Ejercicios de Análisis y Tratamiento de Datos en Geofísica y Meteorología. Curso 2021-22

Alberto Calzada Chávez

Enero de 2021

Índice

1 Introducción

En este trabajo se pretende resolver una serie de ejercicios propuestos utilizando el lenguaje de programación MATLAB. Los ejercicios tratan los distintos conceptos vistos en clase sobre la teoría y el análisis de señales temporales. En el primer ejercicio (Sec. 2) trabajaremos con señales de tipo sísmico, mientras que en el segundo (Sec. 3) lo haremos con señales meteorológicas. El código escrito en MATLAB para la resolución de los ejercicios puede consultarse en el Apéndice A.

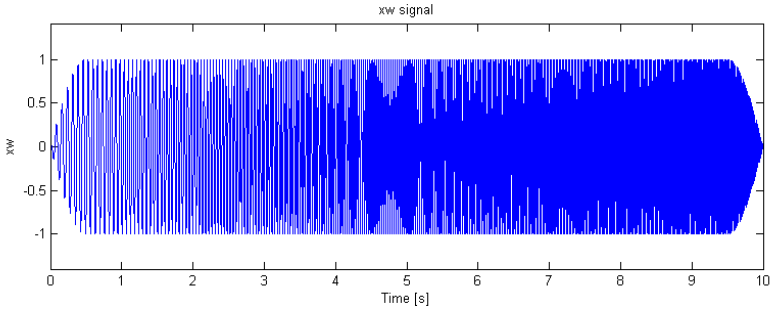
2 Ejercicio 1: Análisis de señales sísmicas

2.1 Ejercicio 1.1: Análisis de la señal de un vibrador sísmico

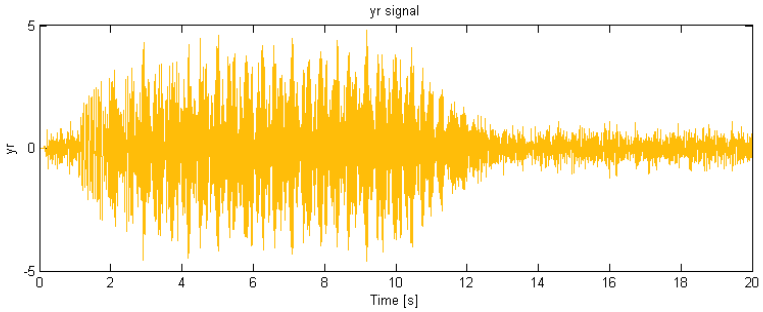
En este primer ejercicio trabajamos con la señal de un vibrador sísmico. Los datos, que encontramos en el fichero **vibroseis.mat**, cuentan con las siguientes variables:

- **xw**: Señal generada por un vibrador sísmico.
- **yr**: Simulación de la respuesta del suelo a la excitación xw.
- **dt**: Intervalo de muestreo.

Lo primero que se nos pide es representar xw e yr en función del tiempo. En la Fig. 1 hemos representado estas dos señales.



(a) Señal xw.



(b) Señal yr.

Figura 1: Señales xw e yr del conjunto de datos vibroseis.mat

Lo siguiente que hacemos es calcular la transformada de Fourier de xw para representar su módulo en función de la frecuencia (líneas 43 a 58 de A.1). Según nos dice el enunciado del ejercicio, la señal xw está generada mediante un barrido senoidal con frecuencias desde unos 10Hz hasta los 60Hz. Por ello, esperamos que el módulo de la transformada de Fourier en este rango de frecuencias sea visiblemente mayor que para el resto de frecuencias.

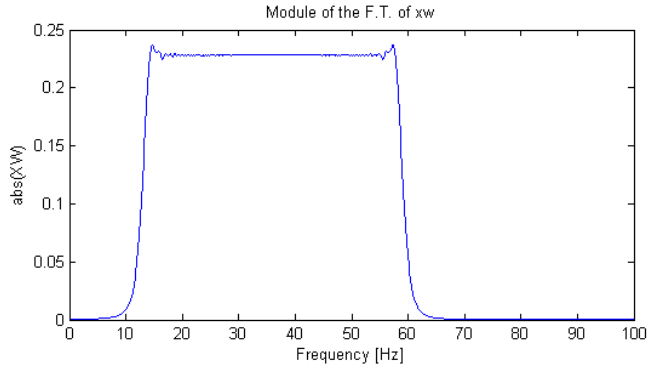


Figura 2: Módulo de la transformada de Fourier de xw en función de la frecuencia.

En la Fig. 2 podemos ver que, efectivamente, las frecuencias que componen la señal xw se encuentran en el rango 10 a 60Hz. Con este ejemplo se puede entender muy bien la utilidad de la transformada de Fourier en el análisis de series temporales. Esta herramienta matemática nos permite cambiar del dominio del tiempo al de la frecuencia nuestras señales para trabajar con éstas en el que más nos convenga.

Una vez vista la composición en frecuencias de xw, pasamos a ver su espectrograma, lo que corresponde al código escrito entre las líneas 60 y 68 de A.1. La función **spectrogram** [MathWorks 2021f] de MATLAB realiza la transformada de Fourier discreta de la señal que le pasemos como argumento. Para ello, nosotros hemos escogido una ventana de 1024 puntos. La resolución en frecuencias que conseguimos entonces es $\Delta f = 1/1024dt = 0.98Hz$, más que suficiente para observar con nitidez el rango de 50Hz (de 10 a 60Hz) en el que estamos interesados.

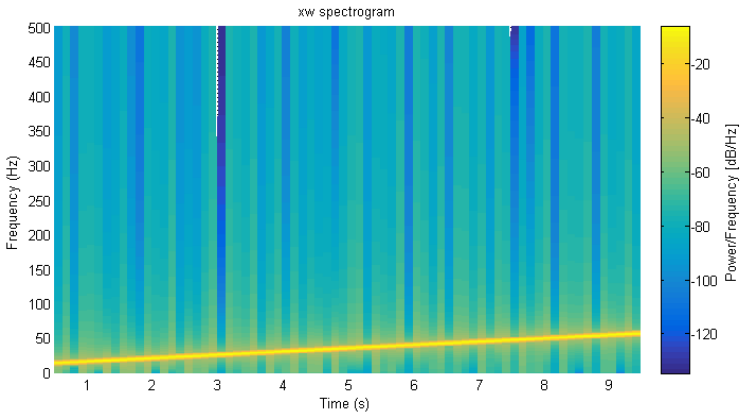


Figura 3: Espectrograma de la señal xw.

El espectrograma de una señal nos va a mostrar el contenido en frecuencia estimado de una señal a lo largo del tiempo. En la Fig. 3 hemos representado el espectrograma de xw. Como podemos observar, la frecuencia de la señal se encuentra en el intervalo 10 a 60Hz, como ya sabíamos. Además dependencia con el tiempo es lineal, desde los 10Hz en un inicio hasta los 60Hz a los 10s. Esto coincide con lo que observamos en la Fig. 1a, donde vemos que inicialmente los «picos» de la señal senoidal están mas separados (menor frecuencia) que al final.

Para acabar, vamos a analizar la correlación cruzada entre las señales xw e yr, la respuesta del suelo, para determinar los tiempos de viaje de las reflexiones. El código escrito para

realizar este análisis es el comprendido entre las líneas 70 y 88 de A.1.

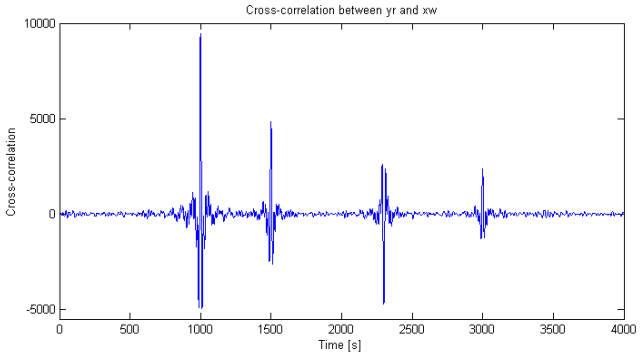


Figura 4: Correlación cruzada entre las señales xw e yr.

En la Fig. 4 hemos representado la correlación cruzada entre xw e yr. Tal y como podemos observar, hay determinados instantes para los que la correlación se dispara. Estos tiempos corresponden a los tiempos de viaje de las reflexiones de la señal sísmica. Utilizando la herramienta de gráficos de MATLAB para obtener las coordenadas de los puntos de una señal, determinamos que los tiempos de viajes son: 1000s, 1500s, 2300s y 3000s.

2.2 Ejercicio 1.2: Análisis de la componente vertical del sismograma de Hokaido

Pasamos ahora a analizar la componente verticaldel terremoto de Hokaido, Japón, registrado en la estación de Sierra Elvira. Las variables a utilizar en este ejercicio se encuentran en el fichero **japon.mat** y son las siguientes:

- **wx**: Componente vertical de la velocidad del suelo.
- **t**: Vector de tiempo en minutos.

Además, tenemos un factor de conversión de la velocidad del suelo para pasar de número de cuentas a $\mu m/s$. En nuestro caso vamos a trabajar en segundos, por lo que le aplicamos el factor de conversión a wx y multiplicamos por 60 el vector t. Esto lo hacemos en A.1 entre las líneas 93 y 116.

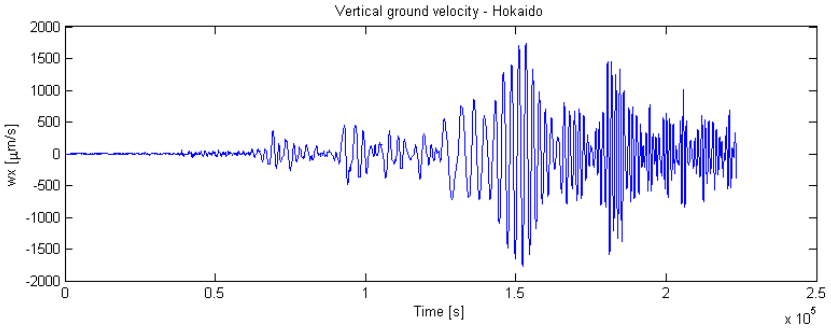


Figura 5: Velocidad vertical del suelo registrada en la estación de Sierra Elvira, Granada, tras el terremoto de Hokaido, Japón.

A modo de referencia para el resto del ejercicio, en la Fig. 5 se ha representado la velocidad vertical del suelo wx.

Lo primero que se nos pide en este ejercicio es estimar el rango de periodos del tren de ondas superficiales. Para ello, utilizamos la herramienta gráfica de MATLAB que mencionamos

antes y calculamos la diferencia temporal entre picos consecutivos de wx. En particular, escogemos un par de picos consecutivos alrededor de $1.5 * 10^5 s$ y otro par de picos en el entorno de $1.8 * 10^5 s$. Escogemos estos instantes pues es alrededor de estos donde se concentran las ondas de mayor amplitud. Los periodos observados son:

$$t_2 - t_1 = 1.534 * 10^5 s - 1.510 * 10^5 s \rightarrow T_1 = 2.4 * 10^3 s \quad \text{y} \tag{2.1}$$

$$t_2 - t_1 = 1.818 * 10^5 s - 1.806 * 10^5 s \rightarrow T_2 = 1.2 * 10^3 s \ . \tag{2.2}$$

Una vez estimados los periodos, se nos pide diseñar un filtro paso-baja de tipo FIR para dejar pasar únicamente las ondas superficiales, filtrando las de mayor frecuencia. Puesto que no estamos seguros de si el segundo periodo estimado corresponde a ondas superficiales o no, hemos construido un filtro que únicamente deja pasar ondas con periodo igual o mayor que T_1 y otro que deja pasar ondas con periodo igual o mayor que T_2 (incluyendo, entonces, las de periodo T_1).

Para diseñar los filtros lo primero que necesitamos son las frecuencias de corte a utilizar. A partir de T_1 y T_2 (Ec. (2.1) y (2.2), respectivamente), es fácil estimar éstas:

$$f_1 = \frac{1}{T_1} = 4.1667 * 10^{-4} Hz \rightarrow fc_1 = 0.0005 Hz \ , \tag{2.3}$$

$$f_2 = \frac{1}{T_2} = 8.3333 * 10^{-4} Hz \rightarrow fc_2 = 0.0009 Hz \ . \tag{2.4}$$

Puesto que para construir los filtros vamos a utilizar la función **fir1** [MathWorks 2021c] de MATLAB, necesitamos ahora definir la longitud de nuestros filtros. Para ello tenemos que tener en cuenta que la longitud elegida (en número de puntos) debe corresponder a un periodo de tiempo mayor que el correspondiente a la frecuencia de corte escogida. En nuestro caso, a la frecuencia de corte fc_1 le corresponde un periodo de tiempo de $1/0.0005 Hz = 2 * 10^4 s$. Dividiendo por dt tenemos que estos son unos $1.67 * 10^3$ puntos. Por su parte, a la frecuencia de corte fc_2 le corresponde un periodo de tiempo de $1/0.0009 Hz = 1.1111 * 10^3 s$. Dividiendo por dt tenemos que estos son unos $0.9259 * 10^3$ puntos.

De esta forma, escogemos la siguiente longitud para cada uno de los filtros:

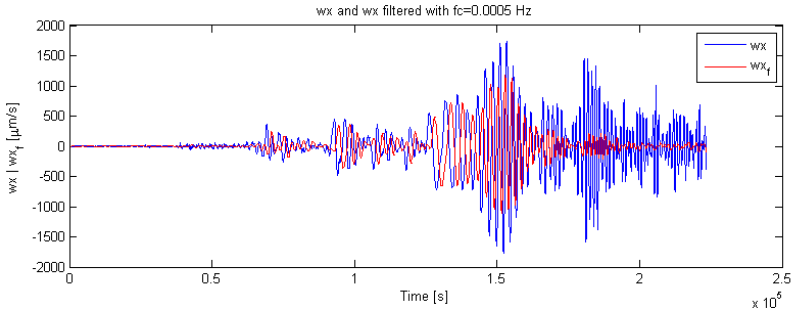
$$l\_filtro_1 = 3 * 10^3 \text{ puntos} \tag{2.5}$$

$$l\_filtro_2 = 2 * 10^3 \text{ puntos} \tag{2.6}$$

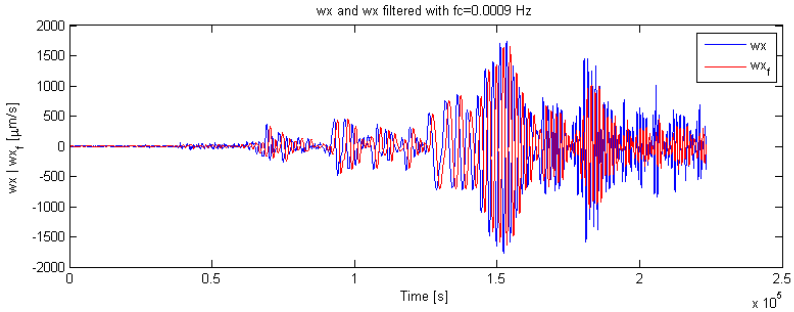
Por último, tenemos que tener en cuenta que la frecuencia que hay que utilizar en con la función **fir1** es la de corte normalizada con la Frecuencia de Nyquist, que se define como la mitad de la frecuencia de muestreo. La Frecuencia de Nyquist corresponde a la frecuencia más alta para la que disponemos de información al muestrear una señal con una frecuencia determinada. Otra particularidad a tener en cuenta cuando diseñamos filtros FIR es que tan solo tenemos un coeficiente a y es igual a 1. Todo esto que estamos explicando corresponde al código escrito entre las líneas 118 y 165 de A.1.

Antes de pasar al siguiente punto, es interesante e ilustrador que veamos el efecto que tiene cada uno de los filtros sobre la señal wx. Como hemos dicho, esperamos que el filtro con frecuencia de corte fc_1 deje pasar principalmente la señal con periodo igual o mayor que el que tiene alrededor de $1.5 * 10^5 s$; mientras que el filtro con fc_2 dejará pasar tanto componentes de la señal con este periodo como con el periodo de la señal alrededor de $1.8 * 10^5 s$.

Como podemos ver en la Fig. 6, la señal filtrada utilizando cada uno de los filtros corresponde con lo que esperábamos al diseñarlos.



(a) Señal wx y señal filtrada utilizando filtro FIR con frecuencia de corte $fc_1 = 0.0005Hz$.



(b) Señal wx y señal filtrada utilizando filtro FIR con frecuencia de corte $fc_2 = 0.0009Hz$.

Figura 6: Señal wx y señal filtrada mediante filtro FIR utilizando la frecuencia de corte a) 0.0005Hz y b) 0.0009Hz.

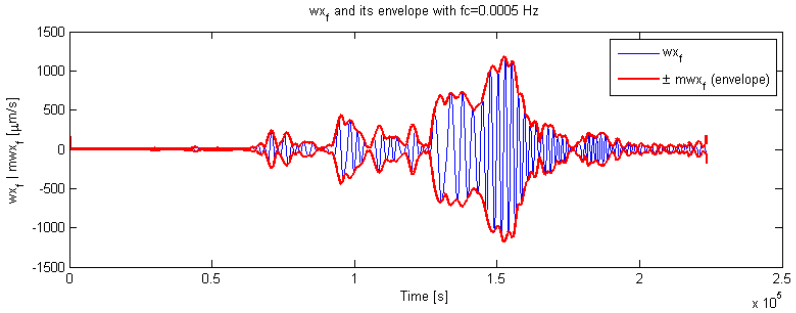
Pasamos ahora a calcular la envolvente de la señal filtrada y a representarla junto a ésta. El código MATLAB que hemos escrito para hacer esto corresponde a las líneas 167 a 184 de A.1.

En primer lugar, calculamos la función analítica correspondiente a wx_f, que es la señal filtrada. Esto lo hacemos utilizando la función **hilbert** [MathWorks 2021d] de MATLAB. El resultado será una función compleja cuya transformada de Fourier no tiene componentes negativas de frecuencia (es por esto por lo que se llama función «analítica»). Como partíamos de una función real (wx_f) hacer esto no supone una pérdida de información.

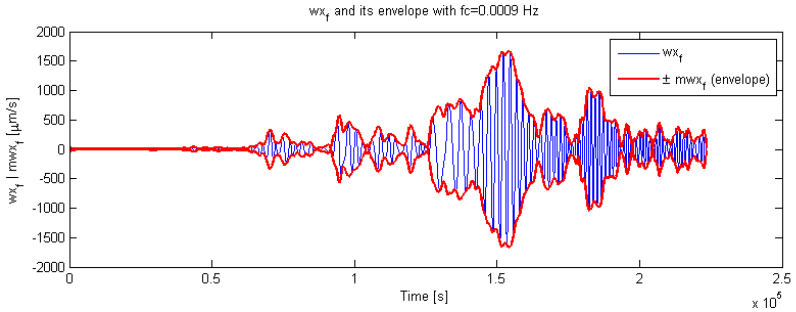
Una vez calculada la función analítica calculamos su módulo y la representamos junto con la señal filtrada. El resultado lo podemos ver en la Fig. 7. En esta figura se puede ver de forma clara la diferencia de la señal filtrada utilizando un filtro u otro.

Para acabar este ejercicio, y con el análisis de las señales de tipo sísmico, vamos a simular la respuesta de un sismómetro de corto periodo. Para ello, vamos a aplicar un filtro Butterworth paso-alta de 1Hz a la señal original y vamos a representar luego la señal filtrada. Esto es lo que hacemos entre las líneas 186 y 204 de A.1.

Como siempre, para diseñar un filtro del tipo que sea tenemos que conocer los parámetros que necesitamos pasarle a la función de MATLAB que vayamos a utilizar. En este caso la función será **butter** [MathWorks 2021a], que nos permite diseñar este tipo de filtros de forma directa y muy sencilla. Para utilizar esta función, tan sólo vamos a necesitar tres parámetros: el orden, la frecuencia de corte normalizada a la frecuencia de Nyquist (y de la que ya hemos hablado) y el tipo de filtro ('high', 'low', 'bandpass', etc). El orden de un filtro tipo Butterworth va a determinar, entre otras cosas, el número de coeficientes del filtro y la pendiente de la banda de paso. Mientras mayor sea el orden más coeficientes tendremos y mayor será la pendiente de la banda de paso. En nuestro caso, hemos elegido un filtro de orden 4 y como tipo de filtro hemos usado 'high', para dejar pasar las frecuencias mayores a la frecuencia de corte que utilicemos.



(a) Señal filtrada wx_f con frecuencia de corte $fc_1 = 0.0005Hz$ y su envolvente.



(b) Señal filtrada wx_f con frecuencia de corte $fc_2 = 0.0009Hz$ y su envolvente.

Figura 7: Señal wx_f filtrada con frecuencia de corte a) 0.0005Hz y b) 0.0009Hz y su envolvente asociada.

Algo que también debemos tener en cuenta es que nosotros estamos trabajando con el tiempo en unidades de segundos, no minutos. Es por ello que, si probamos a utilizar como frecuencia de corte 1Hz, MATLAB nos mostrará un error en la ventana de comandos que nos dice «*The cutoff frequencies must be within the interval of (0,1)*». En nuestro caso, al trabajar en segundos, esto sucede porque la frecuencia de corte que estamos escogiendo es mayor que la frecuencia de Nyquist, siendo entonces la frecuencia de corte normalizada mayor que 1. La solución a esto es simple: cambiar las unidades de la frecuencia de corte a las apropiadas. La nueva frecuencia de corte será

$$fc = \frac{1 \text{ ciclo}}{\text{min}} = \frac{1 \text{ ciclo}}{\text{min}} * \frac{1 \text{ min}}{60\text{s}} = \frac{1}{60} \; Hz = 0.0167 \; Hz \; . \tag{2.7}$$

Una vez determinada la frecuencia de corte correcta, filtramos la señal wx con el filtro que hemos diseñado. El resultado lo podemos ver en la Fig. 8.

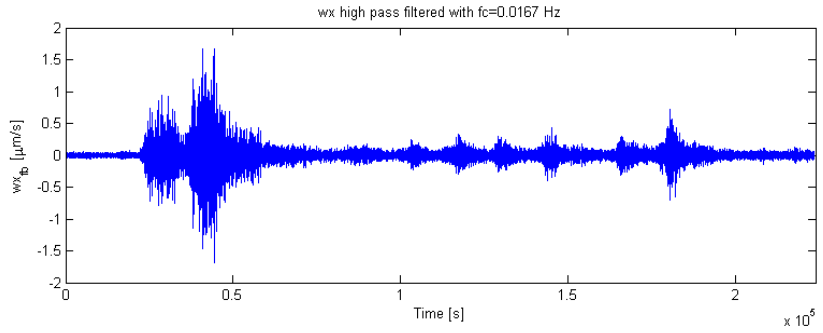


Figura 8: Señal wx filtrada con un filtro Butterworth paso-alta con frecuencia de corte $fc = 1/60Hz$.

Tal y como podemos observar, la amplitud de la velocidad del suelo debida a las compo-

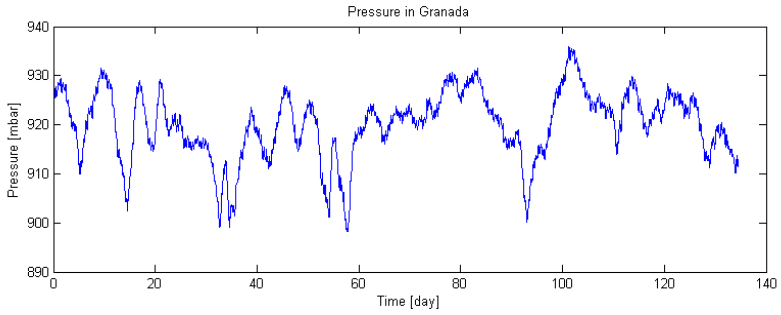
nentes de mayor frecuencia es muy pequeña. Vemos que la respuesta de un sismómetro de corto periodo se concentraría en los primeros instantes del terremoto. En nuestro caso, casi la totalidad de la respuesta de éste se concentra en el intervalo 0.25 a $0.5 * 10^5 s$.

3 Ejercicio 2: Análisis de señales meteorológicas

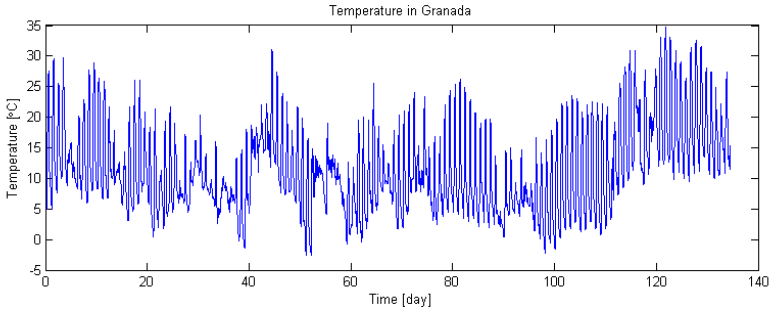
Pasamos ahora a realizar el análisis y resolver los ejercicios asociados a señales de origen meteorológico. El código en MATLAB que hemos escrito para la resolución de este ejercicio puede encontrarse en A.2. Los datos para este ejercicio los encontramos en el fichero **meteo1.mat**, que contiene series de variables meteorológicas de varios meses para un lugar de Granada. Dichas variables son las siguientes:

- **p**: Vector de presión atmosférica en unidades de mbar. Tenemos muestras cada 15min.
- **td**: Vector de tiempo en días.
- **te**: Vector de temperatura del aire.
- **he**: Vector de humedad relativa.
- **ti**: Tiempo inicial de la serie en formato MATLAB.

Lo primero que hacemos es representar tanto la presión como la temperatura en función del tiempo. El código correspondiente lo encontramos en las líneas 25 a 39 de A.2.



(a) Presión atmosférica frente al tiempo.



(b) Temperatura frente al tiempo.

Figura 9: Serie temporal de a) la presión y b) la temperatura en algún emplazamiento de Granada.

Tal y como podemos ver en la Fig. 9 y, en especial, en la serie temporal de la temperatura (Fig. 9b), hay intervalos en los que estas variables toman valores muy altos o bajos de forma continuada. Estos intervalos, como el que va desde el día 90 al 110 aproximadamente, o desde el 120 en adelante, corresponden con cambios estacionales de las condiciones atmosféricas. En

esta serie el día 100 corresponde con el 3 de febrero, lo que es consistente con las temperaturas tan bajas que se han medido.

Pasamos ahora a diezmar la señal a un periodo de muestreo de una hora. La señal que nosotros hemos escogido ha sido la de la temperatura. En el código, esto corresponde con las líneas 41 a 56 de A.2.

Para diezmar la señal vamos a utilizar la función **decimate** [MathWorks 2021b] y no directamente la función **downsample**, ya que la función *decimate* aplica un filtro paso-baja antes hacer el diezmado. Lo hacemos así para protegernos de un posible defecto de «aliasing» tras diezmar la señal. Al diezmar una señal la frecuencia de muestreo disminuye (al aumentar el intervalo de muestreo). Si la señal estaba compuesta por componentes en frecuencia mayores que la mitad de la nueva frecuencia de muestreo, es decir, mayores que la nueva frecuencia de Nyquist, se producirá aliasing.

Puesto que la temperatura, variable *te*, se encuentra muestreada cada 15min, tenemos que diezmarla un factor 4. Para diezmar el vector de tiempo, *td*, basta con utilizar la función *downsample* puesto que es una función lineal y diezmarlo con *decimte* lo deformaría.

A continuación, vamos a crear un filtro paso-alta para eliminar la tendencia estacional presente en los datos y que ya comentamos al inicio del ejercicio, lo que corresponde con las líneas 58 a 87 de A.2. Con dicho objetivo, nosotros hemos decidido utilizar un filtro tipo Butterworth paso-alta. Hemos escogido éste y no uno tipo FIR puesto que el primero no deforma la potencia de la señal en la banda de paso. Además, en este caso no nos preocupa que la pendiente de la banda de transición no sea muy acusada, ya que la tendencia estacional no empieza a ninguna frecuencia concreta. Incluso puede ser beneficioso tener cierta «contaminación» de frecuencias por debajo de la frecuencia de corte que escojamos.

Para eliminar la tendencia estacional hemos escogido una frecuencia de corte de un mes, por lo que:

$$fc = \frac{1 \text{ ciclo}}{\text{mes}} = \frac{1 \text{ ciclo}}{\text{mes}} * \frac{1 \text{ mes}}{30 \text{ días}} = \frac{1 \text{ ciclo}}{30 \text{ días}} . \tag{3.1}$$

Además, para evitar problemas a la hora de filtrar los datos, le restamos primero la media de la serie a ésta y, tras filtrarla, se la volvemos a sumar.

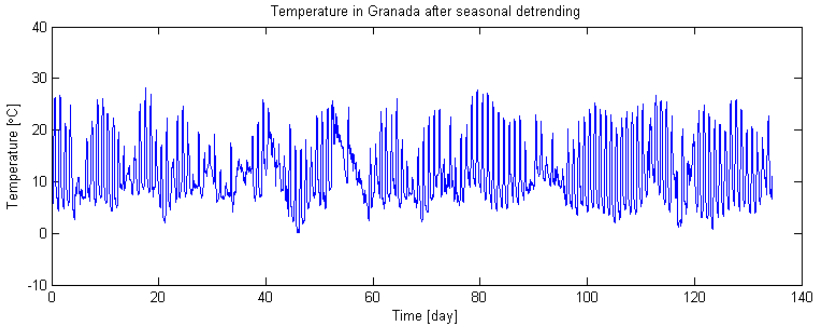


Figura 10: Serie de la temperatura en Granada tras filtrarla mediante un filtro paso-alta con frecuencia de corte $fc = {}^{1 \text{ ciclo}}/_{30 \text{ días}}$. Esta serie corresponde a la serie de temperaturas sin la tendencia estacional.

El resultado lo podemos ver la Fig. 10, donde se sigue observando la variación diaria de la temperatura pero ya no se observan tendencias al alza o la baja dependiendo del momento del año. Hemos de decir que este resultado, aunque esperado, nos ha impresionado. Podemos que hemos eliminado de manera efectiva la tendencia estacional de la temperatura.

Ahora vamos a hacer algo parecido pero vamos a eliminar la fluctuación diaria de la temperatura. De esta forma, esperamos que la señal resultante corresponda a la tendencia estacional.

El código correspondiente lo encontramos en A.2 entre las líneas 89 y 118. De forma análoga a lo que hicimos antes (Ec. (3.1)), escogemos ahora la frecuencia de corte siguiente

$$fc = \frac{1 \text{ ciclo}}{\text{semana}} = \frac{1 \text{ ciclo}}{\text{semana}} * \frac{1 \text{ semana}}{7 \text{ días}} = \frac{1 \text{ ciclo}}{7 \text{ días}}, \tag{3.2}$$

para filtrar aquellas frecuencias mayores que las correspondientes a un periodo de una semana.

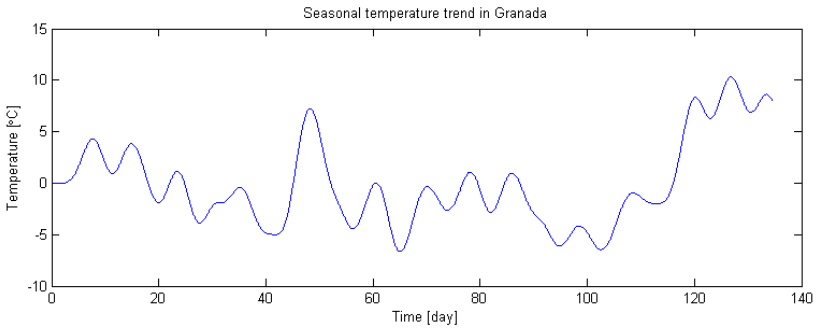


Figura 11: Tendencia estacional de la temperatura en Granada tras filtrarla mediante un filtro paso-baja con frecuencia de corte $^{1 \text{ ciclo}}/_{7 \text{ días}}$.

El resultado lo podemos ver en la Fig. 11. Se puede observar con total claridad lo que comentábamos al inicio del ejercicio.

Pasamos ahora a calcular la envolvente de la fluctuación diaria de la temperatura. Esto corresponde con el código entre las líneas 120 y 204 de A.2.

Si antes para eliminar la fluctuación diaria hemos utilizado un filtro paso-baja con la frecuencia de corte $^{1 \text{ ciclo}}/_{7 \text{ días}}$, para quedarnos con esta fluctuación utilizaremos un filtro con la misma frecuencia de corte pero de tipo paso-alta. Una vez que aplicamos el filtro, y antes de sumar de nuevo la media, calculamos la función analítica y su módulo de la misma forma que lo hicimos en la Subsec. 2.2.

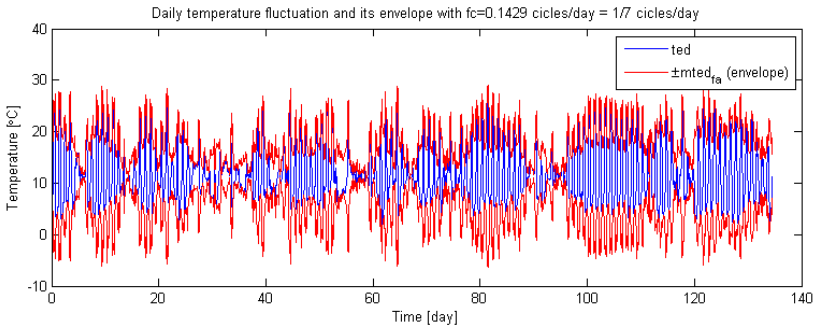


Figura 12: Fluctuación diaria de la temperatura en Granada y su envolvente tras aplicar un filtro paso-alta con frecuencia de corte $^{1 \text{ ciclo}}/_{7 \text{ días}}$.

El resultado lo podemos observar en la figura Fig. 12. Sin embargo la envolvente tiene muchos picos, lo que hace muy difícil apreciar verdaderamente su forma. Para tratar de verla de forma algo más clara, vamos a realizarle un suavizado utilizando para ello un filtro de Butterworth paso-baja. El código para realizar este suavizado se encuentra en el código (A.2) entre las líneas 147 y 190 y es opcional realizarlo, tan solo hay que modificar la variable *quiere_suavizar* para aplicarlo o no aplicarlo.

Tras probar con distintas frecuencias de corte para este filtro, hemos decidido utilizar una frecuencia de corte $fc_s = {}^{1 \text{ ciclo}}/_{1.5 \text{ días}}$.

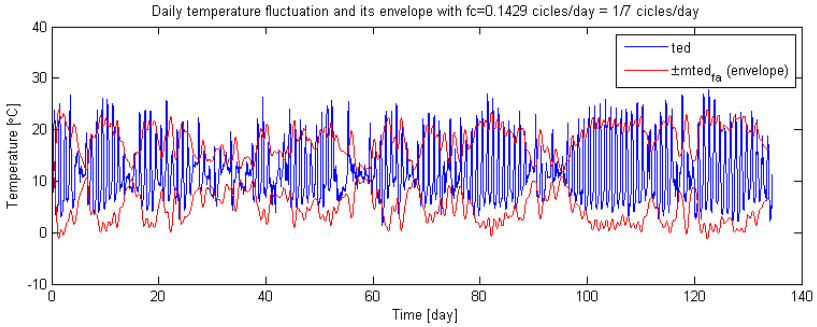


Figura 13: Fluctuación diaria de la temperatura en Granada y su envolvente suavizada tras aplicar un filtro paso-alta con frecuencia de corte $fc = 1\ \text{ciclo}/7\ \text{días}$. La frecuencia de corte utilizada en el suavizado de la envolvente ha sido $fc_s = 1\ \text{ciclo}/1.5\ \text{días}$.

El resultado del suavizado lo podemos ver en la Fig. 13. Ahora se aprecia mucho mejor la forma de la envolvente. Sin embargo, y como podemos ver en la figura, el suavizado tiene un defecto y es que disminuye la amplitud de la señal. Es por ello que hay que ser cuidadoso con la frecuencia fc_s que escojamos, ya que a medida que sea más pequeña el suavizado será mayor pero la envolvente se parecerá, cada vez menos, a la fluctuación diaria de la temperatura.

Además de aplicarle este suavizado a la envolvente, también hemos tratado de corregirle el desfase introducido por el filtro. Para ello, hemos calculado la correlación cruzada entre la envolvente tras aplicar el filtro y la fluctuación de temperaturas. Hemos calculado entonces el desfase como el «lag» para el que la correlación cruzada es máxima. Dependiendo de si la señal suavizada se adelanta o se atrasa, corregimos el desfase. Esto corresponde con las líneas 161 a 190 de A.2 y el resultado está ya incorporado en la Fig. 13.

Pasamos ahora a calcular la transformada de Fourier de la fluctuación diaria de temperaturas, lo que corresponde con las líneas 206 a 217 de A.2. Para realizar el dicho cálculo utilizamos la función **trfour** la cual, a diferencia de la función **tfour**, no aplica la ventana de Hamming.

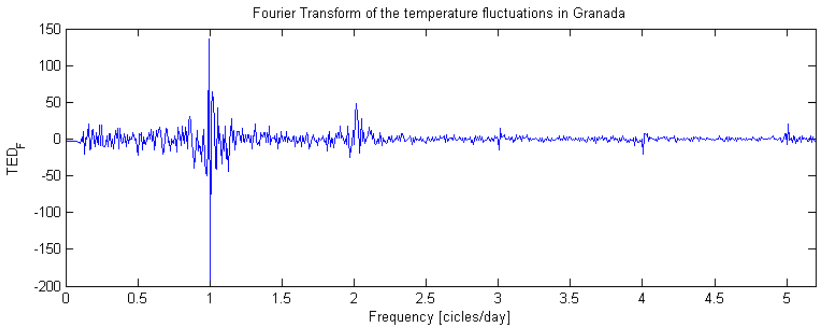


Figura 14: Transformada de Fourier de la fluctuación diaria de temperaturas en Granada.

La transformada de Fourier no es más que una operación matemática que utilizamos en el campo del análisis de señales para cambiar una función del dominio del tiempo al dominio de la frecuencia, y viceversa. Esto nos permite realizar operaciones con la serie en el dominio que más nos convenga.

En el caso que nos atañe, podemos ver en la Fig. 14 la transformada de Fourier de la fluctuación diaria de temperaturas frente a la frecuencia. Vemos que las frecuencias para las que la amplitud es mayor coinciden con múltiplos enteros de la frecuencia $f = 1\ \text{ciclo}/\text{día}$, es

decir, estas especies de «frecuencias de resonancia» son

$$f_{res} = n\frac{1 \text{ ciclo}}{\text{día}} \ , \ n \in \mathbb{N} \ . \tag{3.3}$$

Este fenómeno es algo que tiene cierto sentido si tenemos en cuenta dos cosas. Por un lado, hemos filtrado la serie que estamos utilizando de aquellas frecuencias menores que $^{1 \text{ ciclo}}/_{7 \text{ días}}$. Y, por otro, que existe una periodicidad diaria obvia en las magnitudes físicas atmosféricas, y la temperatura no es una excepción. Es por esto que la frecuencia de mayor amplitud es la de $^{1 \text{ ciclo}}/_{1 \text{ día}}$ y que corresponde con este ciclo diario.

Por último, y para acabar el ejercicio, vamos a realizar una predicción lineal de la temperatura con un modelo autorregresivo. El código escrito para hacerlo lo encontramos entre las líneas 219 y 252 de A.2.

Para crear el modelo autorregresivo utilizamos la función **lpc** [MathWorks 2021e]. Esta función crea un filtro de predicción lineal y toma como primer argumento la señal a predecir y como segundo argumento el orden del predictor. En nuestro caso, y tras distintas pruebas, hemos escogido un predictor de orden 20 para predecir la serie de temperaturas. Puesto que el filtro que la función **lpc** crea es de tipo FIR, al aplicarlo debemos de recordar que tan solo tenemos un coeficiente a y que éste vale 1.

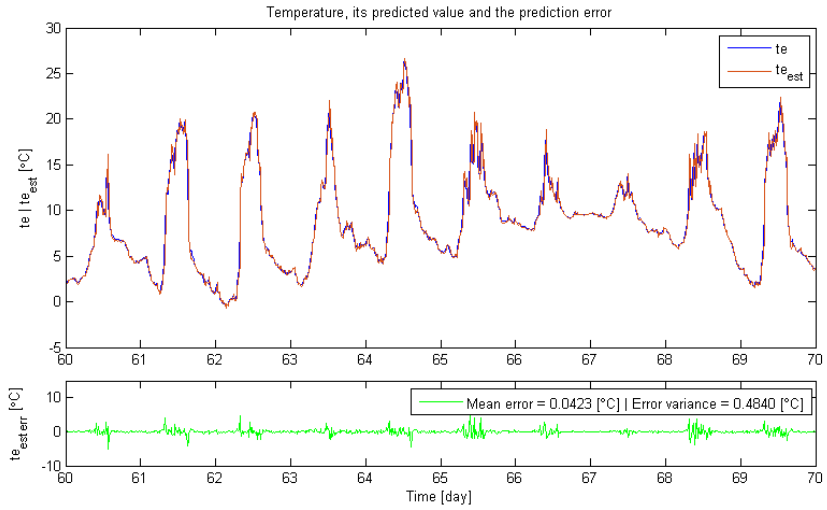


Figura 15: Temperatura predicha en Granada mediante un modelo autorregresivo de predicción lineal de orden 20 (arriba) y su error (abajo).

En la Fig. 15 podemos ver la serie de temperaturas predicha, entre el día 60 y 70, con el modelo autorregresivo lineal que hemos creado. Podemos ver que el error es mayor en los picos, como es normal. Aún así, para un filtro de orden 20 el error medio cometido es menor que $0.05°C$, y la varianza de éste es también menor que $0.5°C$.

Referencias

MathWorks (2021a). *butter - MATLAB documentation*. URL: https://www.mathworks.com/help/signal/ref/butter.html (visitado 01-01-2022).

– (2021b). *decimate - MATLAB documentation*. URL: https://www.mathworks.com/help/signal/ref/decimate.html (visitado 01-01-2022).

– (2021c). *fir1 - MATLAB documentation*. URL: https://www.mathworks.com/help/signal/ref/fir1.html (visitado 01-01-2022).

MathWorks (2021d). *hilbert - MATLAB documentation*. URL: https://www.mathworks.com/help/signal/ref/hilbert.html (visitado 01-01-2022).
– (2021e). *lpc - MATLAB documentation*. URL: https://www.mathworks.com/help/signal/ref/lpc.html (visitado 01-01-2022).
– (2021f). *spectrogram - MATLAB documentation*. URL: https://www.mathworks.com/help/signal/ref/spectrogram.html (visitado 01-01-2022).

A Scripts

En este apéndice mostramos los dos scripts de MATLAB escritos para la resolución de los ejercicios planteados.

A.1 ejercicio_1.m

Script en lenguaje MATLAB para la resolución del Ejercicio 1 (Sec. 2).

```
% ============================================================
% Ejercicio de Análisis y Tratamiento de Datos en Geofísica y
% Meteorología. Curso 2021-22.
% ---------------------------
% Autor: Calzada Chávez, Alberto.
% Email: alcalzada95@correo.ugr.es
% ============================================================

% Ejercicio 1: Análisis de señales sísmicas
% =========================================
%% Ejercicio 1.1: Análisis de la señal de un vibrador sísmico
% ============================================================
clear, clc

% Definimos el parámetro 'Position' de las figuras
pos = [200 200 900 300];

% Cargamos los datos
load vibroseis.mat;

%% 1.1.1 Representar xw y yr en función del tiempo:
% ------------------------------------------------
% Creamos vectores tiempo
tx = (1:length(xw))*dt;
ty = (1:length(yr))*dt;

% Representamos xw
figure('Position', pos)
plot(tx,xw,'b')
ylim([-1.4 1.4])
xlabel('Time [s]')
ylabel('xw')
title('xw signal')
xlim([0 10])
```

```
35
36  % Representamos yr
37  figure('Position', pos)
38  plot(ty,yr,'r')
39  xlabel('Time [s]')
40  ylabel('yr')
41  title('yr signal')
42
43  %% 1.1.2 Trans. Fourier de xw. Representar módulo en función de la
44  %  frecuencia:
45  % ------------------------------------------------------------------
46  % Calculamos Transformada de Fourier con trfour
47  [f,XW] = trfour(xw,length(xw),dt);
48
49  % Calculamos su módulo
50  mXW = abs(XW);
51
52  % Representamos el módulo de XW en función de la frecuencia
53  figure('Position',[200 200 600 300])
54  plot(f,mXW,'b')
55  xlim([0 100])
56  xlabel('Frequency [Hz]')
57  ylabel('abs(XW)')
58  title('Module of the F.T. of xw')
59
60  %% 1.1.3 Representación del espectrograma de xw:
61  % ------------------------------------------------
62  % Representamos el espectrograma
63  figure('Position',[200 200 800 400]);
64  spectrogram(xw,1024,900,1024,1/dt,'yaxis');   % Δf = 1/1024dt = 0.98Hz
65  cb = colorbar;
66  ylabel(cb,'Power/Frequency [dB/Hz]')
67  title('xw spectrogram')
68  colormap(parula)
69
70  %% 1.1.4 Cálculo de la correlación entre yr y xw:
71  % ------------------------------------------------
72  % Calculamos la correlación cruzada
73  [cc,lags]=xcorr(yr,xw);
74
75  % Representamos
76  figure('Position',[200 200 800 400])
77  plot(lags,cc,'b')
78  xlabel('Time [s]')
79  ylabel('Cross-correlation')
80  title('Cross-correlation between yr and xw')
81  xlim([0 4e3])
82  ylim([-5500 10000])
83
84  % Tiempos de viaje de las reflexiones:
85  % t1 = 1000s
86  % t2 = 1500s
87  % t3 = 2300s
```

```
% t4 = 3000s

%% Cerramos todas las figuras
close all

%% Ejercicio 1.2: Análisis sismograma vertical del terremoto de Hokaido
% ========================================================================
clear, clc

% Definimos el parámetro 'Position' de las figuras
pos = [200 200 900 300];

% Cargamos los datos
load japon.mat;

% Convertimos el número de cuentas a velocidad del suelo
f_conv = 1500;
wx = wx/f_conv;    % μm/s

% Convertimos minutos a segundos
t = t*60;
dt = t(2)-t(1);

% Representamos wx
figure('Position', pos)
plot(t,wx)
xlabel('Time [s]')
ylabel('wx [\mum/s]')
title('Vertical ground velocity - Hokaido')

%% 1.2.1 Estimación el rango de periodos del tren de ondas superficiales
%  y diseño de un filtro FIR paso-baja
% ------------------------------------------------------------------------
% Transformada de Fourier
[f,WX] = trfour(wx,length(wx),dt);

% Calculamos el módulo de la TF de wx
mWX = abs(WX);

% Represtamos el módulo de la TF frente a la frecuencia
figure('Position', pos)
plot(f,mWX)
xlabel('Frequency [Hz]')
ylabel('abs(WX)')
title('Module of F.T. of the vertical ground velocity - Hokaido')
xlim([0,0.0012])

% Estimación periodos (y frecuencias asociadas) de ondas superficiales
% a partir de la figura de wx:
% Periodo 1: t_2 - t_1 = 1.534*10^5 - 1.510*10^5  -> T_1 = 2.4*10^3 s  -> f_1 = 4.1667*10^-4 Hz
% Periodo 2: t_2 - t_1 = 1.818*10^5 - 1.806*10^5  -> T_2 = 1.2*10^3 s  -> f_2 = 8.3333*10^-4 Hz

% Frecuencia de corte estimada a partir del periodo estimado
```

```
fc = 0.0005;      % frecuencia de corte
fny = 1/dt/2;     % frecuencia de Nyquist (=fs/2)

% Un periodo de la frecuencia de corte corresponde a un periodo
% de 1/0.0005 s = 2e4 s, lo que corresponde a 2e4/dt = 1.6667e3 puntos.
% Tenemos que coger un filtro de mayor longitud, por ejemplo de 3e3 puntos.
% ----
% Un periodo de la frecuencia de corte corresponde a un periodo
% de 1/0.0009 s = 1.1111e3 s, lo que corresponde a 1.1111e3/dt = 0.9259e3 puntos.
% Tenemos que coger un filtro de mayor longitud, por ejemplo de 2e3 puntos.
l_filtro = 3e3;

% Creamos el filtro
b = fir1(l_filtro,fc/fny);

% Aplicamos el filtro
wx_f = filter(b,1,wx);

% Representamos la señal y la señal filtrada
figure('Position',pos)
plot(t,wx,'b',t,wx_f,'r')
legend('wx','wx_f')
xlabel('Time [s]')
ylabel('wx | wx_f [\mum/s]')
title(sprintf('wx and wx filtered with fc=%.4f Hz', fc))

% 1.2.2 Representación de la señal filtrada junto son su envolvente
% ------------------------------------------------------------------
% Calculamos la señal analítica
wx_a = hilbert(wx_f);

% Calculamos su envolvente
mwx_a = abs(wx_a);

% Representamos la señal filtrada junto con su envolvente
figure('Position', pos)
plot(t,wx_f,'b')
hold on
plot(t,-1*mwx_a,'r',t,mwx_a,'r','Linewidth',1.5)
hold off
legend('wx_f','\pm mwx_f (envelope)')
xlabel('Time [s]')
ylabel('wx_f | mwx_f [\mum/s]')
title(sprintf('wx_f and its envelope with fc=%.4f Hz', fc))

%% 1.2.3 Diseño filtro Butterworth paso-alta para simular la respuesta
%  sismómetro y representación de la señal filtrada
% ------------------------------------------------------------------------
% Redefinimos la frecuencia de corte
fc = 1/60;  % 1 ciclo/min * 1 min/60s = 1/60 Hz

% Definimos el filtro de Butterworth
[b,a] = butter(4,fc/fny,'high');
```

```
194
195 % Aplicamos el filtro
196 wx_fb = filter(b,a,wx);
197
198 % Representamos la señal filtrada
199 figure('Position', pos)
200 plot(t,wx_fb)
201 xlabel('Time [s]')
202 ylabel('wx_{fb} [\mum/s]')
203 title(sprintf('wx high pass filtered with fc=%.4f Hz', fc))
204 xlim([0 2.24e5])
205
206 %% Cerramos todas las figuras
207 close all
```

A.2 ejercicio_2.m

Script en lenguaje MATLAB para la resolución del Ejercicio 2 (Sec. 3).

```
1  % ============================================================
2  % Ejercicio de Análisis y Tratamiento de Datos en Geofísica y
3  % Meteorología. Curso 2021-22.
4  % ----------------------------
5  % Autor: Calzada Chávez, Alberto.
6  % Email: alcalzada95@correo.ugr.es
7  % ============================================================
8
9  %% Ejercicio 2: Análisis de la señal meteorológica
10 % ------------------------------------------------
11 clear, clc
12
13 % Definimos el parámetro 'Position' de las figuras
14 pos = [200 200 900 300];
15
16 % Cargamos los datos
17 load meteo1.mat;
18
19 % Definimos vector de tiempo tipo string
20 td_str = datestr(td+ti);
21
22 % Calculamos el intervalo de muestreo
23 dt = td(2)-td(1);
24
25 %% 2.1 Representar la presión y la temperatura
26 % -------------------------------------------
27 % Representamos la presión (izqda.) y la temperatura (dcha.)
28 figure('Position',pos)
29 plot(td,p)
30 xlabel('Time [day]')
31 ylabel('Pressure [mbar]')
32 title('Pressure in Granada')
```

```
33
34  % Representamos la temperatura
35  figure('Position',pos)
36  plot(td,te)
37  xlabel('Time [day]')
38  ylabel('Temperature [\circC]')
39  title('Temperature in Granada')
40
41  %% 2.2 Diezmar señal a un periodo de muestreo de 1 hora
42  % -----------------------------------------------------
43  % (Entiendo que se refiere a la temperatura)
44  % Diezmamos la temperatura a intervalos de 1 hora. Puesto que la tenemos en
45  % intervalos de 15min -> submuestreamos a 1/4
46  ted = decimate(te,4);    % Submuestreo temperatura
47  tdd = downsample(td,4);  % Submuestreo tiempo
48
49  % Calculamos el nuevo intervalo de muestreo
50  dtd = tdd(2)-tdd(1);
51
52  figure('Position',pos)
53  plot(tdd,ted)
54  xlabel('Time [day]')
55  ylabel('Temperature [\circC]')
56  title('Temperature in Granada')
57
58  %% 2.3 Filtrado paso-alta de la temperatura para eliminar tendencia
59  %  estacional
60  % ------------------------------------------------------------------
61  % Calculamos y restamos la media
62  mted = mean(ted);
63  ted = ted-mted;
64
65  % Definimos un filtro tipo Butterworth para eliminar tendencia estacional
66  % Queremos filtrar, por ejemplo, frecuencias por debajo de 1 ciclo/mes.
67  % 1ciclo/mes * 1mes/30días = 1/30 ciclos/día
68  fc = 1/30;
69  fny = 1/dtd/2;
70  [b,a] = butter(6,fc/fny,'high');
71
72  % Representamos la respuesta en frecuencia del filtro
73  %figure
74  %freqz(b,a,500,1/dtd)
75
76  % Aplicamos el filtro y le sumamos la media a la señal filtrada y a la
77  % original
78  ted_f = filter(b,a,ted)+mted;
79  ted = ted+mted;
80
81  % Representamos la señal filtrada
82  figure('Position',pos)
83  plot(tdd,ted_f)
84  ylim([-10 40])
85  xlabel('Time [day]')
```

```
ylabel('Temperature [\circC]')
title('Temperature in Granada after seasonal detrending')

%% 2.4 Filtrado paso baja de la temperatura para eliminar la fluctuación
%  diaria
% ------------------------------------------------------------------------
% Calculamos y restamos la media
mted = mean(ted);
ted = ted-mted;

% Definimos un filtro tipo Butterworth para eliminar fluctuación diaria
% Queremos filtrar, por ejemplo, frecuencias por encima de 1 ciclo/semana.
% 1ciclo/semana * 1semana/7días = 1/7 ciclos/día
fc = 1/7;
fny = 1/dtd/2;
[b,a] = butter(6,fc/fny,'low');

% Representamos la respuesta en frecuencia del filtro
%figure
%freqz(b,a,500,1/dtd)

% Aplicamos el filtro y le sumamos la media a la señal filtrada y a la
% original
ted_ff = filter(b,a,ted)+mted;
ted = ted+mted;

% Representamos la señal filtrada
figure('Position',pos)
plot(tdd,ted_ff)
%ylim([-10 40])
xlabel('Time [day]')
ylabel('Temperature [\circC]')
title('Seasonal temperature trend in Granada')

%% 2.5 Envolvente de la fluctuación diaria de la temperatura
% ---------------------------------------------------------
% Calculamos y restamos la media
mted = mean(ted);
ted = ted-mted;

% Creamos un filtro opuesto al del apartado anterior
fc = 1/7;
fny = 1/dtd/2;
[b,a] = butter(6,fc/fny,'high');

% Representamos la respuesta en frecuencia del filtro
%figure
%freqz(b,a,500,1/dtd)

% Aplicamos el filtro y sumamos la media
ted_f = filter(b,a,ted);   % ted ya tiene su media restada

% Calculamos la señal analítica
```

```
ted_fa = hilbert(ted_f);

% Calculamos su envolvente
mted_fa = abs(ted_fa);

% Sumamos la media a la señal filtrada
ted_f = ted_f+mted;

% SUAVIZADO OPCIONAL ------
% ========================
quiere_suavizar = 0; % 1 si queremos suavizar la envolvente; 0 si no queremos.
if quiere_suavizar
    % Suavizamos la envolvente con filtro butter (no altera la potencia de
    % la señal en la banda de paso). Filtramos frecuencias mayores de
    % 1ciclo/1.5días = 0.667ciclos/día
    fc_s = 1/1.5;
    fny = 1/dtd/2;
    [b,a] = butter(4,fc_s/fny,'low');

    % Aplicamos el filtro butter
    mted_fa = filter(b,a,mted_fa);

    % Estudiamos la correlación entre la fluctuación diaria y su envolvente
    % para corregir el desfase introducido por el filtro
    [cc,lags] = xcorr(ted_f,mted_fa);

    % Print lag de la correlación máxima
    [mccv,mcci] = max(cc);
    lag = lags(mcci);
    sprintf('lag of max cross-correlation: %d puntos',lag)

    % Representamos la correlación
    figure('Position',pos)
    plot(lags,cc)
    xlabel('Lag [points ~ 15min]')
    ylabel('Cross-correlation')
    title('Cross-correlation of ted_f and mted_{fa}')

    % Corregimos el lag de la envolvente
    if lag>0 % si la envolvente suavizada se retrasa
        mted_fa = mted_fa(1+abs(lag):end);
    else     % si la envolvente suavizada se adelanta
        mted_fa = mted_fa(1:end-abs(lag));
    end

    % Definimos el vector de tiempo para la señal filtrada como
    tdd_f = tdd(1:end-abs(lag));
else
    % Definimos el vector de tiempo para la señal filtrada como
    tdd_f = tdd;
end
% ========================

```

```
% Representamos la señal filtrada junto con su envolvente (sumando la
% media de ted, es decir, sumando mted)
figure('Position', pos)
plot(tdd,ted_f,'b')
hold on
plot(tdd_f,-mted_fa+mted,'r','Linewidth',1)
plot(tdd_f, mted_fa+mted,'r','Linewidth',1)
hold off
ylim([-10 40])
legend('ted','\pmmted_{fa} (envelope)')
xlabel('Time [day]')
ylabel('Temperature [\circC]')
title(sprintf('Daily temperature fluctuation and its envelope with fc=%.4f cicles/day = 1/7 cicles/day',fc))

%% 2.6 Transformada de Fourier de la fluctuación diaria de la temperatura
% ------------------------------------------------------------------------
% Calculamos la TF
[f,TED_F] = trfour(ted_f,length(ted_f),dtd);

% Representamos la transformada frente a la frecuencia
figure('Position',pos)
plot(f,TED_F,'b')
xlabel('Frequency [cicles/day]')
ylabel('TED_F')
title('Fourier Transform of the daily temperature fluctuations in Granada')
xlim([0 5.2])

%% 2.7 Predicción lineal de la temperatura con un modelo autorregresivo y
%  representar el error de predicción
% ------------------------------------------------------------------------
% Calculamos los coeficientes del modelo autorregresivo
[a,g] = lpc(te,20);  % probamos predictor con 20 coeficientes

% Realizamos la predicción
te_est = filter([0 -a(2:end)],1,te);

% Calculamos el error de predicción
te_est_err = te-te_est;  % Error=diferencia entre señal real y estimada

% Representamos la temperatura, la temperatura estimada y el error de
% predicción en la misma figura
figure('Position',[100 100 900 500])
subplot(4,1,1:3)
plot(td,te,'b')
hold on
plot(td,te_est,'Color',[0.8500 0.3250 0.0980])
hold off
legend('te','te_{est}')
ylabel('te | te_{est} [\circC]')
title('Temperature, its predicted value and the prediction error')
xlim([60 70])

subplot(4,1,4)
```

```
plot(td,te_est_err,'g')
xlabel('Time [day]')
ylabel('te_{est_ err} [\circC]')
legend_str_1 = sprintf('Mean error = %.4f', mean(te_est_err));
legend_str_2 = sprintf(' Error variance = %.4f', g);
legend(strcat(legend_str, ' [\circC] | ', legend_str_2, ' [\circC]'))
xlim([60 70])
ylim([-10 15])

%% Cerramos todas las figuras
close all
```