ITU-R BS.1770-1 filter specifications

(unofficial)

January 15, 2011

1 Pre-filter

The pre-filter is a second-order IIR high-shelf filter with the transfer function

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

and the filter coefficients in Table 1.

Assuming those filter coefficients were obtained with the design in Table 2, solving for V_h, V_b, V_l, Q, Ω gives

$$\begin{aligned} V_h &= 1.584864701130855 \ (\approx 4\text{dB}), \\ V_b &= 1.258720930232562 \ (\approx \sqrt{V_h}), \\ V_l &= 1.0, \\ Q &= 0.7071752369554196, \\ \Omega &= 0.1105318322704944 \end{aligned}$$

and with $\Omega = \tan\left(\pi \frac{f_c}{f_s}\right)$ and $f_s = 48000$

$$f_c = 1681.974450955533.$$

Table 1: Filter coefficients for the pre-filter ([1]):

i	a_i	b_i
0	1.0	1.53512485958697
1	−1.69065929318241	−2.69169618940638
2	0.73248077421585	1.19839281085285

Table 2: normalized ($a_0 = 1$) biquad filter ([2])

b_0	b_1	b_2	a_1	a_2
$\frac{V_l\Omega^2+V_b\frac{\Omega}{Q}+V_h}{\Omega^2+\frac{\Omega}{Q}+1}$	$\frac{2(V_l\Omega^2-V_h)}{\Omega^2+\frac{\Omega}{Q}+1}$	$\frac{V_l\Omega^2-V_b\frac{\Omega}{Q}+V_h}{\Omega^2+\frac{\Omega}{Q}+1}$	$\frac{2(\Omega^2-1)}{\Omega^2+\frac{\Omega}{Q}+1}$	$\frac{\Omega^2-\frac{\Omega}{Q}+1}{\Omega^2+\frac{\Omega}{Q}+1}$

V_h: high-pass gain factor
V_B: band-pass gain factor
V_l: low-pass gain factor
Q: Q factor
Ω: $\tan\left(\pi\frac{f_c}{f_s}\right)$

2 RLB filter

The RLB filter is a second-order IIR high-pass filter with the normalized transfer function

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

and the filter coefficients in Table 3.

Assuming those filter coefficients were obtained with the design in Table 4, solving for V_h, V_b, V_l, Q, Ω gives

$$\begin{aligned} V_h &= G \text{ (optional global gain factor; should be set to 1)}, \\ V_b &= 0.0, \\ V_l &= 0.0, \\ Q &= 0.5003270373238773, \\ \Omega &= 0.002495965915335466 \end{aligned}$$

and with $\Omega = \tan\left(\pi\frac{f_c}{f_s}\right)$ and $f_s = 48000$

$$f_c = 38.13547087602444.$$

Table 3: Filter coefficients for the RLB filter ([1]):

i	a_i	b_i
0	1.0	1.0
1	−1.99004745483398	−2.0
2	0.99007225036621	1.0

Table 4: normalized ($a_0 = b_0 = 1$) biquad filter (modified from [2] to normalize b_0)

b_1	b_2	a_1	a_2
$\frac{2(V_l\Omega^2 - V_h)}{V_l\Omega^2 + V_b\frac{\Omega}{Q} + V_h}$	$\frac{V_l\Omega^2 - V_b\frac{\Omega}{Q} + V_h}{V_l\Omega^2 + V_b\frac{\Omega}{Q} + V_h}$	$\frac{2(\Omega^2 - 1)}{\Omega^2 + \frac{\Omega}{Q} + 1}$	$\frac{\Omega^2 - \frac{\Omega}{Q} + 1}{\Omega^2 + \frac{\Omega}{Q} + 1}$

V_h: high-pass gain factor
V_b: band-pass gain factor
V_l: low-pass gain factor
Q: Q factor
Ω: $\tan\left(\pi\frac{f_c}{f_s}\right)$

References

[1] Rec. ITU-R BS.1770-1, *Algorithms to measure audio programme loudness and true-peak audio level*

[2] Brian Neunaber, *Parameter Quantization in Direct-Form Recursive Audio Filters*

A Maxima code

A.1 Pre-filter

```
fpprec: 300;

a0 : 1 + K / Q + K^2;
a1 : 2 * (K^2 - 1);
a2 : 1 - K / Q + K^2;
b0 : Vh + Vb * K / Q + Vl * K^2;
b1 : 2 * (Vl * K^2 - Vh);
b2 : Vh - Vb * K / Q + Vl * K^2;

sol : solve([b0 / a0 =  153512485958697 / 100000000000000,
             b1 / a0 = -269169618940638 / 100000000000000,
             b2 / a0 =  119839281085285 / 100000000000000,
             a1 / a0 = -169065929318241 / 100000000000000,
             a2 / a0 =   73248077421585 / 100000000000000],
            [Vh, Vb, Vl, Q, K]);

Vh : rhs(sol[2][1]);
Vb : rhs(sol[2][2]);
Vl : rhs(sol[2][3]);
Q :  rhs(sol[2][4]);
K :  rhs(sol[2][5]);

sol2 : solve(K = tan(%pi * fc / 48000), fc);
sol3 : solve(Vh = 10^(G / 20), G);
sol4 : solve(x*log(Vh) = log(Vb), x);

fc : rhs(sol2[1]);
G : rhs(sol3[1]);
x : rhs(sol4[1]);

/* Output floating point numbers
   for use in C code */
float(bfloat(Vh));
float(bfloat(Vb));
float(bfloat(Vl));
float(bfloat(Q));
float(bfloat(K));
float(bfloat(fc));
```

A.2 RLB filter

```
fpprec: 300;

a0 : 1 + K / Q + K^2;
a1 : 2 * (K^2 - 1);
a2 : 1 - K / Q + K^2;
b0 : Vh + Vb * K / Q + Vl * K^2;
b1 : 2 * (Vl * K^2 - Vh);
b2 : Vh - Vb * K / Q + Vl * K^2;

sol : solve([b1 / b0 = -2,
             b2 / b0 =  1,
             a1 / a0 = -199004745483398/100000000000000,
             a2 / a0 =   99007225036621/100000000000000],
            [Vh, Vb, Vl, Q, K]);

Vh : rhs(sol[3][1]);
Vb : rhs(sol[3][2]);
Vl : rhs(sol[3][3]);
Q :  rhs(sol[3][4]);
K :  rhs(sol[3][5]);

sol2 : solve(K = tan(%pi * fc / 48000), fc);

fc : rhs(sol2[1]);

/* Output floating point numbers
   for use in C code */
float(bfloat(Vh));
float(bfloat(Vb));
float(bfloat(Vl));
float(bfloat(Q));
float(bfloat(K));
float(bfloat(fc));
```