV, IPN, México DF, 2009.

[36] Escalante, S., “Identificación, filtraje y control de sistemas no lineales usandoRedes Neuronales Recurrentes con el aprendizaje Recursivo de Levenberg-Marquardt”, Tesis de maestría, CINVESTAV, IPN, México DF, 2006.

[37] Ali Kazemy, Seyed Amin Hosseini and Mohammad Farrokhi Second Order Diagonal Recurrent Neural Network. , IEEE Transactión on NN`s , 2007. pp. 251-256. DOI. 10.1109/isie.2007.4374607

[38] Hush, D. R. and B. G. Horne, “Progress in supervised Neural Networks” IEEE Signal Processing Magazine, pp. 8-39, Jan. 1993. DOI 10.1109/79.180705

[39] Van Der Smagt, P., “Minimisation methods for training feedforward Neural Networks”,Neural Networks, Vol. 7, No. 1, pp. 1-11, 1994. doi:10.1016/0893-6080(94)90052-3

[40] Billings, S.A., Jamaluddin H.B., and Chen, S., “A comparison of the backpropagation and Recursive Prediction Error algorithms for Training Neural Networks”, Mechanical Systems and Signals Processing, Vol. 5, No.3, pp. 233-235, 1991. doi:10.1016/0888-3270(91)90045-7

[41] Asirvadam, V. S., McLoone, S. F., and Irwin G. W., “Parallel and separable Recursive Lavenberg-Marquardt Training Algorithm”, IEEE Workshop on Neural Networks for Signal Processing, 2002, Proceedings of the 2002 12th, pp. 129-138, 4-6 Sept. 2002. DOI: 10.1109/NNSP.2002.1030024

[42] Asirvadam, V. S., “Adaptive Regularizer for Recursive Neural Network Training Algorithms” The 11th IEEE International Conference on Computational Science and Engineering – Workshops, 2008. DOI: 10.1109/CSEW.2008.55

[43] Ngia, L. S., and Sjöberg J., “Efficient Training of Neural Nets for Nonlinear Adaptive Filtering Using a Recursive Levenberg-Marquardt Algorithm” , IEEE Trans. on Signal Processing, Vol. 48, pp. 1915-1927, July 2000. DOI: 10.1109/78.847778

[44] Ngia, L. S., and Viberg, M., “Adaptive Neural Nets Filter Using a Recursive Levenberg-Marquardt search direction”, IEEE Signals, Systems & Computer, Vol.1, pp. 697-701, Nov. 1998. DOI: 10.1109/ACSSC.1998.750952

[45] Govindhasamy, J. J., McLoone, S. F. and Irwin, G. W., “Second-Order Training of Adaptive Critics for Online Process Control”, IEEE Transactions on Systems, MAN, and Cybernetics—part B: CYBERNETICS, VOL. 35, NO. 2, APRIL 2005. DOI: 10.1109/TSMCB.2004.843276

[46] Hagan, M., “Neural Network Design”, PWS Publishing Company, 1996. **ISBN**: 0-9717321-0-8

[47] Hagan, M. and Menhaj, M., “Training Feedforward Networks With the Marquardt Algorithm”, IEEE Trans. On Neural Networks, Vol. 5, No. 6, Nov. 1994. Page(s): 989 - 993. DOI: 10.1109/72.329697.

[48] Fletcher, R., “Practical Methods of Optimization”, Wiltshire, U.K. Wiley, 1997. **ISBN**-10: 0471494631; **ISBN**-13: 978-0471494638

[49] Parlos, A. G., Kil T Chong and Amir F. Atiya, “Application of the Recurrent Multilayer Perceptron in Modeling Complex Process Dynamics”, IEEE Trans. On NN, Vol. 5, No 2, pp. 225-265, March 1994. DOI: 10.1109/72.279189

[50] Geman S., Benenstock E., Doursat R., “Neural networks and the bias/variance dilemma”, Neural Computation 4, pp 1–58, 1992

[51] Laundau, I. Lozano, R., and M’saad M. “Adaptive control”, Springer, Gran Bretaña, 1998. **ISBN**-10: 354076187X; **ISBN**-13: 978-3540761877

[52] Chen, Fu-C and Khalil, H.K., “Adaptive Control of Nonlinear Systems Using Neural Network”, Int. J. Control, Vol. 55, No. 6, pp. 1299-1317, 1992. ISBN: 0-7803-2685-7

[53] Rovithakis, G. A. and Christodoulou, M.A., “Adaptive Control of Unknown Plants Using Dynamical Neural Networks”, IEEE Trans. On Systems, Man and Cybernetics, Vol. 24, No. 3, pp. 400-412, March 1994. DOI: 10.1109/21.278990

[54] Omatu, S. Khalid, M., and Yusof, R., “Neuro-Control and Its Applications”, London, Springer Verlag, 1995 . **ISBN-13:** 9783540199656

[55] Jagannathan, S and Lewis, F. L., “Identification of Nonlinear Dynamical Systems Using Multilayered Neural Networks”, Automatica, Vol.32, No. 12, pp. 1707-1712,1996 doi: 10.1016/S0005-1098(96)80007-0

[56] Jin, L., Nikiforuk, P. N. and Gupta, M. M., “Adaptive Model Reference Control of Discrete- Time Nonlinear Systems Using Neural Networks”, Control Theory and advanced Technology, Vol. 10, No. 4, Part 3, pp. 1379-1399, Sept. 1995

[57] Bhat, N., Mc Avoy, T. J., “Use of Neural Networks for Dynamic Modeling and Control of Chemical Process System”, Comput. Chem. Eng., 14 (4/5), pp. 575-583, 1990. doi:10.1016/0098-1354(90)87028-N

[58] Breusegem, V. V., Thibault, J., Cheruy, A., “Adaptive Neural Models for On- Lines Prediction in Fermentation”, The Canadian Journal of Chemical Engineerng. J. Chem. Eng. 69 (2), pp. 481-487, April 1991. doi: 10.1002/cjce.5450690212

[59] Barouh, P Georgieva and J Barrera Cortes, An Integral Plus State Adaptive Neural Control of Aerobic Continuos Stirred Tank Reactor, Enginnerring Letters, vol 13, No 2., pp 84-92.

[60] Barouh, L. A. Hernandez, J.R. Valle, J. Barrera Cortes, Sliding Mode control of aerobic bioprocess using recurrent neural identifier. Preprints of the 16th IFAC World Congress, Prague, Czech Republic, July 3-8 2005, paper Tu-A21-TO/4, 01526.pdf

[61] Xiaolin Hu and P. Balasubramaniam, "Chapter 4: Recurrent Neural Network Identification and Adaptive Neural Control of Hydrocarbon Biodegradetion Processes", edited by In-Tech, 2008. ISBN 978-953-7619-08-4