EEC 263: MATLAB Assignment 1

Due on Wednesday, Apr. 23, 2014

Wenhao Wu

Contents

Problem 1

Problem 1: (a)

$$
\begin{aligned}
\begin{pmatrix} A_1(z) \\ A_1^R(z) \end{pmatrix} &= \begin{bmatrix} 1 & \gamma \\ \gamma & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 1+\gamma z^{-1} \\ \gamma + z^{-1} \end{pmatrix},
\end{aligned} \tag{1}
$$

$$
\begin{aligned}
\begin{pmatrix} A_2(z) \\ A_2^R(z) \end{pmatrix} &= \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{pmatrix} A_1(z) \\ A_1^R(z) \end{pmatrix} \\
&= \begin{pmatrix} z^{-2}+2\gamma z^{-1}+1 \\ z^{-2}+2\gamma z^{-1}+1 \end{pmatrix},
\end{aligned} \tag{2}
$$

therefore $A_2(z) = z^{-2} + 2\gamma z^{-1} + 1$, its two zeros are

$$
Z_{1,2} = -\gamma \pm \sqrt{\gamma^2 - 1}. \tag{3}
$$

Problem 1: (b)

$$
\begin{aligned}
R_Y(k) &= R_X(k) + R_V(k) \\
&= \frac{A^2}{2}\cos(2\pi f_0 k) + r\delta(k),
\end{aligned} \tag{4}
$$

therefore

$$
S_Y(e^{j\omega}) = \frac{\pi}{2}A^2 \sum_{k=-\infty}^{+\infty} \left[\delta(\omega - 2\pi f_0 + 2\pi k) + \delta(\omega + 2\pi f_0 + 2\pi k)\right] + r. \tag{5}
$$

According to Parseval's theorem, we have

$$
\begin{aligned}
E[E^2(t;2)] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \|A_2(e^{j\omega})\|^2 S_Y(e^{j\omega}) d\omega \\
&= \frac{1}{2\pi}\int_{-\pi}^{\pi} \left[4\gamma^2 + 8\cos(\omega)\gamma + (2\cos(2\omega)+2)\right] S_Y(e^{j\omega}) d\omega.
\end{aligned} \tag{6}
$$

Without loss of generality, assuming $0 < f_0 \le 1/2$, the integration in (6) results in

$$
E[E^2(t;2)] = (4r + 2A^2)\gamma^2 + 4A^2\cos(2\pi f_0)\gamma + [2r + A^2 + A^2\cos(4\pi f_0)], \tag{7}
$$

therefore the optimum reflection coefficient γ to minimize the MSE is

$$
\hat{\gamma} = -\frac{\frac{A^2}{2r}\cos(2\pi f_0)}{1 + \frac{A^2}{2r}}, \tag{8}
$$

and the corresponding MSE is

$$
MSE(2) = 2r\frac{1 + (1 + 2\cos^2(2\pi f_0))\frac{A^2}{2r}}{1 + \frac{A^2}{2r}}. \tag{9}
$$

As a result, γ can be used to estimate f_0 by

$$\hat{f}_0 = \frac{1}{2\pi}\cos^{-1}\left(-\hat{\gamma}\frac{1+\frac{A^2}{2r}}{\frac{A^2}{2r}}\right), \tag{10}$$

and when $A^2/2r \to \infty$, we have

$$\begin{aligned} \hat{\gamma} &\to -\cos(2\pi f_0), &(11)\\ MSE(2) &\to (2 + 4\cos^2(2\pi f_0))r. &(12) \end{aligned}$$

Problem 1: (c)

Denote

$$\mathbf{E} = \begin{pmatrix} E(T) \\ E(T-1) \\ \cdots \\ E(2) \end{pmatrix}, \tag{13}$$

$$\mathbf{Y} = \begin{bmatrix} 2Y(T-1) & Y(T)+Y(T-2) \\ 2Y(T-2) & Y(T-1)+Y(T-3) \\ \cdots & \\ 2Y(1) & Y(2)+Y(0) \end{bmatrix}, \tag{14}$$

$$\boldsymbol{\gamma} = \begin{pmatrix} \gamma \\ 1 \end{pmatrix}. \tag{15}$$

Then we have

$$\begin{aligned} J &= \frac{1}{T-1}\mathbf{E}^T\mathbf{E} \\ &= \frac{1}{T-1}\boldsymbol{\gamma}^T\mathbf{Y}^T\mathbf{Y}\boldsymbol{\gamma} \\ &= \frac{1}{T-1}(a\gamma^2 + 2b\gamma + c) \end{aligned} \tag{16}$$

where

$$\mathbf{Y}^T\mathbf{Y} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \tag{17}$$

is a positive semi-definite matrix. Then the optimal γ to minimize J is

$$\hat{\gamma} = -\frac{b}{a}. \tag{18}$$

Inspired by (11), one simple way to estimate f_0 from γ

$$\hat{f}_0 = \frac{1}{2\pi}\cos^{-1}(-\gamma). \tag{19}$$

The rationale here is that this estimator provides a good estimation when the signal to noise ratio is large, while when signal to noise ratio is low any estimator tends to perform poorly anyway.

Problem 1 continued on next page. . .

Problem 1: (d)

noisin.m

```
function [X, Y] = noisin(A, f0, phi, r, T)

X = A * cos(2 * pi * f0 * (0 : T) + phi);
Y = X + sqrt(r) * randn(1, T + 1);

end
```

conlat.m

```
function [gamma, E, J] = conlat(Y, T)

if (size(Y, 2) ~= T + 1)
    error('Y must be a 1-by-(T + 1) vector!');
end

YMat = [2 * Y(T : -1 : 2)', Y(T + 1 : -1 : 3)' + Y(T - 1 : -1 : 1)'];
YMatSqr = YMat' * YMat;
gamma = - YMatSqr(1, 2) / YMatSqr(1, 1);

[E, ~] = latcfilt([gamma, 1], Y);
E = E(3 : T + 1);
J = norm(E) ^ 2 / (T - 1);
```

plotXYEJ.m

```
function h = plotXYEJ(X, Y, E, J)

T = size(X, 2) - 1;
if (T + 1 ~= size(Y, 2) || T - 1 ~= size(E, 2))
    error('Size of X, Y, E do not match!');
end

h = figure;
subplot(2, 1, 1), plot(0 : T, X, 'b--', 'linewidth', 2), hold on;
subplot(2, 1, 1), plot(0 : T, Y, 'ro-', 'linewidth', 2), hold on;
subplot(2, 1, 1), grid on, set(gca, 'fontsize', 18), legend('X', 'Y'), grid on
    , xlabel('t');

subplot(2, 1, 2), plot(2 : T, J ^ (1 / 2) * ones(1, T - 1), 'b--', 'linewidth'
    , 2), hold on;
subplot(2, 1, 2), plot(2 : T, E, 'ro-', 'linewidth', 2), hold on;
subplot(2, 1, 2), grid on, set(gca, 'fontsize', 18), legend('J^{1/2}', 'E'),
    grid on, xlabel('t');
```

These functions are called in the script file **Main.m**

```
clear all;
close all;
clc;

%% 1. Simulation settings
A = 10;
f0 = 0.25;
phi = 0;
r = 1;
T = 20;

%% 2. Simulation
[X, Y] = noisin(A, f0, phi, r, T);
[gamma, E, J] = conlat(Y, T);
gammaRef = - A ^ 2 / (2 * r) * cos(2 * pi * f0) / (1 + A ^ 2 / (2 * r)); %
    optimal gamma with a priori

%% 3. Postprocessing and visualization
plotXYEJ(X, Y, E, J);
f0Est = acos(-gamma) / (2 * pi);
f0EstRef = acos(-gammaRef * (1 + A ^ 2 / (2 * r)) / (A ^ 2 / (2 * r))) / (2 *
    pi);

disp(['gamma = ', num2str(gamma)]);
disp(['J = ', num2str(J)]);
disp(['Estimated f0 = ', num2str(f0Est)]);
disp('-----------------------------------');
disp(['gamma (with a priori) = ', num2str(gammaRef)]);
disp(['Estimated f0 (with a priori) = ', num2str(f0EstRef)]);
```

When $A = 10$, $f_0 = 0.25$, $T = 20$, $\Phi = 0$, $r = 1$, the results are shown in Fig. 1.

Problem 1 continued on next page. . .

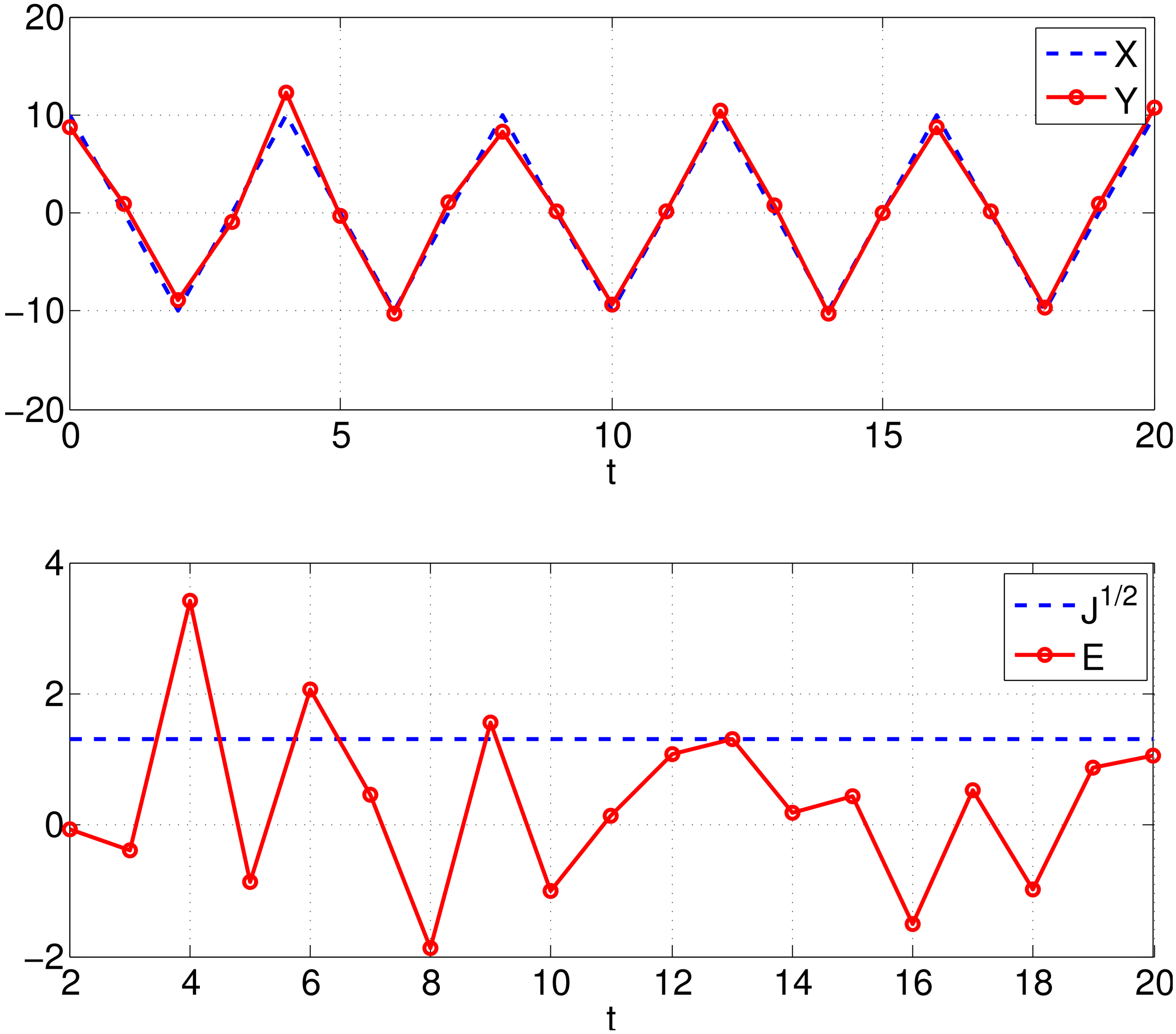

Figure 1: $X(t)$, $Y(t)$ and $E(t)$.

Problem 1: (e)

1. $A = 10,\ f_0 = 0.25$

T	γ	J	$\hat{f}_0$
10	-0.018646	2.8708	0.24703
20	0.006966	2.2364	0.25111
50	-0.0030675	2.7354	0.24951
100	-0.0011253	1.4265	0.24982

2. $f_0 = 0.05$, $T = 100$

A	γ	J	$\hat{f}_0$
1	-0.20207	3.165	0.21762
4	-0.84035	5.0733	0.091176
10	-0.94341	2.304	0.053799
20	-0.94852	3.9141	0.051291

3. $A = 10$, $T = 100$

f_0	γ	J	$\hat{f}_0$
0.01	-0.97846	5.6564	0.033095
0.05	-0.93032	5.3695	0.059767
0.1	-0.79622	3.7875	0.10342
0.25	0.0020769	2.0953	0.25033
0.5	0.98901	6.927	0.47638

It seems that the estimation of f_0 becomes more accurate as T and A grows, while the estimation of f_0 is more accurate when $f_0 \approx 0.25$.

Problem 2

Problem 2: (a)

Denote the tap weight of the desired Wiener filter as $\mathbf{a} = [a_0, a_1, \ldots, a_{N-1}]^T$. The normal equation we need to solve is

$$\mathbf{K}_Y \mathbf{a} = \mathbf{K}_{YX} \tag{20}$$

where $\mathbf{K}_Y$ is the auto covariance matrix of $[Y(t), Y(t-1), \ldots, Y(t-(N-1))]^T$

$$\mathbf{K}_Y = \begin{bmatrix} 1 & a & \cdots & a^{N-1} \\ a & 1 & \ddots & a^{N-2} \\ \vdots & \ddots & \ddots & \vdots \\ a^{N-1} & a^{N-2} & \cdots & 1 \end{bmatrix} + r\mathbf{I} \tag{21}$$

where $\mathbf{K}_{YX}$ is the cross covariance matrix between $[Y(t), Y(t-1), \ldots, Y(t-(N-1))]^T$ and $X(t)$

$$\mathbf{K}_{YX} = [1, a, \ldots, a^{N-1}]^T \tag{22}$$

Since $\mathbf{K}_Y$ is central symmetric Toeplitz, (20) can be efficiently solved with Levinson recursion. We design the matlab function filterWienerFIR() to design the FIR Wiener filter and return the MSE. **filterWienerFIR.m**

```
function [coeff_flt, MSE] = filterWienerFIR(a, r, N)

coeff_prd = zeros(N, 1);
coeff_flt = zeros(N, 1);

M = 1 + r; % MSE(0)
coeff_flt(1) = 1 / (1 + r);

for n = 0 : (N - 2)
    delta = (a .^ ((n + 1) : -1 : 1)) * [1; coeff_prd(1 : n)];
    gamma = - delta / M;
    coeff_prd(1 : n) = coeff_prd(1 : n) + gamma * coeff_prd(n : -1 : 1);
    coeff_prd(n + 1) = gamma;
    M = M * (1 - gamma ^ 2);

    if (M == 0)
        break;
    end

    mu = 0 + a .^ ((n + 1) : -1 : 1) * coeff_flt(1 : (n + 1));
    omega = (a ^ (n + 1) - mu) / M;
    coeff_flt(1 : (n + 1)) = coeff_flt(1 : (n + 1)) + omega * coeff_prd((n +
        1) : -1 : 1);
    coeff_flt(n + 2) = omega;
end

MSE = 1 - a .^ (0 : (N - 1)) * coeff_flt;
```

Problem 2: (b)

When $r = 1$, $a = 0.8$, the MSE and filter tap weights are Note that in Example 7.3.2 in the textbook, the

N	1	2	5	10	20
MSE	0.5	0.40476	0.37546	0.375	0.375
a_0	0.5000	0.4048	0.3755	0.3750	0.3750
a_1		0.2381	0.1883	0.1875	0.1875
a_2			0.0952	0.0938	0.0938
a_3			0.0498	0.0469	0.0469
a_4			0.0293	0.0234	0.0234
a_5				0.0117	0.0117
a_6				0.0059	0.0059
a_7				0.0030	0.0029
a_8				0.0016	0.0015
a_9				0.0009	0.0007
a_{10}					0.0004
a_{11}					0.0002
a_{12}					0.0001
a_{13}					0.0000
a_{14}					0.0000
a_{15}					0.0000
a_{16}					0.0000
a_{17}					0.0000
a_{18}					0.0000
a_{19}					0.0000

causal IIR Wiener filter results in a MSE of 0.375, which is approximatedly the same as the FIR filter when $N \geq 10$.

Problem 2: (c)

When $r = 1$, $N = 10$, MSE versus a are plotted in Fig. 2. The larger a is, i.e.the more $X(t)$ are related to $Y(t), \ldots, Y(t - (N-1))$, the more accurate the Wiener filter will be.

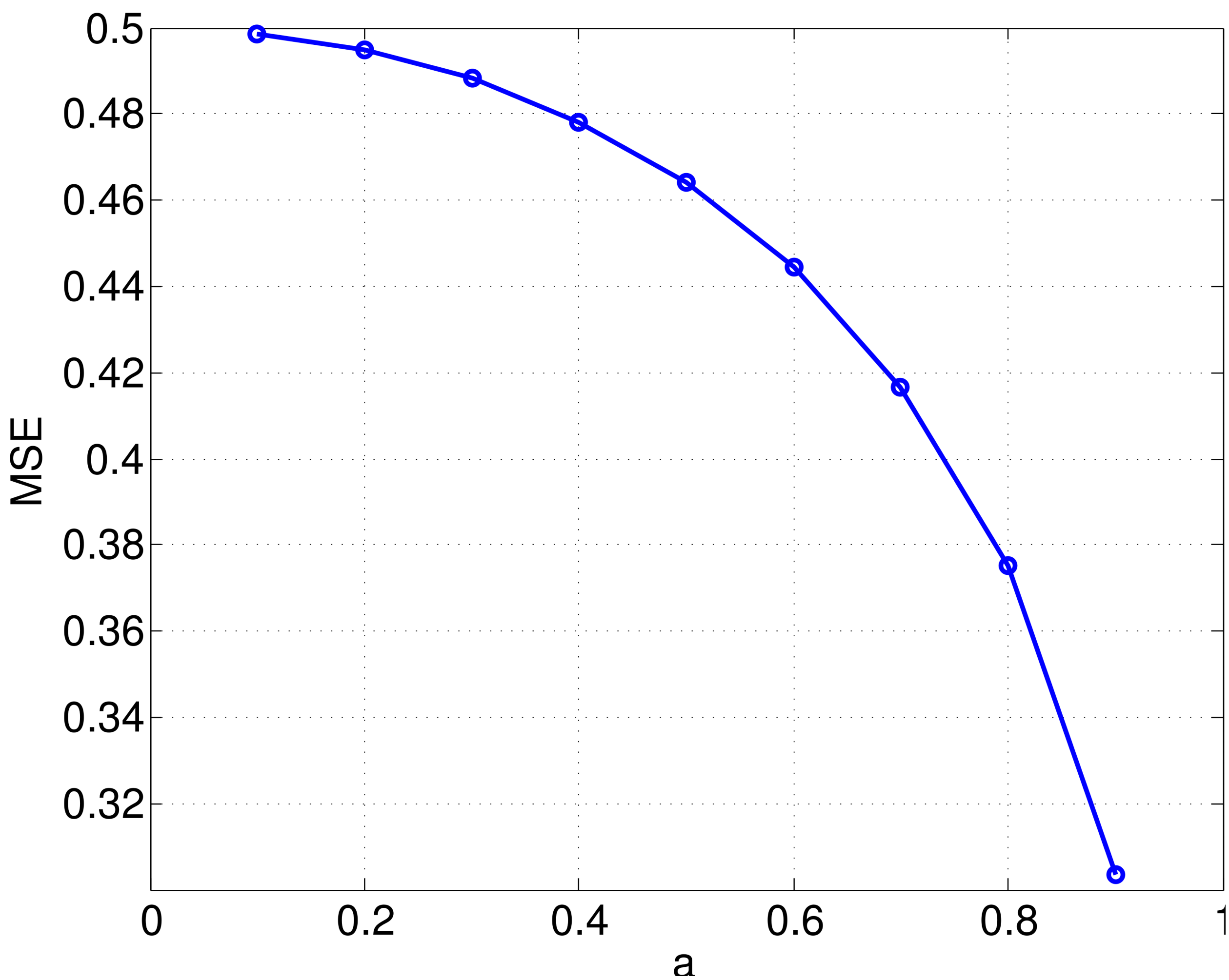

Figure 2: MSE versus a.

Problem 3

Problem 3: (a)

$X(t)$ and $Y_1(t)$ are shown in Fig. 3.

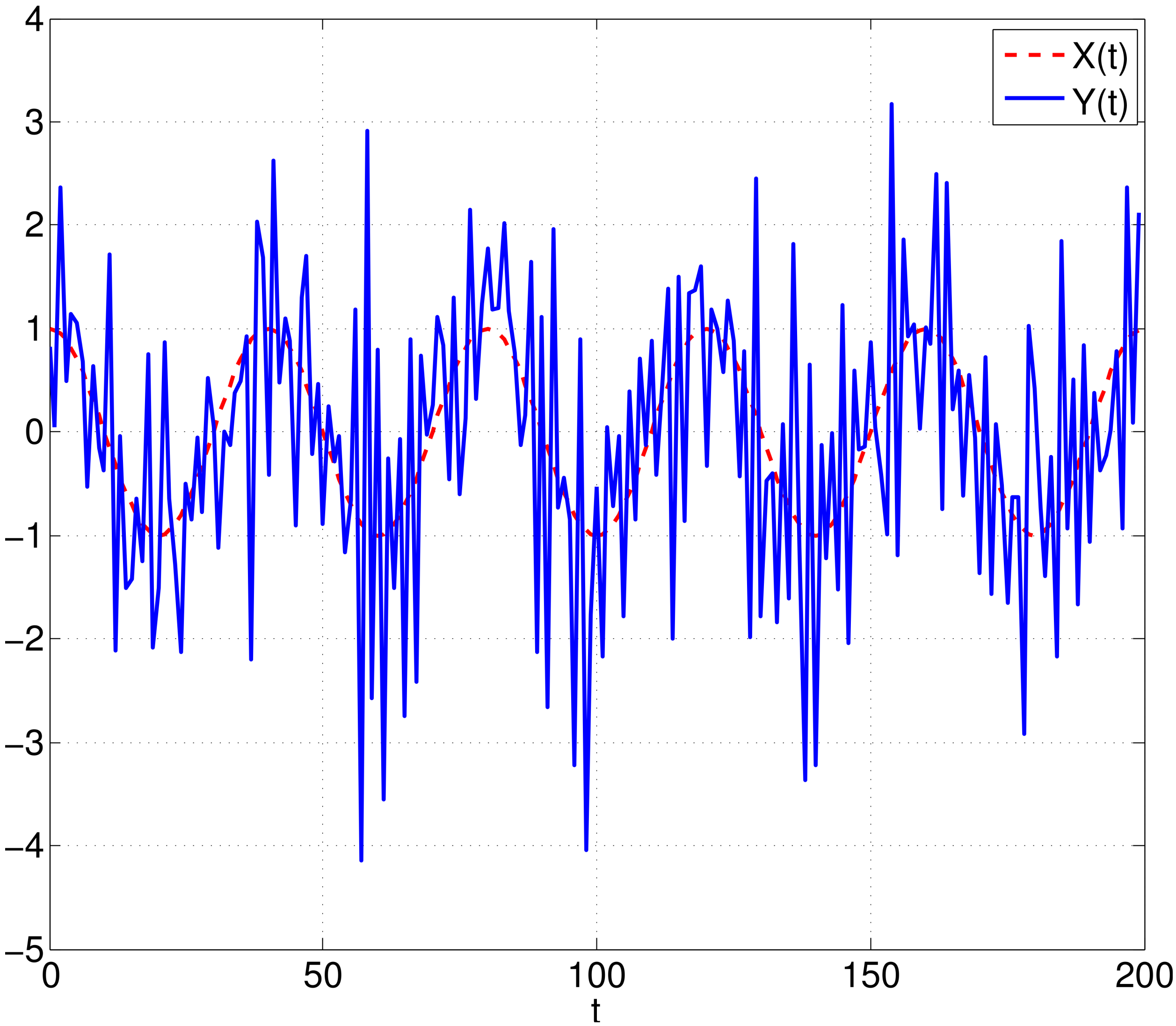

Figure 3: $X(t)$ and $Y_1(t)$.

Problem 3: (b)

Again we use Levinson recursion to get the optimum FIR filter.
noisin.m

```
function coeff_flt = filterWienerFIR(r12, r2)

M = size(r12, 2);
if (size(r2, 2) ~= M)
    error('Both r12 and r2 must be 1-by-M');
end

coeff_prd = zeros(M, 1);
coeff_flt = zeros(M, 1);

MSE_prd = r2(1);
coeff_flt(1) = r12(1) / r2(1);

for m = 0 : (M - 2)
    delta = r2((m + 2) : -1 : 2) * [1; coeff_prd(1 : m)];
    gamma = - delta / MSE_prd;
    coeff_prd(1 : m) = coeff_prd(1 : m) + gamma * coeff_prd(m : -1 : 1);
    coeff_prd(m + 1) = gamma;
    MSE_prd = MSE_prd * (1 - gamma ^ 2);

    if (MSE_prd == 0)
        break;
    end

    mu = 0 + r2((m + 2) : -1 : 2) * coeff_flt(1 : (m + 1), 1);
    omega = (r12(m + 2) - mu) / MSE_prd;
    coeff_flt(1 : (m + 1)) = coeff_flt(1 : (m + 1)) + omega * coeff_prd((m +
        1) : -1 : 1);
    coeff_flt(m + 2) = omega;
end
```

Problem 3: (c)

The noise cancellation results are shown in Fig. 4, Fig. 5 and Fig. 6.

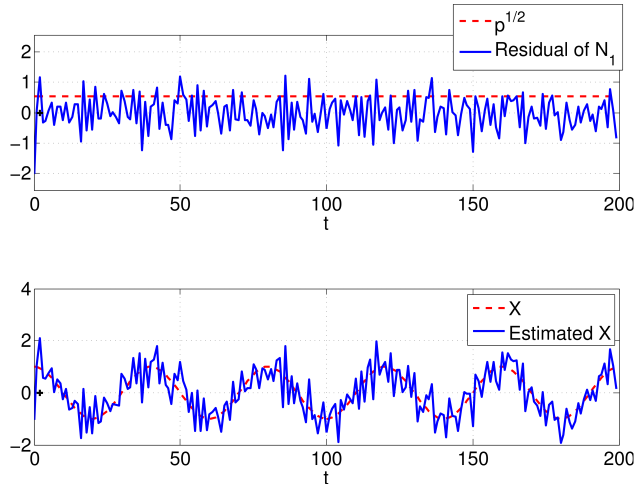

Figure 4: $M = 2$.

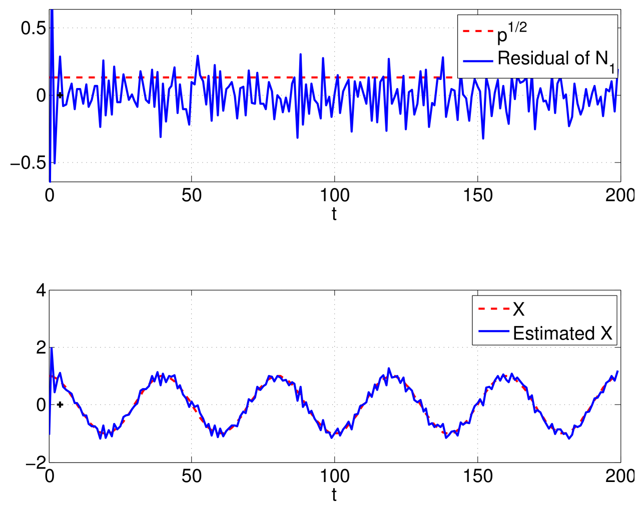

Figure 5: $M = 4$.

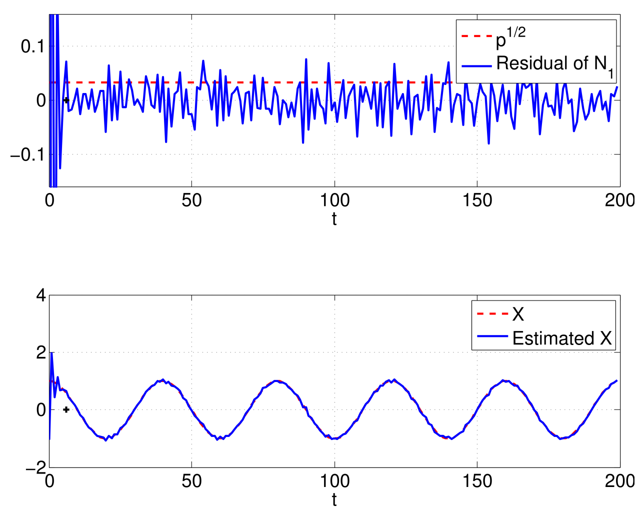

Figure 6: $M = 6$.

Problem 3 continued on next page...

Problem 3: (d)

In terms of sample MSE, the comparison between the Wiener filter computed with theoratical and sampled autocorrelation and cross correlation is shown in Table 1.

Table 1: Comparison between the MSE of the Wiener filters evaluated with theoratical and sampled autocorrelation and cross correlation.

M	Theoretical MSE	Sampled MSE(exact correlation)	Sampled MSE(sampled correlation)
2	0.28741	0.33904	0.33654
4	0.017963	0.024541	0.02542
6	0.0011227	0.0048859	0.0061211

Problem 3: (e)

The sampled MSE results of the Wiener filter evaluated with sampled correlation with channel leakage are shown in Table 2, whhich illustrate that leakage indeed results in an increase in MSE.

Table 2: Effect of channel leakage on the sampled MSE.

M	$b = 0$	$b = 0.1$	$b = 0.2$
2	0.29162	0.29371	0.29613
4	0.038405	0.040677	0.043277
6	0.023206	0.025529	0.028177