First-Order Low/Highpass Filter Design

Yangang Cao

February 13, 2019

The signal can be seen as a set of partials having different frenquencies and amplitudes. The filter can modify the amplitude of partials according to their frenquency.

1 Definition of Low/Highpass Filters

The two types of filters can be defined according to the following classification:

- **Lowpass (LP)** filters select low frenquencies up to the cut-off frenquency f_c and attenuate frenquencies higher than f_c. Additionally, a resonance may amply frenquencies around f_c.
- **Highpass (HP)** filters select high frenquencies higher than f_c and attenuate frenquencies below f_c, possibly with a resonance around f_c.

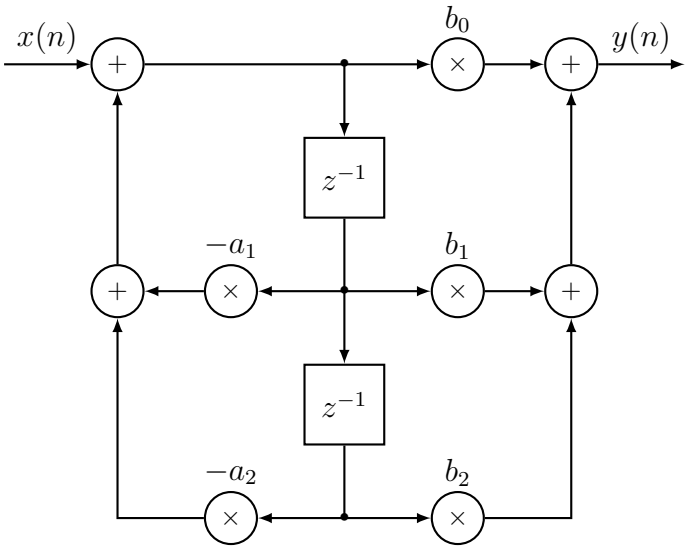
The lowpass with resonance is very often used in computer music to simulate an acoustical resonating structure; the highpass filter can remove undesired very low frequencies.

2 Canonical Form

There are various ways to implement a filter, the simplest being the canonical filter, as shown in following figure for a second-order filter, which can be implemented by the defference equations

$$x_h(n) = x(n) - a_1 x_h(n-1) - a_2 x_h(n-2)$$

$$y(n) = b_0 x_h(n) + b_1 x_h(n-1) + b_2 x_h(n-2),$$



and leads to the transfer function by setting $a_2 = b_2 = 0$, this reduces to a first-order filter which, can be used to implement an allpass, lowpass or highpass with the coefficients of following table

	b_0	b_1	a_1
Lowpass	K/(K+1)	K/(K+1)	(K-1)/(K+1)
Highpass	1/(K+1)	-1/(K+1)	(K-1)/(K+1)
Allpass	(K-1)/(K+1)	1	(K-1)/(K+1)

where K depends on the cut-off frequency f_c by

$$K = \tan(\pi f_c / f_S).$$

3 First-Order Allpass-Based Filter

In this section we introduce a special class of parametric filter structures for lowpass, highpass, bandpass and bandreject filter functions. These filter structures are easily tunable by changing only one or two coefficients. They play an important role for real-time control with minimum computational complexity.

The basis for parametric first- and second-order IIR filters is the first- and second-order allpass filter. We will first discuss the first-order allpass and show simple lowpass and highpass filters, which consist of a tunable allpass filter together with a direct path.

A first-order allpass filter is given by the transfer function

$$A(z) = \frac{z^{-1} + c}{1 + cz^{-1}}$$

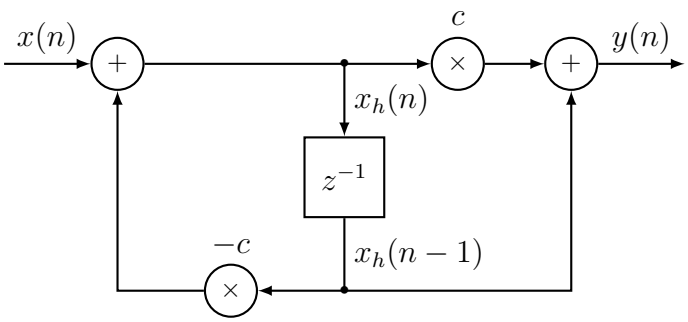
$$c = \frac{\tan(\pi f_c / f_S) - 1}{\tan(\pi f_c / f_S) + 1},$$

and the corresponding difference equations

$$x_h(n) = x(n) - cx_h(n-1)$$

$$y(n) = cx_h(n) + x_h(n-1),$$

which can be realized by the following block diagram,



and corresponding state and output equations are:

$$x_h(n) = -cx_h(n-1) + x(n)$$

$$y(n) = (1-c^2)x_h(n-1) + cx(n).$$

Actually, the state and output equations represent the same meaning as the difference equations, but difference in format. The difference equations are usually easy to comprehend, howere, state and output equations are more widely used in modern control theory, so we implement all kinds of filters by this format. The first-order allpass filter implementation can be obtained by the following **Matlab** code.

```
function y = firstallpass(audio, para)
% y = firstallpass(audio, para)
% Author: Yangang Cao
% Applies a allpass filter to the input signal.
% para is the normalized cut-off frequency in (0,1)
c = (tan(pi*para/2)-1) / (tan(pi*para/2)+1);
x = 0;
x_1 = 0;
for n = 1:length(audio)
    x_1 = -c * x + audio(n);
    y(n) = (1 - c^2) * x + c * audio(n);
    x = x_1;
end
```

4 First-Order Low/Highpass Filter

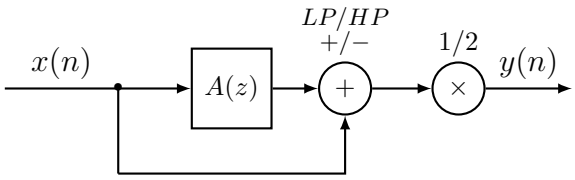
A first-order lowpass/highpass filter can be achieved by adding or subtracting (+/−) the output signal from the input signal of a first-order allpass filter.The transfer function of a lowpass/highpass filter is then given by

$$H(z) = \frac{1}{2}(1 \pm A(z)) \quad (LP/HP + /-)$$

$$A(z) = \frac{z^{-1} + c}{1 + cz^{-1}}$$

$$c = \frac{\tan(\pi f_c/f_S) - 1}{\tan(\pi f_c/f_S) + 1}.$$

where a tunable first-order allpass $A(z)$ with tuning parameter c is used. The plus sign (+) denotes the lowpass operation and the minus sign (–) the highpass operation. The block diagram in following figure represents the operations involved in performing the low/highpass filtering.



The difference equations of first-order lowpass filter are:

$$x_h(n) = x(n) - cx_h(n-1)$$
$$y(n) = \frac{1+c}{2}x_h(n) + \frac{1+c}{2}x_h(n-1),$$

and corresponding state and output equations are:

$$x_h(n) = -cx_h(n-1) + x(n)$$
$$y(n) = \frac{1-c^2}{2}x_h(n-1) + \frac{1+c}{2}x(n).$$

A first-order lowpass filter implementation can be obtained by the following **Matlab** code.

```
function y = aplowpass(audio, para)
% y = aplowpass(audio, para)
% Author: Yangang Cao
% Applies a lowpass filter to the input signal.
% para is the normalized cut-off frequency in (0,1)
c = (tan(pi*para/2)-1) / (tan(pi*para/2)+1);
x = 0;
x_1 = 0;
for n = 1:length(audio)
    x_1 = -c * x + audio(n);
    y(n) = ((1-c^2)/2) * x + (1+c)/2 * audio(n);
    x = x_1;
end
```

The difference equations of first-order highpass filter are:

$$x_h(n) = x(n) - cx_h(n-1)$$
$$y(n) = \frac{1-c}{2}x_h(n) + \frac{c-1}{2}x_h(n-1),$$

and corresponding state and output equations are:

$$x_h(n) = -cx_h(n-1) + x(n)$$
$$y(n) = \frac{c^2-1}{2}x_h(n-1) + \frac{1-c}{2}x(n).$$

A first-order highpass filter implementation can be obtained by the following **Matlab** code.

```
function y = aphighpass(audio, para)
% y = aphighpass(audio, para)
% Author: Yangang Cao
% Applies a highpass filter to the input signal.
% para is the normalized cut-off frequency in (0,1)
c = (tan(pi*para/2)-1) / (tan(pi*para/2)+1);
x = 0;
x_1 = 0;
for n = 1:length(audio)
    x_1 = -c * x + audio(n);
    y(n) = ((c^2-1)/2) * x + (1-c)/2 * audio(n);
    x = x_1;
end
```