

# Digital State-Variable Filters

JULIUS O. SMITH III

*Center for Computer Research in Music and Acoustics (CCRMA)  
Department of Music, Stanford University, Stanford, California 94305 USA*

jos@stanford.edu

January 12, 2019

This tutorial covers realization of a continuous-time second-order lowpass filter as a state-space model which is then digitized via both forward and backward Euler schemes to produce the so-called *Chamberlin form* of a digital *state-variable filter* having desirable numerical properties. Finally, the FAUST form of the filter is derived.

## 1 Normalized Second-Order Continuous-Time Lowpass Filter

The transfer function of a *normalized* second-order lowpass can be written as

$$H_l(s) = \frac{1}{\tilde{s}^2 + \frac{1}{Q}\tilde{s} + 1}$$

where the normalization maps the desired -3dB frequency  $\omega_c$  to 1, *i.e.*,

$$\tilde{s} \triangleq \frac{s}{\omega_c},$$

and the “quality factor”  $Q$  is defined as

$$Q \triangleq \frac{\omega_c}{2\alpha}$$

where  $\alpha$  is a convenient definition for *bandwidth* in radians per second, given by minus the real part of the complex-conjugate pole locations  $p$  and  $\bar{p}$  in the  $s$  plane:

$$p, \bar{p} = -\alpha \pm j\sqrt{\omega_c^2 - \alpha^2}$$

Here we assume  $0 < \alpha < \omega_c$ , so that the poles have nonzero imaginary parts.

A second-order *Butterworth* lowpass filter is obtained for  $Q = 1/\sqrt{2}$ . Larger  $Q$  values give the “corner resonance” effect often used in music synthesizers.

## 2 Bode Plots for Second-Order Butterworth Filters

Filters of this type are nicely viewed in a *Bode plot* which shows the magnitude frequency response (in dB) versus a log frequency axis. In matlab we can say, for example,

```

sys = tf(1,[1,sqrt(2),1]);
bode(sys);

```

to see the frequency response of our normalized second-order Butterworth lowpass filter.

Note that our lowpass is easily converted to a bandpass or highpass filter by changing the transfer-function numerator from 1 to  $s$  or  $s^2$ , respectively:

```

bode(tf([0 0 1],[1,sqrt(2),1])); % lowpass
bode(tf([0 1 0],[1,sqrt(2),1])); % bandpass
bode(tf([1 0 0],[1,sqrt(2),1])); % highpass
bode(tf([1 0 1],[1,sqrt(2),1])); % notch

```

These frequency responses are shown in Fig. 1.

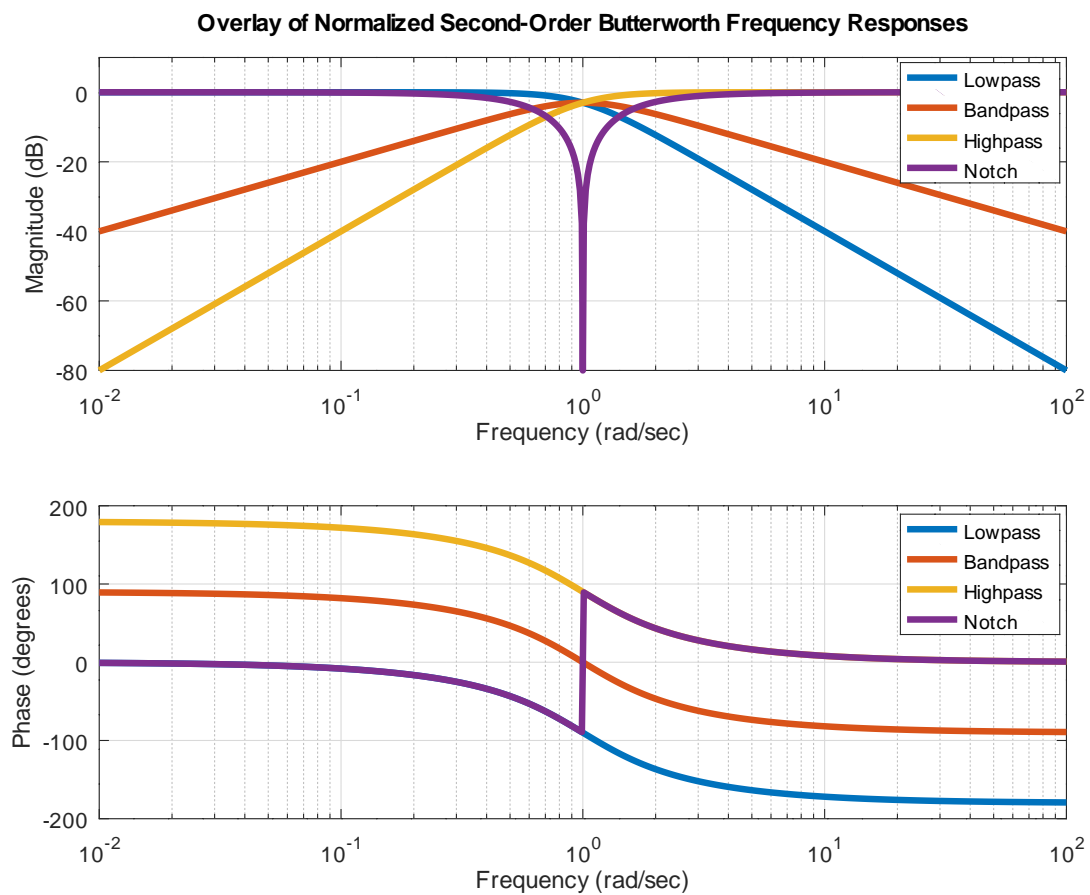


Figure 1: Overlay of normalized second-order Butterworth lowpass, bandpass, highpass, and notch filters.

### 3 Bode Plots for Second-Order Lowpass Filters with Corner Resonance

Figure 2 shows an overlay of frequency responses for various corner resonances.

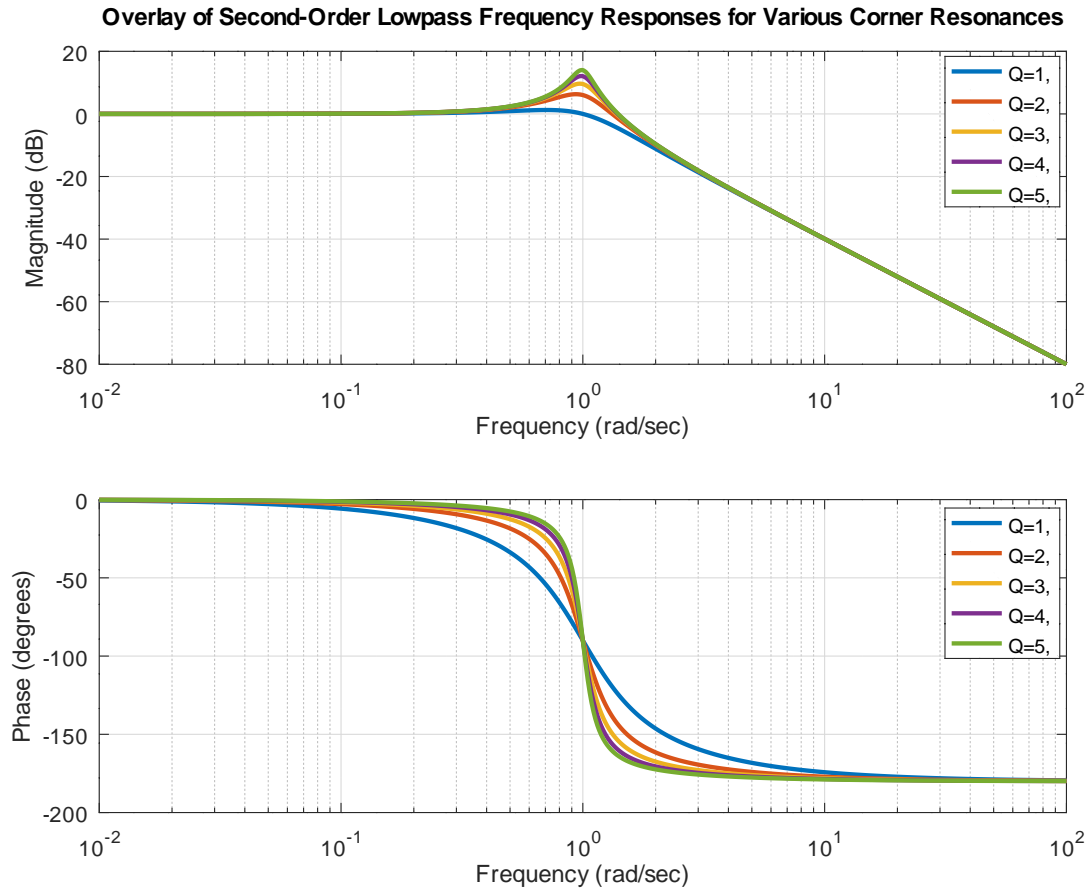


Figure 2: Overlay of second-order lowpass frequency responses for  $Q=1,2,3,4,5$ .

## 4 State Space Realization of Second-Order Continuous-Time Low-pass Filters

It is easy to realize a filter transfer function in state-space form by means of the so-called *controller-canonical form*, in which transfer-function coefficients appear directly in the matrices of the state-space form. In our case, the state-space model becomes

$$\dot{\underline{x}}(t) = \begin{bmatrix} 0 & 1 \\ -a_0 & -a_1 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = [c_0 \quad c_1] \underline{x}(t)$$

where  $a_0 = 1$ ,  $a_1 = \sqrt{2}$ ,  $c_0 = 1$ , and  $c_1 = 0$  for a normalized Butterworth lowpass filter. *I.e.*,

$$\dot{x}_1(t) = x_2(t)$$

$$\dot{x}_2(t) = -x_1(t) - \sqrt{2}x_2(t) + u(t)$$

$$y_l(t) = x_1(t).$$

This system is diagrammed in Fig. 3. Note that due to the chain of integrators in controller-canonical form, we also have available the bandpass and highpass outputs as shown in the figure. Each integrator is typically implemented (in analog circuits) by means of an *operational amplifier* (“op amp”) having a capacitor in its feedback loop.<sup>1</sup>

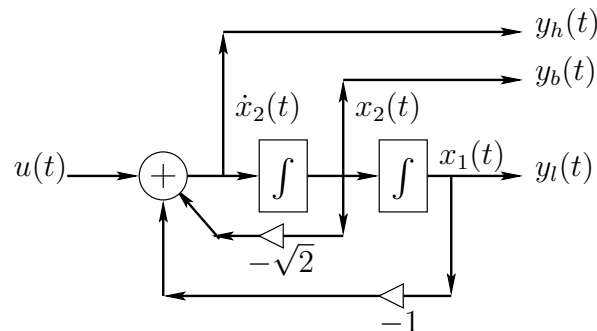


Figure 3: State-space realization of second-order normalized Butterworth lowpass filter.

<sup>1</sup>See, for example, <http://www.analog.com/static/imported-files/tutorials/MT-223.pdf>.

## 5 Digitization of Second-Order Continuous-Time Lowpass Filters in State Space Form

The only elements in Fig. 3 needing modification for digitization are the two integrators, each having transfer function  $1/\tilde{s}$ . Normally the preferred digitization method is the *bilinear transform*:

$$\tilde{s} \leftarrow \frac{1 - z^{-1}}{1 + z^{-1}}$$

However, the bilinear transform cannot be used here due to the presence of feedback which would give a *delay-free loop*.

The *Forward Euler* (FE) finite-difference scheme introduces a sample of delay in the digitization of  $1/s$  that avoids a delay-free loop:

$$\tilde{s} \leftarrow \frac{z - 1}{T} = \frac{1 - z^{-1}}{T z^{-1}}$$

where  $T$  denotes the sampling interval in seconds. There is also a *Backward Euler* (BE) finite-difference scheme:

$$\tilde{s} \leftarrow \frac{1 - z^{-1}}{T}$$

It can be effective to use FE and BE together in alternation to avoid delay build-up in either direction:

$$\tilde{s}^2 = \tilde{s} \cdot \tilde{s} \leftarrow \frac{z - 1}{T} \cdot \frac{1 - z^{-1}}{T} = \frac{z - 2 + z^{-1}}{T^2}$$

Digitizing the two integrators in Fig. 3 via FE and BE respectively and removing the frequency normalization yields

$$\frac{1}{s} \leftarrow \omega_c T \frac{z^{-1}}{1 - z^{-1}}$$

and  $\omega_c T \frac{1}{1 - z^{-1}}$ .

The resulting digital filter is drawn in Fig. 4. This structure is normally called the *Chamberlin form* digital filter section [1].

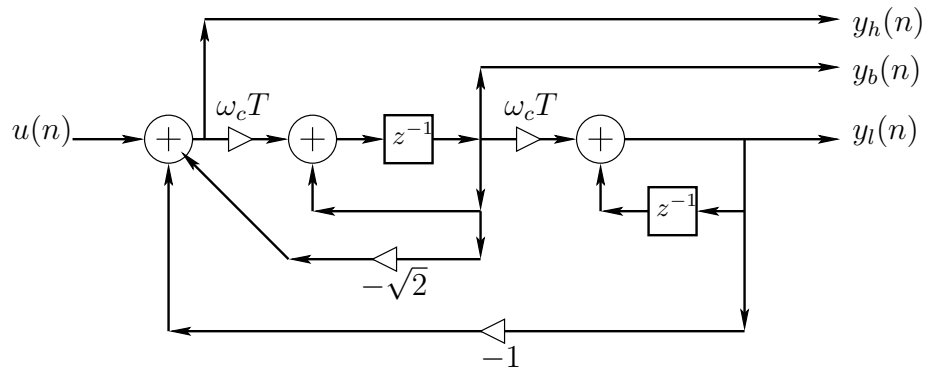


Figure 4: State-space realization in Fig. 3 digitized via forward and backward Euler.

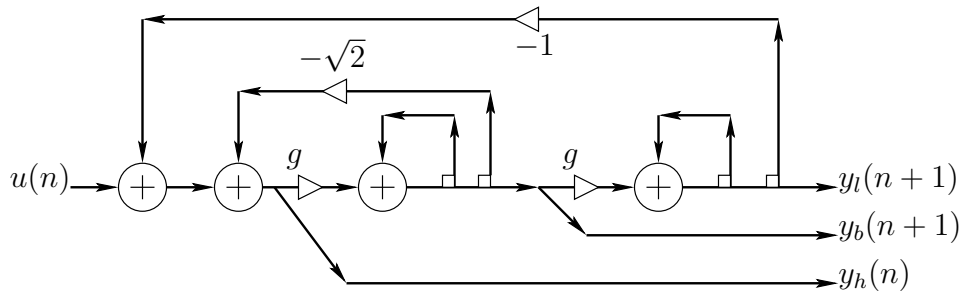
## 6 Faust Encoding of Second-Order Chamberlin Form

To encode the Chamberlin form in the FAUST language, it is helpful to redraw its diagram more like the FAUST compiler would:

- Signal flow strictly left to right, except for feedback
- Feedback paths go up and around instead of down and around
- Delays due to feedback are drawn as such

This is shown in Fig. 5 along with the FAUST encoding of the diagram. Note that the added delay from the digitization of the first integrator (using forward-Euler) has been “pushed” through the second integrator and into the outer feedback loop, resulting in the unit-sample time advances  $y_l(n+1)$  and  $y_b(n+1)$ . One can use `faust2firefox` (or the FAUST online compiler) to display the block diagram directly from the code, as shown in Fig. 6 for the following program:

```
g=0.1; // wcT or 2*sin(wcT/2)
process = ( + : ( + <: (*(g):+~_-),_-)~*(0-sqrt(2)) : (- <: (*(g):+~_-),_-)~*(-1);
```



```
(+ : (+ <: (*(g):+~_-),_-)~*(0 - sqrt(2)) : (- <: (*(g):+~_-),_-)~*(-1)
```

Figure 5: Redrawing of Fig. 4 in a more FAUSTian style.

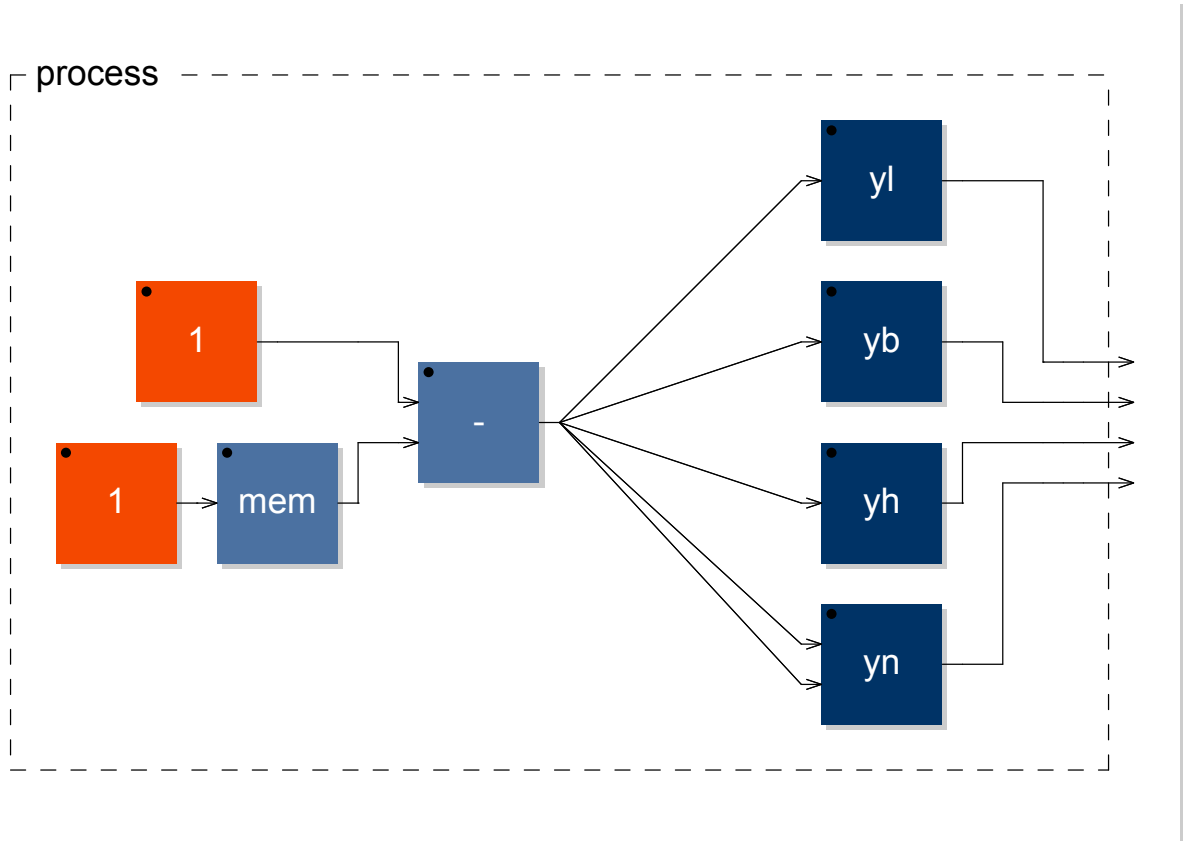


Figure 6: Block diagram generated by `faust2firefox`.



## 7 Bode Plots for Chamberlin Form in Faust

As a final test, let's use `faust2octave` to obtain the four overlaid Bode plots in Fig. 1, but now for the discrete-time system. Our test program is as follows:

```
g=0.1; // wcT or 2*sin(wcT/2) [see exact digital poles]
y = ( + : ( + <: (*(g):+~_),_ )~*(0-sqrt(2)) : ( _ <: (*(g):+~_),_ )~*(-1);
yl = y : _', ! , ! ; // lowpass (delay usually not needed)
yb = y : ! , _', ! ; // bandpass (delay usually not needed)
yh = y : ! , ! , _; // highpass
yn = yl + yh; // notch (delay compensation required here)
```

```
process = 1-1' <: yl,yb,yh,yn; // impulse-response test for faust2octave
```

Running `faust2octave` on this followed by `semilogx(w,db(abs(fft(faustout,8192)(1:4097,:))))` produces the plot overlays shown in Fig. 7.

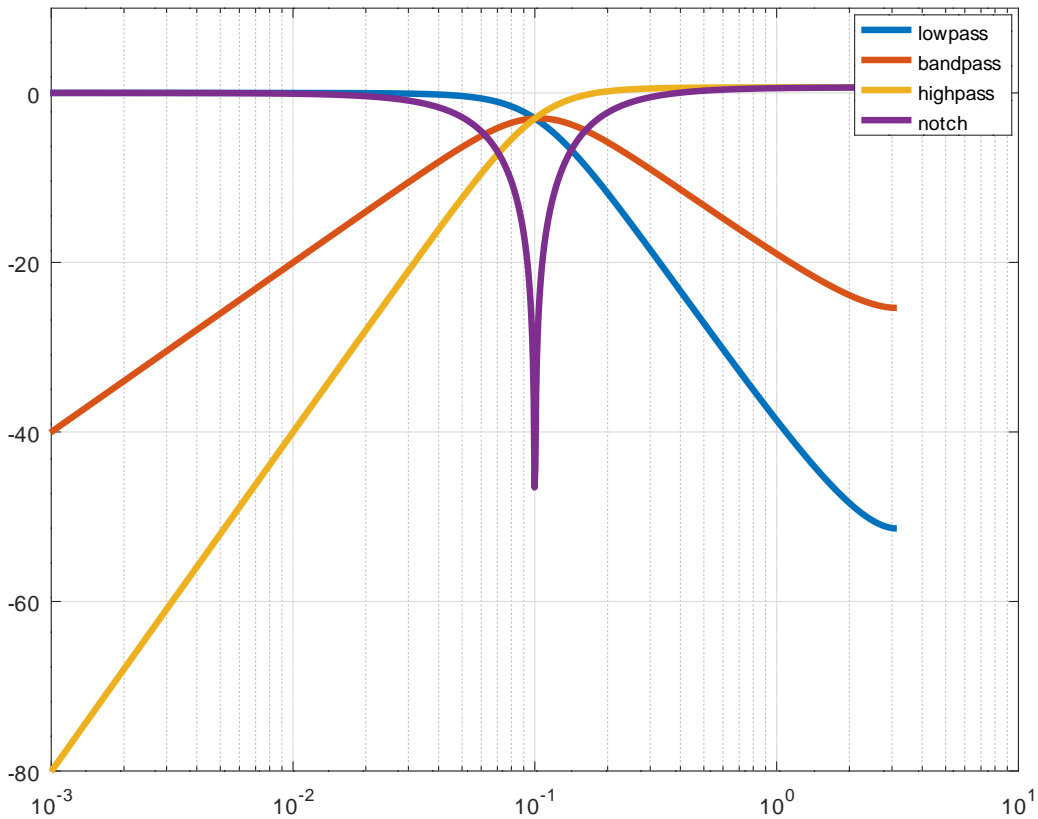


Figure 7: Overlay of FAUST implementations of the digitized Butterworth lowpass, bandpass, highpass, and notch filters in Chamberlin form.

## 8 Further Reading on State Variable Filters and Chamberlin Form

The state-variable filter is described further in [2]. For more about the Chamberlin form and related digital state-variable filters, see, *e.g.*, [4, 6, 3, 5].

### References

- [1] H. Chamberlin, *Musical Applications of Microprocessors, Second Edition*, New Jersey: Hayden Books, 1985.
- [2] D. Colin, “Electrical design and musical applications of an unconditionally stable combination voltage controlled filter / resonator,” *Journal of the Audio Engineering Society*, vol. 19, pp. 923–927, Dec. 1971, <http://www.guitarfool.com/ARP2500/DennisCollinPaper.pdf>.
- [3] J. Dattorro, “Effect design,” *Journal of the Audio Engineering Society*, vol. 45, pp. 660–684, September 1997.
- [4] T. Stilson, *Efficiently Variable Algorithms in Virtual-Analog Music Synthesis—A Root-Locus Perspective*, PhD thesis, Elec. Engineering Dept., Stanford University (CCRMA), June 2006, <https://ccrma.stanford.edu/~stilti/>.
- [5] D. K. Wise, “The modified Chamberlin and Zölzer filter structures,” in *Proceedings of the Conference on Digital Audio Effects (DAFx-06), Montréal, Canada, 2006*.
- [6] U. Zölzer, *Digital Audio Signal Processing*, New York: John Wiley and Sons, Inc., 1999.