CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS
OCTOBER 1991

Department of Music
Report No. STAN-M-73

PHYSICAL MODELING
CCRMA PAPERS PRESENTED AT THE
1991 INTERNATIONAL COMPUTER MUSIC CONFERENCE

Perry R. Cook, Suzanne Hirschman, Julius O. Smith

CCRMA
DEPARTMENT OF MUSIC
Stanford University
Stanford, California 94305

Viewpoints on the History of Digital Synthesis

Keynote Paper, ICMC-91

Julius Orion Smith III

Assoc. Prof. (Research), CCRMA, Music Dept., Stanford University, Stanford, CA 94305
Signal Processing Engineer, NeXT Inc., 900 Chesapeake Dr., Redwood City, CA 94063
email: jos@next.com or jos@ccrma.stanford.edu

Abstract - This essay sketches one view of the development of digital synthesis techniques from the days of Music V to the present. A taxonomy of digital synthesis techniques is proposed, and future projections are ventured. It is anticipated that synthesis in the future will be dominated by spectral and physical models.

Introduction

In a highly stimulating *Science* article, Max Mathews painted an exciting vision of "the digital computer as a musical instrument" (Mathews 1963). "Sound from Numbers," it pointed out, was a completely general way to synthesize sound because the bandwidth and dynamic range of hearing are bounded: "any perceivable sound can be so produced." The promise of computer music was that *the computer is capable of generating any sound that could ever come from a loudspeaker.* In *The Technology of Computer Music* (Mathews 1969), Max wrote

> "The two fundamental problems in sound synthesis are (1) the vast amount of data needed to specify a pressure function—hence the necessity of a very fast program—and (2) the need for a simple, powerful language in which to describe a complex sequence of sounds."

Problem (1) has been solved to a large extent by the march of technology. Digital processor performance has increased at the rate of 40-50% per year for the past fifteen years, and the trend shows no sign of weakening. At present, multiple voices of many synthesis techniques can be sustained in real time on a *single-chip*, general-purpose computer.

Problem (2) remains unsolved, and cannot, in principle, ever be completely solved. Nobody has the time to type in every sample of sound for a musical piece, (unless the piece is very short), nor does anyone know *how* to directly type the samples of natural-sounding waveforms. Therefore, sound samples must be *synthesized* from a much smaller set of numbers, or derived from *recordings* of natural phenomena, or both. In either case, a large number of samples must be specified or manipulated according a much smaller set of numbers. This implies a great sacrifice of generality. Fortunately, the vast majority of waveforms are either musically undesirable or musically equivalent to other waveforms, so we should in fact be able to give up most waveforms without giving up anything of musical value. The fundamental difficulty of digital synthesis becomes finding the smallest collection of synthesis techniques which span the space of musically useful sounds without redundancy. It is helpful when a technique is intuitively predictable. Predictability is good when analogies exist with well-known musical instruments.

Historical View of Digital Synthesis Development

In the Music V program (Mathews 1969), the concept of the *unit generator* was introduced. A unit generator is a fundamental building block for sound synthesis. It takes numeric parameters and/or audio signal(s) as input, and produces an output signal. The parameters are constrained to be constant over the duration of a note, or "event." The unit generators of Music V included an oscillator, filter, adder, multiplier, random number generator, and envelope generator. Thus, Music V unit-generators were basic signal processing/synthesis modules which could be combined to create interesting synthetic sounds. The techniques of *additive, subtractive,* and *nonlinear* synthesis (such as FM) could be implemented quite naturally with these elements. They were similar in function to the sound-generating/processing modules used in analog synthesizers at the time, such as voltage-controlled oscillators (VCO), amplifiers (VCA), and filters (VCF). Analog synthesis, in turn, utilized modules from earlier audio electronics.

Instrument definitions in Music V were written as unit-generator patches. An instrument invocation was essentially a subroutine call with arguments, called "P fields," which plugged into the unit-generator patch. A Music V "score" was essentially a time-stamped sequence of instrument calls. Since the instrument and score definitions in Music V completely specified the music computation in a procedural ASCII format, Music V gave us the musical counterpart of PostScript for 2D graphics (the standard marking language used first for laser printers and more recently for computer displays). Apparently, Music V was born at least three decades too soon to be accepted as the PostScript of the music world. Instead, we got MIDI.

In the years following the availability of Music V, a number of research centers with access to large, mainframe computers and D/A converters extended the music compiler in various ways. At CCRMA, for example, various Music V descendants, such as Mus10, introduced named variables, an Algol-style language for instrument definition, more built-in unit generators, piecewise-linear-functions for use as envelope parameters, and an instrument compiler. Descendants of Music V appeared at Princeton (Paul Lansky) , MIT (Barry Vercoe), and UCSD (Dick Moore) as well a few other places. Computer music blossomed in the seventies, with many software and hardware systems appearing. It would not be feasible to adequately survey parallel developments throughout the world in this short essay, so the remainder of this historical sketch will describe developments from CCRMA's point of view. The CCRMA story is quite applicable to other computer music labs that have invested significantly in digital synthesis hardware.

While the Music V software synthesis approach was very general and powerful—a unit generator could do anything permitted by the underlying programming language—computational costs on a general-purpose computer were dauntingly high. It was not uncommon to be spending hundreds of seconds of computer time for each second of sound produced. Student composers were forced to work between 3AM and 6AM to finish their pieces. Pressure mounted to move the primitive sound-generating algorithms into special-purpose hardware.

In October 1977, CCRMA took delivery of the Systems Concepts Digital Synthesizer (Roads 1989, pp. 333-349: Loy 1981), affectionately known around CCRMA as the "Samson Box," named after its designer Pete Samson. The Samson Box resembled a large, green refrigerator in the machine room at the Stanford AI lab, and it cost on the order of $100,000. In its hardware architecture, it provided 256 "generators" which were waveform oscillators with several modes and controls, complete with amplitude and frequency envelope support, and 128 "modifiers," each of which could be a second-order digital filter, random-number generator, amplitude-modulator, signum function,

allpass controller, and the like. Up to 64K words of delay memory with 32 ports could be used to construct reverberators, other delay effects, and large wavetables. Finally, four D/A converters came with "the Box" to supply four-channel sound output. These analog lines were fed to a 16-by-32 "audio switch" which routed sound to the various listening stations around the lab.

The Samson Box was a very elegant implementation of nearly all known, desirable, unit-generators in hardware form, and sound synthesis was sped up by three orders of magnitude in many cases. Additive, subtractive, and nonlinear FM synthesis and waveshaping were supported very nicely. A lot of music was produced by many composers on the Samson Box over more than a decade. It was a clear success.

The Samson Box, however, was not a panacea. There were very sizeable costs in moving from a general software synthesis environment to a constrained, special-purpose hardware synthesizer. Tens of man-years of effort were poured into software support: A large instrument library was written to manage the patching of hardware unit generators into instruments. Such patching had to be done indirectly via the synthesizer "command stream," that is, instrument procedures in SAIL executed to produce synthesizer commands which were saved in a file. Debugging tools were developed for disassembling, editing, and reassembling the synthesizer command stream. The command stream was difficult to work with, but it was unavoidable in serious debugging work. Software for managing the unique envelope hardware on the synthesizer was developed, requiring a lot of work. Filter support was complicated by the use of 20-bit fixed-point with non-saturating overflow and lack of rounding control. General wavetables were not supported in the oscillators. In general, it simply took a lot of work to make everything work right.

Another type of cost was incurred in moving over to the Samson Box. Research into new synthesis techniques slowed to a trickle. While editing an Algol-like description of a Mus10 instrument was easy, reconfiguring a complicated patch of Samson Box modules was much more difficult, and a lot of expertise was required to design, develop, and debug new instruments on the Box. Many new techniques such as waveguide synthesis and the Chant vocal synthesis method did not map easily onto the Samson Box architecture. Bowed strings based on a physical model could not be given a physically correct vibrato mechanism due to the way delay memory usage was constrained. Simple "Feedback FM" did not work because phase rather than frequency feedback is required. Most memorably, the simple interpolating delay-line, called Zdelay in Mus10, turned out to be incredibly difficult to implement on the Box, and an incredible amount of time was expended trying to do it. While the Samson Box was a paragon of design elegance and hardware excellence, it did not provide the proper foundation for future growth of digital synthesis technology. It was a computer-music production device more than a research tool.

Another problem with supporting special-purpose, computer-music hardware is that it can be obsolete by the time its controlling software is considered useable. Hardware is changing so quickly and software environments are getting so elaborate that we are almost forced to write software that will port easily from one hardware platform to the next. A major reason for the success of UNIX—"the ultimate computer virus"—is that it ports readily to new processors. We simply don't have time to recreate our software universe for new, special-purpose machines. A compromise that works well today is to write all software in a high-level language, but take the time to write hand-coded unit generators for each new processor that comes along. It is possible to implement all known techniques on top of a small number of unit generators which comprise more than 90% of the computational load. The NeXT Music Kit is built according to this model: it is an object-oriented system in which only the UnitGenerator and Orchestra classes must be rewritten for new hardware environments. As a result, the Music Kit can be ported to radically new

processor families in a very short time. Similarly, Bill Schottstaedt at CCRMA wrote "Common Lisp Music" entirely in Common Lisp; a single Lisp macro, "Run," which encloses the sample loop of an instrument definition, tests for the presence of DSP chips (either the one on the NeXT CPU board or five on an Ariel QuintProcessor board) and compiles and loads Lisp code to any DSPs present for acceleration of execution. To port Bill's system to another processor family requires only a Common Lisp implementation on the new computer.

Over the past several years, MIDI-compatible digital synthesizers have been taking over the world as the synthesis engines used by composers and musicians everywhere, including CCRMA. MIDI-compatible digital synthesizers provide far more synthesis power per dollar than we ever saw before. Unfortunately, the development of new techniques is now primarily in the hands of industry. We are no longer likely to read about new synthesis advances in the *Computer Music Journal*. Instead, we continue to hear opaque marketing terms such as "LA Synthesis" with no paper in the literature that explains the technique. On the positive side, musical instrument manufacturers are more likely to hire our students to push forward the degree of sophistication in their synthesizers.

The ease of using MIDI synthesizers has sapped momentum from synthesis-algorithm research by composers. Many composers who once tried out their own ideas by writing their own unit generators and instruments are now settling for synthesis of a MIDI command stream instead. In times such as these, John Chowning would not likely have discovered FM synthesis: a novel timbral effect obtained when an oscillator's vibrato is increased to audio frequencies.

Unlike software synthesis or the Samson Box, MIDI synthesizers require very little effort to control. The MIDI specification simplifies the performance-instrument interface down to that of a piano-roll plus some continuous controllers. In other words, MIDI was designed to mechanize performance on a keyboard-controlled synthesizer. It was not designed to serve as an interchange format for computer-music. MIDI instrument control is limited to selecting a patch, triggering it with one of 128 key numbers, and optionally wiggling one or more controllers to which the patch may or may not respond in a useful way. Rarely is it possible to know precisely what the patch is actually doing, or what effect the controllers will have on the sound, if any. The advantage of MIDI is easy control of preset synthesis techniques. The disadvantage is greatly reduced generality of control, and greatly limited synthesis specification. As Andy Moorer is fond of saying, "no adjustment necessary—in fact, no adjustment possible!"

"Direct digital synthesis makes it possible to compose directly with sound, rather than by having to assemble notes" (Mathews et al. 1974). Thus, part of the promise of computer music was to free composers of the note concept. The note concept become a more abstract *event* which could denote any kind of infusion of information into the sound-computing machinery at a given time. The sound generated by an event could be blended seamlessly with sound generated by surrounding events, obscuring any traditional concept of discrete notes. Hardware synthesizers and MIDI have sent us back to the "Note Age" by placing a wall between the instrument and the note that is played on it. MIDI works against this promise of computer music, however understandably.

MIDI synthesizers offer only a tiny subset of the synthesis techniques possible in software. It seems unlikely that future MIDI extensions will recapture the generality of software synthesis until instrument definitions are provided for, as in the original Music V. A straightforward way to accomplish this would be to define a set of *standard unit generators* for MIDI, and a syntax for patching them together and binding message-parameters and controllers to

unit-generator parameters. That way, in addition to the loosely described, standard timbres of General MIDI, such as "honky-tonk piano," there could also be a handful of simple yet powerful unit generators capable of the General MIDI timbres and much, much more. Adding instrument definitions to MIDI would not significantly increase the size of the typical MIDI file, and the *reproduceability* of a MIDI performance—the whole point of General MIDI—would actually be right. Of course, low-end synthesizers not capable of enough voices specified as unit-generator patches could continue to implement named timbres in their own way, as provided by General MIDI. It would also be helpful if General MIDI would define a few controller parameter names for each timbre, such as "brightness", "legato", and so on, so that greater expressiveness of performance is possible. In the more distant future, it would be ideal to have MIDI instrument definitions specifiable in a popular high-level language, as was done in Mus10.

Happily, software synthesis is making bit of a come-back, thanks to more powerful processors. The Motorola DSP56001 signal processing chip, for example, running at 25 MHz, can synthesize a simple guitar model at 44 kHz in real time. The Music Kit on the NeXT Computer uses the DSP56001 for synthesis. Because the DSP chip is a general-purpose processor with extensions for signal processing, it is much easier to program than most prior music-synthesis engines. While the controlling software for the Samson Box represents perhaps more than a hundred man-years of software effort, the DSP-based synthesis software on the NeXT Computer has absorbed only a few man-years so far, and to reach the same degree of completeness would require considerably less total effort.

For a given cost, DSP chips provide much more synthesis power than do general-purpose processor chips. However, current software development tools are significantly inferior for DSP chips. The DSP56001, for example, is still integrated as a "second computer" requiring its own assembly language, quirks and pitfalls, assembler, compiler, loader, debugger, and user-managed interprocessor communication. The DSP C compiler, for example, has simply not been useful, forcing *all* DSP code to be written in assembly language. Due presumably to the language barrier, separate development tools, and general difficulty of programming DSP chips, there does not appear to have been a significant resurgence of software synthesis research using DSP chips as an implementation vehicle. On the NeXT Computer, the "DSP" is serving mostly as a built-in synthesizer with a set of canned patches to choose from. Hardly anyone takes on programming the DSP; perhaps they feel life is too short to take up yet another computer.

General-purpose processors are not far behind digital signal processing chips in being suitable as software-synthesis engines. Sufficiently powerful chips exist today (at high cost). Already, the IRCAM/Ariel musical workstation uses two Intel i860 RISC processors to perform multivoiced synthesis in real time. A single i860 can out-perform a DSP56001, for example, at real-time music synthesis. (In fairness, one can define the problem so that the DSP chip is faster for that problem.) While DSP chips are less expensive by more than a factor of 10 in terms of dollars per computation per second, and while DSP chips possess valuable built-in conveniences due to being oriented specifically toward signal processing, use of a true general-purpose processor leverages off far superior compilers and development tools. Furthermore, RISC processors are adding hardware features needed for efficient graphics and sound processing. It is probably safe to say that software synthesis is on the threshold of returning forever to the form in which it started decades ago: written in high-level languages on fully general-purpose computers. It is now hard to justify the tens of man-years of software effort required to fully integrate and support special-purpose hardware for computer-music synthesis when one can buy mass-produced, general-purpose processors very cheaply, delivering tens and soon hundreds of megaflops per chip. Quality support of high-level languages and the ability to use immediately all previously existing software and development tools is an advantage not to be taken lightly.

The new RISC chips are not a panacea for synthesis either. One of the promises of RISC is that compiler technology and the processor architecture are developed jointly to make sure high-level language programming can make maximally efficient use of the hardware . This promise has not yet been fulfilled. Hand-coding of unit generators in assembly language still increases the performance by a large integer factor on today's RISC processors relative to what the compilers provide. Unfortunately, optimal assembly-language programming is more work on a RISC processor than it is on an elegantly designed DSP chip. Nevertheless, socio-economic factors indicate that general-purpose processors will enjoy many, many more man-years of software-support effort than any special-purpose processor is likely to see. Given that general-purpose CPU chips now offer very fast, (single-cycle, pipelined), floating-point, multiply-add units, it would be relatively easy to incorporate remaining features of today's signal processing chips—apparently much easier than providing a first-class software development environment for a new, special-purpose piece of hardware.

Taxonomy of Digital Synthesis Techniques

The historical sketch above focused more on digital synthesis *engines* than on *techniques* used to synthesize sound. The traditional categories of synthesis were *additive, subtractive,* and *nonlinear.* In this section, an attempt will be made to organize today's best-known synthesis techniques into the categories displayed in the following table:

Processed Recording	Spectral Model	Physical Model	Abstract Algorithm
Concrète	Wavetable F	Ruiz Strings	VCO,VCA,VCF
Wavetable T	Additive	Karplus-Strong Ext.	Some Music V
Sampling	Phase Vocoder	Waveguide	Original FM
Vector	PARSHL	Modal	Feedback FM
Granular	Sines+Noise (Serra)	Cordis-Anima	Waveshaping
Prin. Comp. T	Prin. Comp. F	Mosaic	Phase Distortion
Wavelet T	Chant		Karplus-Strong
	VOSIM		
	Risset FM Brass		
	Chowning FM Voice		
	Subtractive		
	LPC		
	Inverse FFT		
	Xenakis Line Clusters		

Some of these techniques will now be briefly discussed. Space limitations prevent detailed discussions and references for all techniques. The reader is referred to (Roads and Strawn 1985, Roads 1989) and recent issues of the *Computer Music Journal* for further reading and references.

Sampling synthesis can be considered a descendant of *musique concrète.* Jean-Claude Risset noted: "*Musique concrète* did open an infinite world of sounds for music, but the control and manipulation one could exert upon them was rudimentary with respect to the richness of the sounds, which favored an esthetics of collage" (Roads 1989: Risset 1985). It is interesting that the same criticism can be applied to sampling synthesizers three decades later. A recorded sound can be transformed into any other sound by a linear transformation (some linear, time-varying filter). A loss of generality is therefore not inherent in the sampling approach. To date, however, highly general transformations of recorded material have not yet been introduced into the synthesis repertoire, except in a few disconnected

research efforts. Derivative techniques such as *granular synthesis* are yielding significant new colors for the sonic palette. It can be argued also that spectral-modeling and wavelet-based synthesis are sampling methods with powerful transformation capabilities in the frequency domain.

"Wavetable T" denotes time-domain wavetable synthesis; this is the classic technique in which an arbitrary wave-shape is repeated to create a periodic sound. The original Music V oscillator supported this synthesis type, and to approximate a real (periodic) instrument tone, one could snip out a period of a recorded sound and load the table with it. The wavetable output is invariably multiplied by an amplitude-envelope. Of course, we quickly discovered that we also needed vibrato, and it often helped to add several wavetable units together with independent vibrato and/or slightly detuned fundamental frequencies in order to obtain a chorus-like effect. Panning between wavetables was a convenient way to get an evolving timbre. More than anything else, wavetable synthesis taught us that "periodic" sounds are generally poor sounds. Exact repetition is rarely musical. Electronic organs (the first digital one being the Allen organ) had to add tremolo, vibrato, and the Leslie (multipath delay and Doppler via spinning speakers and horns) as sonic post-processing in order to escape ear-fatiguing, periodic sounds.

"Wavetable F" denotes wavetable synthesis again, but as approached from the frequency domain. In this case, a desired harmonic spectrum is created—either a priori or from the results of a spectrum analysis—and an inverse Fourier series is used to create the period for the table. This approach, with interpolation among timbres, was used by Michael McNabb in the creation of Dreamsong (Roads 1989: McNabb 1981). It was used years earlier in psychoacoustics research by John Grey. An advantage of spectral-based wavetable synthesis is that *phase* is readily normalized, making *interpolation* between different wavetable timbres smoother and more predictable.

Vector synthesis is essentially multiple-wavetable synthesis with interpolation (and more recently, chaining of wavetables). This technique, with four-way interpolation, is used in the Korg Wavestation, for example. It points out a way that sampling synthesis can be made sensitive to an arbitrary number of performance control parameters: Given sufficiently many wavetables plus means for chaining, enveloping, and forming arbitrary linear combinations (interpolations) among them, it is possible to provide any number of expressive control parameters. Sampling synthesis need not be restricted to static table playback with looping and post-filtering. In principle, many wavetables may be necessary along each parameter dimension. Also, it is good to have a wavetable for every combination of parameters, implying that n parameters of control require 2^n wavetables, given two tables per parameter (i.e., no intermediate tables required). If the parameters are instead orthogonal, (e.g., formant bandwidths), n parameters can be implemented using interpolation among $2n$ wavetables. In any case, a lot of memory is likely to be used making a multidimensional timbre space using tables. Perhaps a physical model *is* worth a thousand wavetables.

Principal-components synthesis was apparently first tried in the time domain by Stapleton and Bass at Purdue University (Stapleton and Bass 1988). They computed an optimal set of *basis periods* for approximating a larger set of periodic musical tones via linear combinations. This would then be a valuable complement to vector synthesis since it can provide vectors which combine to span a wide variety of natural sounds. The frequency-domain form was laid out in (Plomp 1976) in the context of steady-state tone discrimination based on changes in harmonic amplitudes. In this domain, the principal components are fundamental *spectral shapes* which are mixed together to produce various spectra.

Additive synthesis historically models a spectrum as a set of discrete "lines" corresponding to sinusoids. The first analysis-driven additive synthesis for music appears to be Jean-Claude Risset's analysis and resynthesis of trumpet tones using Music V in 1964 (Roads 1989: Risset 1985). He also appears to have carried out the first piecewise-linear reduction of the harmonic amplitude envelopes, a technique that has become standard in additive synthesis based on oscillator banks. The phase vocoder has provided analysis support for additive synthesis for many years. The PARSHL program at CCRMA extended the phase vocoder to inharmonic partials, motivated initially by the piano, and Xavier Serra added filtered noise to the inharmonic sinusoidal model (Serra and Smith 1991). *Inverse-FFT* additive synthesis is implemented by writing any desired spectrum into an array and using the FFT algorithm to synthesize each frame of the time waveform (Chamberlin 1980); it undoubtedly has a big future in spectral-modeling synthesis since it is so general. The only tricky part is writing the spectrum for each frame in such a way that the frames splice together noiselessly in the time domain. Post-processing operations can be applied to the ideal desired spectrum to give optimal frame splicing in the time domain.

Linear Predictive Coding (LPC) has been used successfully for synthesis by Andy Moorer, Ken Steiglitz, and Paul Lansky, and earlier (at lower sampling rates) by speech researchers at Bell Labs. It is listed as a spectral modeling technique because there is evidence that the reason for the success of LPC in sound synthesis has more to do with the fact that the *upper spectral envelope* is estimated by the LPC algorithm than the fact that it has an interpretation as an estimator for the parameters of an all-pole model for the vocal tract. If this is so, direct spectral modeling should be able to do anything LPC can do, and more, and with greater flexibility. LPC has proven valuable for estimating loop-filter coefficients in waveguide models of strings and bores, so it could also be entered as a tool for sampling loop-filters in the "Physical Model" column. As a synthesis technique, it has the same transient-smearing problem that spectral modeling based on the short-time Fourier transform has. LPC can be viewed as one of many possible nonlinear smoothings of the short-time power spectrum, with good audio properties.

The *Chant* vocal synthesis technique (Mathews and Pierce 1989: Bennet and Rodet chapter) is listed a spectral modeling technique because it's a variation on *formant synthesis.* The Klatt speech synthesizer is another example. VOSIM is similar in concept, but trading sound quality for lower computational cost. Chant uses five exponentially decaying sinusoids tuned to the formant frequencies, prewindowed and repeated (overlapped) at the pitch frequency. Developing good Chant voices begins with a sung-vowel spectrum. The formants are measured, and Chant parameters are set to provide good approximations to these formants. Thus, the object of Chant is to *model the spectrum* as a regular sequence of harmonics multiplied by a formant envelope. LPC and *subtractive synthesis* also take this point view, except that the excitation can be white noise rather than a pulse train (i.e., any flat "excitation" spectrum will do). In more recent years, Chant has been extended to support *noise-modulated harmonics,* especially useful in the higher frequency regions. The problem is that real voices are not perfectly periodic, particularly when glottal closure is not complete, and higher-frequency harmonics look more like narrowband noise than spectral lines. Thus, a good spectral model should include provision for spectral lines that are somewhat "fuzzy." There are many ways to accomplish this "air brush" effect on the spectrum. Bill Schottstaedt, many years ago, added a little noise to the output of a modulating FM oscillator to achieve this effect on a formant group. Additive synthesis based on oscillators can accept a noise input in the same way, or any low-frequency amplitude- or phase-modulation can be used. Inverse-FFT synthesizers can simply write a broader "hill" into the spectrum instead of a discrete line (sampled window transform); the phase along the hill controls its shape and spread in the time domain. In the LPC world, it has been achieved, in effect, by *multipulse excitation*—that is, the "buzzy" impulse train is replaced by a small "tuft" of impulses, once per period. Multipulse LPC sounds more natural than single-pulse LPC.

The *Karplus-Strong algorithm* is listed as an abstract algorithm because it was conceived as a wavetable technique with a means of modifying the table each time through. It was later recognized as a special case of physical models for strings developed by McIntyre and Woodhouse, which led to its extensions for musical use. The "method of the rounded corner" was originally devised by McIntyre and Woodhouse to speed up model simulation on a digital computer, and the method can be simplified to the Karplus-Strong algorithm. What the Karplus-Strong algorithm showed, to everyone's surprise, was that the "corner rounding function" could be simplified to a multiply-free, two-point average with musically useful results. Waveguide synthesis is a set of extensions in the direction of accurate physical modeling while maintaining the computational simplicity of the method of the rounded corner. It most efficiently models one-dimensional waveguides, such as strings and bores, yet it can be coupled in a rigorous way to the more general physical models in Cordis-Anima and Mosaic (ACROE 1990, Smith 1991).

The Control Problem

Issues in digital-synthesis performance practice are too numerous even to try to summarize here. The reader is referred to the survey chapter by Gareth Loy in (Mathews and Pierce 1989, pp. 291-396). Suffice it to say that the musical control of digital musical instruments is still in its infancy despite some very good work on the problems. Perhaps the problem can be better appreciated by considering that instruments made of metal and wood are played by human hands; therefore, to transfer past excellence in musical performance to new digital instruments requires either providing an interface for a human performer—a growing trend—or providing a software control layer which "knows" how to perform a given score in a given musical context. The latter is a real artificial intelligence problem.

Projections for the Future

Abstract-algorithm synthesis seems destined to diminish in "mind-share" due to the lack of analysis support. It is very difficult to find a wide variety of musically pleasing sounds by exploring the parameters of some mathematical expression. Most sounds are simply uninteresting. The space of sounds we hear in everyday life is but a tiny pin-point in the space of all possible sounds. The most straightforward way to obtain interesting sounds is to draw on nature in some way. Both spectral-modeling and physical-modeling synthesis techniques support incorporation and/or modeling of natural sounds. In both cases the model is determined by some analysis procedure which is capable of computing optimal model parameters for approximating a particular given sound. The parameters are then used to provide desirable variations.

Obtaining better control of *sampling* synthesis will require more general sound transformations. To proceed toward this goal, transformations must be understood in terms of what we hear. The best way we know to understand a sonic transformation is to study its effect on the *short-time spectrum,* where the spectrum-analysis parameters are tuned to match the characteristics of hearing as closely as possible. Thus, it appears inevitable that sampling synthesis will migrate toward spectral modeling.

If abstract methods disappear and sampling synthesis is absorbed into spectral modeling, this leaves only two categories: *physical-modeling* and *spectral-modeling.* This boils all synthesis techniques down to those which model either the *source* or the *receiver* of the sound. If it is agreed that nature-referenced techniques are required to obtain natural sounds, it is difficult to ask for greater generality than that covered by these two categories. Spectral modeling is the more general of the two, since it is capable of constructing an arbitrary stimulus along the basilar membrane of the ear. However, physical models provide more compact algorithms for generating familiar classes of

sounds, such as strings and woodwinds. Also, they are generally more efficient at producing effects in the spectrum arising from attack articulations, long delays, pulsed noise, or nonlinearity in the physical instrument. It is also interesting to pause and consider how invariably performing musicians have interacted with resonators since the dawn of time in music. When a resonator has an impulse-response duration greater than that of a spectral frame (nominally the "integration time" of the ear), as happens with any string, then implementation of the resonator directly in the short-time spectrum becomes inconvenient. A resonator is a lot easier to implement as a recursion than as a super-thin formant in a short-time spectrum. Of course, as O. Larson says: "Anything is possible in software."

Spectral modeling has unsolved problems in the time domain: it is not yet known how to best modify a short-time Fourier analysis in the vicinity of an attack or other phase-sensitive transient. Phase is important during transients and not during steady-state intervals; a proper time-varying spectrum model should retain phase only where needed for accurate synthesis. The general question of *timbre perception* of non-stationary sounds becomes important. *Wavelet transforms* support more general signal building blocks which could conceivably help solve the transient modeling problem. Most activity with wavelet transforms to date has been confined to basic constant-Q spectrum analysis. Spectral models are also not yet terribly sophisticated; sinusoids and filtered noise with piecewise-linear envelopes are a good start, but surely there are other good primitives. Finally, tools for spectral modeling and transformation, such as spectral envelope and formant estimators, peak-finders, pitch-detectors, polyphonic peak associators, time compression/expansion transforms, and so on and on, should be developed in a more general-purpose and sharable way.

The use of granular synthesis to create swarms of "grains" of sound using wavelet kernels of some kind (Roads 1989: Roads 1978) appears promising as a basis for a future *statistical* time-domain modeling technique. It would be very interesting if a kind of wavelet transform could be developed which would determine the optimum grain waveform, and provide the counterpart of a short-time power spectral density which would indicate the statistical frequency of each grain scale at a given time. Such a tool could provide a compact, transformable description of sounds such as rain, breaking glass, and the crushing of rocks, to name a few.

References

Chamberlin 1980. *Musical Applications of Microprocessors.* New Jersey: Hayden Book Co., Inc.

ACROE 1990. Proceedings of the Colloquium on Physical Modeling, Grenoble, France.

Mathews, M. V. 1963. "The Digital Computer as a Musical Instrument." *Science* 142(11):553-557.

Mathews, M. V., et al. 1969. *The Technology of Computer Music.* Cambridge, Mass.: MIT Press.

Mathews M. V., F. R. Moore, and J.-C. Risset 1974. "Computers and Future Music." *Science* 183(1):263-268.

Mathews M. V., and J. R. Pierce, eds. 1989. *Current Directions in Computer Music Research.* Cambridge, Mass.: MIT Press.

Moore F. R. 1990. *Elements of Computer Music.* Englewood Cliffs, New Jersey: Prentice Hall.

Plomp, R. 1976. *Aspects of Tone Sensation.* New York: Academic Press.

Roads, C., and J. Strawn, eds. 1985. *Foundations of Computer Music.* Cambridge, Mass.: MIT Press.

Roads, C., ed.. 1989. *The Music Machine.* Cambridge, Mass.: MIT Press.

Serra X. J., and J. O. Smith 1991. "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic plus Stochastic Decomposition." *Computer Music Journal* 14(4):12-24.

Smith, J.O. 1991. "Waveguide Simulation of Non-Cylindrical Acoustic Tubes." Elsewhere in this proceedings.

Stapleton, J. C., and S. C. Bass 1988. "Synthesis of Musical Tones Based on the Karhunen-Loève Transform." IEEE Tr. ASSP, 36(3):305-319.

Waveguide Simulation of Non-Cylindrical Acoustic Tubes

Julius O. Smith III

Assoc. Prof. (Research), CCRMA, Music Dept., Stanford University, Stanford, CA 94305
Signal Processing Engineer, NeXT Inc., 900 Chesapeake Dr., Redwood City, CA 94063
email: jos@next.com or jos@ccrma.stanford.edu

Introduction

The digital waveguide modeling technique provides an efficient class of synthesis structures for vibrating strings, woodwind air columns, and other one-dimensional waveguides (Hirschman 1991, Smith 1987, Smith 1990). A basic waveguide building block is simply a pair of delay lines: One delay line provides propagation delay for a wave traveling in one direction along a string or tube, while the other delay line provides propagation in the other direction. By adjoining different tube sections via scattering junctions, adding digital filters at strategic points, and providing nonlinear junctions which excite oscillation, models of whole musical instruments, especially members of the string, woodwind, and brass families, can be built. Even the singing human voice has been convincingly simulated using this approach (Cook 1990).

There are several ways the waveguide approach may be extended. Digital filters at selected points in the waveguide can be computed to approximate distributed losses and dispersion in real strings and woodwind bores to an arbitrarily accurate extent using the techniques of linear prediction and system identification (Smith 1983). It is also straightforward to couple a waveguide model to a more conventional simulation framework, such as one which explicitly simulates individual modes of vibration using second-order resonators (Adrien and Morrison 1990, Smith 1987). Since the model has a physical interpretation, nonlinear and time-varying extensions are straightforward (Hirschberg et al. 1991, Smith 1987, Sullivan 1990).

It is possible to extend beyond one-dimensional waveguides to two dimensions (for membrane modeling) or three dimensions (for waves in solids or open air). In these extensions, a waveguide mesh can attempt to fill the vibrating area or volume with sufficient density, much like using a "wire-frame diagram" in computer graphics. Alternatively, the basic principle of commuting losses and dispersion to the medium boundaries can be applied directly to a simulation of the two or three dimensional medium in order to eliminate all multiplies from the interior medium simulation (Smith 1990).

The simplest waveguide models involve only linear, time-invariant, one-dimensional acoustic systems. To a high degree of approximation, a *horn* may be regarded as one-dimensional waveguide, provided that the flare of the horn is small relative to the smallest wavelength of propagation (Beranek 1986, Morse 1976). Morse states: "The analysis of wave motion in a horn is a very complicated matter, so complicated that it has been done in a rigorous manner only for conical and hyperbolic horns." In this context, hyperbolic horns are those of the form $y = y_0 [\cosh(x/h) + T\sinh(x/h)]$, where x is distance along the axis of the horn, y is the horn radius, h is the "scale factor" controlling flare, and T is the "shape factor" which is important near the throat of the horn. The hyperbolic horn family is also known as the *Salmon horn* family. Note that *catenoidal, exponential, conical,* and *cylindrical* horns are all special or limiting cases of the hyperbolic horn. (The hyperboloidal horn, however, is not!)

Waves in a hyperbolic horn are "one-parameter waves," meaning that a coherent wavefront spreads out uniformly along the horn, and a "surface of constant phase" may be defined whose tangent plane is normal to the horn axis. For cylindrical tubes, the surfaces of constant phase are planar, while for conical tubes, they are spherical (Morse 1976). The key property of the horn is that a wave propagates from one end to the other with no "back-scattering" of the wave.

Rather, it is smoothly "guided" from one end to the other. In other words, a horn is a waveguide. The absence of back-scattering means that the entire propagation path may be simulated using a pure delay line. Any losses, dispersion, or amplitude change due to horn radius variation can be implemented where the wave exits the delay line to interact with other components. This is valid because linear, time-invariant systems *commute*.

Non-conical horns are ***dispersive***, i.e., sound speed is not the same at all frequencies (Morse 1976). However, dispersion can be "lumped" into one or more allpass filters as is done for stiff string simulation (Smith 1983).

While we could proceed directly to the waveguide formulation of the entire Salmon horn family, it is simpler to consider first the conical case. All smooth horns reduce to the conical case over sufficiently short distances, and the use of many conical sections is always an alternative to a higher order waveguide model.

Piecewise Conical Acoustic Tubes

In a paper especially relevant to musical acoustics (Caussé et al. 1984), truncated cones were used in the modeling of horns and brass instrument bells. They report that the use of conical sections "leads to faster numerical convergence" relative to cylindrical sections. Each truncated cone is represented by a so-called "transmission matrix" (Pierce 1989).*

The *cone* is a one-dimensional waveguide which propagates a circular section of a *spherical wave* in place of the plane wave which traverses a cylindrical acoustic tube (Ayers et al. 1985, Pierce 1989). The wave equation in the spherically symmetric case is given by

$$c^2 p_x'' = \ddot{p}_x$$

where

$$c \triangleq \text{sound speed} \qquad p_x \triangleq xp(t,x)$$

$$\dot{p}_x \triangleq \frac{\partial}{\partial t} p_x(t,x) \qquad p_x' \triangleq \frac{\partial}{\partial x} p_x(t,x)$$

and $p(t,x)$ is the *pressure* at time t and radial position x along the cone.

It can be seen that the wave equation in a cone is identical to the wave equation in a cylinder, except that p is replaced by xp. Thus, the solution is a superposition of left- and right-going traveling wave components, scaled by $1/x$:

$$p(t,x) = \frac{f\left(t - \frac{x}{c}\right)}{x} + \frac{g\left(t + \frac{x}{c}\right)}{x}$$

where $f(\cdot)$ and $g(\cdot)$ are arbitrary continuous functions.

---

* The transmission matrix is a two-by-two matrix of frequency-dependent elements which when multiplied times the two-vector containing pressure and velocity phasors (complex amplitudes at a single frequency) at the output of the tube segment, produces the pressure and velocity phasors at the input of the segment. Transmission matrices are most often used in the musical acoustics literature to simulate acoustic tubes; for example, Keefe describes clarinet tone-holes this way (Keefe 1982). The difference between the transmission-matrix formulation and the waveguide formulation lies in the choice of acoustic state variables: The transmission-matrix formulation uses pressure and velocity to define the acoustic state at any point along the tube, while the waveguide formulation uses left- and right-going traveling-wave components (either pressure or velocity or some combination of the two) (Smith 1987). Since a transmission matrix may be converted to a corresponding scattering matrix by a two-by-two linear transformation, it is straightforward to rigorously incorporate simulation parameters from the acoustics literature in a digital waveguide model.

Digital Simulation

The discrete-time simulation of the above solution is obtained by simply *sampling* the traveling-wave amplitude at intervals of T seconds, which implies a *spatial* sampling interval of $X \triangleq cT$ meters. Define

$$p^+(n) \triangleq f(nT - x_0/c) \qquad p^-(n) \triangleq g(nT + x_0/c)$$

where x_0 is arbitrarily chosen as the position along the cone closest to the tip. (There cannot be a sample at the tip itself, for a singularity exists there.) Then a section of the ideal cone can be simulated as shown in Figure 1 (where pressure outputs are shown for $x = x_0$ and $x = x_0 + 3X$). A particle-velocity output is formed by dividing the traveling pressure waves by the characteristic impedance. Since the characteristic impedance is now a function of frequency and propagation direction (as can be quickly derived from the Laplace transform of the momentum-conservation equation in a cone), a digital filter will replace what was a real number for cylindrical tubes.

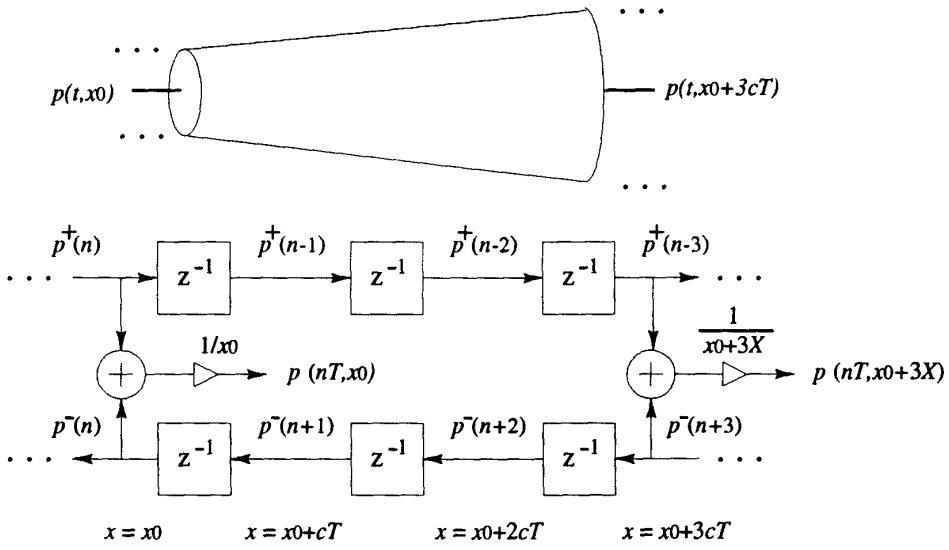


Figure 1. Digital simulation of the ideal, lossless, conical waveguide with observation points at $x = x_0$ and $x = x_0 + 3X = x_0 + 3cT$. The symbol "z^{-1}" denotes a one-sample delay.

Generalized Scattering Coefficients

The generalization of scattering coefficients at a multi-tube intersection as derived in (Smith 1987) results in junction pressure being given by

$$p_J = \left(G_J + \sum_{i=1}^{N} G_i^-\right)^{-1} \sum_{i=1}^{N} (G_i^+ + G_i^-)\, p_i^+$$

where G_i^+ is the complex, frequency-dependent, incoming, acoustic admittance of the ith branch at the junction, G_i^- is the corresponding outgoing acoustic admittance, p_i^+ is the incoming traveling pressure wave phasor in branch i, $p_i^- = p_J - p_i^+$ is the outgoing wave, and G_J is the admittance of a load at the junction, such as a coupling to another simulation. For generality, the formula is given as it appears in the multivariable case.

References

Due to space limitations, only the most recent and/or fundamental references appear below.

J. M. Adrien and J. Morrison, "Mosaïc: A Modular Program for Synthesis by Modal Superposition," *Proc. Colloquium on Physical Modeling,*, Grenoble, 1990.

J. Agullo, A. Barjau, and J. Martinez, "Alternatives to the Impulse Response $h(t)$ to describe the Acoustical Behavior of Conical Ducts," JASA-84#5, pp. 1606–1627, Nov. 1988.

R. D. Ayers, L. J. Eliason, and D. Mahgerefteh, "The Conical Bore in Musical Acoustics," *Am. J. Physics.*, vol. 53, no. 6, pp. 528–537, June 1985.

A. H. Benade, "Equivalent Circuits for Conical Waveguides," JASA-83#5, pp. 1764–1769, May 1988.

L. L. Beranek, *Acoustics,* Amer. Inst. Physics, (516)349-7800 x 481, 1986. (1st ed. 1954.)

R. Caussé, J. Kergomard, and X. Lurton, "Input impedance of Brass Musical Instruments—Comparison between Experiment and Numerical Models," JASA-75#1, pp. 241–254, Jan. 1984.

P. R. Cook, "Identification of Control Parameters in an Articulatory Vocal Tract Model, with Applications to the Synthesis of Singing," Ph.D. Dissertation, Elec. Eng. Dept., Stanford University, Dec. 1990.

J. Gilbert, J. Kergomard, and J. D. Polack, "On the Reflection Functions Associated with Discontinuities in Conical Bores," JASA-87#4, pp. 1773–1780, April. 1990.

A. Hirschberg, J. Gilbert, A. P. J. Wijnands, and A. J. M. Houtsma, "Non-Linear Behavior of Single-Reed Woodwind Musical Instruments," Nederlands Akoestisch Genootschap J., nr. 107, pp. 31–43, March 1991.

S. Hirschman, "Digital Waveguide Modelling and Simulation of Reed Woodwind Instruments," Eng. Dissertation, Elec. Eng. Dept., Stanford University, May 1991.

D. H. Keefe, "Theory of the Single Woodwind Tone Hole," "Experiments on the Single Woodwind Tone Hole," JASA-72#3, pp. 676–699, Sep. 1982.

P. M. Morse, *Vibration and Sound,* Amer. Inst. Physics, (516)349-7800 x 481, 1976 (1st ed. 1936, 2nd ed. 1948).

P. M. Morse and U. Ingard, *Theoretical Acoustics,* McGraw-Hill, New York, 1968.

A. D. Pierce, *Acoustics,* Amer. Inst. Physics, (516)349-7800 x 481, 1989.

J. O. Smith, "Techniques for Digital Filter Design and System Identification with Application to the Violin," Ph.D. Dissertation, Elec. Eng. Dept., Stanford University, June 1983.

J. O. Smith, "Music Applications of Digital Waveguides," (A compendium containing four related papers and presentations.) CCRMA Tech. Rep. STAN-M-67, Stanford University, 1987, (415)723-4971.

J. O. Smith, "Efficient Yet Accurate Models for Strings and Air Columns using Sparse Lumping of Distributed Losses and Dispersion," *Proc. Colloquium on Physical Modeling,*, Grenoble, 1990. CCRMA Tech. Rep. STAN-M-67, Stanford University, (415)723-4971.

J. O. Smith, "Waveguide Simulation of Non-Cylindrical Acoustic Tubes," CCRMA Tech. Rep. STAN-M-?, Stanford University, (415)723-4971. A longer version of this paper.

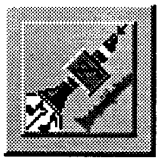
C. R. Sullivan, "Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback," *Computer Music J.*, vol. 14, no. 3, pp. 26–37, Fall 1990.

DIGITAL WAVEGUIDE MODELLING OF REED WOODWINDS: AN INTERACTIVE DEVELOPMENT

Suzanne E. Hirschman, Perry R. Cook, Julius O. Smith (szh, prc @ccrma.stanford.edu, julius_smith@next.com)
Center for Computer Research in Music and Acoustics (CCRMA), Stanford University

ABSTRACT

The digital waveguide filter has proven to be extremely effective for modelling wave propagation in musical instruments. It can be easily connected to independent models of reed mechanisms and bell shaping functions. In addition, the waveguide can be adapted to simulate arbitrary bore shapes and tone hole lattices. The WGF was used as the basis for an interactive modelling environment for a generic cylindrical cane reed-type instrument on the NeXT computer. The user has direct access to parameters defining the reed, the bell, the waveguide length, and the attack envelope. The program offers both reed lookup table and dynamic reed models for reed implementation, and a number of filter types for bell implementation. In addition, a single tone hole, implemented as a two-port scattering junction with "diameter" and placement defined by the user, is included. The ouptut of the simulation is a NeXT soundfile, which can be played and analyzed using a variety of tools on the NeXT computer. In addition, a spectrum panel allows the user to generate and analyze the impulse response of the instrument. Studies performed with this simulation include: implementation of a register hole; influence of attack on steady-state mode excitation; single vs. double reed behavior; influence of bell and cutoff frequency on tone; and influence of reed resonance on tone and stability.



INTRODUCTION

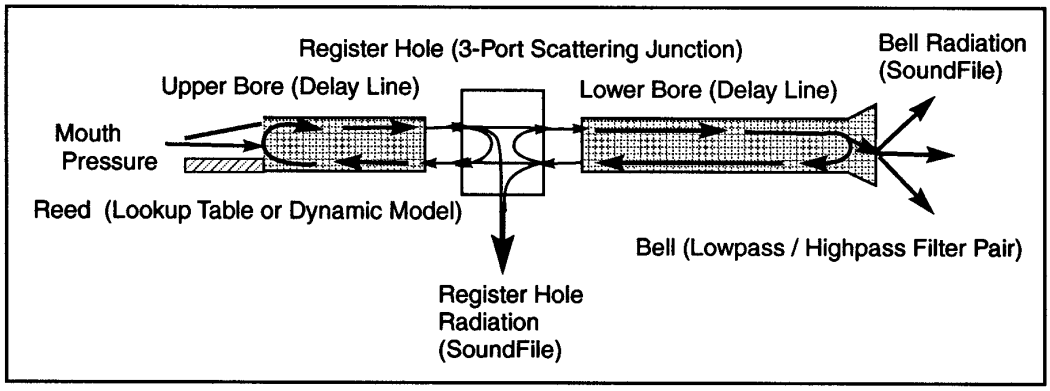
As the field of digitally produced sound, and its associated computer technology develops, so does the potential for creating realistic real-time computer simulations that model acoustic instruments. Simulation has long been a tool for acoustical analysis; in the field of physical modelling, it is rapidly becoming a viable and exciting resource for musical performance.

The primary source of musical sound in an acoustic instrument is the propagation of waves through the instrument medium and the subsequent pressure radiation to the outside air. A good model of the instrument must therefore duplicate the wave behavior, as well as the nonlinear relationships between the medium and its excitation mechanism. Julius Smith has demonstrated that the waveguide digital filter, an offshoot of the normalized ladder/lattice filter, is an ideal structure for this task [1]. The waveguide filter is

essentially a two-way delay line interspersed with partially reflective "scattering" junctions which operate on the oncoming wavefronts. Such a filter implements the wave propagation equations exactly. Just as a tonehole or bore diameter change will create an acoustic barrier within an instrument, allowing only part of the wave to penetrate while reflecting the rest, so can the associated changes in impedance be used to calculate the filter reflection coefficients which will result in similar scattering of the propagated, simulated wavefronts. The waveguide section is extremely modular, and can easily be connected to models of excitation and terminating impedance.

This paper, a summary of a more complete report available in reference [2], discusses the implementation of Smith's clarinet model [3] in an interactive environment on the NeXT computer. The clarinet model in its basic form is particularly elegant and efficient, as its cylindrical bore reduces much of the waveguide to a simple delay line. Inelaborate as it is, it provides a powerful demonstration of the principles of wave propagation, and of the essential coupling between reed and air column, as well as being a potentially rich source of musical sound. The simulation, dubbed the "ClariNeXT Workbench", was designed as an evolutionary, interactive workbench both for exploring acoustical behavior and for evaluating the musical potential, with respect to a real-time digital instrument, of various refinements to the basic model.

THE CLARINEXT MODELLING WORKBENCH



The clarinet simulation was developed in Objective C on the NeXT computer, using the NeXT Interface Builder application to provide the graphical user interface. It was designed to run with the 22050 Hz sampling rate used by the NeXT sound processing software. The simulation contained the following component models:

Cylindrical Bore Sections: modelled by pure delay lines, sized according to the pitch designated by the user;

Register hole: modelled as a 3-port scattering junction at a user-defined location along the bore. A modified 2-port scattering junction was also provided to allow a crude source of bore perturbation;

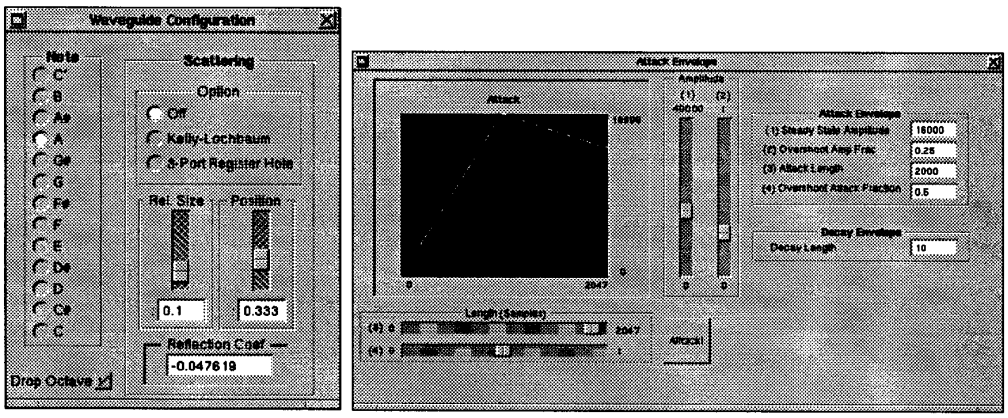
Reed: modelled as a time-varying reflection coefficient based on either 1) Smith's static reed look-up table [3] or 2) a 2nd order dynamic model with user-definable damping ratio and resonant frequency. Additional enhancements to the reed model included a provision for any level in the elasticity of collision against the lay and a scalable "Bernoulli"-like force;

Bell: modelled as a lowpass (reflection) / highpass (transmission) filter pair. For the reflection filter, the

user could select among a simple averager and 2nd, 4th, and 6th order Butterworth filters with a variety of cutoff frequencies. A simple one-pole filter was used for transmission. In addition, the reflection portion of the simulation was reconfigured to allow true complementary transmission;

Inputs: simulation driven by a 4-segment user-defined pressure envelope with optional vibrato and noise. The simulation operates purely on pressure wave propagation; flow is never explicitly calculated;

Outputs: Pressure output at the bell and register hole, as well as the reed tip displacement, were written to NeXT Soundfiles. In addition, an impulse response panel was available so that the user could obtain a spectrogram of the bore resonances.



RESULTS

The simulation successfully achieved a clarinet-like tone that could be modified by user inputs, and matched in behavior many of the experimental and theoretical predictions in the literature. Particularly noteworthy was the addition of the register hole scattering junction, which cleanly elicited the desired register shift of a twelfth. Other realistic behavior included:

Generation of harmonics with input amplitude, and subsequent timbre/volume dependency;

Existence of a threshold blowing pressure and the ratio of nonbeating to beating regime;

Sensitivity of steady-state mode to initial attack profile in marginally mode-stable cases, such as when steady-state attack pressure was below threshold. In extreme cases, with very high initial attack overshoot with respect to the steady-state input pressure, a "wind pluck" could be achieved, where the mode is first excited strongly, and then dies away in transient form as its support vanishes. In general, sensitivity to attack envelope was somewhat less than anticipated;

Effect of bell cutoff frequency on timbre. Adjusting the bell filter cutoff frequency provided a convenient means of modifying instrument tone. However, the bell in this simple form could not provide any of the midrange harmonic boosting that was demonstrated experimentally;

Instrument "squeaking" in the presence of reed dynamics and a low blowing pressure. Although squeaking has been attributed to various causes - usually reed resonance excitation - in the past, the most easily

obtained "squeak" was actually a third register shift which occurred near threshold blowing pressure. This was consistent with the admonishments of reed teachers in this author's experience to avoid squeaking by providing "sufficient support". Although added (and somewhat excessive) damping would suppress the squeak, simply raising the input pressure was the more musically meaningful way to solve the problem. Reed resonance squeaks similar to those reported in the literature were also obtained, by lowering the reed damping below the normal level.

From a musical standpoint, the addition of the register hole and the adjustable cutoff frequency showed much potential. The scattering junction bore perturbations did not reproduce the expected acoustical results, but did produce some interesting multiphonic sounds which could be useful. The reed dynamics did not seem to contribute much musically, at least with this simple model. The "Bernoulli" force did have some aural effects on tone, but could be precalculated in the look-up table the same result. Especially given the dearth of current understanding of how reeds really operate (a situation which Hirschberg is trying to correct, in, for example, [4]), the inclusion of reed dynamic computations in a real-time model seems wasteful at this point. However, with the development of a more complex, less modally stable bore with a tonehole lattice, the sensitivity of the results to currently discardable second order factors would probably be greatly increased.

Although the simulation itself did not run in real-time - a typical one second run took about five seconds to complete - it was still very fast, especially considering the number of output files it wrote to and the amount of testing required because of the many options offered in the user interface. It is conceivable that a real-time runtime ratio could be obtained on a dedicated version of the instrument with streamlined options, even without using the NeXT DSP. For the workbench environment, a real-time ratio is not required. However, the combination of the high running speed and the convenient user interface allowed for a considerable amount of intuitive experimentation and parameteric study which would have been impossible in a less responsive environment.

REFERENCES

1) Smith, J. O, "Music Applications of Digital Waveguides," Stanford University Music Department Technical Report STAN-M-39, Stanford, California: 1987.

2) Hirschman, S. E., "Digital Waveguide Modeling and Simulation of Reed Woodwind Instruments", Engineer Thesis, Stanford University Department of Electrical Engineering, 1991.

3) Smith, J. O., "Efficient Simulation of the Reed-Bore and Bow-String Mechanisms", *Proceedings of the 1986 International Computer Music Conference.* Computer Music Association, 1986.

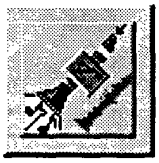
4) Hirschberg, A., R. W. A. van de Laar, et. al., "A Quasi-stationary Model of Air Flow in the Reed Channel of Single-Reed Woodwind Instruments," *Acustica*, vol. 70, pp. 146-154, 1990.

DIGITAL WAVEGUIDE MODELLING OF REED WOODWINDS: AN INTERACTIVE DEVELOPMENT

Suzanne E. Hirschman, Perry R. Cook, Julius O. Smith (szh, prc @ccrma.stanford.edu, julius_smith@next.com)
Center for Computer Research in Music and Acoustics (CCRMA), Stanford University

ABSTRACT

The digital waveguide filter has proven to be extremely effective for modelling wave propagation in musical instruments. It can be easily connected to independent models of reed mechanisms and bell shaping functions. In addition, the waveguide can be adapted to simulate arbitrary bore shapes and tone hole lattices. The WGF was used as the basis for an interactive modelling environment for a generic cylindrical cane reed-type instrument on the NeXT computer. The user has direct access to parameters defining the reed, the bell, the waveguide length, and the attack envelope. The program offers both reed lookup table and dynamic reed models for reed implementation, and a number of filter types for bell implementation. In addition, a single tone hole, implemented as a two-port scattering junction with "diameter" and placement defined by the user, is included. The ouptut of the simulation is a NeXT soundfile, which can be played and analyzed using a variety of tools on the NeXT computer. In addition, a spectrum panel allows the user to generate and analyze the impulse response of the instrument. Studies performed with this simulation include: implementation of a register hole; influence of attack on steady-state mode excitation; single vs. double reed behavior; influence of bell and cutoff frequency on tone; and influence of reed resonance on tone and stability.



INTRODUCTION

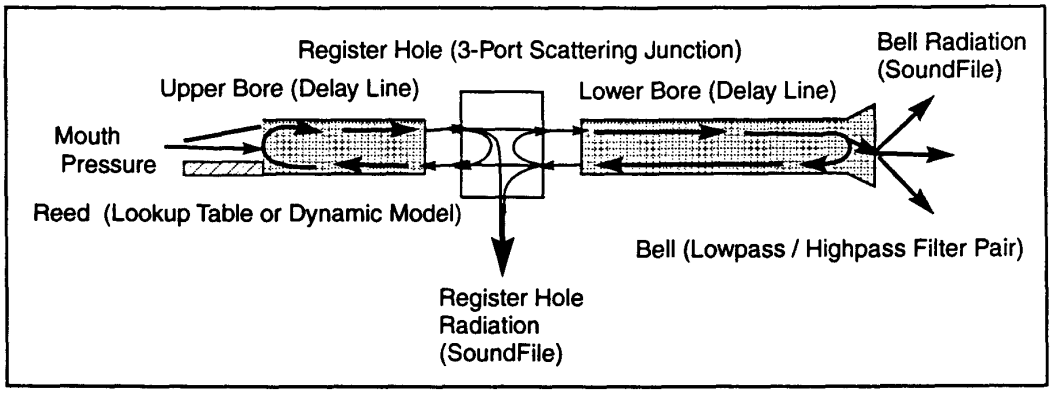
As the field of digitally produced sound, and its associated computer technology develops, so does the potential for creating realistic real-time computer simulations that model acoustic instruments. Simulation has long been a tool for acoustical analysis; in the field of physical modelling, it is rapidly becoming a viable and exciting resource for musical performance.

The primary source of musical sound in an acoustic instrument is the propagation of waves through the instrument medium and the subsequent pressure radiation to the outside air. A good model of the instrument must therefore duplicate the wave behavior, as well as the nonlinear relationships between the medium and its excitation mechanism. Julius Smith has demonstrated that the waveguide digital filter, an offshoot of the normalized ladder/lattice filter, is an ideal structure for this task [1]. The waveguide filter is

essentially a two-way delay line interspersed with partially reflective "scattering" junctions which operate on the oncoming wavefronts. Such a filter implements the wave propagation equations exactly. Just as a tonehole or bore diameter change will create an acoustic barrier within an instrument, allowing only part of the wave to penetrate while reflecting the rest, so can the associated changes in impedance be used to calculate the filter reflection coefficients which will result in similar scattering of the propagated, simulated wavefronts. The waveguide section is extremely modular, and can easily be connected to models of excitation and terminating impedance.

This paper, a summary of a more complete report available in reference [2], discusses the implementation of Smith's clarinet model [3] in an interactive environment on the NeXT computer. The clarinet model in its basic form is particularly elegant and efficient, as its cylindrical bore reduces much of the waveguide to a simple delay line. Inelaborate as it is, it provides a powerful demonstration of the principles of wave propagation, and of the essential coupling between reed and air column, as well as being a potentially rich source of musical sound. The simulation, dubbed the "ClariNeXT Workbench", was designed as an evolutionary, interactive workbench both for exploring acoustical behavior and for evaluating the musical potential, with respect to a real-time digital instrument, of various refinements to the basic model.

THE CLARINEXT MODELLING WORKBENCH



The clarinet simulation was developed in Objective C on the NeXT computer, using the NeXT Interface Builder application to provide the graphical user interface. It was designed to run with the 22050 Hz sampling rate used by the NeXT sound processing software. The simulation contained the following component models:

Cylindrical Bore Sections: modelled by pure delay lines, sized according to the pitch designated by the user;

Register hole: modelled as a 3-port scattering junction at a user-defined location along the bore. A modified 2-port scattering junction was also provided to allow a crude source of bore perturbation;

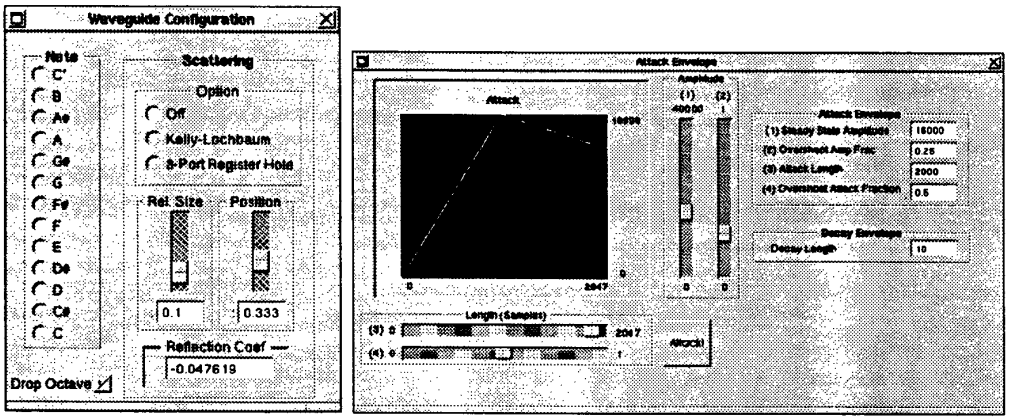
Reed: modelled as a time-varying reflection coefficient based on either 1) Smith's static reed look-up table [3] or 2) a 2nd order dynamic model with user-definable damping ratio and resonant frequency. Additional enhancements to the reed model included a provision for any level in the elasticity of collision against the lay and a scalable "Bernoulli"-like force;

Bell: modelled as a lowpass (reflection) / highpass (transmission) filter pair. For the reflection filter, the

user could select among a simple averager and 2nd, 4th, and 6th order Butterworth filters with a variety of cutoff frequencies. A simple one-pole filter was used for transmission. In addition, the reflection portion of the simulation was reconfigured to allow true complementary transmission;

Inputs: simulation driven by a 4-segment user-defined pressure envelope with optional vibrato and noise. The simulation operates purely on pressure wave propagation; flow is never explicitly calculated;

Outputs: Pressure output at the bell and register hole, as well as the reed tip displacement, were written to NeXT Soundfiles. In addition, an impulse response panel was available so that the user could obtain a spectrogram of the bore resonances.



RESULTS

The simulation successfully achieved a clarinet-like tone that could be modified by user inputs, and matched in behavior many of the experimental and theoretical predictions in the literature. Particularly noteworthy was the addition of the register hole scattering junction, which cleanly elicited the desired register shift of a twelfth. Other realistic behavior included:

Generation of harmonics with input amplitude, and subsequent timbre/volume dependency;

Existence of a threshold blowing pressure and the ratio of nonbeating to beating regime;

Sensitivity of steady-state mode to initial attack profile in marginally mode-stable cases, such as when steady-state attack pressure was below threshold. In extreme cases, with very high initial attack overshoot with respect to the steady-state input pressure, a "wind pluck" could be achieved, where the mode is first excited strongly, and then dies away in transient form as its support vanishes. In general, sensitivity to attack envelope was somewhat less than anticipated;

Effect of bell cutoff frequency on timbre. Adjusting the bell filter cutoff frequency provided a convenient means of modifying instrument tone. However, the bell in this simple form could not provide any of the midrange harmonic boosting that was demonstrated experimentally;

Instrument "squeaking" in the presence of reed dynamics and a low blowing pressure. Although squeaking has been attributed to various causes - usually reed resonance excitation - in the past, the most easily

obtained "squeak" was actually a third register shift which occurred near threshold blowing pressure. This was consistent with the admonishments of reed teachers in this author's experience to avoid squeaking by providing "sufficient support". Although added (and somewhat excessive) damping would suppress the squeak, simply raising the input pressure was the more musically meaningful way to solve the problem. Reed resonance squeaks similar to those reported in the literature were also obtained, by lowering the reed damping below the normal level.

From a musical standpoint, the addition of the register hole and the adjustable cutoff frequency showed much potential. The scattering junction bore perturbations did not reproduce the expected acoustical results, but did produce some interesting multiphonic sounds which could be useful. The reed dynamics did not seem to contribute much musically, at least with this simple model. The "Bernoulli" force did have some aural effects on tone, but could be precalculated in the look-up table the same result. Especially given the dearth of current understanding of how reeds really operate (a situation which Hirschberg is trying to correct, in, for example, [4]), the inclusion of reed dynamic computations in a real-time model seems wasteful at this point. However, with the development of a more complex, less modally stable bore with a tonehole lattice, the sensitivity of the results to currently discardable second order factors would probably be greatly increased.

Although the simulation itself did not run in real-time - a typical one second run took about five seconds to complete - it was still very fast, especially considering the number of output files it wrote to and the amount of testing required because of the many options offered in the user interface. It is conceivable that a real-time runtime ratio could be obtained on a dedicated version of the instrument with streamlined options, even without using the NeXT DSP. For the workbench environment, a real-time ratio is not required. However, the combination of the high running speed and the convenient user interface allowed for a considerable amount of intuitive experimentation and parameteric study which would have been impossible in a less responsive environment.

REFERENCES

1) Smith, J. O, "Music Applications of Digital Waveguides," Stanford University Music Department Technical Report STAN-M-39, Stanford, California: 1987.

2) Hirschman, S. E., "Digital Waveguide Modeling and Simulation of Reed Woodwind Instruments", Engineer Thesis, Stanford University Department of Electrical Engineering, 1991.

3) Smith, J. O., "Efficient Simulation of the Reed-Bore and Bow-String Mechanisms", *Proceedings of the 1986 International Computer Music Conference.* Computer Music Association, 1986.

4) Hirschberg, A., R. W. A. van de Laar, et. al., "A Quasi-stationary Model of Air Flow in the Reed Channel of Single-Reed Woodwind Instruments," *Acustica*, vol. 70, pp. 146-154, 1990.

TBone: An Interactive WaveGuide Brass Instrument Synthesis Workbench for the NeXT Machine

Perry R. Cook
Stanford Center for Computer Research in Music and Acoustics

Abstract

A system for interactive experimentation with waveguide brass instrument synthesis has been constructed on the NeXT machine. Controls affecting individual tubing section length, slide position (in the trombone case), and valve positions determine the overall length of the cylindrical part of the horn. Controls affecting bell flare type and amount determine the characteristics of the bell. A mouthpiece editor allows adjustment of the shape and length. Spectral displays show the response of the horn as affected by changes in physical control parameters. A lip editor controls mass, spring constant, and damping of the lip oscillator. Performance controls allow sweeping of the lip oscillator, breath pressure, slide position, and valve position individually or together. Remarkably convincing brass instrument synthesis is accomplished by setting the variables to physically reasonable values.

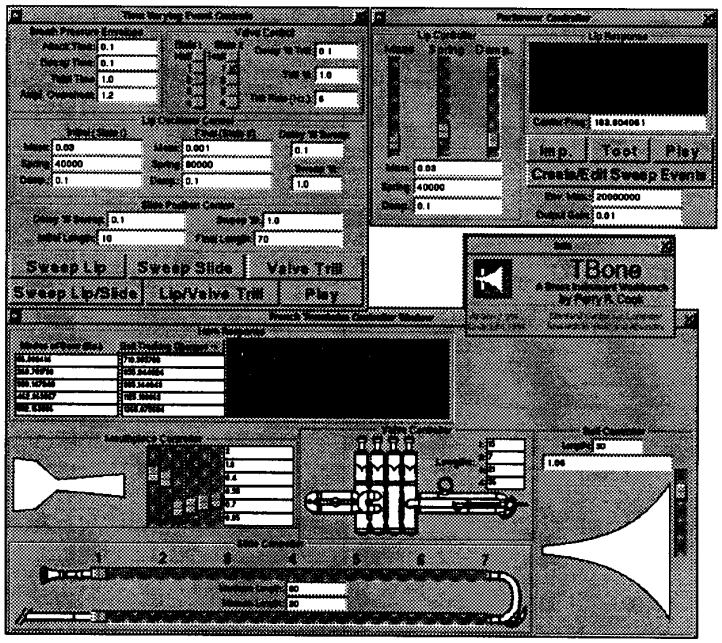


Figure 1: TBone windows provide controllers for the horn, the player, and time varying synthesis.

1 The Instrument

The instrument is composed of a mouthpiece of arbitrary shape, four cylindrical tubing sections switched by valves, a cylindrical tubing section of continuously variable length (trombone slide), and an exponentially flaring bell.

The section of brass instruments which begins at the mouthpiece leadpipe and extends through the network of valves and tuning slides is essentially a cylindrical bore. The freely moving slide and tuning slides of the trombone and valved instruments would not work correctly if the tubing were not cylindrical. These cylindrical sections are well modeled using simple delay line pairs (waveguides) [1]. In the TBone simulator, there are 5 sets of delay-line pairs active at all times, representing a continuously varying slide and four valves. Using these elements, an arbitrary brass instrument can be built. Editors control the length and range of the trombone slide element, and the length of each of the four valve controlled tubing sections. Fractional sample lengths are modeled by using longer delay lines than necessary, and interpolating the contents to the appropriate fractional location. The valves are modeled as four-way scattering junctions, allowing fractional valve positions to be simulated.

The mouthpiece can have arbitrary shape, and is modeled using one scattering junction per spatial sample. The bell is modeled as an exponentially flaring acoustic tube, using one scattering junction per spatial sample. A simple low-pass filter is used to model the reflection characteristics at the end of the bell.

2 The Performer

The performer is composed of a mass-spring oscillator modeling the lip, controls affecting the breath pressure envelope, and controls affecting valve and slide movements. To efficiently model the lip, a number of assumptions are made. The primary oscillation is assumed to take place in the upper lip, which is modeled as a simple mass-spring-damper oscillator. Figure 1 shows the lip oscillator and equivalent model.

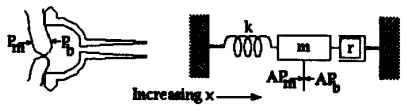


Figure 1: A drawing of the lips as placed on the mouthpiece, and a simplified physical model of the upper lip.

The force equation governing the simplified lip oscillator position x is:

$$f = (P_m - P_b)A - kx - r\dot{x} = m\ddot{x} \qquad (1)$$

Where P_m and P_b are the mouth and bore pressures acting on the lip. A is the area, k is the spring constant, r is the damping coefficient, and m is the mass of the lip. For a simple discrete time simulation, the velocity and acceleration components, $\dot{x}$ and $\ddot{x}$ are replaced with their finite difference approximations.

$$\dot{x} \approx \frac{x(t) - x(t-T)}{T} \qquad (2)$$

and

$$\ddot{x} \approx \frac{x(t+T) - 2x(t) + x(t-T)}{T^2} \qquad (3)$$

Substitution and algebraic manipulation yields a second order recursive digital filter, which when run at the sampling rate approximates the position of the lip as a function of the net force acting on the lip.

$$\frac{X}{F} = \frac{T^2 Z^{-1}}{m - (2m - Tr + T^2 k)Z^{-1} + (m - Tr)Z^{-2}} \qquad (4)$$

As the sampling period approaches zero, the digital filter approximation to the continuous time differential equation becomes more accurate. Once the lip position is calculated, it is used to look up a reflection/transmission coefficient from a table. This table is similar to those

used for massless reed simulations in the models of the clarinet and saxophone[2][3[4]. The lip position is limited at each extreme (closed, or open to the point that it hits the mouthpiece cup) by limiting the position state variable to the minimum or maximum value.

3 Synthesis Examples

Synthesized tones of great variety, both realistic and fantastic, can be produced rapidly with the simulator. Graphical interaction along with fast auditory feedback encourage experimentation and new sounds. To briefly explore the synthesis capabilities of the system, two synthesis examples will be presented here. The first synthesis example is that of a crescendo and decrescendo. Since the lip oscillator is non-linear, the system behaves quite differently in different regions of excitation amplitude. Figure 2 shows the amplitude envelope and the pitch trajectory of the synthesized tone. The measured pitch varied from 230 to 236 Hz. (45 cents), with the tone flattening at both the softest and loudest points.

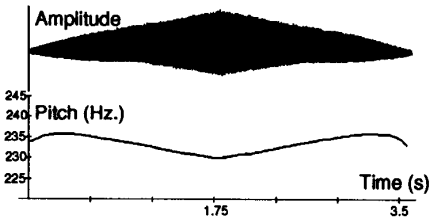


Figure 2: Amplitude envelope and the pitch trajectory of a tone synthesized with linearly increasing then decreasing breath pressure.

The second example is that of a sweep of the lip simulator parameters of mass and spring constant. The lip oscillator was initially configured for a resonance frequency of 183 Hz., then was swept over one second to a final resonance frequency of 714 Hz. The resulting synthesis exhibited an upward sweep in frequency, but since only certain modes are reinforced by the horn simulation, only certain frequencies were favored. The waterfall spectrum plot of Figure 3 shows the spectral evolution of this synthesized tone.

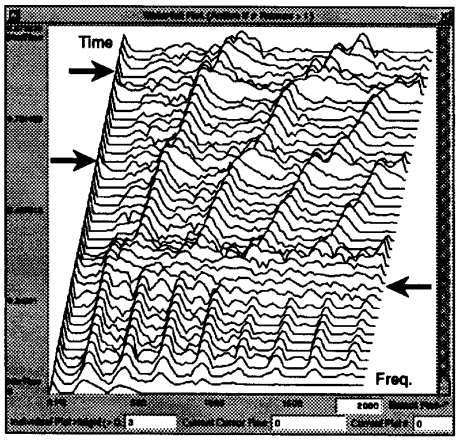


Figure 3: Time varying spectral plot of TBone synthesis performed while sweeping the lip oscillator from 183 to 714 Hz. The arrows show where transitions from one mode of oscillation to another occurred.

References

[1] J. O. Smith, "Musical Applications of Digital Waveguides." Stanford University Center For Computer Research in Music and Acoustics, Report No. STAN-M-39, 1987.

[2] J. O. Smith, "Efficient Simulation of the Reed-Bore and Bow-String Mechanisms." Proceedings of the ICMC, Computer Music Association, 1986.

[3] P. R. Cook, "Implementation of Single Reed Instruments With Arbitrary Bore Shapes Using Digital Waveguide Filters." Stanford University Center For Computer Research in Music and Acoustics, Report No. STAN-M-50, 1988.

[4] S. Z. Hirschman, "Digital Waveguide Modeling and Simulation of Reed Woodwind Instruments." Engineer Thesis, Stanford University Department of Electrical Engineering, 1991.

Non-Linear Periodic Prediction for On-Line Identification of Oscillator Characteristics in Woodwind Instruments

Perry R. Cook
Stanford Center for Computer Research in Music and Acoustics

Abstract

Non-linear periodic prediction techniques are used to identify characteristics of woodwind instruments, and to investigate the non-linear oscillator elements. A Kalman filter implementation of a polynomial non-linear predictor was configured to predict the signal either one or one-half period ahead. Using knowledge about instrument structure, the resulting polynomial coefficients can be algebraically manipulated to yield forms which fit the parameters of standard digital simulation models of such instruments. Distinctions between hard and soft single clarinet reeds are easily seen, as well as distinctions between reeds and jets. A brief discussion is included on the mathematical and modeling differences in non-linearities exhibiting memory vs. memoryless non-linearities.

1 Non-Linear Elements in Musical Systems

Most musical instruments capable of producing a sustained tone can be modeled by combining linear elements with non-linear oscillator elements[1][2]. Often the linear and non-linear elements do not vary with time. The acoustic tubes of the clarinet bore and vocal tract, the bell, and the vibrating string can often be modeled quite well by linear elements (filters or simple delay lines). The non-linear oscillators model the reflection/transmission behavior of the reed, reeds, vocal folds, lips, or bow/string interface. These non-linearities can be modeled to arbitrary accuracy by polynomial functions, or state dependent polynomials in cases such as the bow/string interface where hysteresis is present.

A memoryless polynomial non-linearity has a simple form which contains no delayed versions of the signal (no memory). The mathematical form of an Nth order memoryless non-linear element is:

$$f(x(n)) = \sum_{i=0}^{N} a_i x(n)^i \tag{1}$$

A general non-linear element can be represented as the formation, weighting, and summing of products of the signal and delayed versions of the signal. The memoryless polynomial element can be viewed as a special case of the general non-linear element. The mathematical form of the general non-linear element grows rapidly in complexity for even small orders of non-linearity and delay. This introduces practical considerations in calculating and identifying the coefficients of the polynomial representation of a high-order general non-linearity. An example of a non-linear element with $3rd$ order non-linearity and $2nd$ order delay is:

$$f(x(n)) = \begin{cases} a_{0,0} + a_{1,0}x(n) + a_{2,0}x(n)^2 + a_{3,0}x(n)^3 + a_{1,1}x(n-1) + a_{2,1}x(n-1)^2 + \\ a_{3,1}x(n-1)^3 + a_{1,2}x(n-2) + a_{2,2}x(n-2)^2 + a_{3,2}x(n-2)^3 + \\ b_{0,1}x(n)x(n-1) + b_{0,2}x(n)x(n-2) + b_{1,2}x(n-1)x(n-2) + \\ c_{0,1,1}x(n)x(n-1)^2 + c_{0,2,2}x(n)x(n-2)^2 + c_{1,2,2}x(n-1)x(n-2)^2 + \\ c_{1,1,2}x(n-1)^2x(n-2) + c_{0,1,2}x(n)x(n-1)x(n-2) \end{cases} \quad (2)$$

To investigate some mathematical and computational issues, a simplified clarinet/flute model with no tone holes will be used as an example. If the bell reflection element is 'pushed through' the bore and joined with the reed element, the system of Figure 1 results.

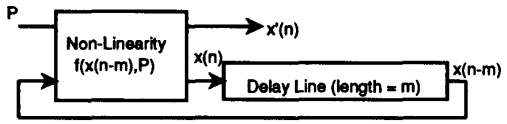


Figure 1: A simplified clarinet/flute model.

The cylindrical bore is modeled by a simple delay line, and the characteristics of bore and reed reflection/transmission are lumped into a single general two input two output non-linear element. Denote the instantaneous breath pressure as $p(n)$, the output as $x(n)$, and the non-linear operation as $F(x,p)$. In the quasi-steady state of oscillation, $p(n)$ can be considered to be a constant, P. The equation governing steady state oscillation is:

$$x(n) = F(x(n-m), P) \quad (3)$$

where m is the period of oscillation in the case of a recorder or flute, and is one-half the period of oscillation in the case of the clarinet. If the behavior of the non-linear operation is reed-like and assumed to be memoryless, the output can be modeled as a memoryless polynomial function of the differential pressure operating on the reed.

$$F(x,P) = \sum_{i=0}^{N} a_i(P-x)^i \quad (4)$$

Often a function $F'(x)$ is desired which models the normalized characteristics of a particular physical non-linearity, such as a reed reflection table. Relating Equation 4 to the coefficients of the physical non-linearity of a reed table can be accomplished through algebraic manipulation, and simplifies if the constant steady-state pressure is normalized to 1:

$$x(n) = F'(x(n-m))x(n-m) + (1 - F'(x(n-m))) \quad \Longrightarrow \quad F'(x) = \sum_{i=0}^{N-1} k_i x^i \quad (5)$$

The coefficients k_i and a_i are related by an upper diagonal matrix whose entries are the binomial coefficients of Pascal's Triangle. For a 4th order predictor, the coefficients of the reed non-linearity are related to those of the predictor by:

$$\begin{bmatrix} 1 \\ k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 \\ 0 & -1 & 2 & -3 & 4 \\ 0 & 0 & 1 & -3 & 6 \\ 0 & 0 & 0 & -1 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} \tag{6}$$

Various possibilities for the signs of the entries of the matrix of Equation 6 are determined by the form of the system being identified. These depend on the number of inversions in the system, the input variable P as assumed to be +1, and other mathematical factors. An indication is given as to the correctness of the assumptions, measurements, normalizations, and calculations by the first condition of Equation 6, which states that the sum of the predictor coefficients should be +1. Other forms dictate that the sum should be -1.

2 Identification Method

A modified Kalman filter [3] was used to identify the non-linear function coefficients. The state equations for the memoryless non-linearity are:

$$\begin{bmatrix} X_{k+m} \\ a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & X_k & X_k^2 & \cdots & X_k^n \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} X_k \\ a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} + \begin{bmatrix} W_k \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{7}$$

$$Y_k = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} X_k \\ a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} + V_k \tag{8}$$

where W_k and V_k are zero-mean white Gaussian noise sequences, representing model and measurement noise, respectively.

3 Experimental Results for Memoryless Predictor

Two tones were synthetically generated using a computer clarinet model. One tone used a reed with normal rest position (Reed 1), and the other used a reed with a large opening rest position (Reed 2). Figure 2 shows the results of identification performed on synthesized clarinet tones. Histograms are shown below each reed curve to show how many samples were used to compute each point on the non-linear function. This gives an estimate as to the confidence of regions of the curve.

Two tones were recorded with a microphone mounted inside of a section of pipe joined to a clarinet mouthpiece. The two tones were generated using a soft reed and a hard reed. Figure 2 shows the

results of identification performed on the clarinet tones. The two curves of the soft actual reed were computed using the acoustical signal inside and outside the instrument. The stiffness of each reed is reflected in the slope of the curve as it passes through the origin (rest position). The greater the negative slope, the softer the reed.

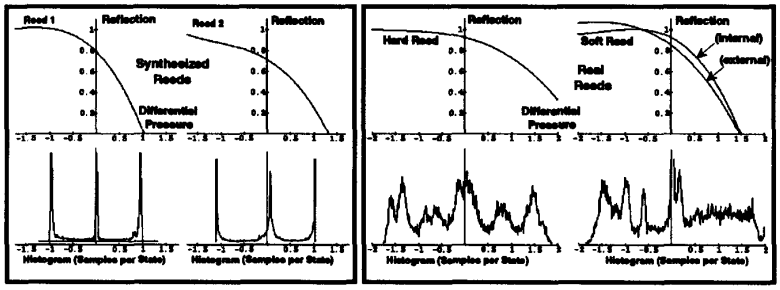


Figure 2: Non-linear reed identification results for synthetic clarinet tones (left), and actual clarinet tones (right) performed with soft and hard reeds.

As a final example, a tenor recorder was recorded while blowing the fundamental (all holes covered), then overblowing to the first overtone one octave above. Figure 3 shows the results of predicting the signal one-half period ahead (interpreting the instrument structure as that of a clarinet), and one period ahead (correct for the recorder). The half-period prediction results yield greatly different jet characteristics for the two tones, while the full-period prediction results yield quite similar jet reflection curves for the two tones.

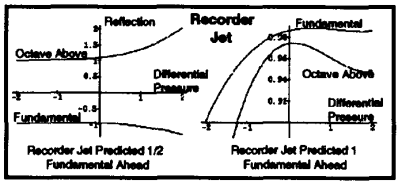


Figure 3: Non-linear identification results for actual recorder tones at half-period (left) prediction, and full-period (right) prediction. The two tones are the fundamental and the overblown octave above.

References

[1] M. E. McIntyre, R. T. Schumacher, and J. Woodhouse, "On the Oscillations of Musical Instruments." JASA, 74, 5, 1325-1345, 1983.

[2] J. O. Smith, "Musical Applications of Digital Waveguides." Stanford University Center For Computer Research in Music and Acoustics, Report No. STAN-M-39, 1987.

[3] T. Kailath, *Lectures on Wiener and Kalman Filtering.* Springer, 1981.