Universidad de Alcalá
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

APLICACIÓN DE ALGORITMOS SSA A LA PREDICCIÓN DE DEMANDA EN CENTROS DE TRANSFORMACIÓN

Autor: Juan Carlos Barbero del Olmo

Tutor/es: Francisco Javier Rodríguez Sánchez

2017

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

APLICACIÓN DE ALGORITMOS SSA A LA PREDICCIÓN DE DEMANDA EN CENTROS DE TRANSFORMACIÓN

Autor: Juan Carlos Barbero del Olmo

Tutor: Francisco Javier Rodríguez Sánchez

TRIBUNAL:

Presidente: Pedro Martín Sánchez

Vocal 1º: J. Antonio Jiménez Calvo

Vocal 2º: Fco. Javier Rodríguez Sánchez

FECHA: 14/07/2017

"No creo que haya nada más excitante para el corazón humano que la emoción sentida por el inventor cuando ve una creación de su mente convirtiéndose en algo exitoso. Estas emociones son las que un hombre no olvida como sus sueños, amigos, amor, todo. "

Nikola Tesla

A Javier, mi tutor,

Ya que, sin su paciencia y ayuda,

esto no habría sido posible.

A mis padres, mi hermana y Rocío,

Por el apoyo incondicional a mi trabajo

Índice

Lista de Figuras

Índice de Tablas

1. RESUMEN

Este TFG pretende explorar una solución a la problemática existente en los sistemas de distribución de energía eléctrica, que por motivos de planificación, ajuste de oferta-demanda, etc., necesitan de una predicción del consumo.

Esta predicción se puede realizar a largo, medio o corto plazo. En este TFG se trata el problema de la predicción de corto plazo, por lo que se realiza con una antelación de 24 horas de horizonte, con el objeto de ajustar la oferta y demanda de energía.

Para realizar la predicción se propone un sistema mixto que aúna dos herramientas: el Análisis de Espectro Singular y Redes Neuronales.

2. SUMMARY

This TFG aims to explore a solution to an existing problem in the distribution systems of electric energy, which for reasons of planning, supply-demand adjustment, etc., need a prediction of consumption.

This prediction can be made long, medium or short term. This TFG treats the problem of short-term prediction, so it is done with a 24-hour horizon ahead, in order to adjust supply and demand for energy.

To perform the prediction, we propose a mixed system that combines two tools: Singular Spectrum Analysis(SSA) and Artificial Neural Networks(ANN).

3. KEY WORDS

Singular Spectrum Analysis

Neural Network

Prediction

Smart Grid

4. RESUMEN EXTENDIDO

La estimación de carga en el sistema eléctrico es el procedimiento de predecir la carga que se va a producir en el sistema, en diversos puntos de la red, antes de que ocurra.

Este procedimiento es utilizado por las compañías eléctricas, entre otras cosas, para equilibrar la demanda y la oferta de suministro eléctrico con el fin de no tener sobrecargas o caídas en el sistema.

También se utilizan estos métodos para dimensionar correctamente la infraestructura de la red (Transformadores, subestaciones, líneas, etc.), además de para planificar correctamente paradas para mantenimiento, tareas de operación, etc.

Por otro lado, con el proceso de desagregación y la apertura del sector energético a diferentes proveedores y operadores de red, la previsión de la carga se está convirtiendo rápidamente en importante para todas las partes involucradas: proveedores de energía, operadores de red, comerciantes y corredores, ya que, a pesar de ser responsables de diferentes aspectos de la red, necesitan de una buena previsión de carga para ejercer con eficacia su rol en el suministro de energía al usuario final.

En general, los datos utilizados para predicciones son datos históricos de la carga, pero varias técnicas de predicción también hacen uso de otra información para ayudar a pronosticar y predecir la carga con mayor precisión, tales como datos meteorológicos históricos y previstos o factores sociales relevantes.

Debido a la naturaleza del problema de pronóstico, la predicción de la carga se clasifica normalmente como problema de predicción de corto, mediano y largo plazo.

El pronóstico a corto plazo se utiliza cuando se requiere información sobre la carga esperada en las próximas dos horas hasta un período de un día, mientras que una predicción a mediano plazo trata de predecir la carga durante unos días y hasta un período de pocas semanas.

La planificación a corto plazo se utiliza para ayudar a la asignación de las centrales que deben entrar en funcionamiento en caso necesario, mientras que la previsión a mediano plazo se puede utilizar para planificar los picos estacionales, así como la programación de mantenimiento.

El pronóstico a largo plazo trata de predecir los requerimientos de carga en el período de unos pocos meses a algunos años y con mayor frecuencia que los otros dos tipos de pronóstico de carga, incorpora otros factores y variables socioeconómicas como el crecimiento de población y población de un área, así como el producto nacional bruto en el proceso de pronóstico. Este tipo de pronóstico a largo plazo informa a las compañías eléctricas acerca de los requisitos futuros para la expansión de la red, las compras de equipos y la contratación de personal y proporciona indicaciones fiables de las tendencias de crecimiento futuro en la demanda de energía.

Dentro de este marco, en este TFG se propone aunar distintos métodos para predecir la carga eléctrica de corto a mediano plazo (24 horas). Además, para conseguir este objetivo, se emplea un historial de 1 año.

El sistema de predicción consiste en la combinación de dos herramientas, a saber, el Análisis de Espectro Singular (en adelante SSA) y Redes Neuronales (en adelante ANN).

El algoritmo de predicción consta de varias etapas: 1) un primer preprocesamiento mediante una herramienta de *detrending*, para eliminar la tendencia del histórico; 2) un procesamiento con el algoritmo SSA, al objeto de descomponer los datos en distintas componentes reconstruidas y un residuo; 3) una tercera fase consistente en la predicción de las componentes extraídas bien con SSA o bien mediante ANNs; 4) una cuarta fase consistente en la predicción de la tendencia y el residuo obtenido mediante SSA, empleando ANNs; y 5) una última fase que consiste en la adición de las distintas predicciones y la comparación entre las mismas.

Se emplean redes neuronales por la capacidad que muestran de encontrar relaciones no lineales entre el histórico de una determinada variable y su comportamiento futuro, además, como las redes neuronales empleadas implementan un algoritmo autoregresivo, son capaces de predecir componentes estocásticas.

Este TFG se encuadra en lo que hoy en día se ha venido en llamar Smart Grids (SG), que son el resultado de aplicar las tecnologías de la información y las comunicaciones a las redes eléctricas. Un caso de uso habitual en el despliegue de las SG, consiste en dotar de inteligencia a los centros de transformación para que puedan autónomamente disponer de una predicción de la carga del día siguiente. Por otro lado, dado que en este TFG se aborda la temática del manejo de información histórica para obtener conclusiones operativas, también podría decirse que el TFG se encuentra dentro del campo del '*big data*' o de la Industria 4.0.

En las siguientes secciones se explicarán, por un lado, el algoritmo y por otro, cada una de las partes del proceso seguido para el ajuste del mismo.

El algoritmo está en parte escrito en Lenguaje R y parte escrito en lenguaje de Matlab. El lenguaje R es un lenguaje de programación estadístico de licencia abierta que surgió a partir del lenguaje S [1] que fue desarrollado en 1993 por Ross Ihaka y Robert Gentleman. Una explicación más extensa del mismo puede ser encontrada en [2] .Tiene una gran acogida en aquellos campos en los que se necesite modelar sistemas no lineales y que requieran de un análisis estadístico como la minería de datos, la investigación biomédica, la bioinformática y las matemáticas financieras.

Para la realización de este proyecto se ha dispuesto de una serie de históricos de consumo de energía de una región de Francia llamada Occitanie, cuyos consumos eléctricos están publicados en la web de la RTE francesa [3], disponibles para ser descargados de libremente.

Para todos los cálculos relacionados con SSA se ha utilizado como referencia el libro "Singular Spectrum Analysis for Time Series" [4].

5. GLOSARIO DE ACRÓNIMOS Y ABREVIATURAS

TFG: Trabajo de Fin de Grado.

SSA: Singular Spectrum Análisis o Análisis de Espectro Singular.

SVD: Singular Value Descomposition o Descomposición del Valor Singular.

RTE: Réseau de transport d'électricité u Operador de transporte de electricidad francés.

ANN: Artificial Neural Network o Redes Neuronales artificiales.

NARX: Nonlinear Autoregressive Network with Exogenous inputs.

IDE: Integrated Development Environment o Entorno de Desarrollo Integrado.

MAPE: Mean Absolute Percentage Error o Error Medio Absoluto.

NMBE: Normalized Mean Bias Error o error de sesgo promedio normalizado.

NRMSE: Normalized Root Mean square Error o Raíz del error cuadrático medio normalizado.

IA: Inteligencia Artificial.

SVR: Regresión de vectores de soporte.

ARIMA: Método de media móvil integrada autoregresiva.

ARMAX: Método de media móvil integrada autoregresiva con entradas exógenas.

SARIMA: Modelo de media móvil integrada autoregresiva estacional.

HW: Modelos de suavización exponencial (Holt-Winters).

FF: *FeedForward* (Prealimentación).

6. MEMORIA

6.1 Introducción

En este apartado se pretende dar una visión general del funcionamiento del algoritmo que se propone.

El algoritmo consiste en la predicción de la carga en MWh que va a tener una determinada zona geográfica con un horizonte de 24 horas, para ello, se utiliza un histórico de la carga de la zona de 1 año.

Para analizar el histórico y extraer la tendencia, se utiliza una herramienta de *"fitting"* (o ajuste de curvas) de Matlab que permite ajustar unos datos a una serie de curvas mediante algoritmos previamente escritos, esta herramienta se utiliza en nuestro caso para extraer la tendencia de los datos.

Una vez se ha extraído la tendencia, mediante el algoritmo SSA se extraen las componentes periódicas que componen la serie temporal que conforma el histórico de datos. Con estas componentes extraídas, se genera un residuo de la descomposición, que puede resultar de utilidad, según se explicará más adelante.

Extraídas componentes, tendencia y residuo, llega la fase de realizar la predicción. Para ello, hay dos alternativas. En el caso de las componentes, se exponen dos variantes: la predicción mediante SSA y la predicción mediante ANNs. Para la predicción de la tendencia solo se ha tenido en cuenta la opción de predecir mediante Redes Neuronales. Para la predicción del residuo se han utilizado, asimismo, Redes Neuronales.

Una vez se han realizado las predicciones, se abordan las diversas maneras de reconstruir, ya que se nos presentan diversas opciones para realizar la predicción, como se ha expuesto en el apartado anterior. Se podría utilizar solo las predicciones de las redes neuronales, utilizar las predicciones de las redes neuronales y las extraídas mediante SSA o una mezcla de ambas. Además, existen unos residuos, que pueden añadirse a las predicciones o desecharse.

Por lo tanto, el algoritmo aquí expuesto emplea una aproximación híbrida, dado el uso combinado de SSA y ANNs. Esta solución híbrida se adoptó porque la descomposición en componentes hace que estas sean mucho más sencillas de predecir de forma separada, ya sea por algoritmos de predicción de SSA o por las redes neuronales. Además, las redes neuronales permiten predecir de forma muy precisa el residuo extraído mediante SSA y la tendencia. Un algoritmo corrige las debilidades del otro, ya que intentar predecir todo a la vez mediante ANNs resulta en una topología compleja de difícil entrenamiento al tratarse de un problema demasiado complejo.

En la Figura 1 se muestra un esquema del algoritmo para mejor comprensión del mismo.

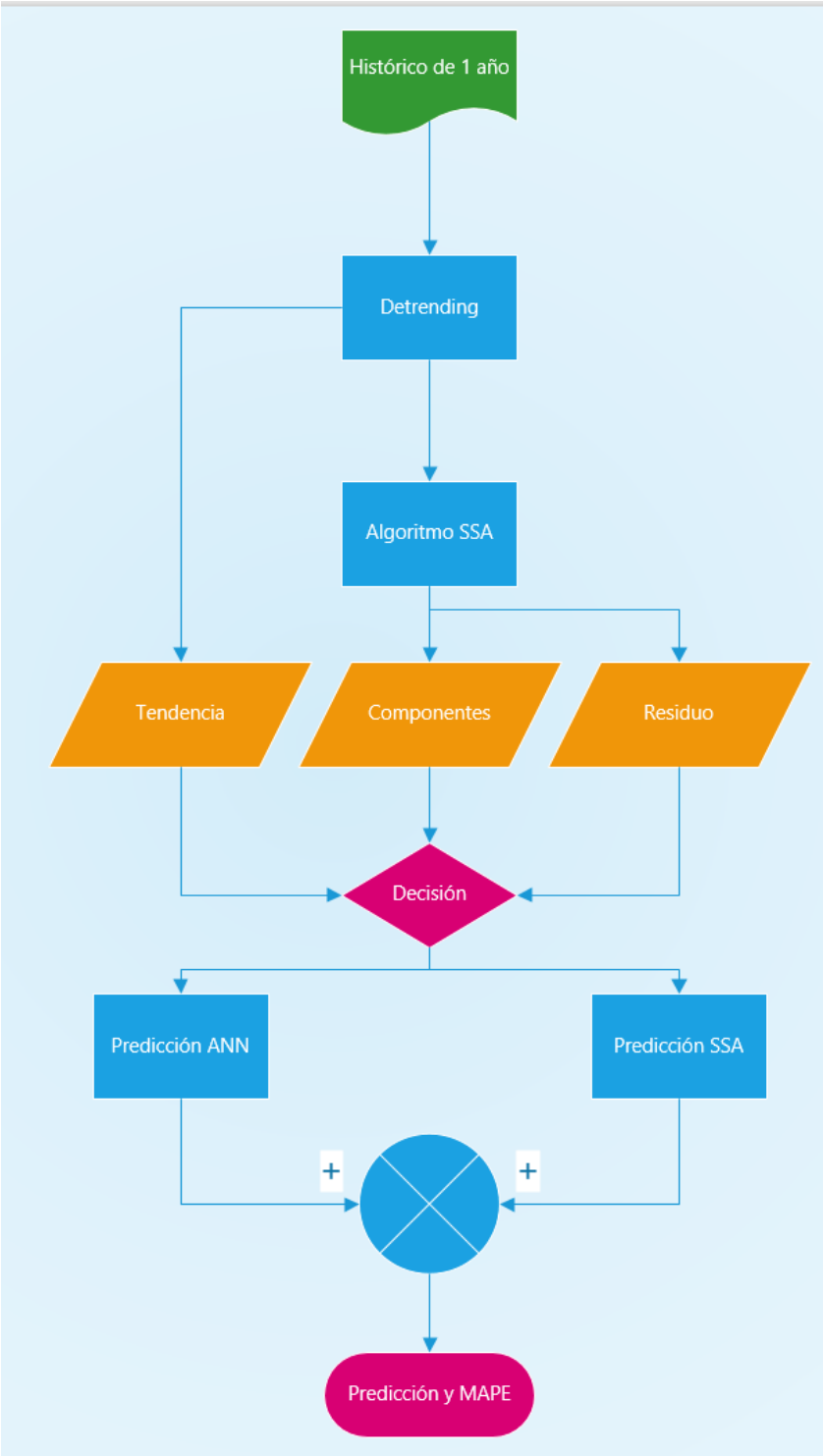


Figura 1. Esquema general del algoritmo

6.2 Revisión de métodos de predicción

Previo a la explicación teórica de los algoritmos que se han utilizado, se va a realizar un breve resumen de las técnicas de predicción que han utilizado para la estimación de la carga eléctrica.

Las compañías utilizan estos métodos para poder gestionar correctamente sus recursos y poder dotar al consumidor de un suministro fiable, por lo que es de suponer que elegirán aquellos métodos que sean más precisos y, además, requieran de un tiempo de cómputo razonable.

Actualmente, el campo de los métodos de predicción está en auge, ya que las empresas demandan algoritmos de predicción para todos los campos imaginables, por lo que aquí solo se van a exponer los más importantes.

La elección del modelo de predicción puede ser complicado, ya que, como ya se ha dicho, dependiendo del horizonte de predicción que se desee, existen distintos modelos (corto plazo, medio plazo y largo plazo).

Se pueden clasificar los algoritmos en dos grandes grupos: Los métodos basados en inteligencia artificial (IA en adelante) y los métodos basados en aproximaciones a series temporales. Para una visión general de todos ellos, véase [5].

Los métodos basados en inteligencia artificial, a su vez se pueden dividir en dos subgrupos: métodos basados en redes neuronales (ANN) y métodos basados en la regresión de vectores de soporte (SVR en inglés).

Tradicionalmente, se han utilizado los métodos de análisis de series temporales, por lo que estos algoritmos están bien asentados. Entre ellos, cabe destacar el método de media móvil integrada autoregresiva Box-Jenkins (ARIMA) [6]. A partir de este método se han realizado diversas variantes, como por ejemplo el ARMAX [7] que es el método de media móvil integrada autoregresiva con entradas exógenas (las redes NARX (*Neural Autoregressive with external inputs)* están basadas en este modelo, como se explicará en el apartado de redes neuronales). Con origen en este método se encuentran los algoritmos más utilizados, como el modelo de media móvil integrada autoregresiva estacional (SARIMA), los modelos de suavización exponencial como Holt-Winters (HW) [8] y el modelo Holt-Winters, con suavizado lineal (SHW) [9].

Estos métodos tradicionales han sido implementados en inteligencia artificial (redes neuronales habitualmente), debido a su elevada eficacia y su capacidad de responder a cambios en el modelo aprendiendo de los históricos. Las redes neuronales se han desarrollado intensivamente desde 1980 en adelante gracias a los avances en la computación, son muy utilizadas debido a que son capaces de resolver problemas no lineales.

Hay diversas arquitecturas (se exponen posteriormente en este trabajo) que consiguen muy buenos resultados en cuanto a error en la predicción, por lo que suponen una alternativa viable para la solución al problema.

Finalmente, aunque los algoritmos de regresión de vectores de soporte suelen ser utilizados en problemas del tipo "clasificación", también pueden emplearse en problemas de predicción. El algoritmo consiste en el mapeado del vector de datos de entrada, en un espacio vectorial donde es más sencillo encontrar una regresión lineal de los mismos, para más información acudir a [10].

6.3 Descomposición SSA

El Análisis de Espectro Singular o SSA como se denotará en adelante, es una técnica de análisis de series temporales que se desarrolló en 1985 y que es muy usada en climatología para predicción. Tiene la ventaja de que es una técnica que no depende del modelo de la dinámica que se esté analizando y permite descomponer una serie temporal en principio compleja, en varias series temporales más sencillas de analizar. El estudio de SSA de este TFG se basa en el libro [4].

El análisis SSA básico consta de varias fases, a saber:

- Empotrado (Embedding)
- Descomposición
- Agrupación
- Reconstrucción

A continuación, se explicarán una a una cada una de las anteriores fases.

6.3.1 Embedding

Para empezar con el análisis, consideraremos una serie temporal de N valores en la cual no son todos nulos.

$$F = (f_0 \dots, f_{N-1})$$

Ecuación 1

En la fase de *embedding* o empotramiento, formaremos una matriz cuyas filas serán los valores de la serie con un conjunto de retardos.

Consideremos ahora un entero positivo L, tal que 1<L<N . L se denomina ventana y es el único parámetro del algoritmo, por lo que es un punto vital para el mismo. Más adelante se discutirá la elección de este parámetro.

El proceso de *embedding* forma K=N-L+1 vectores en retardo. A continuación, se forma la matriz de los vectores en retardo, que a partir de ahora se denominará como matriz de trayectoria (X):

$$X = \begin{pmatrix} f_0 & f_1 & f_2 & \cdots & f_{K-1} \\ f_1 & f_2 & f_3 & \cdots & f_K \\ f_2 & f_3 & f_4 & \cdots & f_{K+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{L-1} & f_L & f_{L+1} & \cdots & f_{N-1} \end{pmatrix}$$

Ecuación 2

Como se puede observar en Ecuación 2, hay L vectores retardados y cada vector tiene longitud N-L.

También es posible observar que dos elementos de X son iguales si la suma de la columna y fila a la que pertenecen es la misma. Es decir, siendo i_1, j_1 la fila y columna de un elemento e i_2, j_2 la fila y columna de un segundo elemento, los dos elementos serán iguales si $i_1+j_1=i_2+j_2$. A este tipo de matrices se les denomina matrices de Hankel.

6.3.2 Descomposición

En esta fase se pretende conseguir la descomposición en valores singulares (SVD en adelante) de la matriz de trayectoria.

Denotemos $S = XX^T$, $\lambda_1, \dots , \lambda_L$ los autovalores de S tomados en orden decreciente de magnitud y $U_1 , \dots , U_L$ el sistema ortonormal de autovectores asociados a esos autovalores.

Si denotamos $V_i = \frac{X^T U_i}{\sqrt{\lambda_i}}$ (i=1, (i=1, …, L), podemos reescribir X como:

$$X = X_1 + \cdots + X_L$$

Ecuación 3

Siendo $Xi = \sqrt{\lambda_i} U_i V_i^T$.

La colección de $\sqrt{\lambda_i}, U_i, V_i$ en adelante se denominará el autotriple i del SVD y es el pilar para la siguiente fase.

6.3.3 Agrupación

Una vez se ha conseguido descomponer la serie inicial en los distintos autotriples, el siguiente paso consiste en agruparlos en distintos subgrupos. La clave utilizada en este proceso se denomina criterio de agrupación o de “grouping”.

Es importante que cuando se agrupen dos o más autotriples, los autotriples agrupados tengan una fuerte correlación entre ellos, y que al final, entre los grupos que se hayan formado, no exista demasiada correlación; idealmente, no debería existir ninguna.

La separabilidad implica que si consideramos dos grupos tal que $X = X_{I1} + X_{I2}$ donde I1 e I2 es el resultado del agrupamiento de autotriples en dos grupos, la reconstrucción total es la suma de las reconstrucciones de los dos grupos:

$$F = F_1 + F_2$$

Ecuación 4

La clave de la agrupación es el criterio a seguir para agrupar autotriples. Cuando se realiza el análisis SSA, una herramienta muy potente es un dibujo de la autocorrelación con el resto de autotriples, este dibujo tiene la siguiente forma (Figura 2):

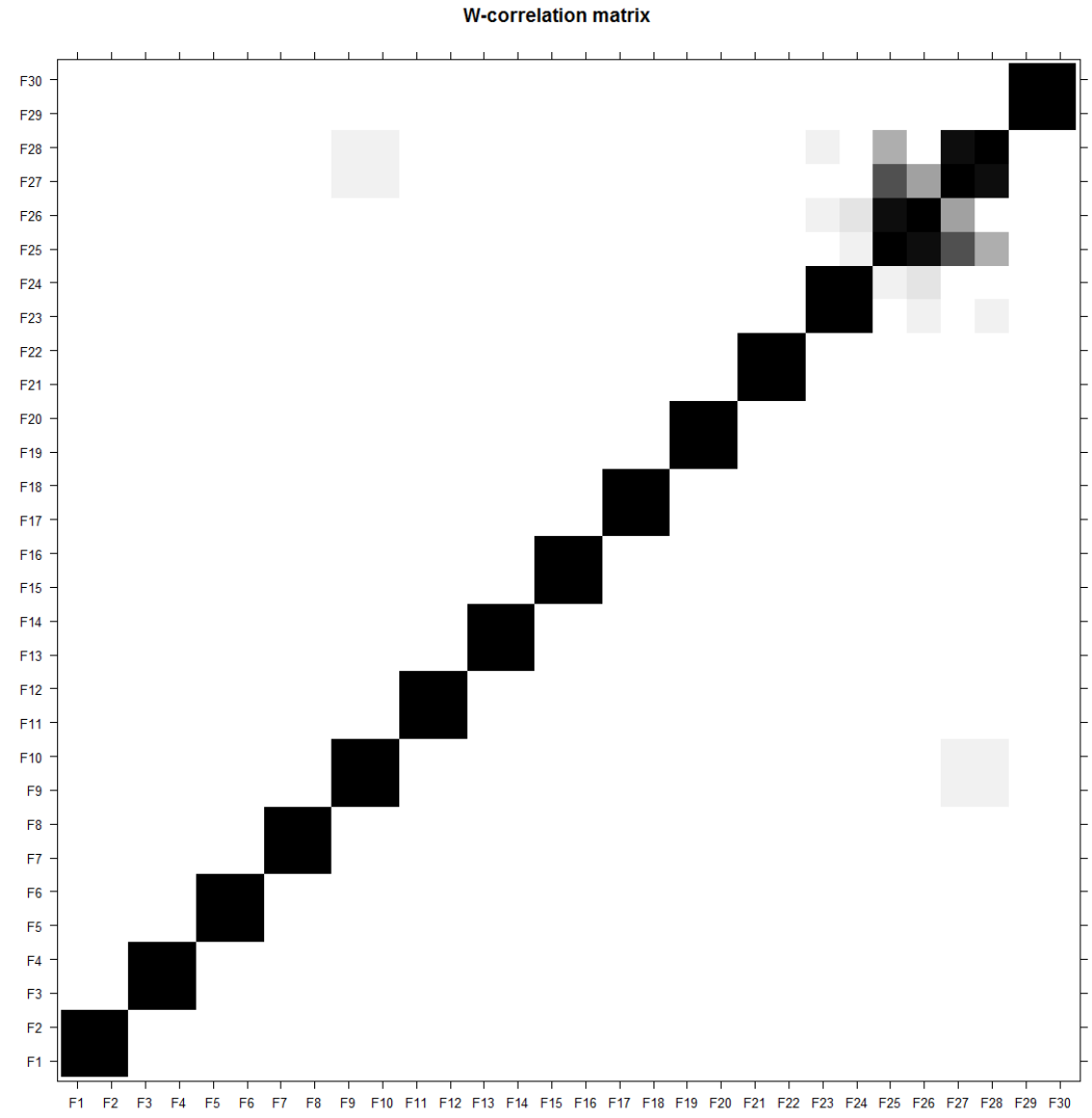


Figura 2. Matriz de correlación

Donde en una escala del blanco al negro se ve la correlación que tienen unos autotriples con otros. En los ejes XY se encuentran los autotriples, numerados del 1 al 30 en función de su peso. En los cuadrados de la figura se puede ver si la correlación es fuerte o débil.

En este caso, cuando hay dos autotriples que están altamente correlacionados, se suele tratar de una componente periódica; esto se puede ver más claramente usando otra herramienta: la reconstrucción de los autovectores, tal y como se puede ver en la Figura 3:

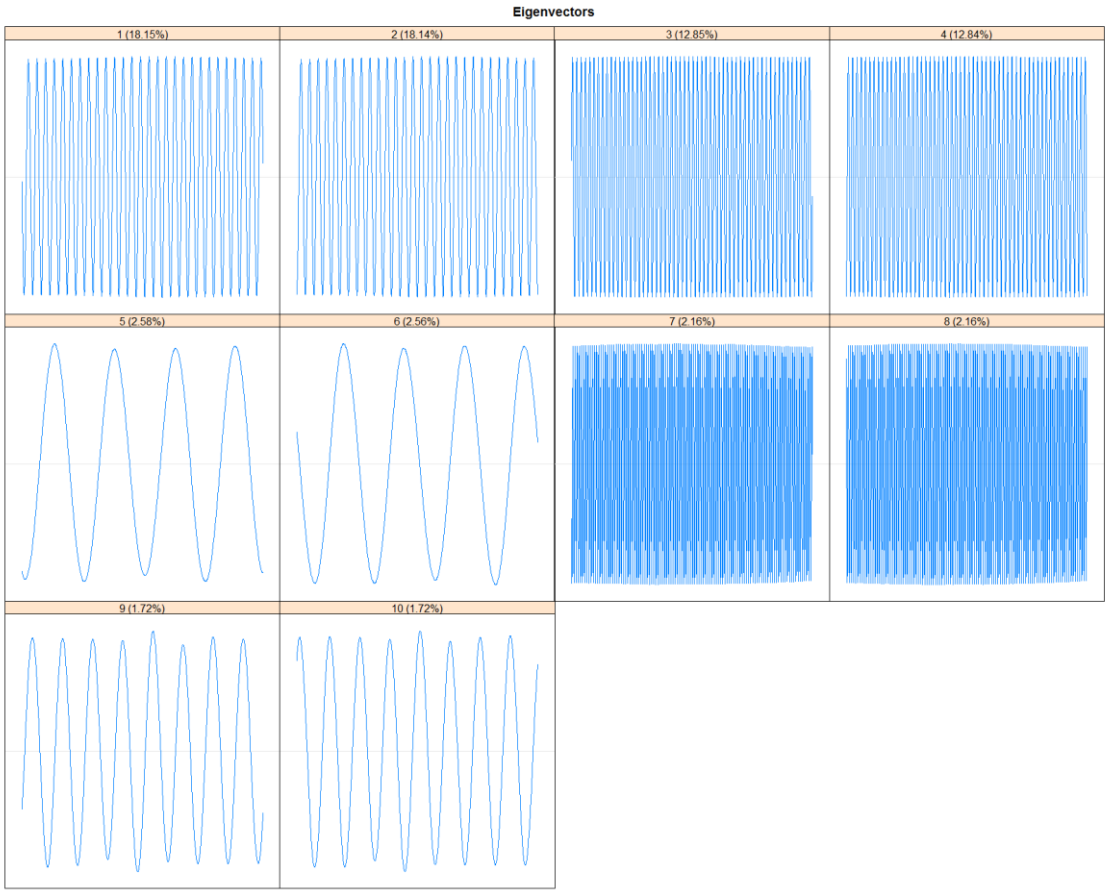


Figura 3. Reconstrucción de Autovectores

Cuando dos autovectores son seno y coseno, respectivamente, del mismo periodo, están altamente correlacionados y se pueden agrupar. Este hecho se puede contrastar con el análisis de autocorrelación anterior.

Para más seguridad, también es recomendable hacer un *scatterplot* (representación gráfica de los autovectores de un autotriple como nube de puntos) de los autotriples agrupados de dos en dos, la figura obtenida es la siguiente (Figura 4):

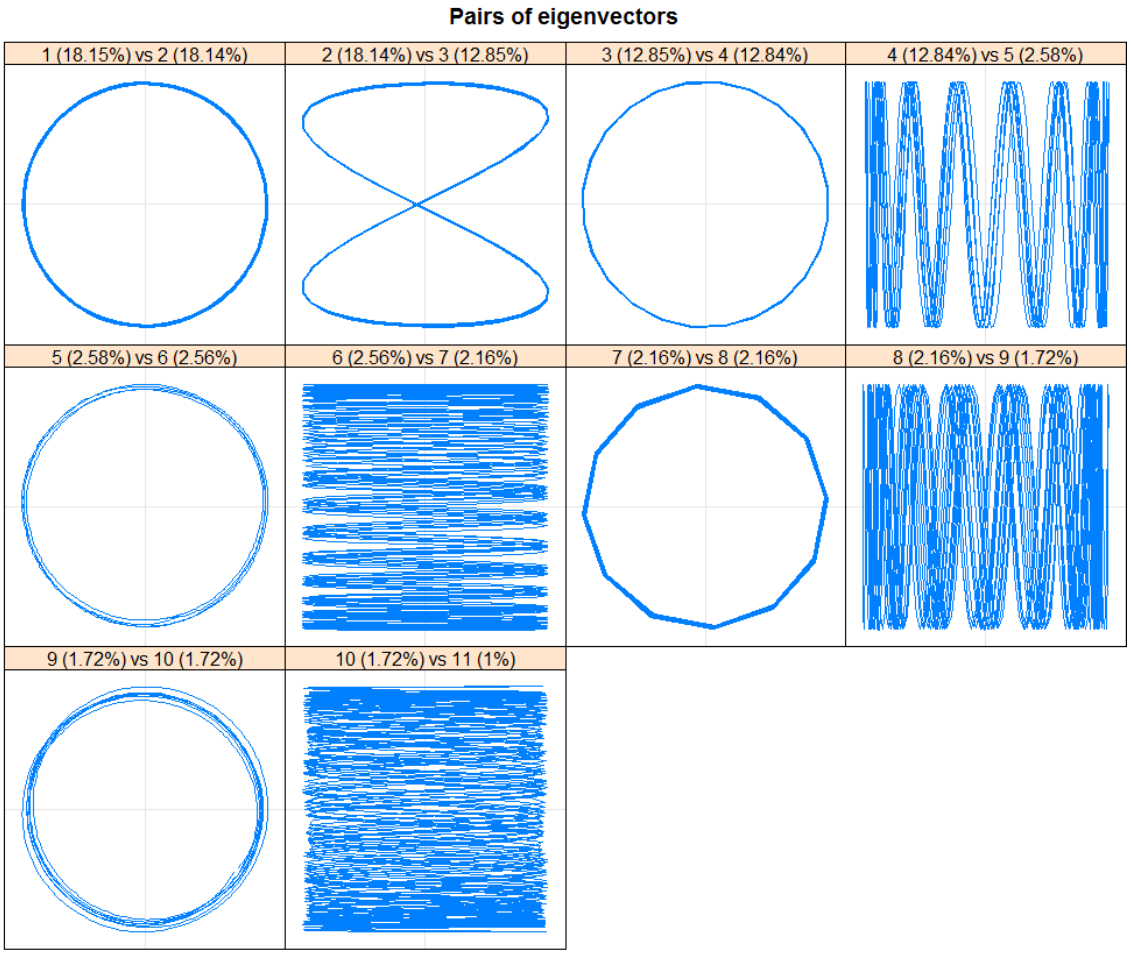


Figura 4. Scatteplot

Como se puede observar en la Figura 4, cuando los dos autovectores forman una componente periódica, en el *scatterplot* aparece un polígono regular. Si no es completamente regular es porque en un caso real, siempre hay algo de ruido en el sistema.

El número de vértices del polígono indica el periodo que tiene la componente en muestras. Por ejemplo, la agrupación del autovector 7 y el 8 forman una componente de periodo L/12 muestras. A continuación, en la Figura 5, se muestra la componente reconstruida:

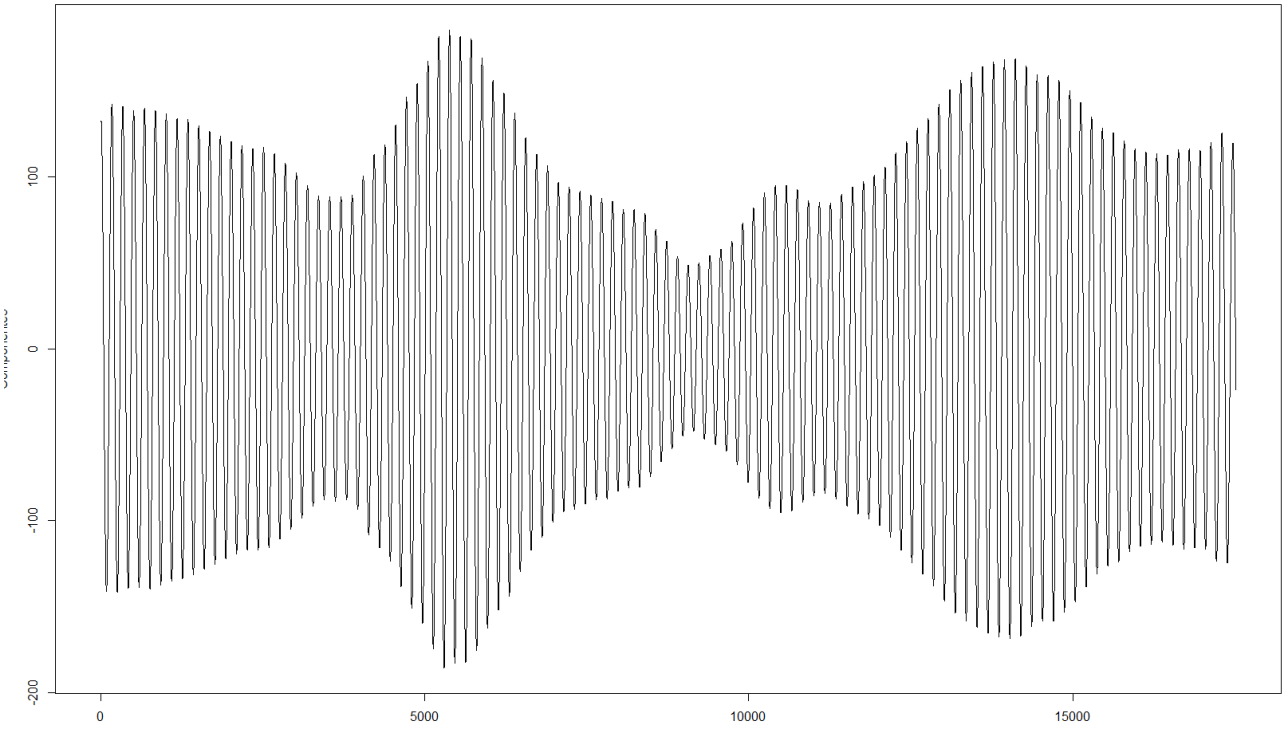


Figura 5. Reconstrucción de Autovectores 7 y 8

En este caso particular, la componente reconstruida tiene una modulación añadida en amplitud. Si hacemos zoom en la señal podremos descubrir que se trata de una componente senoidal de periodo L/12.

Estas son las herramientas más importantes para realizar la agrupación. Además, es importante saber cuándo se ha de dejar de agrupar autotriples y considerar el resto "residuo" o "ruido". Para ello, en el gráfico de autocorrelación (Figura 2) se puede observar que hay un punto a partir del autotriple 25 en el que los autotriples empiezan a estar muy correlacionados unos con otros y es prácticamente imposible emparejarlos de 2 en 2 o incluso de tres en tres. A partir de ese momento se puede considerar residuo el resto de autotriples. Para ver la importancia de la información que se pierde, se puede dibujar una gráfica con la representación del peso de los autotriples y observar cuánta información es la despreciada (Figura 6):

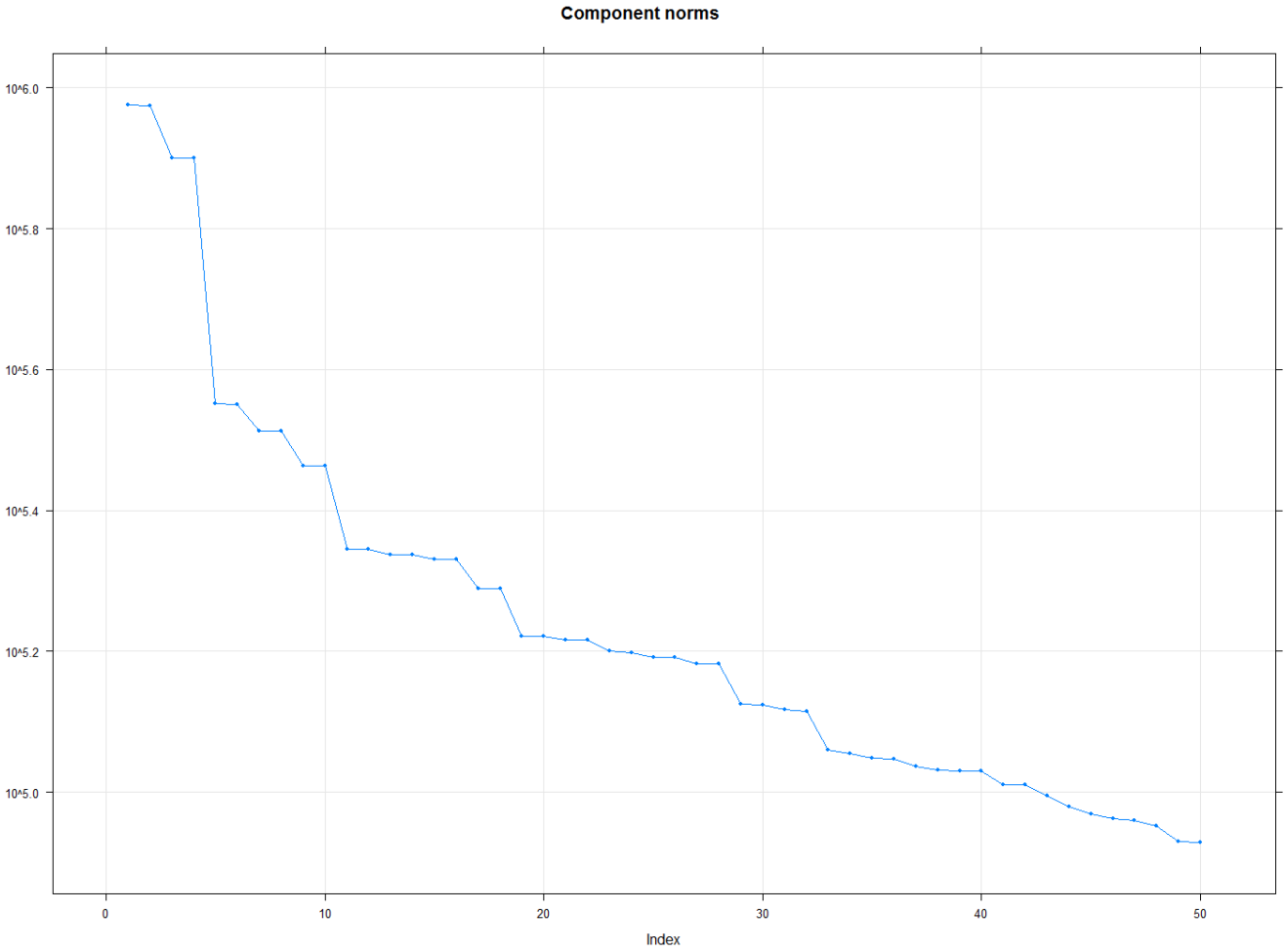


Figura 6. Representación de autovalores

Aquí se puede observar que al dejar de agrupar autotriples, a partir del autotriple 19 se desprecian todos los autotriples, por lo que se estarían perdiendo autotriples de poca potencia. La Figura 15 representa el valor de los autovalores (λ), por lo que podemos concluir que la mayor parte de la información se preserva.

Una vez elegida la agrupación, hay que pasar a la siguiente fase.

6.3.4 Reconstrucción

En la fase de reconstrucción, que es la última, hay que comprobar varias cosas. En primer lugar, que los grupos que se han formado son incorrelados entre sí. Para ello, primero se reconstruyen y posteriormente se aplica un análisis de correlación.

A continuación, hay que comprobar si la reconstrucción realizada es satisfactoria. De ser así, se admite como buena y el proceso concluye.

Vamos a explicar en primer lugar el algoritmo matemático que rige la reconstrucción, llamado **"***Diagonal averaging***".**

Si el agrupamiento ha sido correcto, las matrices que resultan de agrupar los autotriples son matrices de Hankel, igual que la matriz de trayectoria inicial (Ecuación 2).

El procedimiento consiste en hacer una media de las diagonales secundarias de las matrices, para conseguir extraer una serie temporal de la matriz, de la siguiente forma:

$$g_k = \begin{cases} \dfrac{1}{k+1}\displaystyle\sum_{m=1}^{k+1} y^*_{m,k-m+2} & \text{for } 0 \le k < L^* - 1, \\ \dfrac{1}{L^*}\displaystyle\sum_{m=1}^{L^*} y^*_{m,k-m+2} & \text{for } L^* - 1 \le k < K^*, \\ \dfrac{1}{N-k}\displaystyle\sum_{m=k-K^*+2}^{N-K^*+1} y^*_{m,k-m+2} & \text{for } K^* \le k < N. \end{cases}$$

Ecuación 5

Donde:

$$L^* = \min(L, K)$$

$$K^* = \max(L, K)$$

$$N = L + K - 1;$$

$$y_{ij}{}^* = y_{ij} \quad Si\ L < K$$

$$y_{ij}{}^* = y_{ji} \quad Si\ L > K$$

Esta operación reconstruiría cada uno de los grupos que se han formado a partir del agrupamiento y nos debería dar tantas series temporales como grupos. Si eran completamente separables, al sumar estas series temporales obtendríamos una reconstrucción de la serie original.

Para comprobar la separabilidad, se puede obtener una gráfica indicando la correlación entre los distintos grupos formados, como la de la Figura 7:

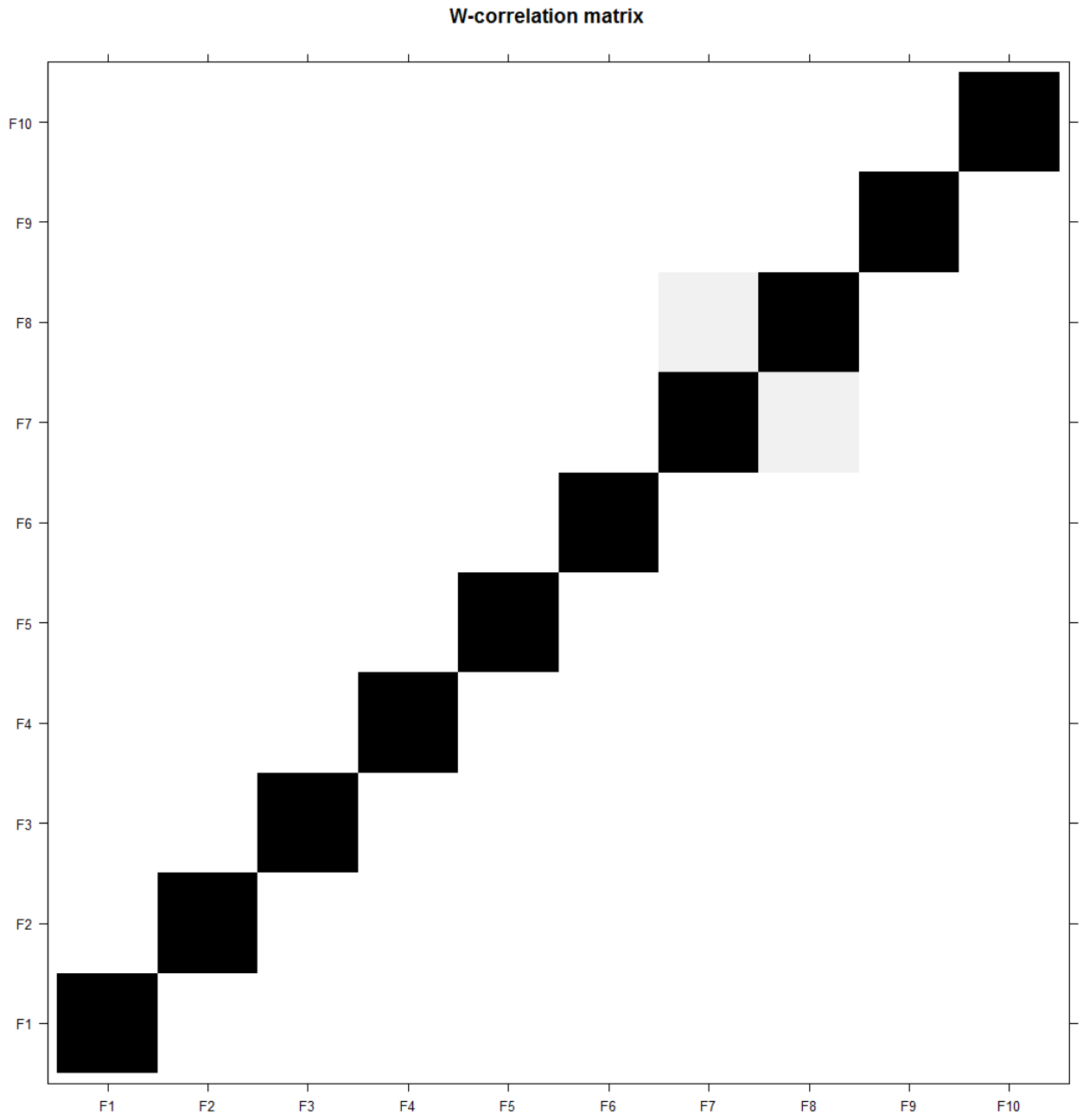


Figura 7. Correlación de grupos reconstruidos

En este caso, los grupos son completamente independientes unos de otros y se obtiene un diagrama de correlación perfecto, por lo que podemos reconstruir la señal y compararla con la original (Figura 8):

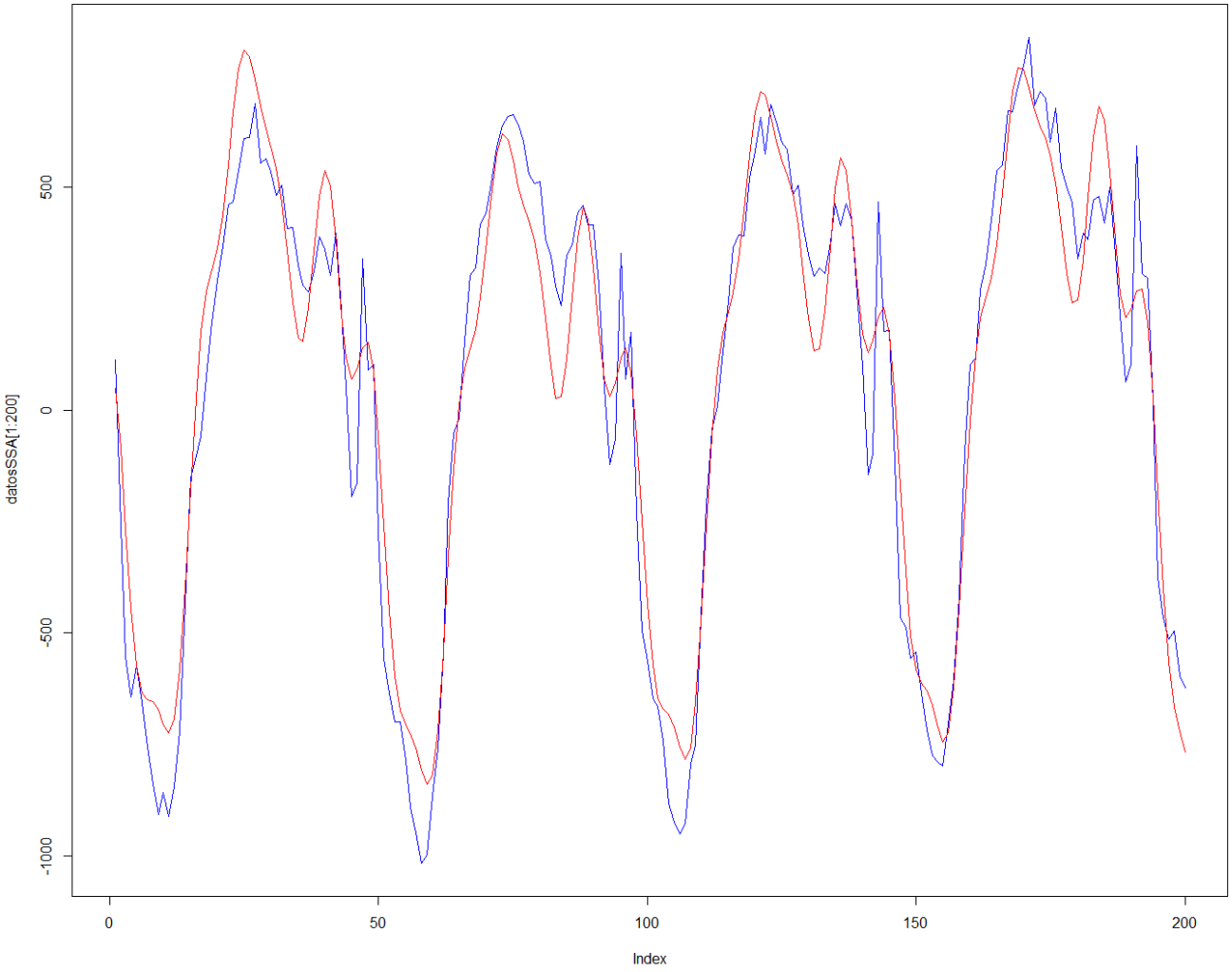


Figura 8. Datos reales vs Predicción

En rojo está el resultado de la reconstrucción mediante SSA y en azul la serie original. Como se puede observar, no es exacta, ya que hemos despreciado una parte de la señal considerándola ruido.

En conclusión , este es el procedimiento a seguir para aplicar SSA en un caso general. Para más detalle se recomienda [11] y [12].

En los siguientes apartados se explicará con detalle la variante a este procedimiento general que se ha utilizado en este TFG. Esta variante se ha elegido para adecuar el método al problema de predicción. Para más información sobre el algoritmo SSA se recomienda revisar [13] y [14].

6.4 Redes Neuronales

6.4.1 Introducción

En la solución adoptada para la resolución de la problemática de la predicción, se ha optado por utilizar redes neuronales, que permiten hacer una predicción de una serie temporal a partir de valores de instantes anteriores de la misma y de una serie de señales auxiliares.

Existen multitud de topologías de redes, cada una con su propia función de transferencia [15]. En la Figura 9 se muestra una topología sencilla y estándar para entender cómo funciona una red neuronal:

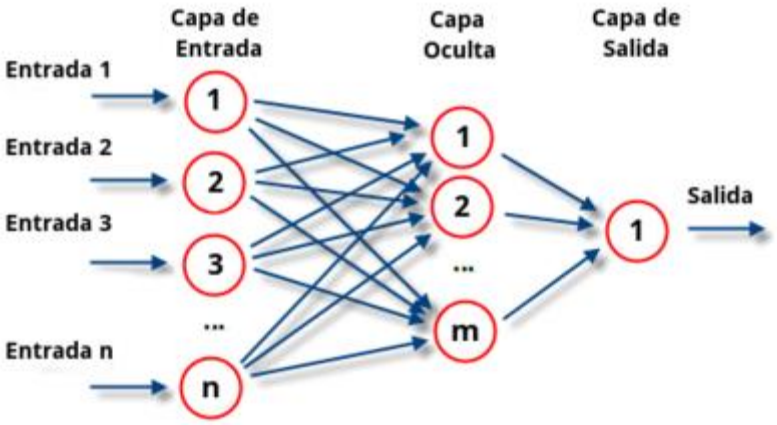


Figura 9. Estructura de red

Como se puede observar en la Figura 9, la red consiste en una serie de "neuronas" que están interconectadas entre sí y organizadas por capas. Transmitiendo la información de unas a otras, previa ponderación, generan salidas (en la Figura 9 solo hay una salida, pero puede haber más).

Las neuronas son unidades matemáticas que constan de entradas que están multiplicadas por un determinado peso (w), una entrada de offset (umbral), una salida y una función de transferencia. Para la función de transferencia de salida se puede optar entre una relación lineal o no lineal. En este último caso, se elige habitualmente la función sigmoidal. En la Figura 10 se muestra el esquema de una neurona.

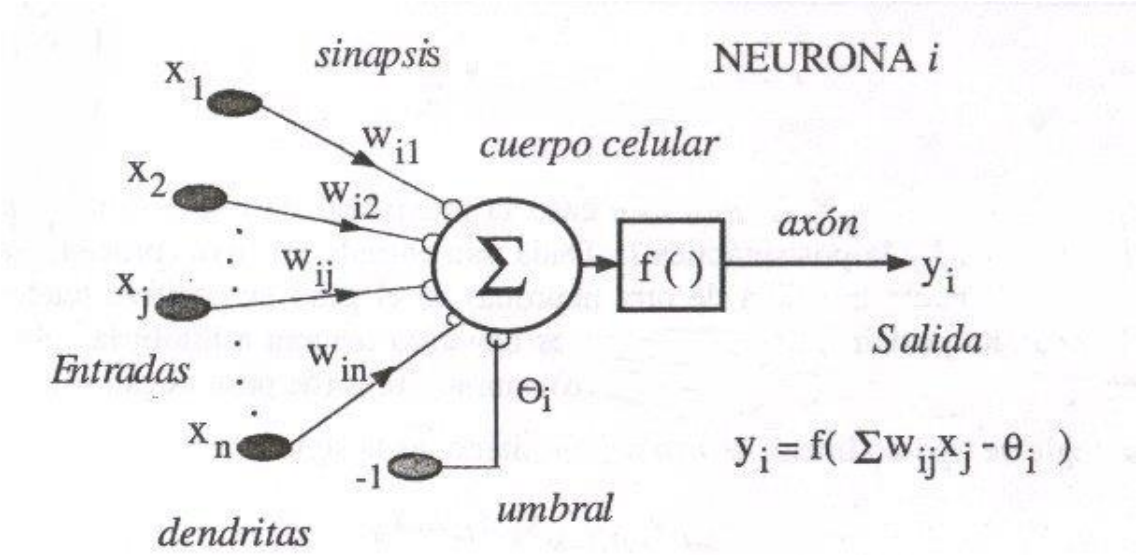


Figura 10. Esquema de una neurona artificial

6.4.2 Topologías

Entre las diversas topologías que existen, merece la pena destacar entre las más utilizadas la topología *FeedForward.* Esta topología es la más sencilla de todas, ya que la red al entrenarse, lo único que busca es una relación directa entre los datos de salida y los de entrada, es decir, la red no está realimentada. El modelo de red más utilizado para este tipo es el Perceptrón, que consiste en una red con una capa de entrada, una o más capas ocultas y una capa de salida. En la Figura 11 se muestra un esquema de una red Perceptrón.

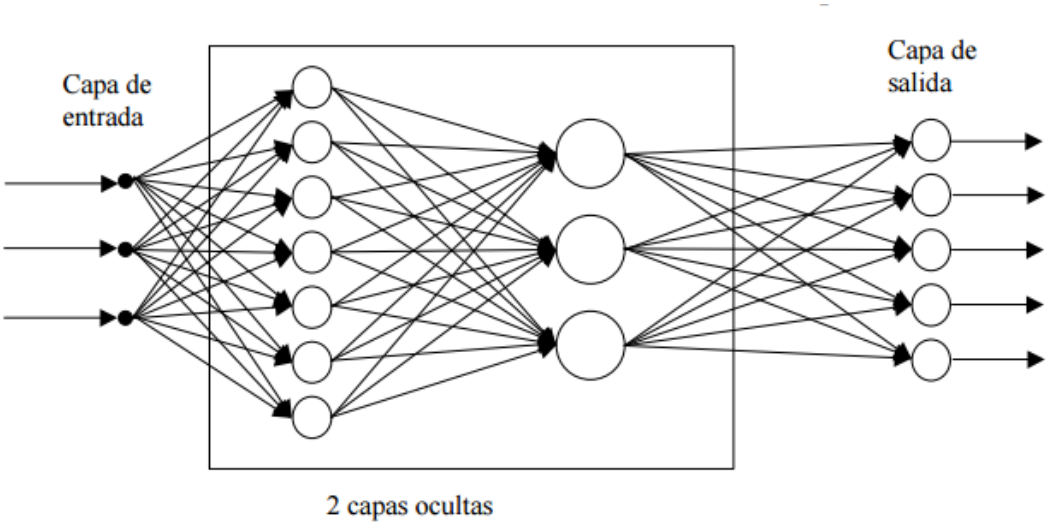


Figura 11. Red FeedForward: Perceptrón

Las redes neuronales *FeedForward* (FF) tienen la capacidad de aproximar cualquier función que sea acotada y derivable [16].

Otra topología de red (la utilizada en nuestro caso) es la topología NARX (*Neural Autoregressive with external inputs*), que se basa en una red *FeedForward* realimentada, según el esquema de la Figura 12:

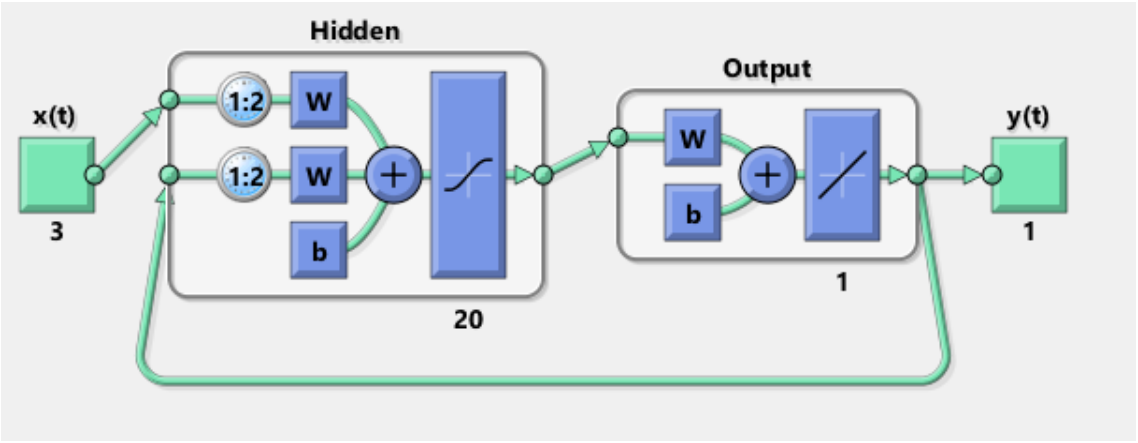


Figura 12. Esquema de Red NARX

Donde x(t) son entradas externas que aportan "ayuda" al entrenamiento de la red, y(t) es la salida de la red que además se incluye como entrada en el instante siguiente y tiene tres capas de neuronas: una capa de entrada, una capa oculta de neuronas con función de transferencia sigmoidal, y una capa de salida con función de transferencia lineal.

La diferencia respecto a la topología *FeedForward* es la realimentación. La salida afecta a la siguiente muestra, lo que provoca que la salida sea más precisa que en una topología que use un lazo abierto

6.4.3 Entrenamiento

El entrenamiento de la red es el proceso más importante a la hora de utilizar las redes neuronales, ya que una mala elección de sus entradas o una mala elección del algoritmo puede resultar en unos pésimos resultados. El entrenamiento consiste en asignar unos pesos a las entradas de todas las neuronas y simular la salida que dan respecto a las entradas, se supone un set de datos de entradas del cual se conocen las salidas, por lo que al realizar la simulación, se analiza el grado de exactitud de la red mediante un indicador que depende del tipo de entrenamiento elegido y se reajustan los pesos de acuerdo al algoritmo de entrenamiento elegido, a cada uno de los procesos de simulación-comprobación-ajuste se le llama iteración.

Existen multitud de algoritmos de entrenamiento para las redes, un buen resumen puede ser encontrado en la página de la referencia [17].

Para el entrenamiento de las redes que nos ocupan, las redes NARX, se han tenido en cuenta tres opciones para el algoritmo: Lavenberg-Marquardt, Regularización Bayesiana y el método de gradientes conjugados escalonados.

El método de Lavenberg-Marquardt [18] (También conocido como método de los mínimos cuadrados amortiguados) consiste en la resolución del problema de ajuste a una curva mediante mínimos cuadrados, es decir, el algoritmo se encarga de ajustar los pesos de las neuronas en cada iteración dependiendo del error cuadrático respecto a la

salida de la red, el algoritmo precisa de calcular matrices Jacobianas, por lo que hay que tener en cuenta que el tiempo de cómputo puede ser elevado si crece el número de neuronas y/o entradas de la red. Este algoritmo tiene el peligro de que, al encontrar un mínimo para la función de ajuste, este mínimo puede tratarse de un mínimo local, y no global, por lo que es necesario cuando se utiliza este método realizar varios entrenamientos sobre la red, ya que unas diferentes condiciones iniciales pueden dar lugar a resultados finales distintos.

El método de la Regularización Bayesiana [19] consiste en la optimización de la función de densidad de probabilidad de dos parámetros llamados α y β que caracterizan la varianza del error cuadrático que comete la red neuronal respecto al que debería ser su valor de salida. De manera rápida, se puede decir que es una derivación del método Lavenberg-Marquardt en la que se calcula el error de la misma manera, pero el algoritmo de ajuste de pesos se basa en una función de probabilidad (Teorema de Bayes) [20], este método requiere del cálculo de matrices Hessianas, por lo que el entrenamiento puede demorarse bastante en el tiempo.

El método de gradientes conjugados escalonados [21] es el método que requiere menos tiempo de los tres que se ha explicado, esto es debido a que consiste en la obtención de gradientes, el indicador utilizado para el cálculo de los nuevos pesos es el error cuadrático, la función utilizada consiste en relacionar el cambio del error respecto al cambio de los pesos, se utiliza una retropropagación, mientras la derivada de la función que relaciona los cambios en el error con los cambios de peso no se anule, se puede seguir iterando. Este método es extraordinariamente rápido, pero los resultados de precisión obtenidos son muy pobres en comparación con los dos métodos anteriores.

El entrenamiento de las redes se ha realizado con Matlab, con la herramienta de redes neuronales para series temporales. Esta herramienta tiene implementados los tres algoritmos que aquí se han explicado, por lo que para la decisión de cuál de los tres se utiliza, se hicieron varias pruebas experimentales. En primer lugar, se desechó la opción de utilizar los gradientes conjugados debido a que producían resultados con muy poca precisión, lo cual generaba una predicción muy pobre. Posteriormente, nos decantamos por el algoritmo de Lavenberg-Marquardt frente a la regularización bayesiana porque la regulación tardaba en entrenar cada red en torno a 2 horas cada una, ya que las redes son bastante complejas debido a los retardos y el gran número de redes en la capa oculta, además la diferencia de resultados entre los dos métodos no era significativa. Nos quedamos entonces con el método Lavenberg-Marquardt, ya que es el punto medio en el compromiso entre tiempo de entrenamiento de las redes y la eficacia del mismo. Este método es más rápido que la regularización bayesiana porque en vez de operar con la matriz Hessiana, calcula Jacobianas, lo que es infinitamente más rápido, pero a su vez es más preciso que el de gradientes conjugados.

En el entrenamiento de las NARX también es importante elegir adecuadamente las entradas externas que se quieran emplear, crear una matriz de columnas en la que cada columna sea una entrada, incluido un historial de la serie temporal a predecir y preparar otra matriz con los objetivos que la red neuronal tenga que alcanzar con las entradas

proporcionadas. Una vez se tienen entradas y objetivos, se dividen todos los datos en 3 sets, un primer set de entrenamiento, que generalmente es más grande (70%), un segundo set de validación, que permite validar el entrenamiento y un tercer set de pruebas, que la red no va a usar en el entrenamiento. Solo se emplea para probar que el error no se dispara cuando aparecen entradas que la red no haya visto previamente.

Para entrenar la red NARX, se realizan ajustes en los pesos para responder correctamente a las entradas y acercarse lo más posible a las salidas objetivo, es decir, la red busca relaciones (en este caso no lineales) entre las entradas y las salidas. Cuantas más neuronas tenga la capa oculta, más complejos serán los problemas que podrá resolver. Además, también se pueden ajustar el número de retardos que tiene la entrada, lo que permite que, en cada uso, se tenga mayor o menor historial disponible para el entrenamiento. Por ejemplo, si se tienen veinte retardos, al pedir una predicción, la red dispondrá de la muestra actual y las 20 anteriores, lo que puede ayudar en el aprendizaje.

Hay que tener cuidado a la hora de entrenar una red, ya que se puede provocar un sobreentrenamiento de las mismas si no se supervisa cuidadosamente el entrenamiento, en la figura se puede observar la gráfica de entrenamiento de una red, se puede observar que hay un punto en el que la red ya no mejora de manera significativa, a partir de ese punto la red se está sobreentrenando y el síntoma de un sobreentrenamiento es que si el patrón de entradas-salidas cambia ligeramente respecto al de entrenamiento, la red puede dar salidas sin ningún tipo de sentido.

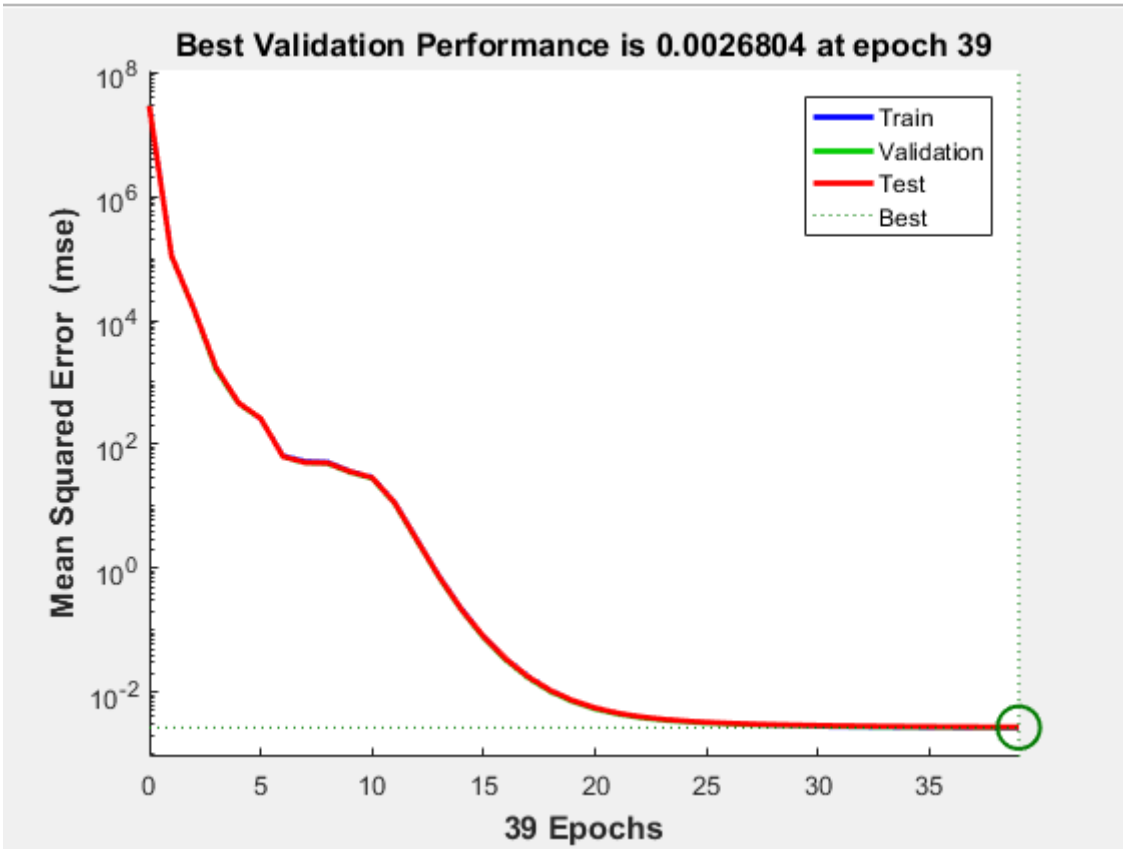


Figura 13. Sobreentrenamiento de una red

Las redes se entrenan en primer lugar en lazo abierto, introduciendo por la entrada de realimentación el valor real de la salida generada anterior, pero posteriormente, una vez se ha completado el entrenamiento de la red, el lazo debe cerrarse de tal manera que las salidas del instante anterior se vean reflejadas en la entrada del siguiente instante de tiempo. En la Figura 14 se muestra el esquema de entrenamiento de una red ejemplo.

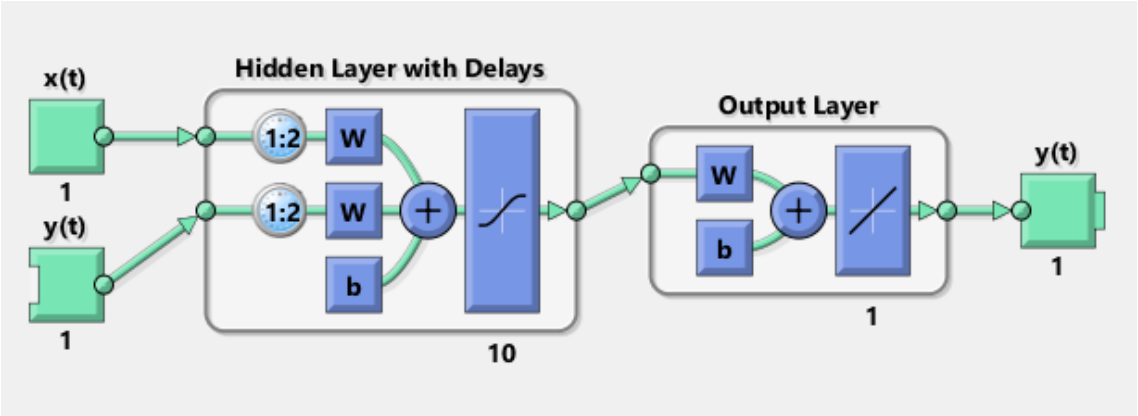


Figura 14. Esquema de entrenamiento de red NARX

Es interesante para entender ampliamente el funcionamiento de una red neuronal el libro referenciado en [22].

Una vez se han explicado los fundamentos utilizados para la solución, vamos a pasar a la explicación de la solución tomada paso por paso.

6.5 Adaptación de Algoritmo SSA

En este apartado se aborda la problemática encontrada a la hora de aplicar el algoritmo SSA a la predicción del consumo energético y las soluciones adoptadas.

En primer lugar, los datos son históricos del consumo de una región de Francia cuyo histórico es de 3 años. El primer problema encontrado es que, para hacer una predicción fiable, se han de tener en cuenta todas las componentes frecuenciales que pueda haber, incluidas las anuales. Conseguir separar las componentes anuales es posible con SSA, pero sería necesario tener varios años (Preferiblemente más de 3) para poder extraer esa componente, aun así, esto no garantizaría que no hubiese una componente con periodo superior al escogido la cual se estuviese ignorando.

Por esta razón, se ha adoptado la solución de realizar un SSA de “ventana”, esto significa que cada vez que se quiera realizar una predicción, se va a tomar del histórico los 365 días anteriores y se va a aplicar el algoritmo SSA a ese histórico, por lo tanto, la L elegida en todos los casos va a ser la misma, ya que no cambia el número de muestras de la serie temporal.

6.5.1 Detrending

El segundo punto a tener en cuenta es que antes de realizar el análisis SSA, se ha realizado un **"*Detrending*"** de los datos, es decir, se ha extraído una serie temporal auxiliar que sería la **"media móvil"** o **"tendencia"** de los datos. Esto es así porque si no se realiza un *detrending* antes de aplicar SSA, existe un autotriple que acapara prácticamente toda la potencia de la señal y no permite ver con claridad el resto de componentes. Para resolver este problema hay dos soluciones: realizar dos análisis, uno para extraer la tendencia y uno posterior para extraer el resto de componentes o realizar un detrending con otra herramienta y aplicar SSA. Se ha elegido la segunda opción porque además permite eliminar componentes de tipo anual que SSA no es capaz de detectar por falta de historial (aparecen en la tendencia).

Para realizar el Detrending se ha utilizado una herramienta que posee Matlab llamada "*Curve Fitting Tool*" que permite hacer ajustes de datos a curvas, lo cual nos facilita la implementación del código, la aplicación posee la siguiente interfaz, mostrada en la Figura 15:

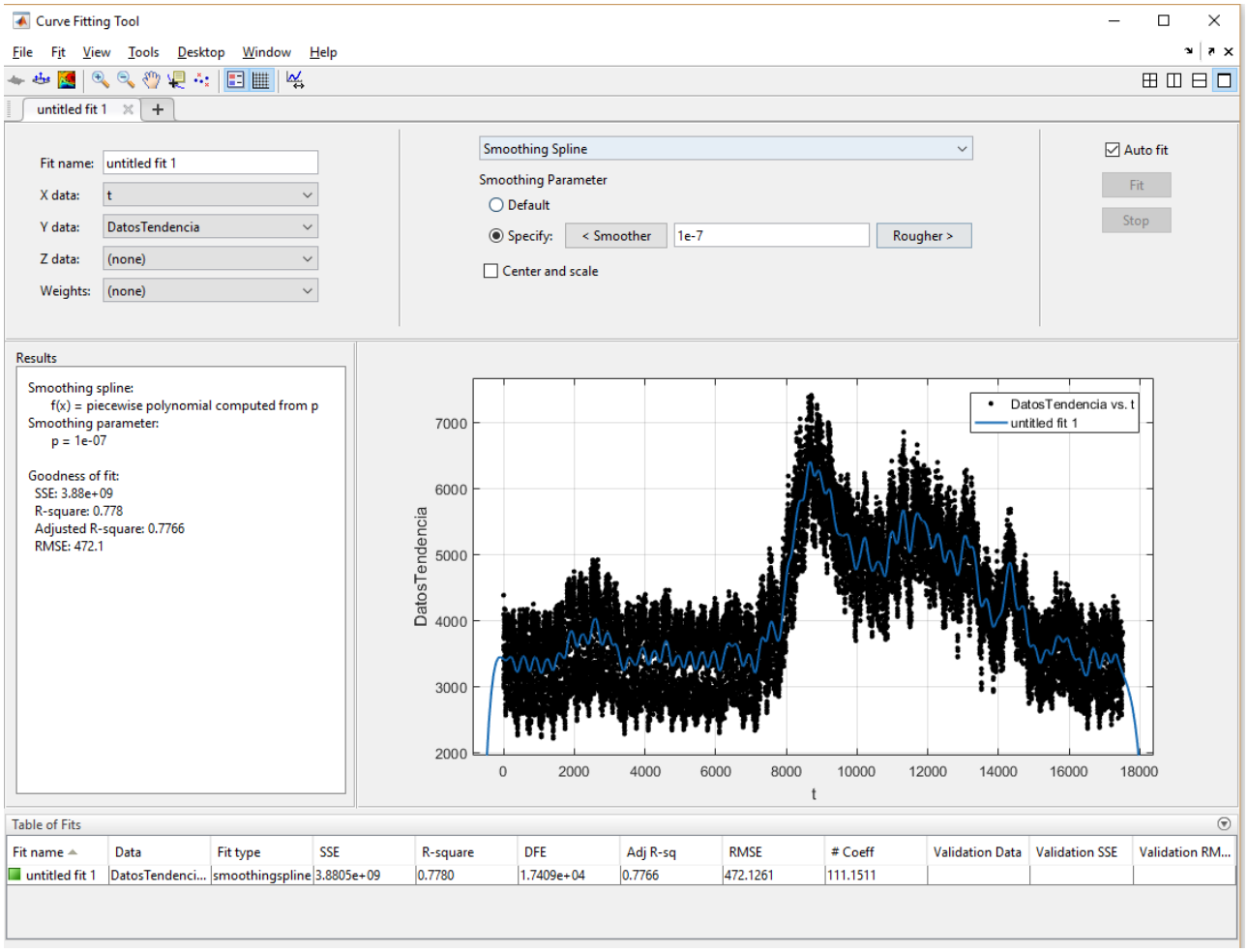


Figura 15. Curve Fitting Tool

La herramienta permite usar varios algoritmos de ajuste, entre ellos se encuentran:

- Ajuste polinómico: Consiste en un ajuste de los datos que se proporcionan mediante polinomios, la herramienta permite elegir el grado del polinomio con el que se quiere ajustar hasta un grado 9.
- Ajuste exponencial: Este algoritmo intenta ajustar los datos a una curva exponencial de dos términos.
- Suma de senos: Este algoritmo intenta ajustar los datos a una suma de senos de hasta 8 términos.
- Suavizado mediante SPLines: Este algoritmo consiste en conseguir una suma de polinomios entre los distintos puntos de los datos de entrada de tal manera de que cada una de las "porciones" sea diferenciable, lo que se consigue con esto es que el resultado sea "suave", es decir, no tiene picos ni cambios bruscos. Posee un único parámetro, el grado de suavizado, lo que permite que la SPLine siga en mayor o menor grado a los datos.

De las opciones expuestas se ha elegido el SPLine por su versatilidad, ya que permite regular el grado de ajuste que se quiere, que, en nuestro caso, ha de ser bajo, ya que solo queremos que siga la tendencia de la señal, no que la siga completamente.

Mediante experimentación, se ha elegido un parámetro de suavidad de valor 1E-7, que permite seguir la tendencia y evitar algunas componentes problemáticas para el SSA como por ejemplo las componentes anuales o de periodo amplio que no queden identificadas mediante la descomposición con un año de histórico. También se evitan los eventos especiales que provoquen cambios no habituales (Como por ejemplo una ola de calor que no suceda todos los años).

Para automatizar el proceso de cara a una aplicación, la herramienta permite generar código Matlab que realiza la función sin necesidad de acudir a la interfaz gráfica, la función sería la siguiente:

```
[Tendencia, info]=CrearSpline(Ejetiempos,Datos,SmoothingParameter)
```

En "tendencia" devuelve la SPLine generada y en "info" información sobre el error cometido en el ajuste.

Una vez se ha realizado el *detrending* de los datos, al histórico se le sustrae la tendencia calculada y esos datos son los que se procesan mediante el algoritmo SSA.

6.5.2 Presentación del IDE

Para desarrollar la implementación del algoritmo SSA se ha utilizado un lenguaje estadístico OpenSource llamado R y un IDE llamado RStudio [23] .

La elección de este lenguaje se explica por la existencia de una librería OpenSource de SSA, llamada RSSA [24]. Esta librería contiene funciones que permiten usar de manera intuitiva muchas de las herramientas anteriormente expuestas. A continuación, vamos

a explicar cómo se puede aplicar SSA sobre los datos sin tendencia usando las herramientas de la librería:

En primer lugar, esta es la pantalla que ofrece Rstudio (Figura 16):

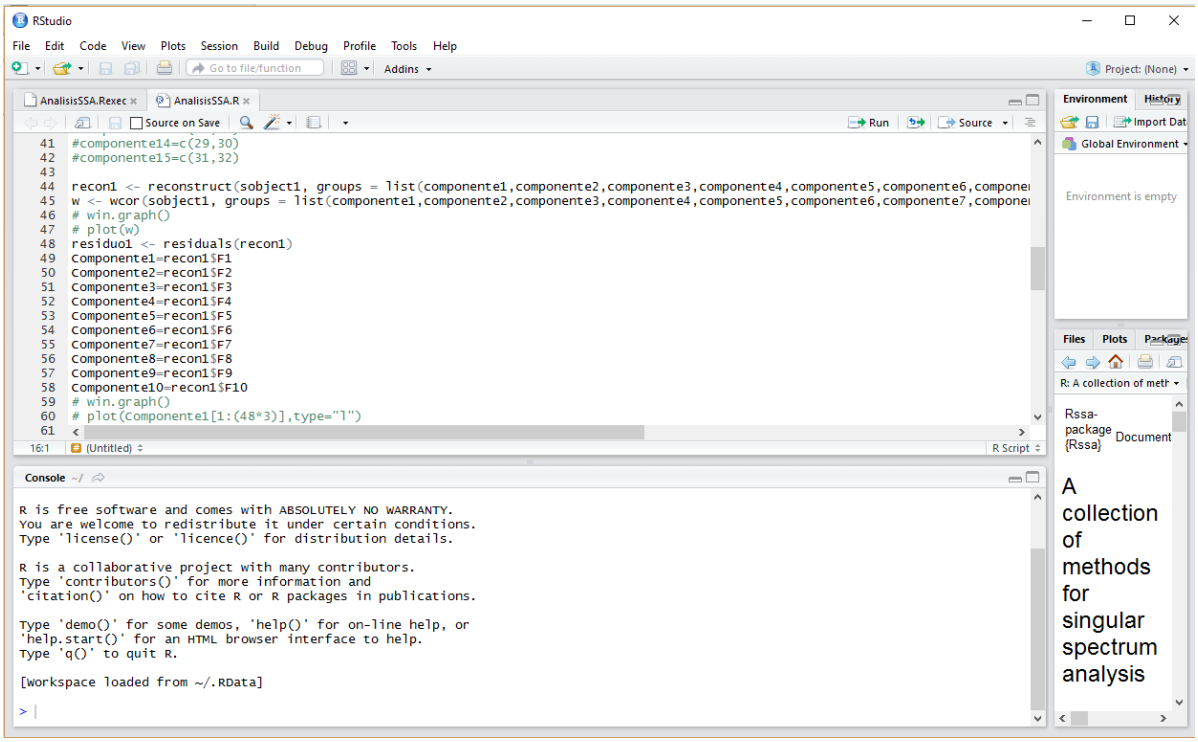


Figura 16. Interfaz de RStudio

El lenguaje es interpretados, por lo que se dispone de un editor de scripts, una ventana de variables y un visor de archivos.

Para aplicar SSA, el primer paso es cargar la librería con la orden:

```
library(Rssa)
```

6.5.3 Descomposición

Una vez cargada la librería y los datos a los que se va a aplicar, hay que definir la ventana que se va a utilizar para el algoritmo (L). Para nuestro caso, se ha elegido L=48*7*4, ya que es la que permite conseguir separar más eficazmente todas las componentes, al ser múltiplo de 48, 7 y 4. Es importante que sea múltiplo de 48 porque 48 muestras son las que hay en un día, 7 porque son los días de la semana (habrá componentes semanales) y 4 porque son las semanas que hay en un mes.

Una vez tenemos la L elegida, se llama a la función que se va a encargar de realizar el embedding y la descomposición. La función es la siguiente:

```
sobject1<-ssa (datosSSA, L=L)
```

Esta función acepta muchos parámetros como el número de autotriples a calcular, los datos que se han de procesar, el método de SVD, la longitud de ventana, etc.

Para nuestro caso particular, los parámetros por defecto nos valen exceptuando la longitud de ventana, que se especifica. El método de SVD es Nrutlan [25], que permite calcular el número de autovectores que se especifiquen, en nuestro caso, por defecto se calculan 50, que se consideran suficientes.

La función devuelve un objeto de la clase *SSA* que contiene información sobre la descomposición y la descomposición en sí misma.

Para poder ver el resultado de una descomposición, se puede recurrir a la función:

```
Summary(sobject1)
```

Que devuelve:

```
> summary(sobject1)

Call:
ssa(x = datosSSA, L = L)

Series length: 17520,   Window length: 1344,    SVD method: nutrlan
Special triples:  0

Computed:
Eigenvalues: 50,        Eigenvectors: 50,       Factor vectors: 0

Precached: 0 elementary series (0 MiB)

Overall memory consumption (estimate): 0.6491 MiB
```

Todos ellos parámetros ya comentados anteriormente.

6.5.4 Agrupación

A continuación, hay que agrupar los autotriples para formar grupos, que a su vez se traduzcan en componentes periódicas. Para ello, se puede recurrir a las herramientas que se explicaron con anterioridad (scatterplot, diagrama de correlación, valores de autovalores, reconstrucciones, etc.). Para ello, vamos a explicar cómo acceder a cada una de estas herramientas con un ejemplo de cada una (Los datos son del historial de un año cogido desde el 1 de enero de 2014 al 31 de Diciembre de 2014.

- *Scatterplot*:

Para poder dibujar el *scatterplot*, se tiene que haber realizado la descomposición y llamar a las siguientes funciones:

```
win.graph()
plot(sobject1,type="paired")
```

El resultado obtenido se muestra en la Figura 17:

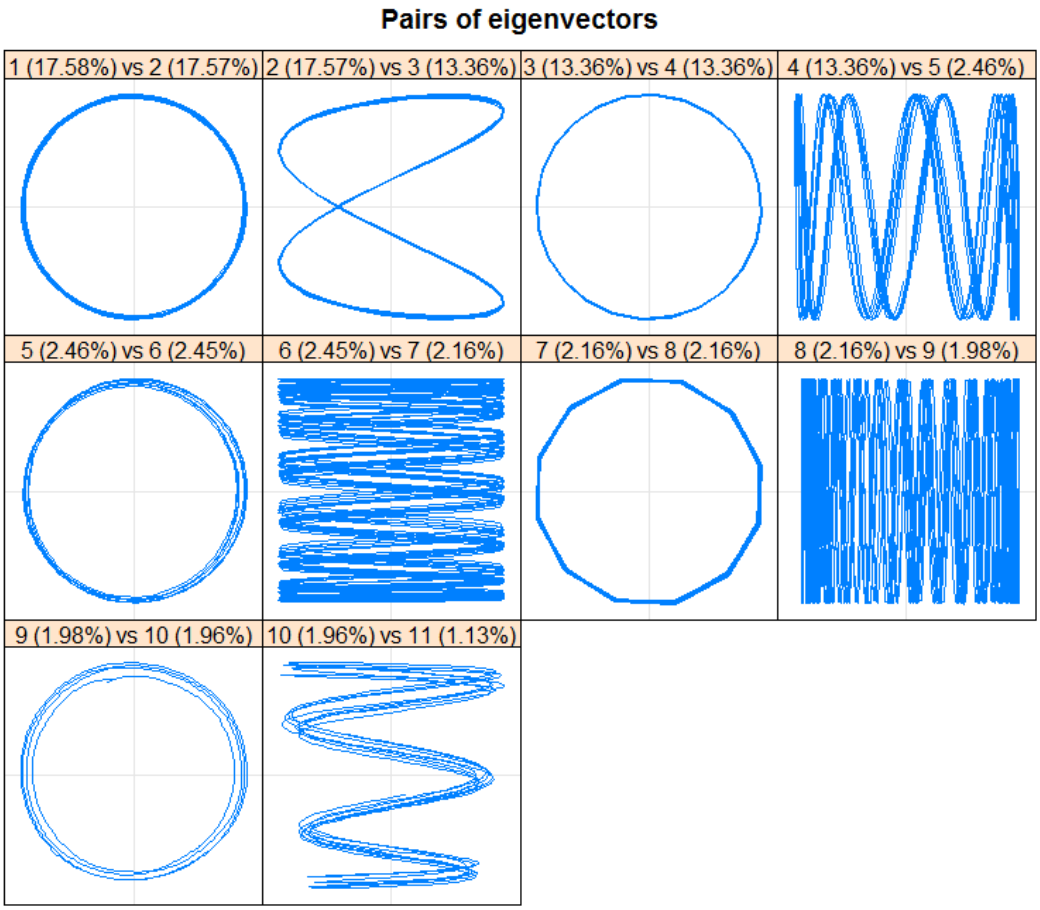


Figura 17. Scatterplot

A la función se le puede especificar el rango de autovectores de la descomposición que se ha realizado anteriormente a incluir en la representación y además, si se quieren representar agrupados de dos en dos o de tres en tres.

Es importante llamar a win.graph() para generar una ventana de gráfico.

- Valores de autovalores:

Para generar esta ventana, muy útil para saber la importancia de un cierto autotriple y discernir entre residuo e información útil, se llama a la siguiente función:

```
win.graph()
plot(sobject1,type="values")
```

Con el siguiente resultado (Figura 18):

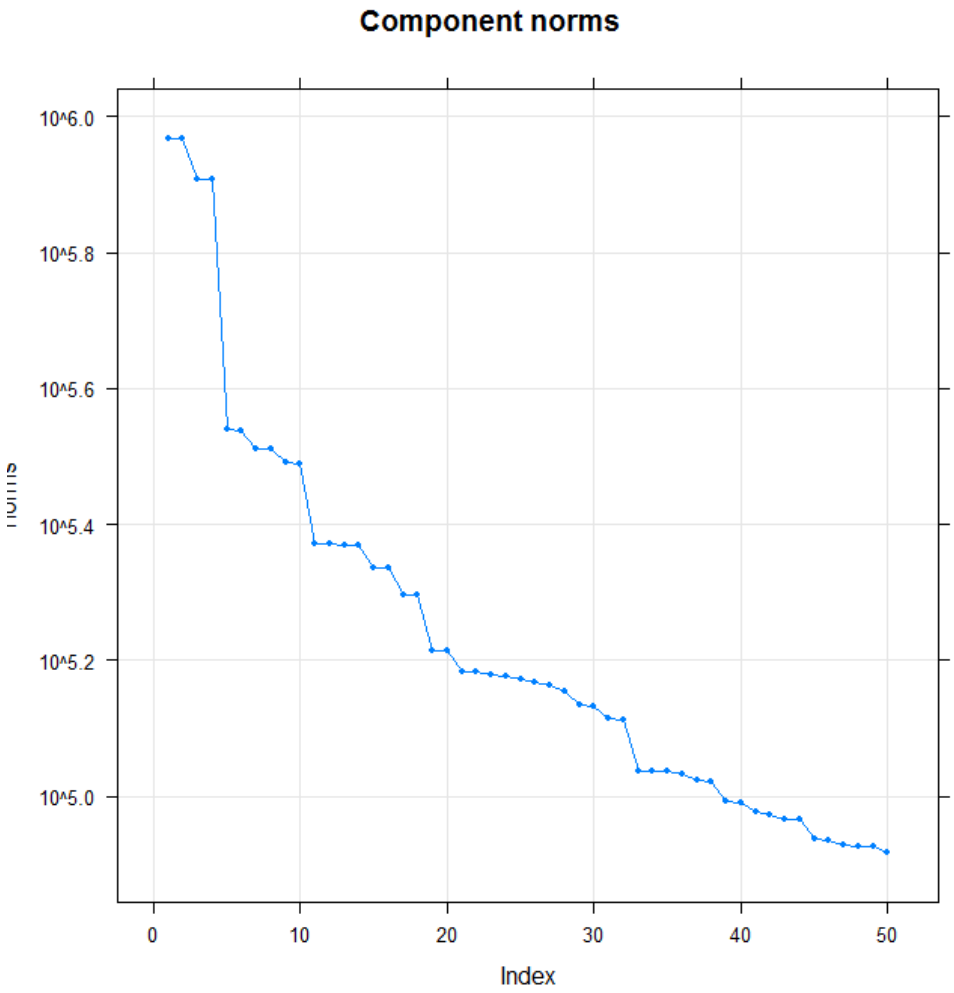


Figura 18. Representación de autovalores

Como en el caso anterior, se le puede indicar cuantos autovalores representar.

- Diagrama de correlación:

Esta es la representación más importante, ya que es la más intuitiva para representar los autotriples que tienen que ir unidos. A partir de este esquema se han elegido los grupos que llevan a definir las componentes a predecir. Se llama con la siguiente función:

```
win.graph()
plot(sobject1,type="wcor")
```

Obteniendo el resultado de la Figura 19:

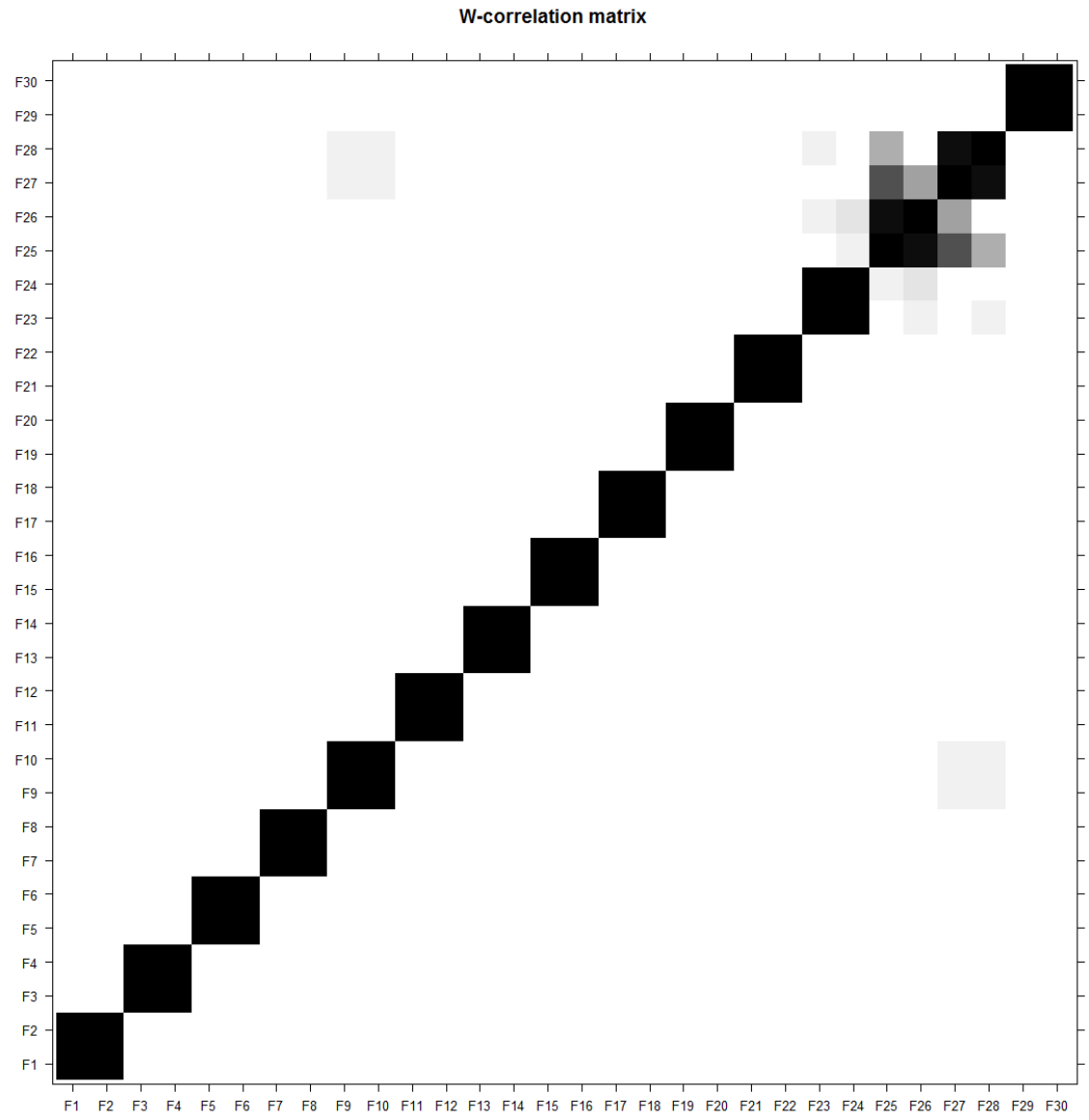


Figura 19.Diagrama de correlación

Como se puede observar en la figura, las componentes se agrupan limpiamente en 10 grupos de dos en dos, hasta que llega un momento en el que se empiezan a correlacionar unas con otras. Esto se repite cuando se eligen otros sets de datos, es decir, cuando el SSA se aplica en otro momento temporal, pero se mantiene el historial de un año.

Esto es muy importante, ya que es la base para poder aplicar el algoritmo de forma automática, si no se cumpliese, cada vez que se aplicase el algoritmo habría que rehacer los grupos (de hecho, si esto se hiciese, el resultado sería más exacto). Dado que el objetivo es conseguir un sistema que no requiera la intervención del usuario, se ha decidido mantener los grupos independientemente de cual sea la fecha elegida para la predicción.

Por lo tanto, se definen los grupos creando un vector para cada grupo con el índice del autotriple:

```
componente1=c(1,2)
componente2=c(3,4)
componente3=c(5,6)
componente4=c(7,8)
componente5=c(9,10)
componente6=c(11,12)
componente7=c(13,14)
componente8=c(15,16)
componente9=c(17,18)
componente10=c(19,20)
```

6.5.5 Reconstrucción

A continuación, para realizar la reconstrucción se llama a la función *reconstruct* con los siguientes parámetros:

```
recon1 <- reconstruct(sobject1,
groups =list(componente1,componente2,componente3,componente4,
componente5,componente6,componente7,componente8,componente9,
componente10))
```

Como primer parámetro se le pasa el objeto donde está la descomposición SSA que se ha hecho previamente; como segundo parámetro, se le pasa una lista con todos los grupos que se han definido. La salida de la función es un objeto de la clase *reconstruct* que contiene todas las series temporales reconstruidas.

Por otro lado, al hacer una reconstrucción sin usar todos los autotriples, es inevitable que quede un residuo de la serie que no ha sido reconstruido, este residuo puede ser extraído mediante la siguiente función:

```
residuo1 <- residuals(recon1)
```

En nuestro caso, es muy importante obtener el residuo porque lo vamos a utilizar para hacer una estimación del mismo mediante una red neuronal del tipo NARX, como se explicará posteriormente.

Como se comentó anteriormente, tras realizar la reconstrucción de los grupos, hay que comprobar que no hay correlación entre ellos, para ello, hay que calcular las correlaciones y dibujarlas. Se puede realizar con las siguientes funciones:

```
w <- wcor(sobject1,
groups =list(componente1,componente2,componente3,componente4,
componente5,componente6,componente7,componente8,componente9,
componente10))
win.graph()
```

```
plot(w)
```

Esta secuencia de código, devuelve el siguiente gráfico (Figura 20):

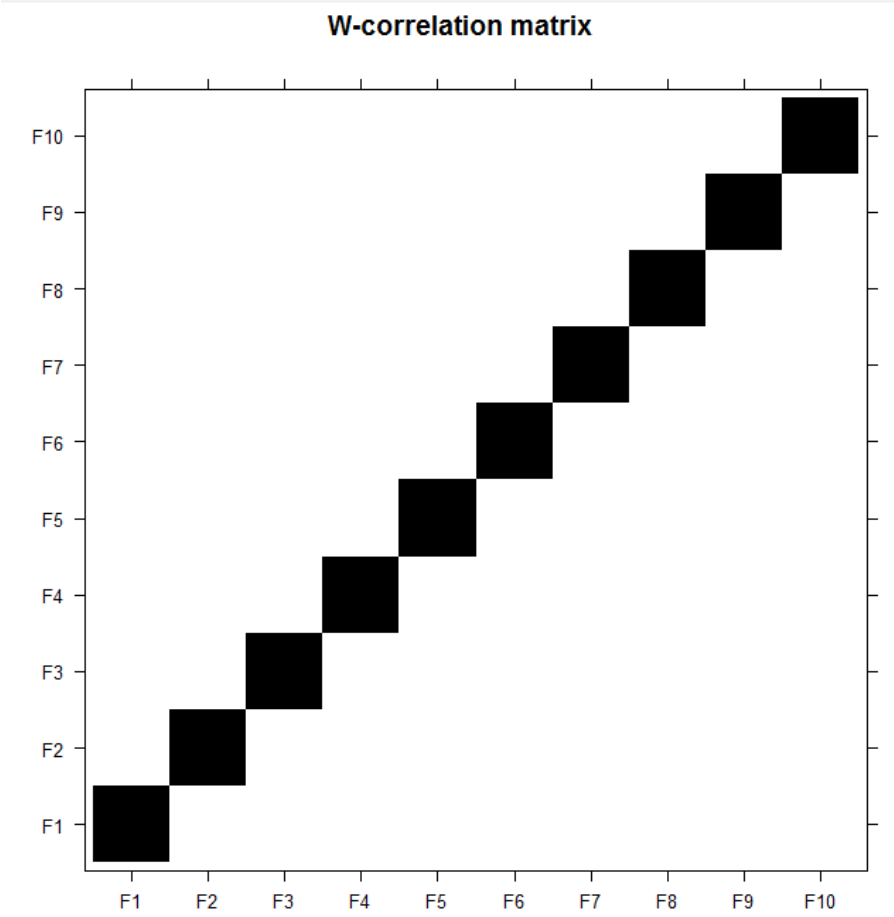


Figura 20. Correlación entre grupos

Donde cada F representa cada una de las reconstrucciones realizadas. Se puede observar que todos los grupos son independientes entre sí, lo que indica que la serie es separable y que los autotriples se han reunido de manera correcta (Se han agrupado aquellos que estaban correlacionados sin dejar en otros grupos autotriples también correlacionados).

6.5.6 Predicción

Finalmente, una vez se tienen las componentes, como se quiere predecir el día siguiente, hay que realizar una predicción de 48 muestras con la siguiente función:

```
PredSSA<-forecast(sobject1,
groups =list(c(componente1,componente2,componente3,componente4,componente5,
componente6,componente7,componente8,componente9,componente10)),
method = "recurrent",
len = 48)
```

Como se puede observar, la función tiene 4 argumentos (puede tener más, pero no son relevantes). El primer argumento es el objeto de la descomposición, el segundo argumento es la lista de grupos que se quieren utilizar para realizar la predicción, el tercero y más importante, es el método que se quiere utilizar para realizar la predicción. Hay 4 métodos: recurrente [26], vectorial [27], bootstrap-recurrente [28] y bootstrap-vectorial [28]. Se ha elegido entre ellos el recurrente, ya que al realizar diferentes pruebas, se llegó a la conclusión de que en el caso que está bajo estudio, existe una ecuación linear que permite la predicción de las componentes elegidas. Además, en el caso de las dos opciones con Bootstrap, como no se han elegido más de 20 componentes, el algoritmo no converge y da un error, por lo que la decisión estaba entre la predicción vectorial y la predicción por componentes, la diferencia no era apreciable y en tiempo de cómputo era más ventajosa la opción de recurrente.

La predicción se basa en la búsqueda de una fórmula recurrente lineal o LRF, que, si la separación está bien hecha, debería existir. Esta búsqueda puede llevar mucho tiempo.

Una vez que se ha realizado la predicción, esta se almacena en el objeto de salida.

Finalmente, se guardan las variables de interés como las componentes reconstruidas, la predicción de las mismas y los residuos en formato CSV para que puedan ser cargadas por la siguiente fase, en este caso desarrollada en Matlab. La explicación del procesamiento SSA concluye aquí, aunque más adelante se explicará qué se hace con la tendencia y qué se hace con los residuos.

Algún ejemplo de cómo predecir mediante SSA puede ser encontrado en [29] y [30].

6.6 Predicción mediante Redes Neuronales

En una tercera etapa, se importan los resultados obtenidos del análisis SSA como las componentes, la predicción de las componentes y los residuos. Además, se dispone de la tendencia del histórico que se ha extraído en la primera etapa.

6.6.1 Predicción de la Tendencia

Como la tendencia no se ha podido predecir mediante el algoritmo SSA, se va a utilizar una red neuronal para hacerlo. Para ello, se ha preparado un set de datos con el primer año de historial, se ha extraído su tendencia y con ellos se ha entrenado la red neuronal.

Mediante observación directa de la tendencia, se puede observar que depende de la época del año en la que se esté. Además, se puede observar que de un día a otro no tiene grandes cambios, por lo que se decide que las entradas a la red neuronal van a ser 2, por un lado, se le va a pasar el día del año en el que se encuentra mediante un seno con periodo de 365 días, y por otro se le va a pasar la muestra de la tendencia del día anterior a la misma hora. En cuanto a los retrasos, se decide poner solo dos, ya que la tendencia no cambia significativamente de un día para otro.

Por otro lado, el número de neuronas para la capa oculta es de 30, se eligió este número de neuronas después de un procedimiento experimental consistente en realizar entrenamientos a distintas configuraciones para la red, se fue entrenando redes cambiando las neuronas ocultas de 5 en 5 ascendentemente hasta que se observó que a partir de 30 neuronas la mejora en la salida no era significativa y no justificaba el tiempo de entrenamiento que crecía de manera exponencial.

El esquema de entrenamiento es el siguiente (Figura 21):

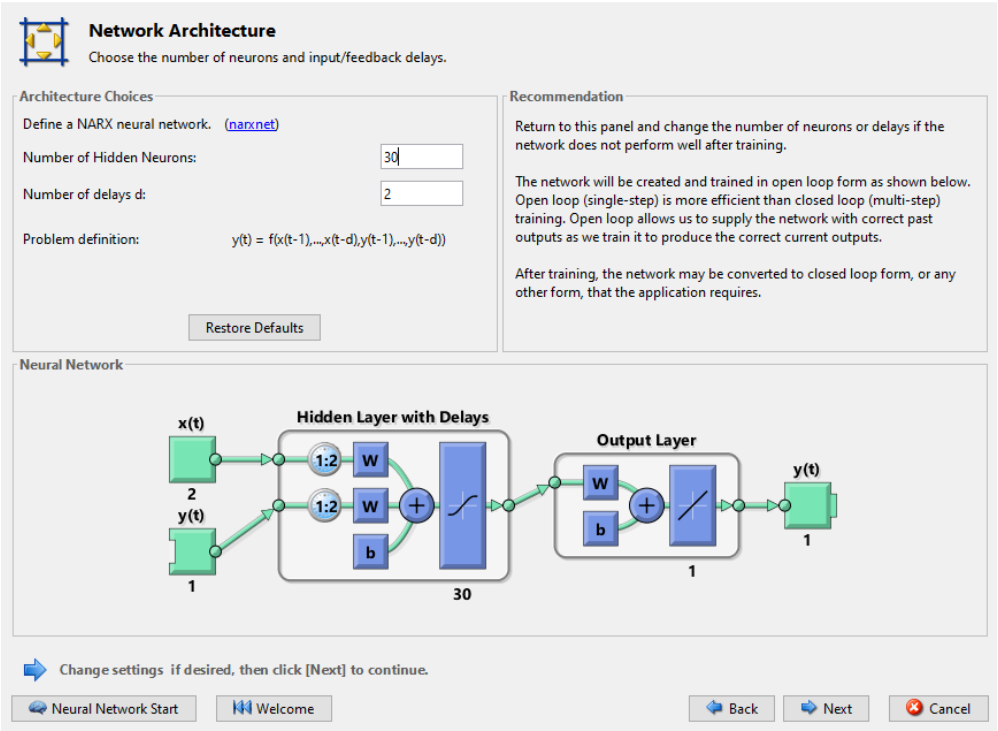


Figura 21.Esquema de entrenamiento NARX

Como se puede observar en la Figura 21, el entrenamiento se realiza en lazo abierto.

Es importante puntualizar, que del historial de un año se ha reservado un tercio para pruebas posteriores sobre la red (aparte del 15% que reserva la herramienta por si sola).

El entrenamiento se realiza mediante el algoritmo Lavenberg-Marquardt (Figura 22):

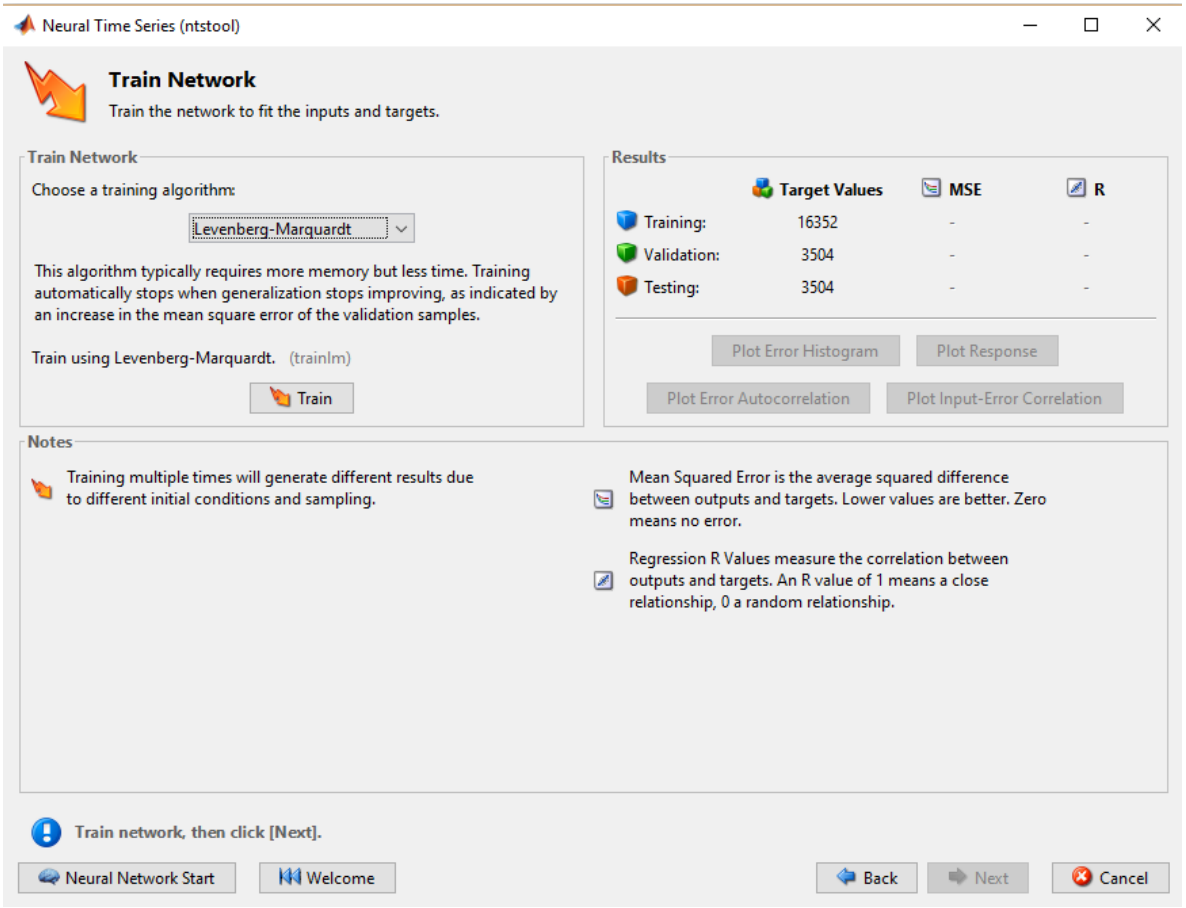


Figura 22. Interfaz algoritmo de entrenamiento

Tras 39 iteraciones, se obtiene el resultado de la red en la Figura 23.

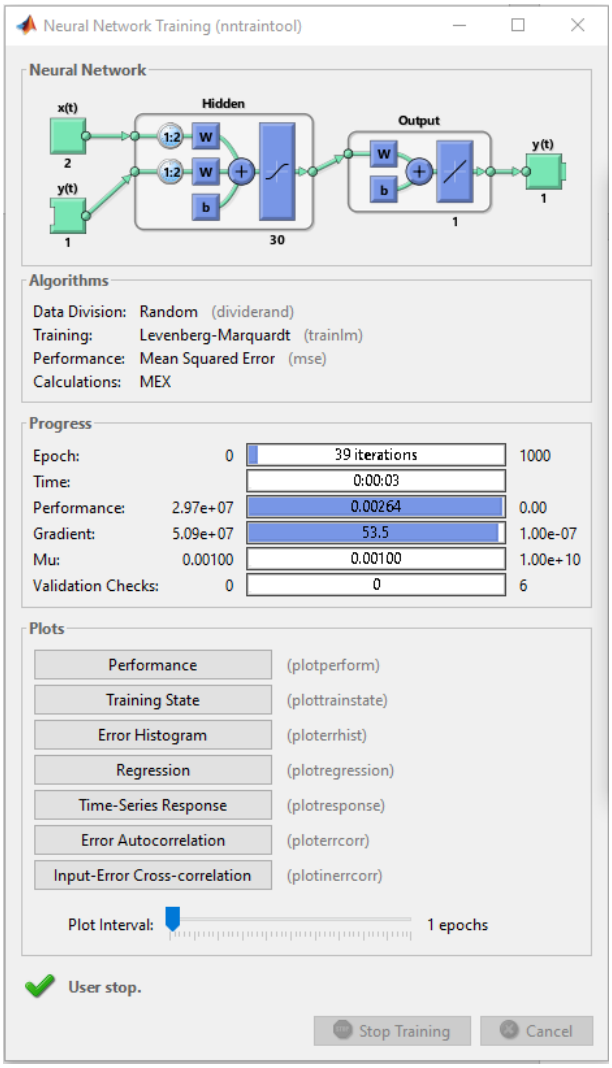


Figura 23. Interfaz de entrenamiento en curso

Como se puede observar, 39 iteraciones han llevado tan solo 3 segundos, lo que es extremadamente rápido, hay que estar muy atento al gráfico de eficiencia del entrenamiento para evitar sobreentrenar la red.

El resultado del entrenamiento de la red, se muestra en el gráfico de la Figura 24:

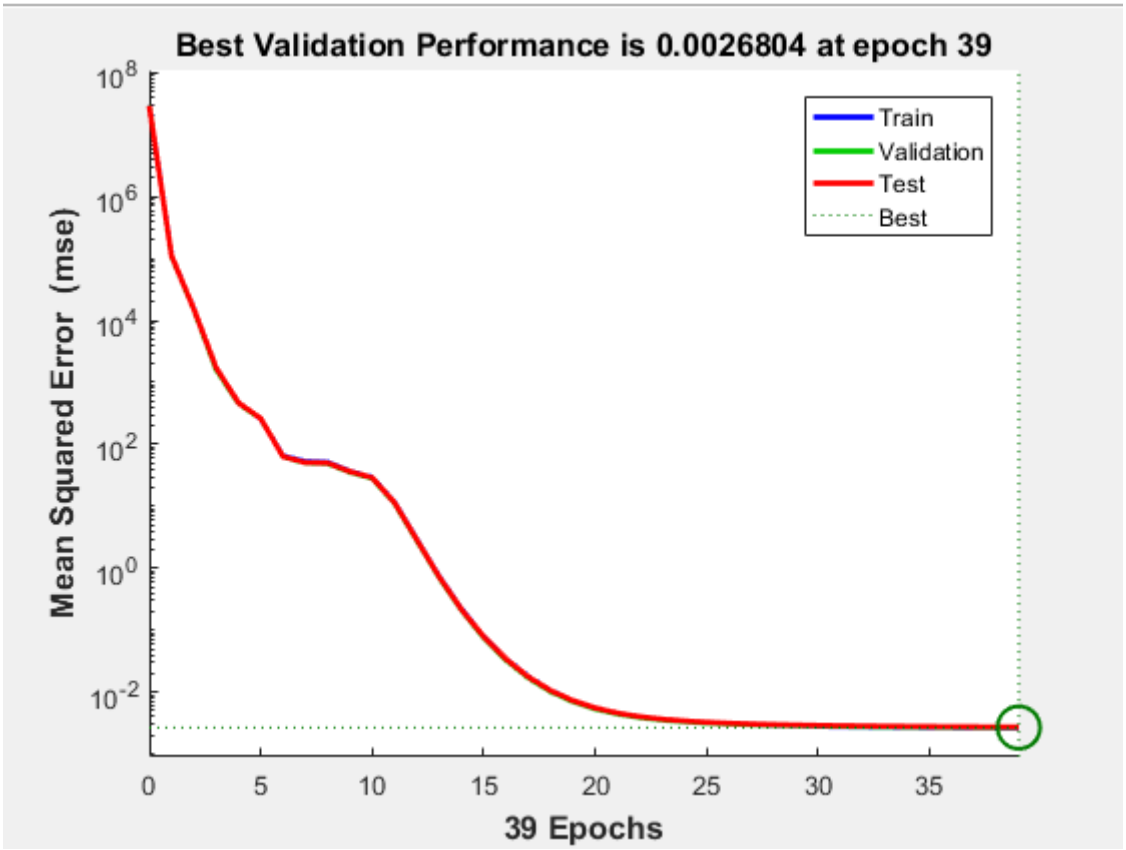


Figura 24. Eficiencia de entrenamiento

Como se puede observar, en la iteración 39 el error de entrenamiento se estanca. La red puede seguir entrenándose, pero se provocaría sobreentrenamiento, en cuyo caso la red no es capaz de responder acertadamente a casos distintos a los entrenados.

Finalmente, se prueba la red con el set de datos reservados para pruebas, para así poder ver su desempeño. La Figura 25 muestra el resultado de estas pruebas:

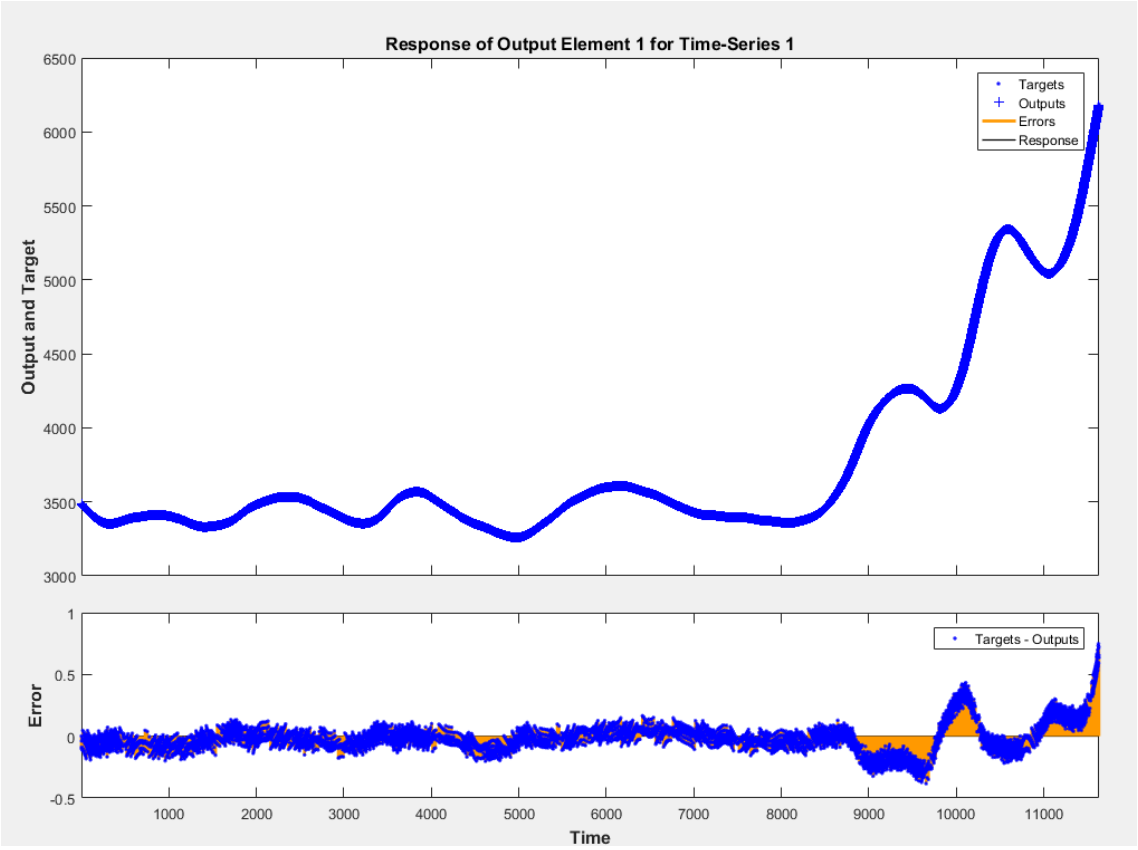


Figura 25. Simulación de la red

Como se puede observar, la red comete un error máximo de 1 frente a 6000, por lo que concluimos que es un resultado más que aceptable.

Esta es la explicación detallada de como se ha entrenado la red de tendencia, esta red necesita ser reentrenada con nuevos historiales cada cierto tiempo (3-4 años) ya que la escala puede cambiar y las tendencias de consumo podrían llegar a cambiar de forma drástica.

6.6.2 Predicción del Residuo

En una primera versión de la solución aquí expuesta, se planteó la posibilidad de no predecir el residuo que se extraía del análisis SSA, pero se desechó esa idea porque el residuo es lo suficientemente importante como para tener una influencia significativa en la predicción, si se quiere que esta se ajuste a la realidad.

Por ello, se plantea la posibilidad de utilizar otra red neuronal NARX para la predicción del residuo, mucho más compleja que la utilizada para la tendencia, ya que el conjunto de datos contiene mucho más ruido que en el caso de la tendencia, además, las relaciones que hay que establecer son mucho más complejas.

Para ayudar a la red a calcular la salida, en este caso se ha decidido dotar a la red de 96 retardos (lo que equivale a dos días de muestras anteriores, ya que cada día son 48

muestras, una cada media hora). Gracias a estos retardos, la red puede saber lo que ha ocurrido en los dos días atrás.

Además, recibe como entradas el residuo del día anterior a esa misma hora, la hora del día, el día de la semana y el día del año en el que se encuentra, para ayudar a la red a ubicarse.

Finalmente, se ha dotado a la capa oculta de 40 neuronas, como se puede observar, el número es superior que en el caso anterior, esto se debe a que los datos son más ruidosos, por lo que la relación entrada-salida es más compleja. Se siguió el mismo método que en el caso anterior, se fue aumentando el número de neuronas hasta que se llegó a un punto en el que, aunque se aumentase el número de neuronas, los resultados de la red no mejoraban, además de que se aumentaba el tiempo de entrenamiento. La estructura de la red es la mostrada en la Figura 26:

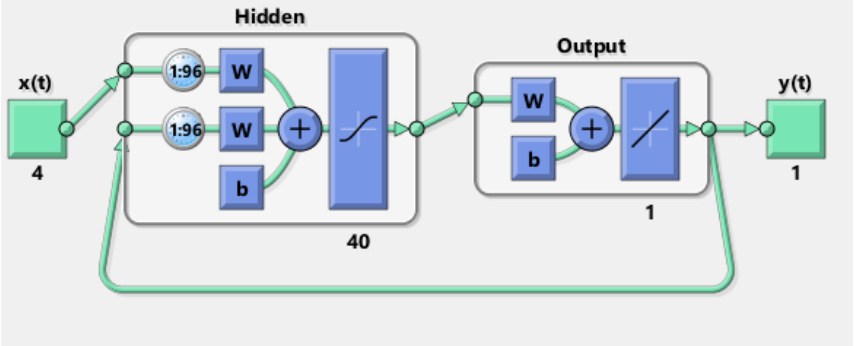


Figura 26. Esquema de NARX realimentada

El proceso de entrenamiento y pruebas de la red es el mismo que en el caso anterior, por lo que no se explicará el proceso. Aun así, mostraremos las figuras con los resultados obtenidos. En la Figura 27 muestra la eficiencia del entrenamiento y como al final de la iteración 11 ya se estanca el aprendizaje de la red.

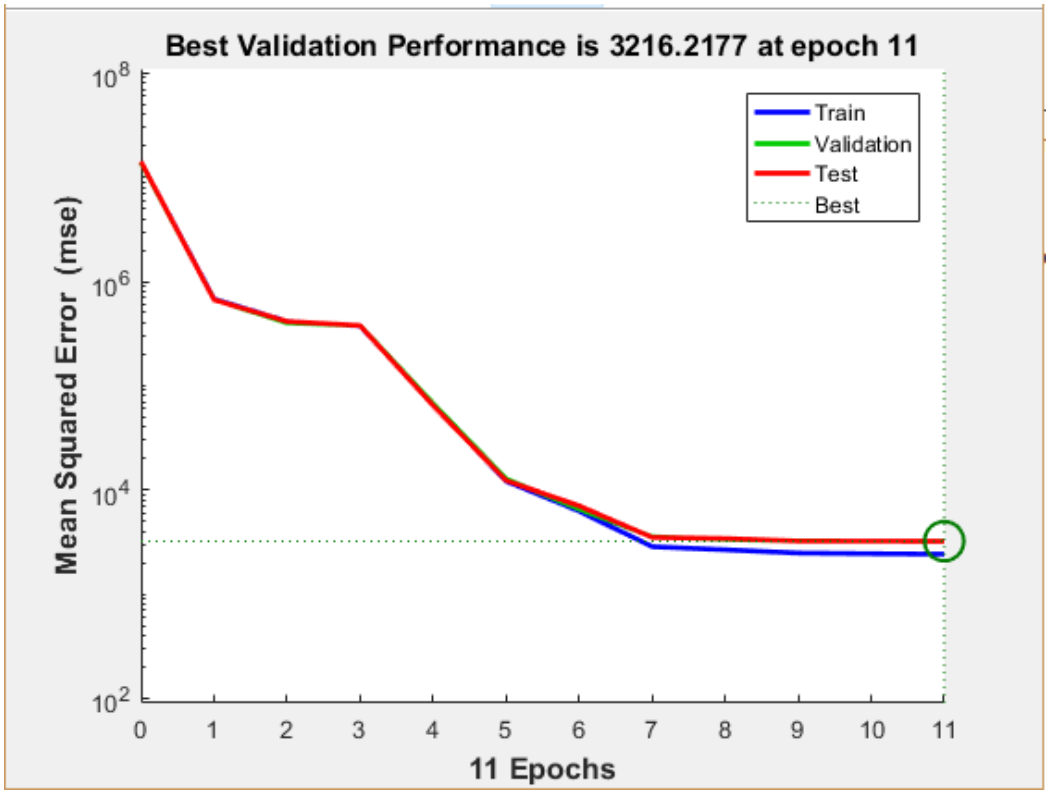


Figura 27. Eficiencia de entrenamiento de Red de Residuos

En la Figura 28 se muestra el resultado de probar con el set de datos de pruebas, en la gráfica de encima es la respuesta y debajo se muestra el error de cada una de las muestras.

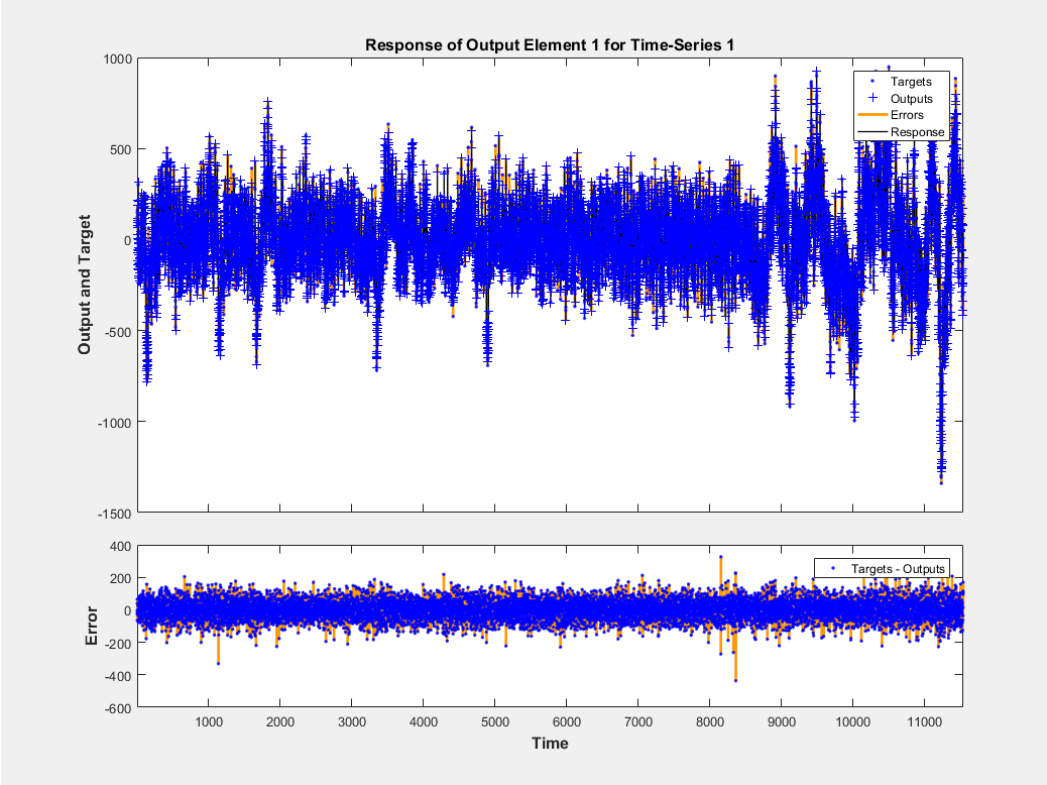


Figura 28. Simulación Red de Residuos

A continuación, como no se aprecia demasiado bien los errores que se cometen, se ha dibujado en la Figura 29 el histograma de errores cometidos por la red en la simulación de la Figura 28.

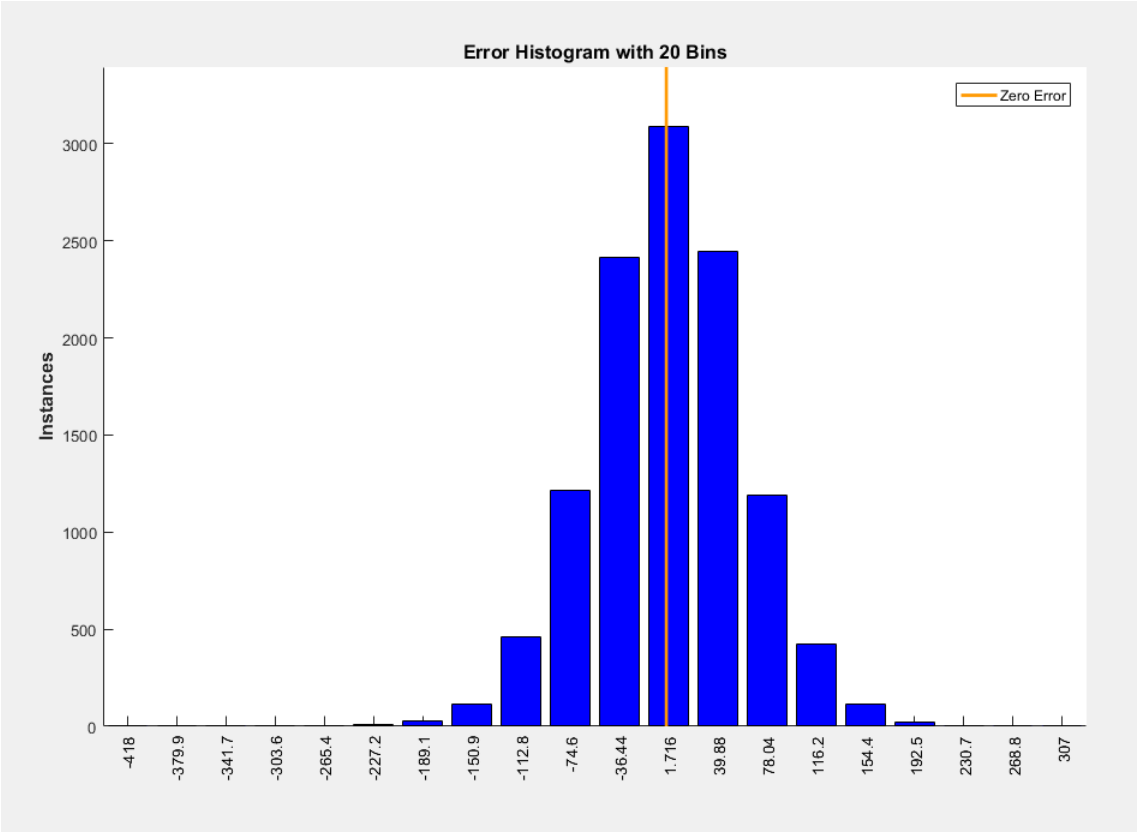


Figura 29. Histograma de errores de Red de Residuos

Como se puede observar, la red comete en la mayoría de las veces error 0 o cercano a él, por lo que se puede afirmar que la red está bien entrenada.

Finalmente, se pone como ejemplo de la complejidad de entrenamiento de la red el estado final de la herramienta cuando se ha terminado en la Figura, se puede observar que el entrenamiento se ha demorado 47 minutos solo para hacer 11 iteraciones.

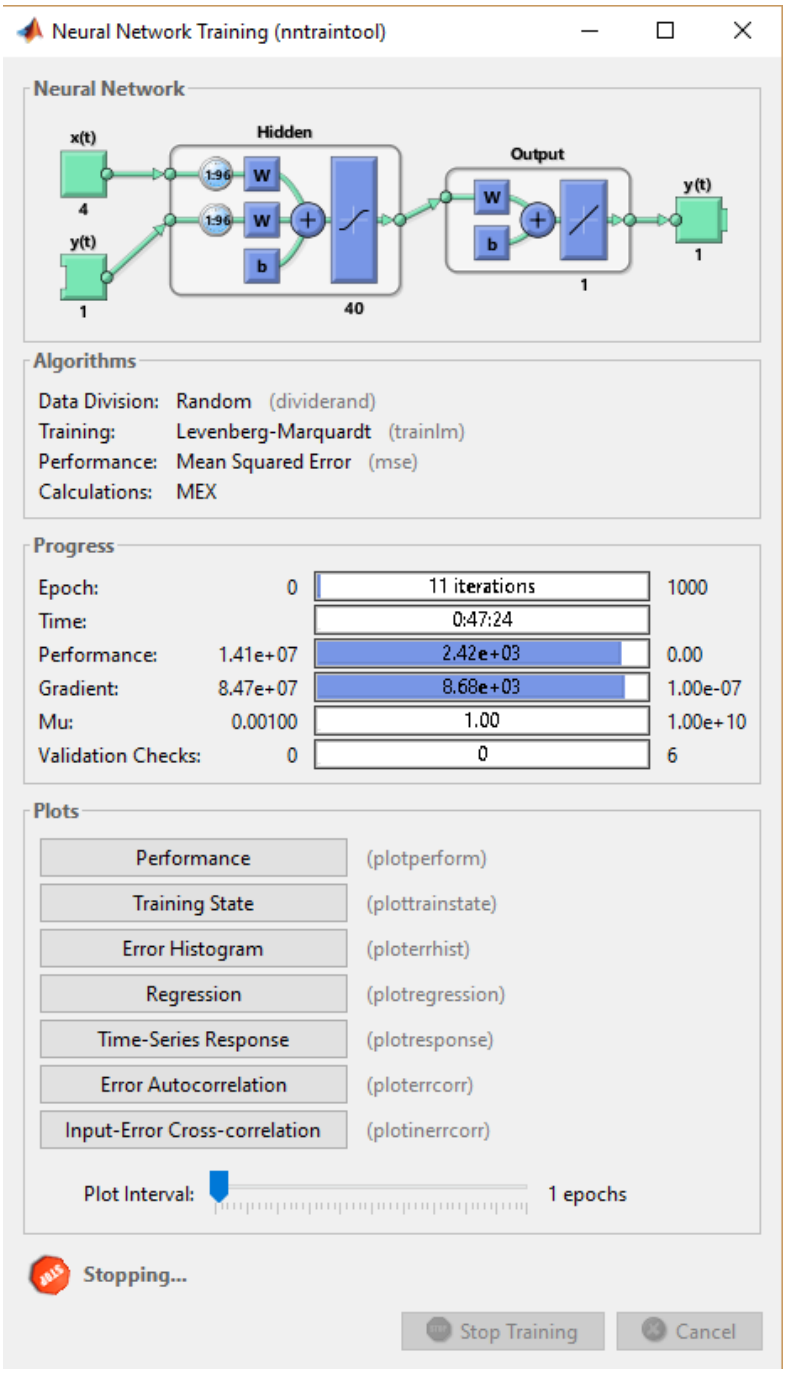


Figura 30. Herramienta de entrenamiento de redes: Red de Residuos

6.6.3 Predicción de las Componentes

Como se indicó anteriormente, las componentes extraídas con el algoritmo SSA pueden ser predichas mediante el método de fórmula lineal recurrente. No obstante también se planteó la posibilidad de que podrían ser predichas mediante redes neuronales. Por tanto se ha entrenado una red para cada una de las 10 componentes elegidas. El proceso es similar a los dos casos anteriores, todas ellas tienen en común que se han utilizado solo 2 retardos (se trata de casos sencillos) y se han usado 20 neuronas en la capa oculta, el método para determinar el número de neuronas de la capa oculta es el mismo empleado anteriormente, los retardos se dejaron al mínimo ya que no son necesarios, ya que una componente periódica es fácilmente predecible solo con las dos muestras anteriores debido a su periodicidad.

Las entradas dependen de cada componente, para decidir qué entrada se tiene que utilizar se observó la componente y su periodo, si el periodo es inferior a un día de duración, se le pasa la hora del día, si el periodo es similar al de una semana, se le pasa el día de la semana. No hay componentes anuales, ya que son absorbidas por la tendencia.

7. COMPARACIÓN DE MÉTODOS DE PREDICCIÓN Y RESULTADOS

Una vez explicadas las herramientas de predicción, vamos a exponer los diferentes casos de uso de predicción del día siguiente al elegido, principalmente son:

- **CASO 1:** Predicción mediante SSA de Componentes Periódicas + Predicción de Tendencia mediante Redes Neuronales. En la Figura 31 se muestra el esquema utilizado.

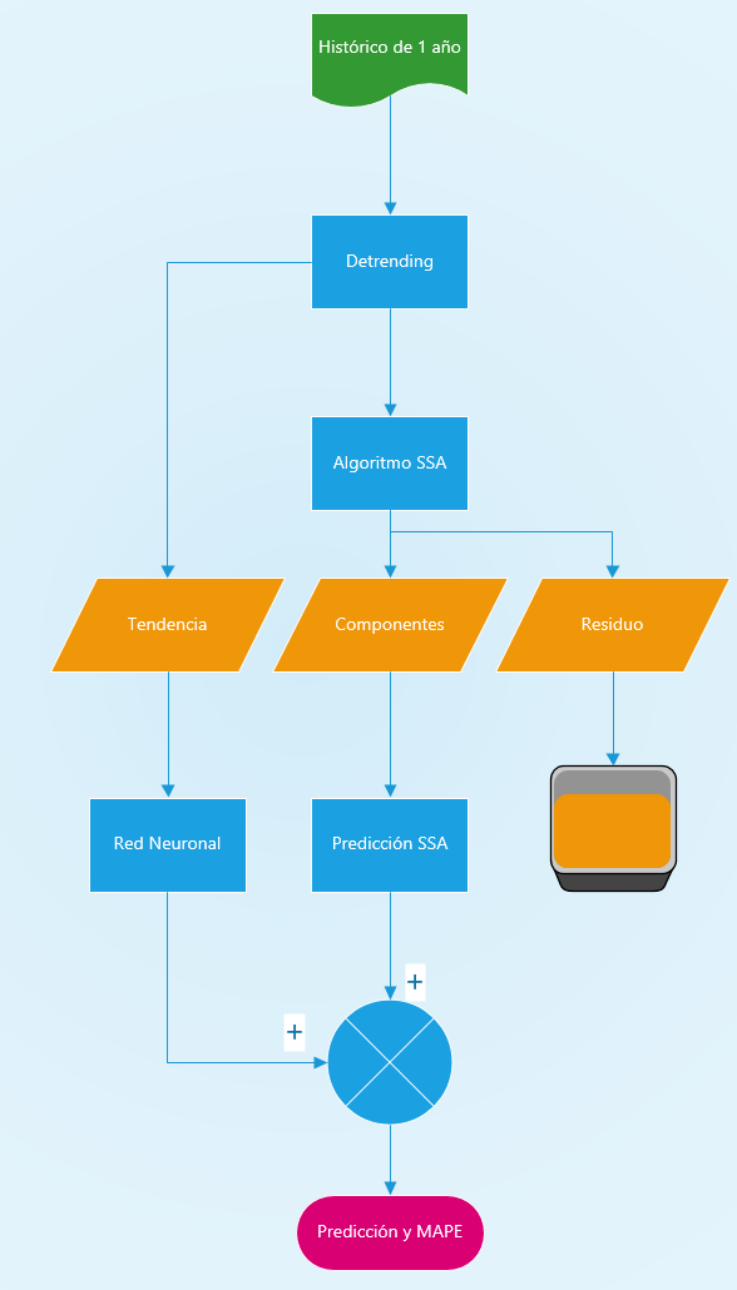


Figura 31. Esquema Caso 1

- **CASO 2:** Predicción mediante Redes Neuronales de Componentes Periódicas + Predicción de Tendencia mediante Redes Neuronales. En la Figura 32 se muestra el esquema utilizado. Este caso se diferencia del CASO 1 en que las Componentes periódicas se predicen mediante redes neuronales en vez de hacerlo mediante la fórmula recurrente de SSA.

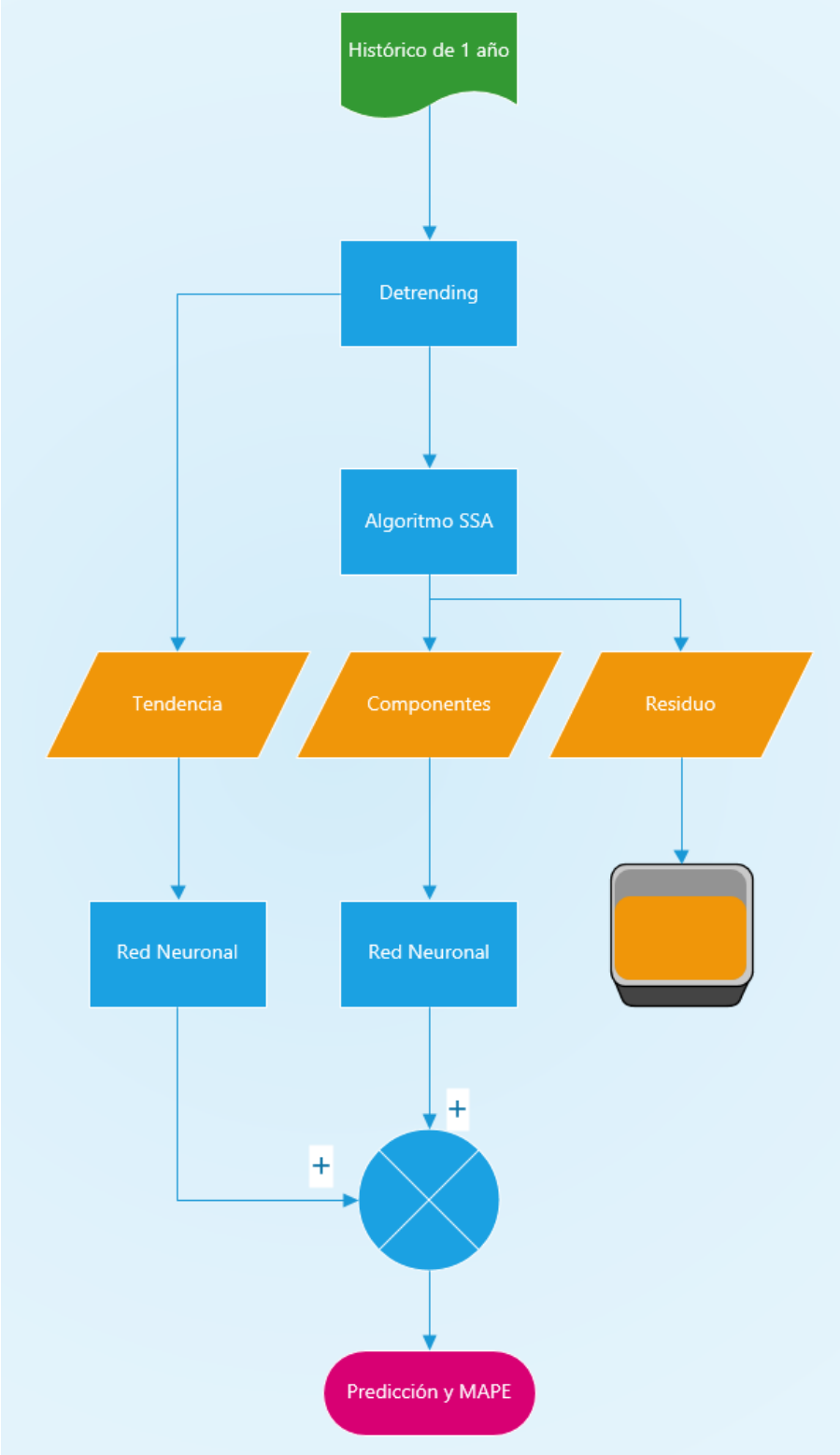


Figura 32. Esquema Caso 2

- **CASO 3:** Predicción mediante SSA de Componentes Periódicas + Predicción de Tendencia mediante Redes Neuronales + Predicción de Residuos mediante Redes Neuronales. En la Figura 33 se muestra el esquema utilizado. Este caso es igual que el CASO 1 pero se le añade la predicción del residuo, en vez de desecharlo como en el CASO 1, esto añadirá previsiblemente más precisión a la predicción si suponemos que hay información en los residuos.

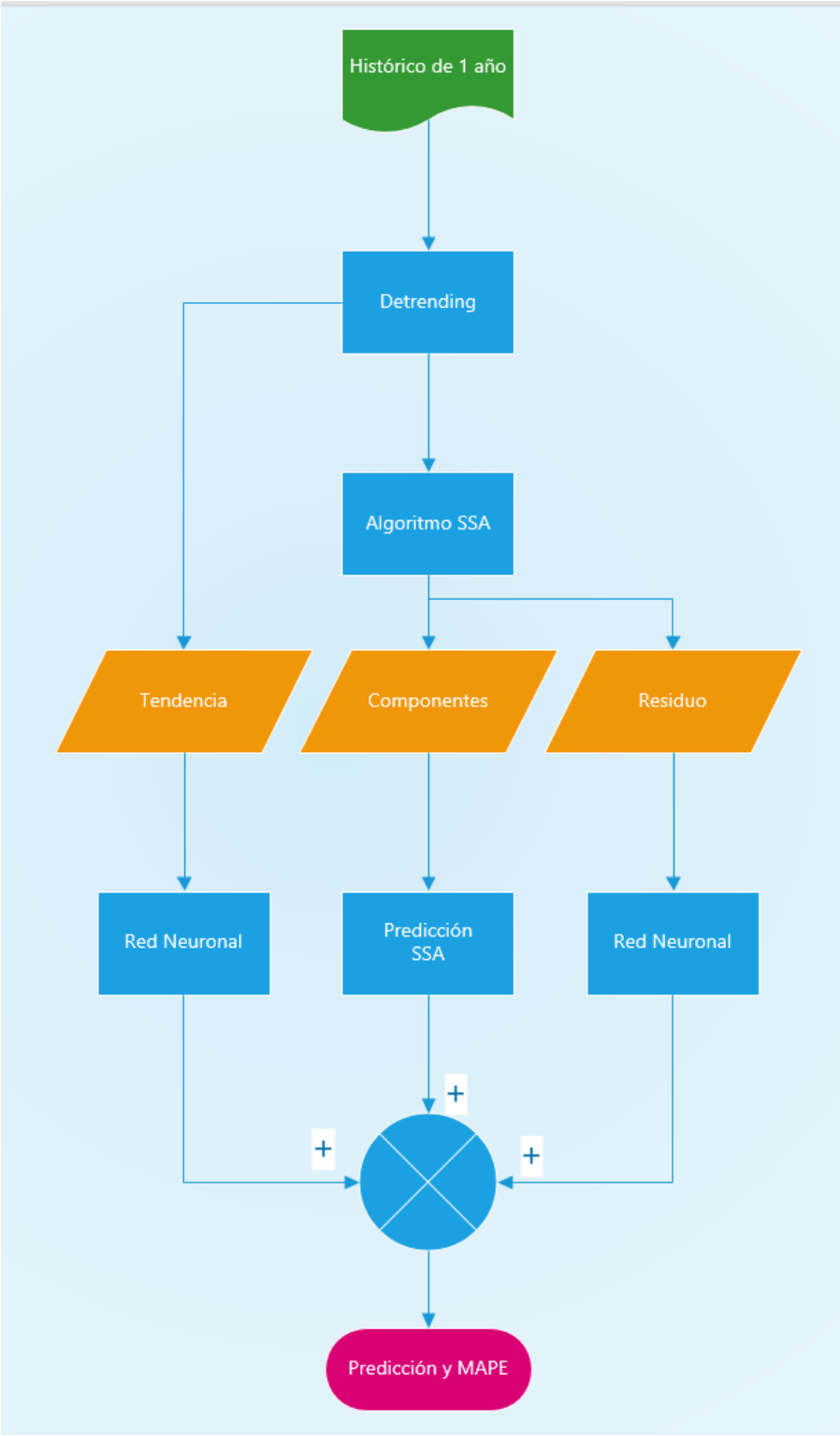


Figura 33. Esquema Caso 3

- **CASO 4:** Predicción mediante Redes Neuronales de Componentes Periódicas + Predicción de Tendencia mediante Redes Neuronales + Predicción de Residuos mediante Redes Neuronales. En la Figura 34 se muestra el esquema utilizado. Este caso es igual que el CASO 2 pero se le añade la predicción del residuo, en vez de desecharlo como en el CASO 2, esto añadirá previsiblemente más precisión a la predicción si suponemos que hay información en los residuos.

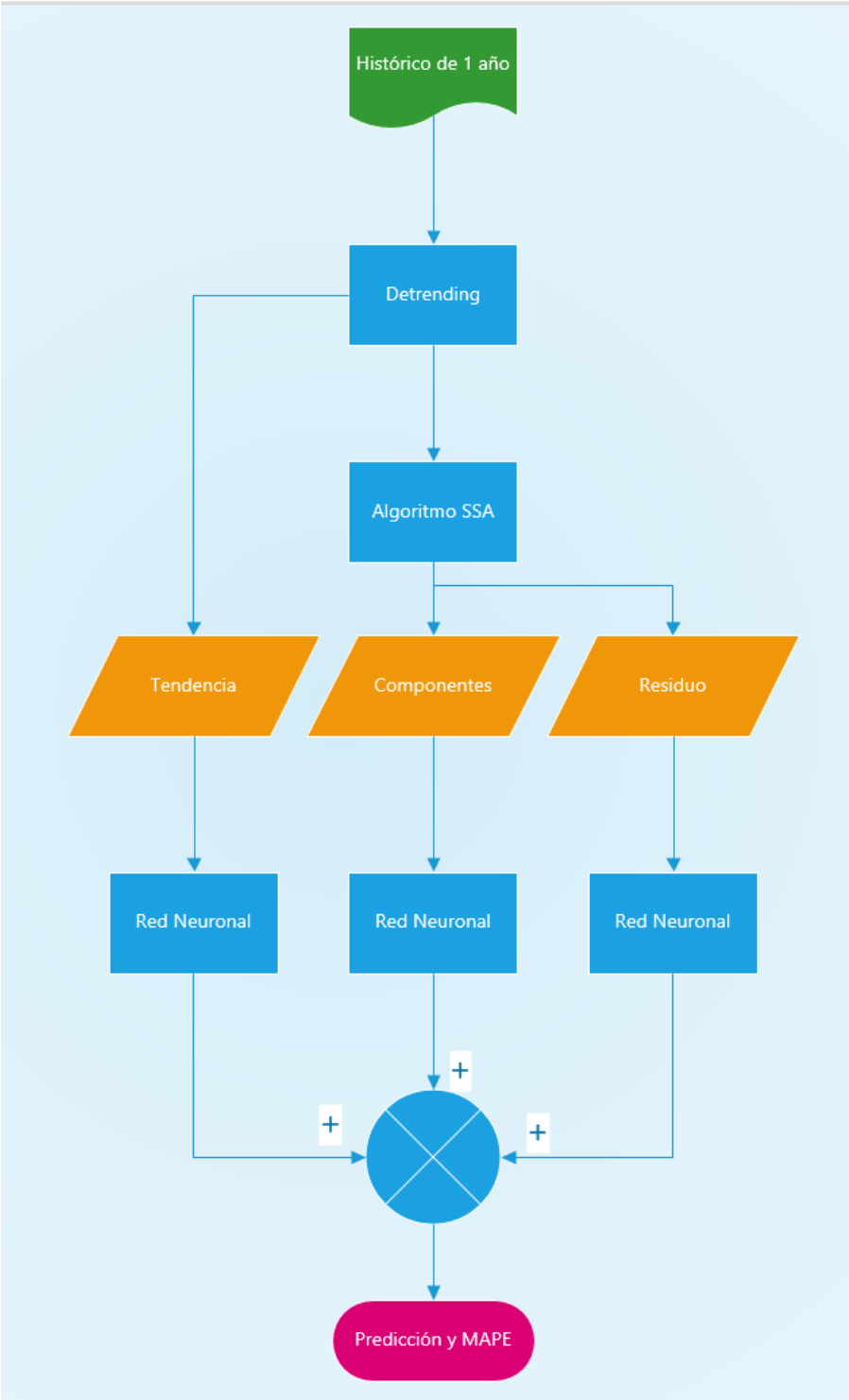


Figura 34. Esquema Caso 4

Estas son todas las opciones planteadas en este TFG. Para cuantificar las prestaciones de cada solución es necesario alguna clase de índice que nos permita comparar las bondades de las predicciones.

Existen multitud de indicadores de la bondad de una predicción, vamos a realizar una pequeña revisión de los mismos a título informativo. Para la medida del error de la predicción se suelen utilizar 3 indicadores, que son los más populares: el **MAPE**, que significa **"Mean Absolute Percentage Error",** el **NMBE** que significa **"Normalized Mean Bias Error"** y el **NRMSE** que significa **"Normalized Root Mean Square Error"**. Vamos a explicar brevemente en qué consiste cada uno.

El MAPE o error porcentual medio absoluto permite obtener el porcentaje de error medio absoluto a lo largo de una predicción. Es un índice ampliamente utilizado en la comparación de predicciones. La expresión matemática se muestra en la Ecuación 6:

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{V_r - V_p}{V_r}\right| (\%)$$

Ecuación 6

Donde n es el número de muestras de la predicción, V_r es el valor real y V_p el valor predicho.

EL NMBE o error de sesgo promedio normalizado permite obtener el porcentaje de error medio a lo largo de una predicción. Se diferencia del caso anterior en que no se usa el valor absoluto. La expresión matemática se muestra en la Ecuación 7:

$$NMBE = \frac{100}{n}\sum_{t=1}^{n}\left(\frac{V_r - V_p}{V_r}\right) (\%)$$

Ecuación 7

Donde n es el número de muestras de la predicción, V_r es el valor real y V_p el valor predicho.

EL NRMSE o Raíz del error cuadrático medio normalizado permite obtener la varianza del error medio a lo largo de una predicción, valores más pequeños indican menor varianza en el error. La expresión matemática se muestra en la Ecuación 8Ecuación 7:

$$NRMSE = \sqrt{\frac{100}{n}\sum_{t=1}^{n}\left(\frac{V_r - V_p}{V_r}\right)^2}$$

Ecuación 8

Donde n es el número de muestras de la predicción, V_r es el valor real y V_p el valor predicho.

El método elegido finalmente es el MAPE, debido a que permite hallar el error medio en valor absoluto, ya que en nuestro caso no nos interesa saber si el error se comete por encima o por debajo, además de que si se usase el NMBE podría dar un 0% de error de media debido a que hay errores negativos y positivos, mientras que el MAPE es en valor absoluto. El NRMSE se desecha porque no nos interesa tanto la varianza del error como el error en sí cometido. Por lo que se concluye que se va a elegir el MAPE.

A continuación, se muestran un conjunto de pruebas, donde se comparan los distintos MAPE y resultados gráficos de las predicciones. Las pruebas aquí mostradas son mejores y peores casos del set de pruebas que se ha realizado. Para el set de pruebas, como se dispone de datos de tres años, se pueden hacer pruebas de solicitar predicciones de 2014 y 2015, por lo que las pruebas se han realizado escogiendo un día de cada mes, por cada mes de cada año, lo que hace un set de 24 pruebas, siempre atendiendo a escoger días distintos de la semana y días distintos de un mes a otro.

Predicción del 1 de Junio del 2014:

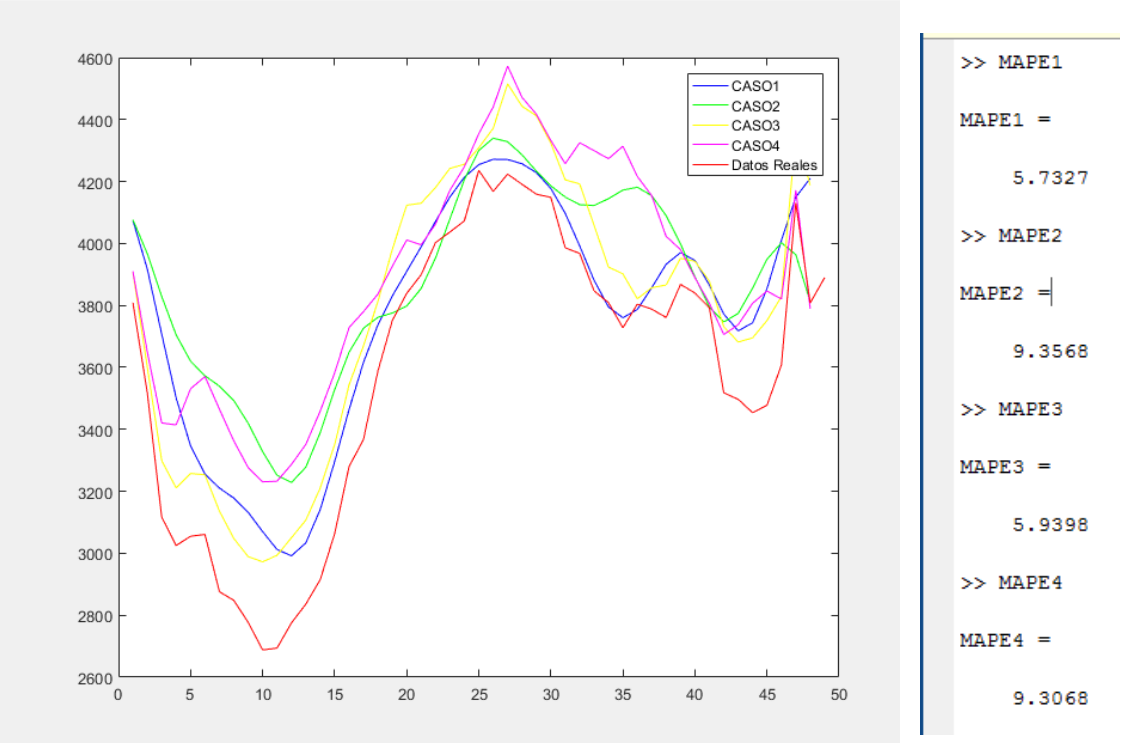
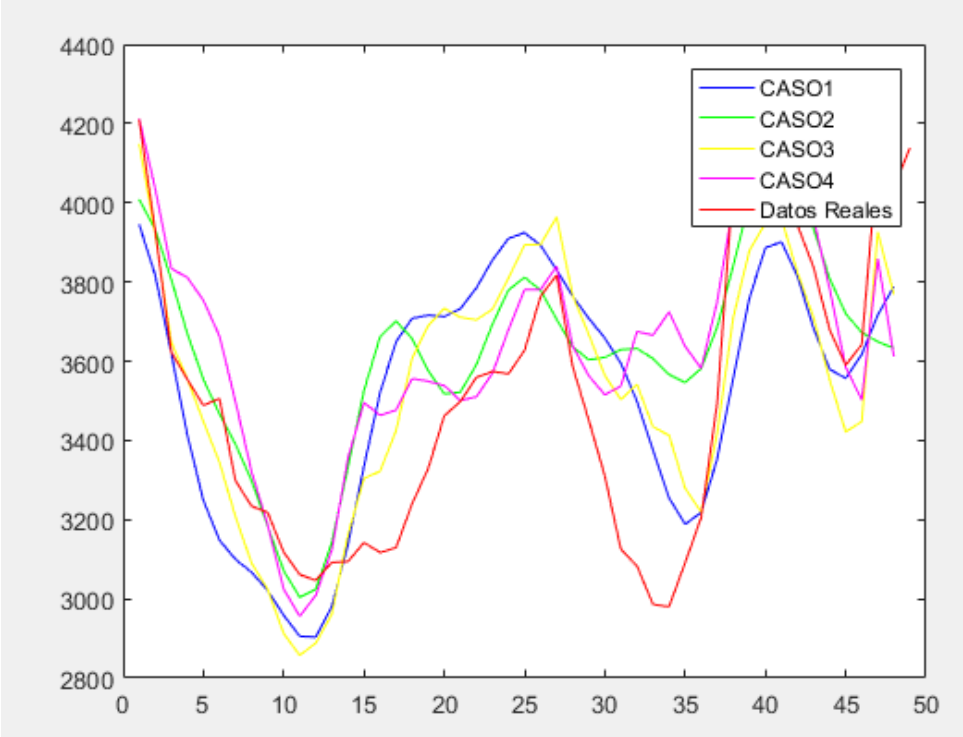


Figura 35. Predicción del 1 de Junio del 2014

En la prueba de la Figura 35, el caso 1 y el caso 3 son los más ajustados a la realidad, un poco mejor el caso 1. Esta prueba es la peor del set de pruebas que se ha realizado, como se puede observar, tiene MAPES que oscilan entre 5 y 9, por lo que tiene bastante error. Como se comentó, antes de decidir qué esquema de predicción es el mejor, se hicieron 24 pruebas escogidas al azar entre los 24 meses.

Predicción del 1 de Noviembre del 2015:




```
>> MAPE1

MAPE1 =

    6.6799

>> MAPE2

MAPE2 =

    6.1170

>> MAPE3

MAPE3 =

    5.6678

>> MAPE4

MAPE4 =

    5.8598
```

Figura 36. Predicción del 1 de Noviembre del 2015

En la Figura 36 se puede observar que el mejor caso es el caso 3, seguido del caso 4, por lo que de momento nos quedaríamos con el caso 3 que es el más ajustado a los datos reales.

Este caso es un caso medio en el que el MAPE se encuentra en torno al 6%, la media de las pruebas realizadas (El set completo de 24) mostraba que la media del MAPE del caso 3 es de aproximadamente un 5%, mientras que las medias de los casos 4 y 5 rondaban el 8%. Aun así, vamos a mostrar la tercera prueba que aquí se documenta, que corresponde con un día en el que la predicción es mejor que la media.

Predicción del 19 de Marzo del 2014:

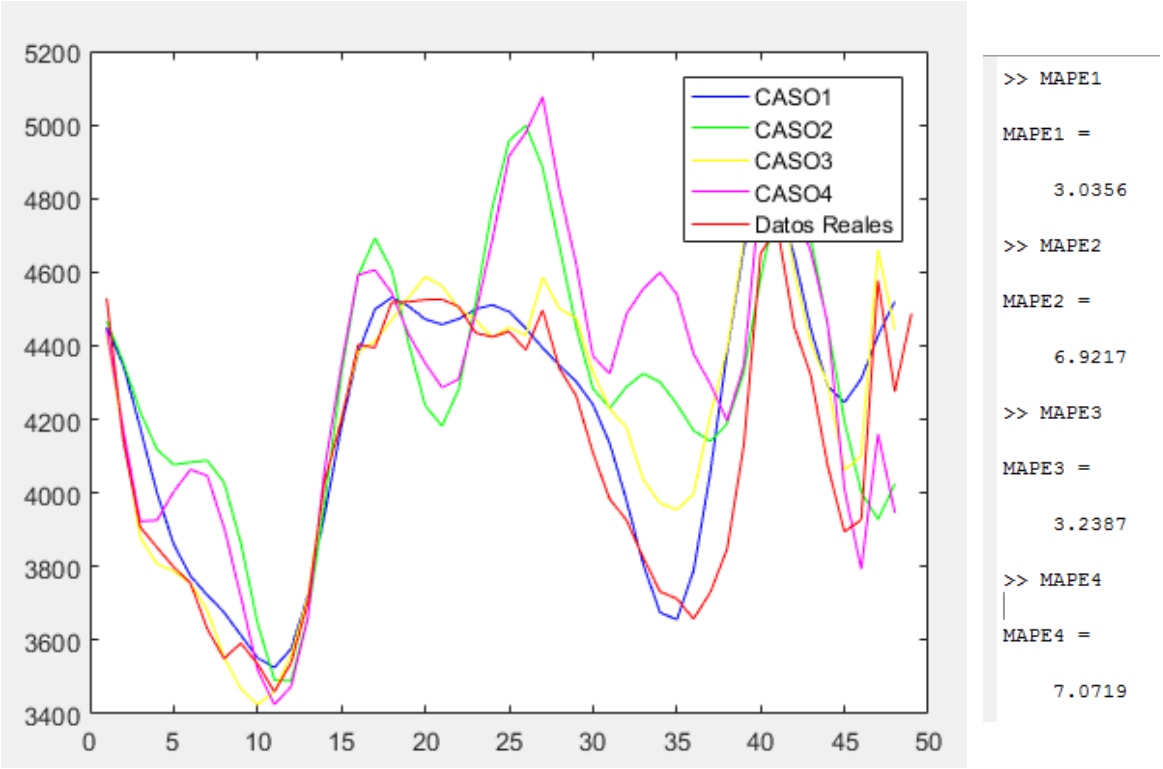


Figura 37. Predicción del 19 de Marzo de 2014

En la Figura 37, el mejor caso sería el Caso 1, seguido de cerca por el caso 3. Como se puede observar, el CASO 3 tiene en esta prueba un MAPE de 3.2387, muy por debajo del 5 % que se tiene de media, por lo que este se trata de uno de los mejores casos probados.

Aunque no se incluyen los resultados gráficos de más pruebas en el documento, se han realizado pruebas aleatorias en todos los meses de los dos años que se pueden utilizar (recordemos que necesitamos al menos un año de historial). De estas pruebas se concluye que son responden sensiblemente mejor los casos 1 y 3. Ambos tienen en común que la predicción de las componentes se ha realizado mediante SSA y además la tendencia se ha predicho mediante la Red Neuronal.

Las pruebas realizadas demuestran que ambos casos son muy parecidos, pero que es ligeramente mejor el que incluye la predicción de residuos mediante la red neuronal, por lo que será nuestra propuesta final para la predicción.

En la Tabla 1 se muestran los resultados obtenidos para las 24 pruebas realizadas:

Fecha de prueba	MAPE 1 (%)	MAPE 2 (%)	MAPE 3 (%)	MAPE 4 (%)
6/01/2014	6.763	4.075	6.101	2.995
11/02/2014	2.331	13.943	2.338	13.415
19/03/2014	3.036	6.922	3.239	7.072
17/04/2014	6.200	8.485	6.209	8.526
7/05/2014	4.911	8.854	5.205	8.156
1/06/2014	4.430	8.980	3.845	9.518
9/07/2014	6.705	11.379	8.452	12.474
31/08/2014	5.788	10.434	5.026	9.655
14/09/2014	4.837	8.607	3.767	8.385
29/10/2014	5.121	20.045	5.055	19.544
6/11/2014	6.414	14.052	6.395	14.674
27/12/2017	8.272	5.909	6.185	4.727
15/01/2015	4.283	5.948	4.653	5.554
9/02/2015	3.390	8.234	3.279	8.937
21/03/2015	5.177	5.036	4.352	4.207
11/04/2015	9.677	7.875	10.840	9.501
15/05/2015	3.400	3.371	2.071	1.940
23/06/2015	3.964	3.052	1.796	2.446
28/07/2015	2.780	2.947	2.224	3.076
07/08/2015	7.134	11.348	7.338	11.602
17/09/2015	11.112	9.926	10.043	8.964
04/10/2015	4.612	7.713	3.397	7.096
01/11/2015	6.680	6.117	5.668	5.860
10/12/2015	8.007	5.547	5.410	3.479
Media	**5.626**	**8.283**	**5.120**	**7.992**

Tabla 1. Pruebas aleatorias realizadas

Como se puede observar, el promedio de los MAPE obtenidos permite discernir cual es el mejor caso de predicción, el caso 3, seguido de cerca por el caso 1. Por lo que se puede extrapolar que la predicción de las componentes es más precisa mediante el uso de los algoritmos de predicción SSA (Fórmula lineal recurrente).

Por último, se ha desarrollado una aplicación de interfaz gráfica de matlab que permite obtener en 2 pasos la predicción, tal y como se explica en el apartado dedicado al manual de usuario.

8. CONCLUSIONES Y TRABAJOS FUTUROS

8.1 Conclusiones

En este TFG se ha diseñado un sistema que permite generar una predicción de carga con un horizonte de 24 horas a partir de un historial de carga de un año. Para ello, el sistema se basa en una herramienta híbrida basada en el algoritmo de descomposición SSA y redes neuronales (ANN) del tipo NARX. Para su utilización, se pone a disposición del usuario una interfaz gráfica realizada en el entorno MATLAB que permite cargar un histórico de usuario y permite realizar predicciones de manera simplificada.

Las conclusiones a las que se ha llegado tras la realización de este TFG son las siguientes:

En primer lugar, no es sencillo predecir el futuro de manera precisa, teniendo que recurrir a algoritmos matemáticos que hacen que la solución sea más compleja. Por otro lado, estas herramientas matemáticas son de aplicación a distintos ámbitos, más allá de la predicción de carga eléctrica.

En segundo lugar, también se llega a la conclusión de que la solución adoptada no es definitiva, sino que, en caso de ser utilizada de forma comercial, debería ser revisada de manera periódica, reajustando las redes neuronales de acuerdo a los nuevos datos que se van obteniendo. Además, habría que revisar de manera periódica que los parámetros del algoritmo SSA se siguen ajustando a la estructura de los datos. Por ejemplo, lo más probable es que hubiese que cambiar el agrupamiento que se ha realizado inicialmente o ajustar el cálculo de la tendencia, haciéndolo más o menos estricto.

En tercer lugar, se aprende de la realización de este TFG que la predicción del consumo de una zona es muy variable, pero se puede encontrar una explicación para las subidas y bajadas del mismo. Siempre se puede achacar a variaciones a lo largo del año, fluctuaciones a lo largo de la semana u otros parámetros.

8.2 Trabajos Futuros

Este trabajo podría ampliarse de diversas maneras, las cuales podrían conseguir una mejora en la predicción obtenida.

Sería muy interesante aplicar esta misma filosofía para realizar una predicción del consumo teniendo en cuenta más variables como la temperatura ambiente, ya que tiene una gran dependencia de ésta, o el PIB del país, ya que es lógico que cuanto más rico sea el país, más energía se consumirá en el mismo.

También podría estudiarse la posibilidad de sustituir las redes NARX por redes convolucionales o redes que autoaprendan, ya que evitaría tener que reentrenarlas.

9. PLANOS Y DIAGRAMAS

En este TFG se ha escrito una serie de fuentes que son los que se utilizan en la aplicación, estos se pueden dividir en dos grupos, aquellos que son en código MATLAB y aquellos que han sido escritos en lenguaje R, por lo tanto, en este listado de código se va a dividir en dos partes, el código MATLAB y el código R. Todos estos ficheros fuentes estarán disponibles en la carpeta llamada "Fuentes" que se encuentra dentro del empaquetado de la aplicación.

Para facilitar el seguimiento del flujo de código del programa, se adjunta en la Figura un diagrama de flujo con el funcionamiento de la aplicación y las llamadas a funciones.

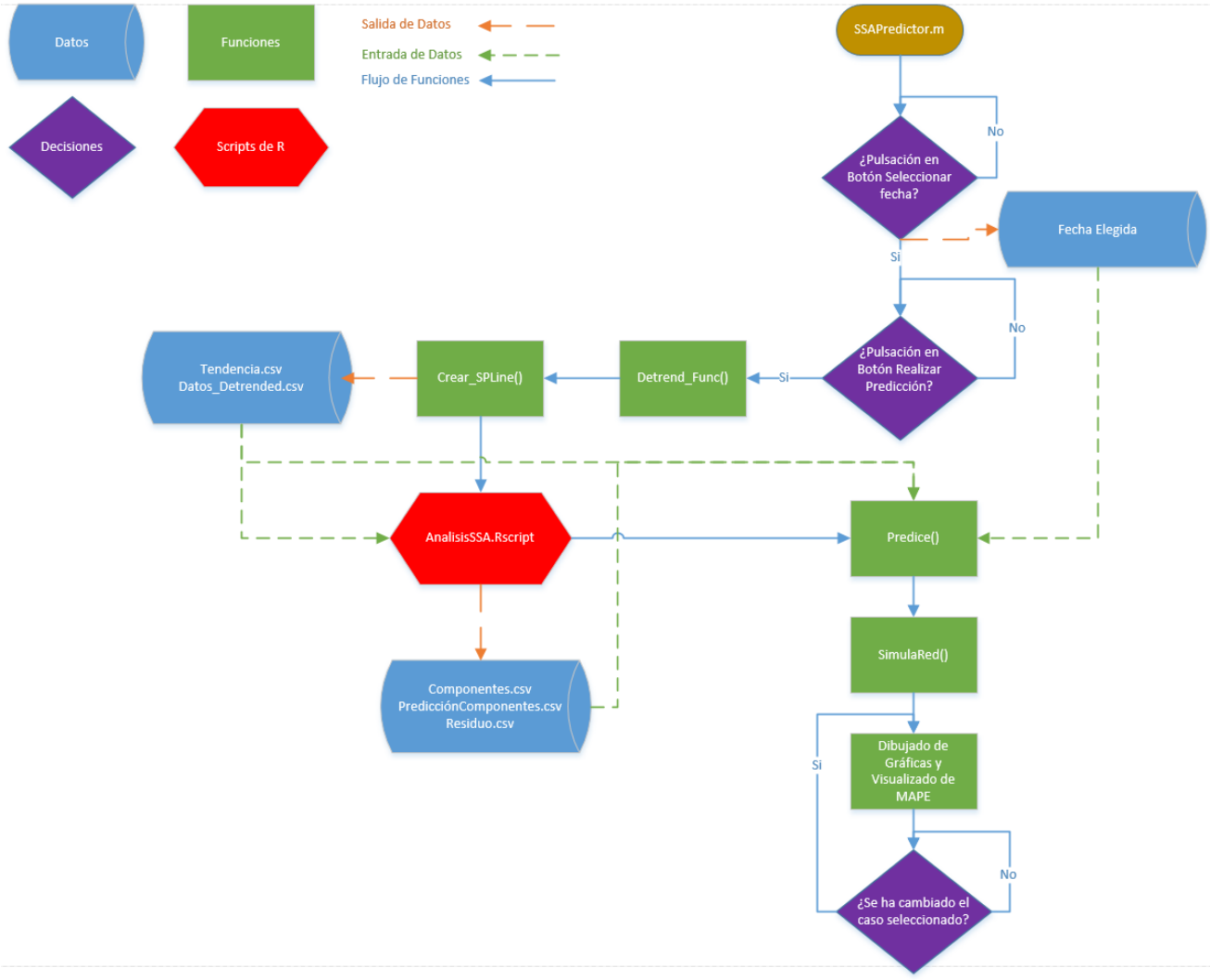


Figura 38. Diagrama de flujo de la aplicación

9.1 Fuentes de Matlab

El código de Matlab se basa en la aplicación que se ha diseñado, esta aplicación llama a su vez a funciones que se han escrito, se podrá observar que, a la hora de simular las redes, se hacen llamadas a scripts que se llaman "RedComponentex_m.", estos son scripts que se obtienen al generar las redes neuronales, no se listarán todos, se listará uno para que sirva de ejemplo.

9.1.1 SSAPredictor.m

Esta función es la encargada de ejecutar la aplicación, en ella se realizan llamadas a las otras funciones, son 2:

- Detrend_Func()
- Predice()

El código de la aplicación es el siguiente:

```
function varargout = SSA_Predictor(varargin)
% SSA_PREDICTOR MATLAB code for SSA_Predictor.fig
% Aplicación Para predicción de carga


% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SSA_Predictor_OpeningFcn, ...
                   'gui_OutputFcn',  @SSA_Predictor_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before SSA_Predictor is made visible.
function SSA_Predictor_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SSA_Predictor (see VARARGIN)
```

```
% Choose default command line output for SSA_Predictor
global ready
handles.output = hObject;
ready=0;
% Update handles structure
guidata(hObject, handles);
%Carga de logos
axes(handles.axes3)
matlabImage = imread('uah.jpg');
image(matlabImage)
axis off
axis image
axes(handles.axes4)
matlabImage = imread('logodepeca.jpg');
image(matlabImage)
axis off
axis image



% --- Outputs from this function are returned to the command line.
function varargout = SSA_Predictor_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Función de respuesta a la pulsación del botón de Seleccionar
Fecha.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global val1
%Invocación del calendario
uicalendar('DestinationUI', {handles.text2,
'String'},'InitDate',datetime('1-Jan-2014','Locale','en_US'));
waitfor(handles.text2,'String');
%Captura de la fecha elegida
val1 = get(handles.text2,'String');
Fechaelegida=datetime(val1,'Inputformat','dd-MMM-yyyy');

% --- Función de respuesta a la pulsación del botón de Realizar
Predicción.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global i
global val1
global PrediccionCASO1
global PrediccionCASO2
global PrediccionCASO3
global PrediccionCASO4
global MAPE1
global MAPE2
```

```
global MAPE3
global MAPE4
global ready
%Preparación de ejes para dibujar el detrend
axes(handles.axes1)
%Ejecución del detrending
Detrend_func;
%Llamada a la descomposición SSA con R
system('AnalisisSSA.Rexec')
%Carga de datos
load('Datos.mat');
load('Fechas.mat');
%Lectura del valor del desplegable de los casos
contents = cellstr(get(handles.popupmenu1,'String'));
axes(handles.axes2)
%Predicción del día elegido
[PrediccionCASO1,PrediccionCASO2,PrediccionCASO3,PrediccionCASO4,MAPE1
,MAPE2,MAPE3,MAPE4]=Predice(i-48);
%Actualización de MAPE y gráfica en función del caso elegido
switch(contents{get(handles.popupmenu1,'Value')})
        case 'Caso 1'
           axes(handles.axes2)
           hold off
           plot(PrediccionCASO1,'b')
           hold on
           plot(Datos(i:i+48),'r')
           legend('CASO1','Datos Reales')
           cadena=sprintf('MAPE Caso 1= %.3f',MAPE1);
           set(handles.t_MAPE1,'String',cadena);
           set(handles.t_MAPE2,'String','');
           set(handles.t_MAPE3,'String','');
           set(handles.t_MAPE4,'String','');
        case 'Caso 2'
           axes(handles.axes2)
           hold off
           plot(PrediccionCASO2,'g')
           hold on
           plot(Datos(i:i+48),'r')
           legend('CASO2','Datos Reales')
           cadena=sprintf('MAPE Caso 2= %.3f',MAPE2);
           set(handles.t_MAPE2,'String',cadena);
           set(handles.t_MAPE1,'String','');
           set(handles.t_MAPE3,'String','');
           set(handles.t_MAPE4,'String','');
        case 'Caso 3'
           axes(handles.axes2)
           hold off
           plot(PrediccionCASO3,'k')
           hold on
           plot(Datos(i:i+48),'r')
           legend('CASO3','Datos Reales')
           cadena=sprintf('MAPE Caso 3= %.3f',MAPE3);
           set(handles.t_MAPE3,'String',cadena);
           set(handles.t_MAPE1,'String','');
           set(handles.t_MAPE2,'String','');
           set(handles.t_MAPE4,'String','');
        case 'Caso 4'
           axes(handles.axes2)
           hold off
           plot(PrediccionCASO4,'m')
           hold on
```

```
            plot(Datos(i:i+48),'r')
            legend('CASO4','Datos Reales')
            cadena=sprintf('MAPE Caso 4= %.3f',MAPE4);
            set(handles.t_MAPE4,'String',cadena);
            set(handles.t_MAPE1,'String','');
            set(handles.t_MAPE2,'String','');
            set(handles.t_MAPE3,'String','');
        case 'Comparativa'
            axes(handles.axes2)
            hold off
            plot(PrediccionCASO1,'b')
            hold on
            plot(PrediccionCASO2,'g')
            plot(PrediccionCASO3,'k')
            plot(PrediccionCASO4,'m')
            plot(Datos(i:i+48),'r')
            legend('CASO1','CASO2','CASO3','CASO4','Datos Reales')
            cadena=sprintf('MAPE Caso 1= %.3f',MAPE1);
            set(handles.t_MAPE1,'String',cadena);
            cadena=sprintf('MAPE Caso 2= %.3f',MAPE2);
            set(handles.t_MAPE2,'String',cadena);
            cadena=sprintf('MAPE Caso 3= %.3f',MAPE3);
            set(handles.t_MAPE3,'String',cadena);
            cadena=sprintf('MAPE Caso 4= %.3f',MAPE4);
            set(handles.t_MAPE4,'String',cadena);
    end
ready=1;

% --- Función de respuesta ante cambio en el desplegable.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1
global PrediccionCASO1
global PrediccionCASO2
global PrediccionCASO3
global PrediccionCASO4
global MAPE1
global MAPE2
global MAPE3
global MAPE4
global i
global ready
%Carga de datos
load('Datos.mat');
load('Fechas.mat');
contents = cellstr(get(hObject,'String'));
%Actualización de MAPE y gráfica en función del caso elegido
if ready==1
    switch(contents{get(hObject,'Value')})
        case 'Caso 1'
            axes(handles.axes2)
            hold off
            plot(PrediccionCASO1,'b')
            hold on
            plot(Datos(i:i+48),'r')
```

```
            legend('CASO1','Datos Reales')
            cadena=sprintf('MAPE Caso 1= %.3f',MAPE1);
            set(handles.t_MAPE1,'String',cadena);
            set(handles.t_MAPE2,'String','');
            set(handles.t_MAPE3,'String','');
            set(handles.t_MAPE4,'String','');
        case 'Caso 2'
            axes(handles.axes2)
            hold off
            plot(PrediccionCASO2,'g')
            hold on
            plot(Datos(i:i+48),'r')
            legend('CASO2','Datos Reales')
            cadena=sprintf('MAPE Caso 2= %.3f',MAPE2);
            set(handles.t_MAPE2,'String',cadena);
            set(handles.t_MAPE1,'String','');
            set(handles.t_MAPE3,'String','');
            set(handles.t_MAPE4,'String','');
        case 'Caso 3'
            axes(handles.axes2)
            hold off
            plot(PrediccionCASO3,'k')
            hold on
            plot(Datos(i:i+48),'r')
            legend('CASO3','Datos Reales')
            cadena=sprintf('MAPE Caso 3= %.3f',MAPE3);
            set(handles.t_MAPE3,'String',cadena);
            set(handles.t_MAPE1,'String','');
            set(handles.t_MAPE2,'String','');
            set(handles.t_MAPE4,'String','');
        case 'Caso 4'
            axes(handles.axes2)
            hold off
            plot(PrediccionCASO4,'m')
            hold on
            plot(Datos(i:i+48),'r')
            legend('CASO4','Datos Reales')
            cadena=sprintf('MAPE Caso 4= %.3f',MAPE4);
            set(handles.t_MAPE4,'String',cadena);
            set(handles.t_MAPE1,'String','');
            set(handles.t_MAPE2,'String','');
            set(handles.t_MAPE3,'String','');
        case 'Comparativa'
            axes(handles.axes2)
            hold off
            plot(PrediccionCASO1,'b')
            hold on
            plot(PrediccionCASO2,'g')
            plot(PrediccionCASO3,'k')
            plot(PrediccionCASO4,'m')
            plot(Datos(i:i+48),'r')
            legend('CASO1','CASO2','CASO3','CASO4','Datos Reales')
            cadena=sprintf('MAPE Caso 1= %.3f',MAPE1);
            set(handles.t_MAPE1,'String',cadena);
            cadena=sprintf('MAPE Caso 2= %.3f',MAPE2);
            set(handles.t_MAPE2,'String',cadena);
            cadena=sprintf('MAPE Caso 3= %.3f',MAPE3);
            set(handles.t_MAPE3,'String',cadena);
            cadena=sprintf('MAPE Caso 4= %.3f',MAPE4);
            set(handles.t_MAPE4,'String',cadena);
    end
```

```
end

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

9.1.2 Detrend_Func.m

En este script se realiza el *detrending* de los datos, es la primera función que se utiliza cuando se quiere realizar la predicción, dentro de la función se realiza una llamada a la función CrearSPLine() que genera la SPLine de la Tendencia.

El código es el siguiente:

```
function Detrend_func()

global val1
global i
%Carga de datos
load('Datos.mat');
load('Fechas.mat');
longitudhistorial=365*48;
Fechaelegida=datetime(val1,'Inputformat','dd-MMM-yyyy');
close(figure(1))
%Búsqueda de fecha en el listado de fechas
for i=1:length(Fechas)
    if(Fechaelegida==Fechas(i))
        break;
    end
end
t=1:longitudhistorial;
%Creación de la SPLine (Tendencia)
[Tendencia,info]=CrearSpline(t,Datos(i-longitudhistorial:i-1),1e-
7);%1e-9
Tendencia=Tendencia(t);
Tendencia=Tendencia';
Datos_Detrended=(Datos(i-longitudhistorial:i-1)-Tendencia);
%Exportado de los datos sin tendencia y la tendencia
csvwrite('Datos_Detrended.csv',Datos_Detrended');
csvwrite('Tendencia.csv',Tendencia');
```

9.1.3 CrearSPline.m

Esta función se encarga de crear una SPLine que se ajuste según el parámetro de entrada (Parámetro de suavizado). El código fuente es el siguiente:

```
function [fitresult, gof] = CrearSpline(t, Datos, parameter)

%Prepara los datos como columnas
[xData, yData] = prepareCurveData( t, Datos );

% Tipo de ajuste y selección de parámetro
ft = fittype( 'smoothingspline' );
opts = fitoptions( 'Method', 'SmoothingSpline' );
opts.SmoothingParam = parameter ;

% Ejecución del ajuste a SPLine
[fitresult, gof] = fit( xData, yData, ft, opts );

% Dibujado del resultado
h = plot( fitresult, xData, yData );
legend( h, 'Histórico de 1 año', 'Tendencia');
xlabel t
ylabel Datos
grid on
```

9.1.4 Predice.m

Esta función es la más importante de todas, ya que es la encargada de hacer llamadas a las funciones que simulan las redes neuronales. Devuelve directamente los datos de predicción y los MAPES asociados a cada uno de los casos.

Hace llamadas a la función SimulaRed() para cada una de las redes neuronales.

El código es el siguiente:

```
function
[PrediccionCASO1,PrediccionCASO2,PrediccionCASO3,PrediccionCASO4,MAPE1
,MAPE2,MAPE3,MAPE4] = Predice(muestra)

%Carga de datos de las componentes SSA, Resíduos, Tendencia y los
datos del histórico
load('Datos.mat');
load('Fechas.mat');
Componente1=csvread('Componente1.csv',1,1)';
Componente2=csvread('Componente2.csv',1,1)';
Componente3=csvread('Componente3.csv',1,1)';
Componente4=csvread('Componente4.csv',1,1)';
Componente5=csvread('Componente5.csv',1,1)';
Componente6=csvread('Componente6.csv',1,1)';
Componente7=csvread('Componente7.csv',1,1)';
Componente8=csvread('Componente8.csv',1,1)';
Componente9=csvread('Componente9.csv',1,1)';
Componente10=csvread('Componente10.csv',1,1)';
Residuo=csvread('Residuo.csv',1,1)';
Tendencia=csvread('Tendencia.csv')';
%Carga de la predicción de las componentes mediante SSA
PrediccionSSA=csvread('ForecastSSA.csv',1,1)';
%Carga de los objetos de las redes neuronales
```

```
load('RedResiduos.mat');
load('RedTendencia.mat');
load('RedComponente1.mat');
load('RedComponente2.mat');
load('RedComponente3.mat');
load('RedComponente4.mat');
load('RedComponente5.mat');
load('RedComponente6.mat');
load('RedComponente7.mat');
load('RedComponente8.mat');
load('RedComponente9.mat');
load('RedComponente10.mat');

%Creación de las señales exógenas
t=1:length(Datos);
senodiario=sin(2*pi*t/48);
senosemanal=sin(2*pi*t/(48*7));
senoanual=sin(2*pi*t/(48*365));

%Preparación de los datos de entrada de la red y simulación de la
misma
EntradaResiduos=[senodiario(muestra-
96:muestra+47);senosemanal(muestra-96:muestra+47);senoanual(muestra-
96:muestra+47);Residuo(end-143:end)];
TargetResiduos=[Residuo(end-95:end) zeros(1,48)];
SalidaR=Simulared(RedResiduos,EntradaResiduos,TargetResiduos,0);

%Preparación de los datos de entrada de la red y simulación de la
misma
EntradaT=[senoanual(muestra-2:muestra+47);Tendencia(end-49:end)];
TargetT=[Tendencia(end-1:end) zeros(1,48)];
SalidaT=Simulared(RedTendencia,EntradaT,TargetT,-1);

%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada1=[senodiario(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente1(end-49:end)];
Target1=[Componente1(end-1:end) zeros(1,48)];
Salida1=Simulared(RedComponente1,Entrada1,Target1,1);

%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada2=[senodiario(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente2(end-49:end)];
Target2=[Componente2(end-1:end) zeros(1,48)];
Salida2=Simulared(RedComponente2,Entrada2,Target2,2);

%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada3=[senosemanal(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente3(end-49:end)];
Target3=[Componente3(end-1:end) zeros(1,48)];
Salida3=Simulared(RedComponente3,Entrada3,Target3,3);

%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada4=[senodiario(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente4(end-49:end)];
Target4=[Componente4(end-1:end) zeros(1,48)];
Salida4=Simulared(RedComponente4,Entrada4,Target4,4);
```

```
%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada5=[senodiario(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente5(end-49:end)];
Target5=[Componente5(end-1:end) zeros(1,48)];
Salida5=Simulared(RedComponente5,Entrada5,Target5,5);

%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada6=[senosemanal(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente6(end-49:end)];
Target6=[Componente6(end-1:end) zeros(1,48)];
Salida6=Simulared(RedComponente6,Entrada6,Target6,6);

%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada7=[senodiario(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente7(end-49:end)];
Target7=[Componente7(end-1:end) zeros(1,48)];
Salida7=Simulared(RedComponente7,Entrada7,Target7,7);

%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada8=[senodiario(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente8(end-49:end)];
Target8=[Componente8(end-1:end) zeros(1,48)];
Salida8=Simulared(RedComponente8,Entrada8,Target8,8);

%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada9=[senodiario(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente9(end-49:end)];
Target9=[Componente9(end-1:end) zeros(1,48)];
Salida9=Simulared(RedComponente9,Entrada9,Target9,9);

%Preparación de los datos de entrada de la red y simulación de la
misma
Entrada10=[senodiario(muestra-2:muestra+47);senoanual(muestra-
2:muestra+47);Componente10(end-49:end)];
Target10=[Componente10(end-1:end) zeros(1,48)];
Salida10=Simulared(RedComponente10,Entrada10,Target10,10);


%Generación de cada uno de los casos de predicción
PrediccionCASO1=SalidaT+PrediccionSSA;
PrediccionCASO2=SalidaT+Salida1+Salida2+Salida3+Salida4+Salida5+Salida
6+Salida7+Salida8+Salida9+Salida10;
PrediccionCASO3=SalidaT+PrediccionSSA+SalidaR;
PrediccionCASO4=SalidaT+SalidaR+Salida1+Salida2+Salida3+Salida4+Salida
5+Salida6+Salida7+Salida8+Salida9+Salida10;


%Cálculo de los MAPEs
MAPE1=0;
for i=1:48
    MAPE1=MAPE1+abs((PrediccionCASO1(i)-
Datos(muestra+47+i))/Datos(muestra+47+i));
end
MAPE1=MAPE1*100/48;

MAPE2=0;
```

```
for i=1:48
    MAPE2=MAPE2+abs((PrediccionCASO2(i)-
Datos(muestra+47+i))/Datos(muestra+47+i));
end
MAPE2=MAPE2*100/48;

MAPE3=0;

for i=1:48
    MAPE3=MAPE3+abs((PrediccionCASO3(i)-
Datos(muestra+47+i))/Datos(muestra+47+i));
end
MAPE3=MAPE3*100/48;
MAPE4=0;
for i=1:48
    MAPE4=MAPE4+abs((PrediccionCASO4(i)-
Datos(muestra+47+i))/Datos(muestra+47+i));
end
MAPE4=MAPE4*100/48;
```

9.1.5 SimulaRed.m

Esta función es la encargada de simular cada una de las redes neuronales, dando a cada una los retardos que necesita y adecuando la inicialización de los estados iniciales de su capa interna (Esto se hace con la función preparets()).

El código es el siguiente:

```
function [Salida] = Simulared(Red,Entrada,Target,nred)

%Comprobación de qué red se está simulando
switch nred
    case -1       %Simulación de red de la Tendencia
        X = tonndata(Entrada,true,false);
        T = tonndata(Target,true,false);
        [Xs,Xi,Ai] = preparets(Red,X,{},T);
        Xs=cell2mat(Xs);
        Xi=cell2mat(Xi);
        Ai=cell2mat(Ai(2,:));
        Salida=RedTendencia_m(Xs,Xi,Ai);
    case 0        %Simulación de red del Residuo
        X = tonndata(Entrada,true,false);
        T = tonndata(Target,true,false);
        [Xs,Xi,Ai] = preparets(Red,X,{},T);
        Xs=cell2mat(Xs);
        Xi=cell2mat(Xi);
        Ai=cell2mat(Ai(2,:));
        Salida=RedResiduos_m(Xs,Xi,Ai);
    case 1        %Simulación de red de la Componente 1
        X = tonndata(Entrada,true,false);
        T = tonndata(Target,true,false);
        [Xs,Xi,Ai] = preparets(Red,X,{},T);
        Xs=cell2mat(Xs);
        Xi=cell2mat(Xi);
        Ai=cell2mat(Ai(2,:));
        Salida=RedComponente1_m(Xs,Xi,Ai);
    case 2        %Simulación de red de la Componente 2
        X = tonndata(Entrada,true,false);
        T = tonndata(Target,true,false);
        [Xs,Xi,Ai] = preparets(Red,X,{},T);
        Xs=cell2mat(Xs);
```

```
    Xi=cell2mat(Xi);
    Ai=cell2mat(Ai(2,:));
    Salida=RedComponente2_m(Xs,Xi,Ai);
case 3        %Simulación de red de la Componente 3
    X = tonndata(Entrada,true,false);
    T = tonndata(Target,true,false);
    [Xs,Xi,Ai] = preparets(Red,X,{},T);
    Xs=cell2mat(Xs);
    Xi=cell2mat(Xi);
    Ai=cell2mat(Ai(2,:));
    Salida=RedComponente3_m(Xs,Xi,Ai);
case 4        %Simulación de red de la Componente 4
    X = tonndata(Entrada,true,false);
    T = tonndata(Target,true,false);
    [Xs,Xi,Ai] = preparets(Red,X,{},T);
    Xs=cell2mat(Xs);
    Xi=cell2mat(Xi);
    Ai=cell2mat(Ai(2,:));
    Salida=RedComponente4_m(Xs,Xi,Ai);
case 5        %Simulación de red de la Componente 5
    X = tonndata(Entrada,true,false);
    T = tonndata(Target,true,false);
    [Xs,Xi,Ai] = preparets(Red,X,{},T);
    Xs=cell2mat(Xs);
    Xi=cell2mat(Xi);
    Ai=cell2mat(Ai(2,:));
    Salida=RedComponente5_m(Xs,Xi,Ai);
case 6        %Simulación de red de la Componente 6
    X = tonndata(Entrada,true,false);
    T = tonndata(Target,true,false);
    [Xs,Xi,Ai] = preparets(Red,X,{},T);
    Xs=cell2mat(Xs);
    Xi=cell2mat(Xi);
    Ai=cell2mat(Ai(2,:));
    Salida=RedComponente6_m(Xs,Xi,Ai);
case 7        %Simulación de red de la Componente 7
    X = tonndata(Entrada,true,false);
    T = tonndata(Target,true,false);
    [Xs,Xi,Ai] = preparets(Red,X,{},T);
    Xs=cell2mat(Xs);
    Xi=cell2mat(Xi);
    Ai=cell2mat(Ai(2,:));
    Salida=RedComponente7_m(Xs,Xi,Ai);
case 8        %Simulación de red de la Componente 8
    X = tonndata(Entrada,true,false);
    T = tonndata(Target,true,false);
    [Xs,Xi,Ai] = preparets(Red,X,{},T);
    Xs=cell2mat(Xs);
    Xi=cell2mat(Xi);
    Ai=cell2mat(Ai(2,:));
    Salida=RedComponente8_m(Xs,Xi,Ai);
case 9        %Simulación de red de la Componente 9
    X = tonndata(Entrada,true,false);
    T = tonndata(Target,true,false);
    [Xs,Xi,Ai] = preparets(Red,X,{},T);
    Xs=cell2mat(Xs);
    Xi=cell2mat(Xi);
    Ai=cell2mat(Ai(2,:));
    Salida=RedComponente9_m(Xs,Xi,Ai);
case 10       %Simulación de red de la Componente 10
    X = tonndata(Entrada,true,false);
```

```
        T = tonndata(Target,true,false);
        [Xs,Xi,Ai] = preparets(Red,X,{},T);
        Xs=cell2mat(Xs);
        Xi=cell2mat(Xi);
        Ai=cell2mat(Ai(2,:));
        Salida=RedComponente10_m(Xs,Xi,Ai);
end
```

9.1.6 RedComponentex_m.m

Este script es un ejemplo de cómo son los scripts que se utilizan para simular las redes neuronales, no vamos a poner todos los scripts porque ocuparía demasiado espacio y no difieren mucho unos de otros (Solo en los pesos y algunos en el número de neuronas y retardos).

Se pone como ejemplo el código de la red de la Componente 1:

```
function [y1,xf1,af2] = RedComponente1_m(x1,xi1,ai2)
%REDCOMPONENTE1_M neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 23-May-
2017 22:31:37.
%
% [y1,xf1,af2] = RedComponente1_m(x1,xi1,ai2) takes these arguments:
%   x1 = 3xTS matrix, input #1
%   xi1 = 3x2 matrix, initial 2 delay states for input #1.
%   ai1 = 20x2 matrix, initial 2 delay states for layer #1.
%   ai2 = 1x2 matrix, initial 2 delay states for layer #2.
% and returns:
%   y1 = 1xTS matrix, output #1
%   xf1 = 3x2 matrix, final 2 delay states for input #1.
%   af1 = 20x2 matrix, final 2 delay states for layer #1.
%   af2 = 1x2 matrix, final 2 delay states for layer #2.
% where TS is the number of timesteps.

% ===== NEURAL NETWORK CONSTANTS =====

% Input 1
x1_step1.xoffset = [-1;-0.866025403784439;-537.346928490387];
x1_step1.gain = [1;1.07179676972449;0.00186094268414719];
x1_step1.ymin = -1;

% Layer 1
b1 = [-1.6965549505937021;1.4630115199497626;-1.5385909579179931;-
0.97229831404201261;-0.16785903781880457;-0.069125211336380998;-
0.95255227153940469;-0.95376427547899711;-0.50704076509352858;-
0.80835833860412065;0.57998146636369907;0.051385423415312943;-
0.21954557579463096;-0.91227607338722794;1.8211168611795066;-
1.5979578729975048;0.52301089295505632;1.7695937925841343;1.2339073382
561567;2.0361177403213668];
IW1_1 = [0.023256305063289372 0.40670652752466724 -0.70005484341520352
0.84055080158468487 0.7087915439389787 1.0109685438363099;-
0.00069382849933307474 -0.43687819197180122 -0.13368736349777333 -
0.00046897218246108726 0.43813331166967434
0.13865517834318322;1.0043860348759481 0.4599742978817824 -
0.78552193006175242 -0.90004667483457623 -0.39928490871336003
0.17673340274624591;-0.022647576105649953 0.4647897808135843 -
```

```
0.39877568650395184 0.015857231122577323 -0.458061557853434
0.42747601704080695;-0.57470978297207131 -0.63007852947707199
1.2852452716711094 0.22978293043985734 0.73231012596917755 -
0.73812751311193181;-0.00093850930920438575 0.026062716443036794 -
0.011959599641242755 0.00067063036536531967 -0.025783225922335774
0.01313315609279595;-0.27008643854391412 -0.45594152225189266
0.55290738727433675 0.90107603785680213 0.33651380368182804 -
0.85899621218706568;-0.27101872943665639 -1.3193518956957542
0.56561138276127076 0.90332405831747553 1.1996007879203023 -
0.87799109488280414;1.272662061674807 0.05438860532753205
0.26822263072702229 -0.43625542354077795 0.91482527656366763
1.2490402633763129;-0.14626516046656018 0.38410015795297975
0.89152464761768069 0.14330265113026444 -0.32665136288779928
1.1577939493854292;1.167244135521134 0.3278563532306758 -
0.84996029023450981 0.17637812970387004 1.4249140515194023 -
0.94401796447686859;-0.0098910118204174041 -0.84042942995923708 -
0.30789435778937951 0.71853665834319602 0.45645129369455767 -
0.67661345361105285;-0.91993156094263395 -0.27057724363639862 -
0.85266176913364355 -1.0080802016525754 0.70623895491513222 -
0.012546622757305819;-1.2019556855333151 0.56220243900584443 -
1.270952530575868 -0.11963044555673336 0.32243291323247253 -
0.062038680973793989;0.047502458692801071 0.64025805143689829
0.89924725322645715 -0.035878698426428574 -0.64273170111484723 -
0.9790095755872632;-1.3853942989461732 0.1949464433246238
0.28065603179197662 0.46109522109195245 -0.41185630929347644 -
0.61394378469368571;0.6174953425210622 -0.38968988427024759
1.0831371444619158 -0.50200205926468522 -0.19227498953412109
1.3300194600659545;1.091689598345184 1.0884780923507589
1.3962388461154078 -0.83069345824396301 0.25484171489084545
0.48644859515296512;-0.079581262100008029 1.1525521386853392
1.7724033565183412 -0.38687206006518043 -0.59992585816135091 -
0.57612618438886232;0.61097394869680932 1.0916144777411465
1.1019947440945257 -0.28331314645461286 0.14362965863516422
0.96711958007365539];
LW1_2 = [0.85931272027298533 -0.44245626350610662;-0.8209002842348887
0.34216700498341873;0.15217834244683862 -
0.40712203611091147;0.39287024353947858 -
0.42918934761072669;0.12063509177979782 -0.95552561850407758;-
0.18972197210876474 0.088394175271269249;0.19659349028765574
0.12502695316576493;0.18409226526909889 0.14399823745976495;-
0.58380785250780409 0.3589283290576642;0.083870892707118477 -
0.61142517350179093;-0.16151586877558996 -0.37462368611513425;-
0.74401663707020282 1.657729314357947;0.65929379418558354
0.12644880461132937;-1.2843960536150323
0.24174771072887249;0.8280166735062795
0.088916412904514445;1.0015485372678126 0.9338791550697717;-
0.86632468625560299 0.46468850034190129;1.1827695376647753
0.67768211491201080;0.34457037189793649 -0.6517526131942436;-
0.56890568741421688 -0.038343563275108357];

% Layer 2
b2 = 0.2066661884416337;
LW2_1 = [-7.5282027120455999e-06 -0.22224990737272685 -
0.00090216182118029061 0.93586938855310431 -0.00030696819694619098 -
9.6614850534229326 0.082344867813933423 -0.082020408032251552
1.3918709970901183e-05 1.5015888789864605e-05 5.3751311188660607e-07 -
4.5020856295286962e-05 9.0462465125700985e-07 -1.9713557967364456e-05
0.028451183194952327 -6.4616872354644856e-06 -5.3623652331341362e-06 -
8.4339783455274878e-06 0.00011266965289151422 3.5797027594858029e-05];

% Output 1
```

```
y1_step1.ymin = -1;
y1_step1.gain = 0.00186094268414719;
y1_step1.xoffset = -537.346928490387;

% ===== SIMULATION ========

% Dimensions
TS = size(x1,2); % timesteps

% Input 1 Delay States
xd1 = mapminmax_apply(xi1,x1_step1);
xd1 = [xd1 zeros(3,1)];

% Layer Delay States
ad2 = [ai2 zeros(1,1)];

% Allocate Outputs
y1 = zeros(1,TS);

% Time loop
for ts=1:TS

      % Rotating delay state position
      xdts = mod(ts+1,3)+1;
      adts = mod(ts+1,3)+1;

    % Input 1
    xd1(:,xdts) = mapminmax_apply(x1(:,ts),x1_step1);

    % Layer 1
    tapdelay1 = reshape(xd1(:,mod(xdts-[1 2]-1,3)+1),6,1);
    tapdelay2 = reshape(ad2(:,mod(adts-[1 2]-1,3)+1),2,1);
    a1 = tansig_apply(b1 + IW1_1*tapdelay1 + LW1_2*tapdelay2);

    % Layer 2
    ad2(:,adts) = b2 + LW2_1*a1;

    % Output 1
    y1(:,ts) = mapminmax_reverse(ad2(:,adts),y1_step1);
end

% Final delay states
finalxts = TS+(1: 2);
xits = finalxts(finalxts<=2);
xts = finalxts(finalxts>2)-2;
finalats = TS+(1:2);
ats = mod(finalats-1,3)+1;
xf1 = [xi1(:,xits) x1(:,xts)];
af2 = ad2(:,ats);
end

% ===== MODULE FUNCTIONS ========

% Map Minimum and Maximum Input Processing Function
function y = mapminmax_apply(x,settings)
  y = bsxfun(@minus,x,settings.xoffset);
  y = bsxfun(@times,y,settings.gain);
  y = bsxfun(@plus,y,settings.ymin);
end

% Sigmoid Symmetric Transfer Function
```

```
function a = tansig_apply(n,~)
  a = 2 ./ (1 + exp(-2*n)) - 1;
end

% Map Minimum and Maximum Output Reverse-Processing Function
function x = mapminmax_reverse(y,settings)
  x = bsxfun(@minus,y,settings.ymin);
  x = bsxfun(@rdivide,x,settings.gain);
  x = bsxfun(@plus,x,settings.xoffset);
end
```

9.2 Fuentes de R

En el código de R solo hay un fichero que es el que se llama desde la aplicación de Matlab, este script solo se encarga de realizar la descomposición de los datos mediante SSA una vez que se ha quitado la tendencia de los datos.

El código se muestra a continuación:

9.2.1 AnalisisSSA.Rscript

```
###### Inicialización de librería ######

library(Rssa)

###### Captura de datos SSA ######

#Captura de datos
datos<-read.csv("Datos_Detrended.csv",header=FALSE,row.names=NULL)

#Formateo de datos como serie temporal
datos=as.ts(as.numeric(unlist(datos[1],use.names=FALSE)))


###### Descomposición SSA ######

longitudserie=length(datos)
datosSSA=datos[1:longitudserie]
#Elección de ventana
L=48*7*4;

#Descomposición SSA
sobject1<-ssa(datosSSA,L=L)
summary(sobject1)

#Agrupación
componente1=c(1,2)
componente2=c(3,4)
componente3=c(5,6)
componente4=c(7,8)
componente5=c(9,10)
componente6=c(11,12)
componente7=c(13,14)
```

```
componente8=c(15,16)
componente9=c(17,18)
componente10=c(19,20)

#Reconstrucción
recon1 <- reconstruct(sobject1, groups =
list(componente1,componente2,componente3,componente4,componente5,compo
nente6,componente7,componente8,componente9,componente10))
residuo1 <- residuals(recon1)

#Separación de componentes
Componente1=recon1$F1
Componente2=recon1$F2
Componente3=recon1$F3
Componente4=recon1$F4
Componente5=recon1$F5
Componente6=recon1$F6
Componente7=recon1$F7
Componente8=recon1$F8
Componente9=recon1$F9
Componente10=recon1$F10

###### Predicción SSA ######

PredSSA<-forecast(sobject1,groups
=list(c(componente1,componente2,componente3,componente4,componente5,co
mponente6,componente7,componente8,componente9,componente10)), method =
"recurrent", len = 48)


###### Exportación de variables ######

write.csv(Componente1, file = "Componente1.csv")
write.csv(Componente2, file = "Componente2.csv")
write.csv(Componente3, file = "Componente3.csv")
write.csv(Componente4, file = "Componente4.csv")
write.csv(Componente5, file = "Componente5.csv")
write.csv(Componente6, file = "Componente6.csv")
write.csv(Componente7, file = "Componente7.csv")
write.csv(Componente8, file = "Componente8.csv")
write.csv(Componente9, file = "Componente9.csv")
write.csv(Componente10, file = "Componente10.csv")
write.csv(residuo1, file = "Residuo.csv")
write.csv(PredSSA$mean, file="ForecastSSA.csv")
```

Estos serían todos los archivos fuentes que se utilizan en la aplicación, están disponibles para su consulta y/o edición en la carpeta de la aplicación en la carpeta “Fuentes”.

10. PRESUPUESTO

Para la realización de este sistema ha sido necesaria la compra de una serie de herramientas y la dedicación de una serie de horas, por lo que en este apartado se recoge el desglose de costes.

Coste del material:

Los costes de materiales se recogen en la Tabla 2.

Esta tabla recoge los costes de equipamiento hardware y software.

Objeto	Precio Unitario(€)	Unidades	Precio Total (€)
MSI GE72 (i7 4 GHz)	1100	1	1200
Monitor LG FHD	250	1	250
Matlab	120	1	120
Rstudio	0	1	0
Licencia de Office	0	1	0
Total (IVA incluido)			1570

Tabla 2. Costes de materiales

Costes de mano de obra

En la Tabla 3 se muestra el precio correspondiente a la mano de obra empleada para la realización de la solución.

Se han empleado aproximadamente 300 horas (5 meses) para la realización del proyecto. De estas 300 horas, aproximadamente unas 240 horas (4 meses) han consistido en labores de investigación y desarrollo software, mientras que el resto del tiempo empleado se ha dedicado a trabajos de realización de la documentación necesaria.

Actividad	Precio (€)	Tiempo(Meses)	Precio Total (€)
Ingeniería	1600	4	6400
Escritor	1000	1	1000
Total (IVA incluido)			7400

Tabla 3. Costes de mano de obra

Coste de ejecución material:

En la Tabla 4 se encuentran los gastos totales de la realización del proyecto:

Coste	TOTAL(€)
Costes Materiales	**1570**
Mano de obra	7400
Total	8970

Tabla 4. Coste de ejecución material

Gastos generales y beneficio industrial:

En este apartado se recogen los gastos producidos por el uso de las instalaciones donde se ha desarrollado el proyecto más el beneficio industrial. Se estima que es un 20% del coste de ejecución material.

Gastos generales y beneficio industrial	**1794€**

Presupuesto de ejecución por contrata:

Se calcula sumando el presupuesto de ejecución material más los gastos generales y de beneficio industrial. El resultado se muestra en la Tabla 5.

Coste	TOTAL(€)
Coste de ejecución material	**8970**
Gastos generales y beneficio industrial	1794
Total	10764

Tabla 5.Total de ejecución por contrata

Honorarios:

Los honorarios facultativos por la ejecución del proyecto se determinan de acuerdo a las tarifas de los honorarios de los ingenieros en trabajos particulares. De acuerdo con la ley 7/1997, de medidas liberalizadoras en materia de suelo y de Colegios Profesionales los Honorarios Profesionales son libres y responden al libre acuerdo entre el profesional y su cliente. Se aplica un coeficiente razonable de un 10% debido a que se trata de un proyecto de i+D.

Los resultados se encuentran en la Tabla 6.

Coste	TOTAL(€)
Importe Ejecución por contrata	**10764**
Honorarios	1076
Total	11840

Tabla 6.Coste Total del proyecto

Por lo que se concluye que el coste total del proyecto es de 11.840€.

11. MANUAL DE LA APLICACIÓN

11.1 Introducción

El objetivo de esta sección es proporcionar al usuario de la aplicación un manual para su correcto uso, ya que la aplicación serviría para un historial distinto al que se da en este Trabajo de Fin de Grado.

Esta aplicación se ha desarrollado para facilitar el uso de la solución y para que sea una interfaz amigable con un usuario final que no tiene por qué estar familiarizado con los mecanismos internos de la misma.

11.2 Requisitos del Sistema

En el estado actual de la solución, la interfaz se ejecuta desde el entorno Matlab, pero en caso necesario, podría ser compilada y ejecutada desde cualquier ordenador sin necesidad de tener Matlab instalado. Como no es el objetivo de este TFG, actualmente un requisito es tener Matlab.

El otro requisito para poder ejecutar la aplicación es tener instalado alguna versión del intérprete de R, ya que parte del código está implementado en R, como ya se comentó anteriormente.

La aplicación ha sido diseñada para trabajar sobre el sistema operativo Windows.

La aplicación ha sido diseñada con Matlab 2016b [31], pero debería ser compatible con versiones anteriores y posteriores.

Cumpliendo estos dos requisitos, el programa puede ser ejecutado.

11.3 Proceso de instalación

Para poder utilizar la aplicación, hay que disponer de la carpeta llamada “Aplicación”, que contiene todos los archivos necesarios para el correcto funcionamiento del programa.

Previo al uso del mismo, hay que seguir dos sencillos pasos para que el entorno Windows ejecute automáticamente el código R:

1. Se debe tener instalada una versión del intérprete de R, puede ser descargado en el enlace [32].

2. Una vez se ha descargado e instalado el programa, hay que asociar la extensión. Rscript al programa mediante el intérprete de comandos de Windows, hay que abrir el intérprete de comandos como administrador:
   Inicio-Buscador-"cmd"-Botón Derecho-Run as administrator

3. Abierto cmd como administrador, hay que ejecutar la siguiente línea:
   ASSOC .Rexec=RScriptExecutable

4. A continuación, hay que buscar la carpeta donde se ha instalado R (Suele estar en C:\Program Files\R\R-3.3.2\bin\x64) y buscar un ejecutable llamado Rscript.exe, localizado el directorio, se copia el path y se introduce en el siguiente comando:
   FTYPE RScriptExecutable="path"\Rscript.exe %1 %*
   Ejemplo:
   FTYPE RScriptExecutable= C:\Program Files\R\R-3.3.2\bin\x64\Rscript.exe %1 %*

Con estos pasos, el sistema debería estar listo para ejecutar la aplicación.

Los pasos también se pueden encontrar en el enlace de la referencia [33].

Si falla algo, siempre se puede asignar manualmente a la extensión Rscript el ejecutable Rscript.exe desde el panel de control-Programas predeterminados.

11.4 Uso de la aplicación

Para ejecutar la aplicación, abra Matlab, cambie el directorio de trabajo a la carpeta donde se encuentra la aplicación llamada “Aplicación” y escriba en la línea de comandos “SSA_Predictor” sin comillas, debería aparecer la ventana mostrada en la Figura 39:

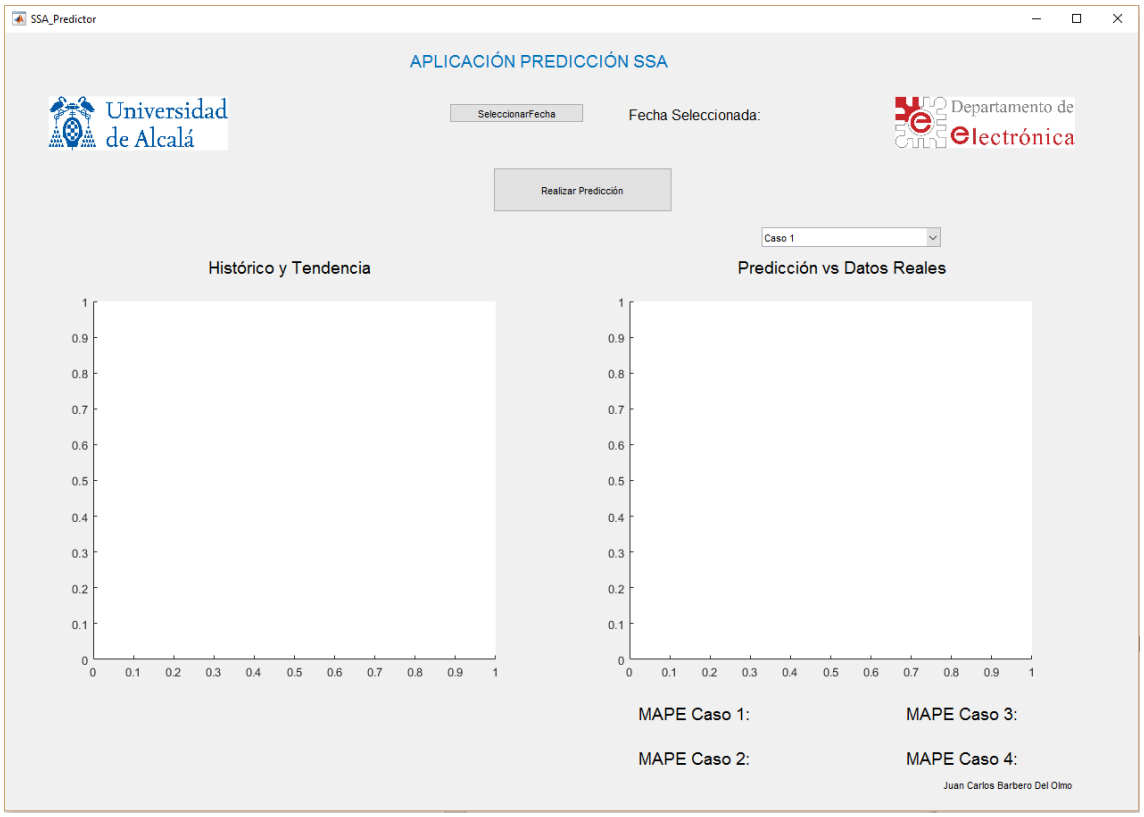


Figura 39. Captura de pantalla principal de la aplicación

Para realizar predicciones de una cierta fecha, hay que hacer clic en el botón de seleccionar fecha y emergerá un calendario que permite seleccionar día mes y año.

Con el historial original del TFG solo se pueden seleccionar todos los días de 2014 y 2015, si se seleccionan fechas fuera de ese periodo, el programa dará un error y la salida no está garantizada.

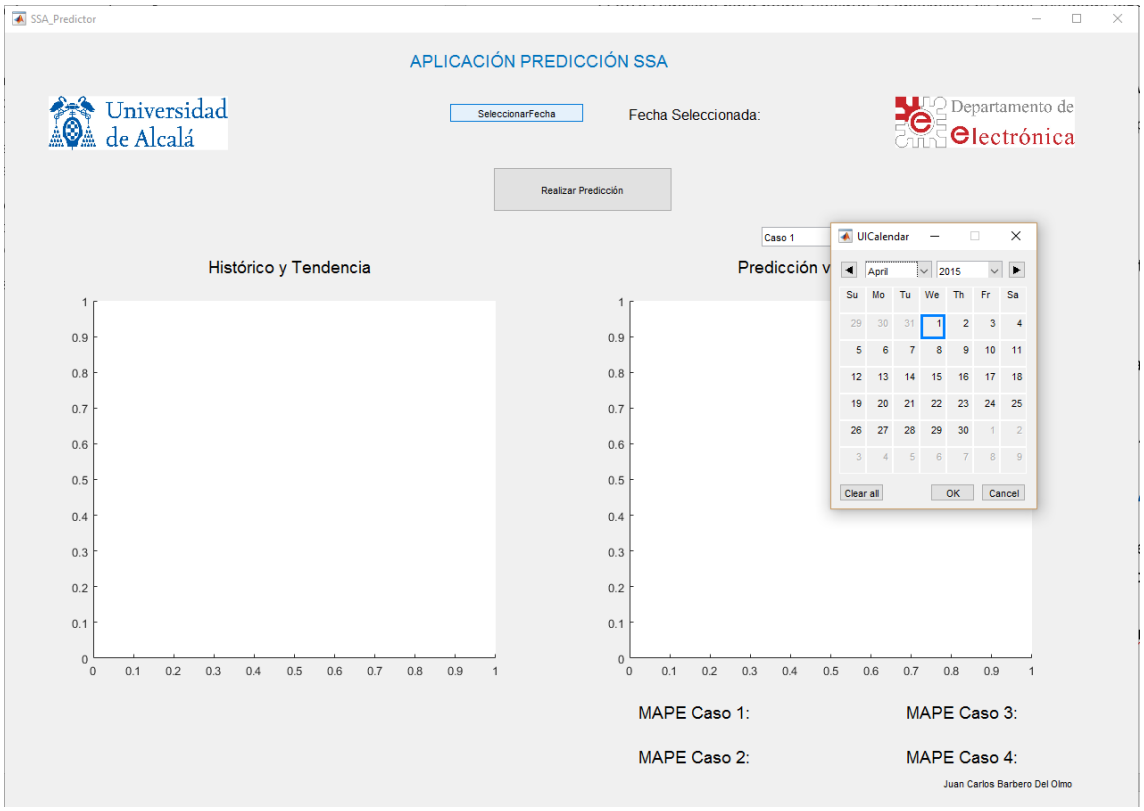


Figura 40. Captura de la elección de día en el calendario

Como se muestra en la Figura 40, simplemente se selecciona el calendario el día deseado y se le da a "OK".

A continuación, se pulsa el botón de "Realizar Predicción y la aplicación se encarga de realizar los 4 casos de predicción que se comentaron en los puntos anteriores. Este proceso puede tardar hasta 1 minuto dependiendo del hardware del que se disponga (Hay que tener paciencia, finalmente debería aparecer una pantalla similar a la de la Figura 41:

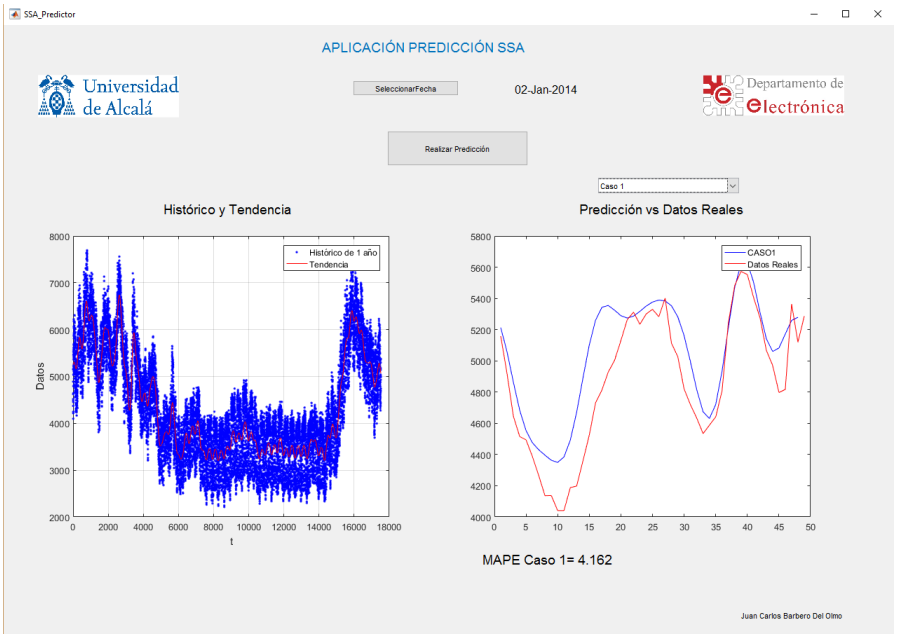


Figura 41. Captura de resultados de la aplicación

Además, como funcionalidad añadida, permite cambiar de un caso a otro de predicción, se dibuja siempre en la gráfica de la izquierda los datos que se han utilizado de historial y la línea de tendencia que se calcula.

En los ejes de la derecha se muestra una de las predicciones o todas a la vez en función de la opción que se haya elegido en el desplegable que hay encima.

El desplegable permite elegir entre los cuatro casos definidos en la explicación del algoritmo y una comparativa de los cuatro a la vez.

Debajo de la gráfica aparece el MAPE (Ecuación 6) que corresponda según la elección, en tanto por ciento.

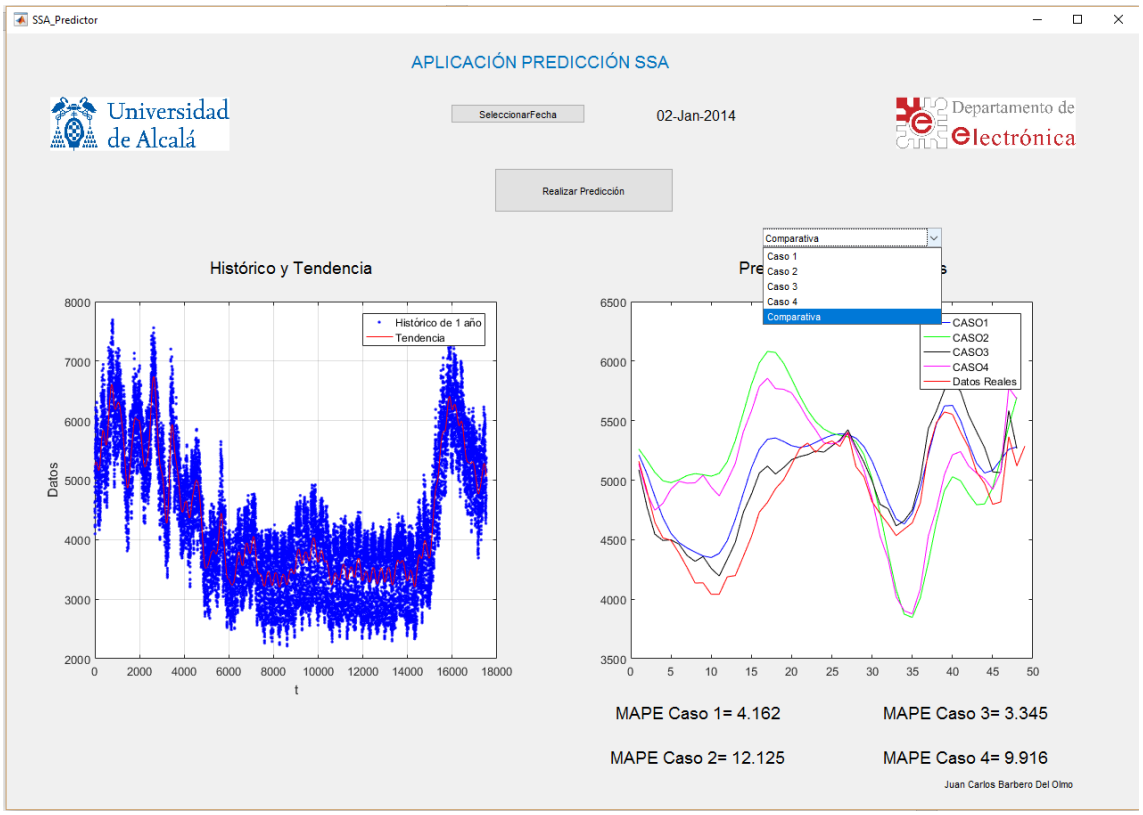


Figura 42. Captura de comparativa de la aplicación

Tal como se muestra en la Figura 42, este es el funcionamiento de la aplicación, como se puede observar, es extremadamente intuitiva, además, sin necesidad de relanzarla, se pueden hacer varias predicciones cambiando la fecha y volviendo a pulsar el botón de realizar predicción.

11.5 Modificación del historial

Como se ha comentado anteriormente, el historial de la aplicación puede ser reducido, ampliado o modificado, para ello, simplemente hay que aportarle el formato adecuado.

En el directorio de la aplicación hay dos archivos llamados “Datos.mat” y “Fechas.mat”, estos dos archivos contienen sendos vectores columna, Datos con los datos de consumo en MWh y Fechas con las fechas de dichos consumos.

Es importante que los datos estén dados cada media hora, ya que el algoritmo está preparado para predecir 48 muestras (24h), otro paso implicaría tener que reentrenar las redes neuronales y volver a configurar el análisis SSA.

Además, sería conveniente para que el algoritmo fuese preciso que se volviese a supervisar el funcionamiento del análisis SSA si se cambia de zona en la que se toman las medidas o si las medidas corresponden a otros patrones de comportamiento (Probablemente habría que cambiar la ventana del algoritmo), por lo que, si se desea realizar este cambio, el lector queda avisado de lo que puede ocurrir si no se revisa el algoritmo.

El algoritmo SSA se encuentra escrito en un fichero llamado AnalisisSSA.Rscript y puede ser abierto por cualquier visor de texto estándar, aunque se recomienda usar un entorno de programación de lenguaje R como RStudio.

Las redes neuronales podrían ser reentrenadas tal y como se explicó en la sección de Entrenamiento de las redes.

12. BIBLIOGRAFÍA

[1] J. M. Chambers y E. T. J. Hastie, «Statistical Models in S».

[2] R. Gentleman y R. Ihaka, «The R language,» *Proceedings of the 28th Symposium on the Interface,* 1997.

[3] «http://www.rte-france.com/en/eco2mix/eco2mix-telechargement-en,» [En línea]. Available: http://www.rte-france.com/en/eco2mix/eco2mix-telechargement-en.

[4] N. Golyandina, V. Nekrutkin y A. A. Zhigljavsky, Analysis of Time Series Structure: SSA and Related Techniques, ISBN 9781584881940.

[5] E. A. G. R. W. a. W. A. Y. I. Ghalehkhondabi, «An overview of energy demand forecasting methods published in 2005–2015,» 2016.

[6] T. H. D. Ngo, The Box-Jenkins Methodology for Time Series Models, 2013.

[7] H.-T. Yang, C.-M. Huang y C.-L. Huang, «Identification of ARMAX model for short term load forecasting: an evolutionary programming approach».

[8] M. Rossi y D. Brunelli, «Forecasting data centers power consumption with the Holt-Winters method».

[9] E. Valakevicius y M. Brazenas, «Application of the Seasonal Holt-Winters Model to Study Exchange Rate Volatility».

[10] C. Cortes y V. Vapnik, «Support-vector networks,» 1995., p. 273–297.

[11] H. Hassani, “Singular Spectrum Analysis: Methodology and Comparison”.

[12] H. Hassani, R. Mahmoudvand y M. Zokaei, «Separability and Window Length in Singular Spectrum Analysis,» 2011.

[13] R. Vautard, P. Yiou y M. Ghil, «Singular-spectrum analysis: A toolkit for short, noisy chaotic signals».

[14] U. Kumar y V. Jain, «Time series models (Grey-Markov, Grey Model with rolling mechanism and singular spectrum analysis) to forecast energy consumption in India».

[15] E. Fiesler, Neural Network Topologies.

[16] K. Hornik, Multilayer feedforward networks are universal approximators, 1989.

[17] «Neural Designer,» [En línea]. Available: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network.

[18] H. B. W. M. Yu, «Levenberg–Marquardt Training,» de *Intelligent Systems*, p. Capítulo 12.

[19] F. Burden y D. Winkler, «Bayesian regularization of neural networks».

[20] T. Bayes, «An Essay towards solving a Problem in the Doctrine of Chances,» 1763.

[21] M. F. Moller, «A Scaled Conjugate Gradient Algorithm for Fats Supervised Learning».

[22] M. T.Hagan, H. B. Demuth y M. Beale, Neural Network Design, 1996.

[23] «RStudio,» [En línea]. Available: https://www.rstudio.com/.

[24] «Rssa,» [En línea]. Available: https://cran.r-project.org/web/packages/Rssa/Rssa.pdf.

[25] A. Kaylor Cline y I. S. Dhillon, «Computation of the Singular Value Descomposition».

[26] N. Golyandina, V. Nekrutkin y A. A. Zhigljavsky, «Analysis of Time Series Structure,» de *Analysis of Time Series Structure*, pp. 107-120.

[27] N. Golyandina, V. Nekrutkin y A. A. Zhigljavsky, «SSA Vector Forecasting,» de *Analysis of Time Series Structure*.

[28] N. Golyandina, V. Nekrutkin y A. A. Zhigljavsky, «Bootstrap con diferentes técnicas,» de *Analysis of Time Series Structure*, pp. 127-139.

[29] R. Mahmoudvand, A. Fatemeh y P. Canas Rodrigues, «Forecasting mortality rate by singular spectrum analysis,» 2014.

[30] H. Li, L. Cui y S. Guo, «A Hybrid Short-Term Power Load Forecasting Model Based on the Singular Spectrum Analysis and Autoregressive Model,» 2014.

[31] «Matlab Guide,» [En línea]. Available: https://es.mathworks.com/discovery/matlab-gui.html.

[32] «CRAN PROJECT,» [En línea]. Available: https://cran.r-project.org/bin/windows/base/old/3.3.2/.

[33] «Manual Rscript,» [En línea]. Available: http://www.r-datacollection.com/blog/Making-R-files-executable/.

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá