Grado en Ingeniería en Tecnologías de Telecomunicación

Trabajo Fin de Grado

APLICACIÓN DE ALGORITMOS SSA PARA LA DETECCIÓN DE ERRORES DE MEDIDA EN REDES DE DISTRIBUCIÓN

Autor: Alejandro López Rodríguez

Tutor/es: Francisco Javier Rodríguez Sánchez

2017

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

Grado en ingeniería en Tecnologías de Telecomunicación

Trabajo Fin de Grado
Aplicación de algoritmos SSA para la detección de errores de medida en redes de distribución

Autor: Alejandro López Rodríguez

Tutor/es: Francisco Javier Rodríguez Sánchez

TRIBUNAL:

Presidente: José Antonio Jiménez Calvo

Vocal 1º: Pedro Martín Sánchez

Vocal 2º: Francisco Javier Rodríguez Sánchez

FECHA: 15 de septiembre 2017

ÍNDICE

Tabla de contenido

Índice de Figuras

Índice de tablas

I. Resumen

En los últimos años, debido a los grandes avances en las tecnologías electrónicas y de comunicación, se han ido introduciendo nuevos equipos en los centros de transformación que permiten una total administración de los mismos. Esto resulta de gran utilidad, ya que permite poder monitorizar voltajes y corrientes, que se pueden utilizar para control y protección.

En este trabajo, utilizando datos históricos de estas medidas, podremos predecir el consumo eléctrico en un centro de transformación mediante técnicas de aplicación de algoritmo *Singular Spectrum Analysis* permitiéndonos detectar errores sistemáticos de medida de forma prácticamente instantánea.

Para la predicción de carga se utilizará el algoritmo *Singular Spectrum Analysis* implantado en librerías de lenguaje R, y para la representación gráfica y detección de errores, utilizaremos las herramientas que otorga MATLAB.

PALABRAC CLAVE: detección de errores, predicción de carga, SSA, Componentes principales, Subestación eléctrica, análisis de métodos de predicción.

II. Abstract

In recent years, due to the great advances in electronic and communications technology, Secondary Substations are being provided with equipment that allows their full management. This is really useful because allow us to monitor voltage and current measurements, that we can use in control and protection of Secondary Substations.

In this work, using historical data from these measurements, we will be able to forecast the electric power consumption in a Secondary Substation using techniques of SSA algorithm, allowing us to make a fast error detection.

For the Load Forecast we will use SSA algorithm implemented in a R package, and for graphic plotting and error detection we will use MATLAB software.

III. Acrónimos y abreviaturas

- SSA: *Singular Spectrum Analysis*
- IDE: *Integrated Drive Electronics* (o Entorno de Desarrollo Integrado)

IV. Resumen Extendido

En los últimos años, debido a los grandes avances en las tecnologías electrónicas y de comunicación, se han ido introduciendo nuevos equipos en los centros de transformación que permiten una total administración de los mismos. Esto resulta de gran utilidad, ya que permite poder monitorizar voltajes y corrientes, que se pueden utilizar para control y protección, y, también, planear estrategias, tanto comerciales, como de diseño. Pero, como contrapartida, estos sistemas pueden tener errores en las medidas, que pueden llevar a tomar decisiones erróneas de funcionamiento, lo cual puede afectar negativamente al funcionamiento de dichos centros de transformación.

Por otro lado, existen datos históricos de carga y consumo, que, mediante diferentes técnicas, permiten realizar estimaciones de carga con una alta precisión.

Dicho esto, ambas ideas se pueden relacionar. Realizando métodos de estimación de carga, utilizando los registros históricos de consumo y carga en los centros de transformación (CT), es posible predecir con bastante fiabilidad cómo se comportará un CT y comparando estas estimaciones con los datos medidos por los equipos "inteligentes" instalados en los CTs, es posible detectar errores sistemáticos en estos sistemas, como pueden ser errores de ganancia y de *offset.*

La predicción del consumo eléctrico ha sido un tema de gran relevancia en los últimos años, en especial para las compañías privadas de este sector, ya que un avance y mejoras en la predicción de consumo pueden suponer nuevas estrategias a desarrollar por las empresas, permitiendo un mejor aprovechamiento de los recursos, conociendo de forma detallada cuál será el consumo eléctrico en las próximas semanas a cada hora. Esto permite también realizar predicciones sobre generación de energía, y, así, ajustar los recursos necesarios para cubrir las necesidades predichas de consumo, permitiendo un ahorro de recursos.

En la actualidad las predicciones de generación eléctrica se vuelven cada vez más complicadas. Esto se debe a la cada vez mayor importancia de las fuentes de energía renovables

y el mayor impacto que tienen en la generación eléctrica, ya que entran en juego nuevas variables como es la meteorología, puesto que energías como la eólica o la solar, dependen directamente de ésta.

El objetivo de este trabajo se centra en la parte de predicción de consumo de potencia, permitiendo detectar errores en las medidas de los nuevos dispositivos electrónicos de las subestaciones. Para ello se utilizarán datos de consumo eléctrico en los últimos años, detectando tendencias y curvas de consumo.

En este trabajo se pretende realizar el proceso completo de análisis, por ello, se utilizarán registros de los últimos años de consumo, de la región francesa de Borgoña, ya que los datos de consumo eléctrico en Francia son públicos, pero a nivel de regiones, o estatal. Dado que el interés de este estudio sería realizar los cálculos para un CT, se toma esta región en consideración debido a que, de las disponibles, es la que presenta un menor consumo, permitiendo asemejar en mayor medida los datos a una zona pequeña como puede ser un CT.

Una vez obtenidos los datos, el siguiente paso a seguir es el de normalizarlos y adecuarlos para facilitar el posterior análisis. Para ello, utilizando la herramienta MATLAB, se pretende realizar una preparación de los datos para su tratamiento en formato CSV. Tras esto, se utilizarán las herramientas que proporcionan las librerías de código abierto del lenguaje de programación R y el entorno Rstudio. Concretamente se utilizará un paquete llamado RSSA, que utiliza algoritmos de análisis singular de espectro *(Singular Spectrum Analisis*, SSA). Esto permitirá obtener las componentes de mayor potencia de los datos, eliminando las componentes de muy baja potencia, eliminando ruido de la señal. De esta forma, resultará mucho más sencillo realizar una buena predicción, que, por otro lado, también derivará en pérdidas de determinadas componentes de baja potencia, que, a fin de cuentas, conlleva una pérdida de información. Por ello, se debe llegar a un compromiso entre sencillez de predicción y calidad de la misma, eligiendo de forma adecuada qué componentes se pueden despreciar.

Una vez que se tenga la información preparada, se procederá a probar distintos métodos de predicción de consumo, que proporciona el propio paquete de RSSA. Una vez realizadas las pruebas, se podrán comparar los distintos métodos, y observar las diferencias entre ellos.

Para la última fase del trabajo, se utiliza MATLAB, para la parte gráfica del proyecto, ya que su ploteado de gráficas aporta más opciones que las disponibles actualmente en R. En esta parte se compararán datos reales de consumo, con predicciones realizadas previamente, pudiendo observar la exactitud de la predicción, y, por otro lado, realizar un *script* que permita detectar, utilizando la señal predicha y la real, si en la real ha ocurrido algún tipo de error, como pueden ser, errores de ganancia o de *offset*.

En este proyecto, como se comentará más adelante en detalle, se parte de un estudio previo y amplio basado en las predicciones de consumo eléctrico y demanda de potencia.

V. Estado del arte

En las últimas décadas, grandes avances en las tecnologías de la comunicación e información han acelerado la introducción y el desarrollo de nuevas tecnologías de banda ancha en sistemas de potencia. Esta dinámica ha facilitado la automatización de los sistemas de potencia en los centros de transformación. Con el objetivo de aprovechar esta tecnología actual, se creó el estándar global IEC 61850 para la automatización de subestaciones, el cual está ganando mucha popularidad en estos años [2].

Debido a estos avances, antiguas subestaciones eléctricas y los nuevos CTs están cambiando para adaptarse a este estándar global. Nuevos elementos electrónicos se encargan de las mediciones de corriente y voltaje, que se usan para controlar y proteger subestaciones eléctricas y CTs. Estas ventajas de control y protección, tambien tienen una parte negativa, y es que pueden ser afectados por errores de medición en el proceso de adqusición de señales, lo que podría llevar a operaciones de protección ineficaces [3].

Por otro lado, el perfil de carga y el consumo eléctrico están fuertemente relacionados con sus datos históricos, dependiendo de ciertas variables, como pueden ser factores económicos o temporales. Esto ha llevado, con el objetivo de controlar la generación y consumo electrico, a que aparezcan nuevos métodos de predicción de carga, como señnalan K.-L. Ho *et al.* y D.Parh *et al.* en [4] y [8].

En este trabajo se pretende, utilizando técnicas de predicción de carga, comparar valores reales con valores estimados y así poder detectar diferentes tipos de errores sistemáticos, como son los errores de ganancia y de *offset*.

Para ello se empleará un algoritmo llamado SSA, el cual se describe en detalle en [5].

El ámbito de la detección de errores se ha tratado en muchos articulos cientficos de diferentes ambitos, como por ejemplo en el articulo de S. Naidu *et al.* [9] donde se discute el proceso de detección de errores en sensores utilizando redes neuronales, o en el artículo “Sensor fault diagnosis method based on fractal dimension” [10] donde se implementa un método basado en la dimensión fractal para detectar errores. En el documento de M. Napolitano *et al.*

[11] también mediante redes neuronales se habla de un método de detecction de errores de sensores en un sistema de control de vuelo. La detección de errores puede ser muy compleja, debido a la multitud de factores que pueden afectar a una señal. Por ello en este documento se tratan solo dos tipos de errores, el de ganancia y el de *offset*, dentro del ámbito del consumo eléctrico en subestaciones y CTs [6].

Centrándose en la detección de errores de CTs, en el trabajo de S. Sheng *et al.* [12] se utiliza identificación por patrones para detectar errores de medida. Con el fin de lograr una alta precisión en la identificación de patrones dentro del plazo impuesto por los sistemas de protección, se implementan los algoritmos *Radial Basis Function Neural Network* (RBFNN) y *Orthogonal Least Square* (OLS). Para detectar estos errores se entregan medidas de los CTs a la red neuronal y devuelve la probabilidad de que ocurra un error.

VI. Contenido

A. Prefacio

1. Introducción

Debido a los grandes avances que sufre la tecnología continuamente, el mundo avanza hacia una digitalización en todos los aspectos. Un ejemplo de esto es el desarrollo de los centros de transformación. Estos centros, en los últimos años, están migrando al concepto de redes inteligentes (*smart grids*), llevándolos a una mayor digitalización, introduciendo nuevos sistemas de medición de consumos, de control de dispositivos, de mecanismos de seguridad, detección de errores, etc. Esto permite entender y controlar mucho mejor todo lo que ocurre en ellos, de forma que permite realizar procesos óptimos, lo que se traduce en un ahorro energético y económico. Concretamente, en este trabajo se centra en los dispositivos de medición, mediante los cuales se tienen datos históricos de consumo y datos de consumo en tiempo real.

Con los datos históricos, se podrán realizar estimaciones de consumo, analizando distintos métodos para ello y viendo sus resultados, y, por otro lado, con los datos en tiempo real se podrán comparar con el estimado para detectar errores en tiempo real, permitiendo una rápida actuación para corregirlos si fuese necesario.

2. Objetivos

Los principales objetivos de este trabajo son los siguientes:

- Analizar mediante algoritmos SSA datos históricos de consumo.
- Llevar acabo predicciones con un error mínimo respecto a la realidad.
- Realizar un algoritmo de detección de errores, que permita detectar errores de ganancia o de offset.
- Proponer un camino a seguir para continuar profundizando en este tema

B. Estudio teórico

1. Técnicas y programas utilizados

En esta sección se habla sobre las distintas técnicas y herramientas utilizadas para llevar a cabo este trabajo.

En primer lugar, se revisará el algoritmo SSA, sus técnicas de predicción y las distintas variantes de dicho algoritmo que se usarán a lo largo del trabajo.

Tras esto, se comenta la parte de detección de errores, y, concretamente, se explican en detalle los dos tipos de errores tenidos en cuenta en este trabajo.

Por último se va a hablar sobre las distintas técnicas y herramientas utilizadas para llevar a cabo este trabajo. En primer lugar, se introducirá la herramienta de *Matlab*, herramienta que se utilizará para preparar el contenido antes de que se analice, para "ploteado" de gráficas y para la realización de un algoritmo de detección de errores. También se menciona brevemente la herramienta *Rstudio*, que contiene las librerías de uso de SSA.

a) SSA

SSA es una nueva técnica de análisis de series temporales. Esta técnica se puede utilizar para un análisis de series temporales clásicas, estadísticas multivariantes, geometría multivariante, sistemas dinámicos y procesamiento de señal. SSA se puede utilizar en distintas áreas, como es la física, las matemáticas, la meteorología, medicinal o, como en este caso, ámbito eléctrico. En definitiva, toda área que se comporte como una serie temporal, puede ser analizada mediante SSA.

El algoritmo SSA se basa en dos diferentes fases, una primera de descomposición, y una segunda de reconstrucción. En la primera etapa, se puede descomponer una serie temporal en distintos componentes, permitiendo obtener los de mayor potencia. En la segunda, se

reconstruye la serie temporal, con la parte que más interese de la señal, como puede ser la tendencia, el ruido, la señal sin ruido, etc.

El primer paso necesario sería definir el tamaño de la ventana a analizar (L) que debe ser como máximo la mitad del total de muestras a analizar. Tras esto se convierte la serie temporal unidimensional $X = (X_1,...,X_T)$ en una matriz multi-dimensional conocida como matriz de trayectoria:

$$\mathbf{X} = [X_1 : \ldots : X_K] = (x_{ij})_{i,j=1}^{L,K} = \begin{bmatrix} x_1 & x_2 & x_3 & \ldots & x_K \\ x_2 & x_3 & x_4 & \ldots & x_{K+1} \\ x_3 & x_4 & x_5 & \ldots & x_{K+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_L & x_{L+1} & x_{L+2} & \ldots & x_N \end{bmatrix} \quad (1)$$

La matriz de Trayectoria es una matriz de Hankel, esto es que sus anti diagonales son constantes.

Tras esto se procede a descomponer la matriz de trayectoria. Para descomponerla, primero se obtiene $S=XX^T$ de donde se consiguen sus autovalores y autovectores.

$$XX^T = PDP^T \quad (2)$$

- X = Matriz de trayectoria
- P = Matriz ortogonal de autovectores de XX^T
- D = Matriz diagonal que contiene los autovalores de XX^T ordenados de mayor a menor.

A continuación, se lleva a cabo la selección de autovectores que interesan de la matriz, permitiendo obtener, como se comentó con anterioridad, la señal sin ruido, el ruido, la tendencia, o lo que sea de mayor interés en función del análisis a realizar.

Por último, se procede a la reconstrucción de la serie de la siguiente forma:

$$\tilde{X} = x_{i,j} = \sum_{k=1}^{L} P_{ik} P_{ik}^T X \quad (3)$$

Permitiendo obtener una nueva serie temporal unidimensional, que, pese a haber perdido información, será más sencilla de procesar y de predecir, ya que en el proceso se han eliminado componentes de poco interés. Por ello, hay que llegar a un compromiso sobre la información que se quiere manejar, la facilidad de proceso y el tamaño de ventana a aplicar, de forma que se pueda obtener el resultado esperado óptimo.

b) Detección de Errores

El error se define como la diferencia entre la salida obtenida y el valor ideal de la misma, el cual es el esperado.

$$\Delta x = x_r - x_i \tag{4}$$

En este TFG se considera como ideal la señal predicha, de esta forma, si el error supera un valor determinado se considera que ha habido un error de medición y/o de funcionamiento. Se deja un margen de error debido a que la predicción, por muy precisa que pueda llegar a ser, siempre tendrá ligeras diferencias entre el comportamiento real y el estimado, por lo que, de no dejar dicho margen, cualquier mínima diferencia podría hacer pensar que hay un error de medida, cuando, seguramente, se tratase de un error de predicción.

Existen gran variedad de errores que pueden verse reflejados en las medidas, desde errores sistemáticos a errores aleatorios. Como, por ejemplo, errores térmicos, errores de derivas temporales, errores de tolerancias, etc.

En este trabajo se tendrán en consideración dos tipos de errores, el error de *offset* y el error de ganancia.

- Error de *offset* es aquel que desplaza toda la señal una misma cantidad, constante en el tiempo o proporcional a la entrada aplicada al sistema. La forma de detectar si existe un error de *offset* habitual es comprobar la salida del sistema, cuando no hay señal a la entrada. De esta forma, teniendo un sistema conocido, se es capaz de observar el error de *offset* presente. En este trabajo, para detectar dicho error se comprobará que la señal medida se encuentra constantemente desviada de la predicha, de forma que siempre tenga el mismo error en todos sus puntos. A continuación, se puede ver un ejemplo de error de offset (Figura 1):

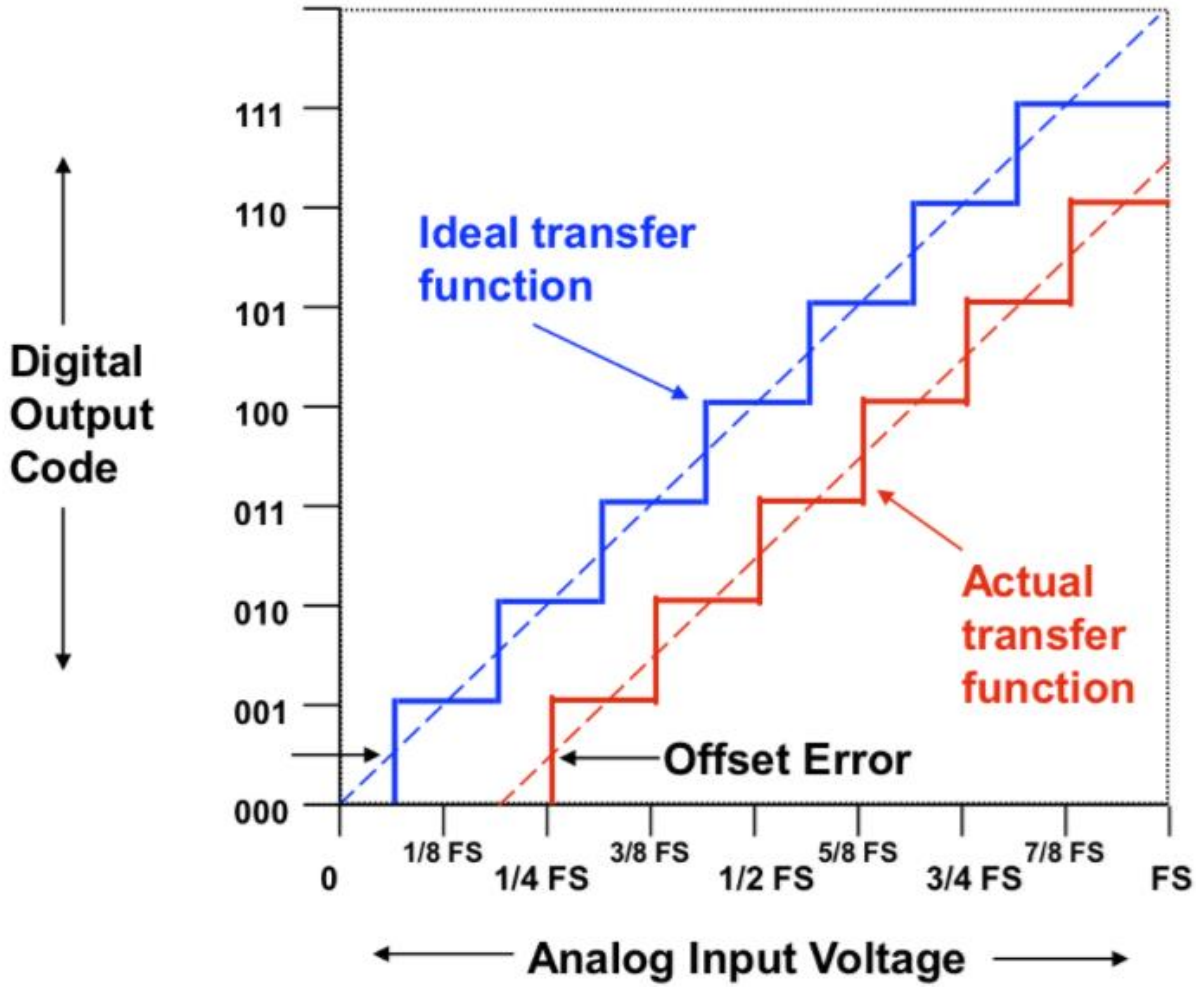


Figura 1: Función ideal vs función con error de offset de un ADC

- Error de ganancia: representa el error en la pendiente de la salida real, en comparación de la ideal. Este error, a diferencia del de *offset*, no es constante en todos sus puntos. Es proporcional a la señal de entrada, por lo que, a mayor valor, el error de ganancia será mayor. Por este motivo también se le conoce como error de pendiente (Figura 2):

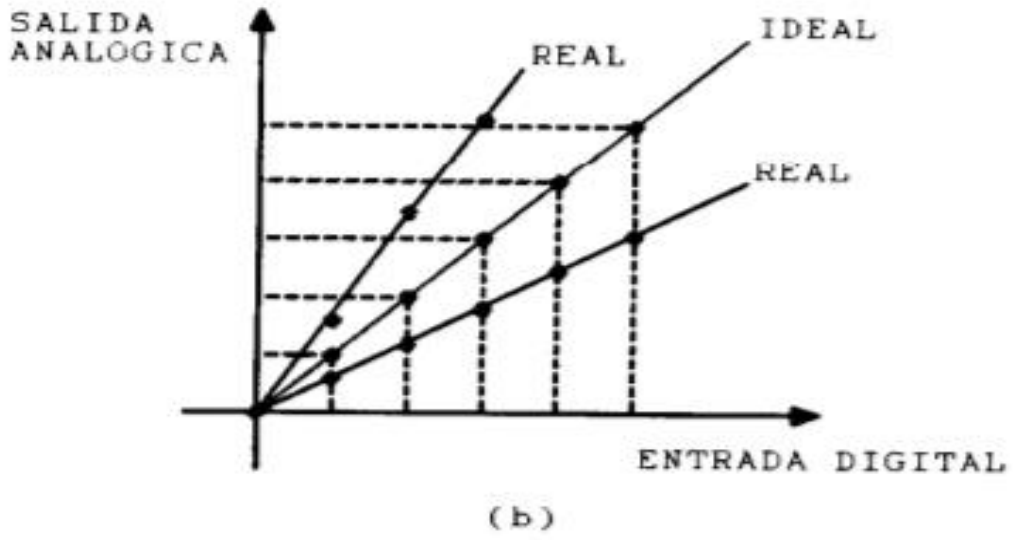


Figura 2: Función ideal vs función con error de ganancia de un DAC

En la medida de consumo eléctrico, estos errores no se aprecian con tanta facilidad, ya que la entrada no es lineal, sino la composición de muchas señales senoidales de distintas frecuencias, haciendo que sea más difícil medir con exactitud el tipo de error y el grado del mismo.

c) MATLAB

Matlab es un software informático creado en el año 1984 por el matemático y programador informático Cleve Moler. Es un software matemático, que permite cálculo matricial, funciones, representación de datos, etc.

Además de ser una potente herramienta de cálculo, también dispone de paquetes que aumentan sus prestaciones, como puede ser SIMULINK.

En este trabajo se va a utilizar *Matlab* en dos etapas diferenciadas, una para análisis y preparación de datos y otra para representación de resultados.

d) Rstudio

Rstudio es un Entorno de Desarrollo Integrado (IDE) para el lenguaje de programación R. Este IDE proporciona una consola, un editor de sintaxis, ayuda, autocompletado de código y muchas facilidades a la hora de trabajar con R. Su principal objetivo es proporcionar un entorno amigable para trabajar con R.

R es un lenguaje de programación orientado al estudio estadístico de software libre. Es por esto por lo que existen de un gran número de paquetes desarrollados por usuarios.

Por este motivo se ha utilizado R en este proyecto, para la utilización de un paquete llamado RSSA que contiene funciones del algoritmo SSA previamente programadas, que serán de gran utilidad para este trabajo.

2. Conclusiones

En este capítulo, correspondiente al estudio teórico, se ha explicado, de forma introductoria, las herramientas y procedimientos que se usarán a lo largo del trabajo, de forma que, tras esto, sea más sencillo hablar del proceso llevado a cabo en el mismo, entendiendo los procesos y la forma con la que se llega a obtener determinados resultados y conclusiones.

Para ampliar conocimiento sobre estas herramientas, en la bibliografía del trabajo se pueden encontrar los documentos de los que ha sido extraída la información, de forma más amplia y en detalle, ya que, si se quisiera llegar a tal nivel de detalle sobre cada materia en este trabajo, además de perderse el verdadero objetivo, la extensión sería demasiado amplia como para abarcarla en este documento.

C. Desarrollo del trabajo

1. Introducción

En este capítulo se va a desarrollar el grueso del trabajo, documentando el desarrollo completo, comenzando desde la parte de análisis y acondicionamiento de datos, continuando con la parte de predicción, y finalizando con la parte correspondiente a detección de errores.

2. Visión global del sistema

Para el trabajo se utilizarán como entrada de datos la base de datos de consumo eléctrico de la región francesa de Borgoña de los años 2014 y 2015. Estos datos proporcionan el consumo eléctrico en intervalos de media hora de la región durante todo el año.

Con estos datos, se procederá a aplicar el algoritmo SSA, obteniéndose los autovectores y autovalores, y componentes de mayor importancia del consumo.

Con este algoritmo, como se explicó en el capítulo anterior, se recompondrá una señal de consumo eléctrico de la que se eliminarán las componentes de mayor frecuencia y menor potencia (ruido) para tener una señal limpia y más sencilla de analizar, a través de la cual, realizar una predicción de consumo, por distintos métodos que permite el paquete RSSA, y se compararán con el consumo real, de forma que se pueda valorar que método es el más apropiado para este fin.

Por último, se introducirán, de forma controlada, errores sistemáticos conocidos a la señal real, para, poder detectar estos errores al diferenciarse la señal predicha de la analizada.

Con todo esto se podrán formular unas conclusiones y unas pautas a seguir, que permitan profundizar más en el tema en estudios posteriores.

3. Preparación de los datos

En este apartado, como se ha comentado anteriormente, se preparará la base de datos de consumo eléctrico de la región francesa de Borgoña de los años 2014 y 2015, ya que son los datos más recientes que se pueden obtener. Los datos vienen en un formato que muestra el consumo eléctrico en dicha región durante el año entero, en intervalos de 30 minutos, además del desglose de a qué recurso energético pertenece dicho consumo. Esto da 48 muestras de consumo diario, que son 336 muestras de consumo semanales, 1344 muestras mensuales, 17520 muestras anuales y 35040 muestras en los dos años que se van a analizar. Concretamente en este trabajo, se analizarán las muestras de dos años, menos una semana, la cual se utilizará para comprobar la predicción con el resultado real.

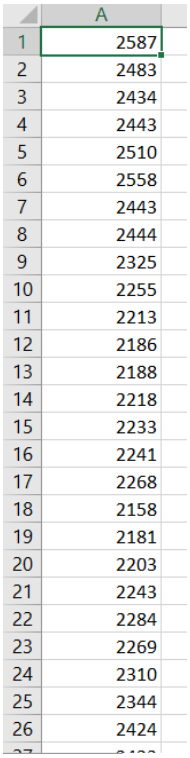
La base de datos obtenida tiene la siguiente forma (Figura 3):

Périmètre	Nature	Date	Heures	Consommation	Thermique	Nucléaire	Eolien	Solaire	Hydraulique	Pompage	Bioénergies	Ech. physiques
Bourgogne-Franche-Comté	Données définitives	01/01/2015	0:00	2772	97	-	84	0	201	0	23	2368
Bourgogne-Franche-Comté	Données définitives	01/01/2015	0:15									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	0:30	2892	104	-	91	0	201	0	22	2474
Bourgogne-Franche-Comté	Données définitives	01/01/2015	0:45									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	1:00	2812	104	-	88	0	53	0	22	2545
Bourgogne-Franche-Comté	Données définitives	01/01/2015	1:15									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	1:30	2811	104	-	78	0	53	0	22	2555
Bourgogne-Franche-Comté	Données définitives	01/01/2015	1:45									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	2:00	2906	104	-	80	0	53	0	21	2648
Bourgogne-Franche-Comté	Données définitives	01/01/2015	2:15									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	2:30	2955	106	-	81	0	53	0	22	2693
Bourgogne-Franche-Comté	Données définitives	01/01/2015	2:45									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	3:00	2826	104	-	85	0	53	0	21	2562
Bourgogne-Franche-Comté	Données définitives	01/01/2015	3:15									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	3:30	2924	109	-	88	0	53	0	22	2652
Bourgogne-Franche-Comté	Données définitives	01/01/2015	3:45									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	4:00	2788	104	-	80	0	53	0	21	2531
Bourgogne-Franche-Comté	Données définitives	01/01/2015	4:15									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	4:30	2756	103	-	66	0	53	0	21	2514
Bourgogne-Franche-Comté	Données définitives	01/01/2015	4:45									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	5:00	2696	102	-	69	0	53	0	22	2451
Bourgogne-Franche-Comté	Données définitives	01/01/2015	5:15									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	5:30	2686	102	-	47	0	53	0	21	2463
Bourgogne-Franche-Comté	Données définitives	01/01/2015	5:45									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	6:00	2697	104	-	40	0	52	0	21	2479
Bourgogne-Franche-Comté	Données définitives	01/01/2015	6:15									
Bourgogne-Franche-Comté	Données définitives	01/01/2015	6:30	2716	104	-	33	0	57	0	22	2499

Figura 3: Base de datos de consumo de Borgoña

Como se puede ver, la base de datos no se encuentra de forma que sea sencillo su manejo. Para adaptar mejor la base de datos, se va a utilizar un *script* de *Matlab*, que permita obtener las muestras de consumo de los dos años y unirlas en un solo archivo de extensión *.csv

Con este *script*, se obtendrá la columna de consumo (figura 4), eliminando los valores vacíos, de forma que se obtienen los valores de la siguiente forma:



	A
1	2587
2	2483
3	2434
4	2443
5	2510
6	2558
7	2443
8	2444
9	2325
10	2255
11	2213
12	2186
13	2188
14	2218
15	2233
16	2241
17	2268
18	2158
19	2181
20	2203
21	2243
22	2284
23	2269
24	2310
25	2344
26	2424

Figura 4: Datos de consumo adecuados para tratamiento SSA

El *script* utilizado para preparar los datos para su análisis en SSA, se encuentra en el apartado VIII - Código.

A continuación, se muestra una gráfica del consumo eléctrico a lo largo de los dos años (Figura 5)

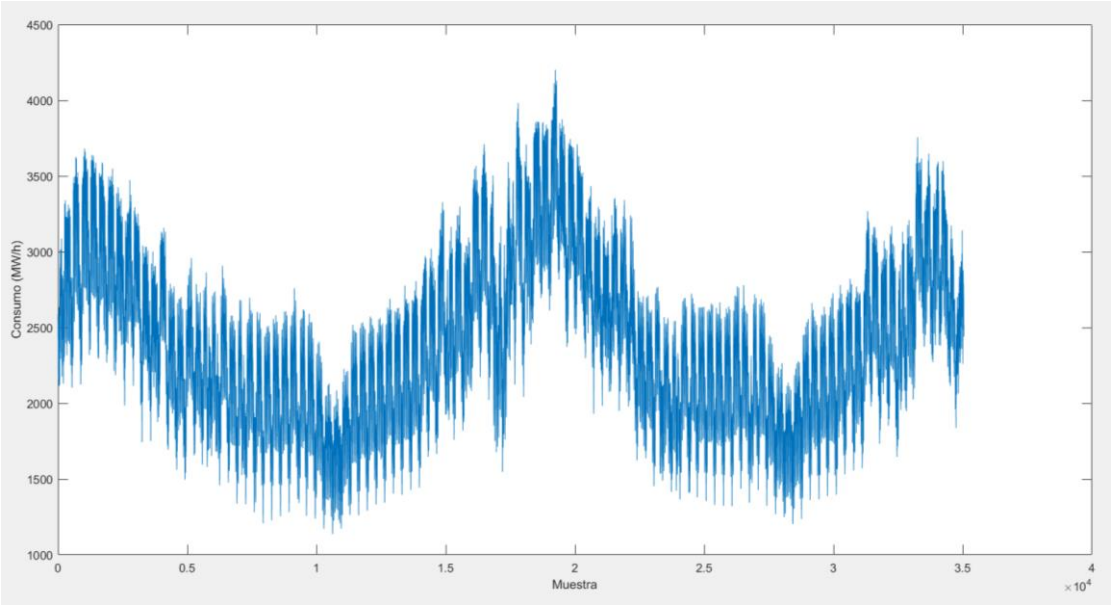


Figura 5: Gráfica de consumo eléctrico en 2 años

4. Aplicación de SSA

Para utilizar SSA se dispondrá un paquete de funciones del software libre R, llamada RSSA, realizado por Anton Korobeynikov, Alex Shlemov, Konstantin Usevich y Nina Golyandina en el año 2016, que aporta funciones de descomposición, reconstrucción, predicción, etc. Gracias a este paquete, se ahorrará el realizar toda la matemática que viene detrás del algoritmo SSA, utilizando funciones ya diseñadas.

Una vez se han exportado los datos al software *RStudio*, el primer paso a seguir es el de seleccionar un tamaño de ventana apropiado, donde el máximo sería la mitad de la serie. En este caso se va a utilizar un tamaño de ventana de 336 muestras, correspondiente a una semana.

En primer lugar, tras aplicar el algoritmo SSA, se puede observar la información interesante que este aporta. Por ejemplo, la matriz de correlación (Figura 6), donde se observan las componentes, ordenadas de mayor a menor potencia, y la correlación entre ellas, quedando de la siguiente forma:

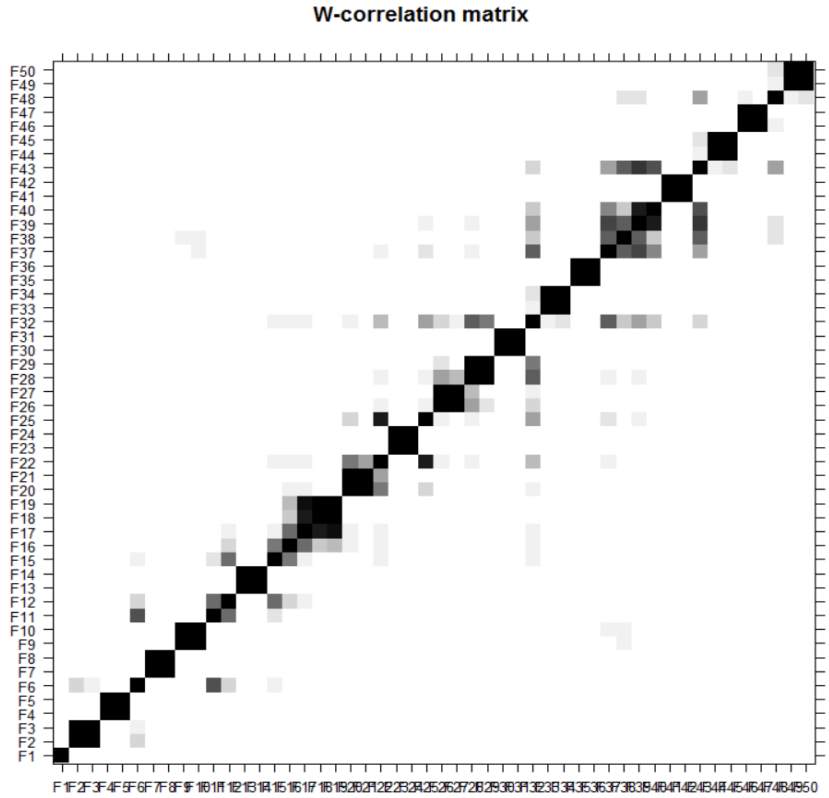


Figura 6: Gráfica de correlación de componentes frecuenciales tras SSA

También se pueden ver las distintas componentes, con su potencia asociada (Figura 7):

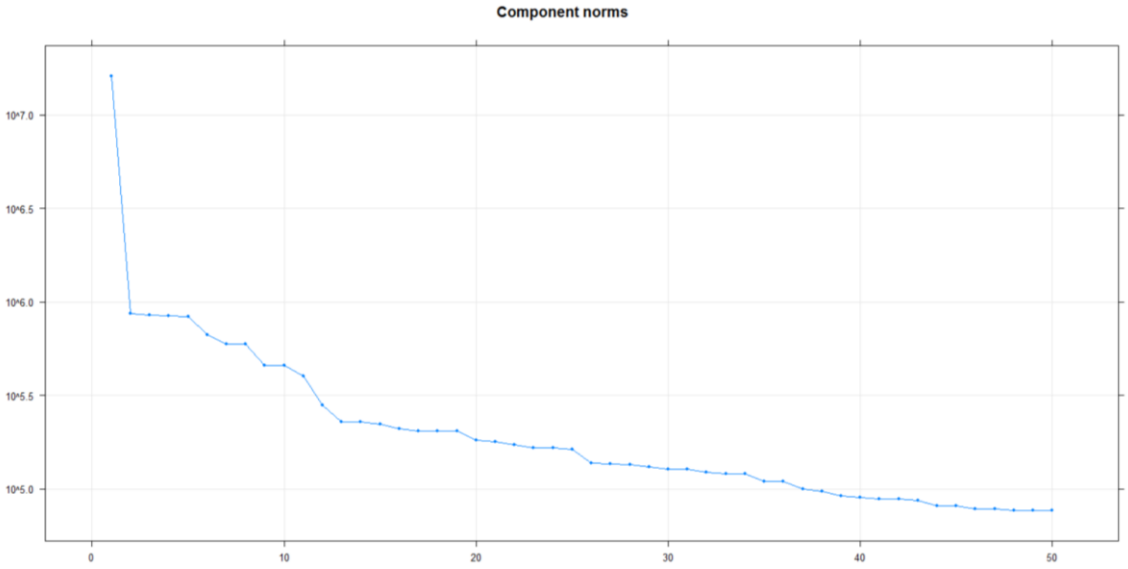


Figura 7: Gráfica de potencia de componentes frecuenciales tras SSA

Los distintos autovectores (Figura 8) con su potencia y frecuencia asociada:

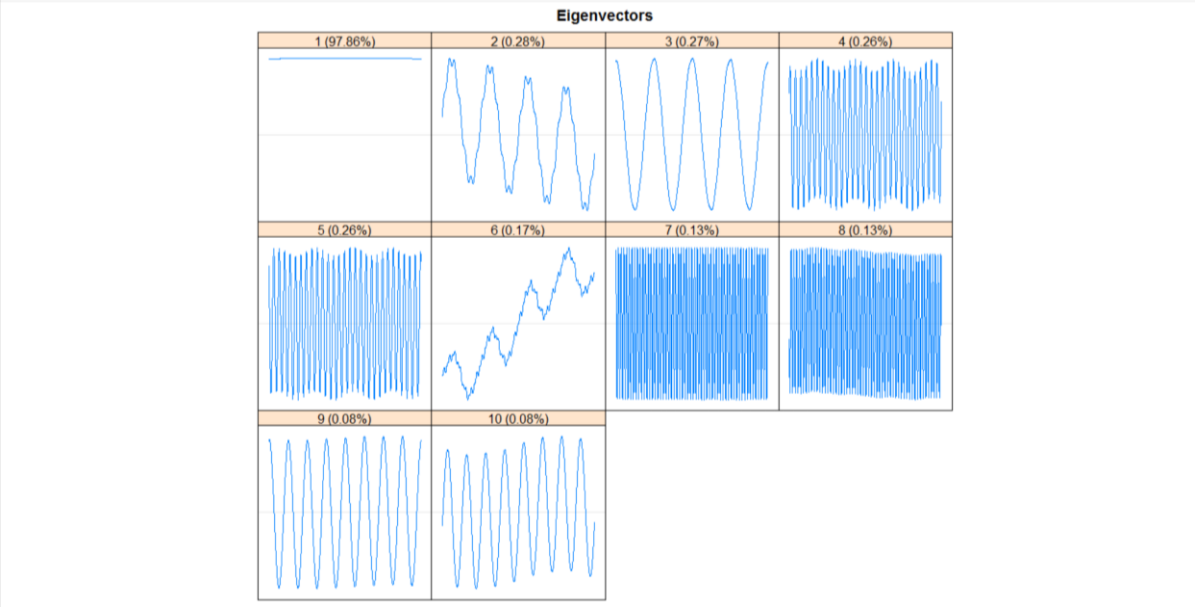


Figura 8: Gráfica de autovectores SSA

Los pares de autovectores (Figura 9), y su relación entre sí:

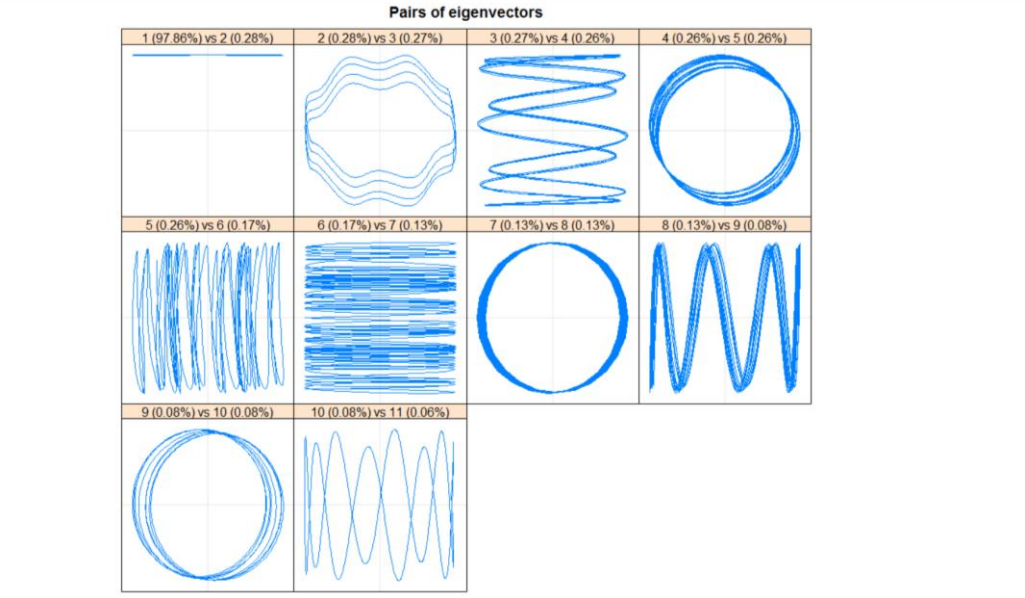


Figura 9: Relación de autovectores SSA

En esta gráfica se muestra la covarianza y la influencia entre sí (dependencia) entre los distintos autovectores, donde, a mayor forma circular, mayor es la relación entre ellos, lo que permite agruparlos.

Y, por último, las primeras 50 señales senoidales que componen la serie temporal (Figura 10):

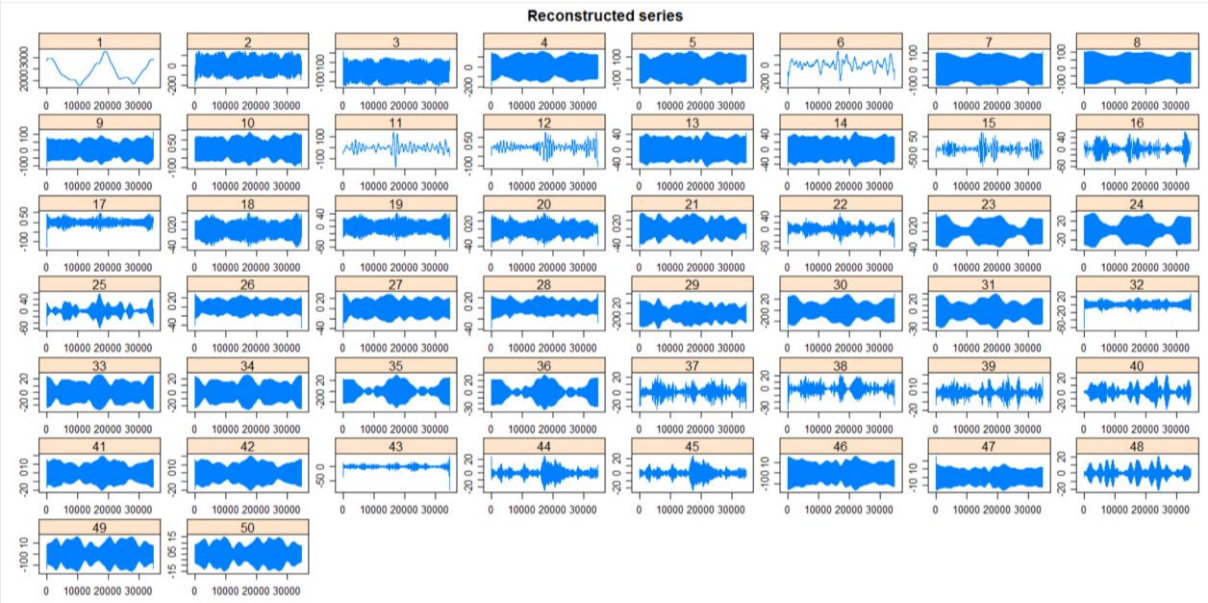


Figura 10: Primeras 50 series reconstruidas por SSA

Tras esto, a elección del usuario, se tiene la posibilidad de reconstruir la señal utilizando las componentes de mayor interés para cada caso. Para ello el paquete RSSA proporciona dos funciones clave, *reconstruct* y *residuals*.

La primera de ellas permite reconstruir una serie temporal, utilizando las componentes que más interesen para cada caso. La función tiene la siguiente forma:

reconstruct(x, groups, ..., drop.attributes = FALSE, cache = TRUE)

Donde:

- X = señal SSA de la cual obtener las componentes
- Groups = lista de componentes a seleccionar para la reconstrucción
- drop.attributes = valor lógico (TRUE o FALSE) si es verdadero, los atributos de la señal de entrada no se copiaran a la reconstruida

- cache = valor lógico (TRUE o FALSE). Si es verdadero los resultados automáticamente se procesarán de nuevo mediante SSA.
- La señal devuelve (con cache a FALSE) la serie temporal reconstruida.

La segunda función, *residuals*, permite obtener "lo que queda" tras una reconstrucción, es decir, si se realiza una reconstrucción, obteniendo la parte correspondiente a la tendencia, si se utiliza *residuals* a continuación, se obtendrá la serie temporal correspondiente a la señal sin su tendencia. La función tiene la siguiente forma:

residuals(object, groups, ..., cache = TRUE)

Donde:

- Object = respuesta obtenida de la función *reconstruct*
- Groups = componentes a utilizar en la señal, del resto de *reconstrcut*, si no se indica, utiliza todas las componentes restantes
- Cache = valor lógico (TRUE o FALSE) si es verdadero los resultados automáticamente se procesarán de nuevo mediante SSA
- La señal devuelve (con cache a FALSE) la serie temporal residual.

En este caso, tras analizar varias posibilidades de número de muestras a utilizar para la predicción, finalmente se llegó a la conclusión de realizar el *forecasting* (predicción) sobre la señal quitando el menor número posible, muy poco significativas, para no perder contenido de la señal y realizar una predicción lo más acertada posible. Por ello, se van a utilizar las 50 primeras, que es con las que se queda el algoritmo SSA de este paquete.

Tras esto viene el proceso de predicción, para ello, el paquete de R, RSSA, proporciona diferentes métodos de predicción, cada uno con distintas posibilidades de edición, para aumentar o disminuir la calidad de la predicción. Estos cambios también aumentan o reducen el tiempo necesario para llevar a cabo la predicción, que también dependerá de las especificaciones del ordenador con el que se realicen dichas predicciones.

Como se ha comentado, el paquete de SSA proporciona distintos métodos para llevar a cabo la predicción. Las series temporales se suelen componer de distintos patrones. Estos

patrones se suelen poder explicar mediante relaciones de causa y efecto. Los distintos patrones o componentes más comunes que se suelen observar son, por ejemplo, la tendencia, el efecto estacional, cambios cíclicos y cambios aleatorios. La identificación de estos patrones es muy importante a la hora de decidir qué modelo de predicción utilizar. Dichos métodos son los que se encuentran a continuación, donde se explicará tanto los parámetros del método en sí del paquete, como una explicación a rasgos generales de cada uno de ellos:

- *Forecast*: es el método genérico de predicción. Con él se pueden realizar los distintos métodos que el paquete dispone, pasándole como un parámetro el método a utilizar. A continuación, se pueden ver los parámetros del método:

  forecast(object, groups, len = 1, method = c("recurrent", "vector", "bootstrap-recurrent", "bootstrap-vector"), ..., drop = TRUE, drop.attributes = FALSE, cache = TRUE)

  - Object = objeto SSA del que se quiere realizar la predicción
  - Groups = grupo de *eigentriples* ($\sqrt{Autovalor}$, Autovector, vector factorial) para realizar la predicción
  - Len = Nº de muestras a predecir a continuación de la última muestra de la serie temporal
  - Method = método de predicción que se utiliza, se habla de ellos más adelante
  - Drop = valor lógico, si es verdadero, la salida se fuerza a la propia serie
  - drop.attributes = valor lógico, si es verdadero, las rutinas de pronostico no interfieren con el tiempo
  - cache = valor lógico (TRUE o FALSE) si es verdadero los resultados automáticamente se procesarán de nuevo mediante SSA

- *bforecast*: realiza la predicción utilizando el método *bootstrap*, con sus dos variantes, *bootstrap-recurrent* y *bootstrap-vector*. El método *Bootstrap* [7] es un método de computación intensivo propuesto por el estadístico estadunidense Bradley Efron, que vino a remplazar los antiguos métodos de predicción mediante herramientas algebraicas. Este método se utiliza para aproximar la distribución de muestreo de un estadístico. El método original no es muy eficiente para la predicción de series temporales, ya que son estructuras de datos dependientes entre sí. Por ello, a lo largo de los años se han desarrollado variantes del método, que permiten estimar como será una serie temporal mediante este método de forma fiable. Además, a día de hoy, debido a la potencia de los ordenadores y de las herramientas de cálculo de las que se dispone, es más sencillo y menos costoso en tiempo realizar predicciones mediante este método de gran fiabilidad.

  El método de RSSA tiene la siguiente forma:

  bforecast(x, groups, len = 1, R = 100, level = 0.95, type = c("recurrent", "vector"), ..., drop = TRUE, drop.attributes = FALSE, cache = TRUE)

  - x = Objeto SSA sobre el cual se va a realizar la predicción
  - Groups = grupo de *eigentriples* para realizar la predicción
  - Len = Nº de muestras a predecir a continuación de la última muestra de la serie temporal
  - R = Nº de replicaciones del método bootstrap
  - Level = niveles de confianza para los limites
  - Type = tipo de método a realizar (recurrente o vectorial)
  - Drop = valor lógico, si es verdadero, la salida se fuerza a la propia serie
  - drop.attributes = valor lógico, si es verdadero, las rutinas de pronostico no interfieren con el tiempo
  - cache = valor lógico (TRUE o FALSE) si es verdadero los resultados automáticamente se procesarán de nuevo mediante SSA

- *rforecast*: utiliza el método de predicción recurrente. Este método es similar a *boostrap*, utilizando recurrencia para predicción de datos. El algoritmo proyecta secuencialmente los vectores de inclusión incompletos (ya sea originales o de series reconstruidas) sobre el subespacio abarcado por los *eigentriples* seleccionados de la descomposición para derivar los valores perdidos (finales) de dichos vectores. De esta forma, los elementos previstos de la serie se producen uno a uno.

  rforecast(x, groups, len = 1, base = c("reconstructed", "original"), only.new = TRUE, reverse = FALSE, ..., drop = TRUE, drop.attributes = FALSE, cache = TRUE)

  - x = Objeto SSA sobre el cual se va a realizar la predicción
  - Groups = grupo de *eigentriples* para realizar la predicción
  - Len = Nº de muestras a predecir a continuación de la última muestra de la serie temporal
  - Base = Serie utilizada como una "semilla" de pronóstico: original o reconstruida de acuerdo con los argumentos
  - only.new = valor lógico, si es verdadero solo se devuelven los valores predichos, en lugar de la serie entera.
  - Reverse = valor lógico, indica en el tiempo en el que avanza la predicción, si es falso, avanza en el orden temporal, si no, al contrario
  - Drop = valor lógico, si es verdadero, la salida se fuerza a la propia serie
  - drop.attributes = valor lógico, si es verdadero, las rutinas de pronóstico no interfieren con el tiempo
  - cache = valor lógico (TRUE o FALSE) si es verdadero los resultados automáticamente se procesarán de nuevo mediante SSA

- *vforecast:* el método *vforecast* utiliza la predicción vectorial. Este método difiere del anterior en la forma en que continua la serie de vectores en el subespacio, expandiendo los autovectores elegidos y derivando la salida respecto a ellos.

  vforecast(x, groups, len = 1, direction = c("row", "column"), only.new = TRUE, ..., drop = TRUE, drop.attributes = FALSE)

  - x = Objeto SSA sobre el cual se va a realizar la predicción
  - Groups = grupo de *eigentrilples* para realizar la predicción
  - Len = Nº de muestras a predecir a continuación de la última muestra de la serie temporal
  - Direction = Dirección del pronóstico en caso de SSA multicanal
  - only.new = valor lógico, si es verdadero solo se devuelven los valores predichos, en lugar de la serie entera.
  - Drop = valor lógico, si es verdadero, la salida se fuerza a la propia serie
  - drop.attributes = valor lógico, si es verdadero, las rutinas de pronóstico no interfieren con el tiempo

5. Predicción usando los distintos métodos

En este apartado se va a hablar sobre los resultados obtenidos en las distintas predicciones y con distintos parámetros, intentando llegar a la conclusión de cuál es más efectivo, tanto en tiempo de procesamiento como en exactitud de la salida.

Para ello se va a utilizar un *script* de *Matlab*, con el que se muestran los valores correspondientes a la serie temporal original, hasta una semana antes, la semana que ocurrió realmente y la semana predicha, donde se podrá comparar las dos gráficas, de forma visual.

- *Bootstrap-recurrent*

En este punto se analiza el método *bootstrap-recurrent* introduciéndole distintos parámetros de entrada:

  - L=48*7, groups = list(1:24) y R = 10

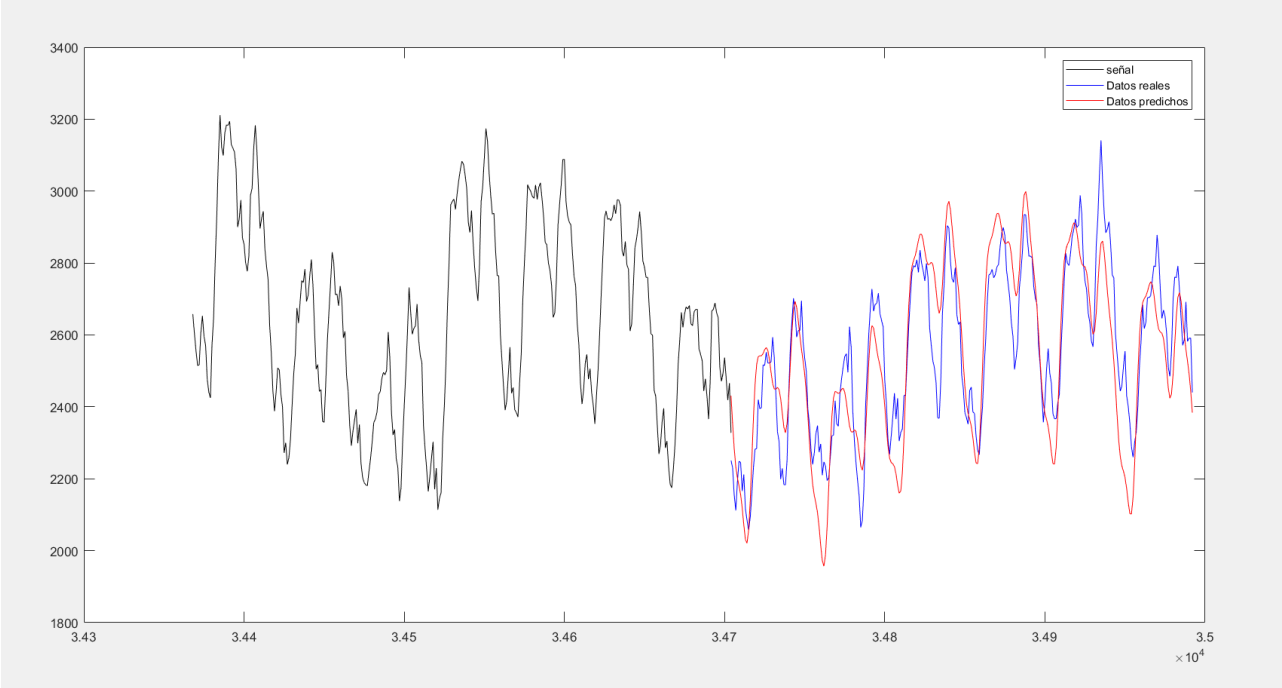


Figura 11: Gráfica de predicción vs real

- L=48*7, groups = list(1:48) y R = 10

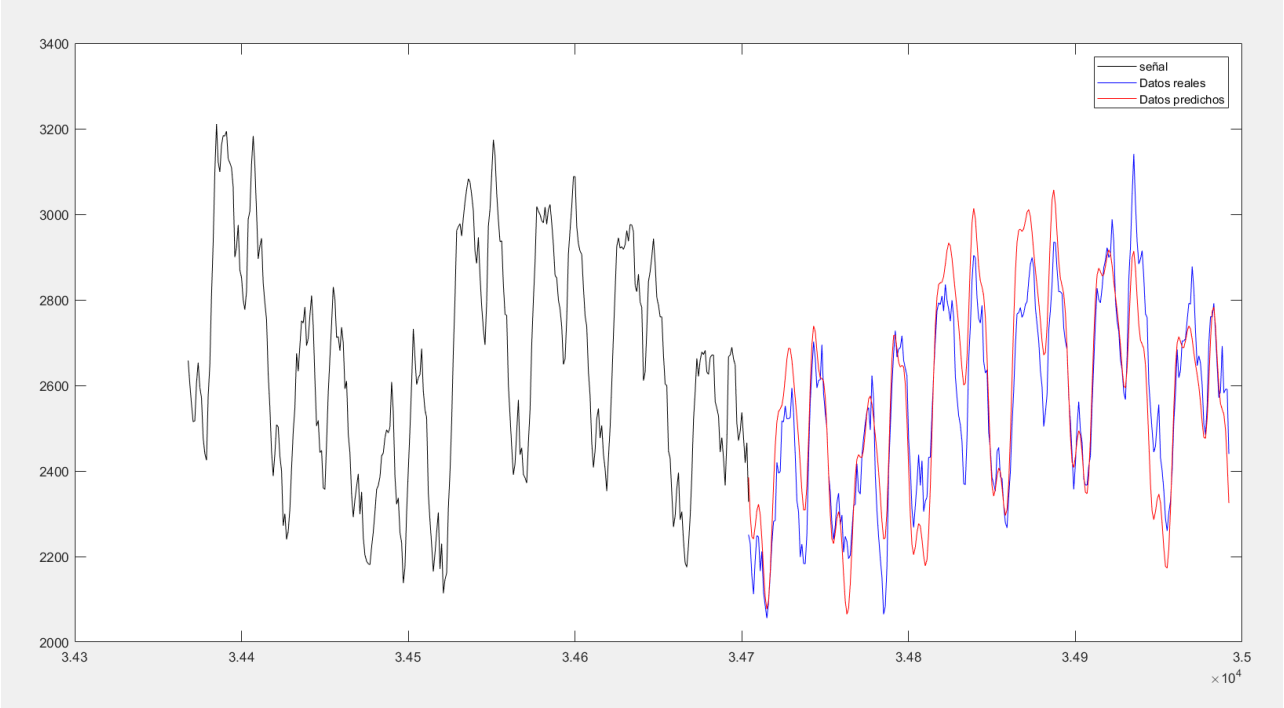


Figura 12: Gráfica de predicción vs real

- L=48*7, groups = list(1:24) y R = 100

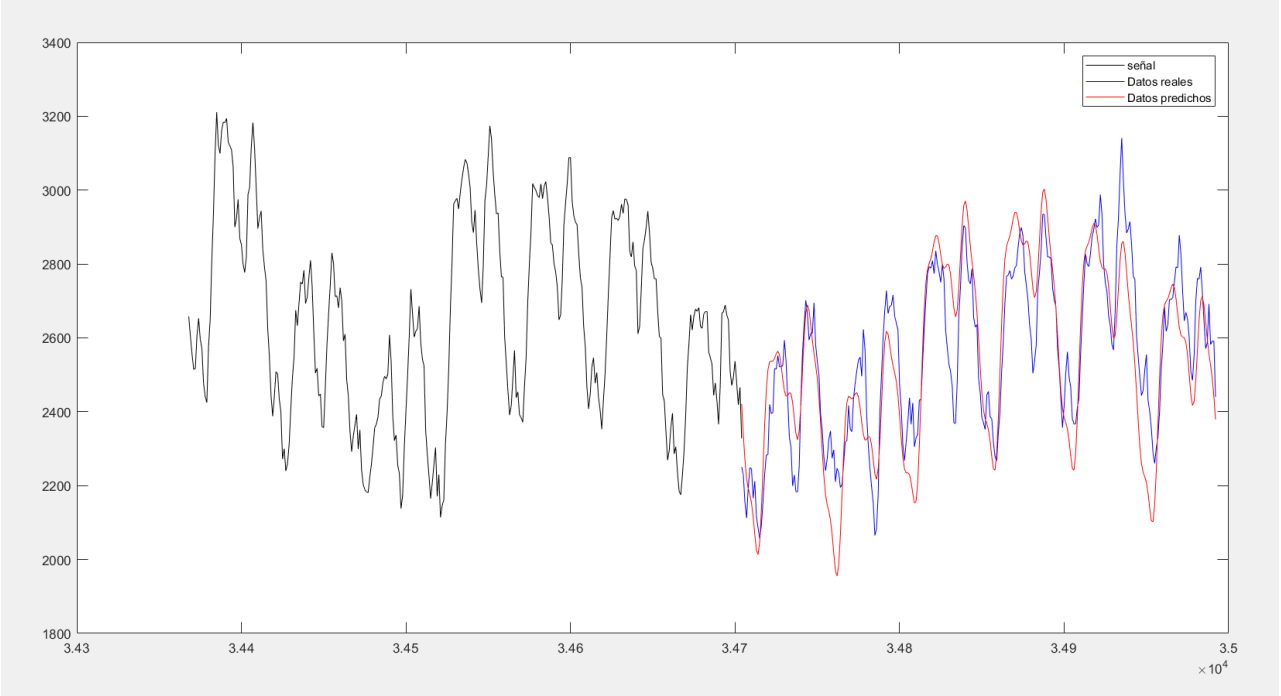


Figura 13: Gráfica de predicción vs real

- L=48, groups = list(1:24) y R = 10

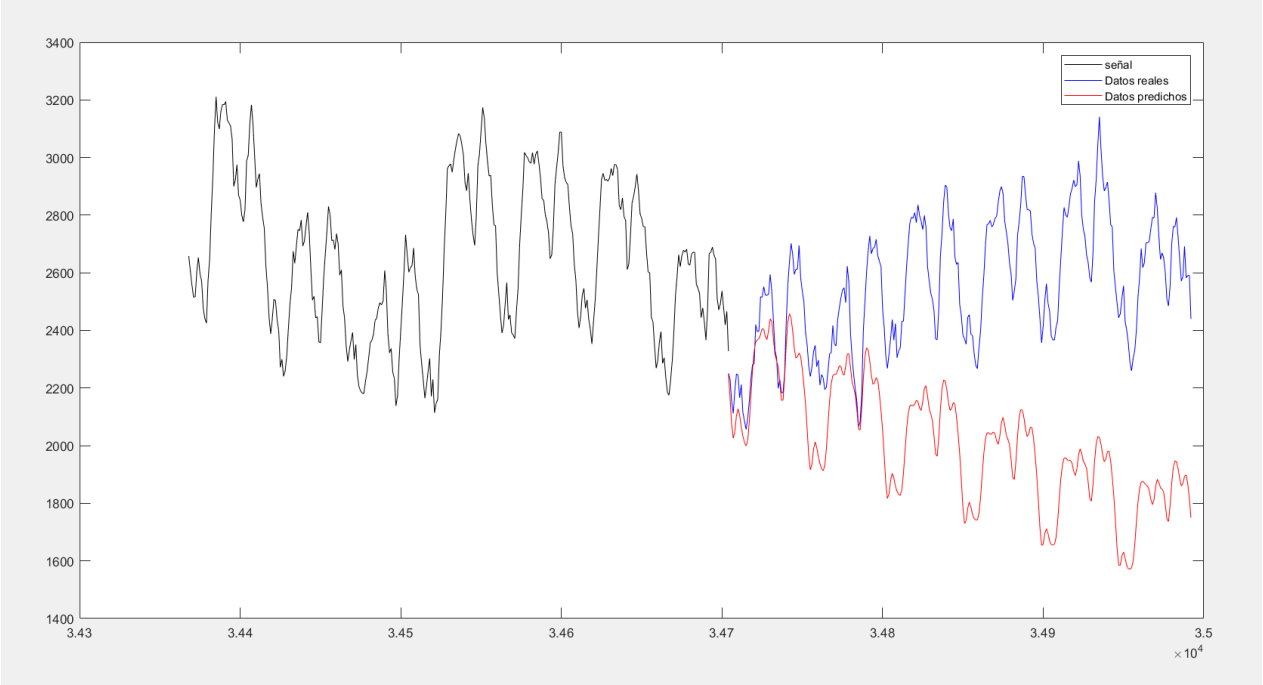


Figura 14: Gráfica de predicción vs real

- L=48*7, groups = list(1:48) y R = 30

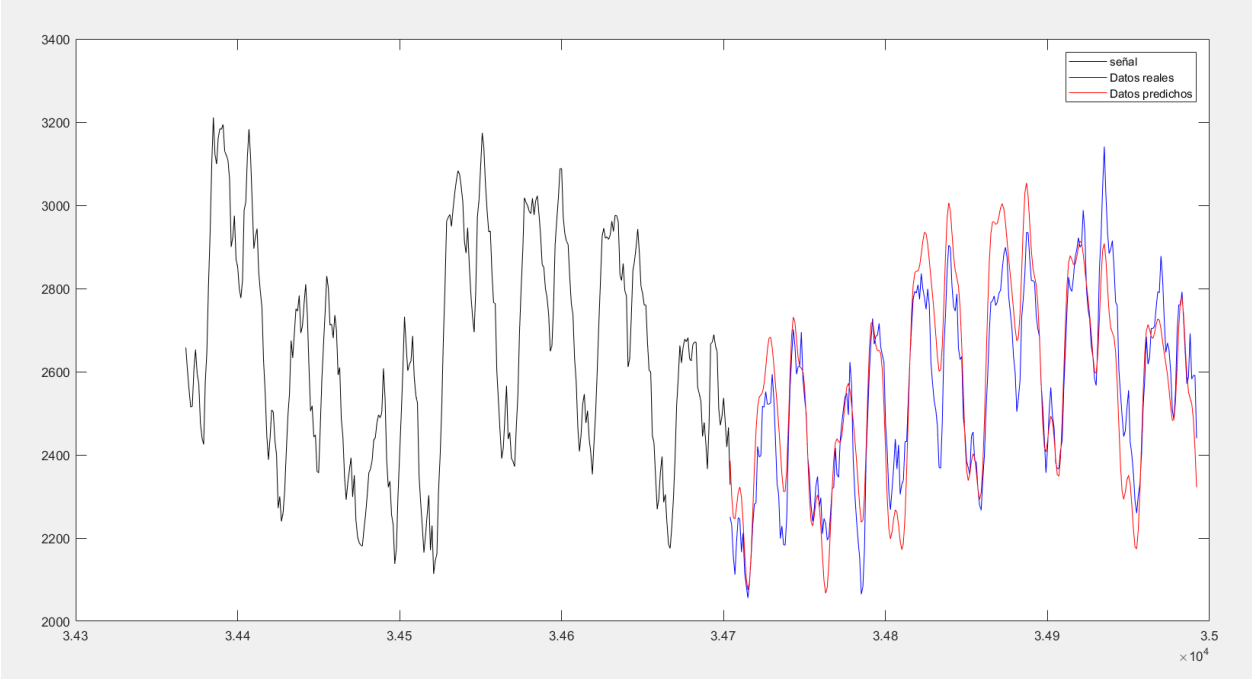


Figura 15: Gráfica de predicción vs real

- L=48*7*4, groups = list(1:48) y R = 30

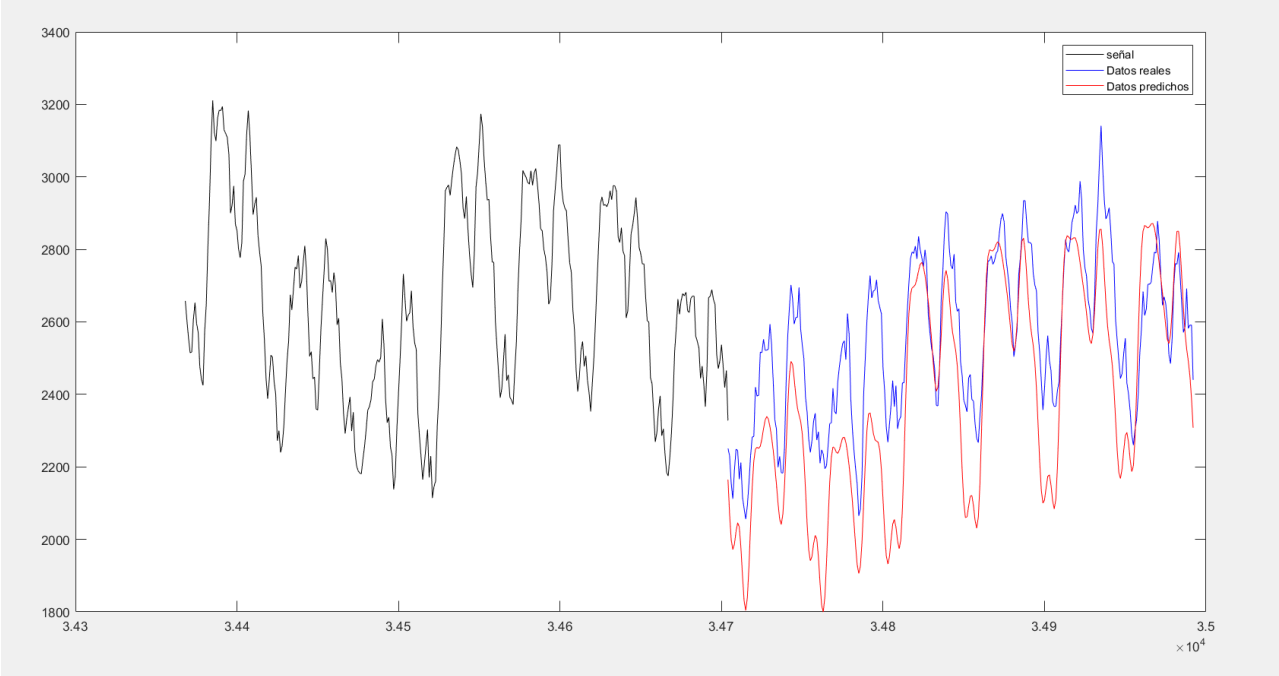


Figura 16: Gráfica de predicción vs real

La primera conclusión que se saca en claro de las predicciones es que el tamaño ideal de la ventana es de las muestras de una semana, 7*48, ya que si se utilizan menos, como en la que se utiliza un día de ventana, se pierde información referente a la tendencia, por lo que la predicción cae más rápido de lo que debería, debido a que solo tiene en cuenta la salida más reciente, sin tener demasiado en cuenta la tendencia global. En cambio, si la ventana se aumenta demasiado, como en el último caso, a un mes, se pierde información de los cambios más concretos, lo que nos lleva a que, a pesar de más o menos, seguir bien la tendencia, los fallos diarios son bastante significativos, por ello se concluye que el tamaño de ventana ideal para este supuesto es de una semana.

Tras esto, también se puede apreciar que a mayor número de muestras utilizadas en la predicción (48 contra 24) más preciso es el resultado. Esto tiene como contrapunto, que cuanto más se eleve el número de muestras, el tiempo de predicción se hace insostenible, pudiendo durar horas o más.

El tercer cambio que se puede apreciar es el número de concurrencias del método *Bootstrap* para el que habría que ir probando, para llegar al número ideal, ya que, si es muy bajo, no realiza bien la predicción, pero si es muy alto tampoco, por lo que hay que llegar a un compromiso. Este parámetro, al igual que el número de componentes, es muy influyente en el tiempo de respuesta, llevándolo a grandes tiempos de espera si se incrementa demasiado.

Estos resultados llevan a concluir que para la predicción *bootstrap-recurrent*, al menos para predecir los datos a una semana vista, los valores ideales para llevarla a cabo serían una ventana de tamaño del número de muestras de una semana, 48 componentes, ya que, al igual que con R, si seguimos aumentando, al tener demasiada información, los resultados son de peor calidad. Por último, el valor ideal de R ronda entre 20-60, siendo 60 ya demasiado pesada en tiempo de ejecución.

- Bootstrap-vector

A continuación, se realiza el mismo proceso llevado a cabo para *bootstrap-recurrent*, para analizar este otro método.

  - L=48*7, groups = list(1:24) y R = 10

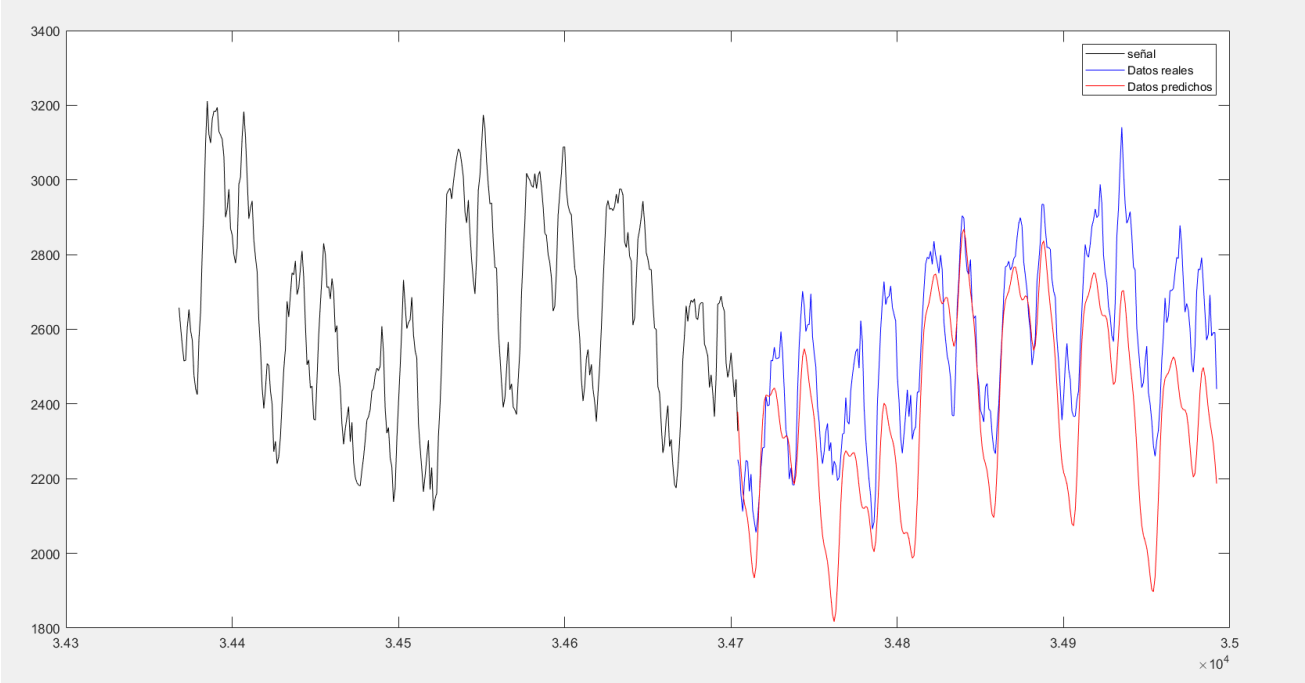


Figura 17: Gráfica de predicción vs rea

- L=48*7, groups = list(1:48) y R = 10

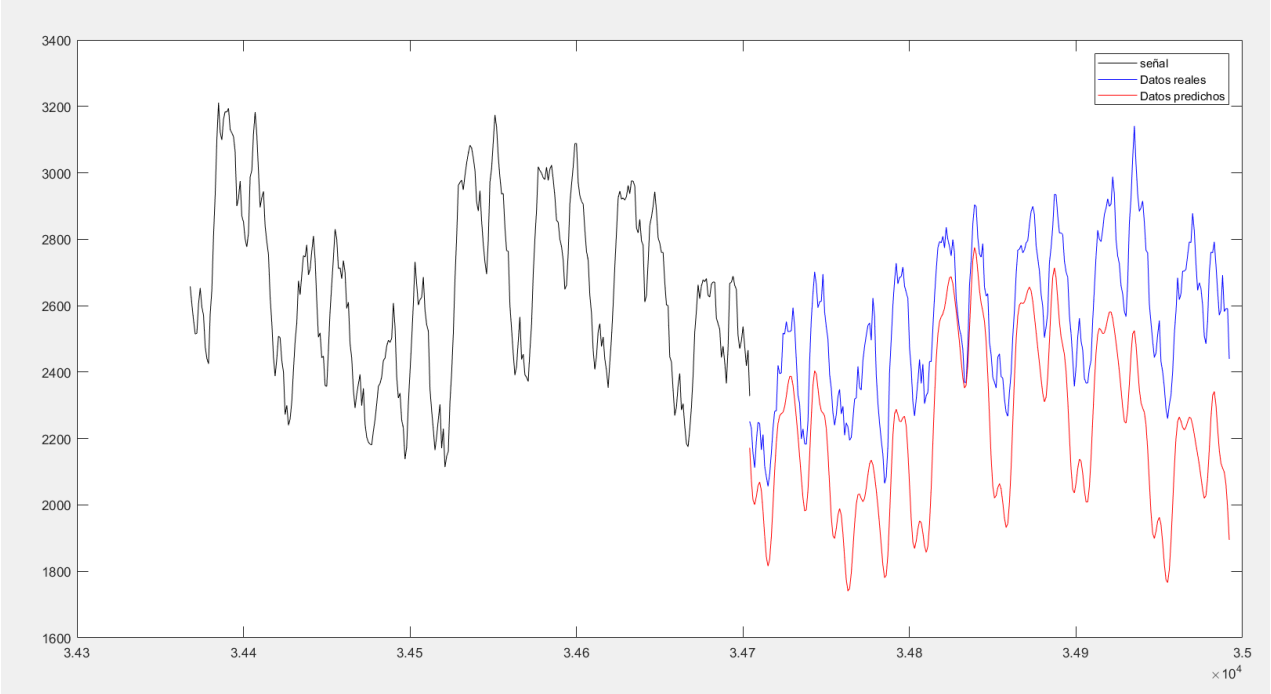


Figura 18: Gráfica de predicción vs real

- L=48*7, groups = list(1:48) y R = 100

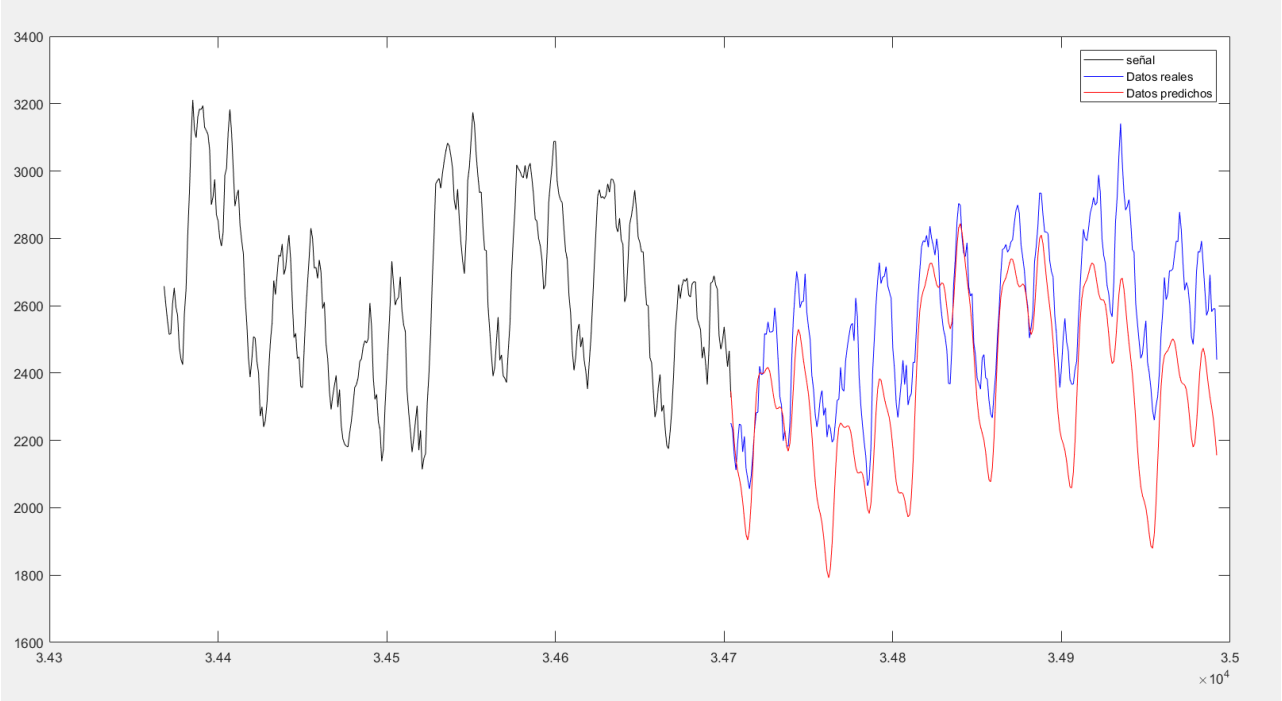


Figura 19: Gráfica de predicción vs real

- L=48*7, groups = list(1:24) y R = 50

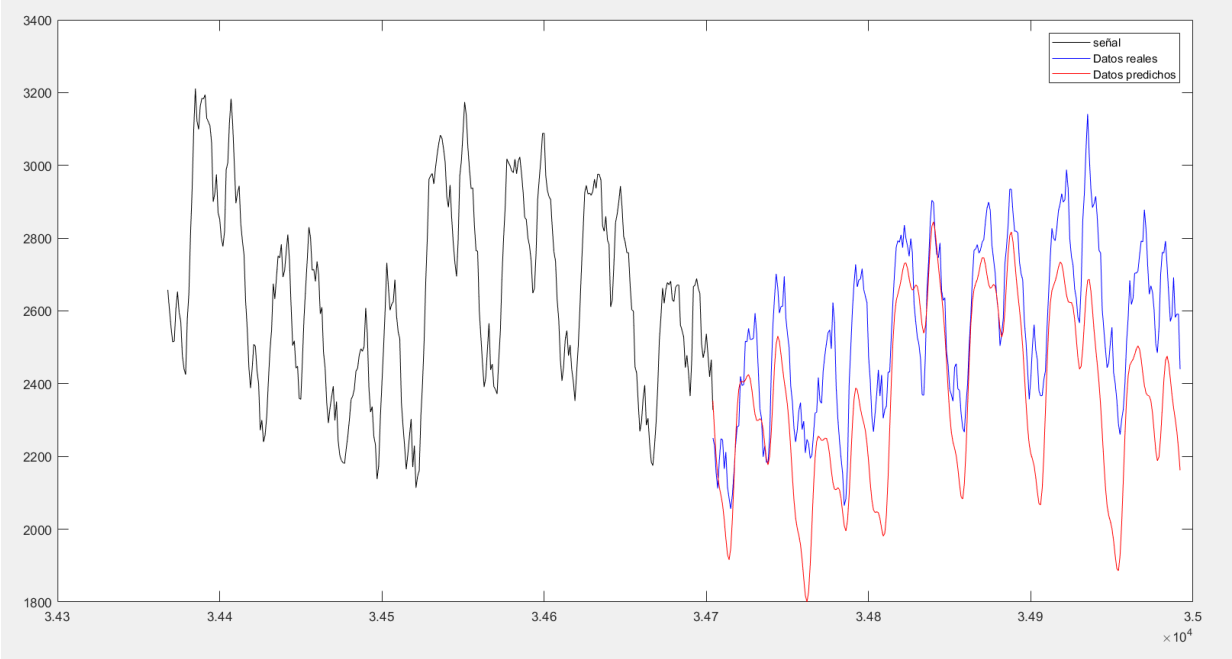


Figura 20: Gráfica de predicción vs real

- L=48*7, groups = list(1:48) y R = 50

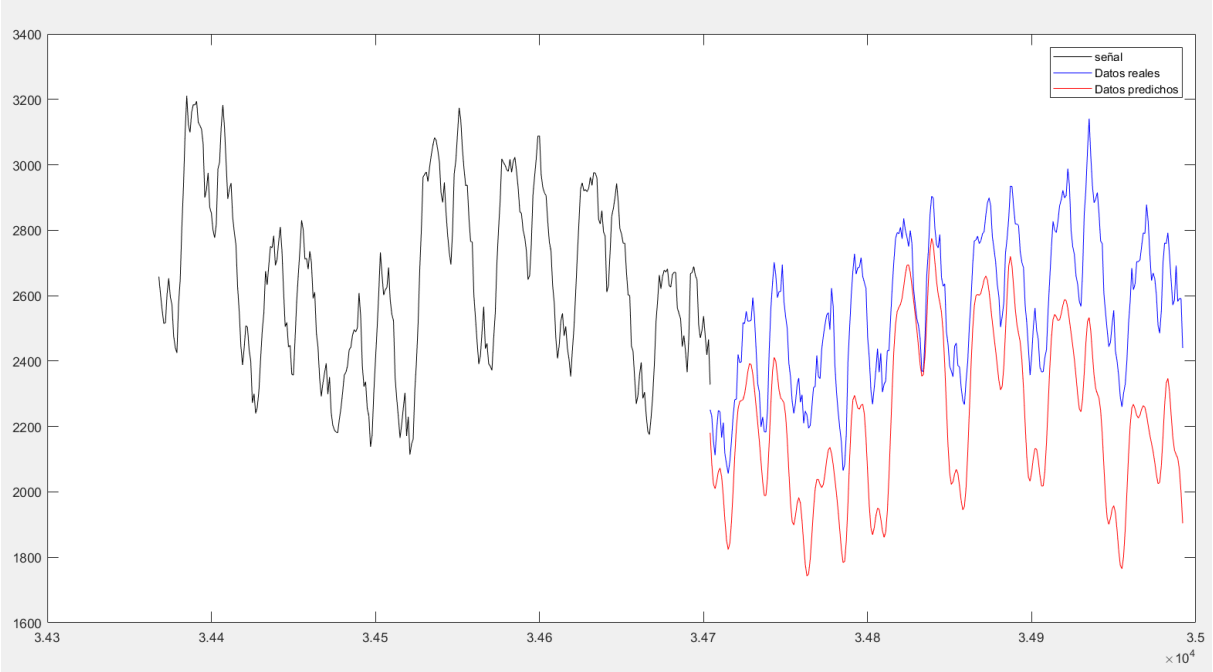


Figura 21: Gráfica de predicción vs real

Lo primero que se puede apreciar de las pruebas realizadas es que este método, a simple vista, es peor que *bootstrap-recurrent*, que analizamos anteriormente, ya que, independientemente de los parámetros probados, los resultados son peores que en el caso anterior. Pero esto puede deberse a esta predicción en concreto, por ello, más adelante, se analizará de forma matemática, si esto que parece apreciarse a simple vista, es así realmente.

Una gran diferencia que se puede observar con respecto al caso anterior es que este funciona mejor con 24 componentes que con 48. Además, este caso, si se comporta mejor a más R, o al menos, las diferencias no son tan apreciables como en el caso *brecurrent*.

Al cambiar el tamaño de ventana, se repiten las conclusiones del apartado anterior, como era de esperar, ya que el tamaño de ventana influye principalmente a la hora de realizar el algoritmo SSA y no tanto en el tipo de predicción a realizar.

- Recurrent

El método *recurrent* será el tercer método que se analice, tanto este método como vector, no usan *bootstrapt*, por lo que no se tendrá el parámetro R con el que cambiar los datos, por lo que no se tienen tantas posibilidades de prueba distintas como en los casos anteriores.

- L=48*7 y groups = list(1:24)

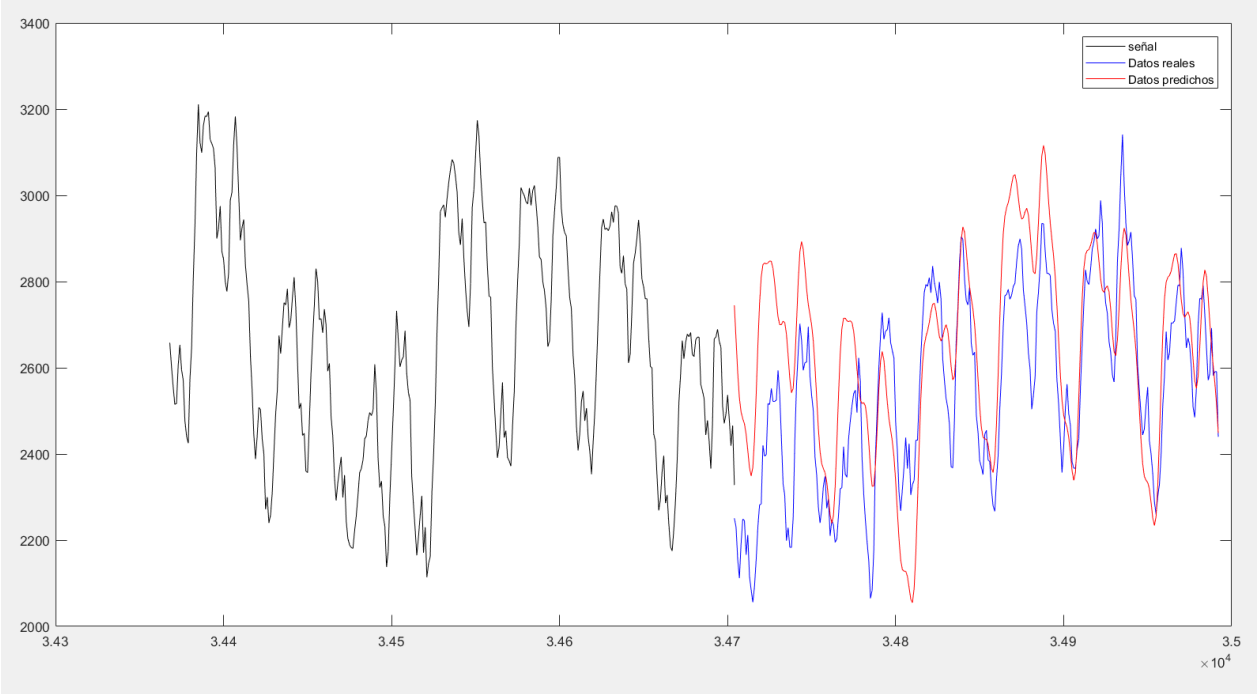


Figura 22: Gráfica de predicción vs real

- L=48*7 y groups = list(1:48)

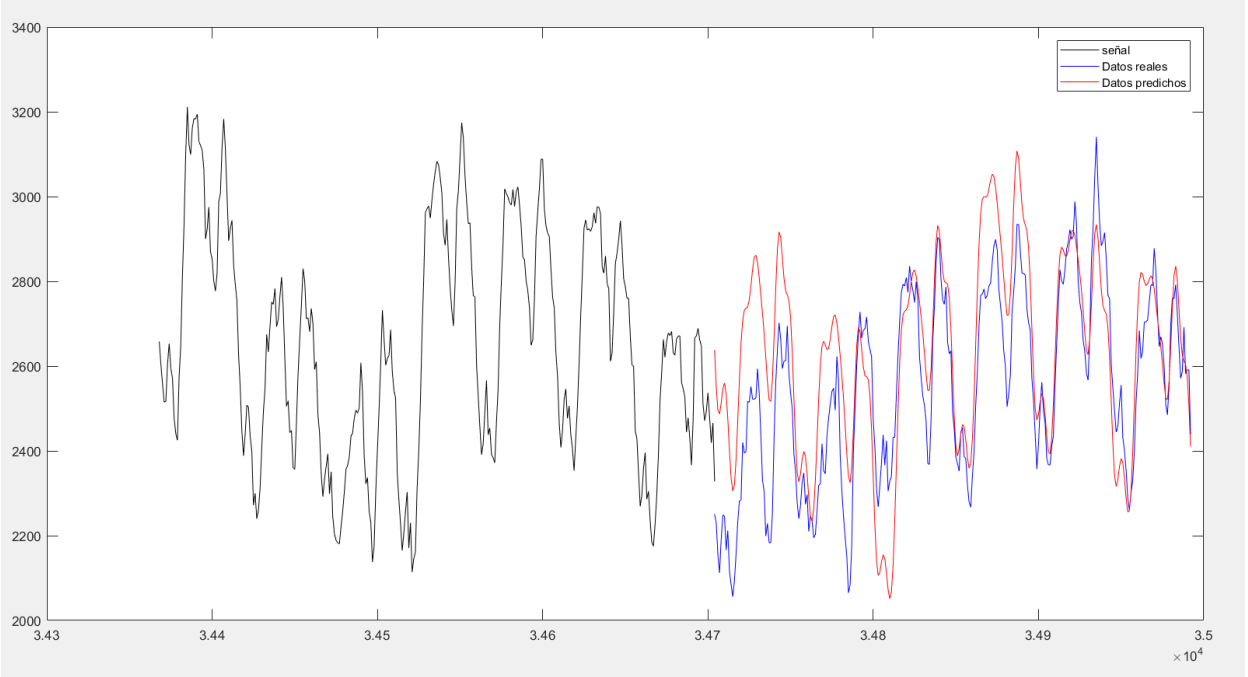


Figura 23: Gráfica de predicción vs real

- L=48*7 y groups = list(1:96)

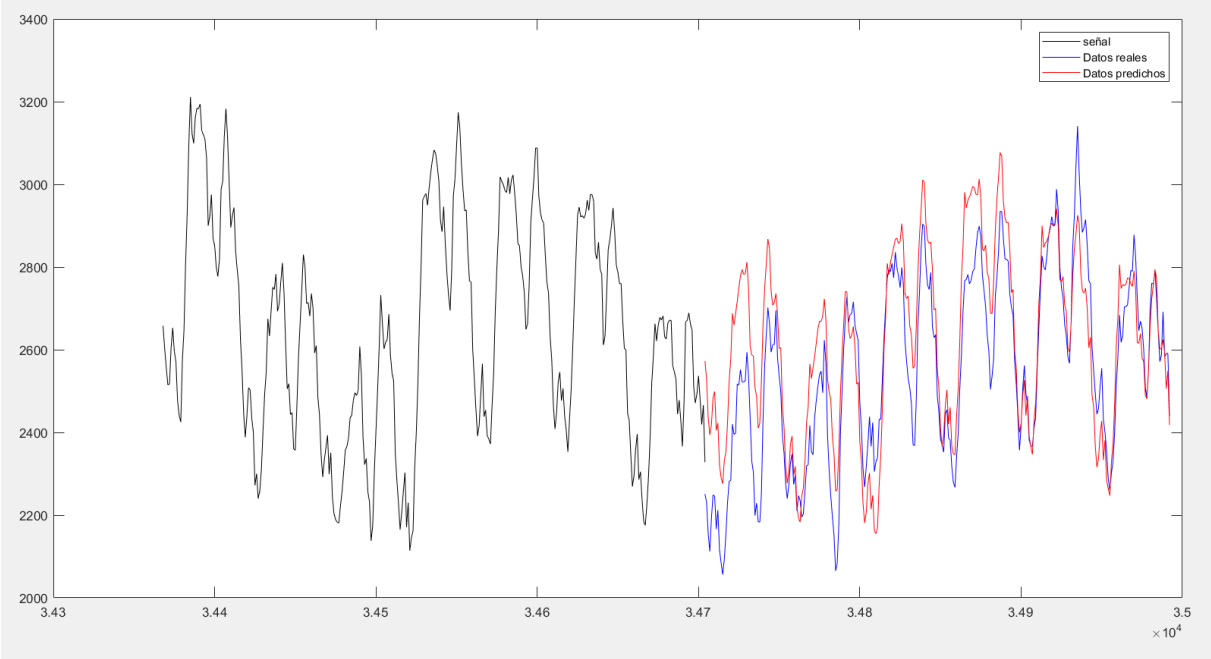


Figura 24: Gráfica de predicción vs real

- L=48*7 y groups = list(1:192)

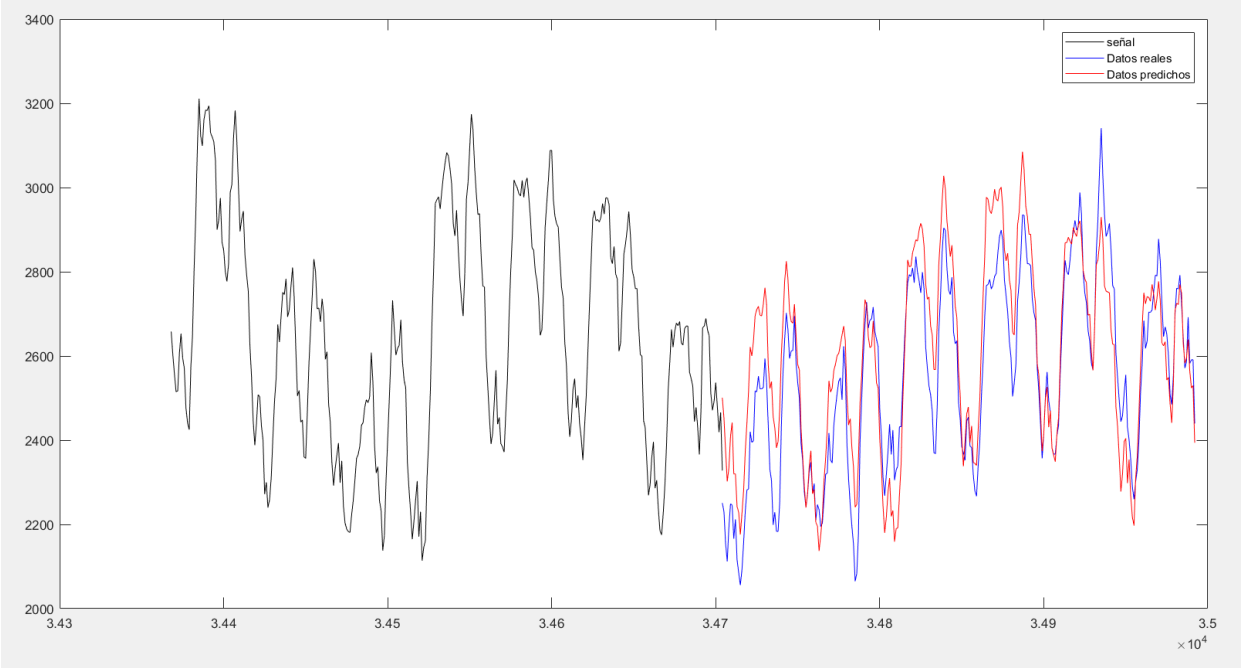


Figura 25: Gráfica de predicción vs real

Como se puede ver, este método es de gran precisión, además de ser muy rápido, a mayor número de componentes, se obtiene un resultado más exacto.

El único problema de esta predicción apreciable es un pequeño transitorio (1 día) en el que la predicción no es tan fiable como en el resto de la semana, pero que, aumentando el número de componentes a analizar en la predicción va desapareciendo poco a poco, aunque aumenta ligeramente el tiempo de ejecución.

- Vector

Último método que analizar, al igual que en caso anterior, no es un método *bootstrap*, por lo que no se dispone de R, por ello, al igual que ocurría en el caso anterior, no tenemos tantas posibilidades para analizar.

  - L=48*7 y groups = list(1:24)

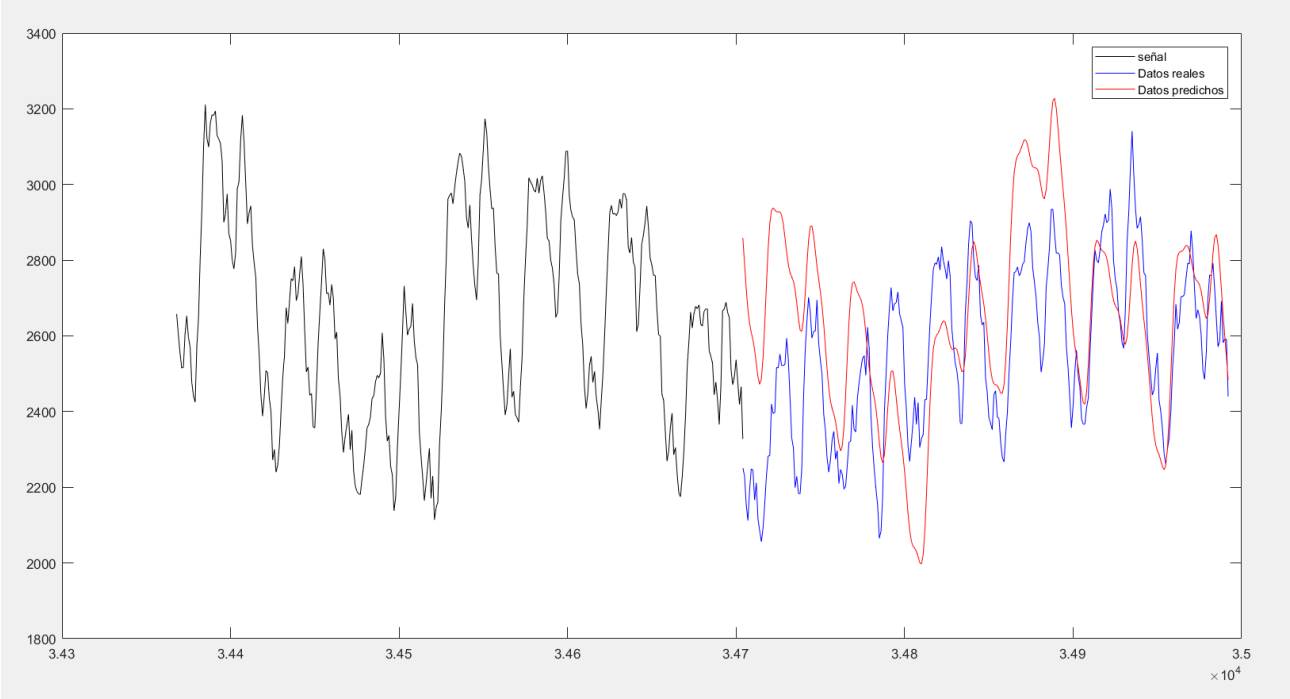


Figura 26: Gráfica de predicción vs real

- L=48*7 y groups = list(1:48)

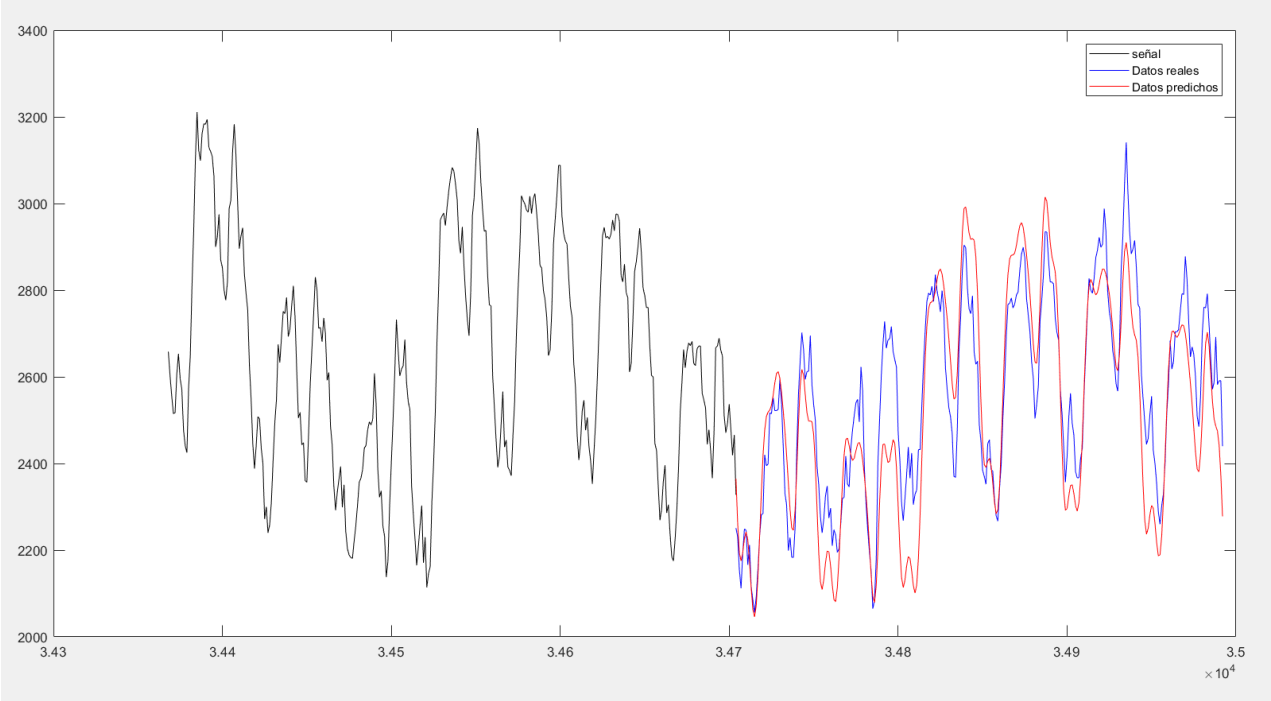


Figura 27: Gráfica de predicción vs real

- L=48*7 y groups = list(1:96)

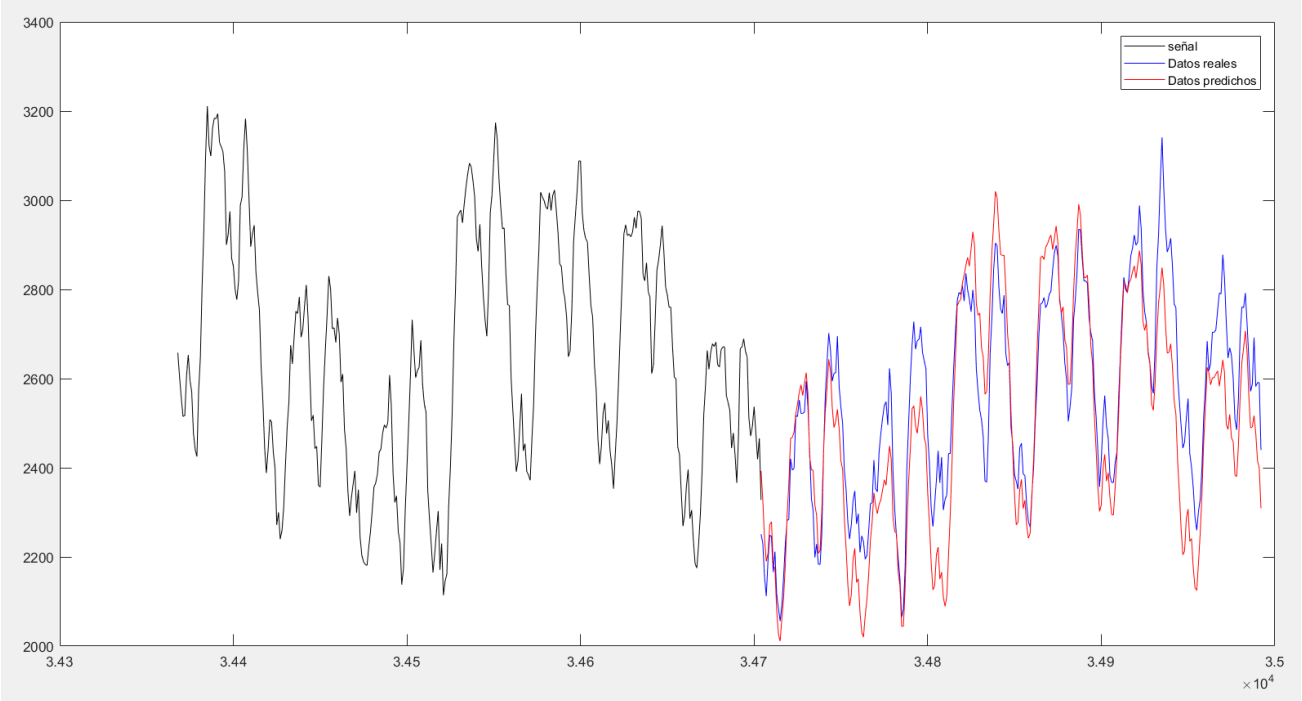


Figura 28: Gráfica de predicción vs real

- L=48*7 y groups = list(1:192)

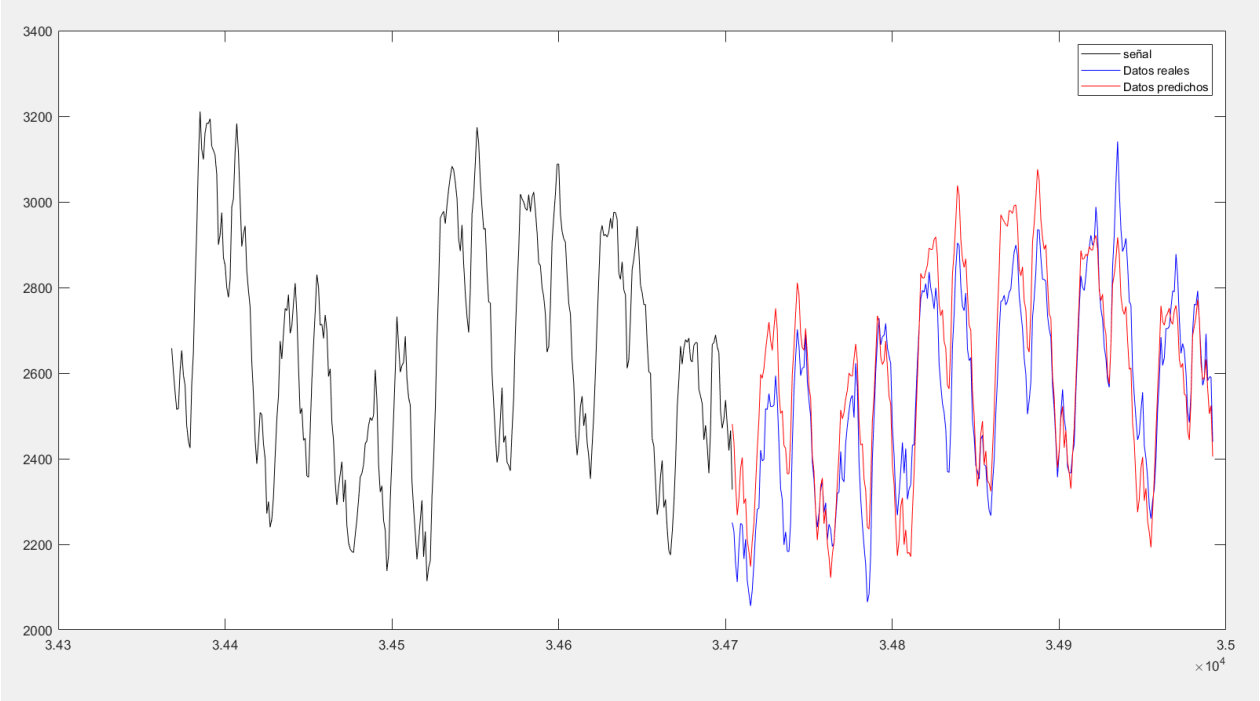


Figura 29: Gráfica de predicción vs real

En este método, al igual que en el anterior, a mayor cantidad de componentes la predicción se vuelve más fiable, pero, por el contrario, el transitorio inicial es mucho menor, lo que lo hace ser un método más fiable, al menos para predicciones a corto plazo (1 semana).

Tras todas estas pruebas, se puede concluir que, al menos para el caso que nos interesa tratar en este trabajo, el método más eficiente y exacto es el método vector, ya que, como podemos observar, es de alta precisión, y, aumentando más las componentes, aunque tarde más, se podrían obtener aun resultados más exactos.

Para corroborar estás observaciones a simple vista, se va a analizar el error que comete en cada método en sus condiciones óptimas. Para ello, con cada método se va a realizar 12 predicciones distintas, una por mes del año, y se calculará el error por distintos métodos existentes como puede ser MAPE.

Una vez calculado el error de cada uno de los meses del año por un mismo método, se realizará una media, en la que se pueda observar el error global que comete el método.

Se repetirá este proceso para todos los métodos y se comparan como se puede ver en la tabla 1. De esta forma, se asegura utilizar el método que comete menor error en distintas predicciones. Las predicciones se harán teniendo en cuenta el histórico de un solo año antes de la fecha a predecir, en lugar de dos, debido a que el histórico de datos disponible es de dos años, por lo que reducirlo a uno permite mover la ventana anual utilizando siempre un histórico de igual longitud. Esto es, si se quiere predecir en diciembre se utilizaría una ventana desde diciembre del año anterior a este, y si fuese abril, lo mismo (desde abril hasta abril) lo cual no se podría conseguir con el histórico de dos años, por falta de datos anteriores a 2014.

MAPE (*Mean Absolute Percentage Error)* es un método para observar la precisión de una predicción. Es un método útil para medir grandes variaciones, aunque, por la fórmula que aplica, no funciona bien si el valor real pasa por 0 debido a la problemática de dividir entre 0. La fórmula la podemos ver a continuación:

$$MAPE = \frac{1}{n} \cdot \sum_{i=1}^{n} \left| \frac{E_i - A_i}{E_i} \right| \times 100 \ \ (\%) \tag{5}$$

También se valorará otro método para tener dos distintas medidas, y así poder elegir el método de predicción más optimo con una mayor cantidad de información. Este método es el NMBE (*Normalized Mean Bias Error*) cuya formula es la que se muestra a continuación. El problema de este método es que proporciona errores positivos y negativos, por lo que al hacer la media de los errores a lo largo de los 12 meses, errores positivos y negativos se anulan, por lo que, se utilizarán los valores en valor absoluto, teniendo en cuenta, que pueden ser tanto positivos como negativos.

$$NMBE = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(\frac{E_i - A_i}{E_i} \right) \times 100 \ \ (\%) \tag{6}$$

Método	*Booststrap-recurrent*	*Booststrap-vector*	*Recurrent*	VECTOR
MAPE	5,71%	7,41%	5,79%	6,60%
NMBE	4,30%	5,38%	4,46%	5,72%

Tabla 1: resultados MAPE y NMBE

Como se puede ver, ambos métodos de *Recurrent* son más precisos que los métodos de vector. Por ello, para la detección de errores, utilizaremos el método *Booststrap-recurrent*.

Además de los métodos utilizados para la medición del error, existen muchos más métodos posibles, pero, en este caso, se utilizarán éstos como criterio.

6. Detección de errores

Una vez que la predicción se ha llevado a cabo y se tienen resultados sobre los que estimar cómo va a ser el consumo eléctrico a lo largo del periodo que interese, se puede comenzar a analizar los resultados en tiempo real, de forma que, en comparación con la predicción, hubiese una gran diferencia entre el consumo real y el estimado, se podría detectar de forma rápida y precoz que algo está sucediendo, permitiendo tomar medidas a la mayor brevedad posible, y corregir los posibles problemas de forma rápida y eficaz.

En este trabajo se van a analizar dos tipos de errores sistemáticos, de ganancia y de offset.

- Error de *offset*: aquel que desplaza toda la señal una misma cantidad, constante en el tiempo. La forma habitual de detectar si existe este tipo de error es comprobar la salida del sistema, cuando no hay señal a la entrada, de esta forma, teniendo un sistema conocido, es posible observar el error de *offset* presente. En este trabajo, se comprobará que la señal medida se encuentra constantemente desviada de la predicha, de forma que siempre tenga el mismo error en todos sus puntos.

$$X = X \pm X * offset \tag{7}$$

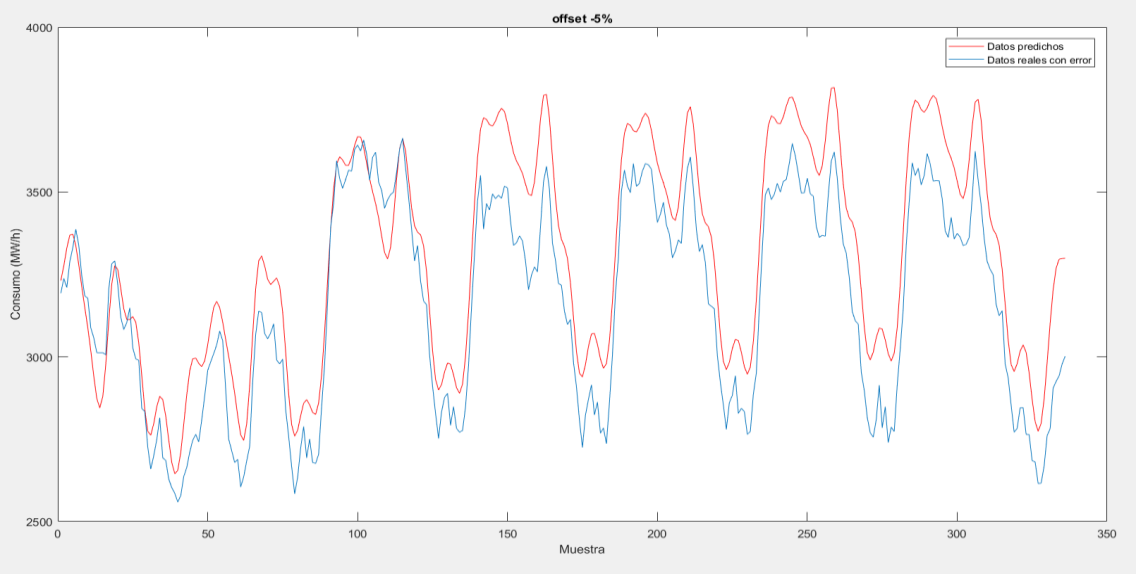


Figura 30: Offset -5%

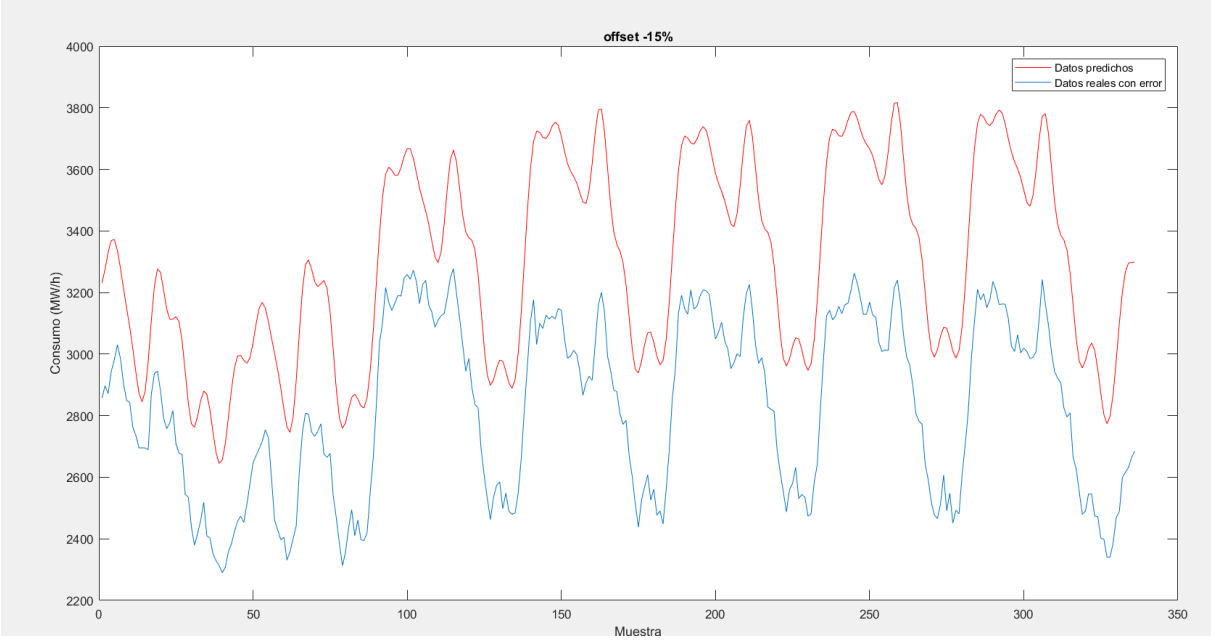


Figura 31: Offset -15%

- Error de ganancia: representa el error en la pendiente de la salida real, en comparación de la ideal. A diferencia del error de *offset*, no es constante en todos sus puntos. Es proporcional a la señal de entrada, por lo que, a mayor valor, el error de ganancia será mayor. Debido a esto también se le conoce como error de pendiente

$$X = X * (1 - E_{ganancia}) \tag{8}$$

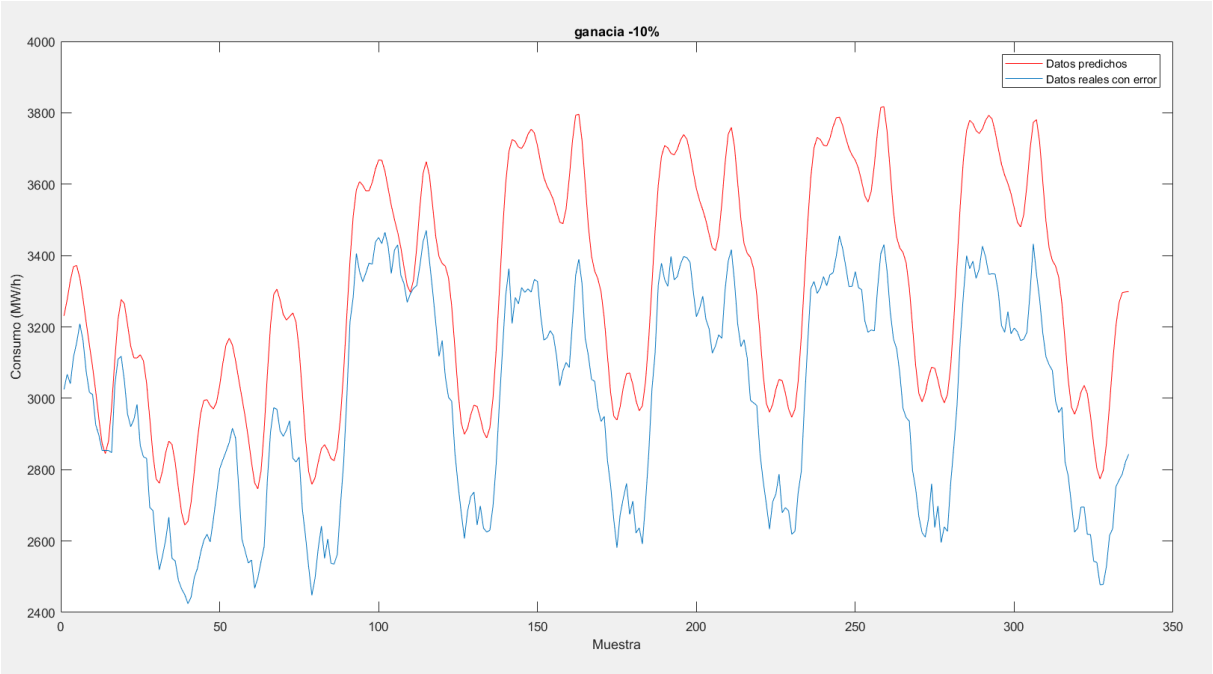


Figura 32: Ganancia -10%

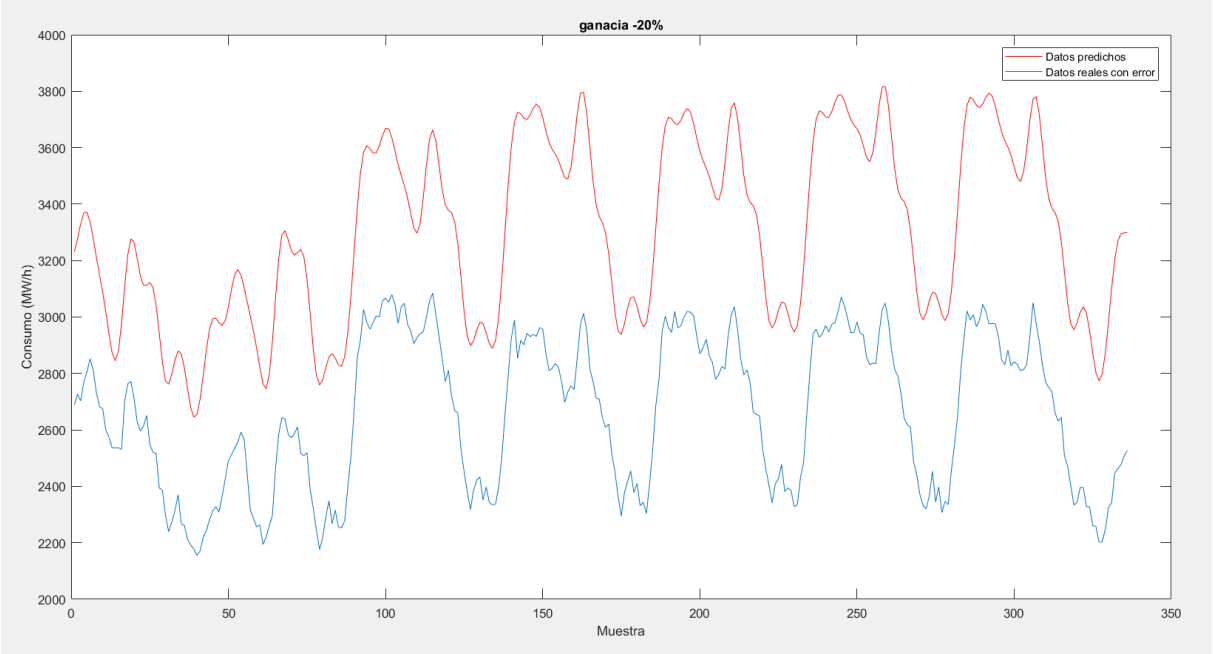


Figura 33: Ganancia -20%

Visto de esta forma, parece sencillo diferenciar qué error existe en la señal, pero, lo más habitual, es que se encuentren los dos errores ocurriendo de forma simultánea, además de otros errores que en este trabajo no se tienen en cuenta, quedando la fórmula de la siguiente forma:

$$X_{erronea} = X * \alpha + \beta \tag{9}$$

Donde $X_{erronea}$ representa la señal con errores, X la señal predicha considerándola como ideal, α como error de ganancia y β el error de *offset*. Como se ha comprobado en el apartado anterior, la predicción no está exenta de error, sino que tiene un nivel de error propio del proceso matemático seguido para predecir el consumo. Por ello, la ecuación se puede rescribir de la siguiente forma:

$$X_{erronea} = \left((X_{ideal} + E_x) * \alpha\right) + \beta = X_{ideal} * \alpha + E_x * \alpha + \beta \tag{10}$$

Donde X_{ideal} representa el valor predicho y E_x el error de predicción. En cambio, para este apartado se va a suponer que este error es despreciable, para facilitar los cálculos.

Para el error de ganancia, utilizando la ecuación 9 y aplicando derivadas, se puede obtener:

$$\frac{\Delta X_{erronea}}{\Delta t} = \frac{\Delta(X_{ideal} * \alpha + \beta)}{\Delta t} = \frac{\Delta X_{ideal}}{\Delta t} * \alpha \quad (11)$$

Y de aquí, se despeja α:

$$\alpha = \frac{\Delta X_{erronea}/\Delta t}{\Delta X_{ideal}/\Delta t} = \frac{\Delta X_{erronea}}{\Delta X_{ideal}} \quad (12)$$

Y, una vez obtenido el error de ganancia, despejando de la ecuación (9), se puede obtener el error de *offset*

$$\beta = X_{erronea} - X_{ideal} * \alpha \quad (13)$$

Tras esto, si se quiere el valor del *offset* en porcentaje, se tendría que dividir el valor obtenido entre el valor de la señal ideal.

Con estas dos fórmulas se podría determinar el error en tiempo real, y planificar unas pautas de actuación en caso de que el error superase unos valores umbrales determinados.

A continuación, se procederá a introducir un error controlado y ver cómo afecta dicho error al MAPE. Para ello, se utilizará un *script* de Matlab, en el cual, aplicando las fórmulas anteriormente indicadas (12-13), se obtendrá tanto el MAPE como el NMBE.

Para decidir cuándo hay error o cuando no, habría que llegar a un compromiso, ya que el error entre distintas predicciones puede variar (por ejemplo, en el proceso de calcular el MAPE para los distintos métodos, en determinados meses se obtuvo un MAPE superior al 10% y en otros inferior al 2% para un mismo método). Por ello, es complicado definir un valor restrictivo, porque en una predicción más exacta, por ejemplo, MAPE=4%, con error de *offset* puede llegar a un MAPE de 10%, el cual es el obtenido en otro mes sin error. Por ello, en este trabajo, se va a considerar que existe error cuando el valor supere el 15% avisando con un mensaje cuando esto ocurra. Con este aviso, es posible permitir que la detección de errores se haga en tiempo real, ya que cada media hora se introducirán los nuevos valores reales, se compara con la predicción y se calcula el MAPE, avisando si ha ocurrido algo inesperado.

Para demostrar esto que se comentó anteriormente, se han realizado pruebas con una predicción, que sin errores tiene un MAPE de 1.85%. A continuación, se pueden observar algunas gráficas de varias pruebas, y en las Tablas 2, 3 y 4 los resultados totales obtenidos:

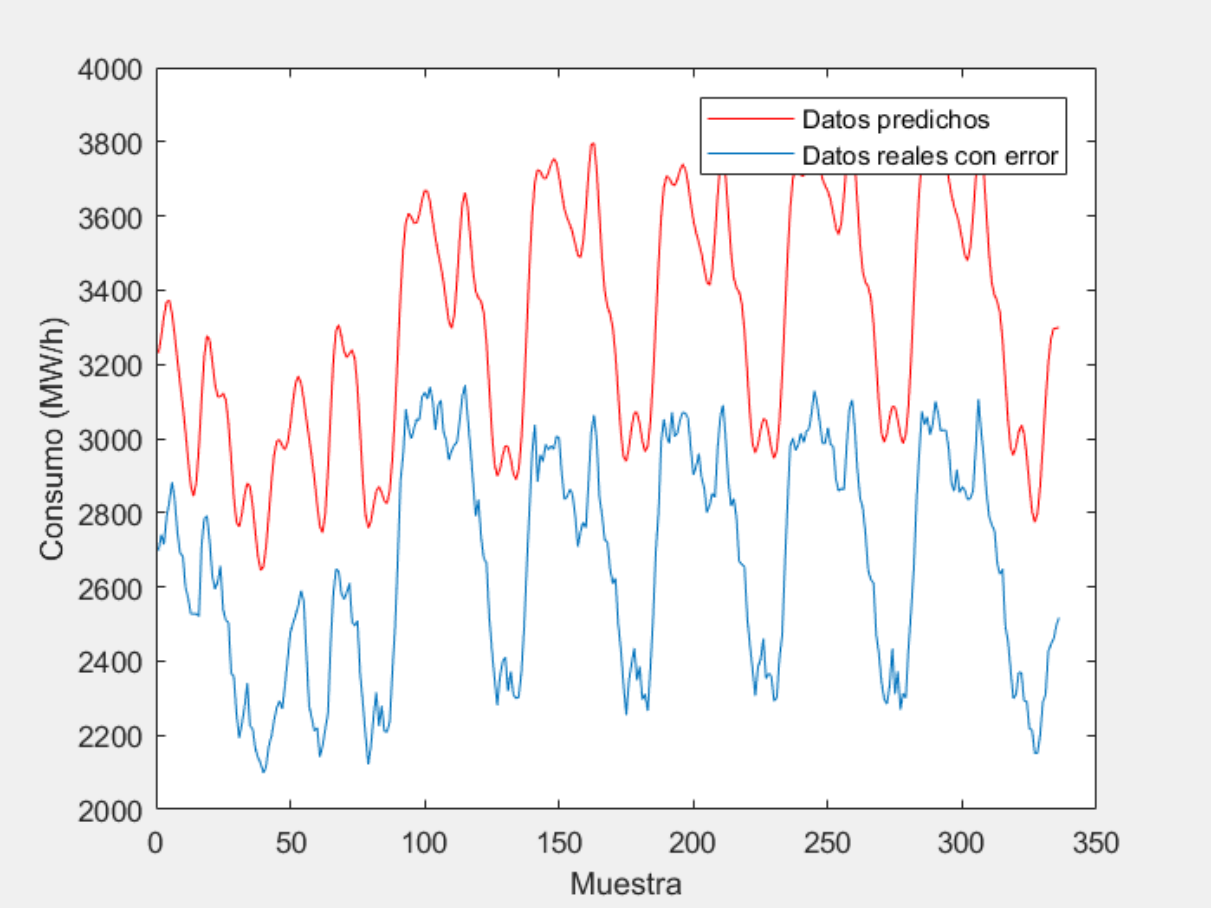


Figura 34: Offset -10% y ganancia 10%

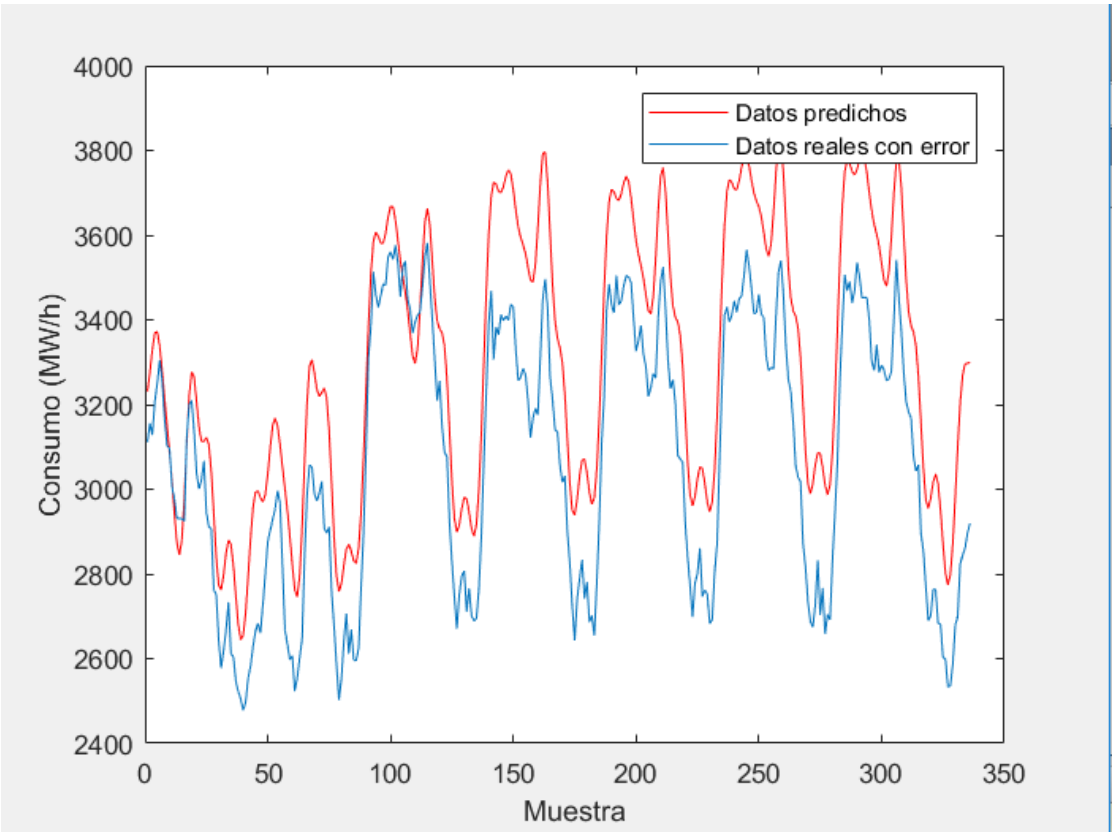


Figura 35: Offset -2.5% y ganancia -5%

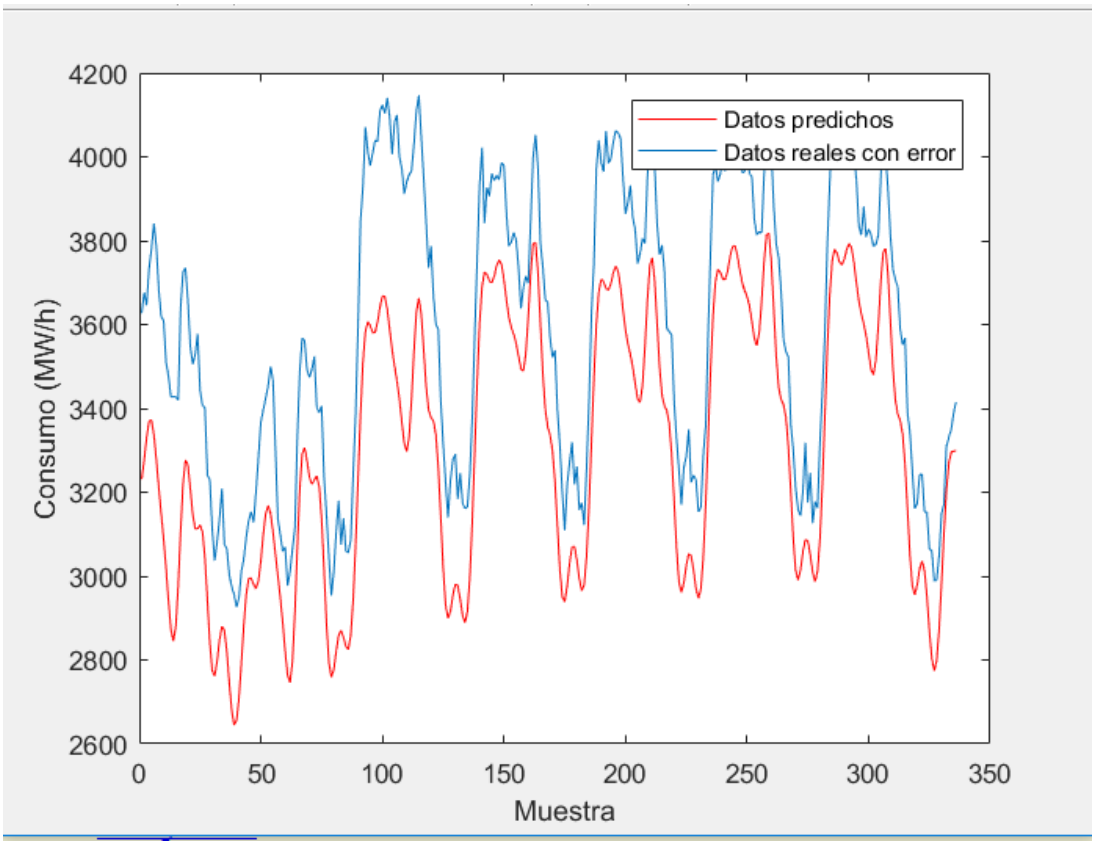


Figura 36: Offset 3% y ganancia 5%

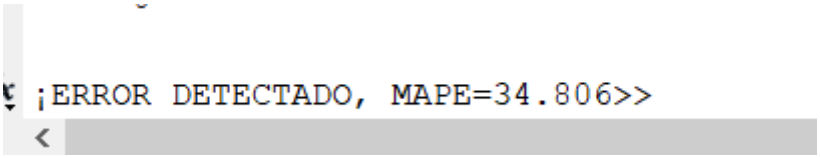


Figura 37: Mensaje de Error cuando MAPE supera el 15%

NIVEL DE ERROR (%)	MAPE (%)	NMBE (%)
-10	10.86	-10.85
-7.5	8.00	-7.89
-5	5.38	-5.08
-2.5	3.13	-2.42
0	1.85	0.10
2.5	2.72	2.50
5	4.80	4.70
7.5	6.98	6.98
10	9.07	9.07

Tabla 2: MAPE y NMBE con error de offset

NIVEL DE ERROR (%)	MAPE (%)	NMBE (%)
-10	11.03	-11.85
-7.5	8.09	-8.02
-5	5.38	-5.10
-2.5	3.13	-2.42
0	1.85	0.10
2.5	2.83	2.54
5	4.90	4.80
7.5	7.09	7.09
10	9.21	9.21

Tabla 3: MAPE y NMBE con error de ganancia

		OFFSET									
		-10		-5		0		5		10	
GANANCIA	-10	24.7	-24.7	17.4	-17.4	11	-11	5.4	-5.2	1.8	-0.1
	-5	17.3	-17.3	10.9	-10.9	5.3	-5.1	1.8	0	4.7	4.7
	0	10.8	-10.8	5.3	-5	1.8	0.1	4.8	4.7	9	9
	5	5.3	-5	2	0.1	4.9	4.8	9.1	9.1	13	13
	10	2.25	0.2	4.9	4.9	9.2	9.2	13.1	13.1	16.6	16.6

Tabla 4: MAPE Y NMBE con ambos errores

El tiempo de ejecución del *script* es de 0.619558 en un ordenador con procesador Intel core I7 de cuarta generación, por lo que, se podría aplicar a sistemas de detección en tiempo real, ya que no supone un consumo de tiempo significativo.

VII. Conclusiones

En este trabajo se ha realizado el desarrollo completo de un proyecto que podría ser utilizado en los centros de transformación, para mejorar su respuesta contra errores no deseados, ya sean solo de medida, o errores más graves, los cuales también se podrían detectar en tiempo real. Para ello, solo requerirían de un histórico de consumo y una aplicación que desempeñe este proceso.

Como se ha podido comprobar, el aplicar el algoritmo SSA puede ser muy útil a la hora de realizar predicciones en series temporales como el consumo eléctrico, debido a la eliminación de componentes ruidosas. También, la librería de SSA de R es de gran utilidad, debido a que permite aplicar SSA a la señal, evitándonos la parte matemática del algoritmo e incluye funciones de predicción, que hemos utilizado en este trabajo. Las predicciones tienen una gran exactitud con lo que ocurre en la realidad, aunque, como hemos observado, no todos los métodos son igual de exactos, para una serie de estas características, pero siendo todos bastante buenos.

Matlab permite facilitar muchos procesos matemáticos y de representación gráfica, facilitando en gran medida la repetición de procesos, con la creación de scripts.

Como objetivo de este trabajo, también se pretende guiar el curso de futuros desarrollos, alcanzando una mayor profundidad en la predicción, que va más allá del alcance de este proyecto. Una mejora muy interesante podría ser la introducción de una nueva serie temporal de predicción temporal, tanto de temperatura, como de precipitaciones, ya que, en mayor o menor medida, el consumo eléctrico en domicilios varía en función de estos parámetros, ya que, si nos enfrentásemos a un mes de verano más frio de lo habitual, o un invierno cálido se vería reflejado en el consumo doméstico en calefacción, luz, etc.

Otra parte interesante sería comparar estas predicciones con las obtenidas de redes neuronales, que es otra forma útil de predicción de series temporales.

Vistos estos dos ejemplos se puede observar que aún quedan muchas posibles mejoras en este ámbito, tanto en el tema de las predicciones de consumo, utilizando más parámetros importantes en el consumo como los mencionados, como en el ámbito de las predicciones de series temporales propiamente dicho, desarrollando nuevos métodos y nuevos algoritmos que permitan realizar predicciones aún más exactas y en un tiempo menor.

En último lugar, remarcar que en el trabajo llevado a cabo se han llegado a unas conclusiones y resultados razonablemente satisfactorios, llevándose a cabo según lo previsto y comprobado que mediante SSA se pueden conseguir predicciones de elevada precisión, pudiendo incluso elevar aún más la precisión de los resultados, ajustando valores como la ventana, a analizar, o el número de componentes, pudiendo buscar valores óptimos para estos datos en función del resultado esperado o de la entrada a analizar.

VIII. Código

- Base_datos.m

Código que lee la base de datos y devuelve los valores que utilizaremos eliminando el resto y guardándolos en formato csv.

```
clear all;

A=xlsread('C:\Users\Alejandro\Desktop\datos2014.xlsx','datos2014','E
2:E35041');
A(isnan(A))=0;
A=A';
A(A==0) = [];
i =[1:1:17520];
plot(i,A)
figure();
plot(i(1:48),A(1:48))
figure();
plot(i(1:96),A(49:144))
A=A';
csvwrite('datos2014.csv',A)

B=xlsread('C:\Users\Alejandro\Desktop\datos2015.xlsx','datos2015','E
2:E35041');
B(isnan(B))=0;
B=B';
B(B==0) = [];
i =[1:1:17520];
plot(i,B)
figure();
plot(i(1:48),B(1:48))
figure();
plot(i(1:96),B(49:144))
B=B';
csvwrite('datos2015.csv',B)
```

- Bvector.R

Codigo de predicción en R de datos de una semana mediante el método bootstrap-vector, para el resto de métodos el código es el mismo, cambiando el método y los parámetros necesarios en el mismo.

```
datos=read.csv("Datostotalsinsem.csv",header=FALSE,row.names=NULL)
datos=as.ts(as.numeric(unlist(datos[1],use.names=FALSE)))
L=48*7
library(Rssa)
datosssa=ssa(datos,L)
f <- forecast(datosssa, groups = list(1:24), method = "bootstrap-
vector", len = 336, R = 50)
pred=f$mean
write.csv(as.integer(pred), file="salida.csv",row.names=FALSE)
```

- Comparacionprep.m

Código que se encarga de comparar en gráficas la salida de la predicción con la señal real. En primer lugar muestra la señal real, en la segunda figura muestra la comparación de señal real y la predicha y la última muestra un día de la señal predicha.

```
clear all;

datosreal = csvread('C:\Users\Alejandro\Documents\mi
proyecto\datostotal.csv');
datosreal=datosreal';
datospred = csvread('C:\Users\Alejandro\Documents\mi
proyecto\prediccion.csv');
i=[1:1:35040];
figure()
plot(i,datosreal)
ylabel 'Consumo (MW/h)';
xlabel 'Muestra';
figure()
plot(i(end-672:end-336),datosreal(end-672:end-336),'black')
hold on;
plot(i(end-336:end),datosreal(end-336):end),'b')
plot(i(end-336:end),datospred(end-336:end),'r')
legend('señal','Datos reales','Datos predichos');
figure()
plot(i(end-336:end-288),datospred(end-336:end-288),'r')
```

- Seleccion.m

Con este código seleccionamos de que mes del año queremos hacer la predicción, y nos devuelve el histórico de un año anterior al mes elegido y le resta una semana más, ya que, en nuestro caso, predecimos una semana ya existente para comparar con dicha semana la predicción.

```
clear all;

datosreal = csvread('C:\Users\Alejandro\Documents\mi
proyecto\datostotal.csv');

mes=1;
m_anio=17520;
m_total=m_anio*2;
m_sem=336;
m_mes=1460;

for i=1:(m_anio)
    datoselegidos(m_anio-(i-1))=datosreal(m_total-(i-1)-((12-
mes)*m_mes));
end

datoselegidos=datoselegidos';
csvwrite('C:\Users\Alejandro\Documents\mi
proyecto\datosanio.csv',datoselegidos);
for i=1:(m_anio-336)
    datoselegidossinsem(i)=datoselegidos(i);
end
 datoselegidossinsem=datoselegidossinsem';
csvwrite('C:\Users\Alejandro\Documents\mi
proyecto\datosaniosinsem.csv',datoselegidossinsem);
```

- Mape.m

Este archivo se encarga de calcular el MAPE y el NMBE de la predicción llevada a cabo.

```
clear all;
datosreal = csvread('C:\Users\Alejandro\Documents\mi
proyecto\datosanio.csv');
datosreal=datosreal';
datospred = csvread('C:\Users\Alejandro\Documents\mi
proyecto\salida.csv');

sumam=0;
sumab=0
i=0


for i=0:336


        sumam=(sumam+abs((datosreal(end-(336-i))-datospred(end-(336-
i)))/(datosreal(end-(336-i)))));
        sumab=(sumab+((datosreal(end-(336-i))-datospred(end-(336-
i)))/(datosreal(end-(336-i)))));
end

MAPE=(100*sumam)/288;
NMBE=(100*sumab)/288;
```

IX. Presupuesto

En este punto describiremos el presupuesto teórico aproximado que costaría llevar a cabo este proyecto. Se van a incluir los costes de materiales y las tasas profesionales.

- Coste de materiales

Producto		Precio (€)	Duración	Uso	Total
Hardware	PC Windows i7	700	4 años	8 meses	116,66 €
Precio Hardware total					116,66 €
Software	MATLAB	120	vitalicio	vitalicio	120,00 €
	Rstudio	0	vitalicio	vitalicio	- €
	RSSA package	0	vitalicio	vitalicio	- €
Precio Software total					120,00 €
Precio total de material					236,66 €

Tabla 5 Costes material

- Coste mano de obra

Trabajador	Precio (€)	Tiempo (meses)	Total (€)
Ingeniero	2000	7	14.000,00 €
Mecanógrafo	1000	1	1.000,00 €
Coste total			15.000,00 €

Tabla 6 Coste mano de obra

- Coste de ejecución de material

Coste	Total
Coste del material	236,66 €
Coste de mano de obra	15.000,00 €
Coste total	15.236,66 €

Tabla 7 Ejecución de material

- Gastos generales y beneficio industrial

Gastos generales y beneficio industrial	Total
(25% del coste de ejecución del material)	3.809,16 €

Tabla 8 Gastos generales y beneficio industrial

- Presupuesto de ejecución por contrata

Coste	Total
Coste de ejecución del material	15.236,66 €
Gastos generales y beneficios industriales	3.809,16 €
Coste total	19.045,82 €

Tabla 9 Ejecución por contrata

- Honorarios

Coste	Total
Redacción del proyecto	860,00 €

Tabla 10 Honorarios

- Importe total

Ejecución por contrata	19.045,82€
Honorarios de redacción	860€
Total	19.905,82 €

Tabla 11 Costes totales

Con estos datos, el coste total asciende a 19905.82 €.

X. Bibliografía

[1] «Adjusting and calibrating out offset and gain error in a precision dac,» [En línea]. Available: http://www.maximintegrated.com/en/app-notes/index.mvp/id/4602.

[2] M. Schumacher, C. Hoga y J. Schmid, «Get on the digital bus to substation automation,» *Power and Energy Magazine,* pp. 51-56, 2007.

[3] S. Sheng, D. Xianzhong, W. Chan y L. Zhihuan, «Erroneous measurement detection in substation,» *International Journal of Electrical Power,* vol. 31, nº 7, pp. 351-355, 2009.

[4] K.-L. Ho, Y.-Y. Hsu y C.-C. Yang, «Short term load forecasting using a multilayer neural network,» *Power Systems, IEEE Transactions on,* vol. 7, nº 1, pp. 141-149, 1992.

[5] N. Golyandina, V. Netrutkin y A. A. Zhigljavsky, Analysis of Time Series Structure: SSA and related techniques, Chapman & hall/CRC, 2001.

[6] J. Moriano, F. J. Rodríguez, P. Martín, J. A. Jiménez y B. Vuksanovic, «A New Approach to Detection of Systematic Errors in Secondary Substation Monitoring Equipment Based on Short Term Load Forecasting,» *Sensors,* p. 14, 2016.

[7] C. Cordeiro y M. Neves, «The bootstrap methodology in time series,» de *17th COMPSTAT Symposium of the IASC*, Roma, 2006.

[8] D. Park, M. El-Sharkawi, R. Marks, L. Atlas y M. Damborg, «Electric load forecasting using,» *Power Systems, IEEE Transactions on,* vol. 6, nº 2, pp. 442-449, 1991.

[9] S. Naidu, E. Zafiriou y T. McAvoy, «Use of neural networks for sensor failure detection in a control system,» *Control Systems Magazine,* vol. 10, nº 3, pp. 49-55, 1990.

[10] Q. Pengpeng, L. Jinguo, Z. Wei y L. Yangmin, «Sensor fault diagnosis method based on fractal dimension,» *Control and Decision Conference (CCDC), 2013 25th Chinese,,* pp. 2955-2960, 2013.

[11] M. Napolitano, C. Neppach, V. Casdorph, S. Naylor, M. Innocenti y G. Silvestri, «Neural-network- based scheme for sensor failure detection, identification, and accommodation,» *Journal of Guidance, Control, and Dynamics*, vol. 18, nº 6, pp. 1280-1286, 1995.

[12] S. Sheng, D. Xianzhong, W. Chan y L. Zhihuan, «Erroneous measurement detection in substation automation system using ols based rbf neural network,» *International Journal of Electrical Power & Energy Systems,* vol. 31, nº 7-8, pp. 351-355, 2009.

[13] A. Korobeynikov, A. Shlemov, K. Usevich y N. Golyandina, «Rssa: A Collection of Methods for Singular Spectrum Analysis,» 2016.