Subtractive Synthesis without Filters

John Lazzaro and John Wawrzynek
Computer Science Division
UC Berkeley
lazzaro@cs.berkeley.edu, johnw@cs.berkeley.edu

1. Introduction

The earliest commercially successful electronic music synthesizer keyboards used analog circuits to implement subtractive sound synthesis. These monophonic instruments, produced by manufacturers such as Moog, Arp, Emu, and Roland in the 1970's, produce signature sounds that remain musically useful to this day.

One approach to modeling the voices of these analog instruments on a digital computers is to emulate the signal processing performed by each oscillator, filter, and amplifier of the original instrument. Readers interested in taking this approach may consult other chapters in this book, which describe efficient implementations for signal processing building blocks.

In this chapter, we take a different approach. In Section 2, we analyze the basics of analog subtractive synthesis, using the simple patch of a square-wave oscillator processed by a low-pass filter with a dynamically variable cutoff frequency. In Section 3, we show how a single mathematical function can model both the oscillator and filter together, and describe an efficient formation for this function (Moorer, 1976; Winhamand & Steiglitz, 1970). Section 4 shows additional applications of this technique.

2. Subtractive Synthesis

The block diagram in Figure 1 shows the spectral processing that underlies subtractive synthesis. In this diagram, an oscillator with a fixed square waveform shape is coupled to a low pass filter. Two properties of the system are open to dynamic control: the pitch of the oscillator and the cutoff frequency of the filter. Not shown are post-processing blocks to shape the amplitude envelope of the filter output.

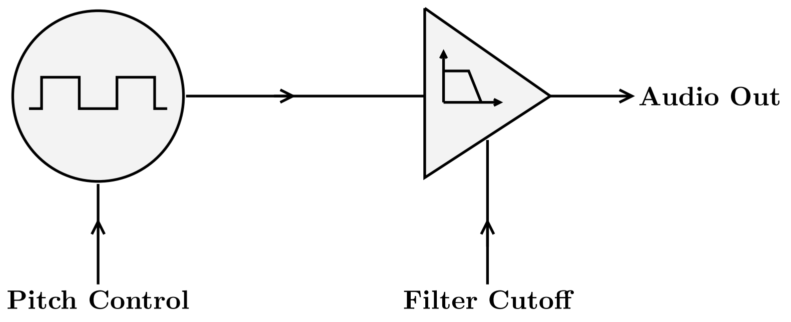

Figure 1. Block diagram for simple subtractive synthesis: a square wave oscillator processed by a low pass filter whose cutoff frequency is under dynamic control.

Most of the 70's-era analog synthesizers are monophonic keyboard instruments. The keyboard input section of these instruments generates an analog voltage that codes the position of the currently depressed key. In a typical setup, this signal provides the baseline oscillator pitch and filter cutoff control signals, so that the spectral signature of the sound scales across the keyboard.

To animate this static sound, oscillator pitch and filter frequency are dynamically varied around this baseline. Dynamic variation may be applied via specialized circuits (low-frequency oscillators, envelope generators triggered by a note depression) and by manual controllers (wheels, paddles, or joysticks that generate a continuous signal).

A square wave oscillator generates a signal that can be described by this function, expressed as an infinite series:

$$\text{square}(p) = \sin(2\pi p) + \frac{1}{3}\sin(3 \cdot 2\pi p) + \frac{1}{5}\sin(5 \cdot 2\pi p) + \ldots. \tag{1}$$

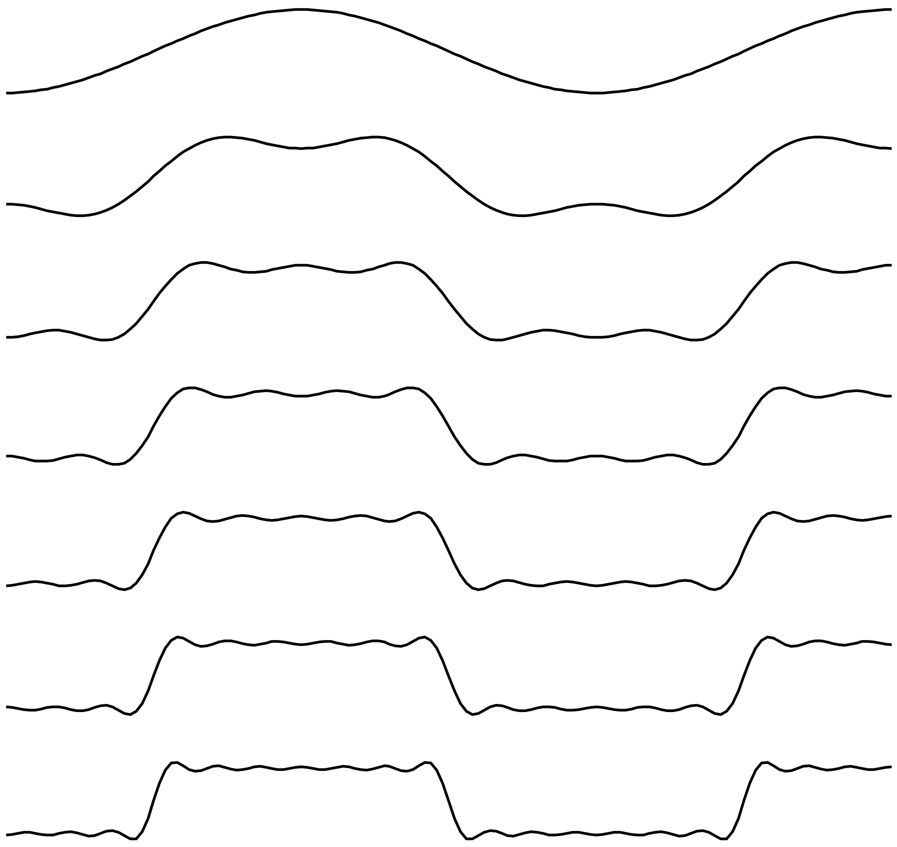

Figure 2. Plots show how the series terms in Equation 1 combine to produce a square wave. The top plot shows the first term of Equation 1, the second plot from the top shows the sum of the first 2 terms of Equation 1, the third plot from the top shows the sum of the first 3 terms of Equation 1, etc.

We can write this function more compactly using summation notation:

$$\text{square}(p) = \sum_{k=0}^{\infty} \frac{1}{2k+1} \sin(2\pi(2k+1)p). \tag{2}$$

The function takes the *phase pointer* p as an argument. To produce one complete cycle of the square wave, p is swept from 0.0 to 1.0. To provide intuition that this series really does produce a square wave, we show in Figure 2 a set of waveforms, showing the result of summing the first two terms, first three terms, first four terms, etc. As we would expect from a sound with a clear pitch, a square wave produces a *harmonic* series – i.e. all sinusoidal frequencies are integral multiples of the lowest frequency component.

This series formulation lets us plot the spectrum of the square wave directly, by plotting the coefficients in front of each sine component. Figure 3a shows this spectral plot in decibel (dB) units, which correspond to how humans perceive relative amplitudes (a 10dB amplitude increase approximately doubles the perceived loudness at a given frequency).

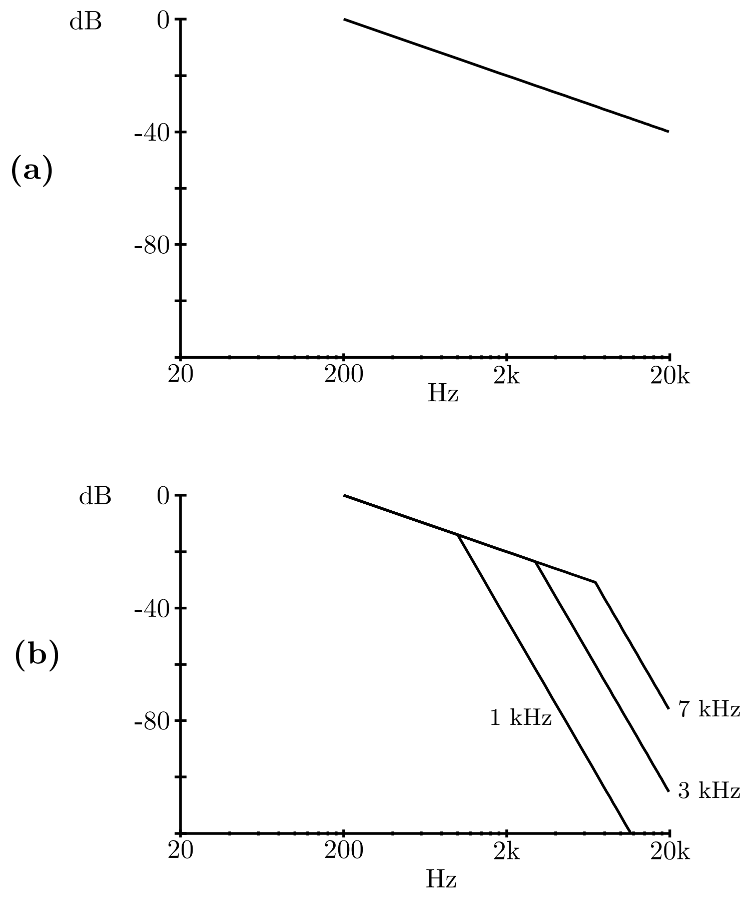

Figure 3. **(a)** Spectral shape of a 200 Hz square wave. **(a)** Spectral shape of a 200 Hz square wave, filtered by 24dB/octave low pass filters with cutoff frequencies of 1 kHz, 3 kHz, and 7 kHz as marked.

In Figure 3b, we plot the spectrum for the complete system shown in Figure 1 – a square wave oscillator processed by low pass filter – for several different values of the filter cutoff frequency. Dynamic variation of cutoff frequency acts to interpolate the spectrum between the shown patterns.

As we stated in the introduction, one approach to implementing subtractive synthesis on digital computers is to emulate the signal processing performed by each component of the instrument. For the simple system in Figure 1, independent modules for alias-free square wave generation (Stilson & Smith, 1996) and dynamic filters would be designed and interconnected.

In this chapter, however, we consider the problem at a higher level of abstraction. Three properties make the system shown in Figure 1 a good framework for analog musical instruments:

[1] The type of spectral variation shown in Figure 3b is musically interesting.

[2] This spectral variation occurs by varying a single parameter in a monotonic way.

[3] The method efficiently maps into analog circuits.

In the next section, we a present digital synthesis method that maintains properties [1] and [2] above, while mapping efficiently onto the digital abstraction.

3. Subtractive Synthesis without Filters

Our goal in this section is to find a single simple algorithm to compute both the oscillator and filter blocks in Figure 1. One way to approach this problem is to modify the square-wave generation function shown in Equation 2, so that it incorporates low pass filtering. The function shown below is an example of this method; this function is a simplified form of a library function contained in the MPEG 4 Structured Audio signal processing language (Scheirer and Vercoe, 1999; ISO, 1999).

$$B(p,a) = \frac{1-|a|}{1-|a^{H+1}|} \sum_{k=0}^{H} a^k \cos(2\pi(k+1)p). \tag{3}$$

Like Equation 2, this series is a harmonic series, and it uses a phase pointer p to plot out one complete cycle of the waveform over the range $0.0 \leq p \leq 1.0$. The key difference between this function and Equation 2 is the new parameter a, which has a role similar to the filter cutoff frequency in the system shown in Figure 1.

Examining Equation 3, we see that a is used to implement parametric scaling: each sine component $2\pi(k+1)p$ is scaled by the value a^k. Figure 4a shows the low pass spectral shapes that this equation generates, for values of a in the range $0.0 < a < 1.0$.

Equation 3 is a finite series summation. Using the series summation techniques described in (Moore, 1990; Moorer, 1976; Winham & Steiglitz, 1970) it is possible to derive an exact closed form solution for Equation 3:

$$B(p,a) = \frac{(1 - a\cos(\theta))(S_1\cos(\theta) - S_2\cos(N\theta)) - a\sin(\theta)(S_2\sin(N\theta) - S_1\sin(\theta))}{1 - 2a\cos(\theta) + a^2} \tag{4}$$

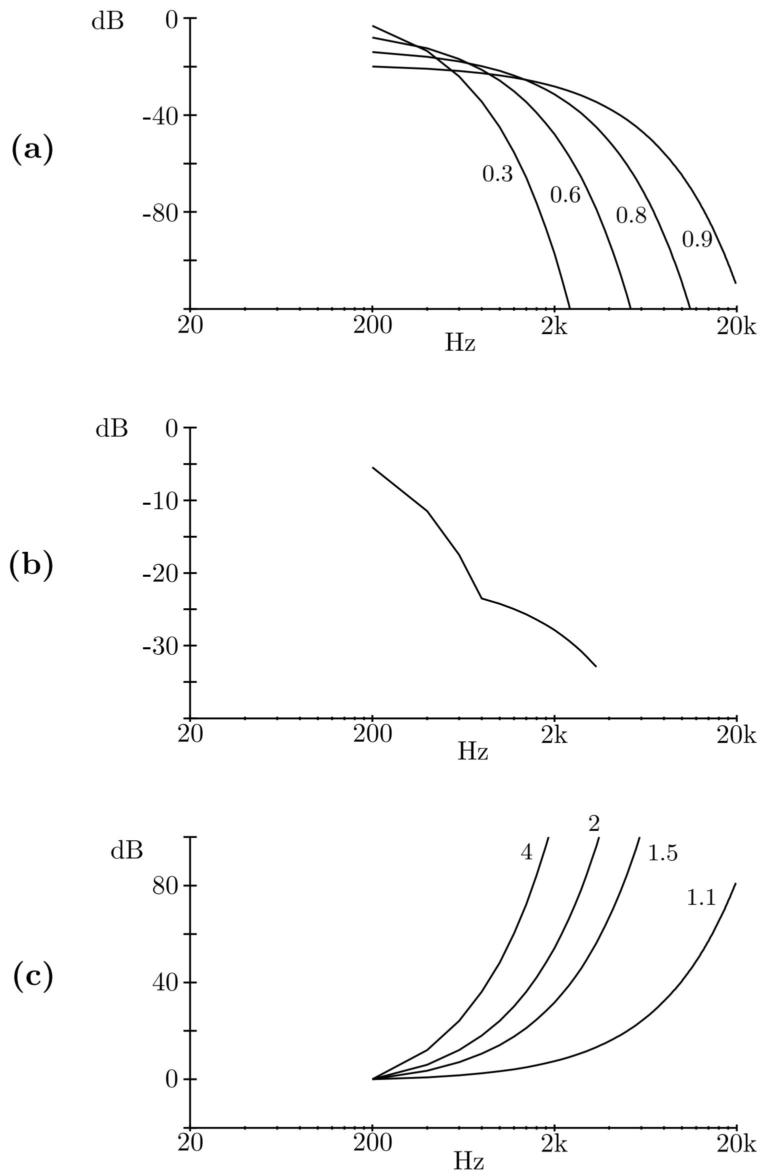

Figure 4. **(a)** Spectral shapes for a 200 Hz waveform produced using Equation 3, marked with its a value. **(b)** Spectral shape for a 200 Hz waveform produced using Equation 7. **(c)** Spectral shapes for a 200 Hz waveform produced using Equation 3, for values of $a > 1$ (as marked), with spectra shifted to start at 0 dB.

Where:

$$\theta = 2\pi p$$

$$N = H + 2$$

$$S_1 = \frac{1 - |a|}{1 - |a^{H+1}|}$$

$$S_2 = \frac{1 - |a|}{(1/a^{H+1}) - (|a^{H+1}|/a^{H+1})}$$

By using Equation 4, we can generate a waveform value using an amount of computation that is independent of the number of sinusoidal components. If a is a constant, a waveform data point calculation requires a phase pointer advance, two $[\sin(x), \cos(x)]$ calculations, nine multiplies, 5 additions, and one divide.

We can use Equation 4 to directly compute waveform data points for a subtractive synthesis system. Several issues arise in a practical implementation, which we discuss below.

Aliasing

In a typical real-time computer music application, audio output samples are sent to a digital-to-analog converter at a fixed audio sample rate (for example the compact-disc sampling rate of 44,100 Hz). To avoid unpleasant audio artifacts due to aliasing (see chapter XXX for more details on aliasing), the output samples should not have frequency components higher than one half the audio sample rate (for the compact disc sampling rate, frequency components above 22,050 Hz should not be generated).

Examining Equation 3, we see that the highest frequency component of the generated waveform is $\cos(2\pi(H+1)p)$. In a typical computer music application, we are sweeping the phase pointer p over the range $0.0 < a < 1.0$, at a bounded rate. For example, in a MIDI application, a maximum frequency for a note may be estimated by considering the MIDI note number value and the maximum depth of all pitch modulations. Given this maximum frequency, and the output sampling rate of the system, we can choose the integral value of H that ensures aliasing can not occur.

Numerical Issues

To produce clean waveforms over a range of a values, Equation 4 must be computed using at least single precision (32 bit) IEEE floating point arithmetic or equivalent. In addition, values of a around unity should be handled with care, as the denominator evaluates to zero for $a = 1$, causing a divide-by-zero error. A simple solution is to detect all a values within an empirically determined $a = 1 \pm \epsilon$ "dangerous regime," and replace these values with the nearest safe value.

Trigonometric Computations

The simplest (and slowest) way to compute the sine and cosine functions in Equation 4 is to use the math library functions. A much faster alternative is to use a table lookup approach. We have implemented a table-driven system that uses a single 2560 element floating-point table to represent one and one quarter cycles of the sinusoid. We directly address into this table for sine evaluations, and apply a quarter-cycle offset for cosine evaluations; no interpolation is done.

The lack of interpolation only results in significant artifacts when the denominator of Equation 4 is very close to zero. We have found it most efficient to check for this rare condition, and recompute the values using the library sine and cosine functions.

4. Cascading Multiple Functions

In more advanced synthesis applications, we may wish to cascade several copies of the function shown in Equation 3, to build up a complex spectral shape. The function shown below simplifies this task, by adding a new parameter L to specify the lowest frequency component in the signal.

$$B(p,a) = \frac{1-|a|}{1-|a^{H+1}|} \sum_{k=L}^{L+H} a^{k-L}\cos(2\pi(k+1)p). \tag{5}$$

Like Equation 3, this function can also be expressed in closed form, as:

$$B(p,a) = \frac{(1-a\cos(\theta))(S_1\cos(Q\theta) - S_2\cos(N\theta)) - a\sin(\theta)(S_2\sin(N\theta) - S_1\sin(Q\theta))}{1-2a\cos(\theta)+a^2} \tag{6}$$

Where:

$$\theta = 2\pi p$$

$$N = L + H + 2$$

$$Q = L + 1$$

$$S_1 = \frac{1-|a|}{1-|a^{H+1}|}$$

$$S_2 = \frac{1-|a|}{(1/a^{H+1}) - (|a^{H+1}|/a^{H+1})}$$

Note that if L has a value of zero, Equation 6 reduces to Equation 4, as expected. When notating a cascade of functions, we use the notation $B(p,H,L,a)$. For example, the equation:

$$f(p) = B(p, 3, 0, 0.5) + 0.5073\, B(p, 12, 4, 0.92) \quad (7)$$

describes a spectrum with a spectrum whose partials fall in amplitude quickly for lower frequencies, and more gradually for higher frequencies. The weighting value 0.5073 was chosen to match the spectral amplitudes at the crossover point. Figure 4b shows the spectrum produced by this function.

In a cascaded configuration, values $a > 1$ may be useful to generate spectral sections of increasing slope. Figure 4c shows the spectral shapes generates by values of a in this regime.

Acknowledgements

Thanks to Eric Scheirer, Robin Davies, and Perry Cook.

References

ISO (International Standards Organization) (1999). International Standard ISO 14496 (MPEG-4), Part 3 (Audio), Subpart 5 (Structured Audio). Geneva, CH: ISO.

Moore, R. F. (1990). *Elements of Computer Music.* Prentice-Hall: Englewood Cliffs, New Jersey, pp. 271-273.

Moorer, J. A. (1976). "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulae," Journal of the Audio Engineering Society 24:717-727.

Scheirer, E. D., and Vercoe, B. L. (1999). "SAOL: The MPEG-4 Structured Audio Orchestra Language." *Computer Music Journal* 23:2 (Summer), pp 31-51.

Stilson, T. and Smith, J. O. (1996). "Alias-Free Digital Synthesis of Classic Analog Waveforms," *1996 International Computer Music Conference, Hong Kong.*

Winham, G., and Steiglitz, K. (1970). "Input Generators for Digital Sound Synthesis," Journal of the Acoustical Society of America 47 2:ii, pp. 665-666.

Annotated References

The main focus of the chapter is the efficient implementation of Equation 3, which is similar to the defining equation of the **buzz()** core opcode in MPEG 4 Structured Audio. To learn more about Structured Audio, consult this easy-to-read introductory article:

Scheirer, E. D., and Vercoe, B. L. (1999). "SAOL: The MPEG-4 Structured Audio Orchestra Language." *Computer Music Journal* 23:2 (Summer), pp 31-51.

and this comprehensive MPEG document:

ISO (International Standards Organization) (1999). International Standard ISO 14496 (MPEG-4), Part 3 (Audio), Subpart 5 (Structured Audio). Geneva, CH: ISO.

The chapter states that Equation 3 above can be converted into the closed-form solution shown in Equation 4, by using a series summation technique. Several references will teach you more about this series summation. An short introduction can be found in R. F. Moore's computer music textbook:

Moore, R. F. (1990). *Elements of Computer Music.* Prentice-Hall: Englewood Cliffs, New Jersey, pp. 271-273.

For more information, readers should look up these research publications which introduced the technique:

Winham, G., and Steiglitz, K. (1970). "Input Generators for Digital Sound Synthesis," *Journal of the Acoustical Society of America* 47 2:ii, pp. 665-666.

Moorer, J. A. (1976). "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulae," *Journal of the Audio Engineering Society* 24:717-727.

To compare this technique for pulse generation with other approaches, consult this review article:

Stilson, T. and Smith, J. O. (1996). "Alias-Free Digital Synthesis of Classic Analog Waveforms," *1996 International Computer Music Conference, Hong Kong.*