Methods for Synthesizing Very High Q Parametrically Well Behaved Two Pole Filters

Max Mathews
Julius O. Smith III

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

Stockholm Musical Acoustic Conference (SMAC)

August 6-9, 2003

Abstract

Techniques for synthesizing two pole filters are well known. A number of techniques introduce unpleasant sounding transients in the filter response when the frequency or damping of the filter is rapidly changed. We will demonstrate a difference equation for a digital filter in which both the frequency and the damping can be changed without producing discontinuities in the filter output. The technique is based on the well known property of the product of complex numbers. In polar form, the magnitude of the product of two numbers is the product of their magnitudes and the angle of the product is the sum of their angles. Successive multiplies can produce a rotating vector whose real or imaginary parts are samples of constant amplitude sine waves or of exponentially damped sine waves. The frequency and damping of the resulting waves can be changed without changing the amplitude of the waves. These properties can be used to make a digital filter whose input, frequency, and damping can all be functions of time in a useful way. Two alternative structures are additionally proposed, having better numerical properties for low-cost fixed-point implementations. A program to demonstrate some musical applications of these filters will be shown.

Contents

1 Introduction

Filters, whose parameters can be varied in time provide an important new resource in musical timbres. High Q filters are also desirable since most musical instruments involve high Q mechanical filters. Unfortunately, tuning a high Q filter can introduce artifacts which change the energy in the filter and produce unpleasant transients in the sound. A mechanical filter consisting of an object on a spring provides a simple example of the problem. The resonant frequency of the object-spring system can be changed by either changing the stiffness of the spring or the mass of the object. But if the mass of the object is changed while the object is moving, the kinetic energy in the object will be changed. Likewise, if the stiffness of the spring is changed while the spring is stretched, the potential energy stored in the spring will be changed. Any change of energy will cause a change in the amplitude of the oscillation.

2 Oscillator based on Complex Number Multiplication

A well known difference equation exists which can compute samples of parametrically well behaved sinusoids [1, 2]. The frequency of the sinusoid can easily be changed without changing the amplitude. The equation is a property of the multiplication of two complex numbers.[1] Complex numbers can be represented as two dimensional vectors in a plane in which the x axis is the real part of the number and the y axis is the imaginary part of the number. The polar coordinate form of the vector is a magnitude and an angle (measured from the x axis). The magnitude of the product of two numbers is the product of the magnitudes of the two numbers and the angle of the product is the sum of the angles of the two numbers.

We will use the following definitions and relations:

$$\begin{aligned} j &\triangleq \sqrt{-1} \\ z &\triangleq \text{a complex number} \\ r &\triangleq |z| \\ \theta &\triangleq \angle z \end{aligned}$$

In terms of real and imaginary components:

$$z = x + jy$$

where

$$\begin{aligned} x &= r\cos(\theta) \\ y &= r\sin(\theta) \end{aligned}$$

Let a sequence of complex numbers

$$z(n), \quad n = 0, 1, 2, 3, ...$$

be formed from the product of two complex numbers

$$z(n+1) = z_1 z(n), \tag{1}$$

[1] http://ccrma.stanford.edu/~jos/mdft/Complex_Numbers.html

where $z_1 \stackrel{\Delta}{=} e^{j\theta_1} = \cos(\theta_1) + j\sin(\theta_1)$. The initial state is taken to be $z(0) = 1$, i.e., the initial magnitude and angle are $r(0) = 1$ and $\theta(0) = 0$, respectively. By the rules of complex number multiplication, we have

$$r(1) = r(2) = \cdots = r(n) = 1$$

and

$$\theta(n) = n\theta_1.$$

The imaginary part of $z(n)$ is the desired sine wave:

$$y(n) = \sin(n\theta_1).$$

The magnitude of the sine wave is always 1. The period of the sine wave is $2\pi/\theta_1$ samples.

By changing θ_1, the period can be changed without affecting the magnitude.

The actual difference equations to compute the sine wave are obtained by writing Eq. (1) as real and imaginary parts. This yields two difference equations which can be extrapolated to compute $x(n)$ and $y(n)$:

$$x(n+1) = x_1 x(n) - y_1 y(n) \quad (2)$$

$$y(n+1) = y_1 x(n) + x_1 y(n) \quad (3)$$

where $x_1 = \cos(\theta_1)$ and $y_1 = \sin(\theta_1)$.

3 High Q Filter Difference Equations

3.1 Homogeneous Solution

Equations (1–3) can be modified to compute the response of a finite Q filter simply by changing r_1 to some value different from unity. From Eq. (1), $|z(n)|$ will be r_1 raised to the nth power, and $y(n)$ will be

$$y(n) = r_1^n \sin(n\theta_1) \quad (4)$$

If r_1 is less than one, Eq. (4) shows the homogeneous solution to Eq. (1) is a sinusoid times a decaying exponential. This solution is the response of a two pole filter.

Equations (2–3) are unchanged except that the coefficients x_1 and y_1 become

$$x_1 = r_1 \cos(\theta_1) \quad (5)$$

$$y_1 = r_1 \sin(\theta_1) \quad (6)$$

If $y(n)$ is the input to a digital-to-analog converter operating at a sampling rate of f_s samples per second, then x_1 and y_1 can be written

$$x_1 = e^{-\frac{1}{\tau f_s}} \cos\left(2\pi \frac{f}{f_s}\right) \quad (7)$$

$$y_1 = e^{-\frac{1}{\tau f_s}} \sin\left(2\pi \frac{f}{f_s}\right) \quad (8)$$

where

$$\tau \triangleq \text{time in seconds for filter to decay to } 1/e \tag{9}$$

$$f \triangleq \text{resonant frequency of filter in Hz} \tag{10}$$

$$f_s \triangleq \text{sampling rate in samples per second} \tag{11}$$

Equations (7–10) are valid for f less than $f_s/2$. For f between $f_s/2$ and f_s, the generated frequency will be reflected about $f_s/2$.

Negative values of τ will generate exponentially increasing sinusoids which can be musically useful.

3.2 Forced Solution

The output of our digital filter is $y(n)$. We have not so far provided a mechanism to input a series of samples of a time function into the filter. An input is necessary if we wish to use the filter in a traditional way—that is, to "filter" samples of a sound wave. A number of possibilities exist for adding input samples to our difference equation. We have chosen to add input samples to $x(n)$. With this modification, Equations (2–3) become

$$x(n+1) = x_1 x(n) - y_1 y(n) + u(n) \tag{12}$$

$$y(n+1) = y_1 x(n) + x_1 y(n) \tag{13}$$

where $u(n)$ are samples of the input to the filter. The transfer function of the filter is then

$$H(z) \triangleq \frac{Y(z)}{U(z)} = \frac{y_1 z^{-2}}{1 - 2x_1 z^{-1} + (x_1^2 + y_1^2) z^{-2}} \tag{14}$$

$$= \frac{r_1 \sin(\theta_1) z^{-2}}{1 - 2r_1 \cos(\theta_1) z^{-1} + r_1^2 z^{-2}} \tag{15}$$

This is a so-called "all pole" filter, since its two zeros are at $z = \infty$ and therefore do not affect the magnitude response. This choice of input can be interpreted as driving the poles in *series*, since the transfer functions of series (cascade) filter combinations simply multiply.

We have compared both the sound and the spectrum of our filter output with the outputs of other filter difference equations realizations and we have detected no differences.

We may alternatively add the input samples $u(n)$ on the right-hand side of Eq. (13) to obtain the transfer function

$$H(z) = \frac{z^{-1} - x_1 z^{-2}}{1 - 2x_1 z^{-1} + (x_1^2 + y_1^2) z^{-2}} \tag{16}$$

$$= \frac{z^{-1} - r_1 \cos(\theta_1) z^{-2}}{1 - 2r_1 \cos(\theta_1) z^{-1} + r_1^2 z^{-2}}. \tag{17}$$

This choice of input introduces a finite zero at $z = x_1 = r_1 \cos(\theta_1)$, which is the real part of the location of both poles. This can be interpreted as driving the poles in *parallel*, since denoting the

complex conjugate of $z = x + jy$ by $\overline{z} = x - jy$, we have

$$\begin{aligned} z_1^n + \overline{z}_1^n &\leftrightarrow \frac{1}{1 - z_1 z^{-1}} + \frac{1}{1 - \overline{z}_1 z^{-1}} && (18) \\ &= \frac{2 - 2\mathrm{re}\left\{z_1\right\} z^{-1}}{1 - 2\mathrm{re}\left\{z_1\right\} z^{-1} + |z_1|^2 z^{-2}} && (19) \\ &= 2\frac{1 - x_1 z^{-1}}{1 - 2x_1 z^{-1} + r_1^2 z^{-2}} = 2zH(z). && (20) \end{aligned}$$

3.3 State Space Formulation

Rewriting the filter in state-space form,[2] we have

$$\left[\begin{array}{c} x(n+1) \\ y(n+1) \end{array}\right] = \left[\begin{array}{cc} x_1 & -y_1 \\ y_1 & x_1 \end{array}\right] \left[\begin{array}{c} x(n) \\ y(n) \end{array}\right] + \left[\begin{array}{c} b_1 \\ b_2 \end{array}\right] u(n)$$

or

$$\begin{aligned} \underline{x}(n+1) &= \boldsymbol{A}\underline{x} + \boldsymbol{B}u(n) && (21) \\ y(n) &= \boldsymbol{C}^T\underline{x}(n) + Du(n) && (22) \end{aligned}$$

where $\boldsymbol{C}^T = [c_1, c_2] = [0, 1]$ and $D = 0$. As is well known, the transfer function of a state-space model (A,B,C,D) is given by

$$\begin{aligned} H(z) &= \boldsymbol{C}^T(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + D && (23) \\ &= \frac{e_1 z^{-1} + e_2 z^{-2}}{1 - 2r_1\cos(\theta_1)z^{-1} + r_1^2 z^{-2}}, && (24) \end{aligned}$$

where

$$\begin{aligned} e_1 &= c_1 b_1 + c_2 b_2 && (25) \\ e_2 &= -c_1(b_1 x_1 + b_2 y_1) + c_2(b_1 y_1 - b_2 x_1). && (26) \end{aligned}$$

Thus, under all choices of input or output, there is at most one real finite zero at $z = -e_2/e_1$.

3.4 Phase-Preserving Restrikes

An interesting third (nonlinear) input possibility is to add the input to the *instantaneous magnitude* of $z(n)$. This mode is useful for "restriking" a decaying filter oscillation in a *phase-preserving* manner. The result is that the amplitude of a decaying oscillation can be jumped without altering its phase at all.

Adding a real signal $u(n)$ to the magnitude of $z(n)$ is equivalent to scaling $z(n)$ by $g(n) = u(n)/r(n) + 1$, where $r(n) = |z(n)|$, to get

$$\left[\frac{u(n)}{r(n)} + 1\right] z(n) = [u(n) + r(n)]e^{j\theta(n)}.$$

[2] `http://ccrma.stanford.edu/~jos/filters/State_Space_Models.html`

Since amplitude jumps can be perceived as "clicks" when they are too fast, it is useful to delay amplitude jumps until the next zero-crossing of the output $y(n)$. Waiting for a zero-crossing in $y(n)$ is equivalent to waiting for the phase of $z(n)$ to reach either 0 or π.

Waiting for phase 0 to scale the state $z(n) = x(n)$ by some gain factor g is equivalent to adding an input impulse with amplitude $u(n) = (g-1)|z(n)| = (g-1)x(n)$ to $x(n)$ at that time. Therefore, another approach to achieving smooth filter restriking is to delay them until the next positive-going zero-crossing in the output signal $y(n)$.

3.5 Numerical Considerations

Since the difference equations (12–13) are based simply on the multiplication of two complex numbers, their extrapolation is guaranteed to be stable for any value of f provided the computer arithmetic is perfect. Round-off errors in computer arithmetic will limit the maximum value of τ, the accuracy of the resonant frequency, and the stability of the extrapolation in ways we have not studied in detail. Using 64 bit floating point arithmetic seems to be entirely adequate for musical purposes. For lower precision implementations, the alternate forms described in the following two subsections may be considered.

3.5.1 Modified Coupled Form Resonator

Robust behavior in fixed-point implementations using short word lengths, including perfect stability in the undamped case, can be obtained by replacing the complex multiply in Equations (12–13) with a damped version of the so-called "modified coupled form" (MCF) sine oscillator [2]:

$$x(n+1) = r_1[x(n) - \epsilon y(n)] + b_1 u(n) \quad (27)$$

$$y(n+1) = r_1[\epsilon x(n+1) + y(n)] + b_2 u(n) \quad (28)$$

where $\epsilon = 2\sin(\theta_1/2)$, and nominally $b_1 = 1$ and $b_2 = 0$. This recursion is highly insensitive to round-off error. When excited by an impulse ($u(n) = \delta(n)$) with no damping ($r_1 = 1$), the signals $[x(n), y(n)]$, viewed as coordinates in the complex plane, follow a fixed *elliptical orbit* over time. As $\epsilon \to 0$, the ellipse approaches a perfect circle. The complex multiply algorithm of Equations (12–13), on the other hand, generates an exact *circle* for all frequencies in the absence of quantization errors.[3] The component sinusoids $x(n)$ and $y(n)$ remain samples of pure sinusoids at the same frequency, as in the complex multiply case. The elliptical orbit is due only to the components having a relative phase difference other than 90 degrees. The relative phase approaches 90 degrees as the oscillation frequency (pole angle) approaches zero.

[3] In the field of *computer graphics*, the MCF has been called the "magic circle" algorithm, because it can be used to recursively draw "circles" on the screen that always close on themselves, even in low-precision arithmetic. While the drawing may look like a circle (when sufficiently many points are computed along its circumference), it is really an ellipse.

3.5.2 Digital Waveguide Resonator

Another attractive second-order filter structure, based on the "digital waveguide oscillator" [1], is the *digital waveguide resonator* (DWR):

$$\begin{aligned} x'(n) &= gx(n) && (29)\\ v(n) &= c'[x'(n) + y(n)] && (30)\\ x(n+1) &= v(n) - y(n) + \tilde{b}_1 u(n) && (31)\\ y(n+1) &= x'(n) + v(n) + b_2 u(n) && (32) \end{aligned}$$

where[4]

$$\begin{aligned} g &= r_1^2 && (33)\\ \tilde{b}_1 &= b_1\sqrt{\frac{1-c'}{1+c'}} && (34)\\ c' &= \sqrt{\frac{1}{1+\frac{\tan^2(\theta)(1+g)^2+(1-g)^2}{4g}}} && (35)\\ &\approx 1 - \frac{\tan^2(\theta)(1+g)^2 + (1-g)^2}{8g}. && (36) \end{aligned}$$

Note that $c' = \cos(\theta_1)$ when $g = 1$ (undamped case). Like the MCF, the DWR produces steady sinusoidal oscillations indefinitely, even for short word lengths. This happens because the resonance tuning is controlled by only one coefficient, when the damping goes to zero. As would be the case for any oscillator structure which is controlled by one coefficient affecting tuning only, quantization of that coefficient can only affect tuning as well. Unlike the MCF and complex multiply, which require four multiplies per sample, the DWR requires only two multiplies per sample. Moreover, when the decay is set to $\tau = \infty$ ($r_1 = 1$), one of the multiplies in the DWR disappears, leaving only *one* multiply per sample for sinusoidal oscillation. As a result, the DWR appears to be best suited for VLSI implementation. As an added bonus, the x and y outputs of the DWR are in exact phase quadrature, like the complex-multiply case considered first above. Note, however, that the choice of input (to either the $x(n)$ or $y(n)$ state variables) results different amplitude scaling.

Figure 1 shows an overlay of initial impulse responses for the three resonators discussed above. The decay factor was set to $r_1 = 0.99$, and the output of each multiplication was quantized to 16 bits, as were all coefficients. The three waveforms sound and look identical. (There *are* small differences, however, which can be seen by plotting the differences of pairs of waveforms.)

Figure 2 shows the same impulse-response overlay but with $r_1 = 1$ and only 4 significant bits in the coefficients and signals. The complex multiply oscillator can be seen to decay toward zero due to coefficient quantization ($x_1^2 + y_1^2 < 1$). The MCF and DWR remain steady at their initial amplitude. All three suffer some amount of tuning perturbation.

4 Program to Produce and Filter Sound Waves

To experiment with the timbres that can be synthesized with our high Q filters, we wrote a test program, `filter.cpp`. The program runs on an 800mHz PC running a Linux operating system. The program can

[4] `http://ccrma.stanford.edu/~jos/waveguide/Digital_Waveguide_Oscillator.html`

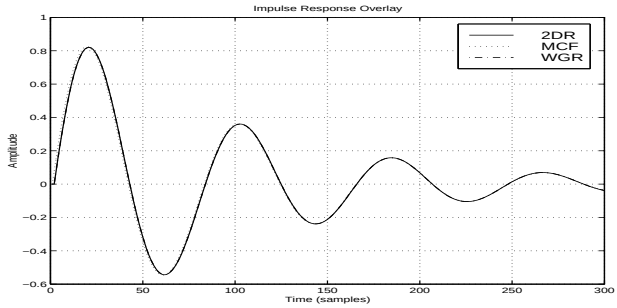

Figure 1: *Overlay of three resonator impulse responses, with* $\boldsymbol{B}^T = [1, 0]$ *and* $\boldsymbol{C}^T = [0, 1]$*, for the (1) complex-multiply resonator (labeled "2DR" for "2D rotation"), (2) modified coupled form (MCF), and (3) second-order digital waveguide resonator (DWR).*

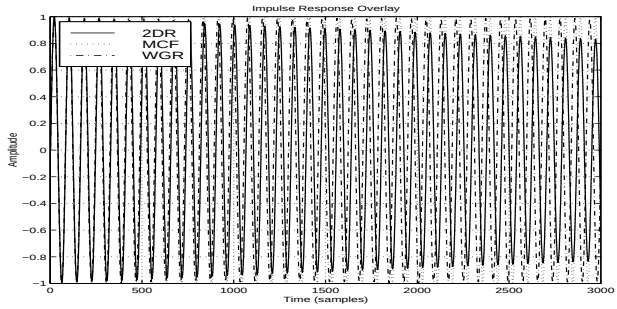

Figure 2: *Overlay of three resonator impulse responses, as in Fig. 1, but with* $r_1 = 1$ *and quantization of coefficients and signals to 4 significant bits.*

- Compute up to 200 filters whose outputs are summed and sent to a d-to-a converter (16bit, monaural, 44.1kHz) running as a real-time synthesis process.
- Accept impulse inputs to blocks of filters to generate homogeneous filter responses.
- Accept samples from a sound wave put through an a-to-d converter (16bit, monaural, 44.1kHz) running as a real-time effects processor.
- Accept samples from `.wav` files in the computer hard disk as input.
- Dynamically change the resonant frequencies and damping of blocks of filters in response to either computer keyboard key presses or midi input commands from a midi keyboard.
- Display either waveforms or envelopes of various sampled waves in the computer program

Computations are done in double precision floating point arithmetic. The program has not been carefully optimized. Only about 30 percent of the computer power seems to run all 200 filters.

Tests where parametric changes (both frequencies and decay times) of blocks of filters were suddenly changed showed that neither visible nor audible transients were produced by the parametric changes. Thus it would appear that both frequency and decay can be used as rapidly changing time-varying parameters to form "compositional lines" in compositions.

In the presentation of this paper at the SMAC 03 meeting, we will demonstrate various timbres and effects that can be produced by the filters.

5 Conclusions

Although no significant compositions employing our high Q parametrically well behaved two pole filters have yet been written, we believe the filters will provide an important new resource for composers, particularly for generating new timbres.

6 Sound Examples

Sound examples from the accompanying CD-ROM are available on-line:

`http://ccrma.stanford.edu/~jos/smac03maxjos/SoundExamples.html`

References

[1] J. O. Smith and P. R. Cook, "The second-order digital waveguide oscillator," *Proceedings of the 1992 International Computer Music Conference, San Jose*. 1992, pp. 150–153, Comp. Mus. Assoc., `http://ccrma.stanford.edu/~jos/wgo/`.

[2] J. W. Gordon and J. O. Smith, "A sine generation algorithm for VLSI applications," in *Proceedings of the 1985 International Computer Music Conference, Vancouver*. 1985, CMA.