Matlab© Exercise II: Active Noise Control with an FIR filter

Ruben Biesheuvel† and Mars Geuze‡

†Student number 4076680
‡Student number 4109139

February 28, 2015

Question 1

Question 1.2

```
%% Initialization

 close all
 clear all

% Load uncorrupted signal d
% N is the number of samples (signal length)
load gong.mat;
[N,k]=size(y);
d = y;

% Generate zero-mean white noise sequence g with standard deviation 0.35
sg = 0.35;
g = sg*randn(N,1);
g = g - mean(g);

% Generate noise sequences v1 and v2
a1 = [1 -0.90]; b1 = [1 -.2];
a2 = [1 -0.95]; b2 = [1 -.3];
v1 = filter(b1,a1,g);
v2 = filter(b2,a2,g);

% Generate the corrupted signal x
x = d + v1;

% You can uncomment one of the following to inspect the signals:
% plot(d); sound(d, Fs);
% plot(v1); sound(v1);
% plot(v2); sound(v2);
% plot(x); sound(x);
```

```
%% Exercise 1: Determining the optimal FIR Wiener Filter
% Goal: reconstruct d from x and v2 by estimating v1 from v2

%n = input('Order filter: ');
% Let n vary between the desired filter orders
n = [1 2 4 6];
Stdd = zeros(4,1);
W_tot = zeros(6,4);
Sound_diff = zeros(4,length(x));
for k = 1:4
% First we determine Rv2 and Rv1v2 needed to set up the Wiener-Hopf
% Equations:
%
%  Rv2 W = Rv1v2 (=Rxv2)
%

% Calculate the first two values of rv2 (i.e. rv2(0) and rv2(1))
% using eq (3.116) from Hayes
% You can find the 'dimpulse' function on the TU computers
h = dimpulse(b2, a2, 20);
c(1,1) = b2(1)*conj(h(1)) + b2(2)*conj(h(2));
c(2,1) = b2(2)*conj(h(1));

 rv2 = zeros(200,1);
% rv2(2) = (sg^2.*(c(1,1)*a2(2)-c(2,1)));
% rv2(1) = c(1,1)*sg^2 - a2(2)*rv2(2);
rv2(1) = (sg^2*c(1,1) - a2(2)*sg^(2)*c(2,1))/(1-a2(2)^2);
rv2(2) = sg^2*c(2,1) - a2(2)*rv2(1);

% Calculate the rest of rv2 until it becomes (almost) zero
% We only determine one side of the auto-correlation function and then
% mirror, to get the double sided ACF (centered at index 200)
for i=3:200,
    rv2(i) = -1*a2(2)*rv2(i-1);
end
rv2_ds = [rv2(end:-1:1); rv2(2:end)];

% Next we determine the the cross-correlation function between v1 and v2
bb = conv(b1,a2);
aa = conv(b2,a1);
rv1v2_ds = filter(bb,aa,rv2_ds);

% Put rv2 and rv1v2 into matrix form for the Wiener-Hopf equations
Rv2 = zeros(n(k),n(k));
for i = 1:n(k)
    for j = 1:n(k)
        Rv2(i,j) = rv2_ds(200+j-i);
    end
end
Rv1v2 = zeros(n(k),1);
for i=1:n(k),
    Rv1v2(i,1) = rv1v2_ds(200+i-1);
end

% Solve for the optimal filter
W = Rv2\Rv1v2;
v1e = filter(W,1,v2);
de  = x - v1e;

% Save the necessary data necessary to evaluate the sound
W_tot(1:length(W),k) = W;
```

```
Stdd(k) = std(d-de);
Sound_diff(k,:) = de;
end
```

Table 1: Output coefficients $w(j)$ for the optimal FIR Wiener filter

	Filter order m			
Coefficient $w(j)$	1	2	4	6
$w(0)$	0.7759	0.9209	0.9935	0.9995
$w(1)$	0	-0.1623	0.0327	0.0487
$w(2)$	0	0	-0.0765	-0.0242
$w(3)$	0	0	-0.2255	-0.0514
$w(4)$	0	0	0	-0.0859
$w(5)$	0	0	0	-0.1916

Question 2

Table 2: Standard deviation σ_W between the sound $d(n)$ and the estimated sound $x(n) - \hat{v}_1(n)$

Filter order m	σ_W
1	0.2075
2	0.1996
4	0.1674
6	0.1362

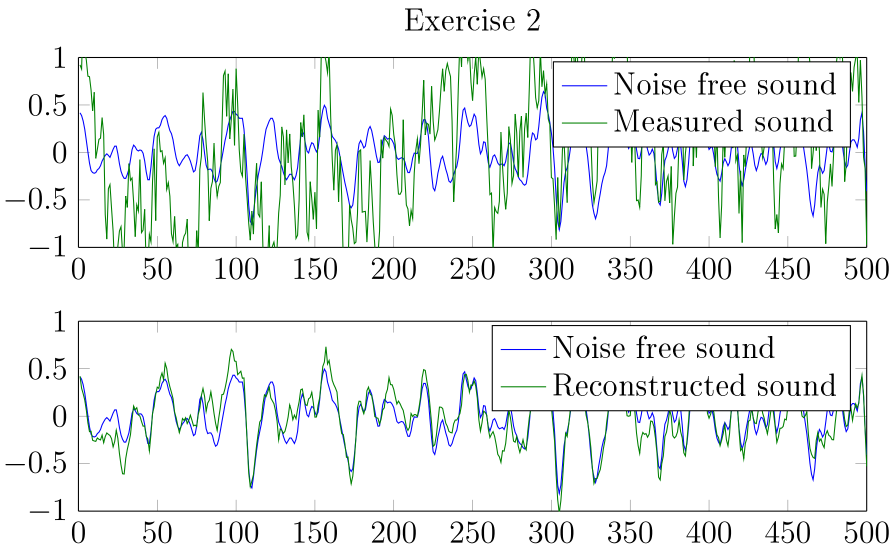

Figure 1: Subplot 1 shows the original sound and the noise corrupted signal. Subplot 2 shows the original sound and the filtered sound with optimal Wiener filter $W(z)$ of order 6

Question 3

```
%% Exercise 3: Filter by approximating the correlation functions
% Goal: Approximate the auto- and cross correlation functions used in
%       the Wiener-Hopf equations from the signals

%n = input('Order filter: ');
% Let n vary between the desired filter orders
n = [1 2 4 6];
Stdd2 = zeros(4,1);
W_tot2 = zeros(6,4);
for k = 1:length(n)
% Construct an n by (N-n+1) matrix V2 containing shifted versions of v2
V2 = zeros(n(k), N-n(k)+1);
for i=1:n(k)
    V2(i,:) = v2(n(k)+1-i:N+1-i);
end

% Use V2 and v1 to estimate Rv2 and Rv1v2
rv2e = zeros(n(k),1);
for i = 1:n(k)
    rv2e(i) = sum(V2(1,:).*V2(i,:));
end

% Put rv2e in a Toeplitz matrix like in Exercise 2
rv2e_ds = [rv2e(end:-1:1);rv2e(2:end)];
Rv2e = zeros(n(k),n(k));
for i = 1:n(k)
    for j = 1:n(k)
        Rv2e(i,j) = rv2e_ds(n(k)+j-i);
    end
end

Rv1v2e = zeros(n(k),1);
for i=1:n(k),
    Rv1v2e(i,1) = sum(x(n(k):end).*(V2(i,:).'));
end

% Calculate filter using Wiener-Hopf equations and reconstruct the signal
w = Rv2e\Rv1v2e;
v1e = filter(w,1,v2);
de = x - v1e;

% Save all the variables for the different values of n
std(d-de);
Stdd2(k) = std(d-de);
W_tot2(1:length(w),k) = w;
end
```

Table 3: Output coefficients $w(j)$ for the estimated FIR Wiener filter

	Filter order m			
Coefficient $w(j)$	1	2	4	6
$w(0)$	0.7679	0.9200	0.9960	0.9995
$w(1)$	0	-0.1689	0.0339	0.0513
$w(2)$	0	0	-0.0783	-0.0243
$w(3)$	0	0	-0.2342	-0.0548
$w(4)$	0	0	0	-0.0885
$w(5)$	0	0	0	-0.1956

Table 4: Standard deviation σ_w between the sound $d(n)$ and the estimated sound $x(n) - \hat{v}_1(n)$

Filter order m	σ_w
1	0.2177
2	0.2089
4	0.1750
6	0.1423

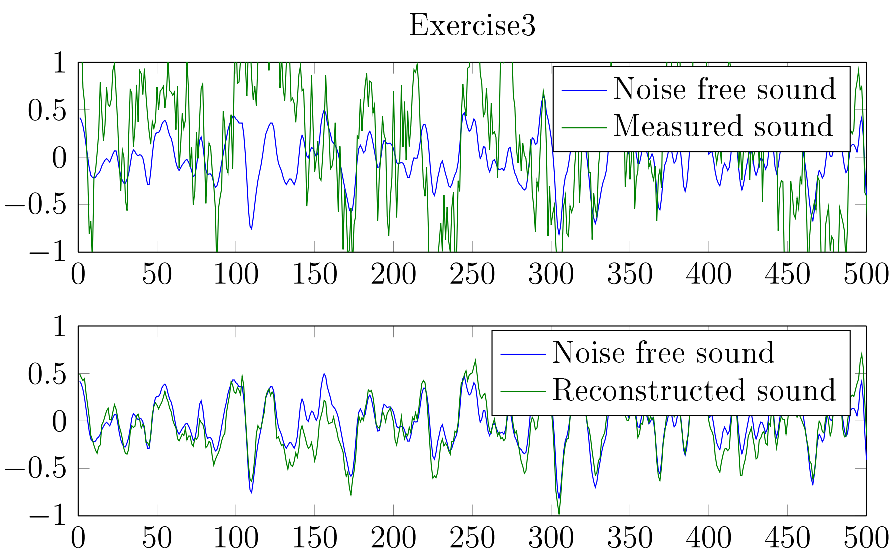

Figure 2: Subplot 1 shows the original sound and the noise corrupted signal. Subplot 2 shows the original sound and the filtered sound with estimated Wiener filter $w(z)$ of order 6