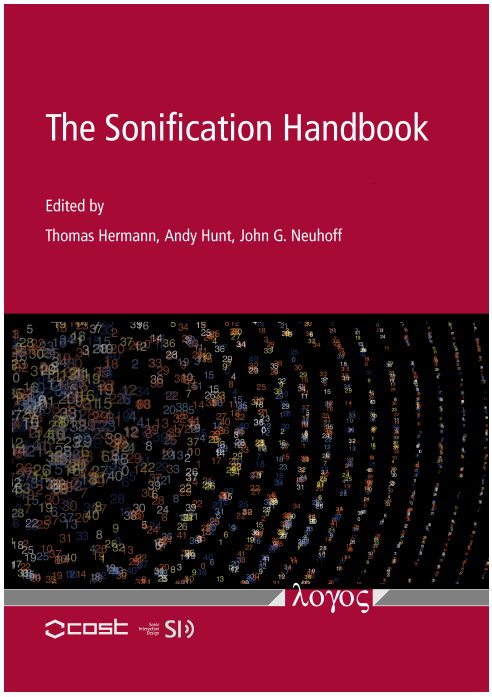


The Sonification Handbook

Edited by
Thomas Hermann, Andy Hunt, John G. Neuhoff

Logos Publishing House, Berlin, Germany
ISBN 978-3-8325-2819-5
2011, 586 pages

Online: http://sonification.de/handbook
Order: http://www.logos-verlag.com

Reference:
Hermann, T., Hunt, A., Neuhoff, J. G., editors (2011). *The Sonification Handbook*. Logos Publishing House, Berlin, Germany.

Chapter 9

Sound Synthesis for Auditory Display

Perry R. Cook

This chapter covers most means for synthesizing sounds, with an emphasis on describing the parameters available for each technique, especially as they might be useful for data sonification. The techniques are covered in progression, from the least parametric (the fewest means of modifying the resulting sound from data or controllers), to the most parametric (most flexible for manipulation). Some examples are provided of using various synthesis techniques to sonify body position, desktop GUI interactions, stock data, etc.

Reference:
Cook, P. R. (2011). Sound synthesis for auditory display. In Hermann, T., Hunt, A., Neuhoff, J. G., editors, The Sonification Handbook, chapter 9, pages 197–235. Logos Publishing House, Berlin, Germany.

Media examples: http://sonification.de/handbook/chapters/chapter9

Chapter 9

Sound Synthesis for Auditory Display

Perry R. Cook

9.1 Introduction and Chapter Overview

Applications and research in auditory display require sound synthesis and manipulation algorithms that afford careful control over the sonic results. The long legacy of research in speech, computer music, acoustics, and human audio perception has yielded a wide variety of sound analysis/processing/synthesis algorithms that the auditory display designer may use. This chapter surveys algorithms and techniques for digital sound synthesis as related to auditory display.

Designers of auditory displays and systems employing sound at the user interface need flexible and powerful means to generate and process sounds. So the approach here is to present the basic technology behind each algorithm, then view it by examining the parameters it provides to the sound/interaction designer to allow manipulation of the final sonic result.

This chapter first walks through most of the major means for synthesizing audio, in descending order from non-parametric to highly parametric, eventually summarizing those in a concluding section. Techniques that begin with pre-recorded audio, then analyze, process, and resynthesize sound, will be followed by techniques that synthesize audio directly "from scratch" or models. But before we proceed, we will take a moment to clearly define "parametric."

9.2 Parametric vs. Non-Parametric Models

Here the word "parametric" is used as it is used in engineering "system identification" (fitting a model to the observable inputs and outputs of some system of interest). A "parametric"

model is defined here as one that has a (relatively) few variable parameters that can be manipulated to change the interaction, sound, and perception. A highly parametric model of sound is the technique known as Linear Predictive Coding (LPC, discussed in a later section), which uses just a few numbers representing the spectral shape, and the (usually voice) source, to represent thousands of PCM samples. A close representation of an original block of samples can be resynthesized by running a source/filter model, informed by the extracted parameters. Further, we can synthesize longer or shorter (by just running the model slower or faster) while retaining all other parameter values such as pitch and spectral shape. We can also synthesize with higher or lower pitch, or turn sung vowels into whispers (less pitch, more noise), because the few parameters we have are meaningful and influential. As the LPC example points out, extracting a few powerful parameters from raw audio is closely related to audio compression and coding.

As a counter example, a non-parametric model of a particular sound would be the raw samples (called PCM as discussed in the next section), because the "model" has no small set of parameters that allows us to modify the sound in meaningful ways. The Fourier Transform (discussed at length two sections from now), while powerful for many reasons, is also a non-parametric model, in that it turns N time waveform samples into N frequency values, but those (equal number of) parameters don't allow us to manipulate the interaction, sound, and perception in a low-dimensional and flexible manner.

Of course, "parametricity" (not likely a word, but used here to represent how parametric a model/technique is), is relative, and a little tricky. If we find a way to represent 10,000 samples of 8-bit wave data with 9,000 bytes, maybe by just running those samples through a standard text symbol compressor such as WinZip, we will have reduced the size of the representation, but the 9,000 bytes aren't really parameters, since they do little to let us manipulate the "resynthesis." On the other hand, we could "code" every song released commercially in digital form by a fairly small and unique "tag" representing the serial number of the CD release, and the track number. This one small tag number is arguably not a "parameter", since even the slightest change in this number will yield a totally different recording of a totally different song.

Thus the definition of a "parameter" in this chapter is: **a (likely continuous) variable that, when changed slightly, yields slight changes in the synthesized sound, and when changed greatly, makes great changes.** The parametricity of the algorithm is determined based on the space of possible output sounds, relative to the number of such parameters. Herein lies flexible power for auditory display, because we can map data to those powerful parameters.

9.3 Digital Audio: The Basics of PCM

Digital audio signals are recorded by *sampling* analog (continuous in time and amplitude) signals at regular intervals in time, and then *quantizing* the amplitudes to discrete values. The process of sampling a waveform, holding the value, and quantizing the value to the nearest number that can be digitally represented (as a specific integer on a finite range of integers) is called Analog to Digital (A to D, or A/D) conversion [1]. A device that does A/D conversion is called an Analog to Digital Converter (ADC). Coding and representing waveforms in sampled digital form is called Pulse Code Modulation (PCM), and digital

audio signals are often called PCM audio. The process of converting a sampled signal back into an analog signal is called Digital to Analog Conversion (D to A, or D/A), and the device which does this is called a Digital to Analog Converter (DAC). Low-pass filtering (smoothing the samples to remove unwanted high-frequencies) is necessary to reconstruct the sampled signal back into a smooth continuous time analog signal. This filtering is usually contained in the DAC hardware.

The time between successive samples is usually denoted as T. Sampling an analog signal first requires filtering it to remove unwanted high frequencies, to avoid "aliasing." Aliasing is caused by under-sampling frequencies higher than half the sample rate, causing them to not be accurately represented, as shown in Figure 9.1. The pre-filtering must eliminate all frequencies higher than half the sampling rate.

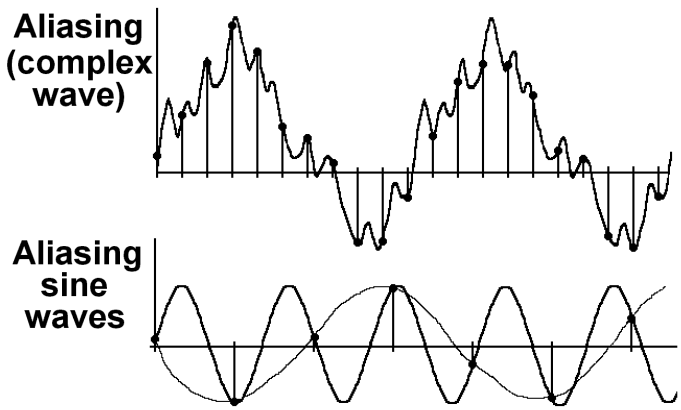


Figure 9.1: Because of inadequate sampling rate, aliasing causes important features to be lost.

The next step in Analog to Digital Conversion is to hold each waveform value steady for a period (using a Pulse) while a stable measurement can be made, then associating the analog value with a digital number (Coding). So PCM means to Modulate the analog signal with a Pulse, measure the value for that instant, then Code it into a digital number. Analog signals can have any of the infinity of real-numbered amplitude values. Since computers work with fixed word sizes (8-bit bytes, 16 bit words, etc.), digital signals can only have a finite number of amplitude values. In converting from analog to digital, rounding takes place and a given analog value must be quantized to the nearest digital value. The difference between quantization steps is called the quantum (not as in quantum physics or leaps, but that's just the Latin word for a fixed sized jump in value or magnitude). Sampling and quantization is shown in Figure 9.2. Note the errors introduced in some sample values due to the quantization process.

Humans can perceive frequencies from roughly 20 Hz to 20 kHz, thus requiring a minimum sampling rate of at least 40 kHz. Speech signals are often sampled at 8kHz ("telephone quality") or 11.025 kHz, while music is usually sampled at 44.1 kHz (the sampling rate used on audio Compact Disks), or 48 kHz. Some new formats allow for sampling rates of 96 kHz, and even 192 kHz.

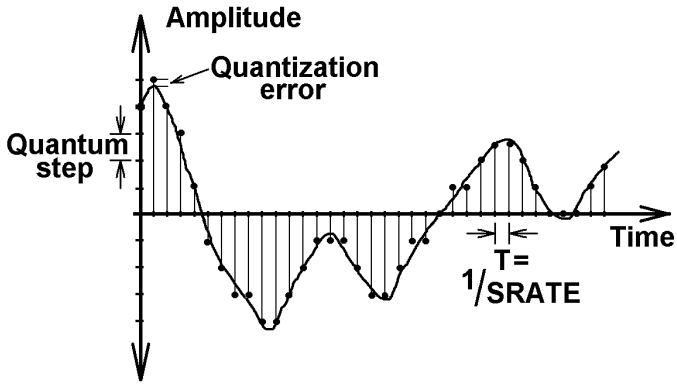


Figure 9.2: Linear sampling and quantization.

In a digital system, a fixed number of binary digits (bits) are used to sample the analog waveform, by quantizing it to the closest number that can be represented. This quantization is accomplished either by rounding to the quantum value nearest the actual analog value, or by truncation to the nearest quantum value less than or equal to the actual analog value. With uniform sampling in time, a properly band-limited signal can be exactly recovered provided that the sampling rate is twice the bandwidth or greater, but only if there is no quantization. When the signal values are rounded or truncated, the amplitude difference between the original signal and the quantized signal is lost forever. This can be viewed as an additive noise component upon reconstruction. Using the additive noise assumption gives an approximate best-case signal to quantization noise ratio (SNR) of approximately 6N dB, where N is the number of bits. Using this approximation implies that a 16 bit linear quantization system will exhibit an SNR of approximately 96 dB. 8 bit quantization exhibits a signal to quantization noise of approximately 48 dB. Each extra bit improves the signal to noise ratio by about 6 dB. Exact formulas for this are given in [1].

Most computer audio systems use two or three types of audio data words. As the data format used in Compact Disk systems, 16 bit (per channel) data is quite common. High definition formats allow for 24 bit samples. 8-bit data is common for speech data in PC and telephone systems, usually using methods of quantization that are non-linear. In non-linear quantization systems (mu-law or a-law) the quantum is smaller for small amplitudes, and larger for large amplitudes.

9.3.1 PCM (Wavetable, Sampling, Concatenative (Speech)) Synthesis

The majority of speech, music, and sound "synthesis" today is accomplished via the playback of stored PCM (Pulse Code Modulation) waveforms. Single-shot playback of entire segments of stored sounds is common for sound effects, narrations, prompts, segments of music, etc. Most high quality modern electronic music synthesizers, speech synthesis systems, and PC software systems for sound synthesis use pre-stored PCM as the basic data. This data is sometimes manipulated by filtering, pitch shifting, looping, and other means to yield the

final output sound(s).

For speech, the most common synthesis technique is "concatenative" synthesis [2]. Concatenative phoneme synthesis relies on end-to-end splicing of roughly 40 (for English) pre-stored phonemes. Examples of vowel phonemes are /i/ as in beet, /I/ as in bit, /a/ as in father, /u/ as in boot, etc. Examples of nasals are /m/ as in mom, /n/ as in none, /ng/ as in sing, etc. Examples of fricative consonant phonemes are /s/ as in sit, /sh/ as in ship, /f/ as in fifty. Examples of voiced fricative consonants are /v/, /z/ (visualize). Examples of plosive consonants are /t/ as in tat, /p/ as in pop, /k/ as in kick, etc. Examples of voiced plosives include /d/, /b/, /g/ (dude, bob, & gag). Vowels and nasals are pitched periodic sounds, so the minimal required stored waveform is only one single period of each. Consonants require more storage because of their noisy (non-pitched, aperiodic) nature. Sound and movie examples **S9.1** and **S9.2** demonstrate concatenative voice/speech synthesis.

The quality of concatenative phoneme synthesis is generally considered quite low, due to the simplistic assumption that all of the pitched sounds (vowels, etc.) are purely periodic. Also, simply "gluing" /s/ /I/ and /ng/ together does not make for a high quality realistic synthesis of the word "sing." In actual speech, phonemes gradually blend to each other as the jaw, tongue, and other "articulators" move with time.

Accurately capturing the transitions between phonemes with PCM requires recording transitions from phoneme to phoneme, called "diphones". A concatenative diphone synthesizer blends together stored diphones. Examples of diphones include see, she, thee, and a subset of the roughly 40x40 possible combinations of phonemes. Much more storage is necessary for a diphone synthesizer, but the resulting increase in quality is significant.

Changing the playback sample rate on sampled sound results in a shift in pitch, time, and spectral shape. Many systems for recording, playback, processing, and synthesis of music, speech, or other sounds allow or require flexible control of pitch (sample rate). The most accurate pitch control is necessary for music synthesis. In sampling synthesis, this is accomplished by dynamic sample rate conversion (interpolation), which has three steps; band-limiting, interpolation, and re-sampling. The band-limiting is the same as is required for sampling, so if the new sample rate is lower than the original, frequencies higher than half the new rate must be removed.

Interpolation is the process of filling in the smooth waveform between existing samples, and can be accomplished by fitting line segments to the samples (not the best method, due to artifacts from the jagged edges), higher order curves (splines), or other means. The provably correct way (from engineering mathematics) to interpolate is to fit a *sinc* function to the samples, defined as:

$$\mathrm{sinc}(t/T) = \frac{\sin(\pi t/T)}{\pi t/T}, \quad \text{where } T = 1/\mathrm{SRATE} \tag{1}$$

The sinc function is the ideal reconstruction filter, but comes at a significant computational cost, so the designer of a high quality sample rate converter will choose an appropriately truncated sinc function [3] to meet the quality constraints of the system. Once the smooth waveform is reconstructed, it can then be re-sampled at the new target sample rate.

Note that stored voiced speech phonemes (pitched vowels are periodic, so can be stored as a single period and synthesized by looping that single period), or the phoneme components

of diphones, can be shifted in pitch by sample rate conversion, allowing "prosody" to be imparted on synthesized speech. However, the pitch cannot be shifted too far in either direction, because the spectral properties shift accordingly, and the synthesized speaker can begin to sound like a chipmunk (shifted upward, therefore sounding like a smaller head size), or a giant (shifted downward, large head). PCM speech synthesis can be improved further by storing multi-samples, for different pitch ranges, genders, voice qualities, individual speaker voices, accents, etc.

9.3.2 Making PCM Parametric

In fact, storing the individual phonemes or diphones for speech synthesis is a form of hand-crafted parameterization. Noting that there are only 40 phonemes, and actually only a few hundred important diphones (any given language uses only a small subset of commonly occurring phoneme combinations), the index number of each phoneme/diphone can be considered a low-dimensional parameter. Combined with pitch shifting by interpolation, and the ability to loop phonemes for arbitrary lengths, a speech synthesizer becomes a form of parametric synthesizer. But from our definition, we desire *continuous* parameters that influence the sound as a reliable function of how much we perturb them.

For musical sounds, it is common to store only a loop, or wavetable, of the periodic component of a recorded sound waveform and play that loop back repeatedly. This is sometimes called "Wavetable Synthesis" [4], primarily in musical synthesis. In speech and other sound synthesis the more common term is "concatenative." For more realism, the attack or beginning portion of the recorded sound can be stored in addition to the periodic steady state part. Figure 9.3 shows the synthesis of a trumpet tone starting with an attack segment, followed by repetition of a periodic loop, ending with an enveloped decay (or release). "Envelope" is a synthesizer/computer music term for a time-varying change applied to a waveform amplitude, or other parameter. Envelopes are often described by four components; the *Attack Time*, the *Decay Time* ("decay" here means the initial decay down to the steady state segment), the *Sustain Level*, and the *Release Time* (final decay). Hence, envelopes are sometimes called *ADSR's*.

Originally called "Sampling Synthesis" in the music industry, any synthesis using stored PCM waveforms has now become commonly known as "Wavetable Synthesis". Filters are usually added to high-quality wavetable synthesis, allowing control of spectral brightness as a function of intensity, and to get more variety of sounds from a given set of samples. Thus making the model more parametric.

As discussed in the previous section, a given sample can be pitch-shifted only so far in either direction before it begins to sound unnatural. This can be dealt with by storing multiple recordings of the sound at different pitches, and switching or interpolating between these upon resynthesis. In music sampling and speech synthesizers, this is called "multi-sampling". Multi-sampling also might include the storage of separate samples for "loud" and "soft" sounds. Linear or other interpolation is used to blend the loudness of multi-samples as a function of the desired synthesized volume. This adds realism, for loudness is not simply a matter of amplitude or power; most sound sources exhibit spectral variations as a function of loudness due to driving energy and non-linearity. There is usually more high frequency energy ("brightness") in loud sounds than in soft sounds. Filters can also be used to add

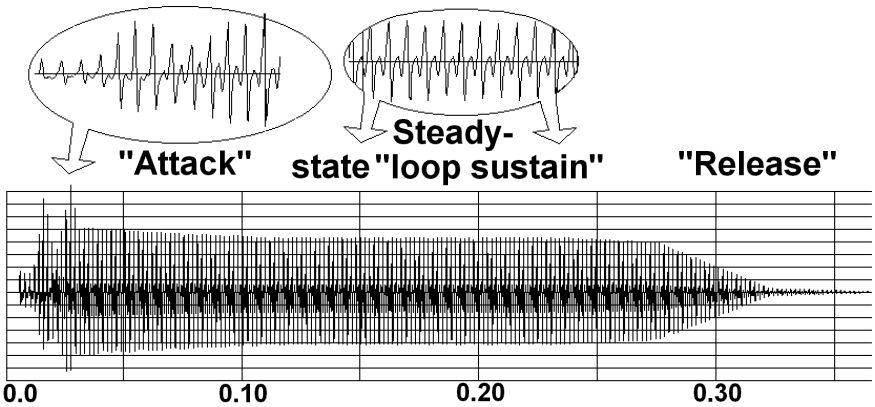


Figure 9.3: Wavetable synthesis of trumpet tone.

spectral variation.

A common tool used to describe the various components and steps of signal processing in performing digital music synthesis is the "synthesizer patch" (historically named from hooking various electrical components together using patch cords). In a patch, a set of fairly commonly agreed building blocks, called "unit generators" (also called modules, plug-ins, operators, op-codes, and other terms) are hooked together in a signal flow diagram. This historical [5] graphical method of describing signal generation and processing affords a visual representation that is easily printed in papers, textbooks, patents, etc. Further, graphical patching systems and languages have been important to the development and popularization of certain algorithms, and computer music in general. Figure 9.4 shows a PCM synthesizer patch with attack and loop wavetables whose amplitudes are controlled by an envelope generator, and a time-varying filter (also controlled by another envelope generator). As with all synthesis, panning (placement in stereo or more channels) can be controlled as an additional parameter.

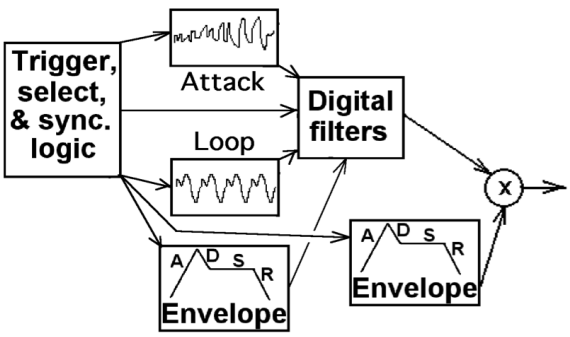


Figure 9.4: Block diagram of a wavetable synthesizer "patch", showing connections of unit generators such as wavetables, ADSR envelopes, digital filters, etc.

9.4 Fourier (Sinusoidal) "Synthesis"

Lots of sound-producing objects and systems exhibit sinusoidal modes, which are the natural oscillatory frequencies of any acoustical system. A plucked string might exhibit many modes, with the strength of each mode determined by the conditions of the terminations, and the nature of the excitation pluck (plucking at the end vs. the center). Striking a metal plate with a hammer excites many of the vibrational modes of the plate, determined by the shape of the plate, and by where it is struck. A singing voice, struck drum head, bowed violin string, struck bell, or blown trumpet exhibit oscillations characterized by a sum of sinusoids. The recognition of the fundamental nature of the sinusoid gives rise to a powerful model of sound synthesis based on summing up sinusoidal modes.

These modes have a very special relationship in the case of the plucked string, a singing voice, and some other limited systems, in that their frequencies are all integer multiples (at least approximately) of one basic sinusoid, called the "fundamental." This special series of sinusoids is called a "harmonic series", and lies at the basis of the "Fourier Series" representation of oscillations, waveforms, shapes, etc. The Fourier Series [6] solves many types of problems, including physical problems with boundary constraints, but is also applicable to any shape or function. Any periodic waveform (repeating over and over again) F_{per} can be transformed into a Fourier series, written as:

$$F_{\text{per}}(t) = a_0 + \sum_m \left[b_m \cos(2\pi f_0 m t) + c_m \sin(2\pi f_0 m t)\right] \tag{2}$$

Which states mathematically that any periodic function can be expressed as a sum of harmonically (integer multiples of some fundamental frequency) related sine and cosine functions, plus an arbitrary constant.

The limits of the summation are technically infinite, but we know that we can (and should) cut off frequencies at ½ the sampling frequency for digital signals. The a_0 term is a constant offset, or the average of the waveform. The b_m and c_m coefficients are the weights of the "m^{th} harmonic" cosine and sine terms. If the function $F_{\text{per}}(t)$ is purely "even" about $t = 0$ $(F(-t) = F(t))$, only cosines are required to represent it, and all of the c_m terms would be zero. Similarly, if the function $F_{\text{per}}(t)$ is "odd" $(F(-t) = -F(t))$, only the c_m terms would be required. An arbitrary function $F_{\text{per}}(t)$ will require sinusoidal harmonics of arbitrary (but specific) amplitudes and phases. The magnitude A and phase θ of the mth harmonic in the Fourier Series can be found by:

$$A_m = \sqrt{b_m^2 + c_m^2} \tag{3}$$

$$\theta_m = \text{ArcTan}(c_m/b_m) \tag{4}$$

Phase is defined relative to the cosine, so if c_m is zero, θ_m is zero. As a brief example, Figure 9.5 shows the first few sinusoidal harmonics required to build up an approximation of a square wave. Note that due to symmetries only odd sine harmonics $(1, 3, 5, 7)$ are required. The amplitudes of the sine waves are expressed as $1/M$, where M is the harmonic number. Using more sines improves the approximation of the resulting synthesis, moving toward a pure square wave.

The process of determining the sine and cosine components of a signal or waveform is called "Fourier Analysis", or the "Fourier Transform". If the frequency variable is sampled (as is

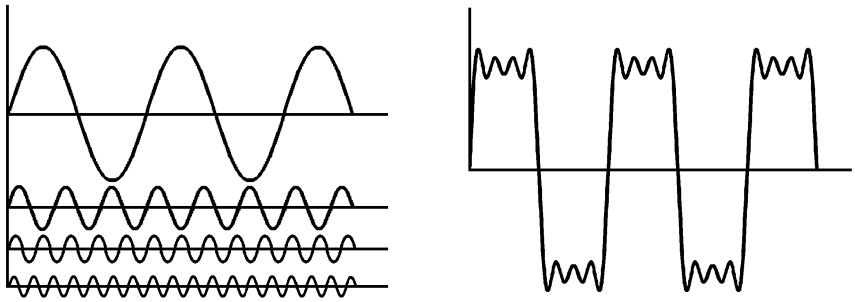


Figure 9.5: A sum of odd harmonics approximates a square wave.

the case in the Fourier Series, represented by m), and the time variable t is sampled as well (as it is in PCM waveform data, represented by n), then the Fourier Transform is called the "Discrete Fourier Transform", or DFT. The DFT is given by:

$$F(m) = \sum_{n=0}^{N-1} f(n) \left[\cos(2\pi mn/N) - j\sin(2\pi mn/N)\right] \tag{5}$$

Where N is the length (in samples) of the signal being analyzed. The inverse DFT (IDFT) is similar to the Fourier Series:

$$f(n) = \frac{1}{N} \sum_{m=0}^{N-1} F(m) \left[\cos(2\pi mn/N) + j\sin(2\pi mn/N)\right] \tag{6}$$

The convention is to use lower case for the time domain and upper case for the frequency domain. So $f(n)$ is the time-waveform (a sound for example), and $F(m)$ represents the spectral description.

The imaginary number $j = \sqrt{-1}$ is used to place the cosine and sine components in a unique mathematical arrangement, where odd ($x(-n) = -x(n)$) sine terms of the waveform are represented as imaginary components, and even ($x(-n) = x(n)$) cosine terms are represented as real components. This gives us a way of talking about the magnitude and phase in terms of the magnitude and phase of $F(m)$ (a complex number). There is a near-mystical expression of equality in mathematics known as Euler's Identity, which links trigonometry, exponential functions, and complex numbers in a single equation:

$$e^{j\theta} = \cos(\theta) + j\sin(\theta)\ . \tag{7}$$

We can use Euler's identity to write the DFT and IDFT in shorthand:

$$F(m) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi mn/N} \tag{8}$$

$$f(n) = \frac{1}{N} \sum_{m=0}^{N-1} F(m) e^{j2\pi mn/N} \tag{9}$$

Converting the cosine/sine form to the complex exponential form allows lots of manipulations that would be difficult otherwise. But we can also write the DFT in real number terms as a form of the Fourier Series:

$$f(n) = \frac{1}{N} \sum_{m=0}^{N-1} F_b(n) \cos(2\pi mn/N) + F_c(n) \sin(2\pi mn/N) \tag{10}$$

where

$$F_b(m) = \sum_{n=0}^{N-1} f(n) \cos(2\pi mn/N) \tag{11}$$

$$F_c(m) = \sum_{n=0}^{N-1} -f(n) \sin(2\pi mn/N) \tag{12}$$

The Fast Fourier Transform (FFT) is a computationally efficient way of calculating the DFT. There are thousands of references on the FFT [6], and scores of implementations of it, so for our purposes we'll just say that it's lots more efficient than trying to compute the DFT directly from the definition. A well crafted FFT algorithm for real input data takes on the order of $N \log_2(N)$ multiply-adds to compute. Comparing this to the N^2 multiplies of the DFT, N doesn't have to be very big before the FFT is a winner. There are some downsides, such as the fact that FFTs can only be computed for signals whose lengths are exactly powers of 2, but the advantages of using it often outweigh the pesky power-of-two problems. Practically speaking, users of the FFT usually carve up signals into small chunks (powers of two), or "zero pad" a signal out to the next biggest power of two.

The Short-Time Fourier Transform (STFT) breaks up the signal into (usually overlapping) segments and applies the Fourier Transform to each segment individually [7]. By selecting the window size (length of the segments), and hop size (how far the window is advanced along the signal each step) to be perceptually relevant, the STFT can be thought of as a simple approximation of human audio perception. Figure 9.6 shows the waveform of the utterance of the word "synthesize", and some STFT spectra corresponding to windows at particular points in time.

9.4.1 Direct Fourier "Synthesis"

Fourier synthesis is essentially just the process of reconstructing the time domain waveform from the sines and cosines indicated by the Fourier Transform. In other words, it is the Inverse Fourier Transform. As such it is essentially the same as PCM synthesis, providing no meaningful parameters for transformations. There are ways to parameterize, however.

Using the Short Time Fourier Transform, the "Phase Vocoder" (VoiceCoder) [8, 9] processes sound by calculating and maintaining both magnitude and phase. The frequency bins (basis sinusoids) of the DFT can be viewed as narrowband filters, so the Fourier Transform of an input signal can be viewed as passing it through a bank of narrow band-pass filters. This means that on the order of hundreds to thousands of sub-bands are used.

The Phase Vocoder has found extensive use in computer music composition and sound design. Many interesting practical and artistic transformations can be accomplished using

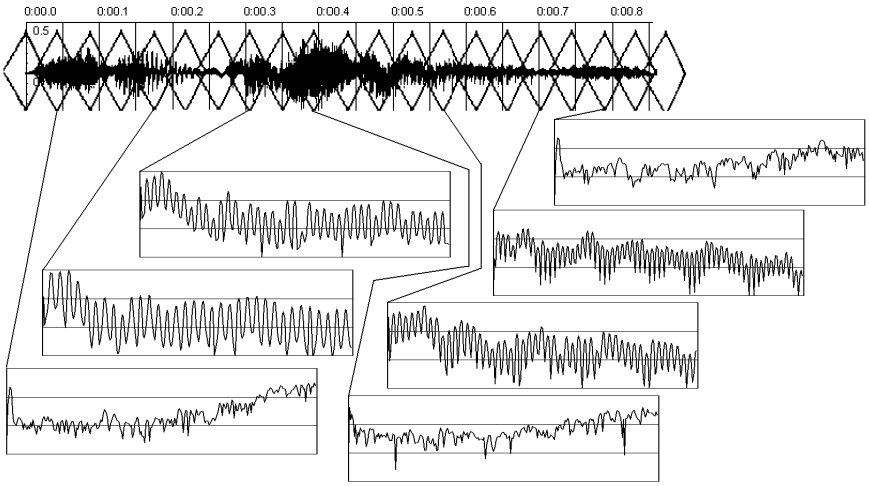


Figure 9.6: Some STFT frames of the word "synthesize".

the Phase Vocoder, including nearly artifact-free and independent time and pitch shifting, as demonstrated in sound example **S9.3**. A technique called "cross synthesis" assumes that one signal is the "analysis" signal. The time-varying magnitude spectrum of the analysis signal (usually smoothed in the frequency domain) is multiplied by the spectral frames of another "input" (or filtered) signal (often "brightened" by high frequency pre-emphasis), yielding a composite signal that has the attributes of both. Cross-synthesis has produced the sounds of talking cows, "morphs" between people and cats, trumpet/flute hybrids, etc.

These techniques are useful for analyzing and modifying sounds in some ways, but for auditory display, we can do more to make Fourier-related methods more parametric.

9.4.2 Making Fourier More Parametric

While the Fourier Transform is not parametric as directly implemented, we can use the Fourier Transform to extract useful parametric information about sounds. A brief list of audio "features" (also called descriptors) that can be extracted is:

- Gross Power in each window. If the audio stream suddenly gets louder or softer, then there is a high likelihood that something different is occurring. In speech recognition and some other tasks, however, we would like the classification to be loudness invariant (over some threshold used to determine if anyone is speaking).
- Spectral Centroid, which relates closely to the brightness of the sound, or the relative amounts of high and low frequency energy.
- Rolloff: Spectra almost always decrease in energy with increasing frequency. Rolloff is a measure of how rapidly, and is another important feature that captures more information about the brightness of an audio signal.
- Spectral Flux is the amount of frame-to-frame variance in the spectral shape. A steady

sound or texture will exhibit little spectral flux, while a modulating sound will exhibit more flux.

- Mel-Frequency Cepstral Coefficients, which are a compact (between 4 and 10 numbers) representation of spectral shape. LPC coefficients are sometimes used in this way as well.
- Low Energy is a feature, defined as the percentage of small analysis windows that contain less power than the average over the larger window that includes the smaller windows. This is a coarser time-scale version of flux, but computed only for energy.
- Zero-Crossing Rate is a simple measure of high frequency energy.
- Harmonicity is a measure of the "pitchyness" (and pitch) of a sound.
- Harmonics to Noise Ratio is a measure of the "breathiness" of a sound.
- Parametric Pitch Histogram is a multi-pitch estimate.
- Beat/Periodicity Histogram is a measure of beat (rhythm) strength and timing.

All of these can be extracted and used to understand, classify, and describe sounds, as is done in audio analysis, music information retrieval, content-based query, etc. [10]. However they are not sufficient for direct synthesis.

If we inspect the various spectra in Figure 9.6, we can note that the vowels exhibit harmonic spectra (clear, evenly spaced peaks corresponding to the harmonics of the pitched voice), while the consonants exhibit noisy spectra (no clear sinusoidal peaks). Recognizing that some sounds are well approximated/modeled by additive sine waves [11], while other sounds are essentially noisy, "spectral modeling" [12] breaks the sound into deterministic (sines) and stochastic (noise) components. Figure 9.7 shows a general Sines+Noise Additive Synthesis model, allowing us to control the amplitudes and frequencies of a number of sinusoidal oscillators, and model the noisy component with a noise source and a spectral shaping filter.

The beauty of this type of model is that it recognizes the dominant sinusoidal nature of many sounds, while still recognizing the noisy components that might be also present. More efficient and parametric representations, and many interesting modifications, can be made to the signal on resynthesis. For example, removing the harmonics from voiced speech, followed by resynthesizing with a scaled version of the noise residual, can result in the synthesis of whispered speech.

One further improvement to spectral modeling is the recognition [13] that there are often brief (impulsive) moments in sounds that are really too short in time to be adequately analyzed by spectrum analysis. Further, such moments in the signal usually corrupt the sinusoidal/noise analysis process. Such events, called transients, can be modeled other ways (often by simply keeping the stored PCM for that segment). As with Fourier synthesis, Spectral Modeling is most useful for transformation and modification of existing sounds. Indeed, some meaningful parameters on noise, spectral shape, and transient extraction could be exploited during resynthesis for auditory display. An excellent reference to Fourier and frequency domain techniques, and signal processing in general, can be found in [14].

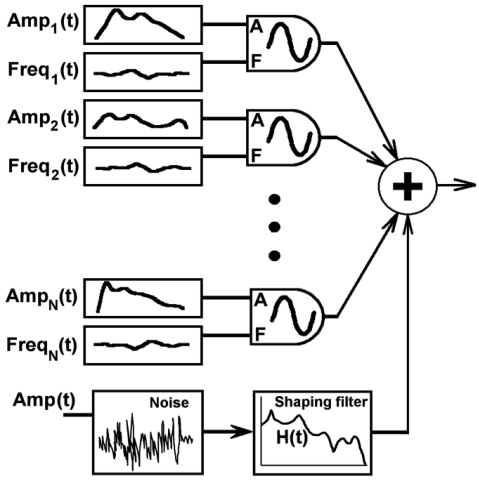


Figure 9.7: Sinusoidal model with filtered noise added for spectral modeling synthesis.

9.5 Modal (Damped Sinusoidal) Synthesis

The simplest physical system that does something acoustically (and musically) interesting is the mass-spring-damper [15]. The differential equation describing that system has a solution that is a single exponentially-decaying cosine wave. Another common system that behaves the same way is a pendulum under small displacements. The swinging back and forth of the pendulum follows the same exponentially-decaying cosine function. Yet one more system, the Helmholtz resonator (a large cavity, containing air, with a small long-necked opening, like a pop bottle), behaves like a mass-spring-damper system, with the same exponentially damped cosine behavior.

The equations describing the behavior of all of these systems, where $m =$ mass, $r =$ damping, and $k =$ spring constant (restoring force) is:

$$\frac{d^2y}{dt^2} + \frac{r}{m}\frac{dy}{dt} + \frac{k}{m}y = 0 \tag{13}$$

$$y(t) = y_0 e^{(-rt/2m)} \cos\left(t\sqrt{\frac{k}{m} - \left(\frac{r}{2m}\right)^2}\right) \tag{14}$$

Where y is the displacement of the mass, dy/dt is the velocity of the mass, and d^2y/dt^2 is the acceleration of the mass. Of course, most systems that produce sound are more complex than the ideal mass-spring-damper system, or a pop bottle. And of course most sounds are more complex than a simple damped exponential sinusoid. Mathematical expressions of the physical forces (and thus the accelerations) can be written for nearly any system, but solving such equations is often difficult or impossible. Some systems have simple enough properties and geometries to allow an exact solution to be written out for their vibrational behavior. An ideal string under tension is one such system.

This section will present some graphical arguments and refers to the previous discussion of the Fourier Transform to further motivate the notion of sinusoids in real physical systems. The top of Figure 9.8 shows a string, lifted from a point in the center (half-way along its length). Below that is shown a set of sinusoidal "modes" that the center-plucked string vibration would have. These are spatial functions (sine as function of position along the string), but they also correspond to the natural frequencies of vibration of the string. At the bottom of Figure 9.8 is another set of modes that would not be possible with the center-plucked condition, because all of these "even" modes are restricted to have no vibration in the center of the string, and thus they could not contribute to the triangular shape of the 'initial central pluck condition' of the string. These conditions of no displacement, corresponding to the zero crossings of the sine functions, are called "nodes." Note that the end points are forced nodes of the plucked string system for all possible conditions of excitation.

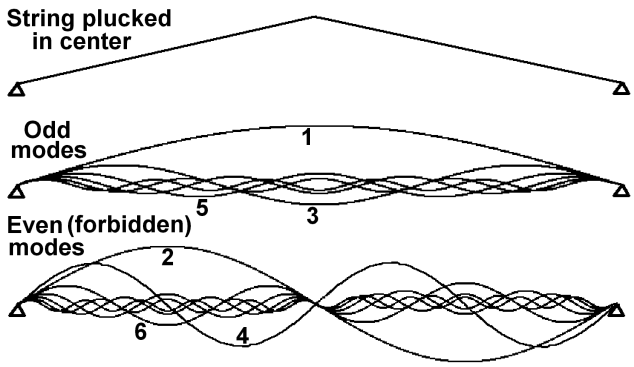


Figure 9.8: Plucked string (top). Center shows sinusoidal "modes" of vibration of a center-plucked string. Bottom shows the even modes, which would not be excited by the center-plucked condition. End points are called "nodes" of no-vibration.

Physical constraints on a system, such as the pinned ends of a string, and the center plucked initial shape, are known as "boundary conditions." Spatial sinusoidal solutions like those shown in Figure 9.8 are called "boundary solutions" (the legal sinusoidal modes of displacement and vibration) [16].

Just as Fourier Boundary methods (Fourier solutions taking into account physical limits and symmetries) can be used to solve the one-dimensional string, we can also extend boundary methods to two dimensions. Figure 9.9 shows the first few vibrational modes of a uniform square membrane. The little boxes at the lower left corners of each square modal shape depict the modes in a purely 2-dimensional way, showing lines corresponding to the spatial sinusoidal nodes (regions of no displacement vibration). Circular drum heads are more complex, but still exhibit a series of circular and radial modes of vibration. The square membrane modes are not integer-related inharmonic frequencies. In fact they obey the relationship:

$$f_{mn} = f_{11}\sqrt{(m^2 + n^2)/2} \tag{15}$$

where m and n range from 1 to (potentially) infinity, and f_{11} is $c/2L$ (speed of sound on the membrane divided by the square edge lengths).

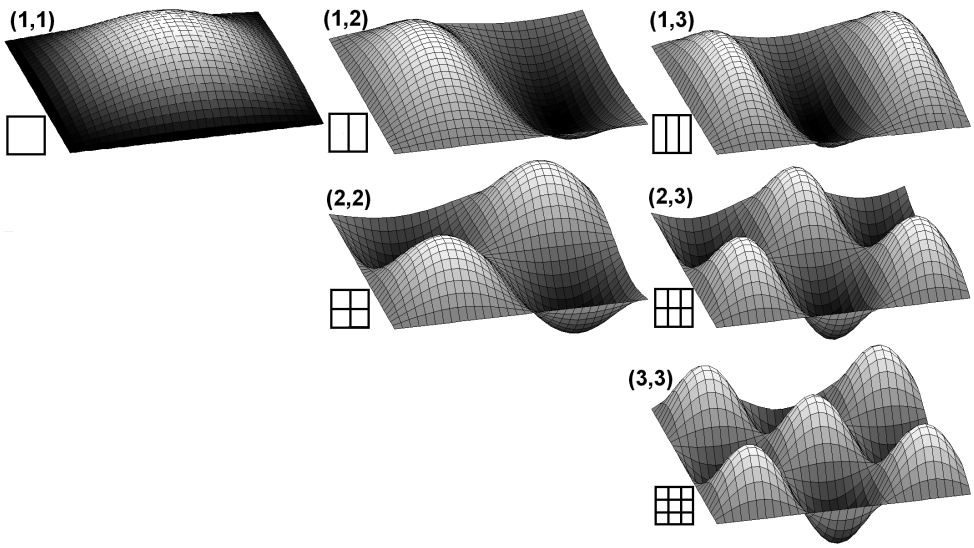


Figure 9.9: Square membrane vibration modes.

Unfortunately, circles, rectangles, and other simple geometries turn out to be the only ones for which the boundary conditions yield a closed-form solution in terms of spatial and temporal sinusoidal components. However, we can measure and model the modes of any system by computing the Fourier Transform of the sound it produces, and by looking for exponentially decaying sinusoidal components.

We can approximate the differential equation describing the mass-spring-damper system of Equation 13 by replacing the derivatives (velocity as the derivative of position, and acceleration as the 2nd derivative of position) with sampled time differences (normalized by the sampling interval T seconds). In doing so we arrive at an equation that is a recursion in past values of $y(n)$, the position variable:

$$\frac{y(n) - 2y(n-1) + y(n-2)}{T^2} + \frac{r}{m}\frac{y(n) - y(n-1)}{T} + \frac{k}{m}y(n) = 0 \qquad (16)$$

where $y(n)$ is the current value, $y(n-1)$ is the value one sample ago, and $y(n-2)$ is the twice-delayed sample. Note that if the values of mass, damping, spring constant, and sampling rate are constant, then the coefficients $((2m + Tr)/(m + Tr + T^2k)$ for the single delay, and $m/(m + Tr + T^2k)$ for the twice delayed signal) applied to past y values are constant. DSP engineers would note that a standard Infinite Impulse Response (IIR) recursive filter as shown in Figure 9.10 can be used to implement Equation 16 (the Z^{-1} represents a single sample of delay). In fact, equation 16 (called the 2nd order 2-pole feedback filter by Digital Signal Processing engineers) can be used to generate an exponentially decaying sinusoid, called a "phasor" in DSP literature [17]. Here the term "filter" is used to mean anything that takes a signal as input, yields a signal as output, and does something interesting between (not strictly a requirement that it do something interesting, but why bother if not?). The connection between the 2nd order digital filter and the physical notion of a mode of vibration forms the basis for Modal Sound Synthesis [18], where a spectrally rich source

such as an impulse or noise is used to excited modal filters to generate a variety of natural sounds.

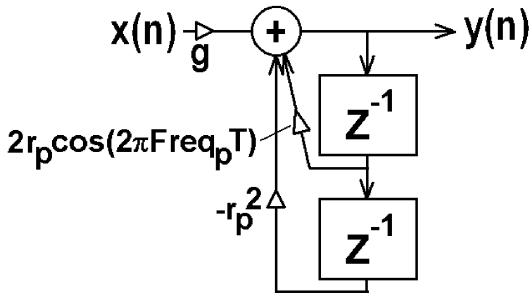


Figure 9.10: Two-Pole resonant filter

9.5.1 Making Modes Parametric

The extraction of modes, and synthesis using a resonant filter, can do a lot toward parameterizing many types of sounds. Stiff metal and glass objects, and some other systems tend to exhibit relatively few sinusoidal modes. In some cases, the location of excitation (striking or plucking) can be related to the excitation level of each mode (as was the case above with our center-plucked string). "Damping" in the system relates to the speed of decay of the exponentials describing each mode. High damping means rapid decay (as when we mute a guitar string). Thus, strike amplitude, strike location, and modal damping can become powerful parameters for controlling a modal synthesis model. The frequencies of the modes can be changed together, in groups, or separately, to yield different sonic results.

Figure 9.11 shows a general model for modal synthesis of struck/plucked objects, in which an impulsive excitation function is used to excite a number of filters that model the modes. Rules for controlling the modes as a function of strike position, striking object, changes in damping, and other physical constraints are included in the model. The flexibility of this simple model is demonstrated in sound example **S9.4**.

Modal synthesis is a powerful technique for auditory display, because we can control timbre, pitch, and time with a few "knobs". The nature of modal synthesis, where each sound begins with an impulsive excitation, and decays exponentially thereafter, lends it to alerts and alarms, and systems where rhythmic organization is an important part of the design of the auditory display.

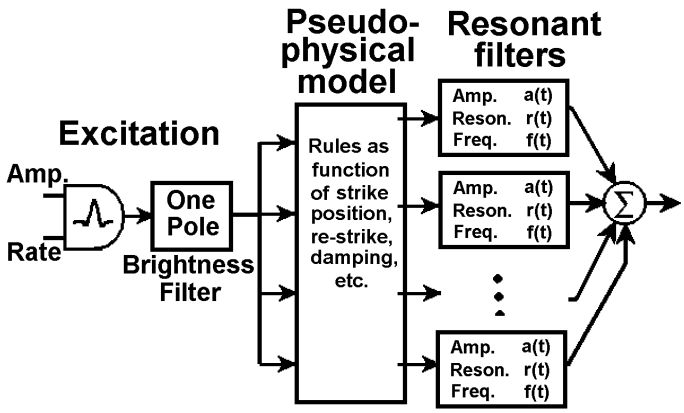


Figure 9.11: Flexible parametric modal synthesis algorithm.

9.6 Subtractive (Source-Filter) synthesis

Subtractive synthesis uses a complex source wave, such as an impulse, a periodic train of impulses, or white noise, to excite spectral-shaping filters. One of the earliest uses of electronic subtractive synthesis dates back to the 1920/30s, with the invention of the "Channel Vocoder" (for VOiceCODER) [19]. In this device, the spectrum is broken into sections called sub-bands, and the information in each sub-band is converted to a signal representing (generally slowly varying) power. The analyzed parameters are then stored or transmitted (potentially compressed) for reconstruction at another time or physical site. The parametric data representing the information in each sub-band can be manipulated in various ways, yielding transformations such as pitch or time shifting, spectral shaping, cross synthesis, and other effects. Figure 9.12 shows a block diagram of a channel vocoder. The detected envelopes serve as "control signals" for a bank of band-pass "synthesis filters" (identical to the "analysis filters" used to extract the sub-band envelopes). The synthesis filters have gain inputs that are fed by the analysis control signals.

When used to encode and process speech, the channel vocoder explicitly makes an assumption that the signal being modeled is a single human voice. The "source analysis" (upper left of Figure 10.12) block extracts parameters related to finer spectral details, such as whether the sound is pitched (vowel) or noisy (consonant or whispered). If the sound is pitched, the pitch is estimated. The overall energy in the signal is also estimated. These parameters become additional low-bandwidth control signals for the synthesizer. Intelligible speech can be synthesized using only a few hundred numbers per second. An example coding scheme might use 8 channel gains + pitch + power, per frame, at 40 frames per second, yielding a total of only 400 numbers per second. The channel vocoder, as designed for speech coding, does not generalize to arbitrary sounds, and fails horribly when the source parameters deviate from expected harmonicity, reasonable pitch range, etc. This can result in artifacts ranging from distortion, to rapid shifts in pitch and spectral peaks (often called "bubbling bells"). But the ideas of sub-band decomposition, envelope detection, and driving a synthesis filter bank with control signals give rise to many other interesting applications and implementations of

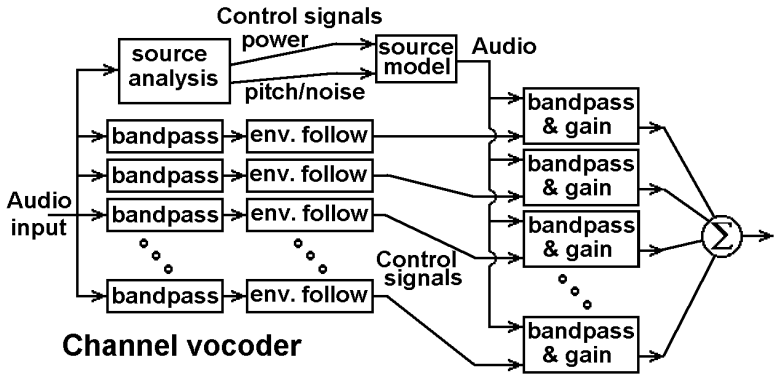


Figure 9.12: Channel vocoder block diagram.

the vocoder concepts. MPEG coding/compression, audio/speech analysis, and audio effects all use these ideas.

A family of filter-based frequency transforms known as "Wavelet Transforms" has been used for analysis and synthesis of sound. Instead of being based on steady sinusoids such as the Fourier Transform, Wavelet Transforms are based on the decomposition of signals into fairly arbitrary functions (called "wavelets") [20] with useful properties such as compactness (constrained in time or frequency), efficient computation, or other.

Some benefits of wavelet transforms over Fourier transforms are that they can be implemented using fairly arbitrary filter criteria, on a logarithmic frequency scale rather than a linear scale as in the DFT, and that time resolution can be a function of the frequency range of interest. This latter point means that we can say accurate things about high frequencies as well as low. This contrasts with the Fourier transform, which requires the analysis window width be the same for all frequencies, meaning that we must either average out lots of the interesting high-frequency time information in favor of being able to resolve low frequency information (large window), or opt for good time resolution (small window) at the expense of low-frequency resolution, or perform multiple transforms with different sized windows to catch both time and frequency details. There are a number of fast wavelet transform techniques that allow the sub-band decomposition to be accomplished in essentially $N \log_2(N)$ time, like the FFT.

While the channel vocoder, and other sub-band models, are interesting and useful for processing and compressing speech and other sounds, by itself the vocoder isn't strictly a synthesizer. Factoring out the voice source parameters and filter energies does reduce the sound to a few descriptive numbers, and these numbers can be modified to change the sound. But very few systems actually use a channel vocoder-like structure to perform synthesis. Spectral shaping of noise or arbitrary signals can be used for auditory display, and thus the channel vocoder ideas could be useful.

9.6.1 Linear Predictive Synthesis (Coding)

Modal Synthesis, as discussed before, is a form of Subtractive Synthesis, but the spectral characteristics of modes are sinusoidal, exhibiting very narrow spectral peaks. For modeling the gross peaks in a spectrum, which could correspond to weaker resonances, we can exploit the same two-pole resonance filters. This type of source-filter synthesis has been very popular for voice synthesis.

Having origins and applications in many different disciplines, Time Series Prediction is the task of estimating future sample values from prior samples. Linear Prediction is the task of estimating a future sample (usually the next in the time series) by forming a linear combination of some number of prior samples. Linear Predictive Coding (LPC) does this, and automatically extracts the gross spectral features by designing filters to match those, yielding a "source" that we can use to drive the filters [21, 22]. Figure 9.13 shows linear prediction in block diagram form (where each Z^{-1} box represents a sample of delay/memory). The difference equation for a linear predictor is:

$$y(n) = \hat{x}(n+1) = \sum_{i=0}^{m} a_i x(n-i) \tag{17}$$

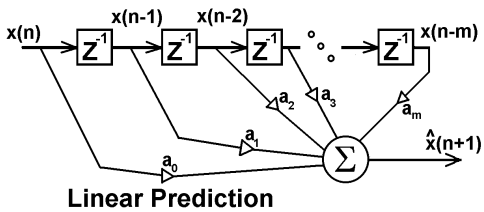


Figure 9.13: A linear prediction filter.

The task of linear prediction is to select the vector of predictor coefficients

$$A = [a_0, a_1, a_2, a_3, \ldots, a_m] \tag{18}$$

such that $\hat{x}(n+1)$ (the estimate) is as close as possible to $x(n+1)$ (the real sample) over a set of samples (often called a frame) $x(0)$ to $x(N-1)$. Usually "close as possible" is defined by minimizing the Mean Square Error (MSE):

$$\text{MSE} = \frac{1}{N} \sum_{n=1}^{N} [\hat{x}(n) - x(n)]^2 \tag{19}$$

Many methods exist for arriving at the predictor coefficients a_i which yield a minimum MSE. The most common method uses correlation or covariance data from each frame of samples to be predicted. The difference between the predicted and actual samples is called the "error" signal or "residual". The optimal coefficients form a digital filter. For low order LPC (delay order of 6–20 or so), the filter fits the coarse spectral features, and the residue contains the remaining part of the sound that cannot be linearly predicted. A common and popular use of

LPC is for speech analysis, synthesis, and compression. The reason for this is that the voice can be viewed as a "source-filter" model, where a spectrally rich input (pulses from the vocal folds or noise from turbulence) excites a filter (the resonances of the vocal tract). LPC is another form of spectral vocoder as discussed previously, but since LPC filters are not fixed in frequency or shape, fewer bands (than some vocoders) are needed to dynamically model the changing speech spectral shape.

LPC speech analysis/coding involves processing the signal in blocks and computing a set of filter coefficients for each block. Based on the slowly varying nature of speech sounds (the speech articulators can only move so fast), the coefficients are relatively stable for milliseconds at a time (typically 5-20ms is used in speech coders). If we store the coefficients and information about the residual signal for each block, we will have captured many of the essential aspects of the signal. Figure 9.14 shows an LPC fit to a speech spectrum. Note that the fit is better at the peak locations than in the valleys. This is due to the nature of the coefficient-computation mathematics, which performs a "least-squares error minimization criterion." Missing the mark on low-amplitude parts of the spectrum is not as important as missing it on high-amplitude parts. This is fortunate for audio signal modeling, in that the human auditory system is more sensitive to spectral peaks (poles, resonances), called "formants" in speech, than valleys (zeroes, anti-resonances).

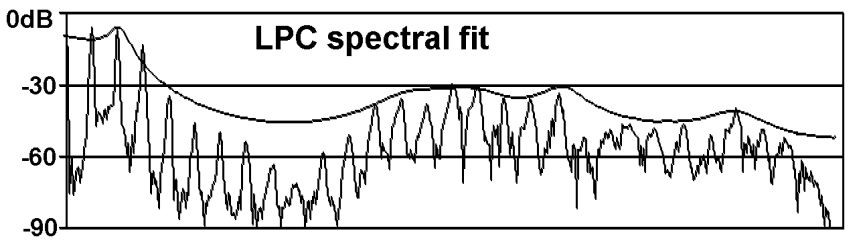


Figure 9.14: 10th order LPC filter fit to a voiced /u/ ("ooo") spectrum.

Once LPC has been performed on speech, inspecting the residual shows that it is often a stream of pulses for voiced speech, or white noise for unvoiced speech. Thus, if we store parameters about the residual, such as whether it is periodic pulses or noise, the frequency of the pulses, and the energy in the residual, then we can recreate a signal that is very close to the original. This is the basis of much modern speech compression. If a signal is entirely predictable using a linear combination of prior samples, and if the predictor filter is doing its job perfectly, we should be able to hook the output back to the input and let the filter predict the rest of the signal automatically. This form of filter, with feedback from output to input, is called "recursive." The recursive LPC reconstruction is sometimes called "all pole", referring to the high-gain "poles" corresponding to the primary resonances of the vocal tract. The poles do not capture all of the acoustic effects going on in speech, however, such as "zeroes" that are introduced in nasalization, aerodynamic effects, etc. However, as mentioned before, since our auditory systems are most sensitive to peaks (poles), LPC does a good job of capturing the most important aspects of speech spectra.

Any deviation of the predicted signal from the actual original signal will show up in the error signal, so if we excite the recursive LPC reconstruction filter with the residual signal itself,

we can get back the original signal exactly. This is a form of what engineers call "identity analysis/resynthesis", performing "deconvolution" or "source-filter separation" to separate the source from the filter, and using the residue to excite the filter to arrive at the original signal.

9.6.2 The Parametric Nature of LPC

Using the parametric source model also allows for flexible time and pitch shifting, without modifying the basic timbre. The voiced pulse period can be modified, or the frame rate update of the filter coefficients can be modified, independently. So it is easy to speed up a speech sound while making the pitch lower, still retaining the basic spectral shapes of all vowels and consonants. Cross-synthesis can also be accomplished by replacing the excitation wave with an arbitrary sound, as shown in sound example **S9.5**.

In decomposing signals into a source and a filter, LPC can be a marvelous aid in analyzing and understanding some sound-producing systems. The recursive LPC reconstruction filter can be implemented in a variety of ways. Three different filter forms are commonly used to perform subtractive voice synthesis [23]. The filter can be implemented in series (cascade) as shown in Figure 9.15, factoring each resonance into a separate filter block with control over center frequency, width, and amplitude. The flexibility of the parallel formant model is demonstrated in sound and movie examples **S9.6** and **S9.7**. The filter can also be implemented in parallel

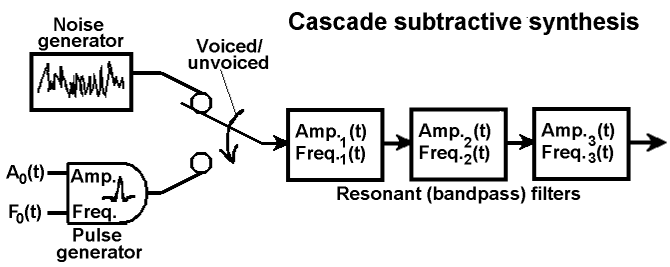


Figure 9.15: Cascade factored formant subtractive synthesizer.

(separate sub-band sections of the spectrum added together), as shown in Figure 9.16.

One additional implementation of the resonant filter is the ladder filter structure, which carries with it a notion of one-dimensional spatial propagation as well [24]. Figure 9.17 shows a ladder filter realization of an 8th order (output plus eight delayed versions of the output) IIR filter (Infinite Impulse Response, or feedback filter).

9.6.3 A Note on Parametric Analysis/Synthesis vs. Direct Synthesis

Note that most of our synthesis methods so far have relied (at least initially or in motivation) on analyzing or processing recorded sounds:

- PCM takes in a time-domain waveform and manipulates it directly;

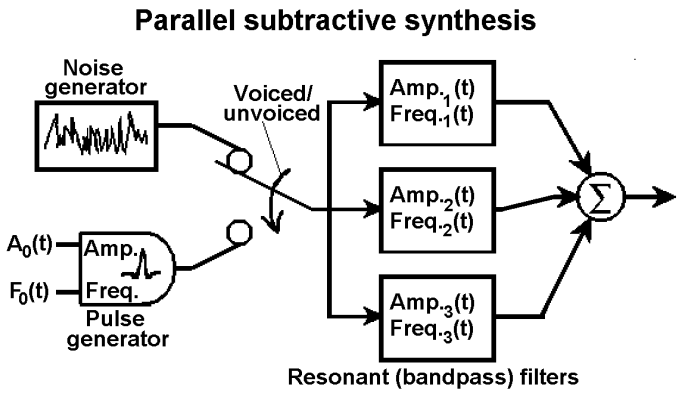


Figure 9.16: Parallel factored formant subtractive synthesizer.

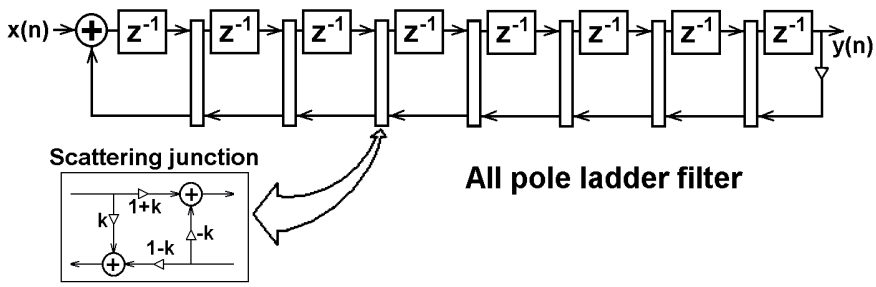


Figure 9.17: Ladder filter implementation of all-pole LPC filter.

- Fourier determines the sinusoidal components of a time-domain waveform;
- LPC determines the gross shape of a spectral filter and the source that, when driven through the filter, will yield an approximation of the original waveform.

As each technique was examined, ways were determined to extract or derive low(er)-order parameters for resynthesis that could be useful for auditory display. Based on this background knowledge and these techniques, the next sections look at methods for synthesizing directly from parameters, not necessarily relying on an original recording to be analyzed and manipulated.

9.7 Time Domain Formant Synthesis

FOFs (fonctions d'onde formantique, Formant Wave Functions) were created for voice synthesis using exponentially decaying sine waves, overlapped and added at the repetition period of the voice source [25]. Figure 9.18 depicts FOF synthesis of a vowel. FOFs are composed of a sinusoid at the formant center frequency, with an amplitude that rises

rapidly upon excitation, then decays exponentially. The control parameters define the center frequency and bandwidth of the formant being modeled, and the rate at which the FOFs are generated and added determines the fundamental frequency of the voice.

Note that each individual FOF is a simple "wavelet" (local and compact wave both in frequency and time). FOFs provide essentially the same parameters as formant filters, but are implemented in the time domain.

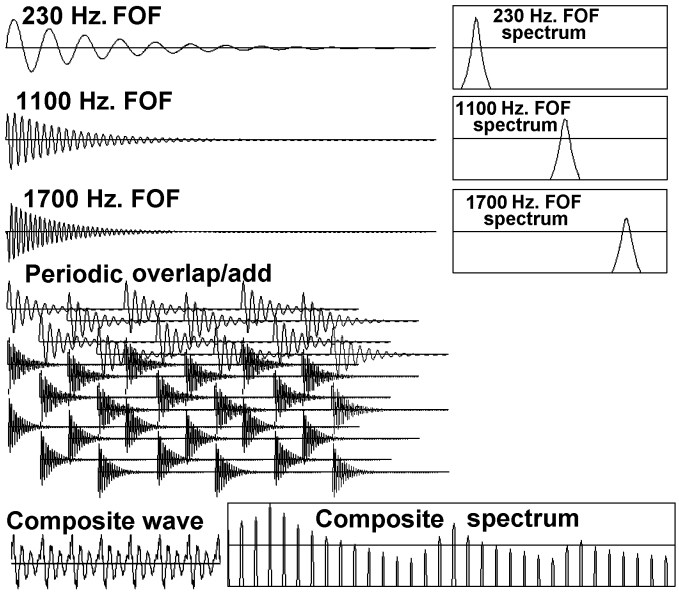


Figure 9.18: FOF synthesis of a vowel.

9.8 Waveshaping and FM Synthesis

Waveshaping synthesis involves warping a simple (usually a saw-tooth or sine wave) waveform with a non-linear function or lookup table [26, 27]. One popular form of waveshaping synthesis, called Frequency Modulation (FM), uses sine waves for both input and warping waveforms [28]. Frequency modulation relies on modulating the frequency of a simple periodic waveform with another simple periodic waveform. When the frequency of a sine wave of average frequency f_c (called the carrier wave), is modulated by another sine wave of frequency f_m (called the modulator wave), sinusoidal sidebands are created at frequencies equal to the carrier frequency plus and minus integer multiples of the modulator frequency. Figure 9.19 shows a block diagram for simple FM synthesis (one sinusoidal carrier and one sinusoidal modulator). Mathematically, FM is expressed as:

$$y(t) = \sin(2\pi t f_c + \Delta f_c \sin(2\pi t f_m)) \tag{20}$$

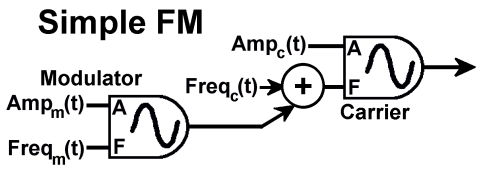


Figure 9.19: Simple FM (one carrier and one modulator sine wave) synthesis.

The index of modulation, I, is defined as $\Delta f_c / f_c$. Carson's rule (a rule of thumb) states that the number of significant bands on each side of the carrier frequency (sidebands) is roughly equal to $I + 2$. For example, a carrier sinusoid of frequency 600 Hz., a modulator sinusoid of frequency 100 Hz., and a modulation index of 3 would produce sinusoidal components of frequencies 600, {700, 500}, {800, 400}, {900, 300}, {1000, 200}, and {1100, 100} Hz. Inspecting these components reveals that a harmonic spectrum with 11 significant harmonics, based on a fundamental frequency of 100 Hz, can be produced by using only two sinusoidal generating functions. Figure 9.20 shows the spectrum of this synthesis. Sound example **S9.8** presents a series of FM-tones with increasing modulation index.

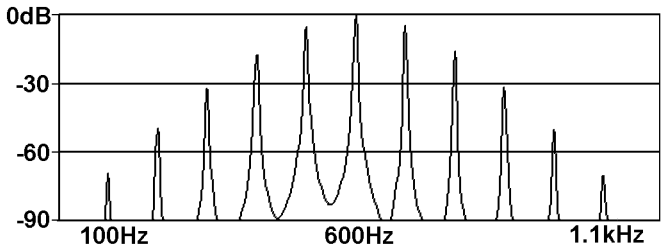


Figure 9.20: Simple FM with 600 Hz carrier, 100 Hz modulator, and index of modulation of 3.

Selecting carrier and modulator frequencies that are not related by simple integer ratios yields an inharmonic spectrum. For example, a carrier of 500 Hz, modulator of 273 Hz, and an index of 5 yields frequencies of 500 (carrier), 227, 46, 319, 592, 865, 1138, 1411 (negative sidebands), and 773, 1046, 1319, 1592, 1865, 2138, 2411 (positive sidebands). Figure 9.21 shows a spectrogram of this FM tone, as the index of modulation I is ramped from zero to 5. The synthesized waveforms at $I = 0$ and $I = 5$ are shown as well.

By setting the modulation index high enough, huge numbers of sidebands are generated, and the aliasing and addition of these results in noise. By careful selection of the component frequencies and index of modulation, and combining multiple carrier/modulator pairs, many spectra can be approximated using FM. The amplitudes and phases (described by Bessel functions) of the individual components cannot be independently controlled, however, so FM is not a truly generic sinusoidal, waveform, or spectral synthesis method.

Because of the extreme efficiency of FM (its ability to produce complex waveforms with the relatively small amounts of computer power to run a few oscillators) it became popular

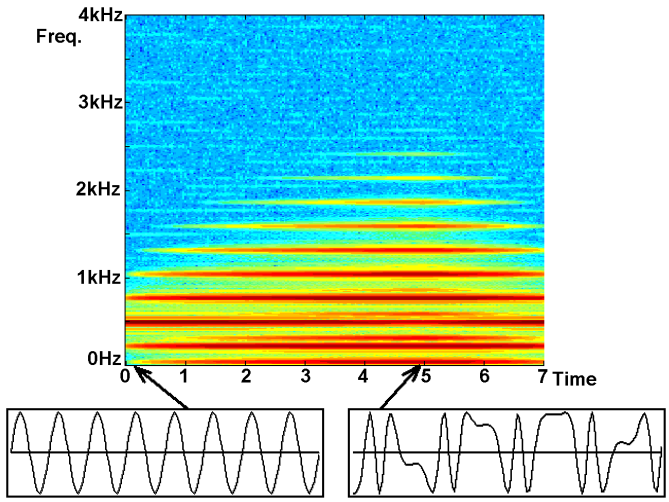


Figure 9.21: Inharmonic simple FM with 500 Hz carrier and 273 Hz modulator. The index of modulation is ramped from 0.0 to 5.0 then back to 0.0.

in the 1980s as a music synthesis algorithm. FM is sometimes used for auditory displays, partly due to popular commercial hardware, and partly due to the rich variety obtainable through manipulation of the few parameters. Carrier and Modulator frequencies determine harmonicity, inharmonicity, and pitch; the index of modulation determines spectral spread; and envelopes control time and spectral evolution. The sound/sonification designer must be careful with carrier/modulator ratio (inharmonicity), however, as often a small-seeming change can result in large categorical perceptual shifts in the resulting sound. Multiple carrier/modulator pairs lend more flexibility and more accurate spectral control. Using multiple carriers and modulators, connection topologies (algorithms) have been designed for the synthesis of complex sounds such as human voices [29], violins, brass instruments, percussion, etc.

9.9 Granular and PhISEM Synthesis

Much of classical physics can be modeled as objects interacting with each other. Lots of little objects are often called "particles." Granular synthesis involves cutting sound into "grains" (sonic particles) and reassembling them by adding, or mixing them back together [30]. The "grains" or "MicroSounds" [31] usually range in length from 10 to 100 ms. The reassembly can be systematic, but often granular synthesis involves randomized grain sizes, locations, and amplitudes. The transformed result usually bears some characteristics of the original sound, just as a mildly blended mixture of fruits still bears some attributes of the original fruits, as well as taking on new attributes due to the mixture. A FOF-Wavelet-related granular method is "Pulsar" synthesis [31]. Granular synthesis is mostly used as a music/composition type of signal processing, but some also take a more physically motivated viewpoint on

sound "grains" [32].

The PhISEM (Physically Informed Stochastic Event Modeling) algorithm is based on pseudo-random overlapping and adding of parametrically synthesized sound grains [33]. At the heart of PhISEM algorithms are particle models, characterized by basic Newtonian equations governing the motion and collisions of point masses as can be found in any introductory physics textbook. By modeling the physical interactions of many particles by their statistical behavior, exhaustive calculation of the position, and velocity of each individual particle can be avoided. By factoring out the resonances of the system, the "wavelets" can be shortened to impulses or short bursts of exponentially decaying noise. The main PhISEM assumption is that the sound-producing particle collisions follow a common statistical process known as "Poisson", (exponential probability of waiting times between individual sounds), Another assumption is that the system energy decays exponentially (for example, the decay of the sound of a maraca after being shaken once). Figure 9.22 shows the PhISEM algorithm block diagram.

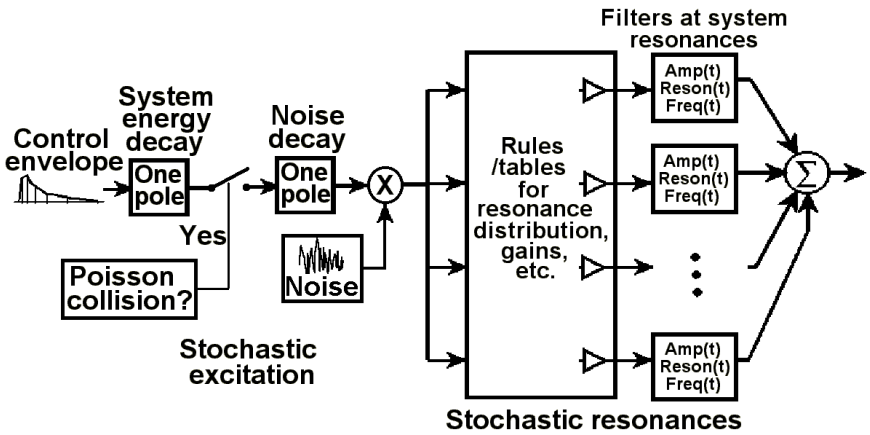


Figure 9.22: Complete PhISEM model showing stochastic resonances.

The PhISEM maraca synthesis algorithm requires only two random number calculations, two exponential decays, and one resonant filter calculation per sample. Other musical instruments that are quite similar to the maraca include the sekere and cabasa (afuche). Outside the realm of multi-cultural musical instruments, there are many real-world particle systems that exhibit one or two fixed resonances like the maraca. A bag/box of hard candy or gum, a salt shaker, a box of wooden matches, and gravel or leaves under walking feet all fit pretty well within this modeling technique.

In contrast to the maraca and guiro-like gourd resonator instruments, which exhibit one or two weak resonances, instruments such as the tambourine (timbrel) and sleigh bells use metal cymbals, coins, or bells suspended on a frame or stick. The interactions of the metal objects produce much more pronounced resonances than the maraca-type instruments, but the Poisson event and exponential system energy statistics are similar enough to justify the use of the PhISEM algorithm for synthesis. To implement these in PhISEM, more filters are used to model the individual partials, and at each collision, the resonant frequencies of the

filters are randomly set to frequencies around the main resonances. Other sounds that can be modeled using stochastic filter resonances include bamboo wind chimes (related to a musical instrument as well in the Javanese anklung) [34].

Granular and particle models lend themselves well to continuous interactive auditory displays, where the parameters can be adjusted to modify the perceived "roughness", damping, size, number of objects, etc. Inspired by the work of Gaver's Sonic Finder [35], the earcons of Blattner [36] and auditory interfaces of Brewster [37], and others, Figure 9.23 shows a simple auditory display for desktop dragging and scrolling that uses PhISEM models to indicate whether the mouse is on the desktop (sonic "texture" of sand) or on the scrollbar of a web browser (tambourine model, with pitch mapped to location of the scrollbar in the window). This is demonstrated in movie example **S9.9**.

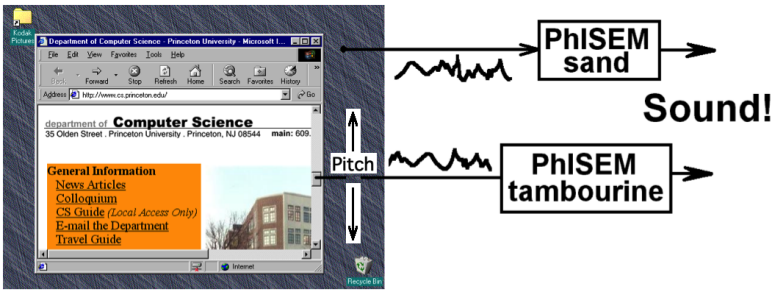


Figure 9.23: Sonically enhanced user interface.

9.10 Physical Modeling Synthesis

There is a simple differential equation that completely describes the motions of an ideal string under tension. Here it is, without derivation:

$$\frac{d^2y}{dx^2} = \frac{1}{c^2}\frac{d^2y}{dt^2} \tag{21}$$

The derivation and solution proof can be found in [16]. This equation (called "the wave equation") means that the acceleration (up and down) of any point on the string is equal to a constant times the curvature of the string at that point. The constant c is the speed of wave motion on the string, and is proportional to the square root of the string tension, and inversely proportional to the square root of the mass per unit length. This equation could be solved numerically, by sampling it in both time and space, and using the difference approximations for acceleration and curvature (much like was done with the mass-spring-damper system earlier). With boundary conditions (such as rigid terminations at each end), the solution of this equation could be expressed as a Fourier series, as was done earlier in graphical form (Figure 9.12). However, there is one more wonderfully simple solution to Equation 21, given by:

$$y(x,t) = y_l\left(t + \frac{x}{c}\right) + y_r\left(t - \frac{x}{c}\right) \tag{22}$$

This equation says that any vibration of the string can be expressed as a combination of two separate traveling waves, one traveling left (y_l) and one traveling right (y_r). They move at rate c, which is the speed of sound propagation on the string. For an ideal (no damping or stiffness) string, and ideally rigid boundaries at the ends, the wave reflects with an inversion at each end, and will travel back and forth indefinitely. This view of two traveling waves summing to make a displacement wave gives rise to the "Waveguide Filter" technique of modeling the vibrating string [38, 39]. Figure 9.24 shows a waveguide filter model of the ideal string. The two delay lines model the propagation of left and right going traveling waves. The conditions at the ends model the reflection of the traveling waves at the ends. The -1 on the left models the reflection with inversion of a displacement wave when it hits an ideally rigid termination (like a fret on a guitar neck). The -0.99 on the right reflection models the slight amount of loss that happens when the wave hits a termination that yields slightly (like the bridge of the guitar which couples the string motion to the body), and models all other losses the wave might experience (internal damping in the string, viscous losses as the string cuts the air, etc.) in making its round-trip path around the string.

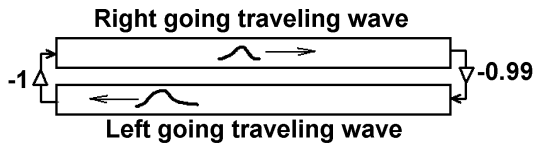


Figure 9.24: Waveguide string modeled as two delay lines.

Figure 9.25 shows the waveguide string as a digital filter block diagram. The $Z^{-P/2}$ blocks represent a delay equal to the time required for a wave to travel down the string. Thus a wave completes a round trip each P samples (down and back), which is the fundamental period of oscillation of the string, expressed in samples. Initial conditions can be injected into the string via the input $x(n)$. The output $y(n)$ would yield the right-going traveling wave component. Of course, neither of these conditions is actually physical in terms of the way a real string is plucked and listened to, but feeding the correct signal into x is identical to loading the delay lines with a pre-determined shape.

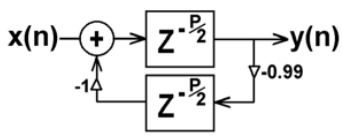


Figure 9.25: Digital filter view of waveguide string.

The impulse response and spectrum of the filter shown in Figure 9.25 is shown in Figure 9.26. As would be expected, the impulse response is an exponentially decaying train of pulses spaced $T = {}^{P}/_{\text{SRate}}$ seconds apart, and the spectrum is a set of harmonics spaced $F_0 = {}^{1}/_{T}$ Hz apart. This type of filter response and spectrum is called a "comb filter", so named because of the comb-like appearance of the time domain impulse response, and of the frequency domain harmonics.

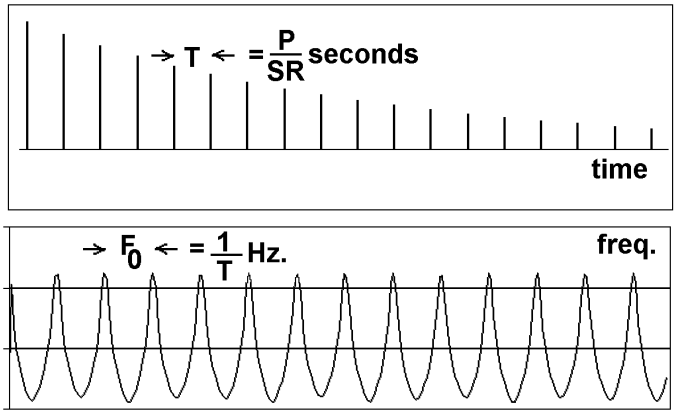


Figure 9.26: Impulse response and spectrum of comb (string) filter.

The two delay lines taken together are called a "waveguide filter." The sum of the contents of the two delay lines is the displacement of the string, and the difference of the contents of the two delay lines is the velocity of the string. If we wish to pluck the string, we simply need to load ½ of the initial string shape into each of the upper and lower delay lines. If we wish to strike the string, we would load in an initial velocity by entering a positive pulse into one delay line and a negative pulse into the other (difference = initial velocity, sum = initial position = 0). These conditions are shown in Figure 9.27.

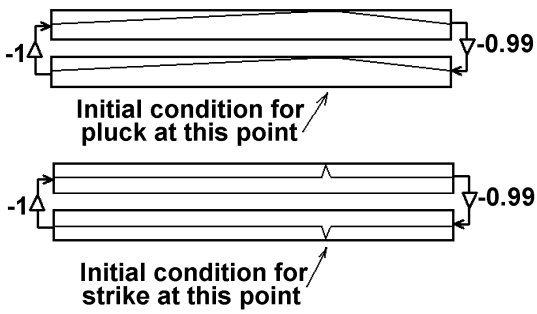


Figure 9.27: Waveguide pluck and strike initial conditions.

9.10.1 Making the String More Real (Parametric)

Figure 9.28 shows a relatively complete model of a plucked string using digital filters. The inverse comb filters model the nodal (rejected frequencies) effects of picking, and the output of an electrical pickup, emphasizing certain harmonics and forbidding others based on the pick (pickup) position [40]. Output channels for pickup position and body radiation are provided separately. A solid-body electric guitar would have no direct radiation and only

pickup output(s), while a purely acoustic guitar would have no pickup output, but possibly a family of directional filters to model body radiation in different directions [41].

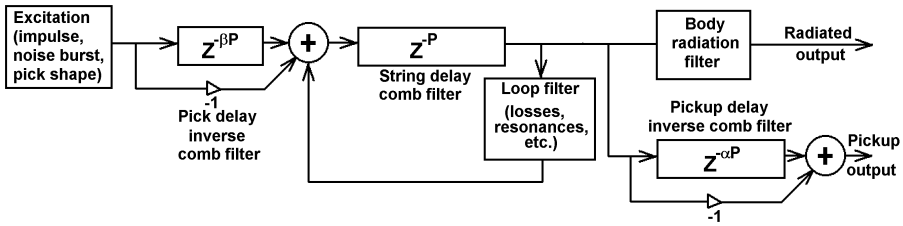


Figure 9.28: Fairly complete digital filter simulation of plucked string system.

9.10.2 Adding Stiffness

In an ideal string or membrane, the only restoring force is assumed to be the tension under which it is stretched. We can further refine solid systems such as strings and membranes to model more rigid objects, such as bars and plates, by noting that the more rigid objects exhibit internal restoring forces due to their stiffness. We know that if we bend a stiff string, it wants to return back to straightness even when there is no tension on the string. Cloth string or thread has almost no stiffness. Nylon and gut strings have some stiffness, but not as much as steel strings. Larger diameter strings have more stiffness than thin ones. In the musical world, piano strings exhibit the most stiffness. Stiffness results in the restoring force being higher (thus the speed of sound propagation as well) for high frequencies than for low. So the traveling wave solution is still true in stiff systems, but a frequency-dependent propagation speed is needed:

$$y(x,t) = y_l(t + {}^{x}/_{c(f)}) + y_r(t - {}^{x}/_{c(f)}) \tag{23}$$

and the waveguide filter must be modified to simulate frequency-dependent delay, as shown in Figure 9.29.

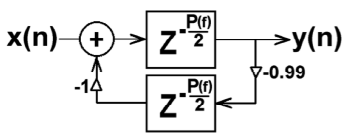


Figure 9.29: Stiffness-modified waveguide string filter.

For basic stiff strings, a function that predicts the frequencies of the partials has the form:

$$f_n = nf_0(1 + Bn^2) \tag{24}$$

where B is a number slightly greater than 0, equal to zero for perfect harmonicity (no stiffness), and increasing for increasing stiffness. This means that $P(f)$ should follow

the inverse of the $\sqrt{1+Bn^2}$ factor (round-trip time or period gets shorter with increasing frequency). Typical values of B are 0.00001 for guitar strings, and 0.004 or so for piano strings.

Unfortunately, implementing the $Z^{-P(f)/2}$ frequency-dependent delay function is not simple, especially for arbitrary functions of frequency. One way to implement the $P(f)$ function is by replacing each of the Z^{-1} with a first order all-pass (phase) filter, as shown in Figure 9.30 [40]. The first order all-pass filter has one pole and one zero, controlled by the same coefficient. The all-pass filter implements a frequency-dependent phase delay, but exhibits a gain of 1.0 for all frequencies. The coefficient α can take on values between -1.0 and 1.0. For $\alpha = 0$, the filter behaves as a standard unit delay. For $\alpha > 0.0$, the filter exhibits delays longer than one sample, increasingly long for higher frequencies. For $\alpha < 0.0$ the filter exhibits delays shorter than one sample, decreasingly so for high frequencies.

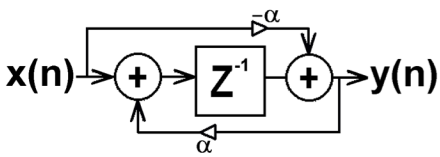


Figure 9.30: First-order all-pass filter.

It is much less efficient to implement a chain of all-pass filters than a simple delay line. But for weak stiffness it is possible that only a few all-pass sections will provide a good frequency-dependent delay. Another option is to implement a higher-order all-pass filter, designed to give the correct stretching of the upper frequencies, added to simple delay lines to give the correct longest bulk delay required.

For very stiff systems such as rigid bars, a single waveguide with all-pass filters is not adequate to give enough delay, or far too inefficient to calculate. A technique called "Banded Waveguides" employs sampling in time, space, and frequency to model stiff one-dimensional systems [42]. This can be viewed as a hybrid of modal and waveguide synthesis, in that each waveguide models the speed of sound in the region around each significant mode of the system. As an example, Figure 9.31 shows the spectrum of a struck marimba bar, with additional band-pass filters superimposed on the spectrum, centered at the three main modes. In the banded waveguide technique, each mode is modeled by a band-pass filter, plus a delay line to impose the correct round-trip delay, as shown in Figure 9.32.

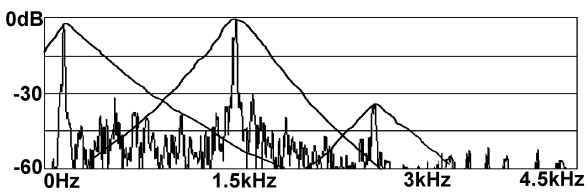


Figure 9.31: Banded decomposition of struck bar spectrum.

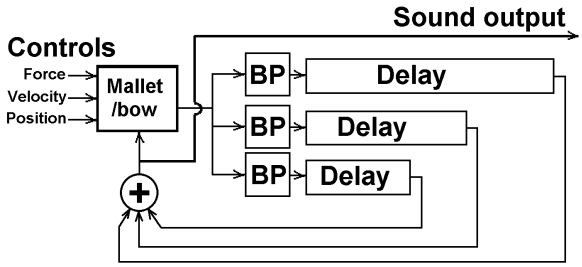


Figure 9.32: Banded waveguide model

9.10.3 Auditory Display with Strings and Bars

Plucked strings and banded waveguide models have many of the same advantages as modal synthesis, but usually with less computational cost. Pluck/strike location, damping, harmonicity/inharmonicity, and other parameters are easily manipulated to yield a wide variety of resulting sound. As an application example, Figure 9.33 shows the normalized stock prices of Red Hat Linux and Microsoft, for one year, February 2001–2002. It's pretty easy to see the trends in the stocks, but what if we wanted to track other information in addition to these curves? We might be interested in the daily volume of trading, and seemingly unrelated data like our own diastolic blood pressure during this period (to decide if it's really healthy to own these stocks). Figure 9.34 shows the five normalized curves consisting of two stock prices, two stock volumes, and one daily blood pressure measurement. It clearly becomes more difficult to tell what is going on.

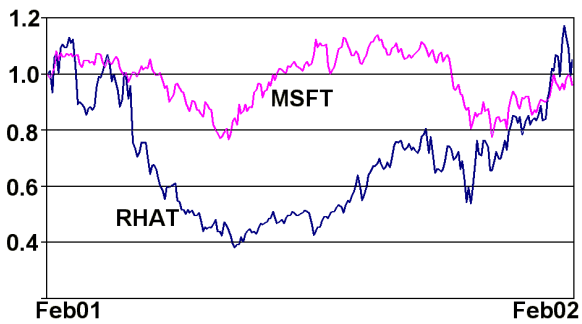


Figure 9.33: Stock prices, normalized to a $1 purchase on day 1.

Of course there are more sophisticated graphical means and techniques we could use to display this data. But some trends or patterns might emerge more quickly if we were to listen to the data. With a suitable auditory mapping of the data, we might be able to hear a lot more than we could see in a single glance. For example, the value of Red Hat could be mapped to the pitch of a plucked mandolin sound in the left speaker, with sound loudness controlled by trading volume (normalized so that even the minimum volume still makes a faint sound). Microsoft could be mapped to the pitch of a struck marimba sound in the right

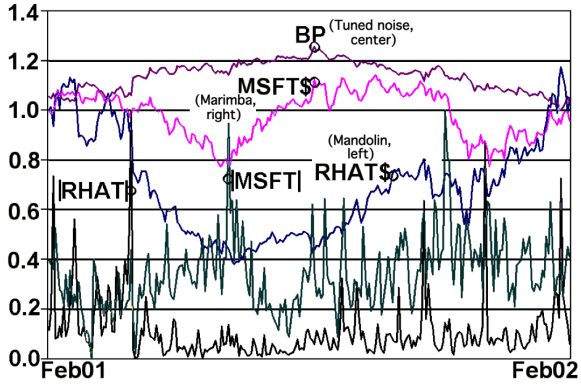


Figure 9.34: Normalized stock prices, plus volumes, plus the blood pressure of someone owning both stocks.

speaker, again with loudness controlled by trading volume. The pitch ranges are normalized so that the beginning prices on the first day of the graphs sound the same pitch. This way, on any day that the pitches are the same, our original (day 1) dollar investment in either stock would be worth the same. Finally, our normalized daily blood pressure could be mapped to the volume and pitch of a tuned noise sound, located in the center between the two speakers. Figure 9.35 shows the waveforms of these three signals. Of course, the point here is not to map visual data to visual data (waveforms), but rather to map to audio and listen to it as in sound example **S9.10**.

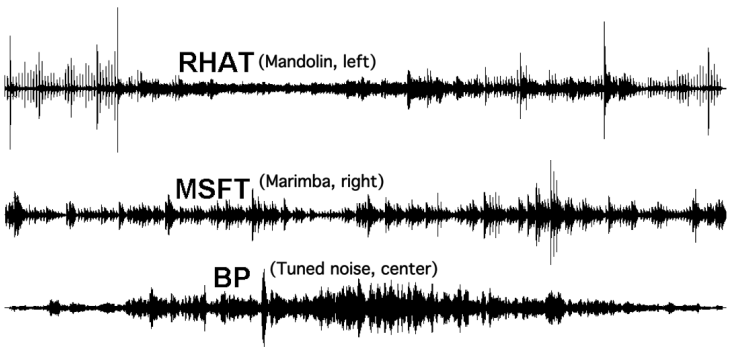


Figure 9.35: Figure 10.35 Audio waveforms of sonified stock prices, volumes, and blood pressure.

9.11 Non-Linear Physical Models

The physical models discussed so far are all linear, meaning that doubling the input excitation causes the output results to double. FM and waveshaping synthesis techniques are also

spectral in nature, and non-linear, although not necessarily physical.

Many interesting interactions in the physical world, musical and non-musical, are non-linear. For example, adding a model of bowing friction allows the string model to be used for the violin and other bowed strings. This focused non-linearity is what is responsible for turning the steady linear motion of a bow into an oscillation of the string [43, 44]. The bow sticks to the string for a while, pulling it along, then the forces become too great and the string breaks away, flying back toward rest position. This process repeats, yielding a periodic oscillation. Figure 9.36 shows a simple bowed string model, in which string velocity is compared to bow velocity, then put through a nonlinear friction function controlled by bow force. The output of the nonlinear function is the velocity input back into the string.

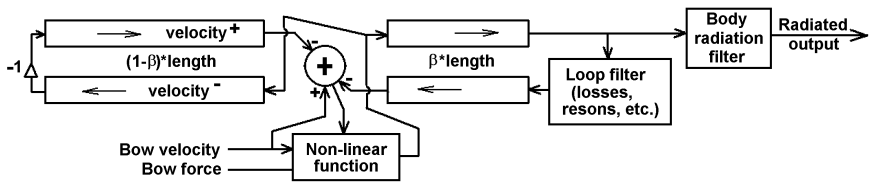


Figure 9.36: Bowed string model.

In mathematically describing the air within a cylindrical acoustic tube (like a trombone slide, clarinet bore, or human vocal tract), the defining equation is:

$$\frac{d^2P}{dx^2} = \frac{1}{c^2} \cdot \frac{d^2P}{dt^2} \tag{25}$$

which we would note has exactly the same form as Equation 21, except displacement y is replaced by pressure P. A very important paper in the history of physical modeling by [43] noted that many acoustical systems, especially musical instruments, can be characterized as a linear resonator, modeled by filters such as all-pole resonators or waveguides, and a single non-linear oscillator like the reed of the clarinet, the lips of the brass player, the jet of the flute, or the bow-string friction of the violin. Since the wave equation says that we can model a simple tube as a pair of bi-directional delay lines (waveguides), then we can build models using this simple structure. If we'd like to do something interesting with a tube, we could use it to build a flute or clarinet. Our simple clarinet model might look like the block diagram shown in Figure 9.37.

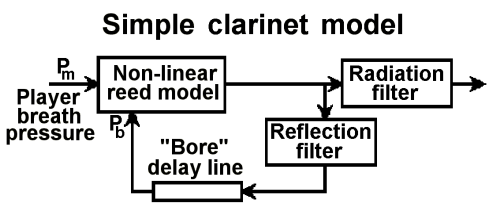
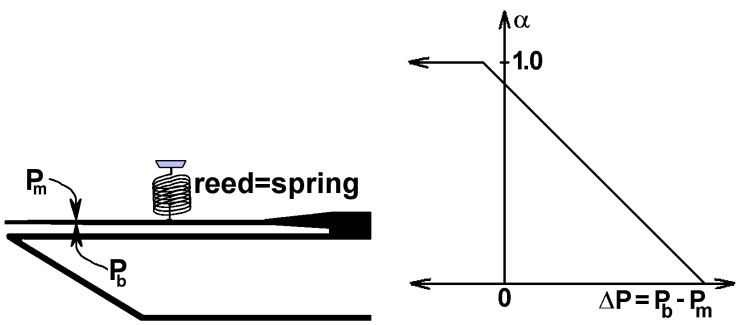


Figure 9.37: Simple clarinet model.

To model the reed, we assume that the mass of the reed is so small that the only thing that must be considered is the instantaneous force on the reed (spring). The pressure inside the bore P_b is the calculated pressure in our waveguide model, the mouth pressure P_m is an external control parameter representing the breath pressure inside the mouth of the player (see Figure 38(a)). The net force acting on the reed/spring can be calculated as the difference between the internal and external pressures, multiplied by the area of the reed (pressure is force per unit area). This can be used to calculate a reed opening position from the spring constant of the reed. From the reed opening, we can compute the amount of pressure that is allowed to leak into the bore from the player's mouth. If bore pressure is much greater than mouth pressure, the reed opens far. If mouth pressure is much greater than bore pressure, the reed slams shut. These two extreme conditions represent an asymmetric non-linearity in the reed response. Even a grossly simplified model of this non-linear spring action results in a pretty good model of a clarinet [44]. Figure 38(b) shows a plot of a simple reed reflection function (as seen from within the bore) as a function of differential pressure. Once this non-linear signal-dependent reflection coefficient is calculated (or looked up in a table), the right-going pressure injected into the bore can be calculated as $P_b^+ = \alpha P_b^- + (1-\alpha)P_m$.



(a) Reed model (b) Reed reflection table

Figure 9.38: Reed model and reflection table

The clarinet is open at the bell end, and essentially closed at the reed end. This results in a reflection with inversion at the bell and a reflection without inversion (plus any added pressure from the mouth through the reed opening) at the reed end. These boundary conditions cause odd-harmonics to dominate in the clarinet spectrum, yielding a square-like wave as we constructed before using odd Fourier harmonics.

We noted that the ideal string equation and the ideal acoustic tube equation are essentially identical. Just as there are many refinements possible to the plucked string model to make it more realistic, there are many possible improvements for the clarinet model. Replacing the simple reed model with a variable mass-spring-damper allows the modeling of a lip reed as is found in brass instruments. Replacing the reed model with an air jet model allows the modeling of flute and recorder-like instruments. With all wind or friction (bowed) excited resonant systems, adding a little noise in the reed/jet/bow region adds greatly to the quality (and behavior) of the synthesized sound.

9.11.1 Auditory Display with Nonlinear Models

Nonlinear synthesis provides some interesting possibilities for auditory display. Since the parameters influence the resultant sound in physically meaningful ways, often the "intuition" for mapping data to parameters is much more natural than in abstract models. However, the nature of many non-linear systems is that a small change in a parameter might make a huge and unpredictable change in the behavior. Such is the case in blowing a clarinet just below the "speaking threshold", resulting in a noisy sound, but by increasing the blowing pressure slightly, the tone changes to a "warm" odd-harmonic oscillation. As any parent of a child studying violin knows too well, slight changes in bowing parameters make gross (literally) changes in the output sound (noisy, scratchy, pitched but irritating oscillation, beautiful sonorous singing quality).

So the interesting and physically meaningful behavior of many non-linear synthesis models is a double-edged sword; rich variety of sounds and responsiveness to small parameter changes, vs. unpredictability and non-linear mapping of parameters to output sound. For this reason, care should be taken in using such systems for reliable and repeatable auditory displays.

9.12 Synthesis for Auditory Display, Conclusion

There are other types of sound synthesis, such as random waveform and/or spectrum generation using genetic algorithms, fractals, neural networks, and other popular techniques that have been applied to a host of other problems in other domains [45]. These techniques can also be applied to the derivation and manipulation of parameters for parametric synthesis models. Scanned synthesis [46] is a hybrid of physical and wavetable synthesis, where a trajectory of a physical model running at one update rate is constantly scanned as a form of self-modifying wavetable. In a way this is a sonification in itself, where a physical process (not necessarily running at audio rate or generating audio itself) generates data that is scanned as a waveform.

Other projects involving the mapping of physical, pseudo-physical, or physically inspired simulations to synthesis range from sonifying a rolling ball or pouring water into a glass [47], to using sound textures to simulate the sound of swords/sticks traveling rapidly through the air [48]. These are examples of the mapping of physical process parameters to the control parameters of various synthesis techniques (like the PhISEM algorithm described above). Others have taken the approach of looking at the target sound to be made for games, sound effects, or other applications, and then deriving custom synthesis "patches" tailored to each sound or class of sound [49]. Again, these methods rely heavily on the basic synthesis methods described in this chapter. These are all examples of using one or more processes or models to control the parameters of sound synthesis, thus related to auditory display and sonification. Further, since many of these include a parametric model for generating synthesis parameters themselves, and since these models themselves have input parameters, they can be used in auditory displays, or for sonifying abstract data. These are examples of "mapping", which is covered at length in other chapters of this book.

On the simple side, just plain old pulses are interesting in many cases, such as the familiar Geiger counter (which could be viewed as a simple case of granular synthesis). Historically, computer researchers would attach an amplifier and speaker to a particular bit or set of

bits inside a computer, and use the resulting pulse wave output to monitor and diagnose information about the behavior and state of the computer (a loop is easy to hear, access to memory, network traffic or collisions, all are possible to learn as direct sonifications). In fact, just placing an AM radio near a computer and tuning to certain frequencies allows the inner workings to be heard, as the sea of pulses at different rates generates Radio Frequency (RF) emissions. These simple and direct mappings rely mostly on the human ability to learn the sound of a process or state, rather than an explicit mapping of data to the parameters of a parametric synthesis algorithm.

On a more neuro-ecological note, the use of speech or speech-like sounds is perhaps the most powerful form of auditory display. Indeed so much of the legacy of sound synthesis comes from research on speech and communications, as our LPC, Formant, FOF, Vocoder algorithms point up. The danger, however, of using speech-like sounds is that they might trigger our linguistic "circuitry" and evoke lots of semantic, emotional, cultural, and other results, which could vary greatly from person to person, and culture to culture. Speech-motivated models are a very powerful tool for conveying even non-speech information, due to our sensitivity to pitch, quality, articulation, breathiness, etc. but designers must be cautious in using them.

Auditory display designers have a rich variety of techniques and tools at their disposal, and with the power of modern computers (even PDAs and cell phones), parametric synthesis is easily possible. The author hopes that more researchers and interaction designers will exploit the potential of synthesis in the future, rather than just using recorded PCM or "off the shelf" sounds.

Bibliography

[1] K. Pohlmann, *Principles of Digital Audio*, McGraw Hill, 2000.

[2] T. Dutoit, *An Introduction to Text-To-Speech Synthesis*, Kluwer Academic Publishers, 1996.

[3] J. O. Smith and P. Gossett, "A flexible sampling-rate conversion method". In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, San Diego*, vol. 2, (New York), pp. 19.4.1–19.4.2, IEEE Press, 1984.

[4] R. Bristow-Johnson, "Wavetable Synthesis 101: a Fundamental Perspective" *AES 101 Conference*, Paper 4400, 1996.

[5] M. V. Mathews, *The Technology of Computer Music*. Cambridge, Massachusetts: MIT Press, 1969.

[6] R. Bracewell, *The Fourier Transform and its Applications*, McGraw Hill, 1999.

[7] R. Portnoff, "Time-scale modifications of speech based on short time Fourier analysis", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29:3, pp. 374–390, 1981.

[8] A. Moorer, "The Use of the Phase Vocoder in Computer Music Applications". *Journal of the Audio Engineering Society*, 26 (1/2), pp. 42–45, 1978.

[9] M. Dolson, "The Phase Vocoder: A Tutorial", *Computer Music Journal*, 10 (4), pp. 14–27, 1986.

[10] *Proceedings of the International Symposium on Music Information Retrieval*, 2003-Present, website at http://www.ismir.net

[11] R. J. McAulay and T. Quatieri, "Speech Analysis/Synthesis Based on a Sinusoidal Representation", *IEEE Trans. ASSP*-34, pp. 744–754, 1986.

[12] X. Serra and J. O. Smith. "Spectral Modeling Synthesis: Sound Analysis/Synthesis Based on a Deterministic plus Stochastic Decomposition". *Computer Music Journal*, 14(4), pp. 12–24, 1990.

[13] T. Verma and T. Meng, "An Analysis/Synthesis Tool for Transient Signals that Allows a Flexible Sines+Transients+Noise Model for Audio", *1998 IEEE ICASSP-98*, pp. 77–78, Seattle, WA, 1998.

[14] U. Zölzer, Ed., *Digital Audio Effects*, John Wiley & Sons, 2002.

[15] P. Cook, *Real Sound Synthesis for Interactive Applications*, AK Peters LTD, 2002.

[16] P. Morse, *Vibration and Sound*, American Institute of Physics, for the Acoustical Society of America, 1986.

[17] K. Steiglitz, *A Digital Signal Processing Primer*, Addison Wesley, 1996.

[18] J. M. Adrien, *The missing link: Modal synthesis*. In: De Poli, Piccialli, and Roads, editors, *Representations of Musical Signals*, chapter 8, pp. 269–297. MIT Press, Cambridge, 1991.

[19] H. Dudley, 1939, "The Vocoder", Bell Laboratories Record, December, 1939. Reprinted in *IEEE Transactions on Acoustics, Speech and Signal Processing* ASSP-29(3):347–351, 1981.

[20] I. Daubechies "Orthonormal Bases of Compactly Supported Wavelets", Communications on Pure and Applied Math. Vol. 41, pp. 909–996, 1988.

[21] B. Atal, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave". Journal of the Acoustical Society of America V47, S65(A), 1970.

[22] A. Moorer, "The Use of Linear Prediction of Speech in Computer Music Applications", Journal of the Audio Engineering Society 27(3): pp. 134–140, 1979.

[23] D. Klatt, "Software for a Cascade/Parallel Formant Synthesizer", Journal of the Acoustical Society of America 67(3), pp. 971–995, 1980.

[24] J. L. Kelly Jr. and C. C. Lochbaum, "Speech Synthesis", in *Proceedings of the Fourth ICA*, Copenhagen, Denmark, pp. 1–4, 1962.

[25] X. Rodet, "Time-Domain Formant-Wave-Function Synthesis", *Computer Music Journal* 8 (3), pp. 9–14, 1984.

[26] M. Le Brun, "Digital Waveshaping Synthesis" in *Journal of the Audio Engineering Society*, 27(4), pp. 250–266, 1979.

[27] D. Arfib, "Digital synthesis of complex spectra by means of multiplication of non-linear distorted sine waves". *Journal of the Audio Engineering Society* 27: 10, pp. 757–768, 1979.

[28] J. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation", Journal of the Audio Engineering Society 21(7), pp. 526–534, 1973; reprinted in Computer Music Journal 1(2), 1977; reprinted in *Foundations of Computer Music*, C. Roads and J. Strawn (eds.). MIT Press, 1985.

[29] J. Chowning, "Frequency Modulation Synthesis of the Singing Voice" in Mathews, M. V., and J. R. Pierce (eds), *Current Directions in Computer Music Research*, pp. 57–64,Cambridge, MIT Press, 1980.

[30] C. Roads, "Asynchronous Granular Synthesis" in G. De Poli, A. Piccialli, and C. Roads, eds. 1991. *Representations of Musical Signals*. Cambridge, Mass: The MIT Press, pp. 143–185, 1991.

[31] C. Roads, *Microsound*, Cambridge, MIT Press, 2001.

[32] C. Cadoz, A. Luciani and J. L. Florens. "CORDIS-ANIMA: A Modeling and Simulation System for Sound Image Synthesis-The General Formalization" *Computer Music Journal*, 17(1), pp. 23–29, 1993.

[33] P. Cook, "Physically Informed Sonic Modeling (PhISM): Percussion Instruments." In *Proceedings of the ICMC,* Hong Kong, pp. 228–231, 1996.

[34] P. Cook, "Physically Informed Sonic Modeling (PhISM): Synthesis of Percussive Sounds", *Computer Music Journal,* 21:3, pp. 38–49, 1997.

[35] W. Gaver, "The SonicFinder: An interface that uses auditory icons", *Human Computer Interaction*, 4(1), pp 67–94, 1989.

[36] M. Blattner, D. Sumikawa, & R. Greenberg, "Earcons and icons: Their structure and common design principles", *Human Computer Interaction*, 4(1), pp 11–44, 1989.

[37] S. Brewster, "Sonically-Enhanced Drag and Drop", *Proc. 5th Int. Conf. on Auditory Display (ICAD98)*, 1998.

[38] K. Karplus and A.Strong. "Digital synthesis of plucked-string and drum timbres". *Computer Music Journal*, 7(2), pp. 43–55, 1983.

[39] J. O. Smith, “Acoustic Modeling Using Digital Waveguides”, in Roads et. al. (eds.), *Musical Signal Processing*, pp. 221–263, Netherlands, Swets and Zeitlinger, 1997.

[40] Jaffe, D. A. and Smith, J. O. Extensions of the Karplus-Strong plucked string algorithm. *Computer Music Journal*, 7(2), pp. 56–69, 1983.

[41] P. R. Cook, and D. Trueman, “Spherical Radiation from Stringed Instruments: Measured, Modeled, and Reproduced”, *Journal of the Catgut Acoustical Society,* pp. 8–14, November 1999.

[42] George Essl and Perry Cook. Banded waveguides: Towards physical modeling of bar percussion instruments. In *Proc. 1999 Int. Computer Music Conf.*, pp. 321–324, Beijing, China,Computer Music Association, 1999.

[43] M. E. McIntyre, R. T. Schumacher, and J. Woodhouse, “On the oscillations of musical instruments.” *J. Acoust. Soc. Am.*, 74(5), pp. 1325–1345, 1983.

[44] J. O. Smith, “Efficient simulation of the reed-bore and bow-string mechanisms”, *Proceedings of the International Computer Music Conference*, pp. 275–280, 1986.

[45] J. Yuen and A. Horner, “Hybrid Sampling-Wavetable Synthesis with Genetic Algorithms”, Journal of the Audio Engineering Society, 45:5, 1997.

[46] B. Verplank, M. Mathews, and R. Shaw, “Scanned synthesis”, *Proceedings of the ICMC*, pp. 368–371, Berlin, 2000.

[47] D. Rocchesso and F. Fontana, (Eds.) *The Sounding Object*, http://www.soundobject.org/SObBook, 2003.

[48] Y. Dobashi, T. Yamamoto, and T. Nishita, “Real-Time Rendering of Aerodynamic Sound Using Sound Textures Based on Computational Fluid Dynamics”, *Proc. ACM SIGGRAPH*, pp. 732–740, 2003.

[49] A. Farnell, *Designing Sound*, Cambridge, MIT Press, 2010.