Homework 3

Table of Contents

ADSI Problem 2.1: Implementation of All-zero lattice filters

Work you way through the MATLAB code in Figure 9.26 and make sure you understand what is going on.

Code can be found in azlatfilt.m.

ADSI Problem 2.2: Find impulse response of an all-zero lattice filter

Consider an all-zero (FIR) lattice filter with $k_1 = 0.65$, $k_2 = -0.34$ and k_3=0.8.

1. Find the impulse response of the filter by tracing an impulse $x(n) = \delta(n)$ through the filter.

We can find the the impulse response of a filter by passing letting the impulse signal$\delta[n]$ pass through the filter:

```
k = [0.65, -0.34, 0.8];
x = [1, 0, 0, 0];
G = 1;
y = azlatfilt(k, x, G)
```

```
y = 1×4
    1.0000    0.1570    0.0032    0.8000
```

The impulse response of the given filter is:

$h[n] = [1, 0.157, 0.0032, 0.8]$

ADSI Problem 2.3: Find the reflection coefficients for an all-zero lattice filter

An all-zero lattice filter has the following system function

$$H(z) = 1 + \frac{13}{24}z^{-1} + \frac{5}{8}z^{-2} + \frac{1}{3}z^{-3}$$

1. Find the reflection coefficients k_1, k_2 and k_3 for the corresponding lattice filter.

We use the MATLAB function ***tf2latc*** to convert transfer function filter parameters to lattice filter form

```
b = [1, 13/24, 5/8, 1/3];
a = 1;
k = tf2latc(b, a)
```

```
k = 3×1
    0.2500
    0.5000
    0.3333
```

We can also try the algorithm (Figure 9.24) from the book

```
k = fir2lat(b)
```

```
k = 1×3
    0.2500    0.5000    0.3333
```

Both functions agree!

The reflection coefficients are $k_1 = \frac{1}{4}, k_2 = \frac{1}{2}, k_3 = \frac{1}{3}$

ADSI Problem 2.5: All-zero lattice filter, find coefficients from system function

A filter has the system function

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

1. Calculate k_1 and k_2 for the corresponding lattice filter and draw the lattice structure.

We cannot use a recursive algorithm because one of the reflection coefficients is equal to 1

```
b = [1, 2, 1];
a = 1;
% tf2latc(b) % This fails because one coeff. is 1
```

Instead we can use a cascade of two single stage filters.

First, we find the two roots:

```
roots(b)
```

```
ans = 2×1
    -1
    -1
```

Next, we factorise the original FIR filter to two cascade filters:

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

We know that a single stage all-zero filter has following difference equation:

$$y[n] = x[n] + k_1 x[n-1]$$

Taking the z-transform we get:

$$Y_{\mathrm{az}}(z) = X_{\mathrm{az}}(z) + k_1 X_{\mathrm{az}}(z) z^{-1}$$

The system function of a single stage all-zero lattice filter is therefore:

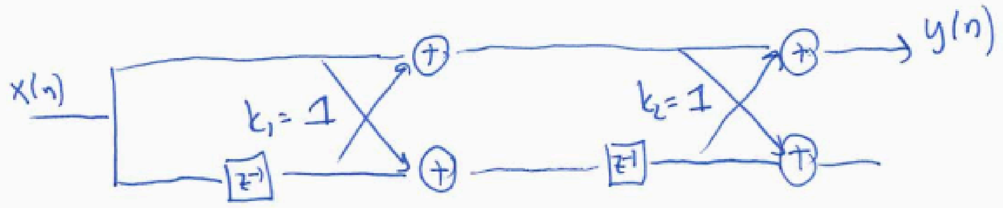
$$H_{\mathrm{az}}(z) = \frac{Y_{\mathrm{az}}(z)}{X_{\mathrm{az}}(z)} = 1 + k_1 z^{-1}$$

It turns out that we can cascade two single stage all-zero lattice filters.

instead we can factorize $H(z) = A_2(z)$

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

so we can instead implement as a cascade of two single stages with $k_1 = 1$ and it turns out that it also works in a single two-stage lattice



So in our case, we have $k_1 = 1$ and $k_2 = 1$.

Let us check if we can get the impulse response of the original filter.

```
k = [1, 1];
lat2fir(k, 1)
```

```
ans = 1×3
     1     2     1
```

```
latc2tf(k) % Same as lat2fir
```

```
ans = 1×3
     1     2     1
```

Success!

ADSI Problem 2.7: Find system function an all-pole filter from reflection coefficients

Determine the system function for an all-pole filter with the reflection coefficients $k_1 = 0.6$, $k_2 = 0.3$, $k_3 = 0.5$ and $k_4 = 0.9$.

We can use MATLAB function latc2tf

```
k = [0.6, 0.3, 0.5, 0.9];
[num, dem] = latc2tf(k, 'allpole')
```

```
num = 1×5
     1     0     0     0     0
dem = 1×5
    1.0000    1.3800    1.3110    1.3370    0.9000
```

The system function is:

$$H(z) = \frac{1}{1 + 1.38z^{-1} + 1.311z^{-2} + 1.337z^{-3} + 0.9z^{-4}}$$

ADSI Problem 2.10: Linear prediction and lattice filters

The idea behind linear prediction is than the signal $\hat{x}(n)$ can be estimated as a weighted linear combination of the p previous samples

$$\hat{x}(n) = -\sum_{k=1}^{p} a_p(k)x(n-k)$$

The file **`signal1.dat`** contains 64 samples from an artificial signal.

1. Load **`signal1.dat`** into MATLAB. Create an autocorrelation of the signal using **`xcorr`** and plot it. What does the autocorrelation tell you about the signal?

Based on the autocorrelation values we can calculate the coefficients for the optimum pth order linear predictor with the normal equations (which we will derive later in the course)

$$r_x(l) = -\sum_{k=1}^{p} a_p(k) r_x(l-k), \qquad l = 1, \dots, p$$

Let $p = 2$.

2. Compute $a_2(1)$ and $a_2(2)$.

Next, consider a two-stage all-zero lattice filter.

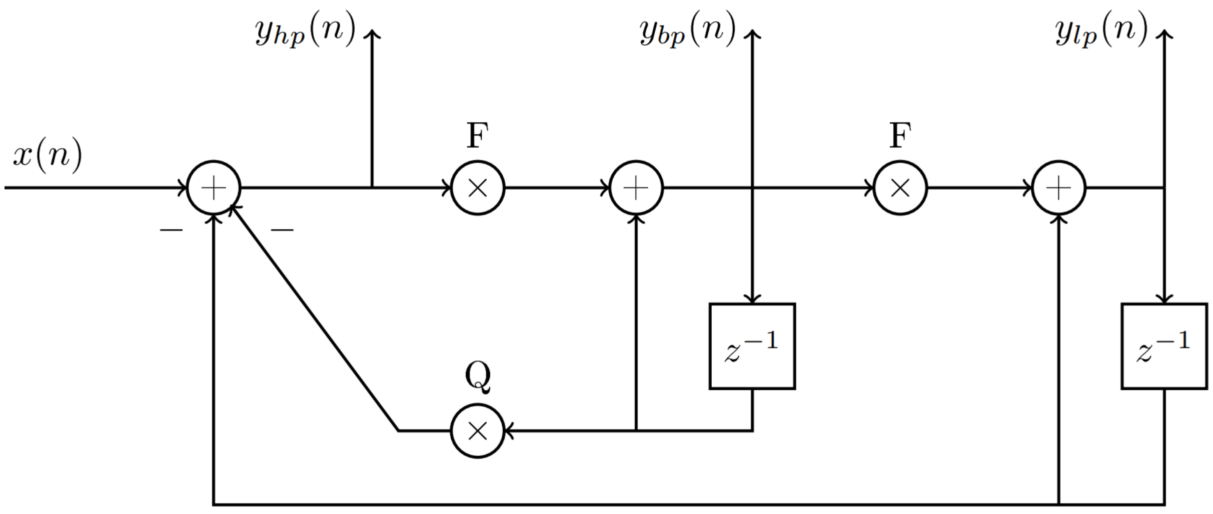
3. What is the relation between the output of the filter, $f_2(n)$, and the above linear prediction?

4. Compute the reflection coefficients k_1 and k_2 from the $a_2(1)$ and $a_2(2)$ values found above.

5. Send the signal through a 2nd order all-zero lattice filter and analyze the prediction error. Is the signal predictable? Repeat for a higher order predictor.

ADSI Problem 2.11 (Optional): The digital state variable filter

One interesting filter structure is the digital state variable filter which has its roots in the analog state variable filter. It can be considered as a standard second order filter, but with the valuable feature that the center frequency and Q-factor are independently tuneable. The structure of the filter is shown here



Where $F = 2\sin(\pi F_c/F_s)$ with F_c the filter corner frequency, F_c is the sampling frequency and Q is the Q-factor of the filter. From the structure, updating equations for the three state variables, $y_{hp}(n)$, $y_{bp}(n)$ and $y_{lp}(n)$ can easily be written down. Index hp, bp and lp denotes highpass, bandpass and lowpass respectively. For $y_{hp}(n)$ the equation becomes

$$y_{hp}(n) = x(n) - y_{lp}(n-1) - Qy_{bp}(n-1).$$

1. Write down the updating equations for the two other state variables.

2. Calculate the transfer function $H_{LP}(z)$ from the input to the lowpass output.

3. Implement the state variable filter as a function in Matlab. Make sure the three equations are updated in the correct order.

4. Test the implementation by sending white noise sampled at 48000 Hz through the filter. Use Fc=1000 Hz and Q=2. Plot the spectra of the three outputs on the same graph and compare with the expected spectra. Repeat for Q=10.

The filter is described in P. Dutilleux "Simple to operate digital time varying filters" AES 86th Convention, Hamburg (1989) In this paper Dutillex shows that the digital state variable filter fails if F_c becomes to large. A rule of thumb is to keep $F_c < F_s/10$.

5. Demonstrate that this is true by using Fc=20000 Hz in the above implementation.

6. A band reject output is also easily obtained with the digital state variable filter. Extend the above drawing with a $y_{br}(n)$ output.

Functions

```
function [y] = azlatfilt(k, x, G)
% AZLATFILT      Function for implementation of an all-zero FIR lattice filter
% From Figure 9.26 (p 515)

    M = length(k);

    % Create an array for the sequence f_m[n]
    f = zeros(1,M);

    % Create an array for the sequence g_m[n]
    g = f;

    % Create an array for the sequence g_m[n-1]
    oldg = zeros(1,M);

    % Keeps track of x[n-1]
    oldx = 0;
    x = G*x;
    y = zeros(size(x));

    for n=1:length(x)
        % Compute f1[n] = x[n] + k1x[n ? 1] -> Equation (9.57a)
        f(1) = x(n)+k(1)*oldx;

        % Compute g1[n] = k1x[n] + x[n ? 1] -> Equation (9.57b)
        g(1) = k(1)*x(n)+oldx;
        oldx = x(n);
        for m = 2:M
            % Compute: fm[n] = fm?1[n] + km gm?1[n ? 1] -> Equation (9.55a)
            f(m) = f(m-1)+k(m)*oldg(m-1);

            % Compute: gm[n] = km fm?1[n] + gm?1[n ? 1] -> Equation (9.55b)
            g(m) = k(m)*f(m-1)+oldg(m-1);

            % Delay
            oldg(m-1) = g(m-1);
        end

        % y[n] = fM[n] -> Equation (9.56b)
        y(n) = f(M);
    end
end


function [k,G] = fir2lat(h)
% Converts FIR filter coefficients to lattice coefficients.
% From Figure 9.24 (p. 514)
    G = h(1);
    a = h/G;
    M = length(h)-1;
    k(M) = a(M+1);
    for m = M:-1:2
        b = fliplr(a);
        a = (a-k(m)*b)/(1-k(m)^2); a = a(1:m);
        k(m-1) = a(m);
    end
end

function h = lat2fir(k, G)
% Converts lattice coefficients to FIR filter coefficients.
% From Figure 9.25 (p. 515)
    a = 1;
    b = 1;
    M = length(k);
    for m = 1:1:M
        a = [a,0]+k(m)*[0,b];
        b = fliplr(a);
    end
    h = G*a;
end
```