2.161 Signal Processing: Continuous and Discrete
Fall 2008

Massachusetts Institute of Technology
Department of Mechanical Engineering

2.161 *Signal Processing - Continuous and Discrete*
Fall Term 2008

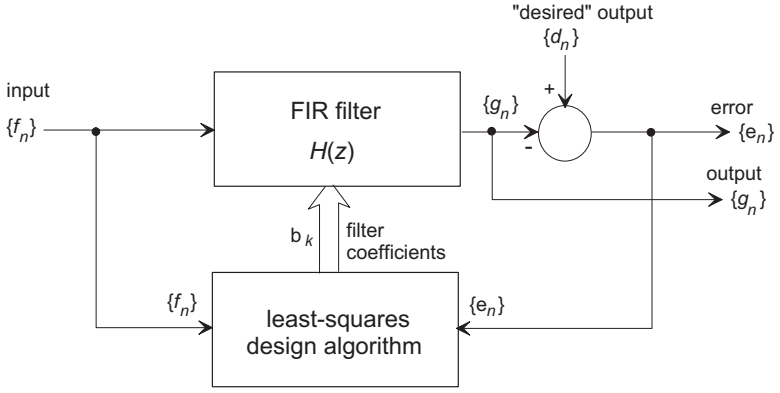
Lecture 24[1]

Reading:

- Class Handout: *MATLAB Examples of Least-Squares FIR Filter Design*
- Proakis and Manolakis: Sec. 12.3 – 12.5
- Stearns and Hush: Ch. 14

1 Least-Squares Filter Design

We now look at a FIR filter design technique that is based on "experimental" data



Given an input sequence $\{f_n\}$, and a "desired" filtered output sequence $\{d_n\}$, the task is to design a FIR filter

$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k}$$

that will minimize the error $\{e_n\} = \{d_n\} - \{g_n\}$ in some sense, where $\{g_n\}$ is the filter output. In particular, we will look at a filter design method that minimizes the mean-squared-error (MSE), where

$$\begin{aligned} \text{MSE} &= \mathcal{E}\left\{e_n^2\right\} \\ &= \mathcal{E}\left\{(d_n - g_n)^2\right\} \\ &= \mathcal{E}\left\{d_n^2\right\} + \mathcal{E}\left\{g_n^2\right\} - 2\mathcal{E}\left\{d_n g_n\right\} \end{aligned}$$

Massachusetts Institute of Technology
Department of Mechanical Engineering

2.161 *Signal Processing - Continuous and Discrete*
Fall Term 2008

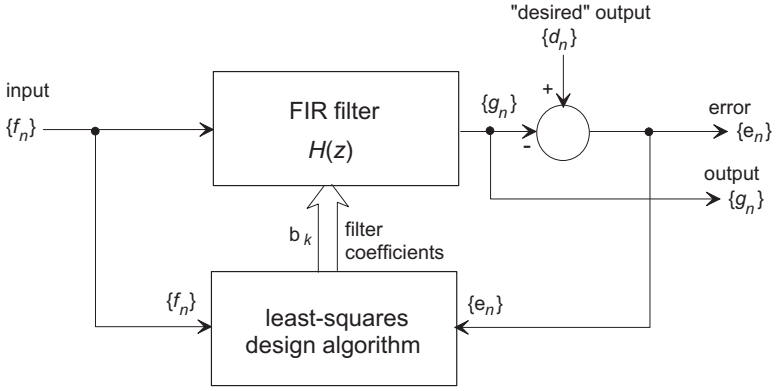
Lecture 24[1]

Reading:

- Class Handout: *MATLAB Examples of Least-Squares FIR Filter Design*
- Proakis and Manolakis: Sec. 12.3 – 12.5
- Stearns and Hush: Ch. 14

1 Least-Squares Filter Design

We now look at a FIR filter design technique that is based on "experimental" data



Given an input sequence $\{f_n\}$, and a "desired" filtered output sequence $\{d_n\}$, the task is to design a FIR filter

$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k}$$

that will minimize the error $\{e_n\} = \{d_n\} - \{g_n\}$ in some sense, where $\{g_n\}$ is the filter output. In particular, we will look at a filter design method that minimizes the mean-squared-error (MSE), where

$$\begin{aligned} \text{MSE} &= \mathcal{E}\left\{e_n^2\right\} \\ &= \mathcal{E}\left\{(d_n - g_n)^2\right\} \\ &= \mathcal{E}\left\{d_n^2\right\} + \mathcal{E}\left\{g_n^2\right\} - 2\mathcal{E}\left\{d_n g_n\right\} \end{aligned}$$

and in terms of correlation functions

$$\boxed{\text{MSE} = \phi_{dd}(0) + \phi_{gg}(0) - 2\phi_{dg}(0)}.$$

For the FIR filter with coefficients b_k, the output is

$$\{g_n\} = \{f_n\} \otimes \{b_n\}$$

and from the input/output properties of linear systems (Lec. 22)

$$\begin{aligned}\phi_{gg}(n) &= \mathcal{Z}^{-1}\left\{\left(H(z)H(z^{-1})\right) \otimes \phi_{ff}(n)\right\} \\ &= \sum_{m=0}^{M-1}\sum_{n=0}^{M-1} b_m b_n \phi_{ff}(m-n).\end{aligned}$$

Similarly,

$$\begin{aligned}\phi_{dg}(n) &= \mathcal{E}\left\{d_m g_{n+m}\right\} \\ &= \mathcal{E}\left\{d_m \sum_{k=0}^{M-1} b_k f_{n+m-k}\right\} \\ &= \sum_{k=0}^{M-1} b_k \mathcal{E}\left\{d_m f_{n+m-k}\right\} \\ &= \sum_{k=0}^{M-1} b_k \phi_{fd}(k-n)\end{aligned}$$

and

$$\phi_{dg}(0) = \sum_{k=0}^{M-1} b_k \phi_{fd}(k)$$

The MSE is therefore

$$\boxed{\text{MSE} = \phi_{dd}(0) + \sum_{m=0}^{M-1}\sum_{n=0}^{M-1} b_m b_n \phi_{ff}(m-n) - 2\sum_{k=0}^{M-1} b_k \phi_{fd}(k).}$$

We now seek the optimum coefficients.

$$\frac{\partial \text{MSE}}{\partial b_n} = 2\sum_{m=0}^{M-1} b_m \phi_{ff}(m-n) - 2\phi_{fd}(n)$$

and setting $\partial\text{MSE}/\partial b_n = 0$ for $n = 0, 1, \ldots, M-1$, we have

$$\sum_{m=0}^{M-1} b_m \phi_{ff}(m-n) = \phi_{fd}(n)$$

which is a set of linear equations in the coefficients b_k:

$$\begin{array}{lcllcl}
b_0\phi_{ff}(0) & + & b_1\phi_{ff}(1) & +\ldots+ \; b_{M-1}\phi_{ff}(M-1) & = & \phi_{fd}(0) \\
b_0\phi_{ff}(-1) & + & b_1\phi_{ff}(0) & +\ldots+ \; b_{M-1}\phi_{ff}(M-2) & = & \phi_{fd}(1) \\
b_0\phi_{ff}(-2) & + & b_1\phi_{ff}(-1) & +\ldots+ \; b_{M-1}\phi_{ff}(M-3) & = & \phi_{fd}(2) \\
\vdots & & & & & \vdots \\
b_0\phi_{ff}(-(M-1)) & + & b_1\phi_{ff}(-(M-2)) & +\ldots+ \; b_{M-1}\phi_{ff}(0) & = & \phi_{fd}(M-1)
\end{array} .$$

Recognizing that the auto-correlation function is an even function ($\phi_{ff}(-n) = \phi_{ff}(n)$), we can write the equations in matrix form

$$\begin{bmatrix}
\phi_{ff}(0) & \phi_{ff}(1) & \phi_{ff}(2) & \cdots & \phi_{ff}(M-1) \\
\phi_{ff}(1) & \phi_{ff}(0) & \phi_{ff}(1) & \cdots & \phi_{ff}(M-2) \\
\vdots & & & & \vdots \\
\phi_{ff}(M-1) & \phi_{ff}(M-2) & \phi_{ff}(M-3) & \cdots & \phi_{ff}(0)
\end{bmatrix}
\begin{bmatrix} b_0 \\ b_0 \\ \vdots \\ b_0 \end{bmatrix} =
\begin{bmatrix} \phi_{fd}(0) \\ \phi_{fd}(1) \\ \vdots \\ \phi_{fd}(0) \end{bmatrix}$$

or

$$\boxed{\mathbf{Rb} = \mathbf{P}}$$

where

$$\mathbf{R} = \begin{bmatrix}
\phi_{ff}(0) & \phi_{ff}(1) & \phi_{ff}(2) & \cdots & \phi_{ff}(M-1) \\
\phi_{ff}(1) & \phi_{ff}(0) & \phi_{ff}(1) & \cdots & \phi_{ff}(M-2) \\
\vdots & & & & \vdots \\
\phi_{ff}(M-1) & \phi_{ff}(M-2) & \phi_{ff}(M-3) & \cdots & \phi_{ff}(0)
\end{bmatrix}$$

is the *correlation matrix*,

$$\mathbf{b} = \begin{bmatrix} b_0 & b_1 & b_2 & \cdots & b_{M-1} \end{bmatrix}^T$$

are the filter coefficients, and

$$\mathbf{P} = \begin{bmatrix} \phi_{fd}(0) & \phi_{fd}(1) & \phi_{fd}(2) & \cdots & \phi_{fd}(M-1) \end{bmatrix}^T$$

is the *cross-correlation matrix*.

The MSE FIR filter coefficients are

$$\boxed{\mathbf{b} = \mathbf{R}^{-1}\mathbf{P}.}$$

Stearns and Hush show that with these coefficients

$$\boxed{(\text{MSE})_{\min} \quad = \quad \phi_{dd}(0) - \mathbf{P}^T\mathbf{b} \quad = \quad \phi_{dd}(0) - \mathbf{P}^T\mathbf{R}^{-1}\mathbf{P}.}$$

- R is a Toeplitz matrix, and efficient algorithms (Levinson-Durbin – see Proakis and Manolakis Sec. 12.4.1) exist for its inversion ($O(n^2)$).
- The development above requires that the processes $\{f_n$ and d_n are stationary.

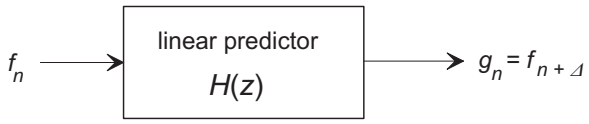
1.1 A Simple MATLAB Tutorial Least-Squares Filter Design Function

```
%------------------------------------------------------------------------
function [B,MSE] = LSQFilt(f,d,M)
%LSQFilt - Demonstration routine for Least-Squares FIR filter design
%[B,MSE] = LSQFilt(f,d,M)
%       f - rowvector of data samples    - length N
%       d - row vector of desired values - length N
%       M - filter order
% Returns:
%       B -   vector of optimal filter coefficients
%       MSE - minimized value of the mean-square-error
%
% Note: This routine is for tutorial purposes only. The Levinson method for
%       toeplitz matrix inversion would be used in practical methods.
%
% Author:  D. Rowell
% Revised: 10/29/07
%------------------------------------------------------------------------
     N = length(f);
% Compute the correlation coefficients.
% Note that matlab defines the cross-correlaton backwards!! and
% we need to reverse the order of the subscripts.
%
     phiff=xcorr(f);
     phifd=xcorr(d,f);
%
% Extract out the central regions (low-lag values) and form
% the autocorrelation matrix.
%
     rff=phiff(N:N+M-1);
     R = toeplitz(rff);
     P=phifd(N:N+M-1);
%
% Compute the optimal filter coefficients
%
     B=inv(R)*P';
%
% and the residual mean-square-error
%
     phidd=xcorr(d);
     MSE=phidd(N) - P*B;
%
%------------------------------------------------------------------------
```

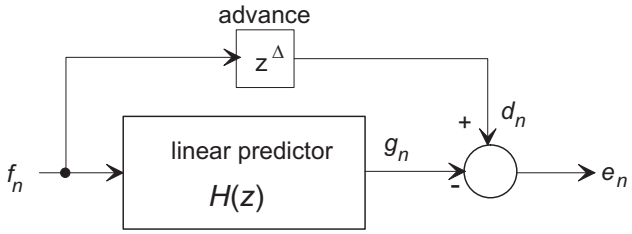
1.2 Application Examples of Least-Squares Filters

1.2.1 The Linear Predictor

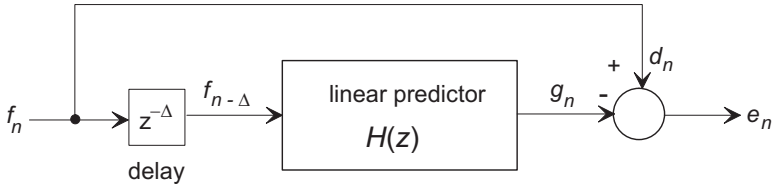
Suppose that we want a filter that will predict the value of a process $\{f_n\}$ Δ steps in the future,



The structure of least-squares filter design is



where $d_n = f_{n+\Delta}$. This system is not realizable however (because the advance block in the forward path is itself a predictor), and a practical design structure uses a delay of Δ steps in the forward path so that the filter design algorithm uses the input *history* to predict the current value.



Once the filter is designed it may be used without the delay to predict the future. The most common form is the one-step ($\Delta = 1$ predictor.

■ Example 1

Stearns and Hush (p. 346) solve the problem of a one-step linear predictor for an sinusoidal input function

$$s_n = \sin\left(\frac{2\pi n}{12}\right), \qquad n = 0, 1, 2, \ldots$$

and show that $\mathbf{b} = \left[\sqrt{(}3) \quad 1\right]^T$.

The following MATLAB code de4signs the filter using the function `LSQFilt()` described above:

```
% One-step linear predictor for a sinusoidal input.
% Define a time vector, and the input vector:
t = 0:199;
s = sin(2*pi*t/12);
% In this case the desired output is the input
d = s;
% To make it causal we must delay the input to the filter
f = zeros(1,200);
f(2:200) = s(1:199);
% Compute the filter coefficients for a first-order filter
[B1,MSE] = LSQFilt(f,d,2)
% Repeat with a second-order model
[B2,MSE] = LSQFilt(f,d,3)
```

and produces the results

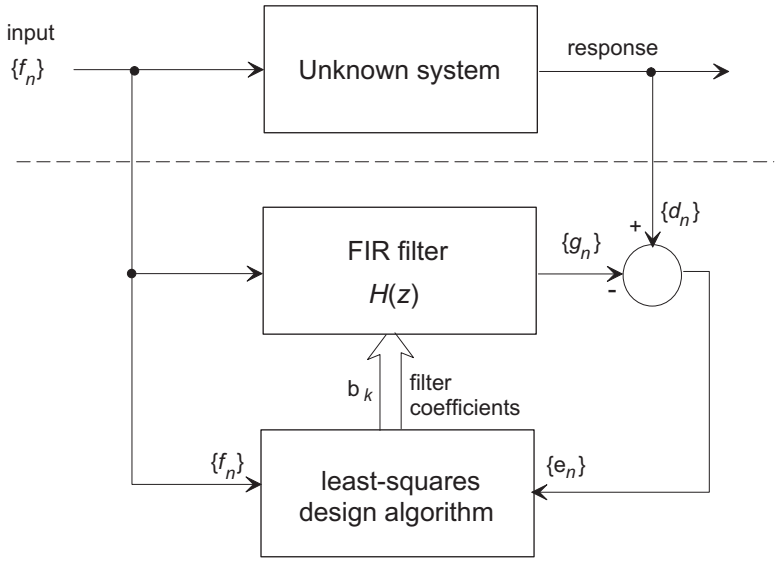
$$B1 = \begin{bmatrix} 1.73205080756888e+000 \\ -1.00000000000000e+000 \end{bmatrix}$$

$$B2 = \begin{bmatrix} 1.73205080756897e+000 \\ -1.00000000000016e+000 \\ 112.354570092066e-015 \end{bmatrix}$$

which agree with the closed-form solutions in Stearns and Hush.

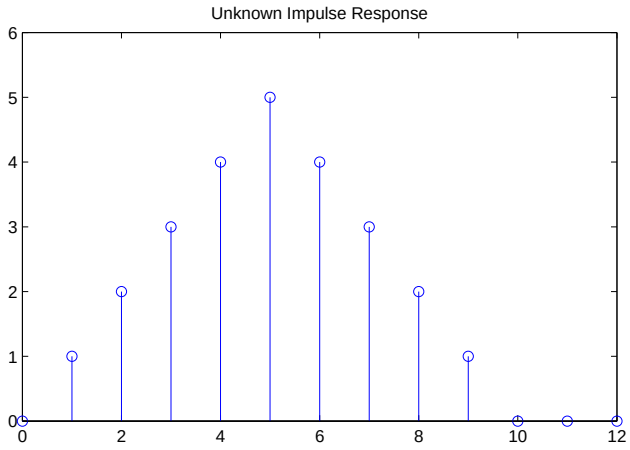
1.2.2 System Identification

Suppose we have an "unknown" FIR system and the task is to determine its impulse response. We can construct an experiment, using white noise to excite the system and record the input and output series. The least-squares filter design method is then used to determine the coefficients of a FIR are used as estimates of the plant impulse response.



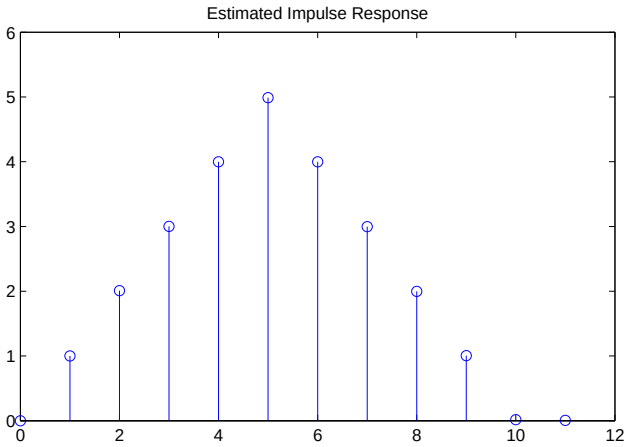
■ Example 2

Use a least-squares filter to estimate the impulse response of an "unknown" FIR system with impulse response



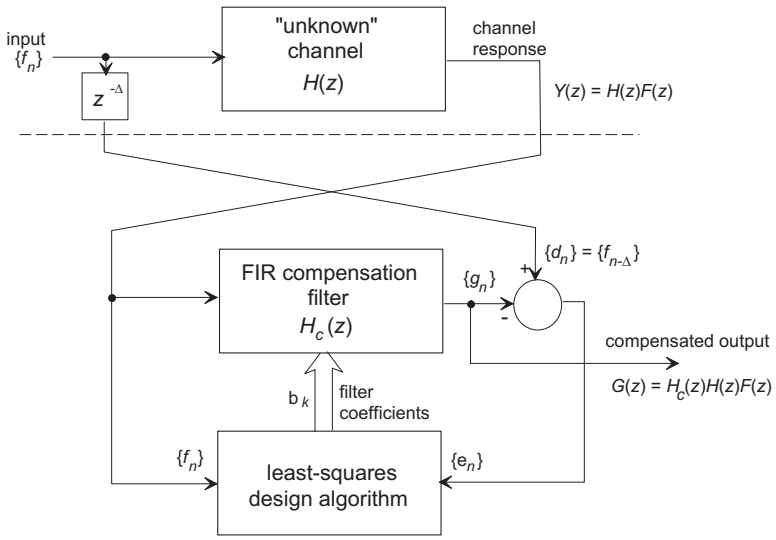

```
% System ID Using LSQFilt
% Create a FIR filter as the "unknown" plant
h = [0 1 2 3 4 5 4 3 2 1 0 0 0];
figure(1); stem(0:length(h)-1,h);
title('Unknown Impulse Response');
%
f = randn(1,1000);
% create output data representing the exerimental
% measurements
y = filter(h,1,f);
%
% Estimate the impulse response from the data
[h_opt,MSE] = LSQFilt(f,y,15);
figure(2); stem(0:length(h)-1,h_opt(1:length(h)));
title('Estimated Impulse Response');
```

giving the following result:



1.2.3 Channel Compensation

Suppose a waveform has been "corrupted" by passing through an unknown LTI filter $H(z)$, and it is desired to recover the original waveform by using a cascade compensating *inverse filter* $H_c(z) = 1/H(z)$ so that $H_c(z_H(z) = 1$. The structure is shown below, and includes an empirically chosen delay element $z^{-\Delta}$ to ensure causality of the compensating filter.



■ Example 3

Use `LSQFilt()`, with white noise as the input, to design a compensation filter for an "unknown" recursive filter.

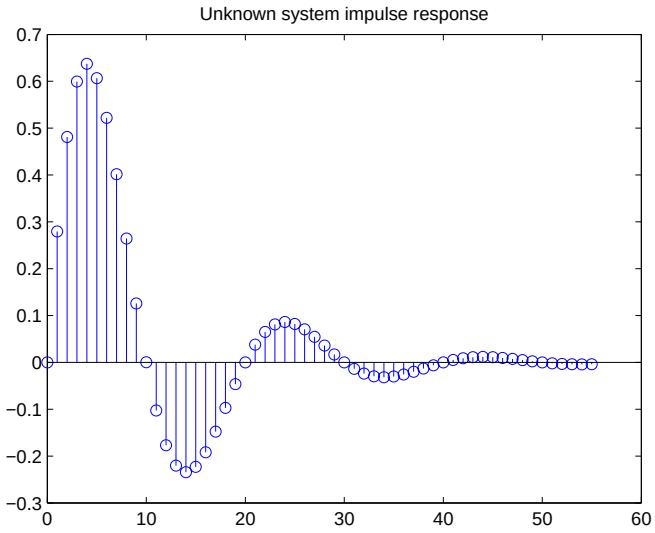
$$H(z) = \frac{0.296}{1 - 1.721z^{-1} + 0.8187Z^{-2}}.$$

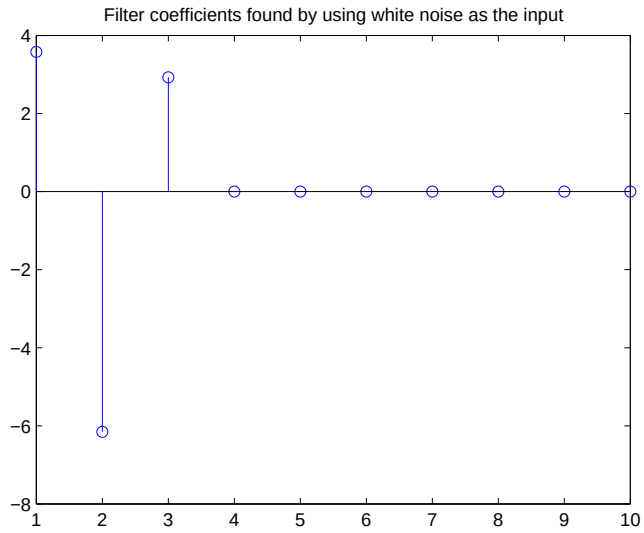
Plot the resulting impulse response of the compensated channel.

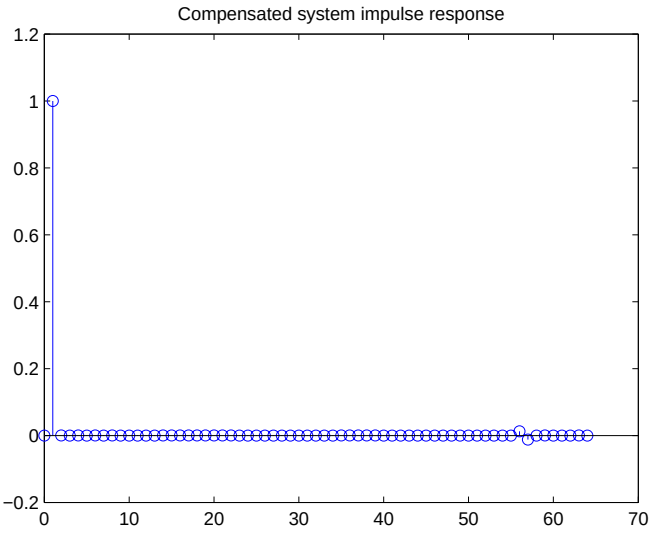
Solution: The following MATLAB scipt uses a filter length $M = 10$, and a delay factor $\Delta = 1$.

```
% Channel compensation using Least-Squares filter design.
% Input the numerator B(z), and denominator A(z) of the "unknown" plant
B_u = [0.2796 0];
A_u = [1 -1.7211 0.8187];
unknown = tf(B_u, A_u ,1);
h_unknown = impulse(unknown);
figure(1), stem(0:length(h_unknown)-1, h_unknown)
title('Unknown system impulse response');
%
% Use white noise as the input signal:
s = randn(1,200);
f = lsim(unknown,s);
% Create the desired output
$ (delay by one step so that resulting filter is causal)
d = zeros(1,200);
d(2:200) = s(1:199);
[B_noise,MSE] = LSQFilt(f,d,10);
figure(2), stem(B)
title('Filter coefficients found by using white noise as the input')
h_compensated = conv(h_unknown, B);
figure(3), stem(0:length(h_compensated)-1, h_compensated)
title('Compensated system impulse response')
```

The output is shown below.





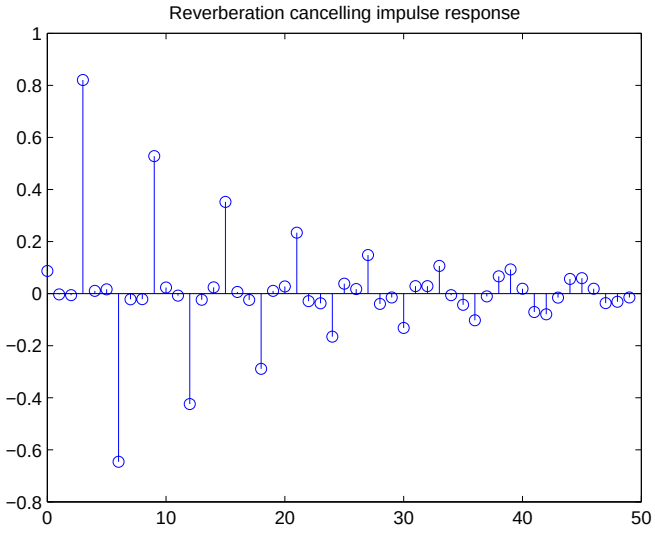


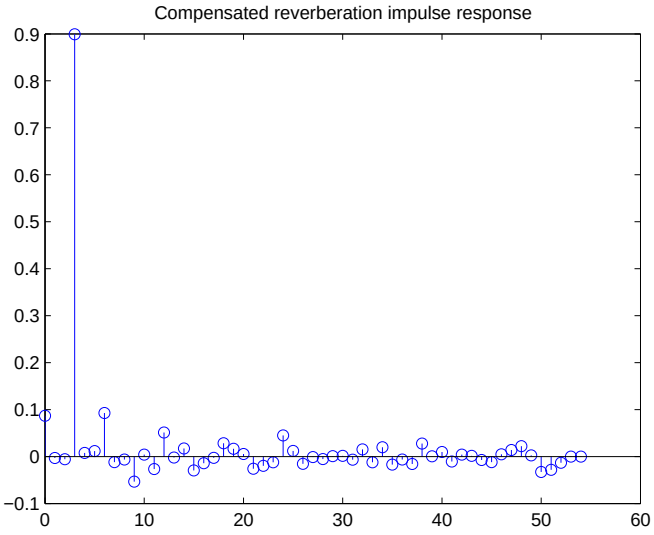
■ Example 4

The following is another example of channel compensation, this time where the channel is corrupted by an echo. For clarity we use a simple "strong" echo, so that the waveform is $y_n = f_n + 0.9f_{n-3}$. White noise is used as the excitation, and a filter length $M = 50$, and a delay $\Delta = 4$ are chosen. Notice that complete echo suppression is not possible with a finite length FIR filter.

```
% Model the environment as a nonrecursive filter:
```

```
b = [1 0 0 0.9 0 0];
% Use white noise as the excitation
s = randn(1,200);
f = filter(b,1,s);
% The desired output is a delayed version of the input:
d = zeros(1,200); d(4:200) = s(1:197);
% Design the filter
[B,MSE] = LSQFilt(f,d,50);
figure(1), stem(0:length(B)-1, B)
title('Reverberation cancelling impulse response')
% Find the overall compensated system impulse response
h_comp = conv(B,b);
figure(2), stem(0:length(h_comp)-1, h_comp)
title('Compensated reverberation impulse response')
```




Compensated reverberation impulse response
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
−0.1
0
10
20
30
40
50
60