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.

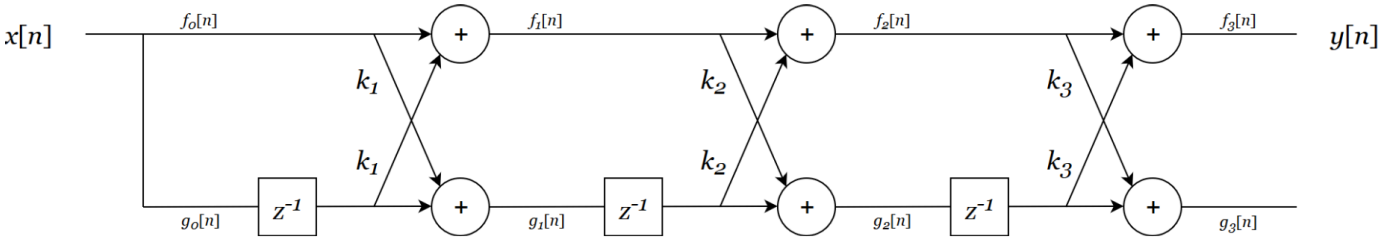
Code can be found in azlatfilt.m.

[✓] 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.

Since the lattice filter has three reflection coefficients, it is a 3-stage all-zero lattice filter.



Here are the formulas needed to trace through the filter.

$$x[n] = f_0[n] = g_0[n],$$

$$y[n] = f_M[n].$$

$$f_m[n] = f_{m-1}[n] + k_m g_{m-1}[n-1], \quad m = 1, 2, \ldots, M$$

$$g_m[n] = k_m f_{m-1}[n] + g_{m-1}[n-1]. \quad m = 1, 2, \ldots, M$$

The input is $x[n] = \delta[n] = [1, 0, 0, 0]$

$n = 1$
$f_0(n) = g_0(n) = x(1) = 1$
$f_1(n) = f_0(n) + k_1 g_0(n-1) = 1 + 0 = 1$
$g_1(n) = k_1 f_0(n) + g_0(n-1) = 0.65 + 0 = 0.65$
$f_2(n) = f_1(n) + k_2 g_1(n-1) = 1 - 0.34 \cdot 0 = 1$
$g_2(n) = k_2 f_1(n) + g_1(n-1) = -0.34 \cdot 1 + 0 = -0.34$
$f_3(n) = f_2(n) + k_3 g_2(n-1) = 1 + 0.8 \cdot 0.0 = \underline{\mathbf{1}}$
$g_3(n) = k_3 f_2(n) + g_2(n-1) = 0.8 \cdot 1 + 0 = 0.8$

$n = 2$
$f_0(n) = g_0(n) = x(2) = 0$
$f_1(n) = 0 + 0.65 \cdot 1 = 0.65$
$g_1(n) = 0 + 1 = 1$
$f_2(n) = 0.65 - 0.34 \cdot 0.65 = 0.429$
$g_2(n) = -0.34 \cdot 0.65 + 0.65 = 0.429$
$f_3(n) = 0.429 + 0.8 \cdot -0.34 = \mathbf{0.157}$
$g_3(n) = 0.8 \cdot 0.429 - 0.34 = 0.0032$

$n = 4$
$f_0(n) = g_0(n) = x(2) = 0$
$f_1(n) = 0$
$g_1(n) = 0$
$f_2(n) = 0$
$g_2(n) = 0$
$f_3(n) = \mathbf{0.8}$
$g_3(n) = 1$

We can also use the MATLAB function **azlatfilt** given in Figure 9.26 in the book.

```
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

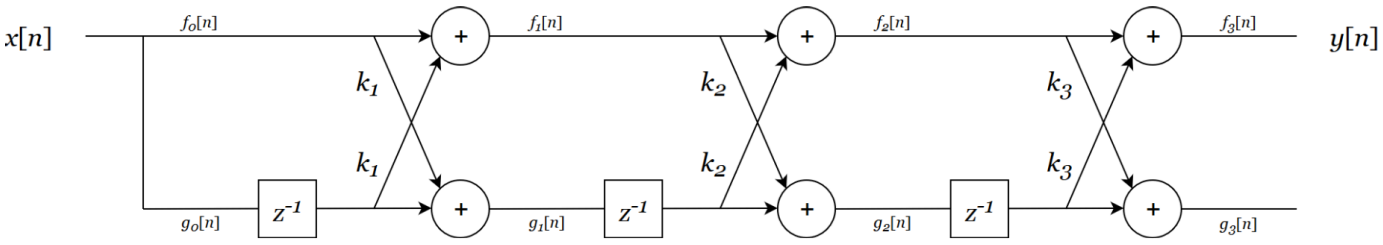
$$H(z) = 1 + \frac{13}{24}z^{-1} + \frac{5}{8}z^{-2} + \frac{1}{3}z^{-3}$$

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

The recursive algorithm works as follows:

1. Compute $A_M(z) = \frac{H(z)}{h[0]}$
2. Compute $k_M = a_M$ where a_M is the last coefficients of $A_M(z)$. For example, if $A_M(z) = 1 + 0.06z^{-1} - 0.42z^{-2} + 0.5z^{-3}$ then $k_M = 0.5$
3. Compute $B_M(z)$ by flipping the coefficients of $A_M(z)$
4. Set $m = M$
5. Compute $A_{m-1}(z) = \frac{1}{1 - k_m^2}[A_m(z) - k_m B_m(z)]$. Notice that this is where the algorithm fails because if $k_m = 1 \rightarrow k_m^2 = 1$ then we will have division by zero.
6. Compute $k_{m-1} = a_{m-1}$ where a_{m-1} is the last coefficients of $A_{m-1}(z)$
7. Compute $B_{m-1}(z)$ by flipping the coefficients of $A_{m-1}(z)$. Alternatively, compute $B_{m-1}(z) = z^{-m-1}A_{m-1}\left(\frac{1}{z}\right)$
8. Set $m = m - 1$
9. Go to step 5 if $m \neq 0$
10. We know that $A_0(z) = B_0(z) = 1$

In this problem, we have a 3-stage all-zero lattice filter:



We are given its transfer function:

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

Since this is a FIR system of the form:

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

We know that $M = 3$, $h[k] = \left[1, \frac{13}{24}, \frac{5}{8}, \frac{1}{8}\right]$ and $h[0] = 1$.

Step 1: Compute $A_3(z)$

$$A_M(z) = A_3(z) = \frac{H(z)}{h[0]} = \frac{H(z)}{1} = 1 + \frac{13}{24}z^{-1} + \frac{5}{8}z^{-2} + \frac{1}{3}z^{-3}$$

Step 2: Compute k_3

The last coefficients of $A_3(z)$ is $\frac{1}{3}$ so $k_3 = \frac{1}{3}$

Step 3: Compute $B_3(z)$

Once we have $A_M(z)$, we can compute $B_M(z)$ by simply flipping the coefficients of $A_M(z)$.

$$B_M(z) = B_3(z) = \frac{1}{3} + \frac{5}{8}z^{-1} + \frac{13}{24}z^{2} + z^{-3}$$

Step 4: Set $m = M$

$$m = 3$$

Step 5: Compute $A_2(z)$

Using the formula $A_{m-1}(z) = \frac{1}{1 - k_m^2}[A_m(z) - k_m B_m(z)]$ we can compute $A_2(z)$

$$A_2(z) = \frac{1}{1 - k_3^2}[A_3(z) - k_3 B_3(z)]$$

$$A_2(z) = \frac{1}{1 - \left(\frac{1}{3}\right)^2}\left[\left(1 + \frac{13}{24}z^{-1} + \frac{5}{8}z^{-2} + \frac{1}{3}z^{-3}\right) - \frac{1}{3}\left(\frac{1}{3} + \frac{5}{8}z^{-1} + \frac{13}{24}z^{2} + z^{-3}\right)\right]$$

$$A_2(z) = \frac{9}{8}\left[1 - \frac{1}{3}\cdot\frac{1}{3} + \left(\frac{13}{24} - \frac{1}{3}\cdot\frac{5}{8}\right)z^{-1} + \left(\frac{5}{8} - \frac{1}{3}\cdot\frac{13}{24}\right)z^{-2} + \left(\frac{1}{3} - \frac{1}{3}\right)z^{-3}\right]$$

$$A_2(z) = \frac{9}{8}\left[\frac{8}{9} + \frac{1}{3}z^{-1} + \frac{4}{9}z^{-2} + 0z^{-3}\right]$$

$$A_2(z) = 1 + \frac{3}{8}z^{-1} + \frac{1}{2}z^{-2}$$

Step 6: Compute k_2

We can read of the last coefficient of $A_2(z)$ which is $k_2 = \frac{1}{2}$

Step 7: Compute $B_2(z)$

Once we have $A_2(z)$, we can compute $B_2(z)$ by simply flipping the coefficients of $A_2(z)$.

$$B_2(z) = \frac{1}{2} + \frac{3}{8}z^{-1} + z^{-2}$$

Step 8: Set $m = 2$

Step 9: Go to step 5

Step 5: Compute $A_1(z)$

We can compute $A_1(z)$ as follows

$$A_1(z) = \frac{1}{1 - k_2^2}[A_2(z) - k_2 B_2(z)]$$

$$A_1(z) = \frac{1}{1 - \left(\frac{1}{2}\right)^2}\left[\left(1 + \frac{3}{8}z^{-1} + \frac{1}{2}z^{-2}\right) - \frac{1}{2}\left(\frac{1}{2} + \frac{3}{8}z^{-1} + z^{-2}\right)\right]$$

$$A_1(z) = \frac{4}{3}\left[1 - \frac{1}{2}\cdot\frac{1}{2} + \left(\frac{3}{8} - \frac{1}{2}\cdot\frac{3}{8}\right)z^{-1} + \left(\frac{1}{2} - \frac{1}{2}\right)z^{-2}\right]$$

$$A_1(z) = \frac{4}{3}\left[\frac{3}{4} + \frac{3}{16}z^{-1} + 0z^{-2}\right]$$

$$A_1(z) = 1 + \frac{1}{4}z^{-1}$$

Step 6: Compute k_1

We can read of the last coefficient of $A_1(z)$ which is $k_1 = \frac{1}{4}$

We are done! The result is $k_1 = \frac{1}{4}$, $k_2 = \frac{1}{2}$ and $k_3 = \frac{1}{3}$

Let us verify our result using 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 if one of the reflection coefficients is equal to 1. In this problem, $k_2 = 1$ so if we attempt to compute $A_1(z)$, it will lead to division by zero:

$$A_1(z) = \frac{1}{1 - k_2^2}[A_2(z) - k_2 B_2(z)] = \frac{1}{1 - 1}[A_2(z) - k_2 B_2(z)]$$

We can try out it out in MATLAB function:

```
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 factorise the original FIR filter to two cascade filters. This can be done by finding the roots and using the formula:

$H(z) = \left(1 - r_1 z^{-1}\right)\left(1 - r_2 z^{-1}\right) \cdots \left(1 - r_M z^{-1}\right)$ where r_i are the roots of the transfer function.

Find the roots:

```
roots(b)
```

```
ans = 2×1
    -1
    -1
```

Factorise the original FIR filter into two cascading filters:

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

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

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

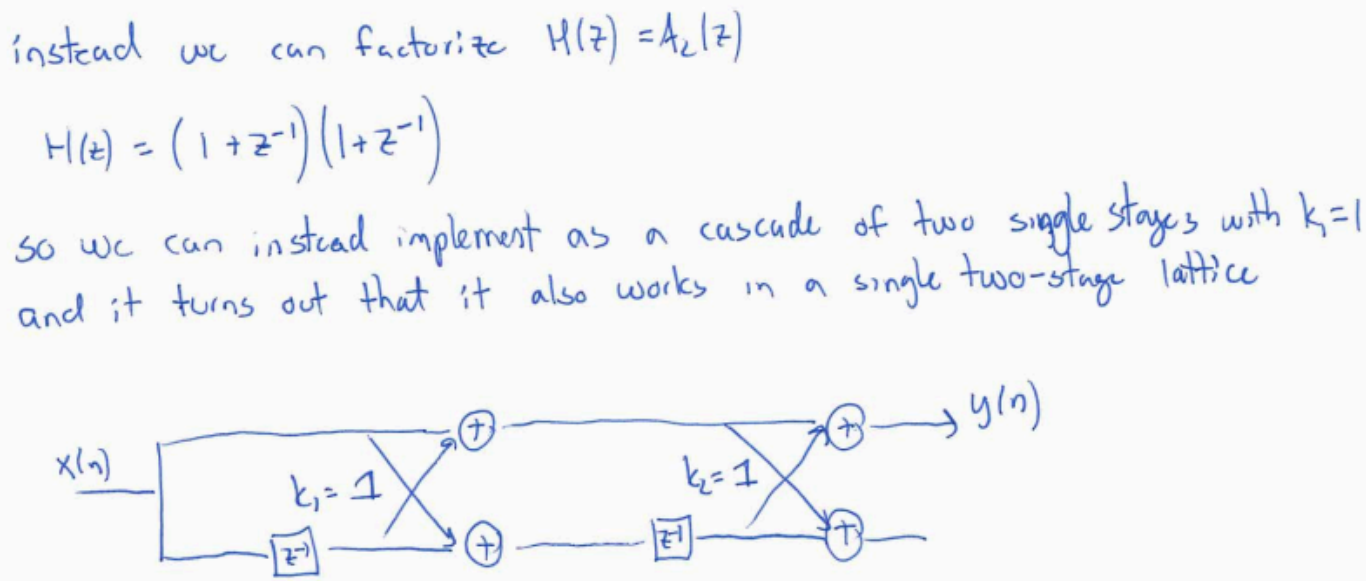
Taking the z-transform we get:

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

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

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

It turns out that we can cascade two single stage all-zero lattice filters the same way as we cascade FIR filters.



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 do this by hand using following two formulas:

$$f_{m-1}[n] = f_m[n] - k_m g_{m-1}[n-1]$$

$$g_m[n] = k_m f_{m-1}[n] + g_{m-1}[n-1]$$

We start computing the difference equations of the last section:

$$f_0[n] = f_1[n] - k_1 g_0[n-1]$$

$$g_1[n] = k_1 f_0[n] + g_0[n-1]$$

Then we compute the next stage:

$$f_1[n] = f_2[n] - k_2 g_1[n-1]$$

$$g_2[n] = k_2 f_1[n] + g_1[n-1]$$

Next, we can substitute $f_1[n]$ into the expression $f_0[n]$:

$$f_0[n] = f_2[n] - k_2 g_1[n-1] - k_1 g_0[n-1]$$
$$f_0[n] = f_2[n] - k_2(k_1 f_0[n-1] + g_0[n-2]) - k_1 g_0[n-1]$$

And so on.

Finally, we will get a large expression in which we replace:

- $g_0[n]$ and $f_0[n]$ with $y[n]$
- $f_4[n]$ with $x[n]$

Alternatively, we can use MATLAB to compute the transfer function:

```
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

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

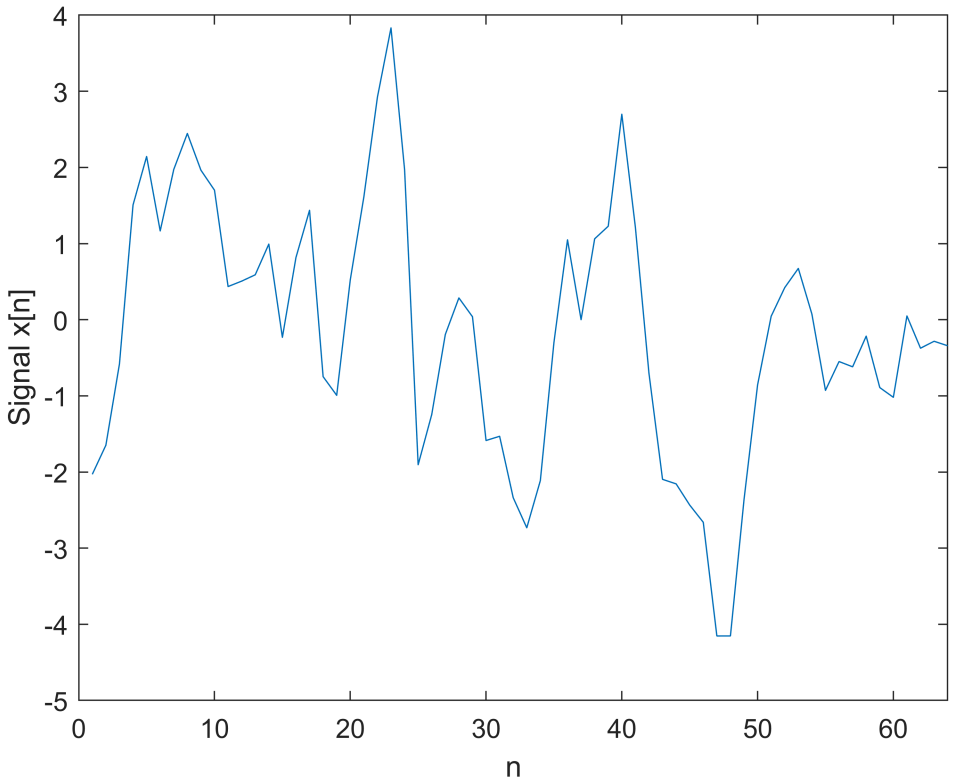
1) Compute autocorrelation from a signal

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?

Let us first plot the signal.

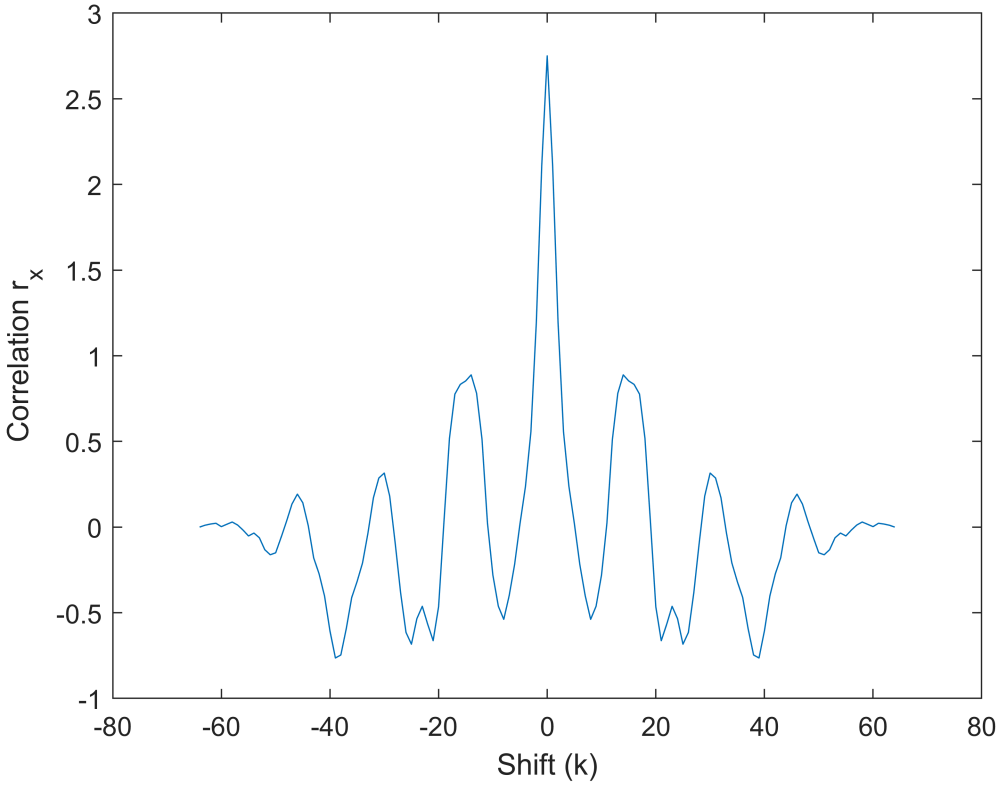
```
x = importdata('signal1.dat');
plot(x);
xlabel('n');
ylabel('Signal x[n]');
xlim([0, numel(x)]);
```



Autocorrelation is the cross-correlation between a signal $x[n]$ with the same signal shifted by some amount $x[n-k]$. So if we start with $k=1$, we compute the correlation between $x[n]$ and $x[n-1]$. For $k=2$, we compute the correlation between $x[n]$ and $x[n-2]$ etc.

Autocorrelation is symmetrical around zero.

```
[r,lags] = xcorr(x,64,'biased');
plot(lags,r);
xlabel('Shift (k)');
ylabel('Correlation r_x');
```



From the plot, we see that when the signal is shifted e.g. by 4, the correlation is low.

```
r(4)
```

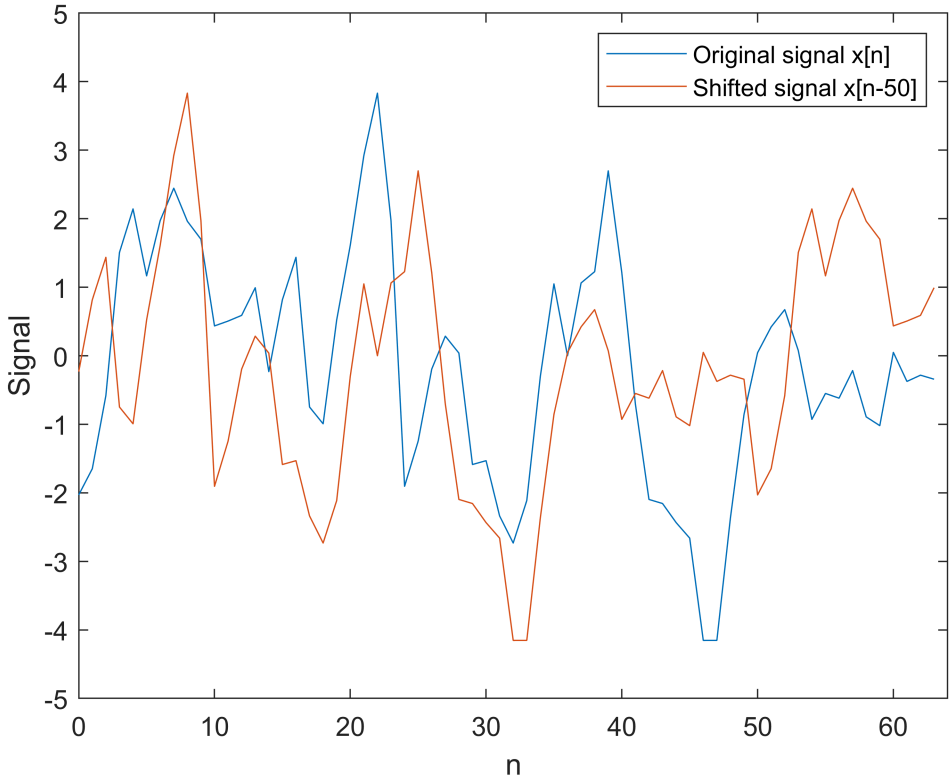
```
ans =
        0.0222826167922265
```

```
r(50)
```

```
ans =
        0.853325213049691
```

This means that the original signal $x[n]$ and the shifted signal $x[n-4]$ are not very similar. The correlation is higher when the signal is shifted by 50:

```
n = linspace(0, 63, 64)';
x_shifted_50 = circshift(x, 50);
plot(n, x, n, x_shifted_50);
legend('Original signal x[n]','Shifted signal x[n-50]')
xlabel('n');
ylabel('Signal');
xlim([0, numel(x)]);
ylim([-5, 5])
```



Shifting the signal by 64 yields the highest correlation. This makes sense because if we shift a signal of length 64 by 64 then we get the original signal $x[n] = x[n-64]$.

```
r(64+1)
```

```
ans =
           2.7506687624622
```

2) Compute the coefficients for the optimum 2nd-order linear predictor

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)

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

Let $p = 2$.

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

When $p = 2$ then we get following two equations:

$r_x(1) = -a_2(1)r_x(0) - a_2(2)r_x(-1)$
$r_x(2) = -a_2(1)r_x(1) - a_2(2)r_x(0)$

In exercise 1) we computed the correlation values r_x. So we know the values for $r_x(-1)$, $r_x(0)$, $r_x(1)$ and $r_x(2)$. This means that we have to solve two equations with two unknowns.

Let us rewrite it:

$r_x(1) + a_2(1)r_x(0) + a_2(2)r_x(-1) = 0$
$r_x(2) + a_2(1)r_x(1) + a_2(2)r_x(0) = 0$

Solve it in MATLAB:

```
syms a2_1 a2_2

% The autocorrelation is centered around zero
% Find the index where n=0
centerIx = find(lags==0);

% Retrive the values r(-1), r(0), r(1) and r(2) in order
rx_m1 = r(centerIx - 1);
rx_0 = r(centerIx + 0);
rx_1 = r(centerIx + 1);
rx_2 = r(centerIx + 2);

eq1 = rx_1 + a2_1*rx_0 + a2_2*rx_m1;
eq2 = rx_2 + a2_1*rx_1 + a2_2*rx_0;

sol = solve([eq1, eq2], [a2_1, a2_2]);

a1 = sol.a2_1;
a2 = sol.a2_2;
% Print the value of a2(1)
vpa(sol.a2_1)
```

```
ans = −1.055448422597235773335769905207 46
```

```
% Print the value of a2(2)
vpa(sol.a2_2)
```

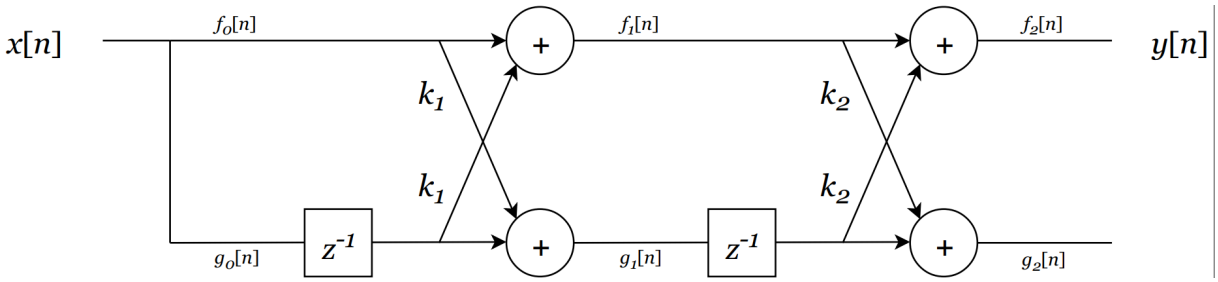
```
ans = 0.375305905228859651705069270410 28
```

The coefficents of the predictor are $a_2(1) \approx -1.0554$ and $a_2(2) \approx 0.3753$

3) Compare the linear predictor with a two-stage all-zero lattice filter

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?



We know that the output of a two-stage all-zero lattice filter is:

$$y[n] = x[n] + k_1(1 + k_2)x[n-1] + k_2 x[n-2]$$

The p'th order linear predictor has the following form:

$$\hat{x}[n] = \sum_{k=1}^{p} a_k \cdot [n-k]$$

So the second-order linear prediction has the form:

$$\hat{x}[n] = a_1 x[n-1] + a_2 x[n-2]$$

Notice that 2nd order linear predictor looks very much like the difference equation for the two-stage all-zero lattice filter.

There is a one-to-one correspondance between their coefficients where $a_1 = k_1(1 + k_2)$ and $a_2 = k_2$.

4) Compute reflection coefficients from the linear predictor

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

From 3) we found that $a_1 = k_1(1 + k_2)$ and $a_2 = k_2$ so:

$$k_2 = a_2$$

$$k_1 = \frac{a_1}{1 + a_2}$$

In 2) we found that values for a_1 and a_2. We will use these to compute k_1 and k_2:

```
k1 = a2/(1+a2);
k2 = a2;
vpa(k1)
```

```
ans = 0.27288903785111455734521239333629
```

```
vpa(k2)
```

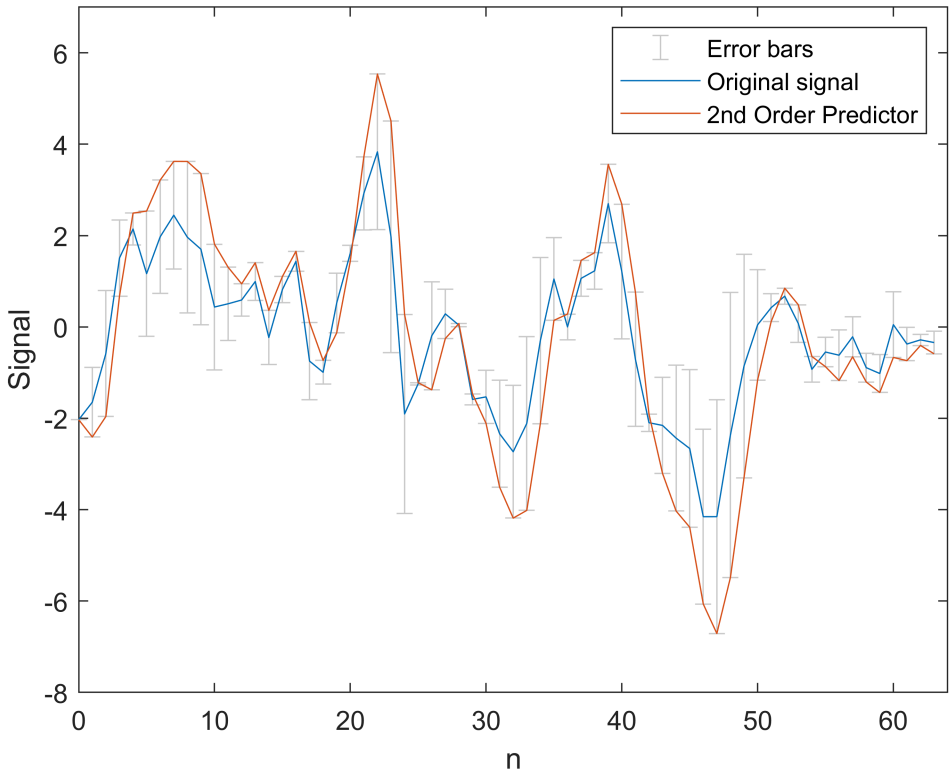
```
ans = 0.37530590522885965170506927041028
```

So we have $k_1 \approx 0.272889$ and $k_2 \approx 0.3753$

5) Compute prediction error

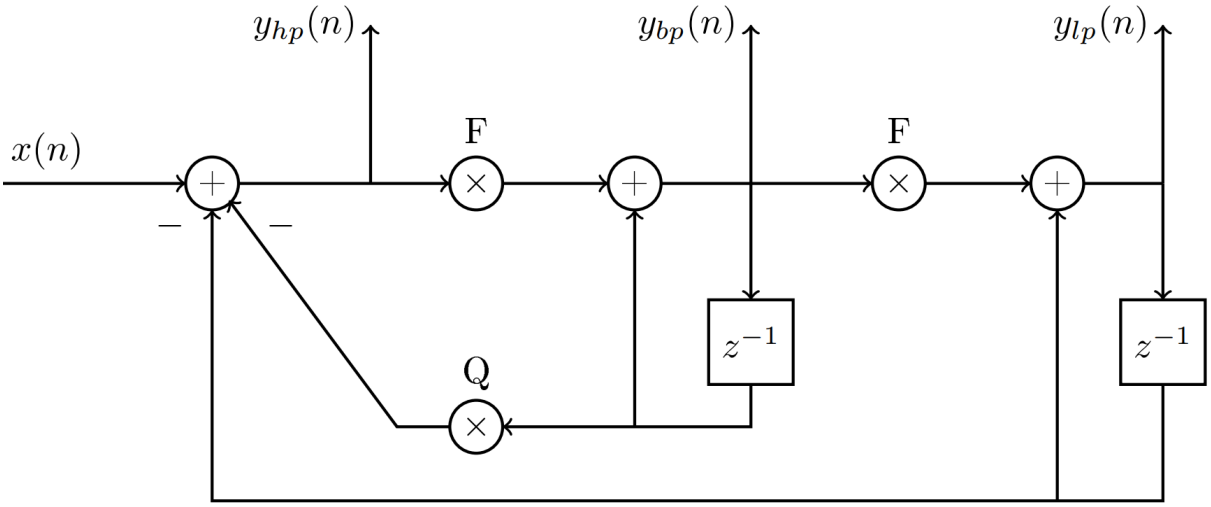
5. Send the signal through a 2nd order all-zero lattice filter and analyze the prediction error. Is the signal predictable? Repeat for a higher order predictor.

```
figure;

% Compute x_hat and err
n = linspace(0, 63, 64);
x = importdata('signal1.dat');
x_hat = azlatfilt([k1, k2], x, 1);
err = x_hat - x;

% Plot
errorbar(n, x, err, 'LineStyle','none', 'Color', [0.8, 0.8, 0.8]);

hold on;
plot(n, x);
plot(n, x_hat);
hold off;

xlabel('n');
ylabel('Signal');
xlim([0, numel(x)]);
ylim([-8, 7])
legend('Error bars', 'Original signal', '2nd 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
```