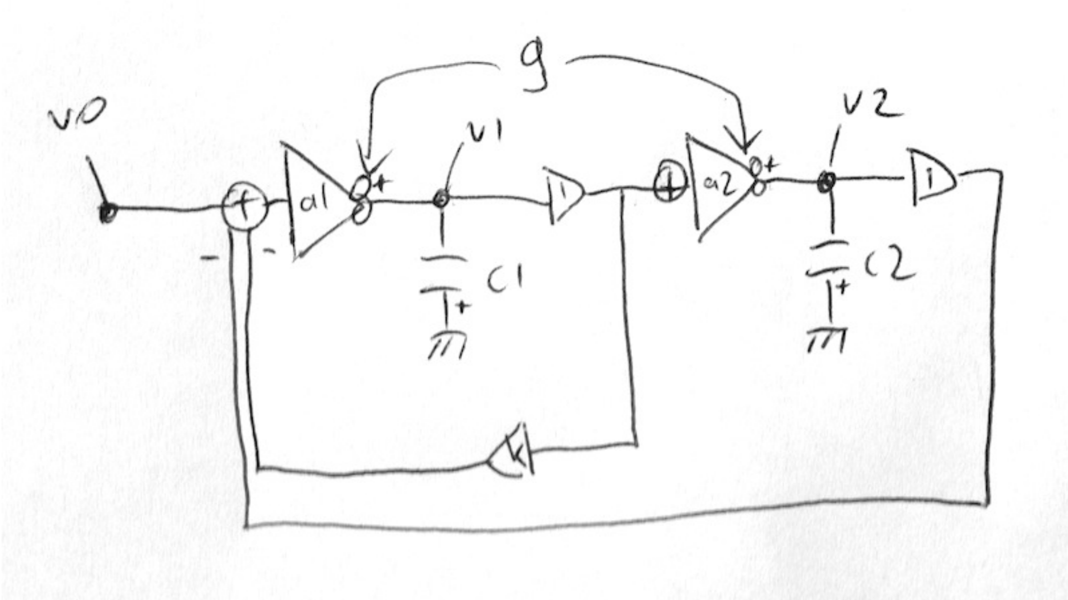
Solving the continuous SVF equations using trapezoidal integration and equivalent currents

© Andrew Simper, Cytomic, 2013, andy@cytomic.com
last updated: 23rd November 2013
corrected typo: 3rd August 2016
added allpass: 20th November 2016



References

[1] http : // en.wikipedia.org / wiki / Capacitor
[2] http : // en.wikipedia.org / wiki / Trapezoidal_rule
[3] http : // en.wikipedia.org / wiki / Nodal_analysis
[4] http : // qucs.sourceforge.net / tech / node26.html
[5] http : // www.ecircuitcenter.com / SPICEtopics.htm
[6] http : // www.musicdsp.org / files / Audio - EQ - Cookbook.txt

Trapezoidal integration of capacitor [1, 2, 4]

$$\mathrm{vc}(t) = \frac{Q(t)}{C} = \frac{1}{C} \int \mathrm{ic}(t)$$

$$\mathrm{vc}^{n+1} = \mathrm{vc}^{n} + \frac{h}{C} \left(1/2\, \mathrm{ic}^{n+1} + 1/2\, \mathrm{ic}^{n}\right)$$

$$\frac{2\,C}{h} \mathrm{vc}^{n+1} - \frac{2\,C}{h} \mathrm{vc}^{n} = \mathrm{ic}^{n+1} + \mathrm{ic}^{n}$$

$$\mathrm{ic}^{n+1} = \frac{2\,C}{h} \mathrm{vc}^{n+1} - \frac{2\,C}{h} \mathrm{vc}^{n} - \mathrm{ic}^{n}$$

We can write the expression with these grouped terms of a scalar and and offset:

$$\mathbf{ic}^{n+1} = \mathbf{gc\, vc}^{n+1} - \mathbf{iceq}^{n} \tag{1}$$

where:

$$\mathrm{gc} = \frac{2\,C}{h}$$

$$\mathrm{iceq}^{n} = \mathrm{gc\, vc}^{n} + \mathrm{ic}^{n}$$

To update $\mathrm{iceq}^{n+1}$for the next time step after you have solved for $\mathrm{vc}^{n+1}$and so ic^{n+1}we have:

$$\mathrm{iceq}^{n+1} = \mathrm{gc\, vc}^{n+1} + \mathrm{ic}^{n+1}$$

$$\mathrm{iceq}^{n+1} = \mathrm{gc\, vc}^{n+1} + \mathrm{gc\, vc}^{n+1} - \mathrm{iceq}^{n}$$

which can be simplified to :

$$\mathbf{iceq}^{n+1} = \mathbf{2\, gc\, vc}^{n+1} - \mathbf{iceq}^{n} \tag{2}$$

Low, Band, High, Notch, and Peak

Solving for terms using nodal analysis (also called KCL, ie the sum of currents at each node is zero) [3,4,5]

Now using equation (1) for the capacitor currents we can write the nodal equations as follows:

$$\mathbf{0 = \mathit{g}\,(v0 - \mathit{k}\,v1 - v2) - (gc\,(v1 - 0) - ic1eq)}, \text{ at node 1} \quad (3)$$

$$\mathbf{0 = \mathit{g}\,(v1 - 0) - (gc\,(v2 - 0) - ic2eq)}, \text{ at node 2} \quad (4)$$

In a circuit we have both capacitance and resistance (conductance) terms that set the cutoff frequency, but for a digial model without loss of generality we can set the capacitors equivalent conductance term gc to 1 and then adjust the g term accordingly to set the cutoff.

```
Clear["Global`*"];
nodev1 = 0 == g (v0 - k v1 - v2) - (gc (v1 - 0) - ic1eq)
nodev2 = 0 == g (v1 - 0) - (gc (v2 - 0) - ic2eq)
subst = {gc → 1};

sln1 = Solve[{nodev1, nodev2} /. subst, {v1}, {v2}][[1]] // FullSimplify
sln2 = (Solve[{nodev1, nodev2} /. subst, {v2}, {v0}][[1]] // FullSimplify)
sln3 = sln2 /. sln1
```

$$0 == \text{ic1eq} - \text{gc v1} + \text{g (v0} - \text{k v1} - \text{v2)}$$

$$0 == \text{ic2eq} + \text{g v1} - \text{gc v2}$$

$$\left\{\text{v1} \to \frac{\text{ic1eq} + \text{g}\,(-\text{ic2eq} + \text{v0})}{1 + \text{g}\,(\text{g} + \text{k})}\right\}$$

$$\{\text{v2} \to \text{ic2eq} + \text{g v1}\}$$

$$\left\{\text{v2} \to \text{ic2eq} + \frac{\text{g}\,(\text{ic1eq} + \text{g}\,(-\text{ic2eq} + \text{v0}))}{1 + \text{g}\,(\text{g} + \text{k})}\right\}$$

Solving for terms using nodal analysis manually by grouping constants and scalars

Group constants and scalars on and solving the sets of equations is the same thing as doing manual row reduction of the matrix equations. This will result in multiple division, one per cancellation of a term, which can be more efficient for larger systems of equations. For the SVF we only have two equations so we can save a division by solving for the terms directly.

Since everything is linear we can express the n^{th} voltage v_n at node n in terms of a linear combination of the other voltages plus and offset:

$$v_n = c_n + \sum_{i=0}^{m-1} g_{n,i}\, v_i\,, \text{ for } i \neq n \quad (5)$$

The nodal equations from (3) and (4) are:

$$0 = g\,(\text{v0} - k\,\text{v1} - \text{v2}) - (\text{gc}\,(\text{v1} - 0) - \text{ic1eq})$$
$$0 = g\,(\text{v1} - 0) - (\text{gc}(\text{v2} - 0) - \text{ic2eq})$$

So looking at the first equation for node 1 we can see there are terms in v0, v1, and v2, similarly at node 2 we have terms in v1 and v2, and since these are already in linear form we can group the coefficients as follows:

$$v1 = c_1 + g_{1,0}\,v0 + g_{1,2}\,v2$$
$$v2 = c_2 + g_{2,1}\,v1$$

We can now substitute v2 into the equation for v1 to solve for v1 in terms of v0 alone:

$$v1 = c_1 + g_{1,0}\,v0 + g_{1,2}\,(c_2 + g_{2,1}\,v1)$$
$$v1 = (c_2 + g_{1,0}\,v0 + g_{1,2}\,c_2) \big/ (1 - g_{1,2}\,g_{2,1})$$

If we want we could further reduce the expression for v1 by grouping terms again, for larger systems of equations where further cancellation of terms is required this can be beneficial:

$$v1 = c_1' + g_{1,0}'\,v0$$
$$v2 = c_2 + g_{2,1}\,v1$$

If you want to know what the constants are we have to look back at the original nodal equations, solving for v1 at node 1 and setting gc=1:

$$0 = g\,(v0 - k\,v1 - v2) - ((v1 - 0) - \mathrm{ic1eq})$$
$$v1\,(1 + g\,k) = g\,v0 - g\,v2 + \mathrm{ic1eq}$$
$$v1 = (\mathrm{ic1eq} + g\,v0 - g\,v2) / (1 + g\,k)$$
$$v1 = \mathrm{c1} + g_{1,0}\,v0 + g_{1,2}\,v2$$

So the first set of constants are:

$$c_1 = \mathrm{ic1eq} / (1 + g\,k)$$
$$g_{1,0} = g / (1 + g\,k)$$
$$g_{1,2} = -g / (1 + g\,k)$$

Doing the same thing and solving for v2 at node v2 we have:

$$0 = g\,(v1 - 0) - ((v2 - 0) - \mathrm{ic2eq})$$
$$0 = g\,v1 - v2 + \mathrm{ic2eq}$$
$$v2 = \mathrm{ic2eq} + g\,v1$$
$$v2 = c_2 + g_{2,1}\,v1$$

So the second set of constants are:

$$c_2 = \mathrm{ic2eq}$$
$$g_{2,1} = g$$

We can now check the results against the direcetly solved solution:

```
ClearAll["Global`*"];
subst = {c2 → ic2eq, g2,1 → g, c1 → ic1eq/(1 + g k), g1,0 → g/(1 + g k), g1,2 → -g/(1 + g k)};
nodev1 = {v1 → FullSimplify[(c1 + g1,0 v0 + g1,2 c2)/(1 - g1,2 g2,1) /. subst]}
nodev2 = {v2 → FullSimplify[c2 + g2,1 v1 /. subst]}
nodev2 /. nodev1
```

$$\left\{v1 \to \frac{\mathrm{ic1eq} + g\,(-\mathrm{ic2eq} + v0)}{1 + g\,(g + k)}\right\}$$

$$\{v2 \to \mathrm{ic2eq} + g\,v1\}$$

$$\left\{v2 \to \mathrm{ic2eq} + \frac{g\,(\mathrm{ic1eq} + g\,(-\mathrm{ic2eq} + v0))}{1 + g\,(g + k)}\right\}$$

In summary every time we want to cancel out a dependancy from the set of linear equations we will require a division. Sometimes these divisions can be grouped together and solved directly, but in general, if there are many equations being solved for the size of the directly solved expressions becomes exponentially larger, so using multiple divisions is more efficient.

Regrouping terms in bounded form

```
FullSimplify[
```
$$\frac{\text{ic1eq} + g\left(-\text{ic2eq} + \text{v0}\right)}{1 + g\,(g + k)} - \left(\frac{1}{1 + g\,(g + k)}\,\text{ic1eq} + \frac{g}{1 + g\,(g + k)}\left(-\text{ic2eq} + \text{v0}\right)\right)]$$

$$\text{FullSimplify}\left[\text{ic2eq} + \frac{g\left(\text{ic1eq} + g\left(-\text{ic2eq} + \text{v0}\right)\right)}{1 + g\,(g + k)} - \left(\text{ic2eq} + \frac{g}{1 + g\,(g + k)}\,\text{ic1eq} + \frac{g^2}{1 + g\,(g + k)}\left(-\text{ic2eq} + \text{v0}\right)\right)\right]$$

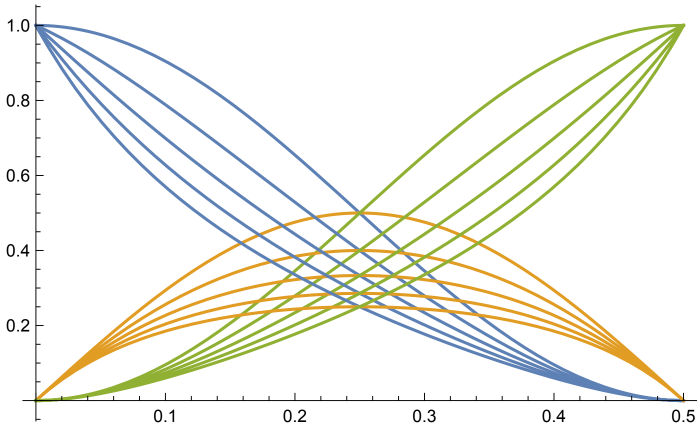
0

0

Plotting regrouped terms

The symmetry of the equations is quite aparent in the plots of these regrouped terms

```
a1 = 1 / (1 + g (g + k)) /. {g → Tan[π wc]};
a2 = g a1 /. {g → Tan[π wc]};
a3 = g a2 /. {g → Tan[π wc]};
Show[Table[Plot[{a1, a2, a3}, {wc, 0, 1 / 2}], {k, 0, 2, 1 / 2}]]
```



Algorithm for low, band, high, notch, and peak with bounded terms

This implementation, apart from keeping scaling terms bounded, computes v1 and v2 directly from v0, so these operations can be done in parallel. The updating of sates as well as the final output calculations can also be done in parallel.

```
Clear
ic1eq = 0;
ic2eq = 0;

Set
g = Tan[π cutoff/samplerate];
k = 1/Q = 2 - 2*res;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;

Tick
(6*, 6+, 12 total ops for low and band)
(7*, 8+, 15 total ops for high)
(7*, 7+, 14 total ops for notch)
(8*, 8+, 16 total ops for peak)
(8*, 7+, 15 total ops for all)

v3 = v0 - ic2eq;
v1 = a1*ic1eq + a2*v3;
v2 = ic2eq + a2*ic1eq + a3*v3;
ic1eq = 2*v1 - ic1eq;
ic2eq = 2*v2 - ic2eq;

low = v2;
band = v1;
high = v0 - k*v1 - v2;
notch = low + high = v0 - k*v1;
peak = low - high = v0 - k*v1 - 2*v2;
all = low + high - k*band = v0 - 2*k*v1;
```

Algorithm using v1 to compute v2 (unbouded g term)

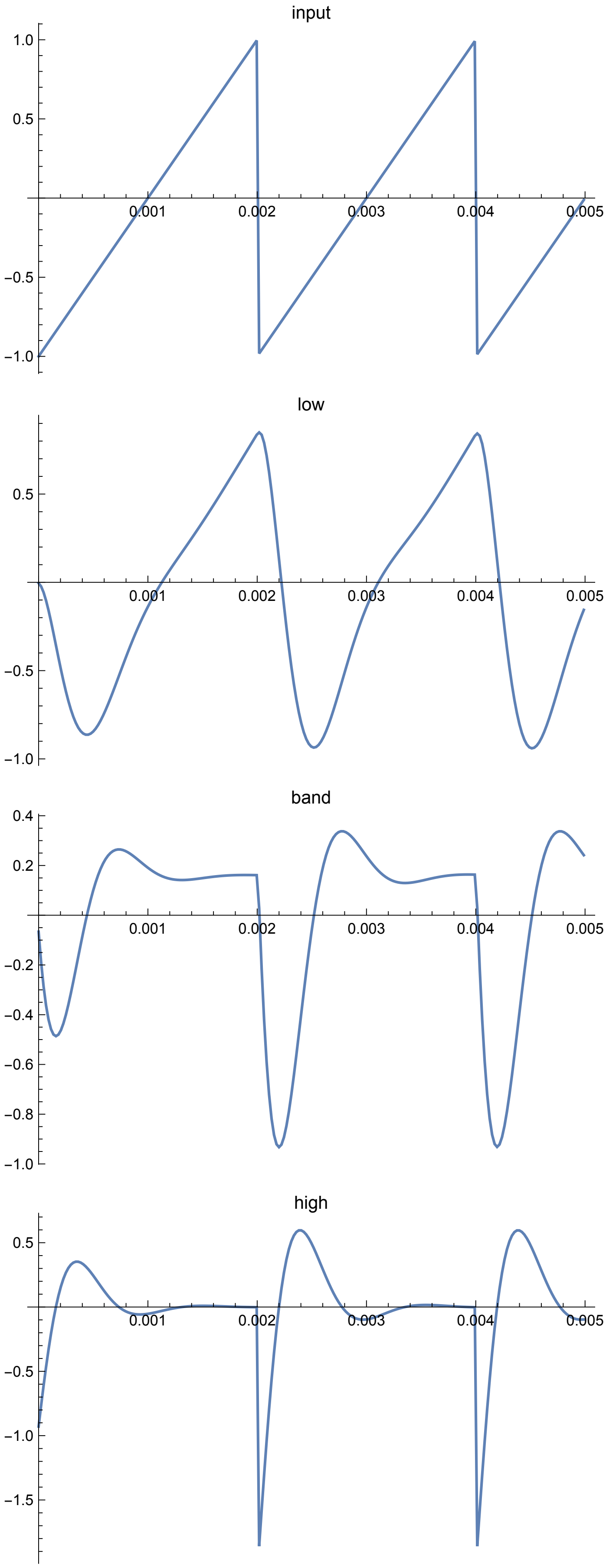
Note that this form requires computation of v1 before v2 can be solved, which places an extra dependancy in the system. Even though there there are two fewer numerical operation to compute this may be slower than the bounded method where operations are kept more parallel. To determine which method is more optimal profiling is required.

```
Clear
ic1eq = 0;
ic2eq = 0;

Set
g = Tan[π cutoff/samplerate];
k = 1/Q = 2 - 2*res;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;

Tick
(5*, 5+, 10 total ops for low and band)
(6*, 7+, 13 total ops for high)
(6*, 6+, 12 total ops for notch)
(7*, 7+, 14 total ops for peak)
(8*, 7+, 15 total ops for all)

v1 = a1*ic1eq + a2*(v0 - ic2eq);
v2 = ic2eq + g*v1;
ic1eq = 2*v1 - ic1eq;
ic2eq = 2*v2 - ic2eq;
```

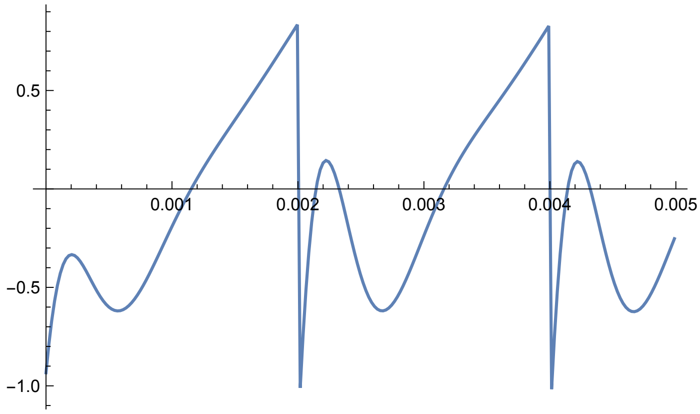
```
low = v2;
band = v1;
high = v0 - k*v1 - v2;
notch = low + high = v0 - k*v1;
peak = low - high = 2*v2 - v0 + k*v1
all = low + high - k*band = v0 - 2*k*v1;
```

Test of algorithm

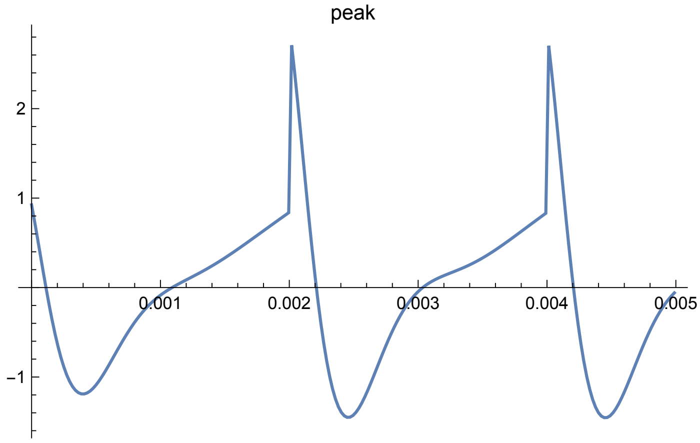
```
ClearState[] := Block[{},
   ic1eq = 0;
   ic2eq = 0;
  ];

SetCoeff[cutoff_, res_, samplerate_] := Block[{},
   g = Tan[π cutoff / samplerate];
   k = 2 - 2 * res;
   a1 = 1 / (1 + g * (g + k));
   a2 = g * a1;
   a3 = g * a2;
  ];

Tick[t_, v0_] := Block[{v1, v2, v3, low, band, high, notch, peak, all},
   v3 = v0 - ic2eq;
   v1 = a1 * ic1eq + a2 * v3;
   v2 = ic2eq + a2 * ic1eq + a3 * v3;
   ic1eq = 2 * v1 - ic1eq;
   ic2eq = 2 * v2 - ic2eq;
   low = v2;
   band = v1;
   high = v0 - k * v1 - v2;
   notch = v0 - k * v1;
   peak = 2 * v2 - v0 + k * v1;
   all = 2 * low + band + high;
   all = v0 - 2 * k * v1;
   Return[{t, v0, low, band, high, notch, peak, all}]
  ];

h = 1.0 / 44 100.0;
t1 = 0.005;
drive = 1;
freq = 500.0;

MySaw[x_] := 2 (x - Floor[x] - 0.5);
MyOsc[x_] := drive MySaw[freq x];

ClearState[];
SetCoeff[1000.0, 0.5, 44 100.0];
tp0 = Table[Tick[t, MyOsc[t]], {t, 0, t1, h}];

ListPlot[{tp0[[All, {1, 2}]]},
 PlotLabel → "input", Joined → True, PlotRange → All]
ListPlot[{tp0[[All, {1, 3}]]}, PlotLabel → "low", Joined → True, PlotRange → All]
ListPlot[{tp0[[All, {1, 4}]]},
 PlotLabel → "band", Joined → True, PlotRange → All]
ListPlot[{tp0[[All, {1, 5}]]}, PlotLabel → "high",
 Joined → True, PlotRange → All]
ListPlot[{tp0[[All, {1, 6}]]}, PlotLabel → "notch",
 Joined → True, PlotRange → All]
ListPlot[{tp0[[All, {1, 7}]]}, PlotLabel → "peak",
 Joined → True, PlotRange → All]
ListPlot[{tp0[[All, {1, 8}]]}, PlotLabel → "all", Joined → True, PlotRange → All]
```


input
1.0
0.5
−0.5
−1.0
0.001
0.002
0.003
0.004
0.005
low
0.5
−0.5
−1.0
band
0.4
0.2
−0.2
−0.4
−0.6
−0.8
−1.0
high
0.5
−0.5
−1.0
−1.5

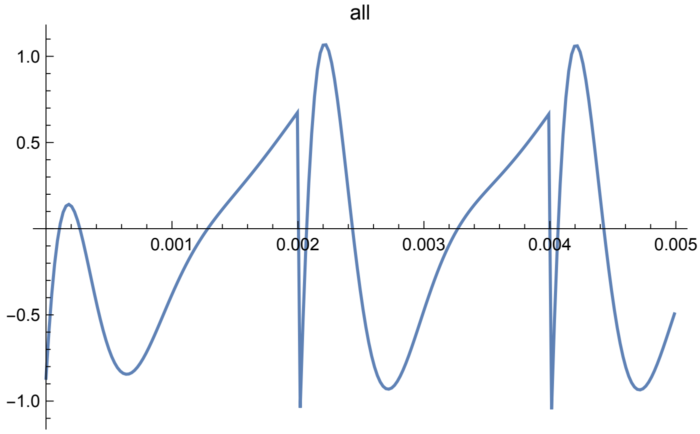
notch



peak



all



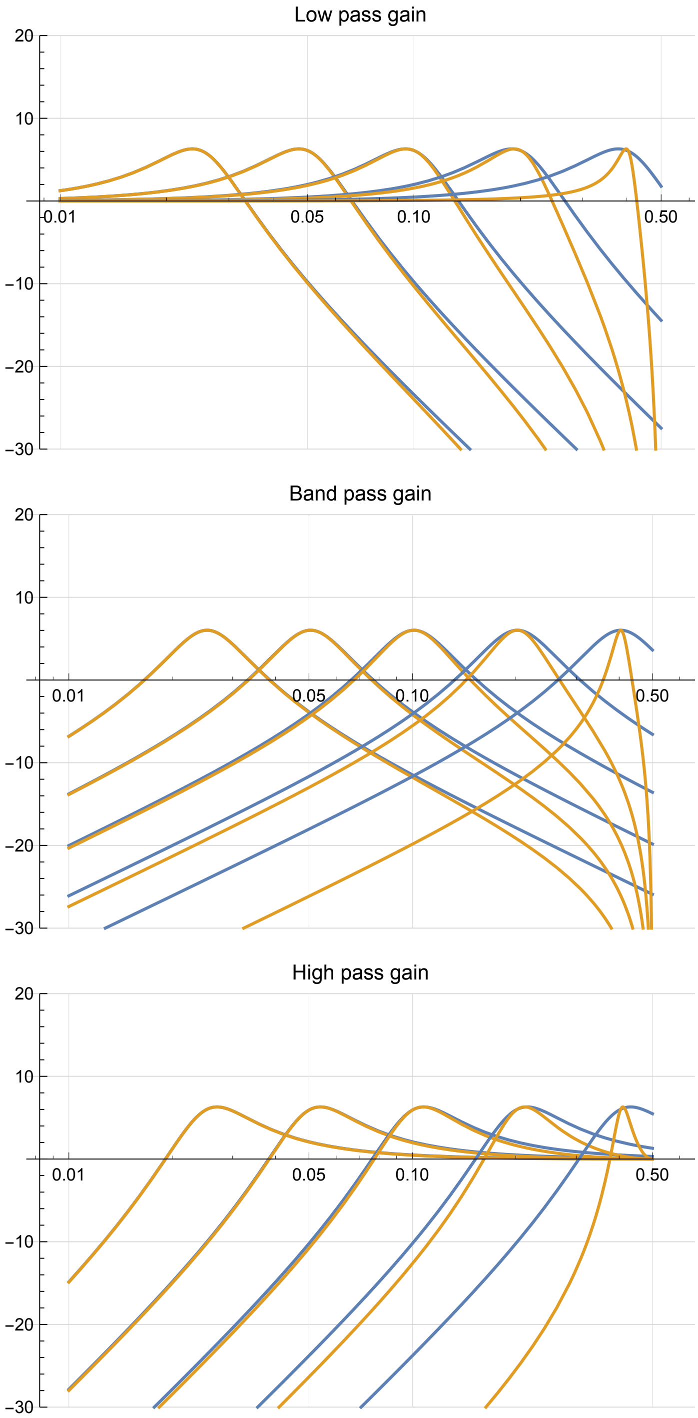
Transfer functions for low, band, high, notch, peak, allpass, continuous and discrete

```
ClearAll["Global`*"];
dB[x_] := 6 Log[2, Abs[x]];
BodePlotSZ[responseS_, responseZ_, title_] :=
  Show[
   Table[LogLinearPlot[{dB[(responseS) /. {g → 2 π 2^wc, k → damp, s → 2 π ⅈ w, A →
          Power[10, gaindb / 40]}], dB[(responseZ) /. {g → Tan[π 2^wc],
         k → damp, z → Exp[2 π ⅈ w], A → Power[10, gaindb / 40]}]},
     {w, 0.01, 0.5}, PlotLabel → title, PlotRange → {-30, 20},
     GridLines → Automatic, GridLinesStyle → LightGray], Evaluate[wcrange]]];
BodePlotSZPhase[responseS_, responseZ_, title_] :=
  Show[
   Table[LogLinearPlot[{Arg[(responseS) /. {g → 2 π 2^wc, k → damp, s → 2 π ⅈ w,
         A → Power[10, gaindb / 40]}], Arg[(responseZ) /. {g → Tan[π 2^wc],
         k → damp, z → Exp[2 π ⅈ w], A → Power[10, gaindb / 40]}]},
     {w, 0.01, 0.5}, PlotLabel → title, PlotRange → {-π, π},
     GridLines → Automatic, GridLinesStyle → LightGray], Evaluate[wcrange]]];
```

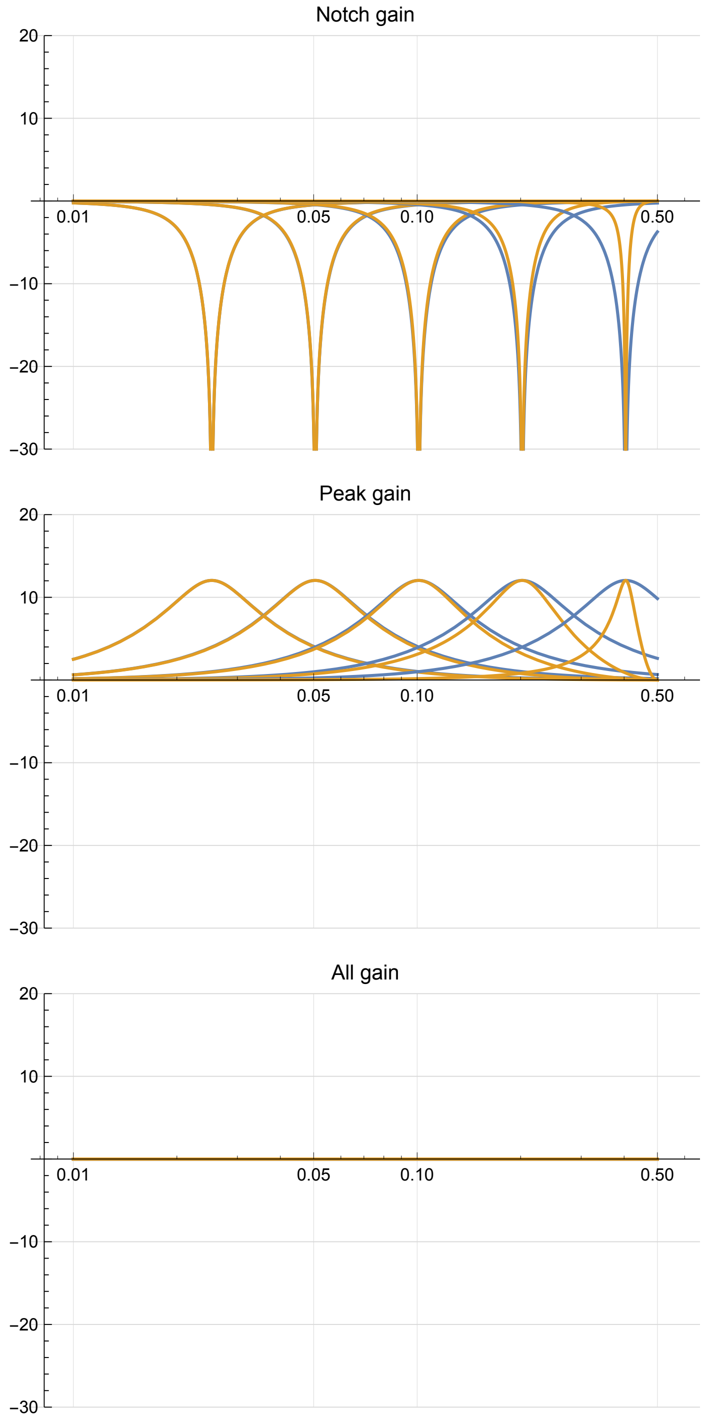
```
nodev1 = g (v0 - k v1 - v2) - (s c (v1 - 0)) == 0;
nodev2 = g (v1 - 0) - (s c (v2 - 0)) == 0;
lps = lp == (v2) / v0;
bps = bp == (v1) / v0;
hps = hp == (v0 - k v1 - v2) / v0;
nps = np == (lp + hp);
pps = pp == (lp - hp);
aps = ap == (lp + hp - k bp);
subst = {c → 1};
slns = Solve[{nodev1, nodev2, lps, bps, hps, nps, pps, aps} /. subst,
    {lp, bp, hp, np, pp, ap}, {v0, v1, v2}][[1]] // FullSimplify

nodev1 = g (v0 - k v1 - v2) - (gc (v1 - 0) - ic1eq) == 0;
nodev2 = g (v1 - 0) - (gc (v2 - 0) - ic2eq) == 0;
ic1eqn = ic1eq == 2 gc (v1 - 0) z^-1 - ic1eq z^-1;
ic2eqn = ic2eq == 2 gc (v2 - 0) z^-1 - ic2eq z^-1;
lpz = lp == (v2) / v0;
bpz = bp == (v1) / v0;
hpz = hp == (v0 - k v1 - v2) / v0;
npz = np == (lp + hp);
ppz = pp == (lp - hp);
apz = ap == (lp + hp - k bp);
subst = {gc → 1};
slnz =
 Solve[{nodev1, nodev2, ic1eqn, ic2eqn, lpz, bpz, hpz, npz, ppz, apz} /. subst,
    {lp, bp, hp, np, pp, ap}, {v0, v1, v2, ic1eq, ic2eq}][[1]] // FullSimplify

dB[x_] := 20 Log[10, Abs[x]];
wcrange = {wc, -6 + Log[2], -1, 1};
damp = 0.5;

BodePlotSZ[lp /. slns, lp /. slnz, "Low pass gain"]
BodePlotSZ[bp /. slns, bp /. slnz, "Band pass gain"]
BodePlotSZ[hp /. slns, hp /. slnz, "High pass gain"]
BodePlotSZ[np /. slns, np /. slnz, "Notch gain"]
BodePlotSZ[pp /. slns, pp /. slnz, "Peak gain"]
BodePlotSZ[ap /. slns, ap /. slnz, "All gain"]
BodePlotSZPhase[ap /. slns, ap /. slnz, "All phase"]
```

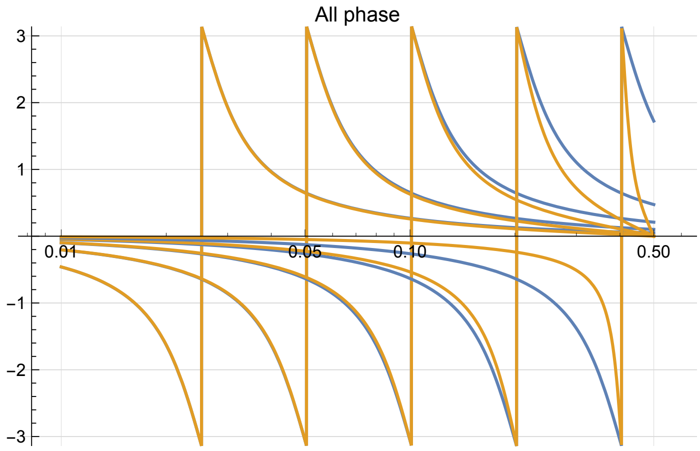
$$\left\{\mathrm{lp} \to \frac{g^2}{g^2 + g\,k\,s + s^2},\ \mathrm{bp} \to \frac{g\,s}{g^2 + g\,k\,s + s^2},\ \mathrm{hp} \to \frac{s^2}{g^2 + g\,k\,s + s^2},\right.$$
$$\left.\mathrm{np} \to \frac{g^2 + s^2}{g^2 + g\,k\,s + s^2},\ \mathrm{pp} \to \frac{(g - s)\ (g + s)}{g^2 + g\,k\,s + s^2},\ \mathrm{ap} \to 1 - \frac{2\,g\,k\,s}{g^2 + g\,k\,s + s^2}\right\}$$

$$\left\{\mathrm{lp} \to \frac{g^2\ (1 + z)^2}{(-1 + z)^2 + g^2\ (1 + z)^2 + g\,k\ \left(-1 + z^2\right)},\ \mathrm{bp} \to \frac{g\ \left(-1 + z^2\right)}{(-1 + z)^2 + g^2\ (1 + z)^2 + g\,k\ \left(-1 + z^2\right)},\right.$$
$$\mathrm{hp} \to \frac{(-1 + z)^2}{(-1 + z)^2 + g^2\ (1 + z)^2 + g\,k\ \left(-1 + z^2\right)},\ \mathrm{np} \to \frac{(-1 + z)^2 + g^2\ (1 + z)^2}{(-1 + z)^2 + g^2\ (1 + z)^2 + g\,k\ \left(-1 + z^2\right)},$$
$$\left.\mathrm{pp} \to \frac{(1 + g + (-1 + g)\ z)\ (-1 + g + z + g\,z)}{(-1 + z)^2 + g^2\ (1 + z)^2 + g\,k\ \left(-1 + z^2\right)},\ \mathrm{ap} \to \frac{(-1 + z)^2 + g^2\ (1 + z)^2 + g\ \left(k - k\,z^2\right)}{(-1 + z)^2 + g^2\ (1 + z)^2 + g\,k\ \left(-1 + z^2\right)}\right\}$$



Low pass gain
20
10
0.01
0.05
0.10
0.50
−10
−20
−30
Band pass gain
20
10
0.01
0.05
0.10
0.50
−10
−20
−30
High pass gain
20
10
0.01
0.05
0.10
0.50
−10
−20
−30


Notch gain
20
10
−10
−20
−30
0.01
0.05
0.10
0.50
Peak gain
20
10
−10
−20
−30
0.01
0.05
0.10
0.50
All gain
20
10
−10
−20
−30
0.01
0.05
0.10
0.50


All phase
3
2
1
−1
−2
−3
0.01
0.05
0.10
0.50

Bell

Matching to RBJ [6] continuous shape $H(s) = \frac{s^2 + (k\,A)\,s + 1}{s^2 + (k/A)\,s + 1}$

The bell shape is formed by adding the bandpass output to the input. There are two degrees of freedom to do this, k = 1/Q, and also the gain of the bandpass signal to be summed to the input. From the denominator we can see the k term is divided by A, the gain term, so we can make this substitution and then solve for the m0, m1, m2 terms by matching them to the numerator coefficients. The m0, m1, and m2 terms are the scales of the input, the bandpass output and the low pass output respectively.

```
slns1 = {bell -> (s^2 + (k A) s + 1) / (s^2 + (k / A) s + 1) // Simplify}

nodev1 = g (v0 - k / A v1 - v2) - (s c (v1 - 0)) == 0;
nodev2 = g (v1 - 0) - (s c (v2 - 0)) == 0;
bells = bell == (m0 v0 + m1 v1 + m2 v2) / v0;
subst = {c → 1};
slns2 = Solve[{nodev1, nodev2, bells} /. subst, {bell}, {v0, v1, v2}][[1]];
slns2 /. {g → 1}

Collect[Numerator[bell /. slns1], s] / Collect[Denominator[bell /. slns1], s]
Collect[Numerator[bell /. slns2], s] / Collect[Denominator[bell /. slns2], s]

c1 = CoefficientList[Numerator[bell /. slns1], s]
c2 = CoefficientList[Numerator[(bell /. slns2) /. {g → 1}], s]
msln =
 Solve[{c1[[1]] == c2[[1]], c1[[2]] == c2[[2]], c1[[3]] == c2[[3]]}, {m0, m1, m2}][[
   1]] // FullSimplify

nodev1 = g (v0 - k / A v1 - v2) - (gc (v1 - 0) - ic1eq) == 0;
nodev2 = g (v1 - 0) - (gc (v2 - 0) - ic2eq) == 0;
ic1eqn = ic1eq == 2 gc (v1 - 0) z^-1 - ic1eq z^-1;
ic2eqn = ic2eq == 2 gc (v2 - 0) z^-1 - ic2eq z^-1;
bellz = bell == (m0 v0 + m1 v1 + m2 v2) / v0;
subst = {gc → 1};
slnz = Solve[{nodev1, nodev2, ic1eqn, ic2eqn, bellz} /. subst,
    {bell}, {v0, v1, v2, ic1eq, ic2eq}][[1]] // FullSimplify

dB[x_] := 20 Log[10, Abs[x]];
wcrange = {wc, -6 + Log[2], -1, 1};
damp = 0.5;

gaindb = 10;
BodePlotSZ[(bell /. slns2) /. msln, (bell /. slnz) /. msln, "Bell boost 10dB"]
gaindb = -10;
BodePlotSZ[(bell /. slns2) /. msln, (bell /. slnz) /. msln, "Bell cut 10dB"]
```

$$\left\{\text{bell} \to \frac{A\left(1 + A\,k\,s + s^2\right)}{A + k\,s + A\,s^2}\right\}$$

$$\left\{\text{bell} \to \frac{A\,m0 + A\,m2 + k\,m0\,s + A\,m1\,s + A\,m0\,s^2}{A + k\,s + A\,s^2}\right\}$$

$$\frac{A + A^2\,k\,s + A\,s^2}{A + k\,s + A\,s^2}$$

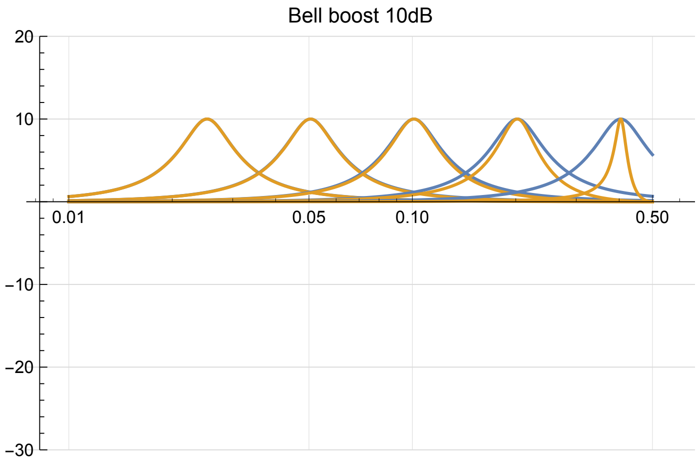
$$\frac{A\,g^2\,m0 + A\,g^2\,m2 + (g\,k\,m0 + A\,g\,m1)\,s + A\,m0\,s^2}{A\,g^2 + g\,k\,s + A\,s^2}$$

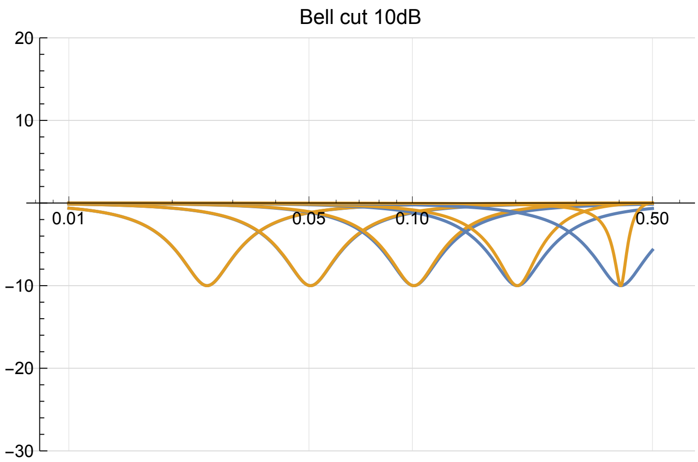
$$\{A,\ A^2\,k,\ A\}$$

$$\{A\,m0 + A\,m2,\ k\,m0 + A\,m1,\ A\,m0\}$$

$$\left\{m0 \to 1,\ m1 \to \frac{(-1 + A^2)\,k}{A},\ m2 \to 0\right\}$$

$$\left\{bell \to \left(g\,k\,m0\,(-1 + z^2) + A\,\left(g\,(1 + z)\,(m1\,(-1 + z) + g\,m2\,(1 + z)) + m0\,\left((-1 + z)^2 + g^2\,(1 + z)^2\right)\right)\right) \Big/ \left(g\,k\,(-1 + z^2) + A\,\left((-1 + z)^2 + g^2\,(1 + z)^2\right)\right)\right\}$$



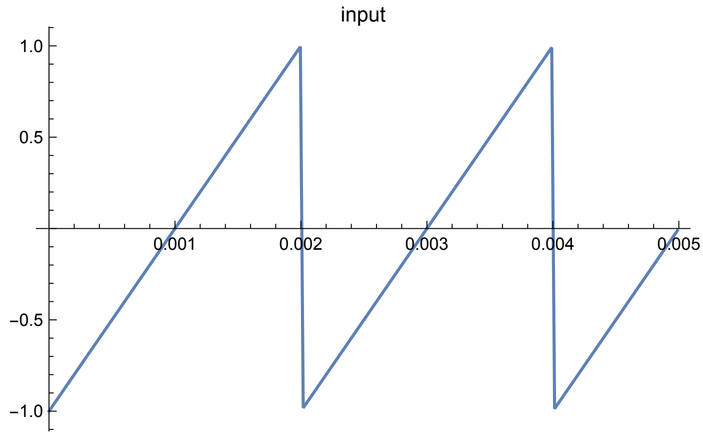


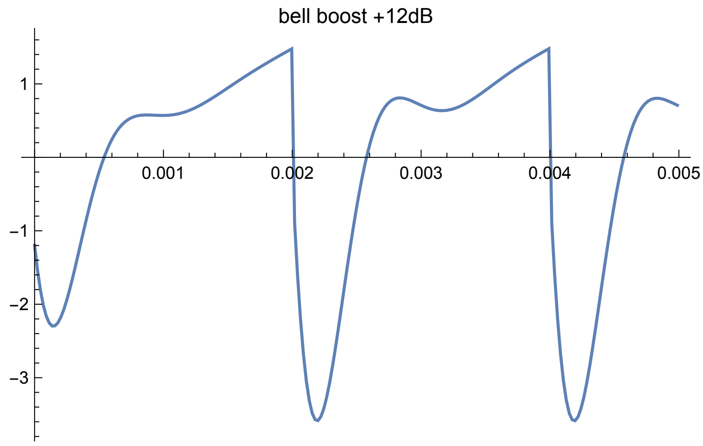
Algorithm bell bounded

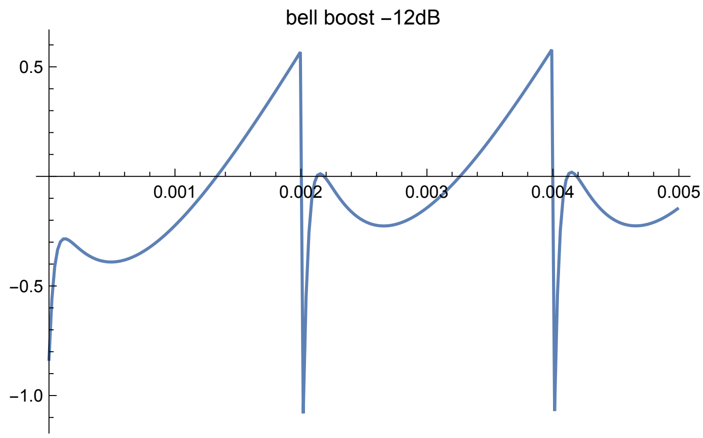
```
Clear
ic1eq = 0;
ic2eq = 0;

Set
A = Power[10, bellgaindB/40];
g = Tan[π cutoff/samplerate];
k = 1/(Q*A);
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 1;
m1 = k*(A*A - 1);
m2 = 0;

Tick
(7*, 7+, 14 total ops for bell)

v3 = v0 - ic2eq;
v1 = a1*ic1eq + a2*v3;
v2 = ic2eq + a2*ic1eq + a3*v3;
ic1eq = 2*v1 - ic1eq;
ic2eq = 2*v2 - ic2eq;

bell = v0 + m1*v1;
```

Test of algorithm bell bounded

```
ClearState[] := Block[{},
    ic1eq = 0;
    ic2eq = 0;
  ];

SetCoeff[cutoff_, Q_, bellgaindB_, samplerate_] := Block[{A},
    A = Power[10, bellgaindB / 40.0];
    g = Tan[π cutoff / samplerate];
    k = 1.0 / (Q * A);
    a1 = 1 / (1 + g * (g + k));
    a2 = g * a1;
    a3 = g * a2;
    m0 = 1.0;
    m1 = k * (A * A - 1);
    m2 = 0.0;
  ];

Tick[t_, v0_] := Block[{v1, v2, v3, bell},
    v3 = v0 - ic2eq;
    v1 = a1 * ic1eq + a2 * v3;
    v2 = ic2eq + a2 * ic1eq + a3 * v3;
    ic1eq = 2 * v1 - ic1eq;
    ic2eq = 2 * v2 - ic2eq;
    bell = v0 + m1 v1;
    Return[{t, v0, bell}]
  ];

h = 1.0 / 44100.0;
t1 = 0.005;
drive = 1;
freq = 500.0;

MySaw[x_] := 2 (x - Floor[x] - 0.5);
MyOsc[x_] := drive MySaw[freq x];

ClearState[];
SetCoeff[1000.0, 0.5, 12, 44100.0];
tp0 = Table[Tick[t, MyOsc[t]], {t, 0, t1, h}];

ListPlot[{tp0[[All, {1, 2}]]},
 PlotLabel → "input", Joined → True, PlotRange → All]
ListPlot[{tp0[[All, {1, 3}]]}, PlotLabel → "bell boost +12dB",
 Joined → True, PlotRange → All]

ClearState[];
SetCoeff[1000.0, 0.5, -12, 44100.0];
tp0 = Table[Tick[t, MyOsc[t]], {t, 0, t1, h}];
ListPlot[{tp0[[All, {1, 3}]]},
 PlotLabel → "bell boost -12dB", Joined → True, PlotRange → All]
```


input
1.0
0.5
-0.5
-1.0
0.001
0.002
0.003
0.004
0.005


bell boost +12dB
1
-1
-2
-3
0.001
0.002
0.003
0.004
0.005


bell boost -12dB
0.5
-0.5
-1.0
0.001
0.002
0.003
0.004
0.005

Low shelf

Matching to RBJ [6] continuous shape $H(s) = \frac{s^2 + k\sqrt{A}\ s + A}{A s^2 + k\sqrt{A}\ s + 1}$

The low shelf filter moves the cutoff frequency lower (divided by Sqrt[A]) as the shelf gain is increaesd so as to keep the frequency of the where half the shelf gain occurs constant. We can then solve for m0, m1, and m2 to match the numerator of the low shelf transfer function.

```
slns1 = {shelf -> A (s^2 + k Sqrt[A] s + A) / (A s^2 + k Sqrt[A] s + 1)} // FullSimplify

nodev1 = g / Sqrt[A] (v0 - k v1 - v2) - (s c (v1 - 0)) == 0;
nodev2 = g / Sqrt[A] (v1 - 0) - (s c (v2 - 0)) == 0;
shelfs = shelf == (m0 v0 + m1 v1 + m2 v2) / (v0);
subst = {c → 1};
eqn = {nodev1, nodev2, shelfs} /. subst;
slns2 = (Solve[{nodev1, nodev2, shelfs} /. subst, {shelf}, {v0, v1, v2}][[1]] //
    Simplify)

Collect[Numerator[shelf /. slns1], s] / Collect[Denominator[shelf /. slns1], s]
Collect[Numerator[shelf /. slns2], s] / Collect[Denominator[shelf /. slns2], s]

c1 = CoefficientList[Numerator[shelf /. slns1], s]
c2 = CoefficientList[Numerator[(shelf /. slns2) /. {g → 1}], s]
msln =
 Solve[{c1[[1]] == c2[[1]], c1[[2]] == c2[[2]], c1[[3]] == c2[[3]]}, {m0, m1, m2}][[
    1]] // FullSimplify

nodev1 = g / Sqrt[A] (v0 - k v1 - v2) - (gc (v1 - 0) - ic1eq) == 0;
nodev2 = g / Sqrt[A] (v1 - 0) - (gc (v2 - 0) - ic2eq) == 0;
ic1eqn = ic1eq == 2 gc (v1 - 0) z^-1 - ic1eq z^-1;
ic2eqn = ic2eq == 2 gc (v2 - 0) z^-1 - ic2eq z^-1;
shelfz = shelf == (m0 v0 + m1 v1 + m2 v2) / (v0);
subst = {gc → 1};
slnz = Solve[{nodev1, nodev2, ic1eqn, ic2eqn, shelfz} /. subst,
     {shelf}, {v0, v1, v2, ic1eq, ic2eq}][[1]] // FullSimplify

dB[x_] := 20 Log[10, Abs[x]];
wcrange = {wc, -6 + Log[2], -1, 1};
damp = 0.5;

(shelf /. slns2) /. msln

gaindb = 10;
BodePlotSZ[ (shelf /. slns2) /. msln,
  (shelf /. slnz) /. msln, "Low shelf boost 10dB"]

gaindb = -10;
BodePlotSZ[(shelf /. slns2) /. msln,
  (shelf /. slnz) /. msln, "Low shelf cut 10dB"]
```

$$\left\{\text{shelf} \to \frac{A\left(A + \sqrt{A}\, k\, s + s^2\right)}{1 + \sqrt{A}\, k\, s + A\, s^2}\right\}$$

$$\left\{\text{shelf} \to \frac{g^2\,(m0 + m2) + \sqrt{A}\, g\,(k\, m0 + m1)\, s + A\, m0\, s^2}{g^2 + \sqrt{A}\, g\, k\, s + A\, s^2}\right\}$$

$$\frac{A^2 + A^{3/2}\, k\, s + A\, s^2}{1 + \sqrt{A}\, k\, s + A\, s^2}$$

$$\frac{g^2\,(m0 + m2) + \sqrt{A}\, g\,(k\, m0 + m1)\, s + A\, m0\, s^2}{g^2 + \sqrt{A}\, g\, k\, s + A\, s^2}$$

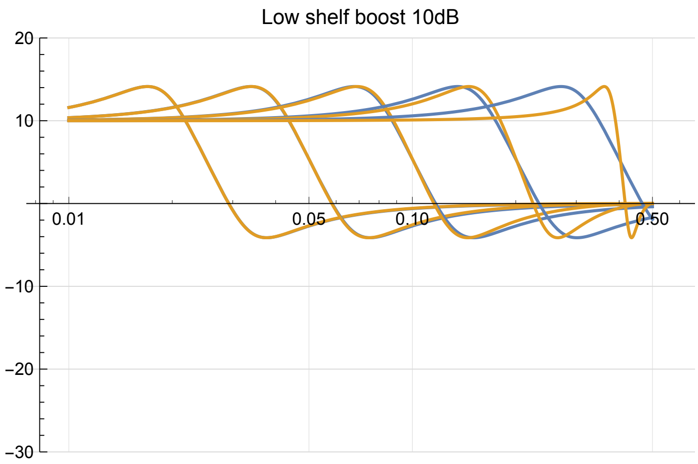
$$\left\{A^2,\ A^{3/2}\, k,\ A\right\}$$

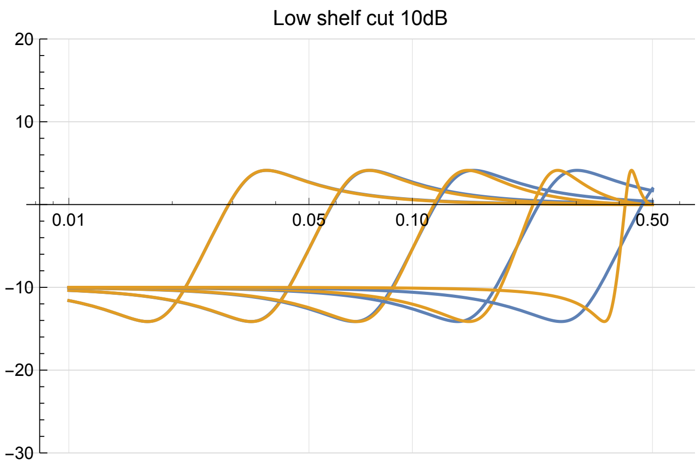
$$\left\{m0 + m2,\ \sqrt{A}\,(k\, m0 + m1),\ A\, m0\right\}$$

$$\left\{m0 \to 1,\ m1 \to (-1 + A)\, k,\ m2 \to -1 + A^2\right\}$$

$$\left\{\text{shelf} \to \left(A\, m0\,(-1 + z)^2 + g^2\,(m0 + m2)\,(1 + z)^2 + \sqrt{A}\, g\,(k\, m0 + m1)\left(-1 + z^2\right)\right) \Big/ \left(A\,(-1 + z)^2 + g^2\,(1 + z)^2 + \sqrt{A}\, g\, k\left(-1 + z^2\right)\right)\right\}$$

$$\frac{A^2\, g^2 + \sqrt{A}\, g\,(k + (-1 + A)\, k)\, s + A\, s^2}{g^2 + \sqrt{A}\, g\, k\, s + A\, s^2}$$



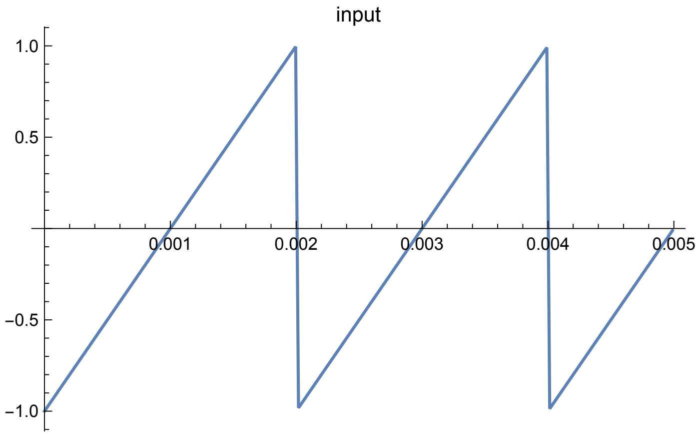


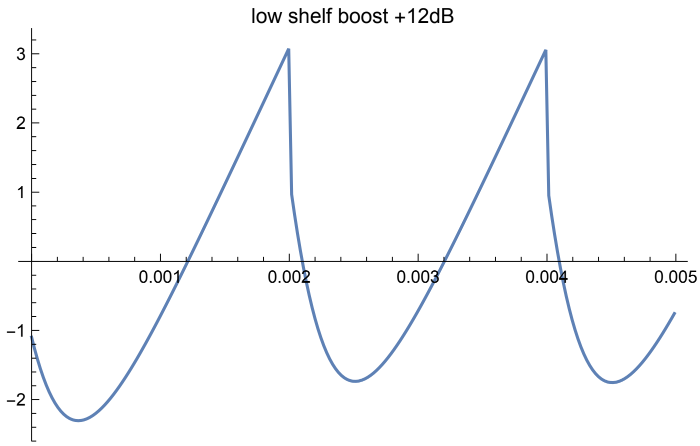
Algorithm low shelf bounded

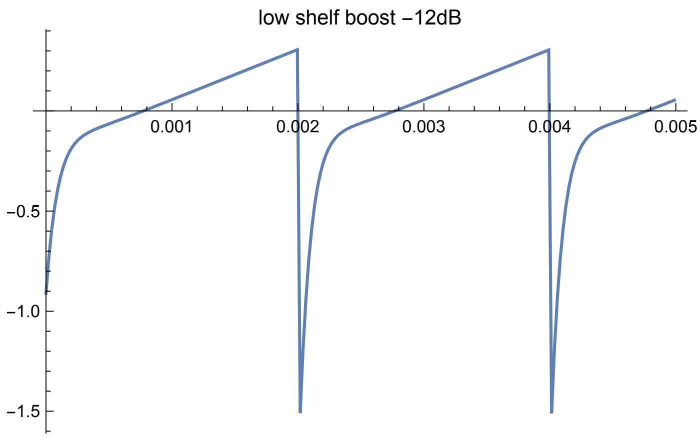
```
Clear
ic1eq = 0;
ic2eq = 0;

Set
A = Power[10, bellgaindB/40];
g = Tan[π cutoff/samplerate]/Sqrt[A];
k = 1/(Q);
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 1;
m1 = k*(A - 1);
m2 = (A*A - 1);

Tick
(8*, 8+, 16 total ops for bell)

v3 = v0 - ic2eq;
v1 = a1*ic1eq + a2*v3;
v2 = ic2eq + a2*ic1eq + a3*v3;
ic1eq = 2*v1 - ic1eq;
ic2eq = 2*v2 - ic2eq;

bell = v0 + m1*v1 + m2*v2;
```

Test of algorithm low shelf

```
ClearState[] := Block[{},
    ic1eq = 0;
    ic2eq = 0;
  ];

SetCoeff[cutoff_, Q_, bellgaindB_, samplerate_] := Block[{},
    A = Power[10, bellgaindB / 40];
    g = Tan[π cutoff / samplerate] / Sqrt[A];
    k = 1 / (Q);
    a1 = 1 / (1 + g * (g + k));
    a2 = g * a1;
    a3 = g * a2;
    m0 = 1;
    m1 = k * (A - 1);
    m2 = (A * A - 1);
  ];

Tick[t_, v0_] := Block[{v1, v2, v3, lowshelf},
    v3 = v0 - ic2eq;
    v1 = a1 * ic1eq + a2 * v3;
    v2 = ic2eq + a2 * ic1eq + a3 * v3;
    ic1eq = 2 * v1 - ic1eq;
    ic2eq = 2 * v2 - ic2eq;
    lowshelf = v0 + m1 v1 + m2 v2;
    Return[{t, v0, lowshelf}]
  ];

h = 1.0 / 44 100.0;
t1 = 0.005;
drive = 1;
freq = 500.0;

MySaw[x_] := 2 (x - Floor[x] - 0.5);
MyOsc[x_] := drive MySaw[freq x];

ClearState[];
SetCoeff[1000.0, 0.5, 12, 44 100.0];
tp0 = Table[Tick[t, MyOsc[t]], {t, 0, t1, h}];

ListPlot[{tp0[[All, {1, 2}]]},
 PlotLabel → "input", Joined → True, PlotRange → All]
ListPlot[{tp0[[All, {1, 3}]]}, PlotLabel → "low shelf boost +12dB",
 Joined → True, PlotRange → All]

ClearState[];
SetCoeff[1000.0, 0.5, -12, 44 100.0];
tp0 = Table[Tick[t, MyOsc[t]], {t, 0, t1, h}];
ListPlot[{tp0[[All, {1, 3}]]},
 PlotLabel → "low shelf boost -12dB", Joined → True, PlotRange → All]
```


input
1.0
0.5
-0.5
-1.0
0.001
0.002
0.003
0.004
0.005


low shelf boost +12dB
3
2
1
-1
-2
0.001
0.002
0.003
0.004
0.005


low shelf boost -12dB
-0.5
-1.0
-1.5
0.001
0.002
0.003
0.004
0.005

High shelf

Matching to RBJ [6] continuous shape $H(s) = A\,\frac{A s^2 + k\sqrt{A}\ s + 1}{s^2 + k\sqrt{A}\ s + A}$

The high shelf filter moves the cutoff frequency higher (multiplied by Sqrt[A]) as the shelf gain is increaesd so as to keep the frequency of the where half the shelf gain occurs constant. We can then solve for m0, m1, and m2 to match the numerator of the high shelf transfer function.

```
slns1 = {shelf -> A * (A * s^2 + k Sqrt[A] s + 1) / (s^2 + k Sqrt[A] s + A)} // FullSimplify

nodev1 = g Sqrt[A] (v0 - k v1 - v2) - (s c (v1 - 0)) == 0;
nodev2 = g Sqrt[A] (v1 - 0) - (s c (v2 - 0)) == 0;
shelfs = shelf == (m0 v0 + m1 v1 + m2 v2) / (v0);
subst = {c → 1};
slns2 = Together[
  Solve[{nodev1, nodev2, shelfs} /. subst, {shelf}, {v0, v1, v2}][[1]] //
   FullSimplify]

Collect[Numerator[shelf /. slns1], s] / Collect[Denominator[shelf /. slns1], s]
Collect[Numerator[shelf /. slns2], s] / Collect[Denominator[shelf /. slns2], s]

c1 = CoefficientList[Numerator[shelf /. slns1], s]
c2 = CoefficientList[Numerator[(shelf /. slns2) /. {g → 1}], s]
msln =
 Solve[{c1[[1]] == c2[[1]], c1[[2]] == c2[[2]], c1[[3]] == c2[[3]]}, {m0, m1, m2}][[
   1]] // FullSimplify

nodev1 = g Sqrt[A] (v0 - k v1 - v2) - (gc (v1 - 0) - ic1eq) == 0;
nodev2 = g Sqrt[A] (v1 - 0) - (gc (v2 - 0) - ic2eq) == 0;
ic1eqn = ic1eq == 2 gc (v1 - 0) z^-1 - ic1eq z^-1;
ic2eqn = ic2eq == 2 gc (v2 - 0) z^-1 - ic2eq z^-1;
shelfz = shelf == (m0 v0 + m1 v1 + m2 v2) / (v0);
subst = {gc → 1};
slnz = Solve[{nodev1, nodev2, ic1eqn, ic2eqn, shelfz} /. subst,
    {shelf}, {v0, v1, v2, ic1eq, ic2eq}][[1]] // FullSimplify


dB[x_] := 20 Log[10, Abs[x]];
wcrange = {wc, -6 + Log[2], -1, 1};
damp = 0.5;

gaindb = 10;
BodePlotSZ[ (shelf /. slns2) /. msln,
 (shelf /. slnz) /. msln, "High shelf boost 10dB"]

gaindb = -10;
BodePlotSZ[(shelf /. slns2) /. msln,
 (shelf /. slnz) /. msln, "High shelf cut 10dB"]
```

$$\left\{\text{shelf} \to \frac{A\left(1 + \sqrt{A}\ k\, s + A\, s^2\right)}{A + \sqrt{A}\ k\, s + s^2}\right\}$$

$$\left\{\text{shelf} \to \frac{A\, g^2\, m0 + A\, g^2\, m2 + \sqrt{A}\, g\, k\, m0\, s + \sqrt{A}\, g\, m1\, s + m0\, s^2}{A\, g^2 + \sqrt{A}\, g\, k\, s + s^2}\right\}$$

$$\frac{A + A^{3/2}\, k\, s + A^2\, s^2}{A + \sqrt{A}\, k\, s + s^2}$$

$$\frac{A\, g^2\, m0 + A\, g^2\, m2 + \left(\sqrt{A}\, g\, k\, m0 + \sqrt{A}\, g\, m1\right)\, s + m0\, s^2}{A\, g^2 + \sqrt{A}\, g\, k\, s + s^2}$$

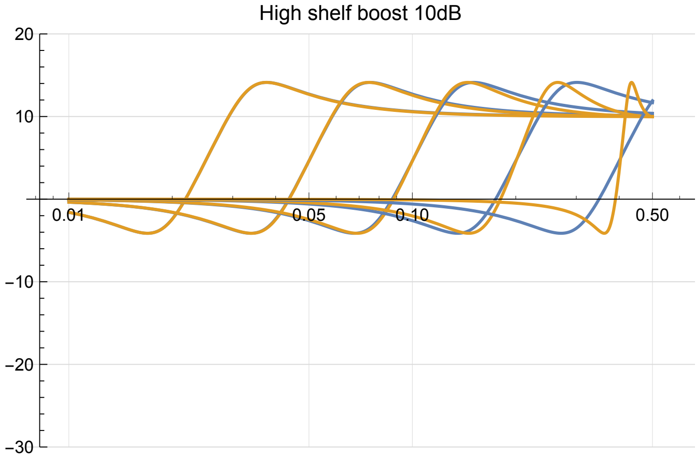
$$\left\{A,\ A^{3/2}\, k,\ A^2\right\}$$

$$\left\{A\, m0 + A\, m2,\ \sqrt{A}\, k\, m0 + \sqrt{A}\, m1,\ m0\right\}$$

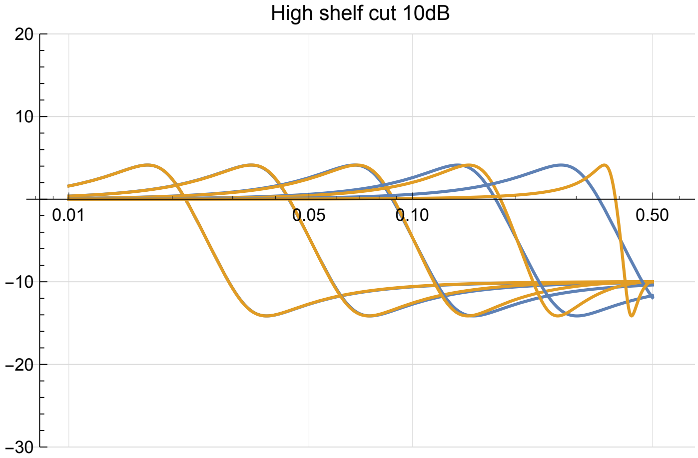
$$\left\{m0 \to A^2,\ m1 \to -(-1 + A)\, A\, k,\ m2 \to 1 - A^2\right\}$$

$$\left\{\text{shelf} \to \left(\sqrt{A}\, g\, (1 + z)\, \left(m1\, (-1 + z) + \sqrt{A}\, g\, m2\, (1 + z)\right) + m0\, \left((-1 + z)^2 + A\, g^2\, (1 + z)^2 + \sqrt{A}\, g\, k\, \left(-1 + z^2\right)\right)\right) \Big/ \left((-1 + z)^2 + A\, g^2\, (1 + z)^2 + \sqrt{A}\, g\, k\, \left(-1 + z^2\right)\right)\right\}$$

High shelf boost 10dB



High shelf cut 10dB

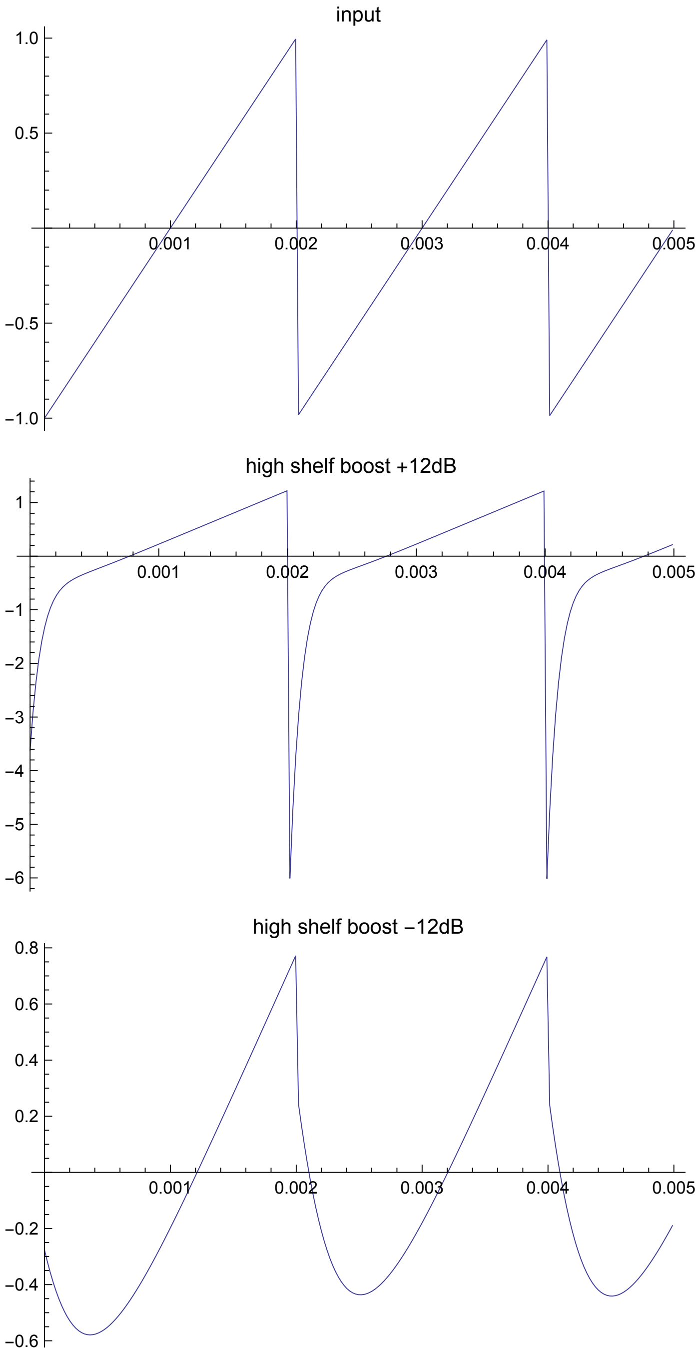


Algorithm high shelf bounded

```
Clear
ic1eq = 0;
ic2eq = 0;

Set
A = Power[10, bellgaindB/40];
g = Tan[π cutoff/samplerate]*Sqrt[A];
k = 1/(Q);
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = A*A;
m1 = k*(1 - A)*A;
m2 = (1 - A*A);

Tick
(9*, 8+, 17 total ops for high shelf)

v3 = v0 - ic2eq;
v1 = a1*ic1eq + a2*v3;
v2 = ic2eq + a2*ic1eq + a3*v3;
ic1eq = 2*v1 - ic1eq;
ic2eq = 2*v2 - ic2eq;

highshelf = m0*v0 + m1*v1 + m2*v2;
```

Test of algorithm high shelf

```
ClearState[] := Block[{},
   ic1eq = 0;
   ic2eq = 0;
  ];

SetCoeff[cutoff_, Q_, bellgaindB_, samplerate_] := Block[{A},
   A = Power[10, bellgaindB / 40.0];
   g = Tan[π cutoff / samplerate] * Sqrt[A];
   k = 1.0 / (Q);
   a1 = 1 / (1 + g * (g + k));
   a2 = g * a1;
   a3 = g * a2;
   m0 = A * A;
   m1 = k (1 - A) A;
   m2 = (1 - A A);
  ];

Tick[t_, v0_] := Block[{v1, v2, v3, highshelf},
   v3 = v0 - ic2eq;
   v1 = a1 * ic1eq + a2 * v3;
   v2 = ic2eq + a2 * ic1eq + a3 * v3;
   ic1eq = 2 * v1 - ic1eq;
   ic2eq = 2 * v2 - ic2eq;
   highshelf = m0 v0 + m1 v1 + m2 v2;
   Return[{t, v0, highshelf}]
  ];

h = 1.0 / 44 100.0;
t1 = 0.005;
drive = 1;
freq = 500.0;

MySaw[x_] := 2 (x - Floor[x] - 0.5);
MyOsc[x_] := drive MySaw[freq x];

ClearState[];
SetCoeff[1000.0, 0.5, 12, 44 100.0];
tp0 = Table[Tick[t, MyOsc[t]], {t, 0, t1, h}];

ListPlot[{tp0[[All, {1, 2}]]},
 PlotLabel → "input", Joined → True, PlotRange → All]
ListPlot[{tp0[[All, {1, 3}]]}, PlotLabel → "high shelf boost +12dB",
 Joined → True, PlotRange → All]

ClearState[];
SetCoeff[1000.0, 0.5, -12, 44 100.0];
tp0 = Table[Tick[t, MyOsc[t]], {t, 0, t1, h}];
ListPlot[{tp0[[All, {1, 3}]]},
 PlotLabel → "high shelf boost -12dB", Joined → True, PlotRange → All]
```


input
1.0
0.5
−0.5
−1.0
0.001
0.002
0.003
0.004
0.005
high shelf boost +12dB
1
−1
−2
−3
−4
−5
−6
0.001
0.002
0.003
0.004
0.005
high shelf boost −12dB
0.8
0.6
0.4
0.2
−0.2
−0.4
−0.6
0.001
0.002
0.003
0.004
0.005

Algorithm for every response

As has been demonstrated we can alter the cutoff, Q, and output mix to obtain all the various continuous bi-quadratic responses. This leads to a generic forumulation where, at the cost of additional computation, we can implement all the different responses with the same generic code, which can be useful for vector implementations.

With the appropriate interpolation you could also smoothly “morph” between all the responses, at each in between point are adjusting the mix of outputs as well as the cutoff and Q, so the resultant filter will remain stable and smooth even with audio rate modulation, although aliasing well result if the resultant modulation causes frequency or amplitude modulation sidebands to exceed the nyquist limit.

```
Clear
ic1eq = 0;
ic2eq = 0;

Set
case low
g = Tan[π cutoff/samplerate];
k = 1/Q;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 0;
m1 = 0;
m2 = 1;

case band
g = Tan[π cutoff/samplerate];
k = 1/Q;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 0;
m1 = 1;
m2 = 0;

case high
g = Tan[π cutoff/samplerate];
k = 1/Q;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 1;
m1 = -k;
m2 = -1;

case notch
g = Tan[π cutoff/samplerate];
k = 1/Q;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 1;
m1 = -k;
m2 = 0;

case peak
g = Tan[π cutoff/samplerate];
k = 1/Q;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 1;
m1 = -k;
m2 = -2;
```

```
case all
g = Tan[π cutoff/samplerate];
k = 1/Q;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 1;
m1 = -2*k;
m2 = 0;

case bell
A = Power[10, bellgaindB/40];
g = Tan[π cutoff/samplerate];
k = 1/(Q*A);
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 1;
m1 = k*(A*A - 1);
m2 = 0;

case low shelf
A = Power[10, bellgaindB/40];
g = Tan[π cutoff/samplerate]/Sqrt[A];
k = 1/Q;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = 1;
m1 = k*(A - 1);
m2 = (A*A - 1);

case high shelf
A = Power[10, bellgaindB/40];
g = Tan[π cutoff/samplerate]*Sqrt[A];
k = 1/Q;
a1 = 1/(1 + g*(g + k));
a2 = g*a1;
a3 = g*a2;
m0 = A*A;
m1 = k*(1 - A)*A;
m2 = (1 - A*A);

Tick
(9*, 8+, 17 total ops for any output)

v3 = v0 - ic2eq;
v1 = a1*ic1eq + a2*v3;
v2 = ic2eq + a2*ic1eq + a3*v3;
ic1eq = 2*v1 - ic1eq;
ic2eq = 2*v2 - ic2eq;

output = m0*v0 + m1*v1 + m2*v2;
```