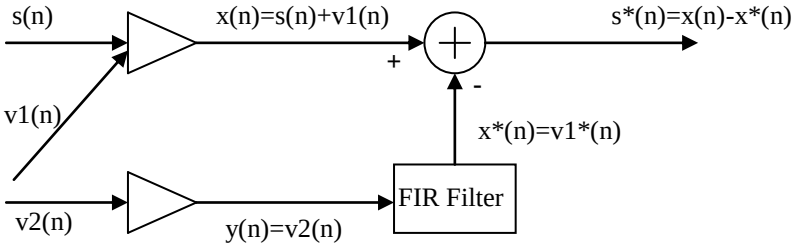
Αλγόριθμος Levinson
Εφαρμογή: Καταστολή Θορύβου με χρήση FIR φίλτρου Wiener

Χάρης Γεωργίου, 1999

Το πρόβλημα της καταστολής θορύβου σε κανάλι μετάδοσης προσεγγίζεται ως εξής. Χρησιμοποιώντας δύο στατιστικά ανεξάρτητα δείγματα του (λευκού) θορύβου, κατασκευάζεται FIR φίλτρο Wiener το οποίο λειτουργεί ως εκτιμητής του μεταδιδόμενου θορύβου σήματος. Αφαιρώντας την εκτίμηση αυτή από το αλλοιωμένο σήμα, προκύπτει μια εκτίμηση του αρχικού σήματος.



Η κατασκευή του FIR φίλτρου Wiener γίνεται χρησιμοποιώντας τον αλγόριθμο Levinson για τον γρήγορο υπολογισμό των συντελεστών του φίλτρου.

Χρησιμοποιώντας λευκό θόρυβο v(n) με μέση τιμή 0 και διασπορά 1, κατασκευάζονται τα σήματα θορύβου v1(n) και v2(n):

$$v1(n)=-0.5*v1(n-1)+v(n)$$
$$v2(n)=0.8*v2(n-1)+v(n)$$

ενώ σαν αρχικό σήμα χρησιμοποιείται μια απλή τριγωνομετρική συνάρτηση:

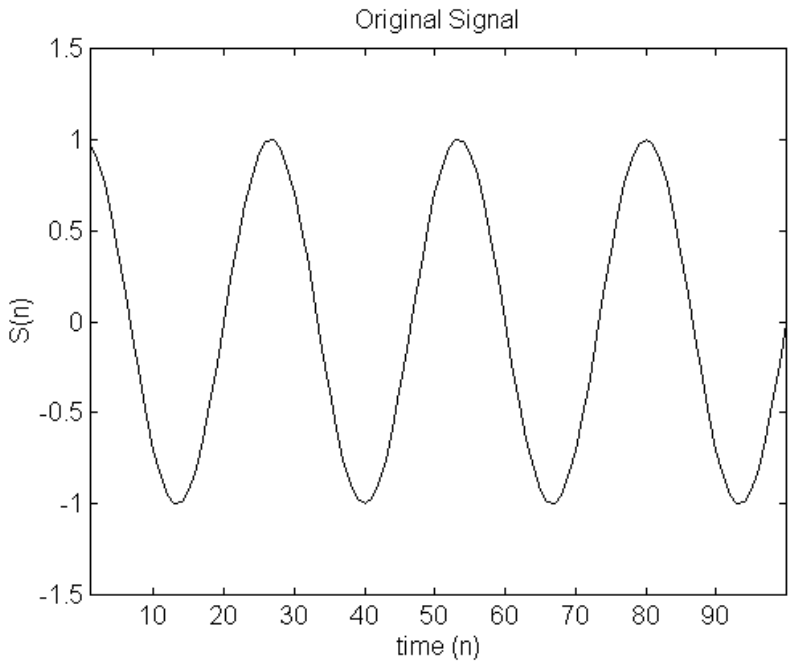
$$s(n)=\cos(\omega_0*n) \quad , \omega_0=0.075$$

Για την κατασκευή του φίλτρου χρησιμοποιούνται οι σχέσεις:

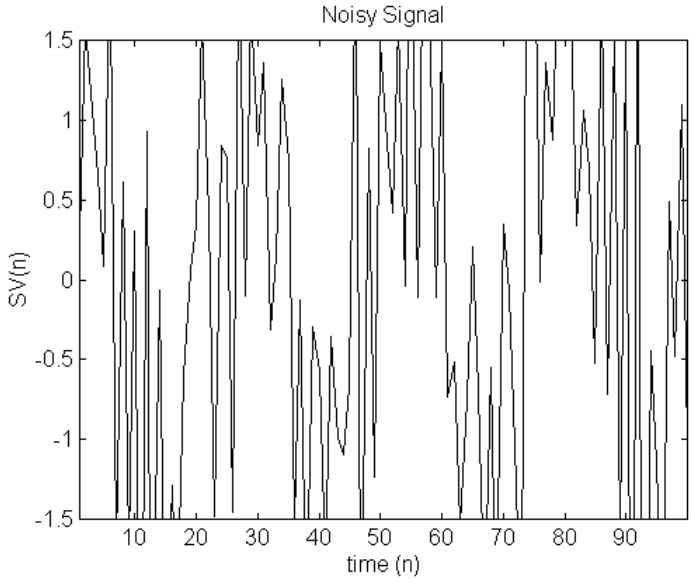
$$R_{yy}(\kappa)=(a_2)^{\kappa}/(1-(a_2)^2) \quad , \kappa>=0$$
$$R_{yx}(-\kappa)=(a_1)^{\kappa}/(1-a_1*a_2) \quad , \kappa>=0$$

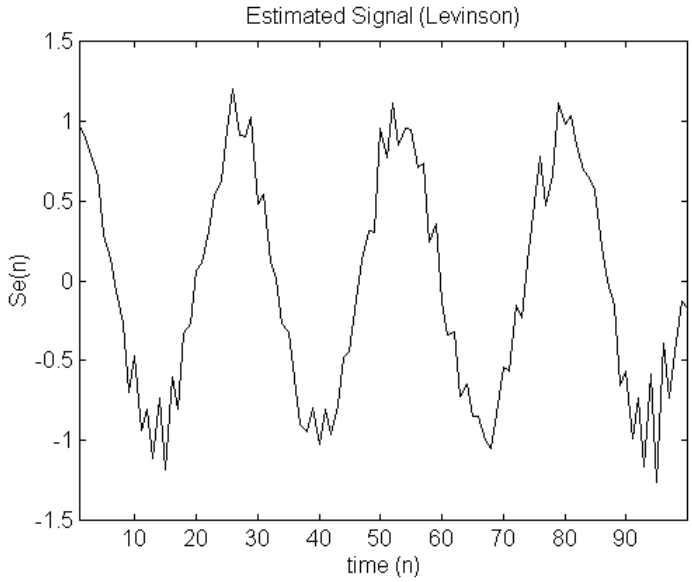
Οι παρακάτω γραφικές παραστάσεις παρουσιάζουν τα αποτελέσματα της εφαρμογής του αλγορίθμου για φίλτρο τάξης m=4 και m=6.

Αρχικό σήμα

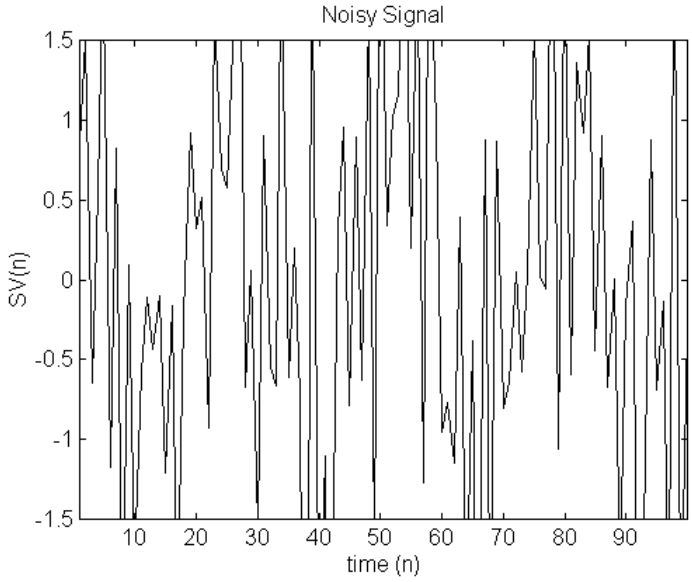

Original Signal
1.5
1
0.5
0
-0.5
-1
-1.5
S(n)
10
20
30
40
50
60
70
80
90
time (n)

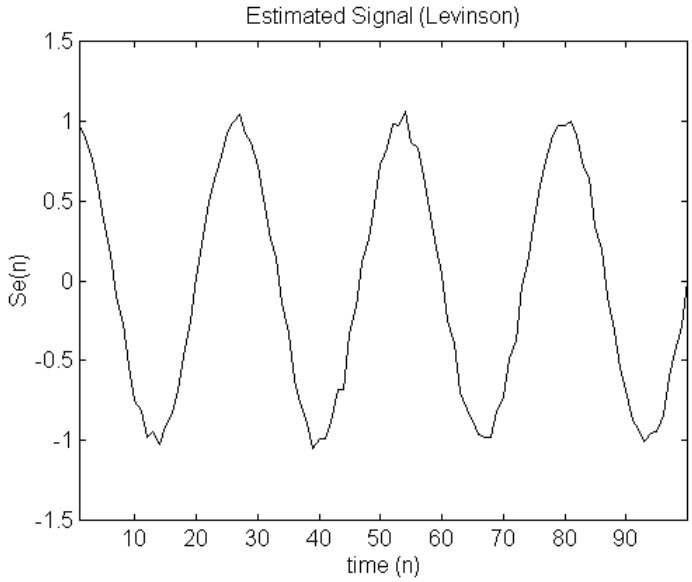
Φίλτρο FIR: m=4





Φίλτρο FIR: m=6





```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                                              %%%
%%%   Noise Cancelling using Wiener FIR filter   %%%
%%%   ----------------------------------------   %%%
%%%                                              %%%
%%%                                              %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


disp('Noise Cancelling Using Wiener FIR Filter, Levinson Solution');

%... initialize constants and variables ...%
clear all;

disp('Initializing variables');
N=100;          % number of samples
M=6;            % filter order
c1=-0.5;        % noise v1(n)=c1*v1(n-1)+v(n)
c2=0.8;         % noise v2(n)=c2*v2(n-1)+v(n)
f0=0.075*pi;    % original signal s(n)=cos(f0*n)
nsc=1.0;        % scale noise range

v=zeros(1,N); v1=zeros(1,N); v2=zeros(1,N);
s=zeros(1,N);
rv=zeros(1,M); Pv=zeros(M,1);

%... create original (s(n)) and white noise (v(n),v1(n),v2(n)) signals ...%
v(1)=nsc*randn; v1(1)=v(1); v2(1)=v(1); s(1)=cos(f0);
for i=2:N,
    v(i)=nsc*randn;
    v1(i)=v1(i-1)*c1+v(i);
    v2(i)=v2(i-1)*c2+v(i);
    s(i)=cos(f0*i);
end

%... create correlation vectors rv(i),Pv(i) and matrix r(i) ...%
for i=1:M,
    rv(i)=(c2^(i-1))/(1-c2^2);
    Pv(i)=(c1^(i-1))/(1-c1*c2);
end


%----- Gauss algorithm (classic solution) -----%

disp('Calculating Gaussian solution');

r=toeplitz(rv(1:M));      % construct matrix r(i) from vector rv(i)
wg=(Pv\r)';               % calculate correct linear system solution


%----- Levinson algorithm -----%

disp('Calculating Levinson solution');

%... initialize variables ...%
a=1;
wl=Pv(1)/rv(1);
p=rv(1);


%... begin main loop ...%
for i=2:M;
```

```
    g(i-1)=-(rv(2:i)*flipud(a))/p;              % calculate (current) 'G'
coefficient
    a=[a;0] + [0;conj(flipud(a))]*g(i-1);       % update 'A' matrix
    p=p*(1 - g(i-1)^2);                         % calculate 'P' coefficient
    k=(Pv(i)-(rv(2:i)*flipud(wl)))/p;           % calculate 'K' coefficient
    wl=[wl;0] + [conj(flipud(a))]*k;            % update 'W' matrix (solution of
order 'i')
end

%... diplay Levinson and Gaussian solution vectors ...%
disp('Gaussian solution:');
wg
disp('Levinson solution:');
wl
disp('...<press any key to continue>...');
pause;


%... plot original signal ...%
disp('Creating original signal plot');
figure(1);
plot(s);
axis([1,N,-1.5,1.5]);
xlabel('time (n)');
ylabel('S(n)');
title('Original Signal');
disp('...<press any key to continue>...');
pause;


%... plot signal noise ...%
%disp('Creating signal noise plot');
%figure(6);
%plot(v1);
%axis([1,N,-1.5,1.5]);
%xlabel('time (n)');
%ylabel('v1(n)');
%title('White Noise V1');
%disp('...<press any key to continue>...');
%pause;


%... calculate and plot noisy signal ...%
disp('Creating noisy signal plot');

sv = s + v1;

figure(2);
plot(sv);
axis([1,N,-1.5,1.5]);
xlabel('time (n)');
ylabel('SV(n)');
title('Noisy Signal');
disp('...<press any key to continue>...');
pause;


%... calculate and plot restored signal (Gauss) ...%
%disp('Creating restored signal plot (Gaussian)');
%
%xe = filter(wg,1,v2);   % FIR filtering
%se = sv - xe;           % calculate estimate
%
%figure(5);
%plot(se);
```

```
%axis([1,N,-1.5,1.5]);
%xlabel('time (n)');
%ylabel('Se(n)');
%title('Estimated Signal (Gaussian)');
%disp('...<press any key to continue>...');
%pause;


%... calculate and plot restored signal (Levinson) ...%
disp('Creating restored signal plot (Levinson)');

xe = filter(wl,1,v2);   % FIR filtering
se = sv - xe;           % calculate estimate

figure(3);
plot(se);
axis([1,N,-1.5,1.5]);
xlabel('time (n)');
ylabel('Se(n)');
title('Estimated Signal (Levinson)');
disp('...<press any key to continue>...');


disp('Process completed.');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Noise Cancelling Using Wiener FIR Filter, Levinson Solution

Initializing variables
Calculating Gaussian solution
Calculating Levinson solution
Gaussian solution:

wg =

    2.0380
    1.2298
    1.1420
    0.7818

Levinson solution:

wl =

    1.0000
   -1.3000
    0.6500
   -0.2321
```

```
Noise Cancelling Using Wiener FIR Filter, Levinson Solution

Initializing variables
Calculating Gaussian solution
Calculating Levinson solution
Gaussian solution:

wg =

    2.0753
    1.2813
    1.2078
    0.8665
    0.7326
    0.5532

Levinson solution:

wl =

    1.0000
   -1.3000
    0.6500
   -0.3250
    0.1625
   -0.0580
```

22 Νοεμβρίου 1999.