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

Se analiza una secuencia $X(n)$ de 32768 muestras, estimando y calculando parámetros de interés, como lo son la autocorrelación, los coeficientes de correlación parcial, a partir de estos se generará un modelo AR con el propósito de ajustar dicha serie, y con los coeficientes autoregresivos diseñar un modelo en variables de estados del sistema para utilizar un filtor de Kalman.
A la secuencia se le agregará ruido blanco gaussiano aditivo (AWGN) para simular el ruido de medición.
Cabe destacar que en este informe se hará referencia a una variable p, esta tiene un rango entre 1 y 9.

0.2. Estimación de la Autocorrelación

Se estiman la autocorrelación mediante el uso de los primeros p elementos de la secuencia brindada. Para ello, se vale del no polarizados (R_{np}) de dicho parámetro. Esta funcion es empleadas para estimar otras funciones mediante información digitalizada.

$$R_{np}(k) = \frac{1}{N-k} \sum_{i=0}^{N-k-1} X(i)X(i+k) \tag{1}$$

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

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 la estimaciones no polarizada, para cada p se obtiene un vector de tamaño $[1 \times p]$ que corresponde a los $\phi_{p,i}$.

Figura 2: Coeficientes de correlación parcial a partir de la matriz de Yule Walker.

Cabe destacar que en ningun caso los coeficcientes de correlacion parcial se hacen nulos, por lo que no hay seguridad de que un modelo Auto Regresivo se ajuste a la secuencia.

0.4. Modelo del proceso

El tipo de modelo a utilizar para el proceso será un Auto Regresivo, se utilizará un AR(p). .
Para el calculo de los ϕ, son los obtenidos de la matriz de Yule Walker

$$\textit{Ecuacion de yule walker} \tag{2}$$

Figura 3: Grafico relevante.

0.5. Filtro de Kalman

0.5.1. Introducción

En el campo de la estadística y teoría de control, los filtros de Kalman, también conocidos como estimación cuadrática media(LQE), es un algoritmo que utiliza una serie de mediciones realizadas por un observador a través del tiempo, el cual contiene ruido estadístico y otras in-certezas, y produce estimaciones de variables , que tienden a ser mucho mas acertadas que otras basadas en una única medicion, al estimar la distribución de probabilidad conjunta sobre las variables por cada tiempo t_k, El filtro es llamado así por su desarrollador, Rudolf E. Kálmán,

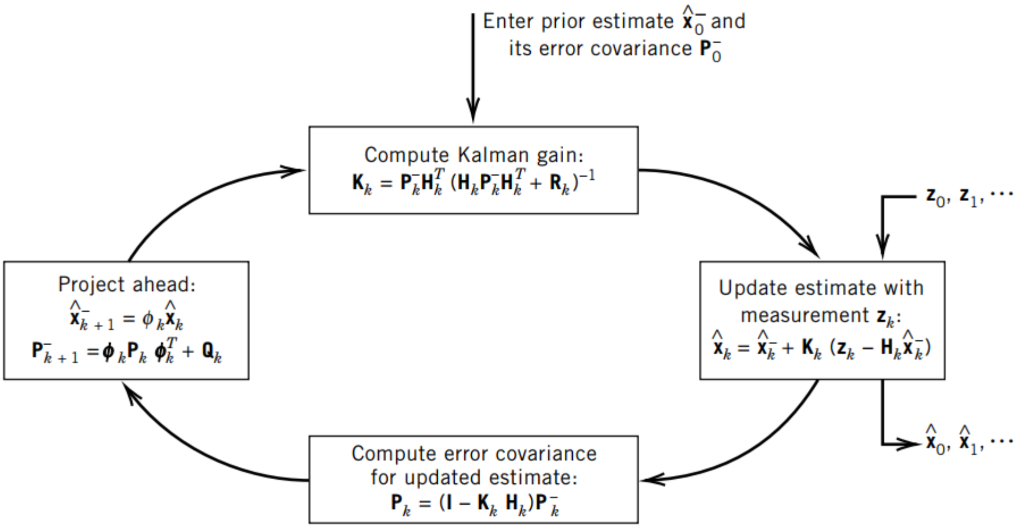

Figura 4: Iteración del filtro de Kalman.

0.5.2. Modelo en variables de estado

Aca explico las matrices y el modelo estas cosas de LaTeXesta bien piola

$$\underset{n\times 1}{\mathrm{Y}} = \underset{n\times p}{X} \times \underset{p\times 1}{\theta} + \underset{n\times 1}{\varepsilon} \tag{3}$$

0.5.3. Implementacion del filtro recursivo

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

0.6. Análisis de resultados

El mierdas comparativo con gráficosa

0.7. Código implementado

El mierda.m

- Main.m:

```
clc
clear
close all
fprintf('Welcome to GT1 matlab script for Kalman filtering\r\n');

%Constants usefull to alter the behaviour of the script
kmax=9;%Value of P
SAMPLES=100;
LOGSPACE_LEN = 10;

%Buffers for variables.
R_vars_buffer=logspace(-2,2,LOGSPACE_LEN);%diferent values for the variance of the measurment
error_improvement = zeros(1,LOGSPACE_LEN);
mse_measure_b=zeros(1,kmax);
mse_filter_b=zeros(1,kmax);

%%actual script
x=load('h06g1.dat')';%loads the sample vector

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)

for j = 1:LOGSPACE_LEN%For each Variance
```

```
    for i = 1:kmax%For every p

        Rxxnp = Rnp(x,i+1); %Estimate the  Autocorrelation function using a non
            polarized estimator
        rxxnp = Rxxnp./Rxxnp(1); %Normalize it

        if(j==1 && i==kmax)
            [phikknp, phiv, phiVarn] = cpar(rxxnp, i+1,1); %Obtain the partial
                correlation coeffients by solving the Yule Walker equation.
        else
            [phikknp, phiv, phiVarn] = cpar(rxxnp, i+1,0); %Obtain the partial
                correlation coeffients by solving the Yule Walker equation.
        end

        % Kalman matrices.
        PHI = [phiv';[eye(i-1) zeros(i-1,1)]]; %Phi matrix for the Kalman filter,
            also known as the State transition model
        %%The first row is the AR coefficients, the other is the Identity
        %%matrix
        H = zeros(i,i);% Observation model
        H(1,1)=1;

        varX=var(x);
        varNoise= var(x)*(1-dot(phiVarn(i,1:i),rxxnp(2:i+1)));
        Q=zeros(i);
        Q(1,1)=varNoise; %Create the Covariance of process noise

        R=eye(i)*R_vars_buffer(j); %R Covariance of observation noise

        z =  xs + sqrt(R(1,1)) * randn(size(xs)); %Measurement  of the signal
        z=make_extended(z,i); %Extends the measurment vector to make it fit for
            the AR model.

        xhat = kalman(z, PHI, H, R, Q); %With the Matrices defined, apply the
            Kalman filter to the sequence

        Yhat= (H*xhat); %Apply the observation matrix to obtain the i
        Yhat=Yhat(1,:);
        zinput=z(1,:);

        measurement_error = (xs-zinput).^2;
        filter_error = (xs-Yhat).^2;
        mse_measure_b(i) = mean(measurement_error);
        mse_filter_b(i) = mean(filter_error);

        if (i == kmax)
            error_improvement(j) = mean((mse_measure_b - mse_filter_b)*100/
                mse_measure_b);
        end

        if(i==kmax && j==LOGSPACE_LEN-4)
            hold on
            plot(zinput, 'g')
            plot(xs, 'k')
            plot(Yhat, 'r')

            legend({'Measurement','Input', 'Estimated'})
            xlabel('Samples [n]');
```

```
                ylabel('Amplitude ');
                title(sprintf(' $$\\sigma_v^2$$ = %.4f ~ p = %i ~ $$MSE_{Input-Measure}$$=%.4f ~ $$MSE_{Input-Filter}$$=%.4f', R_vars_buffer(j), i, mse_measure_b(i), mse_filter_b(i)), 'interpreter', 'latex');
                suptitle('Kalman Filtering Stages');
                grid on;
                hold off
                if(j ~= LOGSPACE_LEN)
                    figure();
                end

            end
        end
end

    semilogx(R_vars_buffer,error_improvement);
    grid on;
    title('Mejora porcentual del error cuadratico medio');
    xlabel(sprintf('$$\\sigma_v^2$$'),'interpreter', 'latex');
    ylabel('MSE %');

function xtended = make_extended(x,k)
    size_ = size(x);
    size_=size_(2);
    xtended=zeros(k,size_);
   for i = 1:k
       xtended(i,:)= [zeros(1,i-1) x(1:size_-(i-1))];
       %aca tengo que hacer que haga el extendido solo cosa de que sol con
       %l maneje todo
   end
end
```

- Cpar.m:

```
function [phikk,phi,phis_triang] = cpar(rxx,kmax,flag_print)

    phikk = rxx(2);
    phis_triang=zeros(kmax-1);
    phi = rxx(2);
    phis_triang(1,1)=rxx(2);
    for i = 2:kmax-1

        R = toeplitz([rxx(1:i)]);
        phi = linsolve(R,rxx(2:i+1).'); %%Resuelvo el sistema de ecuaciones para obtener los phikk
        phikk = [phikk, phi(end)];
        phis_triang(i,:) = [phi' zeros(1,kmax-1-i)];
        if(flag_print)
        fprintf('Yule walker matrices %i\r\n',i)
        phi
        R
        end
    end
end
```

- Rnp.m:

```
function [Rxx] = Rnp(x,kmax)
    N=max(size(x));
    Rxx=0;
```

```
    for i = 0:kmax-1
        Rxx=[Rxx,(sum(x(1:N-i) .* x(i+1:N))*(1/(N-i)))]; %%aplico el algoritmo
    end
    Rxx=Rxx(2:end);
end
```

- kalman.m:

```
function xhat = kalman(z, Phi, H, R, Q)

% z Measurement signal              m observations X # of observations
% Phi State transition model        n X n, n = # of state values
% H Observation model               m X n
% R Covariance of observation noise m X m
% Q Covariance of process noise     n X n

m = size(H, 1); %Number of sensors
n = size(H, 2); %Number of state values
numobs = size(z, 2);    %Number of observations
xhat = zeros(n, numobs);        %Observation

%Use linear least squares to estimate initial state from initial
%xhat(:,1) = H \ z(:,1);

%Initialize P, I
P = ones(size(Phi));
I = eye(size(Phi));

%Kalman Filter
for k = 2:numobs
    %Predict
    xhat_acotado=xhat(:,k-1);

    xhat(:,k) = Phi*xhat_acotado;       %Shamugan 7.133 pag 433
    P = Phi*P*Phi' + Q; %Borwn picture 4.1 pag 147

    %Update
    num = P*H';
    den = (H*P*H' + R);
    K = num/den;
    P = (I - K*H)*P;
    xhat(:,k) = xhat(:,k) + K*(z(:,k) - H*xhat(:,k));
end
```