Digital State-Variable Filters

Fons ADRIAENSEN
fons@linuxaudio.org

1 Introduction

In the analog signal processing world the state-variable filter is widely used. Figure 1 shows the usual way to implement it. There are many reasons for its popularity. Using the right active components for the frequency range of interest, it will always be stable. It provides highpass, resonance (bandpass) and low-pass outputs. Resonance or cutoff frequency and Q-factor can be controlled separately without problem, making it a good choice for e.g. parametric equalisers in audio applications.

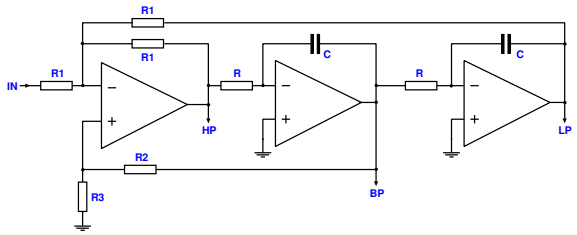

Figure 1: Analog SV filter

In the digital world, the biquad filter shown in Fig. 2 filter is much more popular. Again there are reasons for this. It's a 'textbook' filter that every DSP engineer is supposed to know. Many filter design tools ouput 'second order sections' and provide biquad coefficients for each of them.

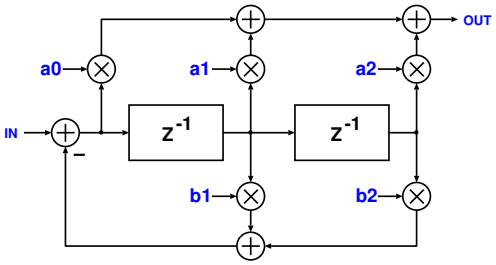

Figure 2: Biquad filter

However, the biquad is not without problems. One of them is potential instability in case the coefficients are interpolated, for example to avoid 'zipper noise' in audio equalisers. Another problem is numerical precision. For very low (relative to sample rate) values of the resonance frequency one finds coefficient values of the form $2-\epsilon$ or $1-\epsilon$, with very small ϵ proportional to design frequency. Even for audio applications at 48 kHz sample rate this means that using 32-bit floating point values can result in significant loss of precision.

The digital state-variable filter does not have these problems. So one may wonder why it is not used more often. The main reason seems to poor documentation.

2 The digital SV filter

If the delay elements in the biquad are replaced by integrators we get something very similar to the analog SV filter. This is shown in Fig 3.

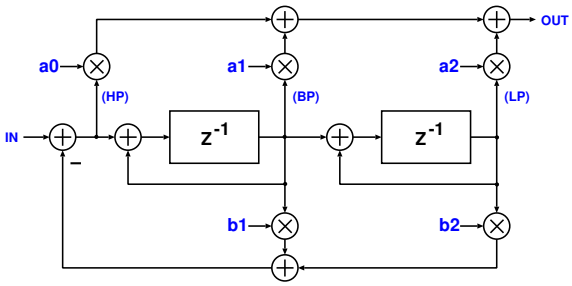

Figure 3: Digital SV filter

By shuffling the multipliers around many equivalent forms of Fig 3 can be found. Figure 4 shows the one that will be used in the remainder of this report. Here the feedback coefficients are constant, and instead we have integrator gains $c1$ and $c2$. This is similar to the analog form, except that since both feedack coefficients are fixed, we now control the Q-factor via the ratio

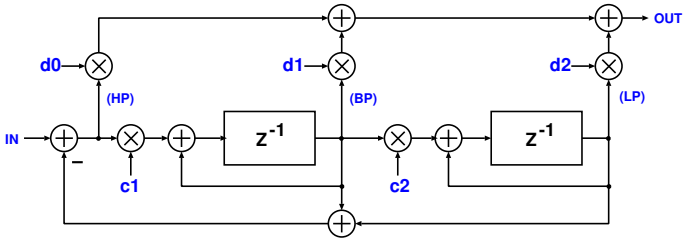

Figure 4: Alternative implementation, used in this report

$c2/c1$, and the bandpass gain will be 1 instead of Q at the resonance frequency.

The **C** code to process a block of n samples would be something like

```
for (i = 0; i < n; i++)
{
    x = inp [i] - z1 - z2;
    out [i] = d0 * x + d1 * z1 + d2 * z2;
    z2 += c2 * z1;
    z1 += c1 * x;
}
```

where $z1$ and $z2$ are the state of the filter.

Given F and Q, and taking frequency warping in to account, the intuitively obvious way to compute $c1$ and $c2$ would be:

$$\begin{aligned} w &= 2 * \tan(\pi * F) \\ c1 &= w/Q \\ c2 &= w * Q \end{aligned}$$

where F is the resonance frequency divided by the sample rate.
For very low values of F, the responses at the points marked (HP), (BP) and (LP) in Fig. 4 will indeed be similar to the those of the analog version. But for higher values of F this is no longer true.

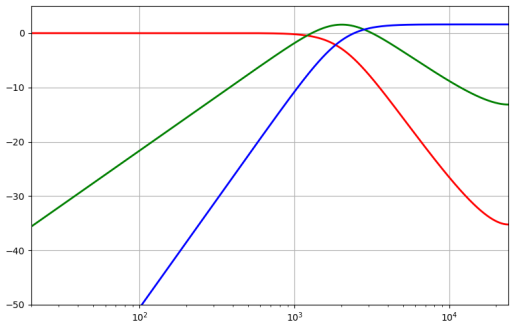

Figure 5: (HP), (BP) and (LP)

Figure 5 shows what happens for $F = 1/24$ and $Q = 0.71$. The passband gains of the highpass and bandpass are too high, and the lowpass and bandpass gains are not zero at the Nyquist frequency (which due to frequency warping maps to infinity in the analog domain).

Things get worse for higher frequencies and Q factors, up to the point where the filter will become unstable.

As shown in the next section, it is actually quite easy to get the correct responses. What is surprising is that there seem to be no other publications pointing out how to do this. Most authors, even Julius Smith (Smith, 2020), just ignore the subject.

3 Exact HP, BP and LP

In order to get the exact responses we need to modify the calculation of $c1$ and $c2$ (this will ensure stability), and combine the (HP), (BP) and (LP) signals in the correct proportions.

The proof of all the equations below is left as an exercise to the reader (it only takes some basic algebra, pen and paper).

The calculation of $c1$ and $c2$ is the same in all three cases. Given F and Q

$$\begin{aligned} w &= 2 * \tan(\pi * F) \\ a &= w/Q \\ b &= w^2 \\ c1 &= (a+b)/(1+a/2+b/4) \\ c2 &= b/(a+b) \end{aligned}$$

3.1 Highpass

For a highpass filter:

$$\begin{aligned} d0 &= 1 - c1/2 + c1 * c2/4 \\ d1 &= 0 \\ d2 &= 0 \end{aligned}$$

The **C** code becomes:

```
d0 = 1 - c1 / 2 + c1 * c2 / 4;
for (i = 0; i < n; i++)
{
    x = inp [i] - z1 - z2;
    out [i] = d0 * x;
    z2 += c2 * z1;
    z1 += c1 * x;
}
```

3.2 Bandpass

For a bandpass filter:

$$\begin{aligned} d0 &= (1 - c2) * c1/2 \\ d1 &= 1 - c2 \\ d2 &= 0 \end{aligned}$$

The **C** code becomes:

```
d1 = 1 - c2
d0 = d1 * c1 / 2
for (i = 0; i < n; i++)
{
    x = inp [i] - z1 - z2;
    out [i] = d0 * x + d1 * z1;
    z2 += c2 * z1;
    z1 += c1 * x;
}
```

3.3 Lowpass

For a lowpass filter:

$$\begin{aligned} d0 &= c1 * c2/4 \\ d1 &= c2 \\ d2 &= 1 \end{aligned}$$

The **C** code becomes:

```
d0 = c1 * c2 / 4
for (i = 0; i < n; i++)
{
    x = inp [i] - z1 - z2;
    z2 += c2 * z1;
    out [i] = d0 * x + z2;
    z1 += c1 * x;
}
```

Here we have used the fact that $d1 = c2$ to eliminate one multiplication and addition.

Note that the three values for each of $d0$, $d1$ and $d2$ sum to unity.

Figure 6 shows the result of using the equations above, again for $F = 1/24$ and $Q = 0.71$.

4 Transformation of an arbitray biquad

It is also easy to transform any given biquad to the state variable form. Using the coefficient names in Figs. 2 and 4:

$$\begin{aligned} c1 &= b1 + 2 \\ c2 &= (1 + b1 + b2)/c1 \\ d0 &= a0 \\ d1 &= (2 * a0 + a1)/c1 \\ d2 &= (a0 + a1 + a2)/(c1 * c2) \end{aligned}$$

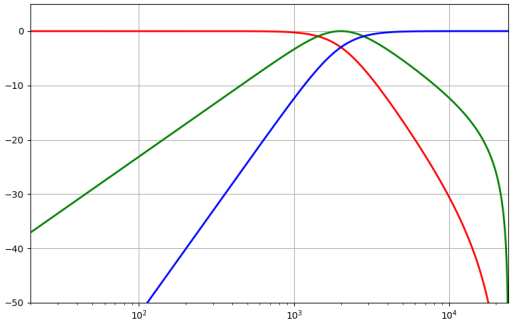

Figure 6: Exact HP, BP, LP

References

Julius O. Smith. 2020. *Digital State-Variable Filters*. https://ccrma.stanford.edu/~jos/svf/svf.pdf.