1

Sound synthesis and physical modeling

Before entering into the main development of this book, it is worth stepping back to get a larger picture of the history of digital sound synthesis. It is, of course, impossible to present a complete treatment of all that has come before, and unnecessary, considering that there are several books which cover the classical core of such techniques in great detail; those of Moore [240], Dodge and Jerse [107], and Roads [289], and the collections of Roads et al. [290], Roads and Strawn [291], and DePoli et al. [102], are probably the best known. For a more technical viewpoint, see the report of Tolonen, Välimäki, and Karjalainen [358], the text of Puckette [277], and, for physical modeling techniques, the review article of Välimäki et al. [376]. This chapter is intended to give the reader a basic familiarity with the development of such methods, and some of the topics will be examined in much more detail later in this book. Indeed, many of the earlier developments are perceptually intuitive, and involve only basic mathematics; this is less so in the case of physical models, but every effort will be made to keep the technical jargon in this chapter to a bare minimum.

It is convenient to make a distinction between earlier, or abstract, digital sound synthesis methods, to be introduced in Section 1.1, and those built around physical modeling principles, as detailed in Section 1.2. (Other, more elaborate taxonomies have been proposed [328, 358], but the above is sufficient for the present purposes.) That this distinction is perhaps less clear-cut than it is often made out to be is a matter worthy of discussion—see Section 1.3, where some more general comments on physical modeling sound synthesis are offered, regarding the relationship among the various physical modeling methodologies and with earlier techniques, and the fundamental limitations of computational complexity.

In Figure 1.1, for the sake of reference, a timeline showing the development of digital sound synthesis methods is presented; dates are necessarily approximate. For brevity, only those techniques which bear some relation to physical modeling sound synthesis are noted—such a restriction is a subjective one, and is surely a matter of some debate.

Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics Stefan Bilbao

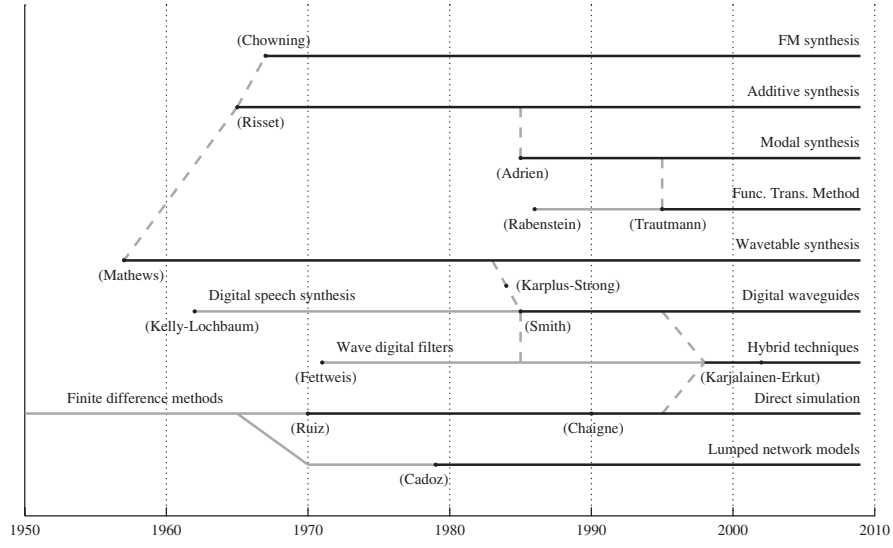


Figure 1.1 Historical timeline for digital sound synthesis methods. Sound synthesis techniques are indicated by dark lines, antecedents from outside of musical sound synthesis by solid grey lines, and links by dashed grey lines. Names of authors/inventors appear in parentheses; dates are approximate, and in some cases have been fixed here by anecdotal information rather than publication dates.

1.1 Abstract digital sound synthesis

The earliest synthesis work, beginning in the late 1950s[1], saw the development of abstract synthesis techniques, based primarily on operations which fit well into a computer programming framework: the basic components are digital oscillators, filters, and stored "lookup" tables of data, read at varying rates. Though the word "synthesis" is used here, it is important to note that in the case of tables, as mentioned above, it is of course possible to make use of non-synthetic sampled audio recordings. Nonetheless, such methods are often lumped in with synthesis itself, as are so-called analysis–synthesis methods which developed in the 1970s after the invention of the fast Fourier transform [94] some years earlier.

It would be cavalier (not to mention wrong) to assume that abstract techniques have been superseded; some are extremely computationally efficient, and form the synthesis backbone of many of the most popular music software packages, such as Max/MSP [418], Pd [276], Csound [57], SuperCollider [235], etc. Moreover, because of their reliance on accessible signal processing constructs such as tables and filters, they have entered the lexicon of the composer of electroacoustic music in a definitive way, and have undergone massive experimentation. Not surprisingly, a huge variety of hybrids and refinements have resulted; only a few of these will be detailed here.

The word "abstract," though it appears seldom in the literature [332, 358], is used to describe the techniques mentioned above because, in general, they do not possess an associated underlying physical interpretation—the resulting sounds are produced according to perceptual and mathematical, rather than physical, principles. There are some loose links with physical modeling, most notably between additive methods and modal synthesis (see Section 1.1.1), subtractive synthesis and

[1] Though the current state of digital sound synthesis may be traced back to work at Bell Laboratories in the late 1950s, there were indeed earlier unrelated attempts at computer sound generation, and in particular work done on the CSIRAC machine in Australia, and the Ferranti Mark I, in Manchester [109].

source-filter models (see Section 1.1.2), and wavetables and wave propagation in one-dimensional (1D) media (see Section 1.1.3), but it is probably best to think of these methods as pure constructs in digital signal processing, informed by perceptual, programming, and sometimes efficiency considerations. For more discussion of the philosophical distinctions between abstract techniques and physical modeling, see the articles by Smith [332] and Borin, DePoli, and Sarti [52].

1.1.1 Additive synthesis

Additive analysis and synthesis, which dates back at least as far as the work of Risset [285] and others [143] in the 1960s, though not the oldest digital synthesis method, is a convenient starting point; for more information on the history of the development of such methods, see [289] and [230]. A single sinusoidal oscillator with output $u(t)$ is defined, in continuous time, as

$$u(t) = A\cos(2\pi f_0 t + \phi) \tag{1.1}$$

Here, t is a time variable, and A, f_0, and ϕ are the amplitude, frequency, and initial phase of the oscillator, respectively. In the simplest, strictest manifestation of additive synthesis, these parameters are constants: A scales roughly with perceived loudness and f_0 with pitch. For a single oscillator in isolation, the initial phase ϕ is of minimal perceptual relevance, and is usually not represented in typical symbolic representations of the oscillator—see Figure 1.2. In discrete time, where the sample rate is given by f_s, the oscillator with output u^n is defined similarly as

$$u^n = A\cos(2\pi f_0 n/f_s + \phi) \tag{1.2}$$

where n is an integer, indicating the time step.

The sinusoidal oscillator, in computer music applications, is often represented using the symbolic shorthand shown in Figure 1.2(a). Using Fourier theory, it is possible to show that any real-valued continuous or discrete waveform (barring some technical restrictions relating to continuity) may be decomposed into an integral over a set of such sinusoids. In continuous time, if the waveform to be decomposed is periodic with period T, then an infinite sum of such sinusoids, with frequencies which are integer multiples of $1/T$, suffices to describe the waveform completely. In discrete time, if the waveform is periodic with integer period $2N$, then a finite collection of N oscillators yields a complete characterization.

The musical interest of additive synthesis, however, is not necessarily in exact decompositions of given waveforms. Rather, it is a loosely defined body of techniques based around the use

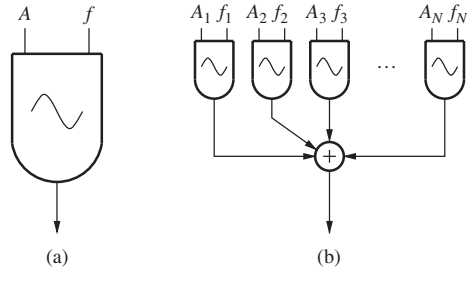


Figure 1.2 (a) Symbolic representation of a single sinusoidal oscillator, output at bottom, dependent on the parameters A, representing amplitude, and f, representing frequency. In this representation, the specification of the phase ϕ has been omitted, though some authors replace the frequency control parameter by a phase increment, and indicate the base frequency in the interior of the oscillator symbol. (b) An additive synthesis configuration, consisting of a parallel combination of N such oscillators, with parameters A_l, and f_l, $l = 1, \ldots, N$, according to (1.3).

of combinations of such oscillators in order to generate musical sounds, given the underlying assumption that sinusoids are of perceptual relevance in music. (Some might find this debatable, but the importance of pitch throughout the history of acoustic musical instruments across almost all cultures favors this assertion.) A simple configuration is given, in discrete time, by the sum

$$u^n = \sum_{l=1}^{N} A_l \cos(2\pi f_l n / f_s + \phi_l) \tag{1.3}$$

where in this case N oscillators, of distinct amplitudes, frequencies, and phases A_l, f_l, and ϕ_l, for $l = 1, \ldots, N$, are employed. See Figure 1.2(b). If the frequencies f_l are close to integer multiples of a common "fundamental" frequency f_0, then the result will be a tone at a pitch corresponding to f_0. But unpitched inharmonic sounds (such as those of bells) may be generated as well, through avoidance of common factors among the chosen frequencies. With a large enough N, one can, as mentioned above, generate any imaginable sound. But the generality of such an approach is mitigated by the necessity of specifying up to thousands of amplitudes, frequencies, and phases. For a large enough N, and taking the entire space of possible choices of parameters, the set of sounds which will *not* sound simply like a steady unpitched tone is vanishingly small. Unfortunately, using such a simple sum of sinusoids, many musically interesting sounds will certainly lie in the realm of large N.

Various strategies (probably hundreds) have been employed to render additive synthesis more musically tractable [310]. Certainly the most direct is to employ slowly time-varying amplitude envelopes to the outputs of single oscillators or combinations of oscillators, allowing global control of the attack/decay characteristics of the resulting sound without having to rely on delicate phase cancellation phenomena. Another is to allow oscillator frequencies to vary, at sub-audio rates, so as to approximate changes in pitch. In this case, the definition (1.1) should be extended to include the notion of instantaneous frequency—see Section 1.1.4. For an overview of these techniques, and others, see the standard texts mentioned in the opening remarks of this chapter.

Another related approach adopted by many composers has been that of analysis-synthesis, based on sampled waveforms. This is not, strictly speaking, a pure synthesis technique, but it has become so popular that it is worth mentioning here. Essentially, an input waveform is decomposed into sinusoidal components, at which point the frequency domain data (amplitudes, phases, and sometimes frequencies) are modified in a perceptually meaningful way, and the sound is then reconstructed through inverse Fourier transformation. Perhaps the best known tool for analysis–synthesis is the phase vocoder [134, 274, 108], which is based on the use of the short-time Fourier transformation, which employs the fast Fourier transformation [94]. Various effects, including pitch transposition and time stretching, as well as cross-synthesis of spectra, can be obtained, through judicious modification of frequency domain data. Even more refined tools, such as spectral modeling synthesis (SMS) [322], based around a combination of Fourier and stochastic modeling, as well as methods employing tracking of sinusoidal partials [233], allow very high-quality manipulation of audio waveforms.

1.1.2 Subtractive synthesis

If one is interested in producing sounds with rich spectra, additive synthesis, requiring a separate oscillator for each desired frequency component, can obviously become quite a costly undertaking. Instead of building up a complex sound, one partial at a time, another way of proceeding is to begin with a very rich sound, typically simple to produce and lacking in character, such as white noise or an impulse train, and then shape the spectrum using digital filtering methods. This technique is often referred to as subtractive synthesis—see Figure 1.3. It is especially powerful when the filtering applied is time varying, allowing for a good first approximation to musical tones of unsteady timbre (this is generally the norm).

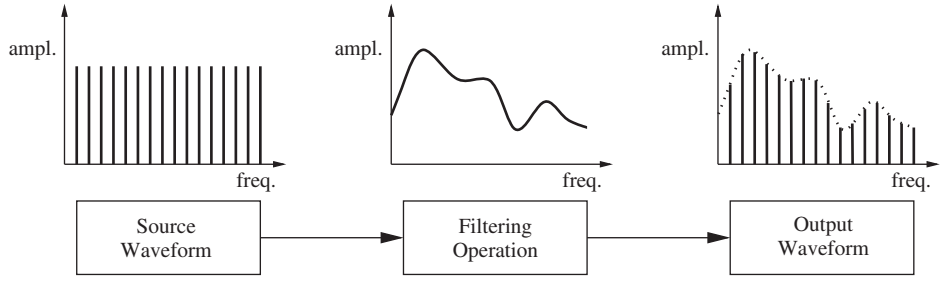


Figure 1.3 Subtractive synthesis.

Subtractive synthesis is often associated with physical models [240], but this association is a tenuous one at best.[2] What is meant is that many linear models of sound production may be broken down into source and filtering components [411]. This is particularly true of models of human speech, in which case the glottis is assumed to produce a wide-band signal (i.e., a signal somewhat like an impulse train under voiced conditions, and white noise under unvoiced conditions) which is filtered by the vocal tract, yielding a spectrum with pronounced peaks (formants) which indicate a particular vocal timbre. In this book, however, because of the emphasis on time domain methods, the source-filter methodology will not be explicitly employed. Indeed, for distributed nonlinear problems, to which frequency domain analysis is ill suited, it is of little use and relatively uninformative. Even in the linear case, it is worth keeping in mind that the connection of two objects will, in general, modify the characteristic frequencies of both—strictly speaking, one cannot invoke the notion of individual frequencies of components in a coupled system. Still, the breakdown of a system into a lumped/distributed pair representing an excitation mechanism and the instrument body is a very powerful one, even if, in some cases, the behavior of the body cannot be explained in terms of filtering concepts.

1.1.3 Wavetable synthesis

The most common computer implementation of the sinusoidal oscillator is not through direct calculation of values of the cosine or sine function, but, rather, through the use of a stored table containing values of one period of a sinusoidal waveform. A sinusoid at a given frequency may then be generated by reading through the table, circularly, at an appropriate rate. If the table contains N values, and the sample rate is f_s, then the generation of a sinusoid at frequency f_0 will require a jump of f_s/f_0N values in the table over each sample period, using interpolation of some form. Clearly, the quality of the output will depend on the number of values stored in the table, as well as on the type of interpolation employed. Linear interpolation is simple to program [240], but other more accurate methods, built around higher-order Lagrange interpolation, are also used—some material on fourth-order interpolation (in the spatial context) appears in Section 5.2.4. All-pass filter approximations to fractional delays are also possible, and are of special interest in physical modeling applications [372, 215].

It should be clear that one can store values of an arbitrary waveform in the table, not merely those corresponding to a sinusoid. See Figure 1.4. Reading through such a table at a fixed rate will generate a quasi-periodic waveform with a full harmonic spectrum, all at the price of a single table read and interpolation operation per sample period—it is no more expensive, in terms of computer arithmetic, than a single oscillator. As will be seen shortly, there is an extremely fruitful physical

[2] A link does exist, however, when analog synthesizer modules, often behaving according to principles of subtractive synthesis, are digitally simulated as "virtual analog" components.

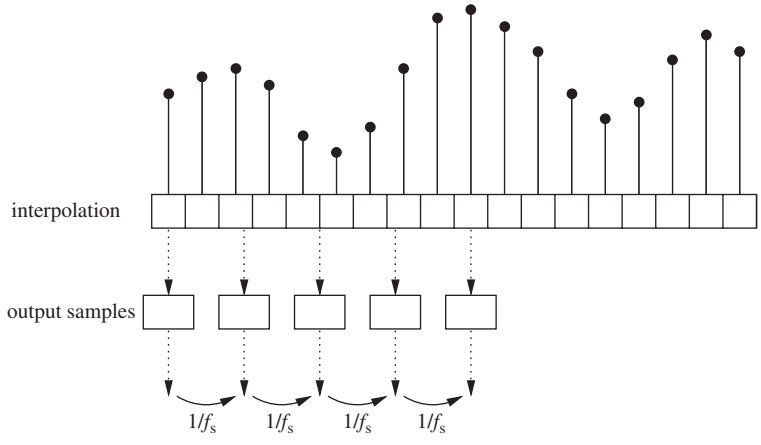


Figure 1.4 Wavetable synthesis. A buffer, filled with values, is read through at intervals of $1/f_s$ s, where f_s is the sample rate. Interpolation is employed.

interpretation of wavetable synthesis, namely the digital waveguide, which revolutionized physical modeling sound synthesis through the same efficiency gains—see Section 1.2.3. Various other variants of wavetable synthesis have seen use, such as, for example, wavetable stacking, involving multiple wavetables, the outputs of which are combined using crossfading techniques [289]. The use of tables of data in order to generate sound is perhaps the oldest form of sound synthesis, dating back to the work of Mathews in the late 1950s.

Tables of data are also associated with so-called sampling synthesis techniques, as a de facto means of data reduction. Many musical sounds consist of a short attack, followed by a steady pitched tone. Such a sound may be efficiently reproduced through storage of only the attack and a single period of the pitched part of the waveform, which is stored in a wavetable and looped [358]. Such methods are the norm in most commercial digital piano emulators.

1.1.4 AM and FM synthesis

Some of the most important developments in early digital sound synthesis derived from extensions of the oscillator, through time variation of the control parameters at audio rates.

AM, or amplitude modulation synthesis, in continuous time, and employing a sinusoidal carrier (of frequency f_0) and modulator (of frequency f_1), generates a waveform of the following form:

$$u(t) = (A_0 + A_1 \cos(2\pi f_1 t)) \cos(2\pi f_0 t)$$

where A_0 and A_1 are free parameters. The symbolic representation of AM synthesis is shown in Figure 1.5(a). Such an output consists of three components, as also shown in Figure 1.5(a), where the strength of the component at the carrier frequency is determined by A_0, and those of the side components, at frequencies $f_0 \pm f_1$, by A_1. If $A_0 = 0$, then ring modulation results. Though the above example is concerned with the product of sinusoidal signals, the concept of AM (and frequency modulation, discussed below) extends to more general signals with ease.

Frequency modulation (FM) synthesis, the result of a serendipitous discovery by John Chowning at Stanford in the late 1960s, was the greatest single breakthrough in digital sound synthesis [82]. Instantly, it became possible to generate a wide variety of spectrally rich sounds using a bare minimum of computer operations. FM synthesis requires no more computing power than a few digital oscillators, which is not surprising, considering that FM refers to the modulation

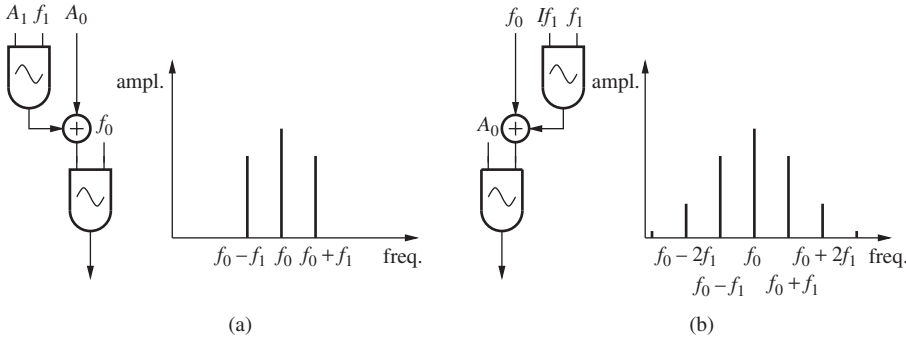


Figure 1.5 Symbolic representation and frequency domain description of output for (a) amplitude modulation and (b) frequency modulation.

of the frequency of a digital oscillator. As a result, real-time synthesis of complex sounds became possible in the late 1970s, as the technique was incorporated into various special purpose digital synthesizers—see [291] for details. In the 1980s, FM synthesis was very successfully commercialized by the Yamaha Corporation, and thereafter permanently altered the synthetic soundscape.

FM synthesis, like AM synthesis, is also a direct descendant of synthesis based on sinusoids, in the sense that in its simplest manifestation it makes use of only two sinusoidal oscillators, one behaving as a carrier and the other as a modulator. See Figure 1.5(b). The functional form of the output, in continuous time, is usually written in terms of sine functions, and not cosines, as

$$u(t) = A_0(t) \sin(2\pi f_0 t + I \sin(2\pi f_1 t)) \tag{1.4}$$

where, here, f_0 is the carrier frequency, f_1 the modulation frequency, and I the so-called modulation index. It is straightforward to show [82] that the spectrum of this signal will exhibit components at frequencies $f_0 + qf_1$, for integer q, as illustrated in Figure 1.5(b). The modulation index I determines the strengths of the various components, which can vary in a rather complicated way, depending on the values of associated Bessel functions. $A_0(t)$ can be used to control the envelope of the resulting sound.

In fact, a slightly better formulation of the output waveform (1.4) is

$$u(t) = A_0(t) \sin\left(2\pi \int_0^t f_0 + If_1 \cos(2\pi f_1 t')dt'\right)$$

where the instantaneous frequency at time t may be seen to be (or rather defined as) $f_0 + If_1 \cos(2\pi f_1 t)$. The quantity If_1 is often referred to as the peak frequency deviation, and written as Δf [240]. Though this is a subtle point, and not one which will be returned to in this book, the symbolic representation in Figure 1.5(b) should be viewed in this respect.

FM synthesis has been exhaustively researched, and many variations have resulted. Among the most important are feedback configurations, useful in regularizing the behavior of the side component magnitudes and various series and parallel multiple oscillator combinations.

1.1.5 Other methods

There is no shortage of other techniques which have been proposed for sound synthesis; some are variations on those described in the sections above, but there are several which do not fall neatly into any one category. This is not to say that such techniques have not seen success; it is rather

that they do not fit naturally into the evolution of abstract methods into physically inspired sound synthesis methods, the subject of this book.

One of the more interesting is a technique called waveshaping [219, 13, 288], in which case an input waveform (of natural or synthetic origin) is used as a time-varying index to a table of data. This, like FM synthesis, is a nonlinear technique—a sinusoid at a given frequency used as the input will generate an output which contains a number of harmonic components, whose relative amplitudes depend on the values stored in the table. Similar to FM, it is capable of generating rich spectra for the computational cost of a single oscillator, accompanied by a table read; a distinction is that there is a level of control over the amplitudes of the various partials through the use of Chebyshev polynomial expansions as a representation of the table data.

Granular synthesis [73], which is very popular among composers, refers to a large body of techniques, sometimes very rigorously defined (particularly when related to wavelet decompositions [120]), sometimes very loosely. In this case, the idea is to build complex textures using short-duration sound "grains," which are either synthetic, or derived from analysis of an input waveform. The grains, regardless of how they are obtained, may then be rearranged and manipulated in a variety of ways. Granular synthesis encompasses so many different techniques and methodologies that it is probably better thought of as a philosophy, rather than a synthesis technique. See [287] for a historical overview.

Distantly related to granular synthesis are methods based on overlap adding of pulses of short duration, sometimes, but not always, to emulate vocal sounds. The pulses are of a specified form, and depend on a number of parameters which serve to alter the timbre; in a vocal setting, the rate at which the pulses recur determines the pitch, and a formant structure, dependent on the choice of the free parameters, is imparted to the sound output. The best known are the so-called FOF [296] and VOSIM [186] techniques.

1.2 Physical modeling

The algorithms mentioned above, despite their structural elegance and undeniable power, share several shortcomings. The issue of actual sound quality is difficult to address directly, as it is inherently subjective—it is difficult to deny, however, that in most cases abstract sound synthesis output is synthetic sounding. This can be desirable or not, depending on one's taste. On the other hand, it is worth noting that perhaps the most popular techniques employed by today's composers are based on modification and processing of sampled sound, indicating that the natural quality of acoustically produced sound is not easily abandoned. Indeed, many of the earlier refinements of abstract techniques such as FM were geared toward emulating acoustic instrument sounds [241, 317]. The deeper issue, however, is one of control. Some of the algorithms mentioned above, such as additive synthesis, require the specification of an inordinate amount of data. Others, such as FM synthesis, involve many fewer parameters, but it can be extremely difficult to determine rules for the choice and manipulation of parameters, especially in a complex configuration involving more than a few such oscillators. See [53, 52, 358] for a fuller discussion of the difficulties inherent in abstract synthesis methods.

Physical modeling synthesis, which has developed more recently, involves a physical description of the musical instrument as the starting point for algorithm design. For most musical instruments, this will be a coupled set of partial differential equations, describing, for example, the displacement of a string, membrane, bar, or plate, or the motion of the air in a tube, etc. The idea, then, is to solve the set of equations, invariably through a numerical approximation, to yield an output waveform, subject to some input excitation (such as glottal vibration, bow or blowing pressure, a hammer strike, etc.). The issues mentioned above, namely those of the synthetic character

and control of sounds, are rather neatly sidestepped in this case—there is a virtual copy of the musical instrument available to the algorithm designer or performer, embedded in the synthesis algorithm itself, which serves as a reference. For instance, simulating the plucking of a guitar string at a given location may be accomplished by sending an input signal to the appropriate location in computer memory, corresponding to an actual physical location on the string model; plucking it strongly involves sending a larger signal. The control parameters, for a physical modeling sound synthesis algorithm, are typically few in number, and physically and intuitively meaningful, as they relate to material properties, instrument geometry, and input forces and pressures.

The main drawback to using physical modeling algorithms is, and has been, their relatively large computational expense; in many cases, this amounts to hundreds if not thousands of arithmetic operations to be carried out per sample period, at a high audio sample rate (such as 44.1 kHz). In comparison, a bank of six FM oscillators will require probably at most 20 arithmetic operations/table lookups per sample period. For this reason, research into such methods has been slower to take root, even though the first such work on musical instruments began with Ruiz in the late 1960s and early 1970s [305], and digital speech synthesis based on physical models can be dated back even further, to the work of Kelly and Lochbaum [201]. On the other hand, computer power has grown enormously in the past decades, and presumably will continue to do so, thus efficiency (an obsession in the earlier days of digital sound synthesis) will become less and less of a concern.

1.2.1 Lumped mass–spring networks

The use of a lumped network, generally of mechanical elements such as masses and springs, as a musical sound synthesis construct, is an intuitively appealing one. It was proposed by Cadoz [66], and Cadoz, Luciani, and Florens in the late 1970s and early 1980s [67], and became the basis for the CORDIS and CORDIS-ANIMA synthesis environments [138, 68, 349]; as such, it constituted the first large-scale attempt at physical modeling sound synthesis. It is also the technique which is most similar to the direct simulation approaches which appear throughout the remainder of this book, though the emphasis here is entirely on fully distributed modeling, rather than lumped representations.

The framework is very simply described in terms of interactions among lumped masses, connected by springs and damping elements; when Newton's laws are employed to describe the inertial behavior of the masses, the dynamics of such a system may be described by a set of ordinary differential equations. Interaction may be introduced through so-called "conditional links," which can represent nonlinear contact forces. Time integration strategies, similar to those introduced in Chapter 3 in this book, operating at the audio sample rate (or sometimes above, in order to reduce frequency warping effects), are employed in order to generate sound output. The basic operation of this method will be described in more detail in Section 3.4.

A little imagination might lead one to guess that, with a large enough collection of interconnected masses, a distributed object such as a string, as shown in Figure 1.6(a), or membrane, as shown in Figure 1.6(b), may be modeled. Such configurations will be treated explicitly in Section 6.1.1 and Section 11.5, respectively. A rather large philosophical distinction between the CORDIS framework and that described here is that one can develop lumped networks which are, in a sense, only quasi-physical, in that they do not correspond to recognizable physical objects, though the physical underpinnings of Newton's laws remain. See Figure 1.6(c). Accurate simulation of complex distributed systems has not been a major concern of the designers of CORDIS; rather, the interest is in user issues such as the modularity of lumped network structures, and interaction through external control. In short, it is best to think of CORDIS as a system designed for artists and composers, rather than scientists—which is not a bad thing!

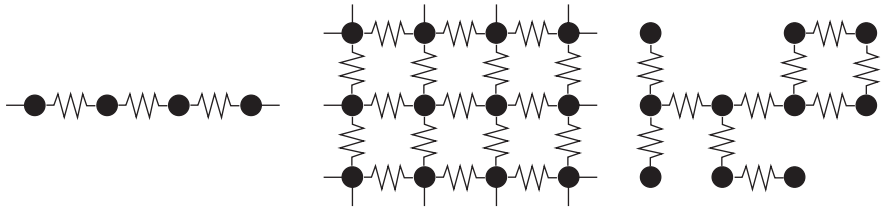


Figure 1.6 Lumped mass–spring networks: (a) in a linear configuration corresponding to a model of a lossless string; (b) in a 2D configuration corresponding to a model of a lossless membrane; and (c) an unstructured network, without a distributed interpretation.

1.2.2 Modal synthesis

A different approach, with a long history of use in physical modeling sound synthesis, is based on a frequency domain, or modal description of vibration of distributed objects. Modal synthesis [5, 4, 242], as it is called, is attractive, in that the complex dynamic behavior of a vibrating object may be decomposed into contributions from a set of modes (the spatial forms of which are eigenfunctions of the given problem at hand, and are dependent on boundary conditions). Each such mode oscillates at a single complex frequency. (For real-valued problems, these complex frequencies will occur in complex conjugate pairs, and the "mode" may be considered to be the pair of such eigenfunctions and frequencies.) Considering the particular significance of sinusoids in human audio perception, such a decomposition can lead to useful insights, especially in terms of sound synthesis. Modal synthesis forms the basis of the MOSAIC [242] and Modalys [113] sound synthesis software packages, and, along with CORDIS, was one of the first such comprehensive systems to make use of physical modeling principles. More recently, various researchers, primarily Rabenstein and Trautmann, have developed a related method, called the functional transformation method (FTM) [361], which uses modal techniques to derive point-to-point transfer functions. Sound synthesis applications of FTM are under development. Independently, Hélie and his associates at IRCAM have developed a formalism suitable for broad nonlinear generalizations of modal synthesis, based around the use of Volterra series approximations [303, 117]. Such methods include FTM as a special case. An interesting general viewpoint on the relationship between time and frequency domain methods is given by Rocchesso [292].

A physical model of a musical instrument, such as a vibrating string or membrane, may be described in terms of two sets of data: (1) the PDE system itself, including all information about material properties and geometry, and associated boundary conditions; and (2) excitation information, including initial conditions and/or an excitation function and location, and readout location(s). The basic modal synthesis strategy is as outlined in Figure 1.7. The first set of information is used, in an initial off-line step, to determine modal shapes and frequencies of vibration; this involves, essentially, the solution of an eigenvalue problem, and may be performed in a variety of ways. (In the functional transformation approach, this is referred to as the solution of a Sturm–Liouville problem [361].) This information must be stored, the modal shapes themselves in a so-called shape matrix. Then, the second set of information is employed: the initial conditions and/or excitation are expanded onto the set of modal functions (which under some conditions form an orthogonal set) through an inner product, giving a set of weighting coefficients. The weighted combination of modal functions then evolves, each at its own natural frequency. In order to obtain a sound output at a given time, the modal functions are projected (again through inner products) onto an observation state, which, in the simplest case, is of the form of a delta function at a given location on the object.

Though modal synthesis is often called a "frequency domain" method, this is not quite a correct description of its operation, and is worth clarifying. Temporal Fourier transforms are not

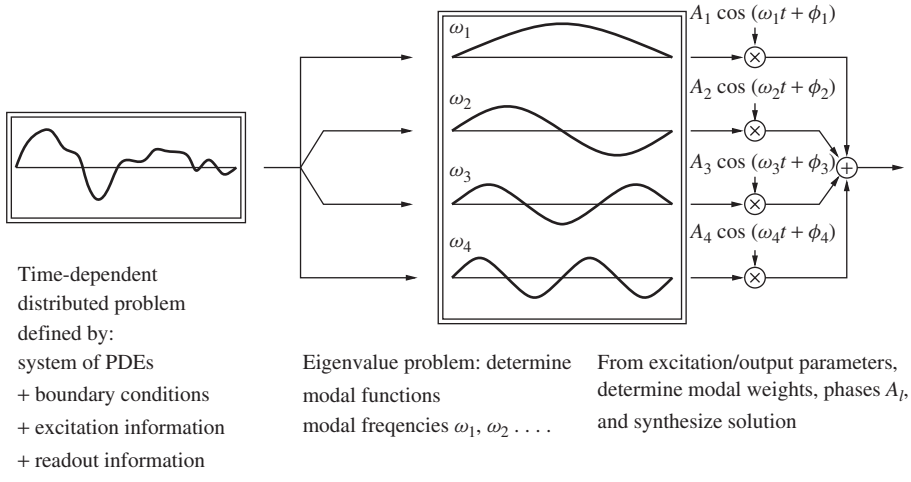


Figure 1.7 Modal synthesis. The behavior of a linear, distributed, time-dependent problem can be decomposed into contributions from various modes, each of which possesses a particular vibrating frequency. Sound output may be obtained through a precise recombination of such frequencies, depending on excitation and output parameters.

employed, and the output waveform is generated directly in the time domain. Essentially, each mode is described by a scalar second-order ordinary differential equation, and various time-integration techniques (some of which will be described in Chapter 3) may be employed to obtain a numerical solution. In short, it is better to think of modal synthesis not as a frequency domain method, but rather as a numerical method for a linear problem which has been diagonalized (to borrow a term from state space analysis [101]). As such, in contrast with a direct time domain approach, the state itself is not observable directly, except through reversal of the diagonalization process (i.e., the projection operation mentioned above). This lack of direct observability has a number of implications in terms of multiple channel output, time variation of excitation and readout locations, and, most importantly, memory usage. Modal synthesis continues to develop—for recent work, see, e.g., [51, 64, 380, 35, 416].

Modal synthesis techniques will crop up at various points in this book, in a general way toward the end of this chapter, and in more technical detail in Chapters 6 and 11.

1.2.3 Digital waveguides

Physical modeling sound synthesis is, to say the least, computationally very intensive. Compared to earlier methods, and especially FM synthesis, which requires only a handful of operations per clock cycle, physical modeling methods may need to make use of hundreds or thousands of such operations per sample period in order to create reasonably complex musical timbres. Physical modeling sound synthesis, 20 years ago, was a distinctly off-line activity.

In the mid 1980s, however, with the advent of digital waveguide methods [334] due to Julius Smith, all this changed. These algorithms, with their roots in digital filter design and scattering theory, and closely allied to wave digital filters [127], offered a convenient solution to the problem of computational expense for a certain class of musical instrument, in particular those whose vibrating parts can be modeled as 1D linear media described, to a first approximation, by the wave equation. Among these may be included many stringed instruments, as well as most woodwind and brass instruments. In essence, the idea is very simple: the motion of such a medium may be

modeled as two traveling non-interacting waves, and in the digital simulation this is dealt with elegantly by using two "directional" delay lines, which require no computer arithmetic at all! Digital waveguide techniques have formed the basis for at least one commercial synthesizer (the Yamaha VL1), and serve as modular components in many of the increasingly common software synthesis packages (such as Max/MSP [418], STK [92], and Csound [57]). Now, some 20 years on, they are considered the state of the art in physical modeling synthesis, and the basic design has been complemented by a great number of variations intended to deal with more realistic effects (discussed below), usually through more elaborate digital filtering blocks. Digital waveguides will not be covered in depth in this book, mainly because there already exists a large literature on this topic, including a comprehensive and perpetually growing monograph by Smith himself [334]. The relationship between digital waveguides and more standard time domain numerical methods has been addressed by various authors [333, 191, 41], and will be revisited in some detail in Section 6.2.11. A succinct overview is given in [330] and [290].

The path to the invention of digital waveguides is an interesting one, and is worth elaborating here. In approximately 1983 (or earlier, by some accounts), Karplus and Strong [194] developed an efficient algorithm for generating musical tones strongly resembling those of strings, which was almost immediately noticed and subsequently extended by Jaffe and Smith [179]. The Karplus–Strong structure is no more than a delay line, or wavetable, in a feedback configuration, in which data is recirculated; usually, the delay line is initialized with random numbers, and is terminated with a low-order digital filter, normally with a low-pass characteristic—see Figure 1.8. Tones produced in this way are spectrally rich, and exhibit a decay which is indeed characteristic of plucked string tones, due to the terminating filter. The pitch is determined by the delay-line length and the sample rate: for an N-sample delay line, as pictured in Figure 1.8, with an audio sample rate of f_s Hz, the pitch of the tone produced will be f_s/N, though fine-grained pitch tuning may be accomplished through interpolation, just as in the case of wavetable synthesis. In all, the only operations required in a computer implementation are the digital filter additions and multiplications, and the shifting of data in the delay line. The computational cost is on the order of that of a single oscillator, yet instead of producing a single frequency, Karplus–Strong yields an entire harmonic series. The Karplus–Strong plucked string synthesis algorithm is an abstract synthesis technique, in that in its original formulation, though the sounds produced resembled those of plucked strings, there was no immediate physical interpretation offered.

There are two important conceptual steps leading from the Karplus–Strong algorithm to a digital waveguide structure. The first is to associate a spatial position with the values in the wavetable—in other words, a wavetable has a given physical length. The other is to show that the values propagated in the delay lines behave as individual traveling wave solutions to the 1D wave equation; only their sum is a physical variable (such as displacement, pressure, etc.). See Figure 1.9. The link between the Karplus–Strong algorithm and digital waveguide synthesis, especially in the "single-delay-loop" form, is elaborated by Karjalainen et al. [193]. Excitation elements, such as bows, hammer interactions, reeds, etc., are usually modeled as lumped, and are connected to waveguides via scattering junctions, which are, essentially, power-conserving matrix operations (more will be said about scattering methods in the next section). The details of the scattering

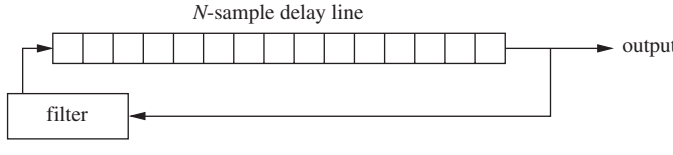


Figure 1.8 The Karplus–Strong plucked string synthesis algorithm. An N-sample delay line is initialized with random values, which are allowed to recirculate, while undergoing a filtering operation.

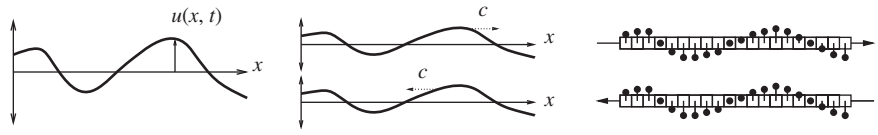


Figure 1.9 The solution to the 1D wave equation, (a), may be decomposed into a pair of traveling wave solutions, which move to the left and right at a constant speed c determined by the system under consideration, as shown in (b). This constant speed of propagation leads immediately to a discrete-time implementation employing delay lines, as shown in (c).

operation will be very briefly covered here in Section 3.3.3, Section 9.2.4, and Section 11.4. These were the two steps taken initially by Smith in work on bowed strings and reed instruments [327], though it is important to note the link with earlier work by McIntyre and Woodhouse [237], and McIntyre, Schumacher, and Woodhouse [236], which was also concerned with efficient synthesis algorithms for these same systems, though without an explicit use of delay-line structures.

Waveguide models have been successfully applied to a multitude of systems; several representative configurations are shown in Figure 1.10.

String vibration has seen a lot of interest, probably due to the relationship between waveguides and the Karplus–Strong algorithm. As shown in Figure 1.10(a), the basic picture is of a pair of waveguides separated by a scattering junction connecting to an excitation mechanism, such as a hammer or plectrum; at either end, the structure is terminated by digital filters which model boundary terminations, or potentially coupling to a resonator or other strings. The output is read from a point along the waveguide, through a sum of wave variables traveling in opposite directions. Early work was due to Smith [333] and others. In recent years, the Acoustics Group at the Helsinki University of Technology has systematically tackled a large variety of stringed instruments using digital waveguides, yielding sound synthesis of extremely high quality. Some of the target instruments have been standard instruments such as the harpsichord [377], guitar [218], and clavichord [375], but more exotic instruments, such as the Finnish kantele [117, 269], have been approached as well. There has also been a good deal of work on the extension of digital waveguides to deal with the special "tension modulation," or pitch glide nonlinearity in string vibration [378, 116, 359], a topic which will be taken up in great detail in Section 8.1. Some more recent related areas of activity have included banded waveguides [118, 119], which are designed to deal with systems with a high degree of inharmonicity, commuted synthesis techniques [331, 120], which allow for the interconnection of string models with harder-to-model resonators, through the introduction of sampled impulse responses, and the association of digital waveguide methods with underlying PDE models of strings [33, 34].

Woodwind and brass instruments are also well modeled by digital waveguides; a typical waveguide configuration is shown in Figure 1.10(b), where a digital waveguide is broken up by scattering junctions connected to models of (in the case of woodwind instruments) toneholes. At one end, the waveguide is connected to an excitation mechanism, such as a lip or reed model, and at the other end, output is taken after processing by a filter representing bell and radiation effects. Early work was carried out by Smith, for reed instruments [327], and for brass instruments by Cook [89]. Work on tonehole modeling has appeared [314, 112, 388], sometimes involving wave digital filter implementations [391], and efficient digital waveguide models for conical bores have also been developed [329, 370].

Vocal tract modeling using digital waveguides was first approached by Cook [88, 90]; see Figure 1.10(c). Here, due to the spatial variation of the cross-sectional area of the vocal tract, multiple waveguide segments, separated by scattering junctions, are necessary. The model is driven at one end by a glottal model, and output is taken from the other end after filtering to simulate

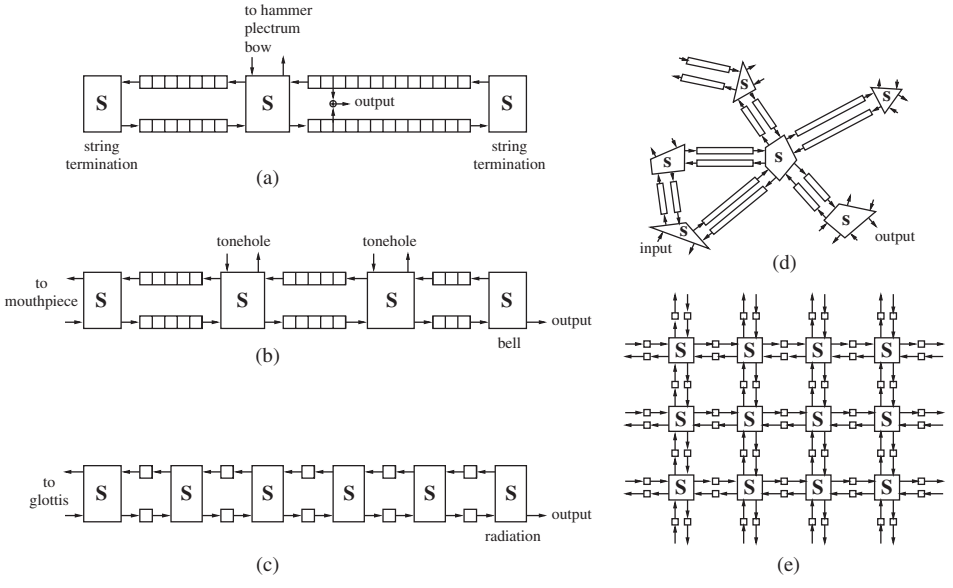


Figure 1.10 Typical digital waveguide configurations for musical sound synthesis. In all cases, boxes marked **S** represent scattering operations. (a) A simple waveguide string model, involving an excitation at a point along the string and terminating filters, and output read from a point along the string length; (b) a woodwind model, with scattering at tonehole junctions, input from a reed model at the left end and output read from the right end; (c) a similar vocal tract configuration, involving scattering at junctions between adjacent tube segments of differing cross-sectional areas; (d) an unstructured digital waveguide network, suitable for quasi-physical artificial reverberation; and (e) a regular waveguide mesh, modeling wave propagation in a 2D structure such as a membrane.

radiation effects. Such a model is reminiscent of the Kelly–Lochbaum speech synthesis model [201], which in fact predates the appearance of digital waveguides altogether, and can be calibrated using linear predictive techniques [280] and wave digital speech synthesis models [343]. The Kelly–Lochbaum model appears here in Section 9.2.4.

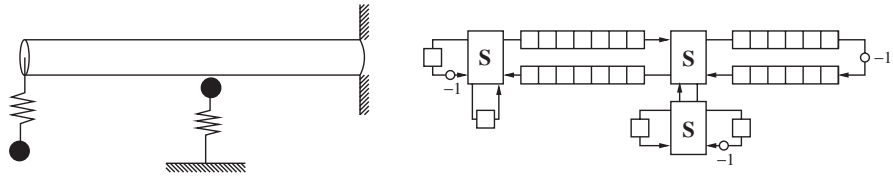
Networks of digital waveguides have also been used in a quasi-physical manner in order to effect artificial reverberation—in fact, this was the original application of the technique [326]. In this case, a collection of waveguides of varying impedances and delay lengths is used; such a network is shown in Figure 1.10(d). Such networks are passive, so that signal energy injected into the network from a dry source signal will produce an output whose amplitude will gradually attenuate, with frequency-dependent decay times governed by the delays and immittances of the various waveguides—some of the delay lengths can be interpreted as implementing "early reflections" [326]. Such networks provide a cheap and stable way of generating rich impulse responses. Generalizations of waveguide networks to feedback delay networks (FDNs) [293, 184] and circulant delay networks [295] have also been explored, also with an eye toward applications in digital reverberation. When a waveguide network is constructed in a regular arrangement, in two or three spatial dimensions, it is often referred to as a waveguide mesh [384–386, 41]—see Figure 1.10(e). In 2D, such structures may be used to model the behavior of membranes [216] or for vocal tract simulation [246], and in 3D, potentially for full-scale room acoustics simulation (i.e., for artificial reverberation), though real-time implementations of such techniques are probably decades away. Some work on the use of waveguide meshes for the calculation of room impulse responses has recently been done [28, 250]. The waveguide mesh is briefly covered here in Section 11.4.

1.2.4 Hybrid methods

Digital waveguides are but one example of a scattering-based numerical method [41], for which the underlying variables propagated are of wave type, which are reflected and transmitted throughout a network by power-conserving scattering junctions (which can be viewed, under some conditions, as orthogonal matrix transformations). Such methods have appeared in various guises across a wide range of (often non-musical) disciplines. The best known is the transmission line matrix method [83, 174], or TLM, which is popular in the field of electromagnetic field simulation, and dates back to the early 1970s [182], but multidimensional extensions of wave digital filters [127, 126] intended for numerical simulation have also been proposed [131, 41]. Most such designs are based on electrical circuit network models, and make use of scattering concepts borrowed from microwave filter design [29]; their earliest roots are in the work of Kron in the 1940s [211].

Scattering-based methods also play a role in standard areas of signal processing, such as inverse estimation [63], fast factorization and inversion of structured matrices [188], and linear prediction [280] for speech signals (leading directly to the Kelly–Lochbaum speech synthesis model, which is a direct antecedent to digital waveguide synthesis).

In the musical sound synthesis community, scattering methods, employing wave (sometimes called "W"), variables are sometimes viewed [54] in opposition to methods which employ physical (correspondingly called "K," for Kirchhoff) variables, such as lumped networks, and, as will be mentioned shortly, direct simulation techniques, which are employed in the vast majority of simulation applications in the mainstream world. In recent years, moves have been made toward modularizing physical modeling [376]; instead of simulating the behavior of a single musical object, such as a string or tube, the idea is to allow the user to interconnect various predefined objects in any way imaginable. In many respects, this is the same point of view as that of those working on lumped network models—this is reflected by the use of hybrid or "mixed" K–W methods, i.e., methods employing both scattering methods, such as wave digital filters and digital waveguides, and finite difference modules (typically lumped) [191, 190, 383]. See Figure 1.11. In some situations, particularly those involving the interconnection of physical "modules," representing various separate portions of a whole instrument, the wave formulation may be preferable, in that there is a clear means of dealing with the problem of non-computability, or delay-free loops—the concept of the reflection-free wave port, introduced by Fettweis long ago in the context of digital filter design [130], can be fruitfully employed in this case. The automatic generation of recursive structures, built around the use of wave digital filters, is a key component of such methods [268], and can be problematic when multiple nonlinearities are present, requiring specialized design procedures [309]. One result of this work has been a modular software system for physical modeling sound synthesis, incorporating elements of both types, called BlockCompiler [189]. More recently the scope of such methods has been hybridized even further through the incorporation of functional transformation (modal) methods into the same framework [270, 279].



(a) (b)

Figure 1.11 (a) A distributed system, such as a string, connected with various lumped elements, and (b) a corresponding discrete scattering network. Boxes marked **S** indicate scattering operations.

1.2.5 Direct numerical simulation

Digital waveguides and related scattering methods, as well as modal techniques, have undeniably become a very popular means of designing physical modeling sound synthesis algorithms. There are several reasons for this, but the main one is that such structures, built from delay lines, digital filters, and Fourier decompositions, fit naturally into the framework of digital signal processing, and form an extension of more abstract techniques from the pre-physical modeling synthesis era—note, for instance, the direct link between modal synthesis and additive synthesis, as well as that between digital waveguides and wavetable synthesis, via the Karplus–Strong algorithm. Such a body of techniques, with linear system theory at its heart, is home turf to the trained audio engineer. See Section 1.3.1 for more comments on the relationship between abstract methods and physical modeling sound synthesis.

For some time, however, a separate body of work in the simulation of musical instruments has grown; this work, more often than not, has been carried out by musical acousticians whose primary interest is not so much synthesis, but rather the pure study of the behavior of musical instruments, often with an eye toward comparison between a model equation and measured data, and possibly potential applications toward improved instrument design. The techniques used by such researchers are of a very different origin, and are couched in a distinct language; as will be seen throughout the rest of this book, however, there is no shortage of links to be made with more standard physical modeling sound synthesis techniques, provided one takes the time to "translate" between the sets of terminology! In this case, one speaks of time stepping and grid resolution; there is no reference to delays or digital filters and, sometimes, the frequency domain is not invoked at all, which is unheard of in the more standard physical modeling sound synthesis setting.

The most straightforward approach makes use of a *finite difference approximation* to a set of partial differential equations [342, 161, 284], which serves as a mathematical model of a musical instrument. (When applied to dynamic, or time-dependent systems, such techniques are sometimes referred to as "finite difference time domain" (FDTD) methods, a terminology which originated in numerical methods for the simulation of electromagnetics [351, 412, 352].) Such methods have a very long history in applied mathematics, which can be traced back at least as far as the work of Courant, Friedrichs, and Lewy in 1928 [95], especially as applied to the simulation of fluid dynamics [171] and electromagnetics [351]. Needless to say, the literature on finite difference methods is vast. As mentioned above, they have been applied for some time for sound synthesis purposes, though definitely without the success or widespread acceptance of methods such as

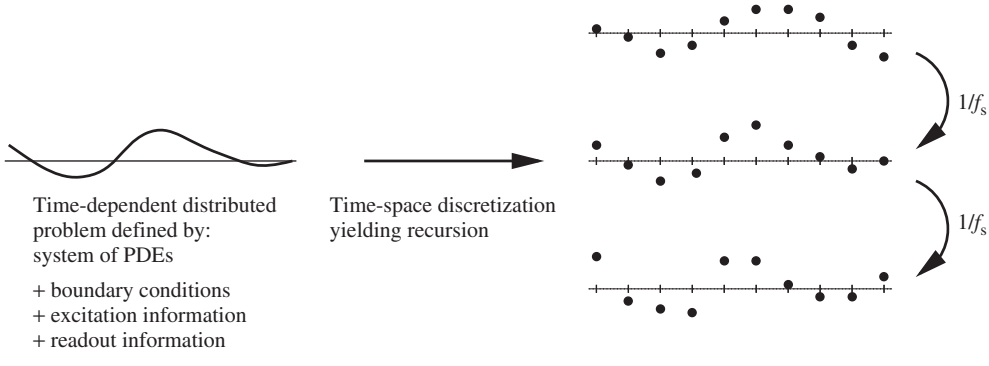


Figure 1.12 Direct simulation via finite differences. A distributed problem (at left) is discretized in time and space, yielding a recursion over a finite set of values (at right), to be updated with a given time step (usually corresponding to the inverse of the audio sample rate f_s).

digital waveguides, primarily because of computational cost—or, rather, preconceived notions about computational cost—relative to other methods.

The procedure, which is similar across all types of systems, is very simply described: the spatial domain of a continuous system, described by some model PDE, is restricted to a grid composed of a finite set of points (see Figure 1.12), at which values of a numerical solution are computed. Time is similarly discretized, and the numerical solution is advanced, through a recursion derived from the model PDE. Derivatives are approximated by differences between values at nearby grid points. The great advantage of finite difference methods, compared to other time domain techniques, is their generality and simplicity, and the wide range of systems to which they may be applied, including strongly nonlinear distributed systems; these cannot be readily approached using waveguides or modal synthesis, and by lumped models only in a very ad hoc manner. The primary disadvantage is that one must pay great attention to the problem of numerical instability—indeed numerical stability, and the means for ensuring it in sound synthesis algorithms, is one of the subjects that will be dealt with in depth in this book. Computational cost is an issue, but no more so than in any other synthesis method (with the exception of digital waveguides), and so cannot be viewed as a disadvantage of finite difference methods in particular.[3]

The most notable early finite difference sound synthesis work was concerned with string vibration, dating back to the work of Ruiz in 1969 [305] and others [169, 170, 19, 58]. Another very important contribution, in the context of vocal tract modeling and speech synthesis, was due to Portnoff [273]. The first truly sophisticated use of finite difference methods for musical sound synthesis was due to Chaigne in the case of plucked string instruments [74] and piano string vibration [75, 76]; this latter work has been expanded by others [232, 33], and extended considerably by Giordano through connection to a soundboard model [152, 154]. Finite difference methods have also been applied to various percussion instruments, including those based on vibrating membranes [139] (i.e., for drum heads), such as kettledrums [283], stiff vibrating bars such as those used in xylophones [77, 110] (i.e., for xylophones), and plates [79, 316]. Finite difference schemes for nonlinear musical systems, such as strings and plates, have been treated by this author [50, 42] and others [23, 22, 24]. Sophisticated difference scheme approximations to lumped nonlinearities in musical sound synthesis (particularly in the case of excitation mechanisms and contact problems) have been investigated [15, 282, 319] under the remit of the Sounding Object project [294]. A useful text, which employs finite difference methods (among other techniques) in the context of musical acoustics of wind instruments, is that of Kausel [196].

Finite difference methods, in the mainstream engineering world, are certainly the oldest method of designing a computer simulation. They are simply programmed, generally quite efficient, and there is an exhaustive literature on the subject. Best of all, in many cases they are sufficient for high-quality physical modeling sound synthesis. For the above reasons, they will form the core of this book. On the other hand, time domain simulation has undergone many developments, and some of these will be discussed in this book. Perhaps best known, particularly to mechanical engineers, is the finite element method (FEM) [121, 93] which also has long roots in simulation, but crystallized into its modern form some time in the 1960s. The theory behind FEM is somewhat different from finite differences, in that the deflection of a vibrating object is modeled in terms of so-called shape functions, rather than in terms of values at a given set of grid points. The biggest

[3] "Time domain" is often used in a slightly different sense than that intended here, at least in the musical acoustics/sound synthesis literature. The distinction goes back to the seminal treatment of McIntyre et al. [236], who arrived at a formulation suitable for a wide variety of musical instruments, where the instrument is considered to be made up of a lumped nonlinear excitation (such as a bow, reed, air jet, or a pair of lips) connected to a linear resonator. The resonator, assumed linear and time invariant, is completely characterized by its impulse response. As such, physical space disappears from the formulation entirely; the resonator is viewed in an input–output sense, and it is assumed that its impulse response is somehow available (it may be measured or calculated in a variety of ways). For time–space finite difference schemes, however, this is not the case. The spatial extent of the musical instrument is explicitly represented, and no impulse response is computed or employed.

benefit of FEMs is the ease with which relatively complex geometries may be modeled; this is of great interest for model validation in musical acoustics. In the end, however, the computational procedure is quite similar to that of finite difference schemes, involving a recursion of a finite set of values representing the state of the object. FEMs are briefly introduced on page 386. Various researchers, [20, 283] have applied finite element methods to problems in musical acoustics, though generally not for synthesis.

A number of other techniques have developed more recently, which could be used profitably for musical sound synthesis. Perhaps the most interesting are so-called spectral or pseudospectral methods [364, 141]—see page 388 for an overview. Spectral methods, which may be thought of, crudely speaking, as limiting cases of finite difference schemes, allow for computation with extreme accuracy, and, like finite difference methods, are well suited to problems in regular geometries. They have not, at the time of writing, found use in physical modeling applications, but could be a good match—indeed, modal synthesis is an example of a very simple Fourier-based spectral method.

For linear musical systems, and some distributed nonlinear systems, finite difference schemes (among other time domain methods) have a state space interpretation [187], which is often referred to, in the context of stability analysis, as the "matrix method" [342]. Matrix analysis/state space techniques will be discussed at various points in this book (see, e.g., Section 6.2.8). State space methods have seen some application in musical sound synthesis, though not through finite difference approximations [101].

1.3 Physical modeling: a larger view

This is a good point to step back and examine some global constraints on physical modeling sound synthesis, connections among the various existing methods and with earlier abstract techniques, and to address some philosophical questions about the utility of such methods.

1.3.1 Physical models as descended from abstract synthesis

Among the most interesting observations one can make about some (but not all) physical modeling methods is their relationship to abstract methods, which is somewhat deeper than it might appear to be. Abstract techniques, especially those described in Section 1.1, set the stage for many later developments, and determine some of the basic building blocks for synthesis, as well as the accompanying notation, which is derived from digital signal processing. This influence has had its advantages and disadvantages, as will be detailed below.

As mentioned earlier, digital waveguides, certainly the most successful physical modeling technique to date, can be thought of as a physical interpretation of wavetable synthesis in a feedback configuration. Even more important than the direct association between a lossless string and a wavetable was the recognition that systems with a low degree of inharmonicity could be efficiently modeled using a pair of delay lines terminated by lumped low-order digital filters—this effectively led the way to efficient synthesis algorithms for many 1D musical systems producing pitched tones. No such efficient techniques have been reported for similar systems in the mainstream literature, and it is clear that such efficiency gains were made possible only by association with abstract synthesis methods (and digital signal processing concepts in particular) and through an appreciation of the importance of human auditory perception to the resulting sound output. On the other hand, such lumped modeling of effects such as loss and inharmonicity is also a clear departure from physicality; this is also true of newer developments such as banded waveguides and commuted synthesis.

Similarly, modal synthesis may be viewed as a direct physical interpretation of additive synthesis; a modal interpretation (like that of any physical model) has the advantage of drastically

reducing the amount of control information which must be supplied. On the other hand, it is restrictive in the sense that, with minor exceptions, it may only be applied usefully to linear and time-invariant systems, which is a side effect of a point of view informed by Fourier decomposition.

As mentioned above, there is not always a direct link between abstract and physical modeling techniques. Lumped network models and direct simulation methods, unlike the other techniques mentioned above, have distinct origins in numerical solution techniques and not in digital signal processing. Those working on hybrid methods have gone a long way toward viewing such methods in terms of abstract synthesis concepts [279, 191]. Similarly, there is not a strong physical interpretation of abstract techniques such as FM (see, though, [403] for a different opinion) or granular synthesis.

1.3.2 Connections: direct simulation and other methods

Because direct simulation methods are, in fact, the subject of this book, it is worth saying a few words about the correspondence with various other physical modeling methods. Indeed, after some exposure to these methods, it becomes clear that all can be related to one another and to mainstream simulation methods.

Perhaps the closest relative of direct techniques is the lumped mass–spring network methodology [67]; in some ways, this is more general than direct simulation approaches for distributed systems, in that one could design a lumped network without a distributed counterpart—this could indeed be attractive to a composer. As a numerical method, however, it operates as a large ordinary differential equation solver, which puts it in line with various simulation techniques based on semi-discretization, such as FEMs. As mentioned in Section 1.2.1, distributed systems may be dealt with through large collections of lumped elements, and in this respect the technique differs considerably from purely distributed models based on the direct solution of PDEs, because it can be quite cumbersome to design more sophisticated numerical methods, and to deal with systems more complex than a simple linear string or membrane using a lumped approach. The main problem is the "local" nature of connections in such a network; in more modern simulation approaches (such as, for example, spectral methods [364]), approximations at a given point in a distributed system are rarely modeled using nearest-neighbor connections between grid variables. From the distributed point of view, network theory may be dispensed with entirely. Still, it is sometimes possible to view the integration of lumped network systems in terms of distributed finite difference schemes—see Section 6.1.1 and Section 11.5 for details.

It should also come as no surprise that digital waveguide methods may also be rewritten as finite difference schemes. It is interesting that although the exact discrete traveling wave solution to the 1D wave equation has been known in the mainstream simulation literature for some time (since the 1960s at least [8]), and is a direct descendant of the method of characteristics [146], the efficiency advantage was apparently not exploited to the same spectacular effect as in musical sound synthesis. (This is probably because the 1D wave equation is seen, in the mainstream world, as a model problem, and not of inherent practical interest.) Equivalences between finite differences and digital waveguide methods, in the 1D case and the multidimensional case of the waveguide mesh, have been established by various authors [384, 386, 334, 333, 41, 116, 313, 312], and, as mentioned earlier, those at work on scattering-based modular synthesis have incorporated ideas from finite difference schemes into their strategy [190, 191]. This correspondence will be revisited with regard to the 1D wave equation in Section 6.2.11 and the 2D wave equation in Section 11.4. It is worth warning the reader, at this early stage, that the efficiency advantage of the digital waveguide method with respect to an equivalent finite difference scheme does not carry over to the multidimensional case [333, 41].

Modal analysis and synthesis was in extensive use long before it debuted in musical sound synthesis applications, particularly in association with finite element analysis of vibrating structures—see [257] for an overview. In essence, a time-dependent problem, under some conditions, may be reduced to an eigenvalue problem, greatly simplifying analysis. It may also be viewed under the umbrella of more modern so-called spectral or pseudospectral methods [71], which predate modal synthesis by many years. Spectral methods essentially yield highly accurate numerical approximations through the use of various types of function approximations to the desired solution; many different varieties exist. If the solution is expressed in terms of trigonometric functions, the method is often referred to as a Fourier method—this is exactly modal synthesis in the current context. Other types of spectral methods, perhaps more appropriate for sound synthesis purposes (and in particular collocation methods), will be discussed beginning on page 388.

Modular or "hybrid" methods, though nearly always framed in terms of the language of signal processing, may also be seen as finite difference methods; the correspondence between lumped models and finite difference methods is direct, and that between wave digital filters and numerical integration formulas has been known for many years [132], and may be related directly to the even older concept of absolute- or A-stability [148, 99, 65]. The key feature of modularity, however, is new to this field, and is not something that has been explored in depth in the mainstream simulation community.

This is not the place to evaluate the relative merits of the various physical modeling synthesis methods; this will be performed exhaustively with regard to two useful model problems, the 1D and 2D wave equations, in Chapters 6 and 12, respectively. For the impatient reader, some concluding remarks on relative strengths and weaknesses of these methods appear in Chapter 14.

1.3.3 Complexity of musical systems

In the physical modeling sound synthesis literature (as well as that of the mainstream) it is commonplace to see claims of better performance of a certain numerical method over another. Performance may be measured in terms of the number of floating point operations required, or memory requirements, or, more characteristically, better accuracy for a fixed operation count. It is worth keeping in mind, however, that even though these claims are (sometimes) justified, for a given system, there are certain limits as to "how fast" or "how efficient" a simulation algorithm can be. These limits are governed by system complexity; one cannot expect to reduce an operation count for a simulation below that which is required for an adequate representation of the solution.

System complexity is, of course, very difficult to define. Most amenable to the analysis of complexity are linear and time-invariant (LTI) systems, which form a starting point for many models of musical instruments. Consider any lossless distributed LTI system (such as a string, bar, membrane, plate, or acoustic tube), freely vibrating at low amplitude due to some set of initial conditions, without any external excitation. Considering the continuous case, one is usually interested in reading an output $y(t)$ from a single location on the object. This solution can almost always[4] be written in the form

$$y(t) = \sum_{l=1}^{\infty} A_l \cos(2\pi f_l t + \phi_l) \tag{1.5}$$

which is exactly that of pure additive synthesis or modal synthesis; here, A_l and ϕ_l are determined by the initial conditions and constants which define the system, and the frequencies f_l are assumed non-negative, and to lie in an increasing order. Such a system has a countably infinite number

[4] The formula must be altered slightly if the frequencies are not all distinct.

of degrees of freedom; each oscillator at a given frequency f_l requires the specification of two numbers, A_l and ϕ_l.

Physical modeling algorithms generally produce sound output at a given sample rate, say f_s. This is true of all the methods discussed in the previous section. There is thus no hope of (and no need for) simulating frequency components[5] which lie above $f_s/2$. Thus, as a prelude to a discrete-time implementation, the representation (1.5) may be truncated to

$$y(t) = \sum_{l=1}^{N} A_l \cos(2\pi f_l t + \phi_l) \tag{1.6}$$

where only the N frequencies f_1 to f_N are less than $f_s/2$. Thus the number of degrees of freedom is now finite: $2N$. Even for a vaguely defined system such as this, from this information one may go slightly farther and calculate both the operation count and memory requirements, assuming a modal-type synthesis strategy. As described in Section 1.2.2, each frequency component in the expression (1.5) may be computed using a single two-pole digital oscillator, which requires two additions, one multiplication, and two memory locations, giving, thus, $2N$ additions and N multiplications per time step and a necessary $2N$ units of memory. Clearly, if fewer than N oscillators are employed, the resulting simulation will not be complete, and the use of more than N oscillators is superfluous. Not surprisingly, such a measure of complexity is not restricted to frequency domain methods only; in fact, *any* method (including direct simulation methods such as finite differences and FEMs) for computing the solution to such a system must require roughly the same amount of memory and number of operations; for time domain methods, complexity is intimately related to conditions for numerical stability. Much more will be said about this in Chapters 6 and 11, which deal with time domain and modal solutions for the wave equation.

There is, however, at least one very interesting exception to this rule. Consider the special case of a system for which the modal frequencies are multiples of a common frequency f_1, i.e., in (1.5), $f_l = lf_1$. In this case, (1.5) is a Fourier series representation of a periodic waveform, of period $T = 1/f_1$, or, in other words,

$$y(t) = y(t - T)$$

The waveform is thus completely characterized by a single period of duration T. In a discrete setting, it is obvious that it would be wasteful to employ separate oscillators for each of the components of $y(t)$; far better would be to simply store one period of the waveform in a table, and read through it at the appropriate rate, employing simple interpolation, at a cost of $O(1)$ operations per time step instead of $O(N)$. Though this example might seem trivial, it is worth keeping in mind that many pitched musical sounds are approximately of this form, especially those produced by musical instruments based on strings and acoustic tubes. The efficiency gain noted above is at the heart of the digital waveguide synthesis technique. Unfortunately, however, for musical sounds which do not generate harmonic spectra, there does not appear to be any such efficiency gain possible; this is the case, in particular, for 2D percussion instruments and moderately stiff strings and bars. Though extensions of digital waveguides do indeed exist in the multidimensional setting, in which case they are usually known as digital waveguide meshes, there is no efficiency gain

[5] In the nonlinear case, however, one might argue that the use of higher sampling rates is justifiable, due to the possibility of aliasing. On the other hand, in most physical systems, loss becomes extremely large at high frequencies, so a more sound, and certainly much more computationally efficient, approach is to introduce such losses into the model itself. Another argument for using an elevated sample rate, employed by many authors, is that numerical dispersion (leading to potentially audible distortion) may be reduced; this, however, is disastrous in terms of computational complexity, as the total operation count often scales with the square or cube of the sample rate. It is nearly always possible to design a scheme with much better dispersion characteristics, which still operates at a reasonable sample rate.

relative to modal techniques or standard time differencing methods; indeed, the computational cost of solution by any of these methods is roughly the same.[6]

For distributed nonlinear systems, such as strings and percussion instruments, it is difficult to even approach a definition of complexity—perhaps the only thing one can say is that for a given nonlinear system, which reduces to an LTI system at low vibration amplitudes (this is the usual case in most of physics), the complexity or required operation count and memory requirements for an algorithm simulating the nonlinear system will be at least that of the associated linear system. Efficiency gains through digital waveguide techniques are no longer possible, except under very restricted conditions—one of these, the string under a tension-modulated nonlinearity, will be introduced in Section 8.1.

One question that will not be approached in detail in this book is of model complexity in the perceptual sense. This is a very important issue, in that psychoacoustic criteria could lead to reductions in both the operation count and memory requirements of a synthesis algorithm, in much the same way as they have impacted on audio compression. For instance, the description of the complexity of an LTI system in terms of the number of modal frequencies up to the Nyquist frequency is mathematically sound, but for many musical systems (particularly in 2D), the modal frequencies become very closely spaced in the upper range of the audio spectrum. Taking into consideration the concepts of the critical band and frequency domain masking, it may not be necessary to render the totality of the components. Such psychoacoustic model reduction techniques have been used, with great success, in many efficient (though admittedly non-physical) artificial reverberation algorithms. The impact of psychoacoustics on physical models of musical instruments has seen some investigation recently, in the case of string inharmonicity [180], and also for impact sounds [11], and it would be useful to develop practical complexity-reducing principles and methods, which could be directly related to numerical techniques.

The main point of this section is to signal to the reader that for general systems, there is no physical modeling synthesis method which acts as a magic bullet—but there certainly is a "target" complexity to aim for. There is a minimum price to be paid for the proper simulation of any system. For a given system, the operation counts for modal, finite difference, and lumped network models are always nearly the same; in terms of memory requirements, modal synthesis methods can incur a much heavier cost than time domain methods, due to the storage of modal functions. One great misconception which has appeared often in the literature [53] is that time domain methods are wasteful, in the sense that the entire state of an object must be updated, even though one is interested, ultimately, in only a scalar output, generally from a single location on the virtual instrument. Thus point-to-point "black-box" models, based on a transfer function representation, are more efficient. But, as will be shown repeatedly throughout this book, the order of any transfer function description (and thus the memory requirements) will be roughly the same as the size of the physical state of the object in question.

1.3.4 Why?

The question most often asked by musicians and composers (and perhaps least often by engineers) about physical modeling sound synthesis is: Why? More precisely, why bother to simulate the behavior of an instrument which already exists? Surely the best that can be hoped for is an exact reproduction of the sound of an existing instrument. This is not an easy question to answer, but, nonetheless, various answers do exist.

[6] It is possible for certain systems such as the ideal membrane, under certain conditions, to extract groups of harmonic components from a highly inharmonic spectrum, and deal with them individually using waveguides [10, 43], leading to an efficiency gain, albeit a much more modest one than in the 1D case. Such techniques, unfortunately, are rather restrictive in that only extremely regular geometries and trivial boundary conditions may be dealt with.

The most common answer is almost certainly: Because it can be done. This is a very good answer from the point of view of the musical acoustician, whose interest may be to prove the validity of a model of a musical instrument, either by comparing simulation results (i.e., synthesis) to measured output, or by psychoacoustic comparison of recorded and model-synthesized audio output. Beyond the academic justification, there are boundless opportunities for improvement in musical instrument design using such techniques. From a commercial point of view, too, it would be extremely attractive to have a working sound synthesis algorithm to replace sampling synthesis, which relies on a large database of recorded fragments. (Consider, for example, the number of samples that would be required to completely represent the output of an acoustic piano, with 88 notes, with 60 dB decay times on the order of tens of seconds, struck over a range of velocities and pedal configurations.) On the other hand, such an answer will satisfy neither a composer of modern electroacoustic music in search of new sounds, nor a composer of acoustic orchestral music, who will find the entire idea somewhat artificial and pointless.

Another answer, closer in spirit to the philosophy of this author, is that physical modeling sound synthesis is far more than just a means of aping sounds produced by acoustic instruments, and it is much more than merely a framework for playing mix and match with components of existing acoustic instruments (the bowed flute, the flutter-tongued piano, etc.). Acoustically produced sound is definitely a conceptual point of departure for many composers of electroacoustic music, given the early body of work on rendering the output of abstract sound synthesis algorithms less synthetic sounding [241, 317], and, more importantly, the current preoccupation with real-time transformation of natural audio input. In this latter case, though, it might well be true (and one can never really guess these things) that a composer would jump at the chance to be freed from the confines of acoustically produced sound if indeed an alternative, possessing all the richness and interesting unpredictability of natural sound, yet somehow different, were available. Edgard Varèse said it best [392]:

> I dream of instruments obedient to my thought and which with their contribution of a whole new world of unsuspected sounds, will lend themselves to the exigencies of my inner rhythm.