4 *Sampling and Additive Synthesis*

Sampling Synthesis

Musique Concrète and Sampling: Background
Looping
Pitch Shifting
Sample-rate Conversion without Pitch Shifting
Problems in Resampling
Data Reduction and Data Compression in Samplers
Data Reduction
Data Compression
Sample Libraries
An Assessment of Samplers
Modeling Note-to-note Transitions

Additive Synthesis

Additive Synthesis: Background
Fixed-waveform Additive Synthesis
The Phase Factor
Addition of Partials
Time-varying Additive Synthesis
Demands of Additive Synthesis
Sources of Control Data for Additive Synthesis

Additive Analysis/Resynthesis

Musical Applications of Additive Analysis/Resynthesis
Methods of Sound Analysis for Additive Synthesis
Data Reduction in Analysis/Resynthesis

Line-segment Approximation
Principal Components Analysis
Spectral Interpolation Synthesis
Spectral Modeling Synthesis
Walsh Function Synthesis

Conclusion

This chapter introduces the method of sound sampling and several forms of additive synthesis. These techniques are fundamental to computer music and should be understood by every musician interested in synthesized sound.

Sampling Synthesis

In popular parlance, *sampling* means making a digital recording of a relatively short sound. The term "sampling" derives from established notions of digital *samples* and *sampling rate*. Sampling instruments, with or without musical keyboards, are widely available. All sampling instruments are designed around the basic notion of playing back prerecorded sounds, shifted to the desired pitch.

Sampling synthesis is different from the classical technique of fixed-waveform synthesis explained in chapter 1. Instead of scanning a small fixed wavetable containing one cycle of a waveform, a sampling system scans a large wavetable that contains thousands of individual cycles—several seconds of prerecorded sound. Since the sampled waveform changes over the attack, sustain, and decay portion of the event, the result is a rich and time-varying sound. The length of the sampling wavetable can be arbitrarily long, limited only by the memory capacity of the sampler. Most samplers provide an interface to an optical or magnetic disk drive so that groups of samples can be loaded into the sampler relatively quickly.

Musique Concrète and Sampling: Background

Composed manipulation of recorded sounds dates back at least as early as the 1920s, when composers such as Darius Milhaud, Paul Hindemith, and Ernst Toch experimented with variable-speed phonographs in concert (Ernst 1977). Magnetic tape recording, originally developed in Germany in the 1930s, permitted cutting and splicing, and therefore flexible editing and rearrangement of sequences of recorded sounds. Tape recorders were not available to musicians until the post–World War 2 period.

After experiments with variable-speed phonographs in the late 1940s, Pierre Schaeffer founded the Studio de Musique Concrète at Paris in 1950 (see figure 4.1). He and Pierre Henry began to use tape recorders to record and manipulate *concrète* sounds. *Musique concrète* refers to the use of microphone-recorded sounds, rather than synthetically generated tones as in pure electronic music. But it also refers to the manner of working with

Figure 4.1 Pierre Schaeffer's studio for musique concrète at rue de l'Université, Paris, 1960. The studio features three tape recorders on the left, along with a disk turntable. On the right is another tape recorder and the multiple-head Phonogène device (see figure 4.2). (Photograph courtesy of the Groupe de Recherches Musicales, Paris.)

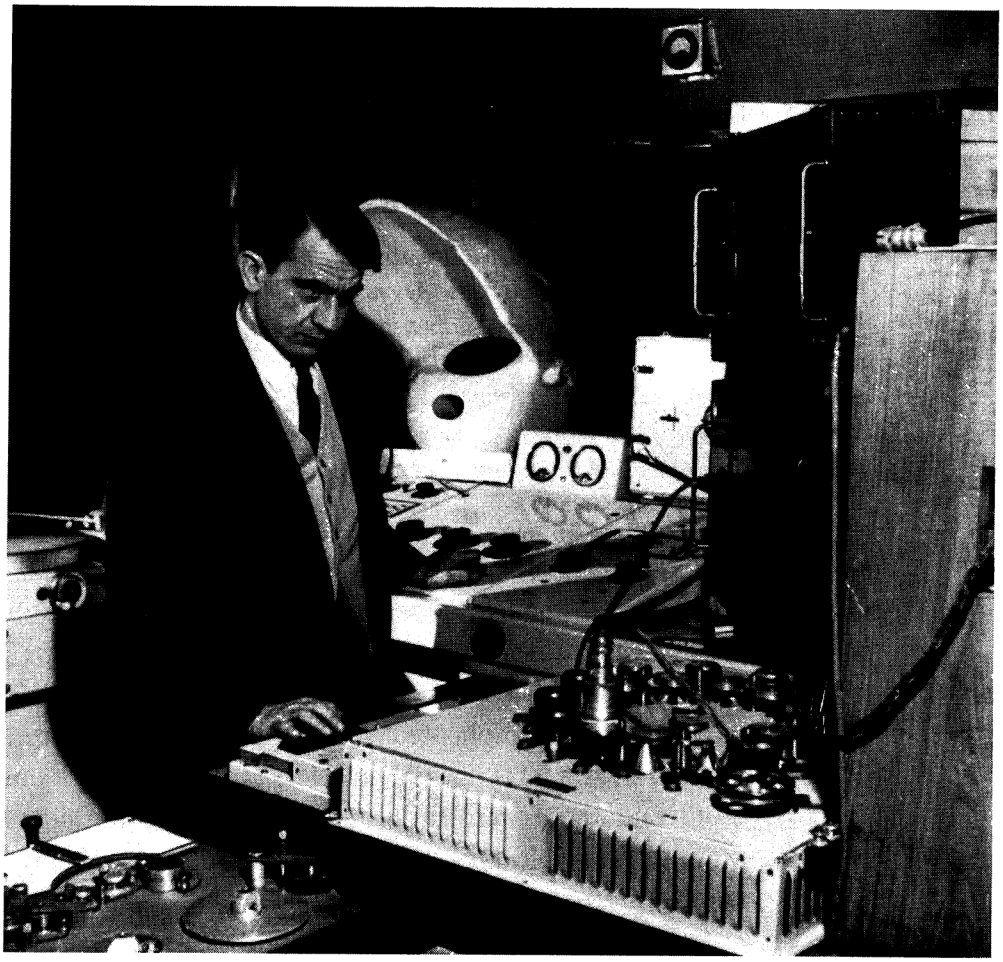


Figure 4.2 Pierre Schaeffer with the Phonogène, a tape-based transposer and time-stretcher, 1953, Paris. (Photograph by Lido, supplied by the courtesy of the Groupe de Recherches Musicales.)

such sounds. Composers of musique concrète work directly with sound objects (Schaeffer 1977; Chion 1982). Their compositions demand new forms of graphic notation, outside the boundaries of traditional scores for orchestra (Bayle 1993).

Modern sampling instruments are based on a principle used in photo-electric and tape-loop devices such as the Edwin Welte's Light-tone Organ (Berlin, 1930s), Sammis's Singing Keyboard (Hollywood, 1936), Pierre Schaeffer's Phonogène (figure 4.2, Paris, early 1950s), Hugh LeCaine's Special Purpose Tape Recorder (Ottawa, 1955), the Chamberlain (Los Angeles, late 1960s), and the Mellotron (London, early 1970s). These devices played either optical disks (encoded with photographic images of waveforms), or magnetic tape loops of sound. Depending on the disk or tape selected and

the key pressed on the musical keyboard, a playback head inside these instruments would play the sound on the disk or tape running at a rate that matched the pitch specified by the depressed key.

The designer of the Singing Keyboard, Frederick Sammis, described the potential of such an instrument in 1936:

> *Let us suppose that we are to use this machine as a special-purpose instrument for making "talkie" cartoons. At once it will be evident that we have a machine with which the composer may try out various combinations of words and music and learn at once just how they will sound in the finished work. The instrument will probably have ten or more sound tracks recorded side by side on a strip of film and featuring such words as "quack" for a duck, "meow" for a cat, "moo" for a cow.... It could as well be the bark of a dog or the hum of a human voice at the proper pitch.* (Frederick Sammis, quoted in Rhea [1977].)

Perhaps the most famous predigital "sampler" was the Mellotron—an expensive instrument containing a number of rotating tape loops. The Mellotron enjoyed popular success with rock groups in the 1970s. They used the instrument to create "orchestral" or "choral" backings on popular songs. But the complicated electromechanical design of the Mellotron made it a temperamental instrument. The tape loops wore out due to head abrasion, and there were failures in the moving parts used in selecting and running multiple tape loops. Despite their problems, Mellotrons piqued interest in the prospect of playing recorded natural sounds on stage.

Several years later, the rise of digital electronics made it feasible to record and store sound in digital memory chips. In the 1970s, however, memory chips were still expensive, so the first "sampling" devices were simple delay units for the recording studio, designed to enrich a sound by mixing it with a sampled version of itself delayed by several milliseconds. (See chapter 10 for a discussion of delay effects.) As memory became cheaper it became possible to store several seconds of sounds for playback on a musical keyboard–based digital sampling instrument. The Fairlight Computer Music Instrument (CMI) was the first commercial keyboard sampler (1979, Australia). The CMI had a resolution of 8 bits per sample and cost over $25,000. Taking advantage of declining costs for digital hardware, the E-mu Emulator (figure 4.3), introduced in 1981, lowered the cost of 8-bit monophonic sampling (Vail 1993). For about $9000, the Emulator offered a total of 128 Kbytes of sample memory.

In order to create a commercial sampling instrument, three basic issues must be addressed: looping, pitch-shifting, and data reduction, which we discuss in the next three sections.

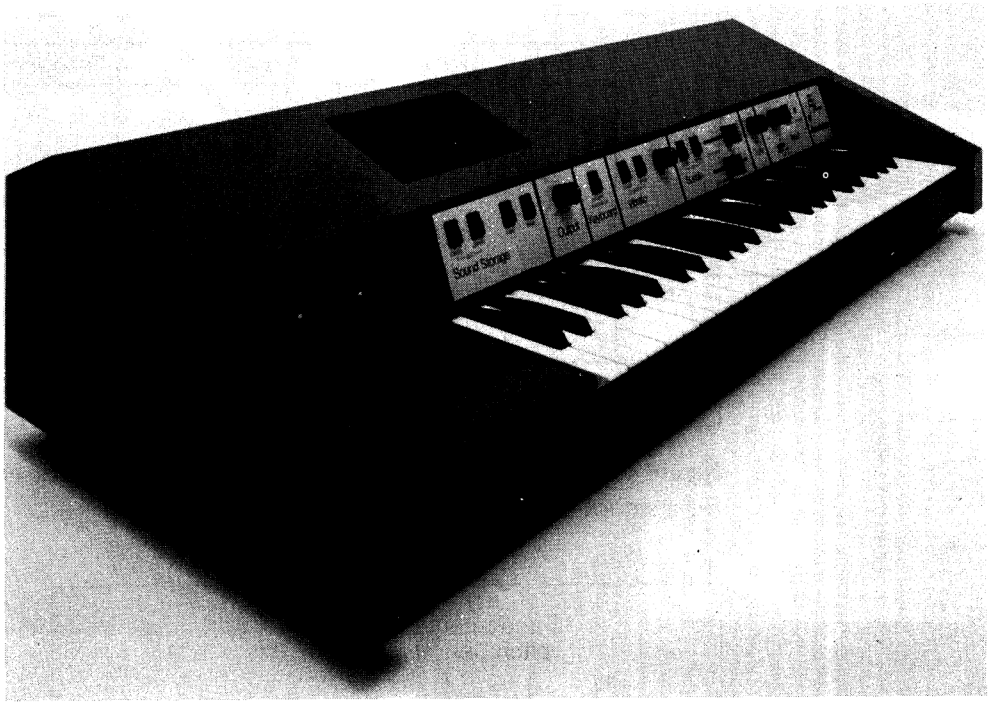


Figure 4.3 The E-mu Emulator sampling keyboard instrument (1981).

Looping

Looping extends the duration of sampled sounds played by a musical keyboard. If the musician holds down a key, the sampler should scan "seamlessly" through the note until the musician releases the key. This is accomplished by specifying beginning and ending *loop points* in the sampled sound. After the attack of the note is finished, the sampler reads repeatedly through the looped part of the wavetable until the key is released; then it plays the note's final portion of the wavetable.

Factory-supplied samples are often "prelooped." But for newly sampled sounds, the responsibility of specifying the begin and end loop points is usually left to the musician who sampled them. Creating a seamless but "natural" loop out of a traditional instrument tone requires care. The loop should begin after the attack of the note and should end before the decay (figure 4.4).

Some samplers provide automatic methods for finding prospective loop points. One method is to perform *pitch detection* on the sampled sound (Massie 1986). (See chapter 12 for a discussion of pitch detection methods.) The pitch detection algorithm searches for repeating patterns in the wavetable that indicate a fundamental *pitch period.* The pitch period is the time

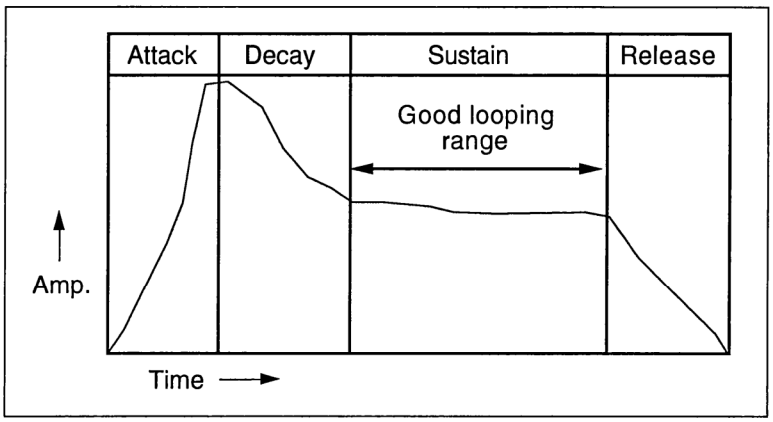


Figure 4.4 Sound with a characteristic ADSR amplitude envelope. The best area for a smooth loop is the sustained portion.

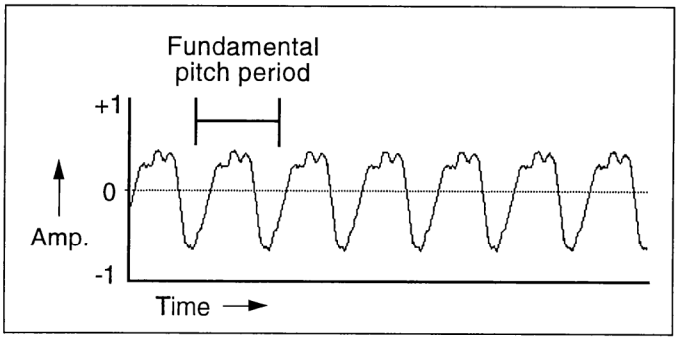


Figure 4.5 The fundamental pitch period is equal to one cycle of a periodic waveform, in this case, a waveform emitted by an alto saxophone.

interval that spans one cycle of a periodic waveform (figure 4.5). Once the pitch has been estimated, the sampler suggests a pair of loop points that match some number of pitch periods in the waveform. This kind of looping algorithm tends to generate smooth loops that are constant in pitch. If the body of the loop is too short, however, the result is similar to the sterile tones of fixed-waveform synthesis. For example, a loop encompassing one or two pitch periods of a violin note negates the time-varying qualities of a bowed string, yielding an artificial tone that has lost its identity.

The beginning and ending points of a loop can either be *spliced* together at a common sample point or *crossfaded.* A splice is a cut from one sound to the next. Splicing waveforms results in a click, pop, or thump at the splice point, unless the beginning and ending points are well matched. Crossfading means that the end part of each looped event gradually fades out while the beginning part slowly fades in again; the crossfade looping process repeats

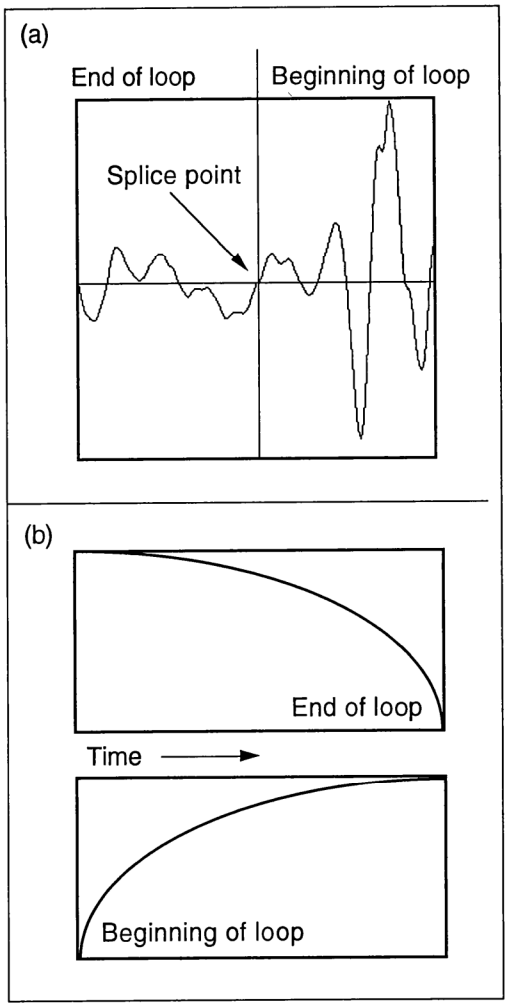


Figure 4.6 Splicing versus crossfading loops. (*a*) A vertical splice of two parts of a waveform at a common zero point. The ending point of the loop splices to the beginning of the same wavetable loop. (*b*) Crossfade looping can be viewed as a fade out of the end of the loop overlapped by a fade in of the beginning of the loop.

over and over as the note is sustained (figure 4.6). Typical crossfade times range from 1 to 100 ms, but crossfades can be extended as long as is desired.

When none of these techniques create a smooth loop, due to vibrato or other variations in the signal, more complicated methods can be brought to bear, such as *bidirectional looping*. A bidirectional loop alternates between forwards and backwards playback (figure 4.7a). Forwards and backwards loops can be layered on top of one another to mask discontinuities in either direction (figure 4.7b). Even more elaborate looping techniques based on spectrum analysis are available. For example, one can analyze the sound, randomize the phase of each spectral component in the loop, and resynthesize (Collins 1993).

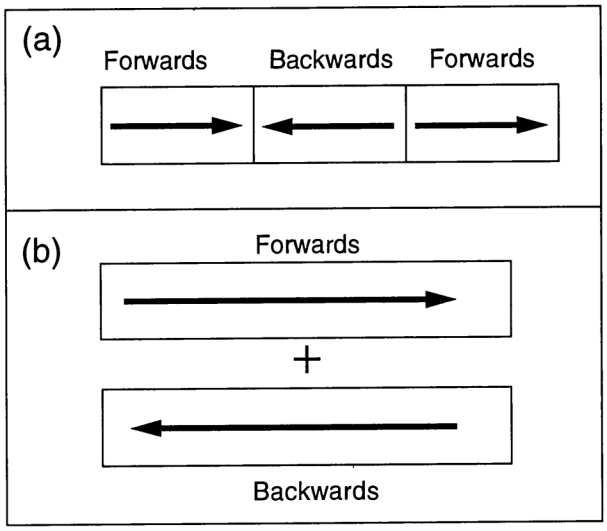


Figure 4.7 Looping methods for smoothing out variations. (*a*) Three cycles of a bidirectional loop. (*b*) In a layered forwards and backwards loop the two versions are added together.

Pitch-shifting

In an inexpensive sampler it may not be possible to store every note played by an acoustic instrument. These samplers store only every third or fourth semitone and obtain intermediate notes by shifting the pitch of a nearby stored note. If you record a sound into a sampler memory and play it back by pressing different keys, the sampler carries out the same pitch-shifting technique. A side effect of simple pitch shifting is that the sound's duration increases or decreases, depending on the key pressed. Chapter 10 describes methods of pitch shifting that preserve the original duration of the sound. Here we stay with simple pitch shifting.

Two methods of simple pitch shifting exist.

Method 1. Varying the clock frequency of the output DAC changes the playback sampling rate; this shifts the pitch up or down and changes the duration.

Method 2. Sample-rate conversion (*resampling* the signal in the digital domain) shifts the pitch inside the sampler and allows playback at a constant sampling rate for all pitches.

Some samplers employ the first method, others use the second method. Both of these methods are called *time-domain* techniques, since they operate directly on the time-domain waveform. This is different from the *frequency-*

domain pitch-shifting techniques discussed in chapter 10. Next we compare these two time-domain methods.

Since method 1 changes the playback sampling rate, it requires a separate DAC for each note that can be played simultaneously on the musical keyboard (typically up to 10 DACs). Each DAC must permit a variable clock rate and have a variable-frequency smoothing filter associated with it. For full transposibility, the DAC and the filter must work over extremely wide operating ranges. For example, if a tone with a pitch of 250 Hz sampled at 44.1 KHz is shifted up six octaves to 16 KHz, the clock frequency of the output DAC must shift up six octaves to 2.82 MHz.

Because of these ranges, either expensive parts must be used or (more typically) the audio performance of the system must be compromised in some ways. For example, one sampler that employs this pitch-shifting method allows only a single semitone of transposition (less than a 6% change of clock frequency) for sounds recorded at its highest sampling rate of 41.67 KHz. In this case the DAC and the filter are never forced to work at a sampling rate higher than 44.1 KHz. Other samplers do not permit any transposition above an arbitrary frequency.

Pitch-shifting method 2 performs sample-rate conversion. Sample-rate conversion, in effect, resamples the signal in the digital domain. This is essentially the same pitch variation technique as used in wavetable-lookup synthesis described in chapter 3. The output DAC's sampling frequency remains constant. Speeding up a sound and increasing its pitch is achieved by resampling at a lower sampling rate. This is analogous to time-lapse photography in which the frame rate is slowed down to achieve a speedup on playback. In a digital audio system samples are skipped in resampling. The number of samples that are skipped is proportional to the amount of pitch shifting that is desired (just as in wavetable-lookup synthesis). The process of skipping samples in resampling is called *decimation* (figure 4.8a). Resampling with decimation is also called *downsampling*. For example, to shift the pitch upwards by three octaves, the signal is downsampled by reading every third sample in playback.

To lower the pitch of a sound and slow it down, the sound is resampled at a higher frequency to stretch it out. This is analogous to the operation of a slow-motion camera that speeds up the frame rate to achieve a slowdown upon playback. In a digital audio system, new intermediate samples are inserted between existing samples by means of *interpolation* (figure 4.8b). Resampling with interpolation is also called *upsampling*.

The relationship between the various resampling rates and pitch-shifting can be confusing at first, because pitch-shifting method 1 and method 2 seem to go in opposite directions to achieve the same aim. Method 1 raises

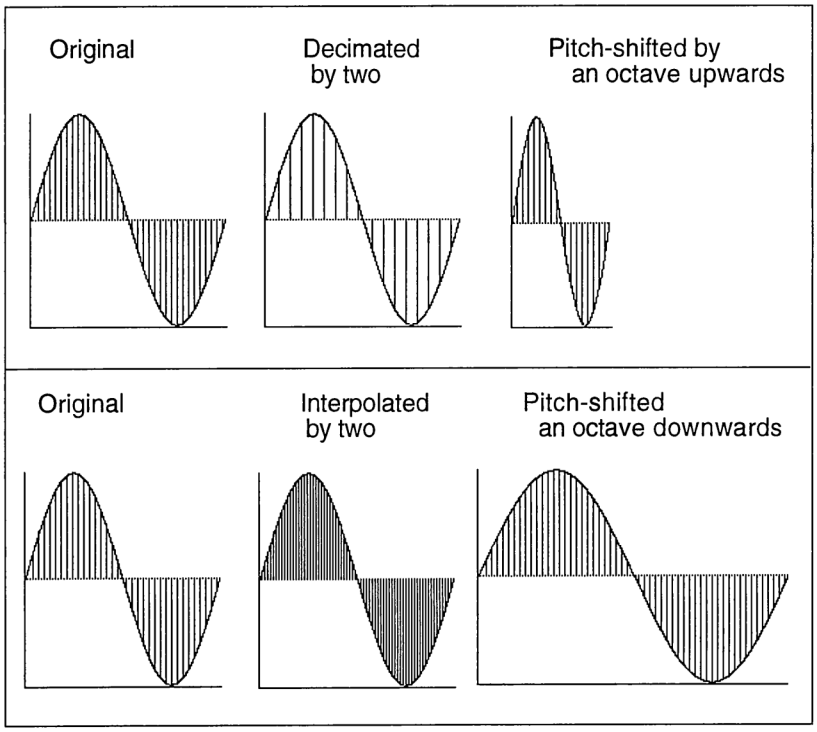


Figure 4.8 Pitch-shifting by sample-rate conversion with a constant playback sampling frequency. (Top) If every other sample is skipped on playback, the signal is decimated and the pitch is shifted up an octave. (Bottom) If twice the number of samples are used by means of interpolation on playback, the signal is shifted down an octave.

pitch by increasing the sampling rate on playback. Method 2, however, raises pitch by decreasing the resampling rate with decimation (downsampling), even though the playback sampling frequency is constant.

So far we have seen how to shift pitch by octave intervals. To shift pitch by any integer ratio, a combination of interpolation and decimation are used (Schafer and Rabiner 1973a; Moorer 1977; Rabiner 1983; Lagadec 1983; Crochiere and Rabiner 1983; Hutchins 1986a; Duncan and Rossum 1988). In particular, to pitch shift by a ratio N/M, we first interpolate by M and then decimate by N. For example, to shift a sound down by an interval of 3/4 (a perfect fourth) we upsample and interpolate by a factor of 4 and then downsample and decimate by a factor of 3. To shift up by a factor of 4/3 we first interpolate by 3 and then decimate by 4.

Sample-rate Conversion without Pitch-shifting

Many digital audio recorders operate at the standard sampling rates of 48 or 44.1 KHz. How can we resample a recording at one of these frequencies

so as to play it back at the other frequency with no pitch shift? In this case the resampling rate is the same as the new output DAC sampling rate.

To convert a signal between the standard sampling rates of 44.1 and 48 KHz without a pitch change, a rather elaborate conversion process is required. First the rates are factored:

$$\frac{48000}{44100} = \frac{2^5 \times 5}{3 \times 7^2} = (4/3 \times 4/7 \times 10/7).$$

These ratios can be implemented as six stages of interpolations and decimations by factors of 2, 3, 5, and 7.

1. Interpolate by 4 from 44,100 to 176,400 Hz
2. Decimate by 3 from 176,400 to 58,800 Hz
3. Interpolate by 4 from 58,800 to 235,200 Hz
4. Decimate by 7 from 235,200 to 33,600 Hz
5. Interpolate by 10 from 33,600 to 336,000 Hz
6. Decimate by 7 from 336,000 to 48,000 Hz

The signal can then be played back at a sampling rate of 48 KHz with no change of pitch.

As long as the input and output sampling rates can be written as a simple fraction, then the conversion process is straightforward. If the rates do not have an integer ratio or they are constantly changing, then more sophisticated mathematical techniques must be used, which we will not venture into here (see Crochiere and Rabiner 1983; Rabiner 1984; Lagadec 1984). This is the case with *flanging effects* (see chapter 10) and audio *scrubbing* (simulating the manual back-and-forth motion of a magnetic tape rocking across a playback head to locate a splice point).

Problems in Resampling

The audio fidelity of resampling is limited by the precision of the hardware used in the conversion. When there are many intermediate resampling stages, a slight loss in fidelity in the form of added noise is to be expected. Aliasing (see chapter 1) can also be a problem. This is because resampling, like the original sampling process, can generate unwanted spectral artifacts due to aliasing. When a sampler skips samples in decimation, for example, it is throwing away intermediate samples. These intermediate samples may have smoothed the waveform's transition between two disjoint points. Thus

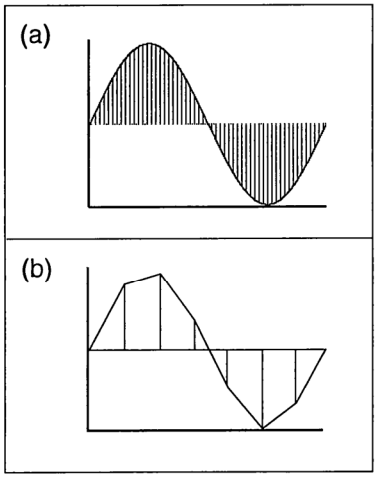


Figure 4.9 With enough decimation, even a sine wave can be turned into a jagged waveform. (*a*) Original sinusoidal waveform. (*b*) Decimation of (*a*) by a factor of eight.

a decimated signal is often full of jagged discontinuities (figure 4.9). At the same time, all frequencies are shifted up, meaning that aliasing can occur on playback. This problem can be reduced to a minimal effect by lowpass filtering the signal after decimation. Filtering smooths the jagged edges of the decimated waveform.

Filtering is also needed in interpolation because simple linear interpolation creates aliased components. Rather than devising a more complicated interpolation scheme, the usual approach in sample-rate conversion is to combine linear interpolation with filtering to shift the frequency content and also minimize aliasing.

Data Reduction and Data Compression in Samplers

The price of semiconductor memory has declined dramatically since it was introduced in the early 1970s. It is still not practical, however, to store a large library of sounds in semiconductor memory. To fit even a subset of such a library into their limited memories, many samplers use *data reduction* or *data compression* strategies to reduce the storage burden. The two are quite different. Data reduction throws away what it considers to be "non-essential" data, while data compression merely makes use of redundancies in the data to code it in a more memory-efficient form. Data compression can reconstitute the original data, while data reduction involves a loss of the original data. Both methods are sometimes grouped under the rubric of *coding* or *encoding* schemes in the audio literature.

Data Reduction

Most samplers do not have facilities for sound analysis and "intelligent" data reduction. In order to reduce the amount of memory needed for storage of audio samples, manufacturers have sometimes taken crude measures that directly affect audio quality. For example, an obvious way to reduce the data stored in a sampler is to limit the sample resolution or quantization (see chapter 1). Some inexpensive sample players use 12 bits or fewer to represent a sample. A variation on this is a *floating-point* encoding scheme that stores the samples in a low-resolution form along with several bits that indicate the original amplitude of the sound (Pohlmann 1989a). Despite shifts in apparent dynamic range, however, the signal-to-noise ratio of the low-resolution samples remains low. Another method is lowering the sampling rate. This diminishes the number of samples stored per unit of time, at the cost of shrinking the audio bandwidth. A third way is to store only every third or fourth note in the range of an instrument and then pitch-shift those samples to play in between pitches. This has the side effect of shifting the spectrum, which is not ideal. If the sound contains any variation like tremolo or vibrato, the rate of these variations is also affected noticeably by pitch-shifting. As the cost of memory declines, there is less and less justification for methods that uniformly compromise audio quality.

A more sophisticated approach to data reduction starts from an analysis stage, which stores sounds in a data-reduced form along with *control functions* that approximately reconstitute it. There are many possible approaches to this analysis and resynthesis. For example, the analysis may take into account *masking phenomena* and throw away those parts of a sound that are supposedly masked by louder parts. (For an introduction to masking, see chapter 23; for further details see Buser and Imbert 1991.) Later in this chapter we look at four experimental data reduction methods based on an additive synthesis model. Several commercial data reduction schemes are built into consumer audio products. This is not the place to enter into a broader discussion of the completeness of the perceptual models on which data reduction schemes are based. Suffice it to say that in any data reduction scheme there is a loss of data leading to a reduction in audio quality. These losses are especially apparent in musical material that exploits the full range of a fine audio system.

Data Compression

To conserve memory space, some systems use data compression techniques to limit the amount of space taken up by a stream of samples. This is done

through the elimination of data redundancies and should not involve any sacrifice in audio quality. One common compression method is *run-length encoding*. The basic idea of run-length encoding is that every sample value is not stored. Rather, each sample that is different from the previous sample is stored, together with a value that indicates how many subsequent samples repeat that value. (For more on audio data compression see Moorer 1979b.)

Sample Libraries

Since a sampler is a type of recording system, the quality of the samples depends on the quality of the recording techniques. Making high-quality samples requires good players with fine instruments, excellent microphones, and favorable recording environments. Arranging all these elements for a large library of sounds takes a great deal of effort. Thus most users of samplers prefer to supplement their collection of samples with libraries prepared by professionals and distributed on magnetic or optical disks.

An Assessment of Samplers

Despite advances in sampling technology, samplers retain a "mechanistic" sound quality that makes them distinguishable from the animated sounds produced by good human performers. Most percussionists, for example, would not mistake the frozen sound of a sampled drum solo from that of a human drummer. In a live performance on acoustic drums, each drum stroke is unique, and there are major differences in the sound depending on the musical context in which the stroke is played. This is not to say that robotic performance is invalid. The commercial success of drum machines proves that lock-step rhythms and unvarying percussion sounds have a major audience.

In any case, it is understandable that the "naturalness" or "realism" of a sampler should be held up as a criterion for judging between different brands. It is well known that a given instrument tone may sound much more realistic on one sampler than it does on another.

Certain instruments, like organs, can be modeled more or less realistically by most samplers. That is, they all can generate a high-quality recording of a pipe or electronic organ. Other instruments like voices, violins, saxophones, electric guitars, and sitars are intrinsically more difficult to capture with existing sampling technology. Individual notes can be captured reasonably well, but when we put the notes together into phrases, melodies, and chords, it is apparent that major chunks of acoustic and performance information have been left out.

Factory-supplied samples model the generic singer, the generic saxophone played by the generic saxophonist, the generic orchestra played in the generic concert hall, and so on. Yet most knowledgeable listeners can tell the difference between two vocalists, saxophonists, and conductors with orchestras. It would be difficult to mistake a MIDI sequencer/sampler rendition of a saxophone solo for the signature style of the John Coltrane original. This points out a fundamental limitation in existing samplers. Beyond a certain point, it is impossible to increase the realism of present samplers without major advances in technology and in understanding of the relationship between sound structure and musical performance. One obvious evolutionary path for samplers is analysis/resynthesis (see chapter 13), which permits flexible, context-sensitive transformations of musical sounds.

In expressive instruments like voices, saxophones, sitars, guitars, and others, each note is created in a musical context. Within a phrase, a note is reached from another note (or from silence) and leads into the successive note (or leads to silence). In addition to these contextual cues, transitional sounds like breathing, tonguing, key clicks, and sliding fingers along strings punctuate the phrasing. Constraints of style and taste determine when context-sensitive effects such as rubato, portamento, vibrato, crescendi and diminuendi, and other nuances are applied.

These problems can be broken into two parts: (1) How can we model the sound microstructure of note-to-note transitions? (2) How can we interpret (analyze) scores to render a context-sensitive performance according to style-specific rules? These questions are the subject of the next two brief sections.

Modeling Note-to-note Transitions

The problem of what happens in note-to-note transitions was the subject of the doctoral research of John Strawn at Stanford University (1985b). He analyzed the transitions in nine nonpercussive orchestral instruments. The time- and frequency-domain plots that emerged from this research graphically depicted the context-sensitive nature of tone successions.

In wind instruments, one of the ways to articulate a transition is by *tonguing*—a momentary interruption of the windstream by an action of the tongue, as if the player were pronouncing the letter *t* or *k*. Figure 4.10 shows a time-domain plot of transitions of a trumpet played tongued (a) and untongued (b). The contrast between the two types of transitions is clear.

Figure 4.11 plots the spectrum of this transition. Strawn's research demonstrated that some transitions are very smooth, with a dip of as little as

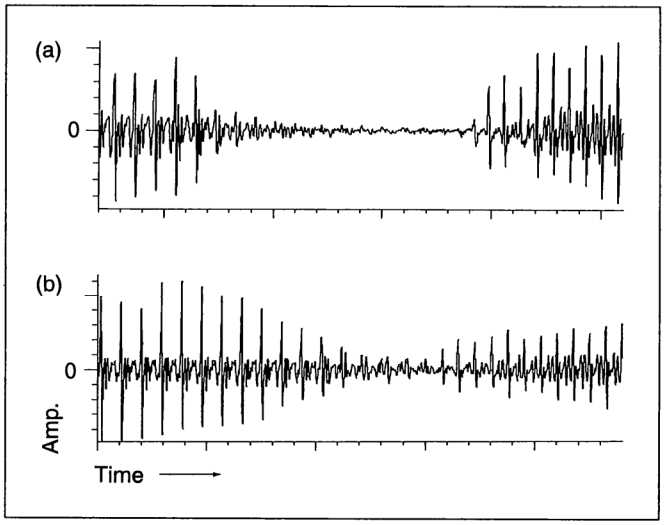


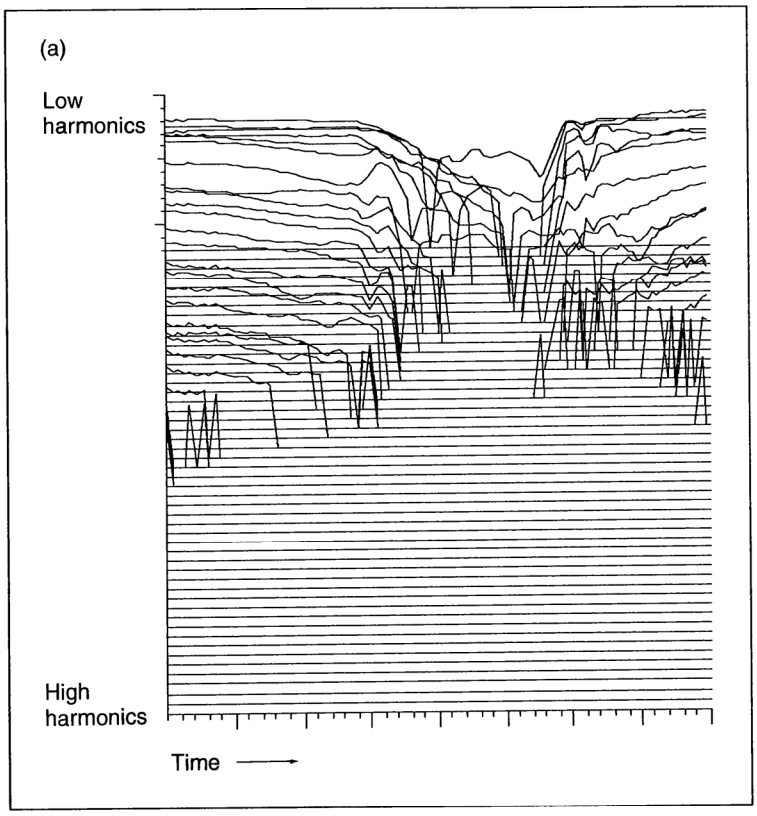
Figure 4.10 Time-domain plot of note-to-note transition of an ascending major third interval for a trumpet played tongued (*a*) and untongued (*b*). The time span for the plots is about 120 ms. (Courtesy of John Strawn.)

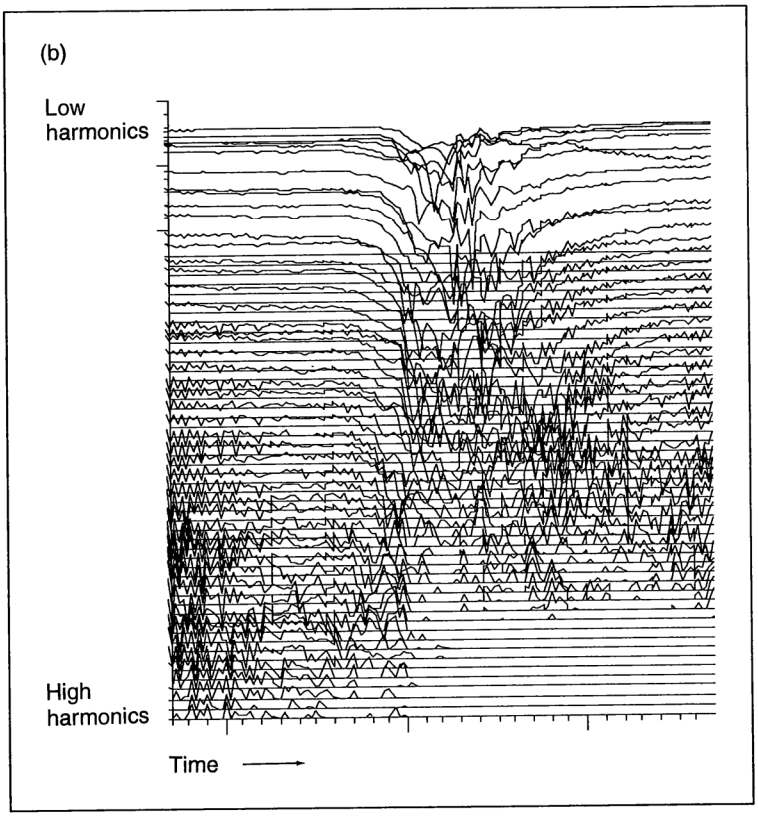
10 dB of amplitude between notes. Other transitions are laden with strong transitional cues in amplitude and spectrum changes that articulate the attack of the second note.

Modeling the microstructure of note-to-note transitions appears to be a tractable problem, since its solution depends on an expectable advance in technology. The problem could be solved by an increase in sampler memory capacities (storing all two-note transitions), fast signal processing, or some combination of the two. The diphone method, for example, stores transition data in a form that allows them to be stretched or compressed (Rodet, Depalle, and Poirot 1988). Holloway and Haken (1992) model transitions as overlapping tracks in a tracking phase vocoder (see chapter 13).

If transitions are to be calculated automatically—as a musician plays on a keyboard, for example, the instrument must be able to make a very quick determination of context. (Chapter 15 discusses the related issue of machine interpretation of musical scores.)

Figure 4.11 Spectrum plots of the transitions shown in figure 4.10. The plots show 50 harmonics plotted over a time span of 300 ms, with lower harmonics at the back. (*a*) Tongued. (*b*) Untongued. Notice how the "hole" in the middle of (*a*) is filled in when the note transition is untongued (more continuous). (Courtesy of John Strawn.)


(a)
Low
harmonics
High
harmonics
Time


(b)
Low
harmonics
High
harmonics
Time

Additive Synthesis

Additive synthesis is a class of sound synthesis techniques based on the summation of elementary waveforms to create a more complex waveform. Additive synthesis is one of the oldest and most heavily researched synthesis techniques. This section starts with a brief history of additive synthesis and explains its fixed-waveform and time-varying manifestations. The next section is devoted to the process of analysis/resynthesis—linking an analysis of a sound to a resynthesis stage based on additive synthesis.

Additive Synthesis: Background

The concept of additive synthesis is centuries old, first being applied in pipe organs by means of their multiple *register-stops*. By pulling on a register-stop, air could be routed to a set of pipes. The air was actually released into the pipe—creating sound—by pressing a key on the organ keyboard. By pulling several register-stops in various proportions one could add together the sound of several pipes for each key pressed on the organ musical keyboard. According to one scholar, "The Middle Ages particularly favored the 'mixtures' in which every note was accompanied by several fifths and octaves based upon it" (Geiringer 1945). This idea of frequency "mixtures" is the essence of additive synthesis.

Additive synthesis has been used since the earliest days of electrical and electronic music (Cahill 1897; Douglas 1968; *die Reihe* 1955; Stockhausen 1964). The massive Telharmonium synthesizer unveiled in 1906 summed the sound of dozens of electrical tone generators to create additive tone complexes (figure 4.12).

Incorporating a miniature version of the Telharmonium's rotating tone generators, the famous Hammond organs were pure additive synthesis instruments (figure 4.13). The power of additive synthesis derives from the fact that it is theoretically possible to closely approximate any complex waveform as a sum of elementary waveforms. Methods exist for analyzing a sound such as a violin tone and resynthesizing it using time-varying combinations of sine waves of various frequencies, phases, and amplitudes. Due to intrinsic limitations in the resolution of the analysis, however, this reconstructed version is never a sample-for-sample replication of the original signal (see chapter 13).

Any method that adds several elementary waveforms to create a new one could be classified as a form of additive synthesis. For example, some forms of granular synthesis discussed in chapter 5 could be called additive synthe-

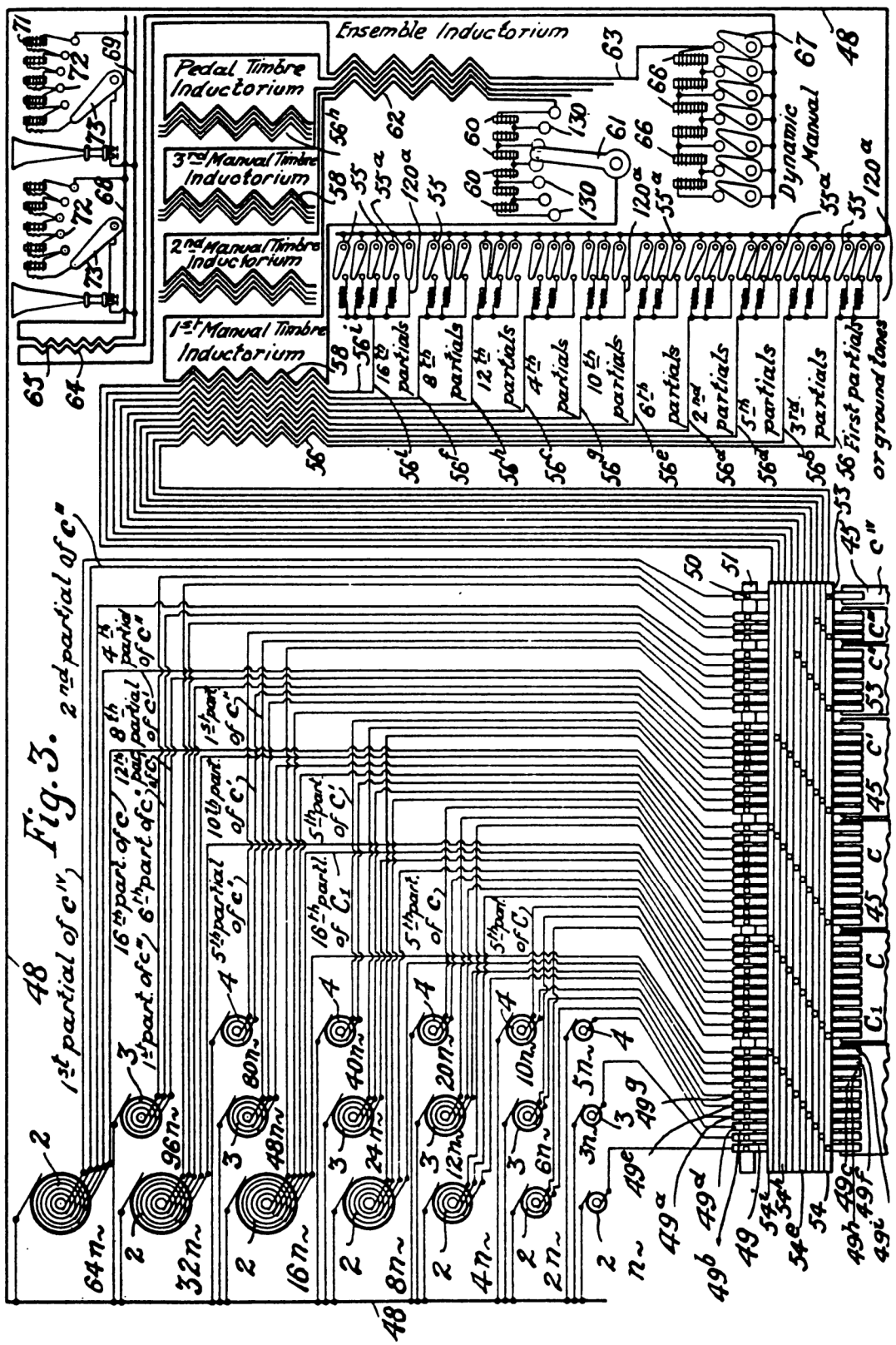


Figure 4.12 Additive synthesis of a complex tone in the Telharmonium. Sine wave harmonics from the tone-generating alternator are fed to bus bars (54). Pressing a key (C in this case) connects each harmonic to a multicoil transformer (56 "inductorium") where they mix. Each harmonic is attentuated to the desired level by the inductors in series with each winding (56a, b, etc.). The tap-switch inductors (60) regulate the amplitude of the mixing transformer output, as do the inductors near the loudspeakers (72, 73) at the listener's end of the transmission line. (Cahill patent drawing, reproduced in Johnson et al. 1970.)

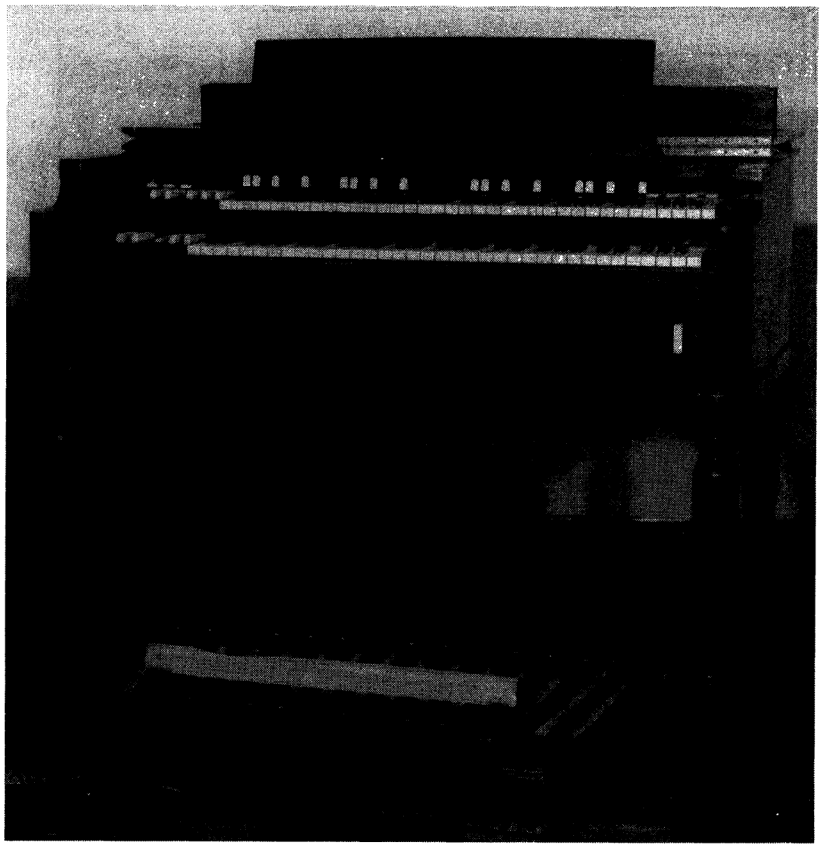


Figure 4.13 Hammond B3 organ, an additive synthesis instrument based on electromechanical tone-wheels. Different mixtures of the various harmonics can be adjusted by pulling "drawbars" above the musical keys. (Photograph courtesy of the Institute of Organology, Kunitachi College of Music, Tokyo.)

sis techniques (Risset and Wessel 1982). However, we have separated these techniques from additive synthesis in this chapter in order to clarify the distinctions between the classical method of sine wave additive synthesis and those methods.

Fixed-waveform Additive Synthesis

Some software packages and synthesizers let the musician create waveforms by *harmonic addition.* In order to make a waveform with a given spectrum the user adjusts the relative strengths of a set of *harmonics* of a given fundamental. (The term "harmonic" as an integer multiple of a fundamental frequency was first used by Sauveur [1653–1716] in 1701.) For example, 400 Hz is the second harmonic of 200 Hz, since 2 times 200 equals 400. The harmonics can be displayed as a *bar graph* or *histogram,* with the height of each bar representing the relative strength of a given harmonic. Figure 4.14 shows a harmonic spectrum and the corresponding waveform.

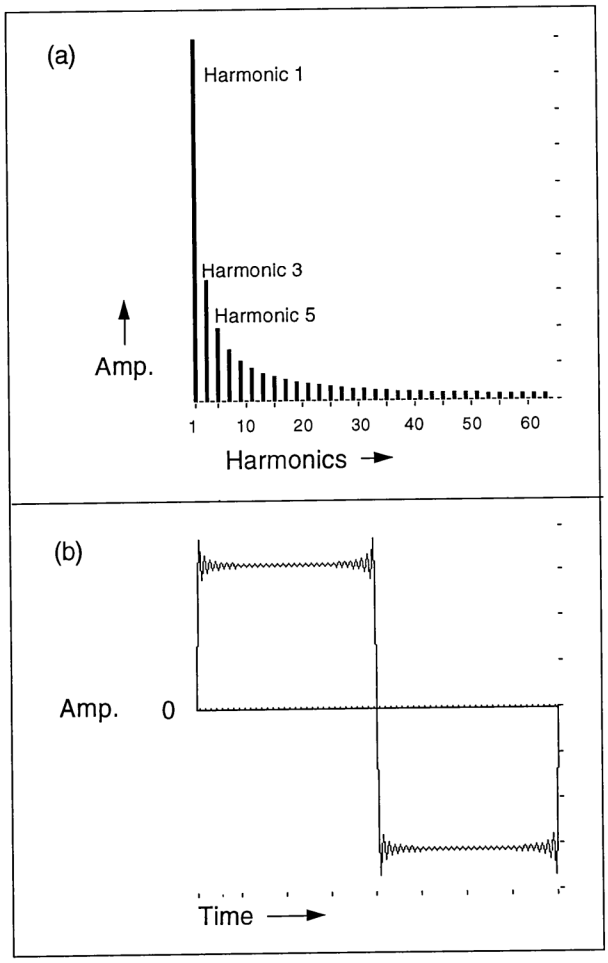


Figure 4.14 Waveform synthesis by harmonic addition. (*a*) Histogram showing the relative strength of the harmonics on a linear scale. In this case, the histogram has energy only in the odd harmonics. The amplitude of the third harmonic is one-third that of the fundamental, the amplitude of the fifth harmonic is one-fifth that of the fundamental, and so on. (*b*) Approximation to a square wave synthesized by harmonic addition using the histogram in (*a*).

Once a desirable spectrum is tuned, the software calculates a waveform that reproduces the spectrum when it is played by a digital oscillator. This spectrum template aligns to different frequencies when one changes the pitch of the oscillator. Figure 4.15 shows successive stages of waveform addition used to create a quasi-square wave.

The Phase Factor

Phase is a trickster. Depending on the context, it may or may not be a significant factor in additive synthesis. For example, if one changes the starting phases of the frequency components of a fixed waveform and resynthesizes the tone, this makes no difference to the listener. And yet such a

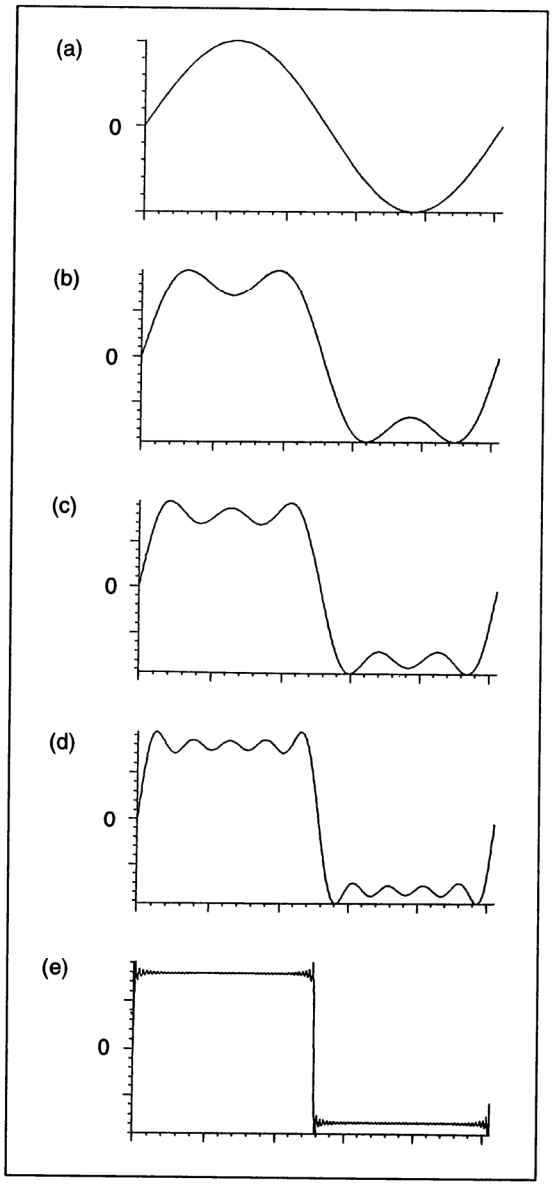


Figure 4.15 Stages of harmonic addition as seen in a series of time-domain waveforms. (*a*) Fundamental only. (*b*) First and third harmonics. (*c*) Sum of odd harmonics through the fifth. (*d*) Sum of odd harmonics through the ninth. (*e*) Quasi-square wave created by summing odd harmonics up to the 101st.

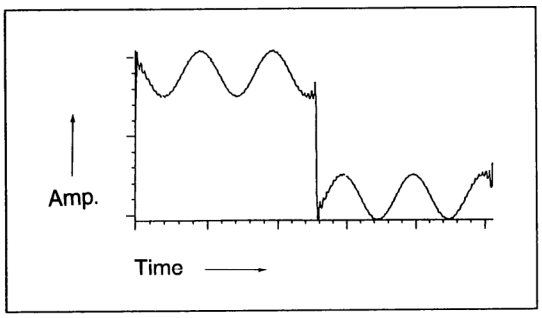


Figure 4.16 Effect of phase in additive synthesis. This waveform is the result of the same mixture of sine waves as in figure 4.15e except that the starting phase of the fifth harmonic is 90 degrees instead of 0 degrees.

change may have a significant effect on the visual appearance of the waveform, as shown in figure 4.16.

Phase relationships become apparent in the perception of the brilliant but short life of attacks, grains, and transients. The ear is also sensitive to phase relationships in complex sounds where the phases of certain components are shifting over time. As we see later in the section on sound analysis and resynthesis, proper phase data help reassemble short-lived components in their correct order, and are therefore essential in reconstructing an analyzed sound.

Addition of Partials

We can generalize from addition of harmonics to addition of *partials*. In acoustics, a *partial* refers to an arbitrary frequency component in a spectrum (Benade 1990). The partial may or may not be a harmonic (integer multiple) of a fundamental frequency *f*. Figure 4.17a shows a spectrum containing four partials: two harmonic and two *inharmonic*. An inharmonic partial is not in an integer ratio to the fundamental frequency. Figure 4.17b is the waveform that results from the sum of the four partials.

Addition of partials is limited in that it succeeds only in creating a more interesting fixed-waveform sound. Since the spectrum in fixed-waveform synthesis is constant over the course of a note, partial addition can never reproduce accurately the sound of an acoustic instrument. It approximates only the *steady-state* portion of an instrumental tone. Research has shown that the *attack* portion of a tone, where the frequency mixture is changing on a millisecond-by-millisecond timescale, is by more useful for identifying traditional instrument tones than the steady-state portion. In any case, a time-varying timbre is usually more tantalizing to the ear than a constant spectrum (Grey 1975).

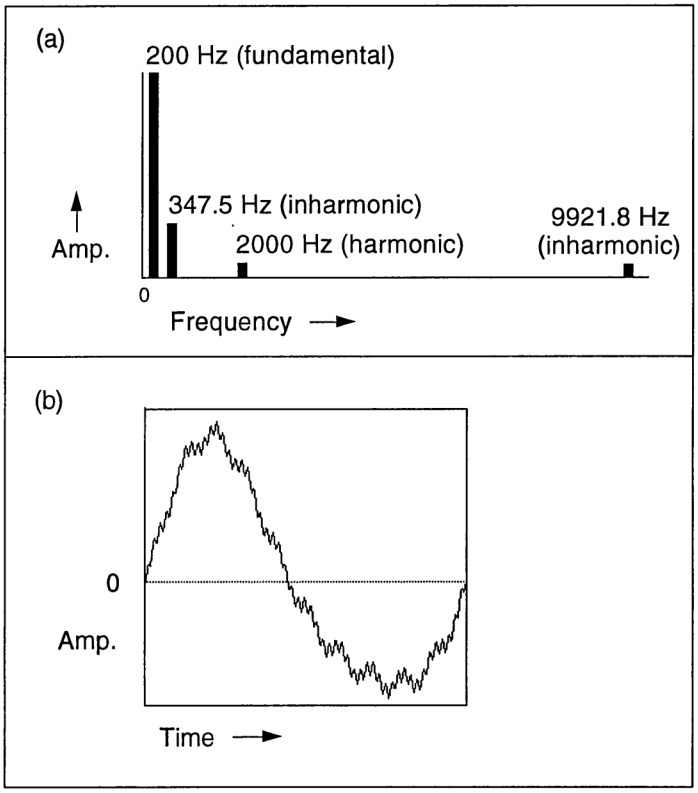


Figure 4.17 Partial addition with four components. The percentage contribution of each component is 73, 18, 5, and 4 percent, respectively. (*a*) Frequency-domain view. (*b*) Time-domain waveform.

Time-varying Additive Synthesis

By changing the mixture of sine waves over time, one obtains more interesting synthetic timbres and more realistic instrumental tones. In the trumpet note in figure 4.18, it takes twelve sine waves to reproduce the initial attack portion of the event. After 300 ms, only three or four sine waves are needed.

We can view the process of partial addition graphically in several ways. Figure 4.19a shows additive synthesis in the analog domain, as it was practiced in the 1950s (Stockhausen 1964). The figure shows several oscillator hardware modules, each with a manually controlled frequency knob. The outputs of the oscillators are routed to a mixer. The composer adjusted the balance of the oscillators in real time to determine the time-varying spectrum. With this setup, manual control was the only option. To precisely realize a time-varying mixture took several people working together (Morawska-Büngler 1988).

Figure 4.19b shows digital additive synthesis. An audio oscillator is represented as a quasi half-circle with a pair of inputs—one for amplitude and

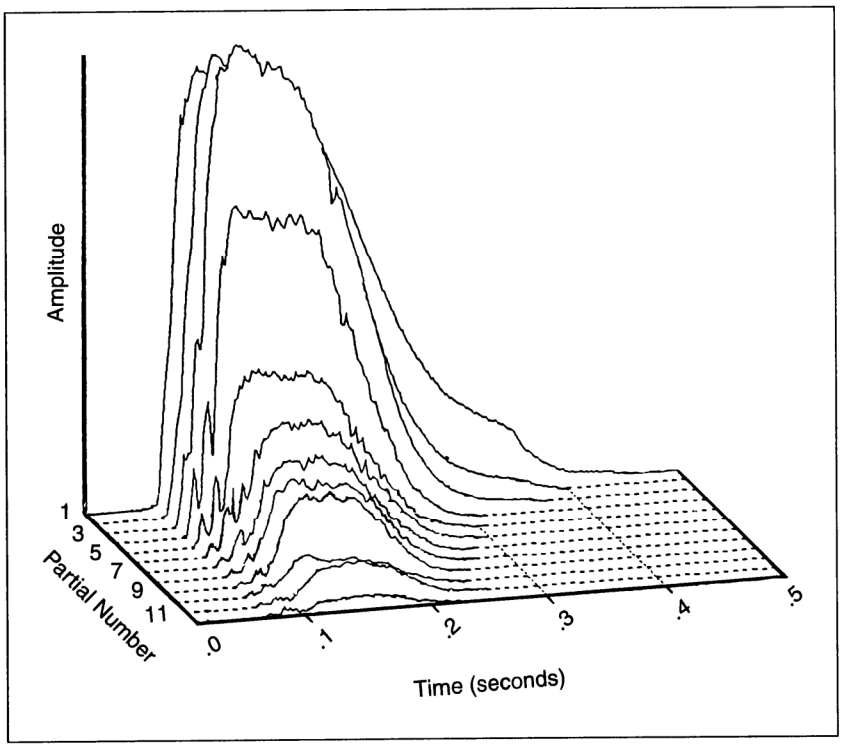


Figure 4.18 Time-varying spectrum plot of twelve partials of a trumpet tone, with the highest partials in the foreground. Time goes from left to right. Notice that the fundamental (at the back) is not the highest amplitude, but it lasts the longest.

one for frequency. To generate a time-varying spectrum, each frequency and amplitude input to the oscillators is not a constant but a time-varying envelope function read over the duration of the event. The sine wave audio oscillators feed into a module that sums the signals. The sum module then passes the additive result to a DAC for conversion to sound.

Demands of Additive Synthesis

Time-varying additive synthesis makes heavy demands on a digital music system. First, it requires large numbers of oscillators. If we make the musically reasonable assumptions that each sound event in a piece may have up to 24 partials (each generated by a separate sine wave oscillator), and that up to sixteen events can be playing simultaneously, we need up to 384 oscillators at any given time. If the system is running at a sampling rate of 48 KHz, it must be capable of generating 48,000 × 384 or 18,432,000 samples/second. Since each sample requires about 768 operations (multiply-adds), the total computational load is over 1.4 billion operations per second, without counting table-lookup operations. Table 20.1 in chapter 20

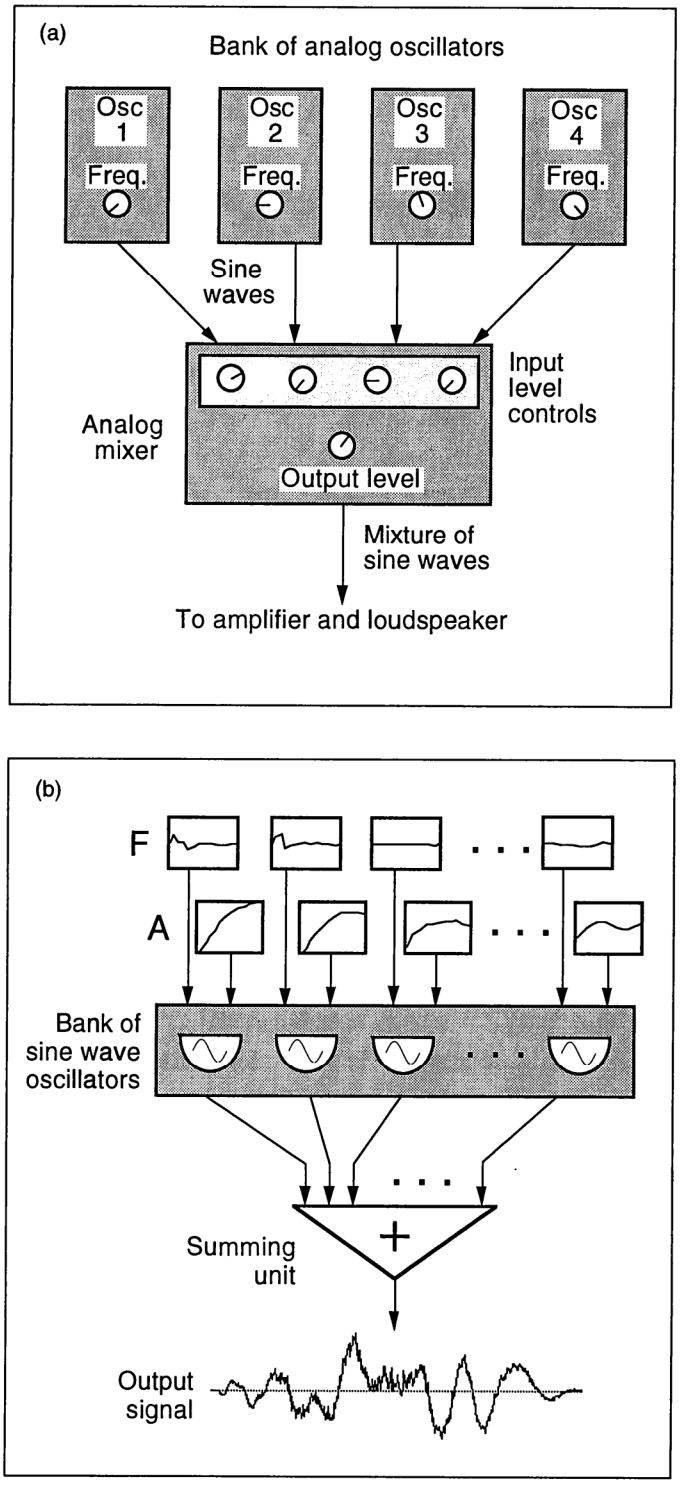


Figure 4.19 Two views of additive synthesis. (*a*) In the analog domain, oscillators feed a mixer. (*b*) Digital additive synthesis. Time-varying additive synthesis with separate frequency (F) and amplitude (A) envelopes. Figure 3.10 shows a more detailed instrument diagram for additive synthesis.

estimates the per-sample requirements for additive synthesis. Such computational demands, although formidable, are not outside the limits of current hardware. For example, one synthesizer specialized for additive synthesis offers the potential of several thousand sine waves in real time (Jansen 1991).

Yet computational power is only one requirement of additive synthesis. The method also has a voracious appetite for control data. If a piece contains 10,000 events (such as a typical orchestral score), each with up to 24 partials, one needs to have 240,000 frequency envelopes and 240,000 amplitude envelopes on hand. Even if the same envelope is used in more than one event, where do the control data come from? This is the subject of the next section.

Sources of Control Data for Additive Synthesis

Effective use of any digital synthesis technique—including additive synthesis—depends on having good control data for the synthesis instrument. To create animated sounds with a rich internal development, one drives the synthesizer with control data; hence, the control data are also referred to as the *driving functions* of the synthesis instrument. Control data can be obtained from several sources:

1. Imported from another domain and mapped into the range of synthesis parameters. For example, some composers have traced the shape of mountains or urban skylines and used these curves as control functions. This is the approach used in the early computer music piece *Earth's Magnetic Field* (1970) by Charles Dodge and in pieces derived purely from geometric, stochastic, or other mathematical or physical models.

2. Generated by a composition program that embodies composer-specified constraints on musical microstructure. An example is John Chowning's *Stria* (1977), realized with additive synthesis of inharmonic spectra.

3. Generated by an interactive composition system that translates high-level musical concepts such as *phrases* (in the Formes language of Rodet and Cointe 1984), *tendency masks* (as in the POD system of Truax 1977, 1985), *sound objects* (as in the SSSP system of Buxton et al. 1978), or *clouds* (as in asynchronous granular synthesis of Roads 1978c, 1991) into synthesis parameters.

4. Entered manually by the composer, using combinations of the previously mentioned sources or the composer's intuitive, theoretical, or empirical knowledge of psychoacoustics. An example of this method is Jean-Claude Risset's piece *Inharmonique* (1970).

5. Supplied from an analysis subsystem that mulches a natural sound and spews out the control data needed to resynthesize it. The data can also be edited in order to create transformations of the original sounds. Trevor Wishart (1988) used sound analysis as an intermediate stage in transforming vocal sounds for his piece *Vox-5* (see also Murail 1991).

Since methods 1 to 4 are based on compositional aesthetics, we do not discuss them further in this chapter. The fifth method requires a subsystem for sound analysis; this is the subject of the next section.

Additive Analysis/Resynthesis

Analysis/resynthesis encompasses different techniques that have a three-step process in common (figure 4.20):

1. A recorded sound is analyzed
2. The musician modifies the analysis data
3. The modified data are used in resynthesizing the altered sound

The concept of analysis/resynthesis is not predicated solely on additive synthesis. It can also be based on subtractive resynthesis (see chapter 5),

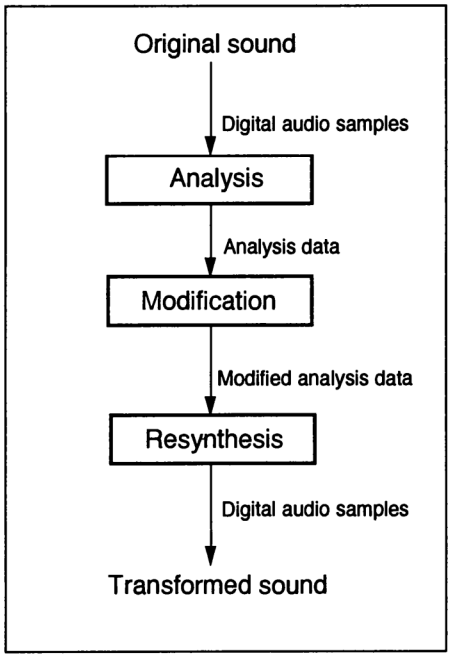


Figure 4.20 General overview of analysis/resynthesis. The modification stage may involve manual edits to the analysis data or modifications via *cross-synthesis* where the analysis data of one sound scale the analysis data from another sound.

combinations of additive and subtractive resynthesis (Serra 1989; Serra and Smith 1990), or other methods (see chapter 13).

Early experiments in additive analysis/resynthesis were carried out by H. Fletcher (of the famous Fletcher-Munson loudness curves) and his associates (Fletcher, Blackham, and Stratton 1962; Fletcher, Blackham, and Christensen 1963). They used entirely analog equipment. When digital additive methods are used for resynthesis, the entire system looks like figure 4.21. The analysis is carried out successively on short segments of the input signal. The process of segmenting the input signal is called *windowing* (discussed in chapter 13 and appendix A). We can think of each windowed segment as being sent through a bank of narrow bandpass filters, where every filter is tuned to a specific center frequency. In practice, a *fast Fourier*

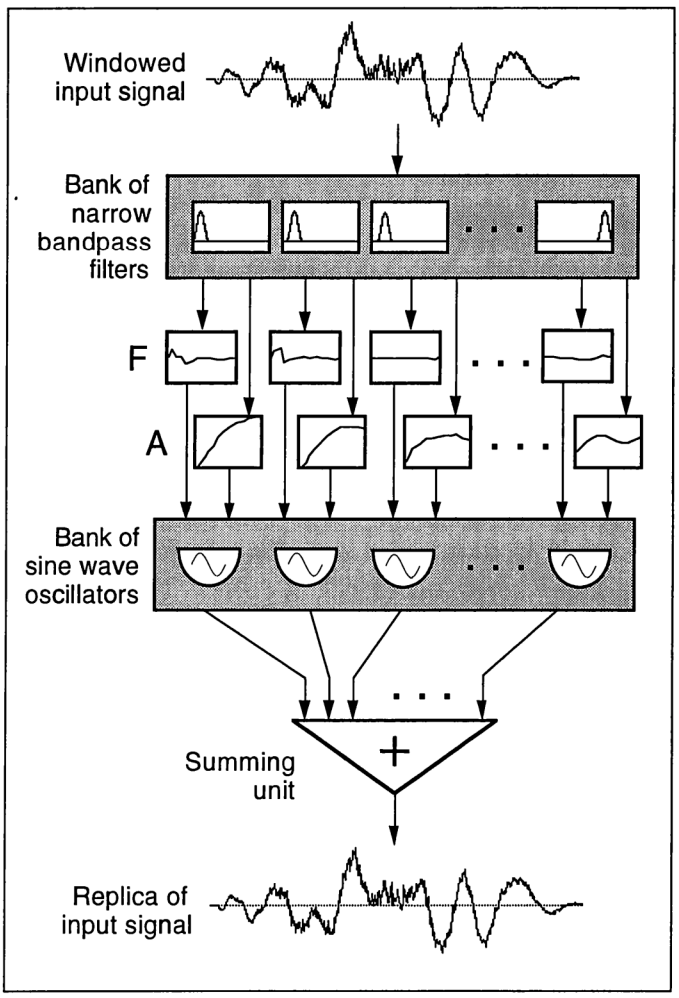


Figure 4.21 Additive analysis/synthesis. A windowed input signal is analyzed by a filter bank into a set of frequency (F) and amplitude (A) envelopes or *control functions* that drive a set of oscillators. If the analysis data are not changed, the output signal should be almost the same as the input signal.

transform (FFT) usually replaces the filter bank and performs essentially the same task in this application, that is, measuring the energy in each frequency band (again, see chapter 13 and appendix A).

The amplitude of the signal coming out of each filter is measured, and this time-varying value becomes the *amplitude control function* for that frequency range. At the same time, the system calculates control functions corresponding to small frequency deviations by looking at the output of adjacent filters (or *analysis bins,* in the case of the FFT).

The frequency and amplitude control functions drive a bank of oscillators in the resynthesis stage. In other words, we are using the information gleaned from the analysis of an existing sound to create the set of control functions needed to resynthesize that sound additively with sine waves. If the input sound is well modeled as a sum of sine waves, the summed signal generated by the oscillators should be much the same as the original input signal.

Of course, straightforward analysis/resynthesis of a sound is not interesting from a musical standpoint. In order to create musically interesting effects, we must modify the data generated by the analysis. This is the subject of the next section.

Musical Applications of Additive Analysis/Resynthesis

After the analysis has been performed, musicians can edit the control functions to create variations of the original input signal. Many different effects are possible with this technique, as listed in table 4.1. Three compositions produced in the 1980s stand as good examples of compositional manipulation of analysis data: *Mortuos Plango, Vivos Voco* (1981) by Jonathan Harvey, *Désintegrations* (1983, Salabert Trajectoires) by Tristan Murail, and *Digital Moonscapes* (1985, CBS/Sony) by Wendy Carlos.

In the Harvey piece, the composer analyzed the sound of a large bell. For each sinusoidal component in the resynthesis, the composer substituted the sound of a sampled boy's voice at the appropriate frequency. The voice samples followed the analyzed frequency and amplitude control functions of the chiming bells, creating an eerie effect of a boy/bell chorus. In the composition by Murail, the composer analyzed traditional instrument tones and created synthetic complements to these tones that blend seamlessly with the sounding instruments yet weave out dramatically when the instruments stop. *Désintegrations* is a classic example of *spectral composition* techniques where the harmonic structure of the work is based on an analysis of instrumental tones (Murail 1991). In *Digital Moonscapes,* Carlos used analysis data as the inspiration for creating an ad hoc synthetic orchestra of per-

Table 4.1 Musical transformations using additive analysis/resynthesis

Musical effect	Technique
Variations of recorded sounds	Change selected frequency or amplitude envelopes by editing or multiplications by arbitrary functions.
Spectrum scaling (without time scaling)	Multiply the frequency of all the partials (possibly excepting the fundamental) by a factor n or by arbitrary functions. Since multiplication does not preserve formant structures, vocal and instrumental sounds may lose their characteristic identity.
Spectrum shifting (without time scaling)	Add a factor n or an arbitrary function to all partials (possibly excepting the fundamental). For small values this preserves formant structures.
Spectrum inversion	Reversing the order of the frequency components before resynthesis, so that the amplitude of the first partial is assigned to the last partial, and vice versa, followed by exchange of the amplitudes of the second and next-to-last components, etc.
Hybrid timbres	Replace some envelopes from one sound with selected envelopes from another sound.
Time expansion and compression without pitch shifting	Extend the duration of the frequency and amplitude envelopes, or change the *hop size* on playback (see chapter 13).
Stretch a percussive timbre into a prolonged synthetic passage	Delay the onset time of each partial and smooth their envelopes.
Timbral interpolation from one instrumental tone to another	Interpolate over time between the envelopes of two instrument tones.
Mutating synthetic sounds	Interpolate between the envelopes of arbitrary synthetic sounds.
Enhance the resonance regions of recorded sounds	Increase the amplitude of selected frequency partials.
Cross-synthesis	Method 1: Use the amplitude envelopes for the partials of one sound to scale the amplitude envelopes of another sound (see *fast convolution* in chapter 10).
	Method 2: Apply the amplitude envelopes from one sound to the frequency (or phase) functions of another sound.
	Method 3: Apply the noise residual from one sound to the quasi-harmonic part of another sound (see, for example, the description of spectral modeling synthesis and the comb wavelet transform in chapter 13).

cussion-, string-, woodwind-, and brasslike timbres, used in an idiomatic orchestral style.

The next section briefly discusses current techniques of sound analysis with additive resynthesis with an emphasis on the data reduction problem. It serves as a prelude to the more detailed treatment in chapter 13 and appendix A.

Methods of Sound Analysis for Additive Synthesis

Many spectrum analysis methods, including *pitch-synchronous* analysis (Risset and Mathews 1969), the *phase vocoder* (Dolson 1983, 1986, 1989b), and *constant-Q* analysis (Petersen 1980; Schwede 1983; Stautner 1983), among others, are variations on the basic technique of Fourier analysis of component frequencies. The practical form of Fourier analysis is the *short-time Fourier transform* (STFT). This method can be thought of as analyzing a sampled sound by extracting successive short-duration overlapping segments (shaped by a window function) and applying a bank of filters to the selected segment. The output of each filter is measured, indicating the amplitude and the phase of the spectrum at that particular frequency. A series of these short-time analyses (akin to the frames of a film) constitute a time-varying spectrum. At the core of the STFT is the FFT, a computationally efficient implementation of Fourier analysis (Cooley and Tukey 1965; Singleton 1967; Moore 1978a, 1978b; Rabiner and Gold 1975).

The phase vocoder (PV) (Flanagan and Golden 1966; Portnoff 1978; Holtzman 1980; Moorer 1978; Dolson 1983; Gordon and Strawn 1985; Strawn 1985b) deserves special mention here, as it is a popular method of sound analysis/resynthesis that has been distributed with several music software packages. The PV converts a sampled input signal into a time-varying spectral format. In particular, it generates a set of time-varying frequency and amplitude curves. Many interesting sound transformations can be achieved by editing and resynthesizing PV data. For example, the phase vocoder can be used for *time compression* or *time expansion* without pitch change. In this effect a sound is made longer or shorter without significantly affecting its pitch or timbre. (See chapter 10 for a discussion of various approaches to time compression/expansion.)

Contrary to the expectations of the researchers who invented them (who were searching for efficient coding techniques), sound analysis techniques may generate an "information explosion" (Risset and Wessel 1982). That is, the analysis data (the control functions) can take up many times more memory space than the original input signal. The amount of data depends partly on the complexity of the input sound, that is, how many sine wave

functions are needed to resynthesize it, and partly on the internal data representation used in the analysis program. Using the tracking phase vocoder, for example, a short sound file that takes up 2 Mbytes may generate tens of Mbytes of analysis data. Such storage requirements make it difficult to build libraries of analyzed sounds, and the volume of data becomes onerous to edit. This situation mandates some form of data reduction of the control data, the subject of the next section.

Data Reduction in Analysis/Resynthesis

Data reduction is important to efficient analysis/resynthesis. Data reduction takes two steps. First, the data—a set of amplitude and frequency control functions—are analyzed. Second, an algorithm transforms the original data into a more compact representation. An important goal of data reduction is to compact data without eliminating perceptually salient features of the input signal. Another important goal in computer music work is that the analysis data must be left in a form that can be edited by a composer. The goal is not simply to save bits; rather, one wants to make it easy to manipulate the data-reduced material (Moorer 1977).

A large body of research work on data reduction of digital audio samples is recorded in the literature, including studies by Risset (1966), Freedman (1967), Beauchamp (1969, 1975), Grey (1975), Grey and Gordon (1978), Charbonneau (1981), Strawn (1980, 1985a, 1985b), Stautner (1983), Kleczkowski (1989), Serra (1989), Serra and Smith (1990), Holloway and Haken (1992), and Horner, Beauchamp, and Haken (1993). Since real-time work is so important to musicians, one goal of analysis/resynthesis research is to speed up data reduction processing and facilitate real-time synthesis from reduced data. Papers by Sasaki and Smith (1980) and Schindler (1984) explain hardware designs for high-speed digital synthesis from reduced data.

Many volumes of engineering literature explore data reduction methods. Here we glance at four techniques that have been applied in computer music: line-segment approximation, principal components analysis, spectral interpolation synthesis, and spectral modeling synthesis. (See also Goldberg 1989 for a description of the genetic algorithm approach, which has recently been applied to synthesis data reduction [Horner, Beauchamp, and Haken 1993].)

Line-segment Approximation

Line-segment approximation of the amplitude and frequency control functions eliminates the need to store a distinct value for every sample analyzed.

Instead, the analysis system stores only a set of *breakpoint pairs,* which are time (x-axis) and amplitude (y-axis) points where the waveform changes significantly. Line-segment approximation represents the overall outline of a waveform by storing only the points of maximum inflection (change). In the resynthesis stage the system "connects the dots," usually by means of straight lines interpolated between the breakpoint pairs.

Initial work with line-segment approximation was done by hand, using an interactive graphics editor to construct functions with four to eight segments each (Grey 1975). A data reduction of a hundredfold was achieved. This manual editing work can also be partially automated, as demonstrated by Strawn (1985a, 1985b). Figure 4.22a shows a perspective plot of the sixteen harmonics of a violin tone, sampled at 25 KHz. Figure 4.22b plots an approximation to (a) using just three line segments.

Going beyond the storage of line-segment approximations, Beauchamp (1975) developed a heuristic technique for inferring the approximate amplitude curve of all harmonics of a tone from the curve of the first harmonic. For simple periodic tones, Charbonneau (1981) found that even more radical data reduction could be achieved. He used simple variations of a single envelope for all amplitude functions of a given tone. (See also Kleczkowski 1989 and Eaglestone and Oates 1990 for refinements of these proposals.)

Principal Components Analysis

The technique of *principle components analysis* (PCA) has been applied in several analysis/resynthesis systems (Stautner 1983; Sandell and Martens 1992; Horner, Beauchamp, and Hakken 1993). PCA breaks down a waveform using the mathematical technique of *covariance matrix* calculation. This results in a set of basic waveforms (the principal components) and a set of weighting coefficients for these basic waveforms. When the components are summed according to their weights, the result is a close approximation of the original waveform.

The advantage of PCA is its potential for data reduction. PCA analysis summarizes the underlying relationships between samples so the fewest number of components account for the maximum possible variance in the signal. The process of determining the principal components and their weighting coefficients is implemented as an iterative approximation that tries to minimize the squared numerical error (difference between the original and the approximation). The first principal component is a fit of a single waveform to the entire data set. The second principal component is a fit to the *residual* (sometimes called *residue*), or difference between the original and first approximation. The third component is a fit to the residual of the

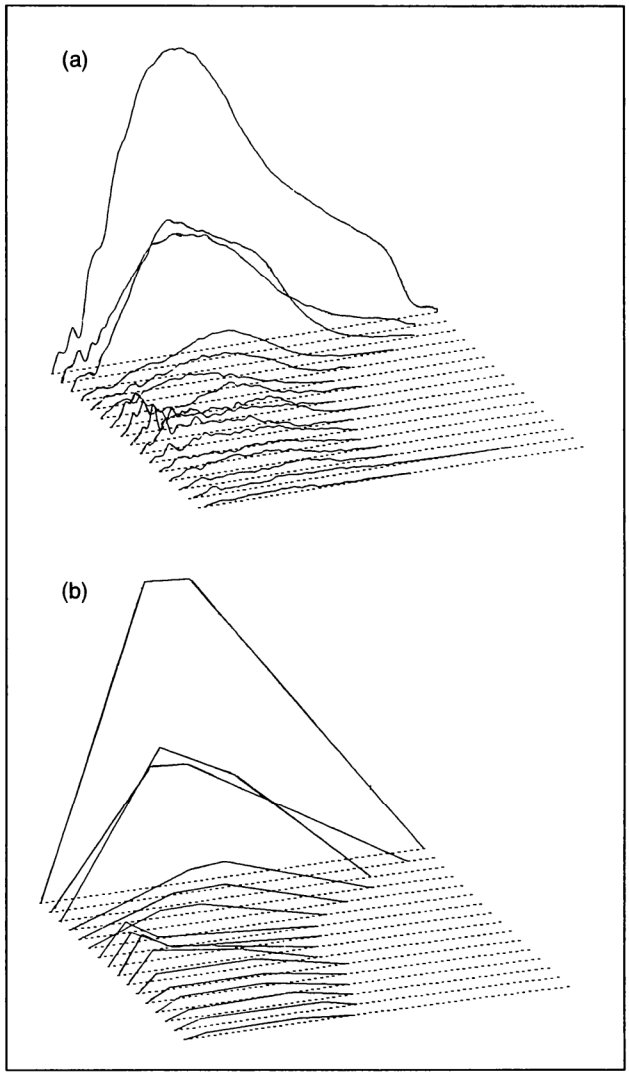


Figure 4.22 Drastic data reduction of analysis data for additive synthesis. Amplitude is plotted vertically, frequency goes from back to front, and time goes left to right. (*a*) Original frequency-vrs-time-vrs-amplitude curve of a violin tone. (*b*) The same violin tone as in (*a*), approximated with only three line segments per partial.

second component, and so on. For further details on PCA see Glaser and Ruchkin (1976).

Spectral Interpolation Synthesis

Spectral interpolation synthesis (SIS) (Serra, Rubine, and Dannenberg 1990) is an experimental technique that generates time-varying sounds by interpolating between analyzed spectra. Rather than crossfading between sampled sounds in the time domain (as in multiple wavetable synthesis discussed in chapter 5), SIS starts from analyses of recorded sounds and

uses additive synthesis to crossfade between the analyses of successive spectra in the frequency domain. An automatic data reduction algorithm is necessary to compress the analysis data into a small set of common spectral paths between two successive sounds and a set of ramp functions that describe the transition from one spectrum to the next. The main difficulty with the procedure appears to be its handling of the attack portion of sounds.

Spectral Modeling Synthesis

Spectral modeling synthesis (SMS) (Serra 1989; Serra and Smith 1990) reduces the analysis data into a *deterministic* component (narrowband components of the original sound) and a *stochastic* component. The deterministic component is a data-reduced version of the analysis that models the most prominent frequencies in the spectrum. These frequencies are isolated by a process of *peak detection* in each frame of the analysis, and *peak continuation,* which tracks each peak across successive frames. SMS resynthesizes these tracked frequencies with sine waves. This is the same method as used in tracking phase vocoders described in chapter 13.

SMS goes beyond this representation, however, by also analyzing the residual or the difference between the deterministic component and the original signal. This is the called the "stochastic" component of the signal. The stochastic component takes the form of a series of envelopes that control a bank of frequency-shaping filters through which white noise is being passed. Thus a composer can transform the deterministic (sine) envelopes and the stochastic (filtered noise) components separately, if desired (figure 4.23). Noisy components remain noisy, even after transformations (such as filtering) are applied to them. This stands in contrast with a pure sine wave model, in which transformations (such as time compression/expansion) on noisy components often turn them into orderly sine wave clusters, denaturing their noisy texture.

Efficient algorithms for generation of *pseudorandom noise* are well known (Knuth 1973a; Keele 1973; Rabiner and Gold 1975). Thus the use of filtered noise results in a tremendous data reduction. In purely sinusoidal resynthesis without this kind of data reduction, noisy components must be approximated with hundreds of sine waves. The control functions for these sine waves take up a great deal of storage space, and the sine wave resynthesis is costly from a computational standpoint.

A problem of accuracy left open by SMS is that the filtered pseudorandom noise it uses to reconstruct the stochastic component is not necessarily the same quality of noise as the original source. In many sounds, "noise" is the result of complicated turbulences that have an audible char-

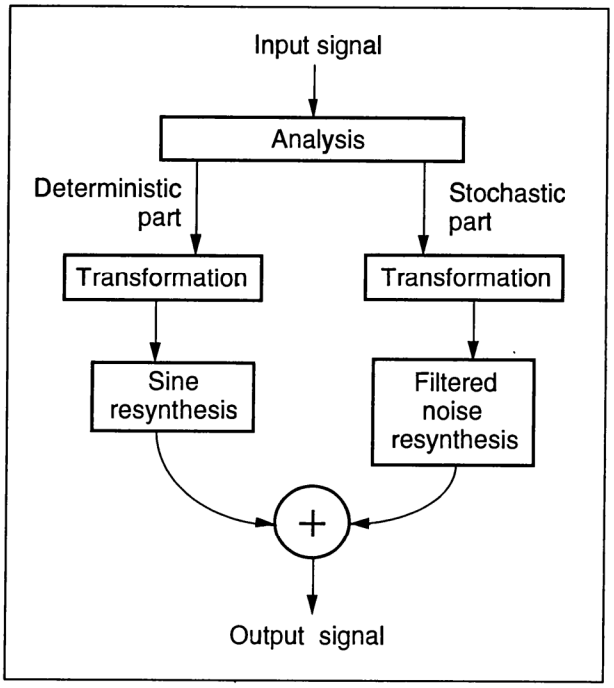


Figure 4.23 Overview of spectrum modeling synthesis. The input signal is divided into a deterministic part and a stochastic part. Each part can be modified separately before resynthesis. (See figure 13.16 for a more detailed view of the analysis stage.)

acter and identity. For some sounds, the approximation by uniform noise leaves room for improvement.

Walsh Function Synthesis

So far we have discussed analysis/resynthesis as a process based mainly on Fourier analysis with resynthesis based on sine wave summation. The Fourier sine wave approach has a long tradition of research and application stemming from the original theorem that states that for periodic signals, a combination of sine waves of various frequencies can be created that approximate arbitrarily closely the original signal. Mathematical research has shown that other groups of waveforms besides sine waves can be used to approximate signals. A family of square waves called the *Walsh functions* can be used to approximate a signal after it has been analyzed by means of the *Walsh-Hadamard transform*. Walsh functions, being rectangular waves, are a kind of "digital domain series," since they take on only two values $+1$ and -1 (Walsh 1923).

Figure 4.24 presents the first eight Walsh functions. Just as with the Fourier series and its sine waves, an arbitrary periodic waveform can be approximated as an additive sum of a finite series of Walsh functions. While

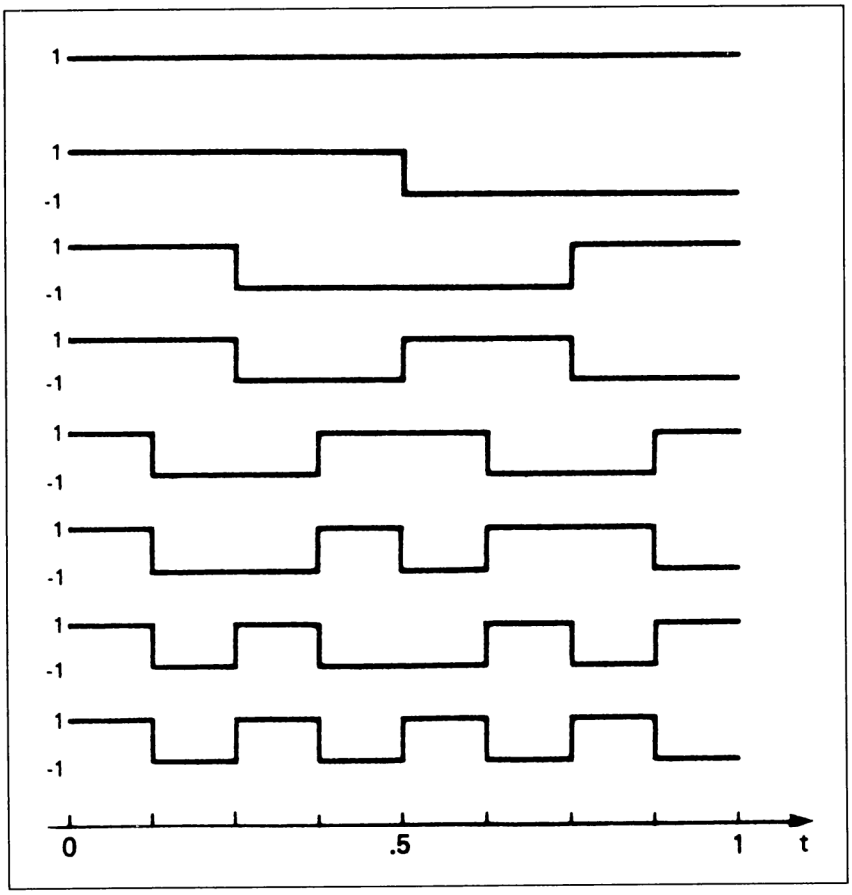


Figure 4.24 The first eight Walsh functions, 0 (top) to 7 (bottom).

the Fourier series builds up waveforms out of component frequencies, Walsh synthesis builds up waveforms using functions of different *sequencies*. Sequency is defined as one-half the average number of zero crossings per second (zps) (Hutchins 1973). Figure 4.25 shows a composite waveform derived by summing several Walsh functions. It suggests how sine wave additive synthesis and Walsh function synthesis are conceptual opposites. That is, the hardest waveform to synthesize in Walsh function synthesis is a pure sine wave. The Walsh approximation to a sine will stay jagged until a very high number of sequency terms are used. Any jaggedness gives an "unsinusoidal" and generally objectionable quality. By contrast, in sine wave synthesis, the hardest waveform to synthesize is one with a rectangular corner, such as a square wave! Figure 4.15, for example, depicts a quasi-square wave constructed by summing 101 sine waves.

The main advantage of Walsh functions in digital sound synthesis is their rectangular shape, a shape that can be computed at high speed by inexpensive digital circuits. A disadvantage of Walsh function synthesis as against

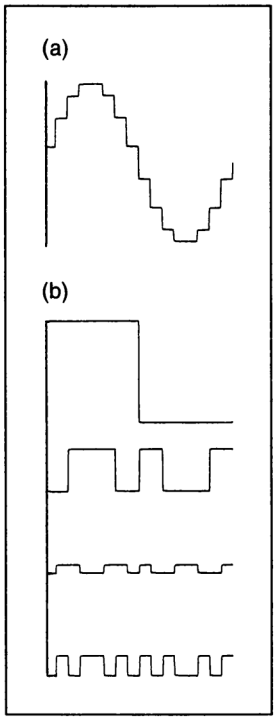


Figure 4.25 Demonstration of Walsh function summation. (*a*) A simple sine wave approximation built by adding the Walsh functions shown in (*b*). (After Tempelaars 1977.)

sine wave synthesis is that individual Walsh functions are not associated with specific harmonics, as they are in sine wave additive synthesis. It is, however, possible to pass mathematically from the Fourier (frequency) domain to the Walsh domain (Tadokoro and Higishi 1978). Thus, one can specify a sound in terms of the addition of various frequency components (partials), and then transform this specification into a set of parameter values for a Walsh function synthesizer. Moreover, natural sounds can be sampled and transformed into the Walsh domain using the Walsh-Hadamard transform and resynthesized using the fast Walsh transform (FWT) (Hutchins 1973, 1975).

A number of music synthesis operations have been redesigned for Walsh signal processing circuits. For example, Hutchins (1973) designed an envelope generator using Walsh function circuits. Rozenberg (1979) and Hutchins (1975) showed how to realize amplitude modulation, subtractive synthesis, frequency modulation, frequency shifting, and reverberation—all in the Walsh domain.

Despite the potential of Walsh function synthesis, only a few experimental devices based on this technique have been built (Hutchins 1973, 1975; Insam 1974). None are commercially available. This is probably due to the

fact that the cost of circuits for sine wave additive synthesis has continued to decline (including memory chips and multipliers), so the economic advantage of Walsh function circuits has diminished. The weight of accumulated research in Fourier/sine wave methods and the more intuitive relationship between frequencies and perception have also contributed to the popularity of sine wave summation in contemporary synthesizer designs.

Conclusion

This chapter has discussed two widely used synthesis techniques: sampling and additive synthesis. The sampler is the mockingbird of musical instruments. Its creative synthesis capabilities may be weak, but it can copy any source through its ability to memorize and playback sound. Because it can also mimic the rich sounds of acoustic instruments, a sampler is among the most popular electronic instruments available.

Additive techniques have been studied in detail for decades. When coupled to an analysis stage, additive synthesis is a powerful means of simulating natural sounds and cloning variations of them. As we have seen, the main drawback of additive techniques is that they achieve a quasi generality by sacrificing computational efficiency. In order to simulate a given sound, a thorough analysis must be performed. This analysis can generate an explosion of data that must go through a data reduction stage before it becomes editable. Sound analysis tools require serious computing power, so in the past they were available only on expensive institutional-grade computing hardware. This situation is changing, with sophisticated sound analysis and editing tools available on even portable computers.

The next chapter discusses two synthesis techniques with links to sampling and additive synthesis, namely multiple wavetable synthesis and granular synthesis. The other technique discussed in chapter 5 is subtractive synthesis, the conceptual opposite of additive synthesis.