CCRMA PAPERS PRESENTED AT THE
1995 INTERNATIONAL COMPUTER MUSIC CONFERENCE
BANFF, CANADA

Marcia L. Bauman, David Berners, Chris Chafe, Perry R. Cook,
David A. Jaffe, Tobias Kunze, Fernando Lopez-Lezcano,
Gary P. Scavone, Julius O. Smith III, Tim Stilson,
Heinrich Taube, Steven D. Trautmann, and Scott A. Van Duyne

CCRMA
DEPARTMENT OF MUSIC
Stanford University
Stanford, California 94305-8180
TABLE OF CONTENTS

Marcia L. Bauman
The International Digital Electroacoustic Music Archive 1

David Berners and Julius O. Smith, III
Super-Spherical Wave Simulation in Flaring Horns 3

Chris Chafe
Adding Vortex Noise to Wind Instrument Physical Models 5

Perry R. Cook
A Hierarchical System for Controlling Synthesis by Physical Modeling 11

Perry R. Cook
Integration of Physical Modeling for Synthesis and Animation 13

David A. Jaffe and Julius O. Smith, III
Performance Expression in Commuted Waveguide Synthesis of Bowed Strings 17

Fernando Lopez-Lezcano
PadMaster: an Improvisation Environment for Real Time Performance 21

Gary P. Scavone
Digital Waveguide Modeling of the Non-Linear Excitation of Single Reed Woodwind Instruments 23

Julius O. Smith, III and Scott A. Van Duyne
Committed Piano Synthesis 27

Tim Stilson
Forward-Going Wave Extraction in Acoustic Tubes 35

Tim Stilson
General Weirdness with the Karplus-Strong String 39
Heinrich Taube and Tobias Kunze

*Capella: A Graphical Interface for Algorithmic Composition* 41

Steven D. Trautmann

*A Physical String Model with a Twist* 45

Steven D. Trautmann

*Toward a CLM Sound Localization Instrument employing Modified Wavefront Reconstruction* 47

Scott A. Van Duyne and Julius O. Smith, III

*Developments for the Commuted Piano* 49
THE INTERNATIONAL DIGITAL ELECTROACOUSTIC MUSIC ARCHIVE
Marcia L. Bauman
Stanford University Center for Computer Research in Music and Acoustics
Tel: (415) 723-4971/E-mail: bau@CCRMA.stanford.edu

Thomas Gerwin
Zentrum für Kunst und Medientechnologie
Tel: 0721/9340-300/E-mail: tg@guido.zkm.de

Abstract

The International Digital ElectroAcoustic Music Archive (IDEAMA) was created to collect, preserve and disseminate historically significant electroacoustic music. It was co-founded in December, 1990, by Stanford University’s Center for Computer Research in Music and Acoustics (CCRMA), and by the Center for Art and Media Technology (ZKM), in Karlsruhe, Germany. Technology for digital sound and text storage and retrieval has been implemented at both locations. This paper describes various phases in the creation of the IDEAMA and plans for its dissemination. The hardware and software used to author and access the IDEAMA’s various disk formats are also discussed.

I. Phase One - Setting up Shop: Administrative Structure

Boards

An international advisory board of renowned composers was formed to help establish the international scope and reputation of the archive. To identify, locate and choose materials for the target collection, CCRMA and ZKM each formed a selection committee comprised of eminent composers, musicologists and other individuals who are well-versed and active in the field.

There are three types of IDEAMA institutions: founding institutions (ZKM and CCRMA), partner and affiliate institutions. The founding institutions have collaborated to establish policies and procedures for creating the archive and its ongoing function. The partner institutions have participated in the formation of the archive, in most cases by contributing materials. They will eventually house the archive, as will the affiliate branches.

Branches

Presently, there are eight formally designated partner institutions: The New York Public Library; the National Center for Science Information Systems (NACSIS) in Tokyo; IRCAM and INA/GRM in Paris; GMEB in Bourges; EMS in Sweden; the IPEM at the University of Ghent, Belgium; and most recently, the Instituut voor Sonologie, of the Konijiklink Conservatory in the Hague, Netherland.

II. Phase II - Creating the Target Collection

CCRMA and ZKM are jointly responsible for collecting archive materials on a regional basis: ZKM focuses on European electroacoustic music, while CCRMA is responsible for music from the Americas, Asia and Australia. The original analog tapes for targeted works, composed between 1940 and 1970, have existed in a number of archives, radio stations, studios and private collections. Over two hundred works have been acquired by CCRMA, and approximately 320 European works are now being processed at ZKM.

ZKM Selections

Sources for the European works include numerous major centers such as INA/GRM Paris; WDR Köln; EMS Stockholm, Experimental Studio Warszaw and the former Studio di Fonologia, Milano. In addition, works from smaller studios and private collections, and from the estate of Hermann Heiss have
been included. The European target collection list contains 480 historical works.

**CCRMA Selections**

Sources for works in the USA include the Columbia-Princeton Electronic Music Center, the defunct San Francisco Tape Music Center (works now housed at Mills College, the University of California at San Diego, and by individual composers), the University of Illinois Experimental Music Center, Bell Telephone Laboratories (personal collection of Max Mathews), various individual composers, and commercial CDs. Significant works by Canadian and Australian composers are presently being sought. The Laboratoris de Investigacion y Produccion Musical (LIPM), the first major center for Latin American electroacoustic music, has digitized approximately 30 works for the target collection. Approximately 50 Japanese works have been provided by the National Center for Science Information Systems and by Dr. Emmanuelle Loubet. Most of the Japanese works were originally produced at the Tokyo studio of NHK radio. Concurrent with the collection of the music, arrangements for copyright clearance and research to obtain information about the music have been ongoing at CCRMA and ZKM.

**III. Phase III - Transferring to Permanent Storage Media and IDEAMA Distribution**

The final phase of the project involves transferring the music to the permanent digital storage media, creating the database and distributing the IDEAMA to partner and affiliate institutions. Works were originally transferred from analog tapes to DAT cassettes. They are now being transferred to disks. Four-channel works are being transferred to CD-ROM. At CCRMA, stereo works are also transferred to CD-ROM, as well as mixed-mode CD and stereo audio CD. CD-ROM is the preferred archival medium because of its greater capacity for error detection, and because it is easier to retrieve and download the sound from it. ZKM uses the Sonic Solutions system to produce stereo audio CDs. Equipment at CCRMA includes the Apple Power Mac 7100; the Apple CD-300 CD-ROM reader; the Micropolis external SCSI hard drive (1.7 gigabytes); digidesign’s ProTools audio card and audio interface; Pro Tools and Sound Designer II audio editing software; OMI’s QuickTOPIX software to author disks; and the Sony CDW 900-E CD recorder.

At ZKM, IDEAMA works will be stored on stereo audio CDs, forming a major component of ZKM’s Mediathek library. Stanford University will be store works on CD-ROMs and house them at the Braun Music Center’s Archive of Recorded Sound. At Stanford and at ZKM, the disks will be placed in a jukebox which will be activated via computer terminal. CD-ROMs and mixed-mode CDs can be accessed using either an Apple computer or a PC. The Claris FileMakerPro commercial database will be used to access catalog information about the music. CD-ROMs also contain a larger text file with information about the music. Partner and affiliate institutions will be able to specify their choice of format for sound, and will receive the catalog database on a floppy disk.

We are presently in this final phase of the IDEAMA project, completing the transfer of materials to CD and CD-ROM and installing the IDEAMA at ZKM’s Mediathek and Stanford’s Archive of Recorded Sound. CCRMA and ZKM are in the process of developing the policies for the distribution of the IDEAMA. We anticipate that the entire IDEAMA (CCRMA and ZKM materials) will be available in April (1996). At the present time, it seems that there will be approximately 140 disks. We look forward to making the IDEAMA accessible on as broad a basis as possible, consistent with the materials in the archive, and at the lowest cost possible.

**Acknowledgements**

At Stanford University, the IDEAMA is supported by CCRMA, the Andrew W. Mellon Foundation and the National Endowment for the Arts. ZKM receives its support from the state of Baden-Wurtemberg and the city of Karlsruhe.
Super-Spherical Wave Simulation in Flaring Horns

David Berners and Julius O. Smith, III
Center for Computer Research in Music and Acoustics (CCRMA)
Dept. of Music, Stanford University, Stanford, CA 94305
dpberner@ccrma.stanford.edu, jos@ccrma.stanford.edu

Abstract: The flared horn is modeled according to Webster’s equation. A change of variables transforms the equation into the form of the one dimensional Schrödinger wave equation. The Schrödinger form facilitates specification of arbitrary axisymmetric wavefronts for the acoustic disturbance within the horn. To provide a physically motivated choice of wavefront shape, Poisson’s equation is solved inside the horn subject to the boundary condition that the normal component of the potential gradient is zero at the boundary of the horn. Since the disturbance within the horn must satisfy the wave equation, the velocity potential satisfies Poisson’s equation when viscous effects and losses are ignored. Physical data from brass instrument bells are used to model musical horns using the Poisson solution, and results are compared to those obtained by traditional models which assume spherical wavefronts.

Acknowledgment

This work was made possible by the generous cooperation of Mr. Greg Hilliard of Frank Holton and Company who provided us with precise physical measurements of a test bell.

1 Introduction

In [Berners and Smith, 1994] a method has been proposed to determine the acoustical properties of flared bores using a form of Webster’s horn equation developed in [Benade and Jansson, 1974]. The method allows any choice of acoustic wavefronts propagating in the bore as long as axial symmetry is preserved. Here we discuss an application of that method using Poisson’s Equation to predict the acoustic wavefronts within a bore.

2 Wavefront Computation

Because of similarities between the acoustic wave equation and the classical field equation for electrostatics, the isophase pressure wave surfaces at relatively low frequencies are equivalent to equipotential surfaces found within an insulator having the shape of the horn wall with an axial potential drop. An insulating boundary with a high relative dielectric constant causes equipotential planes to be perpendicular to the boundary. This is appropriate in comparison to the velocity potential $\Phi$ within an acoustic horn, which also must have equipotential lines which are perpendicular to the boundary due to the fact that

$$\frac{\partial \Phi}{\partial n} = 0$$

at the boundary wall.
Wavefronts within a typical brass bell were computed using the electrostatic Poisson solver RELAX3D which was developed at the TRIUMF Meson Facility in Vancouver, Canada (triumf.ca). The bell boundary was defined by sampled measurements taken from a bell mandrel. Figure 1a shows equipotential surfaces of the solved grid along with the horn boundary. The surfaces become less spherical in regions of greater flare. For a given assumed acoustic wavefront shape, we define the equivalent radius as $r(z) = \frac{1}{2} \sqrt{S(z)}$, where $S(z)$ is the surface area of the wavefront crossing the horn axis at $z$. Figure 1b shows equivalent radii for plane waves, spherical waves, and waves obtained by the Poisson solver within the test horn. It can be seen that the equivalent radius for the Poisson solution falls between those for the planar and spherical wavefronts.

Figure 2a shows the barrier functions for the spherical and Poisson solution wavefronts. As defined in [Berners and Smith, 1994], the barrier is the normalized second derivative of the equivalent radius, $\frac{d^2}{dz^2} r$. The spherical wavefront model produces a higher, narrower barrier which results in a slightly higher cutoff frequency in the reflection coefficient shown in Figure 2b. However, the difference between the two models is minimal in terms of response, and it is likely that the spherical model would be acceptable for this test case.

References


Addition Vortex Noise to Wind Instrument Physical Models

Chris Chafe
Center for Computer Research in Music and Acoustics, Stanford University
cc@ccrma.stanford.edu

ABSTRACT: Flutes and other switching air jet instruments do not exhibit period-synchronous pulsed noise. However, period-synchronous spectral changes were detected in a flute tone. A synthetic flute model is described which imitates turbulent qualities of breath noise with a vortex noise generator circuit.

1 Introduction

Noise in bowed stringed instruments and many other musical oscillators exhibits distinct pitch-synchronous amplitude pulses (Chafe, C). Synthesis quality has been improved by coupling a pulsed noise mechanism to the basic physical model of these instruments. Edge-tone instruments such as flutes and organs are an exception. In the present study, their breath noise is shown to have pitch-synchronous spectral features and a more constant amplitude contour.

Turbulent noise caused by the switching air jet of edge-tone type instruments includes vortex formation and shedding in three dimensions (Verge, M.). As the player excites the air column into oscillation by blowing across the edge, the air jet at the mouthpiece begins to rapidly alternate direction in time with the column's vibration. The process elicits both frictional noise at the constriction where air enters the pipe and air puffs that roll away as the returning pressure wave impedes the entering flow. The formation of vortices and modulation of fricative noise is regulated by the oscillation of the air column. When the instrument starts to “speak,” these are timed by its pitch. The hypothesis that prompted the present investigation is that though breath noise appears fairly constant in amplitude, periodic vortex shedding suggests a source for spectral modulation.

2 Analysis of Pitch-Synchronous Spectral Change

Recordings were made from a plastic “research” flute with no tone holes. Breath noise was extracted from audibly stable portions of tones and examined for pitch-synchronous features. The extraction method used an adaptive linear pitch predictor which allowed the predicted (periodic) signal to be removed (Cook, P. 1993). Spectral fluctuations in the residual (breath noise) signal were detected via transformation to medium-width frequency bands. These fluctuations were projected against the original waveform to reveal pitch-synchronous and longer-term features.

Pitch-synchronous noise extracted from a bowed cello tone is pulsed in comparison with the flute in Figure 2. The latter was further analyzed for the possibility of fast time-varying spectral changes. The noise residual was transformed with a Hamming window and 64-point Fast Hartley Transform. The analyzed tone had a pitch of 204 Hz and was recorded at a sampling rate of 44100 kHz, resulting in a frequency resolution of approximately 690 Hz. per bin and a temporal resolution...
Figure 1: Sufficiently strong fluid flows encountering an obstacle can produce vortex shedding.

Figure 2: Pulsed noise is detected in the residual (top traces) of a cello waveform (left) but not in a flute waveform (right).

of one third of a period per window. Hopping every 8 samples, 27 transforms were measured per period.

A new display method was developed to detect pitch-synchronous features based on a two-dimensional phase portrait of the oscillation. For each time sample in the original flute tone (not the residual), a point is plotted in two dimensions whose axes are instantaneous amplitude vs. instantaneous amplitude at a given time delay (approximately one-quarter of a period). When many periods are plotted, an average cyclic shape emerges. Stable portions of the duty cycle of the oscillation are sharply defined. Variable, or noisier parts of the cycle are visible as blurred regions. Only the extent of variability is observable and the period-to-period structure within these regions is not resolvable.

To better resolve individual period-to-period differences, the phase portrait was projected onto a time spiral, thereby separating each period. In Figure 3, time begins at the perimeter and spirals inward. One orbit around the graph corresponds to a full period of the waveform. The trace can be colored according to spectral qualities of the residual signal at the same instant in time. RGB color balance is directly controlled by (logarithmic) magnitudes of three bins chosen from the spectrum analysis. Here, something similar is rendered in monochrome by displaying an offset from neutral gray. Lighter / darker variation depicts the magnitude difference between bins 4 and 8 (2400 and 5200 Hz.) of the time-varying frequency transform. Color versions can be seen in (Johnstone, B.).
Angular regions (pie slices) with lighter or darker features indicate changes which are pitch-synchronous. Alternation between regions at differing angles is consistent with the notion of periodic vortex shedding governed by the switching air jet. Longer-term features are evident, where after many periods, the angular position of lighter and darker regions reverses. The presence of multiple quasi-stable regimes suggests modeling the system with chaotic components.

3 Synthesis of Vortex Noise

Detailed three-dimensional modeling of vortex formation is a challenging problem in fluid dynamics. It would be overkill for music synthesis via physical models that are (at the present time) largely one-dimensional. The Exhaust Pipe is an example of an efficient, realistic physical model of the flute based on a one-dimensional waveguide (Cook, P. 1995). A reasonably efficient vortex-like noise circuit has been added to enhance its breath noise component.

Vortex-like behavior can be mimicked by a simple one-dimensional chaotic oscillator. By analogy, its behavior represents the effect of vortex shedding on one point in space as successive whorls pass by, Figure 4.

An iterated quadratic map was chosen as the source of chaotic oscillation. The quadratic’s coefficients were found with an automated search suggested in (Sprott, J.), run over a range of values suited for the fixed-point implementation of the real-time flute model. The values picked can produce a strange attractor; nearby values are rich in widely varying behaviors. Within the region shown in Figure 5, the map can produce a variety of spectral mixtures with differing amounts of periodic and random components.

According to the hypothesis, spectral content of the vortex noise generator needs to be modulated pitch-synchronously. Signal-dependent control of one of the map’s coefficients creates this effect. For the map: $x(n) = a + bx(n - 1) + cx(n - 1)^2$, a unit generator was designed with coefficient
Figure 4: Edge-tone vortex shedding can be simplified by emulating its effect measured at one point in space.

Figure 5: The iterated quadratic map exhibits a variety of behaviors depending on one control parameter. The x-axis indicates control value offset and the y-axis the values visited by the function as it iterates on the given control setting.

$b$ as an input. Varying this term causes the unit generator to traverse the region containing the map's variety of behaviors. Pitch-synchronous modulation of spectral content is accomplished by controlling $b$ with the flute's oscillating signal.

Noise injection is somewhat more complicated than before: originally, output of a random number unit generator was directly mixed with incoming breath pressure. In the present form shown in Figure 6, vortex noise is also allowed to interfere with the operating point of the flute's non-linear excitation function, $ax + bx^2 + cx^3$. The noise very lightly modulates the cubic polynomial's coefficient $a$. Slight changes affect the regime of oscillation of the system, which in turn affect modulation of the vortex generator in a highly complex way.
Breath Pressure

Vortex Generator

\[
N_{t+1} = (-0.6 + 0.1R) - 0.8N_{t} + 2(N_{t})^2
\]

Excitation Nonlinearity

Bore

Pitch Control

Figure 6: A flute physical model (shown as a SynthBuilder patch) has been modified to incorporate vortex noise. First, a vortex noise unit generator is included that implements an iterated quadratic map function whose control parameter \( R \) is signal dependent. Secondly, its noise output \( N \) that is injected along with breath excitation is also used to lightly perturb the excitation nonlinearity.

4 Evaluation of Results

The resulting breath noise is well-incorporated in the tone, as opposed to noise that seems to arrive from another source and is simply mixed with the original. The spectral analysis on the right in Figure 3 shows that the synthetic tone compares well with the natural tone in mimicking pitch-synchronous and longer-term spectral features. Sub-period pitch-synchronous alternation of noise spectra is present, and longer-term shifts are present in the phase position of the regimes. The instrument also seems to speak more easily.

No new performer controls are required. The vortex unit generator is essentially governed by blowing pressure. It is quiet if there is no pressure, it begins hissing at the first onset and then changes to a pitch-synchronous sound after the tone "speaks." Voicing alternatives are possible by varying the amount and quality of vortex noise; these can yield different flute types, levels of breathiness and distance effects.

It is likely the technique may apply beyond air jet instruments, for example to the glottis and other winds such as single, double and lip reeds. Common to the group is the existence of an excitation mechanism that possibly incites some degree of periodic vortex shedding.
References


A HIERARCHICAL SYSTEM FOR CONTROLLING SYNTHESIS BY PHYSICAL MODELING

Perry R. Cook
Stanford CCRMA
prc@ccrma.stanford.edu

ABSTRACT: Sound synthesis using physical models provides many expressive possibilities, with an associated cost that all of the parameters must be controlled, and sometimes parameters interact in ways which are not obvious. The way that a human player solves many of these problems is by listening to (and feeling) their instrument and constantly updating the parameters. Here a software architecture and system is described which contains various objects encapsulating the instrument physics, and the physics, expertise, and perception of the player. Inter-object connections to modify instrument control parameters "on the fly" yield more consistent performances.

1. Introduction

Control parameter values for physical modeling synthesis are difficult to obtain by linear or non-linear system identification techniques because of the rapidly varying nature of transients. Further, since many interesting physical models contain a non-linear oscillator, parameters which work well once may not work the next time the same performance is attempted. The way that a human player solves these problems is by setting up instrument control parameters from memory, then adapting the control parameters based on auditory feedback from the instrument. Synthesis by physical modeling can also benefit from a feedback control system such as this, and can further benefit from a more complete simulation of the physics and expertise of the player. A software architecture, evolved from the experience of creating several physical models and controllers (Cook92,92b), will be described which contains various objects encapsulating the instrument physics, player physics, player expertise, and simple aspects of player perception.

2. Objects In a Complete System

Figure 1 shows a general Performer/Instrument class hierarchy. Blocks will be discussed with examples.

A. Instrument Family Acoustics: The physical acoustics characteristics of a particular instrument family, such as the brass family, bowed string family, etc. Al. Acoustics of Specific Instruments within a family, containing all the relevant tone-producing musical acoustics. These are waveguide synthesis models (Smith87), such as Clarlns, Hoselns, FluteIns, and DSPSinger objects used in the ClariNot, HosePlayer, SlideFlute, and SPASM programs.

B. Instrument Performance Physics Model: Models the physics, other than the acoustics modeled in A, of the instrument. Examples include the mass and damping of the trombone slide or trumpet valve (dynamically modeled), or the length of the guitar fretboard (lookup table) to be used in calculating how long it should take a human arm to slide from one position to the other. The bandwidth required for truly coupling the instrument physics to the performer physics could be quite high, although it is unlikely that this level of simulation must take place at audio rates. Bli. These model a specific member of the family.

C. Performer Physics Model: Contains physical limitations of the player, such as the mass/spring/damping of the arm and hand (Janosy et al. 94). Also contains rules for perceptual/physiological guidance of articulators to targets, using instrument physical descriptions acquired by querying B. For example, the jaw can drop only so quickly, and the tongue body moves slower than the tongue tip. The Singer program now includes mass and damping on each articulator (Cook93), so that each articulator drives to the target point according to an individual time constant. Another case might involve modeling the time required for a flute player’s finger to move, possibly modeling the interaction of fingers within the hand. Cii. If desired, the specific instrument case could be addressed, noting that a skilled player develops muscles and strategies
unique to their specific instrument.

**D. Expert Instrument Family:** Contains an expert’s knowledge for controlling instruments of the same family. Examples of such families include bowed strings of the same tuning characteristics, three valved brass instruments, etc. Such knowledge might include coarse settings for lip tension and valve position to form particular notes on brass instruments. The object MIDIController within the HosePlayer program contains a lookup table of slide and lip parameters which give a coarse setting for each MIDI note number.  

**Dij. Expert Specific Instrument:** Detailed settings for the specific member within the instrument family, such as the lip and breath offsets which differentiate a tuba from a trumpet. The fine details that cause a good player to be comfortably proficient on a very specific instrument. This part should probably be adaptive subject to feedback (Szilas et al. 93)(Wessel91), so that the player object can quickly adjust to a new instrument. Some functionality could be implemented by neural networks (Lee et al. 92).

**E. Perceptual Model:** "Listens" to output of physical model and drives control parameters to achieve the desired output. This object should have the capability of a 'musically astute' listener, but not necessarily any knowledge of the specific instrument. The Expert Objects (D) could be sent messages specifying that the sound is flat, unrich, or too loud. The necessary changes to the control parameters would be processed with input from the Expert Objects, then appropriate messages would be sent to the physical model. The Perceptual Model could be composed of simple time domain pitch detector coupled to a 'classical' expert system, or a full blown model of the ear and brain (when such exists).

3. Implementation of a Simple Player/Instrument System

Figure 2 shows the software and connection architecture of a simple system including many of the elements described above. The PlayerController block does all decision making and processing, and various other blocks respond to queries from the PlayerController. The Expert is a simple lookup table, the Perceptual Model is a simple pitch and power detection program (Cook et al. 92). The link from Player to Instrument is accomplished via a MIDI connection, over which simple MIDI Control Change messages are passed. The Instrument consists of a DSP synthesis instrument controlled by an InstrumentController, whose purpose is to convert MIDI messages into parameter changes for the synthesis instrument. Figure 3 shows the user interface panels for the programs implementing these Performer and Instrument functions.

**4. References**


(Cook et al. 93) P. Cook, D. Morrill, and J. Smith, "An A MIDI Control and Performance System for Brass Instruments," ICMC, Tokyo.


(Janosy et al. 94) Z. Janosy, M. Karjalainen, and V. Välimäki, "Intelligent Synthesis Control with Applications to a Physical Model of the Acoustic Guitar," ICMC, Aarhus.

(Lee et al. 92) M. Lee and D. Wessel 1992, "Connectionist Models for Control of Sound Synthesis," ICMC, San Jose.


INTEGRATION OF PHYSICAL MODELING FOR SYNTHESIS AND ANIMATION

Perry R. Cook
Stanford CCRMA
prc@ccrma.stanford.edu

ABSTRACT: In this project, physical models of a flute are used for both sound synthesis and animation. An exhaustive waveguide flute simulation is used for sound synthesis, including a full tone-hole lattice and filters, coupled vortex noise modeling in the jet simulation, and simple inertial and random models of the individual fingers of the flute player. By combining the waveguide flute synthesis model with a numerical specification of the flute dimensions, ray-tracing animation can be combined with waveguide synthesis to create the visual and sonic experience of "driving" around outside and inside the flute. The result of the project is a 90 second short video entitled "Drive By Fluting."

I. The Synthesis Model

In modeling the flute bore as a series of cylindrical waveguides (Smith87) with tonehole junctions, various methods have been proposed for modeling the fractional delay required to place the toneholes at arbitrary positions along the bore at finite sampling rates. Allpass delay interpolation (Jaffe et al. 83) provides flat frequency response at the cost of some undesired phase properties at high frequencies, and some increased complexity in modeling time varying delay. Lagrange interpolation and deinterpolation (Valimaki et al. 93) provides guaranteed phase response at the cost of attenuation in high frequencies. The flute model in this project uses a total of 8 toneholes, 9 waveguide tubing sections requiring two delay lines each, and one additional delay to model the jet propagation effects (Karjalainen91) (Cook92). This results in a total of 19 interpolated delay lines. Figure 1 shows the worst-case gain attenuation as a function of frequency at 1/2 sample of delay when using Lagrange interpolation, for sampling rates of 22 and 44kHz. Figure 2 shows the worst-case tuning error (assuming Total Delay = Sampling Rate / Fundamental Frequency) as a function of frequency at 1/2 sample of delay, which occurs in using all-pass interpolation, again at 2 sampling rates. The selection is a tradeoff between getting the model to oscillate (gain sensitivity), and play in tune (phase sensitivity). For the model of this study running at 22kHz sampling rate, and with primarily non-time varying delays, allpass interpolation was selected. Lagrange interpolation could likely perform well if a high order interpolator were used at 44.1 or higher sampling rates.

Toneholes can be viewed as three-way waveguide scattering junctions, with appropriate filters to model the low-pass reflection and high-pass transmission properties of an open tonehole. Given some assumptions, such as a fixed observation (microphone, listener) point, that all significant radiation comes from the jet and the first open tonehole (Huopaniemi et al. 94), and that all toneholes are roughly the same size when open, reductions in complexity are possible. By further assuming that the reflection filter of the first open tonehole is substantially similar to the reflection filter of the open end of the flute, a single
commuted filter can be placed at only one point in the model, near the excitation end. Given that the focus of this project is to be able to place a virtual microphone at an arbitrary point inside or outside the flute, no such reductions were exploited. The jet model includes a time-domain simulation of coupled-pulsating turbulence (Chafe95), to more accurately model the noise components of the flute. It is interesting to note here that this coupled-noise model causes the instrument to 'speak' more quickly and reliably. Figure 3 shows the complete sound synthesis model in block diagram form.

One more new component in this model is the addition of low-order filters to model the inertial characteristics of the flute player’s fingers (Cook95). In this project, these are modeled by one-pole filters whose pole positions are randomly selected individually to be between 0.99 and 0.9995 each time the fingering is changed. This way each performance is unique, taking a slightly different time for each 'virtual finger' to raise and lower in opening and closing each tone hole. A more elaborate model of the player's fingers, and even entire hand, could be used in a more exhaustive simulation. For ease of interface and for constructing performances, the flute instrument/player combination program responds to solfege ("do", "re", "mi", etc.) messages automatically, setting the correct targets for the fingers based on a simplified recorder fingering chart placed in a lookup table.

![Figure 3 Block diagram of the flute model used for sound synthesis.](image)

Coaxing such a complex instrument to play at all, let alone play in tune, is a very difficult task. Genetic algorithms have been investigated for music synthesis, applied to FM and Wavetable selection (Horner et al. 93), and to a simplified flute model (Vuori et al. 93). These techniques were exploited in this project to explore the high-dimensional parameter space. The instrument in this study was initially adjusted from physical measurements and first principles, then a genetic algorithm was used to randomize parameters around the initial values, measure the success of such random adjustments (using simple criteria based on whether the instrument oscillated at significant power, and at roughly the correct pitch), select successful values, and iterate the process until suitable values were reached.
The model yields outputs from the jet location, all toneholes, and the end. These outputs are, in reality, directional, and the radiation patterns are frequency dependent. It was not undertaken in this project to model these complex behaviors, and it is considered that this area would be the next logical direction to take in moving toward a more exhaustive model. Each output is placed in a stereo field by simple panning combined with appropriate time delay from each source to each virtual 'ear'. The perceptual results would be further improved by using a binaural model to 'place' the individual sources (Huopaniemi et al. 94).

II. The Animation Model and Specification

Measurements from a bamboo flute were used to construct a 3 dimensional specification for animation. Logical intersections of cylinders were used to construct the basic bore and tone hole chimneys. A hemispherical section was used for the end cap. Two other graphical objects were constructed, modeling a pair of lips and one finger. These objects are not modeled in the acoustical simulations, but are included to add variety and interest to the visual display. The resultant file is an industry standard DXF file. Figure 4 shows one frame containing all of the objects. Although it was not done in this project, some animation software packages are capable of kinematic modeling, and this feature could be exploited to compute the inertial trajectories affecting the speed at which the fingers close the toneholes.

Figure 4 Grey scale rendering of the flute animation model, including a pair of lips and one finger.

Key frames were specified corresponding to significant events from a storyboard, and 3D ray tracing and renderings were computed. The musical selection is a three part canon, and three flutes are used in the final production. The musical canon is shown in Figure 5, and the basic storyboard is shown in Figure 6. The storyline begins with a distant shot of the three flutes, then we (the observer) zoom in to the second flute, then closer to the 5th tonehole. An attempt is made to enter the flute via the tonehole, but an interfering finger is encountered and we are pushed away by the collision. We pan around the flute end and zoom to the lips of the player, where we find ourselves sucked into the vocal tract of the flutist, but are are promptly coughed out to a very distant observation point. We zoom in once again, realizing that we can enter the flute via the hole at the end. We drive along inside the flute to the position of the 2nd tonehole, then we exit and zoom away to infinity as the canon ends.

Figure 5 Musical canon used for animation soundscore.

III. Future Directions

It is likely that today on existing (albeit expensive) hardware, this entire simulation could run in real time under user control. It is a certainty that the model as implemented in this project would run in real time on future, less expensive, hardware/software systems. There is much that could be added to the model to make it more realistic, both in visual and aural domains. In a real-time system, a 3D joystick or control
glove using standard Virtual Reality control gestures could be used to fly around the virtual flute world at will. Better animation techniques and software are obvious places for improvement. The next planned project involves a brass instrument based on the TBone model (Cook91).

Figure 6 Animation storyboard. Keyframe times are specified in measure numbers.

IV. References

Performance Expression in Commuted Waveguide Synthesis of Bowed Strings

DAVID A. JAFFE and JULIUS O. SMITH
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University, Stanford, CA 94305
Tel: (415) 723-4971; email: daj@ccrma.stanford.edu / jos@ccrma.stanford.edu

ABSTRACT: In [Smith 1993], an approach was described for implementing efficient real-time bowed string synthesis. Recent work has focused on differentiating the members of the violin family, as well as on the flexibility necessary to create expressive performance. This paper presents a technique for creating smooth transitions between notes, enabling a variety of bowing styles to be synthesized, such as legato, marcato and martele. A method for supporting such left-hand techniques as vibrato and glissando is also given, as is the efficient simulation of pitch-correlated bow noise. Examples from various periods of music history have been convincingly synthesized in real time using the Music Kit and DSP56001 under NEXTSTEP.

1. Motivation—Better Bowed and Plucked String Instruments

The current de facto standard in string synthesis is a MIDI keyboard controlling a sampler. There are two main problems with this approach. First, the keyboard model is inadequate to represent the continuity possible with stringed instruments. Second, sampling is notoriously inflexible when it comes to expression and nuance.

1.1 Efficient Physical Models

Physical modeling side-steps the problems of sampling by beginning, not with an acoustic waveform, but with the physics of the sound-producing mechanism itself. Physical modeling parameters are “just right”, in that they correspond to real-world performance parameters and allow for a great deal of expressive variety in the context of a single perceived source [Jaffe, 1995].

Physical modeling of bowed strings in the past has suffered from great computational expense (as in the case of finite-element modeling) [Ruiz 1971]. In the mid-80s Smith developed a violin model based on non-linear bow dynamics [Smith 1986]. This model, while extremely effective, was still fairly expensive because it required a high-order filter to model the body of the violin. It also suffered from sensitivity to parameter values—if the parameters were not precisely right, it might not even oscillate at all. In 1993, Smith presented an alternative model in which the body filter is permuted with the string and combined with the excitation function, greatly reducing the computational expense of the model and removing the problem of the parameter sensitivity [Smith 1993].

Similarly, the plucked string synthesis technique originally described by the authors [Jaffe/Smith, 1982] can be improved through the use of this refinement. The new version is capable not only of representing a specific type of plucked string instrument (e.g. a viola), but even a particular instrument (e.g. an Amati with a 19th century top and an open crack in the back.)

A detailed discussion of the bowed and plucked string models is beyond the scope of this paper. Instead, we provide an overview, focusing on specific techniques to produce expressive effects. For clarity, we discuss only the bowed string, though most of these techniques apply equally to the plucked string.

1.2 Expressing the Connection Between Notes

Before synthesizing expressive musical phrasing, we need a language to depict it. Following the example of the NeXT Music Kit [Smith, Jaffe, and Boynton, 1989], we define a “phrase” in a manner slightly different from its conventional musical meaning. Our phrase consists of a series of connected notes on a given “voice”, where each voice is capable of producing one simultaneous note at a time. A phrase begins with a noteOn event. If another noteOn is received, it is considered a “smooth rearticulation”. In the case of the violin, this means that either the bow was changed or the left hand fingering was changed or both. If a noteOff is received, the phrase begins its concluding portion, which takes a certain amount of time. If no noteOn is received before the concluding portion has ended, the phrase is considered finished. If, on the other hand, a noteOn does arrive before the concluding portion is finished, the noteOn is again considered a “smooth rearticulation”. Phrases are delineated by one or more samples of silence in a given voice. In MIDI parlance, a phrase corresponds to a series of overlapping notes on the same MIDI channel, with the synthesizer in MIDI Mono Mode.

2. Bowed String Model

The bowed string model consists of three computational blocks, the “bow”, the “string”, and a
"pitch/vibrato" module. The pitch/vibrato module, controls both the bow and the string. The bow feeds its output to the string, which in turn produces the sound.

2.1 Bow

The heart of the bow is a periodically triggered excitation table ("PET"), where the excitation table is derived (off line) from measurements of real instruments. The PET module plays out the excitation table, jumping back to the beginning of the table once per period of sound, similar to the VOSIM and Chant vocal synthesis techniques [Roads 1989]. When the excitation table is longer than the pitch period, overlaps result. These can be managed by a "multi-PET controller" which triggers a "sub-PET" each period, cycling through a given maximum number of sub-PETs. The outputs of the overlapping sub-PETs are added together to form the overall output. For proper tuning, the multi-PET controller "counts down" an extended precision period duration (in samples) having a fractional part. The period length may in effect be rounded to the nearest sample when an excitation table is triggered to avoid the expense of interpolating the samples in the table.

The bow (or pick) position can be represented as an all-zero comb filter that simply subtracts a delayed version of the PET output, with the length of the comb filter corresponding to the distance from the bridge [Jaffe and Smith 1983]. The effect of bow position illusion is further strengthened by simulating the burst of noise that is produced when a string slips under a bow [Chafe 1990]. This noise occurs naturally when the string is slipping under the bow, and its duration per period is obtained by multiplying the period by the distance from the bow to the bridge divided by the string length. In principle, the string-slip noise must be convolved with the impulse response of the instrument body. In our implementation, when the PET controller triggers a new period, it also triggers a bow-noise pulse and provides it with a random number to be used as an amplitude scaling for the entire pulse. The envelope of this pulse consists of a rapid rise followed by a slower decay, mimicking the effect of the convolution that occurs in the real-world case. As a rule of thumb, we found that a decay value of approximately five times the bow position works well. We then pass that signal through a one-pole low-pass filter for control of musical dynamics.

For plucked strings, the PET simply operates in a one-shot mode, e.g., by giving it a desired pitch of 0.

2.2 String

The string portion of the model is a digital waveguide string [Smith 1992] in which a delay line holds approximately one period of sound at all times. The output of the delay line is lowpass filtered, summed with the bow output, and fed to the delay line input. There are three aspects of the string filter that affect the resulting timbre: frequency-independent attenuation, frequency-dependent attenuation and frequency-dependent dispersion [Jaffe/Smith 1983]. For bowed strings, frequency-independent attenuation is the most important of the three, and it controls the sustain quality of the string.

Our strategy for producing smooth glissandi without discontinuities proceeds as follows: We pre-allocate a delay memory block long enough to represent the lowest possible pitch. The delay length, a time-varying signal, controls the offset of the read pointer with respect to the write pointer. If this offset does not fall on a sample boundary, the delay module computes an interpolated value. The interpolation also has the effect of allowing arbitrarily fine tuning. We use simple linear interpolation. Due to the presence of vibrato, the differences in filtering by the linear interpolation for different notes is not significant and amounts to a slight low-pass filtering on average. This slight time-varying filtering effect in the interpolator can be compensated in the string loop filter, or allpass interpolation can be used.

For legato transitions, we cannot abruptly change the length of the delay, because this will cause a sharp discontinuity, which will then be recirculated, resulting in a "spurious pluck." Instead, we branch the string into two different length strings, representing the two pitches. Two delay pointers read the same delay memory, with the pitch period of each driven by its own time-varying offset signal. The output of these readers is then fed to a module that interpolates between the two delay outputs, driven by a cross-fade shaped like a half of a cosine cycle. Before the beginning of a legato transition, the first reader is at the old pitch period and the cosine ramp is at 0. At the start of the legato transition, the second reader is set to the new pitch period and the cosine ramp moves from 0 to 1. For the next legato transition, the process is repeated in reverse. Note that this method is applicable not only to string synthesis, but also to commuted synthesis of any quasi-periodic tone [Smith 1993]. Note also that the highly non-physical nature of this legato algorithm is due to the non-commutativity of the time varying string with the body resonator.

Glissandi and legato transitions done in this way have implications for the bow position filter. To keep a constant tone, violinists tend to keep the bow at a distance from the bridge that is a fixed percentage of the string length. This helps maintain a consistent timbre as the string shortens. In our implementation, this means the comb filter needs to be shortened as the string is shortened so that the zeros continue to fall in the
same place relative to the harmonics of the waveform. While this can be done by using a branched interpolating delay similar to that used in the string, we did not want to pay the extra computational expense. Instead, we simply accept the change in timbre. In fact, changing the bow position slightly between phrases is an effective way to add variety and color.

One last subtlety: When the phrase starts, it is essential to do something to make sure that the garbage left over in the delay memory does not get incorporated into the new note. An efficient approach is to use a switch to break the feedback path of the string. A new phrase begins with this path disconnected. Then, after one period, when the delay line is filled with valid samples, we enable the string feedback path.

2.3 Frequency Control
Since both the bow and string require frequency control, it is convenient to factor it out into its own computational block that produces a signal specifying the instantaneous pitch period of the sound to be synthesized. It combines periodic and random vibrato components with a fundamental frequency which may be enveloped to produce glissandi. A high-quality oscillator should be used to produce the vibrato or noticeable quantization in the frequency signal will result, causing a rough sound in the string. The random vibrato can be produced with a one-pole filtered noise generator.

For legato transitions, the pitch/vibrato controller produces two pitch periods, corresponding to the frequencies of the old note and the new note. Each of these is used to control one branch of the string, as described in section 2.2 on legato transitions. Such a split is not required for the PET. If the pitch period becomes shorter, the multi-PET controller simply starts triggering excitation tables at a faster rate. The question does arise, however, as to when in the course of the legato transition the multi-PET controller should start triggering at the new rate. We found that changing the PET frequency at the midpoint of the transition produced the best results.

Transition times between 15 and 30 milliseconds are most effective. Transition times less than about 10 milliseconds cause audible thumps which sound like the finger of the left hand hammering down on the string, causing an injection of some energy into the string. Transition times greater than 50 milliseconds cause a noticeable inharmonicity consisting of the least common multiple of the two frequencies. The real-world analog would be a forced-oscillation of a string, where the oscillator is at a different frequency from the string.

3. Learning to Play the Synthetic Violin
Making an expressive bowed string score using this model draws upon the same experience needed to coach a human violinist. Thus, a knowledge of string technique and style is invaluable. We discuss a number of examples taken from violin performance practice and show how they can be represented.

3.1 Shifting the Position—The One-Finger Case
When a violinist needs to go from a low position of the left hand to a higher position, or vice versa, he or she performs a shift. A simple glissando, coupled with a bow amplitude envelope that has a notch on rearticulation, simulates a one-finger shift. A typical amplitude envelope is shown here in Music Kit ScoreFile notation in which the breakpoints are specified as a list of (x,y) pairs, where x = breakpoint time and y = target amplitude:

\[
\begin{align*}
&[(0,0)(.15,1)]; \quad \text{// For the first noteOn} \\
&[(0,0)(.05,0.1)(.15,1)]; \quad \text{// For the rearticulation}
\end{align*}
\]

The instrument receives a noteOn and begins playing the first envelope. After 0.15 seconds, the amplitude envelope is at 1.0. Some time later, another noteOn occurs. The amplitude envelope moves to the rearticulation point (an amplitude of 0.1) in 0.05 seconds. Then it moves to the stick point (1.0) in another 0.1 seconds. This creates a notch—the amplitude is reduced during the shift. The following scorefile excerpt performs the shift:

\[
\begin{align*}
&\text{vln (noteOn,1) ampEnv:}[(0,0)(.15,1)] \text{ freq:a4; } \\
&t+1; \quad \text{// Advance time} \\
&\text{vln (noteOn,1) ampEnv:}[(0,0)(.05,0.1)(.15,1)] \\
&\text{freqEnv:}[(0,0)(.15,1)] \text{ freq0:a4 freq1:g4;}
\end{align*}
\]

3.2 Shifting the Position—the Two-Fingered Case
There are two kinds of two-finger position shifts, the traditional type, which is by far the most standard, and the French impressionist type, used in the music of Debussy and Ravel. Both can be implemented as a combination of a glissando and a legato transition. The difference between the two
methods depends on the order of the glissando and legato transition. The traditional method proceeds as follows:

1. Lighten pressure on the currently held left-hand finger.
2. Slide the finger to the new position.
3. Press a new finger down on the finger-board.

This can be implemented as a glissando transition with bow amplitude notch, followed by a legato transition.

In contrast, the French type proceeds as follows:

1. Lighten pressure on the currently held left-hand finger.
2. Press the new finger down on the finger-board.
3. Slide the finger to the new position.

This can be implemented as a legato transition, followed by a glissando transition with bow amplitude notch.

3.3 Bowing Styles

Bowing styles are conveyed largely by the bow amplitude envelope, which tends to be somewhat differently shaped from conventional computer music usage. This is because the actual amplitude envelope of the bowed string model is the that of the string output which resonates at the bow signal frequency. As an example, a simple rectangular envelope for the bow signal produces an exponential rise, steady state, and exponential decay in the final output, and in fact sounds quite natural for short envelope durations. As a result of this indirectness of amplitude control, accents and bow bite must be apparently exaggerated in the bow envelope. For example, a marcato bow stroke might appear to have three times the ordinary amplitude during the attack portion: \((0, 2)(1.3)(1.5, 0.8)(1.8, 0)\);

For bouncing bow strokes, the string is allowed to free-ring as soon as the bow leaves the string. Therefore, an envelope that sustains at a non-zero level until the noteOff occurs is inappropriate.

Here is an example spiccato envelope: \((0, 0)(0.1, 0.1)(0.25, 1)(0.35, 0.5)(1.5, 0.5)(1.2, 0)(1.3, 0)\);

This envelope is also suitable for rapid sautille bowing, a style that works especially well with the bowed string model, since each bow stroke injects energy into the string, interacting with the energy from the previous bow stroke.

A standard broad bow stroke is the martele: \((0, 0)(0.0175, 0.6)(0.025, 1.2)(0.2, 1.2)(0.25, 0.8)(1, 0)\);

Another important bowing style parameter is bow position, which may be adjusted to give sul ponticello or sul tasto effects.

3.4 Legato String Crossing

The most accurate approach to legato string crossings is to allocate a separate bow module and string module for each virtual string. These can share the same pitch/vibrato unit. However, if chords and double stops are also to be supported, a complete bowed string model is needed for each virtual string.

4 References


PadMaster: an improvisation environment for real time performance

Fernando Lopez-Lezcano
CCRMA (Center for Computer Research in Music and Acoustics), Stanford University
(nando@ccrma.stanford.edu)

ABSTRACT: This paper will describe the design and implementation of PadMaster, a real-time improvisation environment running under the NextStep operating system. The system currently uses the Mathews/Boie Radio Drum as a three dimensional controller for interaction with the performer.

1.0 The Radio Drum and the MIDI communication protocol

The current implementation of the Stanford Radio Drum was developed by Max Mathews as a simpler alternative to Boie’s design. The two batons act as radio transmitting antennas. There are five receiving antennas underneath the surface of the drum and the multiplexed A/D converter translates signal strength coming from the five receivers to numbers, which the microprocessor uses to calculate the absolute position of each baton in space. In addition to the batons, the Radio Drum hardware includes two switches and four potentiometers. It has a MIDI interface that it can use to communicate with computers or synthesizers.

The existing general purpose controller program (written by David Jaffe / Andrew Schloss) was completely redesigned. A more efficient and faster protocol was created to enable the computer to use the Radio Drum as a three dimensional controller with six degrees of freedom. This program is one of many that are stored in the Drum’s firmware, and can be activated remotely with a system exclusive message. This is a short description of part of the protocol:

- System exclusive configuration messages: can be used to turn ON or OFF the communication program, set the MIDI channel used by it, dump and load the internal calibration tables, etc.
- Trigger / Release messages: sent by the drum when a baton hits / leaves the surface using continuous controllers 26 through 31. The message includes the x-y position and velocity of the hit or release.
- Switches: sent by the drum when one of the switches changes state using controllers 5E to 5F.
- Poll request: sent by the computer to request the position in space of the batons (channel pressure).
- Poll answer: sent by the drum in response to a poll request message using a series of channel pressure messages. It includes the position of both batons in space and optionally the position of the pots.

2.0 The PadMaster program

PadMaster is written in Objective C and runs on the NeXT workstation, which is connected through MIDI to the Radio Drum. It uses the Radio Drum as a controller and splits the surface of the drum into up to 30 virtual pads, each one independently programmable to react in a specific way to a hit and to the position information stream of one or more axes of control. Pads can be grouped into Scenes, so that the behavior of the surface of the drum can be subtly or radically altered during the course of a performance by dynamically jumping to a different Scene. The screen of the computer displays the virtual
surface and gives visual feedback to the performer on the state of all the pads in the current Scene. The workstation is also connected through MIDI to one or more synthesizers. The virtual pads can be split in two types depending on their function: Performance and Control.

2.1 Performance Pads

Performance Pads can be individually programmed to control the playback of MIDI sequences, note generating algorithms or soundfiles. The graphical representation of the pads on the screen gives instant visual feedback to the performer. Pads change color and status messages dynamically according to their state. A performance pad that is playing remains active even if its corresponding Scene is not currently selected.

2.2 Control Pads

Control Pads are used to trigger actions that globally affect the performance of a Scene. A pad can be programmed to change the current Scene when hit, thus redefining the behavior of the whole surface of the drum. Control pads can also be used to pause, resume or stop all playing pads in the currently selected Scene.

3. Inside a pad

Editable parameters inside each pad can be changed through a standard NextStep inspector window with several editing panes. The first pane can be used to select the action that is executed when the pad is hit. The possible actions include starting / pausing / resuming a sequence, starting a new overlapping sequence or playing the next note of a sequence. It also selects the MIDI port and channel for the pad and allows editing a graphical mapping of hit velocity to note velocity of the played sequence. The second pane edits the tempo options. Tempo can be global, per pad or per sequence inside a pad (when there is more than one instance of a sequence playing). There is a tempo envelope and it is also possible to control tempo with the hit velocity or with any of the six available axes of continuous control. The third pane lets you associate up to three continuous MIDI message streams (pitch bend, pressure or any controller) to the position of up to three of the six axes of control. All these mappings are created through graphical function editors. The fourth pane edits the sequence of notes that are played when the pad is hit. The sequence is expressed as a normal MusicKit format scorefile.

4. PadMaster in performance

PadMaster has been used to compose and perform "Espresso Machine", a piece for PadMaster and Radio Drum, two TG77's and processed electronic cello (Chris Chafe, playing his celletto). The piece is an environment for improvisation in which the PadMaster and celletto performers exchange ideas and play with predetermined materials. The piece is composed in three PadMaster Scenes, each with several groups of related materials that are triggered during the performance. One baton is reserved for triggering pads and the other for continuous three dimensional control.

5. Future developments

PadMaster is currently undergoing a complete rewrite to implement new and improved functionality. This includes resizable pads, triggering of algorithms and soundfiles, multiple computers running remote objects for increased capability (in an ethernet environment), pads with inheritance (useful to be able to group related pads) and a completely rewritten editing paradigm that resembles a database with multiple views.

References:

[2] Carlos Cerana (composer) / Adrian Rodriguez (programmer), MiniMax, a piece for Radio Drum
Digital Waveguide Modeling of the Non-Linear Excitation of Single-Reed Woodwind Instruments

Gary P. Scavone
Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University, Stanford, CA 94305
gary@ccrma.stanford.edu

ABSTRACT: This paper reviews past digital waveguide methods for reproducing non-linear "reed" excitations as well as introducing a new method incorporating reed dynamics. This model is based on a mass-spring-damper system and a non-linear flow control mechanism. In this way, a more physical system is attained which provides better approximations to the control parameters of real musical instruments. Further, it is shown that this model of the reed can be modified to represent the lips of a brass player, and the similarities and differences between these two systems are briefly examined.

1 Introduction

Musical instruments are most clearly distinguished from one another by their transient characteristics, which in turn are defined by a particular method of excitation. Among wind-blown instruments, for example, the various air-driven excitation methods distinguish saxophones from trumpets or flutes. In the context of digital waveguide modeling of musical instruments, it is these same non-linear excitation methods that have proven the most difficult to accurately reproduce. Two effective digital waveguide reed woodwind excitation methods have previously been presented (Smith 1986) (Cook 1992), though these models lack the physical control parameters associated with their real world counterparts. A dynamic waveguide reed model incorporating a mass-spring-damper system and non-linear flow control is presented here. The modeling of the reed in this way for waveguide applications was previously discussed in (Hirschman 1991) for woodwinds and (Cook 1991) for brasses, though the flow control mechanisms implemented were different from that discussed in this paper.

2 Acoustical Aspects of Air-Driven Reed Generators

A variety of acoustical studies of reed-like oscillating systems have been performed (Fletcher and Rossing 1991). Two fundamentally different types of reed generators exist – those in which the reed valve is initially closed and then blown open (as with a brass player's lips) and those in which the reed valve is initially open and then blown closed (as in clarinets and saxophones). In most cases, the reed is modeled using a linear mass-spring-damper system, and the pressure on the reed is taken equal to the difference in oral cavity and bore pressures. The position of the reed in turn governs the flow through the reed aperture, for which Bernoulli's flow equation forms a first approximation. In brass instruments, the lips oscillate close to their resonance frequency, but in woodwind instruments the reed resonance is high compared to the fundamental "driving" frequency, and the effect of the mass is often neglected. Possible modifications to the flow equation include terms to compensate for reed channel inertia and the physical motion of the reed surface. Recent fluid-dynamic studies of flow through a reed aperture have suggested the need to account for viscous flow (Hirschberg et al. 1990). Non-linearity of the reed stiffness has also been discussed (Gilbert et al. 1990).
3 Digital Waveguide Reed Generator Models

3.1 The Time-Varying Reflection Coefficient

The time-varying reflection coefficient is usually implemented in terms of a look-up table. The model is a memory-less system, whereby the reed/bore boundary is characterized by a pressure dependent reflection coefficient. The reflection coefficient is assumed to vary in response to the difference in oral cavity \( (p_{oc}) \) and bore pressures \( (p_b) \).

Assuming continuity of volume velocity at the reed/bore junction,

\[
\frac{p_a}{z_a(p_{\Delta})} = \frac{p^+_b - p^-_b}{z_b}, \quad p_{\Delta} = p_{oc} - [p^+_b + p^-_b]
\]  

(1)

Defining the reflection coefficient

\[
\rho(p_{\Delta}) = \frac{1 + r(p_{\Delta})}{1 - r(p_{\Delta})}, \quad r(p_{\Delta}) = \frac{z_b}{z_a(p_{\Delta})}
\]

(1) can be solved for the reflected bore pressure at the junction, \( p^-_b \):

\[
p^-_b = \rho(p_{\Delta})p^+_b + \frac{1 - \rho(p_{\Delta})}{2}p_{oc}
\]  

(2)

By defining a new term, \( p^+_\Delta = p_{oc} - 2p^-_b \), which is independent of \( p^-_b \) and substituting into (2), we obtain

\[
p^-_b = \frac{p_{oc}}{2} - \rho(p^+_\Delta)\frac{p^+_\Delta}{2}
\]  

(3)

3.2 The Reed Reflection Polynomial

The reed reflection polynomial incorporates the concept of a time-varying reflection coefficient and the dependence of this coefficient on the difference between oral cavity and bore pressures. The essential simplifying assumption is that \( p_{\Delta} \) can be approximated by \( p^+_\Delta \). The polynomial model is derived by considering the reed/bore junction as shown in Figure 1. The portion of \( p^+_\Delta \) reflected back into the bore is given by \( p_b^- \cdot \rho(p_{\Delta}) \), while the portion of the oral cavity pressure which is transmitted into the bore is given by \( p_{oc}(1 - \rho(p_{\Delta})) \). Then \( p_b^- \) is given by:

\[
p^-_b = p^+_\Delta - [p^-_b - p^-_b \rho(p_{\Delta})]
\]  

(4)

Using the above stated approximation for \( p_{\Delta} \) and approximating \( \rho(p_{\Delta}) \) by a second order polynomial function, Eq. 4 becomes:

\[
p^-_b \approx p^+_\Delta - [c_1(p^+_\Delta - p^+_b) + c_2(p^+_\Delta - p^+_b)^2 + c_3(p^+_\Delta - p^+_b)^3]
\]  

(5)

This reed implementation method has proven efficient and effective for real-time DSP synthesis. Unfortunately, the process of determining appropriate polynomial coefficients is rather arbitrary.
3.3 The Dynamic Reed Model

We model the reed as a linear mass-spring-damper system which is acted upon by the difference in oral cavity and bore pressures, as in Figure 2. Within the complete instrument model, the reed functions as a pressure controlled valve. Thus, the reed is not directly "attached" to the waveguide bore system. Rather, we simply desire the reed displacement at any given instant in time so that the volume flow through the reed valve may be computed and injected at the reed/bore junction. The relationship between applied force and displacement, and the corresponding Laplace transform is given by

\[ F_r(s) = \frac{ms^2 + \mu s + k}{ms^2 + \mu s + k} X(s) \]  

where

Both the oral-cavity and the bore pressures act upon the reed, so that the resultant force on the reed is

\[ F_r(s) = A \cdot P(A(s)) = A \cdot [P_{oc}(s) - P_b(s)] \]

where \( A \) is the approximate surface area of the reed exposed to \( P_{oc} \). \( A \) is typically bounded by the width of the reed at its tip and the distance from the reed tip to the player’s lower lip.

The transfer function that relates reed displacement to applied force is found from (6) as

\[ \frac{X(s)}{F_r(s)} = H(s) = \frac{1}{ms^2 + \mu s + k} \]

Using the bi-linear transform to convert from continuous to discrete time, the following digital transfer function results,

\[ \frac{X(z)}{F_r(z)} = H(z) = \frac{1 + 2z^{-1} + z^{-2}}{(1 + \alpha z^{-1}) + (1 - \alpha^2 + \alpha) z^{-2}} \]

where \( \alpha \) is the bilinear transform constant used to control the frequency warping. The displacement found by passing \( A \cdot P_{oc}(t) \) through this biquad section is subtracted from the equilibrium reed aperture spacing to produce the reed displacement. Inelastic beating of the reed is assumed, where the reed is forced against the lay and held there until the driving force decreases below that of the stretched spring. The digital filter of (9) must be reinitialized with the appropriate initial conditions each time this occurs.

Assuming that the Bernoulli flow equation applies to the given situation, the volume flow through the reed aperture is found from

\[ u(t) = A_r(t) \cdot v(t) = A_r(t) \cdot \left( \frac{2p_{oc}(t)}{\rho} \right)^\frac{1}{2} \]

where \( A_r(t) = w \cdot x(t) \) is the time-varying area of the reed aperture, \( w \) is the width of the reed, and \( \rho \) is the density of air.

Finally, we assume continuity of volume velocity at the reed/bore junction and calculate the new traveling wave component of pressure entering the bore as,

\[ u(t) = u^+(t) + u^-(t) = \frac{p^+(t) - p^-(t)}{Z_b} \]

Figure 2: Dynamic Reed Model
where $Z_b = \rho c/A_b$ is the constant acoustic characteristic impedance of the bore. $p_b^-(t)$ represents the traveling wave component of pressure entering the bore while $p_b^+(t)$ represents the traveling wave component of pressure leaving the bore. Solving for $p_b^-(t)$, we have

$$p_b^-(t) = u(t) \cdot Z_b + p_b^+(t)$$

$$= A_r(t) \cdot \left[ \frac{2p\Delta(t)}{\rho} \right]^{\frac{1}{2}} \cdot \frac{\rho c}{A_b} + p_b^+(t)$$

$$= A_r(t) \cdot \frac{c}{A_b} \left[ 2p\Delta(t) \right]^{\frac{1}{2}} + p_b^+(t)$$

(12)

Measurements on a clarinet reed and mouthpiece (Backus 1963) have shown the exponent value in (12) to be on the order of $\frac{2}{3}$. This calculation can be simplified for real-time DSP implementation by the use of a look-up table.

### 4 Results & Future Refinements

The dynamic reed model presented here has been successfully implemented in digital waveguide single-reed woodwind instrument models and produces realistic transient and steady-state behaviors. A few improvements in progress include developing an efficient method to achieve time-varying control over the reed parameters (mass, spring constant, and damping), and exploring the effect of elastic collisions of the reed against the lay. Though not explicitly shown, this model can also be used to represent the lips of a brass player. The model as outlined in Section 3.3 remains intact, though the equilibrium reed aperture spacing is set to zero and no beating occurs. Further, the playing frequency of the system will roughly correspond to the resonance frequency of the lip model.

### References


Commuted Piano Synthesis

Julius O. Smith III and Scott A. Van Duyne
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University, Stanford, CA 94305
jos@ccrma.stanford.edu, savd@ccrma.stanford.edu

ABSTRACT: The "commuted piano synthesis" algorithm is described, based on a simplified acoustic model of the piano. The model includes multiple coupled strings, a nonlinear hammer, and an arbitrarily large soundboard and enclosure. Simplifications are employed which greatly reduce computational complexity. Most of the simplifications are made possible by the commutativity of linear, time-invariant systems. Special care is given to the felt-covered hammer which is highly nonlinear and therefore does not commute with other components. In its present form, a complete, two-key piano can be synthesized in real time on a single 25MHz Motorola DSP56001 signal processing chip.

1 Introduction

In [11, 6], techniques were described for simulating plucked, struck, and bowed string instruments in which the resonating body of the instrument is commuted with the string in order to eliminate the high order digital filter which is otherwise necessary to simulate body resonances. In this technique, the string excitation (e.g., the "pluck" or "strike" force over time) is pre-convolved with the body impulse response, and the resulting waveform can be stored in a wavetable. To synthesize a plucked or struck string tone, the wavetable is simply "played into" the string. The resulting sound is that of a plucked or struck string with all the body resonances of the natural instrument, since they are pre-installed in the excitation. The sound quality is excellent for linearly plucked and struck strings. For bowed strings, the quality is quite good for smooth bowing styles that do not involve too much bow force. This paper describes an extension of this technique to the piano using a linearized model of the piano hammer which depends on striking velocity.

2 The Piano

The piano is an example of a nonlinearly struck string. It is simple in some ways and highly complicated in others. It is simple in that only the hammer velocity matters as a control variable when the string is struck—the finger that presses the key has no significant mechanical connection to the hammer after it is launched into flight toward the string. That means MIDI, for example, provides a sufficient representation for piano performance, and the dimensionality of control (aside from pedals) is confined to one degree of freedom per key, per time instant—the so-called "velocity" parameter.

2.1 String

Piano strings are fairly simple because they are uniform, tightly stretched, and nearly rigidly terminated. As a result, they are highly linear under normal playing conditions. The digital waveguide approach to string modeling [10] therefore works very well for the individual piano strings. The non-negligible stiffness of piano strings poses an increase in the expense of the implementation, resulting in the need for an allpass filter in the "string loop". Allpass filters of order 4—6 or more are
required for good results [7, 14]. Another complicating factor is the non-negligible coupling between strings that are hit by the same hammer [17]. There is also significant coupling among all the strings when the sustain pedal is down. To fully simulate the linear behavior of each string, it is necessary to couple [11] at least three digital waveguides together corresponding to the main types of wave propagation in and along the string (two transverse and one longitudinal). In key ranges in which the hammer strikes three strings simultaneously, nine coupled waveguides are required per key for a complete simulation. This paper will address the case in which only the vertical plane of vibration is simulated for each string, and only one string is implemented per key. In a reasonably high quality implementation, at least the correct number of strings should be implemented, since they are detuned and cause important beating and after-sound effects [17]. However, since extending the present discussion to multiple strings is straightforward, only the single-string case will be treated here.

2.2 Resonator

The soundboard and enclosure as a whole are simple in that they are largely linear, time-invariant components, but they are complex in that they are large. Large vibrating objects generally have many more resonant modes in the range of human hearing than do small objects. Also, waveguide propagation in the soundboard and enclosure is not confined to one dimension as it is in a string. That means a complete digital waveguide model of the piano would require two- and/or three-dimensional waveguide meshes [13] to model the resonating soundboard and piano enclosure. In sum, the sheer size of the piano and its soundboard lead to very expensive direct modeling techniques, even after accounting for the fundamental efficiency advantages of the digital waveguide approach. However, the commuted synthesis technique described below bypasses this difficulty and allows simple “sampling” of the soundboard/enclosure impulse response into a read-only memory which is “played” into the string in a manner modified by the hammer-string collision. The same technique applies equally well to the huge bank of sympathetically vibrating strings obtained when the sustain pedal is down [16].

2.3 Hammer

A more seriously complicating factor is the piano hammer. While only its velocity is necessary to specify its state completely prior to hitting the string, the string collision is highly nonlinear [12]. The nonlinearity comes from the felt covering the hammer: As it compresses, it acts like a spring whose spring-constant is rapidly increasing. Also, the hammer-string interactions are a function of both string and hammer motion, giving potentially complex cases such as the string hitting the hammer a second time after it has already fallen away from the initial strike. Apart from the hammer, the entire instrument can be very well approximated by a linear model. The main difficulty with nonlinearity in this context is that it prevents use of the commuted synthesis technique at first sight. This is because commutativity of system elements is only possible in general for linear, time-invariant elements. In previous work, we could relax the time-invariance requirement to some extent to allow for string vibrato. However, relaxing linearity is much more problematic, especially when the nonlinearity is this severe.

3 Commuted Piano Synthesis

It turns out commuted piano synthesis is possible with both high fidelity and low computational cost in spite of the nonlinear behavior of the hammer-string interaction.

The key observation is to note that the interaction between the hammer and string consists essentially of a few discrete events per hammer strike when the string is initially at rest. That is, the hammer-string interaction can be approximated as one or a few discrete impulses which are filtered using
filters which depend on the hammer-string collision velocity. Figure 1 illustrates the qualitative behavior of the striking force [2, 12]. In this example, the three peaks in the force curve indicate that the hammer stays in contact with the string long enough for return pulse waves from the agraffe to compress the felt two more times before the hammer falls away the string. Also indicated in the figure are where three impulses may be located in order to synthesize the waveform as a superposition of filter impulse responses.

![Figure 1: Example of overlapping hammer-string interaction force pulses. The vertical lines indicate the locations and amplitudes of three single-sample impulses which may be passed through small digital filters to produce the overlapping pulses shown.](image1)

To a large extent, the number of interaction impulses is determined by which string is being struck. Thus, given key number and hammer velocity, one can predict the amplitude and timing of all interaction impulses, for a string initially at rest. There is a slight unpredictability which we neglect having to do with the fact that when the hammer strikes an already vibrating string, the entire history of string vibration influences the exact details of the hammer-string interaction; however, this is a second-order effect which may not even be desirable. We still retain the superposition of the new strike response with any existing vibration, thus preserving the naturally varied colorations of successive strikes on a single string as is characteristic of a physical modeling technique. If unpredictability of the force pulses on restriking is deemed important, one may use random perturbations of the interaction impulse levels as a function of amplitude of vibration prior to the hammer strike. For greatest precision, of course, a rigorous piano hammer model may be run in parallel to compute the hammer-string interaction force in real time as a function of their relative velocities [1, 2, 12, 15].

The creation of a force pulse from a single impulse for a specific dynamic level is shown in Fig. 2. Without loss of generality, we consider force units; dividing by the string wave impedance gives the corresponding velocity injection for the string loop. Other physical variables may be chosen, as discussed in [10].

![Figure 2: Creation of a single hammer-string interaction force pulse as the impulse response of a lowpass filter. The input to the filter is a single nonzero sample (impulse), and the output is the desired hammer-string force pulse. When the amplitude of the input impulse increases, the output pulse increases in amplitude and decreases in width, which means the filter is nonlinear. However, on each specific impulse, the filter operates as a normal linear, time-invariant filter. In this way, the entire piano is “linearized” with respect to each fixed hammer velocity.](image2)
3.1 Illustrative Implementation

One method of creating multiple force pulses from multiple impulses at a specific dynamic level is shown in Fig. 3. The multiple interaction impulses become multiple overlapping impulse responses which feed the summer, and the summer output is fed to the string.

Figure 3: Creation of multiple hammer-string interaction force pulses as the superposition of impulse-responses of a bank of digital filters. The input to each filter is a single impulse, and the sum of their outputs gives the desired superposition of hammer-string force pulses. When the input impulses increase in amplitude, the output pulses become taller and thinner, showing less overlap.

At a specific dynamic level, we have obtained the critical feature that the model is linear and time invariant. That means we may now commute the soundboard/enclosure filter with not only the string, but with the hammer lowpass filter as well. These operations are carried out in going from Fig. 4, which shows a naturally ordered schematic diagram of the complete piano synthesis system, to Fig. 5, which shows the results of commuting the hammer-string assembly with the soundboard and enclosure.

Figure 4: Piano synthesis as described using natural ordering of all elements.

Figure 5: Piano synthesis using commuted ordering. The soundboard and piano enclosure are commuted such that we only need a stored recording of the impulse response of their series combination. The large digital filter required to implement the soundboard and the piano enclosure is thus removed. A change in the hammer-string collision velocity $v_c$ changes the filters and triggers playback of the soundboard/enclosure impulse response.

While the commuted result is valid only for a fixed hammer-string collision velocity $v_c$, that is all we need. For different collision velocities, we simply alter the filters, denoted LPF1 through LPF3 in
Fig. 5, applied to the soundboard/enclosure impulse response. This works because the nonlinearity is confined to the hammer-string collision, and these are discrete, non-overlapping events which can be modeled individually using linear, time-invariant elements, indexed by collision velocity. Note, however, that if a “virtual piano key” is re-struck before the excitation table has finished playing out, that playback must either be prematurely terminated (the low-cost solution), or multiple, overlapping playbacks must be supported, as in commuted bowed-string synthesis [11].

3.2 Excitation Factoring

It is typically more efficient to implement the highest Q resonances of the soundboard and piano enclosure using actual digital filters. By factoring these out, the impulse response is shortened and thus the required excitation table length is reduced. This provides a classical computation versus memory trade-off which can be optimized as needed in a given implementation. The explicit resonators can be conveniently implemented using parametric equalizer sections, one per high-Q resonance. In many practical situations, parametric eq sections may already be available in a separate effects unit.

A possible placement of the resonators is shown in Fig. 6. However, since all elements are linear and time invariant, they may be ordered arbitrarily. For example they could appear before the string. Having the resonators at the end, however, is convenient for defining multiple outputs having different spectral characteristics. Traditionally, resonators, equalization, dynamic comb filtering, and reverberation are implemented as post-processors, and these can all provide a diversity of outputs which can be panned individually into the stereophonic (or N-channel) sound-output stream for added sonic richness.

![Excitation & Nonlinear Filtering](image)

**Figure 6**: Example block diagram of a complete, commuted-piano synthesis system, including resonators which partially implement the response of the soundboard and enclosure, equalization sections for piano color variations, reverberation, comb-filter(s) for flanging, chorus, and simulated hammer-strike echoes on the string, and multiple outputs for enhanced multi-channel sound.

3.3 String Reverb

The sound of all strings ringing can be summed with the excitation to simulate the effect of many strings resonating with the played string when the sustain pedal is down. The string loop filters out the unwanted frequencies in this signal and selects only the overtones which would be excited by the played string. This is just another case of commuting the string with a resonator, where in this case, the resonator includes a bank of sympathetically vibrating strings. Sampling synthesis techniques can be used, for example, to synthesize the sound of all piano strings resonating at the same level.

4 String Interface

In a physical piano string, the hammer strikes the string between its two endpoints, some distance from the agraffe and far from the bridge. This corresponds to the diagram in Fig. 7, where the delay lines are drawn according to their physical interpretation.
Figure 7: Illustration of string excitation in a filtered delay loop arranged to display the physical model interpretation. The delay lines contain samples of traveling force waves in this case, and other wave variables yield a similar diagram. The hammer-string interaction force pulse is summed into both the left- and right-going delay lines, corresponding to sending the same pulse toward both ends of the string from the hammer. One direction is negated relative to the other in a force wave simulation, while both are the same sign in a velocity wave simulation (but then the string terminations would be inverting).

By commutativity of linear, time-invariant elements, Fig. 7 can be immediately simplified to the form shown in Fig. 8, in which each delay line corresponds to the travel time in both directions on each string segment. From a structural point of view, we have a conventional filtered delay loop plus a second input (inverted) which sums into the loop somewhere inside the delay line. The output is shown coming from the middle of the larger delay line, which gives physically correct timing, but in practice, the output can be taken from anywhere in the feedback loop. It is probably preferable in practice to define the output as the delay-line input. That way, other response latencies in the overall system can be compensated to a maximum extent.

Figure 8: Diagram equivalent to Fig. 7, obtained by combining upper- and lower-rail delay lines.

An alternate structure equivalent to Fig. 8 is shown in Fig. 9, in which the second input injection is factored out into a separate comb-filtering of the input. The comb-filter delay equals the delay between the two inputs in Fig. 8, and the delay in the feedback loop equals the sum of both delays in Fig. 8. In this case, the string is modeled using a simple filtered delay loop, and the striking force signal is separately filtered by a comb filter corresponding to the striking point along the string. This factoring adds to the amount of memory needed, but (1) simplifies automatic loop calibration, and (2) the comb filter can be implemented elsewhere, such as in an effects unit. Post-processing comb filters are often used in reverberator design and in virtual pick-up simulation.

The comb-filtering can also be conveniently implemented using a second tap from the appropriate delay element in the filtered delay loop simulation of the string, as depicted in Fig. 10. The new tap output is simply summed (or differenced, depending on loop implementation) with the filtered delay loop output. Note that making the new tap a moving, interpolating tap (e.g., using linear interpolation), a flanging effect is available. Adding more moving taps and summing/differencing their outputs, with optional scale factors, provides an economical chorus or leslie effect. These
extra delay effects cost no extra memory since they utilize the memory that's already needed for the string simulation. While such effects are not traditionally applied to piano sounds, they are applied to electric piano sounds which can also be simulated using the same basic technique.

It is also possible to eliminate explicit comb-filtering corresponding to the hammer striking position. The uniform spacing of the force pulses in the excitation signal $f(t)$ is the same as the delay needed for the striking-position comb filter. As a result, the physical force-injection signal $f(t)$ can be replaced by the comb-filtered version $g(t) = f(t) - f(t - \tau)$, where $\tau$ is the travel time from the striking point to the agraffe and back. The comb filtering can be applied to the excitation table prior to the shaping filter(s), or the shaping filter(s) can be designed to convert the excitation table directly into $g(t)$ rather than $f(t)$. In either case, the final excitation signal $g(t)$ simply drives a single filtered delay loop.

Perhaps it should be emphasized here that for medium to high quality piano synthesis, multiple filtered delay loops should be employed per key rather than the single-loop case discussed here. Each delay loop may correspond to a different string hit by the same hammer, a different polarization plane on a single string, or to a longitudinal wave. Thus, in a good piano synthesizer, there should be at least two filtered delay loops, tuned differently (unequal loop delays), both excited by $g(t)$ in some manner (e.g., they can each received an identical copy), and the delay loop outputs should be at least summed or else realistically coupled as described in [11].

5 Conclusions

A highly efficient computational model for the piano derived from an acoustic model was described. The hammer-string force interactions are modeled as discrete events which can be modeled as one or a few successive impulse responses of low-order digital filters. The soundboard and enclosure filtering is replaced by a look-up table using one or a few read-pointers per note. Further details are given in the companion paper [16], and related techniques are discussed in [3].
References


Forward-Going Wave Extraction in Acoustic Tubes

Tim Stilson
Center for Computer Research in Music and Acoustics (CCRMA)
Stanford University
stilti@ccrma.stanford.edu

ABSTRACT: A simple estimator is developed to extract the forward-going traveling pressure wave from physical measurements of pressure in a tube, which consist of the sum of the forward-going wave and a reverse-going wave. The estimator is then analyzed and limits of performance are discussed. Finally, a few extensions are discussed and analyzed which overcome some limitations.

1 Introduction

The solution to the wave equation in simple acoustic waveguides such as the cylindrical tube prescribes the superposition of two waves traveling in opposite directions down the waveguide (Morse). Any physical measurement of the waves in the tube will always return the sum of the two waves. Sometimes, it is desired to be able to extract just one of these waves from the measurements. For example, to measure bell reflection functions.

The two-sensor method is a well known technique for handling two-way propagation in tubes, and has been explored deeply (Spiekermann et al.), (Gibiat et al.) and (Abom et al), but always in directly working the problem of measuring reflection functions and other properties of the tube. In fact these methods never seem to actually estimate the forward-going wave (or the backward-going wave), instead they directly calculate higher level functions. In this paper, we will derive estimators for the forward-going wave using the two-sensor method. We will assume a tube oriented horizontally, with the two (pressure) waves labeled \( R(x - ct) \) and \( L(x + ct) \), (for 'Right'- and 'Left'-going), where \( x \) is position in the tube, and \( c \) is the wave velocity. The absence of DC flow (and turbulence) is assumed, along with negligible attenuation between sensors.

2 The Two-Sensor Estimator

If we at first assume perfect noiseless measurements, and let the space- and time-separation of the sensors be: \( \Delta x = x_2 - x_1, \Delta t = \frac{\Delta x}{c} \). We use the fact that the right and left going waves propagate to get the following relations between the measurements and waves at the sensor locations:

\[
\begin{align*}
s_1(t) &= R(x_1 - ct) + L(x_1 + ct) \\
s_2(t) &= R(x_2 - ct) + L(x_2 + ct) \\
R(x_2 - c(t + \Delta t)) &= R(x_1 - ct) \\
L(x_2 + c(t - \Delta t)) &= L(x_1 + ct)
\end{align*}
\] (2.1)

We solve this set of equations to get:

\[
\begin{align*}
R(x - vt) &= R(x - v(t - 2\Delta t)) + s_1(t - \Delta t) - s_2(t - 2\Delta t) \\
L(x + vt) &= L(x + v(t - 2\Delta t)) + s_2(t - \Delta t) - s_1(t - 2\Delta t)
\end{align*}
\]

\[
\begin{bmatrix}
\tilde{R}_2(s) \\
\tilde{L}_1(s)
\end{bmatrix} = \begin{bmatrix}
e^{-j\pi f \Delta t} & 1 \\
1 - e^{-j\pi f \Delta t} & -e^{-j\pi f \Delta t}
\end{bmatrix} \begin{bmatrix}
s_1(s) \\
s_2(s)
\end{bmatrix}
\]

This matrix equation is quite useful for analysis of the system, and extends well to more complicated systems, such as those with more sensors. This generalizes well to discrete-time systems, simply by substituting \( z = e^{\frac{j \pi f}{2\Delta t}} \). Note, however, that the delays in the estimator become integer-length, thus restricting the choice of sensor spacings \( \Delta x \) to \( \frac{\pi f}{2\Delta t} \).

Figure 1: Two-Sensor Estimator
2.1 Effects of Noise

Noise is inevitable in the measurements $s_1$ and $s_2$, so the sensor equation is rewritten as:

$$
\begin{bmatrix}
s_1(s) \\
s_2(s)
\end{bmatrix} =
\begin{bmatrix}
1 & e^{-j\pi f \Delta t} \\
e^{-j\pi f \Delta t} & 1
\end{bmatrix}
\begin{bmatrix}
R_2(s) \\
L_1(s)
\end{bmatrix} +
\begin{bmatrix}
v_1(s) \\
v_2(s)
\end{bmatrix}
$$

If we assume that the noises are zero-mean, then the same estimation as before should work, but let's see how it is affected by the noise: if we plug the above equation for $s_1$ and $s_2$ into the estimator and assume the noises are independent, we get the following estimator performance equation:

$$
\begin{bmatrix}
\tilde{R}_2(s) \\
\tilde{L}_1(s)
\end{bmatrix} =
\begin{bmatrix}
R_2(s) \\
L_1(s)
\end{bmatrix} +
\frac{1}{1-e^{-j4\pi f \Delta t}}
\begin{bmatrix}
v_3(s) \\
v_4(s)
\end{bmatrix}
$$

Where $v_3$ and $v_4$ are simply another set of noises obtained from the addition of $v_1$ and $v_2$. We see that the noise gain has poles at DC and at multiples of $\frac{1}{2\Delta t}$ (DC and $\frac{f}{2}$ in discrete-time, using one-sample sensor spacing). Thus, although the transfer function from $R_1$ to $\tilde{R}_1$ is flat, the estimation SNR could get rather low at the pole locations, thus making the estimation quite inaccurate.

This can also be seen by looking at the condition number ($\kappa$) of the sensor matrix vs. frequency (Figure 2). One interpretation of $\kappa$ is the “effective-singularity” of the matrix ($\kappa \rightarrow \infty \Rightarrow$ Singular). Since the condition number approaches $\infty$ at DC and $\frac{f}{2}$, we can say that those frequencies, the sensor matrix is essentially singular, so the individual waves are not estimable. We can use the condition number as a quick check to see if a given sensor configuration is good for estimating the travelling waves.

FIR Estimators: If the noise gain at DC and $\frac{f}{2}$ is too disturbing, one may try an FIR estimator, simply by removing the recursion from the estimator, which removes the poles in the transfer functions (we also removed the extra delay, which became unnecessary). This gives an estimator whose performance is:

$$
\begin{bmatrix}
\tilde{R}_{FR2}(s) \\
\tilde{L}_{FR1}(s)
\end{bmatrix} =
(1-e^{-j4\pi f \Delta t})
\begin{bmatrix}
R_2(s) \\
L_1(s)
\end{bmatrix} +
\begin{bmatrix}
v_3(s) \\
v_4(s)
\end{bmatrix}
$$

Technically, the SNR is the same as before (the zeroes reduce the signal the same amount as the noise was boosted before), but in some cases this may be preferred (ease of analysis, for example).

The zero at $\frac{f}{2}$: These zeroes (at $\frac{f}{2}$ and multiples) come from the fact that these wavelengths fall unobservably between the sensors, and the subtraction in the estimator ($\tilde{R}_1(k) = s_1(k) - s_1(k-1)$ in DTS) causes these frequencies to cancel at the sensors. One possible solution to this problem is to use multiple sensors. The idea being that the zeroes of the various pairs of sensors would not all land at the same frequencies, so that the estimation from one pair could be used to “fill-in” where another is at a zero. Unfortunately, in discrete-time all the possible sensor separations are $n c T_s$, giving zeros at multiples of $\frac{f}{2\Delta t}$ for each sensor pair. Thus all sensor spacings will have a zero at $\frac{f}{2}$. 
Oversampling, however, can be used to get a sensor separation that doesn't have a zero at the original \( f \). For example, we could double the sampling rate and place the sensors at \( x_t, x_{t+2} \), and \( x_{t+3} \) according to the new sampling rate. The first two sensors are separated by one sample in the original sampling rate, and the second pair are spaced by one half-sample in the original sampling rate, so their first zero is effectively at \( f_{\text{orig}} \). These sensors could be combined so that the zero at \( f_{\text{orig}} \) is "filled-in", and then the estimate resampled back to \( f_{\text{orig}} \). The condition number of this method's sensor matrix is shown in Figure 3 for the frequency range of interest corresponding to the larger sensor spacing, which for this derivation is one sample at \( f_{\text{orig}} \). Thus the zero at \( f_{\text{orig}}/2 \) has disappeared (or, for a non-FIR estimator, this means that the noise gain at that frequency has become no longer a problem).

The zero at DC: No combination of sensors can fill-in the zero at DC. In fact, it is difficult to define the right-going wave as being distinct from the left-going wave at DC (in any finite-length tube), which reinforces the implication that DC (and near-DC) is unobservable. All estimation transfer functions will thus fall to zero at DC. The important question is how close to DC the estimator can get without breaking down.

There are a few ways to approach this question. We can extend the multiple-sensor system and add sensor pairs that have less-severe attenuation at low frequencies. For example, a pair with a very wide spacing will have quite a few zeros in the range of interest, but will have regions of little attenuation much closer to DC than the 1-sample sensor spacing (see Figure 4). Multiple such sensor-pair estimates can be combined to get a better response close to DC. Note that a similar result can be had near DC by subsampling a closely-spaced pair of sensors. This can also be viewed as an FIR filter design problem: one of choosing the weightings of a regularly-spaced set of sensors to get a desired estimation frequency response, under the added condition that the transfer function from the unwanted wave be attenuated as far as possible.

### 2.2 Effects of Sensor Misplacement

All of the preceding derivations assumed that the estimator could implement delays precisely equal to the wave propagation delays between the sensors. In the discrete-time estimator case, this may be particularly difficult, since the time delays are fixed, so the sensor placement must be rather precise. Here, we analyze the effects of sensor misplacement.

Say the sensor spacing is \( \Delta x = v\Delta t \) and the delay used in a two-sensor FIR estimator is \( \Delta t_2 \), let the delays be unequal, but quite similar (\( \Delta t_2 - \Delta t_1 = \varepsilon \ll \Delta t_1, \Delta t_2 \)). If we look at \( \tilde{R}_{\text{mis2}}(t) \), we see that \( L(x_1 + vt) \) is no longer fully canceled, we get:

\[
\tilde{R}_{\text{mis2}}(t) = \tilde{R}_{\text{mis2}}(t) + [L(x_1 + vt) - L(x_1 + vt - v\varepsilon)]
\]

\[
= \tilde{R}_{\text{mis2}} + 2e^{-2j\pi f/\Delta t_2} \sin(2\pi f)e^{j\pi f}(f)
\]

Note: if it were possible to fine-tune the sampling rate, then misplacement would not be a problem, since the sampling rate could be chosen to get delays that match the wave-propagation delays.
2.3 Effects of Sensor Spread

Similar to the sensor misplacement problem is the fact that most pressure sensors do not pick up just the pressure at a single point. Instead, they have some area over which they pick up pressure, effectively giving the 'average' of the pressure over the area. It is possible that this averaging could cause an effect similar to misplacement in the estimator. If we model the spreading as a space convolution along the length of the tube, we get

\[ s_1(t) = R(x_1 - vt) \ast Sp_1(x) + L(x_1 + vt) \ast Sp_2(x), \]

and a similar function for \( s_2(t) \). Luckily, if we assume that the sensors have the same spread functions, and that these spreads are symmetric about the pickup point, (both of which are fair assumptions), the linearity of convolution gives the following result for a two-sensor FIR estimator (after we note that in the case of traveling wave measurements, space convolution is equivalent to time convolution): \( \tilde{R}_2(t) = \tilde{R}_1(t) \ast vSp(vt) \). This gives a simple filtering of the signal, which, given the typical size of the spread function, only affects the highest of frequencies.

Thus we find that sensor spread doesn't become a problem for identical, symmetric-pattern sensors.

3 Conclusions

The sensor matrix can be inverted to give a simple forward-wave estimator, but it breaks down at DC and \( \frac{1}{2} \) in the simple discrete-time setup. Multiple sensors (together with oversampling for discrete-time estimators) can reduce the problem at the higher frequency, and can push the break-down point closer to DC. The condition-number of the sensor matrix can be used to gauge the effectiveness of a sensor configuration.

If the sensors are incorrectly placed, or if the estimator implements incorrect delays, the unwanted wave will leak into the estimator output, at a gain of approximately 20 dB per order-of-magnitude placement error. Finally, sensor spread produces no leakage if the sensors have identical spread functions and these functions are symmetric.

References


[Morse] Philip M. Morse Vibration and Sound Acoustical Society of America, 1981


This material is based upon work supported under a National Science Foundation Graduate Fellowship.
ABSTRACT: A Karplus-Strong string is length-modulated by a sine wave whose frequency is very close to the fundamental frequency of the string. The resulting waveform is quite complex and sonically intriguing. The addition of an instability allows the system to regenerate, giving a sound that can go for long periods of time before repeating. The string is analyzed in comparison to FM and time-domain pitch shifting, both of which have similar architectures and actions.

The introduction of non-linearities into digital waveguide models has been shown to add very interesting and satisfying complexity to the resulting sound (Van Duyne, et al. 94). A version of the Extended Karplus-Strong string (Jaffe and Smith 83) is presented which achieves its effect not through a nonlinearity in the string loop, but instead as a disruption of the time-invariance of the loop. Audio-frequency modulation of a string pickup point was explored by Van Duyne (Van Duyne and Smith 92), when the string length is modulated rather than the pickup position, certain effects are similar (an FM-like effect is seen in many cases). However, when the delay-line length is modulated at a frequency near to the fundamental frequency of the string, a more complex interaction occurs, and a self-regenerating sound can be produced which is exceedingly complex, yet quite intriguing.

**How it Acts:** When the modulation index is zero, the system acts just like a normal string model. As the modulation index is increased, the effect appears and increases in intensity until, depending on the string length and $\Delta f$, the sound either kills itself or the regeneration becomes overpowering (see the Regeneration section). Moving the modulating frequency away from the string's frequency reduces the effect to FM, and moving it too close to the string's frequency causes the effect to blow up even at low modulation indices. Finally, everything sounds best at very low frequencies ($\text{freq} < 30 \text{ Hz}$, preferably $\approx 10 \text{ Hz}$) because of the patterns that become audible far above the fundamental frequency.

**Doppler Effects:** Because the string length is being modulated at nearly the same frequency as the string is resonating, half of the wave in the string will be time-compressed as it reflects off the end of the string, because the string is shortening, and the other half of the wave will be expanded, because the string is lengthening. In this case, the string's length changes smoothly, so that the compressed part of the wave appears to have been pitch-shifted upwards, and the expanded part appears to have been pitch-lowered (indeed, a degenerate time-domain pitch shifter can be visualized as a delay whose length is being modulated by a sawtooth wave). Thus we get a simultaneous upward and downward shift of the spectrum of the sound that exists in the string. When $\Delta f$ is not too close to zero, the effect will soon reverse itself, because the modulating oscillator becomes out of phase from the string. Thus sections of the wave that had been shifted up, get shifted down, and vice versa. If $\Delta f$ is too small, waveform will become too distorted before the effect reverses, and will succumb to various numerical effects, which kill it ("blowing up")
Half-Cycle Effects: Quite a few algorithms treat different parts of a cycle differently, such as a string with a passive nonlinear filter (Van Duyne et al 94), where the PNF's action is dependent on the state itself, so can, when placed in a string loop, appear to give two different string lengths for different parts of the signal, which produces a sound quite similar to the WeirdString effect.

A simple variation of the patch modulates the string length by a function of the string state, rather than simply by the string fundamental, which produces a slightly different class of sounds and behaviors ("it sounds sort of like a speed boat on the open sea...").

Regeneration: The signal-controlled one-pole filter in the loop has its gain set such that during half of the cycle, the filter has a large gain around 1/2 rather than an attenuation ($a_{center}$ determines how much of the cycle is gain and how much is attenuation). This causes the loop to become unstable under certain circumstances. If the standard Karplus-Strong loop filter is used for the FIR filter in the loop (a two-point average), the instability is never seen, because the two-point average has a zero at $1/2$, which quells the gain of the one-pole filter, but if the feedback FIR filter is changed, even if only a little bit, so that the zero moves off $1/2$, the instability manifests itself.

The previously-mentioned pitch-shifting effect of the modulation causes the high frequencies generated by the instability to be pushed down in frequency into the rest of the spectrum, thus regenerating the effect over the whole frequency range. Regeneration allows the sound to continue for quite a long time, and adds new dimensions to the effect.

In the shown patch, the "intensity" of the instability is proportional to the modulation index, so the amount of regeneration is connected to the magnitude of the whole effect. If the modulation index gets too high, the regeneration can end up overpowering everything else, and degenerating the sound into noise.

Additional: A stereo version of this patch has been constructed by making a nearly identical copy of the string and placing it on the other channel. Because of the complexity of the action of this system, most differences between the two strings would cause their sounds to quickly become uncorrelated, killing any stereo image, thus only the tiniest of changes is made to the copy: the modulating oscillator is temporarily detuned to cause the modulators to become out of phase between the two channels, otherwise the channels are identical and are excited and controlled identically.

The repetition period of this system can be even further lengthened by adding an LFO tuned to some very low frequency (such as $1/30$ Hz) to the modulation index. This has produced patterns whose general sound 'shape' changes on the scale of minutes.

References


[Van Duyne and Smith, 1992] Scott A. Van Duyne and Julius O. Smith III. Implementation of a Variable Pick-Up Point on a Waveguide String Model with FM/AM Applications, ICMC-92, San Jose. 92

This material is based upon work supported under a National Science Foundation Graduate Fellowship.
Capella: A Graphical Interface for Algorithmic Composition

Heinrich Taube
Zentrum für Kunst und Medientechnologie, Karlsruhe
hkt@zkm.de

Tobias Kunze
CCRMA, Stanford University
tkunze@ccrma.stanford.edu

ABSTRACT: Capella is an object-oriented graphical interface for algorithmic composition in Common Music. It defines classes of browsers and worksheets that implement a consistent set of visualization tools and serve as a graphical front end for the system. The interface currently runs on the Macintosh under Macintosh Common Lisp.

Introduction

Algorithmic composition is a complex activity in which both musical and technological issues must be addressed in parallel. This, in turn, places special requirements on a graphical interface that supports the process. Object-oriented composition environments such as Common Music, DMix, and Mode place additional demands on graphical tools due to the breadth of representation and functionality that these kind of systems implement. Smalltalk environments are able to take advantage of a powerful windowing system provided by Smalltalk itself. Since Common Music was designed to be as portable as possible, without the aid of a native windowing system, almost no attempt to address visualization issues was made until recently. Until now, visual output in Common Music was completely text-based, similar to the type of display one sees when working, for example, in a Unix shell window. Common Music’s command-line driven interpreter, Stella, connects to the system’s toolbox similar to the manner in which a shell connects to Unix. Although it allows powerful input expressions to be formulated, Stella does not allow the inner processes to be easily understood. Capella is a response to some of the communication limitations in Stella, while keeping in mind that graphic representation and mouse based gestures are not always the best or most expedient models to choose for interacting with a complex system. Capella has been designed to be a complement, not a replacement, for the two other modes of interaction supported by the system: command processing from Stella and procedure invocation from Lisp. Common Music simply runs all three modes “in parallel” (Figure 1) and the composer is free to choose whatever is most appropriate to a particular situation.

Capella is still in the early stages of development. Its primary goal is to allow a set of flexible visualization tools to be developed, but it also makes interacting with the system as a whole easier and more transparent. The need for transparency is particularly acute in algorithmic composition workshops, where participants must quickly absorb not just new theoretical concepts, but a specific implementation of them as well.

Browsers and Worksheets

Capella provides two basic classes of windows: browsers and worksheets. Browsers provide context sensitive views on musical objects and permit the associated data to be inspected and manipulated in some class specific manner. A basic premise behind the design of Capella is that there is not a single best way to view objects in the system, but rather, that their visualization depends upon context. Browsers in Capella are therefore not identical to the musical structures they display. Worksheets are windows that support compositional activities such as analysis gathering, musical event editing, musical structure definition, and score output processing. Each class of worksheets provides methods for a generic protocol controlling the creation, enabling, activation, and updating of worksheets and their subviews.

Any number of browsers and worksheets may be open on the screen at the same time but only one browser and worksheet are said to be “active” at any given time. The active browser is called the “focus browser” and typically provides selection constraints for the active worksheet. The system currently defines classes of worksheets for data

---

1 cf. Taube (1994)
2 cf. Oppenheim (1993)
3 cf. Pope (1992)
Figure 1: Capella serves as a graphic "front end" to Common Music. This illustration depicts the three parallel interaction modes possible in Common Music: gesture (Capella), command (Stella) and procedure (Lisp).

Figure 2: A SEE browser with its control pane expanded. Four of the five possible dimensions are depicted for a set of Midi-Note events. The x-axis has been set to time, y-axis to pitch, color to channel, and length to duration. The brightness dimension was not used. The browser's value access functions take care of mapping back and forth between symbolic, floating point or integer pitch representations.

editing, creation and deletion of musical objects, query processing, global preference setting, loading and saving archives, and score processing. Several of the browsers and worksheets will be discussed in more detail in the remaining sections of this paper.

Information Browser

The information browser provides class specific summary information about an object (its type, status, parents, number of subobjects, and so forth) and a table for displaying and editing the current slot values in the object. This table distinguishes between internal, external and "read only" slots. Normally, only external slots are included, and only "writable" slots have their input buffer active for editing. An "inspect" mode overrides this default behavior and gives full display and editing access to all the slots in the object.

Listing Browser

The listing browser (Figure 3) displays a containing object together with its subobjects. Subobjects are displayed by a printer with class specific methods. Subobjects that are "events" include parameter information in their output displays, so when a listing browser displays event data it provides formatted views for each parameterized sound event.

Listings can be used in conjunction with Edit worksheets for generic sequence editing. Operations on the Edit worksheet apply to the current selection in the focus listing browser. Subobjects in a listing may be selected by mouse gesture, by iterative index referencing, and by the application of musically salient predicate selection expressions.

SEE Browser

The SEE (Structured Event Editor) browser (Figure 2) is a visualization and analysis tool that operates on sequences of parametric, ordered data objects. These sequences may be structures declared in the system or be "virtual" sequences of merged output generated from multiple objects by the scheduler.

The SEE browser provides a programmable graphics output window that supports a number of user customization and display hooks. To use a SEE browser, the composer assigns a maximum of five "data parameters" to five dimensions of visualization supported by the browser. The first two dimensions determine the element's location on a Cartesian pane (representing its x- and y-axis, respectively). The third dimension is drawn as an element's "length" in the same direction as either the x- or y-axis. Brightness is used to represent a fourth dimension, and color (ie. hue and saturation) provides a fifth dimension. None of these dimensions are required to participate in the drawing process or be used to display a specific type of data.
**Algorithm Browser**

The algorithm browser (Figure 4) is a structured editor for displaying and modifying program code associated with musical algorithm and generator objects. The browser maintains two similar but completely independent editors for the object’s initialization and “run time” statements. The editors are separated by a divider; moving the divider up or down controls the percentage of space allocated to each editor in the browser. Each editor manages its program statements using a control menu and an internal code table. Statements are usually indexed in the menu under the name of the parameter they effect. Selecting a statement from the control menu installs it in the associated code table. Once a statement is installed in the table its index in the control menu is “greyed out” and cannot be reselected until the statement is deleted or deinstalled from the table.

The control menu can hold alternate expressions affecting the same parameter. This allows a composer to easily compare or test different expressions affecting the same parameter. Code tables permit their structure as well as their contents to be easily modified. Statements can be moved in the table simply by control–mouse–dragging them to a new position. Code tables also perform automatic syntax checks on their input expressions and will “pretty print” their contents upon request. Once code has been developed, the browser can either redefine the algorithm object associated with it, or else “decompile” the code tables into a Lisp expression that, if evaluated, would redefine the algorithm object. This Lisp expression is automatically dumped to a new window containing a Lisp editing buffer.

**Output Browser**

Output browsers permit mixtures of sequential and simultaneous arrangements of object references (layouts) to be
“layed out” and then processed to a specific output stream (Figure 5). The Stream and Layout panes allow the user to create, load, save and select layouts and streams from a menu. Once a particular layout has been selected it is graphically depicted in the layout pane itself.

A Layout is defined in terms of one or more “object references”. Each reference may represent a single object (for example an algorithm, thread or merge), or a subset (chunk) of subobjects from some containing object. A reference is graphically depicted as a box (block) in the layout pane. Blocks are mouse-draggable and self-adjust to the width of their textual content. If a block is moved it “snaps” to its horizontal and vertical neighbors after it has been released. All actions—the creation of new object references, duplication of existing ones, moving, selection, and changing the content—may be performed directly on the layout pane using standard command key combinations. Once the layout contains one or more references and an output stream has been selected, the Return key may be used to activate output processing. Layout processing moves from left to right across the pane: objects within a column are scheduled relative to one another and movement across columns represents “sectional” (non overlapping) divisions. Within a column, timeshifts relative to other objects may be specified using the @<time> qualifier, and block repetition is possible using the repeat qualifier.

Conclusion

Although Capella is a working, functional interface, the project is still in its early stages and much work remains to be done. Short term goals include developing new classes of browsers for musical pattern display and statistical analysis. Intermediate goals include insuring the modularity of native windowing code, and isolating as much graphic behavior as possible in methods on generic windowing operators. Long term goals include porting Capella to other windowing systems and at least one public domain Lisp implementation. The first port will most likely involve X-Windows on the Silicon Graphics line of machines.

References


A Physical String Model with a Twist

Steven D. Trautmann
Center for Computer Research in Music and Acoustics
Stanford University
(415) 723-4971/wishbon@ccrma.stanford.edu

ABSTRACT: A physical model of a string embedded in a Möbius strip is presented. Due to the nature of this embedding and the global non-orientability of the strip, interesting properties arise. Given a string at equilibrium, and an initial velocity at any point, a point half way around will function as a node. The fundamental of the string will be the same as its length, but each point in the string will function as if it were the center point of a string with fixed ends. At least one point of the string will be at equilibrium at any time.

Taking a circular string and embedding it in a Möbius strip (see fig. 1) does not change the derivation of the wave equation since the string can be orientated locally (for a general derivation see Powers, 1979). Thus the ideal approximation of \( \frac{\partial^2 u}{\partial t^2} = \frac{1}{c^2} \frac{\partial^2 u}{\partial x^2} \) still holds locally, though care must be taken due to lack of global orientation.

Figure 1: Möbius strips with embedded strings at equilibrium and displaced respectively

This lack of global orientation can be compensated for artificially by choosing a point on the Möbius strip where the orientation reverses. In this model, the direction of string motion within the strip is orientated relative to the strip itself, and the twisting of the strip in space does not affect the motion of the string. Choosing a point along the strip where the orientation changes, we can "cut" the strip there and flatten it, maintaining (or producing) the boundary conditions, namely that \( u(0, t) = -u(2\pi, t) \) and \( \frac{\partial u}{\partial t}(0, t) = -\frac{\partial u}{\partial t}(2\pi, t) \) and so on for higher derivatives, where the length has been scaled to \( 2\pi \) (see fig. 2).

Notice that since the string is continuous in all its derivatives, its position and each derivative (with respect to time) must equal zero at least at one point in the strip, since each derivative changes sign or is zero at the boundaries. Assuming \( u(x, t) = \phi(x)T(t) \) (that it is separable) the wave equation implies \( \frac{\phi''(x)}{\phi(x)} = \frac{T''(t)}{T(t)} \) (for \( 0 < x < 2\pi \)). This implies both sides are constant, say equal to \( -\lambda^2 \) which means \( \phi''(x) + \lambda^2 \phi(x) = 0 \). The general solution to this equation is \( \phi(x) = a \cos(\lambda x) + b \sin(\lambda x) \). Given the boundary conditions for the flattened strip, \( u(0, t) = -u(2\pi, t) \), it must be the case that
Figure 2: Flattened Möbius strip with boundary conditions

\[ \phi(0) = a \text{ and } \phi(2\pi) = -a \text{ which in turn implies } \lambda_n = \frac{1}{2}, \frac{3}{2}, \ldots, \frac{2n+1}{2} \text{ so that } b \sin(\lambda_n x) = 0 \text{ for all } \lambda_n. \]

Also \( T(t) = A \cos(\lambda ct) + B \sin(\lambda ct) \) so

\[ u(x, t) = \sum_{n=1}^{\infty} a_n \cos \left( \frac{2n+1}{2} \cdot x \right) \left[ A_n \cos \left( \frac{2n+1}{2} \cdot ct \right) + B_n \sin \left( \frac{2n+1}{2} \cdot ct \right) \right] \]

which can be compared to the solution to a string of length \( 2\pi \) bounded on both ends.

\[ u(x, t) = \sum_{n=1}^{\infty} \sin \left( \frac{n}{2} \cdot x \right) \left[ \hat{A}_n \cos \left( \frac{n}{2} \cdot ct \right) + \hat{B}_n \sin \left( \frac{n}{2} \cdot ct \right) \right] \]

Since the wave equation is the usual one, the general solution of the wave equation, \( u(x, t) = f^{+}(x + ct) + f^{-}(x - ct) \) holds in this case. It can be seen that any symmetrical initial condition at \( x_0 \), say \( f(x_0) \) will cancel itself when \( ct = \pi \), due to the relative change in orientation between the left and right going wave. Further, as the waves pass each other the left going wave appears the same as a reflected right going wave due to the difference in direction and orientation, and vice versa. Thus for any symmetrical disturbance, the string in the Möbius strip behaves as a string with fixed ends, the ends being at the node created. Since any initial condition and be broken down into symmetrical components (a countless set of delta-functions if nothing else), the string in the Möbius strip can be viewed as a sum of overlapping strings length \( 2\pi \) with symmetrical initial conditions.

A lossless implementation can be made for either position or velocity. Two possible implementations are given (see fig. 3). The first takes the point of view of the directional signals, which see themselves as constant, but the pick-up inverting each time around, which is one-half the total signal length since it takes two trips to be at the same spot with the same orientation. The second implementation is more akin to fig. 2, where the differently oriented parts of the signal are summed (actually differenced) to create directional signals of half the length. The inversion is lumped within the loop, and the number of pick-ups is reduced from four to two. Losses can be lumped and included in either implementation.

Figure 3: continuous directional signals and flattened Möbius strip implementations respectively

REFERENCE:

Toward a CLM Sound Localization Instrument employing Modified Wavefront Reconstruction

Steven D. Trautmann
Center for Computer Research in Music and Acoustics
Stanford University
(415) 723-4971/wishbon@ccrma.Stanford.edu

ABSTRACT: Audio wavefront reconstruction is similar to visual holography in which optical wavefronts are reconstructed by reproducing phases and amplitudes. In audio there are far fewer speakers than necessary to exactly reproduce a wavefront this way. In order to construct the closest approximation to a wavefront, information about the listening environment is used. The perceived quality of this approximation is further improved by using psychoacoustic principles. Ways to use this approach compositionally are being developed in Bill Schottstaedt's Common Lisp Music.

Thanks to the sampling theorem (Petersen and Middleton, 1962), any bandlimited waveform can be exactly reproduced given a sufficient number of samples in a region. Although a bandlimited audio signal is not spatially bandlimited due to a 1/distance amplitude scaling, the amount of spatial aliasing due to this factor is usually small given adequate distance from the source. A more critical aliasing problem occurs when the number and placement of samples provides inadequate information. The sampling structure used is rectilinear due to the relative ease of manipulation, but arrangements of samples can be made which are more efficient for given circumstances. Based simply on the distance between samples averaged over a 1x1x1 meter volume, the approximate maximum audio frequency obtainable by a given number of samples is shown (speed of sound is 343m/s). When averaged over a 1x1 meter area the approximate maximum audio frequency obtainable by a given number of samples increases.

![Figure 1: Approximate maximum frequency representable by N samples filling a 1 \times 1 \times 1m^3 volume and a 1 \times 1m^2 area respectively](image)

It is easy to show that given N loudspeakers, exactly N points in a free field environment can be made to have any signal (for instance see Romano 1987). In this case sets of filters can be made which solve for all frequencies. Here signals will be treated frequency by frequency. If $H$ is a complex-valued square matrix describing the change in phase and amplitude from N speakers $S$ to N points $P$ at frequency $\omega$, then $S = H^{-1} \cdot P$, so knowledge of the sound field at the sample points at a given frequency can determine the output of the speakers at that frequency. Thus replacing the word sample with the word speaker about gives an idea of how many speakers are required to reproduce and arbitrary sound field up to a given frequency.
Since in general, the number of loudspeakers is too small to reproduce a sound field exactly, the goal of best approximation presents itself. By not reducing the number of sample points needed to describe the sound field at a certain frequency, but just forming the best approximation at those points with fewer loudspeakers, a nearly best approximation to the sound field can be found. By increasing the number of sample points beyond the minimum, both the spatial aliasing problem and the potential for significantly sub-optimal approximations are reduced. A least-squares criterion can be applied to this set of sample points. Here the transfer $H$ from $N$ speakers to $M$ points $P$ is not square, but the minimum norm least-squares solution is $S = (H^* \cdot H)^{-1} H^* P$ where $H^*$ is the conjugate transpose of $H$. (similar approach to Yanagida et al. 1983)

The directivity characteristic and frequency response of the speakers can be taken into account as just an amplitude (and possibly phase) adjustment to the transfer of a particular frequency from the speakers to the sample points. Amplitude scaling at crossover frequencies can be used the same way, and much of the room acoustics can be taken into account in this way as well.

For most signals, the amplitude of the signal components is much more important perceptually than the phase. If the optimization does not already make amplitude more vital than phase, then a simple algorithm can be applied to any solution to “trade phase for amplitude” accuracy. By replacing the samples of the sound field with ones of identical amplitude, but phase based on the best approximation at those points from the speakers, a new approximation can be made in which any improvement will be in amplitude. (The phases will get worse, as will the total error).

Figure 2: Algorithm for trading phase for amplitude accuracy in a sound field (one sample shown)

By inserting a listener into the environment, certain filtering by the ears, head and body can affect the perceived result. Therefore frequencies which are strongly filtered by the head and ear to produce localization cues should be kept in the actual region close to the desired localization if possible. The resulting sound field may be suboptimal with no listeners, but produce better results for listeners. Ambiguous frequencies which are not strongly affected by head related transfer functions can be optimized. Also since the optimization is done in the frequency domain, it works best for sustained signals. Brief signals which aren't modeled well by frequency decomposition may be more distorted by this frequency domain optimization and should be excluded from it.

A procedure using the above techniques to adjust the signals sent to speakers using the short time Fourier transform has been implemented, and it is hoped that a Common Lisp Music instrument can be fashioned to use these techniques.

REFERENCES:
Developments for the Commuted Piano

Scott A. Van Duyne and Julius O. Smith III
Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University, Stanford, CA 94305
savd@ccrma.stanford.edu, jos@ccrma.stanford.edu

ABSTRACT: We present here three developments for the Commuted Piano Synthesis model described initially in the companion paper [3]: (1) a theoretical foundation and calibration scheme for the required linearized piano hammer system; (2) a simple algorithmic synthesis approach for the commuted soundboard impulse response, eliminating the need for any wave table memory; and (3) a calibration method for the coupled string system, required for high quality two-stage piano tone decay.

1 Background

Much study has been made of the piano and its parts, with an eye toward better understanding of the acoustics [8], toward more reliable numerical modeling of the piano physics [1], toward the development of high quality sound synthesis [5, 7], and toward the development of cost-effective sound synthesis [2, 3, 4, 6]. Our interest is in the category of sound synthesis informed by physics and acoustics. A high quality physical approach to piano synthesis has been suggested which combines the Wave Digital Hammer [5] with coupled string synthesis [2], using the 2D Digital Waveguide Mesh [7] as a soundboard resonator, and using allpass filtering methods to stiffen the soundboard and the piano strings [6]. On the other hand, we have proposed an algorithmic method of synthesizing the inharmonic piano tones constructing spectral regions with specially tuned FM oscillator pairs [4]. More recently, we have developed a hybrid method known as Commuted Piano Synthesis [2, 3] which combines the high quality and control of physical modeling synthesis with the cost-effectiveness of sampling “synthesis”. This method takes advantage of the commutativity of linear systems, and replaces the high order soundboard resonator with its own sampled impulse response played into the string at its excitation point. In order to implement Commuted Synthesis, we must linearize the hammer-string interaction, which is the focus of the first part of this paper. We have further hybridized and simplified the Commuted Piano Synthesis method by replacing the sampled soundboard impulse response with a simple algorithmic synthesis method which idealizes the soundboard and makes its physical characteristics easier to control. Lastly, we offer a method to calibrate the coupled piano string algorithm to real physical data using only a simple recording of a hammer hitting one string.

2 Linearizing the Piano Hammer

A fully physical nonlinear model of the hammer-string system has been proposed already [5]. However, in order to implement Commuted Piano Synthesis [3], we must commute the resonant soundboard system through the hammer-string interaction to the point of excitation in the commuted piano model. This requires that we replace the entire hammer-string interaction with a linear filter. Rather inconveniently, the hammer-string interaction is highly nonlinear in two important respects: First, the felt itself is nonlinear in that it gets stiffer the more it is compressed [1]. Second, the hammer leaves the string at some point, which corresponds to a shift in the models from a string interacting with a hammer to a string vibrating freely.
2.1 Linearized Analysis of the Piano Hammer-String System

Impedance of the Un-Terminated Ideal String The impedance experienced at some point on an un-terminated string is purely resistive:

\[ R_S \triangleq \frac{F_S}{V} = 2R_0, \]  

where \( F_S \) and \( V \) are the Laplace transforms of force and velocity at the driving point and \( R_0 \) is the wave impedance of the string, which is dependent on the square root of string tension times string density. The \( 2R_0 \) in the above equation results from taking into account the impedance of both halves of the string, as seen at the driving point.

Figure 1: String Terminated on One Side Only

Impedance of the Terminated Ideal String In the case of the piano hammer-string interaction, waves from the agraffe return and interact with the hammer before it leaves the string for most notes. However, the return waves from the bridge end of the string do not make it back before the hammer leaves the string, except in the very highest notes. Therefore, we formulate a half terminated string impedance taking into account a one sided termination at the agraffe end, as shown in Figure 1. The velocity response of a force impulse at the strike position is an impulse followed by an inverted impulse which returns reflected off the essentially rigid agraffe end of the string \( T \) seconds later:

\[ V = \frac{F_S}{2R_0} \left( 1 - e^{-sT} \right) \implies R_S \triangleq \frac{F_S}{V} = \frac{2R_0}{1 - e^{-sT}} \]  

Impedance of the Ideal Linear Hammer Let us assume that the hammer is of the form shown in Figure 2, essentially a mass and spring system, where the spring represents the felt portion of the hammer. We find that the impedance relation is:

\[ F_H = R_H \left( V - \frac{v_0}{s} \right) \quad \text{where} \quad R_H \triangleq \frac{ks}{s^2 + k/m} \]  

and where \( v_0/s \) represents the step input of the initial striking velocity. \( R_H \) has a zero at DC and two conjugate poles indicating an oscillation frequency of \( \sqrt{k/m} \).

Figure 2: The Linear Mass-Spring Hammer Model

Connecting the Hammer to the String When the hammer is in contact with the string, we take the velocity of the string equal to the velocity of the spring end of the hammer, and the force on the string equal and opposite to the force on the spring, \( F_S = -F_H \). Plugging in the string impedance relation, \( V = F_S/R_S \), we find:

\[ F_S = -F_H \triangleq -R_H \left( V - \frac{v_0}{s} \right) = -R_H \left( \frac{F_S}{R_S} - \frac{v_0}{s} \right) \]
In the un-terminated string case, we define $H_\infty$ as the transfer function from the initial striking velocity step to the force experienced by the string (and, equivalently, by the hammer felt). Taking the hammer to be a simple mass-spring system, we find that the $H_\infty$ transfer function is now a damped second order system, which looks just like the $R_H$ except for the under bracketed damping term (6). For practical physical parameters, this is an over damped system with real poles.

$$H_\infty \triangleq R_H \parallel R_S = \left( \frac{ks}{s^2 + k/m} \right) \parallel 2R_0 = \frac{ks}{s^2 + \frac{k}{2R_0}s + \frac{k}{m}}$$

(6)

For the one side terminated string case, we define $H_T$. Again, we find $H_T$ is like $R_H$ but for the under bracketed damping term, which in this case contains an interesting time delay part.

$$H_T \triangleq R_H \parallel R_S = \left( \frac{ks}{s^2 + k/m} \right) \parallel \left( \frac{2R_0}{1 - e^{-sT}} \right) = \frac{ks}{s^2 + \frac{k}{2R_0} \left( 1 - e^{-sT} \right) s + \frac{k}{m}}$$

(7)

### 2.2 Implementation

Conveniently, we find a recursion relationship between $H_\infty$ and $H_T$, which is independent of the exact nature of the hammer impedance, $R_H$.

$$H_T = \frac{H_\infty}{1 - e^{-sT}H_\infty}$$

(8)

This allows a simple recursive hammer filter implementation of the form in Figure 3:

Figure 3: Step-Driven Recursive Hammer Filter

Since, in this case, the hammer never leaves the string (from the linear system assumption), we may prefer to include a cutoff envelope in the feedback loop to terminate the reflections from the agraffe at some point, or better, break out the first few reflections in a feed forward formulation as in Figure 4:

Figure 4: Feed Forward Hammer Filter

Noting that $H_\infty$ is a differentiated lowpass filter,

$$H_\infty = \frac{ks}{s^2 + \frac{k}{2R_0}s + \frac{k}{m}} \triangleq sL_p,$$

(9)

the step-driven hammer system of Figure 3 may be commuted to an impulse-driven system, as preferred for Commuted Piano Synthesis [3]. This is shown in Figure 5. In this formulation, the hammer feedback loop contains what is fundamentally a lowpass filter and a DC-blocker. It is easy to break this out into a feed forward form as in Figure 4.
2.3 Analysis of Real Hammer-String Interaction Data

Using the Wave Digital Hammer [5] parameterized with measured data provided by [1], we were able compute the forces experienced by terminated and un-terminated middle-C (261 Hz) strings during a hard hammer strike. In the upper left plot of Figure 6, we see the felt compression force curves for a hammer hitting an un-terminated middle-C string (dashed line) and a terminated middle-C string (solid line). The multiple pulses correspond to return waves from the agraffe interacting with the hammer while it is still in contact with the string. Note that in the un-terminated string case, the force curve ramps smoothly to zero and the hammer apparently comes to rest on the string as in an over damped second order system. However, in the terminated string case, the hammer leaves the string when the return waves finally throw it away. The upper right plot of Figure 6 shows the dB magnitude spectra of these force curves. Note that the overall bandwidth of both the terminated string and un-terminated string hammer shock spectra are about the same. The multi-pulse spectrum (solid line) differs from the single-pulse spectrum (dashed line) primarily in a slight ringing of the lower spectrum region.

The lower right plot is the dB magnitude of the complex ratio of the multi-pulse spectrum and the single-pulse spectrum. The several low frequency peaks correspond to the spectral peaks one would expect from the hammer staying in contact with the string at the strike point (about 1/8 the way along the string) for some finite duration. It is of some interest that keeping the hammer in contact with the string introduces spectral peaks about every eight harmonics, whereas an impulsive
strike at the same position on the string introduces spectral nulls every eight harmonics. The piano hammer interaction is a compromise between these two extremes of behavior.

We further note that there are odd looking wiggles in the ratio spectrum, clearly visible around the 5–10 kHz range. These correspond in width to the side lobes one would expect from rectangularly windowing the time domain signal at exactly the point where the hammer leaves the terminated string. Hence, the severe nonlinear effect of the hammer leaving the string (which changes the entire linear system model) turns out in the spectral domain to be a simple convolution by the appropriate rectangular window sinc function.

The lower left plot in Figure 6 shows the inverse transform of the ratio spectrum. This is what is left of the multi-pulse hammer force signal if we de-convolve the single pulse force signal out of it. It appears to be a recursively damped impulse train, with some DC blocking, eventually centering the signal around zero. This is what was predicted by the linear hammer analysis as shown in Figure 5.

2.4 Spectral Modeling Approach to the Multi-Pulse Effect

An alternative approach to hammer filter design is to model the complex ratio spectrum as shown in the lower right plot of Figure 6 directly as a low order filter. This reduces the recursive or feed forward filter design methods of modeling multiple force pulses to a simple spectral equalization filter, as shown in Figure 7.

![Figure 7: Spectral EQ Method of Modeling the Hammer Filter](image)

In this case, we used a fourth order fit for the single pulse hammer lowpass filter. We then made a sixth order equalization filter fit to a few of the significant low frequency features of the ratio spectrum. The right hand plot of Figure 8 shows the equalization filter fit. The left hand plot shows the time domain output of the hammer filter system shown in Figure 7. The thick dotted lines are actual data as generated by the Wave Digital Hammer and the solid lines are the result of the filter fits. Note that the phase information in the sixth-order ratio spectrum fit results in a very good time domain approximation. In general, the coefficients of the lowpass filter part of this structure will be highly dependent on strike velocity, the harder the strike, the wider the bandwidth. However, the equalization part of this structure is reasonably consistent over strike velocity, and, in the simplified model, may be held constant over strike velocity, although it will vary with piano key number.

![Figure 8: Sixth Order Filter Fit to Ratio Spectrum](image)
3 Excitation Synthesis with Nonlinearly Filtered Noise

The impulse response of the piano soundboard is fundamentally a superposition of many exponentially decaying sinusoids, at least in its linear approximation. The reverberant effect of the soundboard occurs as energy from the struck piano strings is coupled into these modes and reverberated. However, if there were some particular modes of the soundboard which were unusually prominent, exhibiting a clear peak in the impulse response spectrum, and having an unusually long decay time, then a string which contained this frequency in one of its partials would couple into this mode more significantly than a string which did not have that frequency among its partials. This could produce unwanted unevenness in the piano tone from note to note. In general, much effort has been applied to the design of real piano soundboards to avoid such situations as these. The idealized piano soundboard should have a smooth, or flat spectral response locally, although it is evident that higher frequency modes decay a little faster than low frequency modes.

It is difficult to design a resonant system with such a flat response without using a very high order filter, for example the 2D Digital Waveguide Mesh [7]. On the other hand, it is easy to model the impulse response of such a system as exponentially decaying white noise, with the possible extension of a time varying lowpass filter applied to model high frequency modes decaying more quickly than low frequency modes. Using such a nonlinearly filtered noise model, we may synthesize any number of reverberant systems which have the characteristic that they have more or less smooth responses over the frequency spectrum, with no particular peaks of importance. The piano soundboard is a system of this kind.

In Figure 9 we show such a soundboard impulse response synthesis system. White noise is being fed into a time varying lowpass filter whose gain and bandwidth are both being controlled by envelopes. One possible implementation of this would use a one-pole lowpass filter whose denominator coefficient is being swept toward $-1$, thereby shrinking the bandwidth. If the numerator coefficient is modified to keep gain at DC constant, the amplitude envelope might even be dispensed with in a simplified system. Alternatively, more elaborate noise filtering systems may be used, possibly breaking the noise into frequency bands which would be enveloped independently to calibrate to some particular impulse response.

![Diagram](image)

Figure 9: Synthesis of Soundboard Tap with Nonlinearly Filtered Noise

Synthesizing Sustain Pedal Effect Just as the dry soundboard impulse response may be commuted to the point of excitation, similarly we may commute the entire sampled impulse response of the soundboard plus open strings with dampers raised to the point of excitation to obtain the resonant effect of the sustain pedal. Further, since there are so many resonating partials, the spectral response is essentially flat and filtered white noise with a long slow decay rate makes a good synthetic approximation.

4 Calibrating Coupled Strings

4.1 The Coupled String Model

Figure 10 illustrates a coupled piano string model for one note of the piano. The Coupling Filter represents the loss at the yielding bridge termination, and controls the coupling of energy between
and among the three strings [2, 8]. Each of the three string loops shown contain two Delay elements, 
the first corresponding to the delay path from the hammer strike point to the agraffe and back, 
the second corresponding to the delay path from the hammer strike point to the bridge and back. 
The relative delay length ratio for most strings is about 1 to 8, although the relative delay lengths 
may be set to model any particular piano string strike position. The input signals $E_1$, $E_2$, and $E_3$ 
are taken from the output of the hammer filter, which has been driven, in turn, by a soundboard 
impulse response, or a nonlinearly filtered noise excitation synthesis. Note that the input signals 
are introduced into the string loops at two points, in positive and negative form: this models the 
spectral combing effect of the relative strike position of the hammer on the string.

The signals $C_1$, $C_2$, and $C_3$ should be set to 1.0 during the sustain portion of the piano sound, 
and should be ramped to some appropriate loop attenuation factor, such as .95, at key release time. Alternatively, some more elaborate release sound model might be used. Note that, for una corda 
pedal effects, one or more of the signals $E_1$, $E_2$, or $E_3$, should be set to zero at key strike time. This 
causes the coupled string system to move quickly into its second stage decay rate, just as is found 
in real piano sounds when the una corda pedal is depressed.

In this coupled string model, the delay lengths are fine-tuned such that the effective pitch of each 
of the three string loops is very nearly equal, but not exactly equal. This is the mechanism by which 
two stage decay [8] is synthesized in the commuted piano synthesis model. The Stiffness Filters, as 
shown in the Figure, are intended to be an allpass filter structure which modifies the phase response 
of the loop so as to create the effect of the natural inharmonicity of the piano string partials. We 
recommend a bank of one-pole allpass filters as described in [6].

![Diagram](Image)

Figure 10: Three Piano Strings Coupled at a Bridge Termination

4.2 Calibrating the Coupling Filter

Ideally, from a physical perspective, we would like to measure empirically the bridge impedance, $R_b$, 
and the string impedance, $R_s$; and then from these measurements compute the desired Coupling 
Filter. However, following the spirit of the simplified string loop model presented above, let us 
say we have already calibrated a single string system and know $L(z)$, a lowpass filter modeling 
the per period attenuation of the tone, and $A(z)$, an allpass filter summarizing the dispersion in 
the string due to stiffness. We have presumably done this by measuring the partial frequencies and
corresponding decay rates of a single piano string. This may be accomplished by physically damping two of the three piano strings in a piano note group with felt, rubber, or some such means, and then recording the sound of the remaining undamped string decaying after it is struck. The decay rate of this single string should not contain very much two-stage decay interference [8] from the other damped strings, but should, instead, produce a reasonable single stage decay from which data about the partial frequencies and their individual decay rates may be extracted.

Loss in a string-bridge system comes almost entirely from the yielding bridge termination itself. That is, the loss from viscous air drag and internal friction is very small compared to termination loss. Therefore, let us simply say that \( L(z) \approx T_f(z) \) is the force wave transfer function at the bridge, that the string is rigidly terminated at the other end so that the force wave transfer function there is unity, and that the dispersion, \( A(z) \), is entirely due to stiffness in the string, and not due to any significant reactive qualities in the bridge. We may therefore write [2],

\[
L \triangleq T_f = \frac{R_b - R_0}{R_b + R_0}
\]

and solve for \( R_b \) in terms of \( L \),

\[
R_b = R_0 \frac{1 + L}{1 - L}
\]

The coupling filter for \( N \) strings coupled at an impedance \( R_b \) is

\[
H_b \triangleq \frac{2}{N + R_b/R_0} = \frac{2}{N + \frac{1 + L}{1 - L}} = \frac{2(1 - L)}{(1 + N) + (1 - N)L} \xrightarrow{N=3} \frac{1 - L}{2 - L}
\]

In summary of this calibration approach, we have measured the sound of a single string decaying, derived the loop loss filter from this data, then re-interpreted this to be the force wave transfer function at the bridge (assuming that almost all of the loss is due to the yielding bridge); from this point, we derive the bridge impedance and thence the \( N \)-string coupling filter.

In the model shown in Figure 10, we have three strings coupled, \( N = 3 \). However, several minus signs have been commuted around in that figure and the Coupling Filter is actually represented by \(-H_b\). To complete the model, the Tuning Filters should be tweaked by a good piano tuner to achieve a fine, full-bodied two-stage decay rate (around 0.4–1.7 Hz detuning between strings).

References


