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. 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.

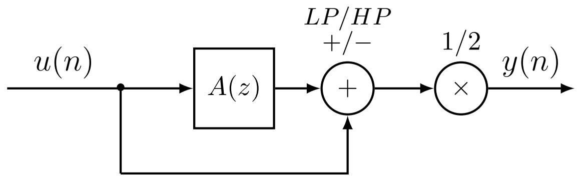
A first-order lowpass/highpass filter can be achieved by adding or subtracting (+/−) the output signal from the input signal of a first-order allpass filter.The transfer function of a lowpass/highpass filter is then given by

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

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

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

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

$$x(n) = u(n) - cx(n-1)$$

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

and corresponding state and output equations are

$$x(n) = -cx(n-1) + u(n)$$

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

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

```
function y = aplowpassunit(audio, para)
% 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

$$x(n) = u(n) - cx(n-1)$$

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

and corresponding state and output equations are

$$x(n) = -cx(n-1) + u(n)$$

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

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

```
function y = aphighpassunit(audio, para)
% 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
```