INSTITUTO TECNOLÓGICO DE BUENOS AIRES

Trabajo Práctico 2

Señales Aleatorias - 2019

Grupo 2:
Máspero, Martina - 57120
Mestanza, Joaquín - 58288
Müller, Malena - 57057
Nowik, Ariel - 58309
Regueira, Marcelo - 58300

July 2, 2019

EJERCICIO 1

En esta parte del trabajo se estiman algunos parámetros de la secuencia aleatoria X(n) del archivo que se nos ha sido enviado.

0.1 ESTIMACIÓN DE LOS PRIMEROS 128 VALORES DE LA FUNCIÓN DE AUTOCORRELACIÓN UTILIZANDO LOS ESTIMADORES NO POLARIZADO Y POLARIZADO

El estimador no polarizado de la autocorrelación está dado por la expresión (de la página 567 del libro "Random Signals", K. Sam Shanmugan):

$$R_{xx_{np}}(k) = \frac{1}{N-k} \sum_{i=1}^{N-k-1} X(i)X(i+k)$$

Donde N es la cantidad de muestras asignadas, que en este caso son 4096. Los valores que toma k son entre 0 y 127, para obtener los 128 primeros valores solicitados.
El estimador polarizado esta dado por la expresión (de la página 571 del libro):

$$R_{xx_p}(k) = \frac{1}{N} \sum_{i=1}^{N-k-1} X(i)X(i+k)$$

Para los mismos valores de k y N previamente mencionados.
Al normalizar con $R_{xx_{NP}}(0)$ cada uno de los valores estimados de esta manera, se obtienen los coeficientes de autocorrelación total $r_{xx_{NP}}(k)$ (no polarizado) y r_{xx_P} (polarizado), que se pueden ver graficados en las figuras 0.1 y 0.2, respectivamente.
El código implementado para realizar esto, es el siguiente:

```
%% ITEM 1
clear all;
clc;

S = load('Archivo_2.mat');
%whos S
%whos -file Archivo_2.mat

N = 4096;
length = 128;
% Estimador no polarizado y polarizado
RxxNP = zeros(length, 1); %contendr\'a el RxxNP para cada valor de k entre 0 y 127
RxxP = zeros(length, 1); %contendr\'a el RxxP para cada valor de k entre 0 y 127
for k = 0:length-1
    sum = 0;
    for i = 0:N-k-1
        sum = sum + (S.x(i+1) * S.x(i+1+k));
    end
    RxxNP(k+1) = (1/(N-k)) * sum;
    RxxP(k+1) = (1/N) * sum;
end

clearvars sum;
clearvars i;
clearvars k;

% Coeficiente de correlacion
```

```
rxxNP = RxxNP/RxxNP(1);
rxxP = RxxP/RxxP(1);

k = 0:1:length-1;
figure;
plot(k,rxxNP)
figure;
plot(k,rxxP)
```

Listing 1: EJ1.m

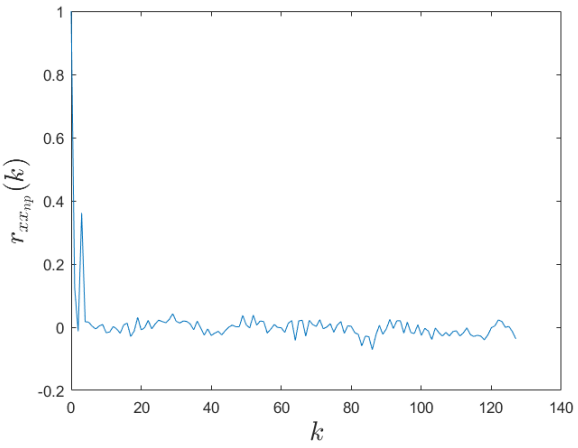

Figure 0.1: $r_{xx}(k)$ a partir del estimador no polarizado.

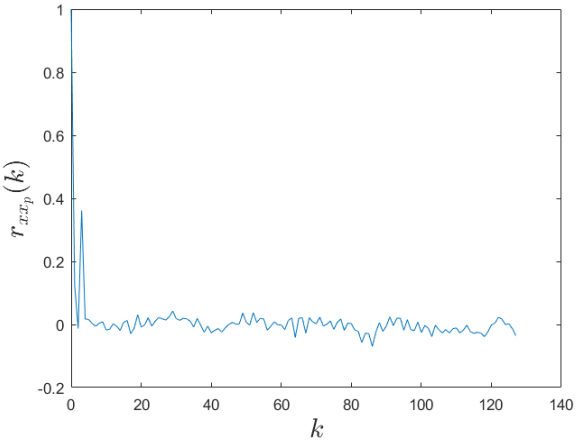

Figure 0.2: $r_{xx}(k)$ a partir del estimador polarizado.

En una primera instancia, se superpuso las dos gráficas para poder denotar diferencias, pero resultaron ser casi iguales. Esto se deriva de que un estimador se encuentra dividido por N, y el otro por $N-k$. Siendo 127 el mayor valor que puede tomar k, sigue siendo pequeño en comparación a N, que es 4096, por lo que no afecta significativamente a los resultados.

0.2 Estimación de los primeros 128 coeficientes de correlación parcial

Para calcular los coeficientes de correlación parcial $\phi_{k,k}$, se arma la matriz de toeplitz correspondiente para cada k, y con ella se resuelve la ecuación de Yule-Walker (de la página 262 del libro):

$$r_{xx} = R \cdot \Phi$$

Donde R es la matriz de coeficientes de autocorrelación (que es una matriz de toeplitz) y Φ el vector de coeficientes $\phi_{p,k}$. De dicho vector, se utiliza solamente el último elemento (donde $p = k$) en cada caso, y se lo almacena en otro vector nuevo para tener todos los coeficientes de correlación parcial. Se graficaron los coeficientes obtenidos tanto para el caso no polarizado como para el polarizado. También resultaron ambos casos valores muy similares, por lo que se separaron las gráficas al igual que antes.
A continuación se presenta el código:

```
%% ITEM 2
length = 127;

%Caso a partir de estimacion de Rxx NO polarizado
partialCorrCoefNP = zeros (1, length) %contendr\'a los coeficientes de
                                %correlaci\'on parcial para k entre 1 y 127
                                %a partir del caso no polarizado.

rxxNPaux = rxxNP
for k = 1:length
    rxxToep = toeplitz(rxxNPaux') % Se genera la matriz de Toeplitz
    rxxMat = rxxToep(1:k,1:k)
    rxxVect = (rxxNPaux(2:k+1));
    corrCoefVect = inv(rxxMat) * rxxVect; % Se resuelve la ecuacion de Yule Walker
    partialCorrCoefNP(k)= corrCoefVect(k);
end

%Caso a partir de estimacion de Rxx polarizado
partialCorrCoefP = zeros(1,length); % contendra los coeficientes de
                                %correlacion parcial para k entre 1 y 127;
                                %a partir del caso polarizado
rxxPaux = rxxP;
for k = 1:length
    rxxToep = toeplitz(rxxPaux'); % Generating Toeplitz Matrix
    rxxMat = rxxToep(1:k,1:k);
    rxxVect = (rxxPaux(2:k+1));
    corrCoefVect = inv(rxxMat) * rxxVect; % Solving Yule Walker Equation
    partialCorrCoefP(k)= corrCoefVect(k);
end
q = 1:length;
figure
stem(q,partialCorrCoefNP)

q = 1:length;
figure
stem(q,partialCorrCoefP)
```

Listing 2: EJ1.m

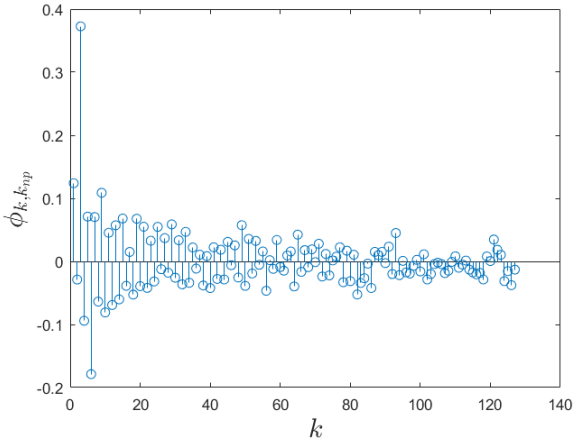

Figure 0.3: Coeficientes de correlación parcial a partir del estimador no polarizado.

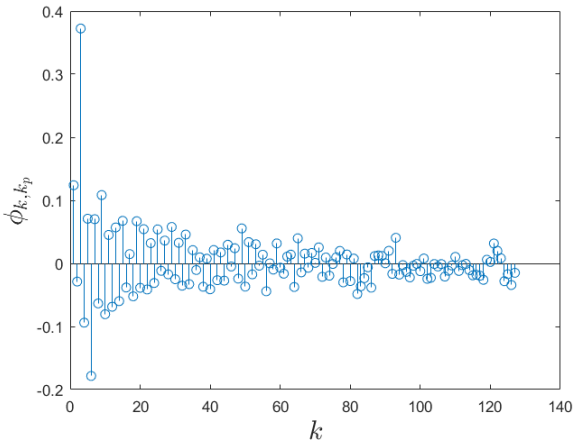

Figure 0.4: Coeficientes de correlación parcial a partir del estimador polarizado.

0.3 Determinación del modelo y orden para ajustar a la secuencia aleatoria $X(n)$

Luego de probar con modelos AR, MA de segundo orden y ARMA(1,1), se determinó que el que mejor ajusta a la secuencia aleatoria $X(n)$ podría ser el AR de orden 2. Teniendo en cuenta que la entrada es una secuencia de ruido blanco y Gaussiano con varianza unitaria, se hallan los parámetros de este modelo. Se planteó solo el caso no polarizado, dado que se obtienen resultados muy similares como ocurrió previamente, por lo que se deja solo uno como caso representativo.

Del vector de los $R_{xx_{np}}(k)$ obtenidos, se toman el $R_{xx_{np}}(0) = 1.3690$, $R_{xx_{np}}(1) = 0.1698$ y $R_{xx_{np}}(2) = -0.0178$.

A partir de la expresión recursiva para la autocorrelación de un modelo AR de orden 2 (de la página 257 del libro):

$$R_{xx}(k) = \phi_{2,1} \cdot R_{xx}(k-1) + \phi_{2,2} \cdot R_{xx}(k-2)$$

Teniendo además en cuenta que $R_{xx}(k) = R_{xx}(-k)$, se plantean dos ecuaciones:

$$\begin{cases} \phi_{2,1} \cdot R_{xx}(0) + \phi_{2,2} \cdot R_{xx}(1) &= R_{xx}(1) \\ \phi_{2,1} \cdot R_{xx}(1) + \phi_{2,2} \cdot R_{xx}(0) &= R_{xx}(2) \end{cases}$$

De donde se obtiene que $\phi_{2,1} = 0.1276$ y $\phi_{2,2} = -0.0288$. Con dichos valores, se calcula analíticamente $R_{xx}(k)$ con la fórmula recursiva previamente planteada y $r_{xx}(k)$, con k entero entre 0 y 127.

El código empleado es el siguiente:

```
%% ITEM 3
% Se lo describe con un modelo AR(2)
% X(n) = phi21*X(n-1) + phi22*X(n-2) + e(n)
phi21 = 0.1276;
phi22 = -0.0288;

%% ITEM 4

Rxx_TNP = zeros(length,1);
Rxx_TNP(1) = RxxNP(1);
Rxx_TNP(2) = RxxNP(2);

for k = 3:length-1
    Rxx_TNP(k) = phi21.*Rxx_TNP(k-1) + phi22.*Rxx_TNP(k-2);
end
rxx_TNP = Rxx_TNP/Rxx_TNP(1);

k = 0:1:length-1;
figure
plot(k,rxx_TNP)

Rxx_TP = zeros(length,1);
Rxx_TP(1) = RxxP(1);
Rxx_TP(2) = RxxP(2);

for k = 3:length-1
    Rxx_TP(k) = phi21.*Rxx_TP(k-1) + phi22.*Rxx_TP(k-2);
end
rxx_TP = Rxx_TP/Rxx_TP(1);

k = 0:1:length-1;
figure
plot(k,rxx_TP)
```

Listing 3: EJ1.m

La gráfica obtenida se muestra a continuación.

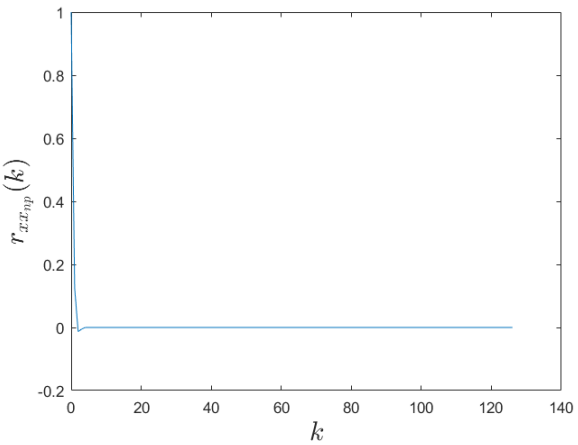

Figure 0.5: $r_{xx}(k)$ a partir del estimador no polarizado, obtenido en forma teórica, en base al modelo AR orden 2.

La curva obtenida se asemeja a la estimada con los valores al principio, y luego se anula prácticamente. Esto es consistente con el rápido decaimiento que posee la curva hallada en la figura 0.1, donde los valores más allá de $k = 10$ se observó que se encuentran todos en el orden de 10^{-2}.

0.4 Estimación de la densidad espectral de potencia de X(n)

En primer lugar, se estima la densidad espectral de potencia a partir de la transformada de Fourier discreta de la autocorrelación no polarizada. Por el mismo motivo mencionado anteriormente, se muestra solamente la resultante del caso no polarizado, dado que la otra es prácticamente similar. Los resultados pueden observarse en la figura 0.6. Se muestra también el código implementado para el cálculo de la transformada.

```
%% ITEM 5
% Por transformada
SxxNP = fft(RxxNP);
mag_SxxNP = abs(SxxNP);
SxxNP(mag_SxxNP<1e-6) = 0;
f = 0:1:length;
figure
plot(f,mag_SxxNP)
```

Listing 4: EJ1.m

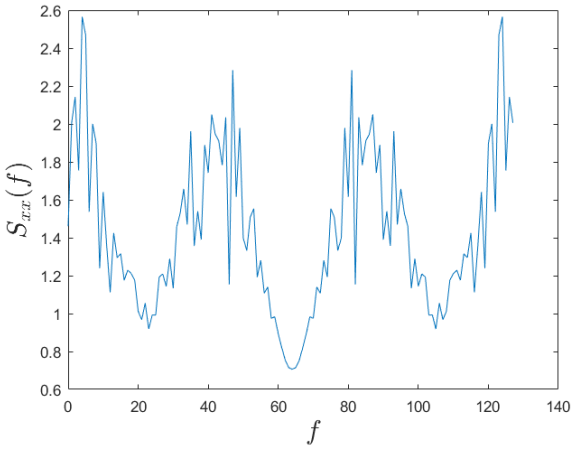

Figure 0.6: Estimación de la densidad espectral de potencia de X(n) a partir de la transformada de Fourier discreta de la estimación de la autocorrelación no polarizada.

Por otro lado, se estima la densidad espectral de potencia a partir de la promediación de periodogramas. Para ello, las muestras iniciales se las divide en 16 grupos de 256 muestras cada uno. En cada grupo, se estiman los primeros 128 valores de la autocorrelación (no polarizado). A cada vector resultante, se le calcula la densidad espectral de potencia, y finalmente se las promedia (página 579 del libro):

$$\overline{S_{xx}}(f) = \frac{1}{n}\sum_{k=1}^{n} S_{xx}(f)_k$$

Donde en este caso $n = 16$. Pueden verse los resultados en la figura 0.7. Se muestra también el código que implementa el procedimiento.

```
% Por periodogramas
Rxx_Vector = zeros(16,128);
% Se divide la entrada en 16 grupos de 256 muestras, calculando
```

```
% a 16 funciones de autocorrelacion sus 128 primeros valores
% para el caso no polarizado
for l = 1:16
    for k = 0:127
        sum = 0;
        for i = 0:256-k-1
            sum = sum + (S.x(256*(l-1)+i+1) * S.x(256*(l-1)+i+1+k));
        end
        Rxx_Vector(l,k+1) = (1/(256-k)) * sum;
    end
end
Sxx_Vector = zeros(128,16);
% Se calcula la densidad espectral de cada uno
for k = 1:16
    Sxx_Vector(:,k) = fft(Rxx_Vector(k,:));
end
Sxx_Vector = Sxx_Vector';
Sxx_Med = zeros(1,128);
% Se estima la densidad promedio
for k = 1:16
    Sxx_Med = Sxx_Med + Sxx_Vector(k,:);
end
Sxx_Med = Sxx_Med/16;
mag_SxxMed = abs(Sxx_Med);
SxxNP(mag_SxxMed<1e-6) = 0;
figure
plot(1:128,mag_SxxMed)
```

Listing 5: EJ1.m

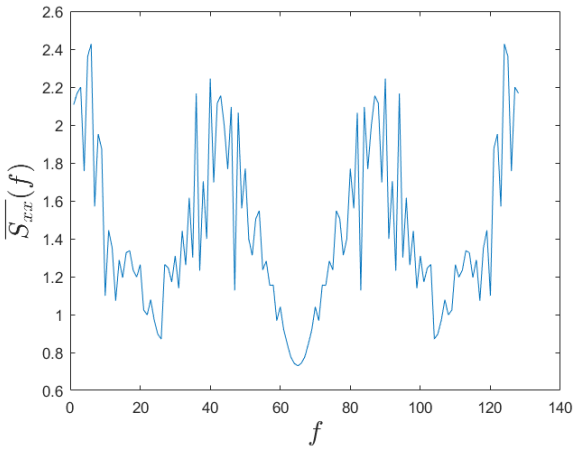

Figure 0.7: Estimación de la densidad espectral de potencia de $X(n)$ a partir de la promediación de periodogramas.

Se observa que las densidades resultantes son bastante similares, con pequeñas diferencias. De esto se desprende que es posible estimar la densidad espectral con grupos de muestras más pequeños, promediando las densidades intermedias. Es decir, sin tomar demasiados datos se logra cometer muy poco error con el estimador promedio.

EJERCICIO 2

Para el segundo ejercicio se pidió calcular Smoothing filters óptimos, aplicados a una señal de voz. Para ello, se utilizó la ecuacion 7.83.b del del libro “Random Signals”, K. Sam Shanmugan, con una pequeña modificación dado que los Smoothing filters tienen respuesta impulsiva no causal. En primer lugar se define nuestra señal como X(n) = S(n) + v(n), donde S(n) es la señal de audio original y v(n) es ruido blanco gaussiano de media nula. La potencia del ruido utilizada es de -30dBw. Se estimaron las autocorrelaciones de X y de S utilizando la ecuación 9.6 del libro:

$$R_{xx}(k) = \frac{1}{N-k} \sum_{i=1}^{N-k-1} X(i)X(i+k)$$

De esta forma, se obtiene luego la respuesta impulsiva y finalmente se realiza la convolución con la señal X, obteniendo el estimador a partir de la ecuación 7.74 del libro.

```
clear;
clc;
%leemos archivo (debe ser mono)
file = 'whereIam8Khz.wav';
info = audioinfo(file);
[data,Fs] = audioread(file);
t = 0:seconds(1/Fs):seconds(info.Duration);
S = compand(data,255,max(data),'mu/compressor'); % esta es la senal
Muestras = Fs*20e-3; %160 si fs 8Khz %fs*20ms = Muestras
Ventanas = cast(floor(length(S)/Muestras),'uint64');
S = S(1:Ventanas*Muestras); %Recortamos la senal en funcion de la cantidad de ventanas
PotRuido = -30; % esta en dB
Noise = wgn(length(S),1,PotRuido);
X_tot = S + Noise;
% La variable size determina la longitud de la respuesta impulsiva
% La longitud seria de 2*size+1
size = 10;
m = -size:size;
shat = double.empty; %en esta variable guardaremos el S estimada
for l=0:Ventanas-1
    td = (1+l*Muestras):(Muestras+l*Muestras);
    Act_X = X_tot(td);
    Act_S = S(td);
    Rxx = get_Rxx(Act_X, Muestras,2*size+1);
    Rss = get_Rxx(Act_S, Muestras,size+1);
    RssMat = 1:size*2+1;
    for i=1:size*2+1
        index =  abs(size+1-i)+1;
        RssMat(i) = Rss( index );
    end
    R = toeplitz(Rxx);
    h = R\RssMat';
    Shift_H = circshift(h,size); % hacemos un shifteo a la
    % respuesta impulsiva dado que no es causal
    estimacion = conv(Act_X,Shift_H,'same');
    shat = [shat estimacion'];
end
subplot(3,1,1);
plot(t(1:length(S)),S)
title('Original')
```

```
subplot(3,1,2);
plot(t(1:length(X_tot)),X_tot)
title('Con ruido')
subplot(3,1,3);
plot(t(1:length(shat)),shat)
title('Estimacion')
player = audioplayer(shat, Fs);
play(player);
%para parar el audio stop(player);
```

Listing 6: Ejercicio2.m

```
function [ Rxx ] = getRxx(x,N,k)
%Esta funcion es para obtener el estimador de la autocorrelacion de X
%Esta en la pagina 567 del Shanmugan
% x es el vector con las muestras
% N es la cantidad de muestras
% k es hasta donde calcular Rxx
Rxx = double.empty ;
for j = 0:k-1
    sumatoria = 0;
    for i = 0:N-j-1
        sumatoria = sumatoria + x(i+j+1)*x(i+1); % calculo la sumatoria
    end
    Rxx(j+1) = (1/(N-j))*sumatoria; % los sumo y divido por el k correspondiente
end
end
```

Listing 7: getRxx.m

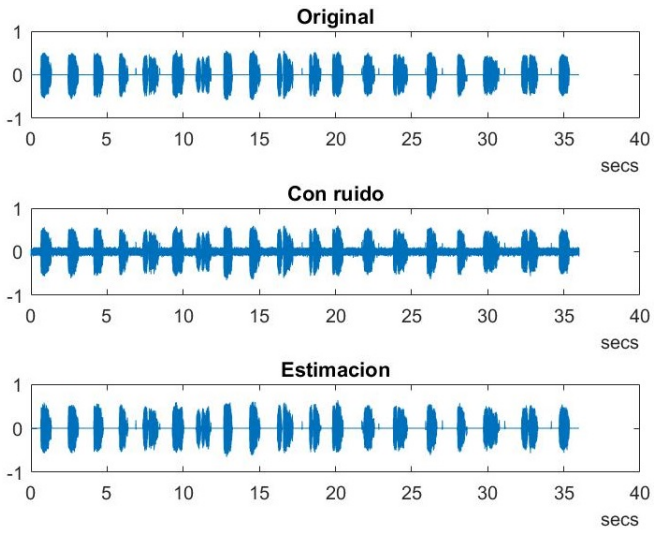

Figure 0.8: Estimación de la señal de prueba 'whereIam8Khz.wav'

Se puede ver en el primer gráfico de la figura la señal original, en la segunda se puede observar como se distorsiona la señal por efecto del ruido y en la tercera la estimación dada por el filtro obtenido. Se nota una gran similitud con la señal original.

0.4.1 Conclusiones

Durante la realizacion de este trabajo se pudo verificar el correcto funcionamiento de los "smoothing filters". Se logró a traves de la implementacion de uno obtener una estimacion muy similar a la señal original.