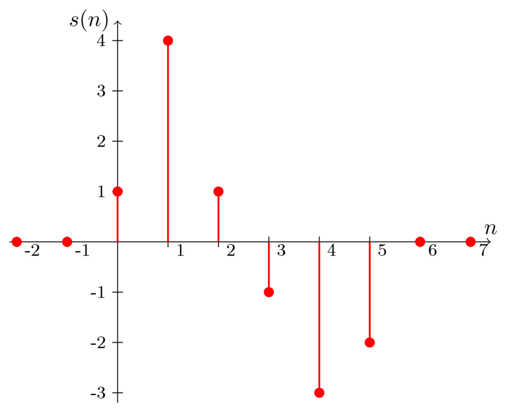
Homework 9

Table of Contents

```
figure('position', [0, 0, 800, 300])
```

[✓] ADSI Problem 6.1: Matched filters

Consider a deterministic signal that is only non-zero in the vicinity of n=0 as shown below and zero for all other values of n.



The signal is disturbed by additive noise from an AR(3) process given by

$$v(n) = -0.5v(n-1) - 0.5v(n-2) - 0.25v(n-3) + w(n)$$

where $w(n) \sim WGN(0, 4)$.

The goal of this problem is to construct a matched filter that can help us distinguish between the presence or absence of the signal in the noise.

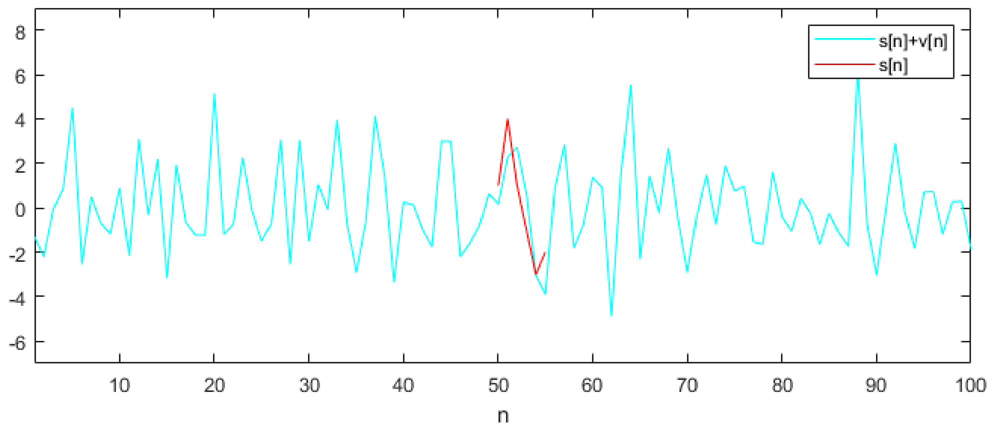
```
clear variables;
```

[✔] 1) Plot a realisation of the signal with the added noise

Create a realization of the noise and add the signal somewhere in the noise. Plot the result and comment on whether the signal is detectable.

```
D = 50; % The place to embed signal into the noise


% Generate the deterministic signal s[n]
s = [1, 4, 1, -1, -3, -2]';

% Generate white noise w[n] with zero mean and variance 4
w_var = 4;
w = sqrt(w_var) * randn(10000, 1);

% Generate noise from an AR(3) process
b = 1;
a = [1, 0.5, 0.5, 0.25];
v = filter(b, a, w);

% Extract 100 samples to remove the transient effect
% at the beginning and at the end of the filtered signal
x = v(500:600);

% Embed the signal s[n] in v[n] from D
s_in_x_start = D;
s_in_x_end =   D + length(s) - 1;
```

```
x(s_in_x_start:s_in_x_end) = x(s_in_x_start:s_in_x_end) + s;

plot(1:length(x),       x, 'c', ...
     s_in_x_start:s_in_x_end, s, 'r') % the signal is completely invisible
legend('s[n]+v[n]', 's[n]')
xlabel('n')
xlim([1,100])
ylim([-7, 9])
```



The signal is cannot be distinguished from the noise.

[✔] 2) Plot the power density spectrum of the noise and spectrum of the signal

Plot the power density spectrum of the noise and spectrum of the signal, see Eq. (14.91).

The noise is generated by an AR(3) process plus some white Gaussian noise:

$$v(n) = -0.5v(n-1) - 0.5v(n-2) - 0.25v(n-3) + w(n)$$

$$\text{where } w(n) \sim WGN(0,4).$$

We can determine the Power Density Spectrum of an ARMA(p,q) process is given by

$$S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2 = \sigma_x^2 \left| \frac{\sum_{k=0}^{q} b_k e^{-j\omega k}}{1 + \sum_{k=1}^{p} a_k e^{-j\omega k}} \right|^2 . \qquad (13.133)$$

The power spectrum of an AR(p) process is given by:

$$S_{yy}(\omega) = \sigma_x^2 \left| \frac{1}{1 + \sum_{k=1}^{p} a_k e^{-j\omega k}} \right|^2$$

For this problem, we have:

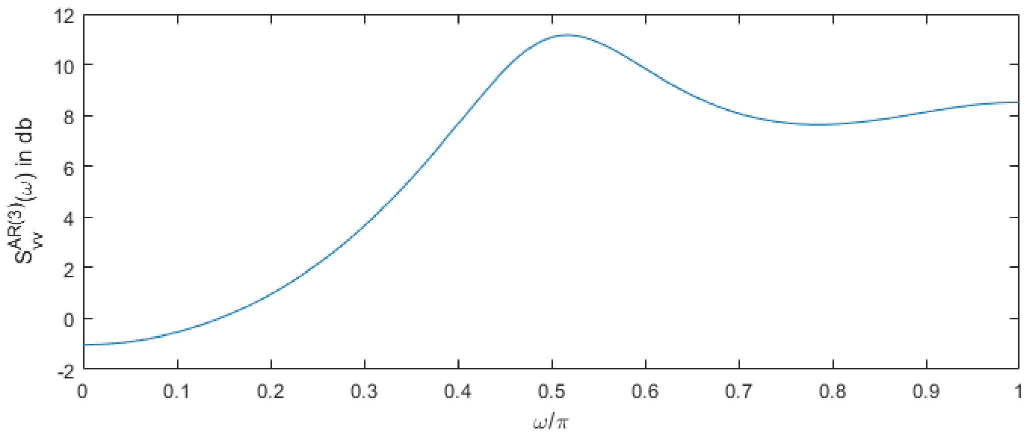
$$S_{vv}(\omega) = 4 \left| \frac{1}{1 + 0.5e^{-j\omega} + 0.5e^{-j2\omega} + 0.25e^{-j3\omega}} \right|^2$$

The algorithm is as follows:

1. Use the coefficents $\{a_1, a_2, \cdots, a_p\}$ for the $\mathrm{AR}(p)$ model,
2. Compute the transfer function for the $\mathrm{AR}(p)$ by computing the sum and finding its reciprocal
3. Compute the conjugate of the transfer function: $|H(e^{j\omega})|^2$
4. Multiply it with the variance σ_x^2

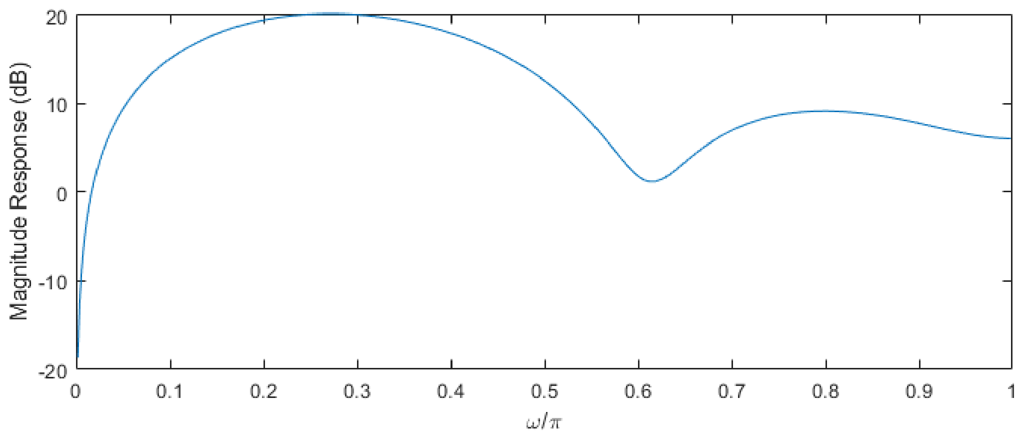
The algorithm is implemented in the functions `ar2psd()` function:

```
N = 256;

a = [0.5, 0.5, 0.25]; % The coefficients of the AR(3) model
w_var = 4; % The variance of white noise
[S_vv, w] = ar2psd(a, w_var, N); % Compute the PDS of AR(3) model

plot(w/pi, pow2db(S_vv))
xlabel('\omega/\pi')
ylabel('S_{vv}^{AR(3)}(\omega) in db')
```



We can plot the spectrum of the signal $s(n)$ using the `freqz()` function:

```
[H,w2] = freqz(s, 1);
plot(w2/pi, 20*log10(abs(H)))
xlabel('\omega/\pi')
ylabel('Magnitude Response (dB)')
```



[✓] 3) Calculate the matched filter

Calculate the matched filter, the frequency response of the matched filter and the optimum signal to noise ratio. You can use `xcorr` *on the noise realization to find* $\boldsymbol{R}_v$*.*

The impulse response of the matched filter is given by:

$$\boldsymbol{h} = \kappa \boldsymbol{R}_v^{-1} \boldsymbol{s}. \qquad (14.97)$$

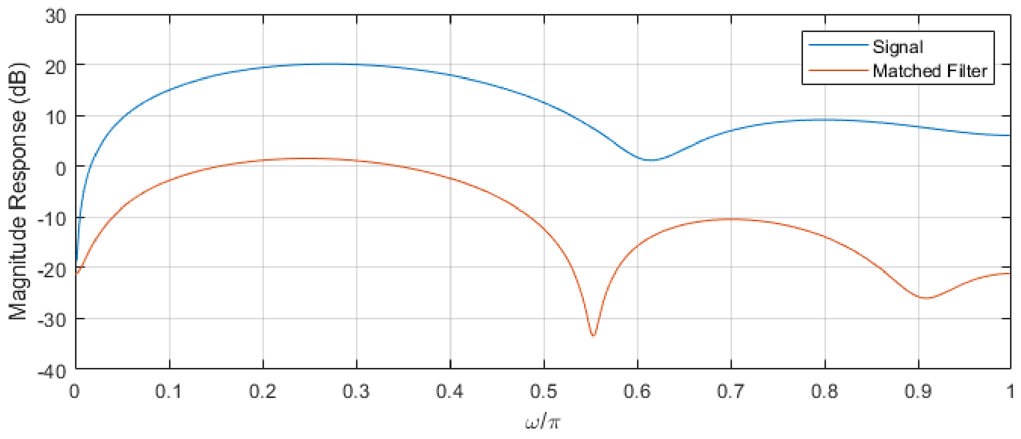
where k is the normalisation factor. Although the maximum SNR can be obtained by any choise of constant κ, we choose the constant by requiring that:

(a) $\boldsymbol{h}^T \boldsymbol{s} = 1,$ which yields $\kappa = 1/\boldsymbol{s}^T \boldsymbol{R}_v^{-1} \boldsymbol{s}$

(b) $E(v_o^2[n]) = \boldsymbol{h}^T \boldsymbol{R}_v \boldsymbol{h} = 1,$ which yields $\kappa = 1/\sqrt{\boldsymbol{s}^T \boldsymbol{R}_v^{-1} \boldsymbol{s}}.$

```matlab
p = numel(s); % Signal length

% The autocorrelation matrix must be p x p since
% its inverse is multiplied by a p-tap signal s(n)
[r_vv, ~] = xcorr(v, p-1, 'biased');
R_vv = toeplitz(r_vv(p:end));

% Compute normalisation factor
k = 1/sqrt(s'*(R_vv\s)); % Same as 1/sqrt(s'*inv(R_vv)*s)

% Compute the filter
h = k*(R_vv\s); % Same as k*inv(R_vv)*s

[H_s, w_s] = freqz(s, 1);
[H_h, w_h] = freqz(h, 1);
plot(w_s/pi, 20*log10(abs(H_s)), w_h/pi, 20*log10(abs(H_h)))
legend('Signal', 'Matched Filter')
xlabel('\omega/\pi')
```

```
ylabel('Magnitude Response (dB)')
grid on;
```



The maximum possible value of the output SNR is given by:

$$\mathrm{SNR_o} = a^2\tilde{\boldsymbol{s}}^{\mathrm{T}}\tilde{\boldsymbol{s}} = a^2\boldsymbol{s}^{\mathrm{T}}\boldsymbol{R}_v^{-1}\boldsymbol{s}. \qquad (14.98)$$

Since the attenuation factor a is not given in this problem, we assume that it is 1:

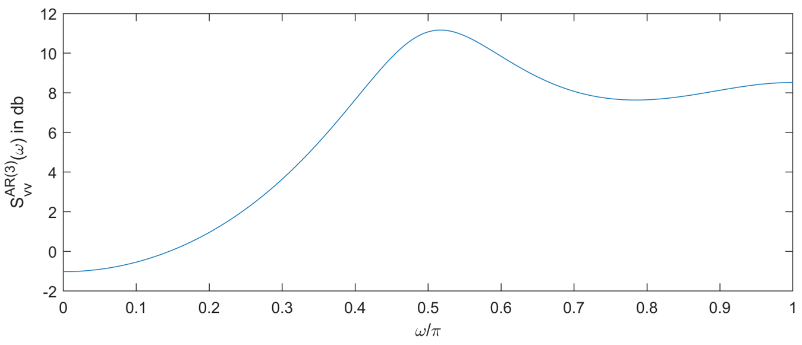
```
a = 1;
SNR = a^2 * s' * (R_vv\s)
```

```
SNR = 11.5754
```

[�] 4) Does the frequency response of the matched filter make sense?

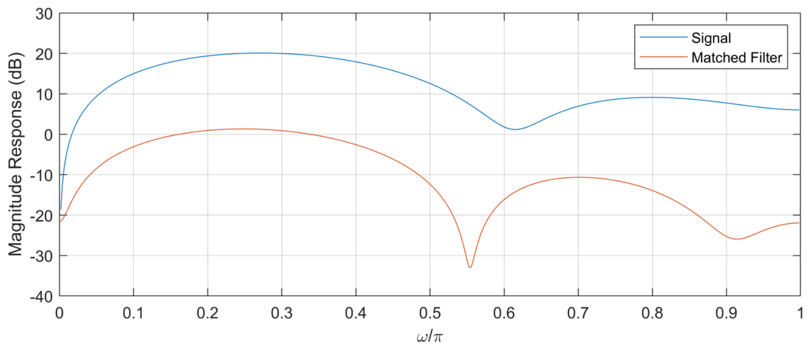
Does the frequency response of the matched filter make sense seen in relation with the spectra from question 2?

In question 2) we found that the AR(3) noise model has peak power at 0.5.



Looking at the frequency response of the signal, we observe that it peaks around 0.25 and dips around 20 dB at around 0.6.

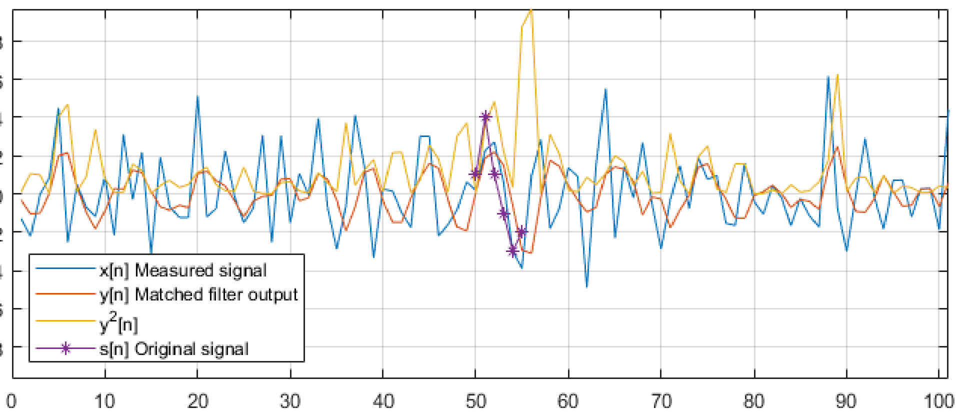
The frequency response of the matched filter shows that it is minimising the noise in the region at around 0.55. The noise power peaks at around 0.5



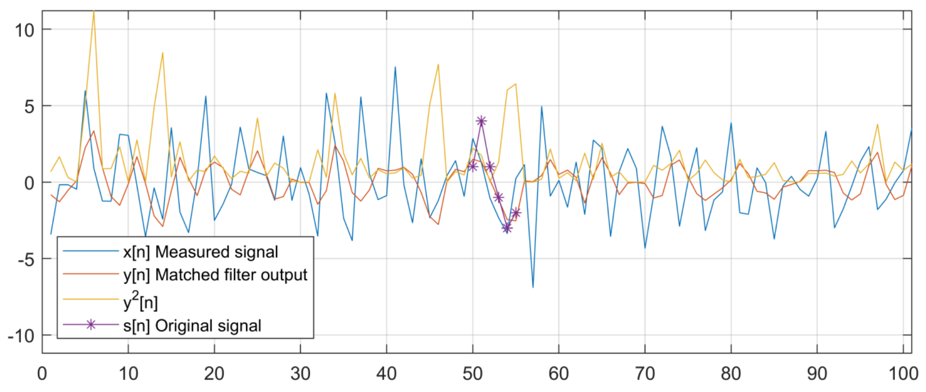
[✔] 5) How efficient is the matched filter?

Plot the signal before and after the matched filter as well as the square of the output of the matched filter on the same graph and comment on the effiency of the matched filter in our quest to detect the presence of the signal.

```
% x(n) is the 100 samples extracted from the noise signal v(n)
% y(n) is the result of feeding x(n) to the matched filter
y = filter(h, 1, x);
y_squared = y.^2;

n = 1:length(x);

n2 = s_in_x_start:s_in_x_end;
xs = s;

plot(n,x, n,y, n,y_squared,  n2,xs,'*-')
xlim([0, numel(n)])
ylim([-max(y_squared), max(y_squared)])
legend({'x[n] Measured signal', 'y[n] Matched filter output', 'y^2[n]', 's[n] Original signal'
    'Location','southwest')

grid on;
```



It's close to impossible to find the orignal signal $s[n]$ in the measured signal $x[n]$. It is also difficult to detect $s[n]$ after filtering $x[n]$ with the matched filter i.e., $y[n]$. It is easier to detect the original signal in the square of the output of the matched filter $y^2[n]$. Running the code multiple times i.e., creating different realisations of the noise, we may see many false positives. The figure below shows an example of this. The highest response is measured at $n = 6$ but we know that the signal is embedded in $n = 50$.



[✓] 6) What happens to the optimum SNR when the noise is twice as powerful?

Is the optimum signal to noise ratio halved if the AR(3) is instead driven by twice as powerful white noise, i.e. w(n) ~ WGN(0,8)? Why or why not?

We will compute the SNR again but this time with $\sigma_w^2 = 8$:

$$\mathrm{SNR_o} = a^2\tilde{\boldsymbol{s}}^\mathrm{T}\tilde{\boldsymbol{s}} = a^2\boldsymbol{s}^\mathrm{T}\boldsymbol{R}_v^{-1}\boldsymbol{s}. \qquad (14.98)$$

```
% Generate white noise w[n] with zero mean and variance 8
w2_var = 8;
w2 = sqrt(w2_var) * randn(10000, 1);

% Put the white noise through an AR(3) process
b = 1;
a = [1, 0.5, 0.5, 0.25];
```

```
v2 = filter(b, a, w2);

% Extract 100 samples to remove the transient effect
% at the beginning and at the end of the filtered signal
x2 = v2(500:600);

p = numel(s); % Signal length

% The autocorrelation matrix must be MxM since
% its inverse is multiplied by a M-tap signal s(n)
[r_vv2, lags] = xcorr(x2, p-1, 'biased');
R_vv2 = toeplitz(r_vv2(p:end));

a = 1;
SNR2 = a^2 * s' * (R_vv2\s)
```

```
SNR2 = 6.6825
```

```
SNR2/SNR
```

```
ans = 0.5773
```

The optimal signal-to-noise-ratio is almost halved when the white noise is doubled. The reduction of SNR is not always decreased by a factor of though. Although the noise power is doubled, the shape of the optimum filter will change because it depends on the measured signal:

$$\boldsymbol{h} = \kappa \boldsymbol{R}_v^{-1} \boldsymbol{s}. \qquad (14.97)$$

[✓] 7) Compute SNR using a 6-tap moving average filter

Assume that instead of the matched filter a simple 6-tap moving average filter is used instead. That is

$$\boldsymbol{h} = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}^T$$

Calculate the signal to noise ratio when the moving average filter is used.

We can compute the SNR using the following formula:

$$\text{SNR}_\text{o} = \frac{s_\text{o}^2[n_0]}{\text{E}(v_\text{o}^2[n_0])} = a^2 \frac{(\boldsymbol{h}^\text{T}\boldsymbol{s})^2}{\boldsymbol{h}^\text{T}\boldsymbol{R}_v\boldsymbol{h}}. \qquad (14.94)$$

```
% h_avg = [1, 0, 1, 1, 1, 1]'./5;
h_avg = [1, 1, 1, 1, 1, 1]'./6;
a = 1;
SNR_avg = a^2 * (h_avg'*s)^2 / (h_avg'*R_vv*h_avg)
```

```
SNR_avg = 0
```

The SNR of the moving average filter is zero (or a number very close to zero). The moving average filter gives us the average of a signal. Since the original signal has zero mean, the average is zero. This is not surprising. If we change the filter, then the SNR is no longer zero.

```
h4 = [1, 0, 1, 1, 1, 1]'./5;
a = 1;
SNR4 = a^2 * (h4'*s)^2 / (h4'*R_vv*h4)
```

```
SNR4 = 1.4075
```

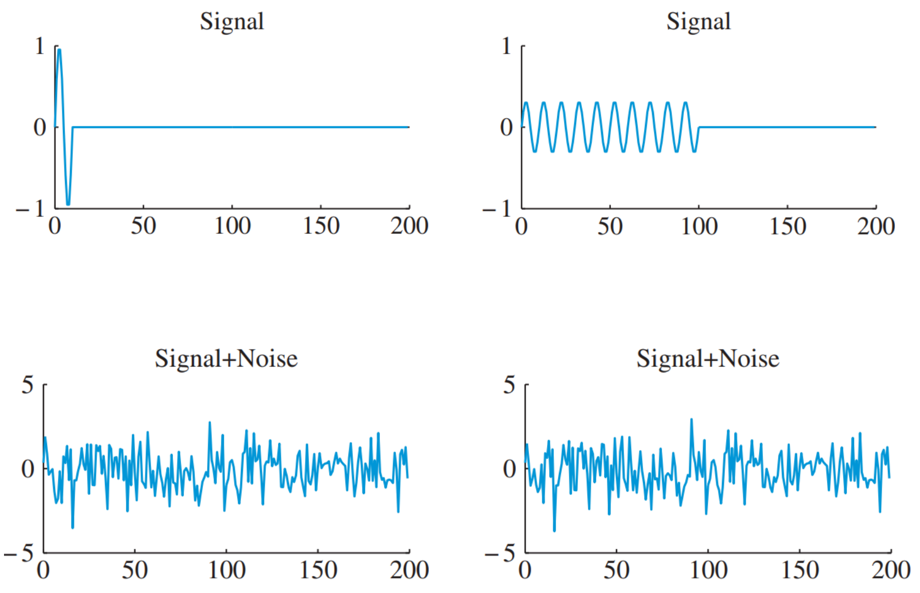
Problem 14.14: Matched filters, compare short vs long signals

```
clear variables;
```

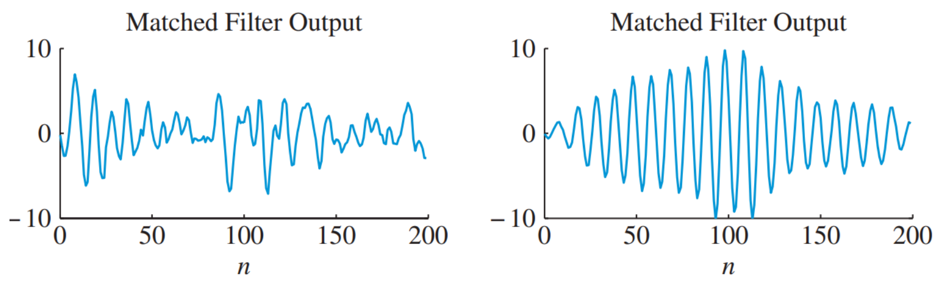
In this problem we discuss in detail the matched filtering problem illustrated in Figure 14.18.

Figure 14.18 shows the operation of a matched filter in white noise.

The two signals have different lengths ($p = 10$ and $p = 100$, respectively) but the same energy.



Notice that the peak response of the matched filter occurs at $n_0 = p - 1$ because there is no time delay i.e. $D = 0$. Normally, the output SNR is maximised at time $n_0 = p + D - 1$. In practice, finding the time delay D depends on the particular application.



The input to the matched filter is given by

$$x[n] = s_I[n] + v[n] = a\,s[n-D] + v[n]$$

where

- a is the attenuation factor
- D is the round-trip delay
- $v[n]$ is the white Gaussian noise with zero mean and unit variance
- $s_I[n] = 0$ outside the interval $0 \le n \le p-1$

[�] 1) Determine the impulse response of the matched filter and the output SNR.

Determine the impulse response of the matched filter and the output SNR. Explain why, in the absence of noise, the output is the ACRS of the desired signal.

[�] To compute the matched filter and the output SNR, we need to know something about the signal $s[n]$ and the noise.

[�] 1.1) *Explain why, in the absence of noise, the output is the ACRS of the desired signal.*

In case of white noise, the impulse response of the matched filter $h[n]$ is equal to the signal that we want to detect $s_I[n]$. As such the output of the matched filter is the same as autocorrelation:

[✓] 2) Determine the matched filter of a short cosine signal

Suppose that $p = 10$ *and* $s_i[n] = \cos\left(2\pi \frac{n}{10}\right)$. *Generate* $N = 200$ *samples of the noisy signal* $s[n]$ *and process it through the matched filter designed for* $s_i[n]$. *Plot the desired, input, and filtered signals and determine when the matched filter output is output.*

The impulse response of the matched filter is given by:

$$\boldsymbol{h} = \kappa \boldsymbol{R}_v^{-1} \boldsymbol{s}. \qquad (14.97)$$

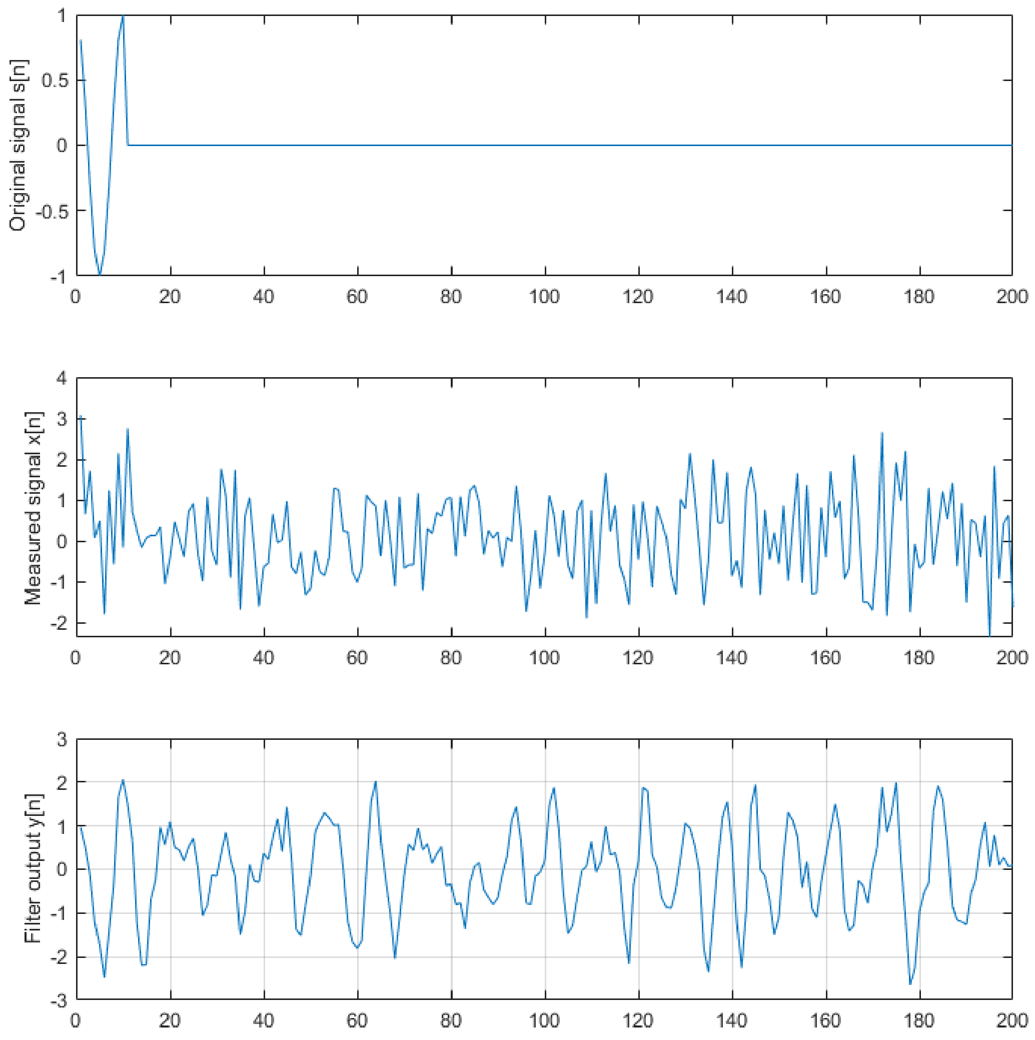
where k is the normalisation factor and $\boldsymbol{R}_v$ is the autocorrelation matrix of the noise.

The output SNR can be computed using the formula:

$$\mathrm{SNR_o} = a^2 \tilde{\boldsymbol{s}}^{\mathrm{T}} \tilde{\boldsymbol{s}} = a^2 \boldsymbol{s}^{\mathrm{T}} \boldsymbol{R}_v^{-1} \boldsymbol{s}. \qquad (14.98)$$

The computation is coded in a MATLAB function named `matched_filter` (see the end of the document)

```
N = 200;
n = (1:N)';

p = 10; % Signal length

% Generate the attenuated signal.
% Ensure that s[n] is zero when n is outside the interval [1, p]
s = zeros(N, 1);
inside_indices = (n >= 1 & n <= p);
s(inside_indices) = cos(2*pi*n(inside_indices)/10);

% Generate the zero-mean WGN with unit variance
v = randn(N, 1);

% Generate the measured signal x[n]
x = v + s;

% Compute the impulse response of the matched filter
[h, SNR] = matched_filter(s(1:p), v);

% Feed the measured signal to the matched filter
y = filter(h, 1, x);

figure('position', [0, 0, 800, 800])
subplot(3,1,1); plot(n,s); ylabel('Original signal s[n]')
subplot(3,1,2); plot(n,x); ylabel('Measured signal x[n]')
subplot(3,1,3); plot(n,y); ylabel('Filter output y[n]')
xlim([0 N])
grid on;
```



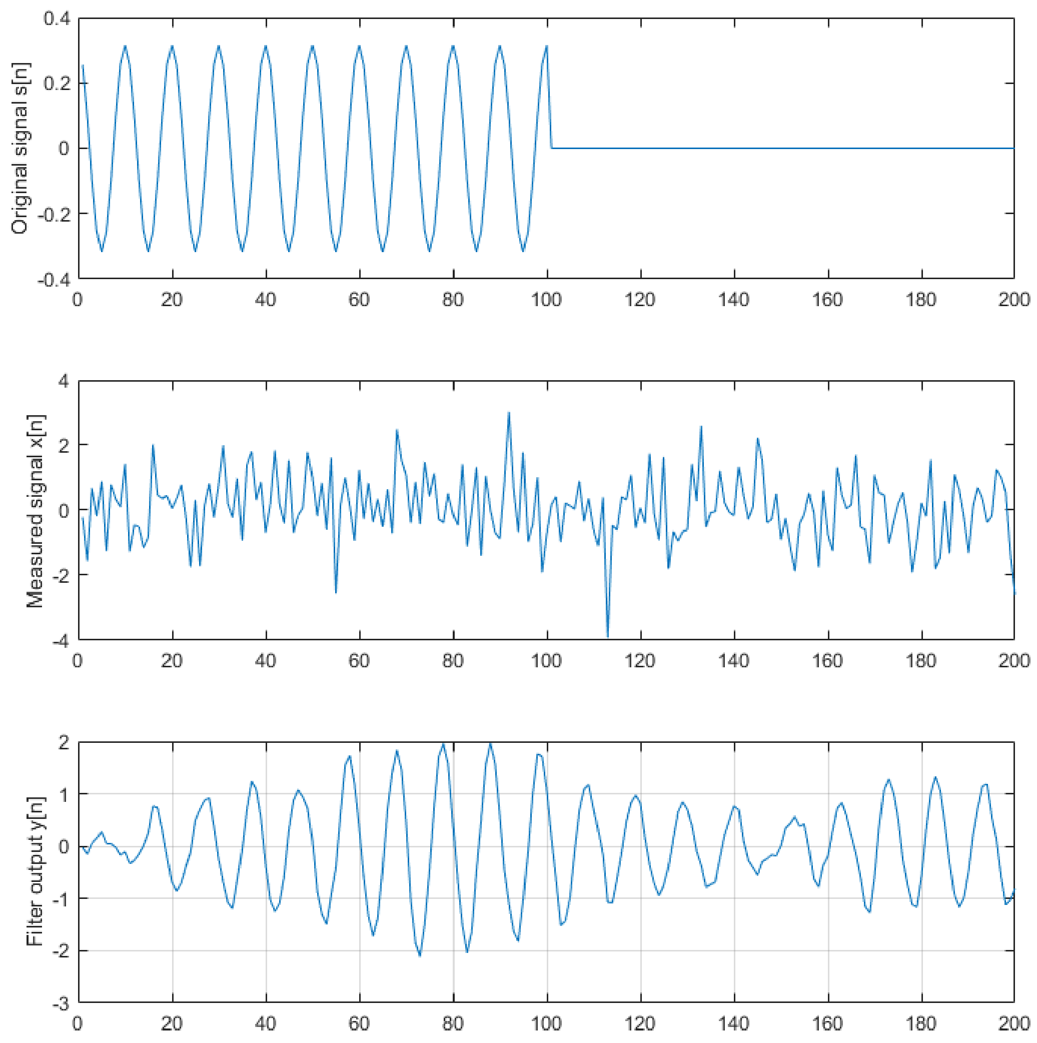
From the filtered output, it is difficult to determine where the original signal is placed.

[✔] 3) Determine the matched filter of a long cosine signal

Repeat 2) for $p = 100$ *and* $s_i[n] = \frac{1}{\sqrt{10}}\cos\left(2\pi\frac{n}{10}\right)$.

```
p = 100; % Signal length

% Generate the attenuated signal.
% Ensure that s[n] is zero when n is outside the interval [1, p]
```

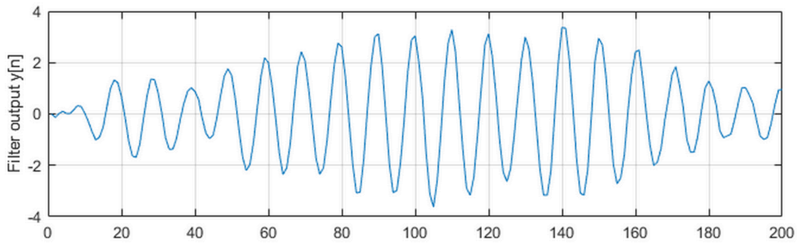
```
s = zeros(N, 1);
inside_indices = (n >= 1 & n <= p);
s(inside_indices) = (1/sqrt(10)) * cos(2*pi*n(inside_indices)/10);

% Generate the zero-mean WGN with unit variance
v = randn(N, 1);

% Generate the measured signal x[n]
x = v + s;

% Compute the impulse response of the matched filter
[h_100, SNR_100] = matched_filter(s(1:p), v);

% Feed the measured signal to the matched filter
y = filter(h_100, 1, x);

figure('position', [0, 0, 800, 800])
subplot(3,1,1); plot(n,s); ylabel('Original signal s[n]')
subplot(3,1,2); plot(n,x); ylabel('Measured signal x[n]')
subplot(3,1,3); plot(n,y); ylabel('Filter output y[n]')
xlim([0 N])
grid on;
```



[�] 4) Which signal can be detected more easily by visual inspection?

Which signal can be detected more easily by visual inspection of the matched filter output? Is this justified by comparing the output SNR in each case?

For certain noise realisations, it is easier to detect the second (longer) signal by visual inspect than the first signal. However, running the code multiple times i.e., creating different realisations of the noise, we may see many false positives. The figure below shows an example of this. The highest response is measured at $n = 140$ but the maximum filter response should occur at $n = p + D - 1 = 100 + 0 - 1 = 99$.



Comparing the output SNR of the two filters, we observe that the SNR of the longer filter is usually better but this is not always the case. Different realisation of the noise process, yields different results:

```
SNR
```

```
SNR = 4.7238
```

```
SNR_100
```

```
SNR_100 = 5.5132
```

Problem 14.35: Matched Filter and white noise

In the book on page 861, the authors claim: "*The term matched filter was introduced because, in the case of white noise, the impulse response is “matched” to the shape of the signal being sought. In fact, the output of the matched filter is proportional to the correlation between the signal segment stored in the filter memory and the signal of interest.*"

This problem investigates this claim.

Consider the two 10-point signals $x_0[n]$ and $x_1[n]$ given below:

$$x_0[n] = \{1, 1, 1, -1, -1, 0, 0, 0, 0, 0\},$$
$$x_1[n] = \{1, 1, 1, 1, -1, 0, 0, 0, 0, 0\}.$$

These signals are sent over a communication channel which adds white noise to the signals. Using a correlation-detector approach, we want to detect signals in white noise. Let $h_0[n]$ denote the matched filter for $x_0[n]$ and let $h_1[n]$ denote the matched filter for $x_1[n]$.

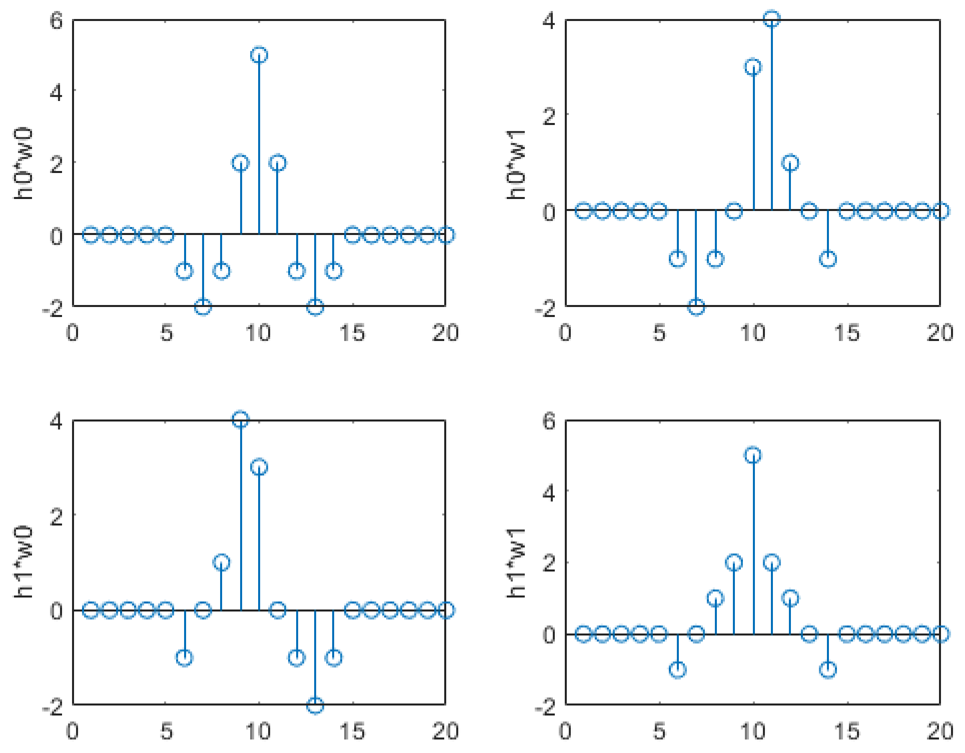
```
clear variables;
```

[�] a) Compare the impulse responses with input signals

Determine and plot the responses of $h_0[n]$ to $x_0[n]$ and $x_1[n]$. Repeat the same for $h_1[n]$. Compare these outputs at $n = 10$.

I don't understand what should be done here!

```
x0 = [1, 1, 1, -1, -1, 0, 0, 0, 0, 0]';
x1 = [1, 1, 1,  1, -1, 0, 0, 0, 0, 0]';

N = 20; n = 1:N;
v=zeros(N,1);%randn(N,1);
w0 = v + [x0;zeros(N-length(x0),1)];
w1 = v + [x1;zeros(N-length(x1),1)];
h0 = flip(x0); %wgn -> just use flipped
h1 = flip(x1);
figure
subplot(2,2,1); stem(filter(h0,1,w0)); ylabel('h0*w0')
subplot(2,2,2); stem(filter(h0,1,w1)); ylabel('h0*w1')
subplot(2,2,3); stem(filter(h1,1,w0)); ylabel('h1*w0')
subplot(2,2,4); stem(filter(h1,1,w1)); ylabel('h1*w1')
```



b) Compute the correlation of input and impulse response

You should notice that the output of the above matched filters at $n = 10$ can be computed as a correlation of the input and the impulse response. Implement such a structure and determine its output for each case in part (a) above.

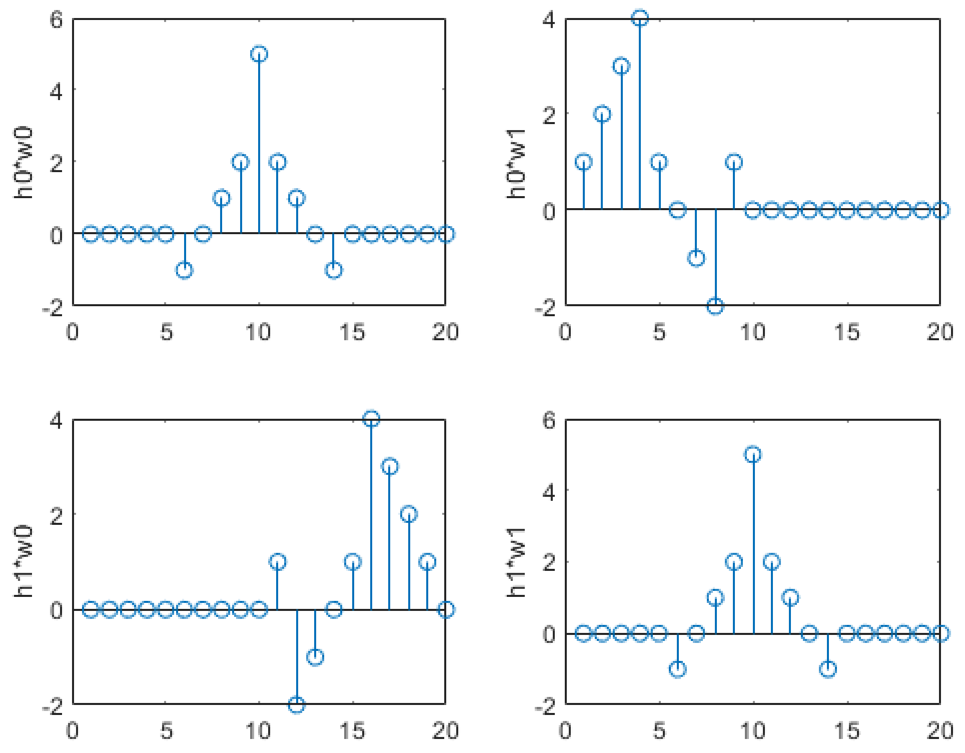
c) Modify the signals so the relations are zero

How would you modify the signal $x_0[n]$ so that the outputs of $h_0[n]$ to $x_1[n]$ and $h_1[n]$ to $x_0[n]$ are zero?

```
% Don't understand what is going on here
x0 = flip(x1); % the inverse, maybe?
corr(x0,x1)
```

```
ans = -0.2195
```

```
w0 = v + [x0;zeros(N-length(x0),1)];
h0 = flip(x0); %wgn -> just use flipped
figure
subplot(2,2,1); stem(filter(h0,1,w0)); ylabel('h0*w0')
subplot(2,2,2); stem(filter(h0,1,w1)); ylabel('h0*w1')
subplot(2,2,3); stem(filter(h1,1,w0)); ylabel('h1*w0')
subplot(2,2,4); stem(filter(h1,1,w1)); ylabel('h1*w1')
```



Problem 14.37: Matched Filter and Square Pulse Signals

Let $s_1[n]$ and $s_2[n]$ be short-length pulses of unit energy as given below:

$$s_1[n] = 1/3,\ 0 \le n \le 8 \quad \text{and} \quad s_2[n] = 0.1,\ 0 \le n \le 99$$

and zero everywhere. These pulses are observed in noise, i.e.

$$x_i[n] = s_i[n] + v[n],\ i = 1, 2$$

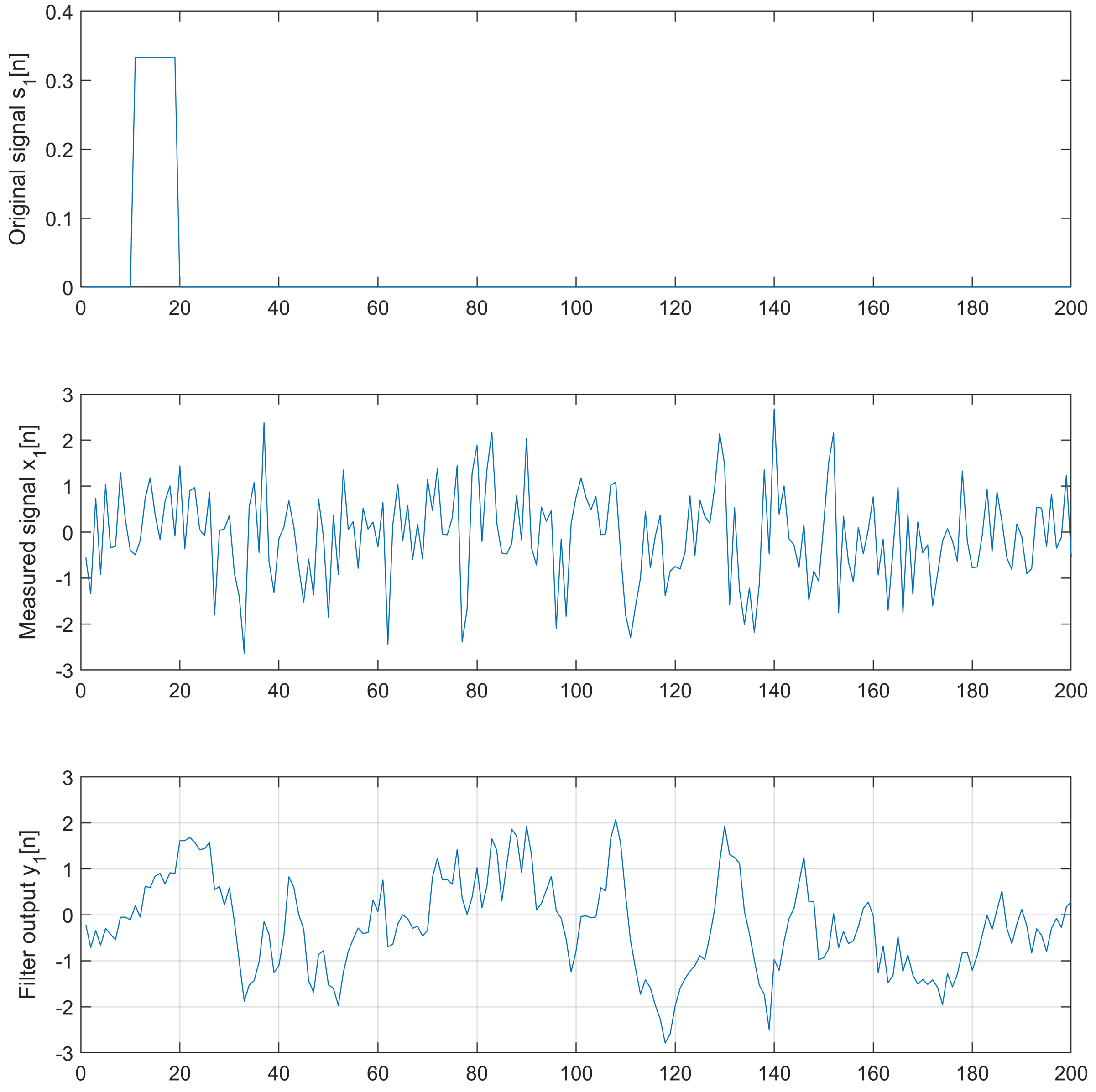
where $v[n] \sim \text{WGN}(0, 1)$.

```
clear variables;
```

[✓] a) Process x1[n] through a matched filter

Generate 200 samples of the noisy x1[n] signal and process it through the matched filter designed for s1[n]. Plot the original, noisy, and filtered signals and determine when the matched filter output is maximum.

```matlab
s1 = 1/3*ones(9, 1);

N = 200;
n = 1:N;
noise_var = 1;
D = 10;
[x1, v1, s1e] = gen_noisy_signal(s1, N, noise_var, D);

% Compute the impulse response of the matched filter
[h1, SNR1] = matched_filter(s1, v1);

% Feed the measured signal to the matched filter
y1 = filter(h1, 1, x1);

figure('position', [0, 0, 800, 800])
subplot(3,1,1); plot(n, s1e); ylabel('Original signal s_1[n]')
subplot(3,1,2); plot(n, x1); ylabel('Measured signal x_1[n]')
subplot(3,1,3); plot(n, y1); ylabel('Filter output y_1[n]')
xlim([0 N])
grid on;
```



[✓] b) Process x2[n] through a matched filter

Generate 200 samples of the noisy x2[n] signal and process it through the matched filter designed for s2[n]. Plot the original, noisy, and filtered signals and determine when the matched filter output is maximum.

Let $s_1[n]$ and $s_2[n]$ be short-length pulses of unit energy as given below:

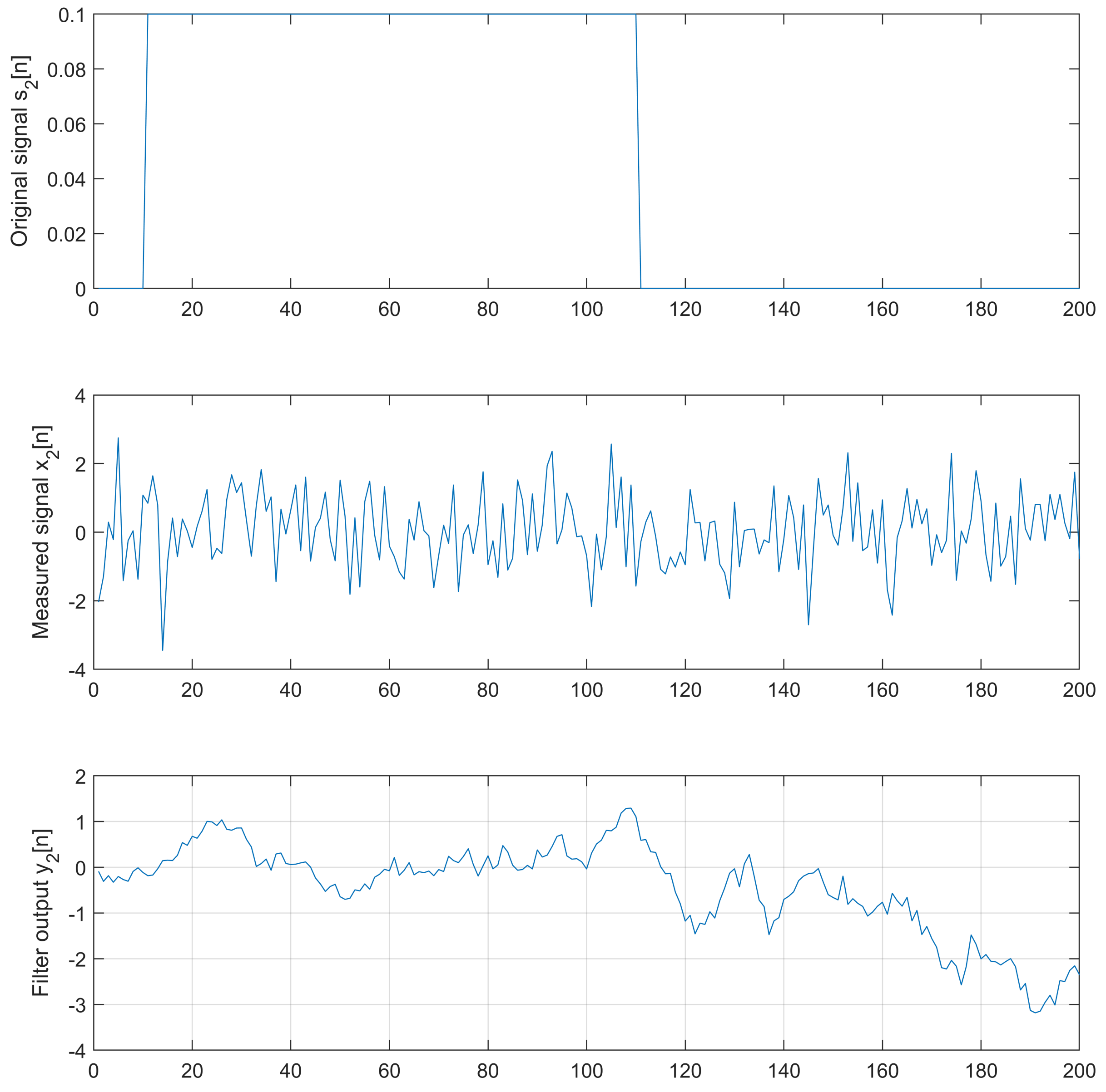
$$s_1[n] = 1/3,\ 0 \le n \le 8 \quad \text{and} \quad s_2[n] = 0.1,\ 0 \le n \le 99$$

and zero everywhere. These pulses are observed in noise, i.e.

$$x_i[n] = s_i[n] + v[n],\ i = 1, 2$$

where $v[n] \sim \mathrm{WGN}(0, 1)$.

```
s2 = 0.1*ones(100, 1);

N = 200;
n = 1:N;
noise_var = 1;
D = 10;
[x2, v2, s2e] = gen_noisy_signal(s2, N, noise_var, D);

% Compute the impulse response of the matched filter
[h2, SNR2] = matched_filter(s2, v2);

% Feed the measured signal to the matched filter
y2 = filter(h2, 1, x1);

figure('position', [0, 0, 800, 800])
subplot(3,1,1); plot(n, s2e); ylabel('Original signal s_2[n]')
subplot(3,1,2); plot(n, x2); ylabel('Measured signal x_2[n]')
subplot(3,1,3); plot(n, y2); ylabel('Filter output y_2[n]')
xlim([0 N])
grid on;
```



[�] c) Discuss the results in (a) and (b)

Discuss your results in (a) and (b) above in terms of filtered waveforms and maximization of the output SNR.

The peak response of the matched filter should occur at $n_0 = p + D - 1$

Since $s_1[n]$ is a 10-tap signal, this point should be $n_0 = 10 + 10 - 1 = 19$

Since $s_2[n]$ is a 100-tap signal, this point should be $n_0 = 100 + 10 - 1 = 119$

We rarely see these points for different realisations of the noise process.

```
[~, peak_n1] = max(y1)
```

```
peak_n1 = 108
```

```
[~, peak_n2] = max(y2)
```

```
peak_n2 = 109
```

What can we conclude?

```
SNR1
```

```
SNR1 = 0.9883
```

```
SNR2
```

```
SNR2 = 2.4961
```

[✓] Exam 2013, Problem 3, matched filter given ACRS for noise, SNR optimum vs non-optimum filter

Let a deterministic signal $s(n)$ be given by

$$s(n) = \begin{cases} 1 & n = 0 \\ 2 & n = 1 \\ 3 & n = 2 \\ 0 & \text{elsewhere} \end{cases}$$

Assume that the signal is corrupted by additive noise and that the autocorrelation function of the noise is $r_v(l) = 0.6^{|l|}$.

```
clear variables;
```

[�] 1) Determine the matched filter for detecting the presence of absense of the signal

Determine the matched filter for detecting the presence or absence of the signal in the noise at time n_0 and comment on the shape of its magnitude response.

The impulse response of the matched filter can be computed using Eq. 14.97:

$$\boldsymbol{h} = \kappa \boldsymbol{R}_v^{-1} \boldsymbol{s}. \qquad (14.97)$$

where k is the normalisation factor. Although the maximum SNR can be obtained by any choise of constant κ, we choose the constant by requiring that:

(a) $\boldsymbol{h}^{\mathrm{T}}\boldsymbol{s} = 1$, which yields $\kappa = 1/\boldsymbol{s}^{\mathrm{T}}\boldsymbol{R}_v^{-1}\boldsymbol{s}$

(b) $\mathrm{E}(v_{\mathrm{o}}^2[n]) = \boldsymbol{h}^{\mathrm{T}}\boldsymbol{R}_v\boldsymbol{h} = 1$, which yields $\kappa = 1/\sqrt{\boldsymbol{s}^{\mathrm{T}}\boldsymbol{R}_v^{-1}\boldsymbol{s}}$.

```
s = [1, 2, 3]';
p = numel(s);
r_vv = 0.6.^abs([0:p-1]);
R_vv = toeplitz(r_vv)
```

```
R_vv = 3×3
    1.0000    0.6000    0.3600
    0.6000    1.0000    0.6000
    0.3600    0.6000    1.0000
```

```
% Compute normalisation factor
k = 1/sqrt(s'*(R_vv\s)) % Same as 1/sqrt(s'*inv(R_vv)*s)
```

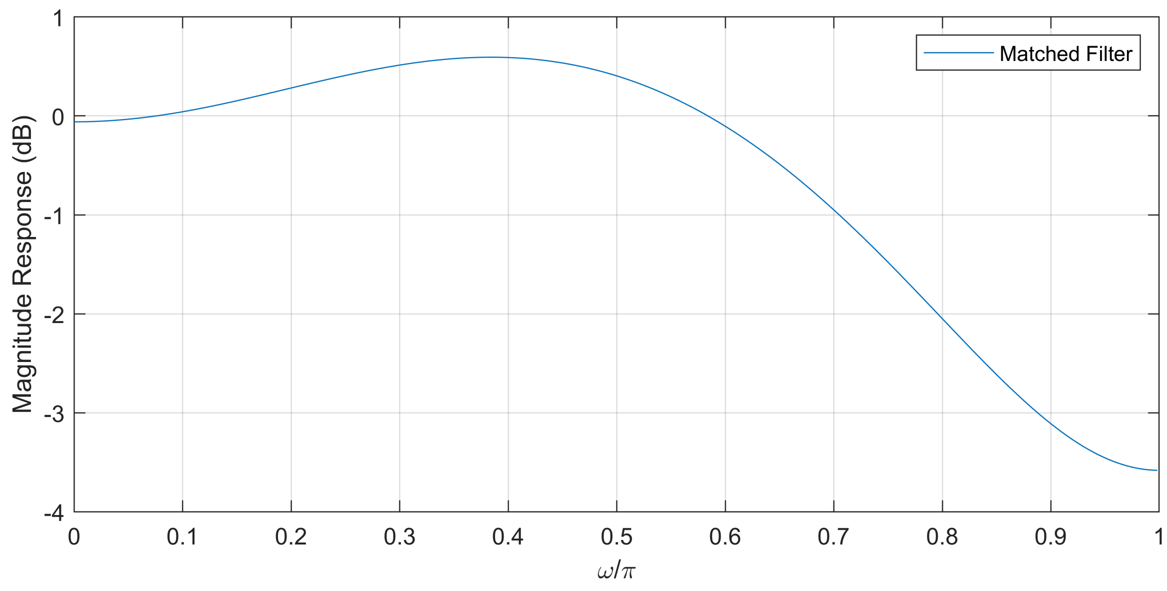
```
k = 0.3310
```

```

% Compute the filter
h = k*(R_vv\s)
```

```
h = 3×1
   -0.1035
    0.1655
    0.9311
```

```
% Plot the magnitude response
figure('position', [0, 0, 800, 350])
[H_h, w_h] = freqz(h, 1);
plot(w_h/pi, 20*log10(abs(H_h)))
legend('Matched Filter')
xlabel('\omega/\pi')
```

```
ylabel('Magnitude Response (dB)')
grid on;
```



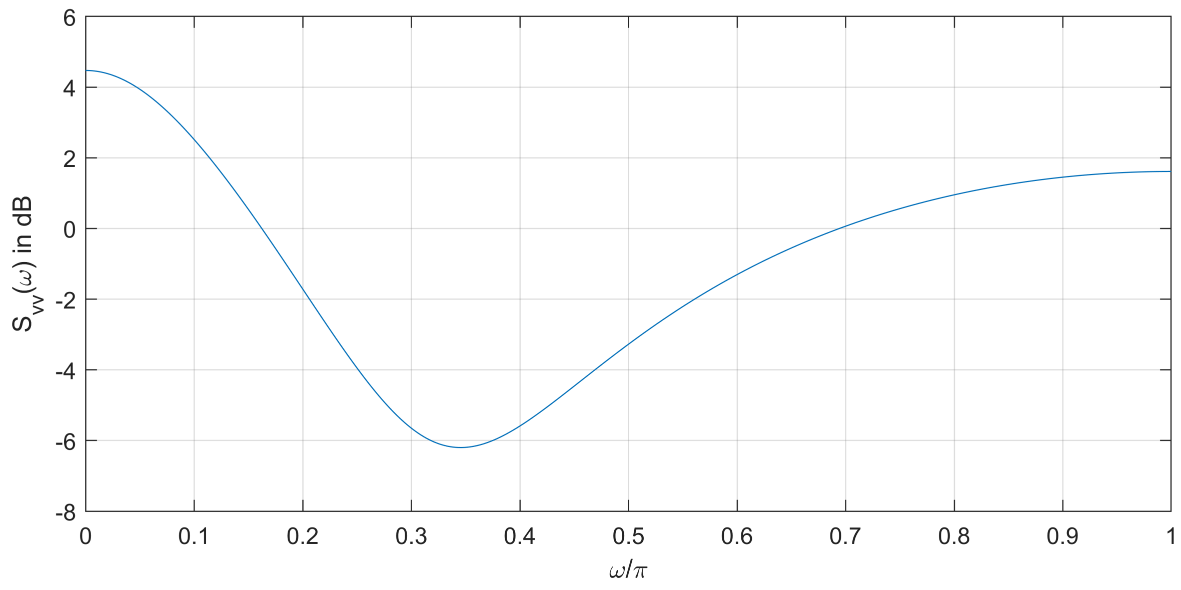
The matched filter passes DC components almost undistorted, has a small gain at midrange frequencies and attenuates high frequency components.

To understand why the matched filter has this shape, we need to look at the power spectrum density of the noise. Since the autocorrelation function is given, we can compute the PSD using the equation (13.119):

$$S_{xx}(\omega) = r_{xx}[0] + 2\sum_{\ell=1}^{\infty} r_{xx}[\ell]\cos\omega\ell, \qquad (13.119)$$

This is coded in the function `psd_from_acrs` (see at the end of the document):

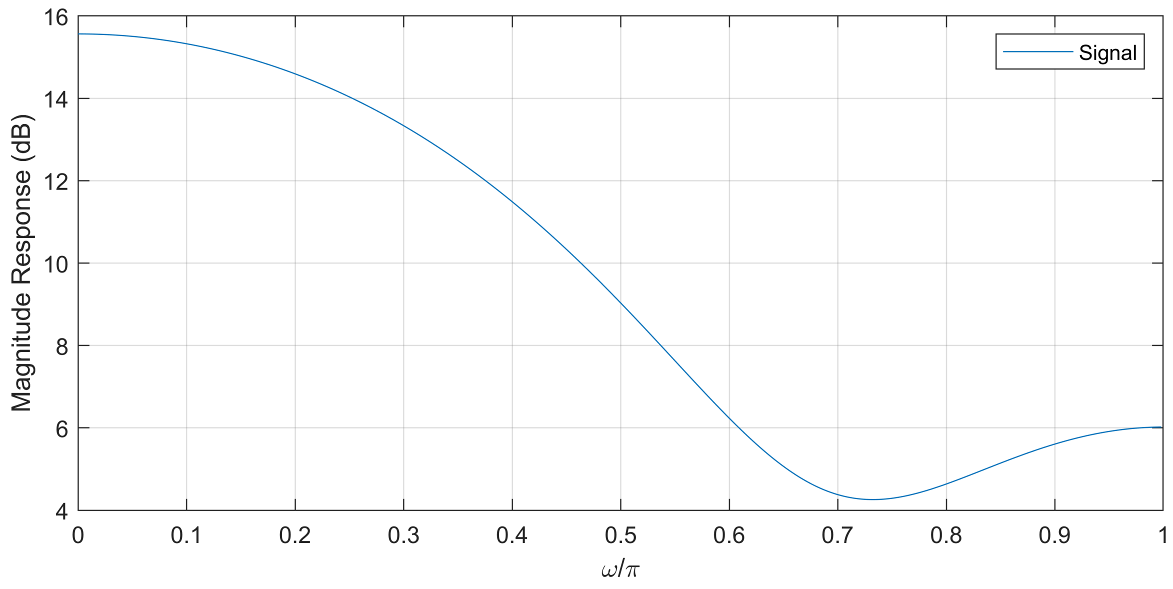
```
r_vv = 0.6.^abs(0:100);
[S_vv, w] = psd_from_acrs(r_vv);

figure('position', [0, 0, 800, 350])
plot(w/pi, pow2db(S_vv))
xlabel('\omega/\pi')
ylabel('S_{vv}(\omega) in dB')
grid on;
```



By plotting the PSD of the noise, we observe that the noise is centred at low frequencies.

Let us plot the magnitude response of the signal:

```
[H,w] = freqz(s, 1);
figure('position', [0, 0, 800, 350])
plot(w/pi, 20*log10(abs(H)));
legend('Signal')
xlabel('\omega/\pi')
ylabel('Magnitude Response (dB)')
grid on;
```



We observe that the signal acts like a low-pass filter.

But how do we explain all this?

[✓] 2) Calculate the optimum signal to noise ratio

The optimum SNR can be compute using Eq. 14.98:

$$\mathrm{SNR_o} = a^2\tilde{\boldsymbol{s}}^\mathrm{T}\tilde{\boldsymbol{s}} = a^2\boldsymbol{s}^\mathrm{T}\boldsymbol{R}_v^{-1}\boldsymbol{s}. \qquad (14.98)$$

Since the attenuation factor a is not given in this problem, we assume $a = 1$:

```
SNR_opt = s'*(R_vv\s)
```

```
SNR_opt = 9.1250
```

[✓] 3) Compute the SNR of a non-optimal filter

Assume that a non-optimum filter $h^T = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$ is used rather than the optimum filter.

3. Determine the decrease in signal to noise ratio when this filter is used instead of the optimum filter.

We can compute the SNR using Eq. (19.94):

$$\mathrm{SNR_o} = \frac{s_o^2[n_0]}{\mathrm{E}(v_o^2[n_0])} = a^2\frac{(\boldsymbol{h}^\mathrm{T}\boldsymbol{s})^2}{\boldsymbol{h}^\mathrm{T}\boldsymbol{R}_v\boldsymbol{h}}. \qquad (14.94)$$

```
h_nopt = [1, 0, 1]';
SNR_nopt = (h_nopt'*s)^2 / (h_nopt'*R_vv*h_nopt)
```

```
SNR_nopt = 5.8824
```

Functions

```
function [S, w] = ar2psd(a, v, N)
% AR2PSD Compute the Power Spectral Density from AR(p) coefficients
%   [S, w] = ar2psd(a, v, N)
% a: AR(p) coefficients
% v: the variance
% N: number of points in the range [1, pi]
% S: the estimated power spectrum
% w: frequencies
    w = linspace(0, 1, N) * pi;

    % Compute the transfer function
    % Used Eq. (13.133) in the book
    H = ones(N, 1);
    for k=1:numel(a)
        H = H + a(k)*exp(-1j * w' * k);
    end
    H = 1./H;

    % Finally compute the PSD
    S = v * H.*conj(H);
end

function [h, SNR] = matched_filter(s, v, a)
% MATCHED_FILTER: Compute the impulse response of a matched filter and
%                 and the corresponding output SNR.
%   [h, SNR] = matched_filter(s, v, a)
% s:   the signal
% v:   a realisation of the additive noise
% a:   the attenuation factor (default=1)
% h:   the impulse response of the matched filter
% SNR: the output SNR
    if nargin < 3
        a = 1;
    end

    p = numel(s); % Signal length

    % The autocorrelation matrix must be p x p since
    % its inverse is multiplied by a p-tap signal s(n)
    [r_vv, ~] = xcorr(v, p-1, 'biased');
    R_vv = toeplitz(r_vv(p:end));

    % The expression `R_vv^{-1} * s` is used multiple times,
    % so compute it once and reuse
    R_vv_inv_s = R_vv\s;

    % Compute normalisation factor
    k = 1/sqrt(s'*R_vv_inv_s);

    % Compute the filter
    h = k*R_vv_inv_s;

    % Compute the output SNR
    SNR = a^2 * s' * R_vv_inv_s;
end

function [x, v, s] = gen_noisy_signal(s_original, N, noise_var, D)
% Embeds a signal into a realisation of a WGN process
%  s_original: the original signal that needs to be embedded
%  N:          the number of samples of the noisy signal
%  noise_var:  the variance of WGN (default=1)
%  D:          the position of embedding
    if nargin < 4
        D = 0;
    end
    if nargin < 3
        noise_var = 1;
    end

    n = (1:N)';
    p = numel(s_original); % Signal length

    % Generate the signal.
    % Ensure that s[n] is zero when n is outside the interval [1, p]
    s = zeros(N, 1);
    s(n >= D+1 & n <= p+D) = s_original;

    % Generate the zero-mean WGN
    v = sqrt(noise_var) * randn(N, 1);

    % Generate the measured signal x[n]
    x = v + s;
end

function [S_xx, w] = psd_from_acrs(r_xx)
% Computes the power spectrum density based on the autocorrelation sequence
% This uses the Equation (13.119)
    w = (0:0.001:pi)';
    S_xx = r_xx(1) * ones(numel(w), 1);
    for l=2:length(r_xx)-1
        S_xx = S_xx + 2*r_xx(l+1) * cos(w*l);
    end
end
```