First-Order Low/Highpass Filter Design

Yangang Cao

February 13, 2019

1. Definition of Low/Highpass Filters

2. First-Order Allpass Filter

3. First-Order Low/Highpass Filter
   - first-order lowpass filter
   - first-order highpass filter

1. Definition of Low/Highpass Filters

2. First-Order Allpass Filter

3. First-Order Low/Highpass Filter
   first-order lowpass filter
   first-order highpass filter

Definition of low/highpass filter:

- **Lowpass (LP)** filters select low frenquencies up to the cut-off frenquency f_c and attenuate frenquencies higher than f_c.
- **Highpass (HP)** filters select high frenquencies higher than f_c and attenuate frenquencies below f_c.

1. Definition of Low/Highpass Filters

2. First-Order Allpass Filter

3. First-Order Low/Highpass Filter
   first-order lowpass filter
   first-order highpass filter

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

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

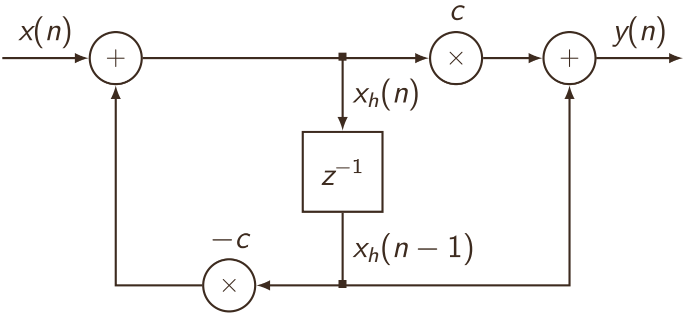
$$c = \frac{\tan(\pi f_c / f_S) - 1}{\tan(\pi f_c / f_S) + 1}$$

and the corresponding difference equations

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

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

Block diagram of first-order allpass filter



Corresponding state and output equations are:

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

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

Matlab code:

```
function y = firstallpassunit(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
```

1. Definition of Low/Highpass Filters

2. First-Order Allpass Filter

3. First-Order Low/Highpass Filter
   - first-order lowpass filter
   - first-order highpass filter

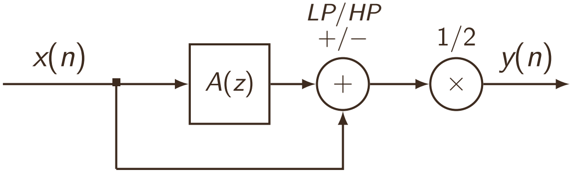
A first-order lowpass/highpass filter can be achieved by adding or subtracting $(+/-)$ the output signal from the input signal of a first-order allpass filter:

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

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

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

Block diagram of first-order low/highpass filter



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:

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

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

Matlab code:

```matlab
function y = aplowpassunit(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:

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

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

Matlab code:

```
function y = aphighpassunit(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
```