Table of Contents

Práctica 8

Teresa González y Miguel Oleo

```
clc
close all
clear
slCharacterEncoding('UTF_8')
```

Introducción

Los filtros adaptativos se encuentran en nuestro dúa a dúa aunque no lo sabiamos. Muchos sistemas de comunicación, como pueden ser los cascos con cancelación de ruido o los telefonos (NCS). Para emplear esta técnica es necesario conseguir información de la señaal a la que queremos quitar el ruido, y el ruido ambiente. Además se necesita un algoritmo de aproximación para ajustar dinámicamente los coeficientes del filtro que vamos a implementar. En esta práctica empleamos LMS.

Diagrama de bloques y datos

Apartado a)

Leemos el fichero y el archivo del audio. Aprovechamos para escucharla y se puede apreciar que hay unas sirenas de fondo (bastante desagradables) y de fondo una voz.

Esta señaal (Dn) está compuesta de un ruido (Rn) más la señal de voz (Sn). El objetivo de esta práctica va a ser minimizar el ruido en la señal con un filtro adaptativo.

```
load('PDS_P8_LE2_G4');
[Dn,fs] = audioread('PDS_P8_LE2_G4_d_n.wav');
sound(Dn,fs);
```

Apartado b)

Leemos el fichero correspondiente a Xn. Este fichero contiene el ruido correlado con la señal Rn.

```
[Xn,fs] = audioread('PDS_P8_LE2_G4_x_n.wav');
sound(Xn,fs);
```

Implementación del algoritmo LMS

Apartado a, b y c)

En este apartado vamos a implementar el algoritmo de LMS para poder ir ajustando Wn y obtener el múnimo ruido posible en En.

Primero inicializamos los vectores a cero (menos los coeficientes del filtro que las dejamos a 1). Es importante insertar M ceros al principio de la señal Xn para poder realizar la convolución completa. Esto se debe a que la primera iteración, cogemos una muestra, a la segunda dos, etc.

Una vez en el bucle realizamos la siguientes operaciones: Filtamos Xn (a través de la convolución). Luego calculamos la diferencia entre Dn (ruido + voz) y Yn (ruido filtrado por Wn), con esto conseguimos reducir el ruido de la salida (En). Por último, calculamos el siguiente coeficiente del filtro aplicando Wn - mu*grad(wn). El gradiente es lo mismo que 2*Rx+W - 2*RdX. En concreto LMS usa una aproximación del gradiente, quedando la fórmula como en la línea 70. Por último reproducimos En y la mejora es notable, el ruido se ha reducido ampliamente, aunque sigue sin ser idéntica a la señal Sn.

```
Wn = ones(M+1,1);
En = zeros(1,length(Xn));
Yn = zeros(1,length(Xn)-M);

Xn = [zeros(M,1); Xn];

for i=1:length(Xn)-M
    Yn(i) = Wn(:,i).'*Xn(i:M+i);
    En(i) = Dn(i)-Yn(i);

    Wn(:,i+1) = Wn(:,i) + 2*mu*En(i)*Xn(i:M+i);

end

sound(En,fs);
```

Apartado c)

Para ver si el valor de mu es óptimo, se puede comparar con el valor máximo permitido que es 1/máximo autovalor de la matriz de autocorrelación. Utilizamos funciones de matlab para sacar el vector y los lags con los que construiremos la matriz Rx, la cual es simétrica y en cuya diagonal principal se encuentra el elemento Rx(m=0). Para comprobar que el resultado de la función toeplitz, evaluamos los valores de Rx en algunos lags y vemos que coinciden en la matriz simétrica resultante. Como se puede ver, el valor de mu (0.0024)<< maxmu (1.2712), por lo que dicho mu que nos dan como dato es coherente a la hora de utilizar LMS.

```
[Rx,m] = xcorr(Xn,'biased');
Rx_matriz = toeplitz(Rx(length(Xn):length(Xn)+M-1));
disp('Rx(m==0)')
Rx(m==0)
disp('Rx(m==1)')
Rx(m==1)
```

```
disp('Rx(m==2)')
Rx(m==2)
autovalores = eig(Rx_matriz);
disp('0<mu<maxMu')
maxMu = 1/max(autovalores)
```

```
Rx(m==0)

ans =

    0.0966

Rx(m==1)

ans =

    0.0928

Rx(m==2)

ans =

    0.0868

0<mu<maxMu

maxMu =

    1.2712
```

Análisis de resultados

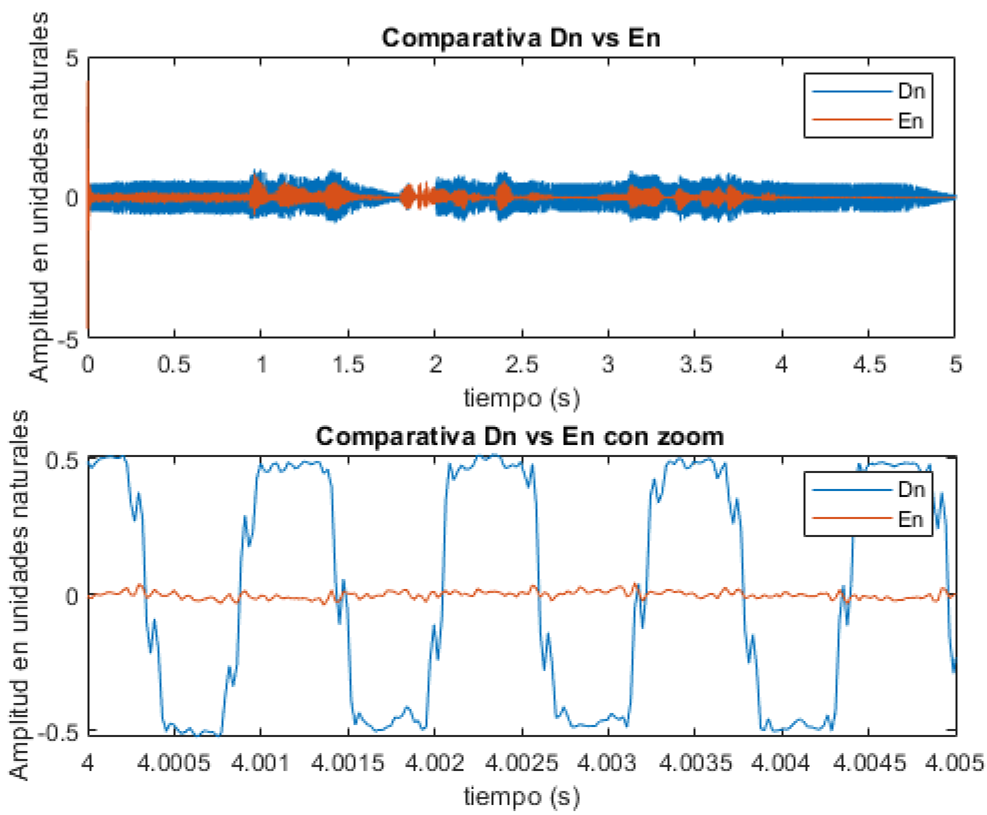
Apartado a)

Como se puede observar en la figura, Dn tiene una amplitud mucho mayor que En; esto es debido a que como hemos explicado, Dn es una señal compuesta por la señal limpia de ruido + ruido que queremos eliminar. Al hacer zoom en la figura, se ve muy claro que en En se elimina una gran parte de dicho ruido presente en Dn, siendo la amplitud resultante considerablemente inferior.

En la imagen con zoom se puede apreciar claramente una especie de señal cuadrada que se corresponde a la sirena y se puede ver como en la señal En están esos tonos practicamente eliminados.

```
dt = 1/fs;
t1= 0:dt:((length(Dn)-1)/fs);
t2= 0:dt:((length(En)-1)/fs);

figure()
subplot(2,1,1)
plot(t1,Dn)
hold on
plot(t2,En)
%xlim([4 4.005])
```

```
xlim([0 5])
title('Comparativa Dn vs En')
xlabel('tiempo (s)');
ylabel('Amplitud en unidades naturales')
legend('Dn','En');

subplot(2,1,2)
plot(t1,Dn)
hold on
plot(t2,En)
xlim([4 4.005])
title('Comparativa Dn vs En con zoom')
xlabel('tiempo (s)');
ylabel('Amplitud en unidades naturales')
legend('Dn','En');
```

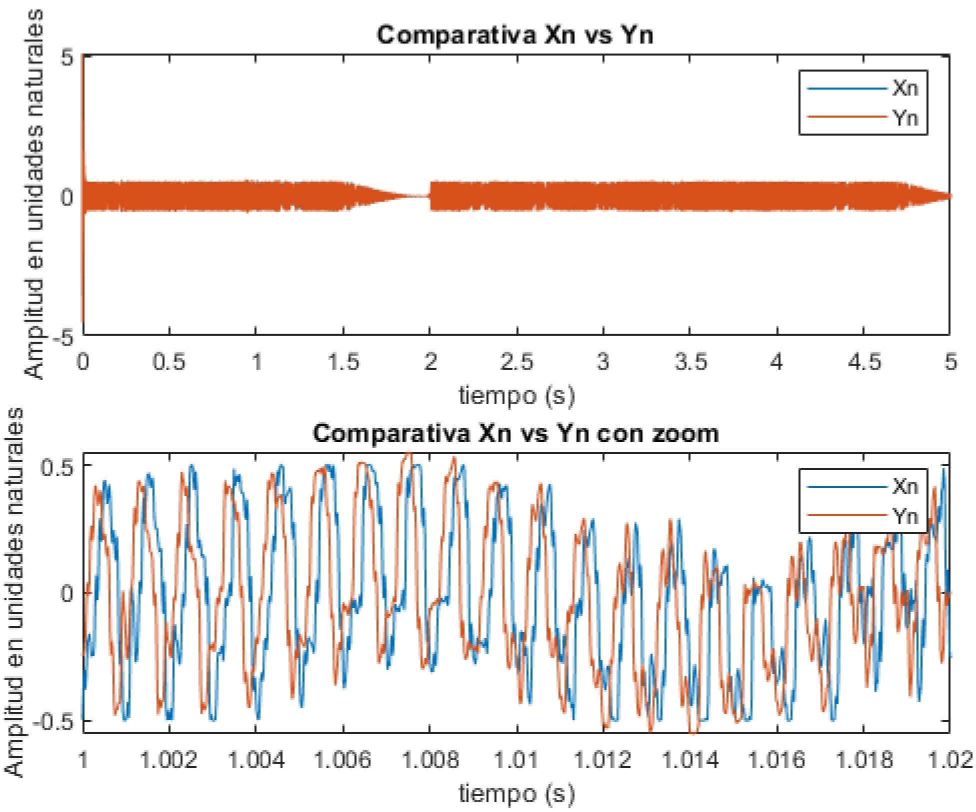


Apartado b)

En la figura se muestran Xn e Yn, las cuales se encuentran desplazadas entre ellas ya que tienen distintos ejes temporales (t1 y t2), porque Xn tiene 10 muestras más debido a los ajustes de tamaño que hemos realizado para algoritmo LMS, aunque 10 muestras de 222267 son despreciables. Se puede ver que las amplitudes son prácticamente idénticas. Por lo tanto, podemos asumir que con el algoritmo LMS se consigue un buen cancelador de ruido, ya que se aisla en Yn todo el ruido que no queremos en la señal de salida.

el la imagen con zoom se puede apreciar como Yn es muy parecida a Xn aunque un poco retardada (por el efecto del filtro).

```
dt = 1/fs;
t1= 0:dt:((length(Xn)-1)/fs);
t2= 0:dt:((length(Yn)-1)/fs);

figure()
subplot(2,1,1)
plot(t1,Xn)
hold on
plot(t2,Yn)
xlim([0 5])
title('Comparativa Xn vs Yn')
xlabel('tiempo (s)');
ylabel('Amplitud en unidades naturales')
legend('Xn','Yn');

subplot(2,1,2)
plot(t1,Xn)
hold on
plot(t2,Yn)
xlim([1 1.02])
title('Comparativa Xn vs Yn con zoom')
xlabel('tiempo (s)');
ylabel('Amplitud en unidades naturales')
legend('Xn','Yn');
```

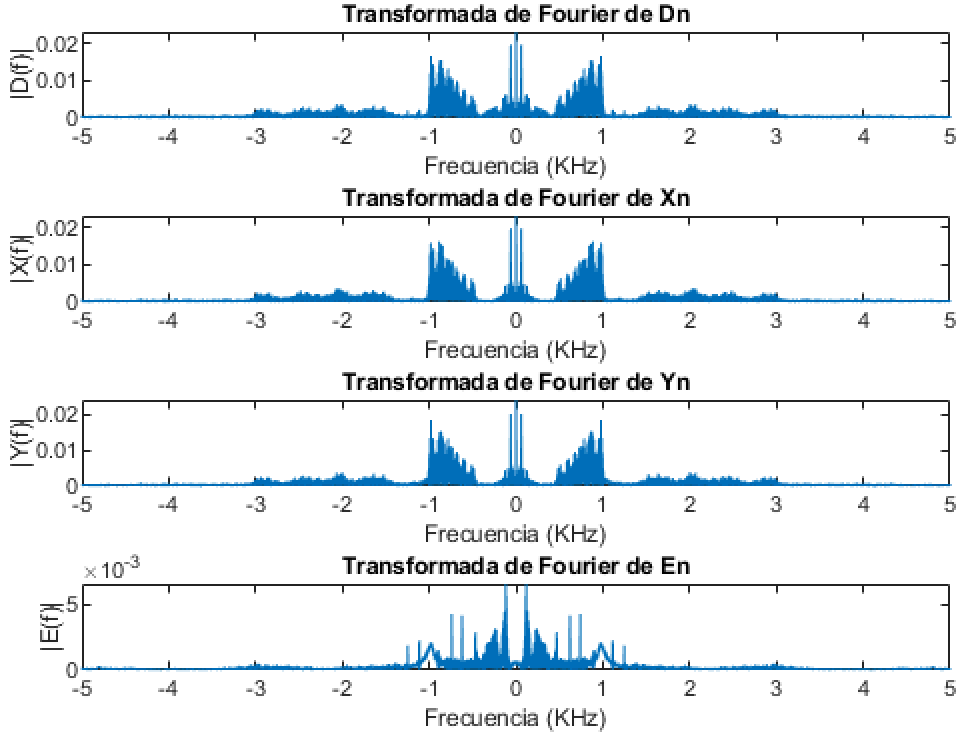


Apartado c)

En esta representación en frecuencia, además de corroborar que Xn e Yn son casi idénticas, se ve como en En hay tonos que desaparecen, y estos son los tonos correspondientes al ruido, que aparecen en Yn (como se ve claramente en torno a 1 Khz. También se muestran las diferencias entre Dn y En, siendo la amplitud menor en esta última, como se ve claramente a mayores frecuencias (en torno a 3 KHZ).

Viendo la transformada de En, se aprecia que, el ruido no se elimina del todo, pero si en gran parte, ya que la potencia de estas frecuencias ha disminuido mucho.

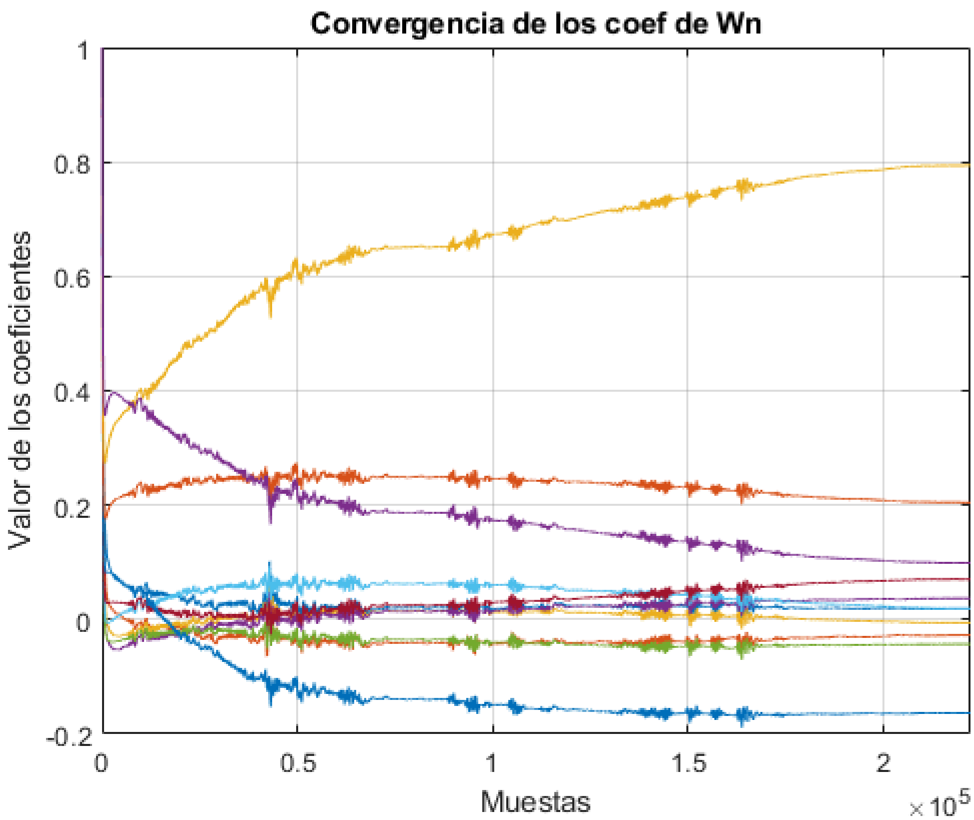
```
Xf = (fft(Xn,length(Xn)))/length(Xn);
f_x = linspace(-fs/2,fs/2,length(Xn));
Df = (fft(Dn,length(Dn)))/length(Dn);
f_d = linspace(-fs/2,fs/2,length(Dn));
Ef = (fft(En,length(En)))/length(En);
f_e = linspace(-fs/2,fs/2,length(En));
Yf = (fft(Yn,length(Yn)))/length(Yn);
f_y = linspace(-fs/2,fs/2,length(Yn));

figure()
subplot(4,1,1)
plot(f_d/1000,fftshift(abs(Df)));
title('Transformada de Fourier de Dn')
xlabel('Frecuencia (KHz)')
ylabel('|D(f)|')
xlim([-5 5])

subplot(4,1,2)
plot(f_x/1000,fftshift(abs(Xf)));
title('Transformada de Fourier de Xn')
xlabel('Frecuencia (KHz)')
ylabel('|X(f)|')
xlim([-5 5])

subplot(4,1,3)
plot(f_y/1000,fftshift(abs(Yf)));
title('Transformada de Fourier de Yn')
xlabel('Frecuencia (KHz)')
ylabel('|Y(f)|')
xlim([-5 5])

subplot(4,1,4)
plot(f_e/1000,fftshift(abs(Ef)));
title('Transformada de Fourier de En')
xlabel('Frecuencia (KHz)')
ylabel('|E(f)|')
xlim([-5 5])
```



Apartado d y e)

Tras haber guardado en el vector Wn los coeficientes del filtro que se van ajustando dinámicamente en cada iteración mediante LMS, representamos dichos coeficientes que corresponden a la evolución de los 10 coeficientes iniciales a lo largo del tiempo. Como se ve en la gráfica, esos coeficientes convergen hacia un valor determinado (tienden a una recta). Se puede ver que los valores hacia los que converge son: 0.8 , 0.2, 0.1, 0.08, 0.04, 0.02, -0.005, -0.03, -0.04, -0.16

```
figure()
plot(Wn.')
title('Convergencia de los coef de Wn')
xlabel('Muestas')
ylabel('Valor de los coeficientes')
xlim([0 222200])
grid on
```



Published with MATLAB® R2020a