Exam 2017

Table of Contents

Problem 1: Decompose a Linear Phase Filter

Linear phase filters with the symmetry property are commonly applied in audio signal processing due to their frequency independent group delay.

Linear phase filters are FIR filters $H(z) = b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}$ with the symmetry property of the coefficients $b_k = b_{M-k}$ for $k = 0,\ 1 \ldots M$. Such filters are commonly applied in e.g. audio signal processing due to their frequency independent group delay.

Consider the following linear phase filter

$$H(z) = 1 + 2.5z^{-1} + z^{-2}.$$

```
h = [1, 2.5, 1];
```

1) Rewrite the system function as a product of a minimum phase filter and an all-pass filter

1. Rewrite $H(z)$ as a product of a minimum phase filter and an all-pass filter, i.e. $H(z) = H_{min}(z)H_{ap}(z)$.

Any system function can be decomposed into a product of a minimum-phase filter and an all-pass filter using the following formula:

$$H(z) = H_{\min}(z)H_{\text{ap}}(z)$$

The minimum phase filter can be computed as follows:

$$H_{\min}(z) = \prod_i -\frac{1}{a_i^*} H_i(z)\left(1 - a_i z^{-1}\right)$$

where $H_i(z)$ corresponds to the part of the transfer function where the i'th zero is inside the unit circle and z_i is the zero outside the unit circle. To get the zero z_i inside the unit circle, we use the trick $a = \frac{1}{z_0}$.

The allpass filter can be calculated:

$$H_{\text{ap}}(z) = \prod_i \frac{z^{-1} - a_i^*}{1 - a_i z^{-1}}$$

The filter decomposition algorithm has following steps:

1. Convert transfer function into pole-zero representation in order to find the zeros that are outside the unit circle
2. Compute a and its conjugate a^*
3. Compute $H_1(z)$ which corresponds to the part of the transfer function where zeros are inside the unit circle
4. Plugin the numbers for the formula for the minimum-phase filter
5. Plugin the numbers for the formula for the allpass filter
6. Put everything together

These steps are coded in MATLAB function (see at the end of the document):

```
[H_min, H_ap] = decompose_min_ap(h)
```

```
H_min =
```

$$\frac{2}{z} + \frac{1}{2\,z^2} + 2$$

```
H_ap =
```

$$\frac{\frac{1}{z} + \frac{1}{2}}{\frac{1}{2\,z} + 1}$$

So we have:

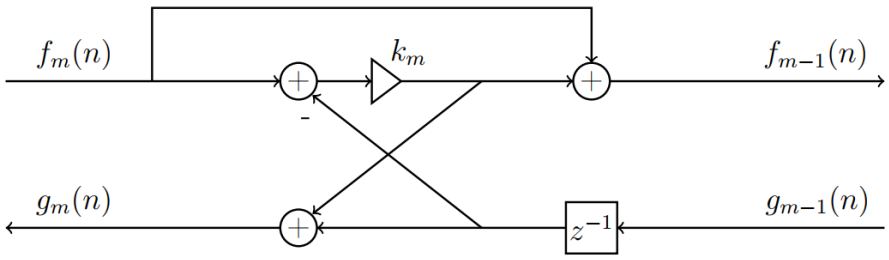
$$H_{\min}(z) = 2 + 2z^{-1} + \frac{1}{2}z^{-2} \quad \text{and} \quad H_{\text{ap}}(z) = \frac{0.5 + z^{-1}}{1 + 0.5z^{-1}}$$

2) Discuss what happens to a signal if it is filtered using only $H_{\min}(z)$ instead of $H(z)$?

The two system functions $H(z)$ and $H_{\min}(z)$ have the same magnitude response, but their phase properties differ. In this particular case the $H(z)$ is a linear phase filter and the group delay through the filter is constant and independent of frequency. For the minimum phase system function the phase is not linear and the group delay becomes frequency dependent. A signal passing through $H_{\min}(z)$ instead of $H(z)$ will thus become phase distorted.

Problem 2: Lattice Structures

Many different lattice structure have been investigated over the years, often tailored to specific requirements. One particular example is based on minimizing hardware requirements and uses only one multiplication per stage. The structure of the *m*'th stage of a one-multiplier lattice filter is shown below. The input and output is connected as for standard all-pole filters.




```
clear variables;
```

1) Determine the two equations relating outputs to inputs

1. Determine the two equations relating outputs $f_{m-1}(n)$ and $g_m(n)$ to inputs $f_m(n)$ and $g_{m-1}(n)$.

$$\begin{aligned} f_{m-1}(n) &= f_m(n) + k_m(f_m(n) - g_{m-1}(n)) \\ &= f_m(n) + k_m f_m(n) - k_m g_{m-1}(n-1) \\ &= (1 + k_m) f_m(n) - k_m g_{m-1}(n-1) \end{aligned}$$

$$g_m(n) = g_{m-1}(n-1) + k_m(f_m(n) - g_{m-1}(n))$$

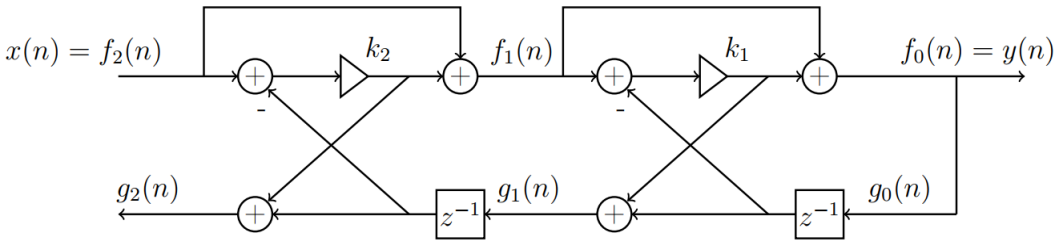
$$= g_{m-1}(n-1) + k_m f_m(n) - k_m g_{m-1}(n)$$

$$= k_m f_m(n) + (1 - k_m) g_{m-1}(n-1)$$

2) When is the first non-zero sample observed at the output?

2. If the input of a two-stage, one-multiplier lattice filter with reflection coefficients k_1 and k_2 is excited by an impulse, $x(n) = \delta(n)$, when is the first non-zero sample observed at the output $y(n)$?

The two-stage, one multiplier lattice looks like this:



We found that the equation relating output to input is:

$$f_{m-1}(n) = (1 + k_m) f_m(n) - k_m g_{m-1}(n-1)$$

Since we want the first output, we do not concern ourselves with the $g_{m-1}(n-1)$.

First, we compute $f_1(n)$:

$$f_1(n) = (1 + k_2) f_2(n) = (1 + k_2) x(n)$$

Next, we compute $f_0(n)$:

$$f_0(n) = (1 + k_1) f_1(n) = (1 + k_1)(1 + k_2) x(n) = y(n)$$

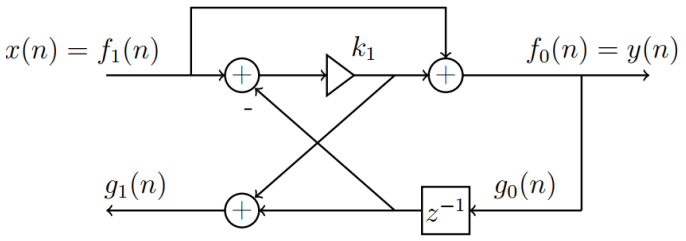
Since there is a direct path without any delays from input $x(n)$ to the output $y(n)$, the first non-zero sample appears instantaneously at the output.

$$y(n) = (1 + k_1)(1 + k_2) x(n)$$

3) Calculate the transfer function for a one-stage lattice filter

3. Calculate the transfer function $H_{G_1}(z) = G_1(z)/X(z)$ for a one-stage, one-multiplier, lattice filter with reflection coefficients k_1 and show that it corresponds to an all-pass filter.

The one-stage, one multiplier lattice looks like this.



The equations relating the inputs and outputs are:

$$f_0(n) = (1 + k_1)f_1(n) - k_1 g_0(n-1)$$

$$g_1(n) = k_1 f_1(n) + (1 - k_1)g_0(n-1)$$

To calculate transfer functions, the two equations are first *z*-transformed:

$$F_0(z) = (1 + k_1)F_1(z) - k_1 z^{-1} G_0(z) \quad \text{Eq. (1)}$$

$$G_1(z) = k_1 F_1(z) + (1 - k_1)z^{-1} G_0(z) \quad \text{Eq. (2)}$$

We know that $F_1(z) = X(z)$ and $F_0(z) = G_0(z) = Y(z)$.

We use the identity $F_0(z) = G_0(z)$ to rewrite our Eq. (1):

$$G_0(z) = (1 + k_1)F_1(z) - k_1 z^{-1} G_0(z)$$

$$G_0(z) + k_1 z^{-1} G_0(z) = (1 + k_1)F_1(z)$$

$$\left(1 + k_1 z^{-1}\right) G_0(z) = (1 + k_1)F_1(z)$$

$$G_0(z) = \frac{(1 + k_1)F_1(z)}{1 + k_1 z^{-1}}$$

Substitutde the new expression into Eq. (2)

$$G_1(z) = k_1 F_1(z) + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}} F_1(z)$$

We know that $F_1(z) = X(z)$:

$$G_1(z) = k_1 X(z) + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}} X(z)$$

$$G_1(z) = X(z)\left(k_1 + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}}\right)$$

$$\frac{G_1(z)}{X(z)} = k_1 + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}}$$

$$\frac{G_1(z)}{X(z)} = \frac{k_1\left(1 + k_1 z^{-1}\right)}{1 + k_1 z^{-1}} + \frac{(1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}}$$

$$\frac{G_1(z)}{X(z)} = \frac{k_1\left(1 + k_1 z^{-1}\right) + (1 - k_1)(1 + k_1)z^{-1}}{1 + k_1 z^{-1}}$$

```
syms k z
expand(k*(1+k*z)+(1-k)*(1+k)*z)
```

ans = $k + z$

$$\frac{G_1(z)}{X(z)} = \frac{k_1 + z^{-1}}{1 + k_1 z^{-1}}$$

An allpass filter has the form:

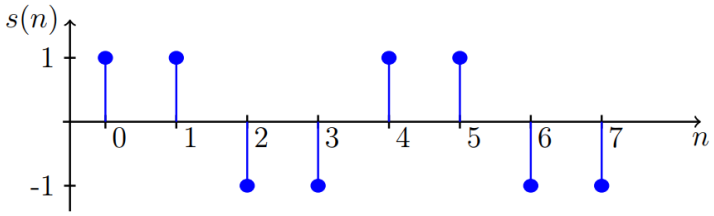
$$H_k(z) = z^{-1}\frac{1 - p_k^* z}{1 - p_k z^{-1}} = \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}} \qquad (5.157)$$

Higher order allpass systems can be obtained by cascading multiple first-order sections, as

$$H_{\text{ap}}(z) = \mathrm{e}^{\mathrm{j}\beta}\prod_{k=1}^{N}\frac{z^{-1} - p_k^*}{1 - p_k z^{-1}}, \qquad (5.158)$$

Problem 3: Detect presence of signal using matched filter

Consider the deterministic signal, *s*(*n*) shown below in blue. The signal is zero for all other values of *n*.



The signal is distorted by additive low frequency noise with autocorrelation $r_v(\ell) = 0.4^{|\ell|}$.

1) Design a matched filter for detecting the presence of the signal and determine the improvement in signal to noise ratio.

The impulse response of the matched filter is given by:

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

where R_v is autocorrelation matrix of noise and κ 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, 1, -1, -1, 1, 1, -1, -1]';

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)
ell = 0:p-1;
r_vv = 0.4.^ell;
R_vv = toeplitz(r_vv);

% Compute normalisation factor (b)
k = 1/(s'*(R_vv\s));     % Using (a)
k = 1/sqrt(s'*(R_vv\s)); % Using (b)

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

% Print the matched filter coefficients
h
```

```
h = 8×1
    0.2292
    0.4431
   -0.4431
   -0.4431
    0.4431
    0.4431
   -0.4431
   -0.2292
```

The optimum SNR at the output is given by:

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

Assuming the attenuation factor $a = 1$:

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

```
SNR_o = 9.7143
```

The SNR at the input is given by:

$$\mathrm{SNR}_i = \frac{(\text{Value of signal at } n = n_0)^2}{\text{power of noise}} = \frac{s^2(n = n_0)}{r_v(0)}$$

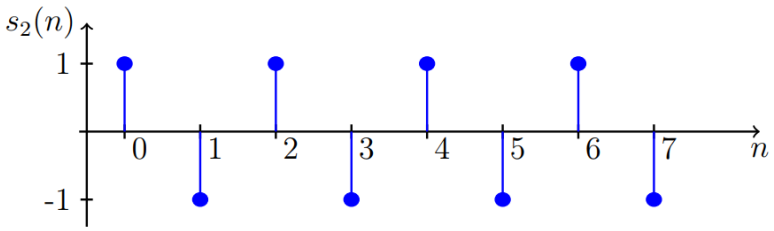
```
SNR_i = s(1)^2/r_vv(1)
```

```
SNR_i = 1
```

The improvement in SNR is 9.7 (almost 10 times)

2) Discuss the improvement of SNR if another signal is used

A second signal $s_2(n)$ is given by



2. Discuss whether the signal to noise ratio will be improved if the signal $s_2(n)$ is used instead of $s(n)$.

If we use $s_2(n)$ instead of $s(n)$, the output SNR becomes 17.3:

```
s = [1, -1, 1, -1, 1, -1, 1, -1]';
p = numel(s); % Signal length
ell = 0:p-1;
r_vv = 0.4.^ell;
R_vv = toeplitz(r_vv);
a = 1;
SNR_o = a^2 * s' * (R_vv\s)
```

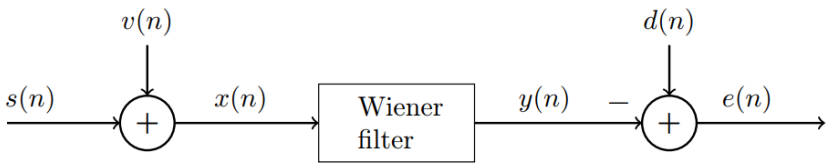
```
SNR_o = 17.3333
```

The $s_2(n)$ is similar to $s(n)$ but it oscillates faster i.e., has higher frequency than the $s(n)$ signal. Since the noise is at low frequency, it is easier to separate the $s_2(n)$ signal from the noise. This means that a higher SNR can be expected from a signal with a higher frequency.

That is why the output SNR for $s_2(n)$ is better than that for $s(n)$.

Problem 4: Recover signal using a Wiener filter

Consider the Wiener filtering problem shown below. A signal $s(n)$ is corrupted by additive, uncorrelated noise $v(n)$ giving the signal $x(n) = s(n) + v(n)$. It is desired to use a 2-tap Wiener filter to recover the signal i.e. $d(n) = s(n)$.



The noise $v(n)$ is generated in an MA(1) process as

$$v(n) = w(n) + w(n-1)$$

where $w(n)$ is a zero-mean gaussian white noise sequence with $\sigma_w^2 = 1$. The autocorrelation function of the signal is

$$r_s(l) = 4 \cdot 0.25^{|l|}.$$

```
clear variables;

M = 2;
ell = 0:M-1;
delta = @(l) ell == l;

r_ss = 4*(0.25).^(abs(ell));
```

1) Compute the autocorrelation function of the noise

1. Show that the autocorrelation function of the noise $v(n)$ is given by

$$r_v(l) = \delta(l-1) + 2\delta(l) + \delta(l+1).$$

In ADSI Problem 4.9, we found that the autocorrelation for an MA(1) process where the input signal is a white noise with unit variance can be described as:

$$r_v(\ell) = \left(b_0^2 + b_1^2\right)\delta(\ell) + b_0 b_1 \left(\delta(\ell-1) + \delta(\ell+1)\right)$$

In this problem $b_0 = b_1 = 1$:

$$r_v(\ell) = \delta(\ell-1) + 2\delta(\ell) + \delta(\ell+1)$$

```
r_vv = delta(-1) + 2*delta(0) + delta(1)
```

```
r_vv = 1×2
     2     1
```

2) Compute the optimum filter coefficients

2. Calculate the crosscorrelation vector $\mathbf{g}$, the autocorrelation matrix $\mathbf{R}_x$ and the optimum filter coefficients.

The second order Wiener filter for estimating the signal $s(n)$ is given by:

$$y(n) = h_1 x(n) + h_2 x(n-1)$$

The optimum Wiener filter coefficients is given by Wiener-Hopf Eq. 14.109:

$$\boldsymbol{h}_{\mathrm{O}} = \boldsymbol{R}_x^{-1}\boldsymbol{g}, \qquad (14.109)$$

where $\boldsymbol{R}_x$ is the autocorrelation matrix of the corrupted signal $s(n)$ given as:

$$\boldsymbol{R}_x = \begin{bmatrix} r_x(0) & r_x(1) \\ r_x(1) & r_x(0) \end{bmatrix}$$

And $\boldsymbol{g}$ is the cross-correlation between the desired signal $s(n)$ and the corrupted signal $x(x)$ given as:

$$\boldsymbol{g} = \begin{bmatrix} r_{\mathrm{sx}}(0) \\ r_{\mathrm{sx}}(1) \end{bmatrix}$$

First, we compute the autocorrelation $r_x(\ell)$:

$$\begin{aligned} r_x(\ell) &= E[x(n)x(n-\ell)] \\ &= E[(s(n)+v(n))(s(n-\ell)+v(n-\ell))] \\ &= E[s(n)s(n-\ell) + s(n)v(n-\ell) + v(n)s(n-\ell) + v(n)v(n-\ell)] \\ &= r_s(\ell) + r_{\mathrm{sv}}(\ell) + r_{\mathrm{sv}}(\ell) + r_v(\ell) \end{aligned}$$

Since $s(n)$ and $v(n)$ are uncorrelated $r_{\mathrm{sv}}(\ell) = 0$. We computed $r_v(\ell) = \delta(\ell-1) + 2\delta(\ell) + \delta(\ell+1)$:

$$r_x(\ell) = r_s(\ell) + r_v(\ell)$$

$$r_x(\ell) = 4 \cdot 0.25^{|\ell|} + \delta(\ell-1) + 2\delta(\ell) + \delta(\ell+1)$$

```
r_xx = r_ss + r_vv;
R_xx = toeplitz(r_xx)
```

```
R_xx = 2×2
     6     2
     2     6
```

Compute the cross-correlation $r_{\mathrm{sx}}(\ell)$:

$$\begin{aligned} r_{\mathrm{yx}}(\ell) &= E[s(n)x(n-\ell)] \\ &= E[s(n)(s(n-\ell)+v(n-\ell))] \end{aligned}$$

$$= E[s(n)s(n-\ell)] + E[s(n)v(n-\ell)]$$

$$= r_s(\ell) + r_{sv}(\ell)$$

$$= r_s(\ell) + 0 \quad \text{(since } s(n) \text{ and } v(n) \text{ are uncorrelated } r_{sv}(\ell) = 0$$

$$= 4 \cdot 0.25^{|\ell|}$$

```
g = r_ss'
```

```
g = 2×1
     4
     1
```

Compute the optimum filter coefficients:

```
h_opt = R_xx\g
```

```
h_opt = 2×1
    0.6875
   -0.0625
```

3) Calculate SNR before and after the Wiener filter

The signal to noise ratio of the input signal is given by

$$\text{SNR}_i = \frac{(\text{value of signal at } n = n_0)^2}{\text{power of noise}} = \frac{s^2(n = n_0)}{r_v(0)} = \frac{r_s(0)}{r_v(0)}$$

We know that:

$$r_s(\ell) = 4 \cdot 0.25^{|\ell|} \quad \text{and} \quad r_v(\ell) = \delta(\ell - 1) + 2\delta(\ell) + \delta(\ell + 1)$$

This means:

$$r_s(0) = 4 \quad \text{and} \quad r_v(0) = 2$$

Thus:

$$\text{SNR}_i = \frac{r_s(0)}{r_v(0)} = \frac{4}{2} = 2$$

The output SNR of the Wiener filter is given by:

$$\text{SNR}_o = \frac{h_{\text{opt}}^T R_s h_{\text{opt}}}{h_{\text{opt}}^T R_v h_{\text{opt}}}$$

```
R_ss = toeplitz(r_ss);
R_vv = toeplitz(r_vv);
```

```
SNR_o = (h_opt'*R_ss*h_opt) / (h_opt'*R_vv*h_opt)
```

```
SNR_o = 2.0991
```

The filter increases the SNR but only slightly. This is expected and is an excellent way to check your results.

Functions

```
function [h]=filter_coeff(rational_system_expr, z)
    h = coeffs(expand(rational_system_expr * z^10), 'all');
    h = h(1:find(h, 1, 'last'));
end

function [H_min, H_ap]=decompose_min_ap(h)
    % Decomposes a filter into minimum-phase filter and all-pass filter
    %  h: the filter coefficients
    syms z;
    rts = roots(h);

    H_outside = 1; % Represents part of H where zeros are outside the unit circle
    H_inside = 1;  % Represents part of H where zeros are inside the unit circle
    zeros_outside = [];
    for i = 1:numel(rts)
        root = rts(i);
        if abs(root) > 1
            H_outside = H_outside * (1 - root*z^-1);
            zeros_outside = [zeros_outside; root];
        else
            H_inside = H_inside * (1 - root*z^-1);
        end
    end

    % Sanity check
    % H = expand(H_inside * H_outside)

    % Compute minimum-phase filter and all-pass filter
    H_min = 1;
    N = numel(zeros_outside);
    H_ap = 1;
    for i = 1:N
        z_i = zeros_outside(i);
        a = 1/z_i;
        a_conj = conj(a);
        H_min = H_min * (-(1/a_conj) * H_inside * (1-a * z^(-1)));
        H_ap = H_ap * (z^(-1) - a_conj) / (1 - a*z^(-1));
    end
    H_min = expand(H_min);
end
```