University of Padua
Degree Course in
ICT for Internet and Multimedia
Academic Year 2017/2018

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

1^{st} Homework, Code

SANTI GIOVANNI - 1177739

VANIN EDOARDO - 1179018

```
clc;
close all;
clear global;
clearvars;

load('signal01.mat', 'x');
N = length(x);

%% SPECTRAL ANALYSIS
L=floor(N/5);
[rx]=autocorrelation(x,L);

w_rect=window(@rectwin,2*L+1);
w_hamming=window(@hamming,2*L+1);
w_bartlett=window(@bartlett,2*L+1);
Pbt3=correlogram(x, w_hamming, rx, L);
Pbt2=correlogram(x, w_rect, rx, L);
%Pbt1=correlogram(x, w_bartlett, rx, L);

X=fft(x);
Pper=(1/N)*(abs(X)).^2;

S=70;    %overlap
D=100;   %window length
w_welch=window(@hamming,D);
[Welch_P, Ns] = welchPSD(x, w_welch, S);
var_Welch=Welch_P.^2/Ns;

b = zeros(1,800);
for i=1:length(b)
    b(i) = 10*log10(0.1);
end
```

```
b(ceil(0.17*800)) = 10*log10(N);
b(ceil(0.78*800)) = 0.8*10*log10(N);

N_ord = 2;
[copt, Jmin, det_R ]=predictor(rx, N_ord);
t=10;
Jvect=zeros(t,1);
for i=1:length(Jvect)
    [c_it, J_it]=predictor(rx, i);
    Jvect(i)=J_it;
end

[a, s_white, d] = AR_Model(N_ord, rx);
[H_w, omega] = freqz(1, [1; a], N, 'whole');

ph = angle(roots([1;a]))/(2*pi);

function [rx]=autocorrelation(x,L)
% Unbiased autocorrelation estimator

K=length(x);
rx=zeros(K, 1);
autoc_full = zeros(K, 1);
   for n=1:K
       xnk=x(n:K);
       xconj=conj(x(1:(K-n+1)));
       autoc_full(n)=(xnk.'*xconj)/(K-n+1);
   end
autoc_full = autoc_full(1:L);
rx(1:L) = autoc_full;
temp = flipud(conj(autoc_full));

if L < K/2
    rx((K-L+1):K) = temp(1:length(temp));
end

end

function [corr] = correlogram( x, window, rx, L )
% Compute the PSD estimate using the correlogram method

K = length(x);
autoc_complete = full_autocorr(x, rx);
full_win = zeros(K, 1);
full_win(1 : L + 1) = window(L + 1 : 2*L + 1);
full_win(K - L + 1 : K) = window(1 : L);

windowed_autoc = autoc_complete .* full_win;
corr = fft(windowed_autoc);
```

```
end

function [a, sw, det_R]=AR_Model(N, rx)
%finds the parameters of an AR model of a r.p.
col=rx(1:N);
row=conj(col);
R=toeplitz(col, row);
r=rx(2:N+1);
det_R=det(R);
a=-inv(R)*r;
sw=abs(R(1,1)+r'*a);
end

function [copt, Jmin, det_R]=predictor(rx, N)

[a, sw, det_R ]=AR_Model(N, rx);
copt=-a;
Jmin=sw;

end

function [est, Ns] = welchPSD(inputsig, win, S)
D = length(win);
K = length(inputsig);
Mw = sum(win .^ 2) * (1/D);
N_s = floor((K-D)/(D-S) + 1);
P_per = zeros(K, N_s);

for s = 0:(N_s-1)
    x_s = win .* inputsig(s*(D-S)+1:s*(D-S)+D);
    X_s = fft(x_s, K);
    P_per(:,s+1) = (abs(X_s)).^2 * (1/(D*Mw));
end
est = sum(P_per, 2) * (1/N_s);
Ns = length(est);
end

close all; clear global; clearvars; clc;

%% GENERATE THE PROCESS x(k), 1 REALIZATION

Nsamples=800;
% Frequencies of the exponentials
f1=0.17;
f2=0.78;
% Generate the white noise (2 components)
sigmaw=0.1;
x = zeros(800,300);

for i=1:300
    % Real part
```

```
    wi=sigmaw*randn(Nsamples,1);
    % Imaginary part
    wq=sigmaw*randn(Nsamples,1);
    % Generate the initial phases
    phi1=2*pi*rand(1);
    phi2=2*pi*rand(1);

    xi=zeros(Nsamples,1);
    xq=zeros(Nsamples,1);
    for k=1:Nsamples
        xi(k)=cos(2*pi*f1*k+phi1)+0.8*cos(2*pi*f2*k+phi2)+wi(k);
        xq(k)=sin(2*pi*f1*k+phi1)+0.8*sin(2*pi*f2*k+phi2)+wq(k);
    end
    x(:,i) = xi + 1i*xq;
end
save('realizations.mat', 'x');


clc; close all; clear global; clearvars;

%% Least Mean Squares Estimation

% Load one realization
load('signal01.mat','x');

% Set parameters
L = floor(length(x)/5);
N = 2;
rx = autocorrelation(x,L);
[a, s_white] = AR_Model(N, rx);
K = L;

max_iter = 800;

c = zeros(N, max_iter + 1);
e = zeros(1, max_iter);

mu_tilde = 0.06;
mu = mu_tilde/(rx(1)*N);

z = x - mean(x);

for k = 1:max_iter
    if (k < N + 1)
        x_in = flipud([zeros(N - k + 1, 1); z(1:k - 1)]);
        y_k = x_in.'*c(:, k);
    else
        x_in = flipud(z((k - N):(k-1)));
        y_k = x_in.'*c(:, k);
    end
    e_k = z(k) - y_k;
```

```
    e(k) = e_k;
    c(:, k+1) = c(:, k) + mu*e_k*conj(x_in);
end

clc; close all; clear global; clearvars;

% Load one realization
load('realizations.mat','x');
% Max number of iterations
max_iter = 800;
% Set parameters
L = floor(size(x,1)/5);
N = 2;
K = L;

% Error vector initialization
e = zeros(size(x,2), max_iter);
C1=zeros(300,801);
C2=zeros(300,801);
for i=1:size(x,2)
    % Autocorrelation
    rx = autocorrelation(x(:,i));
    rx = rx(1:L);
    [a, s_white] = AR_Model(N, rx);
    c = zeros(N, max_iter + 1);
    mu_tilde = 0.06;
    mu = mu_tilde/(rx(1)*N);

    z = x(:,i) - mean(x(:,i));

        for k = 1:max_iter
            if (k < N + 1)
                x_in = flipud([zeros(N - k + 1, 1); z(1:k - 1)]);
                y_k = x_in.'*c(:, k);
            else
                x_in = flipud(z((k - N):(k-1)));
                y_k = x_in.'*c(:, k);
            end
            e_k = z(k) - y_k;
            e(i,k) = e_k;
            c(:, k+1) = c(:, k) + mu*e_k*conj(x_in);
        end
        c1=c(1,:);
        c2=c(2,:);
        C1(i,:)=c1;
        C2(i,:)=c2;
end

c_mean(1,:)=mean(C1);
c_mean(2,:)=mean(C2);
```

```
mean_error = mean(abs(e.^2));

close all; clear global; clearvars; clc;

%% GENERATE THE PROCESS x(k), 1 REALIZATION

Nsamples=800;
% Frequencies of the exponentials
f1=0.17;
f2=0.78;
% Generate the white noise (2 components)
sigmaw=0.1;
% Real part
wi=sigmaw*randn(Nsamples,1);
% Imaginary part
wq=sigmaw*randn(Nsamples,1);
% Generate the initial phases
phi1=2*pi*rand(1);
phi2=2*pi*rand(1);

xi=zeros(Nsamples,1);
xq=zeros(Nsamples,1);
for k=1:Nsamples
    xi(k)=cos(2*pi*f1*k+phi1)+0.8*cos(2*pi*f2*k+phi2)+wi(k);
    xq(k)=sin(2*pi*f1*k+phi1)+0.8*sin(2*pi*f2*k+phi2)+wq(k);
end

% Complex r.p. x(k), 800 samples
x=xi+1i*xq;

save('signal01.mat', 'x');

close all; clear global; clearvars; clc;

% GENERATE THE PROCESS x(k), 1 REALIZATION

Nsamples=800;
% Frequencies of the exponentials
f1=0.17;
f2=0.78;
% Generate the white noise (2 components)
sigmaw=2;
% Real part
wi=sigmaw*randn(Nsamples,1);
% Imaginary part
wq=sigmaw*randn(Nsamples,1);
% Generate the initial phases
phi1=2*pi*rand(1);
phi2=2*pi*rand(1);
```

```
xi=zeros(Nsamples,1);
xq=zeros(Nsamples,1);
for k=1:Nsamples
    xi(k)=cos(2*pi*f1*k+phi1)+0.8*cos(2*pi*f2*k+phi2)+wi(k);
    xq(k)=sin(2*pi*f1*k+phi1)+0.8*sin(2*pi*f2*k+phi2)+wq(k);
end

% Complex r.p. x(k), 800 samples
x=xi+1i*xq;

% save('signal2.mat', 'x');
```