9 Sound Synthesis

Mathematical science . . . has these divisions: arithmetic, music, geometry, astronomy. Arithmetic is the discipline of absolute numerable quantity. Music is the discipline which treats of numbers in their relation to those things which are found in sound.
—Cassiodorus

9.1 Forms of Synthesis

Fourier synthesis can be used to create any periodic vibration (see chapter 3). But Fourier methods are only one of an essentially limitless number of techniques that can be used to synthesize sounds.

9.1.1 Linearity and Synthesis

Linear synthesis techniques can generally be used to reproduce a sound that is identical to the original. The Fourier transform, for example, can be used to reproduce any periodic waveform. This is part of the reason we call it a transform: we can analyze a signal into its transformed state and then use its inverse transform to reproduce the original. Nonlinear techniques generally provide no way to reproduce a sound that is identical to an original, but they may have other compelling advantages, such as being economical to calculate or intuitive to use.

9.1.2 Linear Synthesis

The criteria of linear systems are superposition and proportionality, described here with an emphasis on synthesis.

Superposition Air and water are linear media (at least for the strength of signals we are discussing) because waves superimpose without distortion. For two signals x_1 and x_2, if $F(x_1) = y_1$, and $F(x_2) = y_2$, and if $F(x_1 + x_2) = y_1 + y_2$, then function F is linear (if it also meets the proportionality criterion); otherwise it is nonlinear.

$F(x) = c \cdot X$

Proportionality In air and water, little waves pass through big ones, and vice versa, without being modified by the encounter. Linear systems are independent of amplitude. If $F(x_1 + x_2) = y_1 + y_2$, and $F(ax_1 + bx_2) = ay_1 + by_2$, then function F is linear (if it also meets the superposition

E.G. $F(x) = a x$ [illegible] $F(x_1) = ax_1$ [illegible] $F(x_1 + x_2)$

$F(x_2) = ax_2$ [illegible] $= a(x_1 + x_2)$

criterion); otherwise it is not. Examples of nonlinear media include photographic film and magnetic tape, both of which can saturate if the amplitudes of signals being recorded on them are cumulatively too strong.

9.1.3 Overview of Linear Synthesis Types

Linear transforms are characterized by adding or subtracting weighted basis functions of some kind. In the case of Fourier analysis and synthesis, the basis functions are the family of sinusoids. A non-Fourier linear synthesis technique called *Walsh-Hadamard synthesis* has basis functions that are square waves. The wavelet transform (see chapter 10) has a variety of nonsinusoidal basis functions.

We can synthesize a periodic function corresponding to an arbitrary spectrum via the inverse Fourier transform. Cycling the resulting waveform repeatedly is equivalent to the original infinite periodic function. This is the basic idea of *waveform synthesis* (see section 9.2.5).

Although the Fourier transform is mathematically valid only when applied over all time, it can be adapted to analyze and reproduce sounds that change through time. *Additive synthesis* is an extension of the Fourier transform to signals that can change over time (see figure 9.9). The technique can be implemented, for example, by driving a bank of oscillators with time domain functions representing the frequencies and amplitudes of the sound's components. Since in this case the frequencies need not be harmonics, this technique can also generate inharmonic spectra. The general case of additive synthesis is discussed in chapter 10. Even a simple mixing console can be thought of as a kind of additive synthesizer, in the sense that it superposes and scales its input signals through time.

As the name suggests, *subtractive synthesis* removes energy from a spectrum by filtering. Like additive synthesis, it can be applied to signals that change through time. Subtractive synthesis is discussed under *linear predictive coding* (LPC) (see section 9.5.2).

The advantage of linear synthesis systems is that, since they are based on a transform, we can use them to both analyze and reproduce a sound. But this is generally not a musically interesting thing to do—why not just use the original sound? For musical applications, we generally wish to create effects not otherwise obtainable. We can manipulate the analysis data of linear systems to achieve some very interesting effects, and doing so is usually fairly intuitive because linear techniques involve only superposition and proportionality.

The liability of many linear synthesis systems is that the amount of analysis data can be dauntingly large, requiring huge amounts of calculation or data storage or both in order to produce realistic-sounding synthesis. Nonlinear techniques typically are much more economical, but they are also less general.

9.1.4 Nonlinear Synthesis Types

Any technique that does not meet the superposition and proportionality criteria is a nonlinear technique. This book covers only a small fraction of the amazing variety of such techniques, but it hits many of the high points.

The primary limitation of nonlinear synthesis techniques is that they have no inverse transform, so there is no way to use them to exactly reproduce an original sound. Although no direct analysis is possible, that doesn't mean we can't predict the kinds of sounds we'll get from a nonlinear technique; we just won't necessarily be able to produce a particular sound exactly. It may not be possible for a nonlinear technique to produce a particular type of sound regardless of the parameters used.

Nonlinear systems are generally more computationally efficient than linear synthesis forms. Some, such as frequency modulation, model instrumental timbres very well with few parameters and little computation.

A central element of many nonlinear techniques is *modulation,* which is just a latinate word meaning *change.* In sound synthesis, it has the particular meaning of applying a time-varying change to a signal.

The aspects of signals that are exploited by nonlinear synthesis include

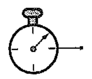
- *Amplitude,* where, for example, a signal is saturated or clipped, or where there is uneven amplitude response to different frequencies (filtering).
- *Frequency,* where new frequencies are produced in response to an input signal.
- *Phase,* where the phase of the output signal is not a linear function of the input phase.

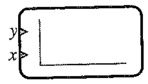
Musically, many nonlinear synthesis systems have a quixotic character that can sometimes be counterintuitive, whereas linear systems tend to offer fewer surprises. Since composition is the art of controlled expectation, nonlinear techniques are very useful in the composer's tool kit. However, they are not a panacea. If overused, the inner structure of nonlinear techniques can become horribly clichéd.

9.2 A Graphical Patch Language for Synthesis

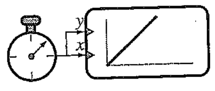
It is worthwhile to have a way to construct sound-generating modules to illustrate the synthesis techniques. We need three elements: a way to keep time, a way to pass signals around, and a way to transform signals.

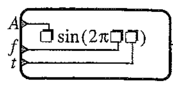
Here's a clock that produces a monotonically increasing value corresponding to the flow of time. We need a way to observe the output of the clock. The function plotter shown here takes two input signals: the x input moves a pen horizontally, and the y input moves the pen vertically.





Connecting the clock to both inputs of the plotter produces a diagonal line. Since the clock's values are applied to both inputs, it moves the pen an equal distance vertically and horizontally through time, producing a ramp at a 45° angle (the identity function).





We need a way to transform inputs to create outputs. The sine wave oscillator shown here implements the function $A\sin 2\pi ft$. The inputs on the left side, A, f, and t, are connected to the indicated terms of the formula. The output appears at the right side and can be the input to another module.

If we wire the sine wave oscillator with the clock and plotter, and supply parameters for A, f, and t, we have the setup shown in figure 9.1. Note that the amplitude input is associated with variable A and frequency with variable f. If left unspecified, these variables indicate parameters we can set any way we like, to obtain, in this case, any desired amplitude and frequency. If an input should receive a particular value, it will be shown associated with a constant.

Let's call a configuration of such modules a *patch*. This patch scales time t by $2\pi f$. The sine of the result is then scaled by A. The time value is also applied to the plotter and moves the pen horizontally. The value of the sine result drives the pen vertically.

In order to represent increasingly complex modules, we can use the standard rules of mathematical equality to define new functions to encapsulate existing definitions. Taking the oscillator, for example, if we let

$$m \cdot g(t) = A\sin 2\pi ft,$$

we could interpret this graphically as shown in figure 9.2.

These diagrams can at times get cluttered with many lines connecting modules, so I occasionally use variables to temporarily hold a value produced in one part of a patch for reuse elsewhere. For example, the oscillator patch could have been drawn as in figure 9.3. The clock sets the value of the variable t, and t is referenced by other module inputs. Here, the variable t holds the output of the clock, and the inputs to the oscillator and plotter reference its value whenever

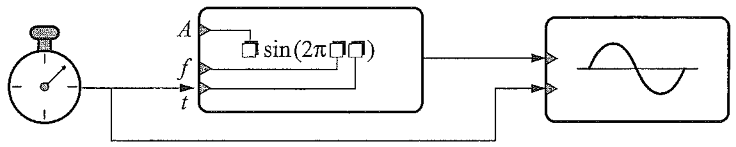


Figure 9.1
Simple oscillator patch.

$A \cdot \mathrm{SIN}(2\pi f \cdot N/N)$

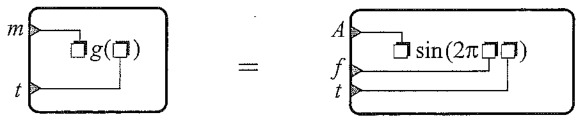


Figure 9.2
Encapsulating patches.

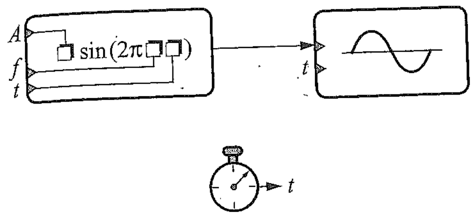


Figure 9.3
Patch using variables instead of interconnections.

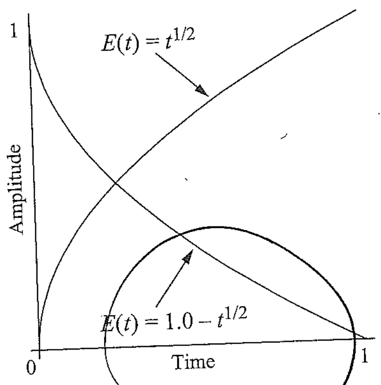


Figure 9.4
Envelope curves.

$t^{\frac{1}{2}} = \sqrt{t}$

OVER 1 SEC
GOES 0 → 1

THIS IS
ODD, NOT
WHAT REAL
SYSTEMS BEHAVE

they need an input. This comes in so handy at times that I may only indicate the variable t when I want to imply the clock signal.

9.2.1 Constructing a Simple Synthesis Instrument

This simple graphical patch language is flexible enough to allow representation of a wide range of synthesis and processing techniques. Let's create a tone that has a natural attack and decay, control over amplitude and frequency, and control over timbre. The oscillator provides a way to generate a particular frequency, but we need a way to control its amplitude through time.

Simple Envelope Generator Let's invent an envelope generator that has a simple exponential attack, variable-length steady state, and exponential decay. We want to be able to control the overall duration, the attack time, and the decay time individually. The function $y = E(t) = t^{1/n}$ produces an exponential curve that goes from 0 to 1 over the range $0 < t < 1$ seconds. The function $y = 1.0 - E(t)$ necessarily goes from 1 to 0 over the same interval. If $n = 1$, $E(t)$ is linear. The variable n specifies the time constant of the function; values of n farther away from 1.0 are more sharply curved (figure 9.4).

We can scale the function to an arbitrary time length by redefining the envelope function to be

$$y = E(t, d, n) = \left(\frac{t}{d}\right)^{1/n}, \qquad \textit{Envelope Kernel} \quad (9.1)$$

where d is the duration in seconds.

If we use one envelope function $E_a(t, d_1, n_1)$ for the attack and concatenate another, $E_d(t, d_2, n_2)$, to its end for the decay, we can use it, for example, to vary the instantaneous amplitude of an oscillator, creating the amplitude envelope shown in equation (9.2):

$$\text{env}(a, n_1, d, n_2, t) = \begin{cases} t < a, & (t/a)^{1/n_1}, \\ t \geq a, & 1.0 - [(t-a)/d]^{1/n_2}, \end{cases} \qquad \textit{Simple Envelope Generator} \quad (9.2)$$

where t is time, a is the attack time, d is the release time, and each n determines the steepness of its part of the envelope curve, with n_1 controlling the curve of the attack and n_2 controlling the curve of the decay. The higher the value of n, the more quickly the function changes. For example, figure 9.5 shows a function with attack time $a = 0.1$ s, release time $d = 0.6$ s, $n_1 = 2$, and $n_2 = 1.5$. Putting it all together, we can construct a simple tone generator with an amplitude envelope, as shown in figure 9.6 where the module env( ) is as defined in (9.2).

In order to realize a tone with this setup, we assign values to the variables and start the clock running at $t = 0$. For example, setting envelope values to those in the previous paragraph, and setting $f = 32$ Hz and $A = 1$ produces the waveform shown in the figure 9.7.[1]

Controlling Frequency If we make the system of representing pitch flexible enough, we can use this simple instrument to study the scale systems described in volume 1, chapter 3.

For equal-tempered pitches, recall volume 1, equation (3.4), repeated here:

$$f_{k,v} = f_R \cdot 2^{(v-4)+k/12},$$

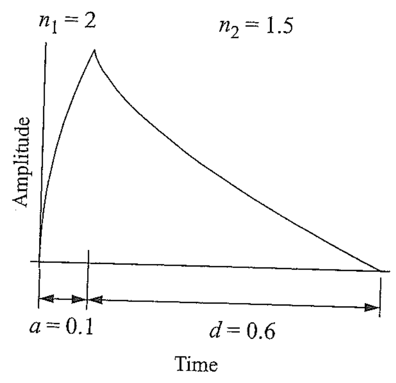


Figure 9.5
Simple envelope generator function.

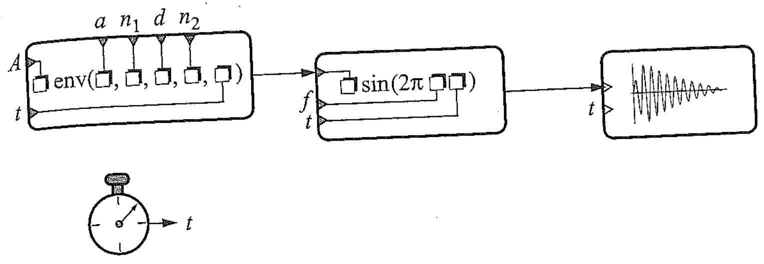


Figure 9.6
Simple synthesis instrument with amplitude envelope.

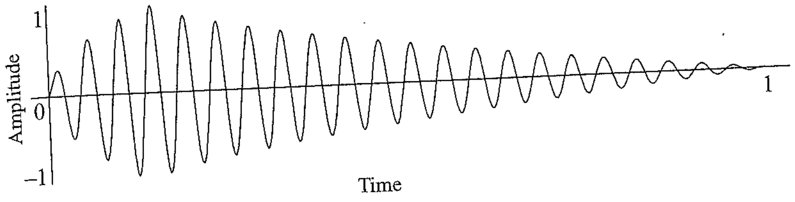


Figure 9.7
Tone produced by the synthesis instrument.

where k is an integer signifying one of the 12 pitch classes numbered 0 to 11, R is the reference frequency, such as A440, and v is the desired octave. If we construct symbols for each chromatic pitch, such as C♯4 = $f(1,4)$, we can set the frequency parameter f = C♯4 to achieve that pitch in the simple instrument shown in figure 9.6.

We can approach just intonation along the lines worked out at the end of volume 1, section 3.8. Recall volume 1, equation (3.11), reproduced here:

$$\mathrm{C}_\pi(v) = R \cdot \frac{16}{27} \cdot 2^{v-4}.$$

We constructed a set of symbols such as $\mathrm{F}_\pi(v) = \mathrm{C}_\pi(v) \cdot 4/3$, which provides the "πthagorean" pitch F_π in any octave v. We can set the frequency parameter $f = \mathrm{F}_\pi(v)$ to achieve Pythagorean pitch F_π in octave v using the simple synthesis instrument. We can also use the more elaborate constructions for pitch in MUSIMAT (see volume 1, appendix sections B.1 and B.2).

Adding Vibrato *Vibrato* is a periodic pitch modulation around a target pitch. Depending upon the instrument, the *vibrato rate* for musical instruments typically ranges from 1 to 7 Hz, and *vibrato depth* ranges from about one tenth of a semitone (100 cents) up to a maximum of about a minor third (think: Wagnerian soprano). Perhaps the simplest way to model vibrato is just to add the output of a slowly time-varying oscillator to the frequency parameter, and use the result as the frequency input of the oscillator generating the waveform. Call the oscillator that generates the

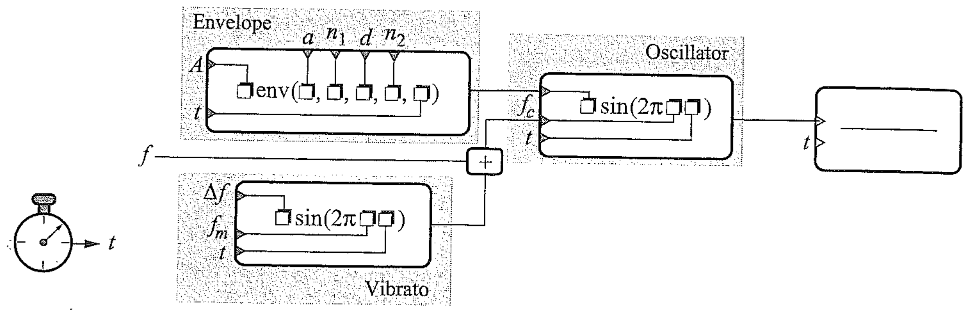


Figure 9.8
Synthesis instrument with vibrato and amplitude envelope.

waveform the *carrier oscillator,* and the oscillator that modulates the target frequency the *modulating oscillator.* The calculation for the frequency f_c of the carrier oscillator is

$$f_c = f + \Delta f \cdot \sin 2\pi f_m t, \qquad \textit{Vibrato} \quad (9.3)$$

where f is the target frequency, Δf is the vibrato depth, and f_m is the vibrato rate. Clearly, if $\Delta f = 0$, there is no vibrato, and the carrier frequency is just the target frequency f. But if $\Delta f \neq 0$, the carrier frequency will rise and fall at the rate of f_m. The patch with vibrato added is shown in figure 9.8.

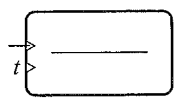
What value do we assign to Δf in order to achieve a desired depth of modulation? Recalling that pitch is logarithmic in frequency, if we want the vibrato depth to be a constant interval, such as a semitone, then Δf must grow with increasing pitch. Recall from volume 1, section 3.2.2, that the size of a tempered semitone ratio is about 1.06. That is, for some frequency f_n, the pitch f_{n+1} a semitone above is $f_{n+1} \cong 1.06 \cdot f_n$. If we set $\Delta f = 1.06 \cdot f$, where f is the target frequency, we actually get a vibrato depth of a whole tone, because the range of the sine function is ± 1.0, which provides a range of a semitone above and below the target pitch. So instead we set $\Delta f = 0.53 \cdot f$ to get a semitone of vibrato depth.

Of course, performers don't produce exactly sinusoidal vibrato. An improvement is to introduce some randomness into the vibrato to achieve a more naturalistic effect. This subject begins to verge into modeling of musical instrument performance. It's an enormous subject; I investigate just the basics.

9.2.2 Static Control over Timbre

With figure 9.8 we have achieved a synthesis instrument that has control over pitch, duration, amplitude, amplitude envelope, and vibrato. But its timbre is a simple sinusoid. In order to improve this, we turn now to ways of synthesizing timbre.

Timbre is actually the subject of key interest in synthesis because it largely determines the perceived quality of the synthetic tone. Musicians often have contradictory aims for timbre. On the



t frequency the *modu-*
tor is

Vibrato (9.3)

brato rate. Clearly, if
uency f. But if $\Delta f \neq 0$,
ibrato added is shown

ulation? Recalling that
tant interval, such as a
, section 3.2.2, that the
$_n$, the pitch f_{n+1} a semi-
frequency, we actually
s ±1.0, which provides
$\Delta f = 0.53 \cdot f$ to get a

vement is to introduce
subject begins to verge
t; I investigate just the

l over pitch, duration,
oid. In order to improve

ely determines the per-
ims for timbre. On the

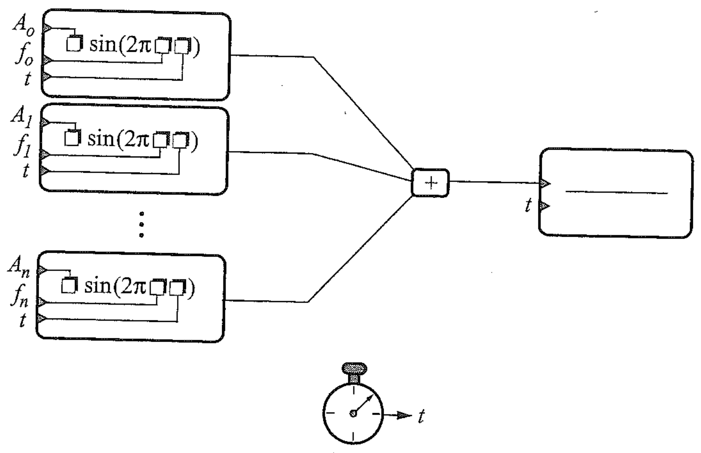


Figure 9.9
Additive synthesis.

one hand, having the most naturalistic possible sound is the goal of those who want to model standard acoustic instruments like pianos and clarinets. For orchestral composers, for example, this provides an inexpensive means to test out musical ideas. Unfortunately, realism in music synthesis is a kind of holy grail, often sought, seldom achieved.

On the other hand, sound synthesis can be used to extend the palette of timbres available to musicians beyond what can be produced by conventional instruments. Synthesis can create sounds that metamorphose from the familiar to the extraordinary. For example, linear predictive coding (LPC) synthesis can create hybrid sounds such as a talking flute. Synthesis can create unearthly sounds, such as the Shepard scale illusion (see volume 1, section 6.4.7).

A straightforward way to control timbre is to specify the spectrum of a sound by controlling the amplitudes and frequencies of a bank of oscillators, as in figure 9.9.

We could express this mathematically as follows:

$$f(t) = \frac{1}{\rho}\sum_{n=1}^{N} A_n \sin 2\pi f_n t, \qquad \textit{Oscillator Bank Synthesis} \quad (9.4)$$

where N is the number of components, starting with the fundamental. Their amplitudes and frequencies are given by vectors A_n and f_n, respectively, each of length N.[2] The sum of the amplitudes A_n is usually normalized so that the amplitude of $f(t)$ does not vary with the number and strengths of the components. This is done by setting

$$\rho = \sum A_n.$$

Oscillator bank synthesis is a generalization of Fourier synthesis (see section 3.1.1) because the amplitudes and frequencies of each oscillator can be any value whatever: we are not constrained to harmonic spectra and periodic waveforms. However, the spectra specified by A_n and f_n are static. Not many musical instruments produce entirely static spectra, so let's extend this to handle dynamic amplitudes and frequencies that can change through time.

9.2.3 Dynamic Oscillator Bank Synthesis

Dynamic frequency control allows realistic synthesis of music instrument tones, glissandos, and vibrato. Dynamic amplitude allows us to trace the spectral evolution of tones. All we have to do is make the amplitudes and frequencies of the spectral components be functions of time as well so that we have an array of amplitude functions $A_n(t)$ and frequency functions $f_n(t)$, as shown in equation (9.5).[3]

$$f(t) = \sum_{n=1}^{N} A_n(t)\sin[2\pi f_n(t)\cdot t]. \qquad \textit{Dynamic Oscillator Bank Synthesis} \quad (9.5)$$

Normalization, as in equation 9.4, can be added if desired.

9.2.4 Advantages and Disadvantages

There are two primary disadvantages of oscillator synthesis. First, many oscillators are typically required to synthesize a realistic sound. A single piano tone might have 30 harmonics containing significant energy. Multiply that by five fingers on both hands, and the number of oscillators required jumps to 300. If the damper pedal is held down, the number of oscillators would be in the thousands.

Second, this approach requires that the amplitudes and frequencies of each spectral component be specified in detail. Furthermore, if the frequencies and amplitudes of the components change through time, as is true with natural musical instruments, the amount of data required to control the oscillators can be many times greater than the resulting signal.[4]

However, this is the most general way of synthesizing sound. Since summation is linear, oscillator bank synthesis can be driven by Fourier analysis, producing sounds that have a dazzlingly realistic quality. Not only can we synthesize any sound we can analyze, we can also synthesize *any sound at all,* limited only by our ability to dream it up. In fact, the combination of a fast enough computer, a loudspeaker, and additive synthesis is perhaps the most general-purpose musical instrument ever created.

But too much generality can be paralyzing. If, for instance, a composer must make every decision at every level, the task can become overwhelming. Therefore, generally, the goal of sound synthesis is to generate sounds that are expressive without requiring extreme micromanagement to produce interesting results.

9.2.5 Waveform Synthesis

Waveform synthesis preserves some of the generality of oscillator bank synthesis at significantly less cost in complexity because the amplitudes and frequencies of the components are constants instead of functions of time. It can be expressed as follows:

$$y(t) = \frac{1}{\rho}\sum_{n=1}^{N} A_n \sin 2\pi nft, \qquad \textit{Waveform Synthesis}\ (9.6)$$

where A_n is an array of component amplitudes of length N. This technique is essentially identical to Fourier synthesis (see section 3.1.1) using oscillators, but it is less general than dynamic oscillator bank synthesis (section 9.2.3) because it is limited to harmonic spectra and periodic signals. The sum of the amplitudes A_n should be normalized if the amplitude of the output should not vary with the number and strengths of the components by setting

$$\rho = \sum A_n.$$

A discrete version of equation (9.6) known as wavetable synthesis works well within the hardware and software constraints of typical modern computer systems, and it is widely used to synthesize sound (see section 9.2.8).

9.2.6 Fourier Series Waveforms

Generalizing equation (9.6) to include all possible real harmonic waveforms, we have

$$f(t) = \sum_{k=0}^{\infty} a_k \cos(2\pi kt) + b_k \sin(2\pi kt), \qquad \textit{Real Fourier Series}\ (9.7)$$

where coefficients a_k and b_k scale the cosinusoidal and sinusoidal contributions, respectively, of harmonic k. Equation (9.7) does not have the frequency parameter f appearing in equation (9.6) because here we want to focus only on wave shape without concerning ourselves about frequency. We can always add frequency back in to realize the specified wave shape at a particular desired frequency. Term a_0 scales any energy at 0 Hz because $\cos k = 1$ when $k = 0$. (b_0 makes no contribution at 0 Hz because $\sin k = 0$ when $k = 0$.) The complex form of the Fourier series is

$$f(t) = \sum_{k=0}^{\infty} (a_k + ib_k)e^{i2\pi kt}. \qquad \textit{Complex Fourier Series}\ (9.8)$$

The behavior of this equation might be a little surprising. For instance, if we set all a_k and b_k to zero except $a_1 = 1$, equation (9.8) reduces to

$$e^{i2\pi t} = \cos 2\pi t + i \sin 2\pi t,$$

while equation (9.7) reduces to just $(\cos 2\pi t) + 0$.

9.2.7 Geometrical Waveforms

Some interesting specimens among the family of wave shapes are implied by equations (9.7) and (9.8). Following are some well-known shapes.

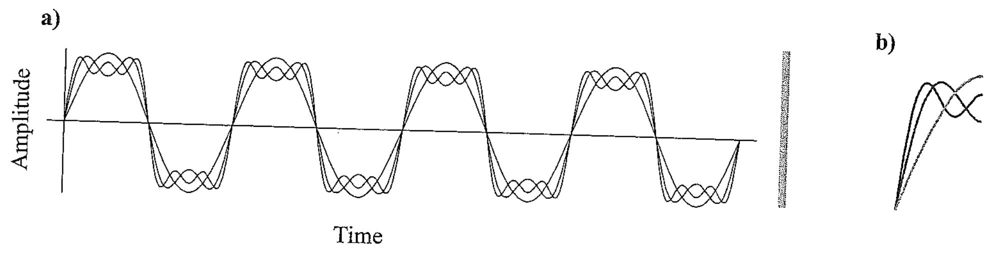


Figure 9.10
Square wave, Fourier series.

Square Waves If we only sum odd-numbered sine-phase harmonics and arrange their amplitudes to be odd reciprocals,

$$f(t) = \sum_{k=0}^{N-1} \frac{1}{2k+1} \sin[2\pi(2k+1)\cdot t], \qquad N > 0, \qquad \textit{Square Wave, Fourier Series} \quad (9.9)$$

then the series converges as $N \to \infty$ to a *square wave*. The series expansion of the first few terms of (9.9) is

$$f(t) = \sin 2\pi t + \frac{1}{3}\sin 2\pi 3t + \frac{1}{5}\sin 2\pi 5t + \frac{1}{7}\sin 2\pi 7t + \dots.$$

The first few waveforms in this sequence ($k = 0$, 1, and 2) are demonstrated in figure 9.10a. When $N = 1$, equation (9.9) produces a sine wave, and for $N > 1$, it produces the sum of a sine wave and odd harmonics. The following table shows the harmonics produced and their amplitudes for the first few values of k.

k	0	1	2	3	4	5	6	7	8
Amplitude	1	1/3	1/5	1/7	1/9	1/11	1/13	1/15	1/17
Frequency	DC	3	5	7	9	11	13	15	17

The spectrum of the square wave contains components that are odd harmonics of the fundamental. The amplitudes diminish with increasing harmonic number. In figure 9.10, note the slight overshoot and ringing that occurs at the ends of the vertical excursion of the waveforms. The effect, called the *Gibbs phenomenon* (or more colorfully, *Gibbs' horns*), is shown magnified in figure 9.10b. The horns indicate that Fourier series functions only approximate the discontinuous points of the non-band-limited square wave function defined in equation (9.10).[5]

The geometric square wave, also called the *non-band-limited square wave,* can be expressed as

$$f(t) = \begin{cases} 1, & 0 \le t < \pi, \\ -1, & \pi \le t < 2\pi, \end{cases} \qquad \textit{Square Wave, Non-Band-Limited} \quad (9.10)$$

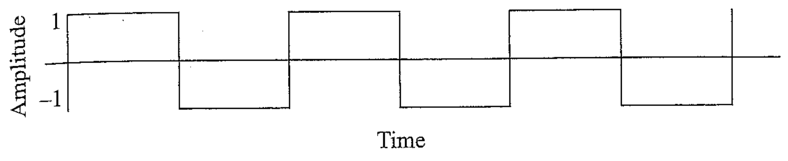


Figure 9.11
Non-band-limited square wave.

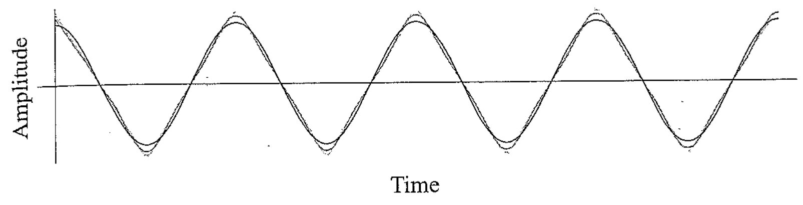


Figure 9.12
Triangular wave Fourier series.

as shown in figure 9.11. This is the same as equation (9.9) with $N = \infty$, so it has an infinite number of odd harmonics.

The non-band-limited square wave takes on only two states through time: −1 and +1; it is binary and discontinuous. This function has two discontinuities per period, where it transits instantaneously from above to below and vice versa.

The *Walsh-Hadamard transform* uses rectangular waves as its basis functions instead of using sinusoids or phasors (see appendix section A.7).

Triangular Waves By summing odd-numbered cosine harmonics and arranging their amplitudes to be squared odd reciprocals,

$$f(t) = \sum_{k=0}^{N-1} \frac{1}{(2k+1)^2} \cos[2\pi(2k+1)\cdot t], \qquad N > 0, \qquad \textit{Triangular Wave, Fourier Series} \quad (9.11)$$

the series converges as $N \to \infty$ to a *triangular wave,* as demonstrated in figure 9.12. The series expansion of the first few terms of equation (9.11) is

$$f(t) = \cos 2\pi t + \frac{1}{9}\cos 2\pi 3t + \frac{1}{25}\cos 2\pi 5t + \frac{1}{49}\cos 2\pi 7t + \cdots.$$

The following table shows the harmonics and amplitudes for the first few values of *k*.

k	0	1	2	3	4	5	6	7	8
Amplitude	1	1/9	1/25	1/49	1/81	1/121	1/169	1/225	1/289
Frequency	DC	3	5	7	9	11	13	15	17

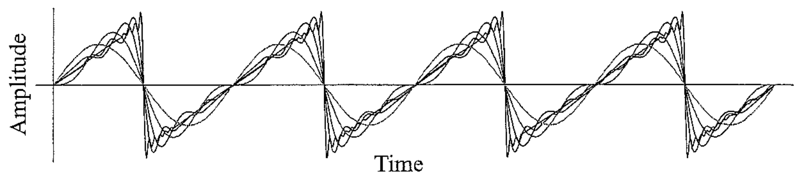


Figure 9.13
Sawtooth wave Fourier series.

The spectrum of the triangular wave contains components that are odd harmonics of the fundamental. The amplitudes diminish with the square of increasing harmonic number. Most of the energy in this signal is in the lowest harmonics.

The geometric triangular wave can be expressed as

$$f(t) = 1.0 - \frac{|t-\pi|}{\pi}. \qquad \textit{Triangular Wave, Non-Band-Limited} \quad (9.12)$$

The Fourier series convergence of the triangular wave is shown in figure 9.12.

Sawtooth Waves The sawtooth wave is created by summing all odd harmonics and subtracting all even harmonics, with amplitudes as the reciprocal of the harmonic number:

$$y(t) = \sum_{k=1}^{N} \frac{(-1)^{k+1}}{k} \sin 2\pi kt, \qquad N > 0, \qquad \textit{Sawtooth Wave, Fourier Series} \quad (9.13)$$

The series converges as $N \to \infty$ to a *sawtooth wave,* as demonstrated in figure 9.13. The series expansion of the first few terms of equation (9.13) is

$$f(t) = \sin 2\pi t - \frac{1}{2}\sin 2\pi 2t + \frac{1}{3}\sin 2\pi 3t - \frac{1}{4}\sin 2\pi 4t + \cdots.$$

The geometric sawtooth wave can be expressed as

$$y(t) = \begin{cases} t, & 0 \le t < \pi, \\ t - 2\pi, & \pi \le t < 2\pi. \end{cases} \qquad \textit{Sawtooth Wave, Non-Band-Limited} \quad (9.14)$$

Sum of Cosines Waves Summing equal-amplitude cosine harmonics,

$$t(t) = \frac{1}{N}\sum_{k=1}^{N} \cos 2\pi kt, \qquad N > 0, \qquad \textit{Sum of Cosines} \quad (9.15)$$

the series converges as $N \to \infty$ to an impulse train signal. The series expansion of the first few terms of equation (9.15) is

$$f(t) = \frac{1}{N}(\cos(2\pi t) + \cos(2\pi 2t) + \cos(2\pi 3t) + \cdots).$$

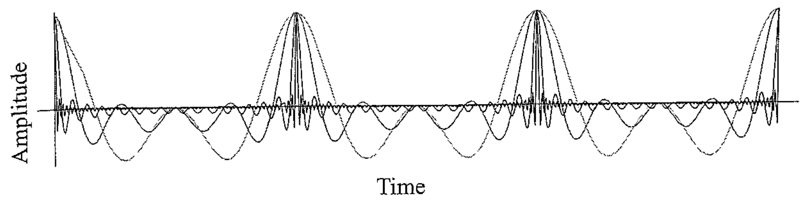


Figure 9.14
Sum of cosines converges to an impulse train signal.

Figure 9.14 shows the sum of cosines for $N = \{2, 4, 16, 32\}$.

In the limit, as $N \to \infty$, the sum of cosines converges to an impulse train function:

$$f(t) = \frac{1}{N} \cdot \frac{1 - \cos 2\pi k}{1 - \cos(2\pi k/N)}. \qquad \textit{Sum of Cosines, Closed Form} \quad (9.16)$$

The complex form of the sum of cosines—I suppose we should call it the *sum of phasors*—is the same as equation (9.15) with a phasor instead of a cosine:

$$f(t) = \frac{1}{N}\sum_{k=1}^{N} e^{i2\pi kt}. \qquad \textit{Sum of Phasors} \quad (9.17)$$

Here is its closed form:

$$f(t) = \frac{1}{N} \cdot \frac{1 - e^{i2\pi k}}{1 - e^{i2\pi k/N}}. \qquad \textit{Sum of Phasors, Closed Form} \quad (9.18)$$

As N grows, the sum of phasors becomes the complex helix form shown in figure 9.15.

A Note about Non-Band-Limited Signals The geometric forms of the equations for all these waveforms exist only in the limit when $N \to \infty$; therefore they have infinite bandwidth. All the preceding geometrical waveforms (except for the sum of cosines) decrease in amplitude (the triangular wave the most quickly) with increasing harmonic number k and eventually become insignificant. However, when synthesizing geometric waveforms in a sampled system such as a computer, one should use the Fourier summations, being careful to adjust the limit of summation N to prevent the highest harmonic from exceeding the Nyquist frequency if aliasing is not desired.

9.2.8 Wavetable Synthesis

The limitations of waveform synthesis are the same as for the Fourier transform: the waveform must be periodic, and the resulting spectrum must be harmonic. But a variant of this approach provides a very efficient way to synthesize an arbitrary harmonic spectrum on a computer.

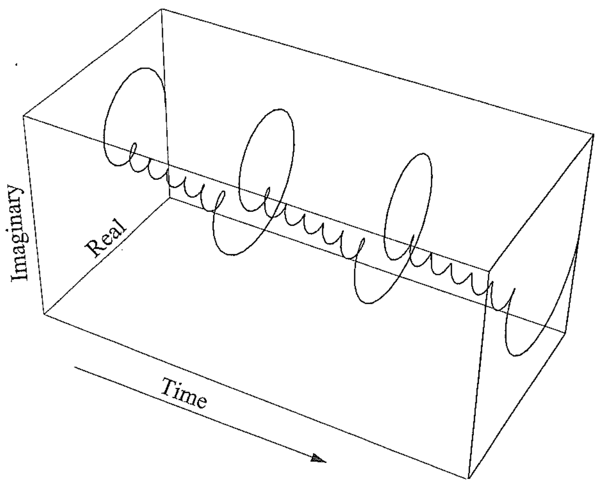


Figure 9.15
Sum of phasors.

Equation (9.6) is written in terms of continuous time t. To synthesize sound using this equation would require either an analog electronic synthesizer or an analog computer because they operate in continuous time. To adapt this approach to a digital computer coupled to a DAC, we must sample this continuous-time function by defining a sampling interval T (see section 1.4). Then defining $t = xT$, the discrete version of equation (9.6) is

$$y(x) = \sum_{n=1}^{N} A_n \sin 2\pi nfxT, \qquad \textit{Discrete Waveform Synthesis} \quad (9.19)$$

where x is the sample index, f is the fundamental frequency, T is sample period and n is the harmonic number.

Creating a Wavetable Since every period of equation (9.19) is the same, we can drastically reduce the amount of computation required to generate it by precomputing one period of the desired waveform and storing it in a table. We then iteratively read out the precomputed sample values from the table in real time to generate the sound. The only real-time computation needed is amplitude scaling and a small amount of arithmetic to determine the order in which to extract the samples from the table. To capture just one period of the waveform requires a modification of equation (9.19):

$$\tau_s = \sum_{n=1}^{N} A_n \sin 2\pi n \frac{s}{L}. \qquad \textit{Wavetable Synthesis} \quad (9.20)$$

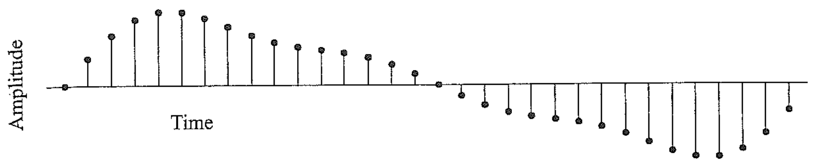


Figure 9.16
Sampled wave table.

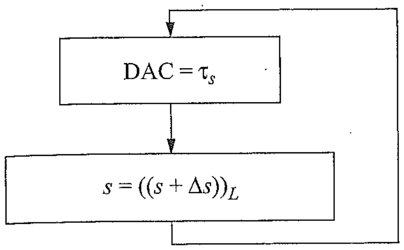


Figure 9.17
Table-lookup oscillator.

Here, τ_s represents a table of length L, indexed by s, that holds one period of the sampled waveform. The amplitudes of the desired harmonic components are given by the array A_n of length N. Figure 9.16 shows a sampled wavetable, given $A_n = \{1, 1/3, 1/5\}$, $N = 3$, and $L = 32$.

Indexing a Wavetable Having constructed a wavetable, we must now develop a means to extract values from it at a rate that will produce a desired synthesis frequency. Suppose the table stores one waveform period composed of $L = 1024$ samples. We must develop a computer program that will read the samples out of the table in any order we specify, one sample at a time. The samples are then converted through a DAC and sent to a loudspeaker. If we read out samples at a rate of $R = 8192$ samples per second, looping back to the beginning of the table when we run off the end, the frequency we'd output is $f = R/L = 8$ Hz. Figure 9.17 shows a simple two-step procedure that performs this operation.

This procedure first outputs the sample of table τ indexed by s to the DAC, then increments s by an amount Δs, then repeats these steps. The variable s is the *index*, and Δs is the *increment*. The index corresponds to the instantaneous phase of the oscillator, and Δs corresponds to the instantaneous frequency. In the example under discussion, $\Delta s = 1$. When s is incremented past the end of the table such that $s \geq L$, then the modulus operator applies, and s is reset back into the range of the table. (The expression $((q))_p$ means the remainder after integer division of q by p. See appendix section A.5.)

If we set $\Delta s = 2$ so that we skip every other sample, we'd run through the table twice as fast, and the frequency would double. Thus, for some sample rate R and table length L, the formula for the frequency f is

$$f = \Delta s \frac{R}{L}. \tag{9.21}$$

In this example we're limited to frequencies that are multiples of 8 Hz so long as Δs must be an integer. But if Δs is a real variable, we could theoretically generate any frequency. For example, if we set $\Delta s = 1.5$, equation (9.21) indicates we'd be able to generate a frequency of 12 Hz. But remember that the wavetable τ is just a list of samples, and there is nothing between them to index, unless we make up rules to define how to interpret the space between sample values. Here are some common choices for how to ascribe meaning to the space between samples.

Truncation Choose the nearest sampled value toward 0. For example, if $s = 3.7$, we throw away the 0.7 and choose sample 3. Truncation is performed with the floor operator, $\lfloor x \rfloor$, which returns the largest integer not larger than x. So, for example, $\lfloor 3.7 \rfloor = 3$. Since truncation simply discards the fractional part of the index, the index it produces can be off by nearly an entire sample. For example, if $s = 3.999$, truncation still indexes sample 3, although sample 4 might be a better choice in this case.

Rounding We can improve on truncation by performing rounding on the index before selecting the sample. For example, if $\Delta s = 3.7$, then because its fractional part $0.7 \geq 0.5$ we round up, choosing sample 4. We can express rounding as follows:

$$x = \begin{cases} x - \lfloor x \rfloor < 0.5, & \lfloor x \rfloor, \\ x - \lfloor x \rfloor \geq 0.5, & \lfloor x \rfloor + 1, \end{cases}$$

which sets x to $\lfloor x \rfloor$ if its fractional part is less than 0.5 and otherwise to $\lfloor x \rfloor + 1$.

Linear Interpolation Another approach is to assume that a straight line joins adjacent sample points and to select a point on this line that is proportional to the fractional part of the index.

For example, suppose we arbitrarily select two samples from a series: $y_{14} = 3.2$, and $y_{15} = 4.5$. Say the real index is $s = 14.7$. Draw a ramp connecting adjacent samples, as shown in figure 9.18, and measure off a distance between the samples corresponding to the size of the fractional part of the index, then measure the distance from the ramp to the x-axis at that point. In this example, the interpolated result would be $y = 0.7y_{14} + (1 - 0.7)y_{15} = 4.11$. In general, we proceed in two steps:

Step 1. $\sigma = s - \lfloor s \rfloor$.

Step 2. $y = \sigma y_{\lfloor s \rfloor} + 1 - \sigma y_{\lfloor s+1 \rfloor}$.

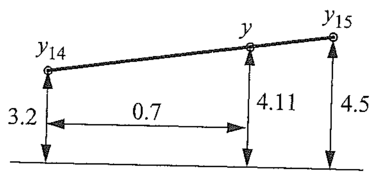


Figure 9.18
Linear interpolation.

Either truncation or rounding can be used effectively to traverse continuous and noncontinuous functions alike because only the actual samples are indexed. Truncation is computationally the simplest approach, rounding is slightly more complicated, and linear interpolation is most complex. Truncation can produce results that err by almost an entire sample, whereas rounding can err by at most half a sample. The error introduced by linear interpolation depends upon the precision of the available arithmetic, but it generally produces an error far less than the other two if indeed the underlying function is at least approximately linear between samples.

In digital audio, truncation and rounding errors appear as a kind of *phase jitter* because the signal is misindexed slightly by the error. It's as though the indexed region of the waveform is shoved forward or backward in time by the size of the truncation or rounding error. Phase jitter is a kind of frequency modulation, so it introduces distortion into the signal. For the truncating oscillator, truncation error can be reduced by increasing the number of samples of the underlying waveform. Linear interpolation also introduces some error because rarely will a linearly interpolated value agree exactly with the actual value of the underlying function; however, the error will generally be less than for truncation or rounding.

Spectrally, linear interpolation is equivalent to a second-order lowpass filter with a triangular impulse response. Take another look at figure 4.4, which shows that the convolution of two rectangular functions is a triangular function. The Fourier transform of the triangular impulse response is the sinc-squared function (see figure 4.36).

The main lobe of the sinc-squared function contains the spectral bandwidth of the original signal. It progressively filters out high-frequency components. The first null in the function occurs at half the sampling rate, so it behaves like a rather poor-quality anti-aliasing filter. The side lobes of the sinc-squared function contain attenuated copies of the original spectral bandwidth and cause aliasing (Smith 2004). The frequency response of linear interpolation is not ideal for at least two reasons. The spectrum is progressively lowpass-filtered near half the sampling rate and is nowhere flat, and aliasing contributed by the first side lobe is down only about 26 dB.

But that's not all. Linear interpolation is commonly used to obtain a fractional delay copy of a signal, for example, to stretch or compress a segment of audio in time. If the point of interpolation sits right on top of a source sample, no spectral change is introduced by the linear interpolation, and the frequency response is allpass. But if the interpolation point is halfway between two samples, the source is lowpass-filtered (by averaging) and suffers distortion from aliasing. If the interpolation point changes through time, which it typically does, linear interpolation introduces an objectionable variable filtering effect.

Additionally, linear interpolation only works if the underlying function from which the samples were derived was smoothly continuous before it was sampled; then interpolation may produce a reasonable approximation of what the function might have been between samples. Otherwise, the result is just a guess.

None of these techniques is ideal, and each introduces some distortion. But they are computationally quite efficient and are widely used. The method of band-limited interpolation (see section 10.2.7) generally yields the highest-quality results, though at greater computational cost.

9.2.9 Table Lookup Oscillator

To construct a table lookup oscillator, we want to choose a sample from table τ as defined in equation (9.20) indexed by s and then advance by Δs samples and repeat, starting over when we fall off the end of the table. Both the index s and increment Δs are real-valued variables, allowing us to progress through τ_s at an arbitrary rate. For simplicity, we'll truncate s to index each sample we output.

Oscillator with Constant Amplitude and Frequency The result will be a waveform with a constant amplitude and frequency. The wave produced depends on the contents of the wavetable τ. For each sample n, we perform the following steps:

Step 1. $y_n = A\tau_{\lfloor s \rfloor}$.

Step 2. $s = ((s + \Delta s))_L$.

Step 3. $n = n + 1$.

Step 4. Repeat.

Static Table Lookup Oscillator Procedure (9.22)

y_n is the array of output samples, A is amplitude, L is the length of the table, the notation $\lfloor x \rfloor$ is the floor function, and the notation $((x))_L$ means the value of x modulo L.

1. We start by outputting the currently indexed value: $y_n = A\tau_{\lfloor s \rfloor}$. The output sample y_n is the table value τ indexed by the floor of the integer table lookup index s and scaled by the amplitude A.

2. Next, we determine which table value to index next time around: $s = ((s + \Delta s))_L$. The next table index s is obtained by adding the current table index and the increment Δs, modulo the length of the table L.

3. Finally, we advance to the next sample time, $n = n + 1$, then repeat the calculation for as many output samples as required.

The increment Δs determines the rate at which we progress through the table, and hence the frequency f of the oscillator. Suppose we have some fixed value of Δs. If we increase the sampling rate R holding Δs constant, the frequency goes up. And if we increase the length of the table L holding Δs constant, the frequency goes down. Since τ holds exactly one period of the waveform, the length L of the table corresponds to 2π radians. Thinking along these lines, we see that the increment Δs corresponds to the frequency f:

$$\Delta s = f\frac{L}{R}.$$

Oscillator Increment and Frequency (9.23)

Thus, to achieve a frequency $f = 8$ with a table length $L = 1024$ and sample rate $R = 8192$, we set $\Delta s = 1$.

Oscillator with Variable Amplitude and Frequency In order to be musically useful, we must allow the oscillator's instantaneous frequency to change through time, for instance, to let the pitch glide up and down over time. The simplest way to accommodate this is to have an array of increments, Δs_n, one for every output sample n, representing a sequence of instantaneous frequencies we wish to synthesize. For every sample we output, we determine the next table value τ_s to pick by taking the current table index s_n and adding the current increment Δs_n to it. The array of increments Δs_n allows us to vary frequency through time. Let's also have an array of amplitudes A_n so that amplitude can vary per sample as well. A procedure that takes these requirements into account is as follows (Mathews 1969):

Step 1. $y_n = A\,\tau_{\lfloor s_n \rfloor}$.

Step 2. $s_{n+1} = ((s_n + \Delta s_n))_L$. *Table Lookup Oscillator Procedure* (9.24)

Step 3. $n = n + 1$.

Step 4. Repeat.

The steps are as before except that potentially we have a different table index s_n and increment Δs_n on each sample n, and a different amplitude A_n on each sample. (I say "potentially" because, of course, we could set all the A_n and Δs_n the same, in which case this would revert to the static oscillator.)

1. We start by outputting the currently indexed value: $y_n = A_n \tau_{\lfloor s_n \rfloor}$. The nth output of the oscillator is formed from the product of the nth amplitude value times the sample indexed by the floor of the nth index value.
2. Next we compute the index of the subsequent output. $s_{n+1} = ((s_n + \Delta s_n))_L$. The next index is obtained by adding the current index and increment, modulo the length of the table.
3. We advance to the next moment, then repeat the calculation: $n = n + 1$.

The instantaneous increment Δs_n corresponds to the instantaneous frequency f_n:

$$\Delta s_n = f_n \frac{L}{R}. \qquad \textit{Instantaneous Oscillator Increment and Frequency} \quad (9.25)$$

We can model the operation of the truncating table lookup oscillator as shown in figure 9.19. For each sample n, the current frequency f_n is multiplied by L/R to create Δs_n. Then Δs_n is added to the previous value of s_n to make the next real index value. We then take the modulus of the real index, producing the new s_n. We take the floor of s_n to give us an integer index that we can use to

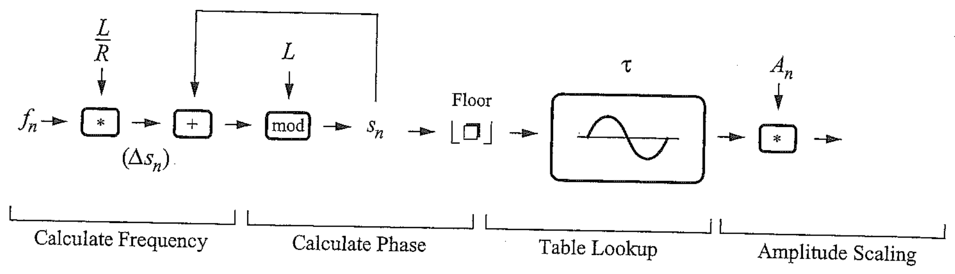


Figure 9.19
Truncating lookup oscillator patch.

select one sample from the wavetable τ. Finally, the result from the table is scaled by the current amplitude A_n.

9.3 Amplitude Modulation

Amplitude modulation (AM) varies the instantaneous amplitude of a signal, usually in a periodic manner. Figure 9.20 shows a sinusoid with a periodically varying instantaneous amplitude and a constant frequency.

Consider the patch shown in figure 9.21. The output of the first cosine wave oscillator, the *modulating oscillator*, is connected to the amplitude input of the second, the *carrier oscillator*.[6] In this way, the amplitude of the carrier oscillator is dynamically controlled by the modulating oscillator. I've chosen to use cosines here because it will help later with the mathematics, but sine waves would have done just as well in practice. In this example, the frequency of the modulating oscillator is f_m, and the frequency of the carrier is f_c. The amplitude input of the modulating oscillator is I, which stands for *modulation index*. Note that a constant value of 1.0 is added to the output of the modulating oscillator before it is fed into the carrier oscillator's amplitude input.

Writing out the equation for this patch, we have

$$f(t) = (1.0 + I\cos 2\pi f_m t)\cos 2\pi f_c t\,.$$

Simplify by letting $\omega_c = 2\pi f_c$, and $\omega_m = 2\pi f_m$:

$$f(t) = \underbrace{(1.0 + I\cos\omega_m t)}_{\text{M}}\underbrace{\cos\omega_c t}_{\text{C}}. \qquad \textit{Amplitude Modulation with Carrier in Output} \quad (9.26)$$

The terms labeled M correspond to the modulator, and the terms labeled C to the carrier. The formula essentially multiplies M and C terms to create the amplitude-modulated signal $f(t)$.

Note that when $I = 0$, the M term equals 1.0 and drops out, so the equation reduces to $f(t) = \cos\omega_c t$, a simple sinusoid at the carrier frequency.

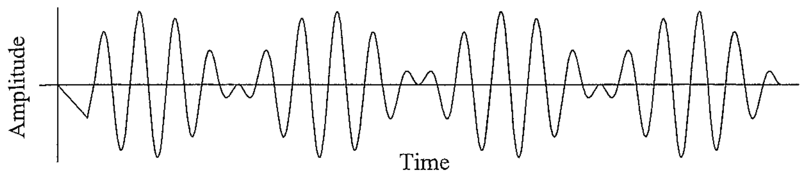


Figure 9.20
Amplitude modulation, carrier present in the output.

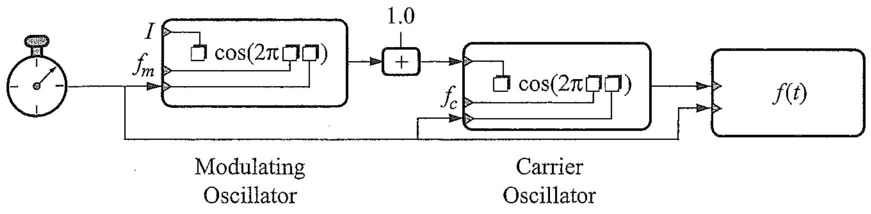


Figure 9.21
Amplitude modulation.

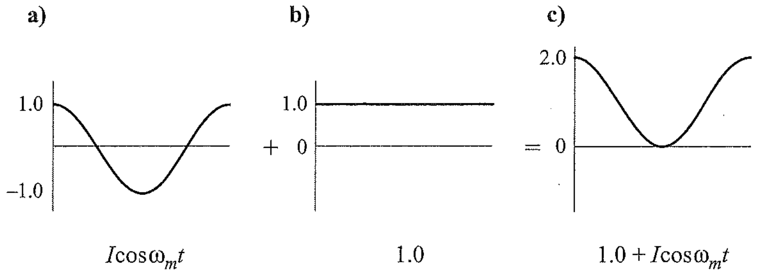


Figure 9.22
Graphical view of the M term.

When $I \neq 0$, the M term essentially plots a cosine wave at frequency ω_m and adds 1.0 to every point. The effect is shown in figure 9.22 for a single period of the modulating cosine wave with $I = 1.0$. Figure 9.22a shows the term $I \cos \omega_m t$ and figure 9.22b shows the constant function 1.0. When summed in 9.22c, they form the M term.

Note that since $I = 1$, in this example the value of the function in figure 9.22c ranges from a maximum of 2 to a minimum of 0. This function is then multiplied by the carrier term C in equation (9.26). When the M term's value is momentarily 0, it nullifies the carrier signal, and when it is at its maximum value of 2, it doubles the amplitude of the carrier signal. In between, the amplitude of the carrier follows the contour of the cosine wave as determined by the M term. If we restrict the range of the index to $0 \leq I \leq 1.0$, the result of the M term is always greater than or equal to 0, as shown in figure 9.22c,

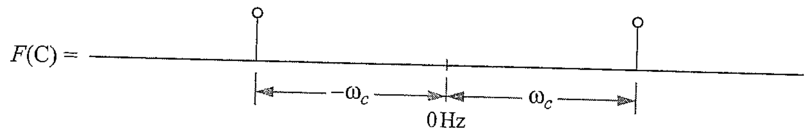


Figure 9.23
Spectrum of C terms.

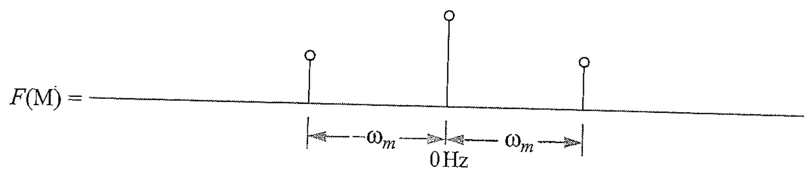


Figure 9.24
Spectrum of M terms.

and so it is an unsigned quantity. When $I = 1$, we say that the *depth of modulation* is 100 percent. If $I = 0$, the depth of modulation is zero. The waveform in figure 9.20 shows a modulation index of 100 percent. (I've also set the carrier and modulator frequencies in that example so that $\omega_m << \omega_c$. Thus many periods of the carrier oscillator are affected by one period of the modulator, making it easier to see the pattern.) The result we see is that the modulating signal dynamically controls the *amplitude envelope* of the carrier signal.

What is happening spectrally? Recall from section 2.6.8 that a real cosine waveform with unity amplitude is the vector sum of two half-amplitude phasors of equal sign and opposite frequency (see especially equation (2.52)). Since the C term in (9.26) is a cosine waveform, its spectrum is a cosine spectrum with half-amplitude frequency components $\pm\omega_c$ (figure 9.23).

The M term in (9.26) is the sum of the constant function 1.0 and a cosine waveform. The spectrum of a constant is just its magnitude at 0 Hz (see section 4.7.1). Thus the spectrum of the M term is a component with magnitude 1.0 at 0 Hz and a cosine spectrum with half-amplitude frequency components $\pm\omega_m$ (figure 9.24).

We can tie these two spectral plots together by recalling that since the M and C terms are multiplied in the time domain, their spectra are convolved in the frequency domain. The consequence is that copies of the M components are placed around each of the components of C (figure 9.25). Thus amplitude modulation can be thought of as frequency-shifting the spectrum of the modulating signal by the frequency of the carrier ω_c.

Since the entire spectrum of the modulating signal is shifted up and down by ω_c and $-\omega_c$, this form of amplitude modulation is sometimes called *double sideband modulation*. In broadcasting applications, it is an unnecessary redundancy to have identical upper and lower sidebands, and some broadcast systems filter out one sideband or the other, a technique known as *single sideband* modulation, in order to reduce the amount of radio frequency spectrum required by the transmitter.

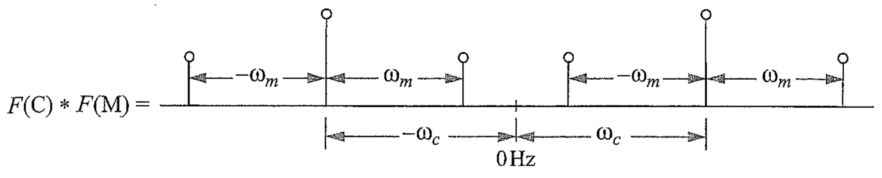


Figure 9.25
Convolution of M and C.

Here is the equation for amplitude modulation again for reference:

$$f(t) = (\underbrace{1.0}_{\text{Duplicate of the carrier}} + \underbrace{I}_{\text{Amplitude of the sidebands}} \cos \omega_m t) \cos \omega_c t.$$

Comparing figure 9.24 and 9.25, note that the spectral component corresponding to the modulation constant 1.0 is convolved to reside at the position of the carrier frequency $\pm\omega_c$. The amplitude of the sidebands that surround the carrier are determined by I.

9.3.1 Finding the Spectrum

We can derive the spectrum for amplitude modulation as follows. First, notice that the equation for amplitude modulation has the general form: $(a + b)c$, where $a = 1.0$, $b = I\cos \omega_m t$, and $c = \cos \omega_c t$. If we expand $(a + b)\,c$ to $ac + bc$ and substitute, we have

$$f(t) = 1.0 \cdot \cos \omega_c t + \underbrace{I\cos \omega_m t \cdot \cos \omega_c t}_{\text{Product of two cosines}}. \tag{9.27}$$

Note that the right-hand terms in (9.27) represent the product of two cosines. There is a handy trigonometric identity that shows what we can do with the product of two cosines (see appendix section A.4):

$$\cos x \cos y = \frac{1}{2}[\cos(x + y) + \cos(x - y)].$$

Substituting $\cos x = I\cos \omega_m t$, and $\cos y = \cos \omega_c t$ from this trigonometric identity into (9.27), we have

$$\begin{aligned} f(t) &= \cos \omega_c t + \frac{I}{2}[\cos(\omega_c t + \omega_m t) + \cos(\omega_c t - \omega_m t)] \\ &= \underbrace{\cos \omega_c t}_{\text{Carrier}} + \underbrace{\frac{I}{2}\cos(\omega_c + \omega_m)t}_{\text{Upper sideband}} + \underbrace{\frac{I}{2}\cos(\omega_c - \omega_m)t}_{\text{Lower sideband}}. \end{aligned} \tag{9.28}$$

Equation (9.28) shows the spectral form of amplitude modulation. The first term represents the ca rier frequency, and the second and third represent the upper and lower sidebands, respectively. No that the carrier frequency does not depend in any way upon ω_m; all the dynamic properties of ampl tude modulation are controlled by the frequencies and amplitudes of the sidebands, which are eac one half of the value of I.

The discussion has assumed that the modulating function is a cosine oscillator, and while th is a good assumption for analysis purposes, in practice any signal can act as the modulating fun tion. For instance, if ω_c is a radio frequency, and $M(t)$ is an announcer's microphone signal, th result could be an AM radio broadcast signal according to

$$f(t) = [1.0 + IM(t)]\cos\omega_c t. \qquad \textit{Amplitude Modulation, General Case} \quad (9.2$$

9.3.2 Ring Modulation

Equation (9.26) for amplitude modulation is a two-quadrant multiply, a signed quantity times a unsigned quantity (see section 3.1.7). Since the M term is always greater than or equal to (because we require $0 \le I \le 1.0$), it is an unsigned quantity. And because the C term is an acoustica waveform, it goes positive and negative, so it is signed.

Now, if we left out the $+1.0$ from the M term in equation (9.26) it would simply be $I\cos\omega_m t$ which is a signed term. The resulting equation,

$$f(t) = (I\cos\omega_m t)\cos\omega_c t, \qquad \textit{Ring Modulation} \quad (9.30$$

is a four-quadrant multiply because the M and C terms are both signed. The spectrum correspond ing to equation (9.30) is shown in figure 9.26.

Ring modulation is the same as amplitude modulation but without the carrier frequency presen in the output spectrum. For a more rigorous proof of the spectral consequences of ring modulation see appendix section A.10.

9.3.3 Musical Uses of Amplitude Modulation

If the frequency of the modulating oscillator is subaudio, amplitude modulation is called *tremolo*. This is the effect, for instance, of the tremolo control on a Fender electric guitar amplifier.

Probably the most important use of amplitude modulation is the central role it plays in spectral analysis and synthesis (see chapter 3). The Fourier transform essentially ring-modulates the probe phasor and the input signal as the first step to determine the spectrum of the sound.

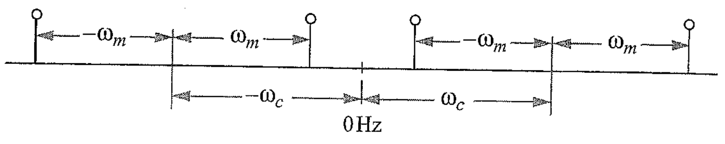


Figure 9.26
Spectrum of ring modulation.

Since there is not necessarily a connection between the spectrum of the carrier and the modulating signal, amplitude modulation and ring modulation are really good ways to create inharmonic spectra. Depending upon the materials and treatment, the effect can still retain enough of the original timbre of the modulating signal to identify its source, though it sounds mutated in a strangely dissonant but still coherent way. Some Hollywood science fiction movies from the 1950s used ring modulation to give the aliens spooky-sounding voices.

9.4 Frequency Modulation

Frequency modulation (FM) varies the instantaneous frequency of a signal in a periodic manner. Figure 9.27 shows a frequency-modulated sinusoid with time-varying frequency and a constant amplitude.

The frequency of a carrier sinusoid is varied by a modulator sinusoid, and the strength of the modulator's effect on the carrier frequency is called the depth of modulation. In figure 9.27 the carrier frequency is about ten times higher than the modulating frequency, and the depth of modulation is quite strong.

The frequency of an oscillator can be varied in time by patching the output of the modulating oscillator to the frequency input of the carrier oscillator (figure 9.28). Writing out the equation for this patch, we have

$$f(t) = A\sin(\omega_c t + \Delta f \sin \omega_m t), \qquad \textit{Frequency Modulation} \quad (9.31)$$

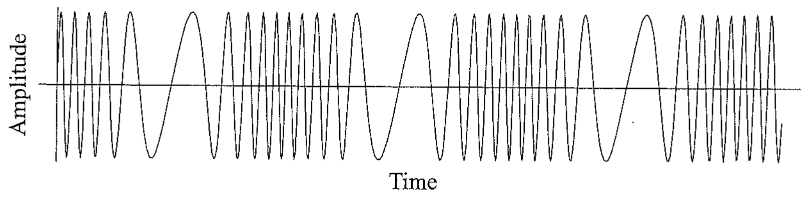


Figure 9.27
Frequency-modulated sinusoid.

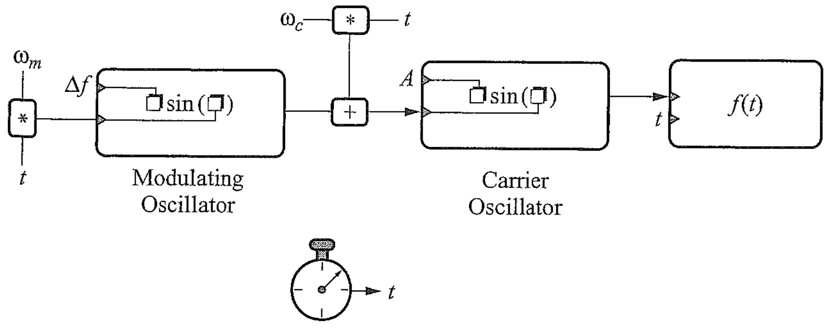


Figure 9.28
Frequency modulation patch.

where A is amplitude, $\omega_c = 2\pi f_c$ is the carrier frequency, and $\omega_m = 2\pi f_m$ is the modulating frequency. The term Δf, called *peak frequency deviation,* determines the amplitude of the modulating oscillator's output, which controls the swing of the carrier oscillator's frequency. If $\Delta f = 0$, equation (9.31) reduces to $f(t) = A \sin \omega_c t$, which is a sine wave at a constant frequency. However, if $\Delta f \neq 0$, a multitude of sidebands appear, positioned above and below the carrier at multiples of $\pm f_m$.

To get a feel for this, let's use equation (9.31) to create some sample spectra with different values of Δf. Let's set $f_c = 1000$ Hz, and $f_m = 100$ Hz, and increase Δf gradually, starting at 0. When $\Delta f = 0$, we have a sine wave at constant frequency with a magnitude spectrum as shown in figure 9.29a.

As Δf grows, we see the carrier decline in amplitude, and additional sidebands at fixed frequencies $f_c \pm nf_m$ enter the spectrum, where n is the integer order of the sidebands. It appears that as Δf increases, energy is stolen from the carrier frequency and distributed over an ever wider number of sidebands. However, in figure 29d, where $\Delta f = 3.6$, the carrier makes a forceful reappearance, even as the number of sidebands continues to grow.

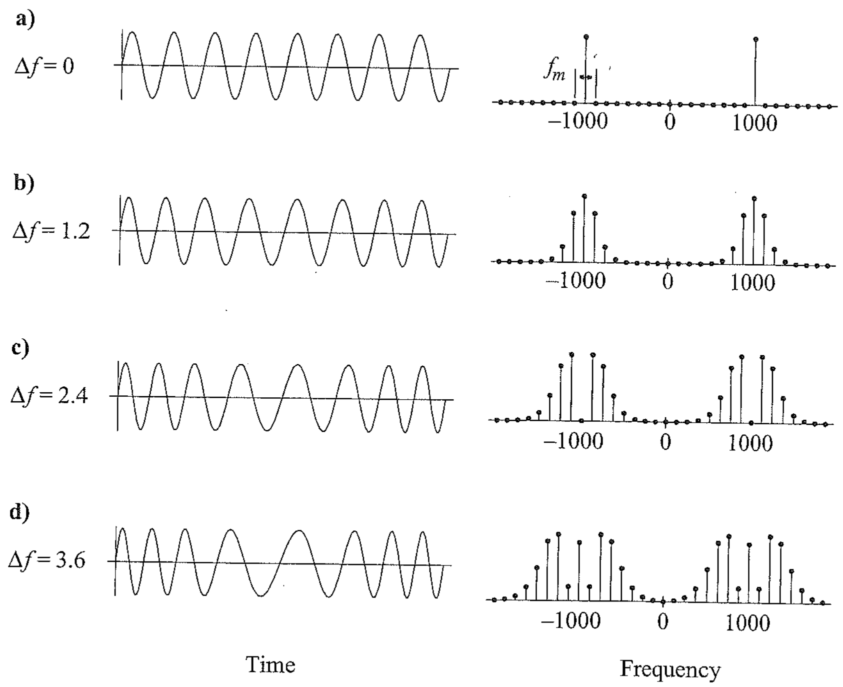


Figure 9.29
Sample FM spectra, various values of frequency deviation.

9.4.1 FM as the Combination of Fixed Frequency Components

How can it be that if we sweep a frequency continuously across a certain limited frequency range it suddenly looks like we have a collection of fixed frequency components spread at discrete points across a wide range of the spectrum? And what accounts for the comings and goings of the sidebands?

It turns out that if fixed frequency sinusoids are combined at exactly the right frequencies, phases, and amplitudes, the result is identical to a frequency-modulated sinusoid. For this to work out, the sidebands of the same order above and below the carrier must be equal in amplitude, and the odd-ordered lower sidebands must be 180° out of phase from the even-ordered sidebands. Additionally, the amplitudes of the various orders of sidebands must be carefully chosen. An example will give a flavor of this.

The three sine waves shown in figure 9.30a are defined as follows:

$$\begin{aligned} x(t) &= 0.98 \sin 2\pi f t, && \text{Carrier} \\ y(t) &= 0.24 \sin 2\pi (f+\Delta f)t, && \text{Upper sideband} \\ z(t) &= -0.24 \sin 2\pi (f-\Delta f)t, && \text{Lower sideband} \end{aligned}$$

where $f = 1$ and $\Delta f = 1/16$. Figure 9.30a shows 16 periods of $x(t)$, just enough time for the upper and lower sidebands to precess against each other once. These three waves are summed, $s(t) = x(t) + y(t) + z(t)$, to create the wave labeled $s(t)$ in figure 9.30b. The lighter sinusoid in

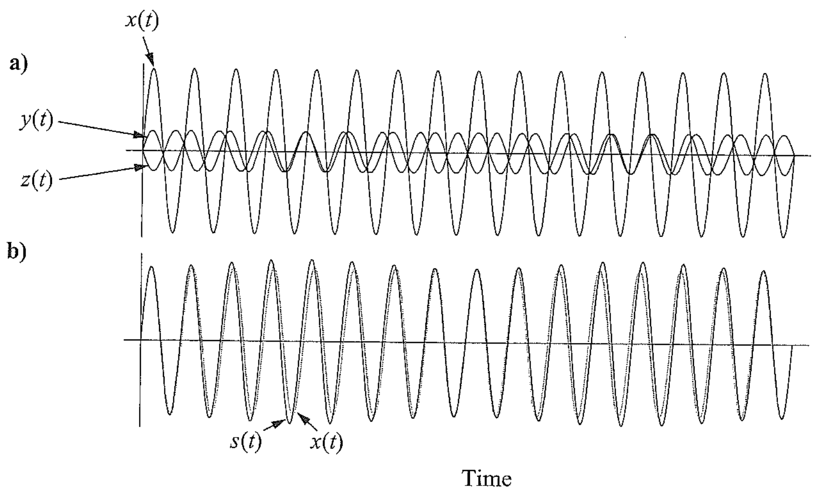


Figure 9.30
Sum of three sine waves creates frequency modulation.

line figure 9.30b is just $x(t)$ shown again for reference. The curves look almost the same, but looking carefully, it's evident that they are not.

Notice that $s(t)$ first lags behind, then advances past $x(t)$. It looks as though the frequency of $s(t)$ changes over time. This is not an optical illusion: in fact, the frequency of $s(t)$ is changing. Summed sinusoids with carefully chosen amplitudes and frequencies are equivalent to a single waveform that changes frequency over time.

Notice in figure 9.30a that when the sidebands are in phase with each other, they are 90° out of phase with respect to the carrier, and when they are out of phase with each other, they are –90° out of phase with the carrier. The sum of the three components appears to have nearly constant amplitude, but it advances and lags in phase relative to the carrier. Figure 9.27 shows this to be exactly the behavior of a frequency-modulated sinusoid: its phase velocity advances and retreats, alternately increasing and decreasing its frequency slightly.

Although the amplitude of $s(t)$ changes slightly, this is because we're summing only three sinusoids together. In order to eliminate the amplitude change, we'd need to sum an infinite number of sinusoids at ever wider frequencies and ever smaller amplitudes according to a particular formula.

What is the formula that determines how to mix the carrier and sidebands together so that a constant-amplitude frequency-modulated sinusoid results?

9.4.2 Return of the Bessel Functions

The amplitudes of the FM-induced sidebands are determined by Bessel functions of the first kind and nth order. Interestingly, they are the same Bessel functions that characterize the motion of vibrating membranes (volume 1, figure 8.21). $J_0(I)$ determines the amplitude of the carrier, $J_1(I)$ determines the amplitude of the first upper and lower sidebands, and in general, $J_n(I)$ determines the amplitude of the nth upper and lower sidebands.

The frequency-modulated sinusoid given in equation (9.31) can be seen to be equivalent to a set of fixed-frequency sinusoids whose amplitudes and phases are determined by the Bessel functions, according to the following trigonometric identity:

$$\begin{aligned} f(t) &= A\sin(\omega_c t + I\sin\omega_m t) \\ &= A\sin(\theta + I\sin\beta) \\ &= J_0(I)\sin\theta + \sum_{n=1}^{\infty} J_n(I)[\sin(\theta + n\beta) + (-1)^n \sin(\theta - n\beta)]. \end{aligned} \tag{9.32}$$

If we expand the terms of the infinite summation in equation (9.32), we obtain

$$\begin{aligned} f(t) = {} & J_0(I)\sin\theta \\ & + J_1(I)[\sin(\theta+\beta) - \sin(\theta-\beta)] \\ & + J_2(I)[\sin(\theta+2\beta) + \sin(\theta-2\beta)] \\ & + J_3(I)[\sin(\theta+3\beta) - \sin(\theta-3\beta)] \\ & + \cdots. \end{aligned} \tag{9.33}$$

The sidebands are at multiples of the modulating frequency, odd lower sideband components are negative, and the amplitudes of successive orders of sidebands are determined by the corresponding orders of Bessel functions.

9.4.3 FM and Bessel Functions

We can make the following observations about equation (9.32) and about Bessel functions to help explain what makes FM synthesis sound the way it does.

- The summation of sidebands goes on to infinity. Only when $I = 0$ can we say that there is absolutely no energy in any sidebands. Therefore, we can say that a frequency-modulated sinusoid has an infinite spectrum, but depending upon the value of I, most of the sidebands have insignificant energy.
- The odd-ordered lower sidebands alternate sign. The term $(-1)^n$ accomplishes this sign reversal for the odd lower sidebands.

Looking at the characteristic shape of the Bessel functions themselves,

- The Bessel functions look somewhat like damped sinusoids, since their peak amplitudes diminish gradually as the modulation index I increases.
- Note that J_0 is different from all the other Bessel orders because $J_0(0) = 1$, whereas for all other orders $n > 0$, $J_n(0) = 0$. So when $I = 0$, $J_0(I)$, which controls the amplitude of the carrier, has all the available energy.
- Higher-order Bessel functions start up from zero more and more gradually. This means that higher-order sidebands have significant energy only when the modulation index I is large.
- Depending upon the index, some Bessel functions will be strongly positive, and others—even nearby components—might be near zero or at zero, or negative. For example, in volume 1, figure 8.22, $J_2(5)$ is almost zero while $J_3(5)$ is near its maximum.
- Also notice in that figure and in figure 9.29d that when $I > 2.5$, some Bessel functions produce a negative scaling coefficient for some values of I.
- As I increases, the amplitudes of the sidebands increase and decrease in strength in a characteristic damped sinusoidal way.

Thinking about all these points, and especially the last one, if we have a way to change the modulation index dynamically, the spectrum will change in complex ways, tracking the ups and downs of the Bessel function curves. Altogether, the spectrum of the sound will exhibit a lively character as we sweep I from one value to another during the time a note is sounding. This is the particular charm of FM synthesized sounds, that they contain an interesting built-in spectral evolution, determined by the Bessel function curves, as the modulation index varies. If an important aim of good synthesis techniques is to elicit complex yet predictable behavior from simple controls, then FM synthesis is certainly a very good technique indeed.

9.4.4 Modulation Index

At this point, we have two equations for frequency modulation. First there is equation (9.31):

$$f(t) = A\sin(\omega_c t + \Delta f \sin \omega_m t),$$

which expresses the strength of the modulating oscillator in terms of peak frequency deviation Δf. Then there is equation (9.32):

$$f(t) = A\sin(\omega_c t + I\sin \omega_m t),$$

which expresses spectra in terms of the Bessel function index I. If I could be related to Δf, we could use the Bessel functions to predict the spectrum we would get if we employ a particular peak frequency deviation. The rest of this section searches out this connection.

The expression for instantaneous phase in equation (9.32) is $\omega_c t + I\sin\omega_m t$. In graph theory, this is an expression of the form $ax + b$, defining a slope, where $a = \omega_c$ determines the rate at which the slope grows, $x = t$ marks units on the x-axis, and $b = I\sin\omega_m t$ is the offset. This is not a linear slope because b is not a constant; rather it is itself a slope that changes steepness through time sinusoidally. But it's still a slope—just a rather bumpy one.

Here is an example. For convenience, let $\theta(t) = \omega_c t + \Delta f \sin\omega_m t$. If we set $\Delta f = 0$, then $\theta(t) = \omega_c t + 0$, which is just a linear ramp function that intersects the y-axis at zero. Curve (a) in figure 9.31 shows the slope of $\theta(t) = 2\pi \cdot 2 + 0$ over a time interval of 4 seconds. The slope of $\theta(t)$ determines the frequency of the sinusoid in equation (9.32), which will be 2 Hz. If, as with function (b), we set $\omega_c = 2\pi \cdot 8 + 0$, it would have a steeper slope, and the frequency of the sinusoid would go up to 8 Hz.

Now let's make this more interesting. Setting $\omega_c = 4\,\text{Hz}$, $\omega_m = 2\,\text{Hz}$, and $\Delta f = 2$, we get curving function (c) in figure 9.31. Here, the instantaneous frequency varies, or *modulates,* according to the instantaneous slope of $\theta(t)$. Where the slope rises more sharply, the frequency goes up; where it rises less sharply, the frequency goes down. The most positive part of the slope of this function will approximately double the frequency when t is in that region; where the slope momentarily goes to zero (where it becomes horizontal), the frequency momentarily goes to zero.

Last, consider the slope of (d) in figure 9.31. Here, $\omega_c = 6\,\text{Hz}$, $\omega_m = 2\text{Hz}$, and $\Delta f = 4.8$. Notice that sometimes the slope of this curve goes negative. That means the value of $\theta(t)$ decreases through time in that region, and the phase velocity of the oscillator is negative. We're producing *negative frequencies* in that region of time. We can leverage these ruminations about $\theta(t)$ into a precise understanding of how modulation index I relates to peak frequency deviation Δf, as follows.

If the ramp function $\theta(t)$ defines frequency, then its derivative defines instantaneous frequency. According to the rules laid out in section 6.1, the derivative of $\theta(t)$ with respect to time can be expressed in terms of equation (9.32):

$$\begin{aligned}\frac{d\theta}{dt} &= \frac{d}{dt}\omega_c t + \frac{d}{dt}I\sin\omega_m t \\ &= \omega_c + I\omega_m \cos\omega_m t.\end{aligned}$$

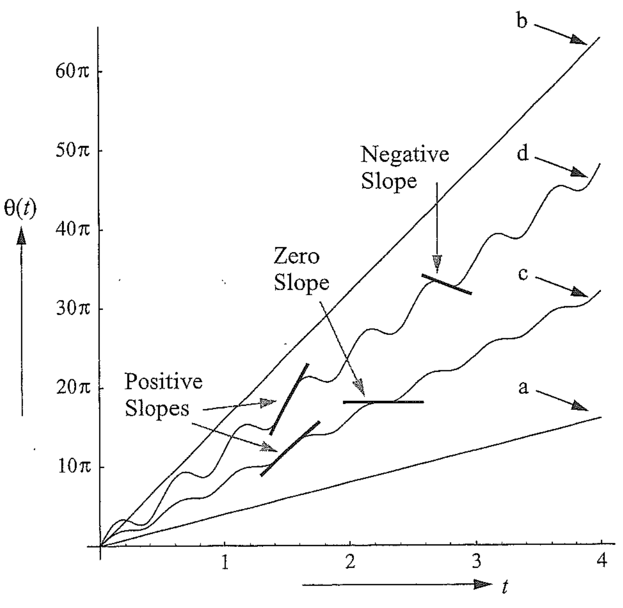


Figure 9.31
Linear and sinusoidally driven slope functions.

Similarly, the derivative of $\theta(t)$ in terms of equation (9.31) is

$$\frac{d\theta}{dt} = \omega_c + \Delta f \cos \omega_m t\,.$$

Since both these expressions define the derivative of $\theta(t)$ with respect to time t, we may equate them:

$$\omega_c + \Delta f \cos \omega_m t = \omega_c + I\omega_m \cos \omega_m t\,.$$

Solving for Δf yields

$$\Delta f = I\omega_m, \qquad \textit{Peak Frequency Deviation} \quad (9.34)$$

and solving for I produces

$$I = \frac{\Delta f}{\omega_m}. \qquad \textit{Modulation Index} \quad (9.35)$$

Equations (9.34) and (9.35) allow us to relate depth of modulation, the modulation frequency, and the index of the Bessel functions. In practical terms, if we want to use FM to create a spectrum that has the strengths of the Bessel functions at some particular index I, and we want the components to be separated in frequency by ω_m, then we must choose a depth of modulation Δf according to these formulas (F. R. Moore 1990, 318).

9.4.5 Calculating FM Spectra

As we've seen, for some positive carrier frequency, as index I grows,

- The spectral bandwidth grows.
- The upper sidebands grow toward higher frequencies.
- The lower sidebands grow toward lower frequencies.

If the index I grows sufficiently, sidebands below 0 Hz eventually become active. What happens to the lower sidebands that venture below 0 Hz? They turn into negative frequencies, or equivalently, we hear a phase-reversed positive frequency. Both interpretations are valid mathematically: a negative frequency can be thought of simply as a phasor spinning in the opposite direction of a corresponding positive frequency. Alternatively, 0 Hz can be thought of as a mirror that reflects negative frequencies back into the positive domain, but with a 180° phase shift. In a nutshell,

$$\sin(\theta) + \sin(-\theta) = \sin(\theta) - \sin(\theta) = 0. \tag{9.36}$$

In other words, reversing the phase of a negative frequency makes it equivalent to a corresponding positive frequency with inverted sign (figure 9.32).

Let's look at a realistic example FM spectrum that includes sidebands that have strayed into negative frequency territory. Let's set $f_c = f_m = 100\,\text{Hz}$, and $I = 4.9$. The upper sidebands will be at 200, 300, 400, . . . Hz, and the lower sidebands will be at 0, −100, −200, . . . Hz. We can calculate the spectrum from the Bessel functions by looking up the strengths of the n sidebands for $J_n(4.9)$, as shown in figure 9.33.

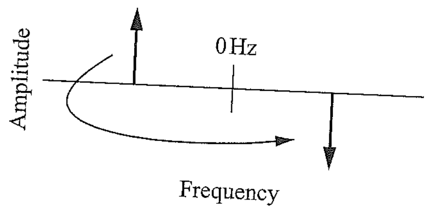


Figure 9.32
Reversing the phase of a negative frequency.

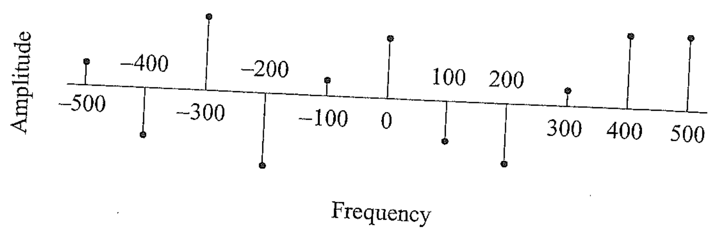


Figure 9.33
Sample FM spectrum.

The Bessel functions determine the relative strengths and signs of the components. The component at 0 Hz corresponds to a positive constant offset in the waveform. If we wrap the negative frequencies around to positive frequencies with a change of sign (or a 180° phase shift), they land on top of and sum with any components at the same frequency. They either add energy to or subtract energy from their positive frequency counterparts, depending upon whether their signs agree. This process is shown in figure 9.34. The result of combining the wrapped negative frequencies is the positive-frequency spectrum shown in figure 9.35. Our ears don't distinguish negative amplitude components from positive components. If we take the magnitude of this spectrum, we obtain what we'd actually hear (figure 9.36).

The wrapped negative components landed on top of positive frequency components because the ratio of carrier to modulating frequency was unity. Whenever $f_m/f_c = 1$, the resulting spectrum will always be harmonic. What about other possible ratios?

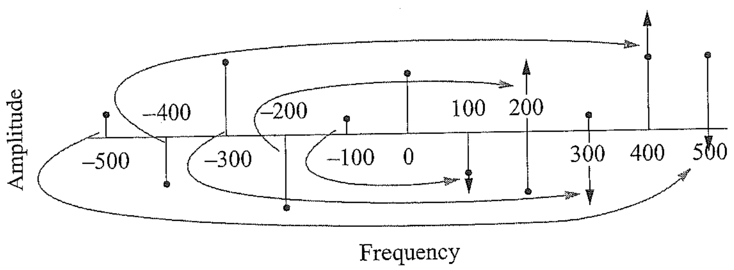


Figure 9.34
Components reflecting around 0 Hz.

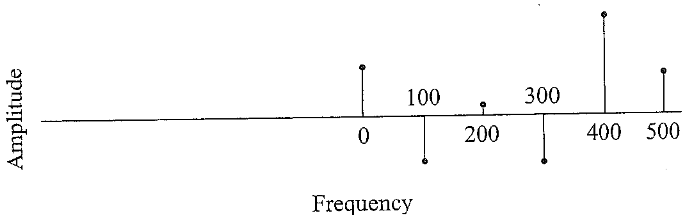


Figure 9.35
Resulting positive spectrum.

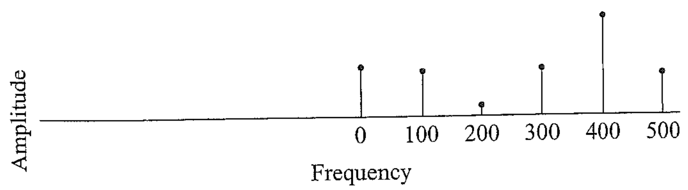


Figure 9.36
Magnitude FM spectrum.

Truax (1977) suggested a *harmonicity ratio,* defined as the ratio of modulating frequency to carrier frequency, such that $H = f_m/f_c$, where H can be either an integer or a real number.

There are two major classes of FM spectra to consider:

- If H is rational, the spectrum is harmonic
- If H is irrational, the spectrum is inharmonic

Rational Harmonicity The preceding example spectrum belongs to the class of rational harmonicity, $H = 1$. As we've seen, this spectrum is harmonic, and the carrier frequency is also the fundamental.

An interesting subclass of spectra consists of $H = 1/m$, where m is a positive integer. For example, if $H = 1/2$, the carrier frequency is twice the modulating frequency. The spectrum is still harmonic, but the carrier is the second component, not the first. In fact, all spectra $H = 1/m$ are harmonic, and the carrier is positioned at component m.

The other interesting subclass is spectra where $H = n$, and $n > 1$. These spectra are also harmonic. Here, however, some components are missing, depending upon the value of n. An important example is $H = 2$, where the modulating frequency is twice the carrier frequency ($f_m = 2f_c$). Sidebands occur at $f_c \pm 2kf_m$ (where $k = 0, 1, 2, \ldots$) leading to a spectrum that omits all even harmonics, making this a convenient way of modeling the clarinet timbre. The first upper sideband component appears at $f_c + 2f_c = 3f_c$, and the next appears at $f_c + 4f_c = 5f_c$ and so on. Going down, the first lower sideband component appears at $f_c - 2f_c = -f_c$, the second at $f_c - 4f_c = -3f_c$, and so on. The negative frequencies wrap around on top of the positive ones, resulting in a spectrum with no even components. In general, for $H = n$, and $n > 1$, only components $kn + 1$ will be present.

Irrational Harmonicity In general, if $H = n/m$, n and m are positive integers, the spectrum is harmonic. But if H is irrational, the spectrum is inharmonic. In this case, the components in the spectrum do not fall at integer multiples of the carrier frequency. Second, the negative frequencies that wrap around 0 Hz typically land between, not on, the positive frequency components, making the spectrum denser. Consider, for example, $H = \sqrt{2}$, which has components spreading out from the carrier at $f_c \pm kf_c\sqrt{2}$, $k = 0, 1, 2, \ldots$.

Again, there are two subclasses of interest. For $H = 1/m$, where m is a positive irrational number, components cluster increasingly densely around the carrier as m increases, mimicking the spectra of drums and gongs. There tends to be no clear fundamental for these timbres; hence they have no distinct pitch.

For $H = n$, $n > 1$, and n is a positive irrational number, the components spread out increasingly widely as n increases. This is a useful class of spectra for mimicking metal bar percussion, where components tend to stretch wider than the integer harmonic sequence. Even though the components are inharmonic, their relative sparseness can still contribute a sense of pitch under the right conditions.

9.4.6 A Historical Note

Prior to the 1970s frequency modulation was used primarily in radio theory to characterize FM broadcast transmission. It was the inspiration of John Chowning to wonder what such spectra would sound like translated into the audio domain. Initially led to the phenomenon by a mistake in a sound synthesis program he had written, Chowning (1973) went on to develop and patent an astonishingly powerful application of FM synthesis of audio spectra that included realistic simulations of every major family of Western musical instrument tones. His work was extended in many directions, both auditory (Schottstaedt 1977) and compositional (Truax 1977; Dashow 1980). Chowning worked with the Yamaha corporation to develop the DX7 synthesizer, the first mass-produced all-digital synthesizer that used audio band FM synthesis.[7] F. R. Moore (1990) asserts that in terms of the numbers of units sold, the DX7 qualified at the time as the most popular keyboard instrument.

9.4.7 Angle Modulation

There is some controversy as to whether what Chowning invented was an application of frequency modulation or phase modulation (PM). In the language of communications theory, both frequency modulation and phase modulation are kinds of *angle modulation*, which is defined as a sinusoid with a time-varying frequency: $f(t) = A \sin \theta(t)$. In frequency modulation, the instantaneous frequency of the carrier is varied from its center frequency by an amount proportional to the instantaneous value of the modulating signal. In phase modulation, the phase of the carrier is controlled by the modulating waveform. Clearly, varying the phase shift of a carrier also modulates its frequency, so FM and PM are very closely related.

We know that instantaneous frequency is the derivative of phase. The instantaneous phase is given by $\theta(t) = \omega_c t + \phi(t)$, and the instantaneous frequency is given by

$$\omega(t) = \frac{d}{dt}\theta(t) = \omega_c + \frac{d}{dt}\phi(t).$$

If the frequency of $f(t)$ is proportional to phase deviation $\phi(t)$, we have *phase modulation*. If the frequency of $f(t)$ is proportional to the derivative of phase deviation, $\frac{d}{dt}\phi(t)$, we have *frequency modulation*. By these criteria, Chowning's method is clearly frequency modulation. Not that it matters much. Schottstaedt (2003) writes, "Of course, you can't tell which is in use either from the waveform or the mathematical expression of the waveform—you have to know what the modulating signal was. That is a roundabout way of saying that in computer music applications there is no essential difference between frequency and phase modulation."

9.4.8 A Few of Chowning's Examples

According to the foregoing theory of FM, we can characterize Chowning's synthesis technique as follows. His method was based upon a frequency modulation patch (see figure 9.28), which by itself produces only a static FM spectrum. To this, Chowning added dynamic amplitude control using an envelope generator (see figure 9.6), and dynamic control over the modulation index I, also using an envelope generator.

One of Chowning's best ideas was that by varying the modulation index I through time, the Bessel functions would cause the spectrum to evolve in interesting ways, and this could be controlled to provide a sense of aliveness that marked the sound of traditional acoustical instruments. Another idea was to exploit particular carrier/modulator frequency ratios to model the spectrum of particular instrument families, for example, using $H = 2$ for clarinets and irrational ratios for percussion.

His method consisted of artistic but reasoned choices for carrier-to-modulator frequency ratios, based on the spectrum of the instrument he was modeling, and suitable choice of amplitude envelope and modulation envelope parameters to make a convincing demonstration. He had available a great deal of practical information about the spectral ballistics of musical instruments from the work of Risset (1969) and Mathews at Bell Telephone Laboratories to help him.

A version of Chowning's classic FM instrument patch appears in figure 9.37. The principal parameters are

A, the peak amplitude

$\omega_c = 2\pi f_c$, the phase velocity of the carrier oscillator, calculated from the carrier frequency f_c

$\omega_m = 2\pi f_m$, the phase velocity of the modulating oscillator, calculated from the modulating frequency f_m

$\Delta f = I/f_m$, the peak frequency deviation, expressed as the ratio of modulation index and modulating frequency

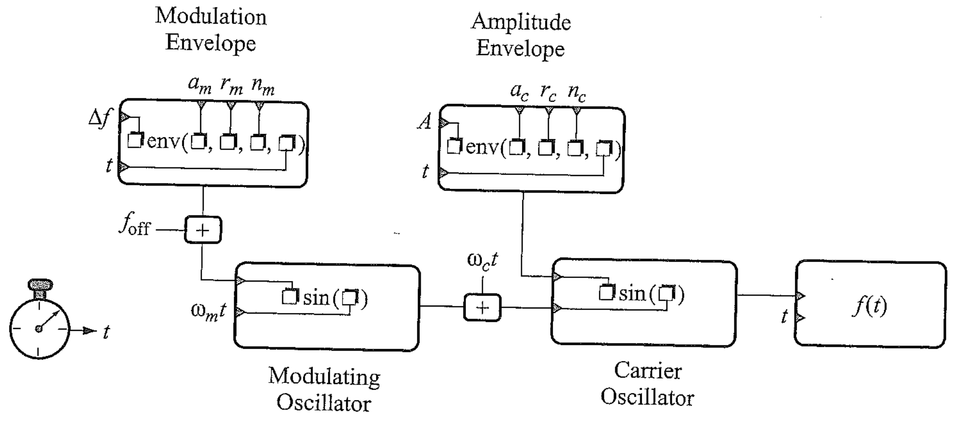


Figure 9.37
Dynamic frequency modulation patch.

$f_{\text{off}} = \alpha I/f_m$, where α is the initial or baseline frequency deviation (The idea is that f_{off} stipulates a minimum spectral brightness for a tone, and its maximum spectral brightness corresponds to $\Delta f + f_{\text{off}}$.)

a_m, r_m and n_m, attack time, release time, and sharpness of attack and release for the frequency deviation envelope

a_c, r_c and n_c, attack time, release time, and sharpness of attack and release for the amplitude envelope

Here are a few of the simulations Chowning created with this setup.

FM Trumpet Risset (1969) had shown that the spectral brightness of trumpets is proportional to their amplitude. Chowning simulated this by having the modulation envelope track the same envelope contour as the amplitude envelope. The spectrum of a trumpet is harmonic, with all components present, suggesting $H = 1$ would make a successful simulation. He made some artistic guesses about the shape of a trumpet amplitude envelope and the value for the peak modulation index that would sound trumpetlike, based on information from Risset, the acoustics literature, and his own ear. The result was his first successful simulation.

FM Bell Chowning knew that the amplitude and spectral brightness of bell tones diminishes exponentially over time, so he coupled the amplitude and modulation envelope trajectories. The spectrum is inharmonic, suggesting an irrational ratio for H such as $\sqrt{2}$ or e. He found various choices for the value for the peak modulation index and synthesized realistic simulations of gongs, vibraphones, and bells.

FM Bassoon Unlike the previous examples, the spectral evolution of the bassoon attack starts off mostly with high frequencies, then fills in lower frequencies as the tone becomes more solid. Chowning realized he should employ $H = 1/n$ for this timbre and make the carrier frequency high, so that as the modulation envelope grows, the effect the ear hears is of lower-frequency components entering after high-frequency components are already present.

FM Clarinet For the clarinet, Chowning used $H = 2$, resulting in a spectrum containing only odd harmonics. He adjusted the amplitude envelope to match a family of clarinet tones and adjusted the modulation index to match its overall spectrum.

FM Voice Chowning (1989) subsequently explored the use of FM to synthesize the singing voice. The most appropriate configuration he found was to have multiple carrier oscillators (typically three) driven by a common modulating oscillator, enabling him to construct rather arbitrary spectra. The frequency of the common modulator is set to the fundamental of the vocal tone to be synthesized. Each carrier frequency is set to whatever integer multiple of the modulating frequency places it nearest the center of each of the three primary vocal formants. Because of this approach, it is impossible to vary continuously between vowel sounds on the same pitch, or to vary the pitch keeping the same vowel. In spite of this limitation, the system was capable in Chowning's masterful hands of delivering compelling vocal synthesis.

Importance of Vibrato for Spectral Fusion Perhaps the most interesting result of Chowning's vocal synthesis system was to show the importance of vibrato (correlated perturbation of harmonic frequency) for the psychoacoustic fusion of vocal timbres. I heard him demonstrate this once. First he played the sound of a rich male voice synthesized with his system, singing an "ah" timbre, with vibrato. Then he played the same timbre without vibrato. I still recognized it as a male voice but no longer as rich. Then he applied a bell-like exponentially decaying envelope with a sharp attack to the timbre, again without vibrato. I thought I was hearing a bell tone; it did not sound like a voice at all.

Finally, he repeated the bell-like envelope with the same timbre, and again I heard a bell, but this time, half-way through, he switched the vibrato back on. I experienced a severe case of cognitive dissonance. I observed my perception do a surprising and quick readjustment from thinking I was hearing bells back to hearing the rich male vocal timbre. For every listener hearing the demonstration, the cognitive interpretation of a vocal timbre won out over the bell the moment the vibrato started.

He did another demonstration to a similar effect with two vocal synthesis instruments set to "sing" the interval of a major third, but without vibrato and using the bell envelope. The timbre started out sounding richly bell-like. The vibrato he subsequently turned on was correlated within each voice but uncorrelated between voices. The complex bell timbre vanished, replaced by two male voices singing. We are hard-wired, evidently, to fuse harmonics that vary according to a correlated time function of frequency. This is not too surprising, given the survival value of sound source separation and system identification that our ancestors must have evolved long ago.

9.4.9 Critique of FM Synthesis

Chowning pioneered the application of FM to audio band spectra. The technique is extremely efficient for digital sound synthesis and is flexible enough for a wide variety of compositional applications. Handled well, it can achieve astonishing effects inexpensively. But otherwise, FM synthesis sounds like . . . well, FM synthesis. The ear is a wonderfully adaptable organ, and our contemporary sonic culture now includes the characteristic burble of an FM synthesizer sweeping through the Bessel functions.

Since FM synthesis is a nonlinear synthesis technique, there is no formal analysis method that would allow us to exactly resynthesize an arbitrary timbre with FM. Using FM to model instrumental timbres is strictly an art.

9.4.10 Composing with FM

Some interesting uses of FM synthesis have a purely compositional motive. Just as composers organize pitch space into musical scales and chords, Barry Truax suggested compositional methods for organizing the carrier-to-modulating frequency ratio (the $c:m$ ratio). Besides the degree of harmonicity defined as $H = f_m/f_c$, Truax (1977) developed other ways to organize the $c:m$ ratio that would help project structure in music composed using FM sounds. His methods included "predicting the precise interval between the carrier and the actual fundamental and relating that interval to just or tempered scales, and predicting sets of $c:m$ ratios producing unique spectra and those producing exactly the same spectrum (i.e., the same set of sidebands)"(70).

The composer James Dashow (1980) has sought ways to treat spectra as musical chords, to convincingly combine inharmonic spectra with traditional performed musical instruments. One of his methods was to pick out, for instance, a pair of chromatic pitches (a diad) and specify them as anchor points in an inharmonic spectrum that would include them as components. If, for instance, the pitches were E♭ and A, he would additionally specify the position of these components in the resulting spectrum; they might be, for example, the 5th and 7th components of a spectrum created using FM synthesis. When sounded together with, for instance, a piano playing E♭ and A, the ear senses the spectral correspondence, and a pleasing kind of musical unity-in-diversity results.

9.4.11 Waveshaping

Waveshaping synthesis is an improvement over FM synthesis in that arbitrary spectra can be directly specified instead of being mandated by the shape of the Bessel function curves. In its simplest form, it only creates harmonic spectra, but it can be extended to create inharmonic spectra.

Waveshaping is also sometimes called *nonlinear distortion synthesis*. A flat mirror reflects light linearly, but a curved mirror distorts the image; waveshaping synthesis is like reflecting a sinusoid off a curved mirror.

We are accustomed to interpreting $\sin \omega t$ as a linear ramp function ωt used to index a sine function, as in figure 9.38a. Another perspective on waveshaping is to say that it turns the tables (so to speak) on the standard sine wave oscillator. Instead of using a ramp function to index the sine function, it uses the sine function to index a ramp function. As shown in figure 9.38b, function W is the identity function that linearly reflects the sinusoid indexing it. The equation for the output waveform is

$$s(t) = W(\sin \omega t). \tag{9.37}$$

What other possible functions are there for W? We can try out any arbitrary reflecting function to see what happens. Figure 9.39 shows a cosine function indexing several kinds of reflecting functions. In figure 9.39a, a cosine reflected by the identity function generates a cosine. In 9.39b, a bump in the identity function creates two corresponding bumps in the cosine wave. In 9.39c, two complementary ramps create two complementary cosines with pointy ends. In 9.39d, a parabola

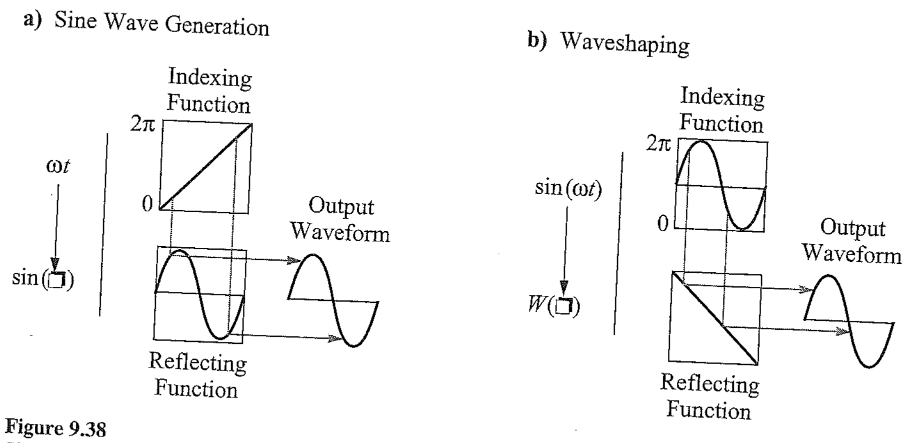


Figure 9.38
Sine wave generation and waveshaping.

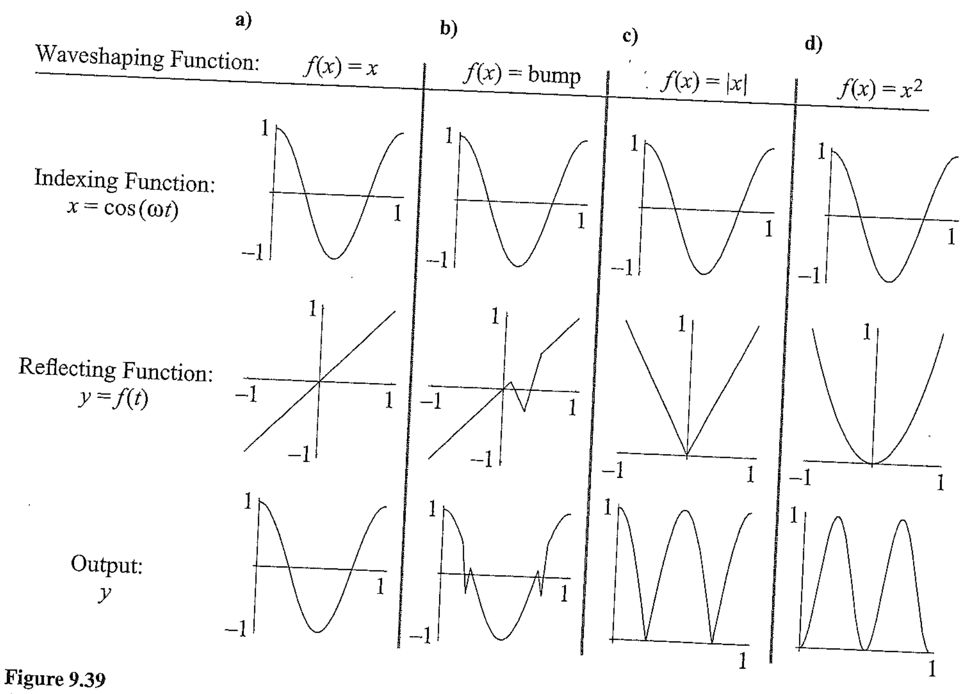


Figure 9.39
Arbitrary waveshaping functions.

warps the cosine wave into two periods of an inverted cosine wave. If it is not the identity function, then the reflecting function distorts the input waveform in a characteristic way. Any such distortion will introduce new spectral components.

Waveshaping as Function Composition Nonlinear waveshaping is an application of function composition. A *composable function* is one that can be another function's argument. For instance, if $y = f(x)$, and $z = g(y)$, then

$$z = g(f(x)) \qquad \textit{Function Composition} \quad (9.38)$$

is the composition of g with f because f is the argument to g.[8] For example, if $y = f(x) = x + 1$, and $z = g(y) = y^2$, then

$$z = g(f(x)) = g(x+1) = x^2 + 2x + 1.$$

We can see that waveshaping as defined by equation (9.37) is clearly based on function composition as follows. If in equation (9.38) we let $W = g$, and $f(x) = \sin \omega t$, then $g(f(x)) = W(\sin \omega t)$.

But frequency modulation can also be viewed as function composition. If $y = f(x) = cx + I \sin mx$, and $z = g(y) = a \sin y$, then

$$\begin{aligned} z &= g(f(x)) = g(cx + I \sin mx) \\ &= a \sin(cx + I \sin mx). \end{aligned} \qquad \textit{FM as the Composition of Functions} \quad (9.39)$$

Compare equation (9.39) to equation (9.31). So FM and waveshaping are closely related synthesis techniques.

Chebyshev Polynomial Reflecting Functions The Chebyshev[9] polynomials are a very useful class of reflecting functions. Their use in sound synthesis was discovered, virtually simultaneously, by LeBrun (1979) in the United States and Arfib (1979) in France. These functions have the following two remarkable properties when used as reflecting functions. By suitable addition of Chebyshev polynomial functions, a reflecting function can be constructed that produces any mixture of harmonics at any combination of strengths. Varying the amplitude of the indexing sinusoid varies the spectral brightness of the output of the Chebyshev polynomial reflecting function.

When the indexing sinusoid has low amplitude, its excursion covers only a small region at the center of the reflecting function and produces relatively simple harmonic spectra. But when its excursion covers the entire range of the reflecting function, the full harmonic spectrum coded in the function is produced. Thus, waveshaping has a capacity (like FM) to generate sounds with dynamic spectral evolution. But waveshaping (unlike FM) allows one to specify an arbitrary harmonic spectrum not limited to those provided by the Bessel functions.

In order to construct a set of known harmonic spectra, we want a set T_n of shaping functions that, when combined, produce the desired spectrum. (We choose T because one romanized version of Chebyshev's name begins with that letter.) Function T_0 should produce 0 Hz, T_1 should produce the fundamental, T_2 should produce the second partial, T_3 the third partial, and so on. We can define the first two functions trivially as follows:

$$T_0(x) = 1.$$

No matter what x is, the output is 1, so this produces 0 Hz.

$$T_1(x) = x.$$

This is the identity function. If x is sinusoidal, the output is identically sinusoidal.

We have T_0 and T_1, but what's the next function in the series? What we're looking for is a function that takes a sinusoidal index function like $\cos\theta$ and gives back the nth harmonic of θ. That is, we want a function T that works like this:

$$T_n(x) = T_n(\cos\theta) = \cos n\theta. \tag{9.40}$$

The solution arises in the context previously considered in ring modulation, which is the product of two sinusoids. Recall the handy trigonometric identity that defines the product of two cosines (see appendix section A.4.1):

$$\cos u \cdot \cos v = \frac{\cos(u+v) + \cos(u-v)}{2}.$$

If we set $u = n\theta$, and $v = \theta$, then this becomes

$$\cos n\theta \cdot \cos\theta = \frac{\cos(n\theta + \theta) + \cos(n\theta - \theta)}{2}$$

$$= \frac{\cos[(n+1)\theta] + \cos[(n-1)\theta]}{2}.$$

There are several interesting things in this equation. Notice on the left-hand side the term $\cos n\theta$. By equation (9.40) we can rewrite this as $T_n(\cos\theta)$. Notice on the right-hand side the terms $\cos[(n+1)\theta]$ and $\cos[(n-1)\theta]$. We can rewrite these as $T_{n+1}(\cos\theta)$ and $T_{n-1}(\cos\theta)$, respectively, yielding

$$T_n(\cos\theta) \cdot \cos\theta = \frac{T_{n+1}(\cos\theta) + T_{n-1}(\cos\theta)}{2}.$$

We can simplify this by letting $\cos\theta = x$, yielding

$$T_n(x) \cdot x = \frac{T_{n+1}(x) + T_{n-1}(x)}{2}.$$

This equation defines T_n in terms of the next function in the series, T_{n+1}, and the previous function in the series, T_{n-1}. If we solve it for T_{n+1}, we have

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \tag{9.41}$$

which is a recursive formula that defines the next function in terms of the current function and the previous function. This is just what we need because we have exactly two solutions at hand: T_0 and T_1. Setting $T_0 = T_{n-1}$ and $T_1 = T_n$, we can use equation (9.41) to find $T_{n+1} = T_2$. Then, having found T_2, we can continue this process recursively to produce T_3, T_4, and so on. The recursive solutions to equation (9.41) are called Chebyshev polynomials of the first kind.

Here's how we derive T_2 using equation (9.41):

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= 2x \cdot x - 1 = 2x^2 - 1. \end{aligned}$$

Similarly we derive T_3 from T_1 and T_2:

$$\begin{aligned} T_1(x) &= x, \\ T_2(x) &= 2x^2 - 1, \\ T_3(x) &= 2x \cdot (2x^2 - 1) - x = 4x^3 - 3x. \end{aligned}$$

The reader can derive T_4, ending up with $8x^4 - 8x^2 + 1$. This series of polynomials gets large and complex rather quickly. What does it have to do with sound synthesis? When we substitute $x = \cos\theta$ in these equations,

$$\begin{aligned} T_0(\cos\theta) &= 1, \\ T_1(\cos\theta) &= \cos\theta, \\ T_2(\cos\theta) &= 2\cos^2(\theta) - 1 \\ &= 2\left[\frac{\cos(\theta+\theta) + \cos(\theta-\theta)}{2}\right] - 1 \\ &= \cos 2\theta + \cos 0 - 1 \\ &= \cos 2\theta. \end{aligned}$$

Continuing this series, we obtain

$$\begin{aligned} T_3\cos\theta &= \cos 3\theta, \\ T_4\cos\theta &= \cos 4\theta, \\ T_5\cos\theta &= \cos 5\theta. \end{aligned}$$

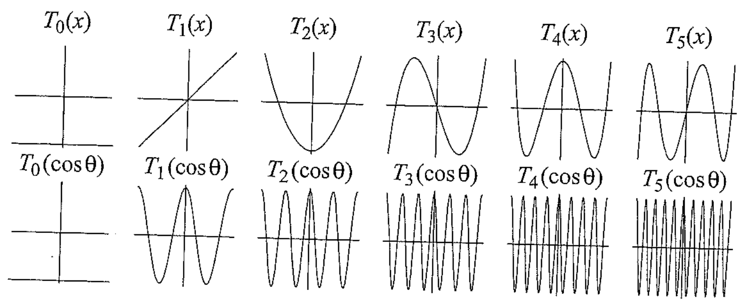


Figure 9.40
Chebyshev polynomial functions.

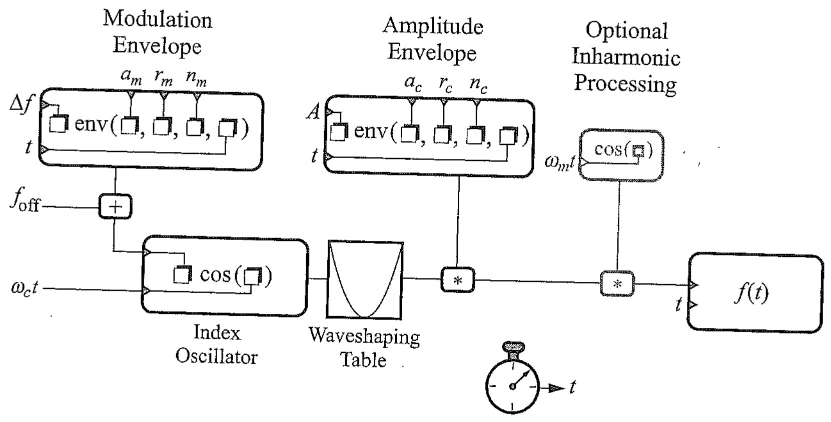


Figure 9.41
Dynamic waveshaping instrument patch.

Remarkably, the Chebyshev polynomial of order n produces the nth harmonic of the cosine index function. Figure 9.40 shows the first four Chebyshev polynomial functions and the corresponding output they produce when they are driven by $\cos\theta$. To achieve a composite spectrum, all we must do is to make a weighted sum of the desired T_n:

$$f(x) = \frac{h_0}{2} + \sum_{k=1}^{N} h_n T_n(x), \tag{9.42}$$

where h_n are the weights of the n components.

A synthesis patch for waveshaping comparable to the patch for FM is shown in figure 9.41. Like FM, the instrument has an amplitude envelope and a modulation envelope allowing amplitude and spectral content to evolve through the duration of a tone, providing a heightened sense of realism. Unlike FM, however, the spectral components are strictly harmonic multiples of the frequency of the index oscillator. If inharmonic spectra are desired, an optional processing step can be added, drawn with light lines in figure 9.41, that multiplies the spectrum of the output by another sinusoid, creating ring modulation (see section 9.3.2).

9.5 Vocal Synthesis

Synthesis is all about characterization and modeling: how to make something like something else. We synthesize either because the synthesized model provides some expressive power we don't otherwise have, or to emulate a sound for less than the cost of the original. For musical applications, we are typically more concerned with using synthesis to enhance expressive power.

Perhaps because the voice is such a rich, expressive sound source, there are many approaches to vocal synthesis. Most models begin by observing that the vocal tract resembles a tube about 17 cm long, made up of sections with varying diameters, connected together in series, and branching at the end toward the nasal tract (figure 9.42). At the head end of the tube, energy can escape through both nose and mouth.

The glottis provides a driving function at the lower end of the vocal tract that produces a broad-spectrum periodic impulse train during voiced speech. Because the glottis can adjust its frequency and amplitude, we can speak with inflection and sing. Figure 9.43 shows two periods of the waveform of the author singing the sound "ah." The impulsive nature of the glottal source is clearly visible.

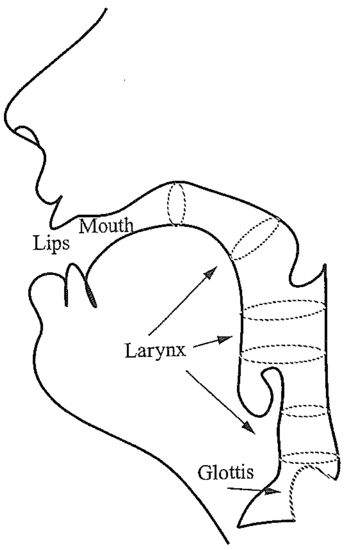


Figure 9.42
Vocal tract.

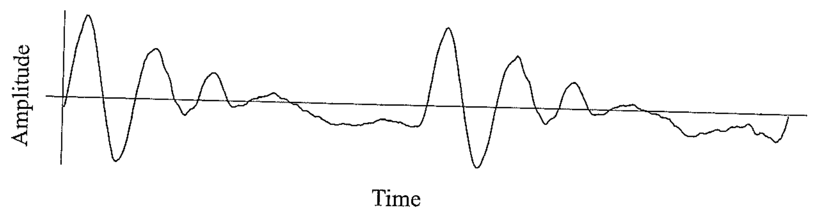


Figure 9.43
Two periods of a vocal waveform.

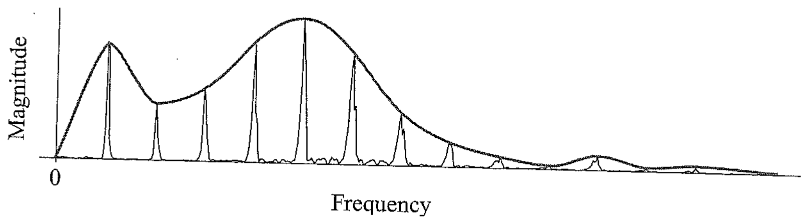


Figure 9.44
Spectrum of voice with formant regions.

Because some reflection occurs where the diameter of each tube section changes, each behaves like a Helmholtz resonator (see volume 1, section 8.3.3). The maxima of this resonant structure, called *formants,* are regions of the vocal frequency response that transmit energy very efficiently. Therefore, components that fall within formant regions tend to be emphasized in the resulting spectrum. The mouth can change its volume, and the lips can change their shape and aperture, allowing some formants to change their frequency and sharpness (see volume 1, equation 8.26). Other formants, corresponding to the inflexible segments, remain relatively fixed. Our hearing uses the formants to help identify speech. Figure 9.44 shows the spectrum of the voice signal in figure 9.43. The spectral envelope of the formants is outlined by the top curve. Three or four formants can be observed in this spectrum.

9.5.1 Waveform Synthesis

Perhaps the easiest way to model the spectrum of a vocal timbre is simply to extract one period of the waveform in figure 9.43 and make it into the wavetable of an oscillator. For example, start with the simple synthesis instrument shown in figure 9.8, but instead of a sine wave oscillator, substitute a table lookup oscillator (section 9.2.9), where the table is one period of figure 9.43. Care must be taken to ensure that there is no discontinuity at the ends of the table, or a click is heard. This can be accomplished by windowing the waveform with a smoothing function. The envelope and vibrato parameters can be adjusted to taste.

This synthesis technique is used by many commercial wavetable synthesizers. Although this simple technique can be refined a great deal and can provide realistic and interesting vocal timbres, the results are limited (see, for example, McNabb, 1981). First, the technique only produces vowels. Second, big changes to the oscillator's fundamental frequency appear to make the physical size of the speaker change. The reason is that as the frequency shifts, the entire spectrum of the sound is shifted linearly up and down in frequency, so the formants change as well. But our hearing equates low-pitched formants with a big person and high-pitched formants with a little person. To be effective over a wide range, many tables must be sampled from the vocalist over his or her entire range, and the nearest one must be selected for the desired fundamental frequency. Individual tables can only be transposed by a few semitones before the size-shifting psychoacoustic effect kicks in. We can do better.

9.5.2 Linear Prediction

Because vocal production depends upon current input (the glottal function) plus current and past outputs (caused by reflections at the mouth and between sections of the vocal tract), the effect of the vocal tract on the spectrum of speech can be modeled reasonably well as an IIR filtering process (see chapter 5).

To model the vocal formants as an IIR filter, we must discover the coefficients of a filter that matches their frequency response. We begin with a difference equation, a *linear predictor,* that expresses each new sample of the speech as a linear combination of previous samples. The coefficients of the linear predictor, the *prediction coefficients,* can be made to provide a highly accurate model of the formants. Estimating the coefficients requires us to solve a set of linear equations and then possibly to adjust the resulting data in order to obtain convergence to a unique and stable solution, but this will prove easier to do than it might sound. The result is a filter that matches the frequency response of the vocal tract. By taking such measurements periodically, the dynamic frequency response of the vocal tract can be characterized over time.

Having characterized the resonances of the vocal tract, the next step is to isolate the pure unfiltered sound of the glottis. To do so, we invert the frequency response of the vocal tract filter and drive it with the original speech sound that we are analyzing, thereby neutralizing the filtering effect of the vocal tract on the glottis. The unfiltered sound source is the *residue signal.* For vowels, the unfiltered glottal function looks like an impulse train and sounds like a buzzer. For consonants, the source sounds like a broadband noise.

We can resynthesize the original sound by driving the residue signal through the (uninverted) vocal tract filter. This analysis/synthesis method, called *linear predictive coding* (LPC), can perform very high-quality speech analysis and synthesis.

Why Do Linear Prediction? LPC's first application was in telecommunications, where the fundamental interest is to reproduce speech at a distance using a representation requiring the least bandwidth to transmit. The original goal was to reduce as much as possible the amount of data transmitted while maintaining intelligibility, and indeed one can achieve dramatic data reduction with LPC.[10]

However, we typically can't afford to suffer any loss of quality for musical applications.[11] For music, there is generally no point in directly resynthesizing the original sound. Why not just use the original? Therefore, LPC's musical usefulness lies in its ability to modify the sound, preferably in ways that are either difficult or impossible to do otherwise.

Musically, LPC can be used to model the speaking and singing voice, woodwind instruments, birds, whales, violins, and any resonant sound source. Since the LPC model separates the speech into a formant filter and a residue signal, we can alter the timing and pace of the analyzed speech or music or modify its spectral content. We can also cross the formant filters from one signal with any other sound to create hybrid sounds, called *cross-synthesis*. For instance, filtering the sound of a flute with the prediction coefficients of a speech segment creates the illusion of a talking flute.[12]

What Does Prediction Have to Do with Filtering? If we know the position and orientation of an airplane a moment ago, and also know how its controls are set, we can predict with fair certainty where it will be a moment from now. Similarly, the response of an IIR filter depends upon its current inputs and past outputs. If we know the coefficients of a filter and its initial conditions, we can predict its value in the future. But prediction implies uncertainty and therefore estimation. For every prediction, there will possibly be an error of some magnitude against the actual outcome.

Premise of LPC If we can predict a signal's behavior in the time domain, we can characterize its behavior in the frequency domain. To predict a signal's time domain behavior, we construct a filter with a response that closely matches the signal's time domain behavior. Since the Fourier transform of the filter's impulse response is its frequency response, the resulting filter characterizes the resonant properties of the signal.

To make this system work, we must have a method of constructing a filter that matches the time domain behavior of the analysis signal. With this in mind, let's consider just the IIR portion of equation (5.41):

$$y_n = x_n - \sum_{s=1}^{N} b_s y_{n-s}. \qquad \textit{IIR Filter as a Linear Predictor} \quad (9.43)$$

(The notation y_{n-s} is equivalent to $y(n-s)$.) The standard interpretation of this is that the filter reads input sample x_n and subtracts from it a weighted combination of past outputs to create the current output y_n. But we can also use this equation as a way to predict the next value of y_n from the previous N samples y_{n-s} for all integers $s = 1, \ldots, N$. In this interpretation, the term x_n is the *difference* between the *true value* y_n and the *predicted value* as indicated by the summation of past outputs in equation (9.43). Isolating x_n in equation (9.43),

$$x_n = y_n + \sum_{s=1}^{N} b_s y_{n-s},$$

we see that x_n is the amount by which the value y_n differs from what would be predicted by recent past outputs alone. A successful predictor would have $|x_n| << |y_n|$ for all n because this would indicate that when the *linear prediction coefficients* b_s are applied to the past outputs, they accurately predict future outputs. Overall, we seek the smallest error ε such that the mean squared error

$$\varepsilon = \sum_{n=0}^{N-1} x(n)^2$$

is as small as possible, given N, the length of the signal being analyzed.

Entropy, Redundancy, and Information If an outcome is *predictable* based on available information, then any additional information of the same kind is *redundant*. If the samples of $y(n)$ can be perfectly predicted from a weighted sum of the previous N samples, then any additional samples would supply no additional information. This redundancy in the signal is what the prediction coefficients are characterizing. If $y(n)$ can be perfectly predicted, that is, if $\varepsilon = 0$, then all we need in order to regenerate it are its prediction coefficients and its initial conditions.

For example, consider the transfer function of the two-pole filter, equation (5.96). Setting $z^{-x} = y(n-x)$, we can directly convert this into the filter equation:

$$y(n) = \delta(n) + 2R(\cos\theta)y(n-1) - R^2y(n-2)$$

where $\delta(n) = 1, n = 0$, else 0.

In terms of equation (9.43), the coefficients of this filter are $b_1 = 2R\cos\theta$ and $b_2 = R^2$. If we set the radius $R = 1$, then the effect is to move the poles of this filter onto the unit circle. If we supply this filter with initial conditions for $y(n-1)$ and $y(n-2)$ such as the values 1.0 and 0, the impulse response of the filter will ring forever, producing a pure sinusoid at frequency θ.

These two filter coefficients can be looked upon as perfect predictors of $y(n)$ because only the coefficients and the initial conditions are required to predict all possible values of $y(n)$. If we set $R < 1$, then $y(n)$ will describe a sinusoid at frequency θ with an exponentially decreasing amplitude. If $R > 1$, $y(n)$ will be a sinusoid at frequency θ but with exponentially increasing amplitude. All these functions can be perfectly predicted from the initial conditions and the two prediction coefficients. Since they perfectly characterize $y(n)$, we can replace $y(n)$ with the initial conditions plus the two prediction coefficients without any loss of information, thereby considerably compressing the amount of data required to characterize the signal.

Now consider a random sampled signal $u(n)$. If it is purely random, there is no way to predict its next value from any number of previous values—in fact, this is a good definition of randomness. There is no way we could compress this signal as we did $y(n)$ unless we constructed a filter that was as long as the signal. But since that would result in no data compression, what would be point? Using the terminology of information theory, we say that $u(n)$ has a high degree of entropy. Similarly, since we wouldn't be able to squeeze any additional redundancy out of the prediction coefficients for $y(n)$, they also have a high degree of entropy.

We can define *information* as the degree of entropy in a signal. We can define *entropy* as the number of states required to characterize a system.[13] We want the resonant properties of speech to have *minimum entropy* because then the states required to characterize the resonance will be merely a handful of prediction coefficients. We want the residue properties of speech to have *maximum entropy* because then we have made sure that everything that can be predicted is accounted for in the prediction coefficients.

In the case of the filter generating a sinusoid, there is no residue signal because there is no information in the signal that is not characterized perfectly in the prediction coefficients. But for real signals, there will always be a residue. The better we are at extracting redundancy into the prediction coefficients, the less predictable the residue becomes, and its entropy increases. The greater its entropy, the broader and more uniform (whiter) its spectrum becomes. So the result of successful linear prediction is that the residue has the flattest spectrum possible.

LPC is attractive in telecommunications because most acoustical signals have high redundancy and low entropy, allowing for a great deal of compression, thereby lowering communication costs. Since the spectrum of speech usually does not change much in 30 ms, we can reduce that 30 ms of sound down to a handful of prediction coefficients plus the residue signal. The magnitude of the residue signal is generally rather small, and for typical speech applications it can be made to fit into 8 bits per sample. This way, intelligible speech can be transmitted at a rate of about 5000 bits per second or less.

The reason LPC is attractive to musicians has more to do with how it divides a signal into an all-pole filter and a residual signal. For cross-synthesis, the residual is not used and needn't be computed because the filter will be applied to an entirely different sound.

Estimating the Prediction Coefficients To make this scheme work, we must have a means of identifying prediction coefficients that squeezes the maximum amount of entropy out of the signal. Let's think about what equation (9.43) is describing. According to the discussion of all-pole filters in chapter 5, equation (9.43) specifies some sort of complex spectral function in the z plane whose frequency response is defined by its N conjugate pole pairs.

We can observe what kind of spectrum a set of coefficients will generate by looking at their Z transform, as defined by equation (5.49). The spectral estimate for a sampled function s_k can be written

$$H(z) = \sum_{k=0}^{N-1} s_k z^k . \tag{9.44}$$

We know from equation (5.99) that the Z transform of equation (9.43) is

$$P(z) = \frac{a}{1+\sum_{n=1}^{N} b_n z^{-n}} . \qquad \textit{All-Poles Model} \tag{9.45}$$

The difference between equations (9.44) and (9.45) is that whereas $H(z)$ can have only zeros of transmission, $P(z)$ can have only poles, corresponding to infinite spectral density at their centers. Filters with poles are good at modeling spectra that have sharp, discrete lines (Dirac delta functions), such as the voice. It is much harder to model the spectral signatures of the voice with filters containing only zeros, so equation (9.45) is generally used. It's called the *all-poles model.*

The Wiener-Khinchin theorem (equation 4.26) states that the Fourier transform of the autocorrelation of a function f is equal to the power spectrum of that function: $F\{\mathrm{corr}(f,f)\} = |F(\omega)|^2$.

That means we have two ways in which to describe spectra: equation (9.45) describes the spectrum of an N-pole filter in terms of its coefficients in the z plane, and equation (4.26) describes the power spectrum of an arbitrary signal in terms of the autocorrelation of a sampled time function. Let's relate these two views.

First, let's convert the all-poles model, equation (9.45), into a power spectrum by taking the absolute value of the square of both sides:

$$|P(z)|^2 = \left|\frac{a}{1+\sum_{n=1}^{N} b_n z^{-n}}\right|^2.$$

Next, remember from equation (4.25) that autocorrelation is written $\phi_n = \mathrm{corr}(f,f)(n)$, and by the Wiener-Khinchin theorem, we know that the Fourier transform of ϕ_n is equal to the power spectrum $|P(z)|^2$. But the Fourier transform is just the Z transform evaluated on the unit circle with $z = e^{i\omega}$. That means we can express the power spectrum of ϕ_n as a Z transform:

$$\Phi_n = \sum_{n=0}^{N-1} \phi_n z^{-n}.$$

Now we have a power spectrum $|P(z)|^2$ specified in terms of the Z transform of the prediction coefficients that we want to discover, and a power spectrum Φ_n specified in terms of the Z transform of the autocorrelation of the signal we want to analyze. When we relate them, $|P(z)|^2 \approx \Phi_n$, we have

$$\left|\frac{a}{1+\sum_{n=1}^{N} b_n z^{-n}}\right|^2 \approx \sum_{n=-N}^{N} \phi_n z^{-n}. \tag{9.46}$$

The left-hand side is the Z transform of the filter we want to design, and the right-hand side is the Z transform of the autocorrelation of the waveform we are trying to match. The value N is both the number of coefficients we want to discover and the extent of the autocorrelation function we will use to find them. Thus, N corresponds to the order of the filter we are seeking. Though we can set N to as high as the total number of autocorrelations available, in practice we want it to be much smaller.

The solution to equation (9.46) can be shown to have the maximum possible entropy of all possible solutions; it is therefore called the *maximum entropy method* (MEM). This equation implies that there is a linear set of relations between the autocorrelation function and the prediction coefficients. To actually solve it requires solving a set of simultaneous equations based on the series expansions of both sides. It can be shown that the filter coefficients satisfy the matrix equation

$$\begin{bmatrix} \phi_0 & \phi_1 & \phi_2 & \cdots & \phi_N \\ \phi_1 & \phi_2 & \phi_3 & \ddots & \phi_{N-1} \\ \phi_2 & \phi_3 & \phi_4 & & \phi_{N-2} \\ \vdots & & & & \vdots \\ \phi_N & \phi_{N-1} & \phi_{N-2} & \cdots & \phi_0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ b_0 \\ b_1 \\ \vdots \\ b_N \end{bmatrix} = \begin{bmatrix} a_0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \qquad (9.47)$$

The matrix form is just a notational convenience. To write out the equations to be solved simultaneously in standard polynomial form would require much more ink. We'd have to write

$$\begin{aligned} (\phi_0 \cdot 1) + (\phi_1 \cdot b_1) + (\phi_2 \cdot b_2) + \cdots + (\phi_N \cdot b_N) &= a_0, \\ (\phi_1 \cdot b_1) + (\phi_2 \cdot b_2) + (\phi_3 \cdot b_3) + \cdots + (\phi_{N-1} \cdot b_{N-1}) &= 0, \\ \vdots \qquad & \quad \vdots \\ (\phi_N \cdot b_N) + (\phi_{N-1} \cdot b_{N-1}) + (\phi_{N-2} \cdot b_{N-2}) + \cdots + (\phi_0 \cdot b_0) &= 0, \end{aligned}$$

and so on. Fortunately, because equation (9.47) is constant along its diagonals, it is a so-called *Toeplitz matrix,* and some very efficient algorithms have been developed to solve them on a computer (Press et al. 1988; Makhoul 1975; Rabiner and Schafer 1978).

Using LPC Generally, we want to set N, the order of the predictor, to a value somewhat higher than the number of sharp spectral features we wish to discover. Though we can set it as high as the number of autocorrelations available, numerical instabilities can arise. Depending on the nature of the signal, a surplus of poles can lead to false spectral peaks. By limiting the number of poles, the spectrum may be smoothed out somewhat, but this is generally a good thing.

The difference between the LPC analysis of a signal and that provided by the DFT is that the LPC spectrum is continuous in frequency whereas the DFT is sampled in frequency. One consequence of this is that the estimated LPC spectrum may have very sharp spectral features that can be overlooked if the spectrum is not evaluated finely enough. One possible way to properly direct the MEM algorithm is to take the DFT of the signal first, then compare it to the spectral features derived by the MEM, using the DFT as a guide as to which spectral features may be spurious.

Another problem with LPC has to do with the stability of the filters derived by the MEM. If, for instance, a component is increasing in amplitude during an analysis frame, MEM will turn that into an unstable recursive filter in order to correctly model its increase in gain. The instability, if mishandled by the LPC analysis/synthesis system, can result in some very loud, very unpleasant sounds that are bad for loudspeakers and listeners alike. The MEM algorithm can be quirky, and under the right conditions it is perfectly capable of putting poles of the filter far outside the unit circle all on its own. If the arithmetic precision of the computer implementing the filter is not sufficient, or if there are other subtle numerical problems, dreadful, deafening howling noises can ensue.

To avoid these problems, we have basically two choices (F. R. Moore 1990):

- Ignore the problems, and hope for the best. This becomes a losing strategy as the number of samples in the analysis frame grows because we'll be running potentially unstable filters for a longer time, increasing the risk of insufficient internal arithmetic precision in the filter. Our chances get slimmer as we either lengthen the analysis frame size or increase the sampling rate (or both).

- Reduce the magnitude of the poles. If we reduce an unstable pole's magnitude so it lies on the unit circle, it goes from an exponentially increasing sinusoid to a constant-amplitude sinusoid. If we bring it inside the unit circle along its radius, it becomes an exponentially decreasing sinusoid. The choice we make depends upon what we want. If it seems that the sound is relatively steady-state, maybe putting the unstable poles on the unit circle is the right thing. If we think the unstable poles are the result of quirky MEM behavior, we can move them inside the unit circle. Or we can kill them off altogether by moving them all the way inside to a magnitude of zero.

LPC as a Data Reduction System A drastic way to economize on transmitted information is actually to discard the residue signal altogether. For vowels, the wave shape of the residue buzz signal remains relatively constant, varying mostly in frequency and amplitude. For consonants, the residue noise signal is spectrally very bright and mostly only varies in amplitude. If the source is a vowel, we can transmit just its frequency and amplitude parameters; if it is a consonant, we just send its amplitude. The parameter that indicates whether the synthesizer should produce noise instead of buzz is called the voicing parameter. These parameters are transmitted along with the prediction coefficients to the synthesizer, which first reconstructs the residue signal, then filters it. The result sounds quite mechanical, but is surprisingly intelligible and highly efficient to implement. This is the basis of the LPC-10e algorithm described in U.S. Federal Standard 1015.[14]

Better-quality speech that does not require much additional transmission bandwidth uses a *codebook,* which is a table of frequently encountered residue signals set up in advance. The analyzer compares the character of the residue signal to the signals in the codebook, choosing the best match using a least-squares fit, then transmits just the index code for this signal. The synthesizer receives the code index, looks up the corresponding residue signal in its identical codebook, and regenerates the signal. This is called Code Excited Linear Prediction (CELP).

The bigger the codebook, the better this approach can model the original speech. But the bigger the codebook, the longer it takes to search for the best match. If the codebook must also take frequency and amplitude information into account, it would have to be quite large. One common way to address this is to have two codebooks, the first of which contains prototype residue signals set up in advance. The second starts off empty and is used during operation as a kind of scratch-pad memory, containing copies of the previous residue signals delayed by an amount that matches the frequency of the signal being encoded. This is the CELP algorithm described in U.S. Federal Standard 1016.[15]

A Caveat LPC succeeds best with sounds that are well modeled as all-pole resonators. The voice is not a pure IIR process because it includes side branches—principally the nasal passages. Side branches introduce zeros in the transfer function of the vocal tract that can't be easily modeled with an all-pole IIR filter. Thus LPC does not model nasal sounds well. More precisely, the spectrum of nasal sounds is not well captured in the prediction coefficients. As a consequence, the nasal components of the spectrum remain in the residue signal. A more complicated approach (based on equation 5.41) is required to model both poles and zeros, the pursuit of which is left to that admirable personage, the interested reader.

9.5.3 FOF Synthesis

FOF synthesis demonstrates a clever use of time/frequency domain symmetry to simulate spectral features of the singing voice and other sounds. It was developed by Rodet (1984).

Each *formant wave function* (*Forme d'Onde Formantique,* or FOF) consists of a windowed sinusoid with a frequency corresponding to the center of one of the vocal formants. By adjusting the shape of the envelope, the spectral bandwidth of each FOF can be adjusted to match the spectral shape of a vocal formant. Although the technique can model many timbres, it excels at modeling vocal resonances.

As shown in figure 9.45, a FOF is a single momentary pulse. To create a continuous tone, a train of FOFs must be generated at a rate corresponding to the desired fundamental frequency. To simulate the principal vocal formants of the voice, between three and five trains of FOFs are generated and summed (figure 9.46).

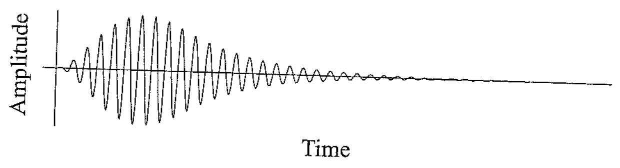


Figure 9.45
Single FOF.

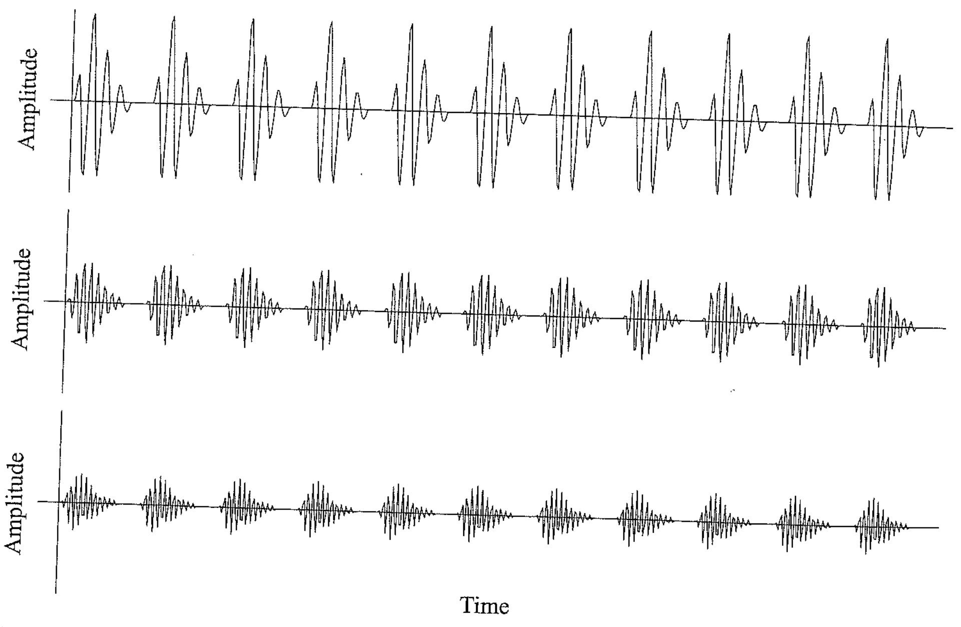


Figure 9.46
Train of FOFs triggered at fundamental frequency.

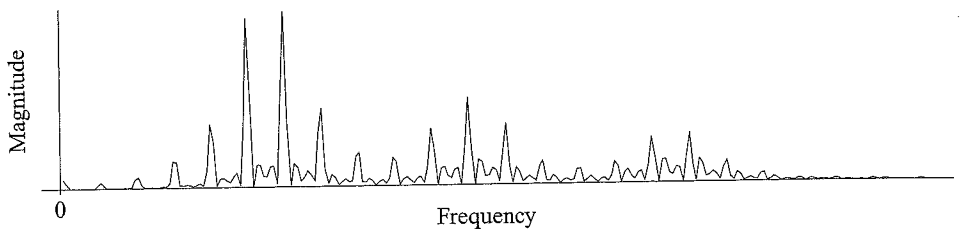


Figure 9.47
Spectrum of figure 9.46.

Figure 9.47 shows the positive frequency magnitude spectrum of the sum of the FOFs in figure 9.46. It has a voicelike spectrum with three formants.

FOF Generation A FOF generator consists of an amplitude envelope generator similar to equation (9.2), controlling the amplitude of a sine wave generator, similar to figure 9.6. The FOF amplitude envelope generator determines the shape of the window applied to the sinusoid.

We have seen the effects of multiplying a sinusoid and a window function in numerous contexts. Multiplying in the time domain convolves in the spectral domain, and so the shape of the window determines the shape of the spectrum of the product. We've seen that windows with sharp edges splatter energy over a broad range of frequencies, whereas smoother windows only slightly broaden the bandwidth of the signal they are multiplied by. FOF synthesis exploits this insight.

Amplitude Envelope Generation The amplitude envelope for a single FOF generator is

$$A(\beta, \alpha, n) = \begin{cases} n < 0, & 0 \\ n < \pi/\beta, & \frac{1}{2}(1 - \cos\beta n)e^{-\alpha n} \quad \text{Attack} \\ n \geq \pi/\beta, & e^{-\alpha n} \quad \text{Decay} \end{cases} \tag{9.48}$$

where n is time in samples. The basic envelope is the exponential function $e^{-\alpha n}$ with a decay rate controlled by α. The attack duration lasts π/β samples. During this time, the function $0.5(1 - \cos \beta n)$ smooths the sharp discontinuity at the beginning of the exponential.

The cosine expression for the attack during $n < \pi/\beta$ is shown in figure 9.48a. The exponential component is shown in 9.48b. Their product is shown in 9.48c. The final decay of the exponent is shown in 9.48d, and the composite envelope is shown in 9.48e.

Figure 9.49 shows how the shape of the envelope sharpens as π/β becomes smaller with α fixed. There is no first- or second-order discontinuity at the end of the attack. Since β determines the impulsiveness of the attack, it primarily determines how broadly energy is distributed in the spectrum.

Figure 9.50 shows the effect of α for a fixed setting of β. As α increases, the decay becomes more rapid. Thus, α also has an impact on spectrum, but not as dramatic as β.

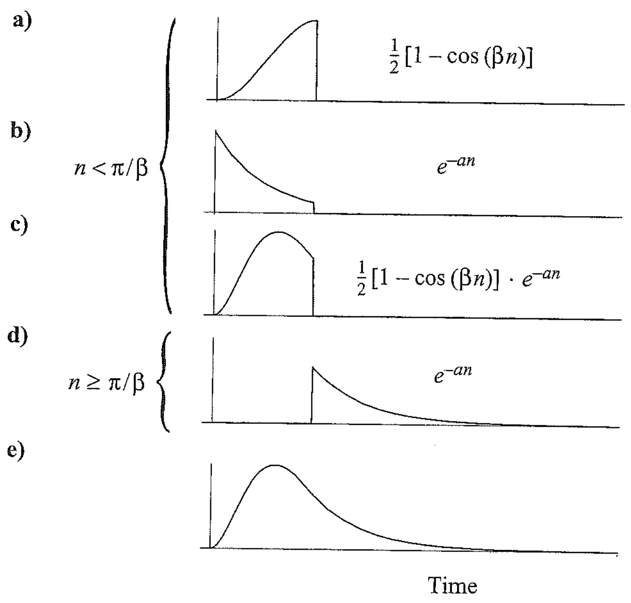


Figure 9.48
FOF envelope components.

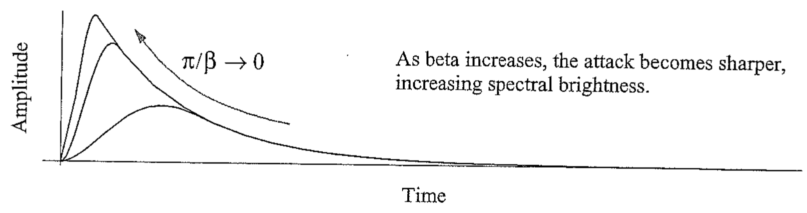


Figure 9.49
FOF envelope, increasing *β*.

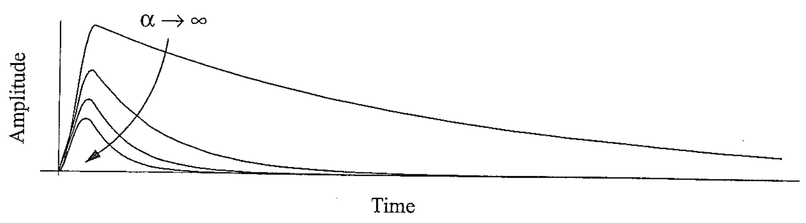


Figure 9.50
FOF envelope, increasing *α*.

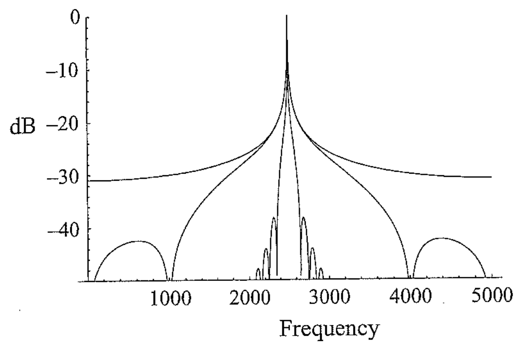


Figure 9.51
FOF spectral brightness as a function of β.

Note also in both figures 9.49 and 9.50 that peak amplitude changes as a function of α and β. In a later step, we will normalize the amplitudes.

FOF Sine Wave Generator Combining the envelope and the sinusoid generator, we have the complete FOF tone generator:

$$s_n(\beta, \alpha, \omega, \phi) = \begin{cases} n < 0, & 0 \\ n < \pi/\beta, & \frac{1}{2}(1 - \cos\beta t)e^{-an}\sin(\omega n + \phi) \\ n \geq \pi/\beta, & e^{-an}\sin(\omega n + \phi) \end{cases} \quad \textit{Forme d'Onde Formantique (FOF)} \quad (9.49)$$

The FOF envelope function scales the sinusoid $\sin(\omega n + \phi)$, where $\omega = 2\pi f$, and f is the center frequency of the formant being synthesized. No filtering occurs here, yet each FOF generates a single impulse with a spectrum that can mimic the frequency response of one vocal formant.

Figure 9.51 shows the magnitude spectrum for various values of β. Large values of β set the narrowest skirts because the attack time is slowest. From narrowest to widest, the attack times correspond to $\pi/\beta = 10$ ms, 1 ms, and 0.1 ms, respectively, with a formant center frequency of 2500 Hz and a sampling rate of 10,000 Hz.

Cascading FOFs To simulate a voice containing three to five formants, we must run three to five trains of FOF synthesizers in parallel (figure 9.52). Each FOF is triggered at a rate corresponding to the fundamental frequency at times τ_n. The bandwidth of each formant range is controlled by β_n, and the band center is controlled by ω_n.

Summary of FOF Synthesis FOF synthesis is computationally less challenging than filter-based approaches to speech synthesis such as LPC. This provides a number of advantages, including simplified arithmetic, no fear that unstable filters will make ballistic projectiles out of the loudspeaker cones, and relatively simple calculation of control parameters. It has been used to synthesize

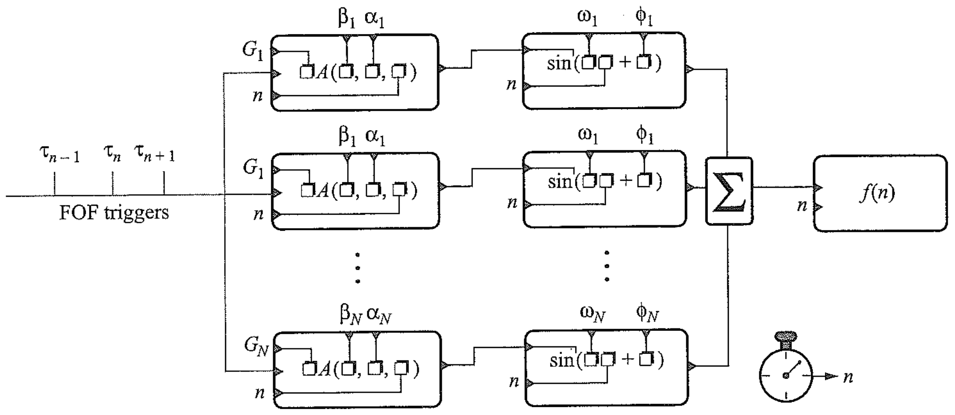

β_1 α_1
G_1
$A(\square, \square, \square)$
n
ω_1 ϕ_1
$\sin(\square\square + \square)$
n
τ_{n-1} τ_n τ_{n+1}
FOF triggers
β_1 α_1
G_1
$A(\square, \square, \square)$
n
ω_1 ϕ_1
$\sin(\square\square + \square)$
n
Σ
n
$f(n)$
β_N α_N
G_N
$A(\square, \square, \square)$
n
ω_N ϕ_N
$\sin(\square\square + \square)$
n
n

Figure 9.52
FOF synthesis of multiple formants.

a great number of vocal, instrumental, and other sounds, including violins and nonresonant instruments such as cymbals.

Its principal disadvantages over LPC are twofold. The FOF model does not include nonvoiced speech, so it can't easily be made to talk. Also, the parameter adjustments required to produce realistic-sounding vocal timbres are quite complex, depending on the timbre of the voice being synthesized, its vocal register, volume, the quality of vibrato, and so on. For analysis-based synthesis such as LPC, the model of the vocal timbre is implicit in the analyzed voice. For FOF synthesis, the rules governing vocal timbre must be made explicit. The rules of speech and singing have been studied extensively by Sundberg (1991), among others.

Rodet, Potard, and Barrière (1984) constructed a program called CHANT (French for *song*), which embeds a set of rules for modeling general vocal synthesis with FOFs. A user of this system supplies high-level parameters, and CHANT controls the FOF synthesis. A more ambitious program called FORMES, based on the Lisp programming language, was written to extend CHANT into a completely general compositional programming environment for FOF synthesis.

9.5.4 Granular Synthesis

FOF synthesis creates complex timbres by combining a sequence of many simple individual sound impulses, each having a characteristic envelope and frequency (see section 9.5.3). If we think of these individual sound impulses as grains of sound, we can then describe FOF synthesis is a kind of *granular synthesis*. This in turn suggests that there could be other forms of granular synthesis, and indeed there are many.

The idea of granular synthesis stems from papers published by Dennis Gabor in the late 1940s. Chapter 10 examines Gabor's ideas in more detail, but for now suffice it to say that in general a grain can be literally any sound, although by convention grains are usually brief, typically on the order of a few milliseconds in duration.

As with FOF synthesis, grains can be windowed sinusoids, or they can be prerecorded sounds, or sounds of any sort whatsoever. Each grain can be thought of as a kind of musical note in the technical sense of that term developed in the discussion of tones, notes, and scores (see volume 1, section 2.2), except that the grains are so brief that they are more like note fragments: hundreds or thousands must be strung together to last long enough to make an audible sound. (Gabor 1946).[16]

Gabor's Kinematical Frequency Converter In addition to setting out the theory of granular synthesis, Gabor (1946) invented a device based on his theory that was capable of changing the time scale of a sound (called time dilation) without changing its pitch, or vice versa. He called this invention the Kinematical Frequency Converter.[17] It was based on a modification of the optical audio track of a film projector.

Movie cameras of his day encoded audio waveform fluctuations as a continuous track on the edge of the film. The most positive amplitude was encoded as transparent, the negative amplitude as black, and the intermediate amplitudes as various shades of grey. The audio track of the film was passed over a narrow slit between an exciter lamp and a photocell. As the audio encoded on the film moved past the slit, light intensity from the bulb was modulated by the relative transparency of the audio track on the film. Variations in light striking the photocell produced an electrical analog of the audio signal encoded on the film, which was then amplified so it could drive a loudspeaker in the theater.

Gabor adapted this as shown in figure 9.53. He added a slotted drum that could rotate at an independent velocity. Each slot would sweep light across the film in the window, thereby projecting onto the photocell the segment of the audio captured in the window. To blend successive sweeps together, Gabor progressively shaded the window so that it was transparent in the center and opaque at its edges, using a Gaussian density distribution. Gabor reported best results when the slots were separated by approximately one half of the window's width.

To get a feel for how it worked, suppose the film is stationary while the wheel rotates. Each slot sweeps over the same area of film, so we hear pulses of sound whose character is determined by the audio encoded on the film that lies within the window. If the window is positioned over a vowel sound such as "a", we would hear "aaaa. . .".

Now suppose the film travels at its normal rate while the wheel rotates at a fixed rate so that we hear the audio as it was originally recorded. But if we now slow the film speed a bit, the pitch of the audio remains the same, but the *rate of spectral evolution* of the sound that we hear is slowed. This is because the rate at which the slots sweep across the audio in the window determines the frequency of audio pulses, but the rate of film movement determines the rate at which new audio material moves into the window. Hence, film speed now only determines the rate of spectral evolution, not frequency. Holding the wheel radial velocity constant while the film moves at a varying rate accomplishes change in tempo without change in frequency.

We can use Gabor's device together with a variable-speed audio recorder to change frequency without changing tempo. Suppose we set the film speed to half the speed of the original

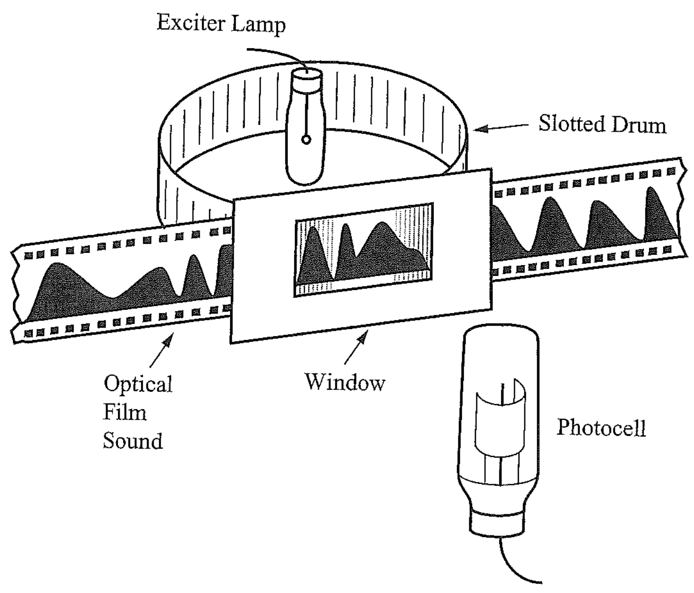


Figure 9.53
Gabor's kinematical frequency converter. From Gabor (1946, 446).

recording. Thus it will take twice as long to play the whole film, though we hear the audio at its original frequency. If we record the output to a machine running at half speed, then play it back at full speed, we will hear the audio transposed up an octave but lasting the length of the original recording. In this way, Gabor was able to demonstrate for the first time independent control of tempo and pitch.

The performance of Gabor's machine was not ideal. The apparatus could produce strong beats depending on the number of slots, their rate, the size of the window, and other factors. He eventually developed an improved version that used magnetic tape. A modern implementation of Gabor's technique, called the phase vocoder, is discussed in chapter 10.

Composers of Granular Synthesis The pioneer of granular synthesis techniques in music was Iannis Xenakis. From exposure to Gabor's work, Xenakis (1971) discovered that "all sound is conceived as an assemblage of a large number of elementary grains adequately disposed in time." He used analog oscillators and tape-splicing techniques to assemble his composition *Analogique A-B* for string orchestra and tape in 1959. Whereas Gabor used grains to modify the time/rate information of existing audio signals, Xenakis did not start with an existing sound. Instead, he proposed a sound grid consisting of successive *screens* of amplitude frequency functions describing the distribution of sound grains at discrete moments through time. He explored musical strategies for filling the screens with clouds of grains. He primarily used stochastic techniques (see volume 1, section 9.13) for his compositional strategies.

Given the small size of the sound grains, the biggest challenge to their use is organizing their distribution in time, frequency, and intensity. The possible organizing principles are seemingly limitless. Rigorous exploration of the possible compositional methodologies had to wait until the common availability of digital computers. Many composers have explored granular synthesis. The composer Curtis Roads (1988) has developed the theory and practice of granular synthesis since 1974 and has continued to research techniques and compose music utilizing granular synthesis. Truax (1988) implemented a real-time granular synthesis system in Canada in the late 1970s and early 1980s. Many implementations of granular synthesis are now commonly available, for example, on the World Wide Web.

Granular synthesis is reminiscent of the post-Impressionist pointillistic painting technique exemplified by the painter Georges Seurat. Other painterly analogies include air-brushing or stenciling. Granular synthesis can also be compared to collage art, in the sense that it starts with ready-made objects (the grains) and composes them into a pattern or skein. This suggests that the principal dimensions of the process are the morphology of grains and their disposition in time and frequency. The interested reader is referred to Roads (2001), whose writings on the subject are definitive and highly recommended.

9.6 Synthesizing Concert Hall Acoustics

Since sound travels at a constant rate, a sound introduced at one end of a tube will propagate to the other end after a constant delay (figure 9.54).

A *delay line* is a simple functional unit that can be used to simulate a sampled traveling wave in an acoustical tube (figure 9.55). We use the notation z^{-N} to indicate a delay of N samples through a delay line (see section 5.11).

It is natural and convenient to think of delay lines as building blocks for acoustic spaces such as concert halls and amphitheaters because the effects of propagation delay in these contexts can be understood as echoes and reverberation. They can also be used as building blocks for digital simulation of physical models of musical instruments. The treatment here prepares the way for a discussion of waveguide models of musical instruments.

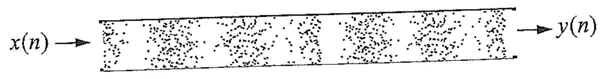


Figure 9.54
Wave propagating in a tube.

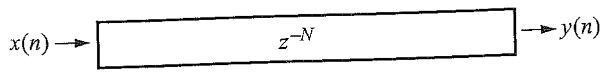


Figure 9.55
N-sample delay line.

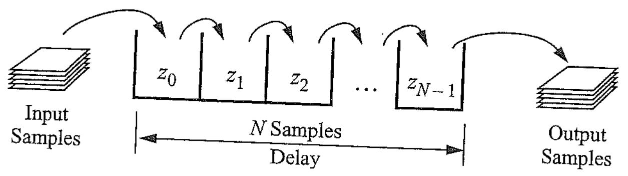


Figure 9.56
Bucket brigade.

9.6.1 Delay Lines

Here are two ways to implement a delay line.

Bucket Brigade Imagine an array of N empty storage bins with a courier who can access them sequentially. As the courier walks past them right to left, he advances each previous sample to each subsequent bin.

Suppose there are N bins labeled $z_0, z_2, \ldots, z_{N-1}$ (figure 9.56). First, the courier removes the sample from bin z_{N-1} and puts it on the Output stack. Next, he removes the sample in z_{N-2} and moves it to bin z_{N-1}. He continues, advancing each previous sample to reside in its subsequent bin. When he advances the contents of bin z_0, he takes a sample from the Input stack and puts it in z_0. Then he returns immediately to the last bin (z_{N-1}) and repeats the whole procedure until all samples have been transferred to the Output stack. The data move through a sequence of bins like a bucket brigade moves a series of buckets from person to person.

The total delay time is the time it takes the courier to move one sample multiplied by N, the number of bins. If the sample interval is T seconds per sample, the total delay time is NT seconds.

In-place Method A simpler approach leaves the data in place. As before, we have an Input stack, an Output stack, and an array of storage bins. For each bin, the courier removes the oldest sample from its bin and puts it on the Output stack, then replaces it with a sample from the Input stack, then moves on to the next oldest bin. The courier will have visited all the other bins before returning to the one he started with, by which time the sample just placed there will have been delayed by N sample times. As before, if the sample interval is T seconds per sample, the total delay time is NT seconds.

Let D be a delay line of length N, i the integer index of the current bin, x the next sample from the Input stack, and y the delayed sample headed for the Output stack. Then we can write the procedure for an in-place delay line as follows:

$y = D(i)$	Remove sample from bin indexed by i; send it to Output stack.
$D(i) = x$	Insert next value from Input stack into the now empty bin.
$i = ((i+1))_N$	Increment i, wrapping around to beginning if it goes off the end.
Return to step 1.	Repeat until there is no more input.

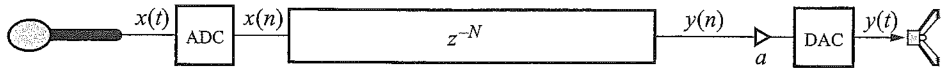


Figure 9.57
Digital delay line.

9.6.2 Modeling Echoes

If the delay line inputs x arrive sequentially from a microphone via an analog-to-digital converter (ADC), and the outputs y are passed sequentially to a digital-to-analog converter (DAC) and then to an amplifier and loudspeaker, the sound from the loudspeaker will be a copy of the input, delayed by N samples (figure 9.57).

Acoustically, this is similar to the echo we might hear standing some distance from a large wall in an open space. So long as the acoustical delay time is great enough to exceed the threshold of the precedence effect (see volume 1, section 6.13.3), if we clap our hands, we will hear a single sharp reflection called a *slap echo* from the wall. Figure 9.57 is an essential building block for commercial delay line audio effects. Its operation is similar to the simple tape delay shown at the beginning of chapter 5.

9.6.3 Modeling Traveling Waves in Air

Note the term a just before the DAC in figure 9.57. We can use it to attenuate the signal, modeling the inverse square law of distance for a spherical wave, so that the delay line can model sound coming from a distance. The proper attenuation value for a depends upon knowing the total delay time d. If a delay line is N samples long and the sample period is T seconds, then the delay time is $d = NT$ seconds. Recalling that the pressure of a spherical waveform drops off as $1/d$, we can model the attenuation of a pressure wave by setting $a = 1/(NT)$. Since intensity drops off as $1/d^2$, we can model the attenuation of intensity by setting $a = 1/(NT)^2$.

Air absorbs varying amounts of high-frequency energy as sound travels a unit distance, depending mostly on humidity but also on temperature and pressure. We can simulate this effect of humidity by adding a lowpass filter to the output of the delay line, and calibrate its attenuation of high frequencies due to humidity conditions from tables in acoustics texts. In practice, the exact attenuation values are less important than the fact that increasingly distant sounds become progressively more muffled.

9.6.4 Multipath Wave Propagation—Comb Filtering

Suppose we have a flutist and a microphone separated by some distance in an open space (figure 9.58). There are two paths sound can take to the microphone: the direct signal path and a reflected signal path off the floor. Suppose further that the two paths together make an isosceles right triangle, as shown. If the length of the reflected path is 2 m, then the length of the direct signal path will be $\sqrt{2} \cong 1.414$m, and the reflected path will arrive at the microphone delayed by $(2 - 1.414)\text{m}/c_s = 0.586/331.1 = 1.76$ ms. The pressure waves of the two paths meet at the

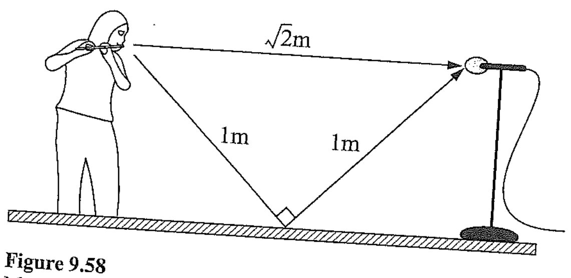


Figure 9.58
Multipath wave propagation.

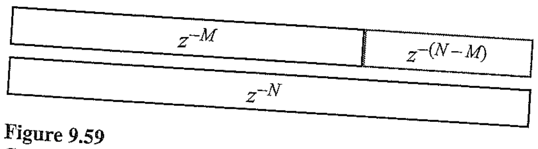


Figure 9.59
Composite delay lines.

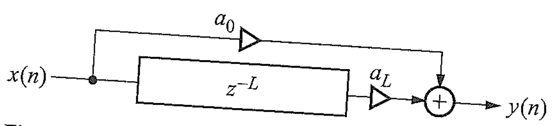


Figure 9.60
Comb filter.

microphone diaphragm and sum together. At some frequencies, the wave fronts reinforce at the diaphragm, while at others they cancel. Thus, the interaction of these two waves at the microphone causes a kind of filtering that colors the tone quality of the instrument being recorded. Most of the surfaces of a good recording studio are made to be absorbent so as to minimize all signal paths but the direct one, thereby reducing spectral coloration from multipath propagation.

We could simulate the preceding effect with two delay lines, one for the direct signal path, another for the reflected path (figure 9.59). But since we are generally more interested in the relative delay between paths than in the absolute delay, we simply delay the reflected path by the difference between the delay lengths, $L = N - M$.

Taking this approach, we can model a dual-path acoustical system as shown in figure 9.60, where a_0 is the signal attenuation experienced by the direct signal path, and a_L is the attenuation through the delay of length L. The summation point ⊕ corresponds to the point where the acoustical paths mix together at the microphone's diaphragm. The filter structure is a *comb filter.*

Here's how to determine the frequencies at which the two paths reinforce and cancel their signals at the microphone's diaphragm. Set a_0 and a_1 to positive nonzero values, and then drive the system with a sine wave of amplitude ± 1. The key is to note that *the phase of the sine wave entering the delay line is also the phase of the sine wave entering the summation point ⊕ via the direct signal* *path, since the direct path has zero delay.* The gain is maximum and equal to $a_0 + a_L$ when a whole number of periods fits in the L samples of the delay line because then both ends of the delay line are exactly in phase. This occurs at frequencies $\omega_k T = k2\pi/L$, $k = 0, 1, 2, \ldots$. Note that these are harmonics of the fundamental frequency $\omega_1 T = 2\pi/L$ of the delay line. Thinking along the same lines, gain is minimum and equal to $|a_0 - a_L|$, where an odd number of half periods fits in the L samples of the delay line because then the ends of the delay line are exactly out of phase. Thus, minima occur at frequencies $\omega_k T = (2k+1)\pi/L$.

Returning to the flute example, the frequency corresponding to a delay of 1.76 ms is 568 Hz. So one period of a 568 Hz sine wave fits the difference between the direct and reflected path, and the microphone diaphragm experiences maxima at that frequency and its harmonics. It experiences minima at one half that frequency, 284 Hz and its harmonics.

We can derive the frequency response of the comb filter as follows. By inspection, we can express the difference equation as

$$y(n) = a_0 x(n) + a_L x(n-L). \tag{9.50}$$

Compare equation (9.50) to the difference equation of the simple lowpass filter given in equation (5.3). They are basically the same except that here the second term is delayed by z^{-L} instead of just z^{-1}. Therefore, the transfer function is

$$H(z) = a_0 + a_L z^{-L}.$$

Evaluating this on the unit circle by setting $z = e^{i\omega T}$, we obtain the complex frequency response,

$$H(e^{i\omega T}) = a_0 + a_L e^{-iL\omega T}. \tag{9.51}$$

Look again at figure 5.10, which shows the transfer function of a simple lowpass filter. Essentially, all that has changed here is that the exponent of z now has an extra coefficient L. Whereas the lowpass transfer function goes counterclockwise 180° as $\omega \to \pi$, this transfer function goes through $L/2$ rotations. We end up with a frequency response like the one in figure 5.13, except repeated $L/2$ times.

Converting equation (9.51), the complex frequency response, into the magnitude response, we have

$$G(\omega) = |H(e^{i\omega T})| = |1 + e^{-iL\omega T}| = 2\left|\cos\frac{L\omega T}{2}\right|. \tag{9.52}$$

Compare this to equation (5.28), the frequency response of a lowpass filter. Figure 9.61 shows the comb filter spectrum for $L = 7$. It should now be clear how the comb filter got its name.

The comb filter spectrum has a series of $L - 1$ *notches* or *nulls* in the Nyquist interval where the energy goes to zero, so we could call it a kind of multiple band reject filter. When heard, this effect is immediately recognizable as *flanging,* which is a technique where a signal is summed with a slightly delayed copy of itself. If the length of the delay varies over time, the number and position of the nulls change, superimposing a kind of hollow swish sound over whatever material is played through it.[18]

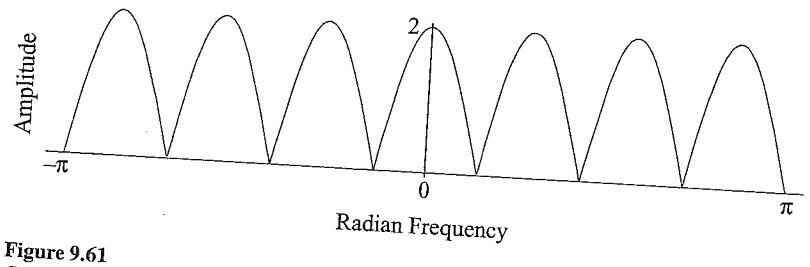


Figure 9.61
Comb filter frequency response.

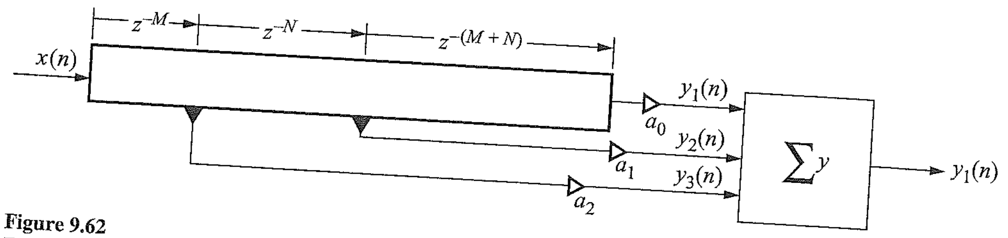


Figure 9.62
Tapped delay line.

9.6.5 Tapped Delay Line

In most natural settings, there are typically many more sources of reflection than there are sources of sound. For example, in a concert hall, sound from a single source on the stage will be reflected by the numerous acoustical features of the hall. We can adapt the delay line to this reality by placing additional outputs, called taps, at various points within the delay line. Each tap registers the signal at its location and sums it with any other taps to create the final output (figure 9.62).

We can simulate how a listener will experience the early reflections in a concert hall. For a simplified example, start with the floor plan of a concert hall, such as shown in figure 9.63. For an arbitrary listening position, trace the distance of the first several room reflections as rays between the stage and the listener. Using the speed of sound, convert these distances to delay times. Now adjust the positions of the taps in the tapped delay line to correspond to these delays. Next, calculate to the inverse square law of spherical spreading, air absorption, and sound absorption by the walls. Set the coefficients for each tap appropriately. Since the tapped delay lines are modeling the sum of reflections at each ear, it might be convenient to have two tapped delay lines, one for each ear. Driving relatively dry music through these tapped delay lines gives a spacious quality to the result.

An interesting effect is created when the taps can be adjusted in real time. With appropriate control software, it becomes possible for a listener to "walk around" in a virtual acoustic space and have the reflections track the listener's movement in the space. Such techniques are used to simulate artificial acoustic environments in virtual reality simulations of physical spaces.

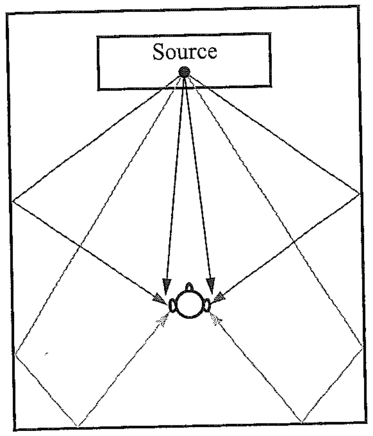


Figure 9.63
Early side reflections.

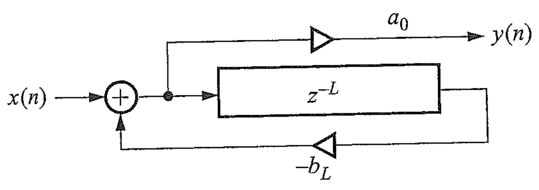


Figure 9.64
Recirculating comb filter.

9.6.6 Recirculating Delay Lines

If we feed the output of a delay line back to its input, it becomes a kind of recursive filter that, for fairly long delays, behaves like the Echoplex tape machine, described at the beginning of chapter 5.

Figure 9.64 shows the output of a recirculating delay line. Inverted copies of sound $x(n)$ delayed by L samples are added back to the delay line's input and also to the output $y(n)$. If $x(n)$ is the sound of a speaker saying "ECHO", then for $a_0 > 0$, $b_L > 0$, and $b_L < 1$, $y(n)$ will sound like "ECHO Echo echo echo echo . . .", mimicking the way a sound dies exponentially as it reflects between the walls of a room. The greater the length of the delay line, the longer it takes the echo to return, and as the feedback gain $b_L \rightarrow 1.0$, the echoes take longer and longer to die away. When $b_L = 0$, this degenerates into a simple gain control of a_0.

Using a recirculating delay line to model room reverberation produces a rather unnatural effect, in that each pass of the sound through the delay produces a copy that is spectrally identical to the original, only quieter (if $b_L < 1$). But in a natural setting, air absorption and absorption by walls is not spectrally flat; rather, high-frequency energy is absorbed at a higher rate, so the successive reflections are progressively more muffled. We can simulate this effect by substituting a simple

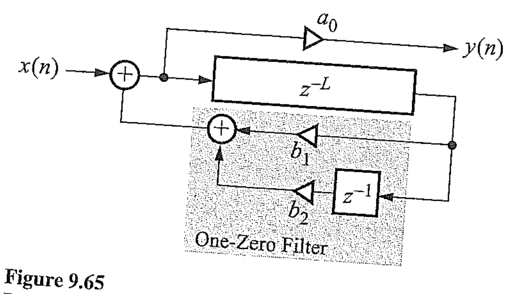


Figure 9.65
Recirculating delay with one-zero filter.

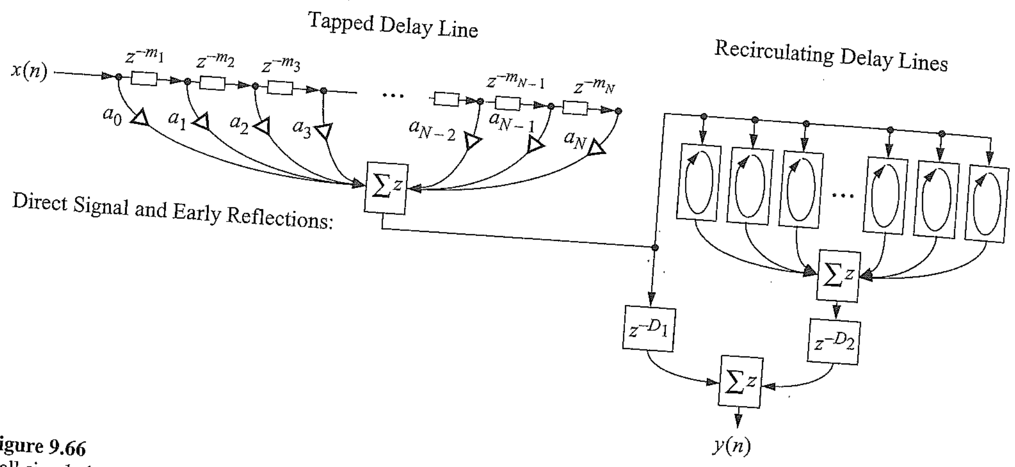


Figure 9.66
Hall simulation system using delay lines.

one-zero lowpass filter for the feedback term (figure 9.65). If we set $b_1 = b_2 = 1/2$, the filter averages adjacent samples, so that each pass through the delay line progressively removes high frequencies. Other coefficient settings accelerate or retard this effect (see section 5.13.1).

9.6.7 Hall Simulation

Schroeder (1970) experimented with combinations of allpass and comb filter elements to create realistic synthesis models of room acoustics. Recall volume 1, figure 7.36, which showed how the impulse response of a room can be analyzed into a direct signal, early reflections, and reverberation tail. The direct signal and early reflections can be modeled with a single tapped delay line. The reverberation tail can be modeled as a set of recirculating delay lines. Moorer (1979a) extended Schroeder's work, adding early reflections as well as air and wall absorption. A flow graph of Moorer's system is shown in figure 9.66.

The input signal $x(n)$ is fed both to the input of a tapped delay line and to the summed output of the tapped delay line. The coefficient a_0 scales the contribution of the direct signal. The lengths of tap delays z^{-m} are chosen to match the geometry of the early reflections of the room and are scaled respectively by coefficients a_m to account for spreading loss, air absorption, wall absorption, and so on. The sum of the direct signal and early reflections constitutes the first 40 to 80 ms of the decay.

The direct signal and early reflections are sent both to a set of recirculating delay lines and to an additional delay that is eventually summed into the final output. The lengths of the recirculating delay lines can be chosen, for example, to model the distance between the three pairs of opposing walls in a typical shoebox concert hall, and their respective gains can model accumulated losses due to spreading and absorption. The output of the recirculating delays is summed, sent through another delay, then summed with the early reflections.

The advantage of Moorer's topology is that all the density of the first N reflections is forwarded to the recirculating delay, which then recirculates the reflections many times, so that the buildup of reflections is quite rapid, as is the case with better concert halls. The delays D_1 and D_2 are set so that the first echo from the recirculating delays coincides with the end of the last echo from the early reflections. This means that either D_1 or D_2 will be zero, depending on whether the total delay of the early reflections is longer or shorter than the shortest recirculating delay line.

Unfortunately, a good deal of fiddling around is required to obtain good-sounding reverberation with this setup. The delays commonly stack up in a highly composite way, even if the delay lengths are chosen to be mutually prime, so that their prime factorizations contain no common factors. Since the delays cascade through the system, their lengths are added together, and the sum of multiple primes is by definition no longer prime. Moorer gives example settings that sound good, but that are heuristically derived.

Performance is improved greatly if lowpass filters are used in the feedback path of the recirculating delays instead of simple gains. However, this introduces another slight problem: the later parts of the reverberation tail become unnaturally tubby because only low-frequency energy can linger. In a natural concert hall, low-frequency energy is lost due to admittance through the walls.[19] This can be overcome by inserting a highpass filter just prior to the final output to simulate the admittance.

The reverberation system developed by Jot (1997) goes a step further, providing control over frequency-dependent reverberation time, allowing independent control over the coloration of a reverberator and its decay time. It is based on a scheme for interconnecting recirculating delay lines that uses a matrix interconnection between a set of recirculating delay lines. (Gerzon 1976). This approach also provides faster and denser buildup of reflections than is obtainable with comb filter networks.

9.7 Physical Modeling

Physical modeling studies the causal interactions of vibrating systems that are the basis of natural sounds. This includes the way energy travels from a performer into and through a musical instrument, how its resonances affect the resulting vibration, and how these vibrations are propagated into the surrounding air. The reverberation and ambient sound systems developed previously are also examples of physical models.

Since it is based on the physics of the vibrating system, physical modeling can also capture the idiosyncrasies of a particular instrument. Physical models capture not only the steady-state behavior of an instrument but also its transient characteristics. As energy floods through a vibrating system, the instrument responds in a characteristic way, and its response provides fertile information to the ear to help characterize the sound source.

There are many approaches to physical modeling. The conventional physical modeling approach was outlined in chapter 7. Starting with a differential equation for an ideal vibration, refinements are added to accommodate other important factors affecting the vibrating system, such as friction, radiation, driving force, and stiffness (see also volume 1, chapter 8, and in this volume, chapters 6 and 8).

Digital waveguide models have been developed that share the advantages of physical modeling but are computationally more efficient as well as intuitively more appealing than classical physical modeling (Smith 1996). Preparation for understanding this material can be found especially in chapter 8. This section focuses on waveguide models, beginning with Karplus-Strong synthesis, a simple extension of the delay line to model plucked strings.

9.7.1 Karplus-Strong Synthesis

We can create a flexible and very inexpensive physical model of a plucked string out of a delay line of length L. First remove the input $x(n)$ from the delay line. Then preload a sequence of random samples into the memory cells of the delay line, $U_0, U_1, \ldots, U_L$, so it looks like figure 9.67a. When the delay line is switched on, it outputs the preloaded random sequence one sample at a time, and if it's connected to an audio DAC with sample rate R, we hear a brief burst of noise as the delay line drains, which sounds kind of like "fffft."

The next step is to recirculate the delay line's output back to its input, as in figure 9.67b. The samples in the recirculating delay line form a periodic sample pattern that the ear detects as a

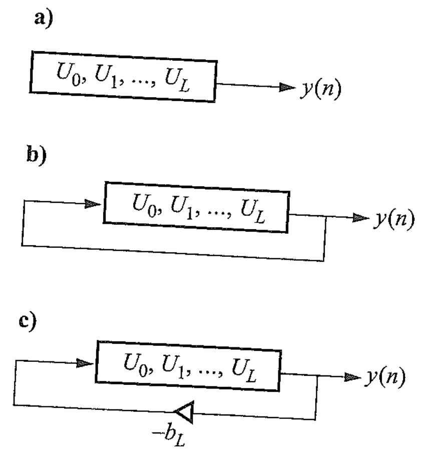


Figure 9.67
Karplus-strong delay lines.

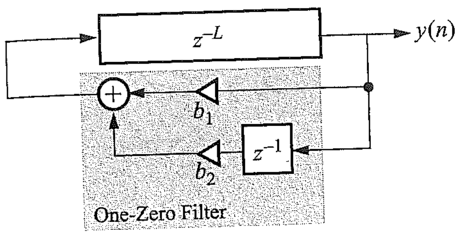


Figure 9.68
Plucked-string synthesis.

complex tone, with a frequency f corresponding to the length of the delay line and the sampling rate, $f = R/L$. Even though the pattern is random, the ear interprets it as a steady, buzzy timbre.

Let's try adding a multiplier b_L on the feedback path (figure 9.67c). Setting the multiplier coefficient b_L to 1 gives us the behavior described in the previous paragraph. Setting it to 0 gives us the one-shot "fffft" sound we started with. But if we set it somewhere in the range $0 < b_L < 1$, then the periodic noise will be attenuated exponentially to silence as it recirculates. During the first pass through the delay line, the samples are scaled by b_L. During the second pass, they are scaled by b_L^2, and on pass n, the samples are scaled by b_L^n. The ear hears this as a complex pitched tone with an exponential decay, suggestive of a plucked or struck string.

A characteristic of plucked or struck string instruments that we'd like to emulate is that the higher harmonics die away more quickly than the lower harmonics, so that the sound becomes more muffled over time. The delay line instrument could be greatly improved if we replace the multiplier b_L with a simple one-zero lowpass filter, as shown in figure 9.68.

If we set $b_1 = b_2 = 1/2$, the filter is just the simple moving average lowpass filter studied in section 5.3. As the preloaded signal recirculates through the delay line, samples pass through the lowpass filter, and on each recirculation, remaining high-frequency energy is attenuated more rapidly than low-frequency energy. The result is that the bright periodic noise we hear at the beginning doesn't just die away, it mellows through time. This lends a dynamical spectral behavior to the simulation that can be made to sound very convincingly like mandolins, banjos, harps, plucked violins, guitars, and drums, depending upon the signal preloaded into the delay line and the kind of filtering to which it is subjected (Karplus and Strong 1983).

The reach of the Karplus-Strong synthesis model was dramatically extended and related to the physics of the plucked string by Jaffe and Smith (1983). The discussion here follows their analysis and that of F. R. Moore (1990, 282).

The dynamic spectral effects introduced by the moving-average filter in the feedback path is what makes this synthesis technique interesting. But some questions must be answered to make this a really useful synthesis technique:

How long does it take for a sound to die away?

Can we control the decay rate of the harmonics?

What is the effect of the phase delay of the filter on frequency?

If we set $b_1 = b_2 = 1/2$ in figure 9.68, the filter's impulse response is

$$y(n) = \frac{x(n) - x(n-1)}{2}.$$

We studied this filter extensively in chapter 5 and know that its transfer function is

$$H(z) = \frac{1 + z^{-1}}{2}.$$

Its frequency response is obtained by taking the magnitude of the transfer function evaluated at $z = e^{i\omega} = e^{i2\pi f/R}$, which we know from chapter 5 evaluates to

$$G(f) = \cos\frac{\pi f}{R}.$$

So the frequency response of the averaging filter is shaped like the first quadrant of a cosine wave, going from 1 to 0 as frequency goes from 0 Hz to the Nyquist rate, $R/2$ Hz. It is a lowpass filter with a gentle roll-off of high frequencies. After the signal has recirculated once around the delay loop, its frequency content is attenuated according to $G(f)$. After the second time around, frequency content is attenuated by $G(f)^2$, and on the nth time around, attenuation is $G(f)^n$.

The phase delay of the filter is the phase of $H(e^{i\omega})$ divided by ω, which we found in chapter 5 evaluates to

$$\angle H(e^{i\omega}) = -\frac{1}{2}.$$

So the averaging filter introduces a one half sample delay to signals passing through. The filter is *linear-phase* because the delay is constant regardless of frequency. We can express the fundamental frequency f_1 of the recirculating delay line as the length of the delay L plus the length of the averaging filter with respect to the sampling rate: $f_1 = R/(L + 0.5)$ Hz. Define the frequency of harmonic k as $f_k = kf_1$. Then the rate at which harmonic k dies away is $G(f_k)^n$.

Jaffe and Smith (1983) determined that the time in seconds required for harmonic k to be attenuated by Q dB is given by

$$\tau_Q(f_k) = \frac{\ln 10^{Q/20}}{f_1 \ln|G(f_k)|}. \tag{9.53}$$

Equation (9.53) demonstrates that the decay time of harmonic k is related to the fundamental frequency of the delay loop f_1 and to the attenuation of the filter at that frequency, $G(f_k)$. This means that higher harmonics decay faster than lower ones, and also that high-pitched tones decay faster overall than low-pitched tones, just as we wanted. Happily, these behaviors map well onto the timbre of plucked and struck string tones. Unhappily, the range of variation from high to low pitch is actually too wide in practice and sounds unnatural. Besides, one size does not fit all: each kind of plucked string instrument has a unique trade-off between high-frequency rolloff and tone length.

To control note duration, we must be able to shorten and lengthen notes arbitrarily. To do this, we must modify the averaging filter characteristics. A simple way to shorten notes is to add an additional gain control ρ in the feedback path, so the impulse response becomes

$$y(n) = \rho\frac{x(n) - x(n-1)}{2}, \qquad 0 < \rho < 1.0,$$

and the frequency response becomes

$$G(f) = \rho\cos\frac{\pi f}{R}.$$

This shortens the overall decay time and also decreases the range of variance between low and high harmonic decay. But we need also to be able to lengthen decay time, especially for high pitches. We can do this by modifying the filter so that high frequencies are less attenuated, thus increasing overall decay time. Jaffe and Smith set the filter coefficients so that $b_1 = 1 - S$, $b_2 = S$, where S is a decay-stretching factor. Together with the decay-shortening factor ρ, the combined impulse response becomes

$$y(n) = \rho[(1-S)x(n) + Sx(n-1)].$$

If $\rho = 1$ and $S = 0.5$, this is exactly the same filter as before. The frequency response of this filter is

$$G(f) = \rho\sqrt{(1-S)^2 + S^2 + 2S(1-S)\cos(2\pi f/R)}. \tag{9.54}$$

If $S \neq 0.5$, the decay time will be longer than for $S = 0.5$. (For 0 or 1, S will not cause any decay; when $S = 0$, the filter passes $x(n)$ directly, and when $S = 1$, it just passes $x(n-1)$ directly.)

The phase response of this modified filter is also more complicated:

$$\angle H(e^{i\omega}) = \frac{\tan^{-1}\dfrac{-S\sin\omega}{(1-S) + S\cos\omega}}{\omega}.$$

If $S \neq 0.5$, the phase delay will no longer be linear, and different frequencies will be delayed by different amounts. Therefore, the value of S will affect the frequency of the harmonics. For $S < 0.5$, higher harmonics will be progressively sharper. Happily, this allows us to mimic the stretching of upper harmonics that occurs in, for example, pianos. For $S > 0.5$, harmonics will be progressively flatter.

The last practical problem has to do with fine-tuning pitch. As defined, the Karplus-Strong model has a fundamental frequency of

$$f_1 = \frac{R}{L + 0.5}\text{Hz.}$$

It would be most convenient to run the instrument at a constant sampling rate R, but since L is also an integer, we are limited to the quantized pitches dictated by this formula. As fundamental frequency rises, L shrinks. At very high frequencies, the frequency jump between successive integer values of L can be so large that we can't match them to any scale. We need a way to achieve fractional delay times through the delay loop so we can achieve arbitrary pitch at a fixed sampling rate. Jaffe and

Smith point out that the allpass filter exactly performs this function (see section 5.13.4). The allpass filter only affects phase and has no effect on frequency. Jaffe and Smith cascade an allpass filter together with the averaging filter to achieve arbitrary delay and hence arbitrary fundamental frequency. They use a simple allpass filter with impulse response

$$y(n) = Cx(n) + x(n-1) - Cy(n-1),$$

which has a transfer function of

$$H(z) = \frac{C + z^{-1}}{1 + Cz^{-1}}, \qquad 0 < C < 1.$$

The phase delay is

$$\angle H(e^{i\omega}) = -\frac{1}{\omega}\tan^{-1}\frac{-\sin\omega}{C + \cos\omega},$$

which, for ease of calculation, is approximately equal to a delay of

$$D = \frac{1 - C}{1 + C} \tag{9.55}$$

at low frequencies, where D is the delay through the allpass filter. The phase delay of the allpass filter is not the same for all frequencies. Upper harmonics are slightly flattened in the range $0 < C < 1$, and are increasingly sharpened in the range $-1 < C < 0$. So long as $|C| < 1$, we can change C dynamically to achieve vibrato effects.

We can achieve simple overall gain control by directly scaling the output of the instrument with an amplitude term. To emulate the piano damper pedal, for example, we could have two values for ρ, a value closer to 1.0 during note sustain, and a value closer to 0 when the damper pedal is lifted. This would accelerate the attenuation of all energy in the delay loop, as does the damper pedal. This technique can also help reduce the possibility that arithmetic quantization errors in the filters would prevent the signal from dying away completely to zero. Another way to help with this problem is to prescreen the random values prior to filling the delay line: we calculate their mean value and substract this value from all samples, so the signal has no DC bias.

It would be convenient if we could specify the time τ required for a tone at frequency f to decay by Q dB. Solving equation (9.53) for $G(f)$, we have

$$G(f) = 10^{-Q/\tau}.$$

Frequency f may be a fundamental or a harmonic frequency. We compare this value to the one produced by the unadulterated delay loop:

$$G_{\text{nom}}(f) = \cos\pi\frac{f}{R},$$

which is the nominal attenuation assuming $\rho = 1$ and $S = 0.5$. If $G_{\text{nom}} < G$, we must lengthen the decay; if $G < G_{\text{nom}}$, we must shorten the decay. Lengthening requires solving equation (9.54) for S. This can be done by arranging equation (9.54) into standard form:

$$(2 - 2\cos\omega)S^2 + (2\cos(\omega) - 2)S + 1 - G(f)^2 = 0.$$

Since it has the general form of $ax^2 + bx + c = 0$, it can be solved by the quadratic formula. Two solutions will result, ranging generally between 0 and 1, on either side of 0.5. Choosing the value less than 0.5 results in harmonic stretching, and it is generally the one we'd want to mimic string instruments.

Last, we need to tune the allpass filter parameters to achieve the correct frequency. The nominal delay introduced by the moving-average filter is S samples. In order to calculate the length of the delay loop L, we must first compute the desired delay $L_d = R/f_1$. The integer part of L_d can be used to set the length of the delay loop L, and the fractional part can be used to fine-tune the pitch with the allpass delay. We extract the integer part of L_d with the floor function, $L = \lfloor L_d \rfloor$. If $L + S$ is less than the desired length, we subtract 1 from L and make up the difference by choosing an all pass coefficient C that produces the desired fractional delay D. Solving equation (9.55) for filter coefficient C in terms of D, we obtain

$$C = \frac{1-D}{1+D}.$$

This value of C is applied to the allpass filter, and the delay length is set to L, thereby obtaining the correct frequency.

There are many extensions to this technique. For example, to effect a softer stroke or pluck, the noise signal can be lowpass-filtered before preloading it into the delay line. To create timbres with only even harmonics, initialize the delay line to contain two identical periods of random values. This doubles the frequency and the decay time. To create drumlike timbres, one can modulate the feedback filter coefficients according to a random variable. Karplus and Strong (1983) suggest a feedback filter with an impulse response of

$$y(n) = \begin{cases} [x(n) + x(n-1)]/2, & b, \\ -[x(n) + x(n-1)]/2, & 1-b, \end{cases}$$

where b is a random variable in the range $0 < b < 1$.

One can also dynamically replace the contents of the delay line with successive samples of an arbitrary sound. The delay line acts rather like the resonating sympathetic strings of a sitar or viola d'amore to capture and sustain frequency content that is harmonically related to its length.

Karplus-Strong synthesis combines computational efficiency with realism and great expressive control. But Karplus-Strong synthesis is just a special case of a much broader and even more powerful synthesis model. Julius Smith (2004) generalized the Karplus-Strong model employing the theory of digital waveguides, to which we turn next.

9.7.2 Waveguide Synthesis

Waveguides can be thought of as acoustic tubes or strings, like the ones discussed in chapter 8, but they can be used to model any one-dimensional wave motion. (I've found it useful to think about waveguides as water-filled troughs, for example.) The discussion here follows Smith (2004).

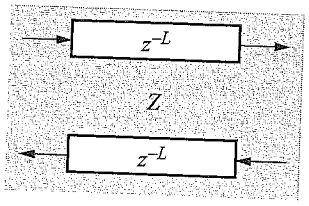


Figure 9.69
Waveguides.

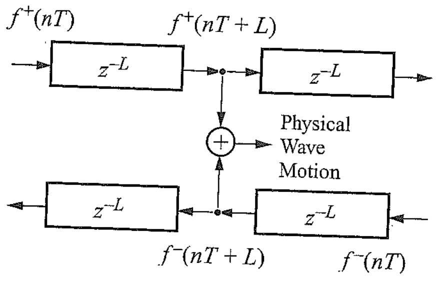


Figure 9.70
Tapping a waveguide delay line.

We observe experimentally that a disturbance in a one-dimensional medium causes waves to propagate away from the disturbing force in both directions (for example, see figure 7.5), and we've seen that solutions of the one-dimensional wave equation have the form of a positive-going (right-traveling) wave $f^+(nT)$ and negative-going (left-traveling) wave $f^-(nT)$ (see chapter 7). We can emulate this property of one-dimensional media by the simple expedient of combining two one-directional delay lines (see section 9.6) that propagate waves in opposite directions according to some characteristic impedance Z, an arrangement called a *waveguide* (figure 9.69).

Waveguides can be used to model any one-dimensional acoustical vibration. Physical force, pressure, or velocity waves can be represented as the sum of the traveling waves in the two delay lines.

We can realize a physical wave at any point along the waveguide by tapping and summing the two delay lines at the desired delay (figure 9.70).

Delay lines model wave propagation in a medium with constant characteristic impedance, but we must also be able to model wave propagation across impedance discontinuities, such as walls, string terminations, or cross-sectional area variation in tubes. The scattering junction (see section 8.15) describes the reflection and transmission of acoustical energy across a change of impedance (see especially figures 8.27 and 8.28). The complex reflection coefficient

$$R_{0,1} = \frac{Z_1 - Z_0}{Z_1 + Z_0}$$

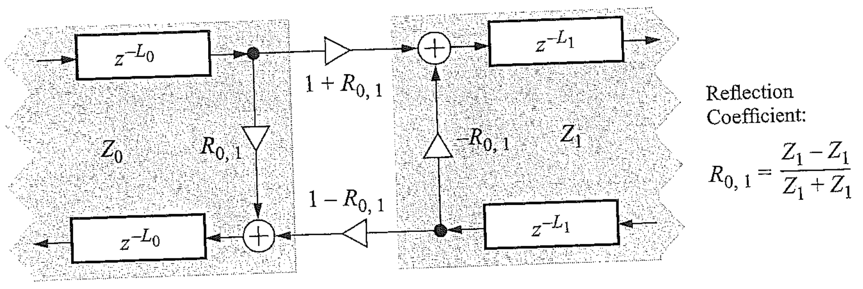


Figure 9.71
Waveguide scattering junction.

determines the way energy is reflected and transmitted by the junction as a function of the impedances to the left (Z_0) and right (Z_1) of the junction.

Combining waveguides with scattering junctions (figure 9.71) allows us to realistically emulate the physics of many natural vibrating systems such as strings, air columns, and halls without having to numerically integrate the wave equation.

9.7.3 Plucked-String Synthesis

We observed in figure 7.8 that a traveling wave on a string reflects from its terminations with a sign inversion. A string and its terminations can be modeled as a waveguide for the string and a pair of scattering junctions for the terminations. By setting the impedances of the string and terminations to appropriate values, the scattering junctions model reflection of energy back into the string at its terminations and also model energy transmission into the body of the instrument.

For example, let the vertical displacement of a string be $u(t)$, made up of a positive-going (right-traveling) wave $u^{+}(t)$ and negative-going (left-traveling) wave $u^{-}(t)$. Suppose it takes $NT = cl$ sample times of duration T for the wave components to propagate the length of a string of length l at speed c. If we sample these traveling wave components at N equidistant points along the string, we can preload the waveguide with these sampled values, and the samples will be indexed in the waveguide as $u(n)$, where n is the index.

Since NT is the propagation time of the string in one direction, $2NT$ is the round-trip string loop delay time. To model this, we must have two waveguides of length N, one to propagate forward, the other to propagate back. Using the terminology of stringed instruments, let's call one of the terminations the bridge at position $x = 0$ along the string, and the other end the nut termination at position $x = N - 1$.

If the terminations are ideally rigid and massive, and the string is ideally flexible and lightweight, then all energy is reflected from the junctions back into the string. Note in figure 9.72 that the signs of the coefficients that feed back energy from one waveguide to the other are negative. This is appropriate for displacement waves, velocity waves or acceleration waves, which reflect with phase inversion. Pressure waves and force waves reflect without phase inversion.

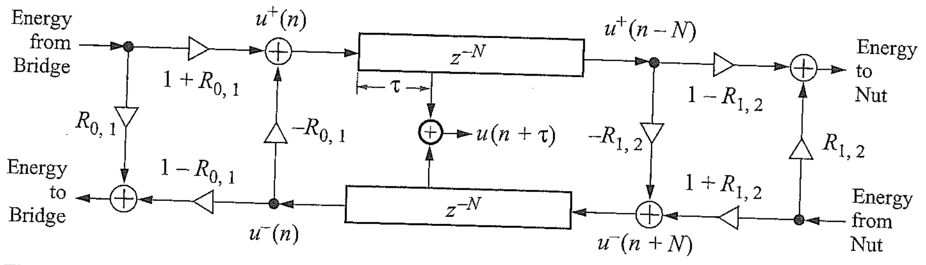


Figure 9.72
Simple waveguide plucked string synthesis.

Since we are modeling displacement of a string, the terminations are inverting, as shown in figure 9.72. If we set

$$R_{0,1} = R_{1,2} = 1 = \frac{Z_{m+1} - Z_m}{Z_{m+1} + Z_m} = \frac{\infty - 0}{\infty + 0},$$

then the scattering junctions actually don't scatter; they reflect all energy back into the string (with phase inversion). By choosing a value for R in the unit interval $0 < R < 1$, some energy is transmitted, and the remainder is reflected back into the string. If $0 < R << 1$, most energy is transmitted at the junction, and the string's vibration ceases quickly, like a banjo string or sound in a room with heavy curtains and carpeting. If $0 << R < 1$, most energy is reflected, and the vibration lasts longer, like a low-pitched piano string or sound in a room with stone surfaces.

We can usefully define the way in which energy is reflected at a scattering junction as the *reflection transfer function*. We can correspondingly define the way energy is transmitted as the *transmission transfer function*. Virtually all the energy at the nut of an unstopped guitar or violin string is reflected, so in these cases, the reflection transfer function at the nut is $R_{1,2} \approx 1$. When a violin string is stopped by the player's finger, energy is lost by friction of the string against the flesh of the finger, and so $R_{1,2}$ is quite a bit lower; consequently, a pizzicato (plucked) violin string tone dies away much more quickly if the string is stopped by the finger than if it is open (unstopped). Since a stopped guitar string is terminated by a metal fret that introduces little friction, its reflection transfer function is about the same whether it is stopped or not. At the bridge end of stringed instruments, some energy is transmitted into the body of the instrument, so $R_{0,1} < R_{1,2}$.

Because the body of a stringed instrument is essentially a Helmholtz resonator, the scattering junction at the bridge end of the string is frequency-dependent, so the value of the transmission transfer function is complex, and its magnitude frequency response has a lower $R_{0,1}$ for frequencies near resonance because at resonance the body is better able to radiate its energy.

Even if a musical instrument has multiple resonances, its overall transmission transfer function will be linear. To see this, imagine if we added an additional resonance to an instrument. For example, the Indian rudra vina (figure 9.73) has a resonating gourd at both the bridge and the nut. The effect of the two resonances on the overall spectrum of the instrument is additive, and overall its

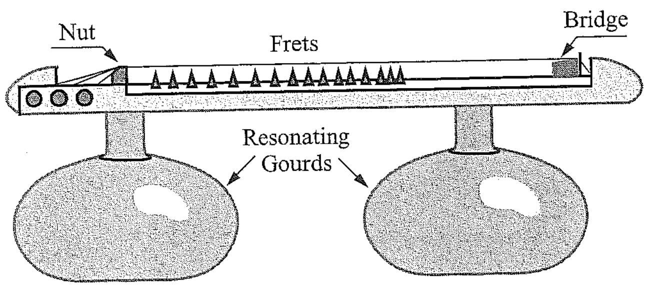


Figure 9.73
Vina.

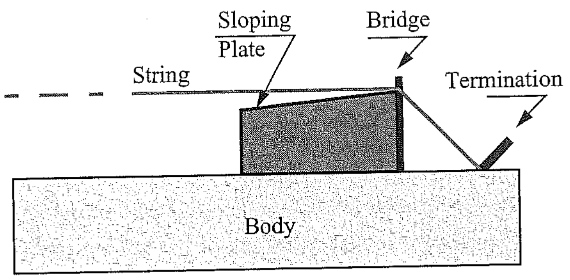


Figure 9.74
Vina bridge.

transmission transfer function is linear. In general, the resonances introduced by the bodies of musical instruments combine linearly.

9.7.4 Nonlinear Scattering Junctions

Not all scattering junctions are linear, and again the vina is a good example. An enlarged view of the bridge (figure 9.74) shows that a sloping bone plate comes up to meet the string as it approaches the bridge. As it vibrates vertically, the end of the string slaps against this plate. The impedance of this junction is nonlinear because the vertical (but not the horizontal) displacing force in the string is constrained by the plate, so that in this case force is not proportional to string displacement. The result is that high-frequency harmonic energy is injected into the string, giving it a characteristic sizzling sound of instruments that have this kind of plate next to the bridge (notably the vina and sitar), and the bandwidth increases with greater vertical excursion of the string.

9.7.5 Clarinet Synthesis

The clarinet headpiece also is a nonlinear scattering junction similar to the vina: the reed is clamped down against a plate in the clarinet headpiece in such a way that it closes against the headpiece in proportion to the pressure difference between the inside of the player's mouth and the inside

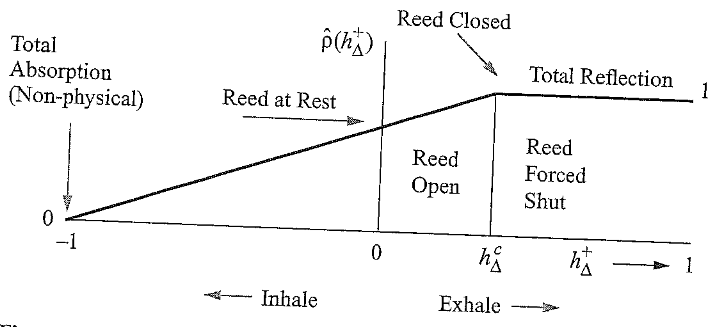


Figure 9.75
Simple clarinet reflection function.

of the clarinet bore. For waveguide synthesis, Julius Smith uses a simple signal-dependent and embouchure-dependent nonlinear reflection coefficient to terminate the bore at the headpiece. Figure 9.75 shows Smith's (2004) qualitative analysis of the reed reflection coefficient as a function of mouth pressure. The formula for this figure is:

$$\hat{\rho}(h_\Delta^+) = \begin{cases} 1 - m(h_\Delta^c - h_\Delta^+), & -1 \le h_\Delta^+ \le h_\Delta^c, \\ 1, & h_\Delta^c \le h_\Delta^+ \le 1, \end{cases} \tag{9.56}$$

for $m = 1/(h_\Delta^c + 1)$. The corner point h_Δ^c is the smallest pressure difference giving reed closure. Embouchure and reed stiffness correspond to the choice of offset h_Δ^c and slope m.

The scattering junction at the bell of the clarinet acts as a highpass filter for transmitted energy and as a lowpass filter for reflected energy, so it is a linear junction that behaves somewhat like a loudspeaker cross-over network.[20] Since the bore of a clarinet is effectively cylindrical, simple waveguides can be used to model the propagation delay in the bore. Because the main control variable is air pressure at the mouth, it is convenient to use pressure wave (noninverting) scattering junctions. The scattering junction for the bell can also serve to model the round-trip absorption losses of the bore, so we don't have to calculate these separately.

Smith's block diagram for a waveguide clarinet is given in figure 9.76. The reflection filter and output filter are complementary lowpass and highpass, filters, respectively, with cross-over frequency at around 1500 Hz for the bell. The output filter implements bell and tone-hole losses; the reflection filter passes the complementary low-frequency energy back into the bore. Smith suggests that the simplest practical implementation for the tone-hole losses is to use the bell filter settings unchanged for the tone holes, as though the clarinet were cut to the length of the fingered tone hole for each note. The cylindrical bore of the clarinet can be modeled with a simple waveguide.

The termination at the headpiece is what distinguishes the clarinet timbre from brasses that also have a cylindrical bore and flared bell. Two input controls determine oscillation: mouth air pressure $p_m(n)$ and embouchure. Ignoring the embouchure for a moment, the reflected signal $p_b^+(n)$ returning through the bore from the bell is summed with the mouth pressure to generate the total pressure at the headpiece.

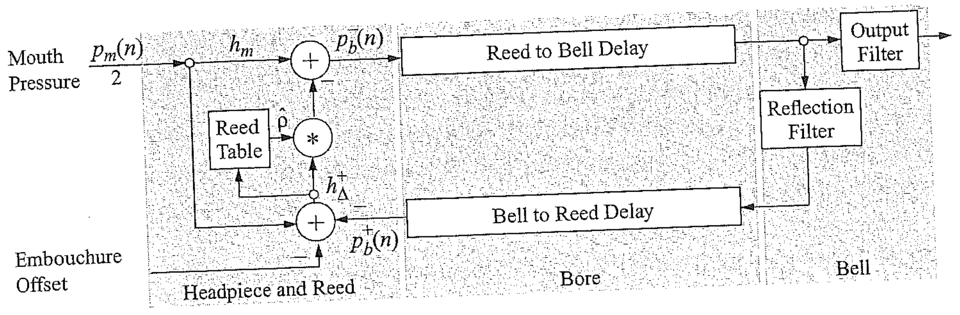


Figure 9.76
Waveguide model of a clarinet.

Total pressure h_Δ^+ is then used to determine how open or closed the reed should be. If the total pressure inside the headpiece is low relative to the mouth pressure, the reed will be sucked shut, and the reflection coefficient of the headpiece should be closer to 1.0, because most energy will be reflected. If the total pressure inside the headpiece is high relative to the mouth pressure, the reed will be blown open, and the reflection coefficient of the headpiece should be closer to zero because most energy will be transmitted into the mouth. The headpiece reflection coefficient $\hat{\rho}$ is therefore determined by using total pressure h_Δ^+ to index the function $\hat{\rho}(h_\Delta^+)$, given in figure 9.75, to establish the instantaneous reflection coefficient, depending upon the instantaneous position of the reed.

The embouchure is primarily a function of two variables: the position of the lips along the length of the reed and the clamping pressure of the lips on the reed. However, a simpler view is to note that these two parameters principally control the position of the pressure drop where the reed begins to open at the knee of the curve in figure 9.75. In light of this, the role of the embouchure can be seen as a simple offset that adjusts the position of the knee, and this is how it is implemented in figure 9.76. This one parameter can then emulate the strength and position of the bite and even the stiffness of the reed. Other effects, such as a general brightening, can be achieved by altering the slope of m in equation (9.56) or by making it an exponential instead of a linear function.

9.7.6 Bowed Strings

The bow divides the string into two sections, creating two nonlinear junctions with parts of the string on either side. A simplified block diagram for the basic waveguide bowed string according to Smith (2004) is shown in figure 9.77. When a rosined horsehair bow is drawn across a violin string, at first the frictional force of the bow drags the string along. When the restoring force of the string exceeds the bow frictional force, the string breaks loose and slides easily against the bow because the function of bow friction versus string velocity is nonlinear in such a way that as bow velocity increases, friction—which is initially quite high—drops quickly to a low point. When the string moves once again in the direction of the bow, it is entrained once again by the bow's frictional force and dragged along. Thus energy is added to the string by the bow as though by tiny plucks synchronized with the movement of the string in the direction of the bow. Since the primary control is bow velocity, the waveguide data are treated as velocity waves. (See section 8.3.)

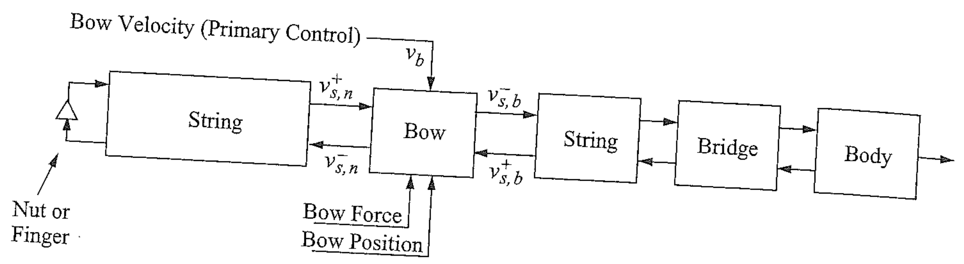


Figure 9.77
Block diagram of a waveguide bowed string.

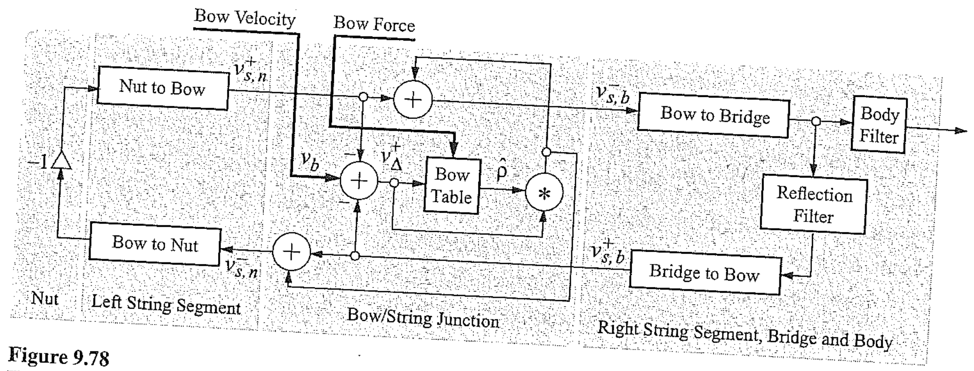


Figure 9.78
Waveguide model for a bowed string.

Figure 9.78 shows a waveguide model for a bowed string. The waveguide at the nut end of the string carries left-going $v^-_{s,n}$ and right-going $v^+_{s,n}$ velocity waves. The waveguide at the bridge end of the string carries right-going $v^-_{s,b}$ and left-going $v^+_{s,b}$ velocity waves. The + indicates waves traveling toward the bow. The fundamental pitch of the system can be adjusted by changing the absolute lengths of the waveguides, and the position of the bow on the string can be adjusted by changing the relative lengths of the bow-end and body-end waveguides. Continuous change in bow position and continuous change in pitch (glissando) can be implemented by interpolating between delay line samples.

The nut can be modeled as a unit reflection coefficient because virtually all energy is reflected from it. The bridge and nut are phase-inverting because we are interpreting the waveguide contents as velocity waves. The bridge has complex-valued losses as a consequence of the frequency response of the body, which acts like a Helmholtz resonator. Fingered notes on a violin are more highly damped and lowpass-filtered in comparison to the bright tone of the open strings. Rather than adding another filter to account for this, we can combine the reflection transfer function of the nut with that of the bridge to economize calculation, as well as the round-trip attenuation and dispersion of wave energy in the strings.

Bow velocity and bow force are the control inputs from the performer that determine the quality of vibration. On a violin, when the difference between bow velocity and string velocity is near zero, the frictional force increases, and the bow entrains the string, increasing its energy by dragging it along. We can simulate this as follows: energy from the left and right string segments arriving at the bow is passed along "underneath" the bow to the other string segment; additionally, depending on the velocities of bow and string, energy will be injected into the string or dissipated from it by the bow.

Here's how the bow/string junction works. In figure 9.78, the right string segment output $v_{s,b}^+$ and left string segment output $v_{s,n}^+$ are passed along "underneath" the bow to their respective string segments $v_{s,n}^-$ and $v_{s,b}^-$ through the adders on the outer rails of the bow/string junction. Additionally, $v_{s,b}^+$ and $v_{s,n}^+$ are sent into the bow/string junction to determine whether energy should be drained or added to the string velocity.

When bow and string velocities are similar, we inject energy; when they are opposed, we do not, or if bow pressure is heavy, we subtract energy.

In order to determine the coefficient of friction between bow and string, we must obtain the differential v_Δ^+ between the instantaneous string velocity at the bow, $v_{s,b}^+ + v_{s,n}^+$, and the bow velocity v_b. That is, we calculate $v_\Delta^+ = v_b - (v_{s,b}^+ + v_{s,n})$. When bow velocity and string velocity are opposed, v_Δ^+ is large, and we want the output of the bow table $\hat{\rho}(v_\Delta^+)$ to be close to zero so that the multiplication that follows the table injects little or no energy into the string. When bow velocity and string velocity are similar, v_Δ^+ is small, and we want the output of the bow table $\hat{\rho}(v_\Delta^+)$ to be large so that the multiplication that follows the table injects more energy into the string, thereby overcoming frictional and dissipative forces in the string and sustaining vibration.

The contents of the bow table defines the way friction and bow/string differential velocity interact. Smith gives a simple quantitative function (figure 9.79). The flat center section of the table shows the bow and string stuck together with a high coefficient of friction. The skirts of the function correspond to reduced friction when the string has broken away from the bow by exceeding v_Δ^c, the breakaway/capture differential velocity.

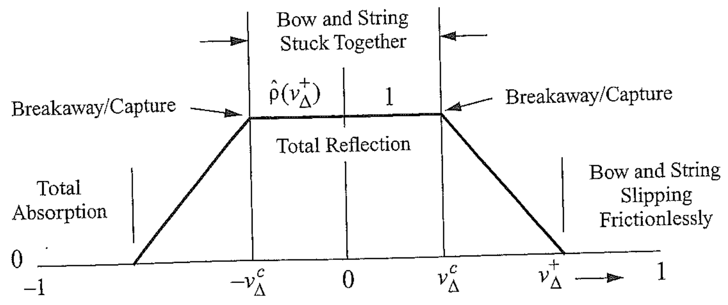


Figure 9.79
Simple string friction function.

The bow table function is defined by

$$\hat{\rho}(v_{\Delta}^{+}) = \frac{r(v_{\Delta}(v_{\Delta}^{+}))}{1 + r(v_{\Delta}(v_{\Delta}^{+}))}, \tag{9.57}$$

where the ratio of the bow impedance Z_b to the string impedance Z_s is given by $r(v_{\Delta}) = 0.25Z_b(v_{\Delta})/Z_s$, and $v_{\Delta} = v_b - v_s$ is the bow/string velocity differential.

The bow impedance function $Z_b(v_{\Delta})$ is the coefficient of friction of the bow against the string; that is, by the definition of impedance, bow force $F_b = Z_b(v_{\Delta}) \cdot v_{\Delta}$. Nominally, $Z_b(v_{\Delta})$ is constant and positive when the magnitude of the bow/string velocity differential is less than the breakaway/capture threshold, that is, when $|v_{\Delta}| < v_{\Delta}^{c}$, where v_{Δ}^{c} is both the capture and breakaway threshold. This is the static coefficient of friction. For $|v_{\Delta}| > v_{\Delta}^{c}$, $Z_b(v_{\Delta})$ rapidly falls toward a relatively low dynamic coefficient of friction, and in an ideal system continues towards a zero coefficient (which is impossible in practice). With a real bow, the breakaway/capture parameter is not a single point but is a function of a hysteresis parameter. Hysteresis in this context means that the transition from stuck to slipping requires slightly greater force than is required to transition from slipping back to stuck. This simplified model ignores hysteresis.

9.7.7 Critique of Waveguides

The simple digital waveguide models of musical instruments share the advantages of physical modeling but are computationally more efficient. Smith suggests that waveguide models require $O(1)$ computations per sample, whereas conventional physical models require $O(N)$, where N is the number of discrete modeling cells, corresponding to the number of samples in the waveguides. For realistic tasks, Smith estimates that waveguides are on the order of 300 times more efficient than numerical integration of the wave equation. It is possible to construct complicated waveguide synthesis models that can be calculated in real time.

If the model is fitted to a suitable physical controller, it can be performed live. And that's a really good thing because, in spite of the uncanny way in which waveguide models seem to efficiently and intuitively capture the acoustical process of musical instruments, they do not capture any of the nuance of a skilled performer. Audition of samples generated by the simple models sound flat and wooden, as if they were performed by an orchestrion. We should not be surprised at this, and in all fairness, this critique must also be leveled at all the other synthesis methods discussed in this chapter. It's only that, with waveguide instruments, the gap between the sometimes quite high degree of realism of the sound and the clumsiness of algorithmic control begs for better ways to perform these models.

Fortunately, this problem is getting some attention. If a physical model is implemented in a real-time computer system and equipped with sensors to track a performer's gestures, it can be performed like a regular instrument, with very convincing results. By changing the parameters of the model, the controls can be used to create a wide variety of different instrumental timbres. We can model the response of an instrument to the articulations of a performer, and we can characterize both the ordinary and degenerate vibrational modes of an instrument. For example, a physical model of a violin can capture how excessive bow pressure produces a wolf tone, or how an overblown clarinet produces a multiphonic, or how a misplaced embouchure on a flute produces a breathy tone.

One of the aims of music synthesis articulated at the beginning of this chapter is to obtain realization of music without performers. But all of the synthesis techniques discussed in this chapter are like instruments without performers, they incorporate no model of the performer/instrument interaction. To obtain a human touch still requires a human. It is certainly an area of fruitful research to characterize and understand human performance.

9.8 Source Models and Receiver Models

Data compression is a potential application for all synthesis techniques described in this chapter. Audio compression techniques such as MPEG achieve compression by employing a psychoacoustic model that removes unhearable components. Thus, MPEG compression can be called a *receiver model,* and by this reasoning, physical modeling is a *source model.* (However, MPEG-4 incorporates both source and receiver models.) Typically, control parameters of source models require far less bandwidth than the parameters of a receiver model. The Musical Instrument Digital Interface (MIDI) also qualifies as a source model. With MIDI, a note can be started with as little as three bytes of data and ended with only two more.

For a more general example, we can interpret common music notation as the control parameters of a source model, namely, a standard symphony orchestra. The memory required to store the notated score of Beethoven's Ninth Symphony as a set of note parameters is small in comparison to the memory required to store an MPEG-encoded representation of an orchestra playing that score. Recent versions of the MPEG standard, such as MPEG-4, incorporate source modeling to allow for this kind of audio compression.

We gain something and lose something with source models. With receiver models, *any* signal the ear can hear can be encoded (with varying levels of quality, depending on the encoding process). With source coding, all one can encode are sounds for which there is a suitable model. To achieve realism, the model must either be performable in real time by a person, or one must incorporate gestural knowledge of a competent performer into the model. It would be ideal if a violin physical model could discriminate, for example, between a Stradivarius and a Guarneri violin, for example, because certainly a competent listener can do so from a high-quality MPEG encoding. Unless the model is performed in a concert space, the model must also supply an acceptable simulation of such an acoustic space.

Thus source models must explicitly embody information that receiver models can take for granted. Therefore, it is unlikely that source models will supplant receiver models, but they will supplement them where appropriate. And there are plenty of places where they are appropriate and desirable, such as to provide musicians with novel and adaptable instruments and to supply composers with realistic simulations of musical instruments that they can use to preview their works.

Summary

Linear synthesis techniques can generally be used to reproduce a sound that is identical to the original. Nonlinear techniques generally provide no way to reproduce a sound that is identical to an original but may have other compelling advantages, such as being economical to calculate or intuitive to use.

Linear transforms add or subtract weighted basis functions of some kind, such as the sinusoids used by the Fourier transform. Additive synthesis combines individual components to create complex timbres. Subtractive synthesis removes energy from a spectrum by filtering. Linear systems are fairly intuitive to use but can require a great deal of analysis data. Nonlinear techniques typically are much more economical, if less general.

We developed a patch system to specify synthesis algorithms and used it to create patches for a wide variety of synthesis techniques. We first got control over pitch, duration, amplitude, amplitude envelope, and vibrato, then investigated a set of synthesis techniques to control timbre. Some sound synthesis techniques are optimized to provide the most naturalistic sound possible. Others produce hybrid sounds, or unearthly sounds, or sounds that metamorphose.

Oscillator bank synthesis generates a weighted sum of fixed-frequency sinusoids at static harmonics of a fundamental. Adding functions of frequency and time to each oscillator provides for time-varying (dynamic) synthesis of arbitrary spectra.

We considered a set of geometrical waveforms obtainable from Fourier series, including square wave, triangular wave, sawtooth wave, and sum of cosines.

By substituting a wavetable for continuous sinusoids, we can discretize oscillator bank synthesis and provide efficient and highly general synthesis. The wavetable can be any discrete function, so the resulting synthesis is completely arbitrary. We developed a table lookup oscillator.

Amplitude modulation (AM) varies the instantaneous amplitude of a signal in a periodic manner. Its spectrum can be viewed as a combination of fixed frequency components. Amplitude modulation without the carrier present is called ring modulation.

Frequency modulation (FM) varies the instantaneous frequency of a signal in a periodic manner. Its spectrum also can be viewed as a combination of fixed frequency components. The amplitudes of the sidebands are controlled by Bessel functions. By varying depth of modulation and the frequency ratio of carrier to modulating oscillator, a wide variety of musical instrument simulations can be created.

Waveshaping synthesis is an improvement over FM synthesis in that arbitrary spectra can be directly specified instead of being mandated by the shape of the Bessel function curves. In its simplest form, it only creates harmonic spectra, but it can be extended to create inharmonic spectra.

Waveform synthesis can be used to model vowel sounds. Linear prediction can synthesize high-quality vocal utterances and is relatively economical to calculate. However, because it relies on recursive filtering techniques, is must be used carefully. If an outcome is predictable based on available information, then any additional information of the same kind is redundant. We can define information as the degree of entropy in a signal. We can define entropy as the number of discrete states required to characterize a system. We want the resonant properties of speech to have minimum entropy because then the states required to characterize the resonance will be merely a handful of prediction coefficients. We want the residue properties of speech to have maximum entropy because then we have made sure that everything that can be predicted is accounted for in the prediction coefficients. Most acoustical signals have high redundancy and low entropy, allowing for a great deal of data compression, thereby lowering communication costs. LPC is attractive to musicians because it divides a signal into an all-pole filter and a residual signal.

Formant wave functions in FOF synthesis consist of windowed sinusoids with frequencies corresponding to the center of each vocal formant. The bandwidth of each FOF can be adjusted to match the spectral shape of a vocal formant. Although the technique can model many timbres, it excels at modeling vocal resonances. FOF synthesis is computationally less challenging than filter-based approaches to speech synthesis such as LPC, but it can't produce consonants.

Granular synthesis, like FOF synthesis, builds up complex sounds out of individual grains of sound. It is an idea related to the work of Dennis Gabor that combines "frequency language" and "time language" in one sonic event.

Concert hall acoustics can be modeled using delay lines, recirculating delay lines, and allpass filters in various combinations. Acoustically, a room is essentially a multipath wave propagation system.

Physical modeling synthesis models the causal interactions of vibrating systems that are the basis of natural sounds. They capture not only the steady-state behavior of an instrument but also its transient characteristics. Karplus-Strong synthesis can be used to create inexpensive physical models of a plucked string. It can be thought of as a simplified form of waveguide synthesis, which is capable of synthesizing also winds, brasses, and bowed strings.

A potential application for all synthesis techniques described in this chapter is data compression. Audio compression techniques such as MPEG achieve compression by employing a psychoacoustic model that removes unhearable components. Thus, MPEG compression can be called a receiver model, and by this reasoning, physical modeling is a source model.