ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

MÉTODO BASADO EN LA RED NEURONAL *PROJECTION PURSUIT LEARNING* PARA LA ESTIMACIÓN DE DIRECCIONES DE LLEGADA PARA UN ARRAY DE ANTENAS

AUTOR: Francisco Javier Mula Cruz

DIRECTOR(ES): Juan Pascual García
Fernando Quesada Pereira

Junio / 2005

Autor	Francisco Javier Mula Cruz
E-mail del Autor	mula@ono.com
Director(es)	Juan Pascual García, Fernando Quesada Pereira
E-mail del Director	juan.pascual@upct.es, fernando.quesada@upct.es
Título del PFC	MÉTODO BASADO EN LA RED NEURONAL *PROJECTION PURSUIT LEARNING* PARA LA ESTIMACION DE DIRECCIONES DE LLEGADA PARA UN ARRAY DE ANTENAS
Descriptores	Projection Pursuit Learning , Redes Neuronales, Dirección de llegada, array de antenas.

Resúmen

En este proyecto se ha implementado la Red Neuronal PPL (Projection Pursuit Learning) para la estimación de los ángulos de las señales que llegan a una agrupación de antenas. Para ello se compararán los resultados obtenidos con los de la red Neuronal RBF que viene programada en la toolbox de Matlab 6.5.

Antes de llegar a los resultados finales se explicará en que consiste la estimación de ángulos de llegada y que son las redes neuronales para entrar en detalle en el entrenamiento de la Red Neuronal PPL.

Titulación	Ingeniero de Telecomunicación
Intensificación	Sistemas y Redes de Telecomunicación
Departamento	Tecnología de la Información y las Telecomunicaciones
Fecha de Presentación	Junio- 2005

CAPÍTULO 1
ESTIMACIÓN DE LA DIRECCIÓN DE LLEGADA

1.1 INTRODUCCIÓN

Los nuevos sistemas *wireless* (inalámbricos) como las comunicaciones celulares, sistemas de comunicación personal (PCSs), y redes de comunicación personal (PCNs) deben satisfacer un incremento en la demanda de cobertura, capacidad y calidad de servicio. Para cumplir el anterior propósito nace la necesidad de desarrollar herramientas para la mejora de los diferentes elementos que conforman un sistema moderno de comunicación. Una de las estrategias para mejorar el rendimiento de los sistemas de comunicación inalámbricos consiste en la aplicación de avanzadas técnicas de procesamiento de señal a aquellas señales provenientes de diversas antenas de una agrupación.

Los sistemas *wireless* presentan limitaciones para el procesamiento de señales temporales de modo que no son capaces de resolver problemas como la interferencia cocanal (CCI), que constituye el factor más serio en la limitación de capacidad del sistema. La probabilidad de interferencia crece conforme aumenta el número de usuarios. Este fenómeno se da especialmente en zonas de gran población en las que por tanto también existirá un gran número de usuarios. Para aumentar el número de usuarios que un sistema puede albergar se aplican técnicas de acceso múltiple al medio. El ancho de banda disponible se puede dividir en fracciones y asignar cada fracción a un usuario distinto dando lugar al acceso múltiple por división en la frecuencia (FDMA). La misma estrategia se puede aplicar en el dominio temporal produciendo el acceso múltiple por división en el tiempo (TDMA). Una técnica que permite compartir el ancho de banda y el tiempo disponible es el acceso por división del código (CDMA) en el que a cada usuario se le asigna un código diferente. Sin embargo las anteriores técnicas de acceso múltiple presentan limitaciones relacionadas con la limitación del ancho de banda disponible, el periodo temporal utilizable o el número de códigos de usuario.

Debido a las limitaciones de los recursos de los que dispone el sistema de comunicaciones es necesario implementar diversas técnicas para incrementar la capacidad del sistema [1]. Un objetivo prioritario para lograr el propósito anterior es la reducción de las interferencias que llegan al receptor tanto si es móvil como si se trata de un receptor fijo. En un sistema que utilice CDMA como técnica de acceso al medio cada usuario no deseado equivale a una fuente de ruido o interferencia. Por lo tanto el sistema no puede incrementar el número de usuarios de forma indefinida debido a que llegado a cierto nivel de señal a ruido en el receptor la comunicación es imposible.

El problema mencionado se puede solucionar empleando una agrupación o array de antenas como primer elemento del sistema receptor. Si se aplica a la agrupación de antenas una técnica de conformación de haz se puede entonces guiar el haz de radiación de la agrupación hacia los usuarios móviles con los que se desea la comunicación y a su vez aplicar nulos de radiación en la dirección de las fuentes de interferencia. Dentro de las técnicas de conformación de haz podemos distinguir entre aquellas que necesitan el

conocimiento de la dirección de llegada ("Direction of Arrival" o DOA) de las señales al array de antenas y aquellas que no hacen uso de esta información [2].

Para aprovechar de una forma más eficiente el espectro radioeléctrico y al mismo tiempo incrementar la capacidad del sistema de comunicación celular se emplea la técnica de reutilización de frecuencia. Dicha estrategia consiste en emplear la misma frecuencia en dos celdas separadas una distancia suficiente para que los usuarios de una celda no interfieran con los usuarios de la otra celda. Las celdas que utilizan las mismas frecuencias están separadas una distancia D que está directamente relacionada con el tamaño de la celda C. Una forma de incrementar el número de usuarios del sistema consiste en aumentar el tamaño de celda C. Esta estrategia conlleva mayores distancias de reutilización D a cambio de incrementar el ratio portadora-interferencia.

Otra forma de encarar el problema del incremento de usuarios se basa en permitir celdas co-frecuencia más próximas de forma que se produce una mayor reutilización de frecuencias. Para poder utilizar celdas co-frecuencias menores es necesario aplicar un método compuesto de dos etapas diferentes. En un primer paso se emplea un algoritmo de cálculo de la dirección de llegada (DOA) de las señales como el algoritmo de clasificación de múltiples señales (MUSIC o "multiple signal classification"). Dicho algoritmo permite la localización del usuario deseado así como de los usuarios cocanal del primero. El algoritmo MUSIC posee una alta resolución para señales con una separación angular pequeña. Una vez se ha calculado la dirección de llegada de cada tipo de señal se actúa como en el caso de los sistemas CDMA. La utilización de técnicas de conformación de haz permite dirigir el máximo de radiación hacia el usuario deseado a la vez que se coloca un nulo de radiación en cada uno de las fuentes de interferencia del slot de frecuencia dado.

El rechazo de las interferencias es muy importante y a veces representa una vía para incrementar la capacidad del sistema permitiendo celdas de co-frecuencia más próximas o lóbulos adicionales para el reuso de frecuencia. Para resolver este problema usaremos primero un algoritmo de superresolución ó en nuestro caso una red neuronal para la estimación de la dirección de llegada (DOA) para localizar a los usuarios. Seguidamente una agrupación de antenas adaptativas puede ser utilizado para guiar su haz de radiación hacia los usuarios móviles de mayor interés y a su vez poner nulos hacia las otras fuentes de interferencia que se encuentran en la misma ranura ó slot de frecuencia.

La dificultad de los algoritmos de superresolución como el MUSIC [3] para la detección del ángulo de llegada reside en su implementación en tiempo real debido a su complejidad computacional. Por eso vamos a estudiar el comportamiento de las redes neuronales en este campo ya que pueden operar en tiempo real debido a su rapidez computacional.

Además los convencionales conformadores de haz requieren antenas altamente calibradas con elementos de idénticas propiedades y por ello a veces ocurre que los algoritmos de superresolución pobremente pueden adaptar los elementos de fallos u otras fuentes de error. Las redes Neuronales basadas en agrupación de antenas no tienen este problema ya que el comportamiento de la antena (Uniforme, no uniformemente espaciada, formada por elementos no uniformes, etc.) puede ser incorporado en su entrenamiento y bajo distintas circunstancias de poner como trabajo.

1.2 DETECCIÓN DE LA DIRECCIÓN DE LLEGADA MEDIANTE REDES NEURONALES

En esta sección vamos a explicar el modelo de propagación que vamos a seguir, es decir, la formación de las señales que llegan a la agrupación o array de antenas y la respuesta de la misma agrupación a las direcciones de incidencia.

Consideremos una agrupación lineal compuesta de *M* elementos, donde *K* (K < M) es el número de ondas planas de banda estrecha centradas a la frecuencia w_0 que inciden en la agrupación procedentes de direcciones $\{\theta_1, \theta_2, \ldots, \theta_K\}$. Usando una representación compleja, la señal recibida en el elemento *i–ésimo* de la agrupación puede ser escrita de la forma:

$$x_i(t) = \sum_{m=1}^{K} s_m(t) e^{-j(i-1)k_m} + n_i(t) \qquad (1)$$

i = 1,2,…, M

Donde $s_m(t)$ es la señal de la *m*-ésima onda, $n_i(t)$ es la señal de ruido recibida en el *i-ésimo* sensor de la agrupación y k_m viene dado por:

$$k_m = \frac{w_0 d}{c} \sin(\theta_m) \qquad (2)$$

Donde *d* es el espaciado entre los elementos de la agrupación y *c* es la velocidad de la luz. Así el modelo de propagación se puede describir de la siguiente forma:

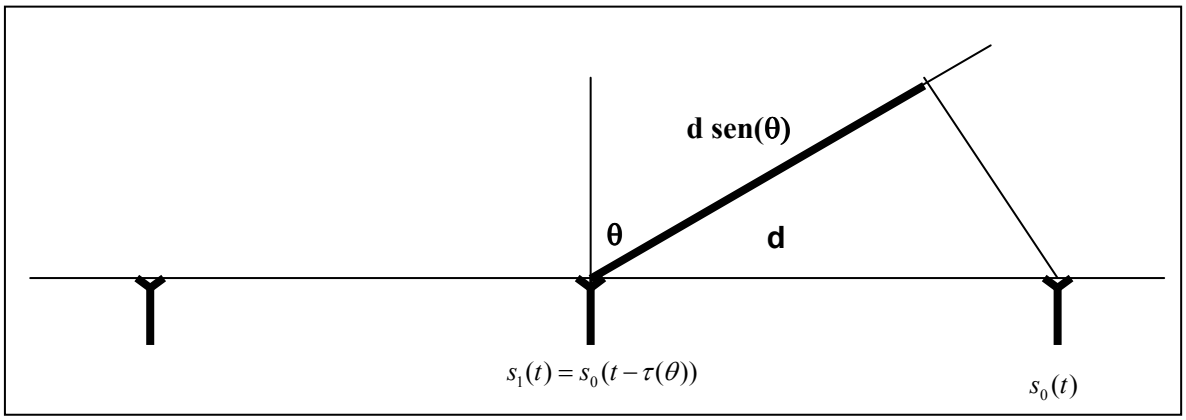


Figura 1.1. Modelo de propagación.

Para que no exista ambigüedad en el ángulo de llegada: $d \leq \lambda/2$

Seleccionando el primer elemento como referencia, la respuesta de la agrupación de antenas a una onda plana incidiendo desde el ángulo θ_i es

$$a(\theta_m) = [1 \;\; e^{-jk_m} \;\; e^{-j2k_m} \;\; \dots \;\; e^{-j(M-1)k_m}]^T \qquad (3)$$

Así formamos la matriz **A (**M x K**)** que es la respuesta de la agrupación a las distintas direcciones de incidencia:

$$\mathrm{A} = [a(\theta_1) \dots a(\theta_m) \dots a(\theta_K)] \qquad (4)$$

Usando notación vectorial podemos escribir la salida en una matriz:

$$\mathbf{X}(\mathrm{t}) = \mathbf{AS}(\mathrm{t}) + \mathbf{N}(\mathrm{t}) \quad (5)$$

Donde **X**(t) es el vector de salida de señales del array, **N**(t) es el vector de ruido y **S**(t) es el vector que contiene los valores de las señales que inciden en el array.

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_M(t) \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ e^{-jk_{11}} & e^{-jk_2} & \cdots & e^{-jk_M} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-jk_1(M-1)} & e^{-jk_2(M-1)} & \cdots & e^{-jk_M(M-1)} \end{bmatrix} \begin{bmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_K(t) \end{bmatrix} + \begin{bmatrix} n_1(t) \\ n_2(t) \\ \vdots \\ n_M(t) \end{bmatrix} \qquad (6)$$

Las señales de ruido recibidas en los diferentes sensores son señales de ruido blanco estadísticamente independientes, de media cero y varianza uno.

Por lo tanto la matriz de correlación $\boldsymbol{R}$ de las señales recibidas puede ser expresada:

$$R = E\{\mathbf{X}(t)\mathbf{X}(t)^H\} = \mathbf{A}E[\mathbf{S}(t)\mathbf{S}^H(t)]\mathbf{A}^H + E[\mathbf{N}(t)\mathbf{N}^H(t)] \qquad (7)$$

La operación que se realiza sobre la agrupación de antenas puede verse como un mapeo o función G: $\mathbf{R}^k \rightarrow \mathbf{C}^M$ desde el espacio de direcciones de llegadas $\{\boldsymbol{\Theta} = [\theta_1, \theta_2, ..., \theta_K]^T\}$ al espacio de salida de los sensores $\{\mathbf{X} = [x(t)_1, x(t)_2, ..., x(t)_M]^T\}$. El mapeo inverso o función inversa F: $\mathbf{C}^M \rightarrow \mathbf{R}^K$ se puede utilizar para hallar las direcciones de llegada de las señales a partir de los datos obtenidos en la matriz de correlación.

El problema consistente en el cálculo analítico de la función F es complicado. Se puede emplear una red neuronal que aproxime la función multidimensional F. **Para poder llevar a cabo esta tarea la red neuronal debe ser entrenada con una serie de datos del espacio $\mathbf{C^M}$ y los correspondientes datos del espacio de direcciones de llegadas $\mathbf{R^k}$.** Una vez entrenada la red neuronal se utiliza para calcular nuevas direcciones de llegada $\boldsymbol{\Theta}$ a partir de cualquier nueva combinación de datos $\mathbf{X}$.

En este proyecto se aplicará la red neuronal conocida como Projection Pursuit Learning para calcular las direcciones de llegadas. A continuación se explicará el funcionamiento general de una red neuronal y cómo pueden ser aplicadas para tratar problemas del tipo expuesto en este punto.

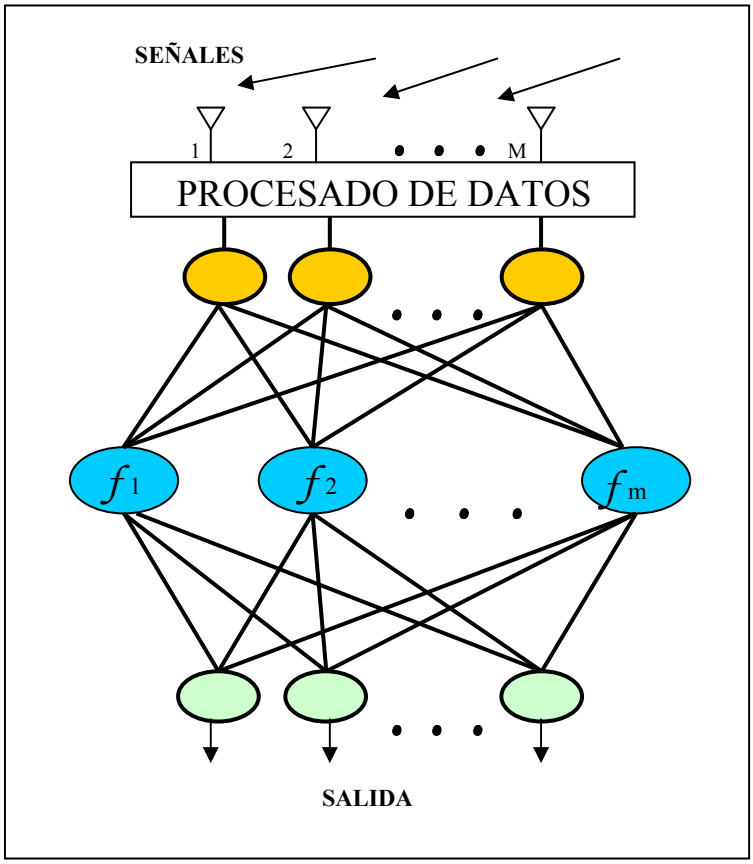


Figura 1.2. Diagrama de Bloques de una red neuronal PPL con una etapa de pre -procesado

El esquema que vamos a utilizar para resolver nuestro problema de detección de ángulos de llegada se puede ver en la *Figura 1.2*. Se puede observar una primera etapa de pre-procesado para transformar las señales que llegan a la agrupación de antenas en entradas para la red neuronal, dicho proceso viene descrito en el capítulo 4. Una vez introducidas las entradas en la red neuronal obtenemos a la salida los ángulos de incidencia de las señales recibidas en la agrupación, pudiendo de esta forma detectar donde se encuentra un mayor número de usuarios, por ejemplo dentro de una celda de telefonía móvil.

CAPÍTULO 2
INTRODUCCIÓN A LAS REDES NEURONALES

2.1 INTRODUCCIÓN A LAS REDES NEURONALES

Algo tan sencillo para el ser humano como es reconocer una cara de otra persona, es el tipo de problema que no es tan fácil de resolver por la vía algorítmica, esto es, la resolución de un determinado problema mediante la obtención de un algoritmo que manipula los datos relativos a dicho problema.

Debido a la complejidad que encuentran las técnicas algorítmicas en tratar diversos tipos de tareas, desde finales de los 50 se ha venido investigando en un conjunto de técnicas que utilizan un enfoque diferente para resolver problemas. Este conjunto de técnicas y herramientas se bautizó con el nombre de Inteligencia Artificial (IA), porque lo que se pretendía era que los ordenadores presentaran un comportamiento inteligente, entendiendo por esto que supieran hacer frente a ciertos problemas de una manera similar a como lo hacen los seres humanos.

Dentro de la IA, se trabajó en dos enfoques distintos. Por un lado, se desarrolló lo que se conoce como el enfoque simbólico. Este enfoque asienta sus bases en la manipulación de símbolos en vez del mero cálculo numérico. La realidad se plasma por medio de una serie de reglas. Herramientas como la lógica de predicados, nos permiten manipular los símbolos y las reglas para obtener nuevas reglas. Este enfoque se presta a ser muy útil en ciertos tipos de problemas aunque en general tiene la desventaja de que a la hora de buscar la solución a un determinado problema los métodos de deducción presentan una explosión combinatoria que hace que requiera bastante tiempo de cálculo.

El otro enfoque tradicional es el enfoque conexionista que consiste en desarrollar un sistema formado por pequeñas unidades de cálculo, en cierta medida muy simples, y hacer mediante conexiones entre ellas, que todo el conjunto sea capaz de resolver una cierta clase de problemas.

La idea que animó el modelo conexionista fue la de imitar el sistema de computación más complejo de los que se conocen hasta ahora, que es el cerebro. El cerebro esta formado por millones de células llamadas neuronas. Estas neuronas son unos procesadores de información muy sencillos con un canal de entrada de información (dendritas), un órgano de cómputo (soma) y un canal de salida de información (axón).

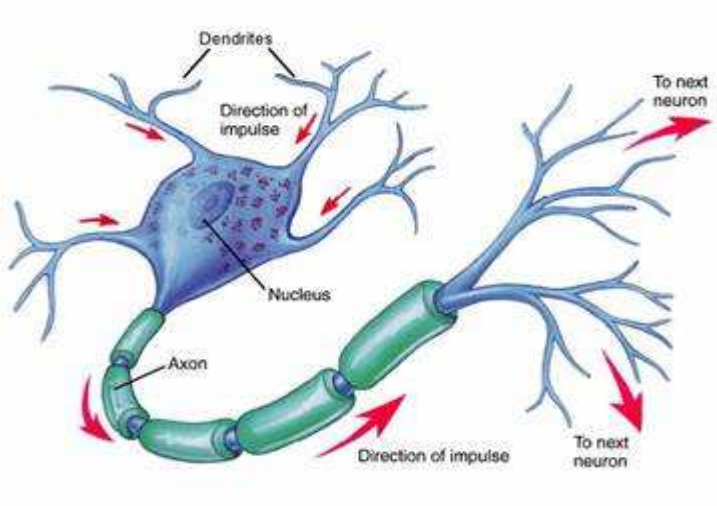


Figura 2.1. *Representación de una neurona biológica*

Por lo tanto en el enfoque conexionista las simples unidades de cálculo son similares a las neuronas y al igual que las unidades de cálculo humanas están interconectadas unas con otras. Por lo tanto los **sistemas que se desarrollan bajo el enfoque conexionista se denominan redes neuronales (RNA).**

Teniendo en cuenta que el cerebro presenta las cualidades de **procesamiento paralelo, procesamiento distribuido y adaptabilidad**, un sistema RNA tiene también estas características. El sistema resulta ser intrínsicamente paralelo porque cada neurona puede realizar cierta cantidad de cálculos sin necesidad de esperar la respuesta de otras neuronas.

El sistema es distribuido. Esto quiere decir que la información no se almacena localmente en ciertas zonas concretas de la RNA sino que se halla presente por toda ella. De igual forma, la computación es también distribuida. Al calcular la respuesta de la red neuronal, intervienen todos y cada uno de los procesadores elementales, los cuales se hallan distribuidos por toda la arquitectura de la red. Además **dicho carácter distribuido hace que la red presente tolerancia a fallos (**si se pierde una parte de las neuronas no se pierde toda la información).

Una red neuronal presenta además un **grado de adaptabilidad** que **se concreta en las capacidades de aprendizaje y generalización**. Por aprendizaje entendemos la capacidad para recoger información de las experiencias y utilizarla para actuar ante situaciones futuras. La generalización está íntimamente relacionada con el aprendizaje, que podría definirse como la capacidad para abstraer la información útil, más allá de los casos particulares. De esta manera, la RNA es capaz de responder ante casos desconocidos de manera apropiada.

Además de las citadas anteriormente podemos destacar las siguientes características de una red neuronal en:

1) **No – linealidad**: Una neurona es básicamente un mecanismo no lineal, y de esta forma una Red Neuronal formada por neuronas interconectadas entre si, es también un sistema no lineal. Esta propiedad es muy importante, especialmente si el sistema que la red intenta modelar es no lineal.

2) **Mapeo de entrada–salida**: Durante el proceso de aprendizaje la red neuronal es alimentada con una colección de pares de datos de entrada-salida conocidos. Una vez finalizada la fase de entrenamiento la red neuronal representa una determinada relación no lineal entre la entrada y la salida. La mencionada relación no lineal constituye un mapeo entre entrada y la salida.

3) **Tipos de Aprendizaje**: Podemos diferenciar dos tipos de aprendizaje. Un conocido paradigma en el entrenamiento de una Red Neuronal es el *Aprendizaje Supervisado.* En este proceso tenemos una colección de pares de entrada – salida conocidas para el aprendizaje de la red. Si cuando presentamos uno de los datos de entrada a la red, la salida calculada no coincide con la deseada los pesos de las interconexiones de la red y el resto de parámetros son modificados para acercarse más a esta salida conocida. Este proceso se repite para todos los pares

de entrenamiento hasta que llegamos a un punto en el cual para cada entrada la salida de la red coincide con la salida deseada. El segundo tipo de aprendizaje se denomina Aprendizaje No-Supervisado. En este caso los parámetros de la red se modifican atendiendo únicamente a los datos de entrada. De este modo los datos de salida no son tomados en cuenta para calcular los parámetros de la red.

4) **Adaptabilidad**: Las redes neuronales tienen la capacidad de *adaptar* sus pesos para cambiar el entorno en el que operan. Por ejemplo podemos entrenar a una Red Neuronal para que reconozca las letras A, B y C del abecedario. Si después queremos que reconozcan dos letras nuevas D y E, solamente tendríamos que re-entrenar a nuestra red adaptando los pesos al nuevo caso.

5) **Implementación en VLSI**: La naturaleza paralela de una Red Neuronal la hace potencialmente rápida en la computación de ciertas tareas usando tecnología VLSI y así la convierte en una herramienta para tareas en tiempo real como procesamiento de señal o reconocimiento de patrones.

6) **Uniformidad de análisis y diseño**: Las Redes Neuronales son procesos de información Universales en el sentido de que la misma notación y diseño es utilizado en todos los campos de aplicación de las redes.

2.2 ARQUITECTURAS DE UNA RED NEURONAL

La arquitectura de una red neuronal queda definida por los elementos de procesado que la constituyen y como éstos están interconectados." Los elementos de procesado (EPs) en una Red Neuronal se distribuyen por CAPAS: conjunto de EPs que se encuentran en el mismo nivel en la estructura. Así según el camino que sigue la información en la red tenemos:

a) ***Redes Feedforward***: sistemas en donde las salidas de las neuronas de una capa sólo se propagan hacia las neuronas pertenecientes a la capa siguiente.

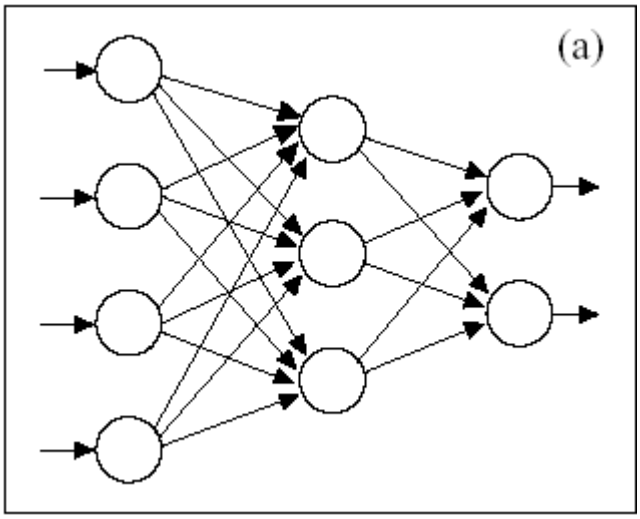


Figura 2.2. Conexión hacia delante (redes progresivas)

b) ***Redes Feedlateral***: sistemas en donde las salidas de las neuronas de una capa, pueden ser entradas a las neuronas de la misma capa. (Aprendizaje Competitivo).

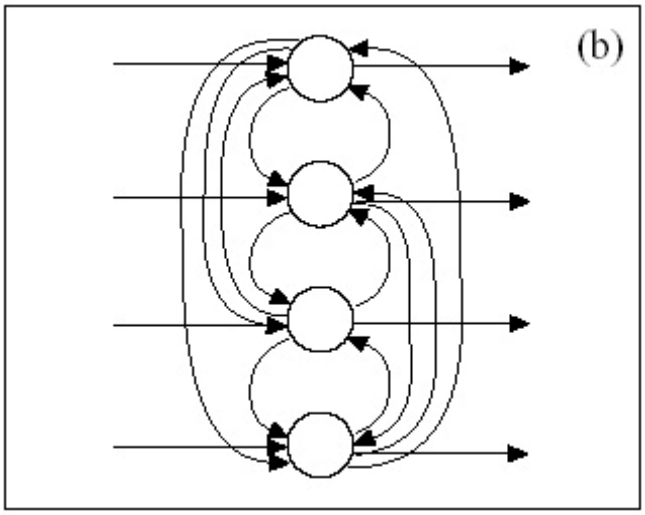


Figura 2.3. Estructura Feedlateral de una red neuronal

c) ***Redes Feedback (o recurrentes)***: Cuando las salidas pueden estar conectadas como entradas de neuronas de niveles previos o del mismo nivel, incluyéndose ellas mismas, la red es de propagación hacia atrás. Las redes de propagación hacia atrás que tiene lazos cerrados son sistemas recurrentes.

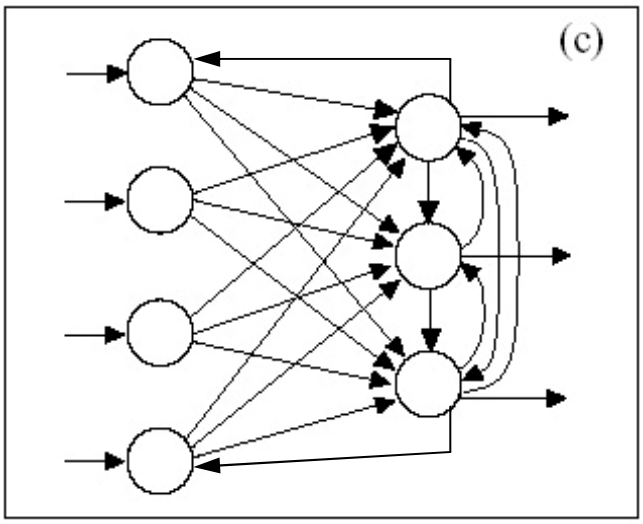


Figura 2.4. Estructura Feedback de una red neuronal

2.3 MODELO DE UNA NEURONA

Una neurona es una unidad de procesado de información fundamental para el funcionamiento de la Red Neuronal. La *figura 2.5* muestra el modelo de una neurona donde podemos identificar tres elementos:

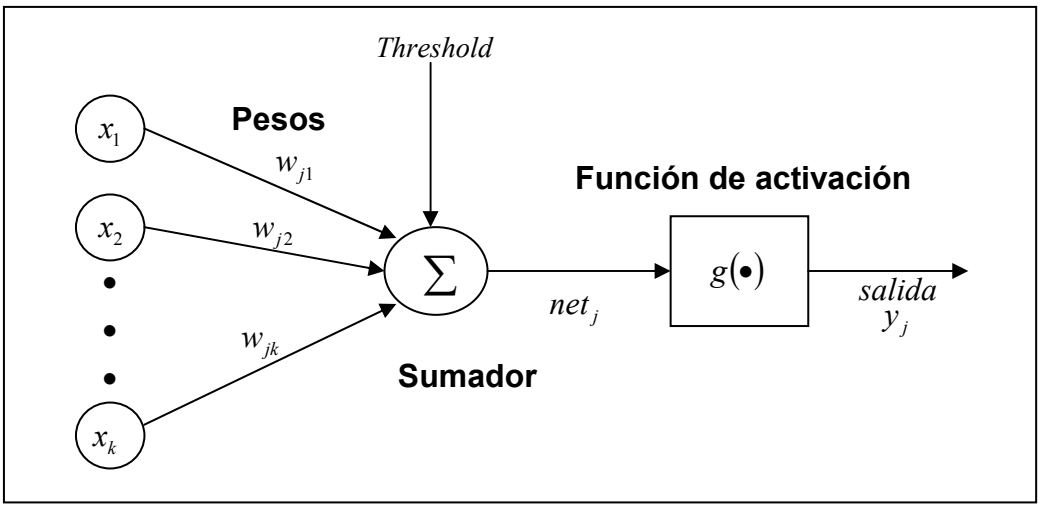


Figura 2.5. Modelo de una neurona

1) Un **conjunto de sinapsis o conexiones** cada una caracterizada por un peso. Una neurona *k* es conectada a otra *j* con su respectivo peso w_{jk}.

2) Un **sumador** de las señales de entrada.

3) Una **función de activación** para limitar la amplitud de la señal de salida de una neurona. Dicha función de activación suele ser de tipo no-lineal.

El modelo de la neurona también incluye un *Threshold* que es en definitiva un peso mas con valor 1 ó -1 que será modificado a través el proceso de aprendizaje de la red.

2.4 TIPOS DE FUNCIÓN DE ACTIVACIÓN

La función de activación $g(\bullet)$ define la salida de una neurona y existe una gran variedad de funciones que se han ido utilizando en las distintas bibliografías. Algunas de las típicas funciones de activación son:

- **Función Threshold**, para este tipo de activación tenemos:

$$g(net) = \begin{cases} 1 : si & net \geq 0 \\ 0 : si & net < 0 \end{cases}$$

- **Función Piecewise – lineal,** para este tipo de activación tenemos:

$$g(net) = \begin{cases} 1 : si & net \geq 1/2 \\ net : si & 1/2 > net > -1/2 \\ 0 : si & net \leq -1/2 \end{cases}$$

- **Función Sigmoidal,** esta es la función de activación mas común empleada en la construcción de Redes Neuronales MLP (MultiLayer Perceptron) que más adelante veremos en el punto 2.6.2:

$$g(net) = \frac{1}{1 + \exp(-a \cdot net)}$$

- **Función Gaussiana,** esta es la función de activación mas común empleada en la construcción de Redes Neuronales RBF que también veremos a continuación en el punto 2.6.3:

$$\varphi(x) = \exp\left(-\frac{1}{2\sigma^2} \left(\|x - c\|^2 \right) \right)$$

c: Centro de la Gaussiana
σ^2 : Varianza

- **Funciones de Hermite,** En nuestro caso es la función de activación que emplearemos y mas adelante serán explicadas de una forma mas detallada. Su expresión viene definida por:

$$g(net) = \sum_{r=1}^{R} c_r h_r(net)$$

h_r : Polinomio de Hermite de orden r

2.5 PROCESOS DE APRENDIZAJE

En el punto 2.1 hicimos una breve explicación de los dos tipos principales de aprendizaje que puede llevar a cabo una red neuronal. A continuación vamos a explicar de forma mas detallada las estrategias de aprendizaje de una red:

1. Entrenamiento supervisado

Es aquel tipo de entrenamiento en el que tenemos conocimiento del entorno en el que estamos trabajando pero desconocido por la red neuronal así tenemos un conjunto de entrada y salidas deseadas que queremos que nuestra red aproxime. Para cada una de estas entradas la red neuronal nos dará una respuesta que compararemos con la salida que deseamos, obteniendo así una función de error que nos permitirá el reajuste de los parámetros de la red para acercarse cada vez mas a estas salidas deseadas.

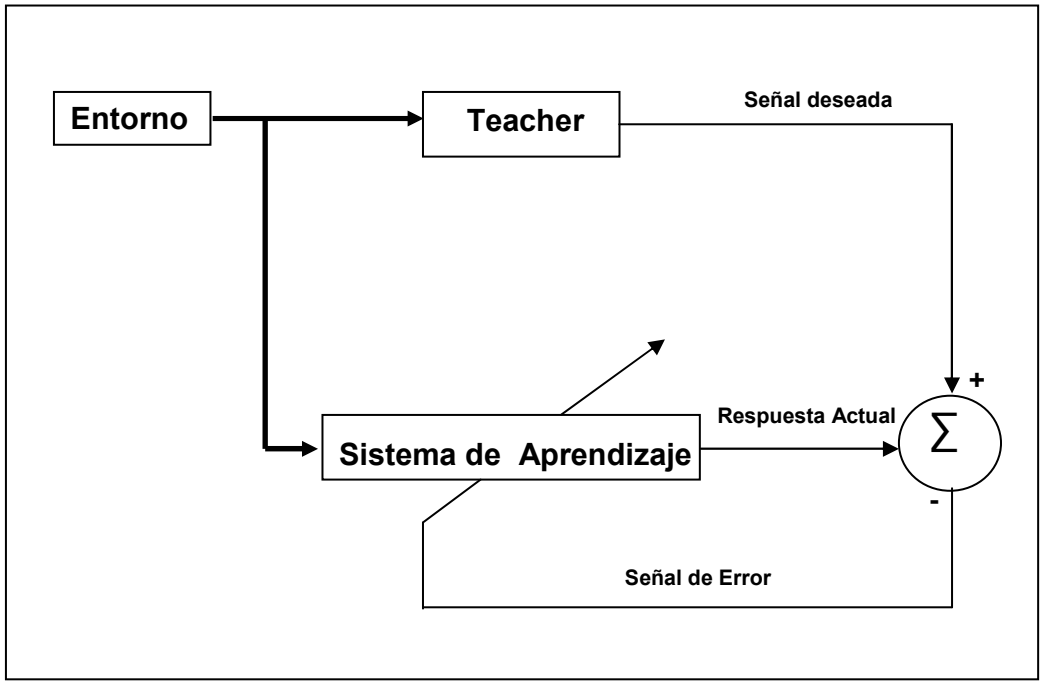


Figura 2.6. Ilustración de un proceso de aprendizaje supervisado

A este tipo de entrenamiento supervisado se conoce como **aprendizaje por corrección de error.** Dentro del entrenamiento supervisado tenemos además de este último:

- **Aprendizaje por Refuerzo**

Aprendizaje más lento que el anterior. No se dispone de un ejemplo completo del comportamiento deseado. No se conoce la salida deseada exacta para cada entrada. Se conoce como debería de ser el comportamiento de manera general ante diferentes entradas. Es un aprendizaje on- line. Relación de entrada-salida a través de un proceso de éxito o fracaso, produciendo una señal (señal de refuerzo) que mide el buen funcionamiento del sistema.

Esta seña "señal de refuerzo" está caracterizada por el hecho de que es menos informativa que en el caso de aprendizaje supervisado mediante ejemplos. Los pesos se ajustan en base a la señal de refuerzo basándose en un mecanismo de probabilidades. "*Si una acción tomada por el sistema de aprendizaje es seguida por un estado satisfactorio, entonces la tendencia del sistema a producir esa particular acción es reforzada. En otro caso, la tendencia del sistema a producir dicha acción es disminuida*". La función del supervisor es más la de un crítico que la de un maestro.

- **Aprendizaje Estocástico**

Este tipo de aprendizaje consiste básicamente en realizar cambios aleatorios en los valores de los pesos y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

2. Entrenamiento no supervisado ó “self- organized”

Las redes con aprendizaje no supervisado, no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas, es decir, no disponemos del conjunto con las salidas deseadas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta. Por ello, suele decirse que estas redes son capaces de autoorganizarse.

De todas formas podemos proveer a la red de una medida de calidad de la representación que la red requiere para el aprendizaje y los parámetros de la red serán optimizados respecto a esta medida.

A continuación en la *figura 2.7* podemos ver el esquema del proceso de aprendizaje no supervisado.

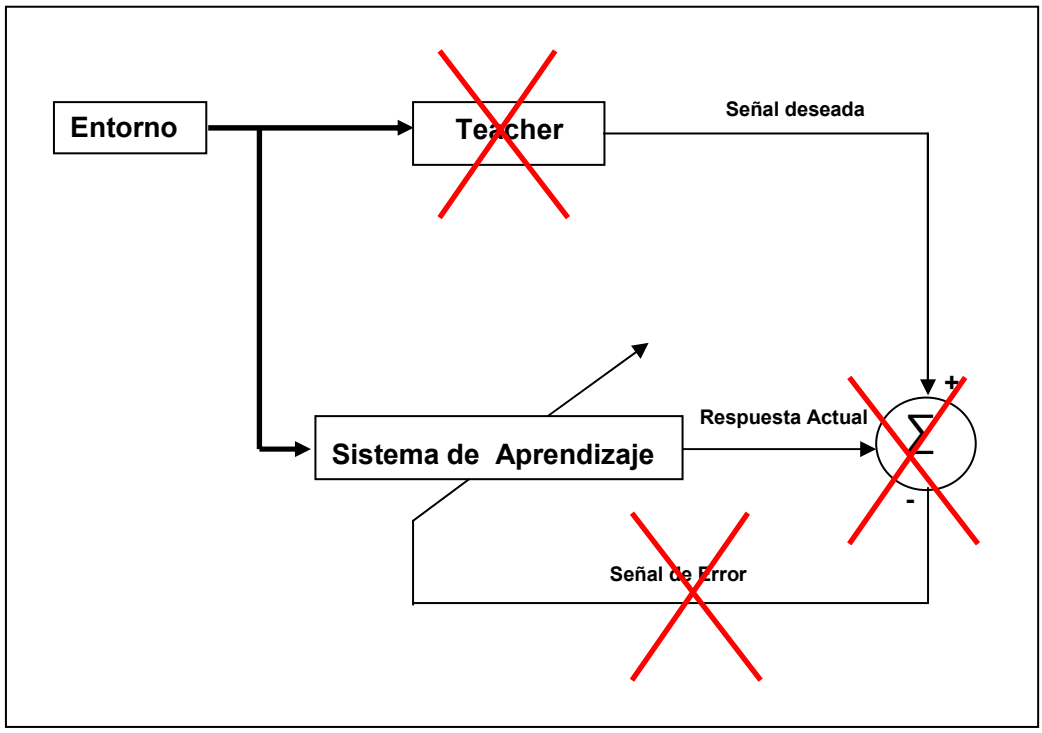


Figura 2.7. Ilustración de un proceso de aprendizaje no supervisado. Las marcas en rojo indican que esos bloques no son posibles.

3. Entrenamiento híbrido

A veces un entrenamiento puramente supervisado no es eficiente y necesitamos una mezcla de ambos. Unas capas de la red tienen un aprendizaje supervisado y otras capas de la red tienen un aprendizaje de tipo no supervisado.

Sea cual sea el método de entrenamiento elegido, la red debe pasar por una fase de testeo. Durante dicha fase, la red es alimentada con una serie de datos de entrada conocidos pero diferentes respecto de los datos de entrenamiento. Así, los pares datos de entrada-salida de test son desconocidos para la red neuronal. Cada salida producida por la red, cuando se presenta un dato de test a la entrada, es comparada con el correspondiente dato de salida conocido. De este modo se comprueba la capacidad de generalización de la red, esto es la capacidad de calcular un valor de salida correcto cuando se presenta a la entrada un dato nunca visto antes por la red.

2.6 CAMPOS DE APLICACIÓN DE LAS REDES NEURONALES

1) **Aproximación**, cuando se requiere diseñar un red neuronal que equivalga a una función que contenga un conjunto de pares de entrada–salida. Los datos de salida pueden haber sido generados bien por un proceso para el que no tenemos

una función analítica que lo represente o bien por una función conocida. En el primer caso se desea diseñar la red para que represente el proceso, de modo que ante nuevas entradas podamos obtener datos de salida fácilmente sin necesidad de reproducir el proceso. En el segundo caso, existe una función o método analítico conocido pero que resulta ser muy costoso de ejecutar en términos de tiempo o recursos consumidos. La red neuronal sustituye la solución analítica a un determinado problema proporcionando valores de salida por un lado rápidamente y por otro sin necesidad de consumir recursos de computación. Para ambos casos la red debe ser sometida a la fase de test. Como se dijo en el punto anterior se presentan nuevos datos a la red de forma que se chequea la capacidad de generalización e interpolación de la red entrenada. Cuando se alimenta la red con un dato de entrada situado entre varios datos de entrada de entrenamiento, la salida producida será resultado de una interpolación no lineal entre los correspondientes valores de salida de entrenamiento. La interpolación llevada a cabo por la red neuronal es de tipo no lineal. Los problemas de aproximación también se denominan problemas de regresión.

2) **Clasificación de patrones**, en este caso cada dato de entrada es asociados a un tipo determinado de clase. Cuando presentamos un dato de entrada o patrón a la red neuronal ésta proporciona la clase a la que pertenece. Los datos de salida por lo tanto pueden tomar una serie de valores discretos. La información contenida en los patrones de entrenamiento y sus respectivas clases asociadas es utilizada por la red para ajustar sus parámetros. Durante la fase de test el conocimiento adquirido en la fase de entrenamiento es utilizado para clasificar nuevos patrones y comprobar así si la capacidad de generalización de la red.

3) **Asociación de patrones,** consiste en construir una memoria distribuida por medio de un mecanismo de asociación**.** Existen dos tipos de mecanismos basados en la asociación, la autoasociación y la heteroasociación. En la autoasociación la red neuronal es entrenada para que almacene una serie de patrones. Generalmente los patrones de entrenamiento son presentados a la red repetidas veces. Durante la posterior fase de test, se presenta una versión distorsionada, mediante la adición de ruido, de los patrones. La red debe entonces recuperar el valor original de los patrones almacenados. En la heteroasociación un conjunto arbitrario de patrones de entrada es asociado con otro conjunto arbitrario de patrones de salida. Generalmente el tipo de entrenamiento en el enfoque autoasociativo es de tipo no-supervisado mientras que en el heteroasociativo es de tipo supervisado.

4) **Predicción**, es un problema de procesamiento temporal de la señal en el que a partir de valores de la señal previos al instante actual, la red será capaz de predecir el valor de la señal correspondiente al instante presente en el que nos encontramos.

En nuestro proyecto nos encontraremos ante un problema de aproximación donde tendremos que estimar los ángulos de las señales que llegan a una agrupación de antenas.

2.7 TIPOS DE REDES NEURONALES

2.7.1 *EL PERCEPTRON MONOCAPA Y MULTICAPA*

El perceptrón monocapa fue de las primeras redes que fueron extensivamente estudiadas a finales de 1950 y principios de los 60, [6]-[13]. Debido a sus limitaciones apareció la red perceptrón multicapa y aunque ya era conocida por aquellas fechas se desconocía de algún algoritmo para poder entrenar este tipo de red. A continuación en la *figura 2.8* podemos apreciar que la red perceptrón monocapa consiste en una única neurona con una estructura del tipo"feedforward":

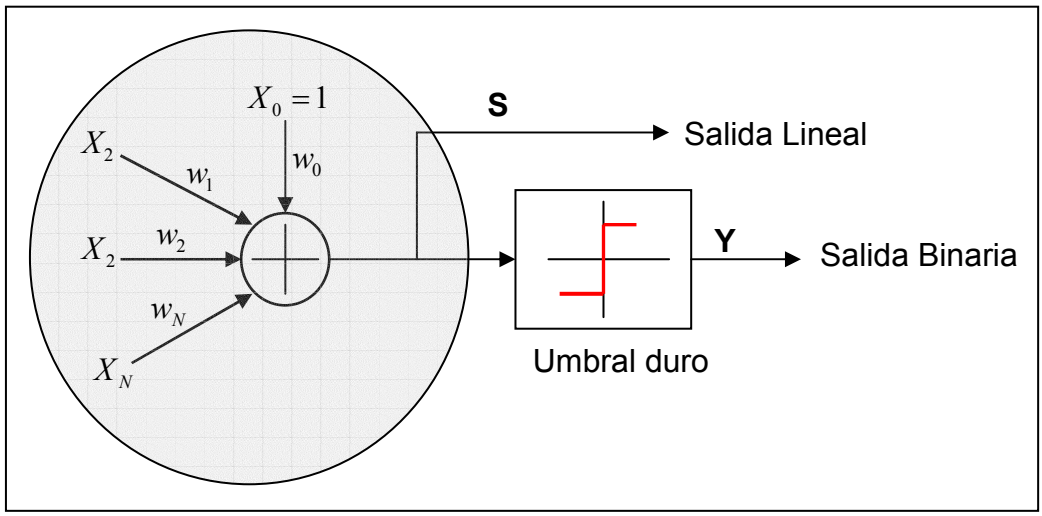


Figura 2.8. Perceptrón monocapa con umbral duro.

Donde:

$$S = Z = W^T \cdot X + W_0 \qquad (8)$$

Y tras pasar por el umbral duro:

$$Y = O = F_w = u(W^T \cdot X + W_0) \qquad (9)$$

Como vemos en la *figura 2.8* el perceptrón monocapa consiste en un buffer con los datos de entrada y en la capa de salida una neurona que lleva una función de activación umbral duro. Más adelante en las redes perceptrón multicapa la sustituiremos por una función derivable para poder resolver problemas de mayor dificultad.

El perceptrón monocapa es entrenado de forma supervisada. Durante la fase de entrenamiento se presenta al perceptrón una colección de pares de datos de entrada-salida. El objetivo de esta fase es la de aprender la relación no-lineal entre la entrada y la salida, de forma que el perceptrón represente un determinado mapeo entrada-salida. La habilidad del perceptrón para generalizar es chequeada durante la fase de test. En esta fase nuevos datos de entrada-salida, diferentes de los de entrenamiento, son presentados al perceptrón.

Una neurona solo puede hacer una separación lineal en el espacio de entradas. Cuando este espacio es de dos dimensiones esta separación lineal consiste en una recta que separa dos semiespacios, cuando es de tres dimensiones consistirá en un plano que separa dos semiespacios y en general cuando es de N dimensiones consistirá en un espacio de N-1 dimensiones que divide el espacio de N dimensiones en dos.

El perceptrón monocapa presenta una serie de limitaciones. Debido a que la función escalón es de tipo lineal **sólo le permite resolver problemas linealmente separables** como comentamos anteriormente. Además al tener una solo neurona **no podemos realizar varias separaciones lineales. Por estas dos razones** en 1960 apareció la *red perceptrón multicapa (MLP)* que subsana los problemas mencionados anteriormente. En la siguiente *Figura 2.9* podemos observar su estructura:

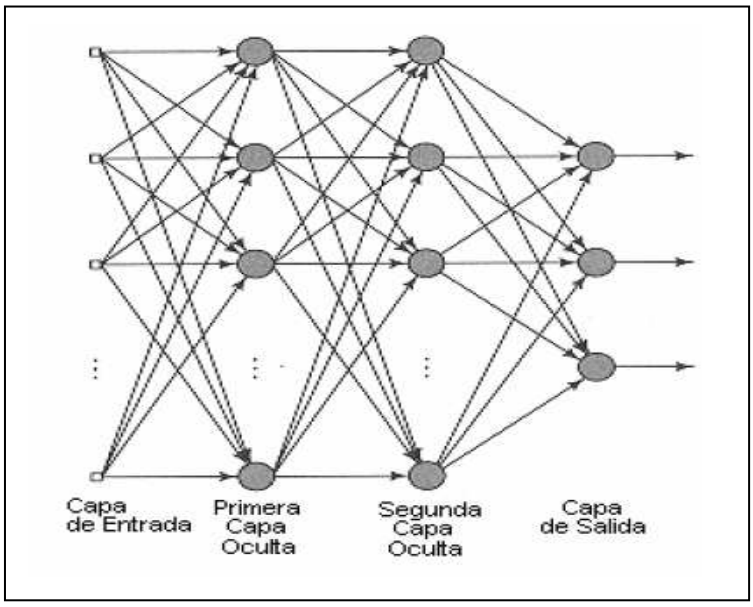


Figura 2.9. Red Perceptrón multicapa.

Donde cada neurona de la red es del tipo:

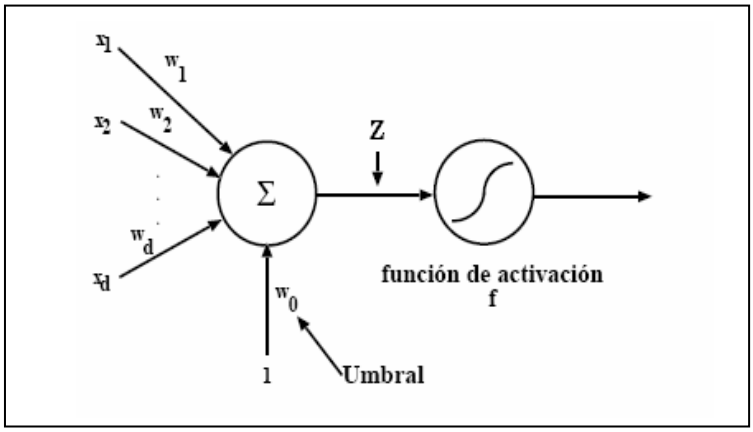


Figura 2.7. Interior de una neurona de la Red Perceptrón multicapa.

De esta forma, podremos resolver problemas más complicados de carácter no lineal por ser multicapa y por poseer una función de activación en cada neurona de tipo no lineal. La salida del perceptrón será derivable, por lo que podremos usar entrenamiento por gradiente.

Los únicos parámetros que se deben calcular durante la etapa de entrenamiento son los pesos de las interconexiones entre las neuronas de la red. Este tipo de red utiliza para el entrenamiento de sus pesos el algoritmo de retropropagación ("backpropagation" en inglés). Al igual que en el perceptrón monocapa, el entrenamiento del perceptrón multicapa es de tipo supervisado. En primer lugar se presenta un dato de entrada como estímulo para la primera capa de las neuronas de la red. Dicho estímulo se va propagando a través de las capas que conforman la red hasta generar una salida multidimensional. El resultado en las neuronas de salida se compara con la salida que se desea obtener y se calcula un valor de error para cada neurona de salida.

A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida hacia todas las neuronas de la capa inmediatamente anterior a la capa de salida que contribuyan directamente a la salida. 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. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona. El proceso descrito se repite para el resto de datos de entrada-salida hasta que el error de salida es menor que un determinado umbral inicial.

La importancia de la red backpropagation consiste en su capacidad de auto-adaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe ente un conjunto de patrones de entrada y sus salidas correspondientes.

2.7.2 *REDES AUTOORGANIZATIVAS*

Diversas áreas del cerebro, especialmente de la corteza cerebral, se hallan organizadas según diferentes modalidades sensoriales. Esta organización de la actividad cortical del cerebro puede describirse mediante mapas ordenados. Inspirado en el mapeo ordenado del cerebro, Kohonen introdujo en 1982 un algoritmo de autoorganización que produce mapas ordenados que simulan cortezas biológicas simplificadas con el objeto de resolver problemas prácticos de clasificación y reconocimiento de patrones [4]. Estos mapas fueron denominados mapas autoorganizativos. Los MAO presentan la propiedad de preservación de la vecindad, que los distingue de otros paradigmas de redes neuronales.

Los MAO son redes neuronales entrenadas de forma no supervisada mediante aprendizaje competitivo. En este tipo de aprendizaje, las neuronas de salida compiten entre ellas para ser activadas, dando como resultado la activación de una sola a la vez. Esta neurona es llamada "neurona ganadora". A diferencia de otras redes neuronales donde sólo se permite que aprenda la unidad ganadora, en **los MAO todas las unidades vecinas a la ganadora reciben una realimentación procedente de la misma, participando de esta manera en el proceso de aprendizaje**. Esta importante característica es también denominada realimentación lateral y puede ser excitatoria, inhibitoria o una combinación de ambas.

Es usual que haya una capa de neuronas de entrada y una capa de salida. Se usan tantas entradas como dimensiones tenga el espacio vectorial de los patrones de entrada (espacio real o binario), y tantas salidas como clases o categorías se quieren utilizar para clasificar los patrones de entrada, de manera que cada nodo de salida representa una categoría.

En la f*igura 2.10* se puede ver la configuración básica de un MAO. Se observan las neuronas de entrada e_i y una red bidimensional de neuronas de salida S_j. Un peso sináptico conecta a la neurona con la S_j. A cada neurona de entrada e_i se le presenta el i-ésimo elemento de cada patrón de entrada x(n). Siendo n la ocurrencia temporal de este patrón.

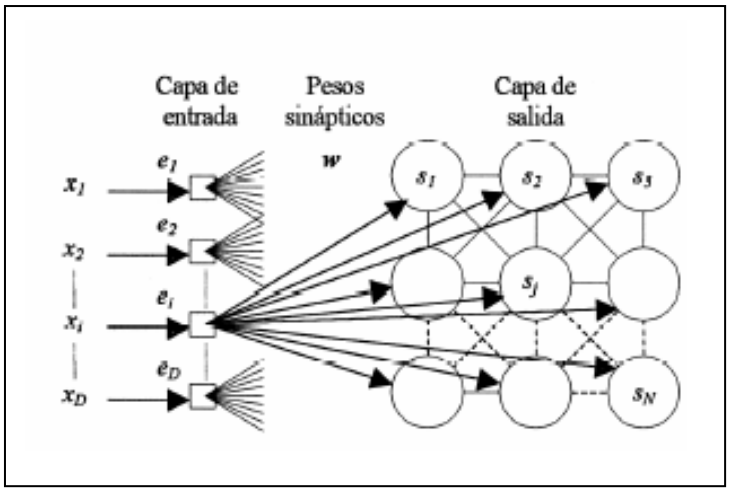


Figura 2.10. Estructura de una MAO.

Además de las conexiones hacia delante, con función excitatoria, se usa una red intracapa, inhibidora, simulando el fenómeno neurológico de la inhibición lateral, de ahí que se la denomina capa lateral. La red hacia delante implementa una regla de excitación de aprendizaje de Hebb. Esta regla refuerza las conexiones entre los pares de unidades entra-salida que se activan simultáneamente. La red lateral es intrínsecamente inhibidora, realiza la labor de seleccionar al ganador, normalmente mediante un método de aprendizaje competitivo, como el "winner-take-all" (el ganador lo toma todo, la unidad con mayor valor de activación toma el valor máximo (pe. 1) y el resto el mínimo (0)).

El arreglo bidimensional de neuronas de salida incluye conexiones entre las neuronas vecinas simulando la realimentación lateral. **Si G es una neurona ganadora durante el entrenamiento de un MAO, las neuronas vecinas que también serán actualizadas quedan en una región determinada por una función de vecindad LG(n). Esta región puede tener diferentes formas y es variable con el tiempo.**

El área cubierta comienza siendo máxima y se reduce a medida que avanza el entrenamiento hasta no incluir ninguna neurona vecina a la ganadora. En la *figura 2.11* se puede observar una región de vecindad cuadrada que disminuye su área en función del tiempo.

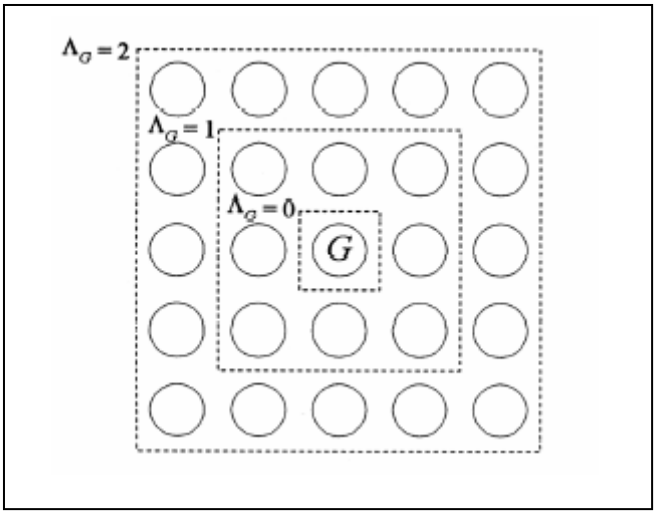


Figura 2.11. Región de vecindad de una MAO.

Durante el aprendizaje se consideran dos etapas**:** la etapa de ordenamiento y la de convergencia. Una vez que se ha entrenado un MAO, los vectores de pesos w_j, que van desde la salida S_j a todas las entradas, determinan los denominados centroides de cada clase.

La principal crítica a estos modelos es que no poseen una de las características generales de las redes neuronales: la información no se halla distribuida entre todas las conexiones, la destrucción de una sola unidad provocaría la pérdida de la información relativa a todo un grupo o categoría de patrones.

Como ejemplo de redes competitivas podemos citar las redes de Kohonen [4] y las arquitecturas ART [5].

Los MAO han sido utilizados con éxito en el reconocimiento de fonemas en discurso continuo, control de brazos robotizados, diseño de circuitos integrados.

2.7.3 *REDES CON FUNCIONES DE BASE RADIAL (RBF)*

En las redes RBF tenemos funciones radiales como función de activación. La estructura de una RBFN se muestra en la *figura 2.12*. Podemos observar que consta de tres capas. En primer lugar la capa de entrada que únicamente toma los datos de entrada y los presenta a la siguiente capa sin alteración. En segundo lugar una sola capa intermedia denominada capa oculta. Esta capa realiza una operación no lineal de forma que el espacio de entrada es transformado en un espacio de dimensión superior que se denominará espacio oculto. La transformación no lineal es llevada a cabo por las neuronas de la capa oculta. Todas las neuronas tienen asociada una misma función de base radial no lineal. La función de base radial posee un determinado número de parámetros variables de forma que cada neurona presentará un comportamiento diferente dependiendo del ajuste de los parámetros variables. Finalmente, tenemos una tercera capa llamada la capa de salida que transforma de manera lineal los datos del espacio oculto. Por lo que ahora tendremos es algo del tipo:

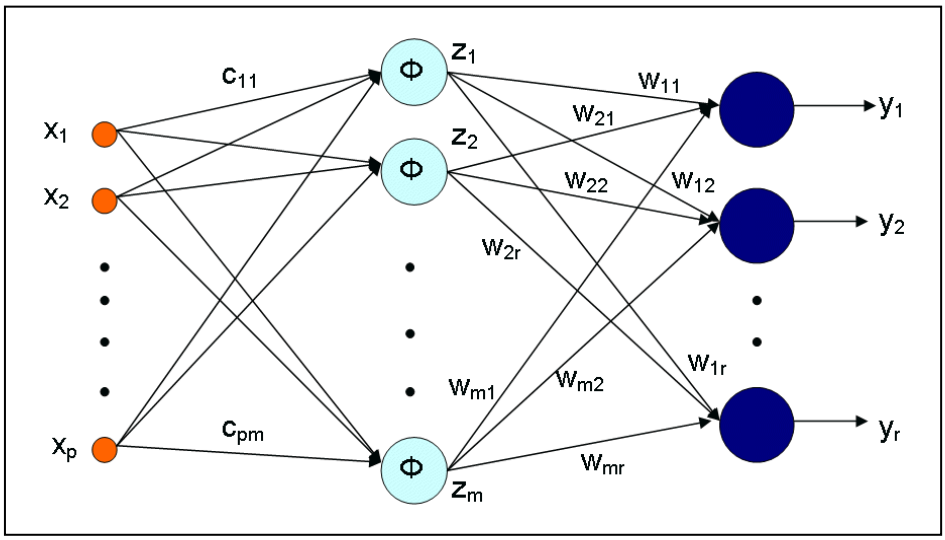


Figura 2.12. Estructura de una red RBF.

Existen diferentes tipos de funciones que pueden ser utilizadas como funciones de base radial. Además de la función gaussiana que es una de las más utilizadas y que hemos visto con anterioridad podíamos mencionar también la función multicuadrática:

$$\varphi(x) = \left(\|x - c\|^2 + d^2 \right)^{1/2}, d > 0 \qquad (10)$$

En [6], se demuestra que una RBFN que utilice una función de base radial de tipo gaussiano constituye un aproximador universal, es decir, la RBFN de este tipo puede aproximar arbitrariamente bien cualquier función multidimensional que sea continua.

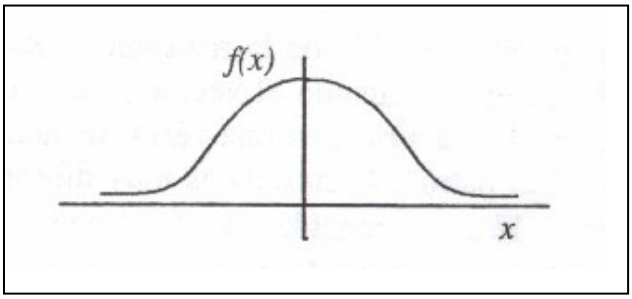


Figura 2.13. Función radial de la capa oculta

Otra diferencia entre las RBFN cuando se utiliza la función gaussiana y las MLP radica en la naturaleza de la función de actuación gaussiana. Como se aprecia en la *Figura 2.13* cuando *x* tiende a infinito la función gaussiana tiende a cero. Por tanto las neuronas realizan una aproximación local ya que proporciona valores significativos distintos de cero para los datos del espacio de entrada que se encuentran cerca (distancia euclídea pequeña) del centro de la gaussiana, ver *Figura 2.14*. Esto no se cumple para las

funciones de activación de las neuronas ocultas de las MLP. La RBF realiza una aproximación de carácter local como ya hemos comentado en cada uno de los centros seleccionados, mientras que la MLP lleva a cabo una aproximación de la función deseada a nivel global.

El valor de la varianza de una neurona para el caso de las RBF da una medida de cuando una muestra activa una neurona oculta para que dé una salida significativa. Geométricamente hablando podemos asimilar la varianza de una neurona a una "anchura" multidimensional.

Donde la salida total del sistema será del tipo:

$$Y = O = \sum_{i=1}^{N} W_i \cdot \exp\left(\frac{\left\| \overline{X} - \overline{C_i} \right\|^2}{2\sigma_i} \right) + W_0 \qquad (11)$$

Donde C_i es el centroide de las neuronas, σi es la varianza de la neurona y W_i son los pesos de la neurona. En el caso de que se use la función gaussiana se puede observar que toma como argumento la distancia euclídea entre el centroide y el vector de entrada dividido por la correspondiente varianza. Como vimos en el apartado 2.7.1. la función de activación de las neuronas en el MLP toma como argumento el producto interior entre el vector de entrada y los pesos de entrada a la neurona.

En el caso de una RBFN con funciones gaussianas el algoritmo de entrenamiento elegido tendrá que calcular los centroides (Ci) y varianzas (σi) de las neuronas, y los pesos de la capa de salida (Wi). El algoritmo de cálculo de los parámetros de las neuronas puede ser tanto supervisado como no-supervisado. Dependiendo de la estrategia de entrenamiento el orden en el que se calculan los parámetros de las neuronas puede ser diferente.

De esta manera se pueden fijar inicialmente los valores de varianzas para pasar a continuación a computar los centroides, [6]. Los centroides de las neuronas pueden ser elegidos en primer lugar mediante un algoritmo no supervisado para posteriormente ajustar las "anchuras" o varianzas de cada neurona dependiendo de la cantidad y lejanía de los centroides vecinos, [6]. El entrenamiento de los pesos de la capa de salida debe ser supervisado ya que se deben tener en cuenta los valores de salida para ajustar los pesos.

El espacio muestral al calcular los centros queda de la siguiente forma:

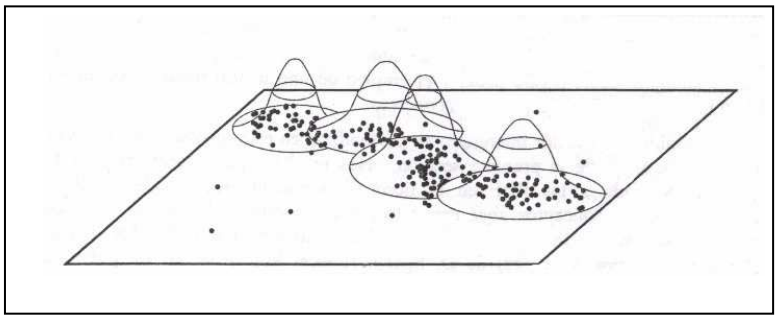


Figura 2.14. Centros en el espacio muestral.

La capacidad de aproximación tanto de las MLP's como de las RBFN's viene determinada por la llamada maldición de la dimensionalidad. Brevemente podemos decir que, para cualquier técnica de entrenamiento que se emplee, el número de parámetros necesarios para aproximar una función se incrementa exponencialmente con la dimensión del espacio de entrada, [6].

2.7.4 *RED PPL (PROJECTION PURSUIT LEARNING)*

Este es el tipo de red que utilizaremos para resolver nuestro problema de detección de direcciones de llegada. Al igual que la RBFN, está compuesta por tres capas cada una de las cuales realiza diferentes operaciones. Una de las diferencias con respecto a las redes anteriormente vistas es que el entrenamiento se realiza neurona a neurona y capa a capa de forma cíclica.

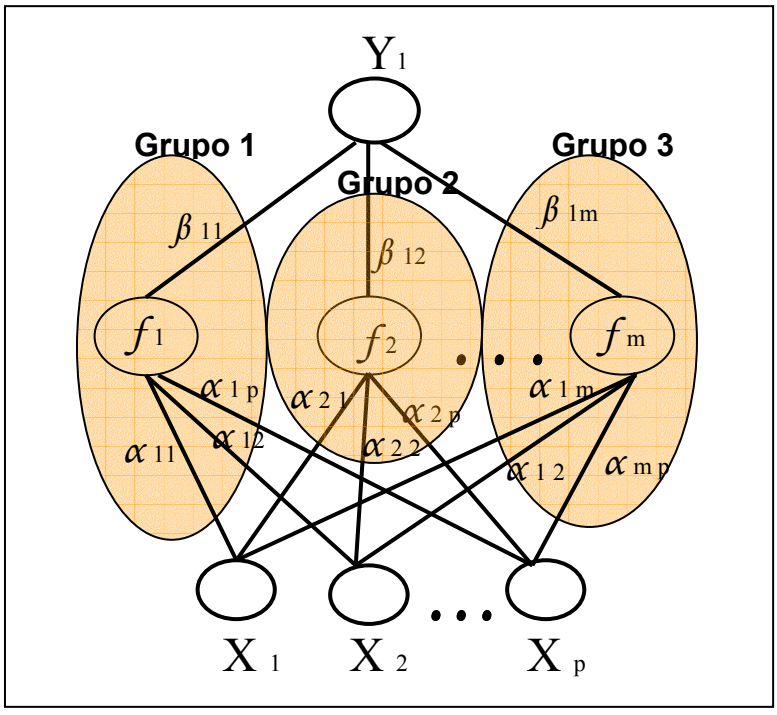


Figura 2.15. Estructura de la red PPL.

Todos los parámetros que tienen que ser estimados en este tipo de red son divididos en grupos asociados a una neurona oculta como muestra la *Figura 2.15*, cada uno de estos grupos es dividido en tres subgrupos:

- Los pesos de la capa de salida β_{ik}
- La función no paramétrica de suavizado f_k
- Los pesos de la capa de entrada α_{kj}

La red PPL empieza actualizando los parámetros del grupo 1 para minimizar la función de perdidas y seguidamente pasa al grupo 2 y así sucesivamente. Pero todo esto lo veremos a continuación en el capítulo 3 entrando en detalles debido a que va ser la red que vamos a utilizar para resolver nuestro problema de detección de direcciones de llegada.

CAPÍTULO 3
RED NEURONAL PROJECTION PURSUIT LEARNING (PPL)

3.1 INTRODUCCIÓN A UNA RED NEURAL PPL

El algoritmo de entrenamiento de una red neuronal PPL (Projection Pursuit Learning) se basa en un procedimiento estadístico propuesto para el análisis de información que contenga múltiples variables. Dicho algoritmo se puede implementar en una red neuronal de dos capas, como se puede observar en la *figura 3.1*. El nombre de este procedimiento se debe a que realiza una serie de proyecciones desde un espacio de datos de una determinada dimensión en un espacio de dimensión menor. Una red neuronal PPL estándar se muestra en la siguiente figura:

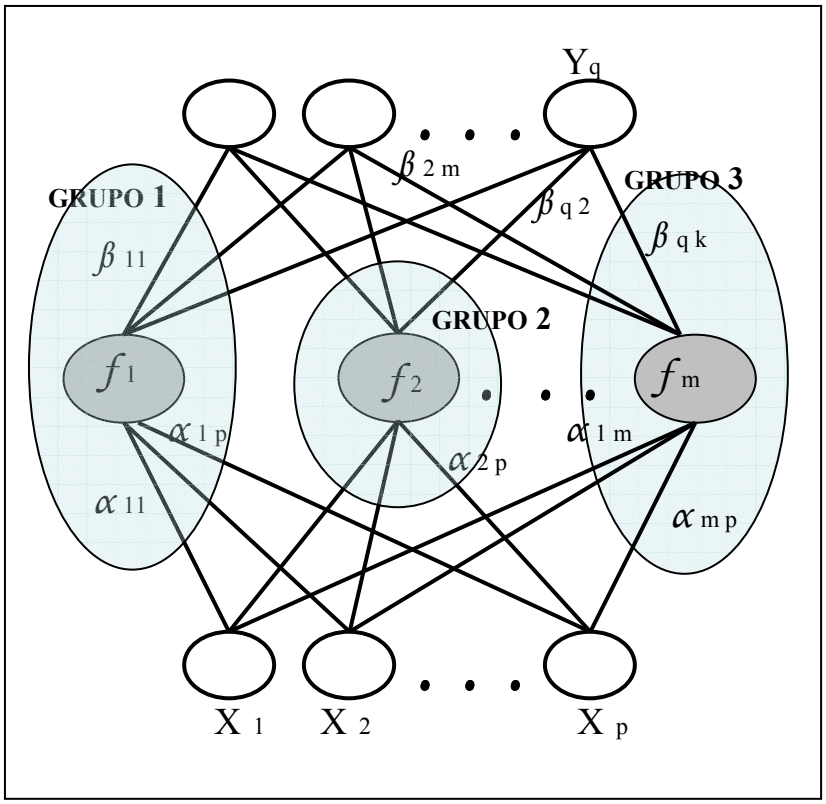


Figura 3.1. Red PPL

La red está compuesta por tres capas donde la primera toma los datos de entrada. Cada coordenada *j* del dato de entrada n-ésimo está conectada a la neurona k por una unión que tiene un peso α_{kj}. El mencionado peso multiplica el valor de la coordenada y denota la importancia de dicha coordenada.

La siguiente capa está situada en una posición intermedia y está formada por una serie de neuronas. La salida de cada neurona es debida a la función de activación $f\left(z = \alpha^T \mathbf{x}\right)$**.** Por último la capa de salida se encarga de calcular los valores de salida de la red PPL. La unión entre la coordenada *i* de salida y la salida de la neurona *k* tiene asignado un peso β_{ik}. Además como ya comentamos en el capítulo anterior la red esta dividida en grupos y estos a su vez en tres subgrupos que son los tres parámetros de la red que tenemos que actualizar en la fase de entrenamiento. Finalmente la salida de la coordenada *i* se forma mediante la suma de los productos de las salidas de las neuronas por sus correspondientes pesos. Por lo tanto la red queda modelada de la siguiente forma:

$$\hat{\mathrm{Y}}_\mathrm{i} = \mathrm{E}\left[\ y_i \mid x_1,\ x_2, x_3, ..., x_p\ \right] = \bar{y}_i + \sum_{k=1}^{m} \beta_{ik} f_k \left(\sum_{j=1}^{p} \alpha_{kj} x_j \right) \qquad (12)$$

Con $\bar{y}_i = \mathrm{E}\left[\ y_i\ \right]$, (media de la salida deseada *i*).

3.2 FUNCIONES DE HERMITE

Las $\{ f \}$ son las funciones de activación que tenemos en las neuronas de la capa intermedia la red neuronal PPL las cuales son paramétricas y se representan como una combinación lineal de funciones de Hermite de la forma:

$$f(z) = \sum_{r=1}^{R} c_r h_r(z) \qquad (13)$$

Donde $\mathbf{z} = \alpha^T x$ son los pesos multiplicados por los datos de entrada, **R** el orden de los polinomios, $\mathbf{c}_r$ son los coeficientes de las funciones de hermite y las $h_r(z)$ **son las funciones de Hermite** que tienen la propiedad de ser ortonormales y vienen definidas por la ecuación:

$$h_r(z) = (r!)^{-1/2} \pi^{1/4} 2^{-(r-1)/2} \mathbf{H}_r(z) \Phi(z) \qquad (14)$$

Donde $H_r(z)$ **son los polinomios de Hermite** construidos de forma recursiva de la siguiente manera:

$$H_0(z) = 1$$
$$H_1(z) = 2z$$
$$H_r(z) = 2\left[z\mathbf{H}_{r-1}(z) - (r-1)\mathbf{H}_{r-2}(z)\right]$$

Para $r = 2, 3, 4, \ldots$

Donde $\Phi(z)$ es la función gaussiana definida como: $\Phi(z) = \dfrac{1}{\sqrt{2\pi}}\, e^{-z^2/2}$ (15)

En la *figura 3.2* podemos observar las cuatro primeras funciones $h_r(z)$ de hermite:

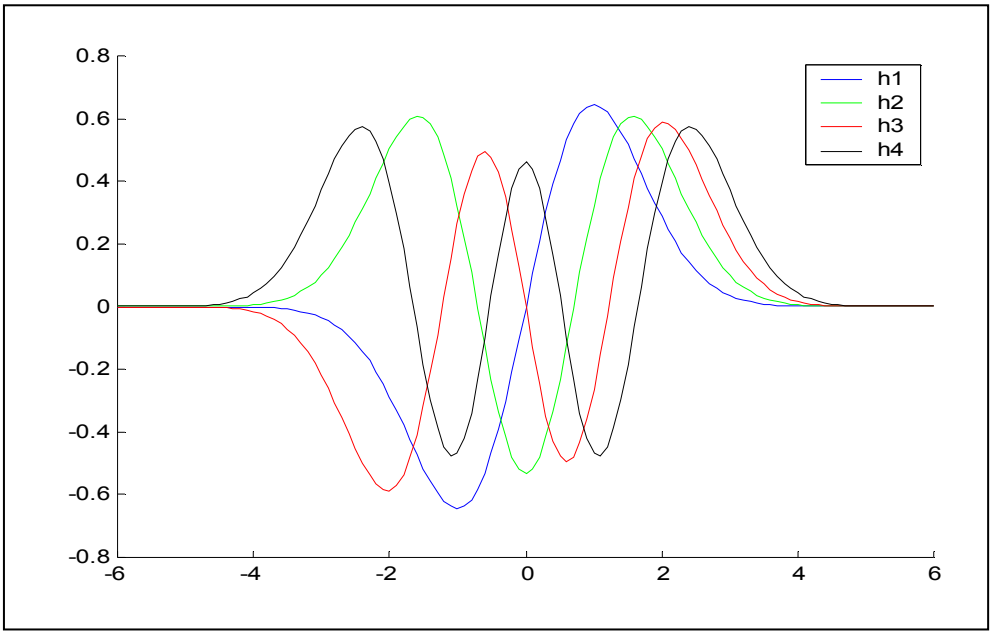


Figura 3.2. Representación de las cuatro primeras funciones de Hermite

Además las derivadas de las funciones de las neuronas intermedias tienen una forma simple:

$$f'(z) = \sum_{r=1}^{R} c_r \left[(2r)^{1/2} h_{r-1}(z) - z h_r(z) \right] \qquad (16)$$

Con esta expresión podremos calcular de una forma rápida y exacta las derivadas sin el uso de largas tablas de regresión. Además los coeficientes óptimos β_{ik} y c_j pueden ser calculados mediante álgebra lineal, por lo tanto la única parte que debemos realizar mediante optimización no lineal es el calculo de los pesos α_{kj} de las conexiones que se dirigen hacia la capa de neuronas intermedias, todo esto lo veremos en los apartados siguientes con más detalle.

3.3 MÉTODO DE APRENDIZAJE DE LA RED PPL

Los tres parámetros de la red, dirección de proyección $\alpha_k^T = (\alpha_{k1} \ldots \alpha_{kp})$, el peso de la capa de salida β_{ik} y las funciones de activación desconocidas f_k son estimadas minimizando el error cuadrático (LS) de la función de error (L):

$$L = \sum_{i=1}^{q} W_i E(\, y_i - \hat{y}_i \,)^2 = \sum_{i=1}^{q} W_i E\left[y_i - E[y_i] - \sum_{k=1}^{m} \beta_{ik} f_k\left(\alpha_k^T \mathbf{x}\right) \right] \qquad (17)$$

Donde $E[y_i]$ denota la media de los ***n*** datos de entrenamiento $(\mathbf{y}_l, \mathbf{x}_l)$. *L = 1,...,n*. Por ejemplo, $E[y_i] = \frac{1}{n}\sum_{l=1}^{n} y_{li} = \overline{y_i}$.

El valor W_i nos permite especificar la contribución de cada salida al error total, típicamente hemos elegido $W_i = 1/Var(y_i)$.

La red PPL aprende neurona - a - neurona y capa - a - capa cíclicamente después de que todos los datos de entrenamiento han sido presentados. En el entrenamiento se aplica el método conocido como "Least Squares**"** (LS) para estimar los pesos de la capa de salida, un método no-paramétrico para estimar la activación no lineal o de activación de cada neurona intermedia y para los pesos de entrada aplicaremos el método de Gauss-Newton [8].

Todos los parámetros a ser estimados son divididos jerárquicamente en *m* grupos (cada uno asociado a una neurona intermedia), y cada uno de estos, denominado por grupo *k*-ésimo, es adicionalmente dividido en tres subgrupos. Por una parte los pesos de la capa de salida $\{\beta_{ik}, \text{i} = 1,\ldots, \text{q}\}$ conectados a la *k*-ésima neurona intermedia. La función no paramétrica f_k de la *k*-ésima neurona intermedia. Finalmente, los pesos de entrada $\{\alpha_{kj}, \text{j} = 1,\ldots, \text{p}\}$, conectados también a la *k*-ésima neurona intermedia. Ver *figura 3.1.*

El algoritmo de entrenamiento de la red PPL empieza actualizando los parámetros asociados con la primera neurona intermedia (primer grupo) y para ello calculamos el nuevo valor de los tres subgrupos $\{\beta_{i1}\}, f_1, \{\alpha_{1j}\}$, consecutivamente (capa - a - capa) para minimizar la función de error (L). Después pasamos a actualizar los subgrupos de la segunda neurona intermedia $\{\beta_{i2}\}, f_2, \{\alpha_{2j}\}$. Una pasada completa termina con los nuevos valores calculados de la neurona intermedia *k*-ésima (la última), actualizando sus respectivos $\{\beta_{im}\}, f_m, \{\alpha_{mj}\}$, a la hora de el entrenamiento de la red realizaremos varias pasadas para optimizar el resultado.

3.4 ACTUALIZACIÓN DE LOS PARÁMETROS DE LA RED

En este apartado vamos a discutir cómo el grupo *k*-ésimo de parámetros $\{\beta_{ik}\}, f_k$ y $\{\alpha_{kj}\}$ es actualizado. Primero vamos a reescribir la función de error en términos de la función residual $R_{i(k)}$ para que nos sean más fáciles las operaciones:

$$R_{i(k)} = y_i - E[y_i] - \sum_{j \neq k} \beta_{ij} f_j\left(\alpha_j^T \mathbf{x}\right) \quad (18)$$

Luego la función de error queda:

$$L = \sum_{i=1}^{q} W_i E\left[R_{i(k)} - \beta_{ik} f_k\left(\alpha_k^T \mathbf{x}\right)\right]^2 = \sum_{i=1}^{q} W_i E[u_i]^2 \quad (19)$$

Expresando la media en forma de sumatorio tenemos:

$$L = \frac{1}{n} \cdot \sum_{l=1}^{n} \sum_{i=1}^{q} W_i \left[R_{li(k)} - \beta_{ik} f_k(\alpha_k^T \mathbf{x}_l) \right]^2 \qquad (20)$$

Donde ***n*** son los datos de entrenamiento y como ya vimos en el dibujo de la red ***q*** el número de salidas. De una forma más detallada:

$$u_i = R_{i(k)} - \beta_{ik} f_k\left(\alpha_k^T \mathbf{x}\right)$$

$$R_{li(k)} = y_{li} - E[y_i] - \sum\nolimits_{j \neq k} \beta_{ij} f_j\left(\alpha_j^T \mathbf{x}_l\right) \qquad (21)$$

Como podemos apreciar $R_{li(k)}$ no depende de k, es decir, de la neurona intermedia donde estamos actualizando los parámetros y esto nos va servir para simplificar las expresiones y tener claridad a la hora de hacer las derivadas requeridas en la actualización de los parámetros alfas, beta y de la función de activación.

3.4.1 *PARÁMETRO ALFA*

Una vez realizada esta simplificación de la función de error vamos a estudiar la actualización de los parámetros α_k para minimizar la función de error L. Para este propósito hemos utilizado el método de Gauss-Newton. Asumiendo que el nuevo vector óptimo de pesos se consigue sumando al peso anterior un incremento delta "Δ" quedando de la siguiente forma $\alpha(k+1) = \alpha(k) + \Delta$, la aproximación mediante la serie de Taylor quedaría:

$$L(\alpha_k + \Delta) \approx L(\alpha_k) + \left(\frac{\partial L(\alpha_k)}{\partial \alpha_k} \right)^T \Delta + \frac{1}{2} \Delta^T \left(\frac{\partial^2 L(\alpha_k)}{\partial \alpha_k^2} \right) \Delta \qquad (22)$$

En cada iteración elegimos un Δ que minimice $L(\alpha_k + \Delta)$ cuyo resultado viene de despejarlo de la siguiente ecuación lineal:

$$\sum_{i=1}^{q} W_i E\left[\left(\frac{\partial u_i}{\partial \alpha_k} \right) \left(\frac{\partial u_i}{\partial \alpha_k} \right)^T \right] \Delta = -\sum_{i=1}^{q} W_i E\left[\left(\frac{\partial u_i}{\partial \alpha_k} \right)^T u_i \right] \qquad (23)$$

Donde la derivada parcial de u_i que utilizamos en la ecuación (23) es:

$$\frac{\partial u_i}{\partial \alpha_k} = \frac{\partial \left(R_{li(k)} - \beta_{ik} \cdot f_k\left(\alpha_k^T \mathbf{x}_l\right) \right)}{\partial \alpha_k} = -\beta_{ik} f'\left(\alpha_k^T \mathbf{x}_l\right) \mathbf{x}_l \qquad (24)$$

Esta derivada es evaluada para cada nueva muestra con el valor de α_k actualizado en la muestra anterior.

Cuando la matriz de la parte izquierda de la ecuación (23) es definida no negativa entonces Δ es una dirección válida. Debido a que en ocasiones la matriz de la parte izquierda de la ecuación (23) puede ser definida negativa se deben introducir soluciones de la forma Δ / c_j, donde $c_j = 2^j$, j = 0, 1, 2,…. De esta forma se busca el c_j que minimice la función de error $L(\alpha_k + \Delta / c_j)$ [7].

3.4.2 *FUNCIÓN DE ACTIVACIÓN*

La actualización de las funciones de activación de las neuronas intermedias se obtiene de una manera sencilla minimizando la función de error *L* respecto f_k para cada valor de entrada a la función $z_{kl} = \alpha_k^T x_l$, donde *l* = 1,2,…, *n.* Por lo tanto vamos a minimizar la función perdidas (25):

$$L = \frac{1}{n} \cdot \sum_{l=1}^{n} \sum_{i=1}^{q} W_i \left[R_{li(k)} - \beta_{ik} \gamma_{kl} \right]^2 \qquad (25)$$

Con respecto $\gamma_{kl} = f_k\left(\alpha_k^T x_l\right)$, obteniendo la siguiente derivada:

$$\frac{\partial L}{\partial \gamma_{kl}} = \frac{W_i}{n} \sum_{l=1}^{n} \sum_{i=1}^{q} \left[2\beta_{ik}^2 \cdot f_k\left(\alpha_k^T \mathbf{x}_l\right) \cdot f_k'\left(\alpha_k^T \mathbf{x}_l\right) - 2R_{li(k)} \cdot \beta_{ik} \cdot f_k'\left(\alpha_k^T \mathbf{x}_l\right) \right] \qquad (26)$$

Igualando a cero y despejando f_k^*:

$$f_k^*\left(\alpha_k^T x_l\right) = \frac{\sum_{i=1}^{q} R_{li(k)} \beta_{ik}}{\sum_{i=1}^{q} \beta_{ik}^2} \qquad (27)$$

Ahora disponemos de un conjunto de datos $\left(z_{kl}, f_k^*\left(z_{kl}\right)\right)$ formado por *n* puntos y nuestro deseo es estimar una nueva función de activación $\hat{f}_k$ que los represente. **Para ello debemos calcular los nuevos valores de los coeficientes que multiplican a las funciones de hermite en la ecuación (13) y así cambiar la forma de la función de la neurona.**

Para este propósito utilizaremos un *suavizador paramétrico,* el suavizado de este conjunto de puntos lo haremos vía LS.

Podemos expresarlo en forma de sistema de ecuaciones:

$\mathbf{y}_k = \left(f_k^*(z_{k1}), f_k^*(z_{k2}), ..., f_k^*(z_{kn})\right)^T$, que son las salidas anteriormente calculadas.
$\mathbf{h}_{kl} = \left(h_1(z_{kl}), h_2(z_{kl}), ..., h_R(z_{kl})\right)^T$, funciones de hermite.

$$\mathbf{H}_k = \begin{pmatrix} \mathbf{h}_{k1}^T \\ \mathbf{h}_{k2}^T \\ \bullet \\ \bullet \\ \bullet \\ \mathbf{h}_{kn}^T \end{pmatrix}$$

$\mathbf{c}_k = (c_{k1}, c_{k2}, ..., c_{kR})^T$, los coeficientes de las funciones de Hermite.

Luego el nuevo valor $\hat{c}_k$ estimado se obtiene mediante LS resolviendo:

$$\min_{c_k} \left\| \mathbf{y}_k - \mathbf{H}_k \mathbf{c}_k \right\|^2 \qquad (29)$$

Y

$$\hat{\mathbf{c}} = \left(\mathbf{H}_k^T \mathbf{H}_k \right)^{-1} \mathbf{H}_k^T \mathbf{y}_k \quad (30)$$

El resultado es la nueva función de activación de las neuronas intermedias:

$$\hat{f}_k(z) = \sum_{r=1}^{R} \hat{c}_{kr} h_r(z) \quad (31)$$

3.4.3 *PARÁMETRO BETA*

Ahora pasemos a ver como se realiza la actualización de los pesos de la capa de salida β_{ik}. Utilizando LS los estimaremos calculando la derivada de L respecto β_{ik} e igualando a cero después para calcular el valor nuevo de los pesos de salida:

$$L = \frac{1}{n} \cdot \sum_{l=1}^{n} \sum_{i=1}^{q} W_i \left[R_{li(k)} - \beta_{ik} f_k(\alpha_k^T \mathbf{x}_l) \right]^2 \qquad (32)$$

$$L = \frac{W_i}{n} \sum_{l=1}^{n} \sum_{i=1}^{q} \left[{R_{li(k)}}^2 + {\beta_{ik}}^2 \cdot {f_k}^2 \left(\alpha_k^T \mathbf{x}_l \right) - 2 R_{li(k)} \beta_{ik} f_k \left(\alpha_k^T \mathbf{x}_l \right) \right] \qquad (33)$$

Haciendo las derivadas para un β_{ik} determinado y sabiendo que $R_{li(k)}$ no depende de k tenemos:

$$\frac{\partial L}{\partial \beta_{ik}} = \frac{W_i}{n} \sum_{l=1}^{n} \left[2\beta_{ik} \cdot f_k^2 \left(\alpha_k^T \mathbf{x}_l \right) - 2 R_{li(k)} \cdot f_k \left(\alpha_k^T \mathbf{x}_l \right) \right], \quad i = 1,2,..., q \qquad (34)$$

Igualando a cero nos queda:

$$\beta_{ik}\sum_{l=1}^{n} f_k^2\left(\alpha_k^T \mathbf{x}_l\right) = \sum_{l=1}^{n} R_{li(k)} f_k\left(\alpha_k^T \mathbf{x}_l\right) \qquad (35)$$

Y finalmente despejando obtenemos el nuevo valor estimado para los pesos de la capa de salida:

$$\beta_{ik} = \frac{E\left[R_{i(k)} \cdot f_k\left(\alpha_k^T \mathbf{x}\right)\right]}{E\left[f_k^2\left(\alpha_k^T \mathbf{x}\right)\right]},\ i = 1,2,\ldots,\ q \qquad (36)$$

Para poder obtener este nuevo valor de β_{ik} primero debemos haber estimado el nuevo valor de α_k , como ya hicimos anteriormente, y la nueva función de activación f_k.

3.5 ALGORITMO DE ENTRENAMIENTO

Los pasos a seguir para entrenar la red neurona PPL son:

1) A α_k, f_k y $\{\beta_{ik}\}$ se les asigna unos valores iniciales.
2) $\hat{\alpha}_k$ es estimado de forma iterativa por el método de Gauss-Newton.
3) Una vez que tenemos el nuevo $\hat{\alpha}_k$, estimaremos el nuevo valor $\hat{f}_k$ suavizando el conjunto de datos $\left(z_{kl}, f_k^*(z_{kl})\right)$.
4) Repetiremos los pasos 2)-3) varias veces.
5) Con los valores mas recientes de $\hat{f}_k$ y $\hat{\alpha}_k$ calcularemos los nuevos valores de los pesos de salida $\{\beta_{ik}\}$.
6) Repetiremos los pasos de 2) a 5) hasta que la función de error *L* es minimizada con respecto a los tres parámetros α_k, f_k y $\{\beta_{ik}\}$ asociados a la neurona intermedia *k*-ésima.

Luego, el procedimiento se repite para (k+1)-ésima neurona intermedia. Una vez el procedimiento anterior se ha repetido para todas las neuronas, se comienza de nuevo por la primera neurona. De este modo el algoritmo de actualización de los parámetros variables de la red opera sobre todas las neuronas de la capa intermedia hasta que el error L es minimizado.

3.6 ESTUDIO DE LA RED CON UN PARÁMETRO BIAS

El valor de R, grado de la función de hermite utilizada como función de activación, debe ser escogido al principio del entrenamiento y a veces resulta crítico para el éxito de la aproximación que deseamos realizar. Una mala selección de R podría llevarnos a malos resultados tanto en el entrenamiento como en el testeo de la red. Si usamos un valor alto suponemos que decaerá el error de aproximación pero a la vez tomamos el riesgo de incrementar el error de estimación, es decir la red es incapaz de generalizar adecuadamente. Dicha degradación de la capacidad de generalización se debe a que un

R alto implica un número elevado de coeficientes de hermite que calcular. Además con un R con un valor fijado de forma heurística no cumpliríamos la propiedad de aproximación universal, [8]**.**

Tenemos que R y el número de neuronas afectan al problema de generalización de nuestra red. Para encontrar la solución óptima es necesario realizar varias simulaciones combinando estos dos parámetros, esta tarea requiere un coste computacional y de tiempo muy alto. Para evitar la excesiva dependencia de la capacidad de la red PPL respecto del parámetro R introducimos un nodo bias o entrada de valor constante θ_j en la capa de entrada (37). La estructura de la red con el nuevo nodo bias quedaría:

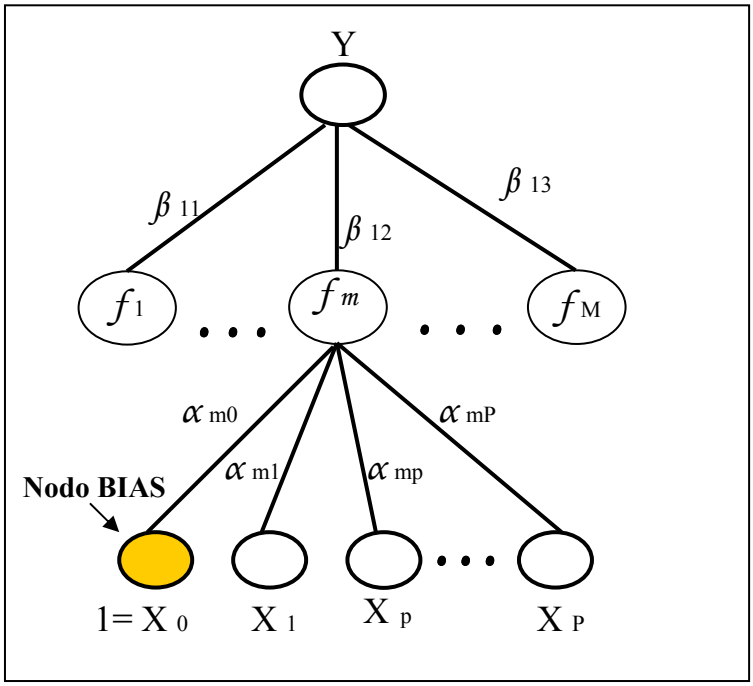


Figura 3.3. Red neuronal PPL con un parámetro bias.

Por lo tanto añadiendo el nodo de bias podemos fijar el parámetro **R, de modo que somos capaces de conseguir la aproximación universal sin la necesidad de realizar una serie de pruebas con diferentes valores de R y número de neuronas**. La introducción del nodo bias proporciona una mejora en el proceso de aproximación.

$$Y = \sum_{j=1}^{n} \beta_j f_j\left(\alpha_j^T x + \theta_j\right) \qquad (37)$$

3.6.1 *DEMOSTRACION EXPERIMENTAL SOBRE LA MEJORA DE LA RED CON UN PARAMETRO BIAS*

Para empezar vamos a ver una demostración simple aproximando la función $h_3(x)$ que es el polinomio de Hermite de grado tres dada por:

$$f(x) = h_3(x) = \frac{2x^3 - 3x}{\sqrt{3}\pi^{1/4}} \exp\left(\frac{-x^2}{2}\right) \qquad (38)$$

Los resultados que mostramos a continuación han sido calculados con una única iteración del algoritmo de entrenamiento de los parámetros sobre las neuronas. Una vez que el algoritmo de entrenamiento llega a la última neurona intermedia empieza otra vez desde la primera hasta la última y así sucesivamente las veces que sea necesario según la dificultad del problema. Para una mayor sencillez y claridad en nuestro primer ejemplo haremos una única iteración del algoritmo de entrenamiento sobre las neuronas de la red.

La configuración para la simulación queda de la siguiente forma:

- Conjunto de entrenamiento de 1000 puntos y un conjunto de test de 2000 puntos generados con una distribución uniforme.
- 9 neuronas intermedias en la red neuronal.
- Un polinomio de hermite de grado 2, es decir, R = 2.
- Una única pasada a las neuronas de la red en el algoritmo de entrenamiento.

Para este propósito veremos en cada simulación realizada dos gráficas:

- La evolución del error en la fase de entrenamiento.
- La salida de la red frente a la señal deseada que queremos aproximar que viene definida en (38).

Estas dos graficas las veremos en el caso de tener y no tener un parámetro bias en la red.

- **SIN NODO BIAS**

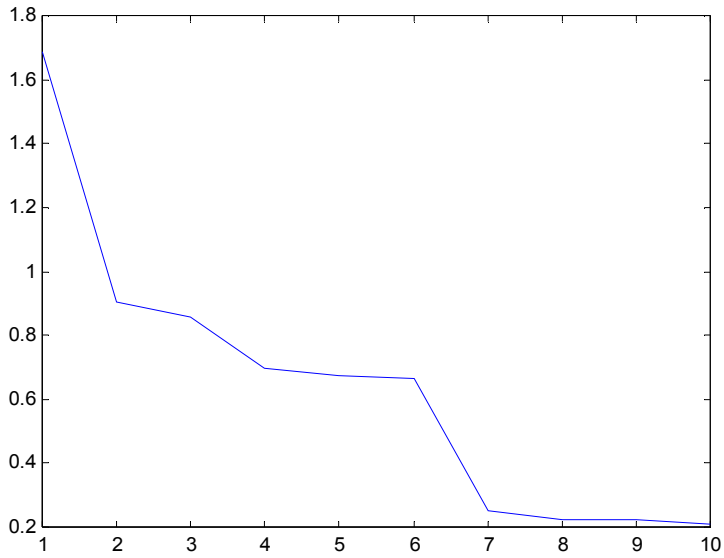


Figura 3.4. Evolución del error en función del número de neuronas entrenadas

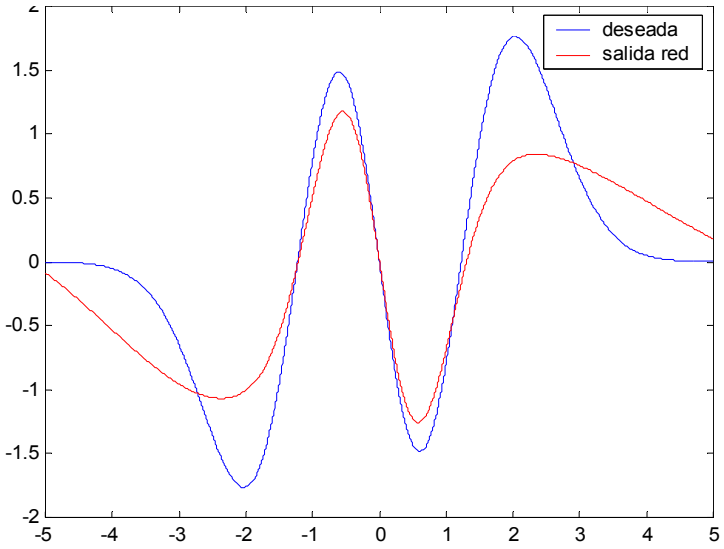


Figura 3.5. Salida de la red frente a la función $h_3(x)$
Con R = 2 y nueve neuronas intermedias

Hemos utilizado nueve neuronas intermedias pero en la *figura 3.4* llega hasta 10 esto es debido por que el paso 1 no es una neurona sino es la evaluación de la red para el valor de inicialización de los parámetros antes de empezar el entrenamiento.

Obteniendo:

MSE = 0.18913
FVU = 0.21025

Para dicha comparación entre la función original y la salida de la Red Neuronal utilizamos el error cuadrático medio (MSE) y la fracción no explicada de la varianza (FVU), dichas ecuaciones vienen definidas por:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left[f(x_i) - f_n(x_i)\right]^2 \qquad (39)$$

$$FVU = \frac{\sum_{i=1}^{N}\left[f(x_i) - f_n(x_i)\right]^2}{\sum_{i=1}^{N}\left[f(x_i) - \overline{f}\right]^2} \qquad (40)$$

- **CON NODO BIAS**

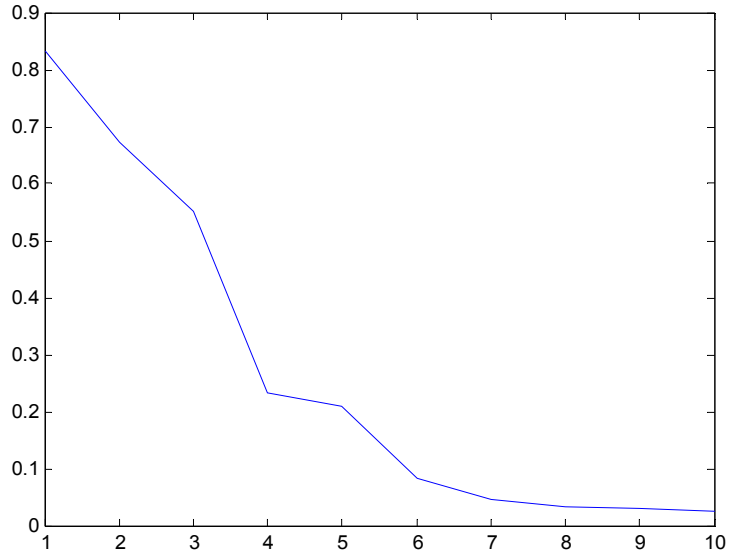


Figura 3.6. Evolución del error en función del número de neuronas

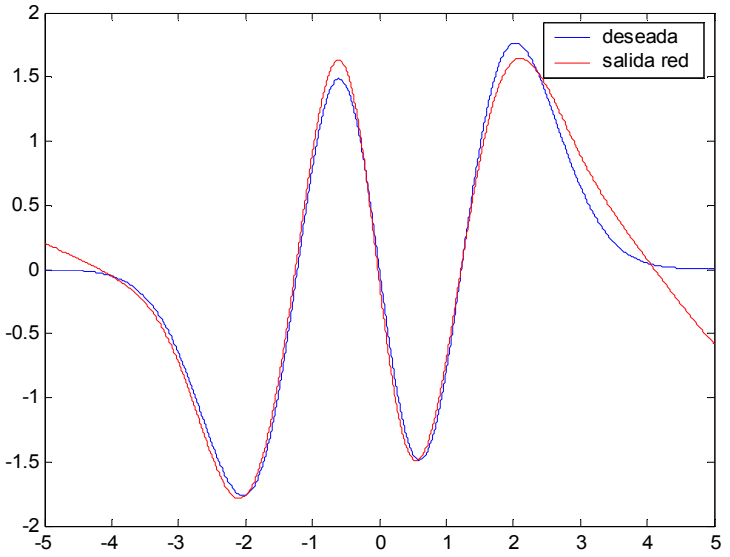


Figura 3.7 Salida de la red frente a la función $h_3(x)$
Con R = 2 y tres neuronas intermedias

Obteniendo:

MSE = 0.02348
FVU = 0.02610

Podemos observar como la aproximación de la red es mejor con el parámetro bias obteniendo un menor error.

Ahora mantenemos la configuración anterior pero reduciremos el número de neuronas para ver con una mayor claridad como influye el nodo bias en los resultados. Ahora disponemos de:

- 3 neuronas intermedias.
- R = 1.

  - **SIN NODO BIAS**

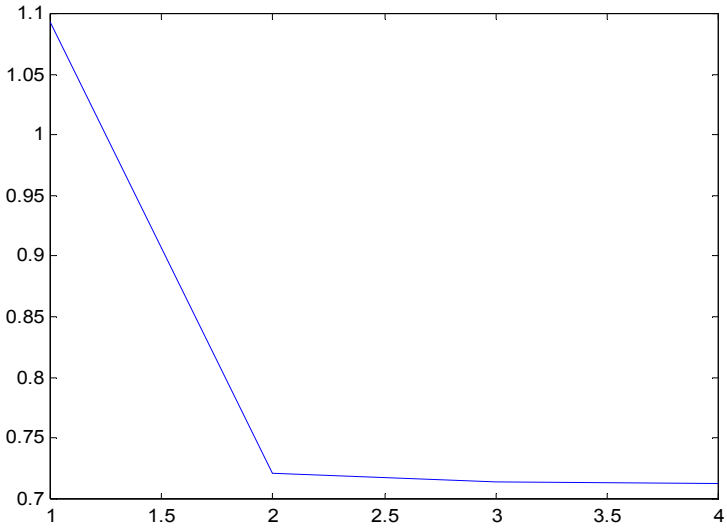


Figura 3.8. Evolución del error en función del número de neuronas

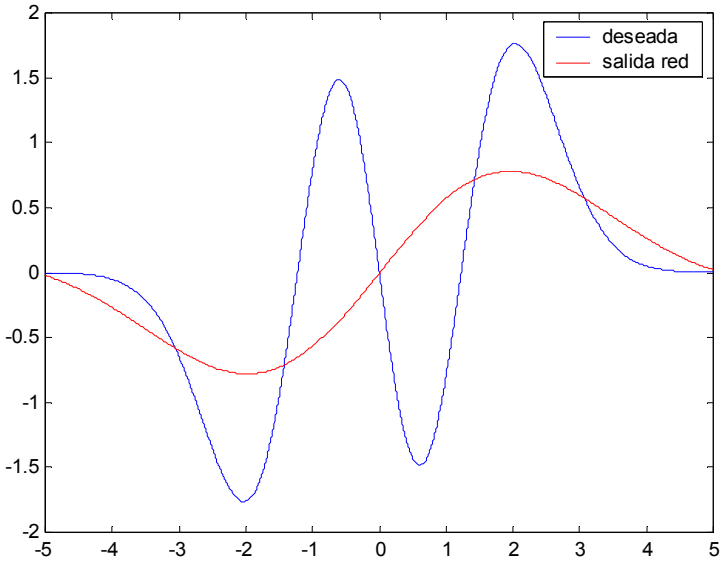


Figura 3.9. Salida de la red frente a la función $h_3(x)$
Con R = 1 y tres neuronas intermedias

Obteniendo:

MSE = 0.64174
FVU = 0.71346

Vemos en las *figuras 3.8 y 3.9* como el error no decae lo suficiente y como la aproximación de la red no es satisfactoria.

- **CON NODO BIAS**

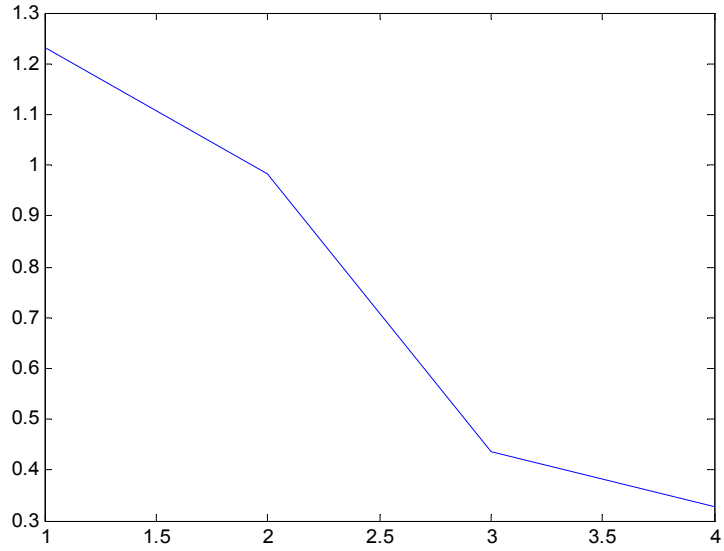


Figura 3.10. Evolución del error en función del número de neuronas

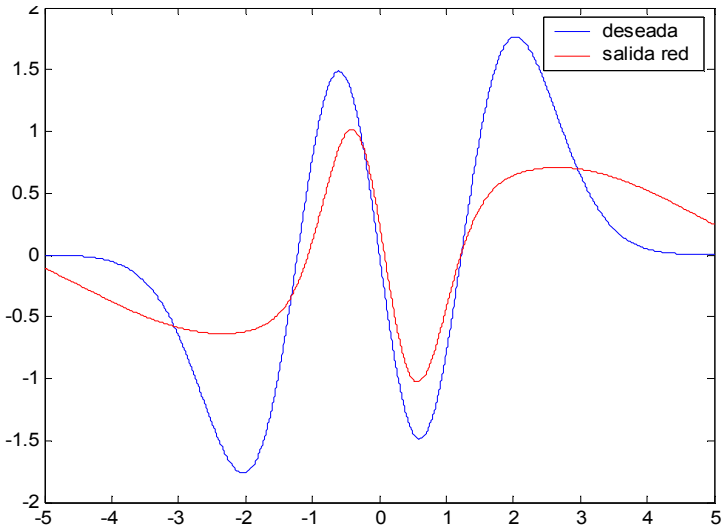


Figura 3.11. Salida de la red frente a la función $h_3(x)$
Con R = 1 y tres neuronas intermedias

Obteniendo:

MSE = 0.29639
FVU = 0.32949

Vemos en las *figuras 3.10 y 3.11* cómo mejora la salida de la red neuronal, y aunque todavía esta lejos de una buena aproximación esta sencilla simulación nos permite ver claramente como mejora la red con un nodo bias añadido a su estructura.

A continuación veremos los resultados aumentando el número de iteraciones, es decir, las veces en que el algoritmo de entrenamiento pasa por todas las neuronas. Luego la configuración de la simulación queda de la siguiente forma:

- 20 iteraciones.
- 9 neuronas intermedias.
- R = 2.

- **SIN NODO BIAS**

Tras varias simulaciones ya que la red depende de los valores iniciales que se calculan de forma aleatoria tenemos en media:

MSE = 0.0257
FVU = 0.0223

A continuación mostramos una de las simulaciones:

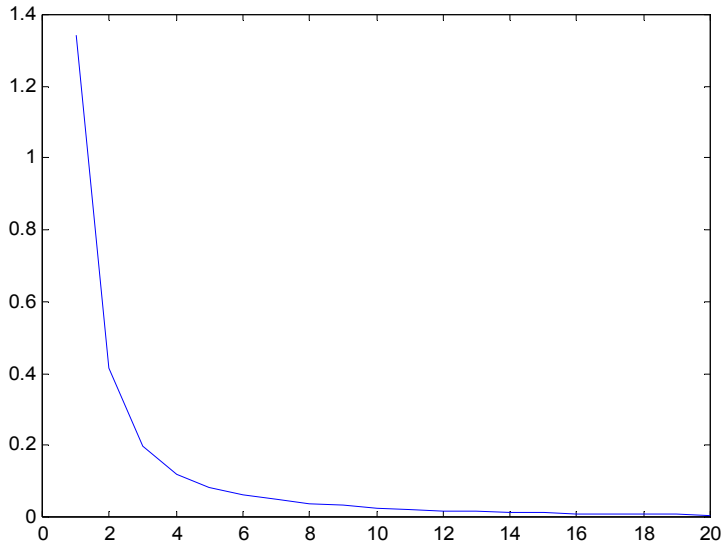


Figura 3.12. Evolución del error en función del número de iteraciones

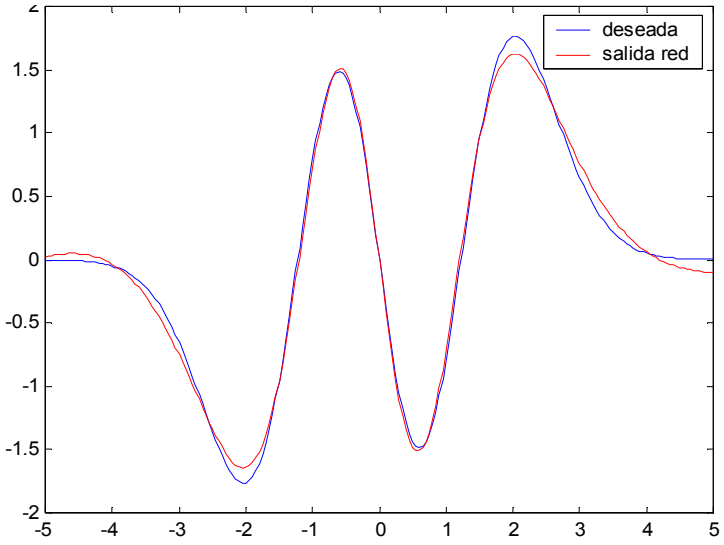


Figura 3.13. Salida de la red frente a la función $h_3(x)$
Con R = 2 y nueve neuronas intermedias

Obteniendo:

MSE = 5.1e-003
FVU = 5.7e-003

- **CON NODO BIAS**

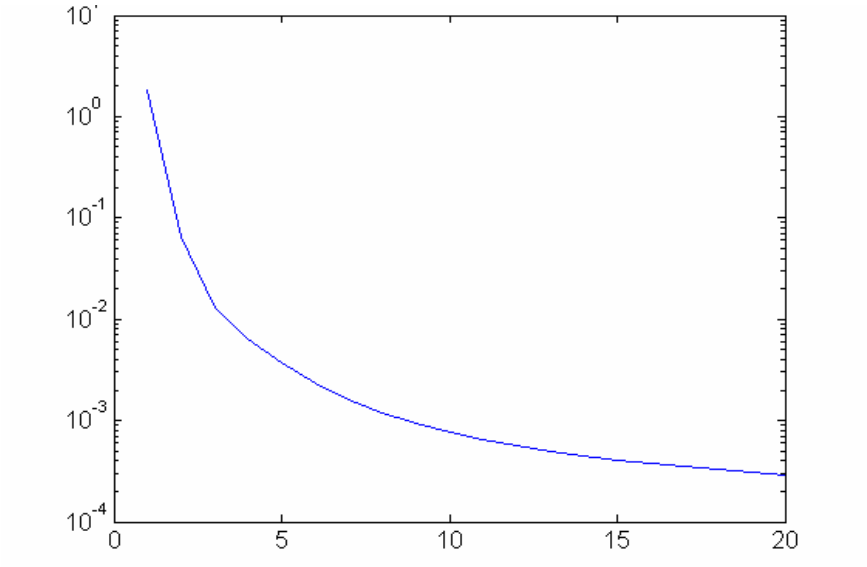


Figura 3.14. Evolución del error en función del número de iteraciones

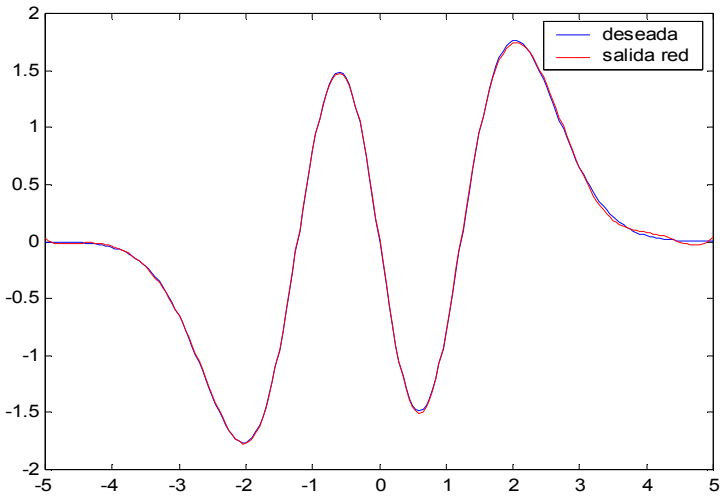


Figura 3.15. Salida de la red frente a la función $h_3(x)$
Con R = 2 y nueve neuronas intermedias

Obteniendo para una de las simulaciones utilizadas para el cálculo de la media:

MSE = 2.4191e-004
FVU = 2.7013e-004

Y para varias simulaciones tenemos en media:

MSE = 8.94e-004
FVU = 9.17e-004

Vemos en la *figura 3.15* que utilizando 20 iteraciones la salida de la red neuronal PPL con un nodo bias añadido a su estructura realiza una muy buena aproximación de la salida deseada.

3.6.2 *DEMOSTRACIÓN EXPERIMENTAL USANDO UNA FUNCIÓN RADIALDE DOS DIMENSIONES*

Ahora vamos a probar la importancia del nodo bias en nuestra red para problemas de regresión de dos dimensiones. Empezaremos con la función radial:

$$f^{(1)}(x_1, x_2) = 24.234\left[r^2\left(0.75 - r^2\right)\right]$$
$$r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$$

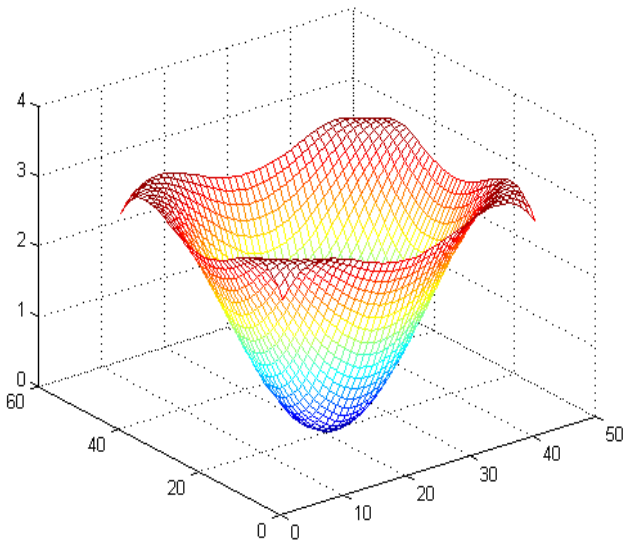


Figura 3.16. Función Radial

Aquí tenemos las tablas de resultados del **MSE** medio para varias simulaciones:

Para el caso de 3 neuronas intermedias

	R =1	**R = 3**	**R = 5**	**R = 7**	**R = 9**
SIN BIAS	0.5749	0.2644	0.2008	0.0797	0.0154
CON BIAS	0.3131	0.0321	0.0401	0.0351	0.0162

Tabla 3.1. MSE para 3 neuronas intermedias en la Red PPL.

Para el caso de 5 neuronas intermedias

	R =1	**R = 3**	**R = 5**	**R = 7**	**R = 9**
SIN BIAS	0.5002	0.0637	0.0072	0.0035	0.0184
CON BIAS	0.1036	0.0245	0.0063	0.0041	0.0031

Tabla 3.2. MSE para 5 neuronas intermedias en la Red PPL.

A continuación vamos a ver algunas de las simulaciones para:

- **3 neuronas intermedias**

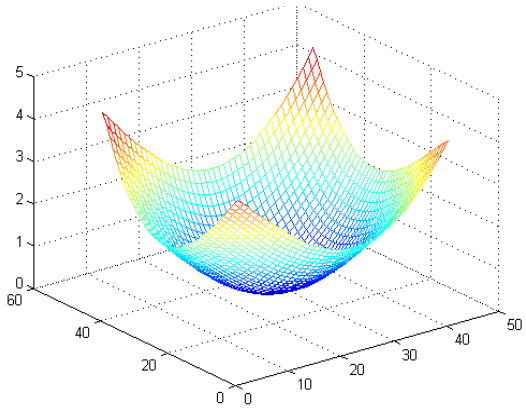


Figura 3.17. 3 neuronas y R = 3, sin BIAS.

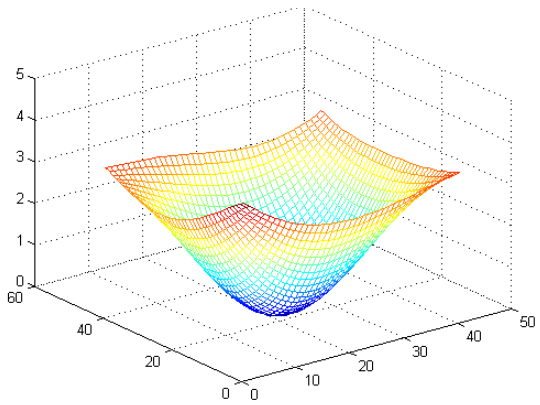


Figura 3.18. 3 neuronas y R = 3, con BIAS.

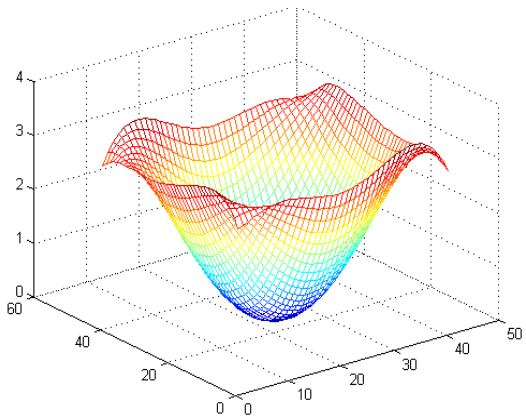


Figura 3.19. 3 neuronas y R = 9, sin BIAS.

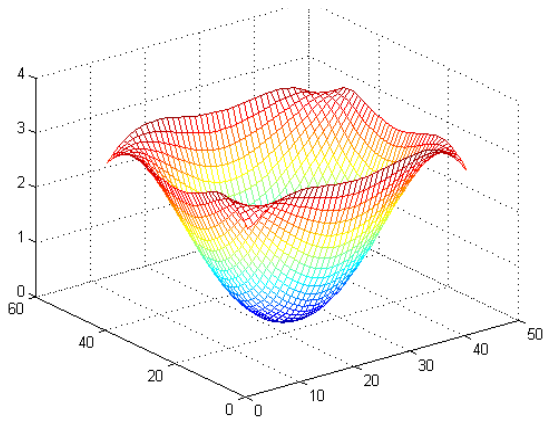


Figura 3.20. 3 neuronas y R = 9, con BIAS.

Se puede ver en la *tabla 3.1* que conforme aumenta el grado del parámetro R la red mejora en la aproximación de la función radial, además con el nodo bias el error que se consigue es menor excepto cuando R es igual a 9 que prácticamente los resultados son iguales. En las f*iguras 3.17 y 3.18* se ve que la aproximación que hace la red con un nodo bias para un R = 3 es mucho mejor.

- **5 neuronas intermedias**

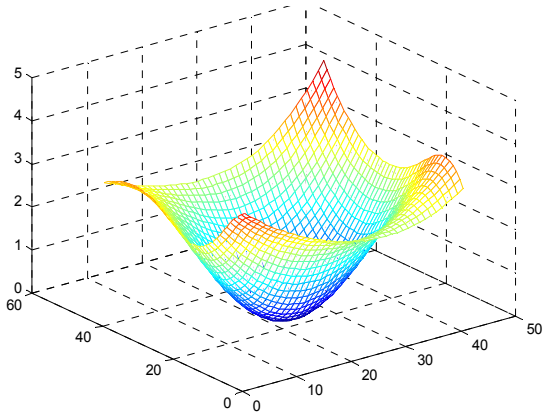


Figura 3.21. 5 neuronas y R = 3, sin BIAS

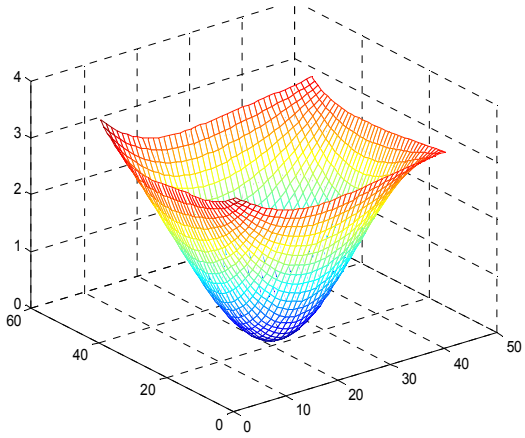


Figura 3.22.5 neuronas y R = 3, con BIAS

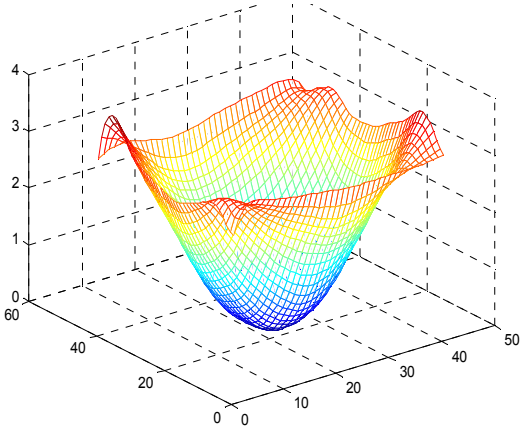


Figura 3.23.5 neuronas y R = 9, sin BIAS

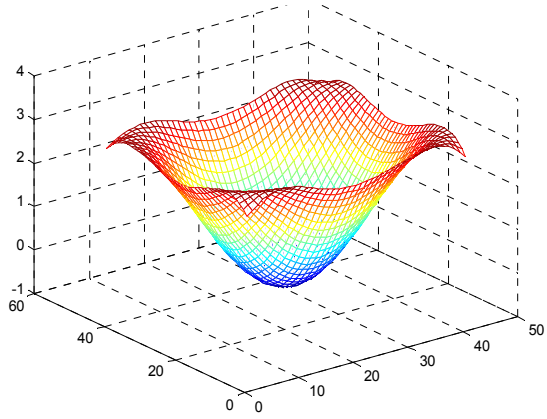


Figura 3.24.5 neuronas y R = 9, con BIAS

En la *tabla 3.2* apreciamos que cuando la red tiene un nodo bias, conforme aumenta el valor de R el error disminuye, sin embargo cuando no tenemos un nodo bias en el diseño de la red obtenemos una mejor aproximación con R igual a 7. Podemos concluir que para cualquier valor de R la red compuesta por un nodo bias consigue un menor error de aproximación, esto lo vemos de una forma más clara observando las *figuras 3.21, 3.22, 3.23 y 3.24.*

3.6.3 *DEMOSTRACIÓN EXPERIMENTAL USANDO UNA FUNCIÓN ADITIVA DE DOS DIMENSIONES*

$$f^{(2)}(x_1, x_2) = 1.3356\left[1.5(1-x_1) + e^{2x_1-1}\cdot\sin\left(3\pi(x_1-0.6)^2\right) + e^{3(x_2-0.5)}\cdot\sin\left(4\pi(x_2-0.9)^2\right)\right]$$

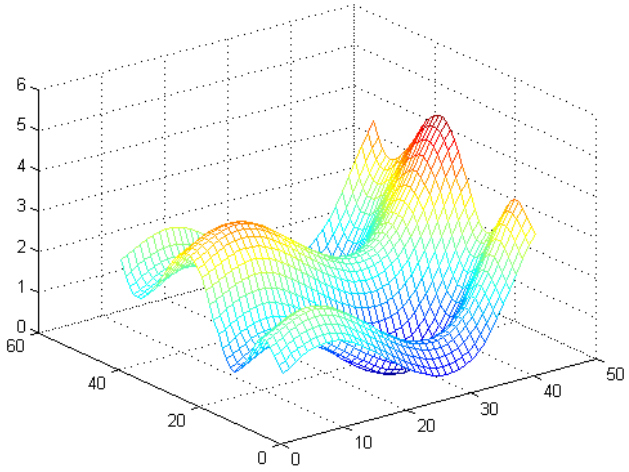


Figura 3.25. Función Aditiva

Aquí tenemos las tablas de resultados del **MSE** para varias simulaciones:

Para el caso de 3 neuronas intermedias

	R =1	**R = 3**	**R = 5**	**R = 7**	**R = 9**
SIN BIAS	0. 55528	0.47662	0.5143	0.2115	0.21412
CON BIAS	0.35409	0.13458	0.2435	0.3705	0.21553

Tabla 3.3. MSE para 3 neuronas intermedias en la Red PPL.

Para el caso de 5 neuronas intermedias

	R =1	**R = 3**	**R = 5**	**R = 7**	**R = 9**
SIN BIAS	0.5464	0.4314	0.25035	0.17584	0.02506
CON BIAS	0.2062	0.1861	0.18453	0.15813	0.00894

Tabla 3.4. MSE para 5 neuronas intermedias en la Red PPL.

A continuación vamos a ver algunas de las simulaciones:

- **3 neuronas intermedias**

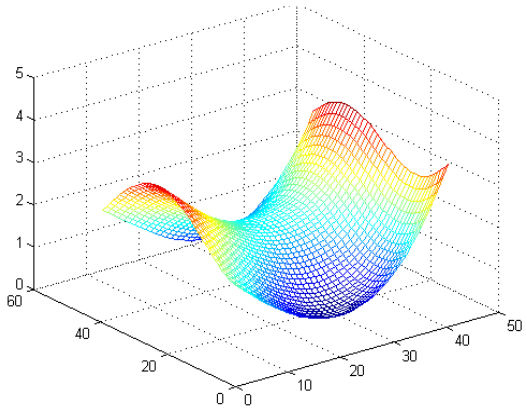


Figura 3.26. 3 neuronas y R = 3, sin BIAS.

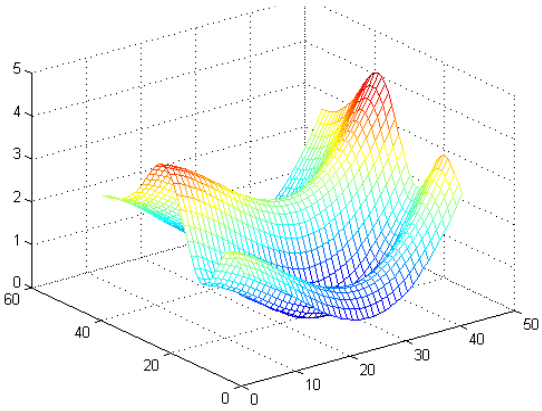


Figura 3.27. 3 neuronas y R = 9, con BIAS.

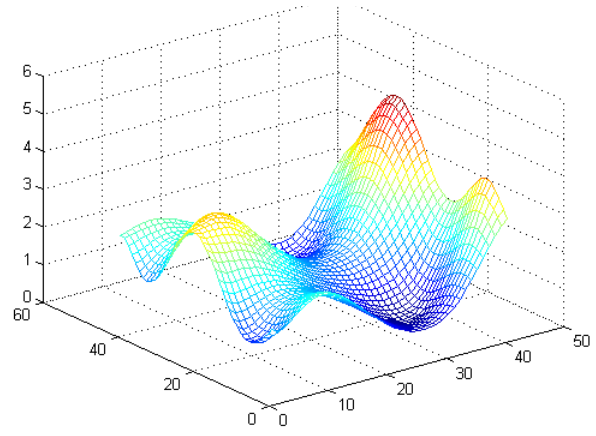


Figura 3.28. 5 neuronas y R = 3, sin BIAS.

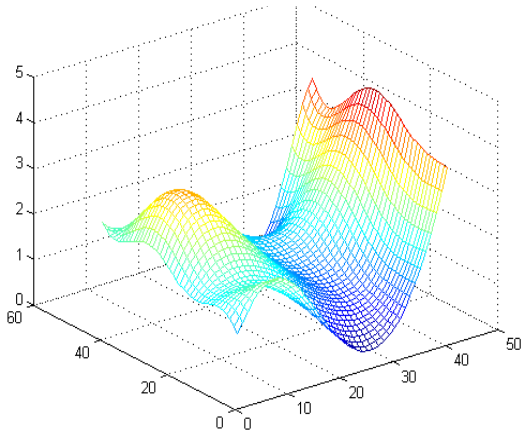


Figura 3.29. 5 neuronas y R = 9, con BIAS.

Esta función aditiva es más complicada de aproximar que la función radial. Para tres neuronas el mejor resultado lo obtenemos con un nodo bias añadido a la estructura de

la red y un valor de R igual 3, esto lo podemos ver en la *tabla 3.3* y de forma grafica en al *figura 3.26*.

- **5 neuronas intermedias**

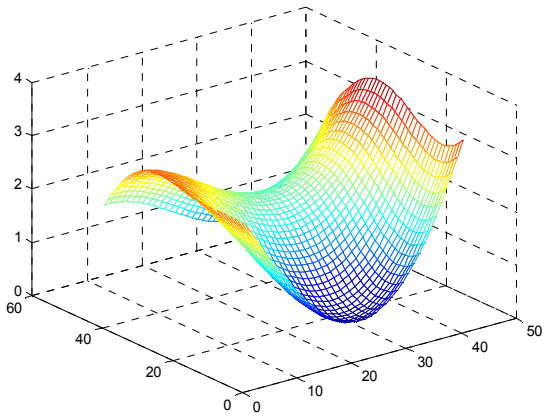


Figura 3.30. 3 neuronas y R = 3, sin BIAS.

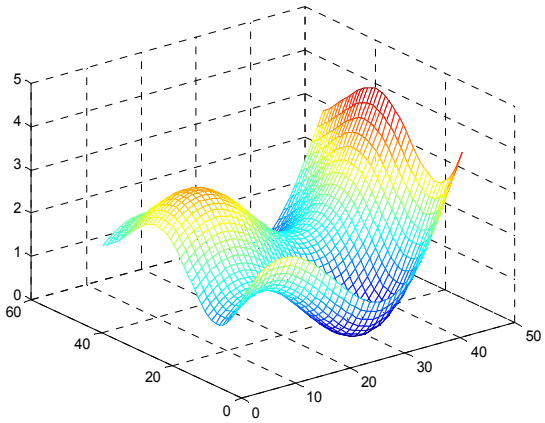


Figura 3.31. 3 neuronas y R = 3, con BIAS.

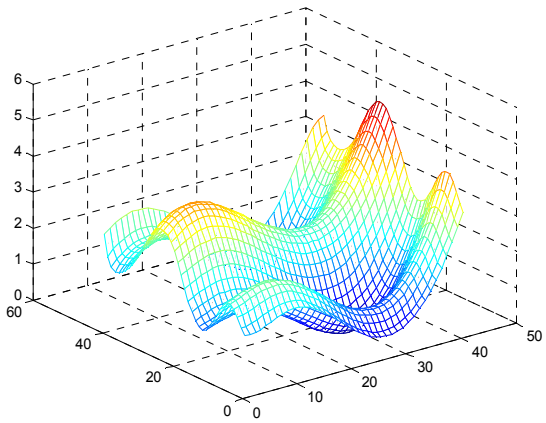


Figura 3.30. 5 neuronas y R = 9, sin BIAS.

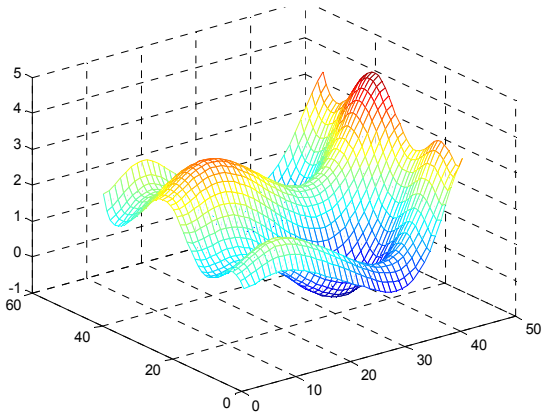


Figura 3.31. 5 neuronas y R = 9, con BIAS.

Al aumentar las neuronas a 5 podemos hacer una mejor aproximación y conforme aumentamos el valor de R obtenemos mejores resultados con un nodo bias añadido a la

estructura de la red neuronal, la mejora de la aproximación se ve de forma gráfica en la *figura 3.31.*

CAPITULO 4: RESULTADOS

4.1 PRE-PROCESADO DE DATOS

Para resolver nuestro problema en particular debemos hacer una trasformación de las señales de llegada para poder conseguir los datos que vamos a introducir a nuestra red neuronal.

Generalmente los algoritmos que trabajan con agrupaciones de antenas utilizan la matriz de correlación para la estimación de dirección de llegada en lugar de la señal **X**(t) ya que la correlación nos aporta mucha mas información sobre las señales recibidas. El vector de entrada a la red de la *figura 1.2* son los elementos de la matriz de correlación ***R*** que se organizan como un solo vector ***b*** de dimensión $2M^2$ ya que la red no trabaja directamente con números complejos y debemos separar las partes reales e imaginarias.

El vector de entrada ***b*** es dividido por su norma. Finalmente, el los valores del vector **z** constituyen la entrada a la red neuronal.

$$\mathbf{z} = \frac{\mathbf{b}}{\|\mathbf{b}\|} \qquad (41)$$

Para explotar la simetría en la matriz de correlación ***R*** solo necesitamos **considerar la parte triangular superior o inferior de la matriz**, en nuestro caso hemos utilizado la parte superior y así queda organizado en un vector de dimensión M(M+1) de partes reales e imaginarias ***b***. Este proceso queda ilustrado de la siguiente forma:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad (42)$$

$$\mathbf{b} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{22} & r_{23} & r_{33} \end{bmatrix} \qquad (43)$$

Ahora separamos las partes reales e imaginarias de este vector quedando ***b*** de dimensión final 2M(M+1), además **también se eliminarán aquellos términos repetidos** debido a que producen una columna linealmente dependiente en la matriz inversa que despejamos para calcular delta, ver ecuación (23), y esto produce errores en su calculo.

Hay que mencionar que el entrenamiento de una red neuronal para detectar el ángulo de llegada de múltiples fuentes no es una tarea fácil. Para tener una idea de cuanto entrenamiento es requerido consideremos el problema de seguir a dos fuentes solamente. Primero consideramos el espacio en la agrupación desde -90 a 90 y empezamos el entrenamiento con una separación angular de 2 º. Esto quiere decir que la primera fuente esta en θ = -90 º y la segunda fuente en θ = -88º, después fijamos la fuente en -89 º y la segunda en -87 º y así sucesivamente hasta cubrir la región de interés (-90 º a 90 º). Si tuviera 3 º de separación entre las dos señales (-90 º y -87 º, -89

º y -86 º, - 88 º y -85º …87 y 90). Luego para la fase de testeo tendríamos señales con un ángulos de llegada diferentes a los de entrenamiento y así nuestra red interpolaría.

A continuación mostramos un resumen de los pasos a seguir para la construcción de los datos de entrada a nuestra red:

1) Construcción de la señal $\mathbf{X}(t) = \mathbf{AS}(t) + \mathbf{N}(t)$.
2) Calculamos la matriz de correlación ***R.***
3) Formamos el vector ***b***.
4) Obtenemos el vector ***z*** normalizando ***b***.
5) Generamos finalmente los pares de entrada y salida $\{ \mathbf{z}^n, \boldsymbol{\Theta}^n, n = 1,2,..., N \}$
6) Los introducimos a la red neuronal PPL para su entrenamiento.

4.2 DESCRIPCIÓN DE LAS FUNCIONES PROGRAMADAS

Ahora vamos a describir brevemente cada una de las funciones programadas:

- **Ángulos**, es la función con la que empieza la simulación de la red, sus argumentos de entrada serán: La resolución entre las señales en la fase de entrenamiento y de test, el paso de los ángulos para la fase de entrenamiento y de test, el número de señales de llegada y el número de elementos de la agrupación. Por ejemplo: ángulos *(10,1,10,0.5,1,2)*, según los dos últimos argumentos tendríamos 1 señal de llegada y 2 elementos en la agrupación. La separación espacial entre las dos señales de llegada será de 10 º tanto en la fase de entrenamiento y de test. La fase de entrenamiento se realizará para un intervalo de 1º entre el espacio de -90º a 90 º y en la fase de test será de 0.5º. Esta función nos llevara en un principio a la fase de entrenamiento de la red y después a la fase de test.

- **Entrena,** en esta función formamos la matriz con los distintos ángulos de entrenamiento que pasaremos como argumento a la *función start* para que nos proporcione los datos de entrada finales que introduciremos a la red y seguidamente llamaremos a la *función parámetros*.

- **Start,** esta compuesta por tres sub-funciones:

    - **genera_x,** crea las señales de llegada a la agrupación.

    - **correlación,** empieza la transformación de las señales como hemos visto en el apartado 4.1.

    - **normalización,** procede a normalizar los datos de entrada a la red.

- **Parámetros,** en esta función queda definido el número de neuronas intermedias, la dimensión de entrada y la dimensión de salida. También se inicializan los pesos de la capa de entrada, los coeficientes de las funciones de activación y los

pesos de la capa de salida. Una vez inicializados los parámetros de la red pasamos a su entrenamiento llamando a la *función de inicialización*.

- **Inicialización,** en esta función se procede a la actualización de los parámetros alfa, beta y de los coeficientes de la función de activación. Por medio de las llamadas a los sub-funciones:

  - **Calcula salida,** que nos proporciona $\frac{\partial u_i}{\partial \alpha_k}$ y u_i para poder despejar delta de la ecuación (23).

  - **Compara delta,** nos asegura que la matriz de la parte izquierda del ecuación (23) quede definida no negativa

  - **Estima_f,** nos calcula los nuevos puntos por los que debe pasar nuestra función de activación, ver (27).

  - **Suaviza_f,** calcula los nuevos coeficientes de las funciones de hermite.

  - **Estima beta,** actualiza los pesos de salida de la red según la ecuación (36).

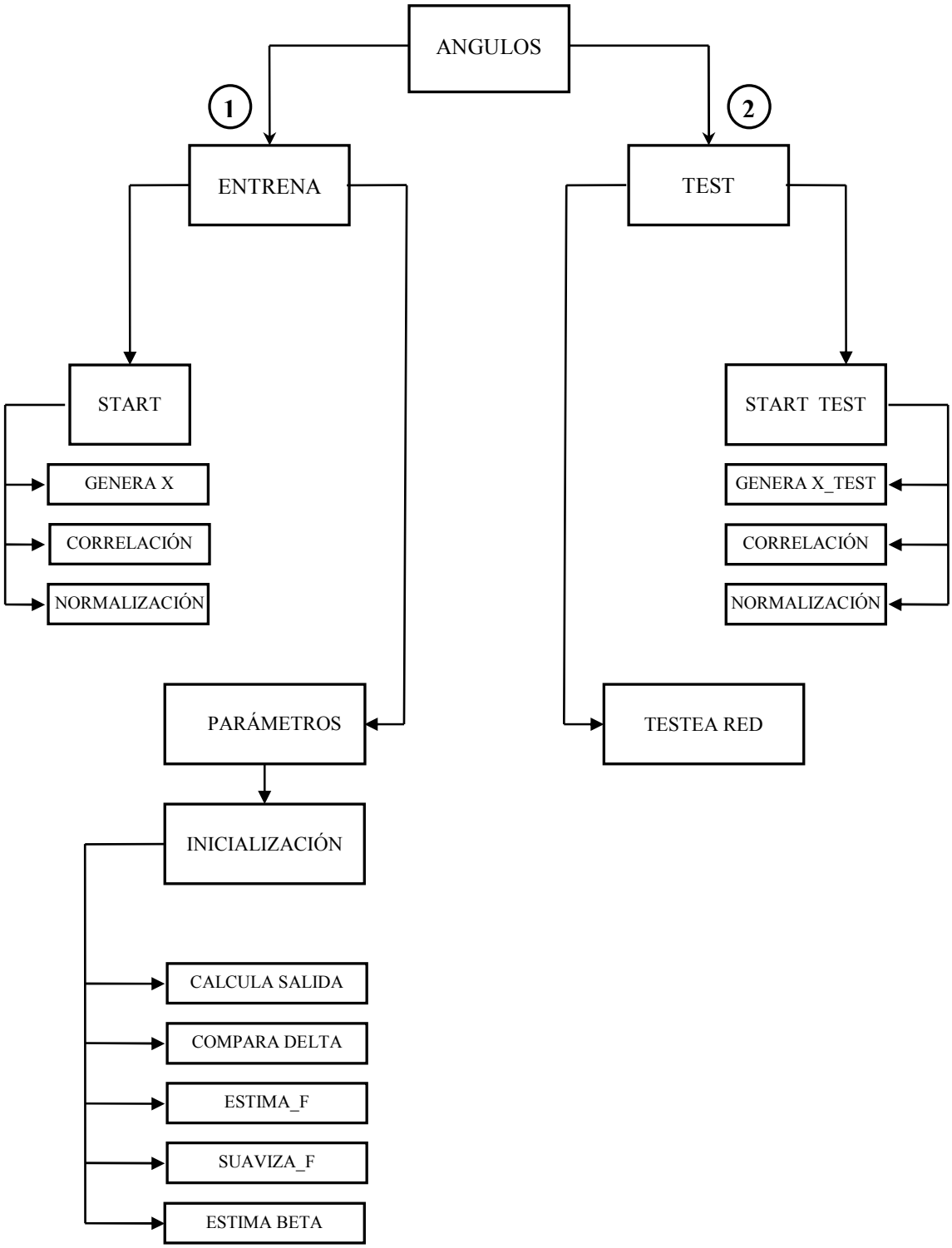
- **Test,** en esta función creamos los datos de entrada para comprobar el funcionamiento de la red con datos desconocidos.

- **Testea red,** una vez calculados en la fase de entrenamiento los parámetros que definen el comportamiento de la red introducimos los nuevos datos de entrada obteniendo finalmente las graficas y medidas de error necesarias para comprobar el comportamiento de la red programada.

En el siguiente punto veremos un esquema de las funciones programadas para que sirva de ayuda para futuras ampliaciones del proyecto. El programa empieza en 1 donde se desarrolla la fase de entrenamiento y después pasa al punto 2 para la fase de testeo.

4.2.1 *ESQUEMA DE LA FUNCIONES*

Vamos a ver un esquema de las funciones utilizadas para simular la red PPL.



4.3 UNA SEÑAL DE LLEGADA

A continuación vamos a proceder a la detección de un distinto número de señales de llegada para distintos elementos de la agrupación de antenas. En las graficas veremos el espacio de llegada de las señales esta acotado entre -90º y 90º.Además cuando tengamos más de una señal de llegada habrá una resolución espacial entre ambas, es decir, las señales de llegada estarán separadas entre si una serie de grados determinados. Procederemos a comparar **nuestra Red Neuronal PPL programada bajo Matlab 6.5 con la red neuronal RBF que viene programada en la toolbox de Matlab 6.5 [9] cuya descripción es la siguiente:**

- Utiliza funciones gaussianas como base radial.
- Para el cálculo de los centros se utiliza el algoritmo OLS ("Orthogonal least square").
- Par el cálculo de los pesos el método de los mínimos cuadrados.
- La varianza queda fijada al principio del algoritmo.

Los distintos parámetros de la red PPL que iremos modificando para el cálculo de resultados son:

- Número de señales de llegada a la agrupación (ó array).
- Número de elementos de la agrupación de antenas.
- La distancia entre los elementos de la agrupación.
- El grado de polinomio de Hermite: variando R.

En las simulaciones las señales de llegada las situaremos cada 1º ó cada 2º en la fase de entrenamiento, barriendo así el rango completo de -90º a 90º. Luego en la fase de test utilizaremos otros ángulos diferentes para comprobar la capacidad de aproximación de la red. Esto nos proporcionará 171 datos en la fase de entrenamiento y 342 datos en la fase de test, la frecuencia de la señal de llegada es de 2 GHz, estos valores son fijos e iremos variando el resto para distintas simulaciones.

- **Red RBF**

  Para una señal de llegada tendremos los siguientes parámetros:

  - **2 elementos** en la agrupación de antenas.
  - Una **dimensión de entrada a la red neuronal igual a 3**.
  - **3 neuronas intermedias**.

Veremos en las gráficas que muestran la salida de la red neuronal RBF cómo 3 neuronas no son suficientes para hacer una buena aproximación de la salida que deseamos , ya que el espacio oculto, es decir , el número de neuronas intermedias debe ser mayor que la dimensión del espacio de entrada.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

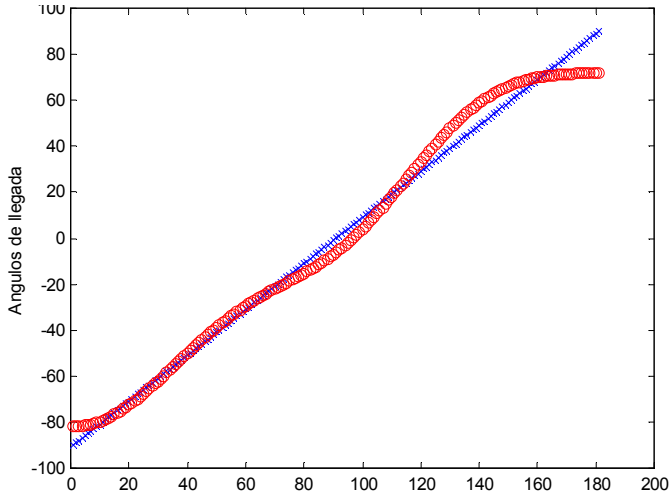


Figura 4.1. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo:

MSE = 0.0098

Para una separación entre los elementos de la agrupación $d = \lambda/6$

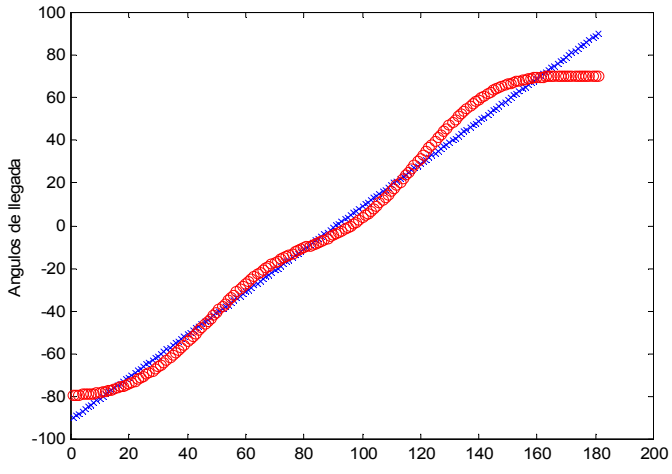


Figura 4.2. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo

MSE = 0.0115

Para una separación entre los elementos de la agrupación $d = \lambda/4$

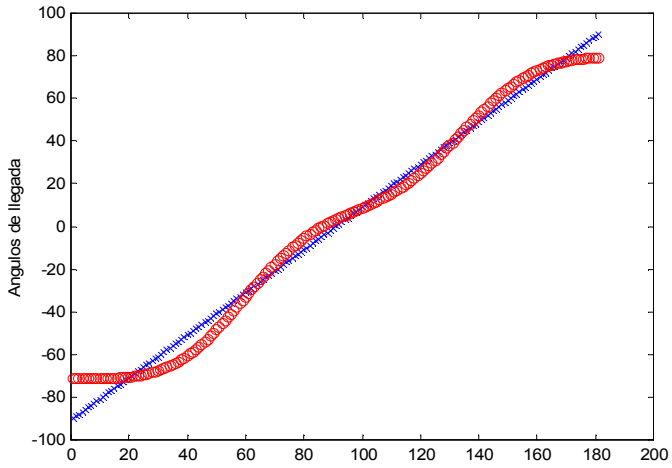


Figura 4.3. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo

MSE = 0.0108

Ahora para **5 neuronas intermedias**:

Para una separación entre los elementos de la agrupación $d = \lambda/8$

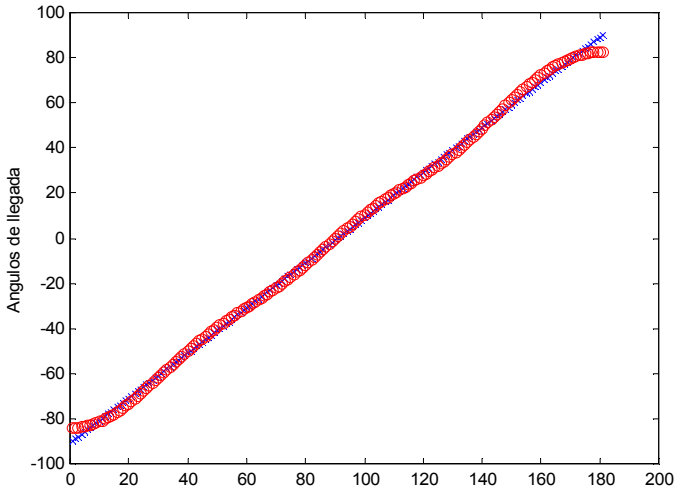


Figura 4.4. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo

MSE = 0.0011

Para una separación entre los elementos de la agrupación $d = \lambda/6$

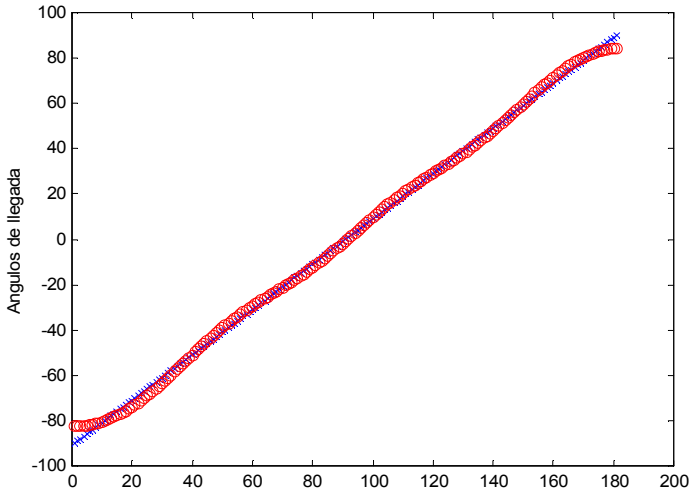


Figura 4.5. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo:

MSE = 0.0011

Para una separación entre los elementos de la agrupación $d = \lambda/4$

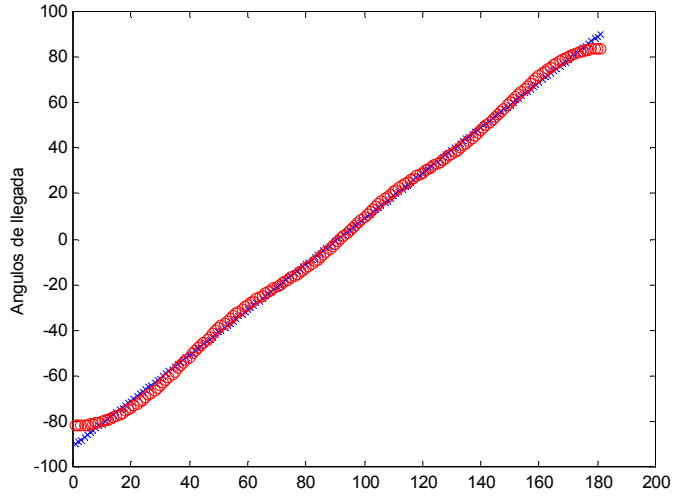


Figura 4.6. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo:

MSE = 0.0014

Vemos que al aumentar el número de neuronas los resultados mejoran ligeramente pero todavía haría falta aumentar el número de neuronas intermedias.

Ahora pasamos a simular con los siguientes nuevos parámetros:

- **3 elementos** en la agrupación de antenas.
- Una **dimensión de entrada a la red neuronal igual a 5**.
- **3 neuronas intermedias**.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

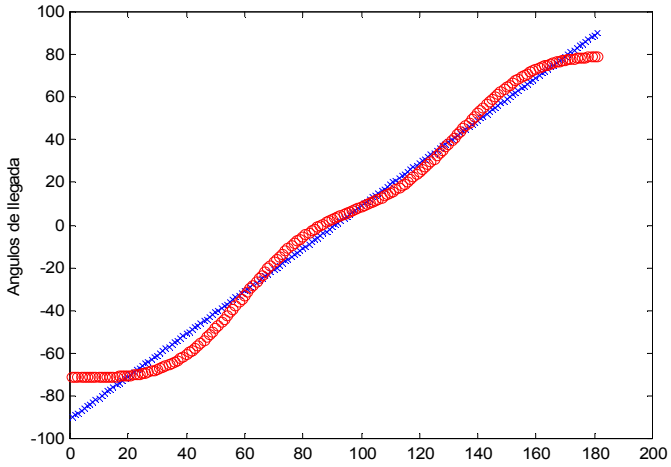


Figura 4.7. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo

MSE = 0.0109

Para una separación entre los elementos de la agrupación $d = \lambda/6$

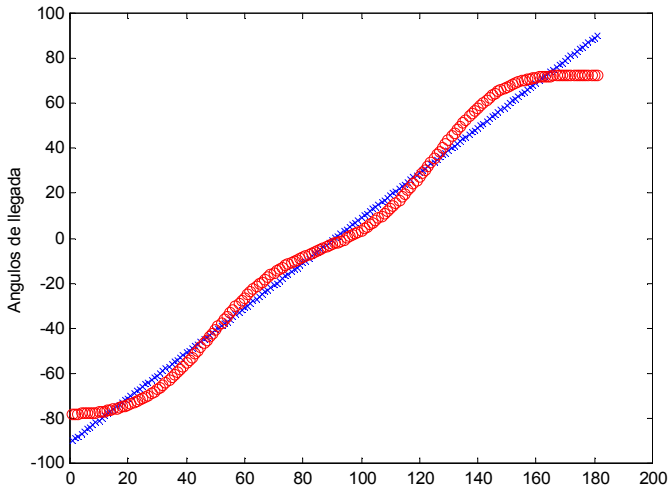


Figura 4.8. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo

MSE = 0.0108

Para una separación entre los elementos de la agrupación $d = \lambda/4$

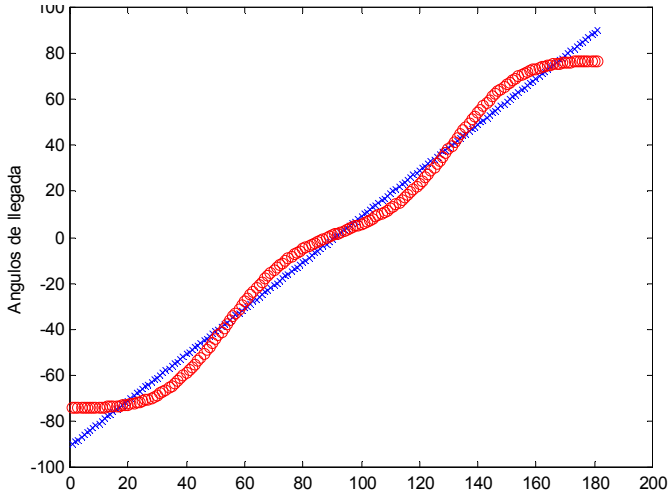


Figura 4.9. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo

MSE = 0.0111

Ahora para **5 neuronas intermedias**:

Para una separación entre los elementos de la agrupación $d = \lambda/8$

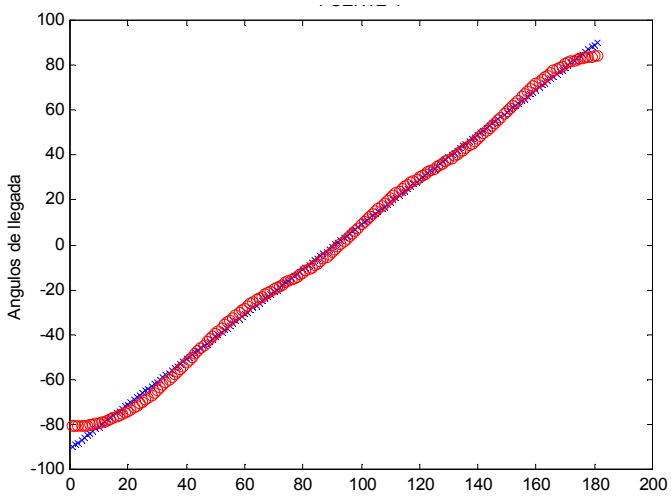


Figura 4.10. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo

MSE = 0.0018

Para una separación entre los elementos de la agrupación $d = \lambda/6$

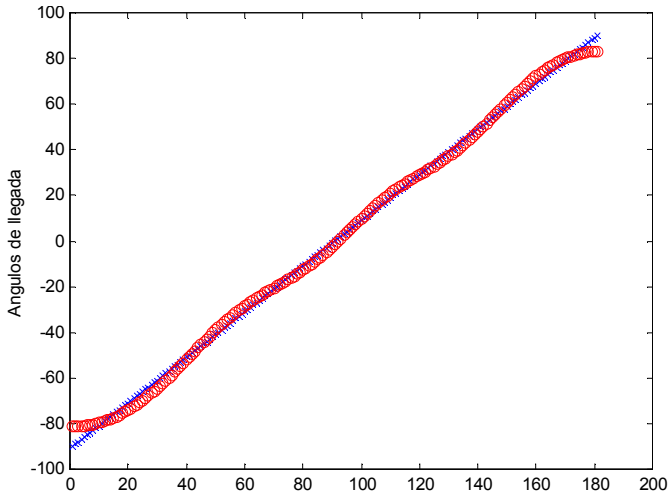


Figura 4.11. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo

MSE = 0.0018

Para una separación entre los elementos de la agrupación $d = \lambda/4$

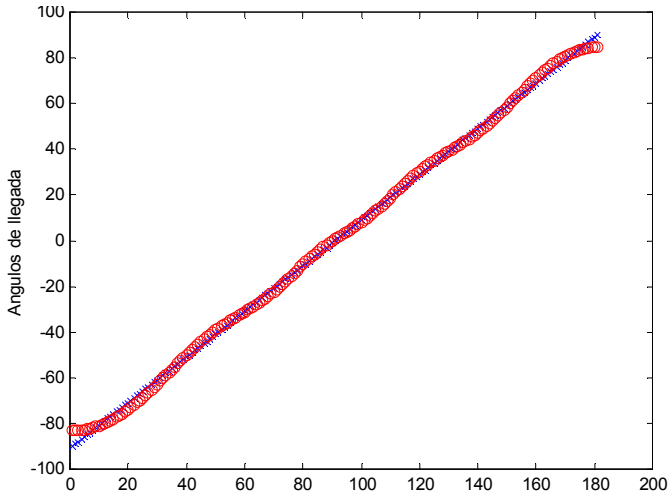


Figura 4.12. Salida de la RBF (rojo) frente a la salida deseada (azul)

Obteniendo

MSE = 9.2050e-004

Ahora veremos el caso:

- **6 elementos** en la agrupación de antenas.
- Una **dimensión de entrada a la red neuronal igual a 11**.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

5 neuronas

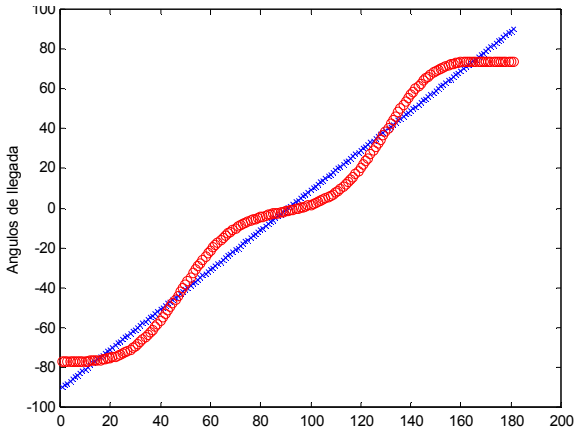


Figura 4.12. Salida de la RBF para 5 neuronas

MSE = 0.0179

7 neuronas

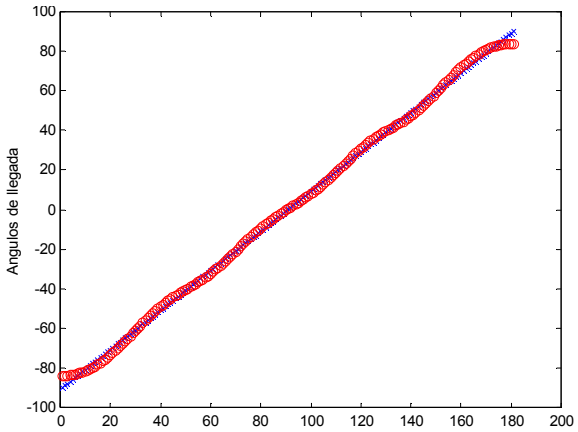


Figura 4.13. Salida de la RBF para 7 neuronas

MSE = 9.3403e-004

11 neuronas

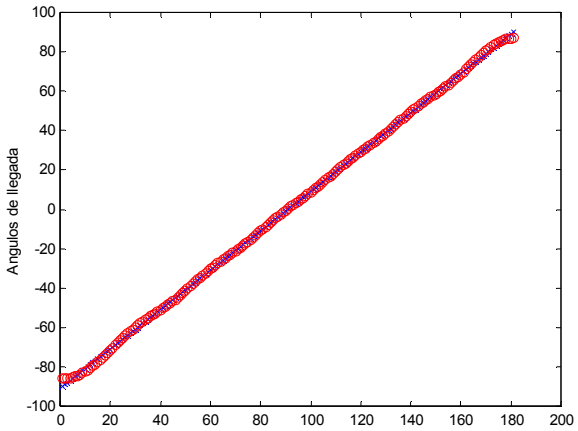


Figura 4.14. Salida de la RBF para 11 neuronas

MSE = 1.9541e-004

15 neuronas

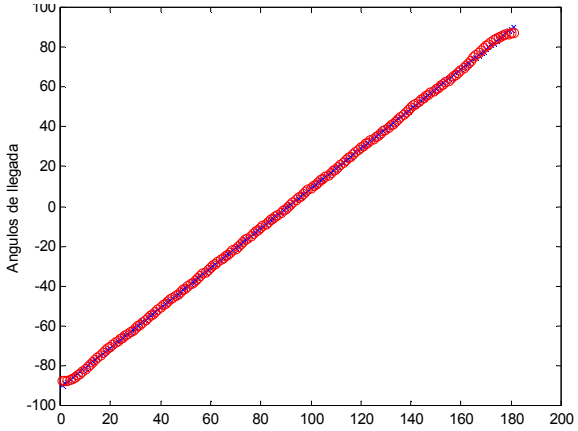


Figura 4.15. Salida de la RBF para 15 neuronas

MSE = 9.8428e-005

Para una separación entre los elementos de la agrupación $d = \lambda/4$

5 neuronas

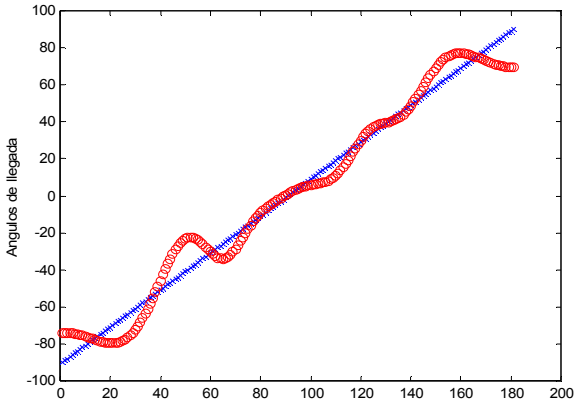


Figura 4.16. Salida de la RBF para 5 neuronas

MSE = 0.0209

7 neuronas

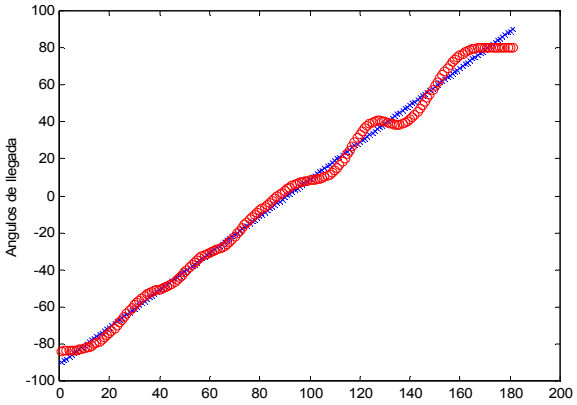


Figura 4.17. Salida de la RBF para 7 neuronas

MSE =0.0043

11 neuronas

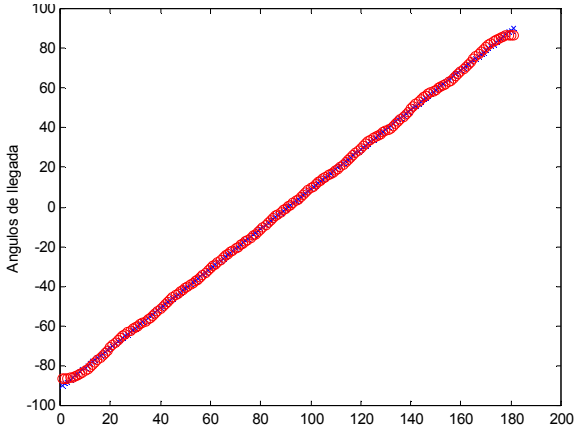


Figura 4.18. Salida de la RBF para 11 neuronas

MSE = 2.2495e-004

15 neuronas

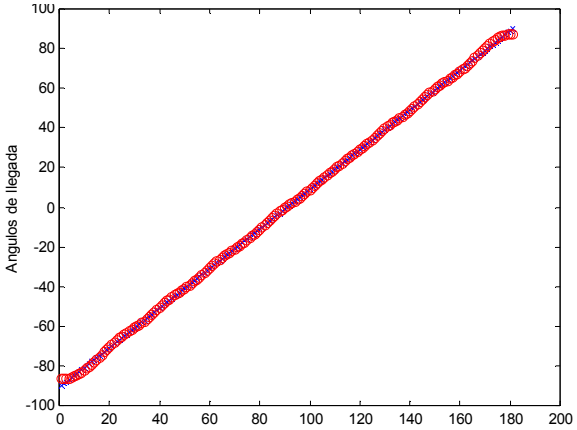


Figura 4.19. Salida de la RBF para 15 neuronas

MSE = 1.3657e-004

Vemos que 5 neuronas no son suficientes para una buena aproximación, **cuando el número de neuronas ocultas supera la dimensión de entrada nos acercamos a resultados satisfactorios como podemos ver en las *figuras 4.15 y 4.19*.**

- **Red PPL**

Para una señal de llegada tendremos los siguientes parámetros:

- **2 elementos** en la agrupación de antenas.
- Una **dimensión de entrada a la red neuronal igual a 3**.
- **3 neuronas intermedias**.

En este caso iremos variando el grado del polinomio de hermite que viene definido pro el parámetro R desde R = 1 hasta R = 9.

Las simulaciones que a continuación vamos a ver son algunas de las muchas realizadas para calcular el error cometido que **al final veremos en unas tablas donde se podrá apreciar la diferencia entre la RBF y la PPL.**

Para una separación entre los elementos de la agrupación $d = \lambda/8$

- **Para R =1**

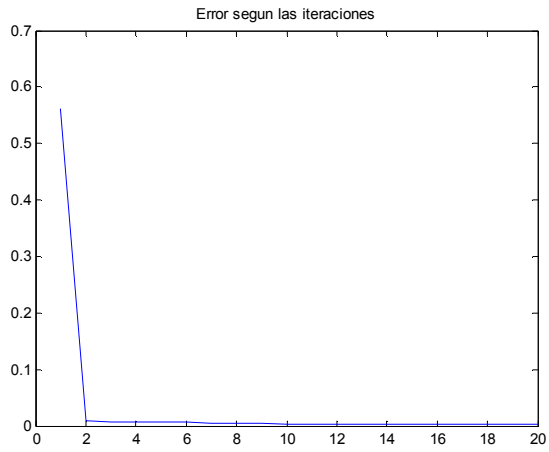


Figura 4.20. Evolución del error según las iteraciones

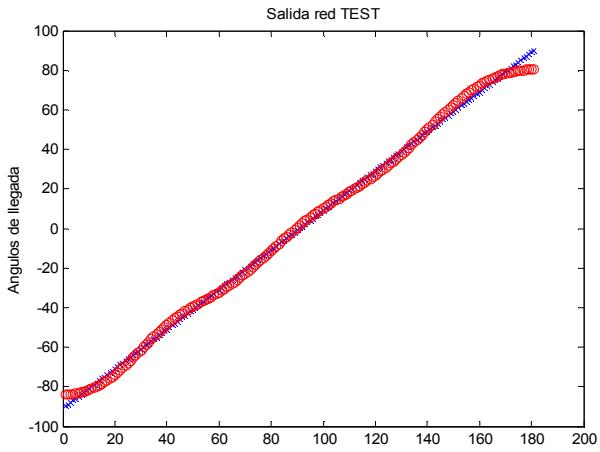


Figura 4.21. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo:

MSE = 0.0018

- **Para R = 5**

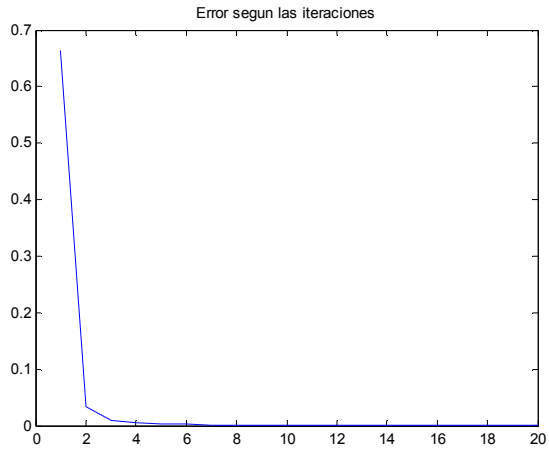


Figura 4.22. Evolución del error según las iteraciones

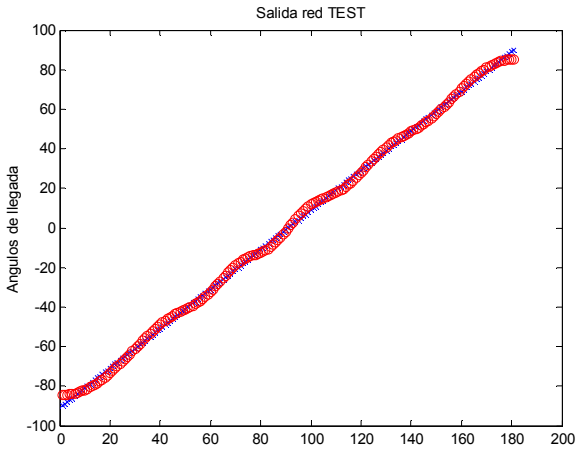


Figura 4.23. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 9.5974e-004**

- **Para R = 9**

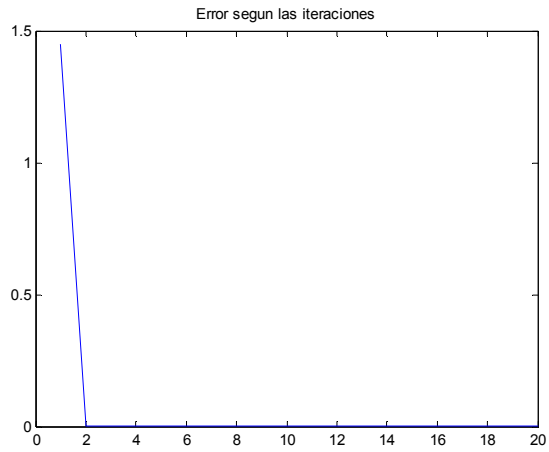


Figura 4.24. Evolución del error según las iteraciones

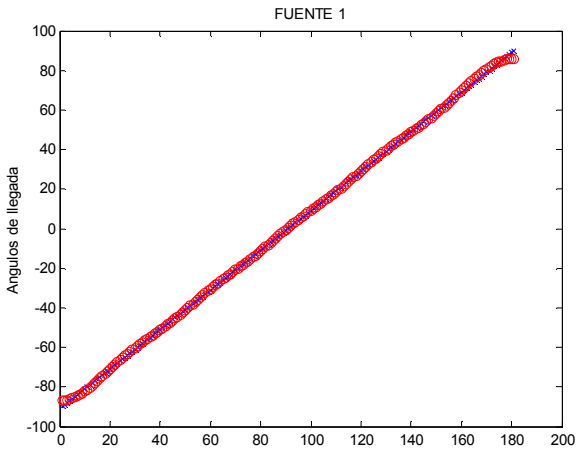


Figura 4.25. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 2.1776e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/6$

- **Para R =1**

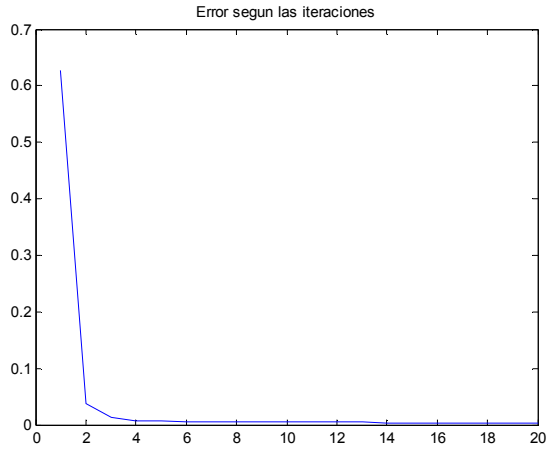


Figura 4.26. Evolución del error según las iteraciones

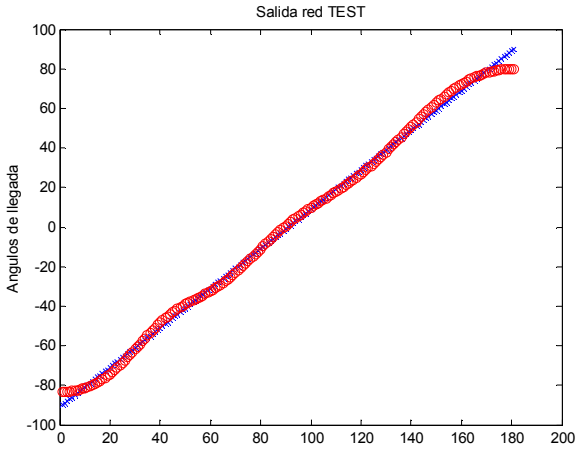


Figura 4.27. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 2e-003**

- **Para R = 5**

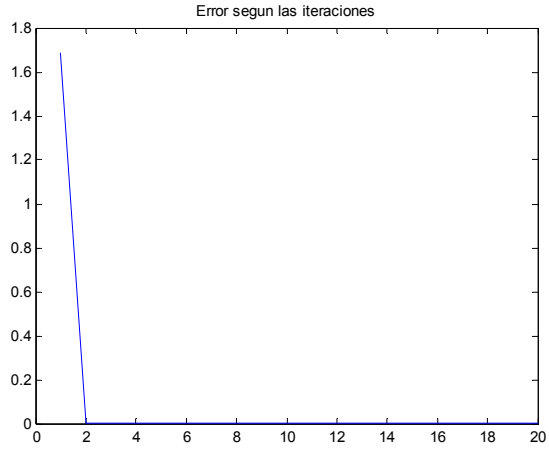


Figura 4.28. Evolución del error según las iteraciones

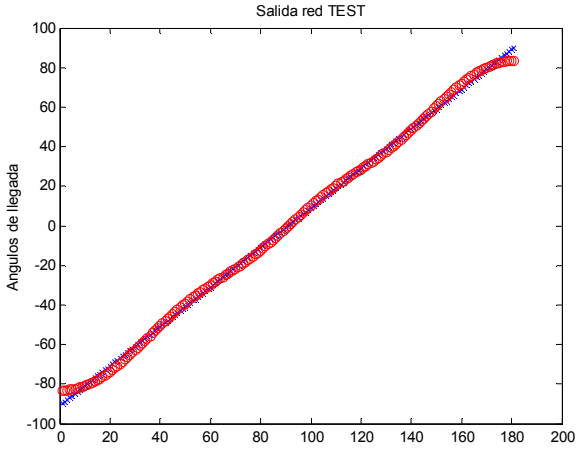


Figura 4.29. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 1.1e-003**

- **Para R = 9**

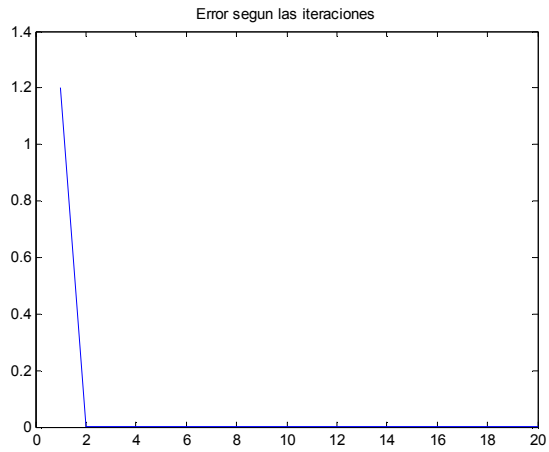


Figura 4.30. Evolución del error según las iteraciones

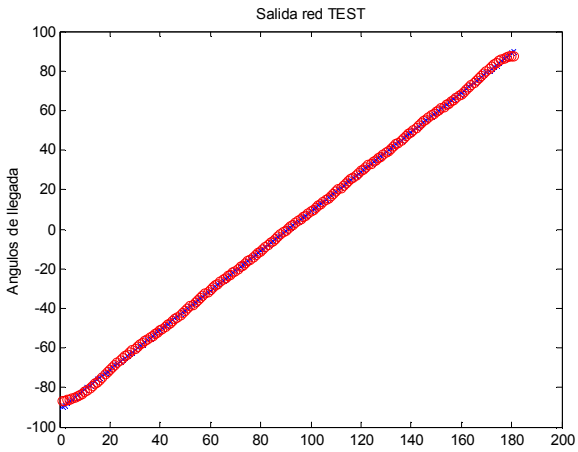


Figura 4.31. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 1.2004e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

- **Para R =1**

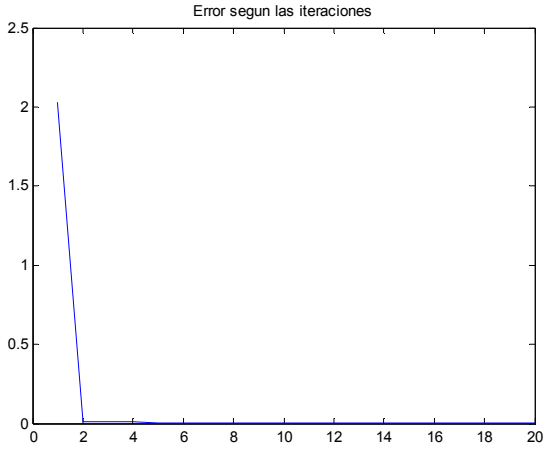


Figura 4.32. Evolución del error según las iteraciones

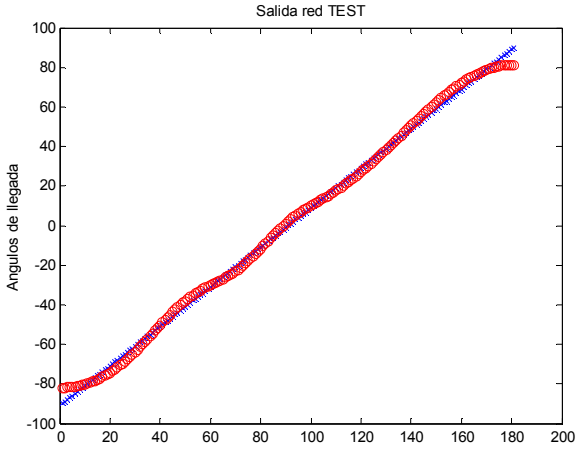


Figura 4.33. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 2e-003**

- **Para R = 5**

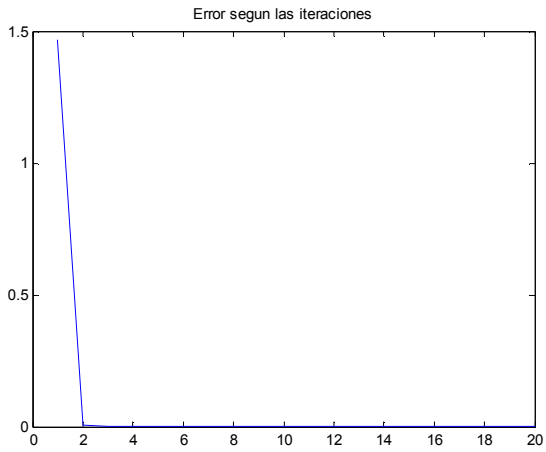


Figura 4.34. Evolución del error según las iteraciones

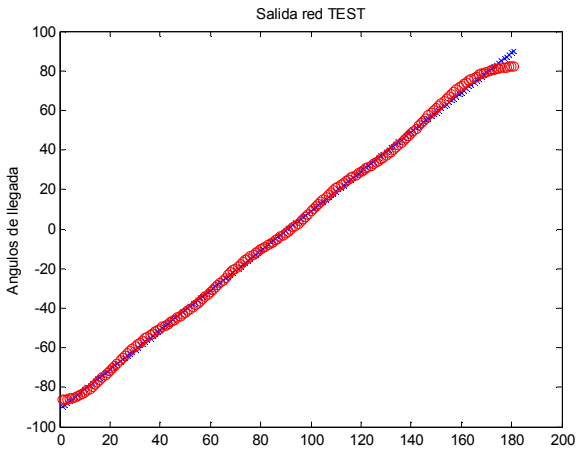


Figura 4.35. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 9.7859e-004**

- **Para R = 9**

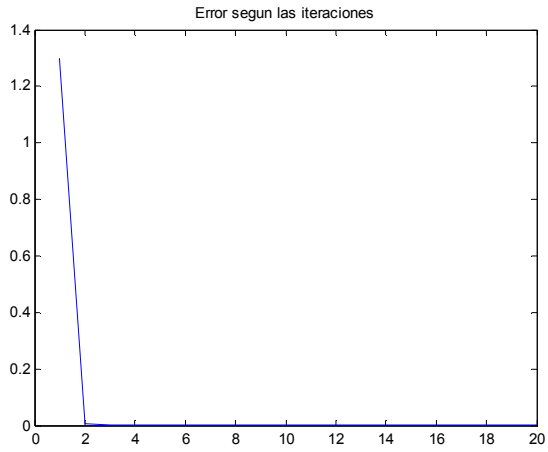


Figura 4.36. Evolución del error según las iteraciones

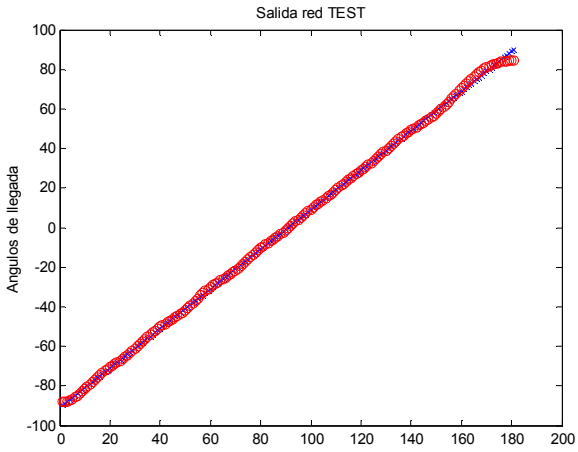


Figura 4.37. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 3.1149e-004**

Podemos apreciar como obtenemos mejores resultados que la red RBF y como el error va disminuyendo conforme aumenta el valor del parámetro R.

Ahora para **5 neuronas intermedias**:

Para una separación entre los elementos de la agrupación $d = \lambda/8$

- **Para R =1**

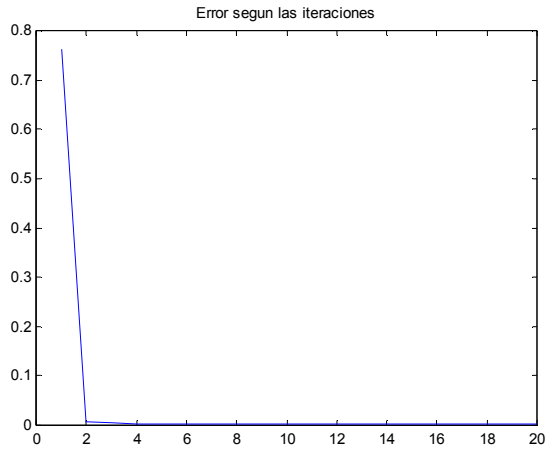


Figura 4.38. Evolución del error según las iteraciones

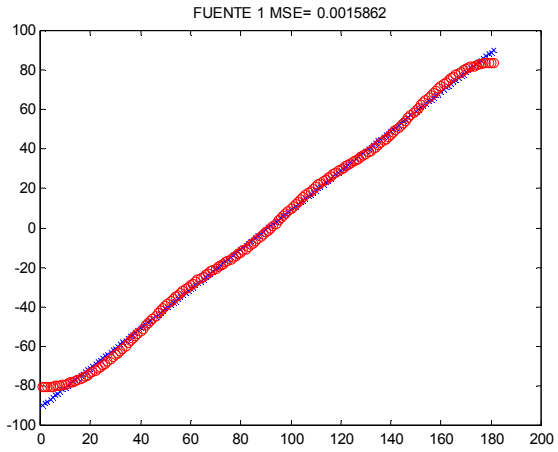


Figura 4.39. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 1.586e-003**

- **Para R = 5**

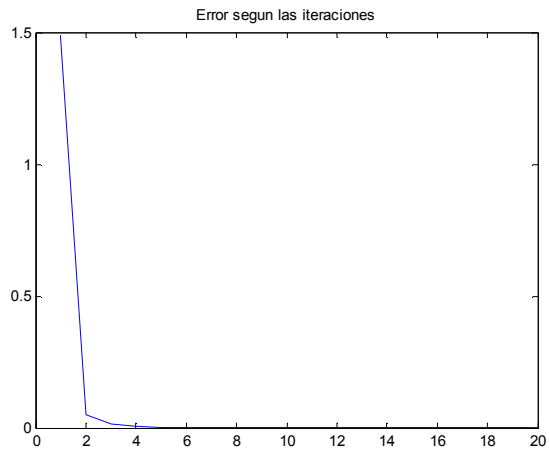


Figura 4.40. Evolución del error según las iteraciones

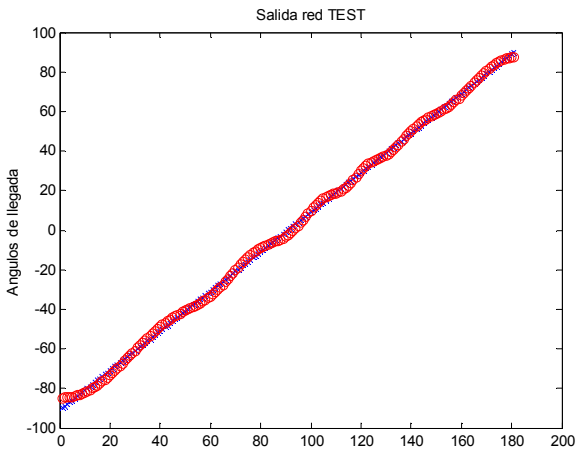


Figura 4.41. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 7.6964e-004**

- **Para R = 9**

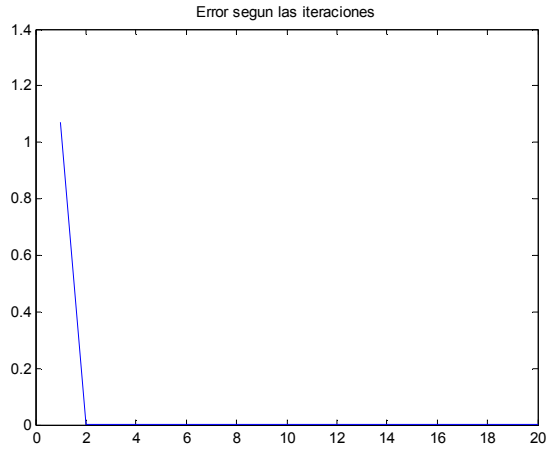


Figura 4.42. Evolución del error según las iteraciones

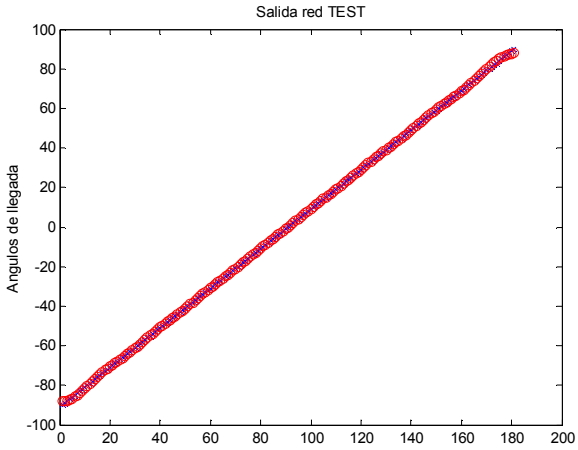


Figura 4.43. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 4.9572e-005**

Para una separación entre los elementos de la agrupación $d = \lambda/6$

- **Para R =1**

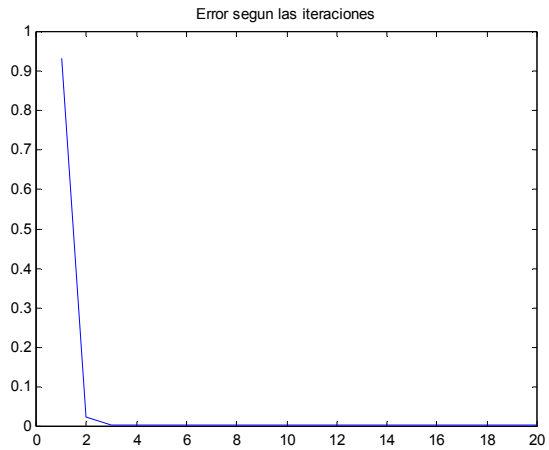


Figura 4.44. Evolución del error según las iteraciones

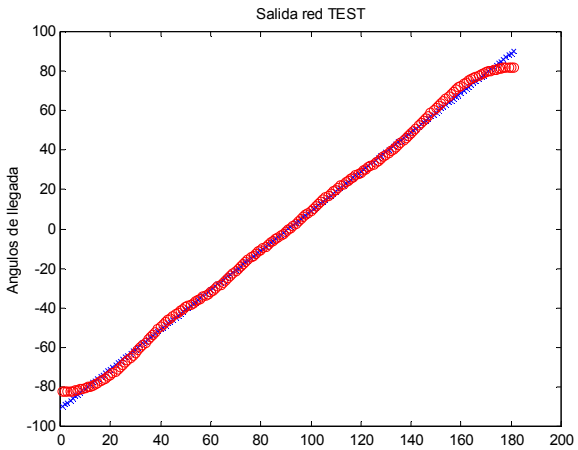


Figura 4.45. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE =1.4e-003**

- **Para R = 5**

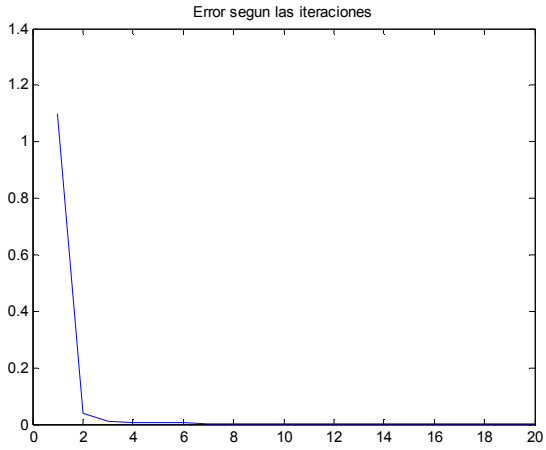


Figura 4.46. Evolución del error según las iteraciones

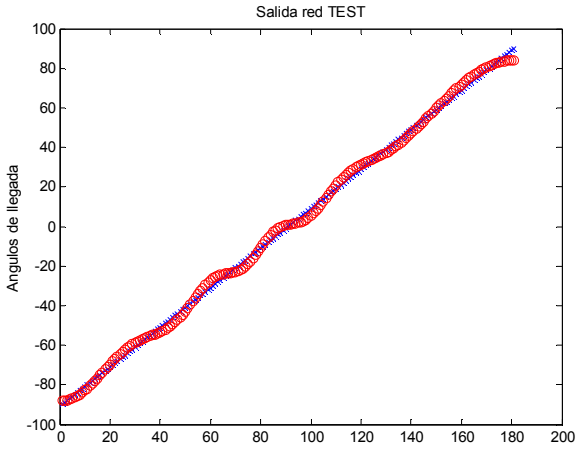


Figura 4.47. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 1.6e-003**

- **Para R = 9**

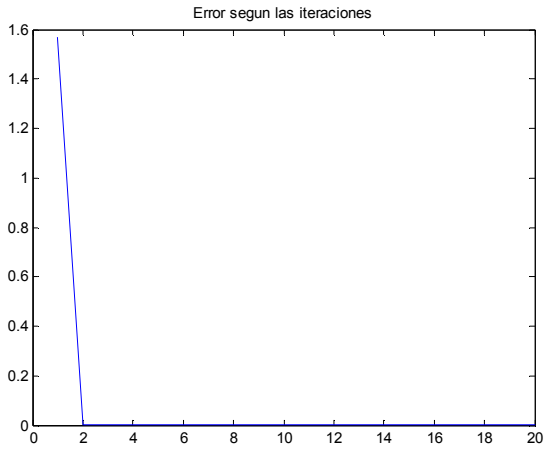


Figura 4.48. Evolución del error según las iteraciones

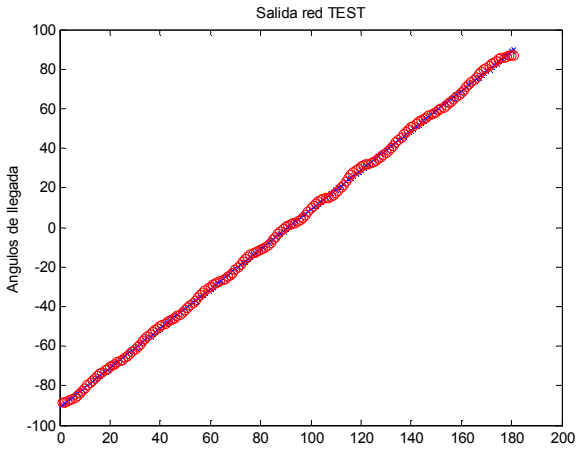


Figura 4.49. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 2.8372e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

- **Para R =1**

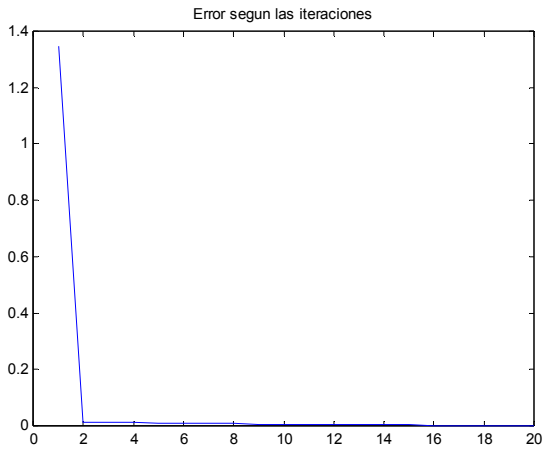


Figura 4.50. Evolución del error según las iteraciones

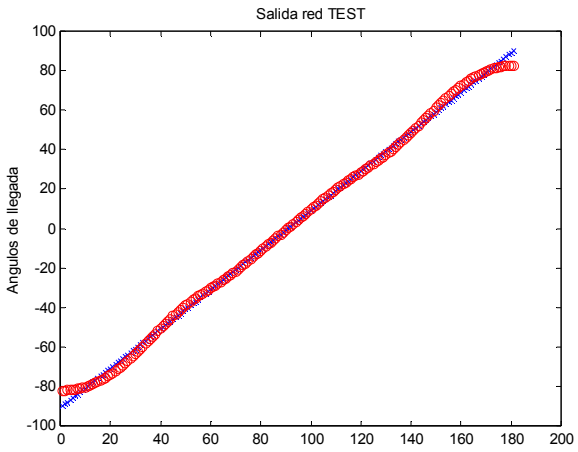


Figura 4.51. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 1.423e-004**

- **Para R = 5**

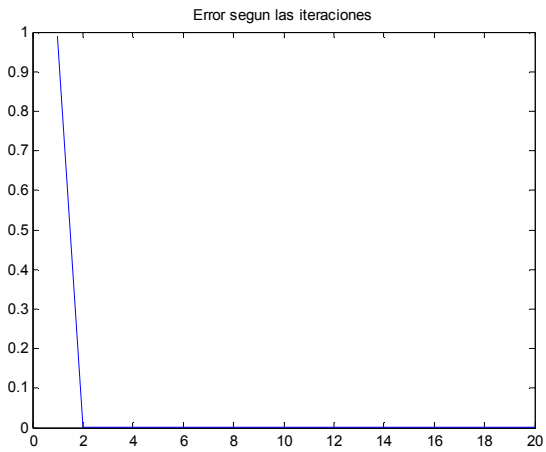


Figura 4.52. Evolución del error según las iteraciones

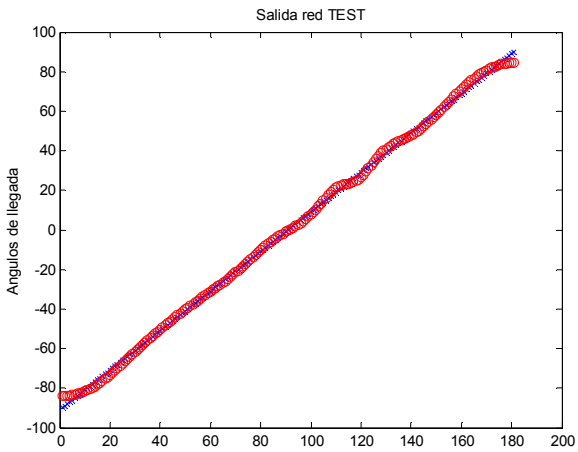


Figura 4.53. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 7.8007e-004**

- **Para R = 9**

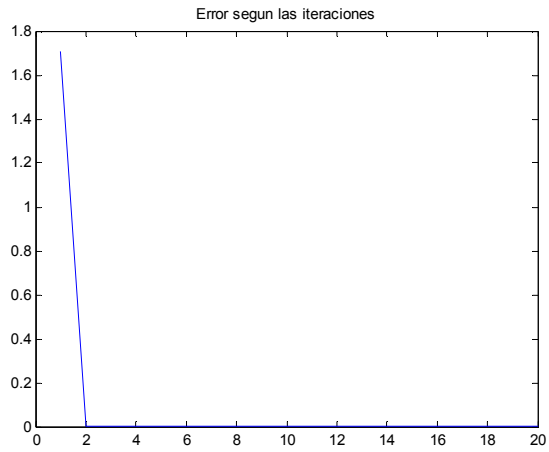


Figura 4.54. Evolución del error según las iteraciones

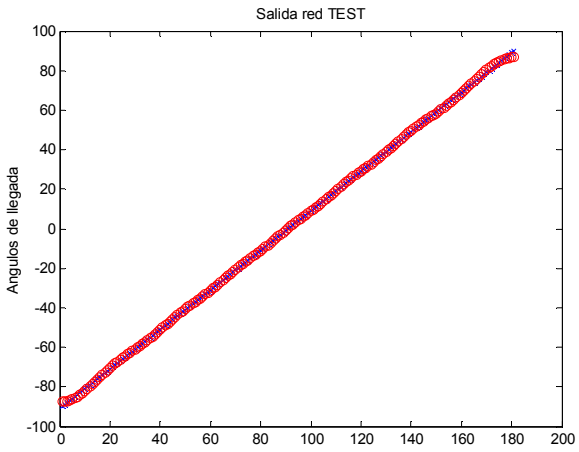


Figura 4.55. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 1.2348e-004**

Ahora pasamos a la siguiente configuración

- **3 elementos** en la agrupación de antenas.
- Una **dimensión de entrada a la red neuronal igual a 5**.
- **3 neuronas intermedias**.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

- **Para R =1**

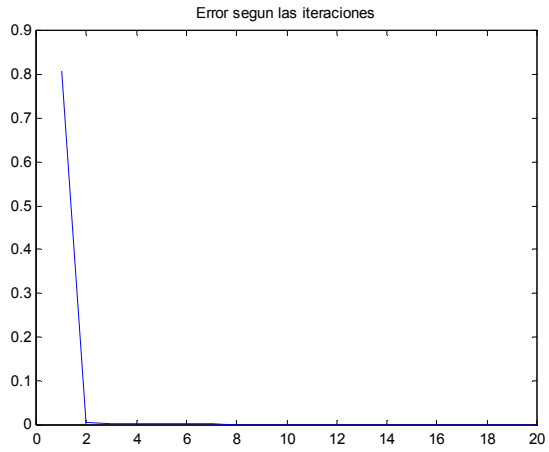


Figura 4.56. Evolución del error según las iteraciones

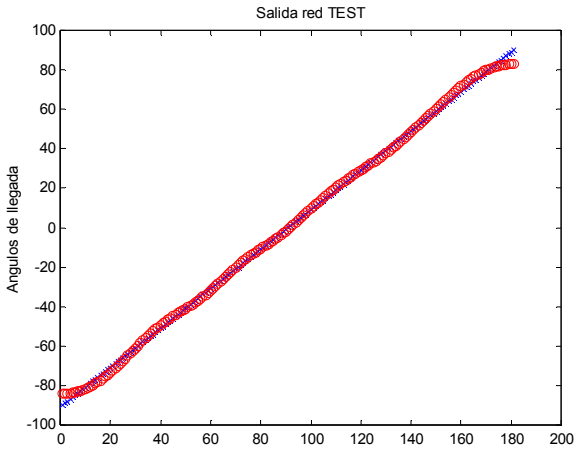


Figura 4.57. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 8.7289e-004**

- **Para R = 5**

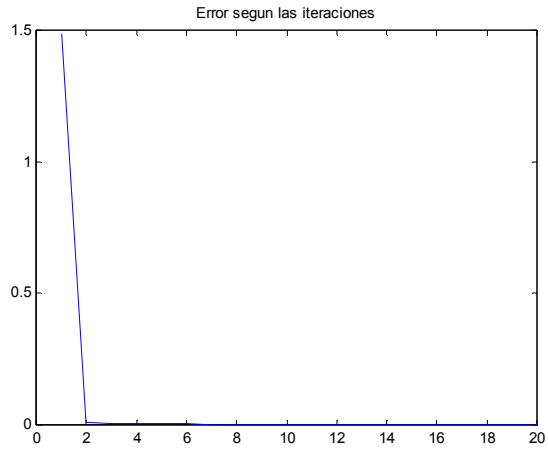


Figura 4.58. Evolución del error según las iteraciones

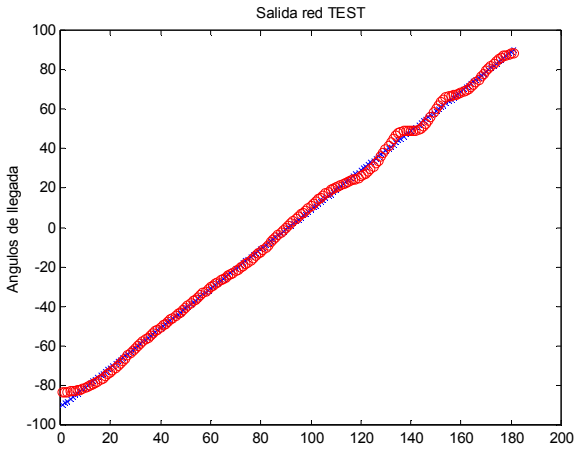


Figura 4.59. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 9.4161e-004**

- **Para R = 9**

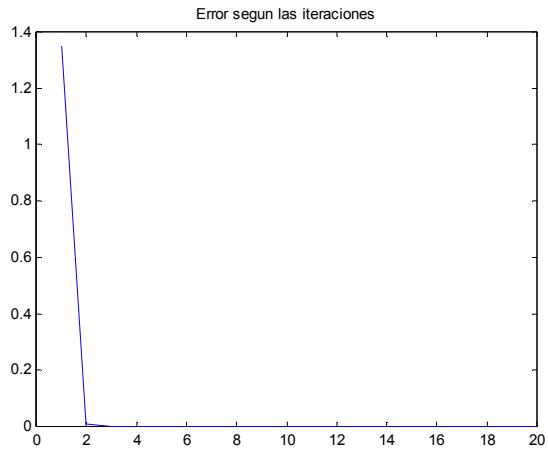


Figura 4.60. Evolución del error según las iteraciones

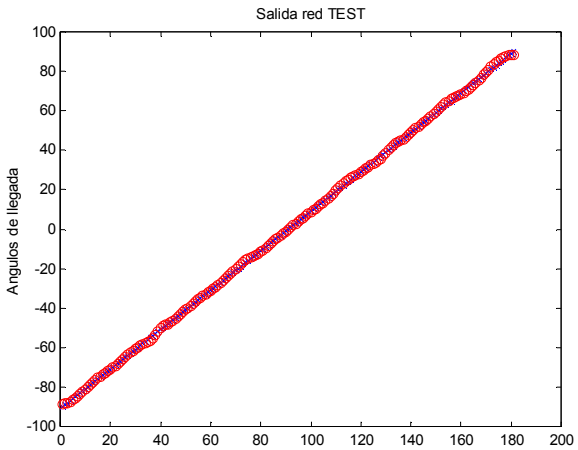


Figura 4.61. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 1.2297e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/6$

- **Para R =1**

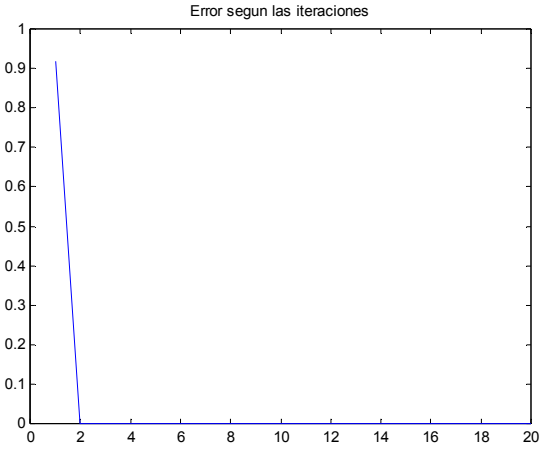


Figura 4.62. Evolución del error según las iteraciones

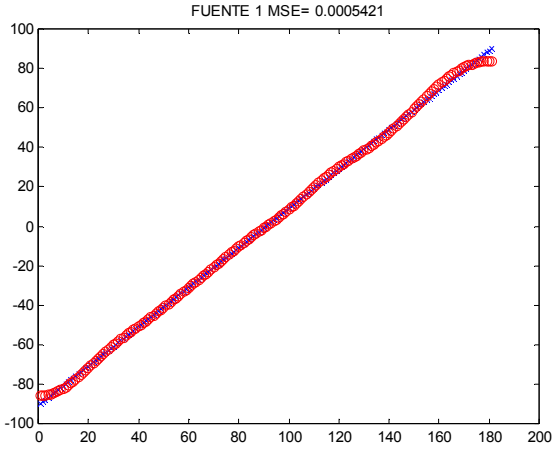


Figura 4.63. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 5.4210e-004**

- **Para R = 5**

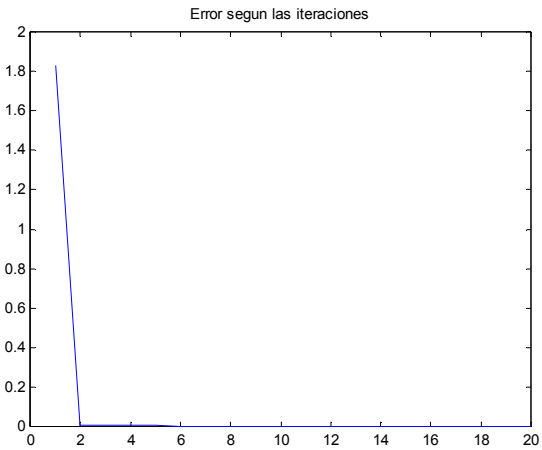


Figura 4.64. Evolución del error según las iteraciones

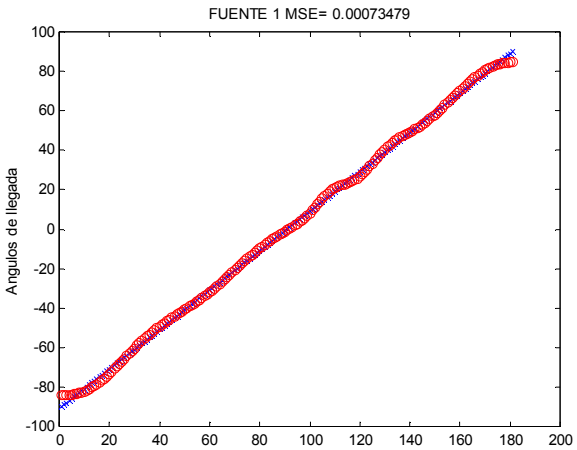


Figura 4.65. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 7.3479e-004**

- **Para R = 9**

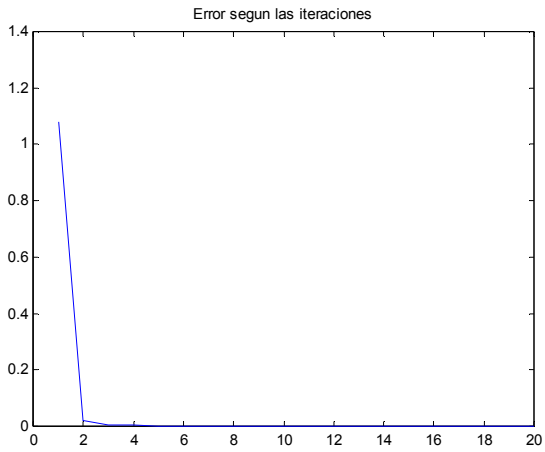


Figura 4.66. Evolución del error según las iteraciones

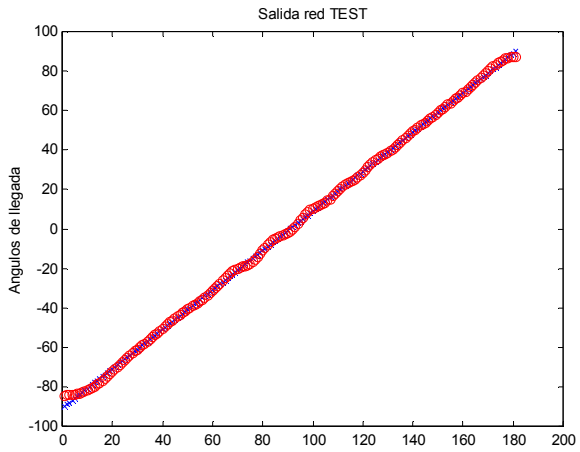


Figura 4.67. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 3.3039e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

- **Para R =1**

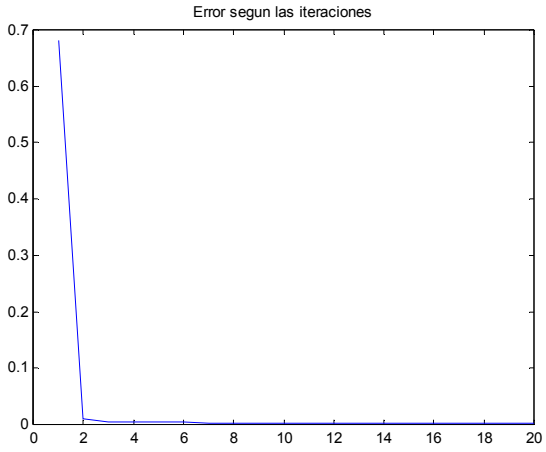


Figura 4.68. Evolución del error según las iteraciones

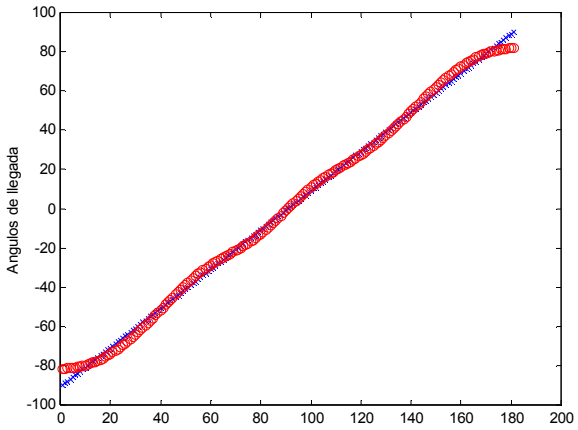


Figura 4.69. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 2e-003**

- **Para R = 5**

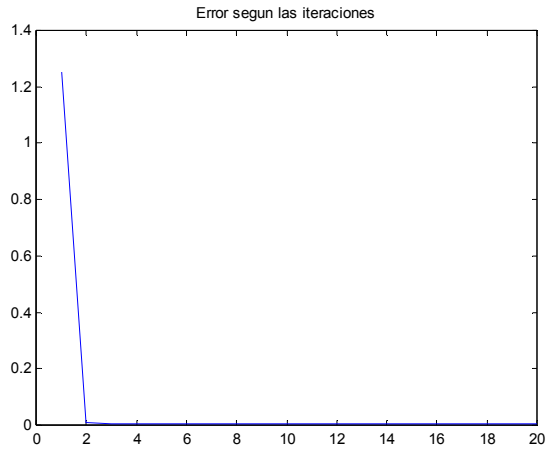


Figura 4.70. Evolución del error según las iteraciones

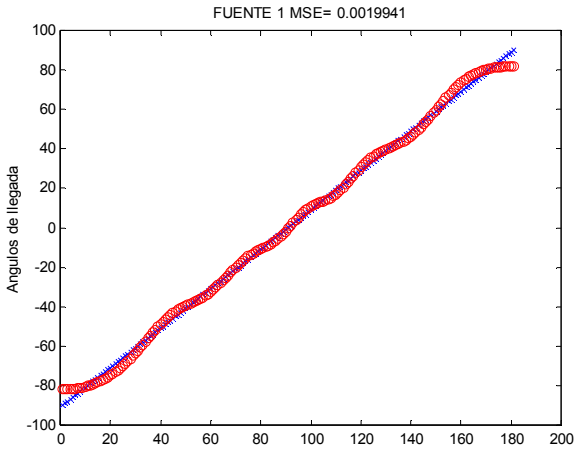


Figura 4.71. Salida de la red PPL (rojo) frente la deseada(azul)

Obteniendo: **MSE = 2.3421e-003**

- **Para R = 9**

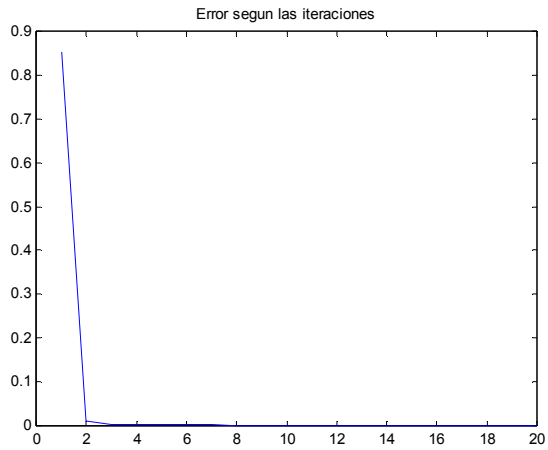


Figura 4.72. Evolución del error según las iteraciones

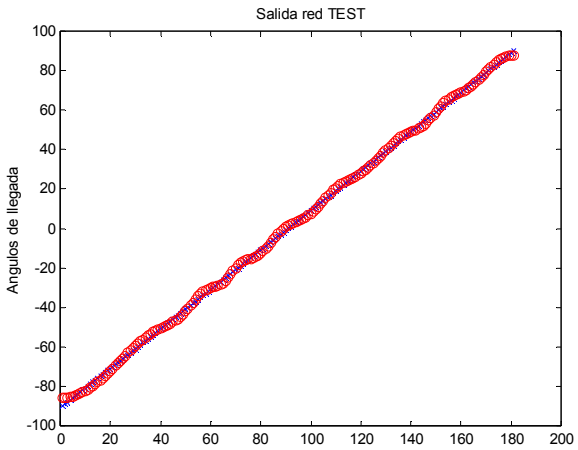


Figura 4.73. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 4.5982e-004**

Ahora para **5 neuronas** intermedias mostraremos los **mejores resultados que fueron obtenidos para el valor de R = 9.** La salida de la red para los distintos valores de R lo podremos observar en las tablas que mas adelante mostraremos a modo de resumen.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

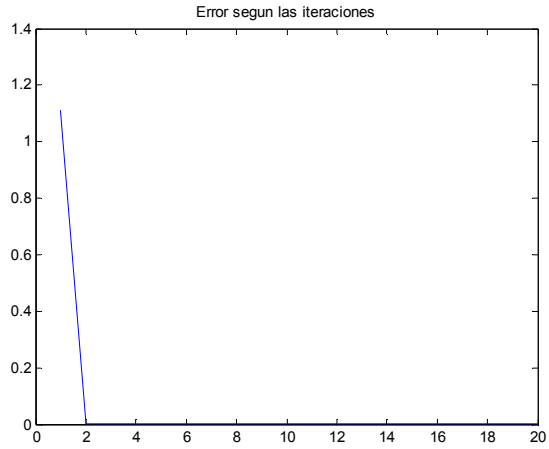


Figura 4.74. Evolución del error según las iteraciones

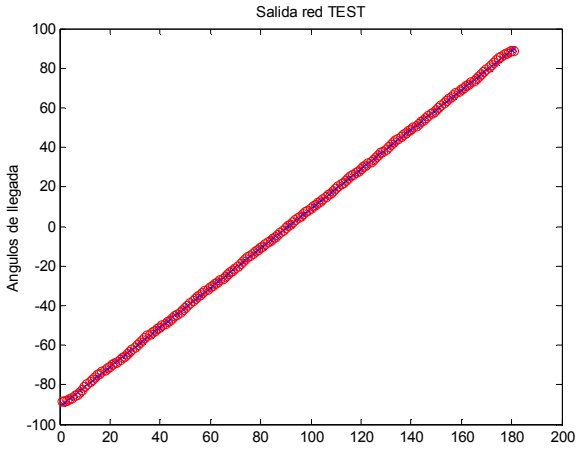


Figura 4.75. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 4.2759e-005**

Para una separación entre los elementos de la agrupación $d = \lambda/6$

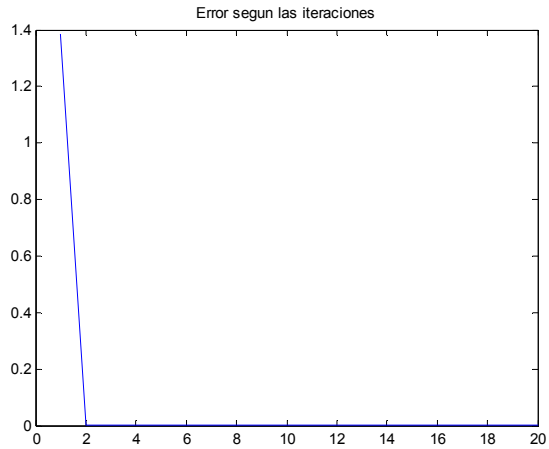


Figura 4.76. Evolución del error según las iteraciones

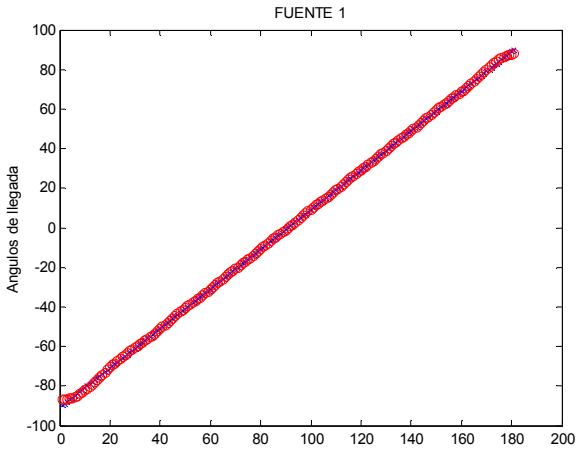


Figura 4.77. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 1.7951e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

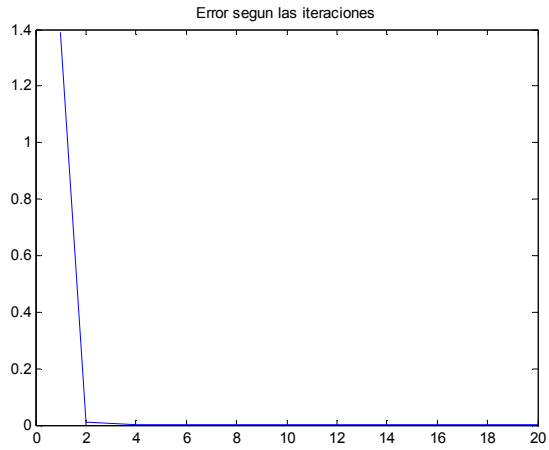


Figura 4.78. Evolución del error según las iteraciones

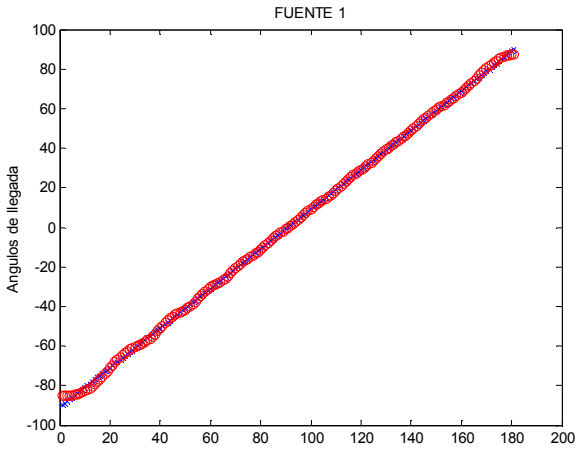


Figura 4.79. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 2.7454e-004**

Ahora pasaremos a una configuración con más elementos en la agrupación de antenas para que de esta forma aumente la dimensión de entrada de la red neuronal y así veamos una mejor comparativa con la RBF en cuestión de necesidad de neuronas intermedias para resolver el problema

- **6 elementos en la agrupación de antenas.**
- **Una dimensión de entrada a la red neuronal igual a 11.**

Las simulaciones que a continuación mostramos para **distinto numero de neuronas intermedias** son para **R =1** con el que **obtenemos los mejores resultados para este problema**. Los resultados para valores distintos de R los mostraremos en las tablas finales.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

- **3 neuronas**

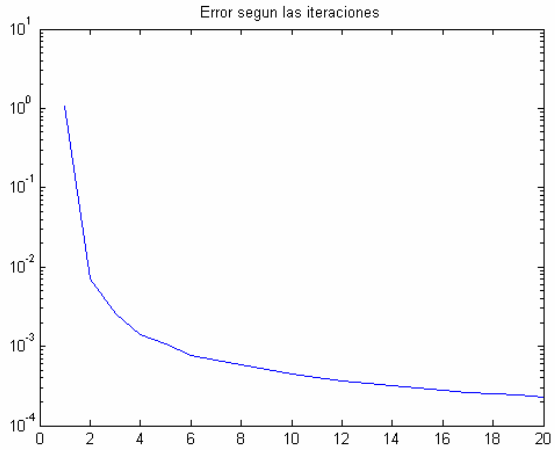


Figura 4.80. Evolución del error según las iteraciones

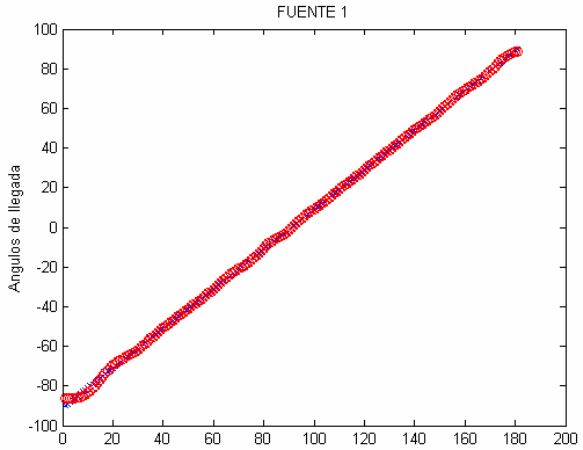


Figura 4.81. Salida de la red PPL (rojo) frente la deseada (azul)

Obteniendo: **MSE = 1.7183e-004**

- **7 neuronas**

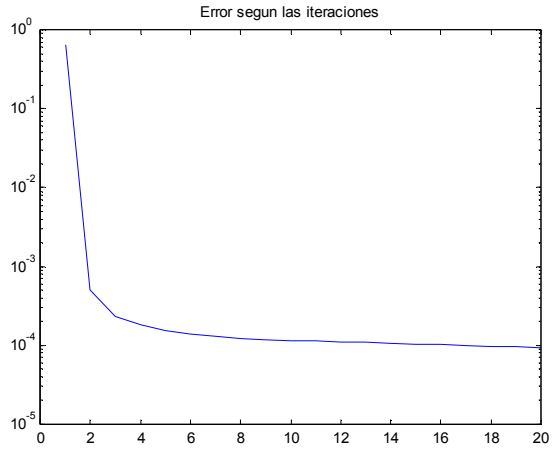


Figura 4.82. Evolución del error según las iteraciones

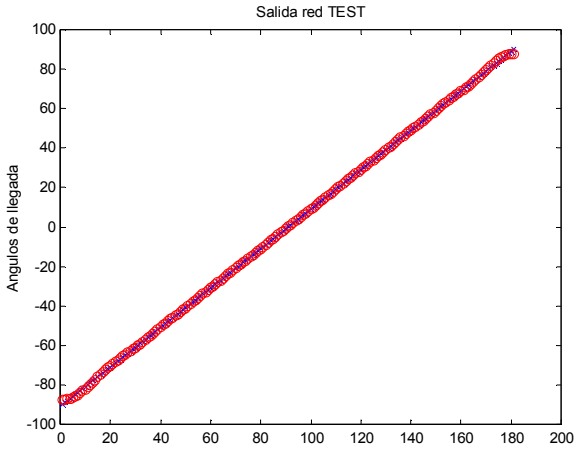


Figura 4.83. Salida de la red PPL (rojo) frente a la deseada (azul)

Obteniendo: **MSE = 6.3726e-005**

- **11 neuronas**

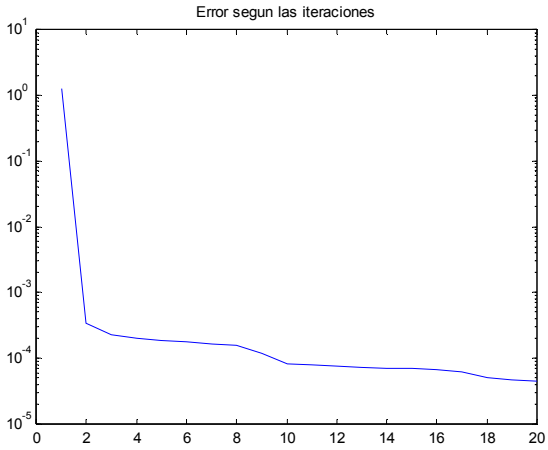


Figura 4.84. Evolución del error según las iteraciones

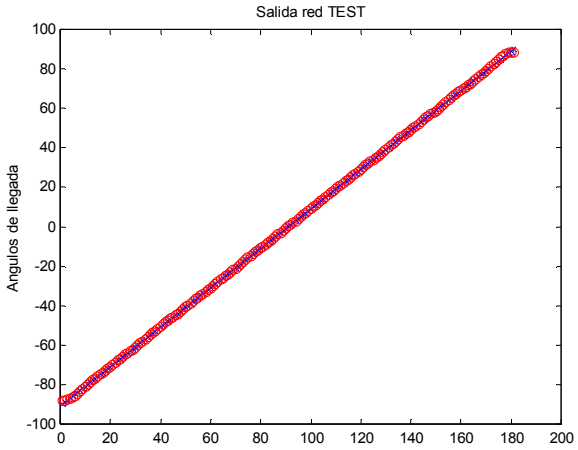


Figura 4.85. Salida de la red PPL (rojo) frente a la deseada (azul)

Obteniendo: **MSE = 2.8906e-005**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

- **3 neuronas**

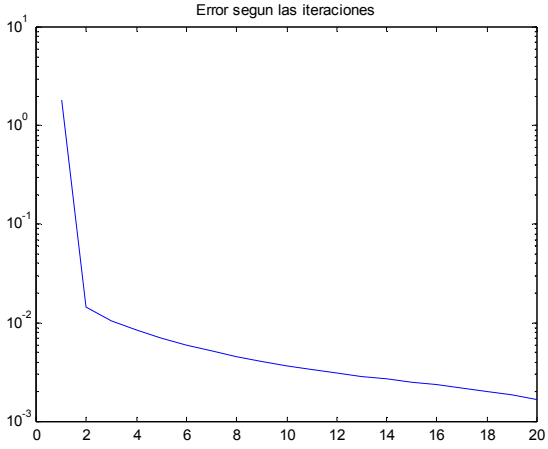


Figura 4.86. Evolución del error según las iteraciones

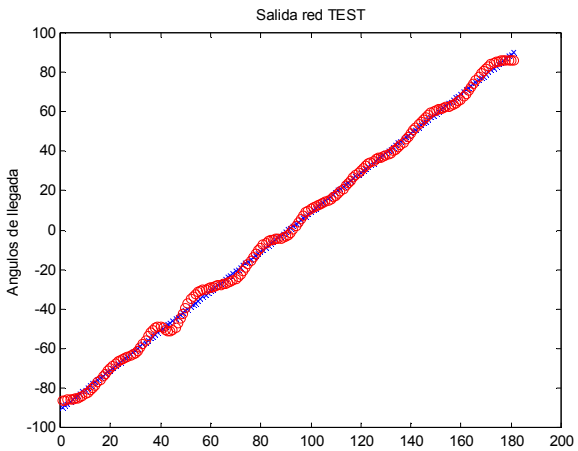


Figura 4.87. Salida de la red PPL (rojo) frente a la deseada (azul)

Obteniendo: **MSE = 1e-003**

- **7 neuronas**

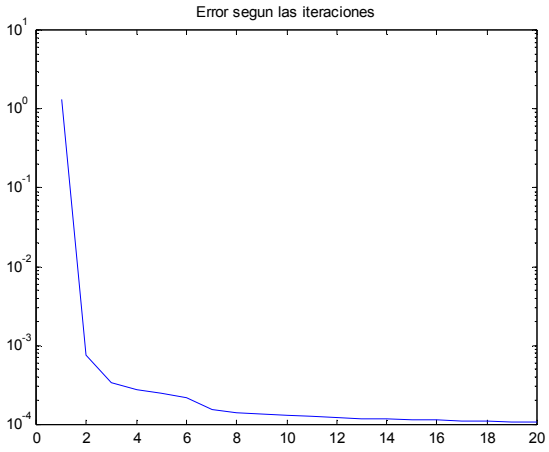


Figura 4.88. Evolución del error según las iteraciones

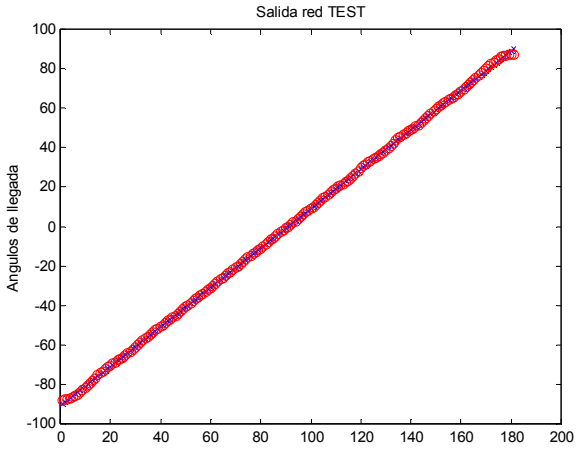


Figura 4.89. Salida de la red PPL (rojo) frente a la deseada (azul)

Obteniendo: **MSE = 7.2184e-005**

- **11 neuronas**

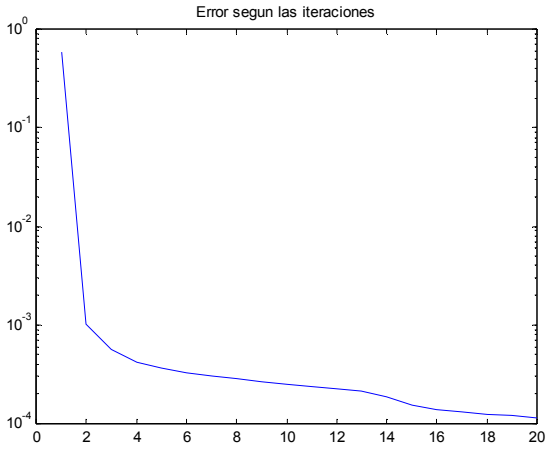


Figura 4.90. Evolución del error según las iteraciones

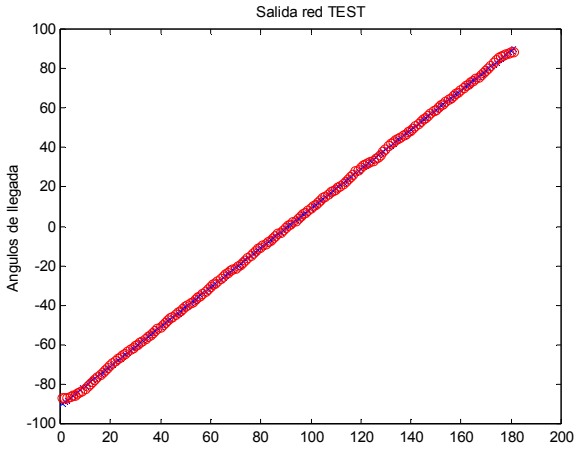


Figura 4.91. Salida de la red PPL (rojo) frente a la deseada (azul)

Obteniendo: **MSE = 5.9600e-005**

A continuación mostraremos las tablas a modo de resumen tanto de los resultados que usamos de guía obtenidos por la RBF de la toolbox de Matlab y los de la red PPL que hemos programado.

4.3.1 *TABLAS DE RESULTADOS*

Los resultados de las tablas es al media de realizar varias simulaciones y el parámetro de medida es el Error cuadrático medio (MSE).

- **RBF**

2 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 3

	$d = \lambda/8$	$d = \lambda/6$	$d = \lambda/4$
Neuronas			
3	**0.0098**	**0.0115**	**0.0108**
5	**0.0011**	**0.0011**	**0.0014**

Tabla 4.1. MSE para 2 antenas en al agrupación y diferentes distancias de separación

3 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 5

	$d = \lambda/8$	$d = \lambda/6$	$d = \lambda/4$
Neuronas			
3	**0.0109**	**0.0115**	**0.0111**
5	**0.0012**	**0.0018**	**0.0019**

Tabla 4.2. MSE para 3 antenas en al agrupación y diferentes distancias de separación

6 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 11

	$d = \lambda/8$	$d = \lambda/4$
Neuronas		
5	**1.79e-002**	**2.091e-002**
7	**9.3403e-004**	**4.3574e-003**
9	**5.3581e-004**	**7.3965e-004**
11	**1.9541e-004**	**2.2495e-004**
13	**1.1610e-004**	**1.9138e-004**
15	**9.8428e-005**	**1.3657e-004**
17	**6.2865e-005**	**9.0113e-005**
22	**4.3259e-005**	**4.6054e-005**

Tabla 4.3. MSE para 6 antenas en al agrupación y diferentes distancias de separación

- **PPL**

2 ANTENAS EN LA AGRPUACION
DIMENSION DE ENTRADA 3

$$d = \lambda/8$$

	R = 1	R = 3	R = 5	R = 7	R = 9
Neuronas					
3	0.00215	0.0018	0.00108	0.00033	0.000198
5	0.00178	0.00103	0.00048	0.000165	0.0000871

Tabla 4.4. MSE para 2 antenas en al agrupación

$$d = \lambda/6$$

	R = 1	R = 3	R = 5	R = 7	R = 9
Neuronas					
3	0.0022	0.00217	0.00104	0.00035	0.000166
5	0.00167	0.0013	0.00145	0.000183	0.000171

Tabla 4.5. MSE para 2 antenas en al agrupación

$$d = \lambda/4$$

	R = 1	R = 3	R = 5	R = 7	R = 9
Neuronas					
3	0.0033	0.00986	0.00113	0.00107	0.000443
5	0.00206	0.0023	0.000869	0.000185	0.000169

Tabla 4.6. MSE para 2 antenas en al agrupación

Observamos como con solo dos antenas en al agrupación con la red PPL somos capaces de obtener resultados bastantes satisfactorios, sin embargo la RBF queda bastante lejos de estos resultados.

3 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 5

$$d = \lambda/8$$

	R = 1	R = 3	R = 5	R = 9
Neuronas				
3	0.000898	0.000845	0.00111	0.000242
5	0.000672	0.000311	0.00061	0.00015

Tabla 4.7. MSE para 3 antenas en al agrupación

$$d = \lambda/6$$

	R = 1	R = 3	R = 5	R = 9
Neuronas				
3	0.000694	0.00268	0.000639	0.000323
5	0.000635	0.000782	0.000679	0.000126

Tabla 4.8. MSE para 3 antenas en al agrupación

$$d = \lambda/4$$

	R = 1	R = 3	R = 5	R = 9
Neuronas				
3	0.00163	0.000964	0.0018	0.000466
5	0.00053	0.0004769	0.00049	0.000483

Tabla 4.9. MSE para 3 antenas en al agrupación

Para 3 antenas en la agrupación los resultados obtenidos con la red PPL siguen siendo bastante mejores que la aproximación que realiza la RBF.

6 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 11

$$d = \lambda/8$$

	R = 1	R = 5	R = 9
Neuronas			
3	1.4715e-004	6.7152e-003	3.2543e-004
5	7.7813e-005	1.4826e-003	3.1364e-004
7	6.8265e-005	7.9701e-003	3.3802e-004
9	5.2314e-005	2.0320e-004	1.9754e-004

Tabla 4.10. MSE para 6 antenas en al agrupación

$$d = \lambda/4$$

	R = 1	R = 5	R = 9
Neuronas			
3	1.3893e-003	3.3401e-003	8.5022e-003
5	7.7386e-005	1.2568e-003	3.5309e-004
7	7.1097e-005	9.6834e-004	2.0775e-004
9	6.1769e-005	6.3644e-004	1.7782e-004

Tabla 4.11. MSE para 6 antenas en al agrupación

Para este ultimo caso de 6 elementos en la agrupación con R = 1 obtenemos los mejores resultados. Utilizando 11 neuronas intermedias igualando así la dimensión de entrada con la capa oculta obtenemos:

$d = \lambda/8$	$d = \lambda/4$
MSE = 3.4236e-005	**MSE = 5.5061e-005**

Vemos que la red PPL aproxima la función deseada con un número menor de neuronas intermedias que la dimensión de entrada, la RBF necesita tener un número de neuronas intermedias mayor que al dimensión de entrada para obtener resultados óptimos, esto se puede apreciar claramente para el último caso de 6 elementos en la agrupación.

4.4 DOS SEÑALES DE LLEGADA

- **Red RBF**

Para dos señales de llegada tendremos los siguientes parámetros:

- **3 elementos** en la agrupación de antenas.
- Una **dimensión de entrada a la red neuronal igual a 9**.
- **5 neuronas intermedias**.

Empezaremos con una separación espacial entre las dos señales de llegada de 5º.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

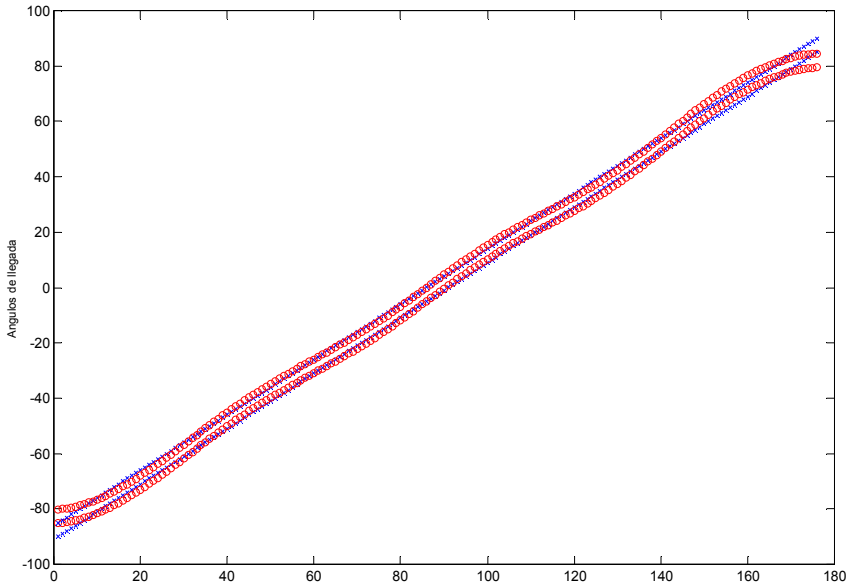


Figura 4.92. Salida de la red RBF (rojo) frente a la deseada.

Obteniendo: **MSE = 0.0015**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

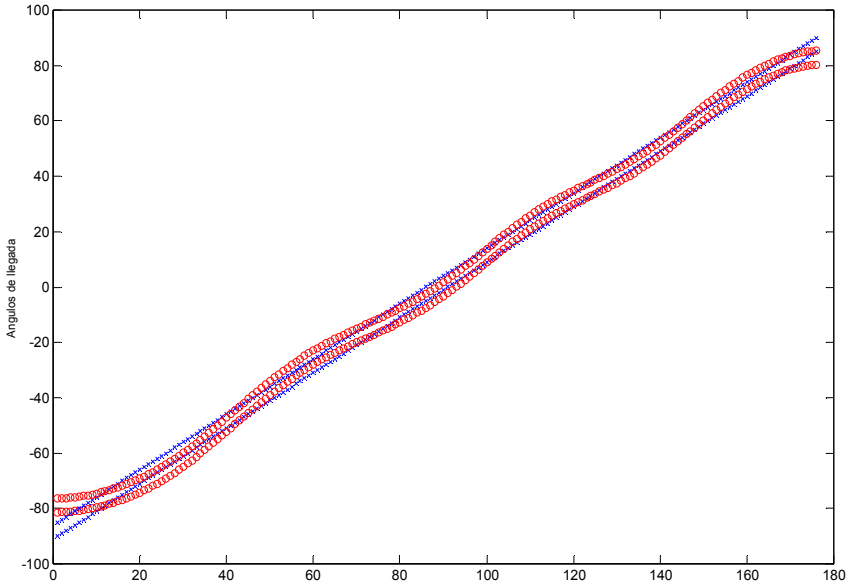


Figura 4.93. Salida de la red RBF(rojo) frente a la deseada.

Obteniendo: **MSE = 0.0035**

Ahora para 7 **neuronas intermedias y una separación espacial de 10º**:

Para una separación entre los elementos de la agrupación $d = \lambda/8$

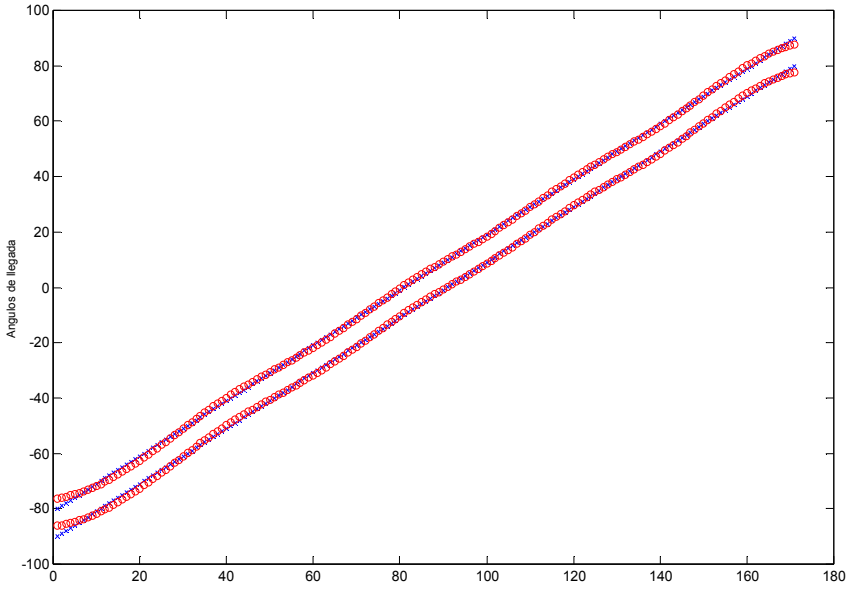


Figura 4.94. Salida de la red RBF(rojo) frente a la deseada.

Obteniendo: **MSE = 4.5999e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

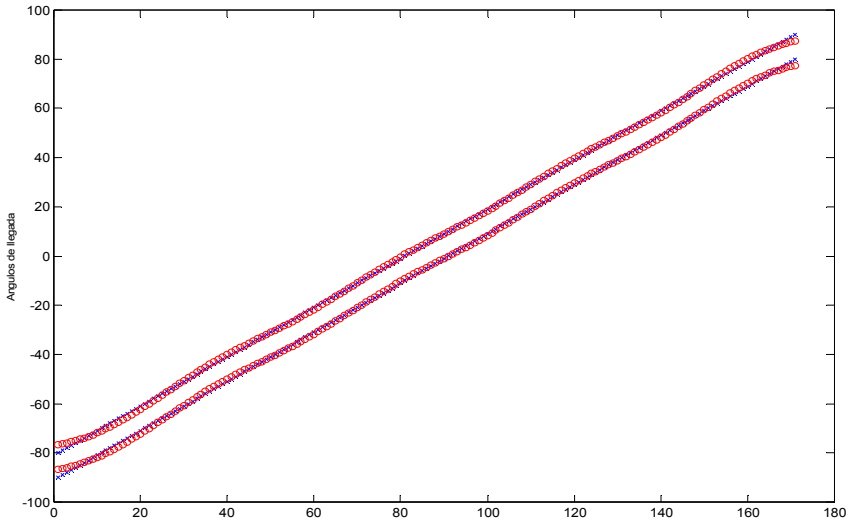


Figura 4.95. Salida de la red RBF (rojo) frente a la deseada.

Obteniendo: **MSE = 4.3130e-004**

Vemos como la red neuronal RBF con pocas neuronas le cuesta aproximar la función de salida y aumentando a 7 neuronas obtenemos una mejora sensible.

Ahora pasamos a la siguiente configuración:

- **6 elementos** en la agrupación de antenas, **dimensión de entrada 36**.
- **11 neuronas intermedias**.
- **10 º de separación espacial entre las dos señales de llegada.**

Para una separación entre los elementos de la agrupación $d = \lambda/8$

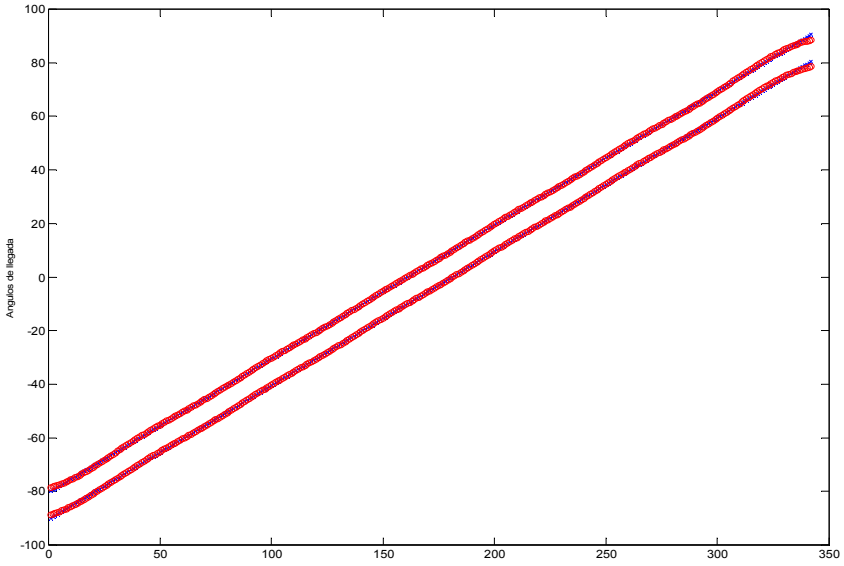


Figura 4.96. Salida de la red RBF(rojo) frente a la deseada.

Para una separación entre los elementos de la agrupación $d = \lambda/4$

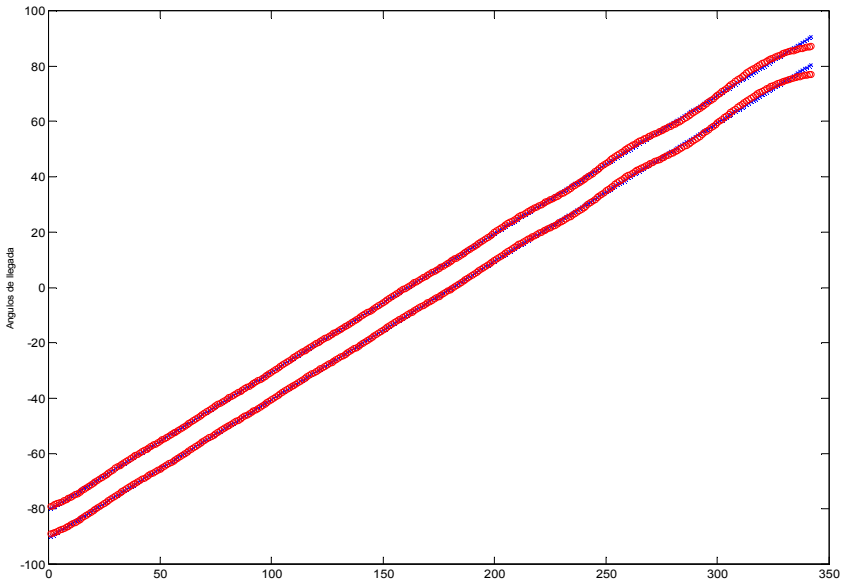


Figura 4.97. Salida de la red RBF(rojo) frente a la deseada.

Obteniendo: **MSE = 7.2076e-005** para una separación $d = \lambda/8$

Obteniendo: **MSE = 2.3647e-004** para una separación $d = \lambda/4$

Vemos que la red RBF cuando tenemos tres elementos en la agrupación, de forma que se genera una dimensión de entrada igual a 9, conforme aumentamos las neuronas intermedias va mejorando su comportamiento. Para 6 elementos de array, que genera una dimensión de entrada igual a 36, con 11 neuronas intermedias para el caso de una separación entre los elementos de la agrupación igual a $\lambda/8$ ya tenemos suficientes neuronas par una buena aproximación, en cambio no podemos decir lo mismo para una separación igual a $\lambda/4$.

- **Red PPL**

Para una señal de llegada tendremos los siguientes parámetros:

- **3 elementos** en la agrupación de antenas.
- Una **dimensión de entrada a la red neuronal igual a 9**.
- **5 neuronas intermedias**.

Las simulaciones que vamos a mostrar a continuación es para un valor R = 1, más adelante en las tablas de resultados veremos la salida para distintos valores del parámetro R.

Empezaremos con una separación espacial entre las dos señales de llegada de 5º.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

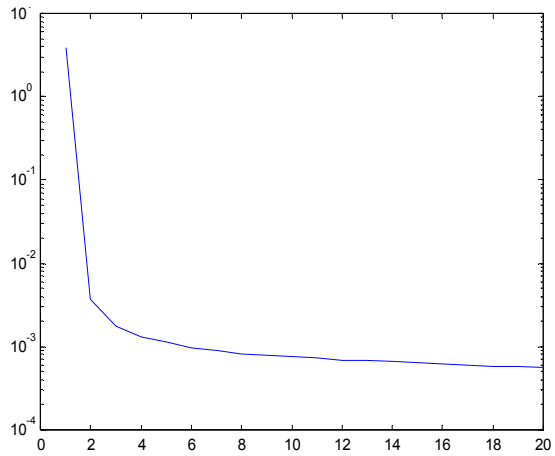


Figura 4.98. Evolución del error según las iteraciones

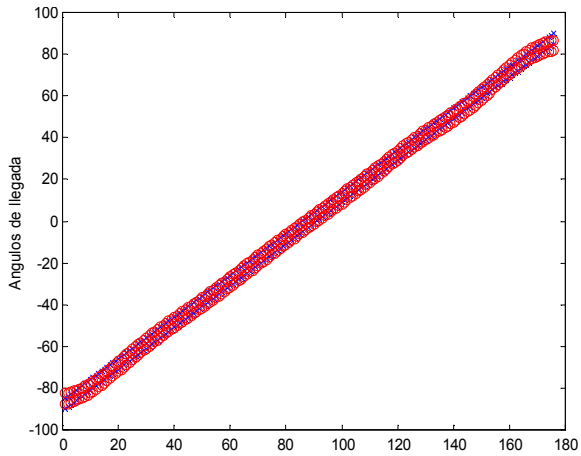


Figura 4.99. Salida de la red PPL (rojo) frente la deseada

Obteniendo: **MSE = 4.9066e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

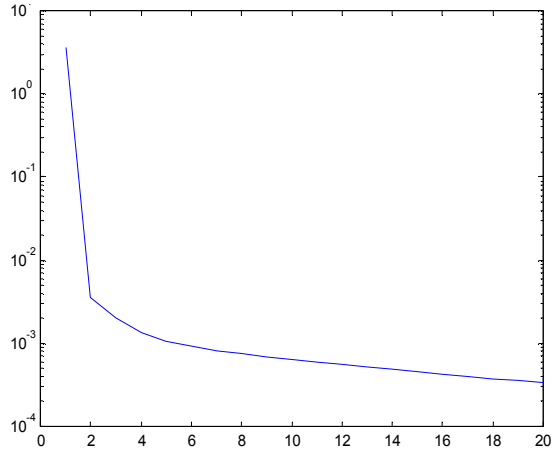


Figura 4.100. Evolución del error según las iteraciones

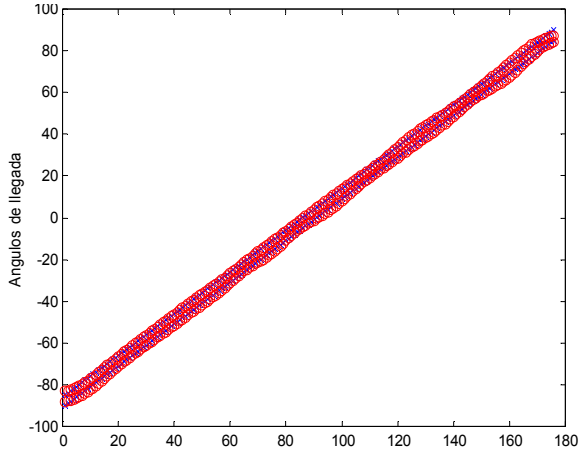


Figura 4.101. Salida de la red PPL (rojo) frente la deseada

Obteniendo: **MSE = 3.6437e-004**

Claramente obtenemos mejores resultados que la red RBF para pocas neuronas.

Ahora para 7 **neuronas intermedias y una separación espacial de 10º**:

Para una separación entre los elementos de la agrupación $d = \lambda/8$

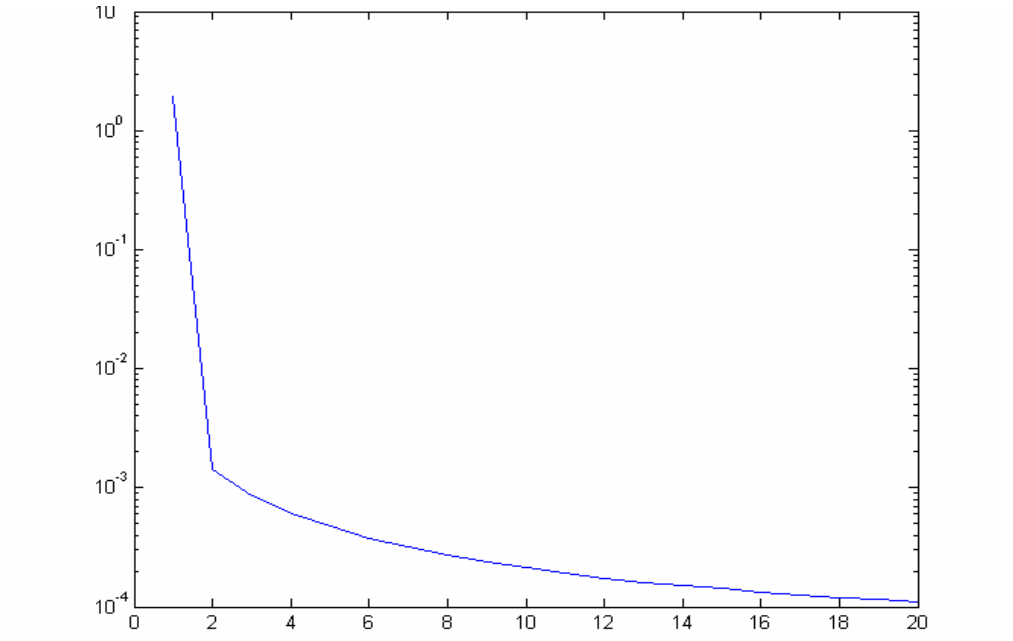


Figura 4.102.Evolución del error durante el entrenamiento según las iteraciones.

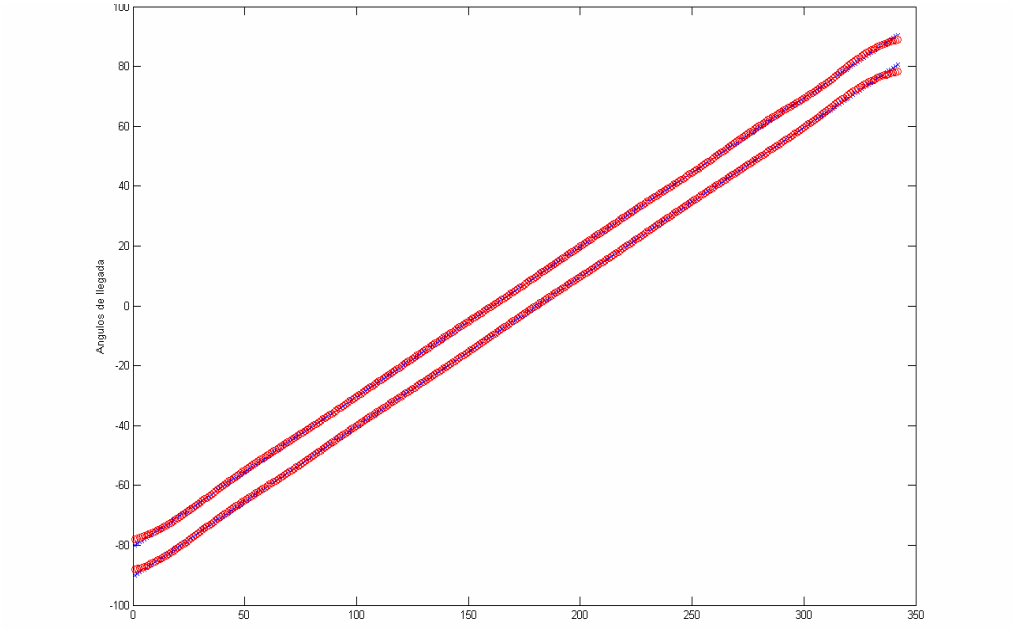


Figura 4.103. Salida de la red PPL (rojo) frente a la deseada.

Obteniendo: **MSE = 1.1680e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

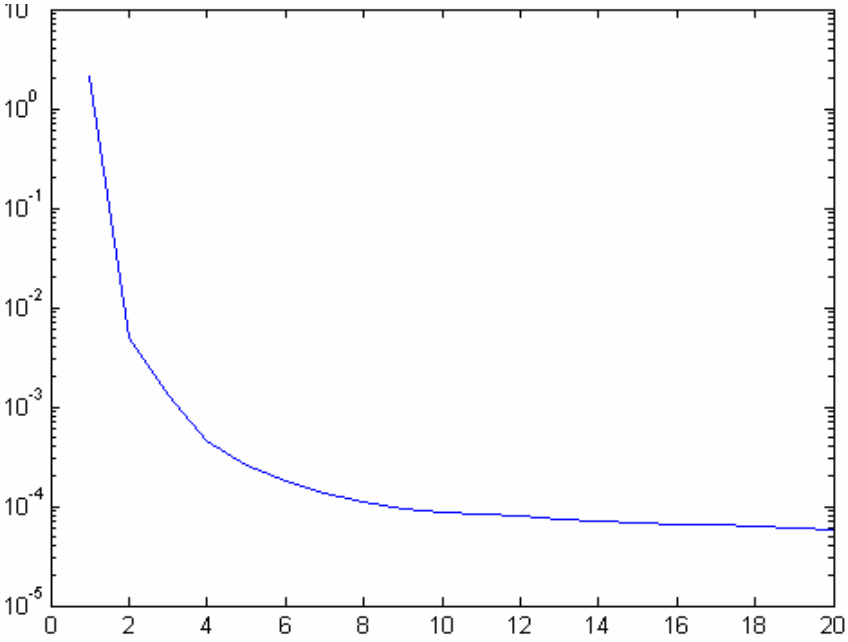


Figura 4.104. Evolución del error durante el entrenamiento según las iteraciones.

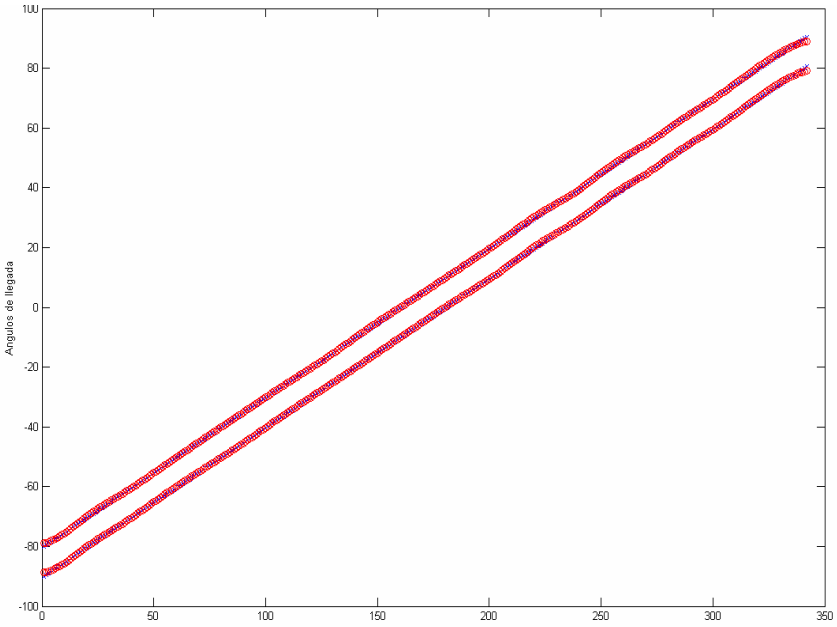


Figura 4.105. Salida de la red PPL (rojo) frente a la deseada.

Obteniendo**:** **MSE = 8.1874e-005**

Con 7 neuronas intermedias seguimos mejorando el resultado de la red PPL frente a la RBF.

Ahora pasamos a la siguiente configuración:

- **6 elementos** en la agrupación de antenas.
- Una **dimensión de entrada a la red neuronal igual a 36**.
- **11 neuronas intermedias**.

Empezaremos con una separación espacial entre las dos señales de llegada de 10º.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

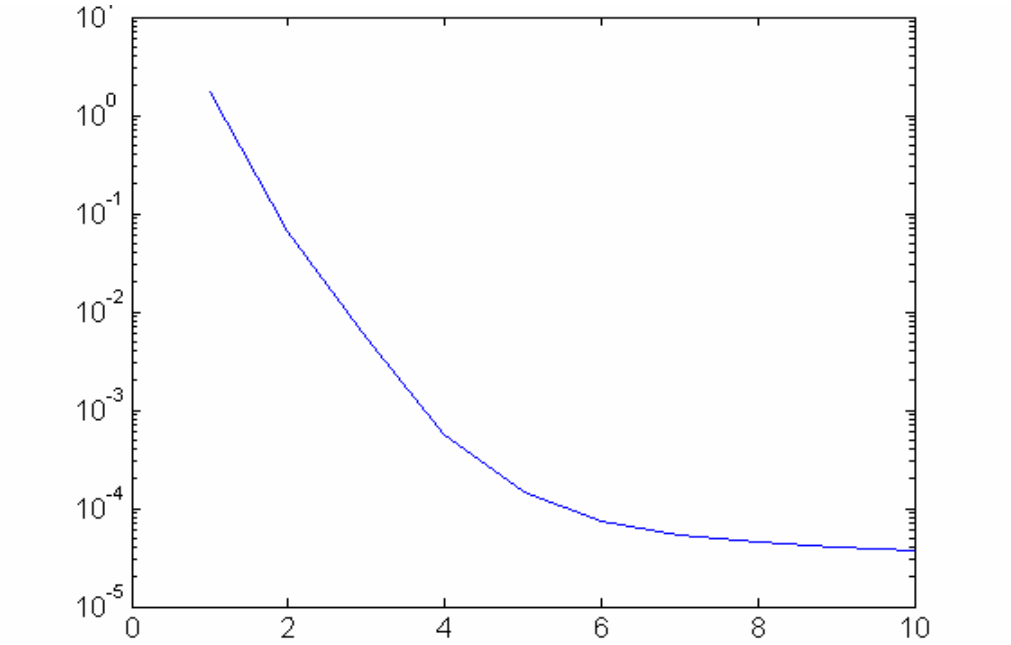


Figura 4.106. Evolución del error durante el entrenamiento según las iteraciones.

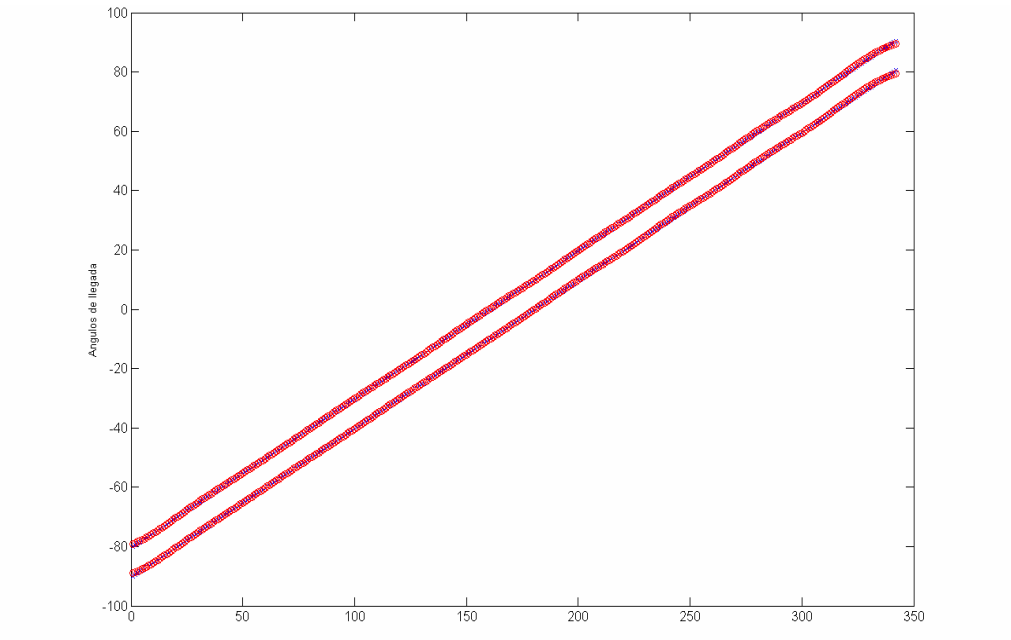


Figura 4.107. Salida de la red PPL (rojo) frente a la deseada.

Para una separación entre los elementos de la agrupación $d = \lambda/4$

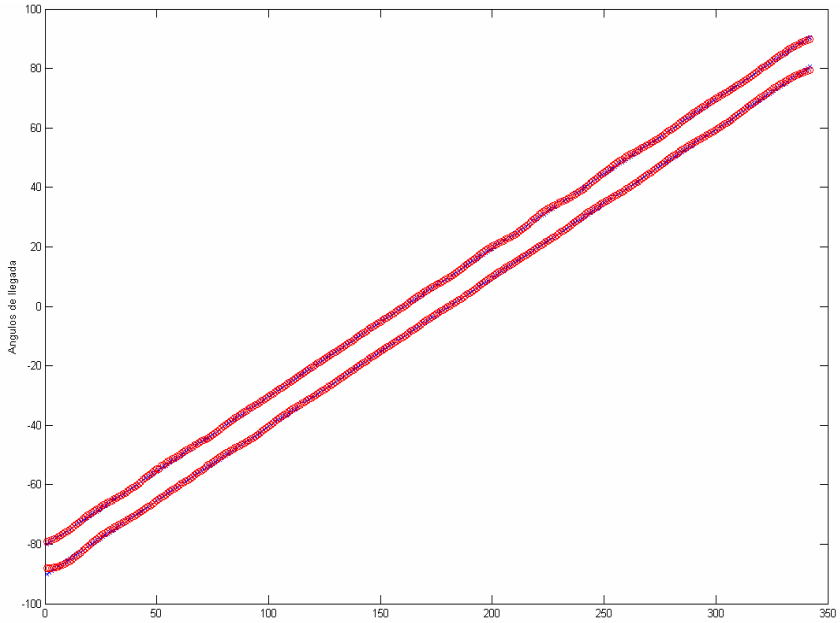


Figura 4.108. Salida de la red PPL (rojo) frente a la deseada

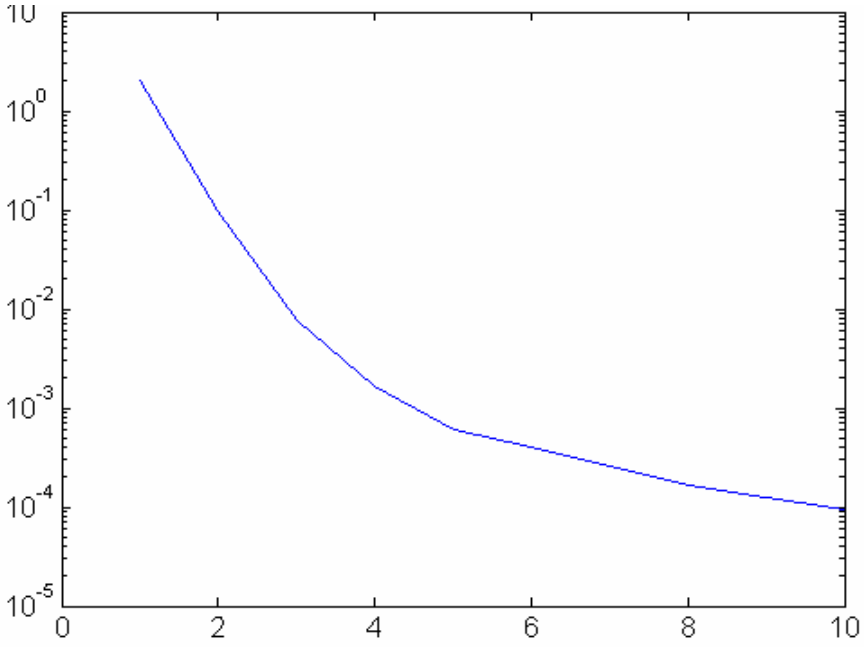


Figura 4.109. Evolución del error durante el entrenamiento según las iteraciones.

Obteniendo: **MSE = 6.3645e-005** para una separación $d = \lambda/8$

Obteniendo: **MSE = 5.4422e-005** para una separación $d = \lambda/4$

La red PPL mejora la aproximación para una separación de los elementos de la agrupación igual a $\lambda/4$ **.** Vamos a ver en el siguiente punto los resultados en tablas.

4.4.1 *TABLAS DE RESULTADOS*

Los resultados de las tablas es al media de realizar varias simulaciones y el parámetro de medida es el Error cuadrático medio (MSE).

- **RBF**

3 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 9

	$d = \lambda/8$	$d = \lambda/4$
Neuronas		
5	**MSE = 1.531e-003**	**MSE = 3.5953e-003**
7	**MSE = 4.5999e-004**	**MSE = 4.7130e-004**
9	**MSE = 2.3459e-004**	**MSE = 3.9183e-004**

Tabla 4.12. MSE para 3 antenas en al agrupación y diferentes distancias de separación

6 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 36

	$d = \lambda/8$	$d = \lambda/4$
Neuronas		
11	**MSE = 7.2076e-005**	**MSE = 2.3647e-004**
18	**MSE = 1.8151e-005**	**MSE = 1.1998e-005**

Tabla 4.13. MSE para 6 antenas en al agrupación y diferentes distancias de separación

- **PPL**

3 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 9

$d = \lambda/8$

	R = 1	**R = 5**	**R = 9**
Neuronas			
5	**MEDIO = 4.9758e-004**	**MEDIO = 1.4928e-003**	**MEDIO =5.8729e-004**

Tabla 4.14. MSE para 3 antenas en al agrupación y varios valores del parámetro R

$d = \lambda/4$

	R = 1	R = 5	R = 9
Neuronas			
5	MEDIO = 4.5032e-004	MEDIO =1.2950e-003	MEDIO =3.5e-003

Tabla 4.15. MSE para 3 antenas en al agrupación y varios valores del parámetro R.

A continuación mostramos los resultados para 7 neuronas intermedias y R = 1, ya que para este valor obtenemos los mejores resultados:

	$d = \lambda/8$	$d = \lambda/4$
Neuronas		
7	MSE = 1.3301e-004	MSE = 9.3166e-005

Tabla 4.16. MSE para 3 antenas en al agrupación.

6 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 36

	$d = \lambda/8$	$d = \lambda/4$
Neuronas		
11	MSE = 6.3645e-005	MSE = 9.8573e-005
18	MSE = 5.4422e-005	MSE = 4.5653e-005

Tabla 4.17. MSE para 3 antenas en al agrupación.

Para el caso de tener 3 elementos en la agrupación con menos neuronas la red PPL obtiene mejores resultados que la red RBF como muestran las *tablas 4.12 para el caso de la RBF y 4.16 para la PPL*. Cuando aumentamos a 6 elementos la aproximación de la señal se hace más fácil y esto nos permite obtener mejores resultados con ambas redes, ver *tablas 4.13 y 4.17*, como tenemos un número suficiente de elementos de array la RBF aunque necesite más neuronas que la dimensión de entrada, con 18 neuronas es suficiente para obtener un resultado similar que la PPL. **La red PPL con un número reducido de neuronas consigue una buena aproximación, suficiente para el problema tratado y mejor que la RBF**.

4.5 TRES SEÑALES DE LLEGADA

- **Red RBF**

Para tres señales de llegada tendremos los siguientes parámetros:

- **4 elementos** en la agrupación, **dimensión de entrada igual a 16**.
- **5 neuronas intermedias**.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

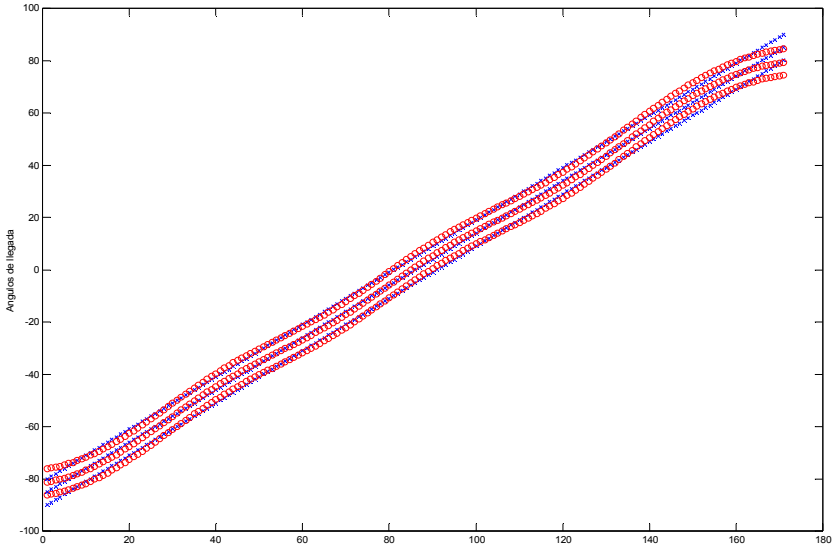


Figura 4.110. Salida de la red RBF (rojo) frente a la deseada

Obteniendo: **MSE = 0.0021**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

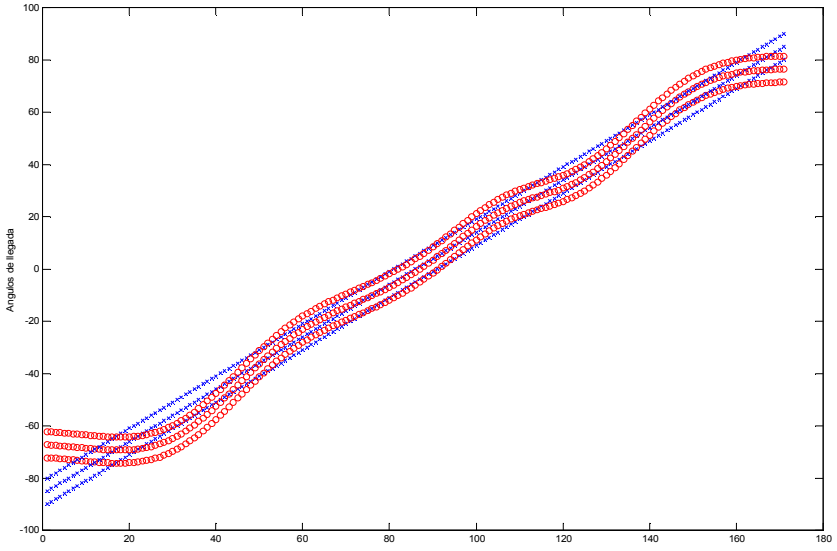


Figura 4.111. Salida de la red RBF (rojo) frente a la deseada

Obteniendo: **MSE = 0.0218**

Ahora para **9 neuronas intermedias**:

Para una separación entre los elementos de la agrupación $d = \lambda/8$

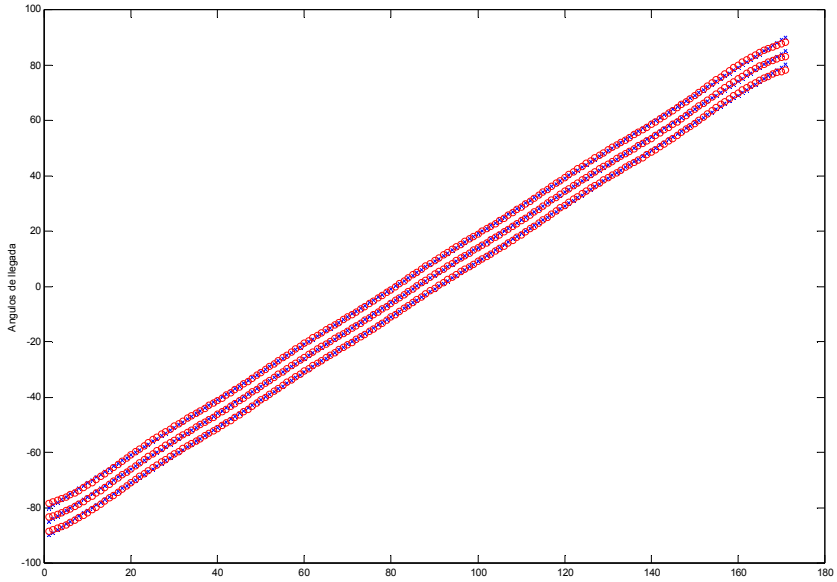


Figura 4.112. Salida de la red RBF (rojo) frente a la deseada

Obteniendo: **MSE = 1.7912e-004**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

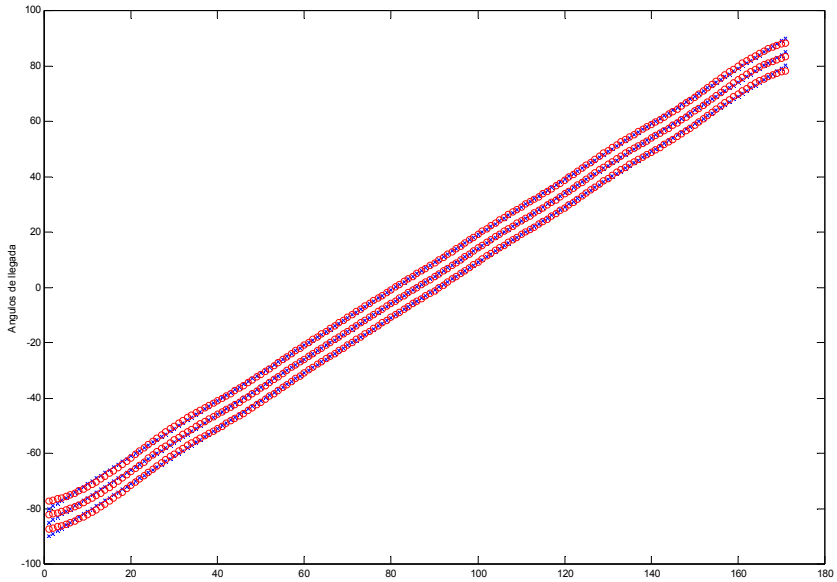


Figura 4.113. Salida de la red RBF (rojo) frente a la deseada

Obteniendo: **MSE = 3.0112e-004**

Para una nueva configuración:

- **6 elementos** en la agrupación, **dimensión de entrada igual a 36**.
- **16 neuronas intermedias**.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

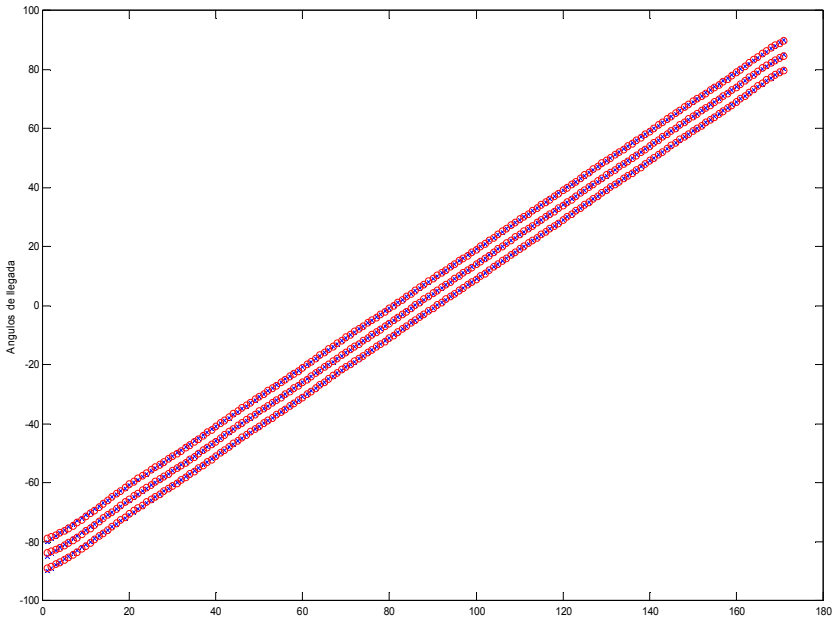


Figura 4.114. Salida de la red RBF (rojo) frente a la deseada

Para una separación entre los elementos de la agrupación $d = \lambda/4$

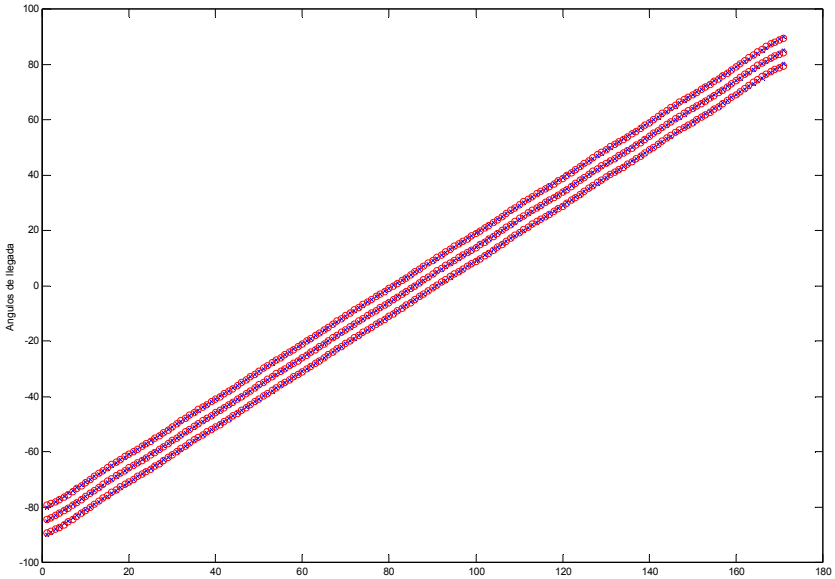


Figura 4.115. Salida de la red RBF (rojo) frente a la deseada

Obteniendo: **MSE = 3.5760e-005**para una separación $d = \lambda/8$

Obteniendo: **MSE = 3.4366e-005**para una separación $d = \lambda/4$

Con pocos elementos de array la detección de las señales es más difícil y le cuesta más a la RBF aproximar la salida deseada. Al aumentar el número de elementos de array se obtienen mejores resultados.

- **Red PPL**

Tenemos la siguiente configuración:

- **4 elementos** en la agrupación de antenas.
- Una **dimensión de entrada a la red neuronal igual a 16**.
- **5 neuronas intermedias**.

Los resultados mas satisfactorios para este caso se obtienen para R = 1 y las simulaciones que a continuación vamos a ver son algunas de las muchas realizadas para calcular el error cometido que al final veremos en unas tablas como en los casos anteriores.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

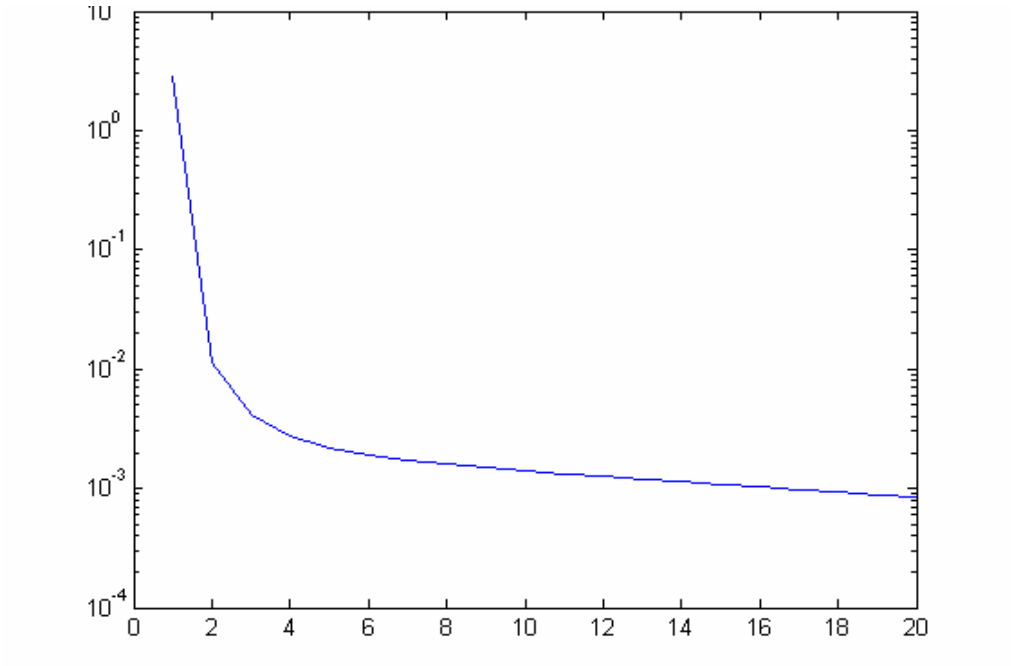


Figura 4.116. Evolución del error durante el entrenamiento según las iteraciones.

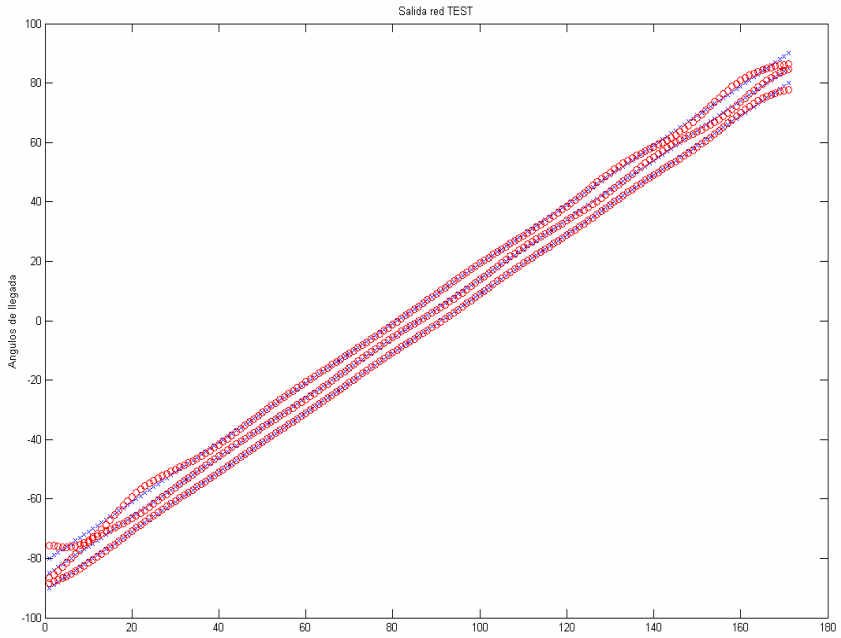


Figura 4.117. Salida de la red PPL (rojo) frente a la deseada

Obteniendo

MSE = 5.5281e-004

Para una separación entre los elementos de la agrupación $d = \lambda/4$

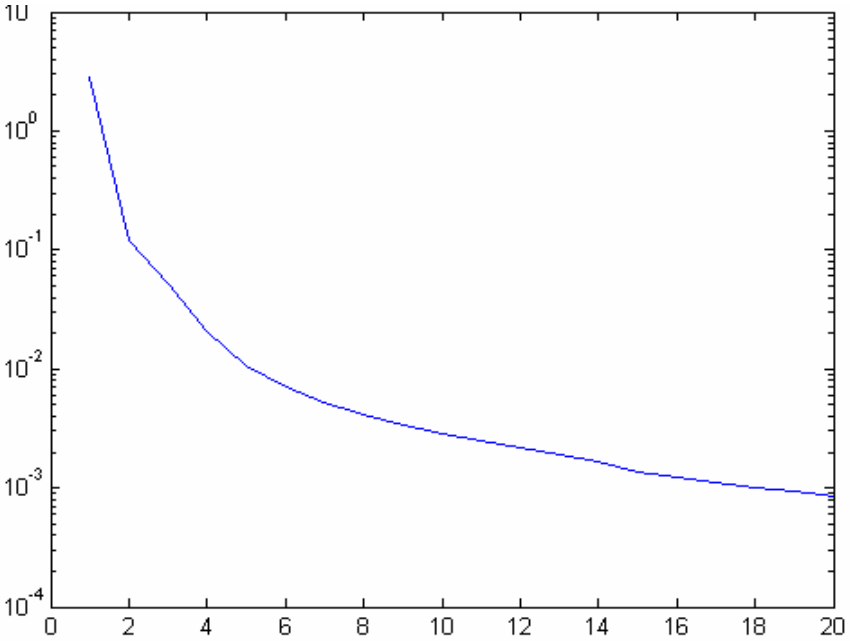


Figura 4.118. Evolución del error durante el entrenamiento según las iteraciones.

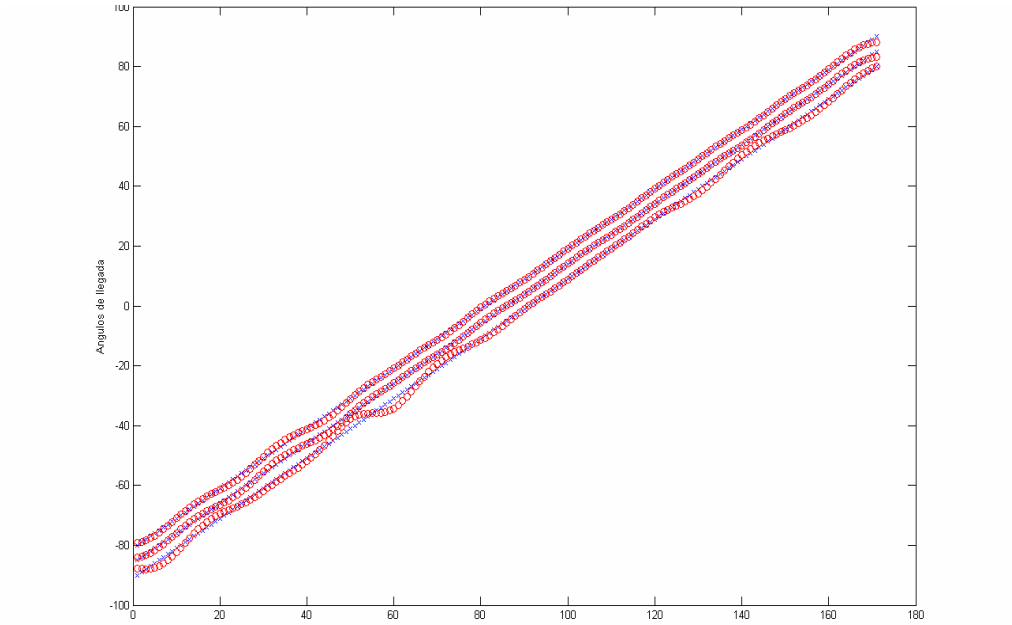


Figura 4.119. Salida de la red PPL (rojo) frente a la deseada

Obteniendo

MSE = 6.2106e-004

Ahora aumentamos las **neuronas intermedias a 9:**

Para una separación entre los elementos de la agrupación $d = \lambda/8$

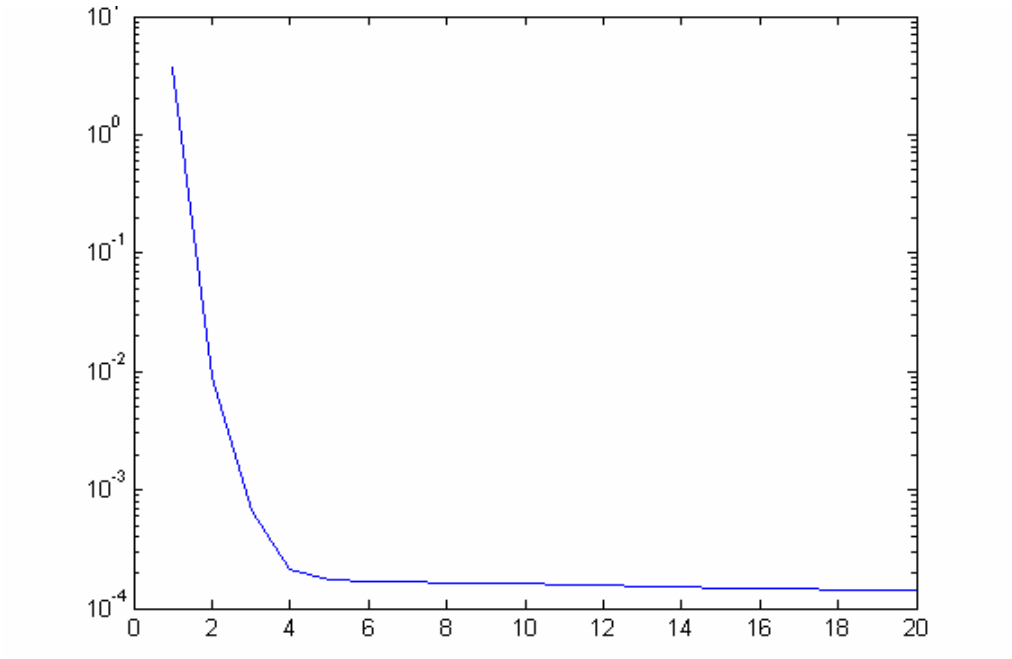


Figura 4.120. Evolución del error durante el entrenamiento según las iteraciones.

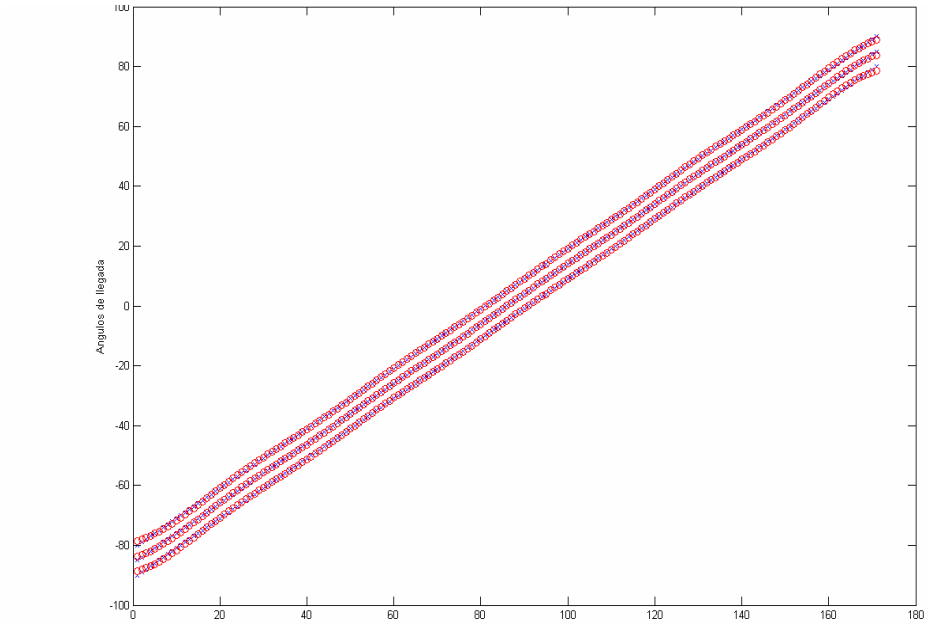


Figura 4.121. Salida de la red PPL (rojo) frente a la deseada

Obteniendo

MSE = 9.3703e-005

Para una separación entre los elementos de la agrupación $d = \lambda/4$

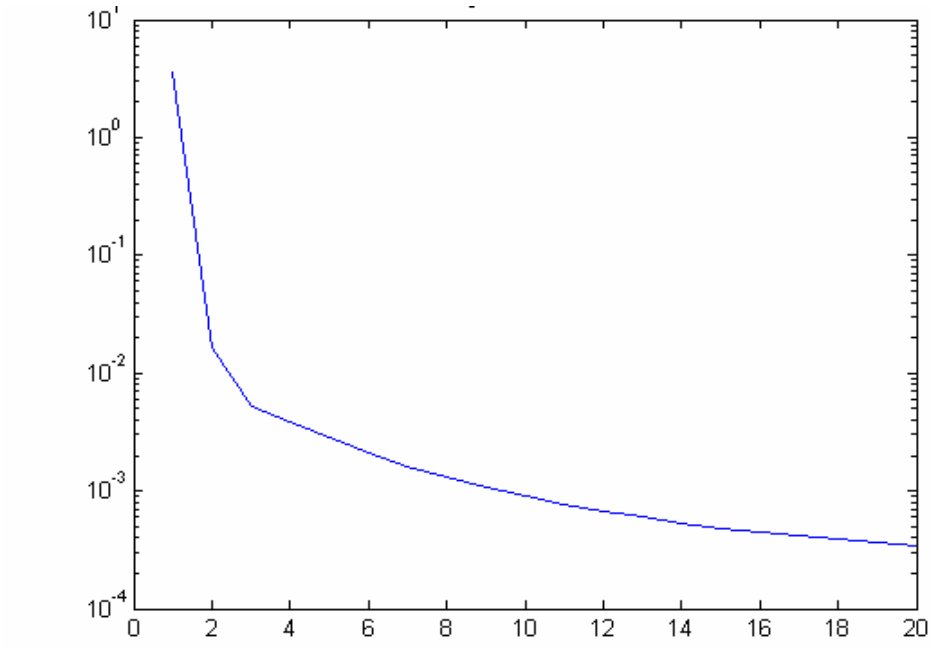


Figura 4.122. Evolución del error durante el entrenamiento según las iteraciones.

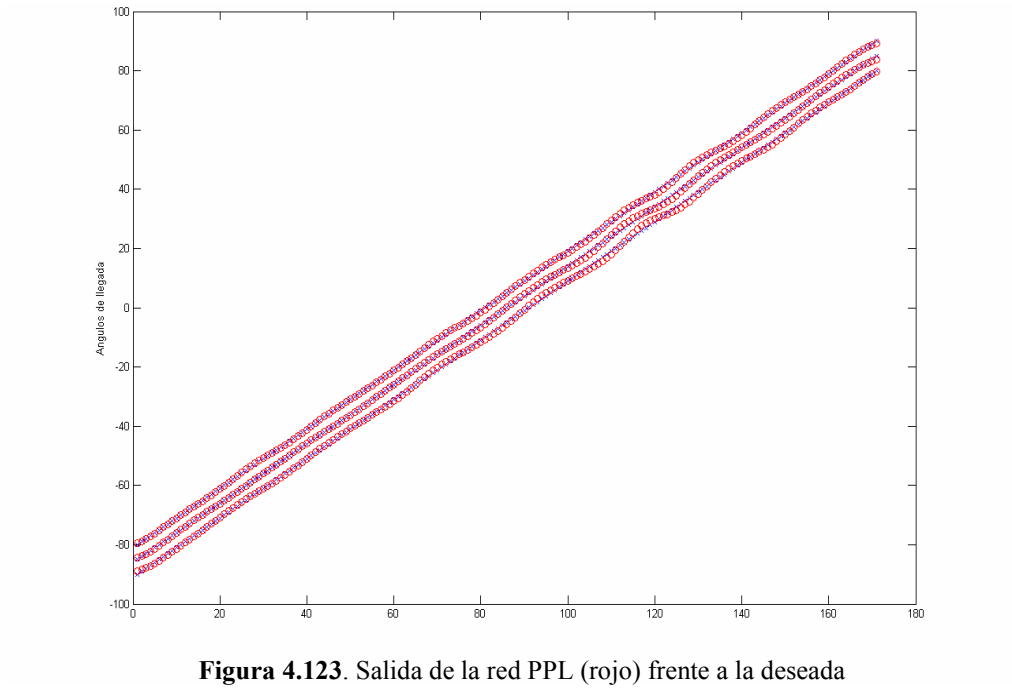


Figura 4.123. Salida de la red PPL (rojo) frente a la deseada

Obteniendo

MSE = 2.3934e-004

Ahora aumentamos las **neuronas intermedias a 11:**

Para una separación entre los elementos de la agrupación $d = \lambda/8$

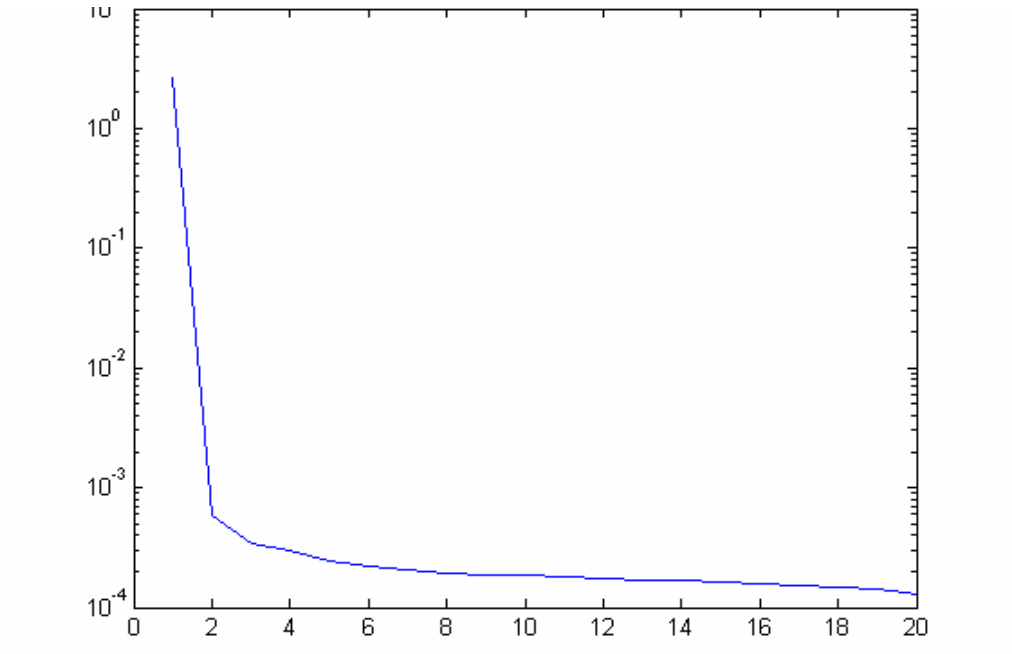


Figura 4.124. Evolución del error durante el entrenamiento según las iteraciones.

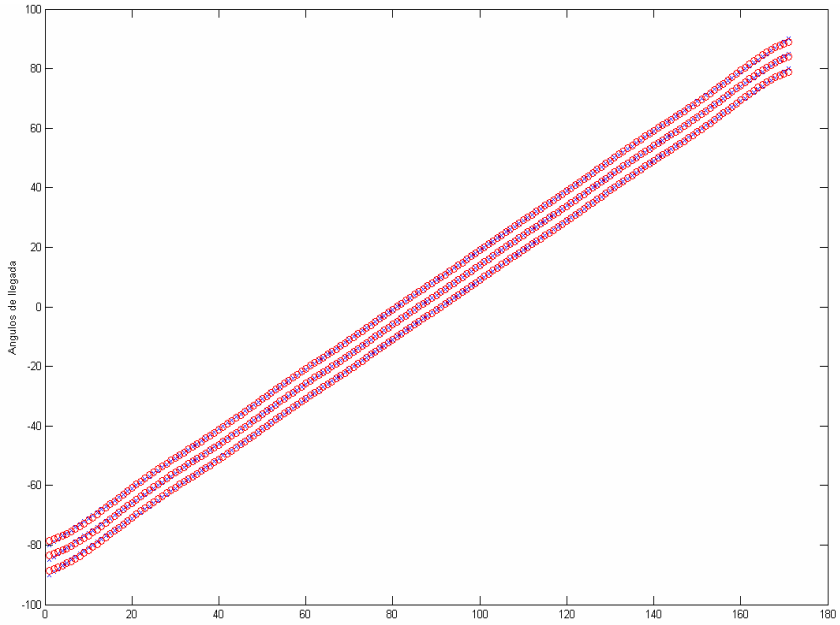


Figura 4.125. Salida de la red PPL (rojo) frente a la deseada

Obteniendo

MSE = 8.4443e-005

Para una separación entre los elementos de la agrupación $d = \lambda/4$

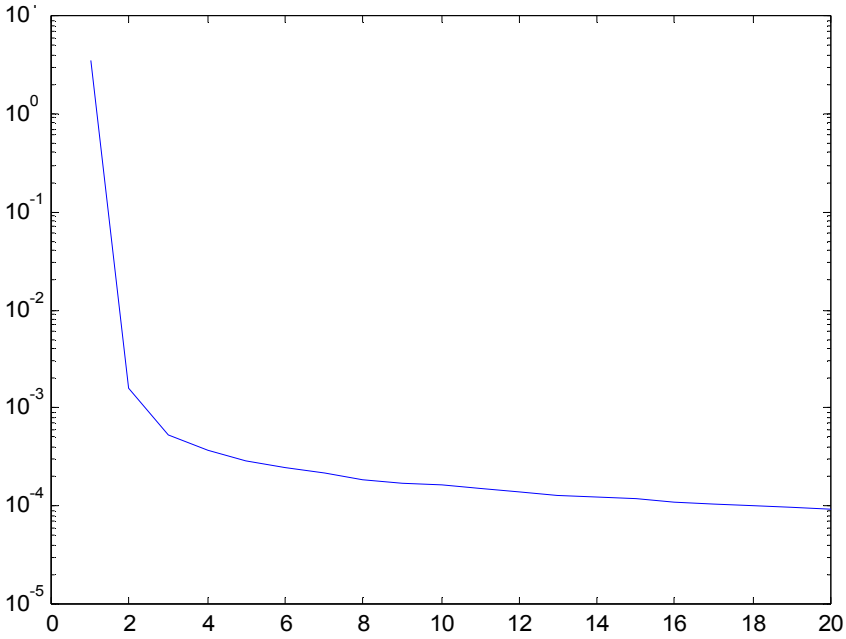


Figura 4.126. Evolución del error durante el entrenamiento según las iteraciones.

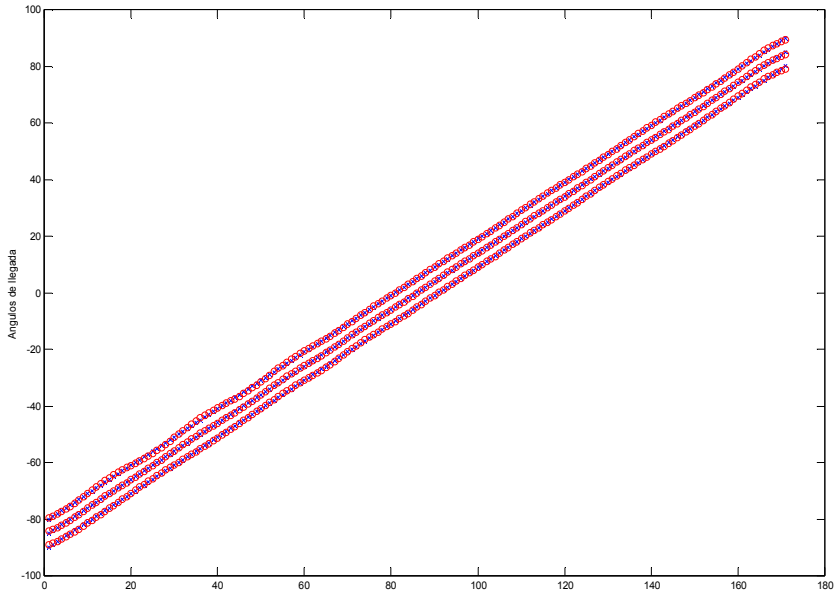


Figura 4.127. Salida de la red PPL (rojo) frente a la deseada

Obteniendo: **MSE = 7.1630e-005**

Para el caso de 4 elementos de array claramente con un menor número de neuronas la red PPL obtiene mejores resultados aproximando las señales de salida.

Ahora cambiamos la configuración aumentando el número de antenas de la agrupación:

- **6 elementos** en la agrupación de antenas, **dimensión de entrada 36**.
- **16 neuronas intermedias**.

Para una separación entre los elementos de la agrupación $d = \lambda/8$

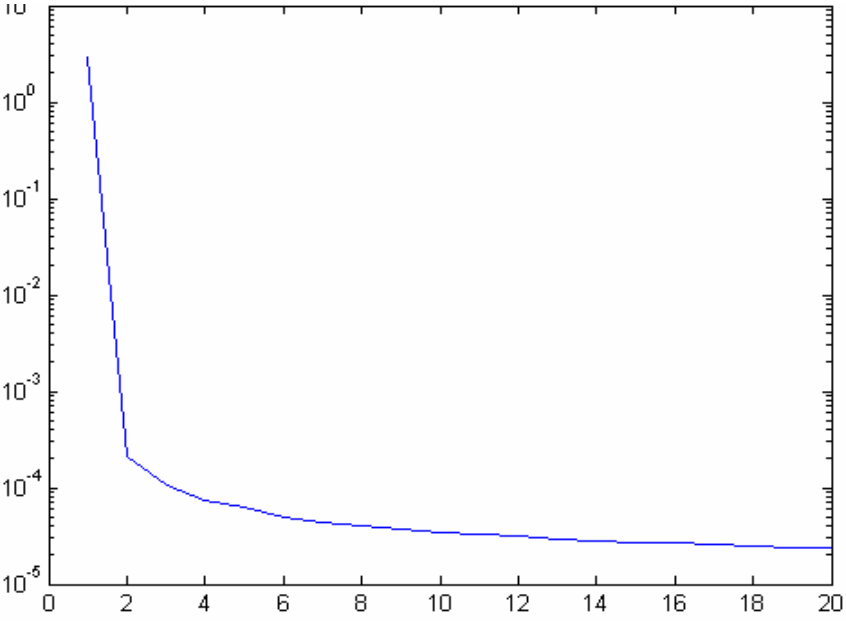


Figura 4.128. Evolución del error durante el entrenamiento según las iteraciones.

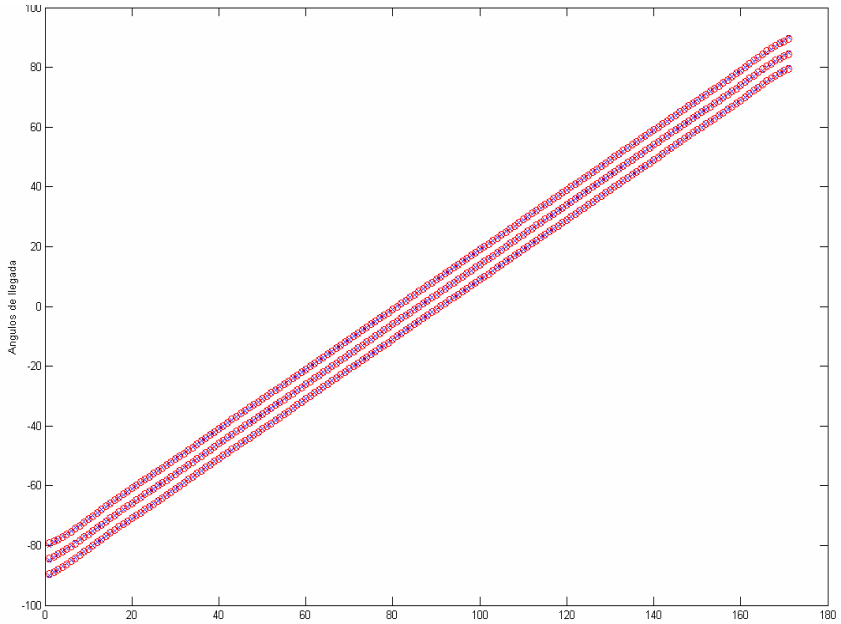


Figura 4.129. Salida de la red PPL (rojo) frente a la deseada

Obteniendo **MSE = 1.4661e-005**

Para una separación entre los elementos de la agrupación $d = \lambda/4$

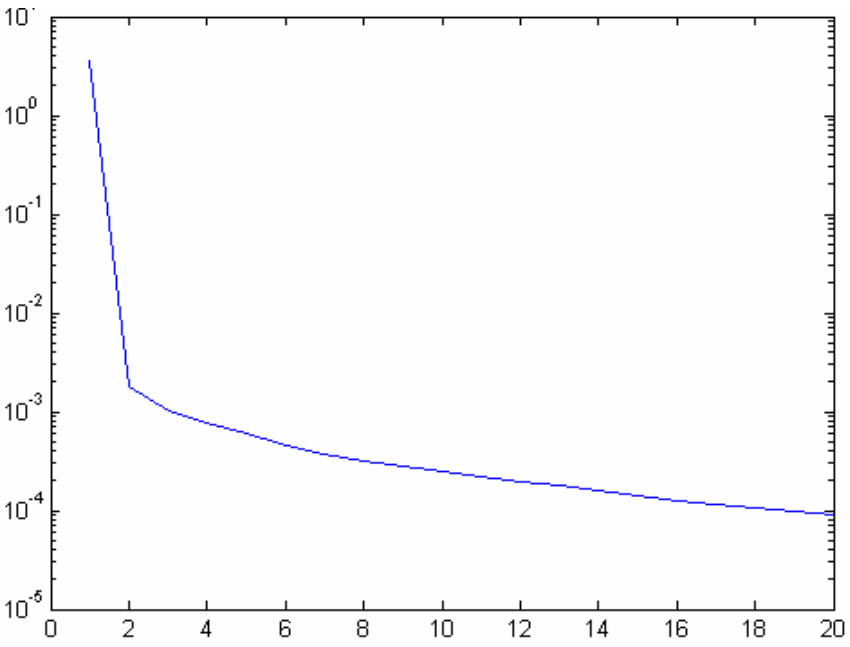


Figura 4.130. Evolución del error durante el entrenamiento según las iteraciones.

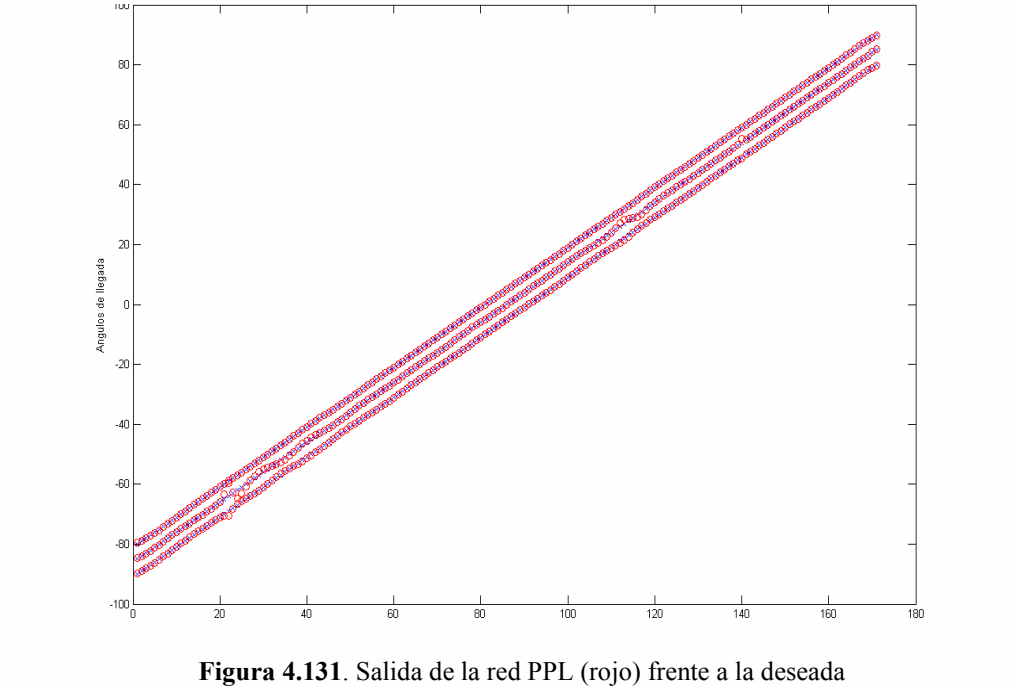


Figura 4.131. Salida de la red PPL (rojo) frente a la deseada

Obteniendo: **MSE = 1.0840e-004**

Los resultados obtenidos para el caso de 6 elementos en la agrupación de antenas los vamos a discutir en el apartado siguiente viendo las tablas.

4.5.1 *TABLAS*

Los resultados de las tablas es al media de realizar varias simulaciones y el parámetro de medida es el Error cuadrático medio (MSE).

- **RBF**

4 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 16

	$d = \lambda/8$	$d = \lambda/4$
Neuronas		
5	**MSE = 2.1e-003**	**MSE = 2.18e-002**
9	**MSE = 1.7912e-004**	**MSE = 3.0112e-004**
11	**MSE = 8.8196e-005**	**MSE = 1.3859e-004**

Tabla 4.18. MSE para 4 antenas en al agrupación y diferentes distancias de separación

6 ANTENAS EN LA AGRPUACIÓN
DIMENSI´0N DE ENTRADA 36

	$d = \lambda/8$	$d = \lambda/4$
Neuronas		
9	**MSE = 3.0876e-004**	**MSE = 7.5805e-004**
16	**MSE = 3.5760e-005**	**MSE = 3.4366e-005**

Tabla 4.19. MSE para 6 antenas en al agrupación y diferentes distancias de separación

- **PPL**

4 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 16

$d = \lambda/8$

	R = 1	**R = 5**	**R = 9**
Neuronas			
5	**MEDIO = 2.1123e-004**	**MEDIO = 7.2184e-004**	**MEDIO =2.1878e-004**

Tabla 4.20. MSE para 4 antenas en al agrupación

$d = \lambda/4$

	R = 1	**R = 5**	**R = 9**
Neuronas			
5	**MEDIO = 3.0917e-004**	**MEDIO = 4.7948e-003**	**MEDIO =7.4417e-004**

Tabla 4.21. MSE para 4 antenas en al agrupación

Ahora dejamos el grado de los polinomios de Hermite con R = 1 que es caso donde obtenemos mejores resultados.

	$d = \lambda/8$	$d = \lambda/4$
Neuronas		
9	**MEDIO = 9.1947e-005**	**MEDIO = 1.1112e-004**
11	**MEDIO = 7.4407e-005**	**MEDIO = 8.4883e-005**
16	**MEDIO = 1.0634e-004**	**MEDIO = 7.9811e-005**

Tabla 4.22. MSE para 4 antenas en al agrupación

6 ANTENAS EN LA AGRPUACIÓN
DIMENSIÓN DE ENTRADA 36

	$d = \lambda/8$	$d = \lambda/4$
Neuronas		
9	**MEDIO = 8.75e-005**	**MEDIO = 1.6671e-004**
16	**MEDIO = 3.2257e-005**	**MEDIO =6.39e-005**

Tabla 4.23. MSE para 4 antenas en al agrupación

Como podemos observar en las *tabla 4.18* para la RBF y la *tabla 4.22* para la PPL con un número pequeño de neuronas intermedias (5 ó 9) la red neuronal PPL aproxima mejor las señales de salida. Por otro lado como ya comentamos en el caso de dos señales de llegada, al aumentar el array de antenas es más fácil la aproximación y obtenemos un menor error en el cálculo. En las *tablas 4.19 y 4.23* observamos la reducción de este error y como ambas redes obtienen resultados similares con el mismo número de neuronas. Así, podemos concluir que la red PPL consigue con un número menor de neuronas una buena aproximación.

4.6 SIMULACIÓN EN PRESENCIA DE RUIDO

En este apartado vamos a probar el funcionamiento de la red neuronal en presencia de ruido blanco. Como vimos en el Capítulo 1 donde explicábamos la formación de las señales que inciden el ruido es introducido en la ecuación (5):

$$\mathbf{X}(t) = \mathbf{A}\mathbf{S}(t) + \mathbf{N}(t)$$

De esta forma recordamos que el ruido aparecerá en la matriz de correlación:

$$R = E\left\{\mathbf{X}(t)\mathbf{X}(t)^H\right\} = \mathbf{A}E\left[\mathbf{S}(t)\mathbf{S}^H(t)\right]\mathbf{A}^H + E\left[\mathbf{N}(t)\mathbf{N}^H(t)\right]$$

Las simulaciones de la red para la detección de ángulos de llegada la vamos a realizar para una SNR (relación señal - ruido) de 10 dB, 20 dB y 30 dB en le caso de tener una y dos señales de llegada.

4.6.1 *UNA SEÑAL DE LLEGADA A LA AGRUPACIÓN*

A continuación vamos a mostrar los resultados obtenidos con la red programada PPL y la RBF para el caso concreto de tener una separación entre los elementos del array de $\lambda/4$. Calcularemos el MSE obtenido por ambas redes para diferentes relaciones señal – ruido (SNR). A continuación vamos a ver las tablas con los resultados obtenidos:

PPL

	SNR = 10 dB	SNR = 20dB	SNR = 30dB
Elementos de array			
3	-15.3309 dB	-23.9005 dB	-28.2359 dB
5	-19.1092 dB	-26.3832 dB	-31.6732 dB

Tabla 4.24. MSE para la red PPL para diferentes relaciones de SNR

RBF

	SNR = 10 dB	SNR = 20dB	SNR = 30dB
Elementos de array			
3	-14.4793 dB	-20.7394 dB	-24.3471 dB
5	-20.1976 dB	-26.9839 dB	-32.0891 dB

Tabla 4.25. MSE para la red RBF para diferentes relaciones de SNR

Si expresamos los resultados en forma de gráfica:

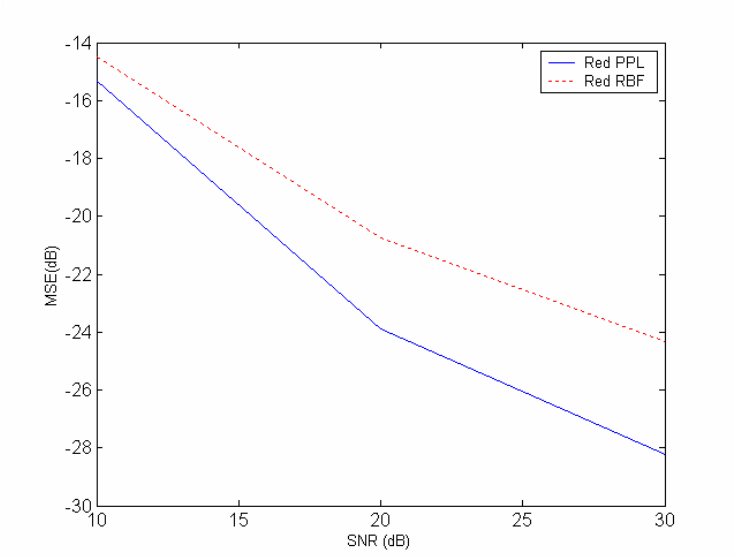


Figura 4.132. MSE para la red PPL y RBF en caso
De tener 3 elementos en el array

Para este caso de 3 elementos en la agrupación hemos utilizado 5 neuronas intermedias para ambas redes. Aumentando el número de neuronas a 8 para el caso de la RBF lograríamos una mejora de -1 dB, siendo todavía mejor la respuesta de la red PPL con menos neuronas que la RBF.

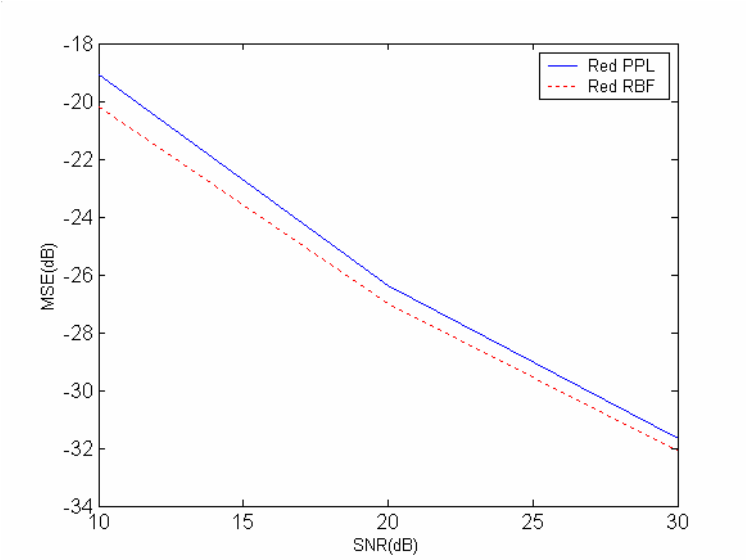


Figura 4.133. MSE para la red PPL y RBF en caso
De tener 5 elementos en el array

Para el caso de tener 5 elementos en la agrupación hemos utilizado 15 neuronas intermedias en ambas redes. La diferencia para tres elementos en la agrupación es mayor ya que con menos elementos es más difícil aproximar la señal. En cambio para cinco elementos en la agrupación y solamente una señal de llegada es mucho más fácil la aproximación de la señal de salida y la respuesta de ambas redes se acerca, y la RBF parece que consigue para este ejemplo menor error siendo la diferencia muy poca.

4.6.2 *DOS SEÑALES DE LLEGADA A LA AGRUPACIÓN*

PPL

	SNR = 10 dB	SNR = 20dB	SNR = 30dB
Elementos de array			
3	-13.7932 dB	-23.0608 dB	-29.0576 dB
5	-17.3251 dB	-27.1178 dB	-32.7147 dB

Tabla 4.26. MSE para 4 antenas en al agrupación

RBF

	SNR = 10 dB	SNR = 20dB	SNR = 30dB
Elementos de array			
3	-13.0955 dB	-19.7317 dB	-22.3144 dB
5	-17.4538 dB	-26.2705 dB	-31.8023 dB

Tabla 4.27. MSE para 4 antenas en al agrupación

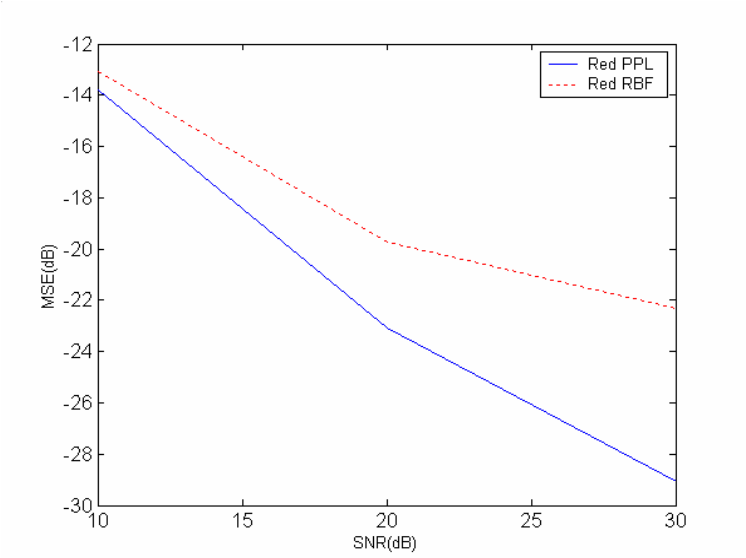


Figura 4.134. MSE para la red PPL y RBF en caso
De tener 3 elementos en el array

El caso de tener 3 elementos en la agrupación nos proporciona una dimensión de entrada en ambas redes neuronales igual a nueve. Los resultados han sido calculados para 5 neuronas intermedias, sin embargo hemos aumentado el número hasta el doble de la dimensión de entrada para la red RBF, es decir, 18 neuronas intermedias. Obteniendo una pequeña mejora:

SNR = 10 dB	SNR = 20dB	SNR = 30dB
-13.2453 dB	**-22.3993 dB**	**-28.9344 dB**

Ahora vemos el caso de tener 5 elementos de array:

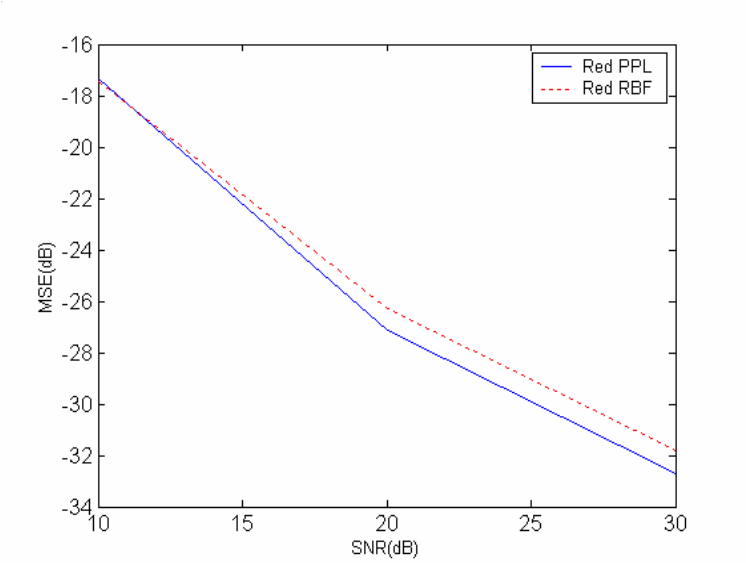


Figura 4.135. MSE para la red PPL y RBF en caso
De tener 5 elementos en el array

Ahora disponemos de 5 elementos de array por lo que tenemos una dimensión de entrada de 25 para ambas redes. Los resultados han sido calculados para un número de 15 neuronas intermedias obteniendo igual que para una señal que la red PPL obtiene mejores resultados que la RBF en presencia de ruido. Los valores del MSE obtenidos son un poco peores ya que estamos comparando dos señales para el mimo número de elementos de array que una señal y es más difícil aproximar dos señales que una.

Igual que en el caso anterior hemos aumentado el número de neuronas para la red RBF hasta 25 donde ya se estabilizan los resultados obteniendo una mejora en su comportamiento. Para 25 neuronas intermedias obtenemos para la red RBF:

SNR = 10 dB	SNR = 20dB	SNR = 30dB
-18.4125dB	**-27.7871 dB**	**-33.4917 dB**

CAPITULO 5: CONCLUSIONES

En este proyecto hemos implementado la red neuronal Projection Pursuit Learning (PPL) para la detección de ángulos de llegada de las señales que inciden en una agrupación o array de antenas. La red ha sido programada en Matlab 6.5, y hemos comparado los resultados obtenidos con una red neuronal RBF ya implementada en la librería "toolbox" de Matlab 6.5. A continuación vamos a comentar las conclusiones obtenidas sobre el comportamiento de la red y sus parámetros, descritos en el capítulo 3, y de los resultados obtenidos al aplicar la red neuronal PPL para la resolución del problema de detección de ángulos de llegada:

Las conclusiones obtenidas son:

- Tanto el parámetro R (orden de los polinomios de hermite) y el número de neuronas intermedias afectan al problema de generalización de nuestra red neuronal PPL. Para encontrar la solución óptima es necesario realizar varias simulaciones combinando estos dos parámetros, esta tarea requiere un coste computacional y de tiempo muy alto**. Para evitar la excesiva dependencia de la capacidad de la red PPL respecto del parámetro R introducimos un nodo bias en la estructura de la red.**

- En las simulaciones realizadas con pocos elementos en la agrupación, la red neuronal PPL se comporta mejor que la RBF obteniendo mejores resultados para un menor número de neuronas intermedias. Al aumentar los elementos de la agrupación seguimos consiguiendo un mejor resultado con la red PPL para pocas neuronas. Solo cuando seguimos aumentando el número de neuronas intermedias el resultado obtenido con la red RBF se aproxima al de la red PPL para el mismo número de neuronas. **Luego concluimos que la red neuronal PPL con un número reducido de neuronas consigue una buena aproximación, suficiente para el problema tratado y mejor que la RBF**.

- Cuando la red neuronal PPL está entrenada los parámetros de la red neuronal permanecen inalterados. **Una red formada por menos neuronas tendrá un comportamiento más rápido**, proporcionando en un menor tiempo los valores de salida. De forma que cuando se utilice en un sistema real superará el comportamiento de la RBF en este aspecto.

Las mejoras futuras que podemos realizar:

- Programar la red usando un lenguaje más eficiente para las redes neuronales como: Fortran ó VHDL.

- Dividir el espacio de llegada en sectores y utilizar una red para cada sector. De esta forma conseguiríamos un mejor cálculo de las señales de llegada. Dicha

división del espacio de entrada permite el cálculo de las direcciones de llegada un mayor número de señales a la agrupación de antenas. Una sola red neuronal no podría calcular la dirección de llegada de todas las señales, en cambio una serie de redes neuronales especializadas por sectores permiten el cálculo de la dirección de las señales en cada uno de los particulares sectores.

BIBLIOGRAFIA

[1] Sistemas de Comunicación, Simon Haykin, John Wiley & Sons, Limusa, 2002.

[2] Adaptive Array Measurements in Communications, M. A. Halim, Arthec House, 2001.

[3] Principles of radar and sonar signal processing, François Le Chevalier, Artech House, Norwood, MA, USA, 2002.

[4] Self-organized formation of topologically correct feature maps, T Kohonen, Biological Cynernetics, vol. 43, no 1, 59-69, 1982.

[5] Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps, G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, D. B. Rosen, IEEE transactions on neural networks, vol. 3, no. 5, septiembre, 1992.

[6] Neural Networks, a comprehensive foundation, S. Haykin, PrenticeHall International, New Jersey, 1999.

[7] Regresión Modeling in Back – Propagation and Projection Pursuit Learning. Jenq-Neng Hwang, Shyh-Rong Lay, Martin Maechler, Doug Martin, Jim Schimert.

[8] Use of Bias Term in Projection Pursuit Learning Improves Approximation and Convergence Properties. Tin-Yau Kwok and Dit-Yan Yeung, IEEE Transactions on Neural Networks, vol.7, no. 5, septiembre 1996.

[9] Matlab Neural network toolbox, version , The Math Works, MA, USA, año.

[10] Radial Basis Function Neural Network fot Direction-of-Arrivals Estimation. Titus Lo, Henry Leung and John Litva, IEEE Signal Processing Letters, vol.1, no. 2, febrero 1994.

[11] Performance of Radial-Basis Function Networks for Direction of Arrival Estimation With Antenna Arrays, Ahmed H. El Zooghby, Christos G. Christodoulou and Michael Georgiopoulos, IEEE Transactions on Antennas and Propagation, vol.45, no. 11, noviembre 1997.

[12] A Neural Network-Based Smart Antenna for Multiple Source Tracking. Ahmed H. El Zooghby, Christos G. Christodoulou and Michael Georgiopoulos, IEEE Transactions on Antennas and Propagation, vol.48, no. 5, mayo 2000.

[13] Applications of Neural Networks in Electromagnetics. Christos Christodoulou and Michael Georgiopoulos.