

Instituto Tecnológico de Buenos Aires

22.67 SEÑALES ALEATORIAS

Trabajo práctico N°3

Grupo 1:

LAMBERTUCCI, Guido Enrique	58009
LONDERO BONAPARTE, Tomás Guillermo	58150
MUSICH, Francisco	58124

Profesor

HIRCHOREN, Gustavo Abraham

Presentado: ??/??/21

Índice

0.1. Introducción	2
0.2. Estimación de la Autocorrelación	2
0.3. Coeficientes de correlación parcial	2
0.4. Modelo del proceso	3
0.5. Filtro de Kalman	4
0.5.1. Modelo en variables de estado	4
0.5.2. Implementacion del filtro recursivo	4
0.6. Análisis de resultados	6
0.7. Código implementado	6

0.1. Introducción

Se analiza una secuencia $X(n)$, estimando y calculando parámetros de interés, como lo son la autocorrelación, los coeficientes de correlación parcial y la densidad espectral de potencia.

0.2. Estimación de la Autocorrelación

Se estiman la autocorrelación mediante el uso de los primeros 128 elementos de la secuencia brindada. Para ello, se vale los estimadores polarizados (R_p) y no polarizados (R_{np}) de dicho parámetro. Estas funciones son las empleadas para estimar otras funciones mediante información digitalizada.

$$\begin{aligned} R_p(k) &= \frac{1}{N} \sum_{i=0}^{N-k-1} X(i)X(i+k) \\ R_{np}(k) &= \frac{1}{N-k} \sum_{i=0}^{N-k-1} X(i)X(i+k) \end{aligned} \quad (1)$$

En ellas se observan los parámetros N , es decir, el largo de $X(n)$, y k , variable que puede tomar los valores $0, 1, \dots, 127$. Mediante el uso de estos estimadores, se normaliza para poder obtener los coeficientes de autocorrelación r_{XXp} y r_{XXnp} .



Figura 1: Grafica de los coeficientes de autocorrelación total estimados.

Se puede observar en la Figura (1) como ambas curvas se encuentran solapadas, haciendo que sea prácticamente imposible distinguirlas. Esto se debe a que existe una relación entre cada estimador, siendo esta

$$R_p(k) = \frac{N-k}{N} R_{np}(k)$$

Ya que, para el caso del vector analizado, se da la condición de que $N = 4096$ y además $N \gg k_{max} = 127$, siendo entonces

$$R_p(k) \approx R_{np}(k)$$

0.3. Coeficientes de correlación parcial

Con los datos ya extraídos y mediante la resolución de la ecuación de Yule-Walker, fue posible obtener los coeficientes deseados. Esto se realizó con los coeficientes totales obtenidos a través de las estimaciones polarizada y no polarizada.



Figura 2: Grafica de los coeficientes de autocorrelación parcial obtenidos.

En la Figura (2) se obtuvo nuevamente una diferencia entre ambas curvas, la cual no es significativa.

0.4. Modelo del proceso

Se procede a determinar que tipo de modelo utilizar para el proceso analizado. Observando la Figura (1), se denota que $r_{XX}(1)$ y $r_{XX}(2)$ son valores distintos de 0 ($-0,603$ y $0,099$ para ambas aproximaciones), mientras que los valores siguientes, si bien no son exactamente 0, son todos menores en modulo a $0,03$, lo que permite aproximarlos a 0. Además, observando la Figura (2), se puede afirmar que los ϕ_{kk} presentan un comportamiento exponencial. Es por ello que se determina que el proceso es un **MA(2)** (**ARMA(0,2)**).

Para el calculo de los θ , se utilizaron las ecuaciones

$$r_{XX}(1) = \frac{R_{XX}(1)}{\sigma_X^2} = \frac{\theta_{2,1} + \theta_{2,1}\theta_{2,2}}{1 + \theta_{2,1}^2 + \theta_{2,2}^2} \quad (2)$$

$$r_{XX}(2) = \frac{\theta_{2,2}}{1 + \theta_{2,1}^2 + \theta_{2,2}^2} \quad (3)$$

Resolviendo dicho sistema, se obtienen los siguientes valores:

$$\begin{aligned} \theta_{2,1} &= -1,280 \\ \theta_{2,2} &= 0,268 \end{aligned} \quad (4)$$

Con lo ya dicho, se procede a estimar los parámetros del proceso y compararlos con los ya obtenidos.



Figura 3: Comparación de los coeficientes de autocorrelación.



Figura 4: Comparación de los coeficientes de autocorrelación normalizados.

0.5. Filtro de Kalman

0.5.1. Modelo en variables de estado

0.5.2. Implementacion del filtro recursivo

A continuación, se estima la la densidad espectral de potencia del vector $X(n)$. Para ello, se emplean dos técnicas distintas. La primera consiste en el uso de la transformada de Fourier de la estimación realizada de las funciones de

autocorrelación.



Figura 5: Periodigramas obtenidos a partir de las estimaciones de R_{XX} .

Como era de esperarse, la diferencia entre el gráfico obtenido a través de la estimación polarizada no difiere tanto de la no polarizada.

La segunda técnica consta de la promediación de periodigramas. Para esto se partió el vector original en 16 grupos de 256 elementos, en cada grupo se calculó los primeros 128 valores de la autocorrelación con el estimador no polarizado, luego a cada vector se le calcula la densidad espectral de potencia y finalmente se las promedia.¹

¹Se utilizó la formula 9.24 del libro



Figura 6: Estimación de la densidad espectral de potencia mediante el uso de promediación de periodogramas.

Finalmente, a modo comparativo, se ilustran las estimaciones obtenidas superpuestas:



Figura 7: Estimaciones de potencia.

Se puede apreciar que son muy similares tanto la promediación de periodogramas con la transformada de la estimación de la función de autocorrelación.

0.6. Análisis de resultados

El mierdas comparativo con gráficos

0.7. Código implementado

El mierda.m

■ Main.m:

```

1  clc
2  clear
3  close all
4  fprintf('Welcome to GT1 matlab script for Kalman filtering\r\n');
5
6  %Constants usefull to alter the behaviour of the script
7  kmax=9;%Value of P
8  SAMPLES=100;
9  LOGSPACELEN = 10;
10
11 %Buffers for variables.
12 R_vars_buffer=logspace(-2,2,LOGSPACELEN);%diferent values for the variance of
    the measurment
13 error_improvement = zeros(1,LOGSPACELEN);
14 mse_measure_b=zeros(1,kmax);
15 mse_filter_b=zeros(1,kmax);
16
17 %actual script
18 x=load('h06g1.dat');%loads the sample vector
19
20 xs=x(1:SAMPLES);%gets a subarray to make it faster to process, if desired SAMPLES
    can be changed to a a value between 1 and size(x)
21
22 for j = 1:LOGSPACELEN%For each Variance
23
24     for i = 1:kmax%For every p
25
26         Rxxnp = Rnp(x,i+1);%Estimate the Autocorrelation function using a non
            polarized estimator
27         rxxnp = Rxxnp./Rxxnp(1); %Normalize it
28
29         if(j==1 && i==kmax)
30             [phiknp, phiv, phiVarn] = cpar(rxxnp, i+1,1);%Obtain the partial
                correlation coeffients by solving the Yule Walker equation.
31         else
32             [phiknp, phiv, phiVarn] = cpar(rxxnp, i+1,0);%Obtain the partial
                correlation coeffients by solving the Yule Walker equation.
33         end
34
35         %Kalman matrices.
36         PHI = [phiv'; eye(i-1) zeros(i-1,1)];%Phi matrix for the Kalman filter,
            also known as the State transition model
37         %The first row is the AR coefficients, the other is the Identity
38         %matrix
39         H = zeros(i,i);% Observation model
40         H(1,1)=1;
41
42         varX=var(x);
43         varNoise= var(x)*(1-dot(phiVarn(i,1:i),rxxnp(2:i+1)));
44         Q=zeros(i);
45         Q(1,1)=varNoise;%Create the Covariance of process noise
46
47         R=eye(i)*R_vars_buffer(j); %R Covariance of observation noise
48
49         z = xs + sqrt(R(1,1)) * randn(size(xs)); %Measurement of the signal

```



```

50     z=make_extended(z,i); %Extends the measurment vector to make it fit for
    the AR model.
51
52     xhat = kalman(z, PHI, H, R, Q); %With the Matrices defined, apply the
    Kalman filter to the sequence
53
54     Yhat= (H*xhat); %Apply the observation matrix to obtain the i
55     Yhat=Yhat(1,:);
56     zinput=z(1,:);
57
58     measurement_error = (xs-zinput).^2;
59     filter_error = (xs-Yhat).^2;
60     mse_measure_b(i) = mean(measurement_error);
61     mse_filter_b(i) = mean(filter_error);
62
63     if (i == kmax)
64         error_improvement(j) = mean((mse_measure_b - mse_filter_b)*100/
            mse_measure_b);
65     end
66
67     if (i==kmax && j==LOGSPACELEN-4)
68         hold on
69         plot(zinput, 'g')
70         plot(xs, 'k')
71         plot(Yhat, 'r')
72
73         legend({'Measurement','Input', 'Estimated'})
74         xlabel('Samples [n]');
75         ylabel('Amplitude');
76         title(sprintf(' $\sigma_v^2$ = %.4f ~ p = % ~ $MSE_{Input-}$
            Measure}$ = %.4f ~ $MSE_{Input-Filter}$ = %.4f', R_vars_buffer(j),
            i, mse_measure_b(i), mse_filter_b(i)), 'interpreter', 'latex');
77         suptitle('Kalman Filtering Stages');
78         grid on;
79         hold off
80         if (j ~= LOGSPACELEN)
81             figure();
82         end
83     end
84 end
85 end
86 end
87
88 semilogx(R_vars_buffer, error_improvement);
89 grid on;
90 title('Mejora porcentual del error cuadratico medio');
91 xlabel(sprintf(' $\sigma_v^2$ '), 'interpreter', 'latex');
92 ylabel('MSE %');
93
94 function xtended = make_extended(x,k)
95     size_ = size(x);
96     size_=size_(2);
97     xtended=zeros(k,size_);
98     for i = 1:k
99         xtended(i,:)= [zeros(1,i-1) x(1:size_-(i-1))];
100         %aca tengo que hacer que haga el extendido solo cosa de que sol con
101         % maneje todo
102     end
103 end

```

■ Cpar.m:

```

1 function [phikk, phi, phis_triang] = cpar(rxx, kmax, flag_print)
2
3     phikk = rxx(2);
4     phis_triang=zeros(kmax-1);
5     phi = rxx(2);
6     phis_triang(1,1)=rxx(2);
7     for i = 2:kmax-1
8
9         R = toeplitz([rxx(1:i)]);
10        phi = linsolve(R, rxx(2:i+1).'); %%Resuelvo el sistema de ecuaciones para
            obtener los phikk
11        phikk = [phikk, phi(end)];
12        phis_triang(i,:) = [phi' zeros(1, kmax-1-i)];
13        if(flag_print)
14            fprintf('Yule walker matrices %d\r\n', i)
15            phi
16            R
17        end
18    end
19 end

```

■ Rnp.m:

```

1 function [Rxx] = Rnp(x, kmax)
2     N=max(size(x));
3     Rxx=0;
4     for i = 0:kmax-1
5         Rxx=[Rxx, (sum(x(1:N-i) .* x(i+1:N)) * (1/(N-i)))]; %%aplico el algoritmo
6     end
7     Rxx=Rxx(2:end);
8 end

```

■ kalman.m:

```

1 function xhat = kalman(z, Phi, H, R, Q)
2
3     %z Measurement signal                m observations X # of observations
4     %Phi State transition model           n X n, n = # of state values
5     %H Observation model                 m X n
6     %R Covariance of observation noise   m X m
7     %Q Covariance of process noise       n X n
8
9     m = size(H, 1); %Number of sensors
10    n = size(H, 2); %Number of state values
11    numobs = size(z, 2); %Number of observations
12    xhat = zeros(n, numobs); %Observation
13
14    %Use linear least squares to estimate initial state from initial
15    %xhat(:,1) = H \ z(:,1);
16
17    %Initialize P, I
18    P = ones(size(Phi));
19    I = eye(size(Phi));
20
21    %Kalman Filter
22    for k = 2:numobs
23        %Predict
24        xhat_acotado=xhat(:,k-1);
25

```

```
26     xhat(:,k) = Phi*xhat_acotado;           %Shamugan 7.133 pag 433
27     P = Phi*P*Phi' + Q; %Borwn picture 4.1 pag 147
28
29     %Update
30     num = P*H';
31     den = (H*P*H' + R);
32     K = num/den;
33     P = (I - K*H)*P;
34     xhat(:,k) = xhat(:,k) + K*(z(:,k) - H*xhat(:,k));
35 end
```