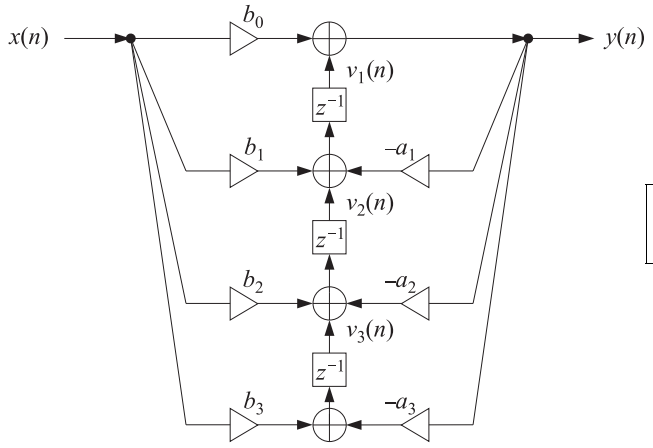
# *Lab 6 – IIR Filtering Experiments*

## *6.1. Transposed Realization*

In Lab-4 you implemented your own version of the function **filter**, specialized to FIR filters, using the transposed realization. In this lab you will implement the IIR case. The transposed realization block diagram is shown in the MATLAB documentation for the function **filter**. An equivalent version is shown below for an order-3 filter.




$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}$$

For an $M$th order filter, the computational algorithm is,

$$\begin{aligned}
y(n) &= v_1(n) + b_0 x(n) \\
v_1(n+1) &= v_2(n) + b_1 x(n) - a_1 y(n) \\
v_2(n+1) &= v_3(n) + b_2 x(n) - a_2 y(n) \\
&\vdots \\
v_{M-1}(n+1) &= v_M(n) + b_{M-1} x(n) - a_{M-1} y(n) \\
v_M(n+1) &= b_M x(n) - a_M y(n)
\end{aligned}$$

$$\mathbf{v}(n) = \begin{bmatrix} v_1(n) \\ v_2(n) \\ \vdots \\ v_M(n) \end{bmatrix} = \text{state vector}$$

The state vector $\mathbf{v}(n)$ represents the current contents of the $M$ delays. The algorithm is an example of a state-space realization. It computes the current output $y(n)$ from the current input $x(n)$ and the current state $\mathbf{v}(n)$ , and then, it updates the state to the next time instant, $\mathbf{v}(n+1)$. Usually, the state vector is initialized to zero, but it can be initialized to an arbitrary vector, say, $\mathbf{v}_{\text{in}}$.

**Lab Procedure**

a. Write a MATLAB function, say, **filtr.m**, that is functionally equivalent to the built-in function **filter** and implements the above algorithm and has the possible syntaxes:

```
y = filtr(b,a,x);
[y,vout] = filtr(b,a,x,vin);

%   b  = order-L numerator coefficient vector,   b = [b0, b1, b2, ..., bL]
%   a  = order-K denominator coefficient vector, a = [ 1, a1, a2, ..., aK]
%   x  = length-N vector of input samples (row or column)
%   y  = length-N vector of output samples (row or column according to x)
% vin  = M-dimensional vector of initial states - zero vector, by default
% vout = M-dimensional final state vector, i.e., final contents of delays
```

Internally, yout function must extend $\mathbf{b}, \mathbf{a}$ to the maximum required order $M = \max(L, K)$ by padding zeros. Test your function on the following case:

```
b = [1,  0.6, -0.4];              % numerator
a = [1, -1.3,  0.2,  0.4];        % denominator
x = [1, 1, 2, 1, 2, 2, 1, 1];
y = [1, 2.9000, 5.7700, 8.3210, 10.3033, 12.2221, 11.8997, 9.7038];    % expected result
```

b. The function **filtr** can be run on a sample by sample basis to generate the successive internal state vectors, for example, using a loop such as,

```
v = zeros(1,M);                          % initial state vector
for n=1:length(x)
   [y(n),vout] = filtr(b,a,x(n),v);      % recycled state vector v
   v = vout;                             % next state
end
```

Add appropriate **fprintf** commands before, within, and after this loop to generate the following table of values for the above example,

```
 n    x       y        v1        v2        v3
-------------------------------------------------
 0    1    1.0000    0.0000    0.0000    0.0000
 1    1    2.9000    1.9000   -0.6000   -0.4000
 2    2    5.7700    3.7700   -1.3800   -1.1600
 3    1    8.3210    7.3210   -3.1140   -2.3080
 4    2   10.3033    8.3033   -4.3722   -3.3284
 5    2   12.2221   10.2221   -6.1891   -4.1213
 6    1   11.8997   10.8997   -7.3657   -4.8888
 7    1    9.7038    8.7038   -7.6688   -4.7599
 8    -       -      5.5462   -7.1006   -3.8815
```

where $v_1, v_2, v_3$ are the internal states at each time instant.

c. The effective *time constant* of a stable and causal IIR filter, measured in samples, may be defined in terms of the maximum pole radius as follows,

$$n_{\text{eff}} = \frac{\ln \epsilon}{\ln R}, \quad R = \max_i |p_i| \tag{6.1}$$

where $p_i$ are the poles of the filter, and $\epsilon$ is a small number, such as $\epsilon = 10^{-2}$ or $\epsilon = 10^{-3}$ for the so-called 40-dB or 60-dB time constants, respectively.

Calculate the 40-dB time constant of the example filter of part (a), and then using your function **filtr**, evaluate and plot its *impulse response* $h_n$ over the interval $0 \le n \le n_{\text{eff}}$. Similarly, evaluate and plot the *unit-step response* over the same time interval. Note that the asymptotic value of the unit-step response (for a stable/causal filter) is given in terms of the filter coefficients by

$$u_\infty = \frac{\sum_{m=0}^{M} b_m}{\sum_{m=0}^{M} a_m}$$

Indicate this value on your graph of the unit-step response.

d. *Envelope detectors* are used extensively in communication systems and in audio effects such as compressors and expanders. A typical envelope detector is a lowpass filter acting on the absolute value of a signal.

In this part, you will determine the envelope of the impulse response $h_n$ that was calulated in the part (c). Consider the following IIR and FIR lowpass filtering operations on the signal $|h_n|$:

$$\begin{aligned} \text{(IIR)} \qquad & h_{\text{env}}(n) = a\,h_{\text{env}}(n-1) + (1-a)\,|h_n| \\ \text{(FIR)} \qquad & h_{\text{env}}(n) = \frac{|h_n| + |h_{n-1}| + |h_{n-2}| + \cdots + |h_{n-N+1}|}{N} \end{aligned} \tag{6.2}$$

where $0 < a < 1$ and $N$ is an integer. It can be shown that the two filters are roughly equivalent if one makes the identification

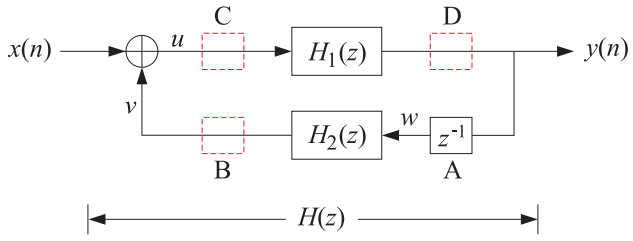
$$N = \text{floor}\left(\frac{1+a}{1-a}\right)$$

For $a = 0.80$, calculate $N$, and carry out the filtering operations of Eq. (6.2). Plot $h(n)$ and $h_{\text{env}}(n)$ over the time interval $0 \le n \le n_{\text{neff}}$. Make different plots for the IIR and the FIR envelopes.

Repeat for $a = 0.94$. Comment on the tradeoffs between the two choices $a = 0.80$ and $a = 0.94$ in terms of their initial transients and their ability to extract the envelope.

## 6.2. Feedback System

A general feedback system is shown below, where the output of filter $H_1(z)$ is fed back into filter $H_2(z)$ and then back to the input, and where the delay $z^{-1}$ can be positioned at the four locations A, B, C, or D.




**Lab Procedure**

a. For each case A,B,C,D, determine the transfer function $H(z)$ of the overall closed-loop system from $x(n)$ to $y(n)$, expressed in terms of the transfer functions $H_1(z)$ and $H_2(z)$.

b. Choose the forward and feedback transfer functions as,

$$H_1(z) = \frac{1+z^{-1}}{1-0.5z^{-1}}, \qquad H_2(z) = \frac{0.4-0.4z^{-1}}{1-0.4z^{-1}} \tag{6.3}$$

Show that for case A, the transfer function $H(z)$ is,

$$H(z) = \frac{1+0.6z^{-1}-0.4z^{-2}}{1-1.3z^{-1}+0.2z^{-2}+0.4z^{-3}} \tag{6.4}$$

What are the transfer functions $H(z)$ for the cases B, C, D?

c. The objective of this part is to use the function **filtr** on a sample-by-sample basis, applied separately on the transfer functions $H_1(z)$ and $H_2(z)$, to implement the feedback system directly, without transforming it first into the equivalent transfer function $H(z)$.

As we saw earlier, the filtering function **filtr** can be invoked repetitively on a sample-by-sample basis using a for-loop that recycles the state vector:

```
s = zeros(1,M);                      % initial state vector
for n=1:length(x)
   [y(n),s] = filtr(b,a,x(n),s);     % recycled state vector s
end
```

To implement the transfer functions $H_1(z)$ and $H_2(z)$ of part (b), you need to define their filter coefficients and states, $\{\mathbf{b}_1, \mathbf{a}_1, \mathbf{s}_1\}$ and $\{\mathbf{b}_2, \mathbf{a}_2, \mathbf{s}_2\}$, and initialize the states to zero, as well as the content of the delay at locations A,B,C,D. For example, initially the content $w$ of the delay at A shown in the above block diagram will be zero. As an example, the pseudo-code implementation of case A will be:

initialize: $w = 0,\ s_1 = 0,\ s_2 = 0$

for each input sample $x$ do:

$[v, s_2] =$ output and state of $H_2$ with input sample $w$ and state $s_2$

$u = x + v$

$[y, s_1] =$ output and state of $H_1$ with input sample $u$ and state $s_1$

$w = y$

Implement this case in a for-loop in Matlab and evaluate the output $y(n)$ when the input is selected to be:

```
x = [1, 1, 2, 1, 2, 2, 1, 1];
```

Check your output by also computing it using the overall transfer function $H(z)$ of Eq. (6.4).

Similarly, formulate the computational steps using a for-loop for cases B, C, D, and calculate the corresponding output signals. Note that each case requires a different ordering of the computations.

Using appropriate **fprintf** commands, print all cases A, B, C. D in a table as shown below, where the second column y represents the output of case A computed with Eq. (6.4),

```
    x         y         yA         yB         yC         yD
-------------------------------------------------------------
 1.0000    1.0000    1.0000    1.0000    0.0000    0.0000
 1.0000    2.9000    2.9000    2.9000    1.0000    1.0000
 2.0000    5.7700    5.7700    5.7700    2.9000    2.9000
 1.0000    8.3210    8.3210    8.3210    5.7700    5.7700
 2.0000   10.3033   10.3033   10.3033    8.3210    8.3210
 2.0000   12.2221   12.2221   12.2221   10.3033   10.3033
 1.0000   11.8997   11.8997   11.8997   12.2221   12.2221
 1.0000    9.7038    9.7038    9.7038   11.8997   11.8997
```

## Example Graphs