CM3106 Chapter 7: Digital Audio Effects

Prof David Marshall
dave.marshall@cs.cardiff.ac.uk
and
Dr Kirill Sidorov
K.Sidorov@cs.cf.ac.uk
www.facebook.com/kirill.sidorov

School of Computer Science & Informatics
Cardiff University, UK

Digital Audio Effects

Having learned to make basic sounds from basic waveforms and more advanced synthesis methods lets see how we can at some digital audio effects.

These may be applied:

- As part of the audio creation/synthesis stage — to be subsequently filtered, (re)synthesised
- At the end of the *audio chain* — as part of the production/mastering phase.
- Effects can be applied in different orders and sometimes in a *parallel* audio chain.
- The order of applying the same effects can have drastic differences in the output audio.
- Selection of effects and the ordering is a matter for the sound you wish to create. There is no absolute rule for the ordering.

FX Pipeline

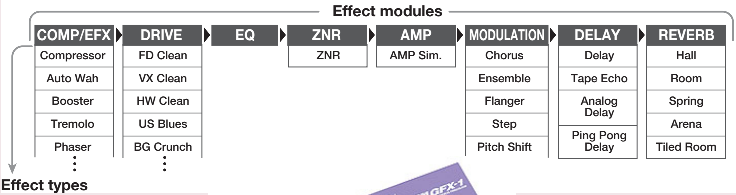
Apply effects in which order?

Some ordering is *standard* for some audio processing, *E.g*:

Compression → Distortion → EQ → Noise Redux → Amp Sim → Modulation → Delay → Reverb

Can also be configurable.

Common for order guitar (and other sources) effects pedal:



Effects Types

Audio effects can be classified by the way process signals:

Basic Filtering: Lowpass, Highpass filter etc., Equaliser

Time Varying Filters: Wah-wah, Phaser

Delays: Vibrato, Flanger, Chorus, Echo

Modulators: Ring modulation, Tremolo, Vibrato

Non-linear Processing: Compression, Limiters, Distortion, Exciters/Enhancers

Spacial Effects: Panning, Reverb, Surround Sound

Basic Digital Audio Filtering Effects: Equalisers

Filtering:

Filters by definition **remove/attenuate** audio from the spectrum above or below some cut-off frequency.

- For many audio applications this a little too restrictive

Equalisation:

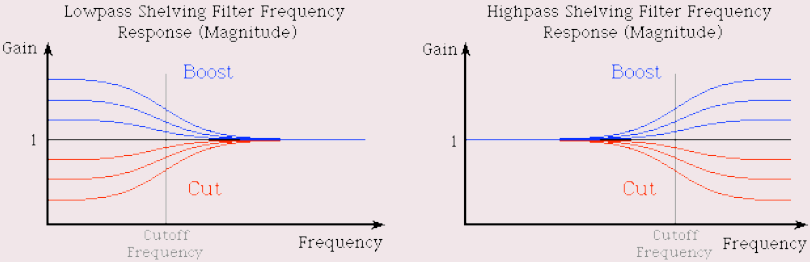
Equalisers, by contrast, **enhance/diminish** certain frequency bands whilst leaving others **unchanged**:

- Built using a series of *shelving* and *peak* filters
- First or second-order filters usually employed.

Shelving and Peak Filters

Shelving Filter:

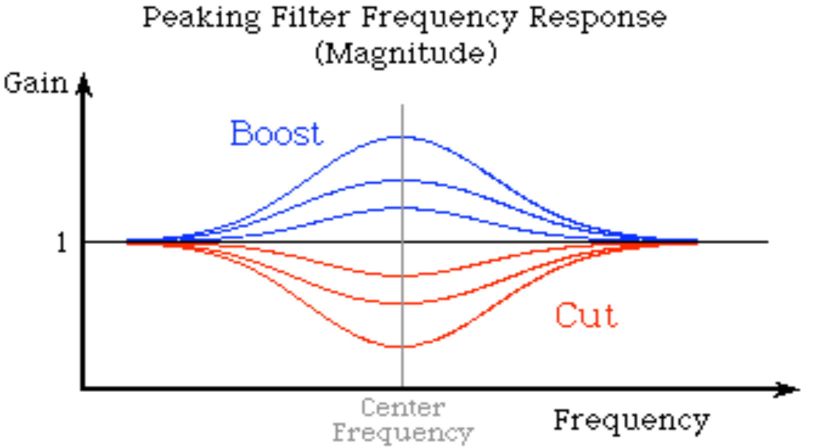
Boost or **cut** the **low** or **high frequency bands** with a **cut-off frequency**, F_c and gain G:



Shelving and Peak Filters (Cont.)

Peak Filter:

Boost or **cut** **mid-frequency** bands with a **cut-off frequency**,F_c, a **bandwidth**, f_b and **gain** G:



Shelving Filters

A First-order Shelving Filter:

Transfer function:

$$H(z) = 1 + \frac{H_0}{2}(1 \pm A(z)) \quad \textbf{where } LF/HF +/-$$

where $A(z)$ is a first-order **allpass** filter — passes all frequencies but modifies phase:

$$A(z) = \frac{z^{-1} + a_{B/C}}{1 + a_{B/C}z^{-1}} \quad \textbf{B=Boost, C=Cut}$$

which leads the following **algorithm**/**difference equation**:

$$\begin{aligned} y_1(n) &= a_{B/C}x(n) + x(n-1) - a_{B/C}y_1(n-1) \\ y(n) &= \frac{H_0}{2}(x(n) \pm y1(n)) + x(n) \end{aligned}$$

Shelving Filters (Cont.)

Shelving Filter Parameters:

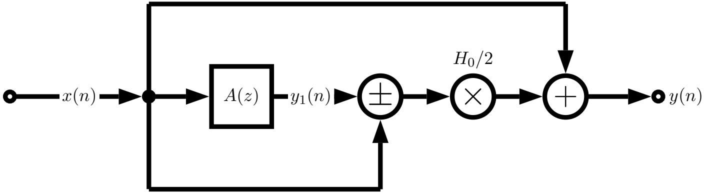
The **gain**, G, in dB can be adjusted accordingly:

$$H_0 = V_0 - 1 \quad \textbf{where} \quad V_0 = 10^{G/20}$$

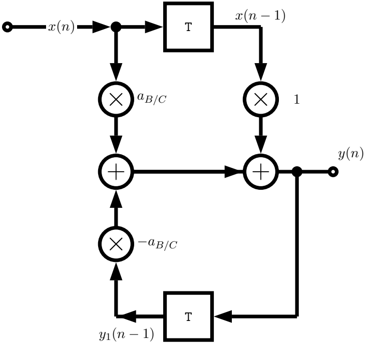
and the cut-off frequency for **boost**, a_B, or **cut**, a_C are given by:

$$\begin{aligned} a_B &= \frac{tan(2\pi f_c/f_s) - 1}{tan(2\pi f_c/f_s) + 1} \\ a_C &= \frac{tan(2\pi f_c/f_s) - V_0}{tan(2\pi f_c/f_s) + V_0} \end{aligned}$$

Shelving Filters Signal Flow Graph



where $A(z)$ is given by:



Peak Filters

A 2nd-order Peak Filter

Transfer function:

$$H(z) = 1 + \frac{H_0}{2}(1 - A_2(z))$$

where $A_2(z)$ is a **second-order allpass** filter:

$$A(z) = \frac{-a_B + (d - da_B)z^{-1} + z^{-2}}{1 + (d - da_B)z^{-1} + a_B z^{-2}}$$

which leads the following algorithm/difference equation:

$$\begin{aligned} y_1(n) &= 1a_{B/C}x(n) + d(1 - a_{B/C})x(n-1) + x(n-2) \\ &\quad -d(1 - a_{B/C})y_1(n-1) + a_{B/C}y_1(n-2) \\ y(n) &= \frac{H_0}{2}(x(n) - y_1(n)) + x(n) \end{aligned}$$

Peak Filters (Cont.)

Peak Filter Parameters:

The **center/cut-off frequency**, d, is given by:

$$d = -cos(2\pi f_c/f_s)$$

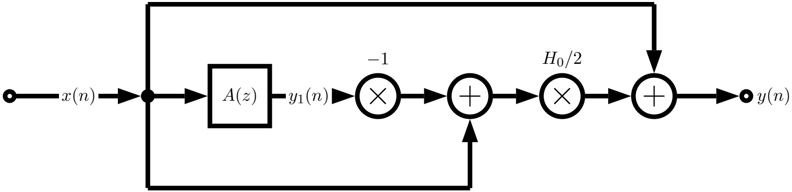
The H_0 by relation to the gain, G, as before:

$$H_0 = V_0 - 1 \quad \textbf{where} \quad V_0 = 10^{G/20}$$

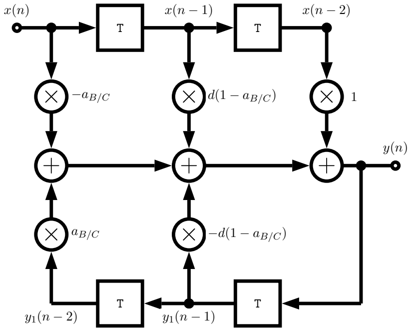
and the bandwidth, f_b is given by the limits for **boost**, a_B, or **cut**, a_C are given by:

$$\begin{aligned} a_B &= \frac{tan(2\pi f_b/f_s) - 1}{tan(2\pi f_b/f_s) + 1} \\ a_C &= \frac{tan(2\pi f_b/f_s) - V_0}{tan(2\pi f_b/f_s) + V_0} \end{aligned}$$

Peak Filters Signal Flow Graph



where $A(z)$ is given by:



Shelving Filter EQ MATLAB Example (1)

shelving.m

```
function [b, a]  = shelving(G, fc, fs, Q, type)
%
% Derive coefficients for a shelving filter with a given amplitude
% and cutoff frequency.  All coefficients are calculated as
% described in Zolzer's DAFX book (p. 50 -55).
%
% Usage:      [B,A] = shelving(G, Fc, Fs, Q, type);
%
%             G is the logrithmic gain (in dB)
%             FC is the center frequency
%             Fs is the sampling rate
%             Q adjusts the slope be replacing the sqrt(2) term
%             type is a character string defining filter type
%             Choices are: 'Base_Shelf' or 'Treble_Shelf'

% Error Check
if((strcmp(type,'Base_Shelf') ~= 1) && ...
         (strcmp(type,'Treble_Shelf') ~= 1))
    error(['Unsupported Filter Type: ' type]);
end
```

Shelving Filter EQ MATLAB Example (2)

shelving.m cont.

```
K = tan((pi * fc)/fs);
V0 = 10^(G/20);
root2 = 1/Q;

% Invert gain if a cut
if (V0 < 1)
    V0 = 1/V0;
end

%%%%%%%%%%%%%%%%%%%%
%    BASE BOOST
%%%%%%%%%%%%%%%%%%%%
if(( G > 0 ) & (strcmp(type,'Base_Shelf')))

    b0 = (1 + sqrt(V0)*root2*K + V0*K^2) / (1 + root2*K + K^2);
    b1 = (2 * (V0*K^2 - 1) ) / (1 + root2*K + K^2);
    b2 = (1 - sqrt(V0)*root2*K + V0*K^2) / (1 + root2*K + K^2);
    a1 =  (2 * (K^2 - 1) ) / (1 + root2*K + K^2);
    a2 =  (1 - root2*K + K^2) / (1 + root2*K + K^2);
```

Shelving Filter EQ MATLAB Example (3)

shelving.m cont.

```
%%%%%%%%%%%%%%%%%%%%
%     BASE CUT
%%%%%%%%%%%%%%%%%%%%
elseif (( G < 0 ) & (strcmp(type,'Base_Shelf')))

    b0 = (1 + root2*K + K^2) / (1 + root2*sqrt(V0)*K + V0*K^2);
    b1 =  (2 * (K^2 - 1) ) / (1 + root2*sqrt(V0)*K + V0*K^2);
    b2 =  (1 - root2*K + K^2) / (1 + root2*sqrt(V0)*K + V0*K^2);
    a1 =  (2 * (V0*K^2 - 1) ) / (1 + root2*sqrt(V0)*K + V0*K^2);
    a2 = (1 - root2*sqrt(V0)*K + V0*K^2) / ...
             (1 + root2*sqrt(V0)*K + V0*K^2);
```

Shelving Filter EQ MATLAB Example (3)

shelving.m cont.

```
%%%%%%%%%%%%%%%%%%%%
%    TREBLE BOOST
%%%%%%%%%%%%%%%%%%%%
elseif (( G > 0 ) & (strcmp(type,'Treble_Shelf')))

    b0 = (V0 + root2*sqrt(V0)*K + K^2) / (1 + root2*K + K^2);
    b1 =  (2 * (K^2 - V0) ) / (1 + root2*K + K^2);
    b2 = (V0 - root2*sqrt(V0)*K + K^2) / (1 + root2*K + K^2);
    a1 =  (2 * (K^2 - 1) ) / (1 + root2*K + K^2);
    a2 =  (1 - root2*K + K^2) / (1 + root2*K + K^2);
```

Shelving Filter EQ MATLAB Example (4)

shelving.m cont.

```
%%%%%%%%%%%%%%%%%%%%
%   TREBLE CUT
%%%%%%%%%%%%%%%%%%%%

elseif (( G < 0 ) & (strcmp(type,'Treble_Shelf')))

    b0 =   (1 + root2*K + K^2) / (V0 + root2*sqrt(V0)*K + K^2);
    b1 =    (2 * (K^2 - 1) ) / (V0 + root2*sqrt(V0)*K + K^2);
    b2 =    (1 - root2*K + K^2) / (V0 + root2*sqrt(V0)*K + K^2);
    a1 =    (2 * ((K^2)/V0 - 1) ) / (1 + root2/sqrt(V0)*K ...
                + (K^2)/V0);
    a2 = (1 - root2/sqrt(V0)*K + (K^2)/V0) / ....
             (1 + root2/sqrt(V0)*K + (K^2)/V0);

%%%%%%%%%%%%%%%%%%%%
%   All-Pass
%%%%%%%%%%%%%%%%%%%%
else
    b0 = V0;
    b1 = 0; b2 = 0; a1 = 0; a2 = 0;
end

%return values
a = [  1, a1, a2];
b = [ b0, b1, b2];
```

Shelving Filter EQ MATLAB Example (5)

Example use: shelving_eg.m

```
infile = 'acoustic.wav';

[ x, Fs] = audioread(infile);% read in wav sample

% Set parameters for Shelving Filter
% Change these to experiment with filter
G = 4; fcb = 300; Q = 3; type = 'Base_Shelf';

[b a] = shelving(G, fcb, Fs, Q, type);
yb = filter(b,a, x);

% Write output wav files
audiowrite('out_bassshelf.wav', yb, Fs);

% Plot the original and equalised waveforms
figure(1), hold on;
plot(yb,'b');
plot(x,'r');
title('Bass Shelf Filter Equalised Signal');
```

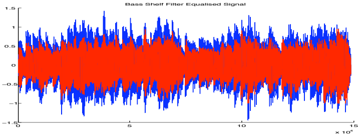
Shelving Filter EQ MATLAB Example (6)

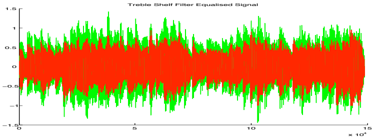
shelving_eg.m cont.

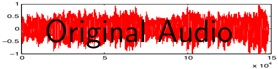
```
% Do treble shelf filter
fct = 600; type = 'Treble_Shelf';

[b a] = shelving(G, fct, Fs, Q, type);
yt = filter(b,a, x);

% Write output wav files
audiowrite('out_treblehelf.wav', yt, Fs);

figure(1), hold on;
plot(yb,'g');
plot(x,'r');
title('Treble Shelf Filter Equalised Signal');
```

Shelving Filter EQ MATLAB Example Output

The output from the above code is (red plot is original audio):







Click on above images or here to hear: original audio, bass shelf filtered audio, treble shelf filtered audio.