#4

# ECE-S 632:
# Fundamentals of Stochastic DSP

Prof. John Walsh

Zexi Liu

Mar 1st, 2010

Electrical and Computer Engineering
Drexel University

**4.1** Project Proposal Write an abstract, complete with background references, on a statistical signal processing problem or problem area interesting to you and/or relevant to your job/research that you would like to investigate for your course project. The abstract is to ensure that your assignment matches your interests. Email me your abstract when it is ready. Over email, you and I will use the abstract to craft a specific mini-research problem of reasonable but limited scope.

The email was sent to Dr. Walsh on Mon 2/22/2010 9:55 PM.

**4.2** Write a MATLAB program which calculates the order 20 forward linear prediction coefficients for a real valued wide sense stationary Gaussian process with mean zero and auto-correlation using the Levinson-Durbin algorithm. Also calculate the MSE of the order 20 linear predictor using the Levinson-Durbin algorithm, and compare it with an empirically observed MSE which you calculate by running the linear predictor on a length 10000 randomly generated sample from the given Gaussian random process and averaging the squared error.

The order 20 forward linear prediction coefficients:

| Order | Coefficients |
|---|---|
| 1 | 0.726636900430018 |
| 2 | 0.00626520788895617 |
| 3 | 0.00412685827773906 |
| 4 | 0.00271833907682490 |
| 5 | 0.00179055519325446 |
| 6 | 0.00117942905352016 |
| 7 | 0.000776883662657109 |
| 8 | 0.000511729304903103 |
| 9 | 0.000337073719683742 |
| 10 | 0.000222029256190734 |
| 11 | 0.000146250425834925 |
| 12 | 9.63358246606418e-05 |
| 13 | 6.34581095886934e-05 |
| 14 | 4.18028923690078e-05 |
| 15 | 2.75404713240350e-05 |
| 16 | 1.81485527615082e-05 |
| 17 | 1.19661860704105e-05 |
| 18 | 7.90002640694090e-06 |
| 19 | 5.23097784495583e-06 |
| 20 | 6.60879642726521e-06 |

The MSE of the order 20 linear predictor (theoretical value):

| Order | Coefficients |
|---|---|
| 0 | 0.800000000000000 |
| 1 | 0.366957265248439 |
| 2 | 0.366903263616263 |
| 3 | 0.366879842199477 |
| 4 | 0.366869681793097 |
| 5 | 0.366865273720419 |
| 6 | 0.366863361210659 |
| 7 | 0.366862531424750 |
| 8 | 0.366862171400546 |
| 9 | 0.366862015194173 |
| 10 | 0.366861947419659 |
| 11 | 0.366861918013769 |
| 12 | 0.366861905255187 |
| 13 | 0.366861899719514 |
| 14 | 0.366861897317704 |
| 15 | 0.366861896275611 |
| 16 | 0.366861895823469 |
| 17 | 0.366861895627294 |
| 18 | 0.366861895542178 |
| 19 | 0.366861895505248 |
| **20** | **0.366861895489225** |

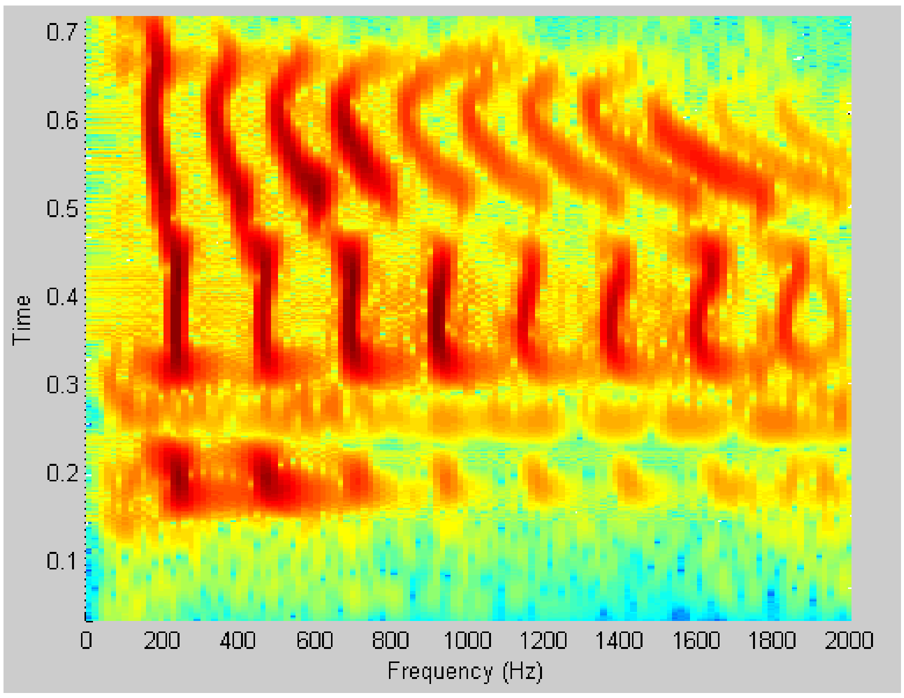
Empirically observed MSE:
**0.225752490827718**

```
% HW4-1
clc;
clear;

R = zeros( 20, 1 );
for i = 1 : 20
   R(i) = 0.2 * exp(-0.5*abs(i)) + 0.6 * exp(-0.25*abs(i));
end
R0 = 0.2 * exp(-0.5*0) + 0.6 * exp(-0.25*0);

f = zeros( 20, 20 );
MSE = zeros( 20, 1 );

MSE0 = R0;
MSE(1) = (1-(R(1)/R0)^2)*MSE0;
f(1,1) = R(1) / MSE0;
for P = 2 : 20
   SUM = 0;
   for i = 1:P-1
      SUM = SUM + f(P-1,i) * R(P-i);
   end
   f(P,P) = ( R(P) - SUM ) / MSE(P-1);
   for i = 1 : P-1
      f(P,i) = f(P-1,i) - f(P,P)* f(P-1,P-i);
   end
   MSE(P) = ( 1 - abs(f(P,P))^2 ) * MSE(P-1);
end

a = levinson( [R0; R], 20 )';

co = f( 20, : )';

X = normrnd( 0, R0, 10000, 1 );      % zero mean Gaussian noise with variance R0
N = numel(X);
Xp = zeros( N, 1 );

XX = conv( X, R );
for i = 21 : N
   Xp(i) = sum( co(1:20) .* XX( i-1:-1:i-20 ) );
end
Err = (XX(1:10000) - Xp).^2;

eMSE = sum(Err(21:10000))/numel(Err(21:10000));
```

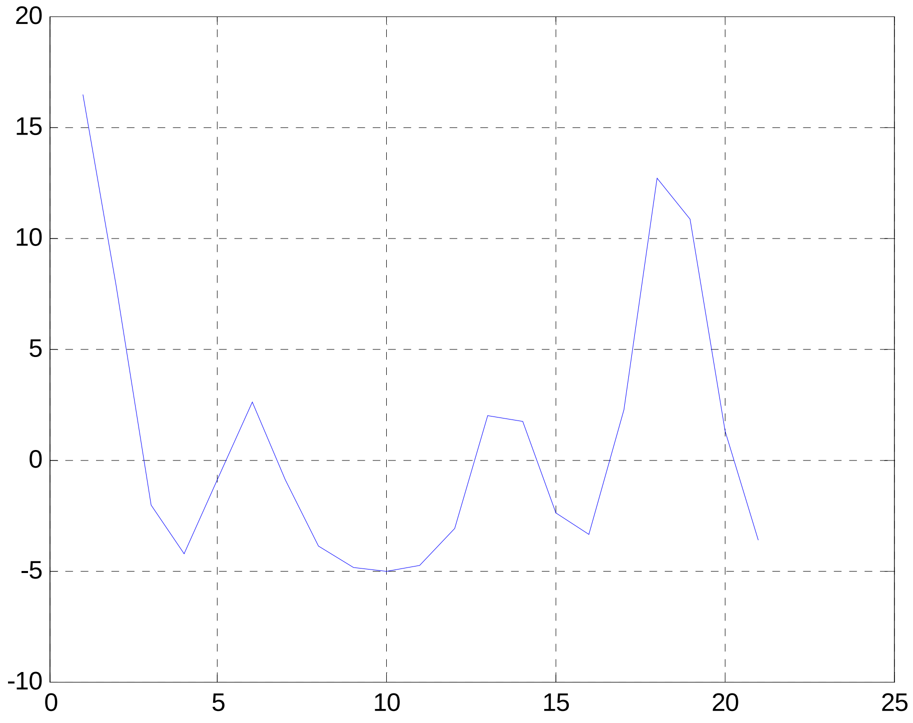
**4.3** Select an audio wav file of your choice that contains speech and load it into Matlab. Speech is a quasi-stationary signal: on short segments (e.g. 32 millisecond blocks) it appears stationary, but it is in its changing spectrum across different such blocks in which carries its information.

(a) Using a spectrogram, find a temporal region of the _le over which the spectrum appears to be constant. Select the portion of the audio _le associated with this temporal region and build an autocorrelation estimate for this region. Generate a linear predictor from this autocorrelation estimate using Levinson Durbin. Record the performance you achieve with this.

The spectrogram of input speech signal.



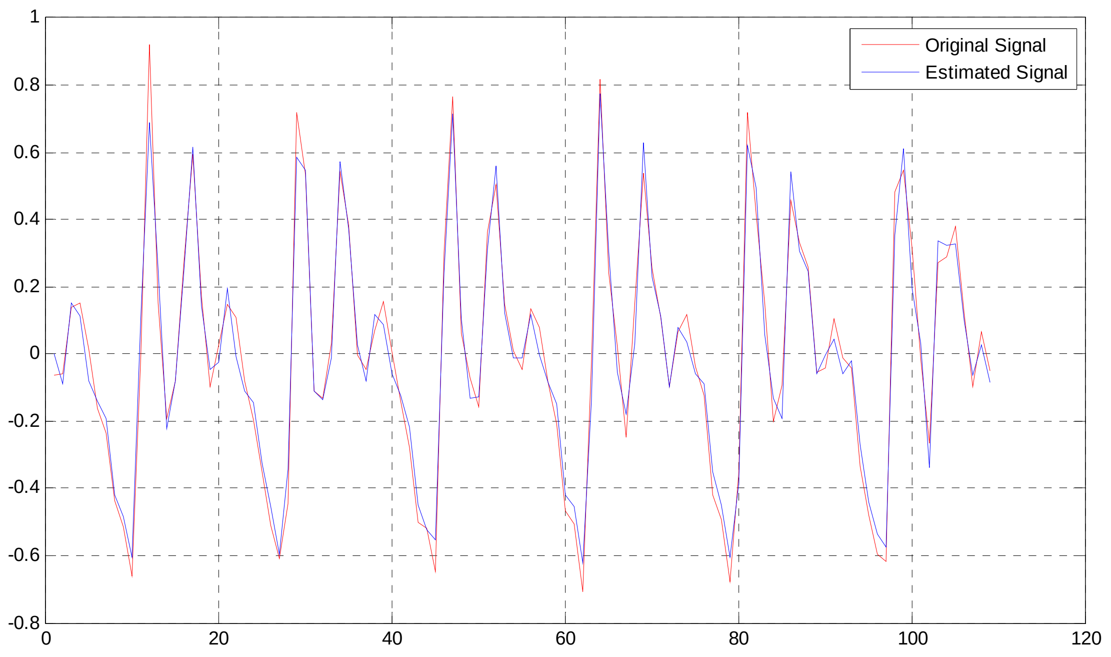
Auto-correlation of the chosen temporal region:



The order 20 forward linear prediction coefficients:

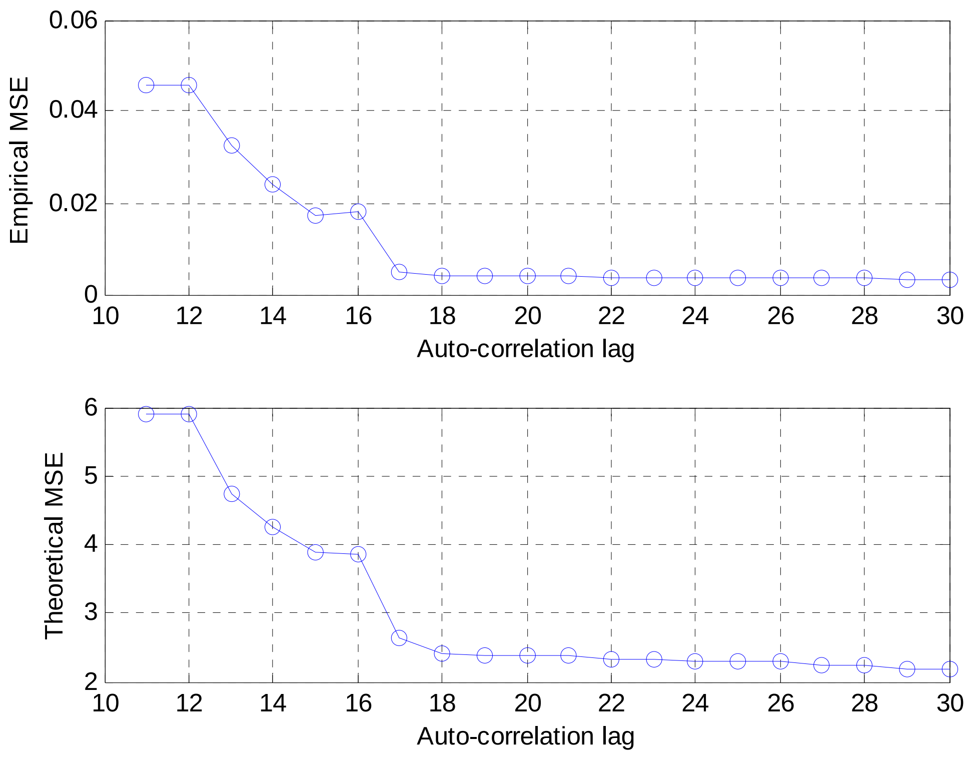
| Order | Coefficients |
|---|---|
| 1 | 0.661028210250749 |
| 2 | -0.596467931594581 |
| 3 | 0.189793085195174 |
| 4 | -0.398600394902704 |
| 5 | 0.247440532874425 |
| 6 | -0.384625222492282 |
| 7 | 0.136756443330365 |
| 8 | -0.354829424009870 |
| 9 | 0.167945542621277 |
| 10 | -0.370210092367595 |
| 11 | 0.0360700626582739 |
| 12 | -0.175716001727205 |
| 13 | 0.0455643928929486 |
| 14 | -0.277896122026717 |
| 15 | 0.116536461547976 |
| 16 | -0.370036261793053 |
| 17 | 0.758057349156431 |
| 18 | -0.365226131399721 |
| 19 | 0.120926824176675 |
| 20 | -0.0149783145425018 |

Original signal compare with estimated signal:



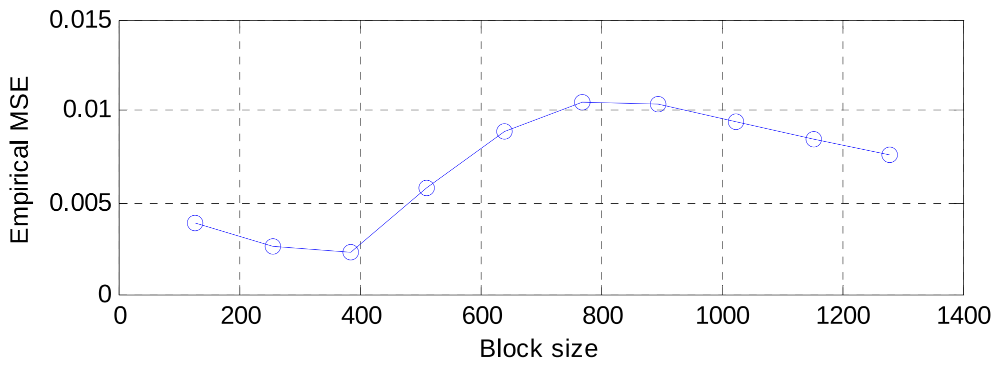
(b) Compare the empirical average squared prediction error over the block you used to calculate the auto-correlation estimate with the theoretical MSE given by Levinson Durbin. Make a plot of these two values as a function of the maximal auto-correlation lag estimated. Why are these two numbers different from one another?

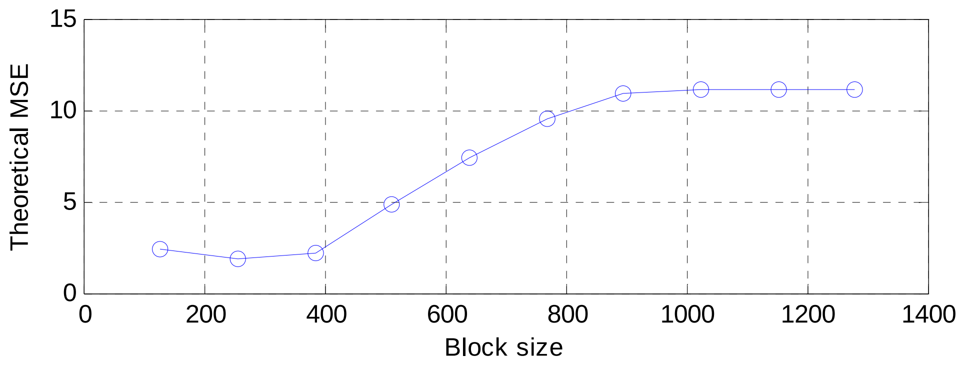
Empirical MSE vs. Theoretical MSE



(c) Repeat this experiment, but by measuring the average squared prediction error over a larger block than the one you used to calculate the auto-correlation estimate. Choose the size of this block to be one over which you can see that spectral properties of the data changing in the spectrogram. How do your estimated squared prediction errors compare now with the ones you obtained with the smaller (original) block? Why?

Block size vs. MSE





```
% HW4-2
eMSE = zeros( 20, 1 );
tMSE = zeros( 20, 1 );
for i = 1 : 20
   [eMSE(i) tMSE(i)] = HW42(i+10, 128);
end
x = 11 : 30;
figure(1);
subplot(2,1,1);
plot( x, eMSE, '-o' );
xlabel('Auto-correlation lag');
ylabel('Empirical MSE');
grid on
subplot(2,1,2);
plot( x, tMSE, '-o' );
xlabel('Auto-correlation lag');
ylabel('Theoretical MSE');
grid on;
```

```
% HW4-2
eMSE = zeros( 8, 1 );
tMSE = zeros( 8, 1 );
k = 1;
for i = 128 : 128 :1280
    [eMSE(k) tMSE(k)] = HW42( 20, i );
    k = k + 1;
end
x = 128 : 128 :1280;
figure(1);
subplot(2,1,1);
plot( x, eMSE, '-o' );
xlabel('Block size');
ylabel('Empirical MSE');
grid on
subplot(2,1,2);
plot( x, tMSE, '-o' );
xlabel('Block size');
ylabel('Theoretical MSE');
grid on;

function [eMSE tMSE]= HW42(lag)

Female = wavread('d.wav');
% figure(1);
% spectrogram(Female,256,250,256,4e3);

R = xcorr( Female(1600:1600+128), lag );
R0 = R(lag+1);
R = R(lag+2:numel(R));

f = zeros( lag, lag );
MSE = zeros( lag, 1 );

MSE0 = R0;
MSE(1) = (1-(R(1)/R0)^2)*MSE0;
f(1,1) = R(1) / MSE0;
for P = 2 : lag
    SUM = 0;
    for i = 1:P-1
        SUM = SUM + f(P-1,i) * R(P-i);
    end
    f(P,P) = ( R(P) - SUM ) / MSE(P-1);
    for i = 1 : P-1
        f(P,i) = f(P-1,i) - f(P,P)* f(P-1,P-i);
    end
    MSE(P) = ( 1 - abs(f(P,P))^2 ) * MSE(P-1);
end

R = [R0; R];
a = levinson( R, lag )';

co = f( lag, : )';

X = Female(1600:1600+128);
N = numel(X);
```

```
Xp = zeros( N, 1 );
for i = lag+1 : N
   Xp(i) = sum( co(1:lag) .* X( i-1:-1:i-lag ) );
end
Err = (X(lag+1:129) - Xp(lag+1:129)).^2;

eMSE = sum(Err)/numel(Err);
tMSE = MSE(numel(MSE));
```

**4.4** Write a MATLAB function which implements the root-MUSIC frequency estimation method. Test it out on a length 1000 signal containing a sinusoid with frequency !$_1$ = 3_=8 and phase 0 with amplitude :6, and a sinusoid at frequency !$_2$ = _=17 with phase _=16 and amplitude :2 observed through complex Gaussian white noise with variance _$_2$ = 1 and mean 0. Try other lengths and observe how the performance of the estimate varies with different lengths and different Monte Carlo trials.

Different lengths and different Monte Carlo trials:

| Length | Trials | Est. frequencies | True frequencies |
|---|---|---|---|
| 10 | 10 | -1.3704 1.2929 | 0.1848 1.1781 |
| | 100 | -1.3160 1.3906 | |
| | 1000 | -1.3121 1.3250 | |
| 100 | 10 | -0.6402 1.1813 | |
| | 100 | -0.7394 1.2539 | |
| | 1000 | -0.6690 1.2889 | |
| 1000 | 10 | 0.2598 1.1900 | |
| | 100 | 0.1417 1.2514 | |
| | 1000 | 0.1705 1.2157 | |
| 10000 | 10 | 0.1884 1.1751 | |
| | 100 | 0.1749 1.1763 | |
| | 1000 | 0.1833 1.1779 | |

```
% HW4-3

clc;
clear;

N = 10000;
T = 1000;
% randn( 'state', 1 );
W = zeros( T, 2 );
P = zeros( T, 2 );
n = 0:N-1;

for i = 1:T
   s = 0.6 * exp(1i*3*pi/8*n) + 0.2 * exp(1i*pi/17*n+pi/16) + normrnd( 0, 1, N, 1 )';
%     Hs = spectrum.music( 2, 20 );
%     pseudospectrum( Hs, s );
   [W(i,:) ~] = sort( rootmusic( s, 2 ) );
end
mean(W)
[ pi/17 3*pi/8 ]
```