CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS
AUGUST 1996

Department of Music
Report No. STAN-M-99

CCRMA PAPERS PRESENTED AT THE 1996 INTERNATIONAL COMPUTER MUSIC CONFERENCE HONG KONG

Chris Chafe, Alex Igoudin, David Jaffe, Matti Karjalainen, Tobias Kunze, Scott Levine, Fernando Lopez-Lezcano, Sile O'Modhrain, Nick Porcaro, William Putnam, Pat Scandalis, Gary Scavone, Julius Smith, Tim Stilson, Heinrich Taube, Scott van Duyne

CCRMA
Department of Music
Stanford University
Stanford, California 94305-8180

TABLE OF CONTENTS

Chris Chafe and Sile O'Modhrain
Musical Muscle Memory and the Haptic Display of Performance Nuance 1

Alex Igoudin and Fernando Lopez-Lezcano
CCRMA Studio Report 5

Tobias Kunze and Heinrich Taube
SEE--A Structured Event Editor: Visualizing Compositional Data in Common Music 7

Scott Levine
Critically Sampled Third Octave Filter Banks 11

Effects Processing on Audio Subband Data 19

Fernando Lopez-Lezcano
PadMaster: banging on algorithms with alternative controllers 23

Nick Porcaro, Pat Scandalis, David Jaffe, and Julius Smith
Using SynthBuilder for the Creation of Physical Models 26

William Putnam and Tim Stilson
Frankenstein: A Low Cost Multi-DSP Compute Engine for Music Kit 28

Gary Scavone
Modelling and Control of Performance Expression in Digital Waveguide Models of Woodwind Instruments 31

Julius Smith and Matti Karjalainen
Body Modelling Techniques for String Instrument Synthesis 35

Tim Stilson and Julius Smith
Alias-Free Digital Synthesis of Classic Analog Waveforms 43

Analyzing the Moog VCF with Considerations for Digital Implementation 47

Scott van Duyne and Julius Smith
The 3D Tetrahedral Digital Waveguide Mesh with Musical Applications 59

Musical Muscle Memory and the Haptic Display of Performance Nuance

Chris Chafe
cc@ccrma.stanford.edu

Sile O'Modhrain
sile@ccrma.stanford.edu

Center for Computer Research in Music and Acoustics, Music Department
Stanford University

Abstract

We have begun exploring extraction and editing of nuances of a performance through the sense of touch. Expressive variations in MIDI piano recordings were obtained, limiting the initial study to timing and velocity information. A force-feedback interface displays in real time an analysis of the performer's musical conception and can be used to graft aspects of one performance onto another.

1 Introduction

A challenging analysis problem has haunted one of the authors for years, usually mentioned in terms of how synthesis could benefit from a deeper understanding of performance. Posed as conjecture, it's to imagine if two string quartets were to perform the same piece on different nights: the first night's performance is competent, and the audience is happy enough about it. The second night the performance is simply stunning, transcendent, and the audience leaves ecstatic. Part of the problem poses the question of imagining the differences in terms of quantities which would be acoustically-measurable differences between the performances. A second, possibly more difficult part of the problem, is in comprehending such a wealth of detail so that the analysis is imageable and useful.

A second interest motivating this study is to further exploit the sense of touch in music editing tasks. Beyond automated mixer controls, digital editing involves only display to the eye and ear. However, in the physical creation of music, sounding events are registered by the hand and ear [Chafe, 1993] [Gillespie, 1995]. Present digital technology can be adapted to incorporate the kinesthetic (muscular), tactile, and vibro-tactile (cutaneous) senses, modalities well-suited for data that depicts time and motion.

Performances of the same music can have vastly different feelings even when constrained by a fully-notated score. For simplicity, a short piano excerpt was chosen for this study and independent renditions were compared in terms of event timings and key velocities. As listeners, we are acutely sensitive to these differences, but it is more likely that we are only aware of their aggregate effect, for instance, the feeling that one passage was played more forcefully than another. What are the note-level differences, how are they structured, and are such structures the basis for the affect?

The hope that differences of affect can be characterized and displayed leads to the further possibility of manipulating recorded or synthesized performances. A computer-controlled force-feedback interface was programmed to display aspects of performance and manipulate them in real time. Haptic display has the advantage of communicating directly to the motor senses, the same that are involved in musical performance. The word "haptic" is employed to describe devices that engage both the kinesthetic and tactile senses. In our work, the quantities displayed to the observer are ideally a replay or recasting of human motor commands which might have created or accompanied a performance. The end-result is a prototype system that allows the observer to feel musical *feeling* through the real-time display of parameters analyzed from performance. Because the controller permits direct interaction with its display, the performance can be edited in an intuitive manner.

2 Method

An excerpt from the opening of Beethoven's Piano Sonata, Opus 109, was recorded by two excellent pianists using a Yamaha Disklavier grand piano, Figure 1. Recorded data was transferred into standard MIDI file format and analyzed in several steps (with the Stella programming environment, a Lisp package for symbolic musical manipulation [Taube, 1993]). First, the two performances were

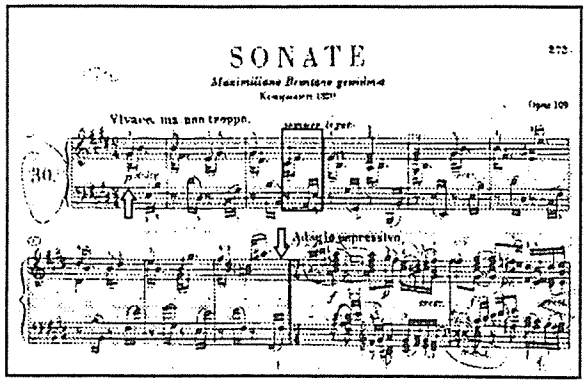


Figure 1: *Two performances of the opening of Beethoven's Piano Sonata, Opus 109, were recorded by Yamaha Disklavier. Note timings and key velocity data were transferred to standard MIDI files.*

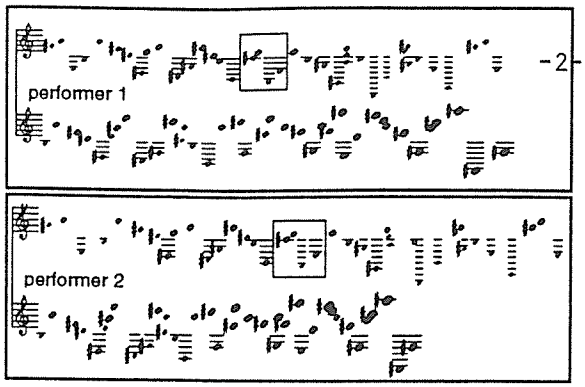


Figure 2: *Distinct short-term shapes are found in raw data displayed from the first 77 notes (marked by arrows in Figure 1). Note placement is proportional to time and size is proportional to key velocity.*

matched up in terms of detected pitches. Our performers were not supervised in any way and were free to submit what they wished. Approximately 2% of the notes did not match up for a variety of reasons, including wrong notes and order differences in chords. Since our project is ultimately directed at acoustically recorded performances, and we expect an even greater error rate in the transcription process, this level of mismatch was acceptable [Chafe and Jaffe, 1986]. A matching algorithm was applied, working from the beginning of the data and pairing equivalent pitches between the two performances. Discrepancies were eliminated and the resulting data set of matched pitches provided the basis for initial experimentation.

2.1 The Moose

Performance data was transferred to a program written in C++ commanding a MIDI synthesizer and the moose, a two-dimensional haptic display device. The moose is essentially a powered mouse-like pointing device. It consists of a puck or manipulandum in the center coupled to two linear voice-coil motors through two perpendicularly oriented flexures. The double flexures conveniently decouple the 2-axis motion of the puck into two single-axis motions at the linear motors. The puck's motion is restricted to an area in the horizontal plane approximately 3 inches on a side.

The moose was designed as part of a larger project based at CSLI, Stanford, to investigate the possibility of using haptic technology to display elements of graphical user interfaces such as window edges, buttons, etc. to blind computer users [O'Modhrain, 1995]. The prototype display has proven the feasibility of the approach, and will continue to be developed alongside our exploration into the use of haptics as a component of a digital music editing systems.

2.2 First Results

Restricting the data to the first eight and a half measures of the Beethoven focused initial analysis on a passage consisting only of running sixteenth-note rhythms. For further simplification, pedal information and durations were ignored. The collected note onsets and key velocities show short-term shapes superimposed on longer-term phrasings. The moose was programmed to directly display key velocity data in the form of an elastic wall. The observer presses the puck to a virtual wall whose stiffness depends on the MIDI velocity being sent to a piano synthesizer. While the performance is sounding, the wall portrays a strong sense of note-to-note variation. As can be seen in Figure 2, some of the note-to-note instantaneous changes are quite abrupt, and a modification was made to display a small mixture of instantaneous key velocity plus a moving average of key velocity whose window is centered on the current note. A rather satisfactory sensation of dynamic phrasing results.

The next refinement consisted of combining onset timings with velocity data to establish an abstract *effort* parameter. Effort, in this sense, represents the directions a conductor might impart to an orchestra. High effort corresponds to faster & louder, low effort to relaxed & softer. However, *ritardando* & *crescendo* can also elicit strong effort, as in the end of the passage studied. A formula to represent these relationships was devised (based on the simplification that the score excerpt only consists of sixteenth notes, which are nominally 125 msec):

$$effort = nv * (1/r + C * r^2)$$

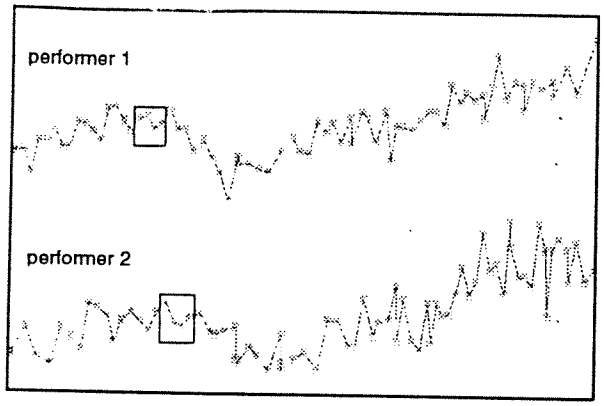


Figure 3: Effort *vs. time is compared for the same passage as Figure 2. The effort quantity is derived from note onset timings and key velocity. Total duration has been normalized for ease of comparison.*

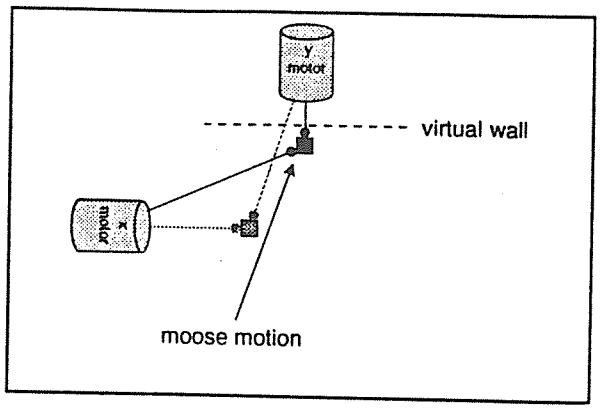


Figure 4: *The moose, a powered mouse, consists of two linear voice-coil motors controlling the location of a puck. Virtual objects and surfaces are displayed by force-feedback. The performance analysis is displayed by changes to a virtual wall's location and compliance in real time while the music is played.*

nv is normalized velocity scaled from 0.0 to 1.0 from the recorded range of velocities, r represents the time interval from the onset of the previous note, and C is a coefficent to bring the nominal rhythm value into range.

Figure 3 shows a graph of effort derived for the same passage as Figure 2. Multi-measure swells correspond to long-term phrasing. Short-term shapes can be seen in note groupings of 2 - 6 notes at a time. The two performances have the keenest difference on this short-term time-scale. Groupings are sometimes similar but shapes are distinct. For example at note 17, a four-note groupin appears (marked by boxes in the figures). Through the effect of a single note, the shape differs between the two performances.

2.3 Manipulations and Muscle Memory

The moose displays the two time scales as separate sensations: a background long-term motion and superimposed, faster foreground shapes. In the background, long-term changes are displayed by averaging the effort parameter with a moving window and causing the virtual wall position to change smoothly. In the foreground, instantaneous effort values affect the wall's compliancy, with higher effort values causing a stiffer spring, Figure 4. The observer quickly trains on differences between the two performances.

A third performance can be created as a product of the first two through linear interpolation of onset rhythms and velocities. The wall's length is used as the interpolation control. At the wall ends, the observer experiences one performance or the other and, in between, an interpolated version. Sliding along the wall in real time allows grafting of one performance to the other.

The prototype system suggests that in-real-time experiences through haptic devices such as the virtual wall can be coupled with sound to offer a rich display for performance analysis and editing. Imaging and memory of patterns is enhanced by appealing to muscle memory. One way to imagine this is to contrast the method with an out-of-time graphical display, such as in Figure 3, or a static haptic display which would project Figure 3 onto a touchable surface. Spatial displays excel for side-by-side pattern discrimination, and performance shapes, such as those briefly discussed, are easily found. Animated spatial displays increase dimensionality, often to include time. The force-feedback system is used to go the other way, to reduce the data into one simplified, intuitive, musical dimension such as the effort parameter. The observer is able to experience, vicariously, the performer's own feeling of effort during performance.

3 Summary: Haptics and Sound

Haptic perception *of the signal* has been lost through changes in music-making technology. The mechanical musical world consists of direct manipulation of sound-producing mechanisms and a sense of their vibration. The analog world replaced this with the feel of various specific control devices or the feel of motion of the recording medium. The digital world has reduced this further to a few general purpose controllers and displays, eg. mouse, keyboard, CRT.

This study has already shown us that there are indeed parameters within music which can be manipulated to allow a performer, composer or mu-

sic editor to traverse the space between two totally different interpretations of the same piece. We have demonstrated that we can make these parameters apparent to the kinesthetic and vibrotactile senses, those same senses which, in live performance, complete the musician's feed-back loop. With a few simple haptic interface tools we can bring back to the editing process some of its former intuitiveness and flexibility [O'Modhrain, 1995].

Specifcally, what we have lost in the transition to mouse-based digital music editing environments is the close contact which sound engineers once enjoyed with their media. We can design new haptic controls and program their "feel" by making them more or less resistant to being moved. A shuttle wheel detent is, for example, easily mediated by motors. And a detent could represent variously manipulator or signal state.

Unique physical operations on sound persist today as metaphors in digital audio editing tools. For example, records are scratched back and forth, tapes are slowed and sped up. The musical arts themselves are strongly influenced by such technologies which often form a basis for new genres of technologically-influenced music. We look forward to enjoying the artistic output inspired by the programmable, multi-modal, and physically coupled interfaces of the future which will feature haptic components.

The authors gratefully acknowledge contributions to the project from our colleagues George Barth, Brent Gillespie, Craig Sapp, and Frederick Weldy. The Archimedes Project at Stanford University's Center for Study of Language and Information provides ongoing support for development of haptic access to graphical user interfaces.

References

[Chafe, 1993] Chris Chafe. Tactile Audio Feedback. *Proceedings of the ICMC, Tokyo*, 1993.

[Taube, 1993] Heinrich Taube. Stella: Persistent Score Representation and Score Editing in Common Music. *Computer Music Journal,* **17**(4), 1993.

[Chafe and Jaffe, 1986] Chris Chafe and David Jaffe. Source Separation and Note Identification in Polyphonic Music. *Proceedings of the IEEE Conference on Acoustics Speech and Signal Processing, Tokyo* (2): pp. 25.6.1-25.6.4, 1986.

[Gillespie, 1995] Brent Gillespie. Haptic Display Of Systems With Changing Kinematic Contraints: The Virtual Piano Action. *Dissertation, Dept. of Mechanical Engineering, available as Stanford Music Department Report STAN-M-92* Stanford University, 1995.

[O'Modhrain, 1995] Sile O'Modhrain. The Moose: A Haptic User Interface For Blind Persons With Application to the Digital Sound Studio. *Stanford Music Department Report STAN-M-95* Stanford University, 1995.

CCRMA Studio Report

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

1.0 The place and the people

The Stanford Center for Computer Research in Music and Acoustics (CCRMA) is a multi-disciplinary facility where composers and researchers work together using computer-based technology both as an artistic medium and as a research tool. CCRMA is located on the Stanford University campus in a building that was refurbished in 1986 to meet its unique needs. The facility includes a large quadraphonic experimental space with adjoining control room/studio, an all-digital recording studio with adjoining control room, a MIDI-based small systems studio, several work areas with workstations, synthesizers and speakers, a seminar room, an in-house reference library, classrooms and offices.

For a detailed tour and more information feel free to visit us in the World Wide Web:

- **http://ccrma-www.stanford.edu/**

The CCRMA community consists of administrative and technical staff, faculty, research associates, graduate research assistants, graduate and undergraduate students, visiting scholars and composers, and industrial associates. Departments actively represented at CCRMA include Music, Electrical Engineering, Mechanical Engineering, Computer Science, and Psychology. CCRMA has developed close ties with the Center for Computer Assisted Research in the Humanities (CCARH), recently affiliated with the Department of Music.

Staff & Faculty: **Chris Chafe**-Associate Professor of Music, Director; **Johannes Goebel**-Technical Director; **Fernando Lopez-Lezcano**-System Administrator/Lecturer; **Heidi Kugler**-Secretary; **Jay Kadis**-Audio Engineer/Lecturer; **Max Mathews**-Professor of Music (Research); **Jonathan Berger**-Associate Professor of Music; **Julius Smith**-Associate Professor of Music and Electrical Engineering; **John Chowning**-Professor of Music, Emeritus; **Leland Smith**-Professor of Music, Emeritus; **John Pierce**-Visiting Professor of Music, Emeritus; **Earl Schubert**-Professor of Speech and Hearing, Emeritus; **Jonathan Harvey**-Professor of Music; **David Soley**-Assistant Professor of Music; **Eleanor Selfridge-Field**-Consulting Professor of Music; **Walter Hewlett**-Consulting Professor of Music; **Marcia Bauman**-Research Associate, IDEAMA Archive; **William Schottstaedt**-Research Associate.

2.0 The activities

Center activities include academic courses, seminars, small interest group meetings, spring and summer workshops, and colloquia. Concerts of computer music are presented several times each year with an annual outdoor computer music festival in July. In-house technical reports and recordings are available, and public demonstrations of ongoing work at CCRMA are held periodically.

3.0 The research

This array of research summaries will give you an idea of the current crop of research at CCRMA and who's doing it:

Computer Music Hardware and Software: "PadMaster, an Interactive Performance Environment. Algorithms and Alternative Controllers", "A Dynamic Spatial Sound Movement Toolkit" **Fernando Lopez Lezcano**; "ATS: Analysis/Transformation/Synthesis; A Lisp Interface for SMS (Spectral Modeling Synthesis; and CLM (Common Lisp Music)" **Juan Carlos Pampin**; "SynthBuilder---A Graphical SynthPatch Development Environment" **Nick Porcaro** and **Pat Scandalis**; "Franken Hardware: On Scalability for Real-Time Software Synthesis and Audio Processing" **Bill Putnam** and **Timothy Stilson**; "Common Lisp Music and Common Music Notation" **William Schottstaedt**; "Music Synthesis and Digital Audio Effects for UltraSparc Processor" **William Putnam**, **Tim Stilson**, and **Julius Smith;**"Rapid Prototyping for DSP, Sound Synthesis, and Effects" **Julius Smith; "**SynthScript - A Sound Synthesis Description Format" **Pat Scandalis, David Jaffe, Nick Porcaro**, and **Julius Smith**; "The CCRMA Music Kit and DSP Tools Distribution" **David Jaffe** and **Julius Smith**; "Capella: A Graphical Interface for Algorithmic Composition" **Heinrich Taube** and **Tobias Kunze**.

Physical Modeling and Digital Signal Processing: "Physical Modeling of Brasses" **David Berners**; "Adding Pulsed Noise to Wind Instrument Physical Models" **Chris Chafe**; "Synthesis of the Singing Voice Using Physically Parameterized Model of the Human Vocal Tract" **Perry Cook**; "Synthesis of Transients in Classical Guitar Sounds", "The "Flutar" a New Instrument for Live Performance" **Cem Duruoz**; "Spectral Operators for Timbral Design" **Jose Eduardo Fornari**; "Voice Gender Transformation with a Modified Vocoder" **Yoon Kim**; "Processing of Critically Sampled Audio Subband Data" **Scott Levine**; "Feedback Delay Networks" **Davide Rocchesso**; "Acoustical Research on Reed Driven Woodwind Instruments for the Purpose of Efficient Synthesis Models" **Gary Scavone**; "FFT-Based DSP and Spectral Modeling Synthesis", "Digital Waveguide Modeling of Acoustic Systems" **Julius Smith**; "A Passive Nonlinear Filter for Physical Models" **John Pierce** and **Scott Van Duyne**; "The Digital Waveguide Mesh" **Scott Van Duyne** and **Julius Smith**; "The Wave Digital Hammer" **Scott Van Duyne** and **Julius Smith**; "The Commuted Waveguide Piano" **Scott Van Duyne** and **Julius Smith**.

Controllers for Computers and Musical Instruments: "Real-time Controllers for Physical Models" **Chris Chafe** and **Perry Cook**; "Ongoing Work in Brass Instrument Synthesizer Controllers" **Perry Cook** and **Dexter Morrill**; "The Touchback Keyboard" **Brent Gillespie**; "The Computer-Extended Ensemble" **David Jaffe**; "Haptic User Interfaces for the Blind" **Sile O'Modhrain** and **Brent Gillespie**; "The Radio Baton Progress" **Max Mathews**; "Optimal Signal Processing for Acoustical Systems" **Bill Putnam**; "Signal Processing Algorithm Design Stressing Efficiency and Simplicity of Control" **Timothy Stilson**.

Psychoacoustics and Cognitive Psychology: "Distance of Sound in Reverberant Fields" **Jan Chomyszyn**; "Embedded Pitch Spaces an The Question of Chroma: An Experimental Approach" **Enrique Moreno**; "Pitch Perception" **John Pierce**; "Psychological Representation of English Vowel Sounds" **Roger Shepard**, **Perry Cook**, and **Daniel Levitin**; "Applying Psychoacoustic Phenomena to the Coordination of Large Speaker Arrays" **Steven Trautmann**.

Computer Music and Humanities: "The International Digital Electroacoustic Music Archive" **Max V. Mathews** and **Marcia L. Bauman**; "The Catgut Musical Acoustics Research Library" **Max Mathews** and **Gary Scavone**; "Impact of MIDI on Electroacoustic Art Music in the mid-1980s" **Alex Igoudin**; "The Chorister-Chorister Interaction: an Ethnography" **Paul von Hippel**.

4.0 The music

Some of the recent (during this past year) compositional works realized at CCRMA:

Michael Alcorn (Visiting Composer / Ireland) -*Double Escapement* (for piano and tape); **alt.music.out**-*Wonderment in Eb* - live jazz/electroacoustic fusion involving 7 performers, several NeXTs, drums and vocals; **Chris Chafe** (CCRMA Director)-*Push Pull*, for Celletto and live electronics; **Cem Duruoz** (MA Graduate Student)-*Cycles*, interactive piece for classical guitar, NeXT (physically modeled SynthBuilder Flute), and Mac (sequencer); **Michael Edwards** (DMA Student) and **Marco Trevisani**-*segmentation fault beta1.0*, for prepared piano and computer; **Doug Fulton** (PhD Graduate Student)-*Holding Betty under Water* for computer generated tape; **David Jaffe** (Visiting Composer / Researcher) - 5th and 6th movements of *The Seven Wonders of the Ancient World*, for Mathews/Boie Radio Drum-controlled Disklavier and an ensemble of plucked string and percussion instruments; **Nicky Hind** (DMA Graduate Student)-*Awakening*, computer-generated sound installation for the 18-th century garden; **Jun Kim**- Reverberation, for two sopranos, percussion and computer processed sounds on tape; **Peer Landa** (Visiting Composer / Norway)-*Downcast* for tape using original C-based software; **Lukas Ligeti** (Visiting Composer / Austria)-*New Music for Electronic Percussion*, performed and processed live, inspired by African drum music; **Fernando Lopez Lezcano** (System Administrator / Lecturer)- *Three Dreams*, tape piece using CLM, performance involves pre-programmed four channel spatialization; *Espresso Machine II* and *With Room to Grow*, in which PadMaster splits the Radio Drum surface into programmable virtual pads, grouped in sets or "scenes"; **Jonathan Norton** (PhD Graduate Student)-*Vicissitudes*, computer music for a documentary about a striving African-American community; **Fiammetta Pasi** (Visiting Composer / Italy)-*Collage*, for stereo tape; **Juan Pampin** (PhD Graduate Student)-*Transcription #1*, for computer controlled Disklavier; **Jorge Sad** (Visiting Composer /Argentina)-*VOX*, *VOXII*, for computer originated tape; **Marco Trevisani** (Visiting Composer / Italy) *Frammenti e Variazioni su Aura*, a Bruno Maderna inspired tape composition.

Recent awards won by CCRMA composers:

Celso Aguiar, for *Piece of Mind*, "Premio Sao Paulo '95", Brazil; **Chris Chafe**, National Endowment for the Arts Composer's Fellowship 1994-95, Green Faculty Fellowship 1995-96; **Kui Dong**, for *Flying Apples*, First Prize, 1994 Alea III International Composition Prize; 1995 Djerassi Foundation for Art, 1995 ASCAP Grants to Young Composers; **David Jaffe** for *The Seven Wonders of the Ancient World*, Collaborative Composer Fellowship, National Endowment for the Arts; **Juan Pampin**, for *Apocalypse was postponed due to lack of interest*, Award, 22e Concours International de Musique Electroacoustique, 1995, Bourges, France; **Jorge Sad**, for *Vox II*, Juan Carlos Paz Electroacoustic Music Prize, 1995, National Foundation for the Arts, Argentina.

SEE—A Structured Event Editor: Visualizing Compositional Data in Common Music

Tobias Kunze
CCRMA, Stanford University
t@kunze.stanford.edu
http://www.stanford.edu/~tkunze

Heinrich Taube
School of Music, University of Illinois
taube@uiuc.edu

Abstract

Highly structured music composition systems such as Common Music raise the need for data visualization tools which are general and flexible enough to adapt seamlessly to the—at times very unique—criteria composers employ when working with musical data. The SEE visualization tool consists of an abstracting program layer to allow for the construction of custom musical predicates out of a possibly heterogenous set of data and a separate program module which controls their mapping onto a wide variety of display parameters. The current version is being developed on a SGI workstation using the X11 windowing system and the OpenGL and OpenInventor graphics standards, but portability is highly desired and upcoming ports will most probably start out with the Apple Macintosh platform.

1 Introduction

Among the vast variety of ways in which music has been put in relation to the visual senses, only a few have been researched or otherwise developed to a noticeable extent. Today, sound and graphics interconnect most prominently in the audiovisual domain, that is in multimedia applications and—more recently—in the area of data sonification, but also in the more arcane areas of music visualization and graphical user interface design as well as in art. These domains, however, are not unrelated: sonification of data may be taken as an inverse process of music visualization and music visualization itself leads seamlessly to music data *manipulation* as in some GUI-designs: it may form half of a genuine music authoring environment. Musical data visualization could, it seems, profit on the other hand from the extensive computer graphics technology and visualization experience scientific data visualization projects today rely upon. In contrast to scientific visualization applications, however, music visualization does not typically deal with an enourmous amount of "flat" data such as data masses acquired by satellite photos, surveys or oceanographic sonic measurements: musical datasets are most often comparably small—but generally include heterogenous and not necessarily commensurable datatypes such as, for instance, notes and rests. They also tend to call for interpretation processes that *evolve* over time to model the changing belief contexts that characterize musical hearing. In short, music visualization differs from data visualization in that it deals with our understanding of music. And musical data, unlike scientific data, may be arbitrarily changed according to the aestethic criteria we decide to apply.

2 Visualization Today

Although a number of promising approaches to signal visualization to faciliate the process of sound (re)creation exist, research in visualization of compositional data is rare and focuses on musicology as opposed to creative applications. More recent examples include the analysis of features of music by Bartók and Webern in the graphical plane by A. Brinkman and M. Mesiti [2] and J.-P. Boon's interesting examination of significant differences between three-dimensional phase portraits of selected three-part compositions by Bach, Mozart, and Schumann [1]. The majority of musicological analysis toolkits, however, doesn't go beyond a symbolic representation of their results (cf., for instance [3]).

Alternative approaches to signal visualization leading to graphical representations of higher-order features of sound data that approximate complexer musical predicates in a raw manner have been presented previously by J. Pressing et al. and B. Mont-Reynaud [8, 7] as well as, most recently, by B. Feiten and G. Behles [5].

Finally, I. Choi et al. [4] and Y. Horry [6] document some specific research into the application of musical concepts using graphical controllers to generate both MIDI and digital sound output.

Graphical user interfaces for compositional oriented software today has begun to venture into the domain of 3D, with more and less success and not

much inspiration. EMAGIC's Notator Audio, for instance, consistently encourages a three-dimensional, albeit conceptual, view of the compositional data. Nevertheless, it sticks with a set of two-dimensional editors to support editing operations and sells a simple two-dimensional control for the independent transposition and stretching of soundfiles in unnecessary perspective 3D look.

Visualization is a central field in the analysis of particularly vast scientific data sets and as such features the most advanced visualization solutions found today. 3D rendering techniques, as well as techniques originally intended to simulate environmental data such as transparency, haze, fog and texture mapping, are widely used to represent abstract qualities such as density or velocity. Also, most scientific visualization packages allow for multiple different graph styles to be combined in a single package. In addition, it had and still has a strong influence on the design and development of graphics packages. As a result, music visualization software packages wanting to take advantage of these highly optimized graphic engines have to adapt to a set of primitives that has been mostly designed with the notion of rendering artificial, "virtual reality" worlds. A particularly blatant problem is the missing support in 3D toolkits for seemingly endless scrolling: for a framework that implies the construction of a perspective, simple scrolling doesn't make sense. Most limitations, however, prove in hindsight to be solvable in terms of a different concept, and scrolling (as opposed to panning) then translates into a matter of *animation*.

3 Graphing Features and Paradigms

Since using a 3D graphics library like OpenInventor implies such an essential commitment to its underlying paradigms and since these paradigms are not necessarily congruent with the demands of a music visualization tool, it is wise to render an account of what its needs are and what it can expect. The current design of the SEE visualization tool followed these major guidelines:

- although the 3D paradigm introduces fundamental changes in the representation of music and imposes particularly high demands on the system, it may be "frozen" to a two-dimensional scene by using an orthographic camera from a front view position and kept reasonably efficient by providing the graphics library with optimizing hints regarding the (invisible) 3D information
- color is widely available today and thus highly recommended for use in data visualization; a particular visualization, however, does not have to use color
- 3D animation is believed to be extremely useful but is still too unexplored to be included in this version of the SEE design
- extensibility of the 3D object library is highly desired on both a scripting and a programming level
- similarly, the 3D toolkit should have provisions to access underlying (2D) graphics libraries
- display styles as well as annotations of axes and various grids should be easy to specify
- different scenes need support for different viewing modes such as spin, fly-through, etc. through reusable components or external applications

The OpenInventor graphics standard meets these guidelines and offers in addition a straightforward interchange file format as well as near compatibility with the current VRML standard.

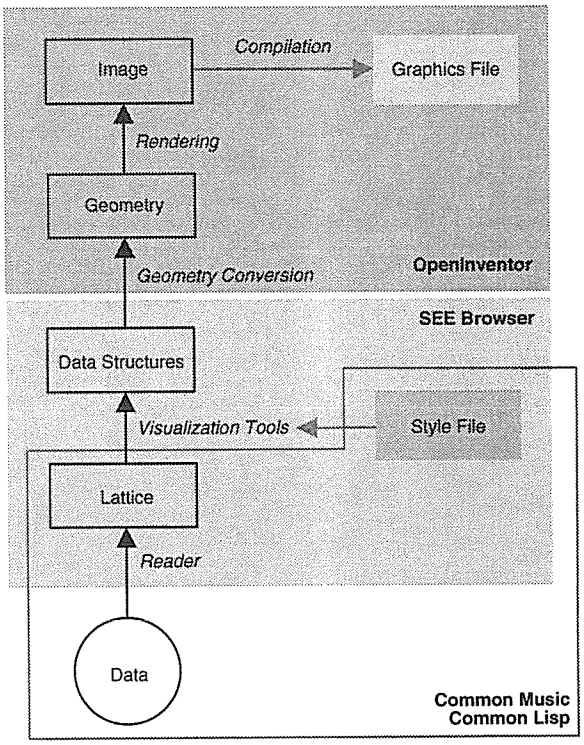


Figure 1: SEE Architecture

4 SEE Architecture

Figure 1 gives an coarse overview of the steps involved in the process of creating a visualization and the tasks SEE has to perform. For the translation from raw data into a unified lattice of raw data structures SEE provides standard readers as well as an programming interface for adding custom data readers. Whether or not it is desirable to have a higher-level interface is unclear as of yet, but such an interface might be added at any point. Fully programmable and easily customizable style files then control the construction

of actual 3D data structures. Graphics tasks, such as global geometry conversion and rendering itself, is taken care of by the OpenInventor toolkit and other custom components, as is the conversion to the Inventor interchange file format.

Since major parts of SEE are implemented as an extension to Common Music's graphical interface, *Capella*, data, and data readers and the visualization style files are written in Lisp. It uses CommonMusic's score representation toolbox and class hierarchy to implement readers with a high degree of polymorphism.

```
1  p1 ← 32
2  p2 ← 37
3  p3 ← 36
4  i ← 0
5  while i ≤ 300
6     do p ← p2 − p3
7        if p ≥ 0
8           then p ← p − 11
9           else p ← p + 13
10       p ← p + p1
11       if p < 24 or p > 108
12          then p ← (p − (36 + mod[p, 12])) + 1
13    p3 ← p2 ← p1 ← p
14    WRITENOTE(note: p, time: i)
15 return
```

Figure 2: Automaton in pseudocode notation

5 An Example Run

To give a complete example of a process of music visualization using the SEE tool, consider the algorithm given in Figure 2. It sets up a history of three pitch variables, initialized to 32, 37, and 36, respectively, to generate 300 notes of a monophonic, pulsating line according to the code in the `while` loop. For simplicity, amplitude, duration and instrumentation have been assigned default values. Figure 3 gives a rendering of this—deterministic—algorithm's musical output in common music notation. Not obvious from the code but readily readable from the score are the all-intervallic structure of the melodic sequence and the strong reduction of the musical material.

In a first step, a default reader translates every event into a cube of unit size and no color, using exclusively the time and pitch information to map it onto the x and y axis, respectively. The camera set to orthographic mode, the display resembles traditional piano-roll notation showing 3 different rising patterns and striking symmetries everywhere that were hard to spot in the score (cf. Figure 4).

Spinning the model in all three dimensions gives an even better overview of the combined pitch-time structure. Figure 5 zooms in on an angle from the bottom-left corner along the slope of the rising patterns, emphasizing the mechanical aspect of the cellular automaton's repetitiveness visible. Moreover, rotating the model clockwise and to the left reveals another set of symmetry axes, perpendicular to each pattern's slope (cf. Figure 6). A zoom on the first pattern together with an added headlight shows the slant of the current viewpoint (cf. Figure 7). As a last example, Figure 8 uncovers the common focal point of every other triad around the center of pattern 3 as approximately an octave above the middle note.

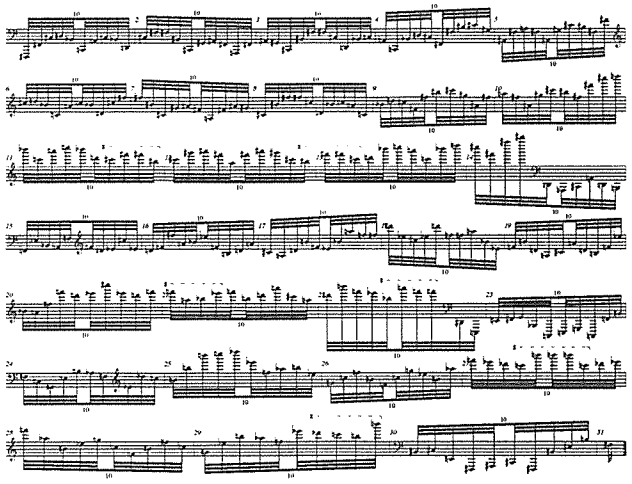


Figure 3: Automaton in traditional notation

Figure 4: Automaton in traditional "piano-roll" notation

Using different data reader methods, the same automaton has been colored according to each note's pitch class and grouped with a vertical line indicating each note's interval in relation to the preceding note. To faciliate optical references, a grid of dotted ledger lines, solid piano system lines and vertical reference lines have been added (cf. Figure 9). A bottom view of this model finally reveals the two dove-tailed states of the automaton, each of which cycles through all intervals, alternating between positive and complementary negative values.

References

[1] **Jean-Pierre Boon** et al.: "Complex Dynamics and Musical Structure" in *Interface* Vol. 19, pp. 3–14 (Lisse, the Netherlands: Swets & Zeitlinger B.V., 1990)

[2] **Alexander R. Brinkman** and **Martha R. Mesiti**: "Computer-Graphic Tools for Music Analysis" in *Proceedings of the 1991 ICMC*, pp. 53–56 (Montreal, Canada: McGill University, 1991)

Figure 5: View from the bottom-left corner

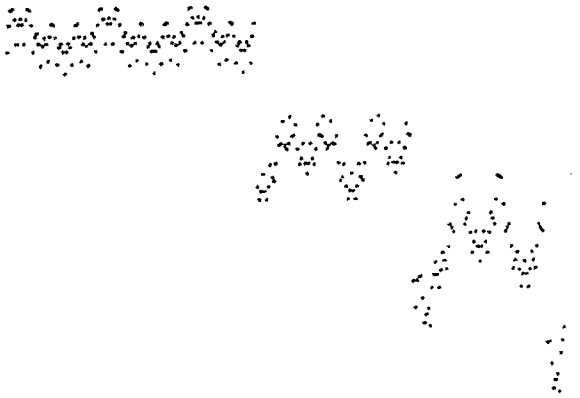


Figure 6: Alternate symmetry axes revealed

[3] **Peter Castine**: "Whatever Happened to CMAP for Macintosh" in *Proceedings of the 1993 ICMC*, pp. 360–362 (Tokyo, Japan: Waseda University, 1993)

[4] **Insook Choi** et al.: "A Manifold Interface for a High-Dimensional Space" in *Proceedings of the 1995 ICMC*, pp. 385–392 (Banff, Canada: The Banff Centre for the Arts, 1995)

[5] **Bernhard Feiten** and **Gerhard Behles**: "Organizing the Parameter Space of Physical Models with Sound Feature Maps" in *Proceedings of the 1994 ICMC*, pp. 398–401 (Aarhus, Denmark: Danish Institute of Electroacoustic Music, 1994)

[6] **Youichi Horry**: "A Graphical User Interface for MIDI Signal Generation and Sound Synthesis" in *Proceedings of the 1994 ICMC*, pp. 276–279 (Aarhus, Denmark: Danish Institute of Electroacoustic Music, 1994)

[7] **Bernard Mont-Reynaud**: "SeeMusic: A Tool for Music Visualization" in *Proceedings of the 1993 ICMC*, pp. 457–460 (Tokyo, Japan: Waseda University, 1993)

[8] **Jeff Pressing** et al.: "Visualization and Predictive Modelling of Musical Signals using Embedding Techniques" in *Proceedings of the 1993 ICMC*, pp. 110–113 (Tokyo, Japan: Waseda University, 1993)

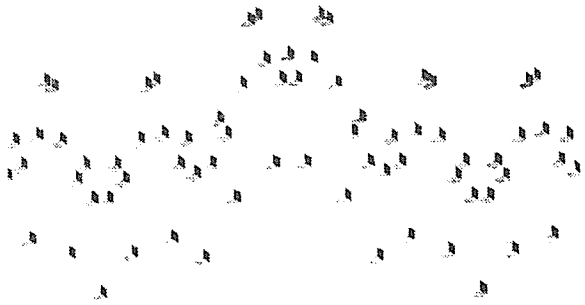


Figure 7: Slant of the current angle of view

Figure 8: Using perspective: focal point of the rising lines in pattern 3

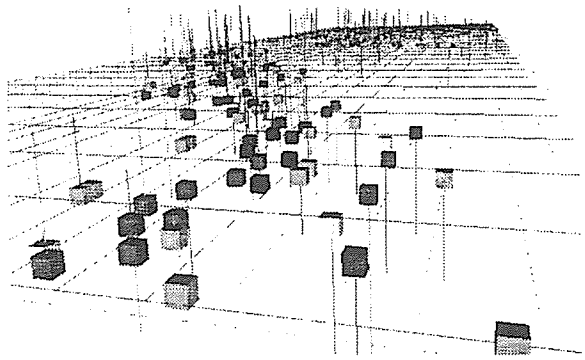


Figure 9: Colored model with lines indicating the note's interval in relation to the preceding note

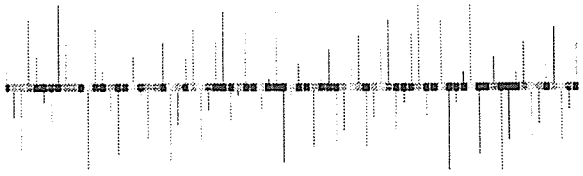


Figure 10: Bottom view shows the behaviour of the automaton's two substates

Critically Sampled Third Octave Filter Banks

Scott N. Levine
Stanford University
Center for Computer Research in Music and Acoustics (CCRMA)
scottl@ccrma.stanford.edu

Abstract

This paper introduces the design of a critically sampled, third octave filter bank. Filter design methods are shown for the octave band filter bank and the third octave sections. In addition, the trade-offs in the design are explained among frequency selectivity, regularity, complexity, latency, and memory.

1 Introduction

The front end of most current audio data compression algorithms use some sort of time-frequency representation to transform the time domain signal to some other domain more closely matched to the human ear. One accepted model is often called the critical band model, which shows how signals psychoacoustically mask one another, as long as they are within a critical band [Zwicker, 1990]. A close approximation to the critical bands of hearing is a third octave filter bank, which is designed in this paper. Thus, if a signal were bandlimited within a critical band, and then quantized, the resultant quantization noise would remain within the band and would be perceptually masked by the original signal. This is the reason that tight frequency selectivity will become an important parameter to optimize later in the paper.

To implement this third octave filter bank, the system first decomposes the input signal into octaves. Then, the octave band signals are further split into third octave sections. Along the way, we will discuss various filter design methods, along with the resulting trade-offs between complexity, latency, and memory.

2 Octave Band Filter Banks

The first step in designing a critically sampled third octave filter bank is to split the input signal into critically sampled octaves. A filter bank is *critically sampled* if the data rate of the original input signal is equal to the sum of the data rates of the transform subbands. Octave band filter banks, also known in the literature as tree-structured filter banks or discrete-time wavelet transforms (DTWT), has been well researched for years [Vetterli and Kovačević, 1995].

The simplest octave band decomposition is a single octave filter bank. This is also known as a two channel, uniform filter bank as seen in figure 1. In orthogonal systems, the lowpass analysis filter, $H_0(z)$, determines the other three filters in the two channel structure.

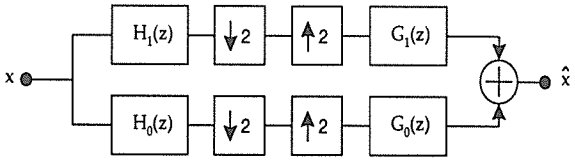


Figure 1: *Two channel, uniform filter bank*

In an octave band decomposition, the lowpassed, downsampled output is iterated through another two channel, uniform filter bank as seen in figure 2. If the original two channel structure is perfect reconstruction, then the iterated octave band tree is also.

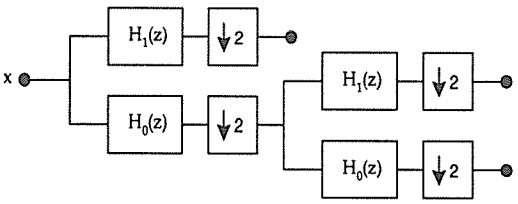


Figure 2: *A three channel, two octave analysis filter bank*

The optimal $H_0(z)$ would be a brick wall lowpass filter, with a cutoff frequency of $\pi/2$. Since this would be an infinite length filter, we must make several trade-offs among frequency selectivity, regularity, and filter length.

2.1 Frequency Selectivity

For the sharpest transition between passband and stopband (tightest frequency selectivity) regions, we first try designing a low pass filter using the remez exchange algorithm [Parks and McClellan, 1972]. It has been

shown that its transition band width is on the order of $\frac{1}{L}$, where L is the length of the FIR filter [Strang and Nguyen, 1996]. But, the smaller the transition region (the tighter the frequency selectivity), for a given length FIR filter, the higher the equiripple sidelobes level will rise. As can be seen in the filter's magnitude responses in figure 3, the aliasing noise floor is at $-70dB$ and the transition width is approximately 0.07π, for $L = 64$.

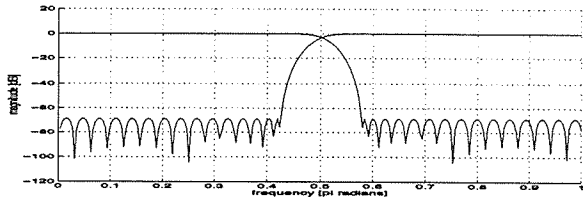


Figure 3: *Magnitude responses of the $H_0(z)$ and $H_1(z)$*

We attempt to reduce the transition width while satisfying the QMF flatness condition, $|H_0(e^{j\omega})|^2 + |H_0(e^{j(\omega+\pi)})|^2 = 1$ [Crochiere and Rabiner, 1983], within some error tolerance. As seen in figure 4, and the ripple of $\pm 0.05dB$.

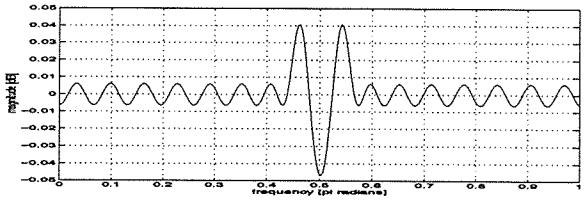


Figure 4: *Reconstruction error with 64 tap QMF filters*

2.2 Regularity

It has been shown that regularity of these filters is an important property for image compression [Rioul, 1993], but there is not a general consensus in the audio compression community. While some say there is no correlation between bit rate and regularity [Philippe, 1995], others mention only a significant improvement for regularity of very short filters (around 4 taps) [Kudumakis, 1995].

Loosely, a filter bank with regular filters will have a smooth impulse response for the lowest frequency subbands. That is, most of the lowest subbands' energy will be bounded within only the low frequencies. Daubechies has shown that a sufficient condition for a filter to be regular would be having some amount of zeros at π [Daubechies, 1988]. If a filter is of length $2N$ and there are N zeros at π, then this becomes a maximally flat Daubechies filter.

The effects of regularity in the frequency domain can be seen in figure 5. The lower figure corresponds to the magnitude response of the lowest subband of a DTWT using the Daubechies maxflat filter, which corresponds to its scaling function. Notice how the energy for this subband is bounded by 0.1π. But, for the upper plot which corresponds to the remez exchange designed lowpass filter (which has no regularity), the scaling function has higher frequency sidelobes of equal height. The longer the remez exchange filter, the lower the sidelobes will be. Both of the shown plots were calculated by iterating length 32 filters.

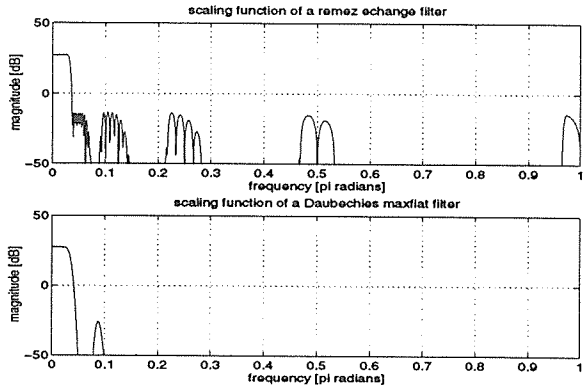


Figure 5: *The magnitude responses of remez exchange (top) and Daubechies maxflat (bottom) iterated filters*

Unfortunately, this high degree of regularity reduces the frequency selectivity, and only has a transition bandwidth on the order of $\frac{1}{\sqrt{L}}$. This Daubechies filter can be contrasted to the remez exchange filter designed in the previous section with no zeros present at π, but with good frequency selectivity.

As a compromise between these two extremes, a third filter was designed with most of the frequency selectivity gained from a standard remez exchange algorithm, but with a varying amount of zeros imposed at π for regularity. The more zeros that are added at π, the more the higher frequency sidelobes, as seen in the top of figure 5, are attenuated.

In figure 6, the difference in magnitude responses between the three 32 tap filters can be seen. Notice that the plots using the remez exchange and the wavelet filter with only four zeros at π [O.Rioul, 1994] are almost identical; except at π, where the dotted line drops off to $-\infty$. Also, notice that the maxflat Daubechies filter has relatively poor frequency selectivity.

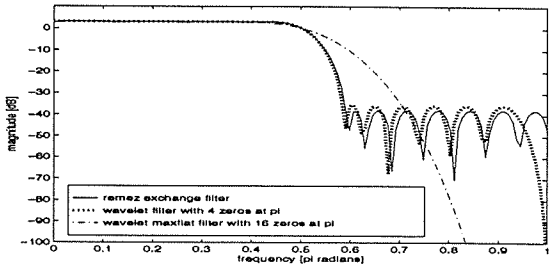


Figure 6: *The magnitude responses various lowpass filter designs*

In conclusion, the trade-offs in this DTWT filter design is between high frequency selectivity and high regularity. A filter designed with the remez exchange algorithm will have high frequency selectivity, but will also have sidelobes in the subbands' iterated filters. The filter with high regularity has poor frequency selectivity, but has very small sidelobes in the iterated filters. The design chosen in this system was to generate long remez exchange filters (64 taps), such that the iterated filters' sidelobes were pushed down $70dB$. Thus, we maintain high frequency selectivity, while the sidelobes are low enough to be considered the noise floor for the application. It will be seen in section 3.2 that the noise floor for the third octave sections will also be placed down $70dB$, thus matching the characteristics of the DTWT filters.

It seems that placing zeros at π made the filters more regular and lowered the sidelobes of the scaling function, but the sidelobes were not lowered enough to warrant losing the high frequency selectivity. In addition, certain algorithms that produced the wavelet filters were not numerically stable above about 40 taps, but the remez exchange algorithm (as implemented in Matlab), could easily compute over one hundred taps.

3 Third Octave Filter Banks

Once the signal has been split into critically sampled octaves, it now gets split into third octave sections. These third octave sections, which can be thought of as uniform bandpass filters on a logarithmic frequency axis, unfortunately have irrational bandwidths. For example, the lowest third octave section (in normalized frequency) will be $r_0 = 2^{1/3} - 1$. The middle section r_1 is of width $2^{2/3} - 2^{1/3}$, and the highest section $r_2 = 2 - 2^{2/3}$.

The subject of splitting a signal into M critically sampled uniform sections of width π/M has been well studied, as well as tree structured banks with bandwidths of $[1/2, 1/4, 1/4]$ or $[1/4, 1/4, 1/8, 1/8]$, for example [Vaidyanathan, 1993].

The topic of constructing perfect reconstruction filter banks with more general rational sampling factors was more recently introduced [Kovačević and Vetterli, 1993]. In this same paper, they mention that third octave sections could be rationally approximated by the factors $[1/4, 1/3, 5/12]$. Notice that these factors will add to unity to guarantee critical sampling.

In this paper, they state two different methods of realizing filter banks with arbitrary rational sampling rates: direct and indirect. The direct method is simpler in complexity and design, but only works for certain rational rates. Unfortunately, the previous third octave fractional sampling rates make this method impossible due to aliasing problems. The indirect method works for any rational rates, but produces frequency shuffling for certain cases. Shuffling denotes the process in which a part of the signal's spectrum has been translated to another part in the spectrum. In order to avoid spectral shuffling, the rates $[6/24, 8/24, 10/24]$ will now be used.

3.1 Indirect Method

The indirect design, as stated in [Kovačević and Vetterli, 1993], first splits a signal into a number of subbands equal to the lowest common denominator of the three third octave sections. In this case, there will be 24 subbands. Then, three groups of subbands will be recombined to form the three, third octave sections. In this design, groups of 6,8, and 10 subbands will be recombined to form the three output signals. This scheme is graphically shown in figure 7.

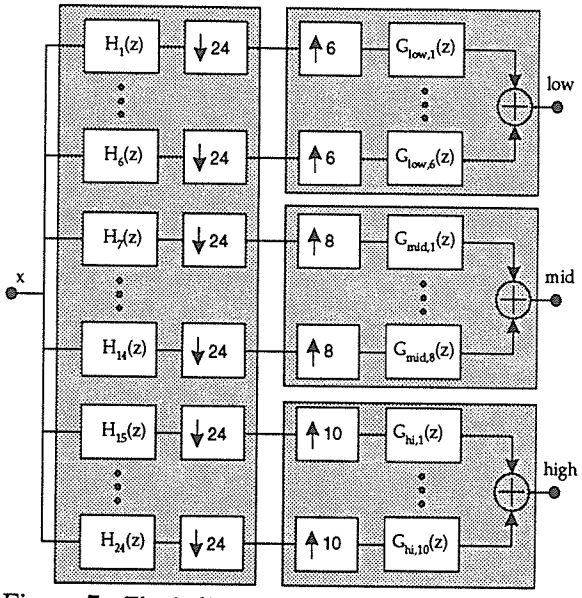


Figure 7: *The indirect method of generating third octave sections*

Each of the shaded boxes in figure 7 are uniform filter banks. The tall box on the left is a 24 channel uniform analysis filter bank, while the three shorter boxes on the right are uniform synthesis filter banks, of 6,8, and 10 channels, respectively. Therefore, it is evident that non-uniform filter banks can be designed from a cascade of uniform banks, while maintaining critical sampling.

3.2 Filter Design

The next design choice is how to design the uniform filter banks shown in figure 7. Because of the low complexity and simple design, we chose to implement pseudo-QMF cosine modulated fil-

ter banks for all the uniform filter banks in this structure [Vaidyanathan, 1993].

For these filter banks, one needs to only design a single lowpass filter prototype, and all other filters will be cosine modulations of this prototype. The prototype lowpass filter for the 8 channel uniform cosine modulated filter bank was downloaded from the WWW site of [Nguyen, 1996], which used the design method of [Nguyen, 1994].

The prototype filters for the 6,10, and 24 channel filter banks were all sample rate converted from this single 8 channel lowpass prototype [Cox, 1986]. The sample rate conversion was performed using upsampling, 32 point lowpass FIR filtering, and downsampling. The ratio of the prototype filter length to the number of filter bank channels is kept constant at a factor of 8 in this design (also known as the overlap factor). The magnitude response of this third octave section can be seen in figure 8.

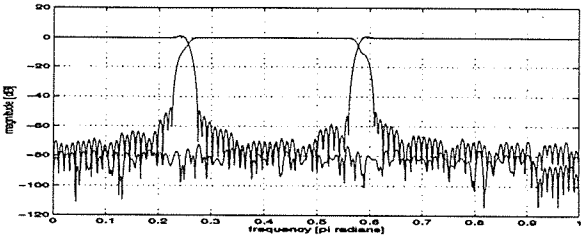


Figure 8: *The third octave section magnitude response*

Notice that at the third octave filter bank boundaries, there is a small amount of ripple, on the order of $1dB$. In most applications, this should not be a problem. To make the response flatter, a "transition" filter could be placed between the uniform filter banks to eliminate any low-level aliasing, as suggested by [Princen, 1995].

With all the pieces now in place, they can be combined to form a critically sampled, third octave filter bank. Initially, we begin with a five octave DTWT structure similar to the one pictured in figure 2. On the output of all the rightmost downsamplers, an individual third octave section, as in figure 7, is placed. An example of the system design of a three channel, two octave case can be seen in figure 9.

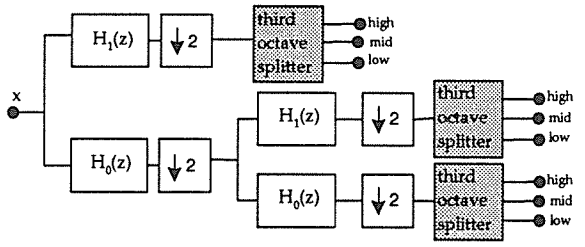


Figure 9: *Three channel, third octave, critically sampled analysis filter bank*

The magnitude response of this system was found by placing an impulse into the system, zeroing out all but one of the third octave sections, and calculating the response. This procedure was performed 15 times, once for each third octave section, as seen in figure 10.

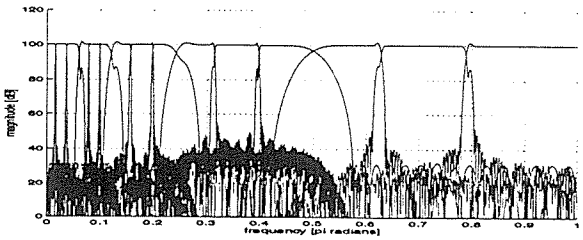


Figure 10: *Third octave filter bank magnitude response*

4 Complexity

While this third octave structure may seem cumbersome, all of the building blocks can be implemented around FFT's and DCT's, which reduces the complexity. First, the complexity of the third octave sections will be discussed, then the DTWT.

4.1 Third Octave Complexity

The complexity of splitting several critically sampled octave band signals into third octave signals is equal to C_{third}, which is independent of the number of octaves present. In fact, C_{third} is equivalent to the complexity of splitting a single fullrate signal into third octave sections.

For example, the complexity of splitting the top octave band signal into third octave sections is $C_{third,1} = \frac{1}{2}C_{third}$. The factor of $\frac{1}{2}$ is due to the fact that the top octave input signal has been downsampled by a factor of two. Given that the lowest two frequency signals have the same data rate, the total complexity for K octaves is:

$$C_{third} = \sum_{i=1}^{K+1} C_{third,i}$$

$$C_{third} = C_{third}\left[\frac{1}{2^K} + \sum_{i=1}^{K}\frac{1}{2^i}\right]$$

$$C_{third} = C_{third} \cdot 1$$

The complexity of a single PQMF cosine modulated filter bank (analysis or synthesis), $C_{QMF}(M,N)$ is on the order of $\frac{N+MlogM}{M}$, where N is the length of the lowpass prototype filter and M is the number of channels in the filter bank. The M in the denominator is because of the decimation by M, N is the cost of FIR convolution, and $MlogM$ is the cost of the cosine modulation using fast DCT's [Konstantinides, 1994].

In the forward third octave splitter, as pictured in figure 7, there is not just one QMF bank, but four. The leftmost bank splits the input into 24

signals, each downsampled by 24. To compensate for this different sampling rate, the complexities of the three synthesis filter banks must be scaled by a factor of $\frac{M_i}{24}, i \in \{low, med, hi\}$. For example, $C_{QMF,low} = \frac{6}{24}C_{QMF}(6, 6l)$, where l is the overlapping factor defined in section 3.2 ($l = 8$ is this design). Therefore, the total third octave complexity for all octaves is:

$$C_{third} = C_{QMF} + C_{QMF,low} + C_{QMF,mid} + C_{QMF,hi}$$

$$= C_{QMF}(24, 24l) + \frac{6}{24}C_{QMF}(6, 6l)$$

$$+ \frac{8}{24}C_{QMF}(8, 8l) + \frac{10}{24}C_{QMF}(10, 10l)$$

In order to synthesize the third octave signals into the original, the same complexity is required. In this system, the total complexity for analysis and synthesis is 47 operations per input sample.

4.2 DTWT Complexity

A convenient fact about the complexity of an octave band filter bank such as the DTWT is that it is bounded by $2C_{DTWT}$, where C_{DTWT} is the complexity involved in one stage of highpass filtering, lowpass filtering, and decimation by two [Rioul and Duhamel, 1992]. Using tricks such as polyphase filtering and FFT overlap-add convolutions, the complexity for a K octave forward and reverse DTWT is on the order of $16 log L(1 - 2^{-K})$, using length L FIR low and highpass filters [Vetterli and Kovačević, 1995]. In this design, the DTWT complexity amounts to 95 operations per input sample.

5 Latency

Along with the complexity, latency is another important metric to consider depending on the application. For any two-way, interactive system, latency must be very low to be useful. If the system only needs to run in real-time, for example decoding compressed audio from file, some amount of latency can be tolerated.

5.1 Third Octave Section Latency

To simplify initial calculations, we will first compute the latency of splitting one fullrate signal into critically sampled third-bandwidth sections. In section 5.3, we will compute the latency of these third octave sections embedded within the subbands of a DTWT.

The delay of any given fullrate pseudo-QMF filter bank of M channels and length N prototype is $D_{QMF}(M, N) = \frac{N-M}{2}$. In the third octave system of figure 7, the left most bank of 24 channels will have a latency of $D_{QMF}(24, 24l) = 84$ samples. The latency of the three parallel QMF banks are all equal to that of the 24 channel bank. Even though the (M, N) parameters are different for the three synthesis banks, they all must be scaled by a factor of $\frac{24}{M}$ to compensate for their lower sampling frequencies.

Therefore, to split a fullrate signal into third-bandwidth sections, the latency is $2 \cdot 84 = 168$ samples. To recombine these third-bandwidth sections back into one fullrate signal, the latency is again doubled to $D_{third} = 336$ samples.

5.2 DTWT Latency

As in the previous section, we will first compute the latency of just the DTWT, D_{WT}, without the third octave sections. In section 5.3, we will combine the two factors.

To begin with, the latency of a two channel QMF structure like that seen in figure 1 is L samples, where L is the length of each of the four filters.

For a three channel, two octave DTWT, a delay line must be inserted into after the highpass and downsampling of the highest channel in order to properly time synchronize all the subbands. The length of this delay line will be twice the latency of the structure embedded after the lowpass and downsampling stage (due to the lower sampling rate). The latency of the entire structure will be $2L + L = 3L$ samples. The factor of $2L$ is twice the length of the delay line inserted (due again, to the lower sampling rate). The extra factor of L is due to the latency of the first iteration of the QMF bank.

To summarize, delays must be inserted after every highpass filter in the DTWT structure. The inserted delay needed for a given octave k, can be recursively calculated:

$$delay(k) = 2 * delay(k - 1) + L$$

where $delay(1) = L$. The total latency for a K octave bank will be $D_{WT} = delay(K)$. In this design for a four octave DTWT, and length 64 tap filters, the latency comes to 945 samples.

5.3 System Latency

To compute the latency for the total system, D_{tot}, it can be solved recursively in the same manner as in section 5.2, except that the initial condition must also include the delay of the third octave sections. Therefore, with $delay(1) = L + D_{third}$, then $D_{tot} = 6321$ samples.

6 Memory Requirements

In this system, the amount of memory required comes from two places: the delay lines inserted into the DTWT to ensure subband time synchronization, and the FIR filters inside the third octave QMF filter banks.

For the DTWT, the memory needed for the inserted delay line lengths increases exponentionally for the number of octaves K:

$$(2^K - (K+1))[L + D_{third}]$$

which comes to 10,374 samples in this design. The number of memory required by the lowpass and highpass filters for the DTWT is $4L(K+1)$, which is 1,260 samples for this system.

For the third octave sections, memory is needed for the polyphase filtering within the QMF banks. A cosine modulated QMF analysis bank of length N prototype lowpass filter requires N elements of delay, while a synthesis bank requires twice as much memory, $2N$ samples. For K octaves, it can be shown that this memory tallies to $6 \cdot 192(K+1)$, which amounts to 6,912 samples. Thus, for all of the previous sources of memory, the entire system requires 18,546 samples of memory.

7 Comparisons to other Filter Banks

There are many other time-frequency representations used in audio data compression, using both uniform and nonuniform filter banks. In this section, we will attempt to just name a few.

7.1 Uniform Filter Banks

The most widely used audio coding algorithms today, MPEG [Brandenburg and Stoll, 1994] and AC-3 [Todd, 1994], both use variations of uniform, cosine modulated lapped transforms [Malvar, 1992]. MPEG uses a 512 point window with a 32 point transform [Rothweiler, 1983], while AC-3 uses a 512 point window with a 256 point transform [Princen, 1987]. Both systems use some form of window length adaptation to compensate for pre-echo effects during transients. These commercial filter banks were primarily chosen due to their low complexity and their low memory requirements, which are large factors when implementing these solutions in hardware. In exchange, the filter banks do not match the critical bands of hearing.

7.2 Wavelet Filter Banks

A recent study in nonuniform filter banks for audio coding was presented by [Sinha and Tewfik, 1993]. They implemented an 8 stage, 29 subbands wavelet packet representation using uniform, two channel FIR filters at each node of the tree. This structure is another approximation to the critical bands of hearing. All the filters used were maximally flat wavelet filters. Thus, their study chose to maximize regularity in exchange for poorer frequency selectivity.

More recently, perfect reconstruction filter banks using IIR filters were developed for audio coding [Creusere and Mitra, 1996]. In order to gain better frequency selectivity, shorter (fourth order) IIR filters were used inside an 32-band full tree-structured allpass filter bank. In exchange, the latency and memory requirements became much larger, while complexity remained low. To gain perfect reconstruction using IIR filters, the synthesis filters must be non-causal. To implement this, long buffers must be used to store up the subband samples in order to perform the time-reversal necessary.

7.3 Final Notes

The third octave filter bank described in this paper attempts to make the closest approximation possible to the critical bands of hearing, while letting latency and memory expenses get large. Low latency is usually only important for two way, real-time communications. For playing sound files from a server, or listening to broadcast audio, latency is not that much of an issue. Low memory is important for low cost, hardware implementations. But, if the algorithm is running in software on a general purpose computer, memory usage is not as vital.

It should also be noted that in any data compression scheme, the filter bank is only one of three major building blocks; the other two being quantization and entropy coding. The quality of an audio data compression system, as a whole, can only be measured with all three of thee blocks working together. Only the filter bank is considered within the scope of this paper.

Acknowledgments

The author would like to thank Dana Massie and the Joint E-mu/Creative Technology Center for the original research project idea, along with their support.

8 Conclusion

This paper introduces the critically sampled, third octave filter bank. This structure, which could be used as a front end for an audio data compression

system, closely models the critical bands of human hearing. Filter design methods were shown for the octave band filters and the third octave filters, as well as calculating the corresponding costs in latency, memory and complexity.

References

[Zwicker, 1990] E. Zwicker and H. Fastl. *Psychoacoustics*, Berlin: Springer-Verlag, 1990.

[Vetterli and Kovačević, 1995] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding*, Prentice Hall, 1995.

[Parks and McClellan, 1972] T. Parks and J. McClellan. *Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase*, IEEE Trans. on Circuit Theory, Vol. CT-19, No. 2, pp. 189-194, May 1972.

[Strang and Nguyen, 1996] G. Strang and T. Nguyen. *Wavelets and Filter Banks*, Wellesley-Cambridge Press, p. 172.

[Crochiere and Rabiner, 1983] R. Crochiere and L. Rabiner. *Multirate Digital Signal Processing*, Prentice Hall, 1983.

[Rioul, 1993] O. Rioul. *On the Choice of "Wavelet" Filters for Still Image Compression*, ICASSP 1993.

[Daubechies, 1988] I. Daubechies. *Orthonormal Bases of compactly supported wavelets*, Commuin. on Pure and Applied Math., 41:909-996, Nov. 1988.

[Philippe, 1995] P. Phillipe, et. al. *On the Choice of Wavelet Filters for Audio Compression*, ICASSP, 1995.

[Kudumakis, 1995] P. Kudamakis, et al. *Wavelets, Regularity, Complexity, and MPEG-Audio*, AES Convention, New York, October 1995.

[O.Rioul, 1994] O.Rioul and P. Duhamel. *A Remez Exchange Algorithm for Orthonormal Wavelets*, IEEE Trans. on Circ. and Sys. II, Vol 41, No. 8, August 1994.

[Vaidyanathan, 1993] P.P.Vaidyanathan.*Multirate Systems and Filter Banks*, Prentice Hall, 1993.

[Kovačević and Vetterli, 1993] J. Kovačević and M. Vetterli. *Perfect Reconstruction Filter Banks with Rational Sampling Factors*, IEEE Transactions on Signal Processing, Vol. 41, no. 6, June 1993.

[Nguyen, 1996] T.Q. Nguyen. `http://saigon.ece.wisc.edu/~waveweb/QMF/COSIN/NPR/N_8_64.html`

[Nguyen, 1994] T.Q. Nguyen. *Near-Perfect-Reconstruction Pseudo-QMF Banks*, IEEE Transactions on Signal Processing, Jan. 1994, pp 65-76.

[Cox, 1986] R. Cox. *The Design of Uniformly and Nonuniformly Spaced Pseudoquadrature Mirror Filters*, IEEE ASSP, Vol. 34, No. 5, October 1986.

[Princen, 1995] J. Princen. *The Design of Nonuniform Modulated Filterbanks*, IEEE Transactions on Signal Processing, Vol 43, No. 11, Nov. 1995.

[Rioul and Duhamel, 1992] O. Rioul and P. Duhamel. *Fast Algorithms for discrete and continuous wavelet transforms*, IEEE Transactions on Information Theory, Vol. 38, pp. 569-586, March 1992.

[Konstantinides, 1994] K. Konstantinides. *Fast Subband Filtering in MPEG Audio Coding*, IEEE Signal Processing Letters, Vol. 1, No. 2, February 1994.

[Malvar, 1992] H. Malvar. *Signal Processing with Lapped Transforms*, Artech House, 1992.

[Rothweiler, 1983] J. H. Rothweiler. *Polyphase Quadrature Filters - A New Subband coding Technique*, International Conference IEEE ASSP 1983, Boston, S.1280-1283.

[Brandenburg and Stoll, 1994] K.Brandenburg and G. Stoll. *ISO-MPEG-1 Audio: A Generic Standard for Coding of High-Quality Digital Audio*, Journal of the Audio Engineering Society, Vol. 42, No. 10, October 1994.

[Princen, 1987] J. Princen, A. Johnson, A. Bradley. *Subband/Transform Coding Using Filter Bank Designs Based on Time Domain Aliasing Cancellation.* ICASSP, 1987. pp. 2161-2164.

[Todd, 1994] C. Todd, G. Davidson, et. al. *AC-3: Flexible Perceptual Coding for Audio Transmission and Storage*, 96th AES Convention, 1994.

[Sinha and Tewfik, 1993] D. Sinha and A. Tewfik. *Low Bit Rate Transparent Audio Compression using Adapted Wavelets*, IEEE Trans. on Signal Processing, Vol. 41, No. 12, December 1993.

[Creusere and Mitra, 1996] C. Creusere and S. Mitra. *Efficient Audio Coding Using Perfect Reconstruction Noncausal IIR Filter Banks*, IEEE Trans. on Speech and Audio Processing, Vol 4, No. 2, March 1996.

Effects Processing on Audio Subband Data

Scott N. Levine
Stanford University
Center for Computer Research in Music and Acoustics (CCRMA)
scottl@ccrma.stanford.edu
http://www-ccrma.stanford.edu

Abstract

This paper will show that computing standard audio effects algorithms such as reverberation, echo, chorus and flange will use less memory and computations when performing the operations on the critically sampled subband data than on the fullrate time domain signal. Not only can effects in the subband domain be obtained to sound close to the effects in the time domain, but new types of effects are now possible because different effects can be placed in separate subbands. The MPEG Audio filter bank, which also splits the audio into subbands, is used in this discussion to show that standard MPEG audio decoders could easily be augmented to include effects processors as well.

1 Introduction

Since the MPEG Audio (layers I,II) standard has become a prominent data compression algorithm in the consumer multimedia market, it exists on many platforms ranging from personal computers to workstations to video games. For most of these applications, some sort of audio post-processing is desired by the end-user. A simple example could be the addition of artificial reverberation for watching a movie in a home theater.

These post-processing algorithms are widely available today, but usually require special hardware for its computation and large amounts of memory. It will be shown that these post-processing algorithms can be performed on the audio subband data itself present inside the MPEG decoding standards [Rothweiler, 1983]. By computing the effects while performing the MPEG decoding, the need for external effects processors is eliminated. Since effects are computed separately for each subband, custom tailored effects can now be placed on different regions of frequencies.

2 Standard Effects Processing Algorithms

Before delving into the benefits of computing effects in a multirate domain, some of the simplified, standard effects algorithms will be quickly explained. For a thorough explanation on these effects, see [Orfanidis, 1996].

2.1 Echo

The simplest effects example would be an echo simulation. An echo algorithm models discrete acoustic reflections from surfaces far away from the listener. If the surfaces are too close in space and the reflections are too close together in time, the ear will not recognize them as discrete echoes, but rather as a reverberation (which will be discussed later). To model an echo, all that is needed is a feedback comb filter with D uniformly spaced poles, as seen in figure 1.

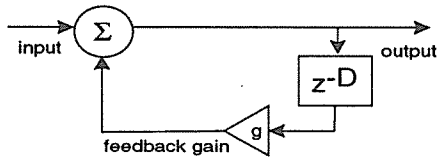


Figure 1: *Echo effect*

2.2 Flange

With digital technology, flanging translates to a delay line whose feedforward read pointer from the delay line to the output is sinusoidally modulated at around 0.2 Hz, as seen in figure 2. To make the structure general, the feedback read pointer from the delay line to the input summation is also shown as modulated. Traditional time domain flangers use no feedback modulation (i.e. a modulation of 0 Hz) due to its inherent pitch modulations. It will be shown later that this feedback modulation frequency is important when designing multiband flangers.

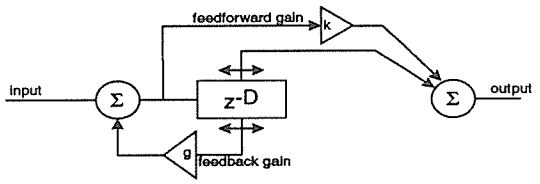


Figure 2: *Flange effect*

2.3 Chorus

The chorus effect is meant to make a single vocalist or instrumentalist sound like a whole group or section performing. When a group of people perform together, they are not all signing or playing at the exact same time, but are slightly non-synchronous. The chorus effect tries to model the time variations between players with several modulated delay lines. Its topology is much like the flanger, except there is no feedback gain, and there are three parallel modulated delay lines instead of one, as shown in figure 3. The delay lines are longer than the flanger, around 25-30 milliseconds and the modulating frequency is still under 1 Hz.

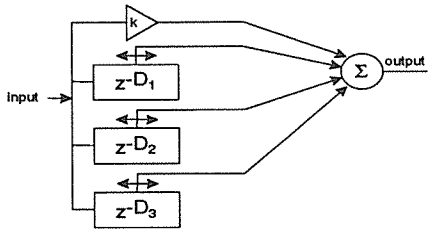


Figure 3: *Chorus effect*

2.4 Reverberation

A huge amount of papers have been published in the area of artificial reverberation, but many of them point towards [Schroeder, 1962] as the pioneer. The basic problem is how to make efficient models of diffuse acoustic reverberation without having to convolve a signal with the impulse response of the room. This response could last for seconds, and would require an FIR filter with tens of thousands of taps. The basic building blocks of these types of artificial reverberators consist of parallel banks of feedback comb filters (FBC), along with a series chain of allpass (AP) filters. The earliest such structure was shown by Schroeder, as seen in figure 4.

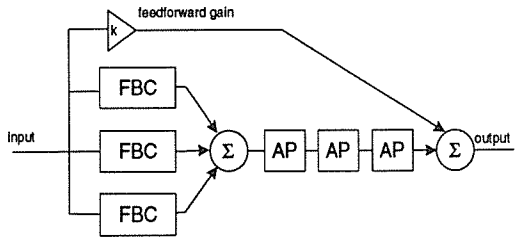


Figure 4: *Reverberation effect*

3 MPEG Filter Bank

When choosing a filter bank in which to implement frequency domain effects, the filter bank from the MPEG audio standards [Brandenburg and Stoll, 1994] proves to be a wise choice. First of all, the 32 channel, uniform cosine modulated pseudo quadrature mirror filter has high frequency selectivity. With a length of 512, the steep window response has their first sidelobes down over $100dB$. The magnitude response of this filter bank is shown in figure 5. Secondly, the computationally expensive matrix multiplication associated with the filter bank can be replaced with efficient, fast discrete cosine transform (DCT) algorithms.

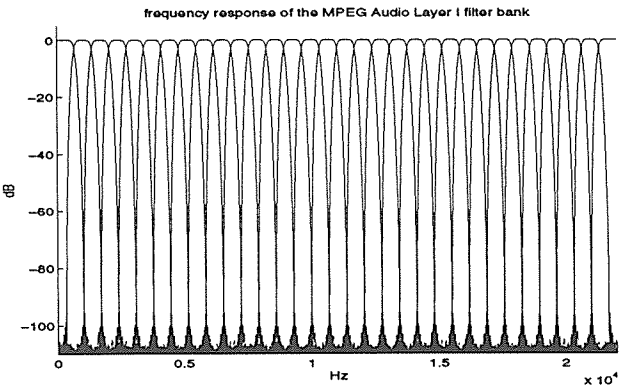


Figure 5: *Magnitude response*

Perhaps as important, the MPEG audio standard and its filter bank is already implemented on millions of platforms around the world. Therefore, the effects described in this paper could easily be inserted into the huge number of existing MPEG audio decoding systems. There would be no need to add other external hardware or software solutions to gain effects processing algorithms; the effects can be combined into the MPEG decoder itself.

4 Subband Effects Processing

This section will explain how effects are transformed into this MPEG filter bank scheme and remain sounding close to the effects computed on the time domain, fullrate signal. Then it will be shown that by altering these effects algorithms, one can gain sizable savings in computation and memory.

4.1 Simple Subband Effects

The first step to mapping an effects algorithm into the individual subbands is to reduce the length of any delay element in the original algorithm by 32, and multiply any modulation frequency by 32. These initial modifications are necessary because

the subbands are critically downsampled by a factor of 32.

As a simple example of the effects transformation, a echo effect is desired with a delay line length of 300 msec. To replicate this effect exactly in the subband domain, first design another comb filter like the one shown in figure 1, with the same feedback scalars but with a $300msec./32 = 9.4msec$ delay line length. This altered comb filter is placed on every subband, as in figure 6. Then the processed subbands are run through the same MPEG synthesis filter bank, and the output will sound just like a standard, fullrate echo effect.

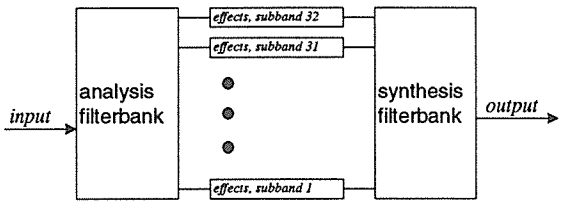


Figure 6: *Subband effects structure*

4.2 Customized Subband Effects

Because there are now 32 parallel, independent effects processors working over different bandwidths, we can now tailor the coefficients in each band separately. This gives the sound designer many more degrees of freedom than using a single fullrate effects algorithm.

4.2.1 The Multiband Flanger

With all these new degrees of freedom, we designed a frequency dependent flanger. Refer to figure 2 for the filter structure that will be placed in all subbands. First of all, we sinusoidally modulate both the feedforward and feedback pointers in all subbands. In a fullrate flanger, feedback modulation is usually avoided due to inherent pitch shifting qualities. But in this structure, any pitch shifting will be limited to a 687 Hz wide subband, so it is not as noticeable. Also, feedback modulation makes the multiband flanger sound closer to the fullrate version, for reasons we have yet to explain.

Secondly, we shorten the delay line lengths as the subband center frequency gets higher. In this system, we begin with 10 msec. delay lines at the lowest frequency subband, and linearly reduce their lengths to 3 msec at the 15th subband (at around 10 kHz). Above this frequency, the energy levels are so low, it is not worthwhile spending the memory and computation to process the signal. Thus, the subband data is passed through dry.

Also, more subtle changes can be heard if the gain magnitudes are decreased towards zero as the subbands get higher in frequency. In this way, the deepest harmonically spaced frequency notches are at lower frequencies only.

4.2.2 Multiband Reverberators

Most commercial reverberators are bandlimited to around 8kHz. This cutoff is due to the fact that the materials from most room interiors only reflect low frequencies. When looking at the magnitude responses of most rooms, the higher frequencies have quicker exponential decay rates than the lower frequencies. This translates to smaller delay line lengths for the higher frequencies. In [Schroeder, 1962], he defines the T_{60} time, which equals the time the reverberating sound level drops by 60 db. In a feedback comb filter, the delay line length D is related to the T_{60} and the feedback scalar $|g| < 1$ by the equation: $D = \frac{1}{3}T_{60} \cdot log(\frac{1}{|g|})$. For a fixed g, as the T_{60} decreases, so does D. With this subband structure in place, one can tailor these three preceding coefficients to resemble generic rooms.

For the included reverb sound example, only the first ten subbands were processed; the other 22 subbands were left dry and unprocessed. The first subband, which is lowest frequency band, had a $T_{60} = 1sec.$ Subbands 2 through 10 had linearly decreasing T_{60} times (with correspondingly shorter delay lines), and subband 10 had a T_{60} time equal to one-third of the first subband. In this algorithm, the feedback factor g stays constant for all subbands. The allpass filters retained the same coefficients for all ten subbands, which used very little memory (under 5 msec.). In order to remove any periodicities in the frequency domain, some low level noise was added to these three parameters. A short-time Fourier transform of this reverberator's impulse response is shown in figure 7.

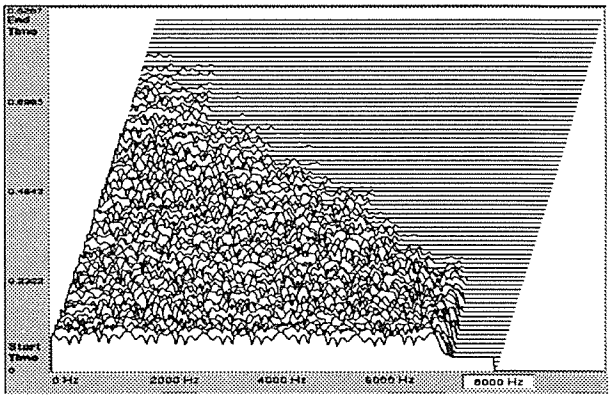


Figure 7: *Reverberator magnitude response*

4.3 Memory and Computational Savings

Assuming the cost of MPEG decoding is already provided for, the amount of calculations and words

of memory required for effects processing of 32 critically sampled subbands are equal to that of a fullrate signal. This property is due to the fact that all the effects delay line lengths have been reduced by a factor of 32. The savings in memory come when the effects algorithms are individually altered for each subband. As shown in the previous two sections, only 10-15 of the total 32 subbands were processed. Additionally, the higher frequency subband effects algorithms used up to one-third the memory of the lower frequency ones. These multiband algorithms can shrink the amount of memory necessary by up to a factor of five, while still sounding like a convincing, fullrate commercial effect.

4.4 Using the Side Information

An optimization problem present is how to get an effects algorithm to sound the best with using a limited amount of memory and computations. Looking at the side information from the psychoacoustic masking function data provided by the MPEG encoder would be a good initial starting point for deciding which subbands are more perceptually relevant. If the MPEG encoder algorithm previously decided to allocate many bits to a certain subband, it would be worthwhile to allocate more memory and computation to the that particular subband's effect algorithm. This process is very efficient since it would be utilizing all the calculations previously performed in the MPEG encoder's complex psychoacoustic model [Brandenburg and Stoll, 1994]. Similarly, the encoder's transient detection information could help these algorithms adaptively change their coefficients in new and novel ways.

4.5 Layering Subband Effects

Instead of allocating all memory and computations for one effects algorithm, one could apply several effects over different (and possibly overlapping) frequency ranges. For example, the user could allocate all available memory for one good reverberator over all audible frequencies. Or, the user could split the memory between a slightly worse reverberator from $0 - 8kHz$, plus a flanger from $2 - 12kHz$ and a chorus from $4 - 10kHz$ by placing the effects in the corresponding subbands.

4.6 Alias Cancellation

If no processing were to take place between the analysis and synthesis filter banks in figure 6, then the any aliasing noise would remain below a $100db$ noise floor. The problem arises when the critically sampled subbands are processed. In [Schoenle, Fliege, and Zölzer, 1993], they avoided critically sampled systems and chose to implement artificial reverberation using an oversampled filter bank. By carrying around twice the amount of subband data, they avoid the problems of aliasing. To stay within the MPEG standards and reduce the memory requirements, their method is avoided.

In the experimental work with this MPEG multirate effects framework, no audible aliasing errors have been noticed. Even if there are aliasing problems, it could easily get perceptually buried under the alterations that the effect itself is creating. In addition, due to the structure of the MPEG filter bank, any aliasing by an effects algorithm in one band will be bandlimited by half a subband (343 Hz) on each side. Therefore, since the aliased regions are highly attenuated, it is most likely that their energy will be masked.

5 Conclusion

This paper has shown that memory and computation can be saved by moving post-processing audio effects such as reverb, chorus, flange and echo from the time domain to the subband domain. These subband domain audio signals are present in the current MPEG Audio compression standards. It was shown that effects processing can be calculated on the subbands while the audio is being MPEG decompressed. This combination of computations eliminates the need for specialized external effects processing hardware. Since the effects are calculated on the audio subbands separately, new and different effects can be placed on different regions of frequencies.

References

[Rothweiler, 1983] J. H. Rothweiler. *Polyphase Quadrature Filters - A New Subband coding Technique*, International Conference IEEE ASSP 1983, Boston, S.1280-1283.

[Orfanidis, 1996] S.J.Orfanidis. *Introduction to Signal Processing.* Prentice-Hall, 1996. pp. 355-383.

[Schroeder, 1962] M.R.Schroeder. *Natural Sounding Artificial Reverberation*, Journal of the AES, Vol. 10, No. 3, July 1962.

[Schoenle, Fliege, and Zölzer, 1993] M. Schoenle, N. Fliege, and U. Zölzer, 1993. *Parametric Approximation of Room Impulse Responses by Multirate Systems*, ICASSP 1993.

[Brandenburg and Stoll, 1994] K.Brandenburg and G. Stoll. *ISO-MPEG-1 Audio: A Generic Standard for Coding of High-Quality Digital Audio*, Journal of the Audio Engineering Society, Vol. 42, No. 10, October 1994.

PadMaster: banging on algorithms with alternative controllers

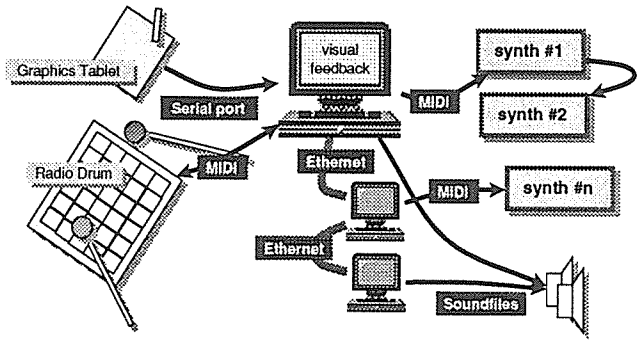
Fernando Lopez-Lezcano (*nando@ccrma.stanford.edu*)
CCRMA (Center for Computer Research in Music and Acoustics), Stanford University

ABSTRACT

This paper will describe the current implementation of PadMaster, a real-time improvisation environment running under the NextStep operating system on both NeXT hardware and Intel PCs. The system was designed with the Mathews/Boie Radio Drum in mind, but can now use alternative controllers, including widely available graphics tablets. The current version adds soundfile playback and algorithms to the preexisting palette of performance options.

1.0 The PadMaster program

PadMaster is a real-time improvisation environment written in Objective C that currently runs on NeXT workstations or Intel PCs running the NEXTSTEP operating system. PadMaster uses the MusicKit [4, 5, 6] as the basic foundation for controlling and performing musical events and the NeXT Soundkit as the basic resource for soundfile playback (an extension is in the works which will allow PadMaster to control other networked workstations for soundfile and or MIDI information playback).



2.0 Basic concepts: a virtual surface, controllers, pads and scenes

The performer interacts through the selected controller with a virtual surface that is modelled on the screen of the computer. This virtual surface contains one of more "Pads" which are the basic performance units of PadMaster. Pads are non-overlapping rectangular sections of the surface (of arbitrary size) and can be triggered through actions of the performer. Pads can control the playback of MIDI information or soundfiles either through sequences or algorithms. A Pad can react in different ways to a trigger event and can also link some of its control parameters to continuous position information provided by the selected controller. A set of Pads that are defined within the same surface are called collectively a "Scene". PadMaster allows an indefinite number of Scenes to be defined. The performer interacts with the current scene and can switch between them during the performance, in effect redefining the behavior of the virtual surface. A Pad can be performing even though its Scene is not the current one, that is, the performer can trigger Pads in one of the Scenes and then move on to another one while the Pads keep performing in the background (they are not visible on the computer screen which always shows the state of the current Scene).

2.1 Controllers

PadMaster was originally designed to be controlled by the Mathews/Boie Radio Drum, which provides for six independent axes of control. The current software architecture has been opened to allow the use of alternative controllers.

2.1.1 The Radio Drum

Triggering of a Pad is effected by striking the surface of the drum with one of the batons and is further modified by the state of the two foot switches. When using the Radio Drum as a controller the number of available pads is constrained by its resolution, as the hit and position MIDI messages create a basic non-linear grid of 128x128 points. For the purpose of calibration and hit detection the surface is split into a matrix of 10x12 tiles, which are the building blocks of Pads. All pads are made up of tiles and can potentially be as small as one tile or as big as the whole drum surface. A frequently used configuration splits the surface into 30 pads arranged in a 5x6 matrix, each pad being composed of four tiles. Although smaller pads can be useful at times, they are not recommended because it can be difficult to get reliable hits (it is possible to inadvertently hit the contiguous pad).

2.1.2 Other MIDI controllers

Software is currently being written to allow the use of alternative MIDI controllers. As an example, a normal MIDI keyboard can be used, where keys are mapped to Pads and pitch bend, modulation wheels or pedals are mapped to axes of position control. Percussion controllers and the Lightning are also being considered as options.

2.1.3 Graphics tablets as performance controllers

As in the previous case the software is not finished at the time of this writing. Tablets provide a performance environment that is close to that of the Radio Baton. The three

dimensional control is missing and with it goes a very important part of the gestural element that makes the Radio Drum so appealing as a controller. Tablets offer two alternative dimensions of control in addition to x-y position information. The first is pressure, which provides (sort of) a third dimension. The second is sensitivity to tilt of the pen, both in the x and y axes. Additional advantages of the tablet include the fact that it does not use MIDI bandwidth (in the case of the PC only, it uses one of the serial ports in NeXT hardware) and is an easily obtainable controller.

In addition to these options the mouse and keyboard can also be used to control most of the performance functions.

2.1.4 Mixing and matching controllers

The fact that now several different controllers might be available, even at the same time, raises the issue on how to map things so that a piece can be written in a "controller independent" way. The current approach being worked on depends on establishing a mappings of axes of control to "virtual axes" for each controller, and then using those when creating the piece. Several maps would enable the performer to play the same piece with different controllers by activating the proper map depending on the detected controller configuration. Controller maps are user configurable.

3.0 The browser

PadMaster is a document oriented program. All the programming that the composer does to define the behavior of the virtual surface can be stored to or loaded from binary documents. Documents are edited through a browser where the composer or performer can selected the scene and pad to be edited. Once the selection is done one of the scrollable panes of the browser shows all editing parameters for the selected scene or pad. The program also offers the possibility of saving or loading the document to text files.

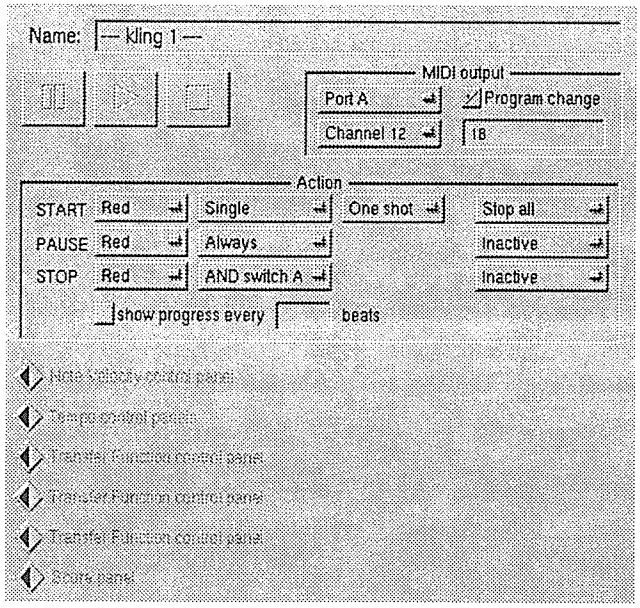
4.0 Performance Pads

A Performance Pad is the basic performance element of the program and can be programmed to control the playback of MIDI information or soundfiles. The graphical representation of the Pads on the screen gives visual feedback to the performer on their performing state.

Each type of Performance Pad has a number of fixed control parameters and a number of optional "Elements" that can be added or removed at will.

4.1 Sequence Pads

Sequence Pads control the performance of MIDI information. The following figure shows the editing pane of a Sequence Pad with all elements closed. Elements and subelements are shown as lines in an outline and can be opened

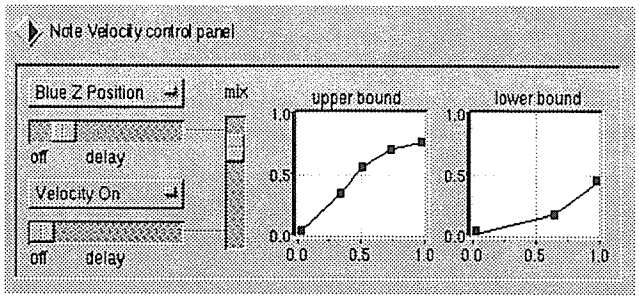


for editing or closed so they don't clutter the editing pane of the browser.

The composer can select the MIDI port, channel and an optional program change number. If the program change number is enabled, PadMaster keeps track of which pads are using a particular combination of port, channel and program change number so that Pads across the document can share the same channel but use it with different patches. Any active Pad using a particular combination disables during its performance any other Pad that uses the same port and channel but a different program change number.

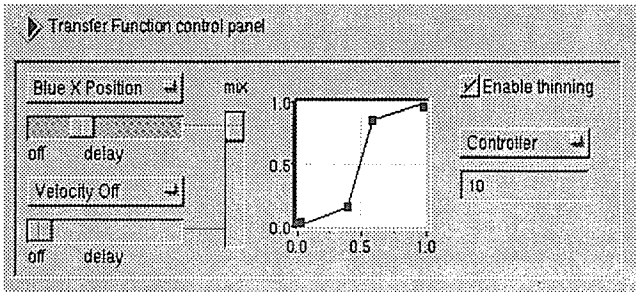
Three types of trigger action can be defined in the current version of the program: start, pause/resume and stop the performance of a Pad. Actions can also have global side effects, for example stopping all performing Pads in the current Scene when the Pad starts performing. The "start" action can be further specified as starting only one sequence, several overlapping sequences with multiple triggers or a note by note performance of the score or algorithm (any of them can perform in single shot or looped modes).

Let's open one of the elements of this particular Pad (in this case the Note Velocity control pane):



The two transfer functions represent the upper and lower limits over which the existing velocity of notes will be

mapped depending on the input parameter. The input itself is a mix of the delayed or smoothed versions of the last trigger velocity and one axis of continuous position information. The position information can come from several sources, depending on the selected controller. In the case of the Radio Drum there are six axes of control plus two additional axes that are computed at trigger time (relative x and y coordinates inside the triggered Pad). Here's another element, this time an open Transfer Function control pane:



As before, a mix of velocity and position information is fed to the transfer function which is used to generate a continuous MIDI information stream (pitch bend, pressure or continuous controllers).

Score elements contain a MusicKit MIDI score that stores the sequence of notes to be played. Several keywords have been added to the scorefile syntax so that performance of the score can be controlled from within the score itself. For example, a note containing the "*pause: 0*" keyword will immediately pause the performance of the Pad.

Algorithm elements contain a small Objective-C program that generates note objects during the performance. PadMaster supplies a well defined API (Application Programmer Interface) to the composer that provides most of the functionality that is necessary to write small algorithms. An algorithm can have some of its parameters controlled by the virtual axes of control.

These are just some examples of the elements that are available. The introduction in the current software release of elements that are in themselves separate objects opens the door to sharing elements between Pads, something very important in simplifying the task of programming the environment (the composer could, for example, program several interesting transfer functions, name them and then use them to create similar behavior in different Pads).

4.2 Soundfile Playback Pads

This type of Pad controls the performance of soundfiles. The composer can specify the file to be played and the action to be performed by the trigger (start, pause/resume and stop as before). An internal manager keeps track of the number of currently performing sounds and enables or disables Pads from being triggered to keep playback reliable.

5.0 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 virtual surface of the drum. Control Pads can also be used to pause, resume or stop all playing Pads in the currently Scene.

6.0 PadMaster in performance

Although in many ways the current version of PadMaster has the same look as the previous, the program has been completely rewritten, almost from scratch in many cases. The result is a much more efficient way of doing things internally. A couple of significant examples are: the delay from triggering to activation of a performer has been reduced; switching between scenes no longer causes an audible pause in the performance and so on...

PadMaster has been used to compose and perform two pieces so far: "Espresso Machine", for PadMaster and Radio Drum, two TG77's and processed electronic cello (Chris Chafe, playing his celletto) and "With Room To Grow" for solo performer using the Radio Drum as controller.

7.0 Future developments

Most of the architectural changes in the current version have opened doors to new functionality, in particular the creation of several controller drivers. One of the most promising future developments is the use of remote workstations through network protocols to enhance the number of controllable devices.

References:

[1] Max Mathews, *The Stanford Radio Drum, 1990*

[2] Carlos Cerana (composer) / Adrian Rodriguez (programmer), *MiniMax, a piece for Radio Drum*

[3] Fernando Lopez-Lezcano, *PadMaster: an improvisation environment for real-time performance*. Proceedings of the 1995 International Computer Music Conf., Banff, Computer Music Association.

[4] J. Smith, D. Jaffe and L. Boynton. *Music System Architecture on the NeXT Computer.* Proceedings of the 1989 Audio Engineering Society Conference, Los Angeles, CA.

[5] D. Jaffe. *Musical and Extra-Musical Applications of the NeXT Music Kit.* Proceedings of the 1991 International Computer Music Conf., Montreal, Computer Music Association, pgs. 521-524.

[6] D. Jaffe, J. O. Smith, N. Porcaro. *The Music Kit on a PC.* Proceedings of the First Brazillian Symposium of Computation and Music, XIV Congress of the Brazillian Society of Computation, Caxambu, MG, 1994. pgs. 63-69.

[7] D. Jaffe and L. Boynton. 1989. *An Overview of the NeXT Music and Sound Kits.* Computer Music Journal, MIT Press, 14(2):48-55.

Demonstration: Using SynthBuilder for the Creation of Physical Models

Nick Porcaro (nick@ccrrma.stanford.edu),
Pat Scandalis (gps@ccrma.stanford.edu),
David Jaffe (daj@ccrma.stanford.edu),
Julius Smith (jos@ccrma.stanford.edu)
CCRMA (http://www-ccrma.stanford.edu)

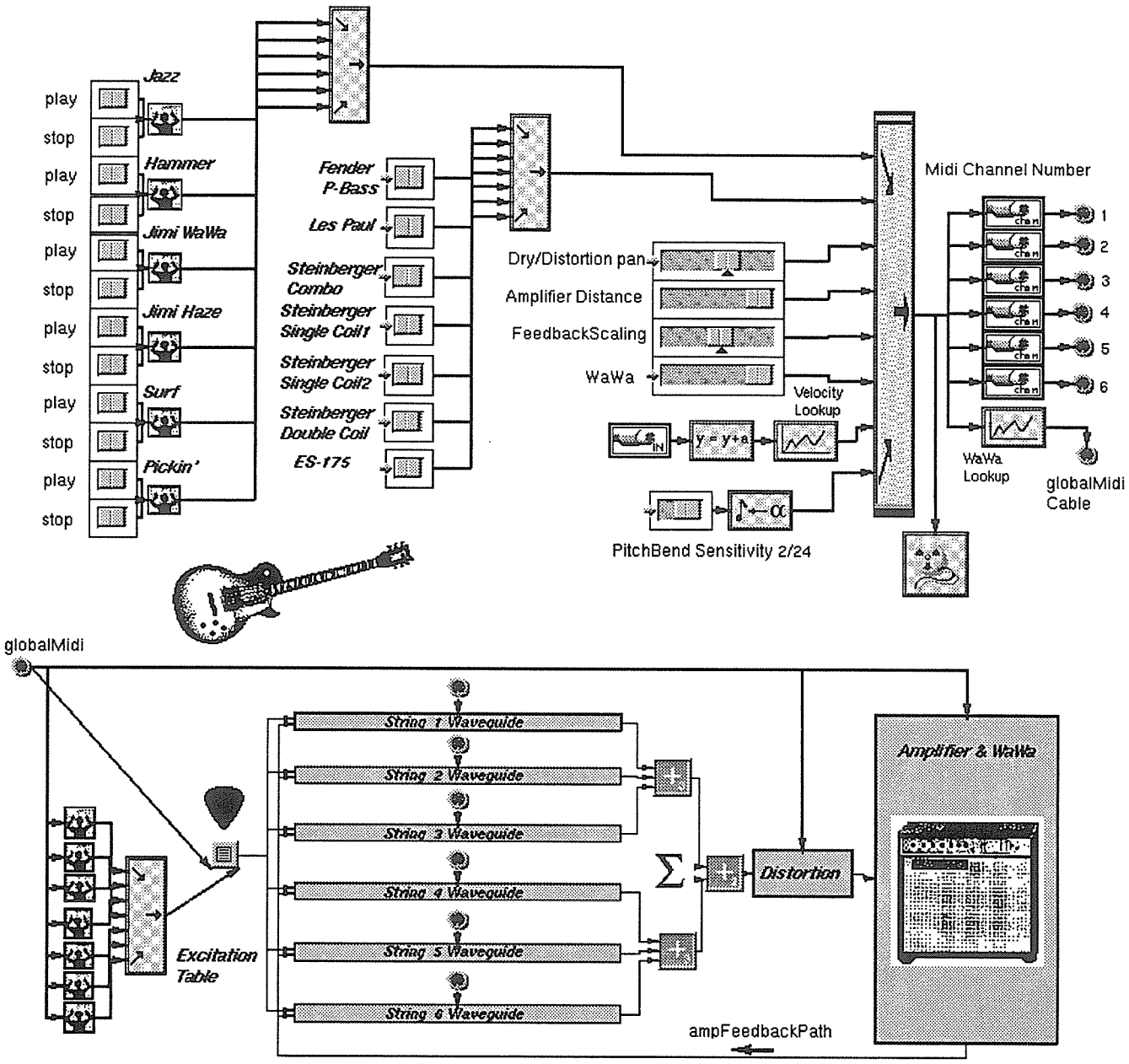
SynthBuilder is a user-extensible, object-oriented, Nextstep Music Kit application for interactive real-time design and performance of synthesizer patches, especially physical models. Patches are represented by networks consisting of digital signal processing elements called unit generators and MIDI event elements called note filters and note generators.

In this demonstration we will present an overview of SynthBuilder, and an example of how to develop an electric guitar physical model. We will also show recently-developed SynthBuilder patches, including bowed strings, piano, guitars, harpsichords, and others.

We are porting our patches to other platforms. Toward this goal, we have developed an interchange format called SynthScript, and defined a portable server, SynthServer, which will enable execution of SynthBuilder patches on other computers. Part of the demonstration will illustrate the importing and exporting of a SynthScript patch.

Recently, SynthBuilder has been significantly optimized and extended. DSP allocation speed, memory usage and drawing performance have been greatly improved. Many new features have been added, including: sound file writing support for various Intel-based DSP cards, an improved driver for the Frankenstein box, new unit generators and note filters, mouse tracking, drop-in subpatches, subpatch variations, inspector improvements, a new trace window, and more robust help/tutorial. These features along with numerous bug fixes and paradigm refinements have enabled rapid development of complex patches.

In the near future, we plan to continue to build more patches and improve SynthBuilder. Among these improvements: tighter SynthScript integration and a mechanism for General MIDI program change.



A six string electric guitar model with distortion, feedback and wawa, implemented in SynthBuilder. This model runs on on a single 72Mhz Motorola 56002 DSP.

Frankenstein: A Low Cost Multi-DSP Compute Engine for Music Kit

William Putnam (putnam@ccrma.stanford.edu)
Tim Stilson (stilti@ccrma.stanford.edu)
CCRMA (http://www-ccrma.stanford.edu/)
Electrical Engineering and Music Departments, Stanford University

Abstract

In this paper, we discuss the design of a multiple DSP system intended for musical synthesis. The current system was designed to be low cost, while still providing enough processing power for real-time complex synthesis voices and extended polyphony. A design based on readily available, low cost DSP processors, along with specific examples of real-time synthesis algorithms made possible by this hardware will be presented.

1 Introduction

This paper discusses the architecture and use of a multiple digital signal processor (DSP) system intended primarily for musical synthesis, but appropriate for other tasks as well. Primary among the design goals, were cost as well as sufficient processing power for polyphonic physical modeling synthesis applications. Furthermore, it was necessary to design a system which would be compatible with the NeXT based Music Kit [Jaffe 1989], as well as our Synth Builder [Porcaro 1995] software without major additional effort. Within these constraints and goals, the decision was made to design a system based on the readily available, low cost Motorola DSP56002 Evaluation Module [Moto 1993]. Our current system incorporates eight of these modules, allowing for several possible fixed processing topologies.

We will first discuss the design and overall philosphy of the system in general terms, and then the architecture in more detail. Folowing this, we will briefly describe available software, and the availability of information for persons wishing to build similar systems. Finally, future work and open issues will be addressed.

2 Motivation and Design Philosophy

At CCRMA, SynthBuilder is one of our primary tools for the development and prototyping of musical synthesis algorithms. Built upon the NeXT's Music Kit, Synth Builder is a graphical environment for building DSP algorithms in a rapid manner. Although efficient in its use of DSP resources, it is very easy to quickly use all available processing power when the designer is allowed to copy and paste high level signal processing blocks. The development of this tool pointed out some important issues concerning the necessary scalability of DSP systems, specifically in the context of musical synthesis.

One of the most critical issues in multiprocessor systems concerns the level of connectivity and communication necessary between processors. For most of our work in the area of musical synthesis, we found that individual algorithms were typically small enough to fit on a single DSP chip. For a large number of applications, it is only when extended polyphony and effects processing are desired that one needs to add multiple processors. Under this philosophy, one only needs to allow for processors to communicate at the audio signal level. The recognition of these factors served as the underlying design philosphy behind the Frankenstein system.

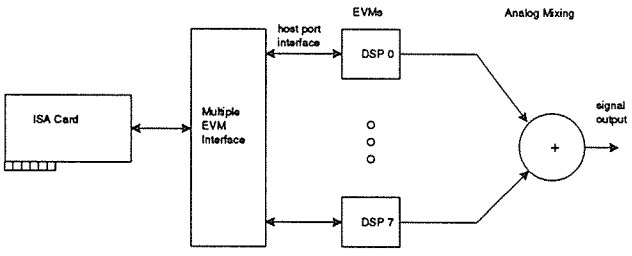


Figure 1: *Frankenstein Architecture.*

3 Architecture

The Frankenstein system was designed to be used with Intel based computers. For the sake of simplicity and cost, the standard ISA interface was

the clear choice. As mentioned previously, primary among the design issues was compatability with existing software. This necessitated a design based on interfacing to the Motorola processor through its 8-bit host port. As shown in Fig. 1, there are several parts to the design. The first is the ISA card which resides inside the computer, and performs the basic interfacing tasks. The second is an external card which handles communication with the eight DSP cards. The final block is the signal routing and mixing which is all done in the analog domain.

Polyphony dictates a parallel connection of DSPs, whereas post processing of a signal requires a cascade connection of DSPs. Ultimately, it would be nice to allow for a completely general interconnection of DSPs, with signal routing and mixing performed digitally. Although such a flexible topology would be extremely useful, and perhaps necessary at some point, it was decided that in the context of our near term needs, a couple of key fixed topolgies would be sufficient. Two of these are depicted in Fig. 2. The first is a topology in which 6 DSPs are connected in parallel, their individual outputs are summed, and fed into a cascade connection of 2 DSP cards. In this topology, the first 6 DSPs are typically used for synthesis of musical instruments, while the last 2 are used for overall effects processing such as reverberation or chorus.

All information concerning this design is available at http://www-ccrma/ putnam/frankenstein/frank.html. The Music Kit along with the appropriate software drivers is available by anonymous ftp from ftp.ccrma.stanford.edu.

Information on the design and construction of a single DSP version of the above system is also avaiable on the previously mentioned web site. This system is low cost, while sufficiently powerful to handle extensive synthesis algorithms.

An MS-Windows based software library is also available which allows for loading of DSP programs as well as subsequent communication between the host and DSP processor. A Matlab toolbox which uses this software has been written and also is available. This toolbox allows one to load and control DSP progams from within the Matlab environment. Available existing examples include real time filtering, a pseudo real-time spectrum analyzer / oscilloscope, and the measurement of room impulse responses.

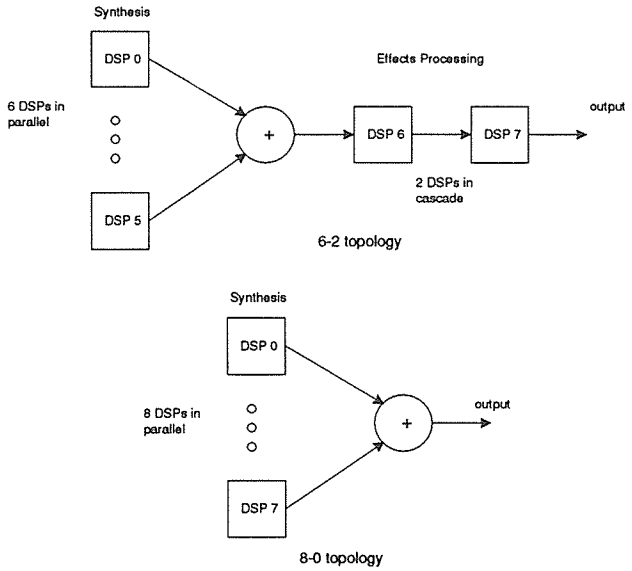


Figure 2: *System Topologies.*

4 Examples

Piano

One of the prime motivations for building this system was the piano model developed by Scott Van Duyne [VanDuyne 1995] [Smith 1995]. Using the Frankenstein hardware it was possible to implement 36 note polyphony at a sampling rate of 44KHz. Each note of the piano includeda bimodal string as well as commuted soundboard model.

Guitar

A guitar model implemented at CCRMA serves as a good example of the amount of processing possible with a single Motorola DSP56002 processor. Running at a processor speed of approximately 70MHz, it has been possible to implement 6 string guitar with wah-wah, distortion, and an amplifier / feedback model. We have also found it possible to run up to 16 simple plucked string models per DSP.

5 Future Work

As implied throughout this paper, this was clearly an exercise in compromise for the sake of design simplicity and cost. Despite this we feel that it has been extremely succesful, and has become an integral part of of our research efforts. One of the major areas that we would like to address is that of signal routing and mixing. It would be nice to do this digitally, both from a noise standpoint, as well as the ability to control processing topologies from software. Other potential improvements include

additional memory, digital audio I/O, and alternative host interfaces sych as PCMCIA or parallel port making it possible to use the systems with laptop computers.

References

[Jaffe 1989] Jaffe, David A., Lee Boyton, "An Overview of the Sound and Music Kits for the NeXT Computer." Computer Music Journal 14(2):48-55,1989.

[Porcaro 1995] Porcaro, Nick, Pat Scandalis, Julius Smith, David Jaffe, Tim Stilson 1995. "SynthBuilder Demonstration – A Graphical Real-Time Synthesis, Processing and Performance System." ICMC Proceedings 1995.

[Moto 1993] *DSP 560002 Digital Signal Processor Users Manual* Motorola.

[VanDuyne 1995] VanDuyne, Scott A., Julius O. Smith 1995. "Developments for the Commuted Piano." ICMC Proceedings 1995.

[Smith 1995] Smith, Julius O., Scott VanDuyne, 1995 "Commuted Piano Synthesis." ICMC Proceedings 1995.

Modeling and Control of Performance Expression in Digital Waveguide Models of Woodwind Instruments

Gary P. Scavone
gary@ccrma.stanford.edu

Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University
Stanford, California 94305 USA

Abstract

Most digital waveguide models of woodwind instruments to date have provided only basic control parameters such as frequency, breath pressure, and vibrato. It is clear that if these models are to gain popularity as real-time performance synthesizers or as composition tools, more flexibility is necessary. This paper discusses the implementation of expressive controls for flutter tonguing, growling, tone-bending, multiphonics, and variation of attack style. These effects are implemented on existing clarinet and flute waveguide instruments using the application SynthBuilder on a NeXTStep computer platform. Finally, control of these expressive effects using MIDI controllers is discussed.

1 Introduction

Research in physical modeling of woodwind instruments has largely focused on methods to accurately represent the instrument bore (Välimäki and Karjalainen, 1994), toneholes (Välimäki *et al.*, 1993), and nonlinear excitation mechanism (Scavone, 1995). The understanding of these issues is far from complete and work should continue to improve the existing models. However, there is a growing demand by composers and musicians to use physical models in new compositions and performance settings. In these cases, the models need more flexibility so as to produce a wide variety of sounds, both similar to real instruments and sounds which would be physically impossible in the real world. This paper first presents methods for achieving such flexibility within the context of digital waveguide modeling. The control of these extensions using MIDI controllers is discussed in the second part of this paper. The implementation of performance expression within the context of stringed instruments was previously discussed by Jaffe and Smith (1995), but issues of real-time MIDI control were not considered.

2 Modeling Performance Expression

The expressive controls discussed here fall into three principal categories – attack variation, breath pressure modification, and bore manipulation. The attack or onset of sound generation in musical instruments is a particularly important aspect of instrument performance and offers enormous expressive flexibility to both the composer and performer. Further, this attack information is a critical element in distinguishing different instruments from one another. Breath pressure modifications, such as flutter tonguing, growling, and singing into the instrument, are possible in all wind instruments, though they are a more common element of woodwind instrument performance. The production of multiphonics, achieved by non-traditional fingerings, is particular to woodwind bores with tonehole lattices.

2.1 Attack Variation

A variety of attack styles are possible in woodwind instruments, ranging from breath attacks to extremely percussive, "slap tongue" effects. Most models typically implement only breath-like styles of attack. Hard tonguing effects are achieved by using the tongue to briefly push the reed against the mouthpiece facing, stopping the reed vibrations and air flow into the mouthpiece. The rapid increase in pressure and air flow into the mouthpiece upon removal of the tongue from the reed, together with noise produced by this highly turbulent initial air flow, produces the resulting attack sound. Lighter tonguing effects are created by briefly interrupting the reed vibrations with the tongue and lesser degrees of flow interruption. The upper half of Figure 1 represents a common method for implementing a breath attack. The breath noise

scaler controls the level of noise present in the steady-state sound. A tongued attack is implemented with an additional burst of DC pressure and noise, as shown in the lower half of Figure 1. The tonguing envelope controls the magnitude and duration of the attack and should have a shape of the form $x\,e^{-x}$. Scaling of the tonguing envelope corresponds to "hardness" of attack and provides an important performance expression control parameter. The relative degree of air flow stoppage is controlled with the tonguing noise scaler.

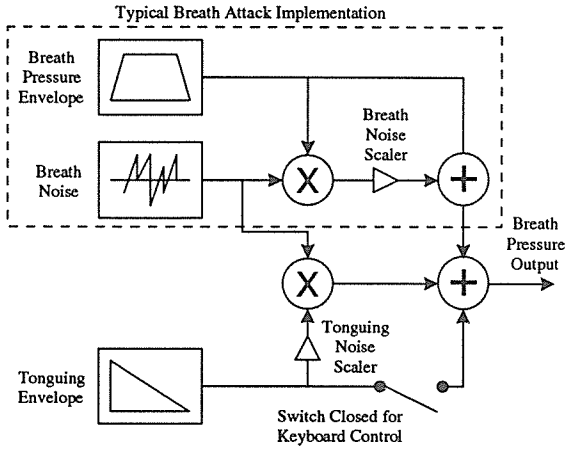


Figure 1: Tonguing System

Slap tonguing is an effect whereby the reed is pulled away from the mouthpiece lay using a suction force between the reed and tongue. When the elastic restoring force of the reed becomes greater than the suction force between the tongue and reed, the reed separates from the tongue and "slaps" against the mouthpiece lay. Varying amounts of breath pressure are then added to produce a range of effects from dry to fully sounding. Implementation of this effect can be accomplished in several ways. One method involves the recording of a dry slap with the mouthpiece removed from the instrument. This signal is then added to the normal breath pressure signal and input to the instrument's nonlinear excitation. This effect can also be achieved by approximating the slap with a predetermined filter impulse response, which is added to the breath pressure signal.

2.2 Breath Pressure Modulation

Several extended performance techniques involve the superposition of higher frequency components with the DC breath pressure applied to the instrument. This type of modification is referred to here as modulation, though not in the strict sense of amplitude or frequency modulation. Flutter tonguing, accomplished using either the tongue or ventricular folds (false vocal folds), simply amounts to the addition of a 15 – 30 Hz signal to the breath pressure. Growling and singing are audio rate modulations of breath pressure. One interesting non-physical extension possible in the digital domain is modulation with speech signals, particularly fricative sounds.

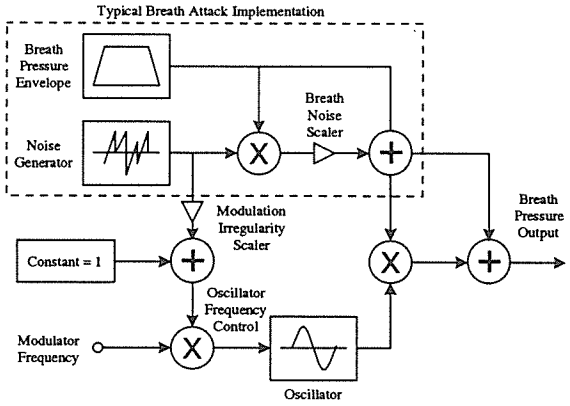


Figure 2: Breath Pressure Modulation System

The implementation of these effects can be achieved using the system depicted in Figure 2. A sinusoidal signal of some desired modulation frequency is added to the original breath pressure and the modulation frequency is randomly varied around its mean to attain a more realistic modulation signal. Modulation of breath pressure with speech can be achieved using recorded signals, though memory considerations in real-time DSP implementations often make this prohibitively expensive. A more desirable implementation provides real-time digital input via a microphone, which can be scaled and added directly to the breath pressure signal. As discussed later in conjunction with wind controllers, a breath pressure sensor sampled in the range of 2 kHz would be ideal for breath pressure control and eliminate the need for most of the system in Figure 2.

2.3 Multiphonics and Pitch Bending

Multiphonics are a common contemporary performance technique produced on musical instruments which have a tonehole lattice. Acoustically speaking, non-traditional fingerings produce air column resonances which are not harmonically related, but which are strong enough to entrain simultaneous inharmonic reed oscillations. The resulting tone is heard as comprised of two or more synchronous and distinct pitches, or as a tone with a rough and beating quality.

The most accurate method of modeling this phenomenon is to implement a full series of toneholes which exactly reconstruct the real instrument behavior. The present understanding of tonehole behavior and interaction does not allow realization of this goal and designs using current tonehole models would prove extremely difficult to properly tune. Further, the complexity of such a model would

make real-time performance difficult to achieve in most situations. An efficient, though non-physical, technique for generating multiphonics is to add more bores to the model, each of different length. In terms of digital waveguide modeling, this corresponds to the addition of more delay lines and a summing operation for feedback to the excitation mechanism, as shown in Figure 3. Each delay line represents a particular resonance and set of overtones, while the input scalers roughly control their relative strengths.

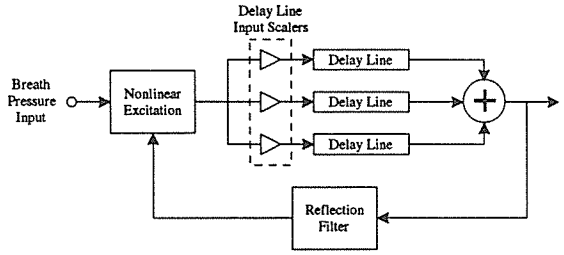


Figure 3: Multiphonic Generation System

Another performance technique is pitch bending. On single reed instruments, equilibrium reed position can be manipulated by the performer's lower jaw, allowing the sounding pitch to be lowered by as much as a quartertone. Some increase in pitch is possible by tightening the embouchure but this effect is much less significant. Oral cavity manipulations allow a further lowering of pitch, the magnitude of which varies over the range of the instrument and can be greater than a fifth. This effect is most easily implemented in a digital waveguide context using a smooth delay line interpolation method. Linear and lagrangian interpolation techniques produce no transients due to filter coefficient changes, but care must be taken to avoid signal discontinuities when changing delay lengths. For the small incremental delay length changes necessary for pitch bend, these techniques generally work well without producing audible discontinuities. Allpass interpolation can prove troublesome because of the transients associated with coefficient modifications in a recursive filter structure. Two methods exist for minimizing these transients in waveguide models (Välimäki *et al.*, 1995; Van Duyne *et al.*, 1996).

3 Controlling Performance Expression

The enormous flexibility originally proclaimed on behalf of physical modeling is finally coming to fruition and in many ways, we are unprepared to control it. Implementation of the expressive techniques discussed above is straight forward and intuitive within the physical modeling context. Finding ways to control these behaviors in real time is far from simple given current MIDI standards and controller technology. It is obvious that MIDI was not designed to handle extended techniques. Is it possible to adequately control these parameters without inventing a new protocol? The clear choice for controlling woodwind synthesis models is a MIDI wind controller. However, the few wind controllers commercially available offer only basic features and prove inadequate for the control of most of the extended techniques discussed in this paper. The MIDI keyboard is also far from ideal in this context, but it must be supported for historical and pragmatic reasons. The limitations of the keyboard are sometimes circumvented by providing wind-like controllers, such as the breath pressure controller supplied with Yahaha's VL1 synthesizer. In order to accommodate the advantages and disadvantages of these two controller types, different control schemes are necessary for each.

3.1 Wind Controller Issues

Flexible attack control requires two degrees of freedom and can be achieved using both breath pressure and velocity MIDI messages. The breath pressure messages control the breath pressure envelope while velocity controls the tonguing noise level, or tonguing noise scaler in Figure 1. In this way, a wide range of combinations of breath attack and tonguing level are possible. The Yamaha WX series of wind controllers generate both breath pressure and velocity MIDI messages via the pressure sensor in its mouthpiece. It is unclear how the velocity messages are determined in the WX controllers, though playing experiments show precise control of this parameter to be difficult. Of all the wind controllers commercially available at present, those of Yamaha are the only ones which generate breath dependent velocity messages. Physically relevant breath velocity messages can be obtained, however, by differentiating the breath pressure signal, so that velocity control using controllers without MIDI velocity output can still be possible if implemented onboard the synthesizer or computer.

The performance techniques based on modulation of the breath pressure signal present significant challenges in developing a realistic means of control. The most physically accurate solution would incorporate a breath pressure sensor that is sensitive enough to detect the modulations in the performer's breath input. Audio rate modulations, however, would require pressure sensor sampling rates on the order of 2 kHz. Under current MIDI standards, message rates can theoretically run as high as 1.5 kHz using running status and 2-byte messages, but such a strategy would be inefficient and hinder control of other aspects of the model. A more ideal solution would be to output breath

sensor readings on a separate data line for real-time digital breath pressure input to the instrument model. Under the limitations of current wind controller technology, one possible scheme for the control of flutter tonguing and growling provides the performer with a foot switch mechanism that allows control of the modulation rate.

Multiphonics present an even greater control problem when using a wind controller. Potentially, non-traditional fingerings could be detected and output with special MIDI parameter values. The Synthophone wind controller provides this flexibility, allowing non-standard fingerings to be programmed with particular parameter values. However, the Yamaha WX and Akai EWI wind controllers output a standard MIDI key number for all fingerings without allowing the key combinations to be reprogrammed. This limitation might potentially be circumvented by using a particular MIDI program change message to control a "multiphonic" mode of operation, but any control scheme developed under this scenario would only function as a poor substitute to the desired behavior. Clearly, the programmable environment offered by the Synthophone should serve as a model for future wind controller development.

3.2 Keyboard Controller Issues

The mapping of MIDI keyboard control mechanisms to wind instrument expressive parameters is less obvious than when using a wind controller. However, the keyboard provides more flexibility than current wind controllers when used in conjunction with a breath pressure sensor. This is largely due to the fact that only one hand is needed to play the keys of the keyboard, leaving the other hand free to modify additional parameters. Without a breath pressure sensor, attack control via the keyboard is completely dependent on key velocity messages, resulting in significant loss of flexibility. In this context, low velocity values might be made to correspond to soft breath attacks and high velocity values to loud, hard tongued attacks. Lost in this scheme would be such effects as strong breath and lightly tongued attacks. The addition of a breath controller gives the keyboard musician much of the same attack flexibility enjoyed by users of wind controllers equipped with breath sensitive velocity detection.

As previously mentioned, natural control of breath pressure modulation requires a high breath pressure sampling rate. Until new controller technologies make this possible, the system of Figure 2 can be implemented and the various modulation parameters can be assigned to such keyboard controllers as modulation wheels or foot pedals. In this instance, the keyboard's wide array of controllers give it more flexibility than the wind controller.

Control of the multiphonic implementation of Figure 3 using a keyboard can be achieved by depressing multiple keys at the same time and assigning the various delay line lengths by the corresponding key numbers.

4 Conclusions

Physical models of woodwind instruments provide flexible control over a wide range of performance expression techniques. The implementation of these effects is reasonably straight forward because of the one-to-one correspondence between the model elements and physical elements. Unfortunately, control of these techniques is less straight forward, even when using a MIDI wind controller. With current technology, schemes can be developed which allow control of performance expression using both wind controllers and keyboards, though such control is not always intuitive or natural. The flexibility of physical modeling should result in the future development of new controller technologies that make such control more natural.

References

ICMC (1995). *Proc. 1995 Int. Computer Music Conf.*, Banff, Canada. Computer Music Association.

Jaffe, D. A. and Smith, J. O. (1995). Performance expression in commuted waveguide synthesis of bowed strings. In ICMC (1995), pp. 343–346.

Scavone, G. P. (1995). Digital waveguide modeling of the non-linear excitation of single-reed woodwind instruments. In ICMC (1995), pp. 521–524.

Välimäki, V. and Karjalainen, M. (1994). Digital waveguide modeling of wind instrument bores constructed of truncated cones. In *Proc. 1994 Int. Computer Music Conf.*, pp. 423–430, Århus, Denmark. Computer Music Association.

Välimäki, V., Karjalainen, M., and Laakso, T. I. (1993). Modeling of woodwind bores with finger holes. In *Proc. 1993 Int. Computer Music Conf.*, pp. 32–39, Tokyo, Japan. Computer Music Association.

Välimäki, V., Laakso, T. I., and Mackenzie, J. (1995). Elimination of transients in time-varying allpass fractional delay filters with application to digital waveguide modelling. In ICMC (1995), pp. 327–334.

Van Duyne, S., Jaffe, D. A., Scandalis, G. P., and Stilson, T. (1996). The signal controlled allpass interpolated delay line. Presented at the CCRMA Affiliates Meeting, 1996.

Body Modeling Techniques for String Instrument Synthesis

Matti Karjalainen

Matti.Karjalainen@hut.fi
http://www.hut.fi/TKK/Yksikot/Osastot/S/Akustiikka/
Laboratory of Acoustics and Audio Signal Processing
Helsinki University of Technology

Julius Smith

jos@ccrma.stanford.edu
http://www-ccrma.stanford.edu/
CCRMA, Music Department
Stanford University

Abstract

Techniques are described for obtaining efficient computational models of stringed instrument resonators such as guitar bodies. Warping the frequency axis to an approximate Bark scale using a first-order conformal map decreases the required body filter order by a factor of 5 to 10 for a given quality level. Structures which implement frequency-warped filters are described. Techniques are described for factoring a body resonator into its least-damped and most-damped modes so that the most-damped modes can be commuted with the string and stored in a shortened look-up table or approximated by a filtered noise burst for commuted synthesis.

1 Introduction

All acoustical string instruments have some kind of a body or soundboard which is the main source of sound radiation. Such an "amplification" is necessary since a vibrating string alone has a very limited capability to move air and radiate efficiently.

Another function of a body or a soundboard is to add coloration and reverberation to the radiated sound. Since acoustic amplification is more or less based on resonating structures, the spectral content of string output signals is thus changed. Due to a relatively slow decay of the body resonances, the temporal structure of the string signals is also changed to exhibit a reverberant quality.

The third acoustic role of a body or a soundboard is to create complex directivity patterns so that the intensity in the radiated sound field is a function of direction. Combined with room acoustics, this adds spaciousness to the perceived sound.

High-quality real-time sound synthesis of string instruments, based on physical modeling and DSP techniques, has been available for several years, [Smith 1983, Karjalainen and Laine 1991, Smith 1993a, Karjalainen *et al.* 1993]. The string itself is very efficiently modeled by digital waveguide filters and extensions of the Karplus-Strong model [Smith 1983, Karplus and Strong 1983, Jaffe and Smith 1983, Smith 1987, Smith 1992, Välimäki *et al.* 1996], and the excitation may be simply implemented as a wavetable or a set of wavetables. The body (or a soundboard), although in most cases a linear and time-invariant system, is found to be computationally very expensive if a full quality synthesis is desired.

Two kinds of efficient solutions are given in literature. The commuting of the body response and consolidation with the string excitation into a wavetable, [Smith 1993a, Karjalainen *et al.* 1993], is a very practical and straightforward method but lacks parametric control of body features. Body filters with a small or moderate number of resonating modes (e.g., [Smith 1983, Stonick and Massie 1992, Bradley and Stonick 1995]) provide another efficient method, but lacking the quality of full-featured instrument bodies. These two approaches can be blended to give parametric control over the most important resonances, while retaining the full richness of remaining resonances in wavetable form.

In this paper, we give a survey of body modeling techniques for model-based sound synthesis by covering methods from full-quality filter models to commuted synthesis, as well as hybrid methods, including new body filter designs. Both traditional and new filter structures are utilized, with discussion of estimation techniques for the calibration of model parameters from measured responses of acoustic instruments. The acoustic guitar is used as the primary example in these studies.

2 Body Impulse Responses

The acoustics of string instrument bodies and soundboards is a relatively widely studied topic [Fletcher and Rossing 1990].

A natural starting point for body modeling is to measure or compute the body impulse response. Figure 9a shows the first 100 milliseconds of the impulse response from the body of an acoustic gui-

tar of classical design. The response was measured by tapping the bridge vertically with an impulse hammer (strings were damped) and by measuring the response with a microphone located one meter in front of the sound hole. Figure 2 depicts the magnitude behavior in the frequency domain for the full impulse response.

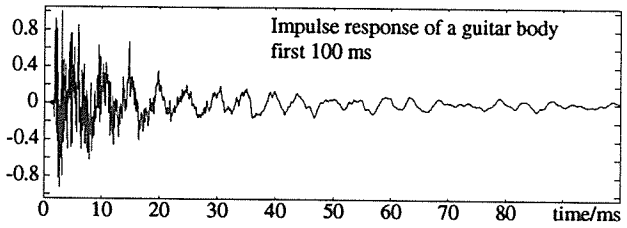


Figure 1: *An example of a body impulse response for an acoustic guitar.*

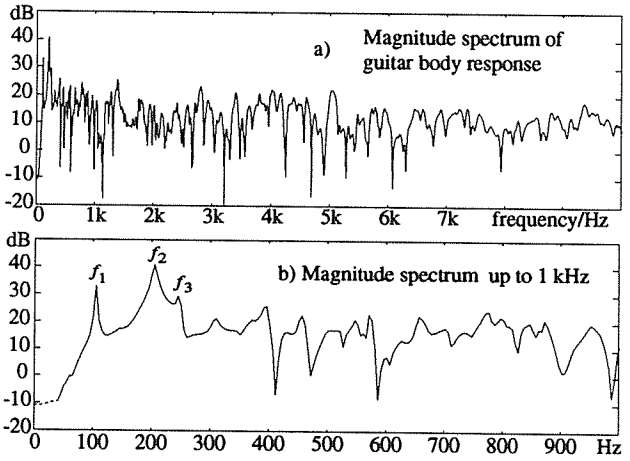


Figure 2: *Magnitude spectrum of the impulse response shown in Figure 1: a) full spectrum, b) low frequencies up to 1 kHz.*

Figure 1 suggests that the impulse response of the body is a combination of exponentially decaying sinusoids, i.e., signal components corresponding to the resonance frequencies of Figure 2. This must be the case if the transfer function of the body is linear, time-invariant, and finite order. In the frequency domain, the signal will be spectrally colored depending on how the harmonics of a string signal are located in relation to peaks or valleys of the body frequency response. It is obvious that both the magnitude spectrum shape and the temporal structure of the impulse response are important from the point of view of auditory perception.

A more comprehensive look at the body response characteristics may be achieved by a time-frequency representation which is more like the audio spectrum analysis of hearing. Figure 3 illustrates a short-time Fourier spectrum plot for the first 70 milliseconds of the response. A general increase in the decay rate at higher modal frequencies can be easily observed. We may also notice that resonance modes do not follow the simple exponential decay that would be linear curves on the dB scale. This ripple can be explained by multiple resonances that interact since they are closer together than the spectral resolution of the analysis. (At very low frequencies the alignment of window in relation to signal cycles may also be a source of ripple.[1])

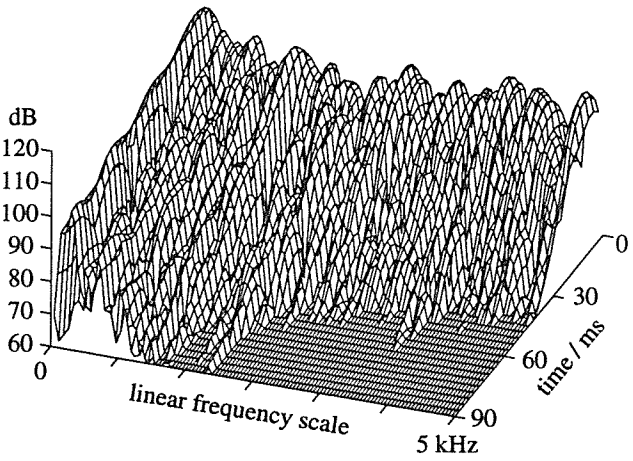


Figure 3: *Time-frequency plot of the guitar body response using short-time Fourier analysis. A Hamming window of 12 ms was used with a 3 ms hop size.*

3 Traditional Digital Filters as Body Models

The signal transfer properties from strings to radiated sound can be considered to be linear and time invariant (LTI) in most string instruments. In this case, an efficient way of implementing the body or soundboard for sound synthesis purposes is by means of digital filtering. Here we first consider the use of traditional filter structures—FIR and IIR filters—as direct implementations of body impulse responses. Then we introduce warped filter techniques and their application to body modeling.

[1] Actually, there is no perfect window size for this analysis since low frequencies require better frequency resolution and high frequencies better temporal resolution. A constant-Q or "wavelet" short-time spectrogram would be closer to an audio transform than the standard short-time Fourier transform.

FIR and IIR Filters as Body Models

A discrete-time LTI system may be represented using the z transform by

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{i=0}^{N} \beta_i z^{-1}}{1 + \sum_{i=1}^{P} \alpha_i z^{-1}} \qquad (1)$$

The most straightforward way to realize a known body response is to use the samples of a measured or computed impulse response as taps in an *FIR filter*, for which the coefficients α_i in (1) are zero. If N is larger than the impulse-response duration, this implements the desired convolution of the string output and the body response yielding a full accuracy body model to the extent that the whole audible portion of the response is available free of noise and artifacts.

An obvious problem with FIR modeling is the filter length N and thus the computational expense of the method. In the current guitar example, in order to cover a period of a single decay time-constant for the lowest mode, an FIR filter of order $N = 5000$ taps is needed when a sampling rate of 22 kHz is used. For a 60 dB dynamic range and full audio bandwidth, an FIR order of about $N = 25000$ is required! In practice, using only the first 100 ms of the response is found to be quite satisfactory, which means a 2200 tap filter for a 22 kHz sampling rate. Even this is computationally much more expensive than a model for six guitar strings, and it may be more than a modern signal processing chip can do in real time. The conclusion is that FIR models are generally impractical unless very efficient FIR hardware is available.

The sharply resonating and exponentially decaying components of a body response imply that IIR filters are more appropriate for efficient synthesis models than FIR filters. In order to see how well straightforward all-pole modeling works, we may apply autoregressive (AR) modeling using the autocorrelation method of *linear prediction* (LP) [Markel and Gray 1976] to the impulse response shown in Figure 1. This yields an all-pole filter model where the coefficients β_i of (1), for $i = 2$ to N, are equal to zero, are the predictor coefficients, and P is the order of the filter. Experiments with all-pole modeling have shown that, in our example, an order of $P = 500$ to 1000 is needed to yield a well matched temporal response. Lower filter orders, although relatively good from a spectral point of view, make the lowest resonances decay too fast. This can be addressed using a spectral weighting function (preemphasis) at the expense of the high-frequency fit.

The next generalization with traditional digital filters is to model the impulse response with a pole-zero (or ARMA) model. We have tried this using Prony's method [Parks and Burrus 1987, pp. 206–209]. The results show that this does not relax the requirements for the order P, but adding 100 zeros or so to the model improves the fit of the transient attack of the impulse response. From the point of view of auditory perception, however, this has only a small effect.

4 Warped Filters

Many filter design and model estimation methods allow for an error weighting function versus frequency in order to control the varying importance of different frequencies. Here, however, we take a different approach: Instead of an explicit weighting function we use frequency scale *warping* that is in principle applicable to any design or estimation technique. The most popular warping method is to use the bilinear conformal mapping [Churchill 1960, Parks and Burrus 1987] since it is the most general conformal mapping that preserves order. It can be used to warp the impulse response, frequency response, or transfer function polynomials. The warped FFT was introduced by [Oppenheim *et al.* 1971] and warped linear prediction was developed by [Strube 1980]. Generalized methods using the FAM functions have been developed by [Laine *et al.* 1994]. Smith has applied the bilinear mapping in order to design filter models for the violin body [Smith 1983].

The bilinear warping is realized by substituting unit delays by first-order allpass sections, i.e.

$$z^{-1} \leftarrow D_1(z) = \frac{z^{-1} - \lambda}{1 - \lambda z^{-1}} \qquad (2)$$

This means that the frequency-warped version of a filter can be implemented by such a simple replacement technique. (Modifications are needed to make warped IIR filters realizable.) The transfer function expressions after the substitution may also be expanded to yield an equivalent IIR filter of traditional form. It is easy to show that the inverse warping can be achieved with a similar substitution but using $-\lambda$ instead of λ.

The usefulness of frequency warping in our case comes from the fact that, given a target transfer function $H(z)$, we may find a lower order *warped filter* $H_w(D_1(z))$ that is a good approximation of $H(z)$. For an appropriate value of λ, the bilinear warping can fit the psychoacoustic Bark scale, based on the critical band concept, relatively accurately [Strube 1980, Zwicker 1990]. For this purpose, an approximate formula for the optimum value of λ as a function of sampling rate is given in [Smith and Abel 1995]. For a sampling rate of 44.1 kHz this yields $\lambda = 0.7233$ and for 22 kHz $\lambda = 0.6288$. When using the warping techniques, the optimality of λ in a specific application de-

pends both on auditory aspects and the characteristics of the system to be modeled.

Warped FIR (WFIR) Filters

The principle of a warped FIR filter (WFIR) is shown in Figure 4a, which may be written as

$$B_w(z) = B\left(D_1^{-1}(z)\right) = \sum_{i=0}^{M} \beta_i [D_1(z)]^i \quad (3)$$

A more detailed filter structure for implementation is depicted in 4b. As the latter form shows, a warped FIR is actually recursive, i.e., an IIR filter with M poles at $z = \lambda$, where M is the order of the filter.

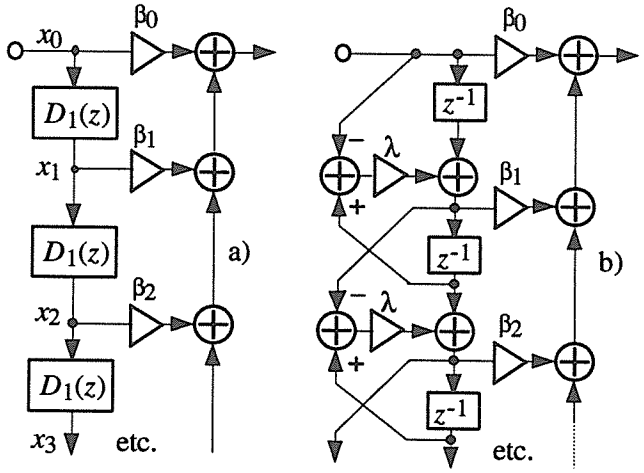


Figure 4: *Warped FIR modeling: (a) general principle, (b) detailed filter structure for implementation.*

A straightforward method to get the tap coefficients β_i for a WFIR filter is to warp the original impulse response and to truncate it by "windowing" the portion that has amplitude above a threshold of interest. (Notice that the bilinear mapping of a signal by (2) is linear but not shift invariant [Strube 1980]). There exist various formulations for computing a warped version of a signal [Strube 1980, Smith 1983, Laine *et al.* 1994]. An accurate and numerically stable method is to apply the FIR filter structure of Figure 4a or 4b with tap coefficients being the samples of the signal to be warped. When an impulse is fed to this filter, the response will be the warped signal. Figure 5 shows the warped ($\lambda = 0.63$) guitar body response as a time-frequency plot for comparison with the original one in Figure 3. As can easily be seen, the warping tends to balance the decay rates and resonance bandwidths for all frequency ranges.

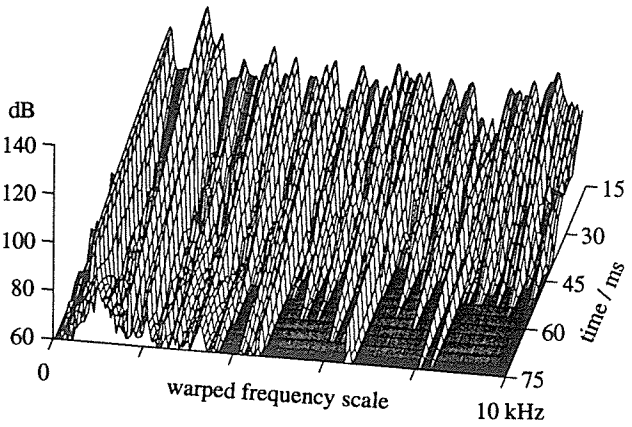


Figure 5: *Time-frequency plot of warped guitar body response. A Hamming window of 24 ms* was used with a 3 ms* hop size, where * denotes warped time.*

Warped IIR (WIIR) Filters

When linear prediction is applied to a warped impulse response it yields a warped all-pole filter. Other methods for warped LP analysis (WLP) are studied in [Laine *et al.* 1994], including an efficient way to compute warped autocorrelation coefficients $r_w(i)$ directly from the original signal. This is based on the warped delay-line structure of Figure 4a, whereby

$$r_w(i) = \sum_n x_0(n) x_i(n) \quad (4)$$

is summed over a time interval or window of interest. After that, the warped predictor coefficients are achieved from warped autocorrelation coefficients as usual [Markel and Gray 1976] to yield a filter model

$$H_w(D_1(z)) = \frac{G_w}{1 + \sum_{i=1}^{R} \alpha_i [D_1(z)]^i} \quad (5)$$

A somewhat surprising observation is that the filter structure of (5) cannot be implemented directly since there will be delay-free loops in the structure for $\lambda \neq 0$. (Of course the bilinear mapping, inherent in the filter structure, may be expanded at design time to yield a normal IIR filter. This, however, can lead to numerical difficulties if the filter order exceeds about 20 to 30 in 64-bit double precision floating-point.) Figure 6 depicts two realizable forms of WIIR filters. Strube [1980] suggested an approximation in which lowpass sections are used instead of allpass sections (see Figure 6a). In practice, it works only for low orders and warping values λ due to excessive high-frequency attenuation otherwise. The version in Figure 6b is more general for warped pole-zero modeling but it has also a more complex structure. The coefficients σ_i can be computed from

coefficients α_i of (5) using a recursion or matrix operation as shown in [Karjalainen 1996].

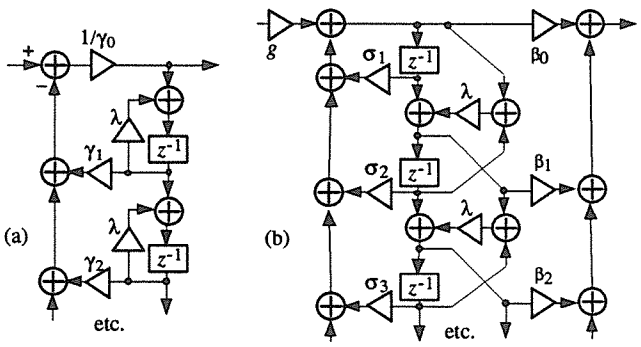


Figure 6: *Filter structures for implementation of warped IIR filters: (a) lowpass structure that does not work with high orders, and (b) modified allpass structure (warped pole-zero filter).*

Body Modeling with Warped Filters

For a given quality level, the warped filter strategy using a Bark-scale warping yields a reduction in filter order by a factor 5 to 10. This means that a WFIR of order M less than 500 gives results similar to those of an FIR filter of order $N = 2000$. For warped all-pole filters, an order of about $R = 100$ is equivalent to normal IIR order of about $P = 500$ to 1000. A body filter resulting from using Prony's method for a warped filter, with $M = 50$ to 100 and $R = 100$ to 200, will represent both transient and decay properties relatively well, although a Bark warping ($\lambda = 0.63$, sampling rate 22 kHz) has a tendency to shorten the impulse response of the highest frequencies a bit too much.

The reduction in filter order due to warping translates to a similar reduction in computational complexity only if unit delays and allpass sections are equally complex computationally. In reality, many digital signal processors have hardware support to run ordinary FIR and IIR filters very efficienly. The complexity of the first-order allpass section used as a warped delay is several times higher than that of a unit delay. Due to this complexity of realization, reduction in computational cost with warped all-pole and IIR structures remains smaller than indicated by the order savings. In a typical case, for the TMS320C30 floating-point signal processor, a WIIR body filter model is only about two times faster than an equivalent normal IIR filter. On the other hand, if the warped filters are converted to conventional structures (using ultra-high precision computations, e.g., in Mathematica), the extra complexity disappears; since the warping preserves order, it does not have to increase filter complexity except when the number of poles is different from the number of zeros in which case the warping (or unwarping) will introduce new poles or zeros so that their number is the same.

A nice benefit of implementing the body filter in warped form is that λ is available as a qualitative *body size* parameter. The size parameter can be modulated to obtain new kinds of effects, or it can be used to "morph" among different members of an instrument family.

5 Body-Model Factoring

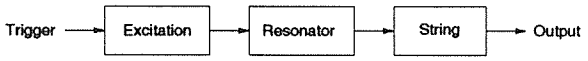


Figure 7: *Schematic model of a stringed instrument in which the string and resonator are commuted relative to their natural ordering.*

Commuted synthesis is a technique in which the body resonator is commuted with the string model, as shown in Fig. 7, in order to avoid having to implement a large body filter at all [Smith 1993a, Karjalainen *et al.* 1993]. In commuted synthesis, the excitation (e.g., plucking force versus time) can be convolved with the resonator impulse response to provide a single aggregate excitation signal. This signal is short enough to store in a look-up table, and a note is played by simply summing it into the string.

A valuable way of shortening the excitation table in commuted synthesis is to *factor* the body resonator into its *most-damped* and *least-damped* modes. The most-damped modes are then commuted and combined with the excitation in impulse-response form. The least-damped modes can be left in parametric form as recursive digital filter sections. Advantages of this factoring include the following:

- The excitation table is shortened.
- The excitation table signal-to-quantization-noise ratio is improved.
- The most important resonances remain *parametric*, facilitating real-time control.
- Multiple body outputs become available.
- Resonators may be already available in a separate effects unit, making them "free."
- A memory vs. computation trade-off is available for cost optimization.

Mode Extraction Techniques

The goal of resonator factoring is to identify and remove the least-damped resonant modes of the

impulse response. In principle, this means ascertaining the precise resonance frequencies and bandwidths associated with each of the narrowest "peaks" in the resonator frequency response, and dividing them out via inverse filtering, so they can be implemented separately as resonators in cascade. If in addition the amplitude and phase of a resonance peak are accurately measurable in the complex frequency response, the mode can be removed by complex spectral subtraction (equivalent to subtracting the impulse-response of the resonant mode from the total impulse response); in this case, the parametric modes are implemented in a parallel bank as in [Bradley and Stonick 1995]. However, in the parallel case, the residual impulse response is not readily commuted with the string.

Various methods are available for estimating the mode parameters for inverse filtering:

- Amplitude response peak measurement
- Weighted digital filter design
- Linear prediction
- Sinusoidal modeling
- Late impulse-response analysis

In the body factoring example presented below, the frequency and bandwidth of the main Helmholtz air mode were measured manually using an interactive spectrum analysis tool. It is also easy to automate peak-finding in FFT magnitude data, as is routinely done in sinusoidal modeling, discussed further below.

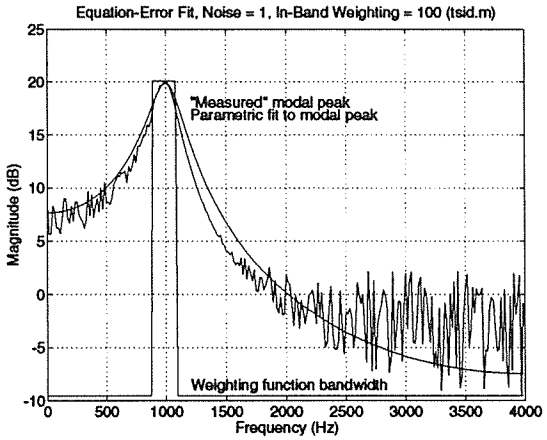


Figure 8: *Illustration of one way to determine the parameters of a least-damped resonant mode.*

Many methods for digital filter design support spectral weighting functions that can be used to focus in on the least-damped modes in the frequency response. One is the *weighted equation-error* method which is available in the matlab `invfreqz()` function. Figure 8 illustrates use of it in a simple synthetic example with only one frequency-response peak in the presence of noise. Unless the weighting function is very tight around the peak, its bandwidth tends to be overestimated.

Another well known method for converting the least-damped modes into parametric form is Linear Prediction (LP) followed by polynomial factorization to obtain resonator poles. LP is particularly good at fitting spectral peaks due to the nature of the error criterion it minimizes [Smith 1983, pp. 43–50]. The poles closest to the unit circle in the z plane can be chosen as the least-damped part of the resonator. It is well known that when using LP to model spectral peaks for extraction, orders substantially higher than twice the number of spectral peaks should be used.

Another way to find the least-damped mode parameters is by means of a *sinusoidal model* of the body impulse response such as is often used to determine the string loop filter in waveguide string models [Smith 1993b, Välimäki *et al.* 1996]. (See, e.g., [Serra and Smith 1990] for further details on sinusoidal modeling and supporting C software). The sinusoidal amplitude envelopes yield a particularly robust measurement of bandwidth. Theoretically, the modal decay should be exponential. Therefore, on a dB scale, the amplitude envelope should decay *linearly.* Linear regression can be used to fit a straight line to the measured log-amplitude envelope of the impulse response of each long-ringing mode. Note that even when amplitude modulation is present due to mode couplings, the ripples tend to average out in the regression and have little effect on the slope measurement. This robustness can be enhanced by starting and ending the linear regression on local maxima in the amplitude envelope.

All methods useable with inverse filtering can be modified based on the observation that late in the impulse response, the damped modes have died away, and the least-damped modes dominate. Therefore, by discarding initial impulse-response data, the problem in some sense becomes "easier" at the price of working closer to the noise floor.

High-Frequency Modes ≈ Noise

Figure 9b suggests that the many damped modes remaining in the shortened body impulse response may not be clearly audible as resonances since their decay time is so short. This is confirmed by listening to shortened and spectrally flattened body responses which sound somewhere between a click and a noise burst. These observations suggest that the shortened, flattened, body response can be replaced by a psychoacoustically equivalent *noise burst*, as suggested for modeling piano soundboards [Van Duyne and Smith 1995]. Thus,

the signal of Fig. 9b can be synthesized qualitatively by a white noise generator multiplied by an amplitude envelope. In this technique, it is helpful to use LP on the residual signal to flatten it. As a refinement, the noise burst can be time-varying filtered so that high frequencies decay faster [Van Duyne and Smith 1995]. Thus, the stringed instrument model may consist of *noise generator* → *excitation amplitude-shaping* → *time-varying lowpass* → *string model* → *parametric resonators* → *multiple outputs*. In addition, properties of the physical excitation may be incorporated, such as comb filtering to obtain a virtual pick or hammer position. Multiple outputs provide for individual handling of the most important resonant modes and facilitate simulation of directional radiation characteristics [Välimäki *et al.* 1996].

Body Factoring Example

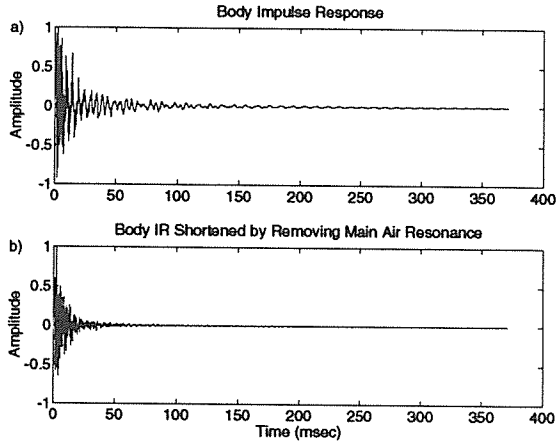


Figure 9: *Impulse response of a classical guitar body before and after removing the first peak (main air resonance) via the inverse filter defined by Eq. (6), with* $a_1 = -1.9963$ *and* $a_2 = 0.9972$.

Figure 9a shows the measured guitar-body impulse-response data plotted in Fig. 1 but extended to its full duration. Figure 9b shows the same impulse response after factoring out a single resonating mode near 100 Hz (the main Helmholtz air mode). As can be seen, the residual response is considerably shorter than the original.

Figure 10a shows the measured guitar-body amplitude response after warping to an approximate Bark frequency axis. Figure 10b shows the Bark-warped amplitude response after the main Helmholtz air mode is removed by inverse filtering. On the Bark frequency scale, it is much easier numerically to eliminate the main air mode.

The modal bandwidth used in the inverse filtering was chosen to be 10 Hz* which corresponds

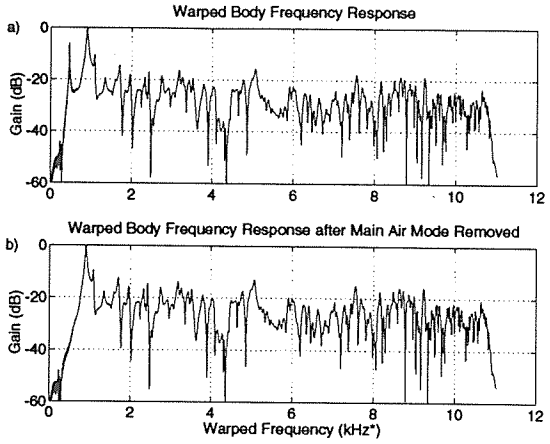


Figure 10: *Normalized Bark-warped amplitude response of a classical guitar body before and after removing the first peak (main air mode) via Eq. (6), with* $a_1 = -1.9801$ *and* $a_2 = 0.9972$.

to a Q of 46 for the main air mode. If the Bark-warping is done using a first-order conformal map [Smith and Abel 1995], its inverse preserves filter order [Smith 1983, pp. 61–67]. Applying the inverse warping to the parametric resonator drives its pole radius from 0.99858 in the Bark-warped z plane to 0.99997 in the unwarped z plane.

Note that if the inverse filter consists only of two zeros determined by the spectral peak parameters, other portions of the spectrum will be modified by the inverse filtering, especially at the next higher resonance, and in the linear trend of the overall spectral shape. To obtain a more *localized* mode extraction (useful when the procedure is to be repeated), we define the inverse filter as

$$H_r(z) \triangleq \frac{A(z)}{A(z/r)} \triangleq \frac{1 + a_1 z^{-1} + a_2 z^{-2}}{1 + a_1 r z^{-1} + a_2 r^2 z^{-2}} \quad (6)$$

where $A(z)$ is the inverse filter determined by the peak frequency and bandwidth, and $A(z/r)$ is the same polynomial with its roots contracted by the factor r. If r is close to but less than 1, the poles and zeros substantially cancel far away from the removed modal frequency so that the inverse filter has only a local effect on the frequency response. In the computed examples, r was arbitrarily set to 0.9, but it is not critical.

6 Conclusions

A number of techniques were discussed for improving the quality of virtual stringed instrument resonators and reducing implementation cost. These methods make it possible to simulate high quality stringed instruments using inexpensive software algorithms. As an example, a basic classical guitar model requires less than two percent of a 120

MHz Pentium processor for each actively vibrating string. Therefore, such models are practical today, even for low-cost multimedia applications. For the future, physical models suggest how to make good use of increased processing power as it comes along.

References

[Bradley and Stonick 1995] Bradley, K., and V. Stonick. 1995. "Automated Analysis and Computationally Efficient Synthesis of Acoustic Guitar Strings and Body." *In: Proc. IEEE Workshop on Appl. Signal Processing to Audio and Acoustics, New Paltz, NY.* New York: IEEE Press. Session 9a, paper 7.

[Churchill 1960] Churchill, R. V. 1960. *Complex Variables and Applications.* New York: McGraw-Hill.

[Fletcher and Rossing 1990] Fletcher, N. H., and T. D. Rossing. 1990. *The Physics of Musical Instruments.* New York: Springer Verlag.

[Jaffe and Smith 1983] Jaffe, D. A., and J. O. Smith. 1983. "Extensions of the Karplus-Strong Plucked String Algorithm." *Computer Music J.*, 7(2):56–69. Reprinted in *The Music Machine*, Roads, C., (ed.), MIT Press, 1989.

[Karjalainen 1996] Karjalainen, M. 1996. *Unpublished manuscript.*

[Karjalainen and Laine 1991] Karjalainen, M., and U. K. Laine. 1991. "A Model for Real-Time Sound Synthesis of Guitar on a Floating-Point Signal Processor." *Pages 3653–3656 of: Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Toronto*, vol. 5. New York: IEEE Press.

[Karjalainen *et al.* 1993] Karjalainen, M., V. Välimäki, and Z. Jánosy. 1993. "Towards High-Quality Sound Synthesis of the Guitar and String Instruments." *Pages 56–63 of: Proc. 1993 Int. Computer Music Conf., Tokyo.* Computer Music Association.

[Karplus and Strong 1983] Karplus, K., and A. Strong. 1983. "Digital Synthesis of Plucked String and Drum Timbres." *Computer Music J.*, 7(2):43–55.

[Laine *et al.* 1994] Laine, U. K., M. Karjalainen, and T. Altosaar. 1994. "Warped Linear Prediction (WLP) in Speech and Audio Processing." *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, Adelaide, Australia*, pp. III:349–352.

[Markel and Gray 1976] Markel, J. D., and A. H. Gray. 1976. *Linear Prediction of Speech.* New York: Springer Verlag.

[Oppenheim *et al.* 1971] Oppenheim, A. V., D. H. Johnson, and K. Steiglitz. 1971. "Computation of Spectra with Unequal Resolution Using the Fast Fourier Transform." *Proc. IEEE*, 59:299–301.

[Parks and Burrus 1987] Parks, T. W., and C. S. Burrus. 1987. *Digital Filter Design.* New York: John Wiley and Sons, Inc.

[Serra and Smith 1990] Serra, X., and J. O. Smith. 1990. "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic plus Stochastic Decomposition." *Computer Music J.*, 14(4):12–24. Software available under http://www.iua.upf.es/~xserra/.

[Smith 1983] Smith, J. O. 1983 (June). *Techniques for Digital Filter Design and System Identification with Application to the Violin.* Ph.D. thesis, Elec. Eng. Dept., Stanford University.

[Smith 1987] Smith, J. O. 1987. "Waveguide Filter Tutorial." *Pages 9–16 of: Proc. 1987 Int. Computer Music Conf., Champaign-Urbana.* Computer Music Association.

[Smith 1992] Smith, J. O. 1992. "Physical Modeling Using Digital Waveguides." *Computer Music J.*, 16(4):74–91. Special issue: Physical Modeling of Musical Instruments, Part I.

[Smith 1993a] Smith, J. O. 1993a. "Efficient Synthesis of Stringed Musical Instruments." *Pages 64–71 of: Proc. 1993 Int. Computer Music Conf., Tokyo.* Computer Music Association.

[Smith 1993b] Smith, J. O. 1993b. "Structured Sampling Synthesis: Automated Construction of Physical Modeling Synthesis Parameters and Tables from Recorded Sounds." *(Presentation overheads, 51 pages), CCRMA Associates Conference*, May.

[Smith and Abel 1995] Smith, J. O., and J. S. Abel. 1995. "The Bark Bilinear Transform." *In: Proc. IEEE Workshop on Appl. Signal Processing to Audio and Acoustics, New Paltz, NY.* New York: IEEE Press. Available online at http://www-ccrma.stanford.edu/~jos/.

[Stonick and Massie 1992] Stonick, V., and D. Massie. 1992. "ARMA Filter Design for Music Analysis/ Synthesis." *Pages II:253–256 of: Proc. Int. Conf. Acoustics, Speech, and Signal Processing, San Francisco.* New York: IEEE Press.

[Strube 1980] Strube, H. W. 1980. "Linear Prediction on a Warped Frequency Scale." *J. Acoustical Soc. of America*, 68(4):1071–1076.

[Välimäki *et al.* 1996] Välimäki, V., J. Huopaniemi, M. Karjalainen, and Z. Jánosy. 1996. "Physical Modeling of Plucked String Instruments with Application to Real-Time Sound Synthesis." *J. Audio Eng. Soc.*, May.

[Van Duyne and Smith 1995] Van Duyne, S. A., and J. O. Smith. 1995. "Developments for the Commuted Piano." *Pages 335–343 of: Proc. 1995 Int. Computer Music Conf., Banff.* Computer Music Association.

[Zwicker 1990] Zwicker, E. 1990. *Psychoacoustics.* New York: Springer Verlag.

Alias-Free Digital Synthesis of Classic Analog Waveforms

Tim Stilson (stilti@ccrma.stanford.edu)
Julius Smith (jos@ccrma.stanford.edu)
CCRMA (http://www-ccrma.stanford.edu/)
Music Department, Stanford University

Abstract

Techniques are presented for alias-free digital synthesis of classical analog synthesizer waveforms such as pulse train and sawtooth waves. Bandlimited impulse trains are generated as a superposition of windowed sinc functions. Bandlimited pulse and triangle waveforms are obtained by integrating the difference of two out-of-phase bandlimited impulse trains. Variations for efficient implementation are discussed.

1 Introduction

Any analog signal with a discontinuity in the waveform (such as pulse train or sawtooth) or in the waveform slope (such as triangle wave) must be *bandlimited* to less than half the sampling rate before sampling to obtain a corresponding discrete-time signal. Simple methods of generating these waveforms digitally contain *aliasing* due to having to round off the discontinuity time to the nearest available sampling instant. The signals primarily addressed here are the impulse train, rectangular pulse, and sawtooth waveforms. Because the latter two signals can be derived from the first by integration, only the algorithm for the impulse train is developed in detail.

2 Related Techniques

Additive synthesis (see [Roads 1996] for a summary of this and other synthesis techniques discussed in this section) can be trivially bandlimited simply by not generating harmonics higher than $F_s/2$. A bandlimited impulse train at fundamental frequency f_1 can be generated using additive synthesis by summing $N = \lfloor (F_s/2)/f_1 \rfloor$ cosine oscillators.

Periodic wavetable synthesis [Mathews 1969] (not to be confused with sample playback synthesis which is also called wavetable synthesis these days) can be made free of aliasing by use of bandlimited interpolation when accessing the wavetable [Smith and Gossett 1984]. In this case, the wavetable contains a 1 followed by all zeros (an impulse).

Discrete-Summation Formulae (DSF) [Moorer 1975] can be used to synthesis a bandlimited impulse train algorithmically based on a closed-form expression for a sum of cosines. The Systems Concepts Digital Synthesizer implemented this method in hardware, and it is used in CSound's buzz and gbuzz unit generators. A disadvantage of DSF relative to the previous two is that, when calculating harmonics all the way to $F_s/2$, the number of harmonics changes with frequency, which causes the highest harmonic to "pop" in or out as the pitch frequency glides down or up. The previous methods (and the method we will discuss) can allow the harmoncs to die out (or come in) slowly and imperceptibly.

Formant synthesis techniques, such as VOSIM [Kaegi and and S. Tempelaars 1978], Chant [Rodet *et al.* 1989], and linear prediction [Roads 1996] can be modeled as an impulse train driving a formant filter where the timing of the impulses is rounded to the nearest sampling instant. This impulse-time rounding causing pitch-period jitter which is a form of aliasing. Eliminating this jitter in Chant or VOSIM requires resampling the filter impulse response each period which would be very expensive. Using a bandlimited impulse train as described here to drive a formant filter will eliminate this pitch-period jitter.

3 Bandlimited Impulse Train (BLIT) Synthesis

The standard operation before sampling is to apply an anti-aliasing filter. The ideal anti-aliasing filter has a continuous-time impulse response that is a sinc function with a zero-crossing interval of one sample:

$$h_s(t) \triangleq \mathrm{sinc}(F_s t) \triangleq \frac{\sin(\pi F_s t)}{\pi F_s t}.$$

The ideal unit-amplitude impulse train with period T_1 seconds is given by

$$x(t) = \sum_{l=-\infty}^{\infty} \delta(t + lT_1)$$

Applying the anti-aliasing filter h_s to this signal and sampling at the sampling rate $F_s = 1/T_s$ gives

$$y(n) = \sum_{l=-\infty}^{\infty} \mathrm{sinc}(n + lP)$$

where $P = T_1/T_s$ is the period in samples (not normally an integer). The above expression can be interpreted as a *time aliasing* of the sinc function about an interval of P samples, and it can be shown to be given by

$$y(n) = (M/P)\mathrm{Sinc}_M[(M/P)n] \qquad (1)$$

where

$$\mathrm{Sinc}_M(x) \triangleq \frac{\sin(\pi x)}{M \sin(\pi x/M)}$$

This function provides a closed-form expression for the sampled bandlimited impulse train (BLIT), and it can be used directly for synthesis in a manner similar to DSF. While P is the period in samples, M is the number of harmonics. It is always odd because an impulse train has one "harmonic" at DC, and an even number of non-zero harmonics, provided no harmonic is allowed at exactly half the sampling rate (which we enforce). Note that M/P is always close to 1. When P is an odd integer, $P = M$, and $y(n)$ is simply $\mathrm{Sinc}_M(n)$. As P departs from M, Eq. (1) implements a time scaling along with a compensating amplitude scaling. We can relate the number of harmonics M to the period P of the impulse train as

$$M = 2\lfloor P/2 \rfloor + 1$$

i.e., M is the nearest odd integer to the period P in samples.

Sum of Windowed Sincs (BLIT-SWS)

A more efficient method for synthesizing digital impulse trains may be based on the windowed-sinc method for general bandlimited interpolation [Smith and Gossett 1984]. The technique is equivalent conceptually to bandlimited periodic wavetable synthesis of an impulse train, as mentioned in the previous section: Bandlimited interpolatation is to convert the sampling rate of a discrete-time unit sample pulse train from a pitch which divides the sampling rate to the desired pitch. The rate conversion causes each unit sample pulse $\delta(n)$ to be replaced by a windowed sinc function $w(t)h_s(t)$ sampled at some phase which generally varies each period.

Because the windowing imposes a finite fall-off rate in the harmonics, some aliasing is inevitable. We can, however, control this by our choice of window. It is also helpful to helpful to have a small oversampling factor so that there is a good sized guard band between the upper limit of human hearing and half the sampling rate. This reduces the window length required.

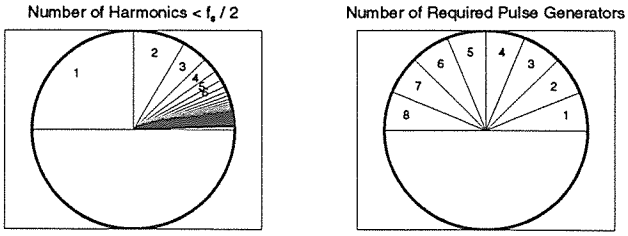


Figure 1: *Comparing number of harmonics to number of overlapped pulse instances for a pulse 8 samples long.*

A further optimization comes from comparing the number of sincs that must be overlapped in the BLIT-SWS method to the number of harmonics of the BLIT that land below $F_s/2$. At very high frequencies, the number of bandlimited harmonics becomes quite small. Indeed, in the top octave, only the fundamental is in band. Thus for a large percentage of the frequency range, it is quite likely that it may be more efficient to generate a BLIT (or any other harmonic waveform) by simple summation of sines. At lower frequencies, we can again revert to the BLIT-SWS method because it is obviously more efficient at low frequencies, where the number of harmonics is very large.

Like additive synthesis and bandlimited wavetable synthesis, and unlike DSF, in BLIT-SWS synthesis the highest harmonic need not audibly "pop" in or out as it comes down from or gets up to half the sampling rate, since the window function can be chosen to exhibit any harmonic falloff rate.

Example Spectrum

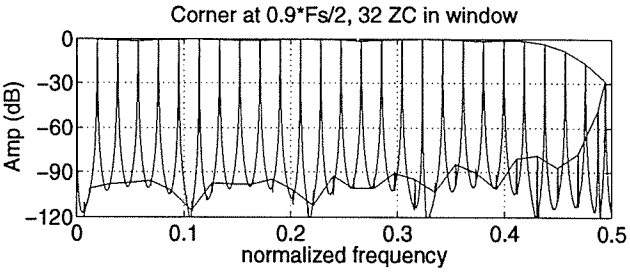


Figure 2: *Spectrum of BLIT-SWS impulse train with a line drawn through the harmonic peaks, both in-band and aliased.*

Figure 2 shows the spectrum of a bandlimited impulse train generated using the BLIT-SWS method with 32 sinc zero-crossings under a Blackman window. Note that, as is done in bandlimited interpolation, the cut-off frequency of the windowed sinc function is lowered below half the sampling rate so that its transition band is folded in half as it falls

into half the sampling rate and reflects. Only the upper 10% of the spectrum is heavily aliased. If the limit of human hearing is 20 kHz, this means we need a 2 kHz guard band, so the sampling rate should be at least 44 kHz.

4 Square-Wave and Sawtooth-Wave Generation

The next major class of analog waveforms are Square waves and Sawtooth waves.[1] We will show how to easily derive these from a BLIT via integrations which are linear transforms (that can be implemented with trivial filters), so that they preserve the bandlimited nature of the BLIT.

Successive Integration of BLIT

Sawtooth

A sawtooth function can be generated as follows:

$$Saw(n) = \sum_{k=0}^{n} BLIT(k) - C_1$$

where $C_1 = \frac{1}{\text{Period}} \sum_0^{\text{Period}} \text{BLIT(k)}$, the DC component of the BLIT.

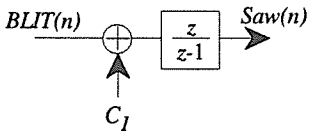


Figure 3: *Direct Sawtooth Generation*

Which is trivally impementable with a single sum and one-pole digital filter. The offet C_1 is the average value of $BLIT$, which should be subtracted off to keep the integration from ramping off to infinity (or saturating).

Rectangle

$$\begin{aligned} Rect(n) &= \sum_{k=0}^{n} \text{BLIT(k)} - \text{BLIT(k} - \text{k}_0) - C_2 \\ &= \sum_{k=0}^{n} \text{BP-BLIT}_{k_0}(k) - C_2 \end{aligned}$$

Where BP-BLIT is a "BiPolar" BLIT, whose pulses alternate sign. See Section 5 for discussion on hacks for efficiently generating BP-BLIT when using DSF BLITs. It turns out that a bipolar impulse train has a DC component of zero, which means that $C_2 = 0$. The rectangle width is controlled with k_0, which can be varied to give PWM (pulse-width modulation). The range of k_0 in these equations is [0,Period]. Depending on the implementation of the BLIT, the PWM control may also be in the range [0,1].

[1] To avoid confusion, we will use the following naming convention: A "square wave" is a rectangle wave with 50% duty cycle (i.e., "rectangle wave" means a wave that can have other duty cycles). A "triangle wave" can have asymmetric up/down slopes (including the 50% duty-cycle version), and a "sawtooth" wave must have infinite-slope transitions (either up or down). Thus a sawtooth wave is a triangle wave with either 0% or 100% duty cycle. All waves can have unipolar, bipolar, or arbitrary-offset versions.

Triangle

$$Tri(n) = \sum_{k=0}^{n} Rect(k) - C_3$$

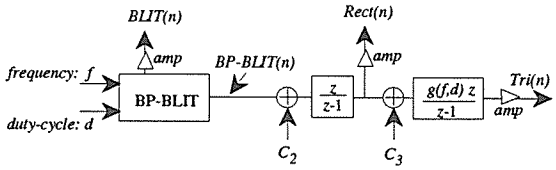


Figure 4: *Rectangle and Triangle Generation*

The offset C_3 is a function of the rectangle wave duty cycle and of a DC offset that arises from the initial conditions of the integration that produces the rectangle wave. For the Triangle to have the same amplitude as the rectangle wave, a frequency- and duty-cycle-dependant scaling must be performed on the Triangle integration:

$$\begin{aligned} Tri(n) &= \sum_{k=0}^{n} g(f,d)(Rect(k) - C_3) \\ g(f,d) &= \frac{2f}{d(1-d)} \end{aligned}$$

Where f is the frequency in units of (cycles/sample), and d is the duty cycle ($d \in [0,1]$).

Appropriate Scalings/Offsets and Non-Steady-State Fixups

In order to keep the integrators from ramping their outputs to infinity, any DC offset in the input to the integrator must be avoided. As already noted, BP-BLIT has no DC offset, so there need be no special offsets for the first integration. The initial conditions of the first integrator, however, can produce a DC offset on the output that must be canceled before the second integration. The value of this offset is also dependant on the duty-cycle of the signal so that the correct (zero-offset) initial condition will depend on: (1) desired phase, and (2) desired duty cycle. It is important to remember that the old state (right before the change) acts as the initial condition for the integrator when the parameters are changed. Since the Second integraotr has a frequency-dependant gain, chaning frequency will also cause an offset that must be accounted for.

Leaky Integrators

The use of pure integrators can numerical errors to accumulate in the integrators, causing unwanted offsets in the signals, making the second integration essentially useless. Therefore, we move the integrator poles wfrom the unit circle slightly. These "leaky" integrators slowly forget bad initial conditions and numerical errors. Forthermore, in steady-state, the outputs of the integrators will have no DC component (because BP-BLIT has none), *regardless of initial conditions*, since the leaky integrators eventually forget them. Thus, if one can live with occasional transient DC offsets (which decay at the leak rate), then just the presence of the leaky integrators can handle all offset cases. These transients can be reduced by temporarily increasing the forgetting rate of the integrators.

If one still requires the absence of any DC offset, transient or not, then the presense of the leaky integrators makes the problem of computing appropriate offsets much more difficult.

Defs of Amplitude

Moore presents a discussion of amplitude compensation in his DSF paper. Similar compensation is necessary in BLIT generation. The compensation to be used depends on how one defines amplitude, which depends on how the signal is to be used. If the signal is to be used as an audio signal, signal power or some psychoacoustic loudness measure is appropriate, but if the signal is to be used as a control signal, a maximum-value (Chebychev) measure is more appropriate.

5 Generating Bipolar BLITs with DSF

Although it is not the most efficient method for generating BLITs, DSF can be used, and is quite interesting theoretically. Here we describe how to generate BP-BLIT using DSF without having to resort to the straight-forward difference-of-shifted-BLITs scheme.

First, we note that BLITs can be generated via DSF by replacing the sin by cos in the DSF formulas.

50% duty cycle: Next, it can be shown that using a negative a in the DSF formula $\sum_{k=1}^{N} a^k \sin(0+kf_1t)$ produces a signal that is shifted from the positive-a signal by exactly half a cycle, this gives a slightly more efficient (or elegant...) way of producing the shifted BLIT than offsetting t. This leads to showing that:

$$\sum_{k=1}^{N} a^k \cos(b+ck) - \sum_{k=1}^{N} (-a)^k \cos(b+ck)$$
$$= 2a \sum_{k=1}^{N/2} (a^2)^k \cos(\,(b+c)+2ck)$$

Thus a 50% duty-cycle bipolar DSF BLIT can be generated almost as efficiently as a single DSF BLIT. For other duty cycles, there is another variation on DSF that is of interest.

If we let a be complex and take either the real or imaginary part of the DSF, this imposes a $\sin(k\angle(a))$ (or cosine) amplitude envelope onto the harmonics, which is equivalent to a comb filtering, which in turn is equivalent to summing a real DSF with a shifted version of itself (possibly with a sign flip). See Figure ??. Thus we generate BP-BLIT simply by making a complex.

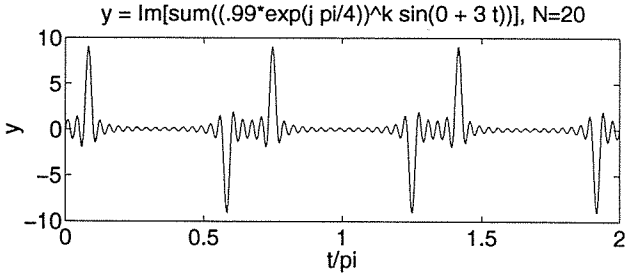


Figure 5: *Using complex multiplies in DSF to generate BP-BLIT*

References

[Kaegi and and S. Tempelaars 1978] Kaegi, and W. a S. Tempelaars. 1978. "VOSIM—A New Sound Synthesis System." *J. Audio Eng. Soc.*, 26(6):418–24.

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

[Moorer 1975] Moorer, J. A. 1975. "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulae." *J. Audio Eng. Soc.*, 24(Dec.):717–727 (Also svailable as CCRMA Report No. STAN-M-5).

[Roads 1996] Roads, C. 1996. *Computer Music Tutorial.* Cambridge, MA: MIT Press.

[Rodet *et al.* 1989] Rodet, X., Y. Potard, and J. Barrière. 1989. "The CHANT Project: From the Synthesis of the Singing Voice to Synthesis in General.". *Pages 449–465 of:* Roads, C. (ed), *The Music Machine.* Cambridge, MA: MIT Press.

[Schafer and Rabiner 1973] Schafer, R. W., and L. R. Rabiner. 1973. "A Digital Signal Processing Approach to Interpolation." *Proc. IEEE*, 61(June):692–702.

[Smith and Gossett 1984] Smith, J. O., and P. Gossett. 1984. "A Flexible Sampling-Rate Conversion Method." *Pages 19.4.1–19.4.2 of: Proc. ICASSP, San Diego*, vol. 2. New York: IEEE Press (An expanded tutorial based on this paper is available at `ftp://ccrma-ftp.stanford.edu/pub/DSP/Tutorials/` The tutorial can be browsed online at `http://www-ccrma.stanford.edu/~jos/`).

A longer version of this paper can be found at (`http://www-ccrma.stanford.edu/ stilti/papers`)

Analyzing the Moog VCF with Considerations for Digital Implementation

Tim Stilson (stilti@ccrma.stanford.edu)
Julius Smith (jos@ccrma.stanford.edu)
CCRMA (http://www-ccrma.stanford.edu/)
Music Department, Stanford University

Abstract

Various alternatives are explored for converting the Moog four-pole Voltage Controlled Filter (VCF) to discrete-time form for digital implementation in such a way as to preserve the usefulness of its control signals. The well known bilinear transform method yields a delay-free loop and cannot be used without introducing an ad-hoc delay. Related methods from digital control theory yield realizable forms. New forms motivated by root locus studies give good results.

1 Introduction

The Voltage-Controlled Filter (VCF) designed and implemented by Robert Moog is an influential filter in the history of electronic music. In this paper, the filter is analyzed in continuous time and then several transformations of the filter into discrete time are analyzed for various properties such as efficiency, ease of implementation, and the retention of certain of the original filter's good properties, such as constant-Q, and separability of the Q and tuning controls. The Root-Locus, a particularly useful tool from control systems, is used extensively in the analysis of the VCFs.

The various transformations that turn continuous-time filters into discrete-time filters each have different characteristics that affect how the properties of the continuous-time system map into the discrete domain. Some transforms that will be studied are the backwards-difference transform and the bilinear transform. In a filter such as the Moog VCF, a possible goal in the move to the discrete domain is to preserve constant-Q. Under our definition of constant-Q, a transformation cannot made which is finite-order rational. We will see how well the rational transforms approximate constant-Q.

In this work, Root-Locus techniques were found to be extremely useful. The Root-Locus comes from control-systems analysis and has particular usefulness in the analysis of systems with sweepable control inputs (inputs intended to have signal-rate updates, such as audio-rate modulation or smooth sweeps of parameters susceptible to zippering). Because the amount of processing available to translate these parameters into algorithm parameters is typically in short supply (so the algorithm is typically designed around these parameters), the parameter usually enters into the filter's equations simply, maybe even linearly. The traditional Root-Locus can plot the locations of the system's poles with variations in the parameter if the parameter enters in linearly, and many techniques in control-system synthesis can be applied to the design to keep the complexity down. The rules of how the root locus works also give the designer new tools and hints for sweepable filter design.

2 The Moog VCF

The VCF used in Moog synthesizers employs the filter structure shown in Fig. 1.

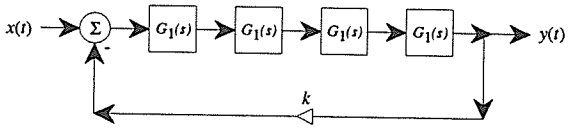


Figure 1: *The Moog VCF.*

The transfer function of each section is

$$G_1(s) = \frac{1}{1 + s/\omega_c}$$

The four real poles at $s = -\omega_c$ combine to provide a lowpass filter with cut-off frequency (-3 dB point) at $\omega = \omega_c$. The overall transfer function with feedback as shown is

$$H(s) \triangleq \frac{Y(s)}{X(s)} = \frac{G_1^4(s)}{1 + kG_1^4(s)} = \frac{1}{k + (1 + s/\omega_c)^4}$$

where g is the feedback gain which is varied between 0 and 4. Each real pole section can be implemented as a simple (buffered) RC section. Moog implemented the RC sections using a highly innovative discrete analog circuit known as the "Moog ladder" [Moog 1965, Hutchins 1975].

At $\omega = \omega_c$, the complex gain of each pole section is

$$G_1(j\omega_c) = \frac{1}{1+j} = \frac{1}{\sqrt{2}} e^{j\frac{\pi}{4}}$$

Therefore, the gain and phase of all four sections are

$$G_1^4(j\omega_c) = \frac{1}{4} e^{j\pi} = \frac{1}{4}(-1)$$

I.e., the total gain is 1/4 and the phase is -180 degrees (inverting). In contrast, at $\omega = 0$, the gain is 1 and the phase is 0 degrees (non-inverting), while at $\omega = \infty$, the gain is 0, and the phase is -360 degrees (also non-inverting). In summary, the four one-pole sections comprise a lowpass filter with cut-off frequency $\omega = \omega_c$, which is *inverting* at cut-off. Therefore, the use of inverting feedback provides *resonance* at the cut-off frequency. This is called "corner peaking" in analog synthesizer VCF design [Hutchins 1975, p. 5d(12)]. As the feedback gain k approaches 4, the total loop gain approaches 1, and the gain at resonance goes to infinity.

Figure 2 shows a family of frequency response functions for the Moog VCF for a variety of feedback levels. As the feedback gain g goes from 0 to 4, the poles of the overall filter expand outward in an "X" pattern from $s = \omega_c$ until the two poles on the right reach the $j\omega$ axis at $\omega = \omega_c$.

Lowpass Nature: Since the one-pole filters are $G_1(s) = \frac{\omega_c}{s+\omega_c}$, we get

$$H(j\omega) = \frac{\omega_c^4}{(j\omega + \omega_c)^4 + k\omega_c^4}$$

so at $\omega \ll \omega_c$, $|H(j\omega)| \approx \frac{1}{1+k}$, and at $\omega \gg \omega_c$, $|H(j\omega)| \approx \frac{1}{\omega^4}$.

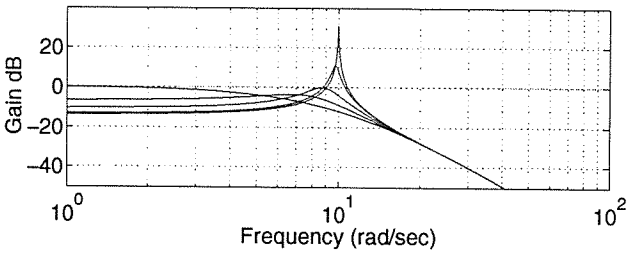


Figure 2: *Amplitude response of the analog Moog VCF for different levels of feedback ($\omega_c = 10 rad/sec$). At $k = 0$, the dc gain is 1 and the filter is a lowpass without corner peaking. Also shown are $k = 4[0.3, 0.6, 0.9, 0.99]$. As k increases, corner peaking develops at the cut-off frequency. At $k = 4$, the lowpass filter oscillates at its cut-off frequency.*

Root-Locus Interpretation

We can also analyze the VCF with the root-locus technique. Root Locus is a method popular in the field of control systems analysis that gives various rules for feedback-loop pole location movement in terms of the open-loop transfer function and the variations of the feedback gain. While originally intended for analysis of control systems, there is no reason why it cannot be used to analyze audio filters (indeed, linear control systems *are* filters, just dealing with different frequency ranges).

Introduction to Root Locus

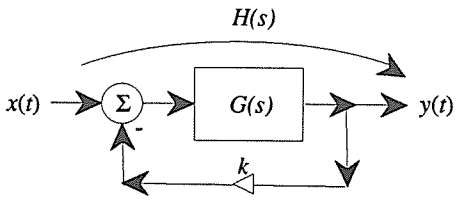


Figure 3: *A simple feedback system.*

Let's assume a system as shown in Figure 3, a simple feedback system with the transfer function $G(s)$ in the forward path. We know from block-diagram algebra that the total (closed-loop) transfer function is:

$$H(s) = \frac{G(s)}{1 + kG(s)}$$

Now, if $G(s) = \frac{N(s)}{D(s)}$, then:

$$H(s) = \frac{N(s)}{D(s) + kN(s)}$$

If $G(s)$ is in the feedback path, then:

$$H(s) = \frac{D(s)}{D(s) + kN(s)}$$

Note that in both cases the poles are the same:

$$D(s) + kN(s) = 0 \quad (1)$$

$$k = -\frac{D(s)}{N(s)} = -G^{-1}(s)$$

Since k is real and positive, we see that the total root-locus (the locus of all points in the s-plane that are roots of eq. 1 as k traverses $[0, \infty)$) is all s for which $\angle G(s) = \pi$.

Two rules for root locus are immediately clear from eq. 1: for $k = 0$, the roots of eq. 1 are the roots of $D(s)$ (the poles of $G(s)$); and for $k \to \infty$, the roots of eq. 1 are the roots of $N(s)$ (the zeros of $G(s)$). Thus as k traverses $[0, \infty)$, the closed-loop poles start at the open-loop poles and head towards the open-loop zeros.

The rules for root locus were developed to aid in hand-drawing the loci, and can be found in any introductory book on control systems (such as Franklin & Powell 1994)[1]. Although it is now trivial to use computers to calculate root-loci via brute-force numerical root finders, familiarity with the rules and the common root-locus shapes allows one to use root-locus as a design tool.

The MoogVCF Analyzed

The Moog VCF is in the simple feedback form of Figure 3. Therefore, we can immediately draw the root locus for changes in the feedback gain.

Root-Locus Rules state that the four coincident poles of the open-loop filters break away from the real axis at 45-degree angles and head to the zeroes at infinity along straight-line asymptotes, which in this case happen to be the same as the break-away lines. Thus, the root locus consists of these 45-degree lines that cross the imaginary axis at $\omega = \omega_c$, the open-loop pole location. Quick calculation also shows that the feedback gain at which this happens is $k = 4$. Thus one has a trivial corner-frequency control via the pole location of the cascaded one-pole filters.

[1]The same rules apply in discrete-time filters (root-locus in z) as in continuous-time (root-locs in s), the only difference being the pole-location interpretation.

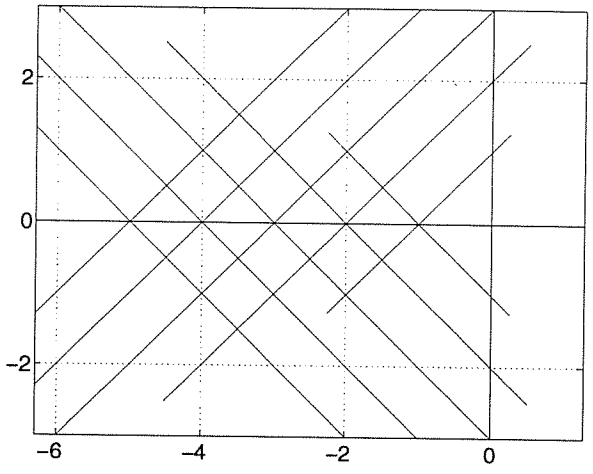


Figure 4: *Root-Loci for the Moog VCF, various ω_c, sweeping $k \in [0, 5]$.*

Resonance Control: One can also evaluate a root locus of the VCF with ω_c, the open-loop pole location as the free variable.[2]

Algebraic solution of $(s + \omega_c)^4 + k\omega_c^4 = 0$ gives

$$s = \omega_c \left(-1 \pm \sqrt{\pm j}\, k^{1/4}\right)$$
$$s = \omega_c \left(-1 \pm k^{1/4} e^{\pm j\pi/4}\right)$$

Which shows the 45-degree root-locus lines mentioned earlier. If we keep k constant, and look at the dominant poles (the ones that approach the $j\omega$ axis), we get:

$$s = ae^{\pm j\alpha}, \text{where}$$
$$\alpha = \frac{\pi}{2} + \tan^{-1}\left(\frac{\sqrt{2} - k^{\frac{1}{4}}}{k^{\frac{1}{4}}}\right)$$

So sweeping ω_c while keeping k constant gives root-locus lines that keep a constant angle from the $j\omega$ axis. This gives the filters a constant Q across sweeps in ω_c, so that k becomes a Q control. Thus the Moog VCF has simple, uncoupled controls of corner frequency and resonance.

[2]Unfortunately, most of the standard root-locus rules do not apply anymore. The root-locus rules work for any system that can be put into the form $A(s) + cB(s) = 0$, so that the coefficients of s in the expanded polynomial are at most *affine* in c (linear plus an offset, i.e. the highest power of c in the polynomial is 1.). In the VCF, the above equation is 4'th-order in ω_c. Thus the root-locus can only be evaluated numerically, or in simple situations, solved algebraically, since there are no simplified rules for the patterns in this "higher-order" root locus (*yet*— work is being done to this end).

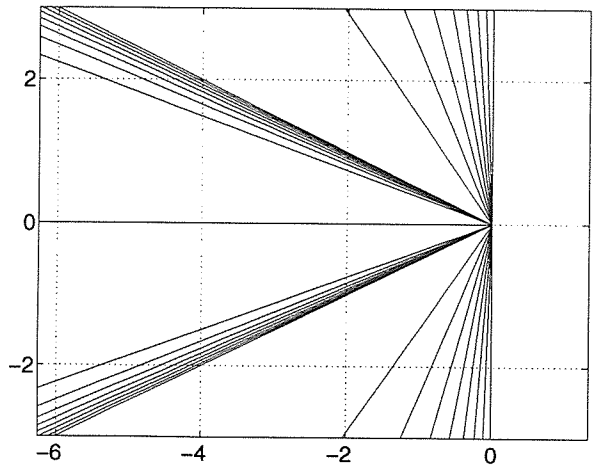


Figure 5: *Root-Locus vs. ω_c, various k.*

2.0.1 Definition of Q

For this paper, Q will be defined in terms of *pole location* rather than the center/bandwidth definition. In particular, it will be defined according to the impulse response of the poles (assuming dominance): "the number of cycles for the envelope of the impulse response to decay to $1/e^{\pi}$"(Morse, p. 25) . With dominance, pole location and impulse-response decay time are essentially equivalent. When we define Q in terms of the impulse response, we can arrive at a discrete-time definition of Q via the impulse-invariant transform $z = e^{sT}$, which gives us constant-Q pole locations as $z = r\ e^{\pm j\theta}, r = e^{-\alpha\theta}$, which are logarithmic spirals in the z-plane, and $Q = 1/(2\sin(\tan^{-1}(\alpha)))$ (thus the Q of the pole z_p is $Q(z_p) = [2\sin(\tan^{-1}(-\ln|z_p|/\angle z_p))]$).

3 Discretizing the VCF

It is desired to create digital filters with frequency and resonance controls as simple and efficient as those in the analog VCF. In particular, we desire filters whose controls (1) are uncoupled, (2) control useful parameters, such as frequency and Q, and (3) are efficient to control (not requiring expensive conversions, such as transcendentals, to get from the desired parameter to the actual control value). Filters based on the Moog VCF topology are explored here because it is hoped that at least some of the good features of the filter will translate well into the digital realm.

In order to preserve controllability, the continuous-time (CTS) VCF equations must be translated to discrete-time (DTS) using some transformation of the transfer function. as opposed to doing a impulse-response discretization, or a DTS filter design based on the CTS frequency response, because these methods typically aren't parameterizable, nor do they preserve any parameterization of the original system.

For similar reasons, the VCF's topology (cascaded one-pole filters with feedback around the whole loop) will be used for the DTS filters. This means that the filter equations to be transformed will be those of the one-pole filters rather than the equations of the whole system. This keeps the controls simple, because otherwise, when the equations for the VCF are determined (i.e. by multiplying out the cascaded filters and collapsing the feedback), the resulting coefficients are no longer simple functions of the controls (including higher powers of the controls and divisions involving the coefficients), which destroys the efficiency of the controls.

Some popular transforms are the Backward-Difference Transform, the Bilinear Transform, and the Pole/Zero Mapping. These techniques accomplish the transform by applying some mapping to convert from the CTS variable s to the DTS variable z. The backward-difference mapping is $s \leftarrow (1 + z^{-1})/T = (z - 1)/(z\ T)$, ($T$ is the sampling period); the bilinear transform is $s \leftarrow 2(z-1)/(z+1)T$; and the pole/zero mapping is $z \leftarrow e^{sT}$ for the poles[3][Franklin & Powell 1990, Ch. 4].

The pole/zero mapping can't be used as a direct substitution for s into transfer functions, because the resulting equation in z is non-rational and thus not implementable. It can be used, however, to guide the design of an equivalent DTS filter, with the poles and zeros in the positions described by the $z \leftarrow e^{sT}$ mapping. This can, still, increase the complexity of a design (and decrease the efficiency) because complex exponentials (or at least transcendentals) can easily crop up in the control equations of the new system.

Implementability:

An unfortunate fact in the discretization of the VCF topology is that most of the above-mentioned transforms will produce one-pole filters that have a delay-free path from input to output. That means when they are placed in the feedback loop, the

[3]And the finite zeros, with all but one of the zeros at infinity being placed at $z = -1$.

system is unrealizable[4]. Thus the systems must be modified to make them realizable, typically by adding a unit delay into the loop. Unfortunately, this addition interferes with many of the features of the filters, including, most notably, causing the controls to no longer be uncoupled.

Therefore, a major part of the design process is finding transforms (or directly designing DTS systems in the Moog VCF topology) that minimize the distortions required in the realizations.

Bilinear Transform

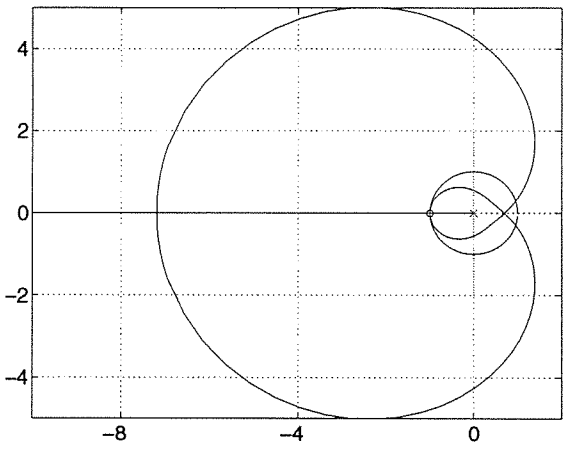


Figure 6: *The Complete Root Locus for a given p, bilinearly-transformed VCF.*

The bilinearly transformed onepole $\frac{s}{s+a}$ is:

$$G_1(z) = \frac{1}{2}(p+1)\frac{z+1}{z+p} \qquad \texttt{(Bilinear)}$$

where

$$p = \frac{a-2}{a+2}$$

so that

$$G(z) = \left(\frac{0.5\ (p+1)(z+1)}{(z+p)}\right)^4$$

and

$$H(z) = \frac{G(z)}{1+kz^{-1}G(z)}$$

$G(z)$ has a delay-free path, so to implement this, a unit delay has been added to the loop. This kills the uncoupled nature of p and k for frequency and resonance control, see Figure 7 (If the controls were uncoupled, the curves would be horizontal).

[4]Unless the feedback loop is collapsed with block-diagram algebra, but as mentioned earlier, this destroys the efficiency of the control.

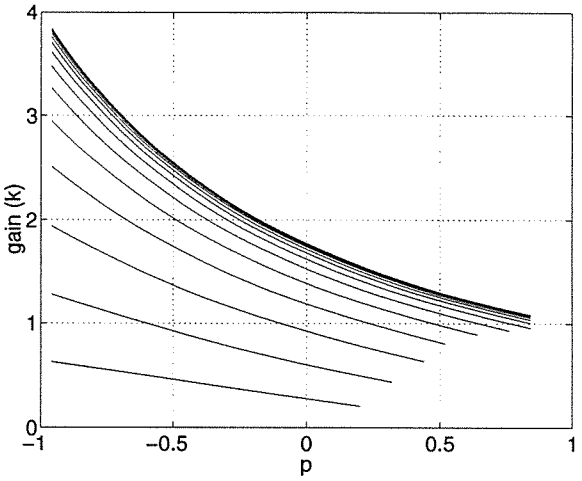


Figure 7: *Feedback gains vs. p to get various Q,* `Bilinear` *case.*

We can see from the Figure 7 that k must be kept below 1.0 if one wants to sweep the whole range of p while keeping k constant yet stay stable at the high frequencies. Unfortunately, this causes the Q at low frequencies to be quite low. The current fix for the coupled controls is to use a "separation" table to scale the feedback gain as function of the pole location p:

$$k_{\text{actual}} = k_{\text{desired}}\text{Table}(p)$$

Where Table(p) is given by the top trace of Figure 7, and $k_{\text{desired}} \in [0, 1)$.

The lower traces in Figure 7 are not simple scalings of the top trace, but the are rather close. This causes the Q along a given sweep of p to rise at very high frequencies, making this not exactly a constant-Q sweepable filter (see Figure 16). This inaccuracy is considered tolerable because it only becomes major for corner frequencies in the top octave, which are typically unused at $f_s = 44.1$ kHz, and if used, typically only for special effects, where total accuracy is not completely necessary.

To get exact constant-Q sweeps, table(p) would also have to be a function of Q, which vastly increases the storage requirements for the table.

Another table lookup must also me done if exact tuning is deemed necessary (Figure 8). Note that at low frequencies, the tuning curve is almost linear, so may be unnecessary. The use of tuning tables

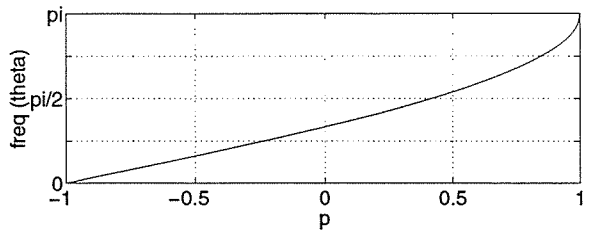


Figure 8: *Tuning curve at infinite Q*,`Bilinear`.

is often less of a problem for efficiency because in many cases exact tuning is needed at slower rates, such as only at the beginning and end of sweeps.

PZMap Onepole Placement

This case acts almost the same as the bilinear case, but it achieves realizability by removing one of the zeros at $z = -1$ instead of adding the unit delay (which puts a pole at $z = 0$). Otherwise it acts similarly, so will not be considered further.

Backward Difference Transform

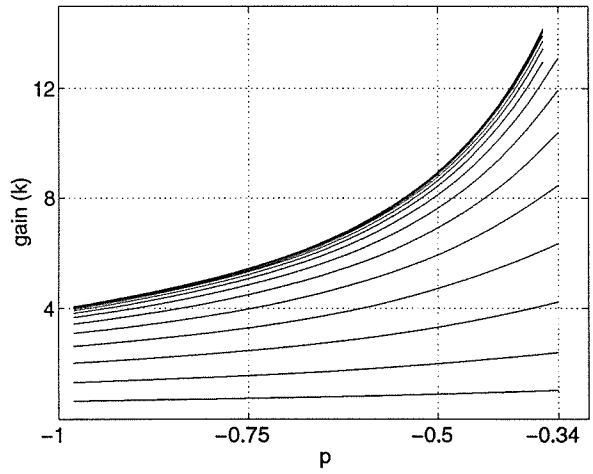


Figure 9: *Feedback gains for various Q*, `Back-Diff`.

The backward-difference transformed onepole $\frac{s}{s+a}$ is:

$$G_1(z) = (p+1)\frac{z}{z+p} \qquad \texttt{(Back-Diff)}$$

where

$$p = a + 1$$

so that

$$G(z) = \left(\frac{(p+1)\,z}{(z+p)}\right)^4$$

Again, this requires an extra delay in the loop to become implementable, And, as in the bilinear case, a table is required for separability. This filter, however, can be used without a separation table with better results than in the bilinear case because, as we can see from Figure 9, the Q *falls* as p increases, for a given k. This allows the user to sweep p without worrying about stability as long as the Q at low frequencies is desirable. For many effects where exact Q isn't necessary, the variation in Q vs. p that this filter presents (Figure 17) may be acceptable.

If more closely constant Q is required, then the techniques described for the bilinear case (the use of a separation table) apply with similar results, although this filter may be able to be implemented slightly more efficiently because of numerator of the onepole is simpler (typically this affects the total system efficiency only slightly, since the bilinear case can be implemented very efficiently.)

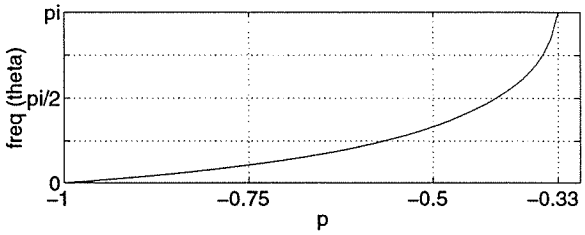


Figure 10: *Tuning curve at infinite Q*, `Back-Diff`.

The tuning curve is more drastic (Figure 10) than in the bilinear case, which makes the use of a tuning table more necessary in practice.

'Compromise' Version

The reader may have noticed that the gain curves for the two preceding cases have nearly opposite behaviors (for the bilinear, Q goes up with p when not using a separation table, and for the backwards difference, it goes down). The big difference between these two architecturally is the placement of the one-pole filters' zeros: the bilinear case places them at $z = -1$, and the backwards difference places them at $z = 0$. This suggests finding an intermediate position that may flatten out the gain curves and give p-sweeps more close to constant-Q.

A few eyeballed tries gave a zero position of $z =$

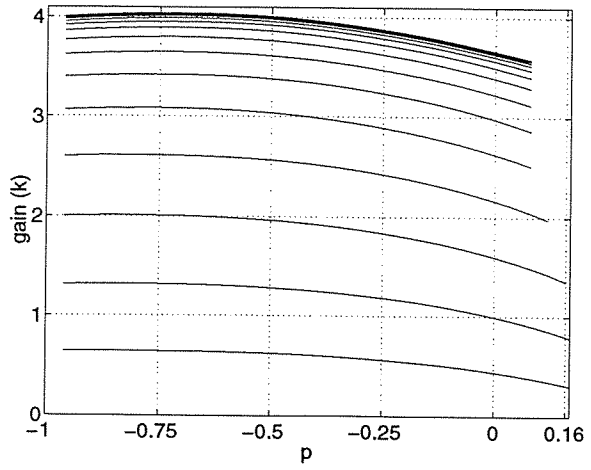


Figure 11: *Feedback gains for various Q,* `Compromise`.

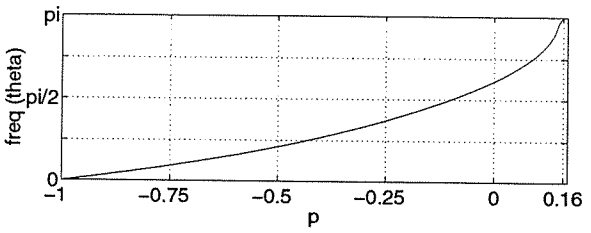


Figure 12: *Tuning curve at infinite Q,* `Compromise`.

-0.3:

$$G_1(z) = \frac{(p+1)}{1.3}\frac{z+0.3}{z+p} \qquad \texttt{(Compromise)}$$

$$G(z) = \left(\frac{(p+1)(z+0.3)}{1.3\,(z+p)}\right)^4$$

Again, a delay is put in the loop.

Referring to Figure 11 and Figure 18, we see that in the frequency range $[0, f_s/4]$ and for Q up to about 100, the filter is quite close to constant-Q and the controls p and k are almost uncoupled controls of frequency and Q, *without the use of a separation table*, although a tuning table may be necessary, as in all these cases.

An optimization could be performed to arrive at the "best" zero location, maybe even optimizing the four zeros to different locations.

Thoughts on Exact Constant-Q: It is likely that the auditory system is not *extremely* sensitive to variations in Q (i.e. the JND is probably large). Unfortunately I don't have any references on the subject other than a mention of a study on speech formant-width sensitivty [Smith86, p. 130]. If true, and if a number for JND (such as percentage) were found, then it would tell us how close to constant-Q we need to get in filters that don't exactly follow constant-Q sweeps, such as all the ones mentioned above. It would also help in the design of stopping conditions for optimization procedures that may be used to design or tweak these kinds of filters. It is likely that there is quite a bit of leeway in the variation of Q with corner frequency that we can tolerate.

On the other hand, JNDs for amplitude are quite small, and since messing with Q usually messes with amplitude (or loudness), this might place a tighter condition on Q. What we really need is a JND for resonance amplitude variation across corner frequency sweeps.

Comparisons

Root Loci: The Root Loci for the above-mentioned filters (Figures 13-15) are quite informative. These plots show dominant-pole locations versus sweeps of p and k (in the bilinear case, k is scaled with the separation table). Constant-Q pole locations are shown on the z-plane grid, so we can see how the filters deviate from constant-Q (at least at high frequencies). The loci also show how the tuning acts versus p.

Note that in all cases, $k = 0$ gives the positive real axis. Also note how the use of a separation table (Figure 13) guarantees stability, at the expense of the extra table lookup.

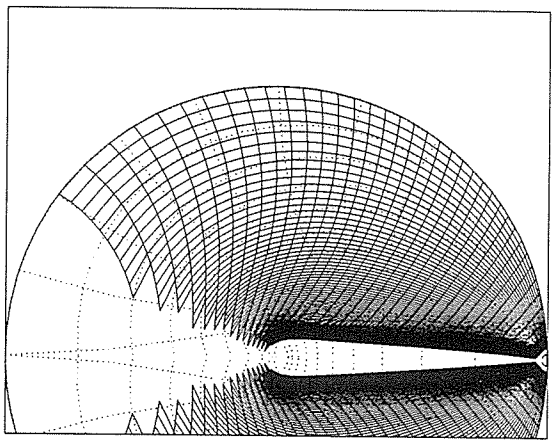


Figure 13: *Dominant pole locations for p and k sweeps,* `Bilinear` *with separation table.*

Constancy of Q: These plots (Figures 16-18) show the frequency ranges and Q ranges over which

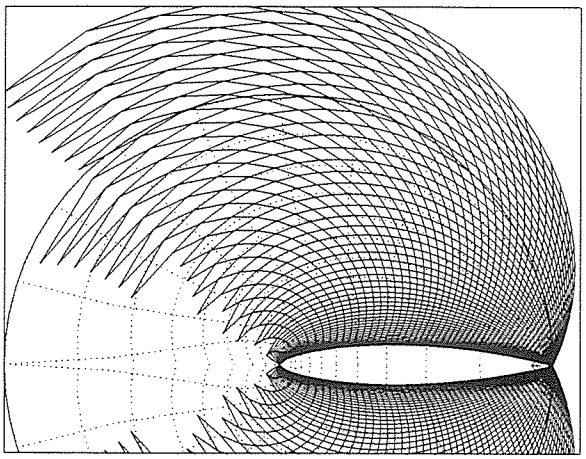


Figure 14: *Dominant pole locations for p and k sweeps,* `Back-Diff`*, no separation table.*

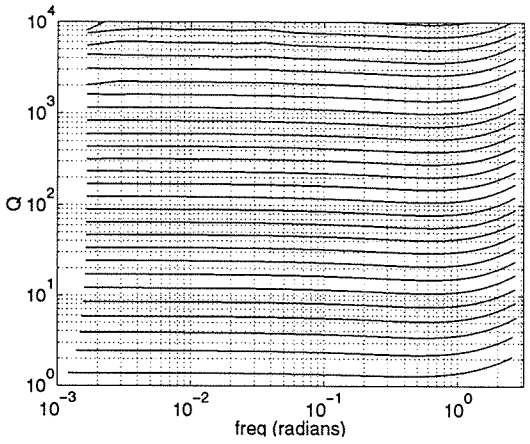


Figure 16: *Q vs. corner freq. for various (pre-scaling) k,* `Bilinear` *with separation table.*

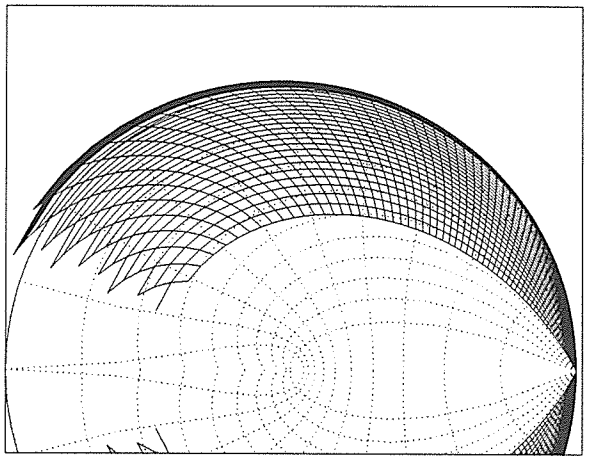


Figure 15: *Dominant pole locations for p and k sweeps,* `Compromise`*, no separation table.*

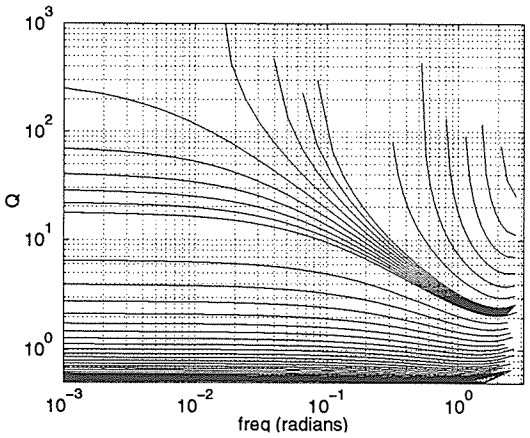


Figure 17: *Q vs. corner freq. for various k,* `Back-Diff`*, no separation table.*

the filters approximate constant-Q (again, this is based on the location of the dominant poles). These show the Q as p varies, with k held constant at various values (except in the bilinear case, where the separation table is used[5]). This type of plot is one of the more useful pieces of information when designing VCFs that are intended to be constant-Q.

Bode Plots: A complete set of Bode plots would take up much too much space, so instead a single p sweep is shown for each filter, with k held constant (with separation table in `Bilinear` case) at a value that gives a medium Q (Figures 19-21).

Oversampling: Another way to approach constant-Q is to oversample. Almost all filters of this type can be tweaked to act very well over a small frequency range. Oversampling reduces the range of desired frequencies significantly, thus making the VCF design problem easier. This is also understandable from the viewpoint that highly oversampled systems are better approximations of continuous-time systems, because the region about $z = 1$ can be linearized down to a rectangular coordinate system (just like the CTS coordinates) by the approximations

[5]Again, separation tables would also work in the other cases to get better curves, but the intention is to find filters for which the use of a separation table is unnecessary. It is necessary in the bilinear case for stability reasons.

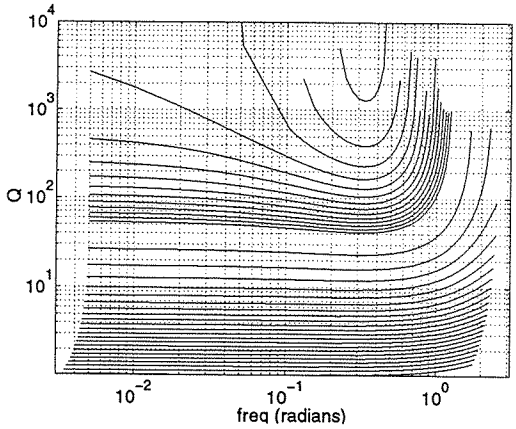


Figure 18: *Q vs. corner freq. for various k,* `Compromise`*, no separation table.*

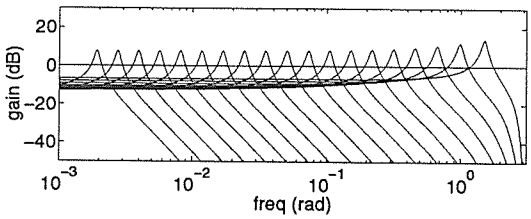


Figure 19: *Bode plots, constant pre-scaling k (medium Q), various p,* `Bilinear` *with separation table.*

$$r\sin(\theta) \xrightarrow{\theta\to 0} r\ \theta$$
$$r\cos(\theta) \xrightarrow{\theta\to 0} r$$

furthermore, for $r \approx 1$,

$$r\sin(\theta) \xrightarrow{\theta\to 0} \theta$$
$$r\cos(\theta) \xrightarrow{\theta\to 0} r$$

On the other hand, oversampling is less efficient. For an oversampling factor of M, the oversampled filter is M times more expensive. This may be useful, however, if because of the oversampling, the cost of the filter can be significantly reduced (cf. the above argument that the design is simpler). Oversampling can also aggravate certain numerical errors, such as coefficient roundoff, because all the poles become bunched up around $z = 1$, which increases sensitivity to the coefficient errors (Franklin & Powell 1990, p. 339)

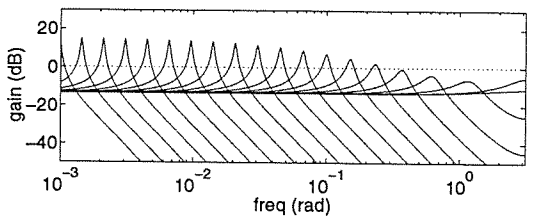


Figure 20: *Bode plots, constant k (medium Q), various p,* `Back-Diff`*, no separation table.*

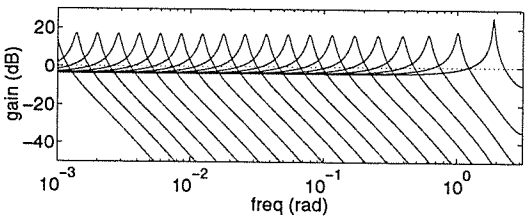


Figure 21: *Bode plots, constant k (medium Q), various p,* `Compromise`*, no separation table.*

Constant-Q Filters, Algebraic Derivation

If efficiency is not a problem, we can directly write the equations for the denominator of the desired filter:

2 poles:

$$(z - e^{-\alpha\theta}e^{-j\theta})(z - e^{-\alpha\theta}e^{j\theta})$$

4 poles:

$$(z - e^{-\alpha\theta}e^{-j\theta})(z - e^{-\alpha\theta}e^{j\theta})(z - a)(z - b)$$

In the four-pole case, the placement of the other two poles is a matter of design. If we compare with the root-locus of the bilinear case (Figure 6), which has the other two poles somewhere closer to the origin, it may be that some good choices for the other poles are: (1) both at $z = 0$, (2) same angle as the main poles, but with a reduced radius, (3) same angle, lower Q, or (4) at the same positions as the main poles (so they become repeated).

Multiplied out, the two-pole denominator is:

$$z^2 + 2e^{-\alpha\theta}\cos(\theta)z + e^{-2\alpha\theta}$$

As a first pass at making this efficient, we could use table lookups for the exponent and cosine, making for 2 table lookups (probably interpolated) and 2 multiplies for each new pole location.

A further efficiency increase comes in constant-rate frequency sweeps, where the update rate is constant. This satisfies the equation:

$$e^{-\alpha\,\theta(t)} = e^{-\alpha(a+bt)} = e^{-\alpha a}e^{-\alpha bt}$$

at $t = t_0 + \Delta t$,

$$e^{-\alpha\,\theta(t_0+\Delta t)} = e^{-\alpha a}e^{-\alpha b t_0}e^{-\alpha b\ \Delta t}$$

So that at the denominator is:

$$z^2 + e_1(t)\cos(\theta(t))z + e_2(t)$$

where

$$e_1(t+\Delta t) = e_1(t)\delta_{e1}$$
$$e_2(t+\Delta t) = e_2(t)\delta_{e2} \text{ or } e_1(t+\Delta t)^2$$

and

$$\delta_{e1} = e^{-\alpha b\ \Delta t}$$
$$\delta_{e2} = \delta_{e1}^2$$

δ_{e1} need only be evaluated at the beginning of the sweep, thus we get rid of one table lookup per Δt. This technique can be used to smooth low-rate θ updates. If necessary, this method can also be used on α sweeps (or both).

4 Root-Locus Filters

Other patterns that show up in root-locus analysis can also be used to create useful sweepable filters. In particular, we can directly look for patterns that are useful for digital filters, rather than finding useful continuous-time patterns and then transforming the filters to discrete-time. We can call filters designed this way "Discrete-Time Root-Locus Filters"[6].

A common pattern in root-loci is a circle surrounding a zero. Circles are particularly interesting from a discrete-time perspective because of the region of stability in z, which is also a circle.

The Two-Pole Constant-Bandwidth Root-Locus Filter

By placing an open-loop zero on the origin, and two poles on the positive real line (so, using the notation of Figure 3, $G(s) = z/(z-a)(z-b)$), we can get a root-locus (in k) that is a circle centered on the origin with a radius that is the geometric average of the pole locations. A particularly simple choice of pole locations is therefore to put them both on the desired radius: $G(s) = z/(z-r)^2$ (Figure 22).

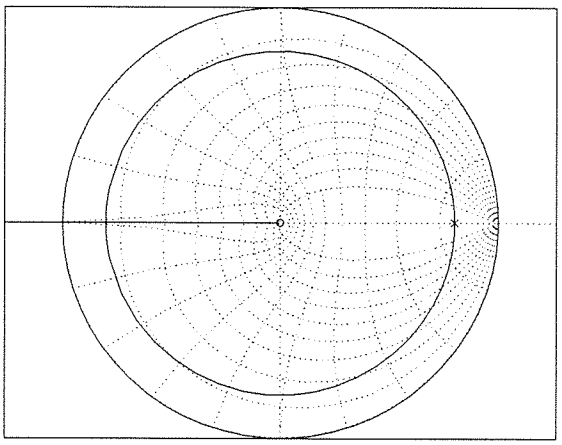


Figure 22: *Root locus of this 2-pole RL Filter,* $r = 0.8$.

We thus have two trivial controls: (1) k controls pole angle (corner frequency), and (2) the open-loop pole location controls the pole radius. Because the root locus is a perfect circle (this can be easily shown), the radius is constant over all frequencies (to the limits of the number system), so stability is not a problem. The controls are also completely uncoupled. frequency is related to k as $k = 2r(1-\cos(\theta))$, and radius as $r =$ (pole location).

This filter is not necessarily any more efficient than a direct-form filter (denom $= z^2 + 2rcz + r^2$), which also has uncoupled radius and angle controls with θ related to c as $c = \cos(\theta)$ — essentially the same control complexity. It may, however, have different numerical properties.

Root-Locus Filters Approximating Constant-Q

It is commonly held that constant-bandwidth filters are less useful than constant-Q filters. We can therefore modify the above root-locus filter to try to approximate constant-Q. The first pass is to note that at large Q, the constant-Q root trajectories look visually like circles, and shift the root-locus circle over to touch the unit circle at $z = 1$, like the constant-Q tracks do. This would give a pseudo-Q control with the open-loop zero location $(G(s) = (z-c)/(z-1)^2)$, with zero locations nearer $z = 0$ giving higher average Q (here the open-loop poles would be fixed at $z = 1$). Unfortunately, root-

[6] By analogy, the Moog VCF is a *Continuous-Time Root-Locus Filter*

locus rules state that the root-locus tracks must leave the real axis at $\pm 90°$,[7] so that at low frequencies, $Q \to \infty$, no matter where the zero is (see Figure 23).

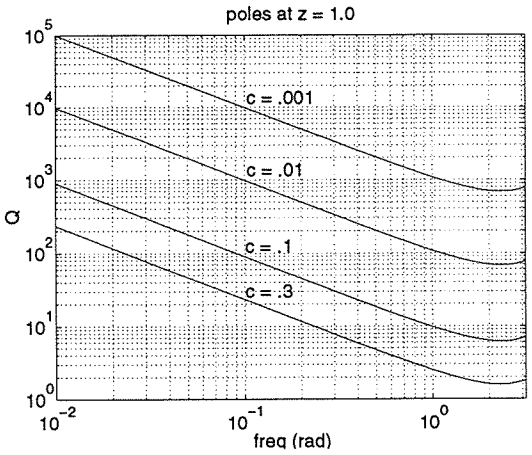


Figure 23: *Q vs. angle, various zero locations.*

The next modification would be to move the open-loop poles in from $z = 1$, so that Q doesn't go to ∞ at DC ($G(s) = (z-c)(z-(1-\epsilon))^2$). This causes Q to go to zero at DC, rise quickly at low frequencies, and then settle in to the same pattern as above at high frequencies (see Figure 24).

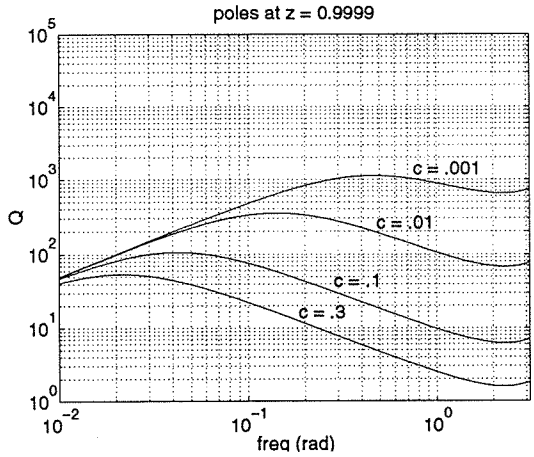


Figure 24: *Q vs. angle, various zero locations.*

Further flattening of Q can be achieved by adding pole/zero pairs inside the unit circle. This technique, well known in control-systems design, is used to locally warp the root locus. A pole/zero pair has a large effect on $\angle G(z)$ near the pair (remember, the root-locus is all z for which $\angle G(z) = \pi$), but away from the pair, they cancel each other and have little effect on the root locus. One can control the effects of the pair by controlling their separation and distance from the locus (close together $\Rightarrow$ more localized effect $\Rightarrow$ most be closer to locus, but has stronger effect because of proximity to locus; further apart $\Rightarrow$ more widespread, but weaker effect due to usually being placed further from the locus).

A quick design using this effect is shown in Figure 25, its root locus is shown in Figure 26. By adding pole/zero pairs and shifting the main open-loop pairs, one can follow an ad-hoc optimization path and minimize the deviation from some desired Q. The filter shown was designed by eyeballed trial-and-error[8], but an optimization procedure could be designed.

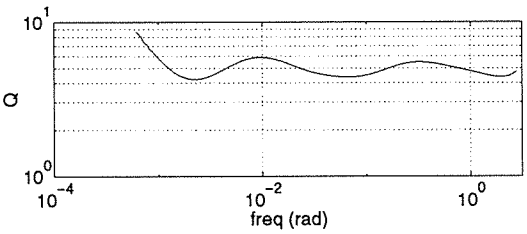


Figure 25: *Q vs. angle, eyeballed minimum-Q-error filter.*

Unfortunately, this technique doesn't easily lend itself to parameterizing Q, because a new optimization may need to be done for each Q (although a pattern could develop upon which a parameterization could be based). Also, it is likely not very efficient, due to the number of pole/zero pairs greatly increasing the order of the system.

5 Conclusion

Implementability issues make the conversion of the Moog VCF to a digital form nontrivial. Once converted using standard techniques, the filter must be tweaked to recover some of the original features. Some transforms preserve features better than others, but best results come from redesigning the fil-

[7] the actual constant-Q tracks leave $z = 1$ at angles greater than $\pm 90°$

[8] The pole/zero pairs were on the real axis to make things easier: poles at z = [.5 .9 .97 .9975 1 1], zeros at z = [.1 .55 .92 .975 .9983]. This particular $Q \approx 5$ is admittedly an easy design compared to a very high Q, but it serves as an example of the idea.

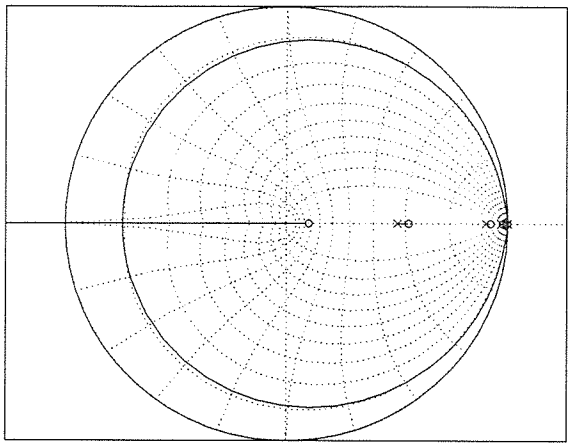


Figure 26: *root locus, eyeballed minimum-Q-error filter.*

ter directly in the discrete domain. Methods from control-systems theory prove useful in this redesign. These methods also suggest new topologies that prove interesting.

References

[Hutchins 1975] Hutchins, B. 1975. *Musical Engineer's Handbook.* Ithaca, New York: Electronotes.

[Moog 1965] Moog, R. A. 1965. "A Voltage-Controlled Low-Pass High-Pass Filter for Audio Signal Processing." *Audio Eng. Soc. Convention*, Preprint 413(Oct.).

[Zwicker 1990] Zwicker, E. 1990. *Psychoacoustics.* New York: Springer Verlag.

[Franklin & Powell 1990] Franklin, G., J. D. Powell, M. L. Workman, 1990 *Digital Control of Dynamic Systems, 2nd Edition* Reading: Addison Wesley

[Franklin & Powell 1994] Franklin, G., J. D. Powell, A. Emami-Naeini, 1994 *Feedback Control of Dynamic Systems, 3rd Edition* Reading: Addison Wesley

[Morse 1981] Morse, P. 1981 *Vibration and Sound* Acoustical Society of America.

[Smith 1983] Smith, J. O. III 1983 "Techniques for Digital Filter Design and System Identification With Applicatio to the Violin." Ph.D. Thesis, Stanford University, Report STAN-M-14

This paper can be found online at the web page: `http://www-ccrma.stanford.edu/~stilti/papers`

The 3D Tetrahedral Digital Waveguide Mesh with Musical Applications

Scott A. Van Duyne
savd@ccrma.stanford.edu

Julius O. Smith III
jos@ccrma.stanford.edu

Center for Computer Research in Music and Acoustics (CCRMA)
Dept. of Music, Stanford University, Stanford, CA

Abstract

The 2D rectilinear digital waveguide mesh algorithm to simulate wave propagation in the ideal membrane was introduced three years ago as a multiply-free, parallel computation scheme suitable for high speed hardware implementation. Since that time, various alternative structures and add-on elements have been developed to make the mesh musically useful. We review some of these developments and outline the new *tetrahedral* mesh structure which now permits efficient *multiply-free* simulation of wave propagation in 3D space.

1 Background

The fundamental intuition-building observation to make, in order to understand how the waveguide mesh algorithm works, is that when you kick a chicken wire fence, waves seem to propagate on it much as on an ideal membrane. The chicken wire fence, like the waveguide mesh, is a regular interconnection of short vibrating string elements joined at nearly lossless scattering junctions. In [9, 10] we showed that a *rectilinear* arrangement of 4-port junctions is mathematically equivalent to the standard finite difference equation approximation to the lossless 2D wave equation. In fact, there are a variety of regular geometric mesh structures which compute valid difference approximations to the wave equation, for example, the hexagonal 3-port structure shown in Figure 1.

The 2D digital waveguide mesh has proven to be effective in the modeling of musical membranes and plates, particularly in combination with recent simplifications in modeling stiffness [8], nonlinearities [7], and felt mallet excitations [7]. One of the more interesting musical applications is the waveguide mesh gong model, a section of which is illustrated in Figure 2: The J's mark ordinary 4-port lossless scattering junctions, in which the four inputs are summed and scaled by one half to form the so-called *junction velocity*, and the four outputs are computed by subtracting the respective inputs from this junction velocity. The PNF's mark *passive nonlinear filters* attached at the upper rim. The passive nonlinear filter is, in its simplest form, an first order allpass filter whose coefficient is varied between two values depending on the sign of the filter state value [7]. This fil-

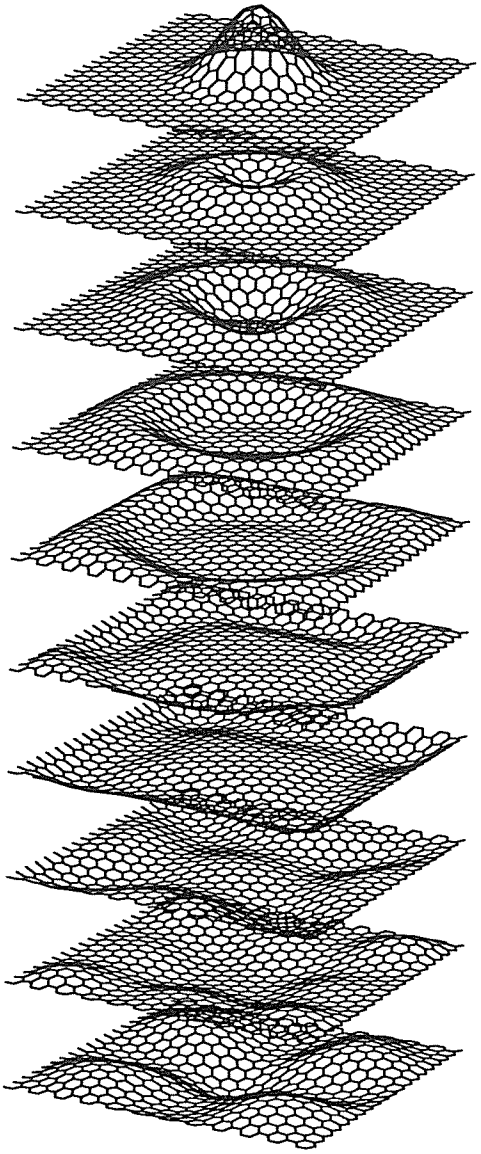


Figure 1: *Chicken Wire Mesh Structure*

ICMC 96

ter structure was developed in collaboration with John R. Pierce, to model the passive, nearly lossless, spreading of energy between modes of vibration in certain important classes of musical instruments. These allpass structures may also be used to simulate a "stretching" of the modal frequencies due to stiffness [8]. In the center of Figure 2, S marks a special time varying scattering junction which is attached to a *wave digital hammer* mallet model through the signals marked v_h^+ and v_h^-. The wave digital hammer simulates the nonlinear compression forces and hysteresis in the soft mallet or piano hammer [7].

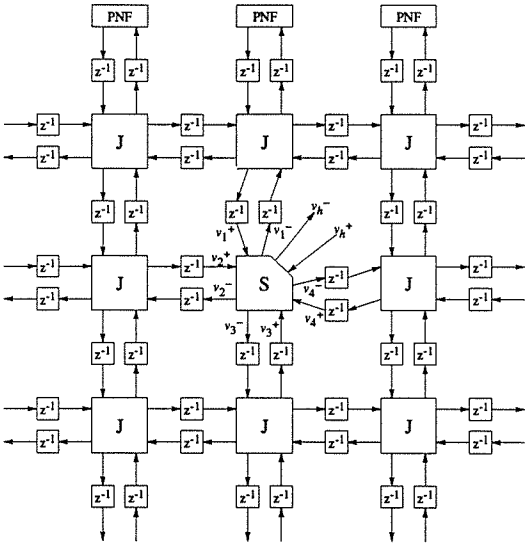


Figure 2: *The Waveguide Gong*

Davide Rocchesso and Federico Fontana have proposed a new percussion instrument based on an efficient 8-port triangular mesh structure with a wave digital air-loading filter and wave digital mallet port *at each junction* [1].

The 3D 6-port rectilinear extension to the mesh had been hypothesized [10], and was applied to the study of room acoustics by Savioja, Rinne, and Takala [2]. Tim Stilson first implemented the 3D rectilinear mesh to study wave propagation in a bent tube [4]. Figure 3 shows several frames from Stilson's animation of a pressure pulse plane wave in an acoustic tube trying to make its way around a U-turn. The computation was actually performed using a dense 3D mesh, and then the results were consolidated into a 2D image representation. Notice that as the *white* pressure pulse rounds the corner, a *black* negative pressure wake develops as some of the pulse reflects and inverts off the turning wall of the tube. This is followed by some significant coupling of energy into the cross-sectional modes of the tube as the pulse continues around out of the turn.

The simulations by both Stilson and Savioja's team used a rectilinear 3D mesh computational structure. However, such a structure requires the

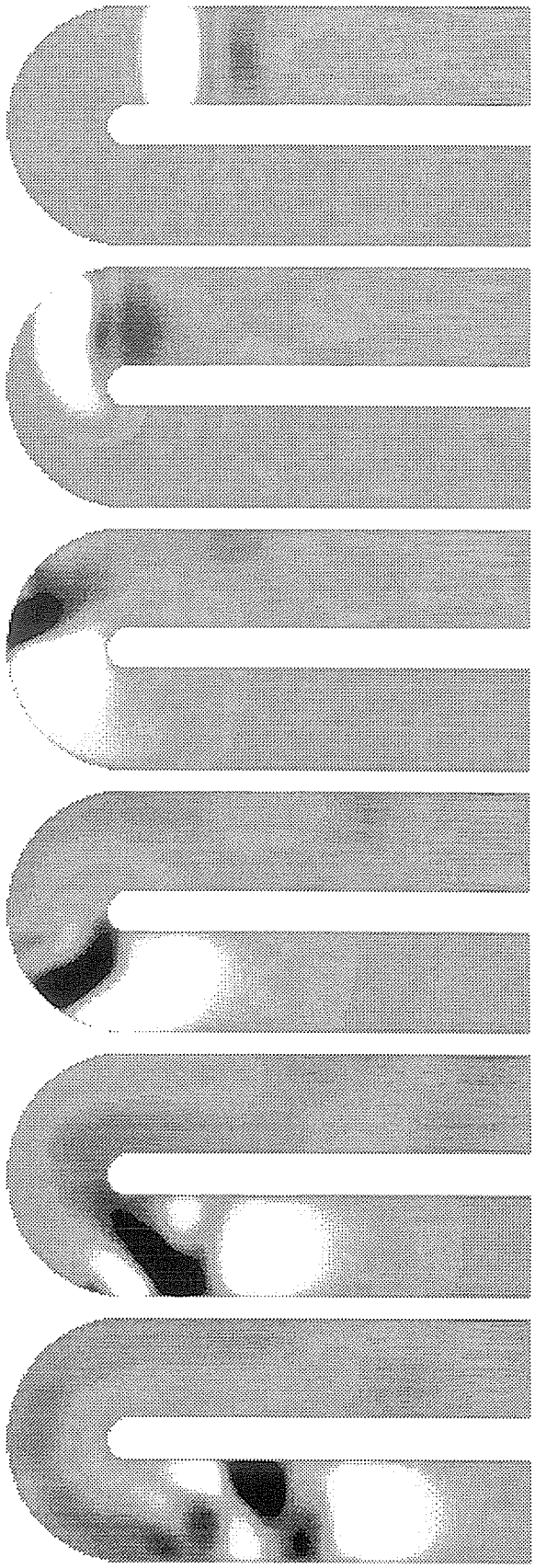


Figure 3: *Stilson's Animation of a Pressure Wave in a Tube Using a Dense 3D Rectilinear Mesh*

use of *6-port* scattering junctions, which make a multiply-free implementation impossible in the isotropic case. Implementation of multiplies in high speed hardware can get expensive, and reduces the practicality of a densely sampled parallel mesh implementation useful for room acoustics or accurate physical simulations. The *4-port* scattering junctions of the 2D mesh required only an internal divide by 2, which could be implemented as an inexpensive right shift in binary arithmetic. However, the 6-port junction requires a divide by 3. The multiply-free cases occur for N-port junctions in which N is a power of two [3].

We describe here a *tetrahedral* distribution of multiply-free 4-port scattering junctions filling 3-space much like the molecular structure of the diamond crystal, where the placement of the scattering junctions corresponds to the placement of the carbon nuclei, and the bi-directional delay units correspond to the four tetrahedrally spaced single bonds between each pair of nuclei. Figure 4 illustrates the structure. The tetrahedral mesh is mathematically equivalent to a finite difference approximation to the 3D lossless wave equation. The frequency- and direction-dependent plane wave propagation speed dispersion error is comparable with that of the rectilinear mesh structure; however, computational and memory requirements are much improved in the tetrahedral structure, and now within the realm of practical high speed hardware implementation. The authors are grateful to Prof. Wen-Yu Su of Chung-Hua Polytechnic Institute of Taiwan for fruitful discussions on the structure and implementation of the tetrahedral mesh [11].

2 What is Dispersion Error?

The term *dispersion* is somewhat overloaded, but in the field of finite difference approximations it refers to an error in the speed of travel of waves. For example, in the solution to ideal membrane equation, waves of all frequencies travel at the same speed in all directions. However, in the standard difference approximation, and, therefore, in the equivalent rectilinear 2D waveguide mesh, plane waves travel at slightly different speeds depending on their frequency and on their direction of travel relative to the orientation of the mesh. In fact, it was shown [10] that all waves traveling in the diagonal directions travel at the same correct speed, but that waves traveling in the directions of the grid axes travel a little slower at higher frequencies. Figure 5 shows a circular wavefront expanding on a 4-port rectilinear mesh. Observe how the wavefront has remained sharp along the diagonal directions, whereas it has smoothed out

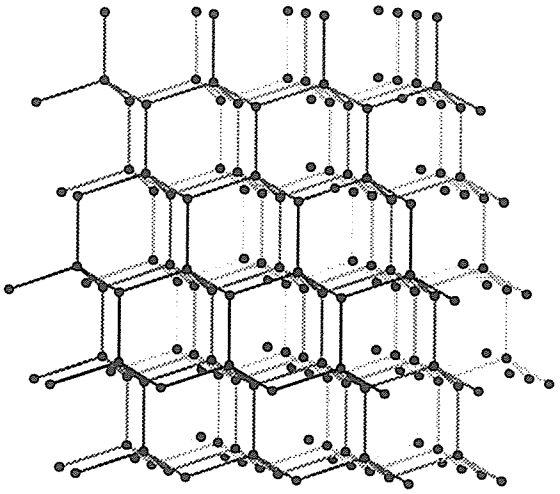


Figure 4: *The Tetrahedral Mesh Structure*

along the axes directions, resulting from the higher spatial frequencies lagging behind.

Figures 6 and 7 illustrate a frequency domain view of wave speed dispersion in various two-dimensional mesh structures. These plots were calculated using a method which will be described in more detail for the 3D tetrahedral mesh case in the succeeding sections. The upper right plot in Figure 6 shows contours of the normalized wave travel speed on the 4-port mesh versus plane wave frequency and direction. The center region of the plot corresponds to low plane wave frequencies; the outer regions of the plot correspond to higher plane wave frequencies. The angular position on the plot, as seen from the frequency plane origin (at the center), corresponds to the direction of plane wave travel on the mesh. Notice that in the diagonal directions of the 4-port mesh, all frequencies travel at full speed, whereas the contour lines show that wave travel speed falls off along the *axes* directions as frequency increases, i.e., nearer the outer edges of the plot. The contour lines are marked off in 1% intervals at 99%, 98%, etc., of full speed. (The dark circles indicate the maximum useful plane wave frequency and will be explained in a subsequent section.) In the lower right of Figure 6 a contour plot of wave speed dispersion in the multiply-free 8-port rectilinear mesh structure is compared. In the 8-port case, things improved over the 4-port case in the axes directions, but got much worse in the diagonal directions.

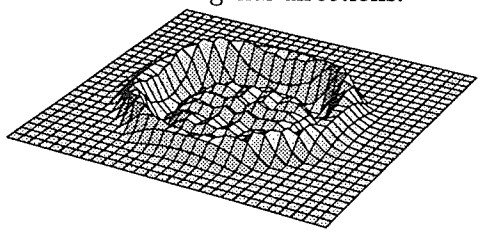


Figure 5: *Time Domain View of the Effects of Dispersion Error in the 4-port Rectilinear Mesh*

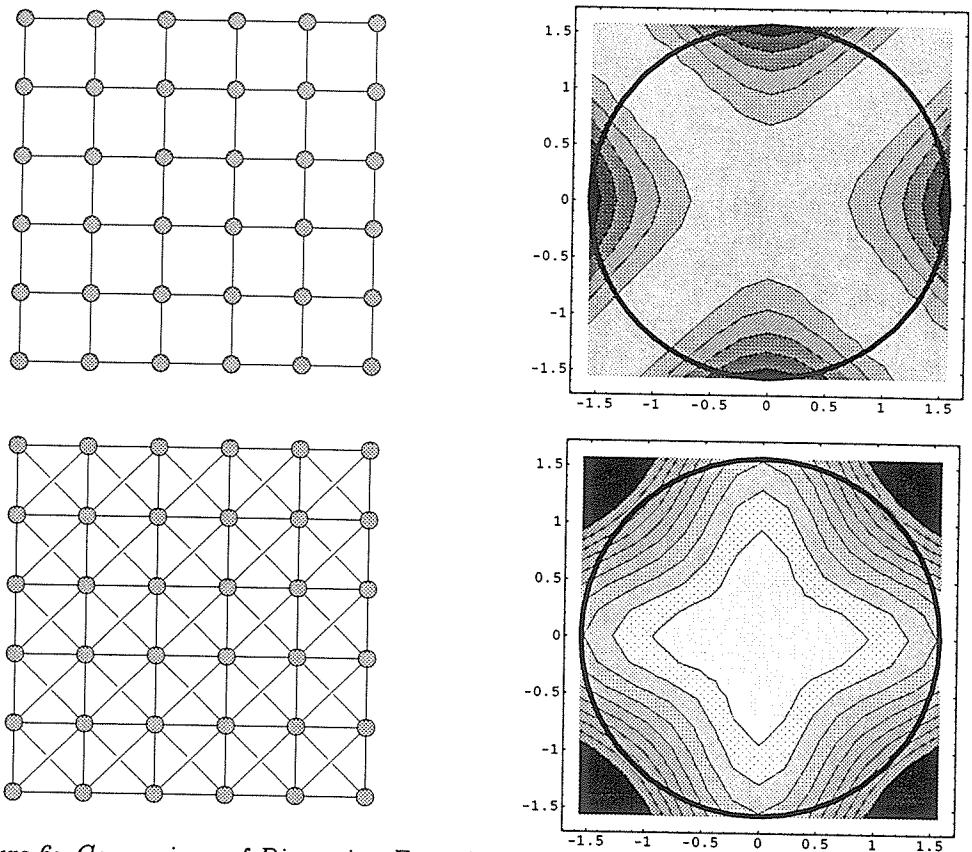


Figure 6: *Comparison of Dispersion Error in the Multiply-free 4-port and 8-port Meshes*

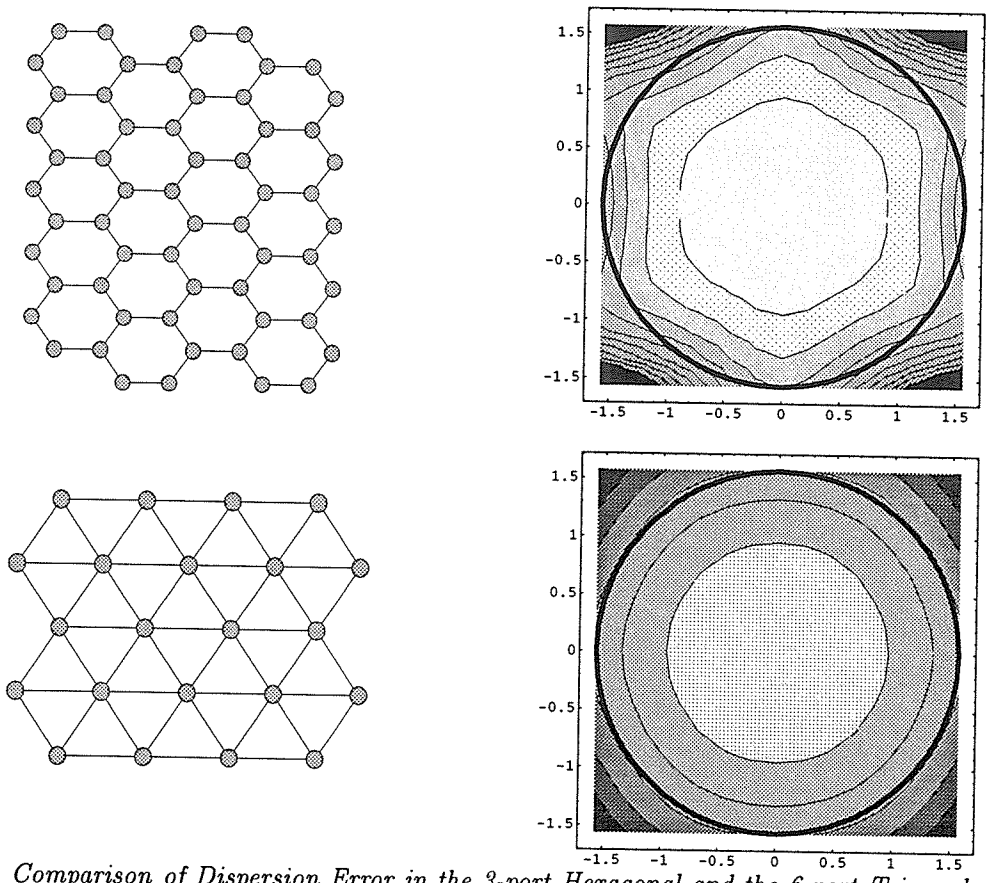


Figure 7: *Comparison of Dispersion Error in the 3-port Hexagonal and the 6-port Triangular Meshes*

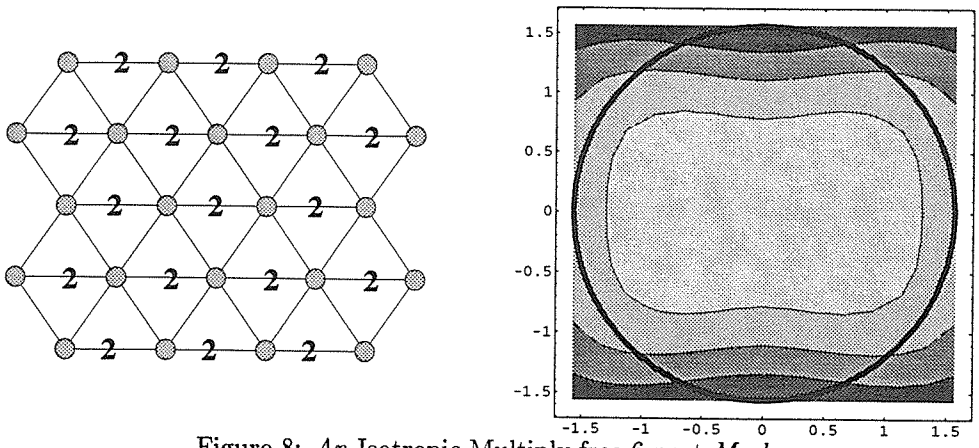


Figure 8: *An* Isotropic Multiply-free 6-port *Mesh*

Figure 7 compares the dispersion properties of the 3-port hexagonal and the 6-port triangular mesh structure. These mesh structures are not multiply-free; however, the triangular mesh seems to exhibit an optimal direction-independent dispersion. This is the underlying mesh used by Fontana and Rocchesso mentioned above [1]. Figure 8 shows a tricky variation to the 6-port triangular structure to achieve a multiply-free junction computation (although it requires one extra binary right shift). Here, two of the six waveguide segments connected at each junction are *twice as thick as*, i.e., twice the wave impedance of, the other four segments. This results in faster wave travel in one direction (much greater than that caused by dispersion error), but may be compensated for by resampling the spatial grid compressing the x-direction by a factor of $\sqrt{5/3}$. There is a similar trick making the 3D rectilinear 6-port mesh multiply-free, though it is rather heavy-handed in light of the tetrahedral alternative.

In a bounded mesh, wave speed dispersion results in a slight mistuning of the higher resonant modes. This mistuning can be adjusted by allpass filtering [8] and/or warping of the membrane boundary in a compensating manner. We note that the high frequency modes of a membrane become so dense that, in musical contexts, this error may not be psychoacoustically important. If higher accuracy is required, then an accordingly higher sampling rate may be used.

3 The Tetrahedral Difference

Rectilinear meshes compute finite difference approximation of the lossless wave equation [2, 10]. It is less obvious in the tetrahedral case. Figure 9 shows a small chunk of the tetrahedral mesh. We take the distance between adjacent junctions to be 1, and the junction point

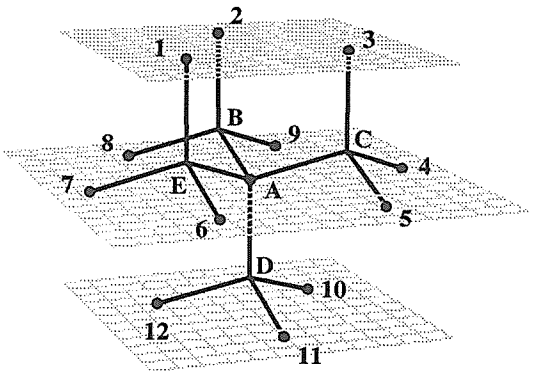


Figure 9: *Tetrahedral Structure Detail*

marked A to lie at the origin of an (x, y, z) cartesian coordinate system. We arrange the junctions $B(0, 2\sqrt{2}/3, 1/3)$, $C(\sqrt{2/3}, -\sqrt{2}/3, 1/3)$, $D(0, 0, -1)$, and $E(-\sqrt{2/3}, -\sqrt{2}/3, 1/3)$ tetrahedrally about point $A(0, 0, 0)$. The line segments between these junction points represent bi-directional delay units as shown in Figure 10.

The equations describing the computation of the lossless 4-port scattering junctions are [3, 9, 10],

$$V_A = \frac{1}{2}\sum_{\Gamma} V_A^{\Gamma+} \quad (1)$$

$$V_A^{\Gamma-} = V_A - V_A^{\Gamma+} \quad (2)$$

where Γ ranges over the four junction points surrounding A, namely $\Gamma \in \{B, C, D, E\}$. V_A represents the junction velocity at junction A. $V_A^{\Gamma+}$ and $V_A^{\Gamma-}$ represent the input and output signals, respectively, of junction A in the direction of junction Γ.

Since the junctions are interconnected with bi-directional delay units, the input to junction A

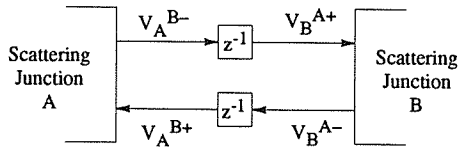


Figure 10: *Bi-Directional Delay Unit*

from the direction of Γ is equal to the output from Γ delayed by one sample. In the $\mathcal{Z}$-transform domain we my write this relationships as,

$$V_A^{\Gamma+} = z^{-1} V_\Gamma^{A-} \tag{3}$$

Using (2) and (3), we obtain an expression for the input signal to junction A from the Γ direction in terms of the junction velocities A and Γ only,

$$\begin{aligned} V_A^{\Gamma+} &= z^{-1} V_\Gamma^{A-} = z^{-1}\left(V_\Gamma - V_\Gamma^{A+}\right) && (4)\\ &= z^{-1}\left[V_\Gamma - z^{-1}\left(V_A - V_A^{\Gamma+}\right)\right] && (5) \end{aligned}$$

which implies,

$$V_A^{\Gamma+} = \left(\frac{z^{-1}}{1-z^{-2}}\right)\left(V_\Gamma - z^{-1} V_A\right) \tag{6}$$

We substitute (6) into (1) to get an expression for the junction velocity V_A in terms of the four surrounding junction velocities V_Γ,

$$V_A = \frac{1}{2}\left(\frac{z^{-1}}{1+z^{-2}}\right)\sum_\Gamma V_\Gamma \tag{7}$$

Unfortunately, the orientations of the tetrahedra vary from point to point. In Figure 9 the tetrahedron around point A and that around point B are in *vertically opposite* orientations. However, consider the relationship between the center point A and the twelve equally spaced junctions marked 1 through 12, which are all equidistant from A, and which are *two* time steps away from A. With some imagination, one can see that the directional relationships between point A and the outer twelve points repeats itself around *every* point in the mesh, regardless of orientation of the inner four points, B, C, D, and E.

Therefore, we take note of the following relationships, which may be derived in a manner similar to (7),

$$V_\Gamma = \frac{1}{2}\left(\frac{z^{-1}}{1+z^{-2}}\right)\left(V_A + \sum_{\gamma_\Gamma} V_{\gamma_\Gamma}\right) \tag{8}$$

where $\Gamma \in \{B,C,D,E\}$ and $\gamma_B \in \{2,8,9\}$, $\gamma_C \in \{3,4,5\}$, $\gamma_D \in \{10,11,12\}$ and $\gamma_E \in \{1,6,7\}$. Plugging (8) back into (7), we get an expression for V_A in terms of the junction velocities of the twelve junctions, V_i:

$$V_A = \frac{1}{4}\left(\frac{z^{-2}}{1+z^{-2}+z^{-4}}\right)\sum_{i=1}^{12} V_i \tag{9}$$

To see that this partial difference equation approximates the 3D wave equation, we first multiply through by the denominator in (9), inverse $\mathcal{Z}$-transform, and gather all the terms onto the left hand side. Then we view the equation as a continuous time and space expression of the form $\mathcal{F}(t,\underline{p}) = 0$, where $\mathcal{F}(t,\underline{p})$ is,

$$\sum_{k=0}^{2} v(t-2k\varepsilon,\underline{p}) - \frac{1}{4}\sum_{i=1}^{12} v(t-2\varepsilon,\underline{p}+\underline{P}_i\varepsilon) \tag{10}$$

and $\underline{p}$ is now the arbitrary spatial position of junction A, and the $\underline{P}_i$ represent the twelve directional vectors from point A to the junction points marked 1 through 12 in Figure 9, respectively. The unit time and space steps are defined as ε.

We may expand (10) in a four dimensional Taylor series about the point $\underline{p} = (0,0,0)$ at time $t = 0$, replacing each term of (10) with something of the form,

$$\sum_{n_t}^{\infty}\sum_{n_x}^{\infty}\sum_{n_y}^{\infty}\sum_{n_z}^{\infty} \frac{v_{\underline{0}}^{(n_t,n_x,n_y,n_z)} t^{n_t} x^{n_x} y^{n_y} z^{n_z}}{n_t! n_x! n_y! n_z!} \tag{11}$$

Collecting terms and computing the limit as the grid size shrinks reveals that

$$\lim_{\varepsilon\to 0}\frac{\mathcal{F}}{(2\varepsilon)^2} = u_{tt} - \frac{1}{3}\left[u_{xx} + u_{yy} + u_{zz}\right] \tag{12}$$

Evidently, the tetrahedral waveguide mesh is equivalent to an finite difference approximation of the continuous 3D wave equation. The apparent wave speed is $c = \sqrt{1/3}$, which is the numerically optimal speed in the Courant-Friedrichs-Lewy sense [5]. (Incidentally, we found it convenient to use the symbol manipulating feature of the mathematics processing language *Mathematica* to verify the algebra.)

To quantify dispersion error in the tetrahedral mesh, we apply a spectral transform analysis directly on the finite difference equation [5, 9]. Essentially, we transform the difference equation into the frequency domain in both time and space, replacing spatial shifts with their corresponding spatial linear phase terms. Then we observe how the spatial spectrum updates after one time sample. With this information, we can determine how fast the various plane waves travel in the mesh at each frequency. There can be no attenuation since the mesh is constructed from *lossless* scattering junctions. Therefore, the only departure from ideal behavior, aside from round-off error, is traveling-wave dispersion.

We may now take the spatial Fourier transform of (9) and replace the spatial positions of the twelve outer junction points with their corresponding linear phase terms, $V_i \longleftrightarrow V(\underline{\omega})e^{j\underline{P}_i^T\cdot\underline{\omega}}$, where $\underline{\omega}$ is the three-dimensional spatial frequency vector, to obtain the following quadratic expression in z^{-2}:

$$1 + bz^{-2} + z^{-4} = 0, \quad b \triangleq 1 - \frac{1}{4}\sum_{i=1}^{12} e^{j\underline{P}_i^T\cdot\underline{\omega}} \tag{13}$$

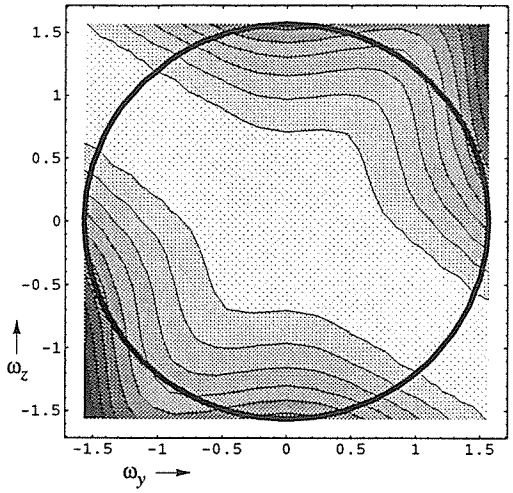


Figure 11: *Tetrahedral Dispersion:* $\omega_x = 0$

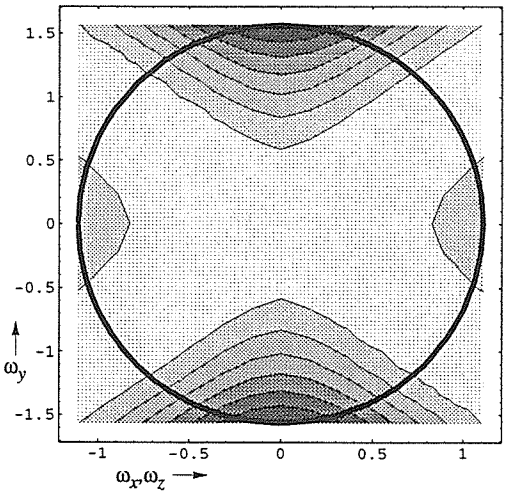


Figure 14: *Rectilinear Dispersion:* $\omega_z = \omega_x$

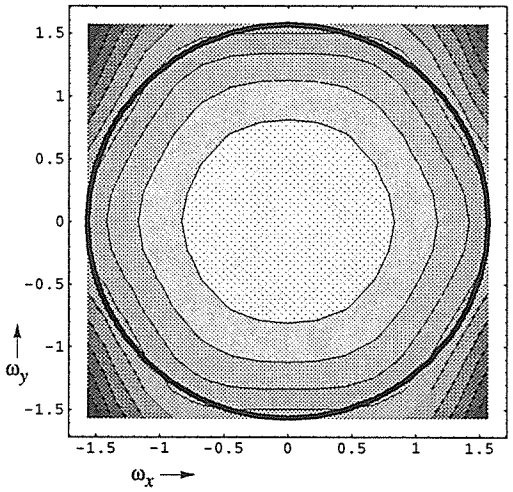


Figure 12: *Tetrahedral Dispersion:* $\omega_z = 0$

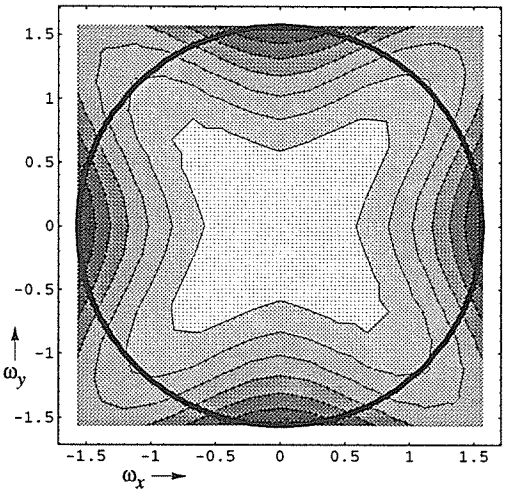


Figure 15: *Rectilinear Dispersion:* $\omega_z = 0$

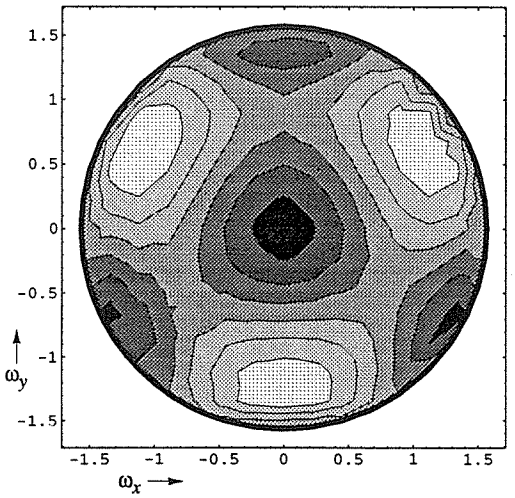


Figure 13: *Tetrahedral Dispersion:* $|\underline{\omega}| = \pi/2$

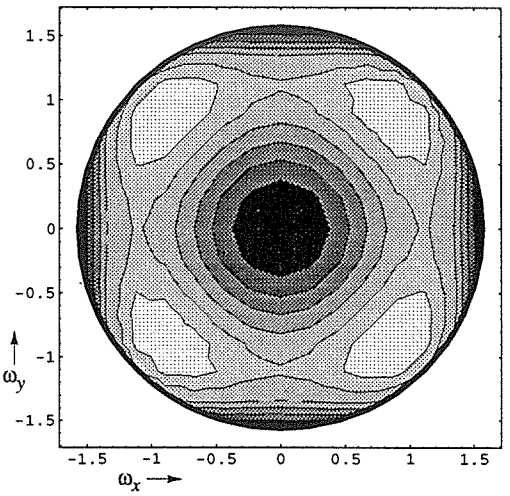


Figure 16: *Rectilinear Dispersion:* $|\underline{\omega}| = \pi/2$

where z^{-2} represents two time samples of delay. Due to the symmetrical orientation of vectors $\underline{P}_i$, as indicated in Figure 9, it may be shown, rather remarkably, that the value of b remains a real number between -2 and 2 for all values of $\underline{\omega}$. Hence, we may define

$$\mathcal{G}^2(\underline{\omega}) \triangleq -\frac{b}{2} \pm j\frac{\sqrt{4-b^2}}{2} \tag{14}$$

where $\mathcal{G}$ is the *spectral amplification factor* of the spatial spectrum after *one* time sample.

Plane waves propagate losslessly, since $|\mathcal{G}| \equiv 1$. We note that the phase of $\mathcal{G}$ corresponds to the spatial phase shift of a plane wave in the direction of travel in one time sample, where

$$\angle\mathcal{G} = \frac{1}{2}\arctan\frac{\pm\sqrt{4-b^2}}{b} \tag{15}$$

Hence, the *phase distance* traveled in one time sample by a spatial plane wave of frequency $|\underline{\omega}|$ and direction $\underline{\omega}$ is $c'(\underline{\omega}) = \angle\mathcal{G}/|\underline{\omega}|$, where $c'(\underline{\omega})$ is the frequency dependent speed of plane wave travel measured in space samples per time sample. (*Phase distance* corresponds to *phase advance* in time domain language.)

Figures 11 and 12 show contour plot slices along the planes $\omega_x = 0$, and $\omega_z = 0$, respectively, of the normalized plane wave speed $c'(\underline{\omega})/c$ in the tetrahedral mesh. The innermost contour line is drawn at 99% of full speed and subsequent lines are drawn at 1% intervals. Because of the spatial sampling interval, there is a Nyquist limit on the spatial frequencies which may be supported on the mesh, namely $|\underline{\omega}| < \pi$. In addition, all transfer functions definable at any one junction, and the denominators of all transfer functions definable between any pair of junctions, are functions of z^{-2}, as may be seen from Figure 10. Therefore, frequencies above $\pi/2$ are not independent, and are constrained to be a copy of the frequencies below $\pi/2$. We have superimposed a circle marking this limit in the contour plots. The central area of each plot corresponds to lower spatial frequencies, and the outer regions correspond to higher spatial frequency. The angular position of a point on each plot indicates the direction of the wave travel in the planar slice being shown. Figure 13 shows the response on the hemispherical surface, $|\underline{\omega}| = \pi/2$, where $\omega_z = \sqrt{(\pi/2)^2 - \omega_x^2 - \omega_y^2}$.

By way of comparison, we show dispersion plots for the 6-port rectilinear 3D waveguide mesh [10, 2] with the same contour line settings. We computed these following a similar procedure as that outlined above for the tetrahedral case. Figure 15 shows a horizontal slice through the origin, and Figure 14 shows a diagonal slice through $\omega_x = \omega_z$. Figure 16, again, shows the response on the hemispherical surface, $|\underline{\omega}| = \pi/2$.

4 Conclusions

Both the rectilinear and the tetrahedral 3D meshes have reasonable dispersion characteristics. And both model a wave speed of $c = \sqrt{1/3}$ space samples per time sample. We compute that the number of tetrahedrally arranged junctions required to fill a given volume is 35% *less* than that required for the rectilinear mesh; and the number of bi-directional delay units required for the tetrahedral mesh is 57% *less* than that required for the rectilinear mesh to fill the same given volume, thus saving substantial memory. Furthermore, the tetrahedral mesh is multiply-free and may be implemented efficiently in high-speed hardware. Applications in concert hall design, acoustical research, musical instrument synthesis, and reverberation are now practical.

References

[1] Rocchesso, D. and F. Fontana. "A New Formulation of the 2D-Waveguide Mesh for Percussion Instruments", *Colloquium on Musical Informatics*, Bologna, November 1995.

[2] Savioja, L; Rinne, T. and Takala, T. "Simulation of Room Acoustics with a 3-D Finite Difference Mesh", *Proc. ICMC*, Århus, 1994.

[3] Smith, J. *Music Applications of Digital Waveguides.* CCRMA, Stanford Univ., Stanford, CA, Tech. Rep. STAN-M-39, 1987.

[4] Stilson, T. Technical presentation, *Eighth Annual Meeting of the CCRMA Associates*, Stanford Univ., May 1994.

[5] Strikwerda, J. *Finite Difference Schemes and Partial Differential Equations.* Wadsworth & Brooks, Pacific Grove, CA, 1989.

[6] Van Duyne, S. and J. Smith. "The Tetrahedral Digital Waveguide Mesh." *Proc. IEEE Workshop on App. of Sig. Proc. to Audio and Acoust.*, Mohonk, 1995.

[7] Van Duyne, S.; Pierce, J. and J. Smith. "Traveling Wave Implementation of a Lossless Mode-Coupling Filter and the Wave Digital Hammer." *Proc. ICMC*, Århus, 1994.

[8] Van Duyne, S. and J. Smith. "A Simplified Approach to Modeling Dispersion Caused by Stiffness in Strings and Plates." *Proc. ICMC*, Århus, 1994.

[9] Van Duyne, S. and J. Smith. "The 2-D Digital Waveguide Mesh." *Proc. IEEE Workshop on App. of Sig. Proc. to Audio and Acoust.*, Mohonk, 1993.

[10] Van Duyne, S. and J. Smith. "Physical Modeling with the 2-D Digital Waveguide Mesh", *Proc. ICMC*, Tokyo. 1993.

[11] Wen-Yu Su. Personal communications, 1994.