UNIVERSITY OF CALIFORNIA, SAN DIEGO

Generative Audio Systems: Musical Applications of Time-Varying Feedback Networks and Computational Aesthetics

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Music

by

Gregory Surges

Committee in charge:

Professor Tamara Smyth, Chair
Professor David Borgo
Professor Sarah Creel
Professor Lei Liang
Professor Miller Puckette

2015

The dissertation of Gregory Surges is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2015

DEDICATION

To Hannah, the love of my life.

TABLE OF CONTENTS

Signature Page . . . iii

Dedication . . . iv

Table of Contents . . . v

List of Figures . . . ix

List of Tables . . . xi

Acknowledgements . . . xii

Vita . . . xiii

Abstract of the Dissertation . . . xv

Chapter 1 Introduction . . . 1
1.1 Motivation . . . 1
1.2 Novel Contributions . . . 3
1.3 Dissertation Organization . . . 4

Part I: Background **6**

Chapter 2 Generative Systems . . . 7
2.1 Generative Audio Systems . . . 8
2.1.1 Feedback Systems . . . 10
2.1.2 Historical Examples of Generative Audio Systems . . . 14
2.2 Discussion . . . 22

Chapter 3 Computational Aesthetics . . . 24
3.0.1 Theories of Computational Aesthetics . . . 26
3.0.2 Exposure, Affect, and Boredom . . . 28
3.0.3 Alternative Approaches to Musical Emotion . . . 32
3.1 Music Information Dynamics . . . 33
3.1.1 Information Rate . . . 35
3.1.2 IR Applied to a Musical Example . . . 39
3.2 Discussion . . . 40

Part II: Complexity and Computational Aesthetics **42**

Chapter 4 Feedback and Complexity in Generative Systems 43
4.1 Feedback as Source Material - *Gates Nos. 1* and *2* 44
4.1.1 *Gates No. 1* 45
4.1.2 *Gates No. 2* 47
4.1.3 Discussion and Further Work 49
4.2 Generativity and Complexity - the *Feld* system 49
4.2.1 System Structure 50
4.2.2 Order and Complexity in the *Feld* system 56
4.2.3 Performance and Presentation 57
4.2.4 Performance Decisions 58
4.2.5 Listening Experience 58
4.3 Conclusion 59

Chapter 5 Machine Learning and Computational Aesthetics in the PyOracle System 61
5.1 Introduction 61
5.2 Machine Improvisation 63
5.3 Audio Oracle 64
5.4 Computational Aesthetic Theories 66
5.4.1 Music Information Dynamics and Information Rate . . . 66
5.4.2 Tuning the AO Algorithm 67
5.5 Music Generation with PyOracle Improviser 70
5.5.1 Composition and Improvisation using PyOracle Improviser 71
5.5.2 Case Study: *Nomos ex Machina* 74
5.6 The `ao` Object 75
5.6.1 Features of the `ao` Object 77
5.7 Conclusion 80
5.8 Acknowledgements 81

Part III: Generative Audio Systems **82**

Chapter 6 Phase Distortion Using Time-Varying All-pass Filters 83
6.1 An Overview of Synthesis Applications of All-pass Filters . . . 84
6.2 The First-Order All-pass Filter 85
6.3 The Second-Order All-pass Filter and its Cascade 88
6.4 Phase Distortion with the Second-Order All-pass 89
6.4.1 A time-varying all-pass filter 89
6.4.2 Simple Application to a Clarinet Sample 97
6.5 Applications 97
6.5.1 Modulation at Sub-Audio Rates 97
6.5.2 Audio-Rate Phase Distortion 98
6.6 Conclusion 101

6.7 Acknowledgements . . . 102

Chapter 7 Generative Feedback Networks Using Time-Varying All-pass Filters . 103
7.1 Introduction . . . 103
7.2 Synthesis Applications of All-pass Filters . . . 104
7.2.1 The First- and Second-order All-pass Filters . . . 104
7.2.2 Frequency-selective Phase Distortion . . . 105
7.3 Stability Analysis of Time-Varying All-pass Filters . . . 106
7.3.1 Instability of the Time-Invariant All-pass Filter . . . 106
7.3.2 Stability Analysis . . . 107
7.4 A Power-Preserving All-pass Filter . . . 109
7.4.1 Differences in Output . . . 112
7.5 All-pass Filters in Feedback Networks . . . 113
7.5.1 Transfer Function Analysis of Time-invariant Feedback Systems . . . 114
7.5.2 Spectra of Time-varying Feedback Systems . . . 117
7.5.3 A Self-Modulating All-pass Feedback Network . . . 119
7.6 Conclusion . . . 123
7.7 Acknowledgements . . . 125

Chapter 8 Evaluation of a Generative Audio System . . . 127
8.1 Introduction and Motivation . . . 127
8.2 The Generative Audio System Component . . . 129
8.2.1 Network Topology . . . 129
8.2.2 Modulation Signal Generation . . . 130
8.2.3 Output of the Generative Audio System . . . 131
8.3 The Computational Aesthetic Analysis Component . . . 134
8.3.1 Audio Oracle Analysis . . . 134
8.3.2 Compositional Applications of IR Analysis . . . 135
8.4 Additional Signal Processing . . . 137
8.5 Evaluation . . . 138
8.5.1 Audio Oracle Evaluation of Synthesis Output Complexity . . . 140
8.5.2 Kolmogorov Complexity Analysis of Synthesis Output . 143
8.5.3 AO Evaluation of the IR-based Adaptive Timer . . . 147
8.5.4 Kolmogorov Complexity Analysis of the Adaptive Timer 150
8.5.5 Summary and Discussion . . . 152
8.6 Conclusion . . . 154

Chapter 9 Conclusion . . . 156
9.1 Recapitulation . . . 156
9.2 Future Work . . . 160

Appendix A Report on the David Tudor Archives at the Getty Center 162
A.1 Background and Goals 162
A.2 Findings 163
A.3 Conclusion 165

Appendix B The Hardware Used in *Feld* 166
B.1 Overview 166
B.2 Related Works 167
B.3 The Devices 168
B.3.1 Common Features 168
B.3.2 USB-Octomod 170
B.3.3 tabulaRasa 171
B.3.4 pucktronix.snake.corral 173
B.4 Further Work and Evaluation of Hardware 174
B.5 Acknowledgements 175

Bibliography 176

LIST OF FIGURES

Figure 2.1: Block diagram of Karplus-Strong string synthesis algorithm 12

Figure 3.1: IR function superimposed on spectrogram of music signal 39

Figure 4.1: An example feedback loop from *Gates No. 1* 45
Figure 4.2: System structure of *Gates No. 2* 48
Figure 4.3: A high-level block diagram of the *Feld* system 50
Figure 4.4: *Feld* signal-processing "node" 55

Figure 5.1: The PyOracle Improviser interface 62
Figure 5.2: An example of an Audio Oracle structure 65
Figure 5.3: The ideal AO model 67
Figure 5.4: An AO with the distance threshold set too low 67
Figure 5.5: An AO with the distance threshold set too high 68
Figure 5.6: Total IR as a function of Distance Threshold for Prokofiev piano work . 68
Figure 5.7: Total IR vs. Distance Threshold for Shakuhachi recording 69
Figure 5.8: Sound-checking *Nomos ex Machina* 73
Figure 5.9: Score for *Nomos ex Machina* 76
Figure 5.10: The `ao` object help patch 78

Figure 6.1: The position of the pole-zero pair for the first-order all-pass filter 86
Figure 6.2: Phase response of a first-order all-pass filter 87
Figure 6.3: Effects of f_π and f_b on the phase response of the second-order all-pass . 90
Figure 6.4: The position of the poles and zeros for the second-order all-pass filter . 91
Figure 6.5: Effect of modulation depth and transition region on phase distortion . . 92
Figure 6.6: Modulating individual components of a clarinet tone 96
Figure 6.7: Clarinet passage with spectral modulation of selected harmonics 98
Figure 6.8: Clarinet passage with $\tilde{f}_\pi(n)$ driven by fundamental frequency estimator 99
Figure 6.9: Example of audio-rate modulation of $\tilde{f}_\pi(n)$ 100

Figure 7.1: Influence on stability of the interaction between pairs of parameters . . 110
Figure 7.2: Effects of instantaneous change in f_π 113
Figure 7.3: Output of time-invariant and power-preserving filter types 114
Figure 7.4: Magnitude spectrum of filter output from both filter types 115
Figure 7.5: Time-invariant filter instability 116
Figure 7.6: Block diagram of all-pass filter cascade in feedback configuration. . . . 117
Figure 7.7: Example pole-zero plots as f_b is varied while f_π is kept fixed 117
Figure 7.8: Example magnitude responses as f_b is varied while f_π is kept fixed . . 118
Figure 7.9: Example magnitude responses as f_π is varied while f_b is kept fixed . . 119
Figure 7.10: Example magnitude responses as N is varied while f_π and f_b are fixed . 120
Figure 7.11: Output spectra of all-pass feedback network with time-varying parameters 121
Figure 7.12: Output spectra of all-pass feedback network with time-varying parameters 122

Figure 7.13: Output spectra of all-pass feedback network with time-varying parameters 123
Figure 7.14: Output spectra of all-pass feedback network with time-varying parameters 124
Figure 7.15: A block diagram of a self-modulating all-pass feedback network 125
Figure 7.16: Spectrogram showing self-organizing network behavior 126
Figure 7.17: Spectrogram showing self-organizing network behavior 126

Figure 8.1: The *AAS-4* feedback network topology 130
Figure 8.2: An example spectrogram from *AAS-4* all-pass filter feedback network . 132
Figure 8.3: Another example spectrogram from *AAS-4* all-pass filter feedback network 133
Figure 8.4: The adaptive parameter switching network 137
Figure 8.5: Plot of signal complexities with automatic thresholds 142
Figure 8.6: Plot of signal complexities with manual thresholds 144
Figure 8.7: Plot of signal orders obtained from FLAC compression 146
Figure 8.8: Plot of signal complexities with automatic thresholds 149
Figure 8.9: Plot of signal complexities with automatic thresholds 152

Figure B.1: USB-Octomod Assembly showing Teensy 2.0 169
Figure B.2: Illustration of USB-Octomod Control Flow 171
Figure B.3: tabulaRasa waveform design interface 172

LIST OF TABLES

Table 8.1: Complexity for frame size of 2048 samples and automatic threshold . . 142
Table 8.2: Complexity for frame size of 2048 samples and manual threshold 143
Table 8.3: Results of calculating order M_K from Kolmogorov complexity K 145
Table 8.4: Complexity for frame size of 8192 samples and manual threshold 148
Table 8.5: Results of computing Kolmogorov complexity K and order M_K 151

ACKNOWLEDGEMENTS

The material in Chapter 5 was originally published in:

Surges, G. and Dubnov, S. "Feature Selection and Composition Using PyOracle." *Proceedings, Workshop on Musical Metacreation, Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.* Boston, MA. 2013.

The material presented in Chapter 6 was originally published in:

Surges, G. and Smyth, T. "Spectral Modulation Using Second-Order Allpass Filters." *Proceedings, 10th Sound and Music Computing Conference.* Stockholm, Sweden. 2013.

The material in Chapter 7 was originally published in:

Surges, G., Smyth, T., and Puckette, M. "Generative Feedback Networks Using Time-Varying Allpass Filters." *Proceedings, International Computer Music Conference.* Denton, TX. 2015. It received the "Best Paper" award.

The material in Appendix B was originally published in:

Surges, G. "DIY Hybrid Analog/Digital Modular Synthesis." *Proceedings, 12th International Conference on New Interfaces for Musical Expression.* Ann Arbor, MI. 2012.

The dissertation author was the primary investigator and author of the above papers.

VITA

2009	B.F.A. in Music Composition and Technology. *High Honors*. University of Wisconsin, Milwaukee.
2010 - 2011	Instructor of Record. University of Wisconsin, Milwaukee.
2011	M.M. in Music Composition and Technology. University of Wisconsin, Milwaukee.
2012 - 2015	Associate Instructor, Music Department. University of California, San Diego.
2014 - 2015	Research Assistant, Qualcomm Institute. La Jolla, California.
2015	Ph.D. in Music. University of California, San Diego.

PUBLICATIONS

Surges, G., Smyth, T, and Puckette M. "Generative Feedback Networks Using Time-Varying Allpass Filters." *Proceedings, 2015 International Computer Music Conference.* Denton, TX. 2015. **Received ICMC 2015 Best Paper Award**.

Dubnov, S. and Surges, G. "Delegating Creativity: Use of Musical Algorithms in Machine Listening and Composition." *Digital Da Vinci: Computers in Music.* Springer New York, 2014. 127-158.

Surges, G. and Dubnov, S. "Feature Selection and Composition using PyOracle." *Proceedings, Workshop on Musical Metacreation, Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.* Boston, MA. 2013.

Surges, G. and Smyth, T. "Spectral Modulation Using Second-Order Allpass Filters." *Proceedings, 10th Sound and Music Computing Conference.* Stockholm, Sweden. 2013.

Surges, G. and Dubnov, S. "Probabilistic Methods of Cognitive Music Modeling and Improvisation." *UCSD Probability and Statistics Day.* La Jolla, CA. 2013.

Surges, G. "PyOracle - Analysis of Musical Structure Using Python." *PyCon.* Santa Clara, CA. 2013.

Surges, G. "DIY Hybrid Analog/Digital Modular Synthesis." *Proceedings, 12th International Conference on New Interfaces for Musical Expression.* Ann Arbor, MI. 2012.

Surges G. "Cytoarchitecture - Music for Digital Improvisors." *23rd National Conference on Undergraduate Research.* La Crosse, WI. 2009.

Burns, C. and Surges, G. "NRCI: Software Tools for Laptop Ensemble." *Proceedings, International Computer Music Conference.* Belfast, Ireland. 2008.

Surges, G. and Burns, C. "Networking Infrastructure for Collaboration Laptop Improvisation." *Proceedings, Spark Festival.* St. Paul, MN. 2008.

ABSTRACT OF THE DISSERTATION

Generative Audio Systems: Musical Applications of Time-Varying Feedback Networks and Computational Aesthetics

by

Gregory Surges

Doctor of Philosophy in Music

University of California, San Diego, 2015

Professor Tamara Smyth, Chair

This dissertation is focused on the development of *generative audio systems* - a term used to describe generative music systems that generate both formal structure and synthesized audio content from the same audio-rate computational process. In other words, a system wherein the synthesis and organizational processes are inseparable and operate at the sample level.

First, a series of generative software systems are described. These systems each employ a different method to create generativity and, though they are not strictly generative audio systems, they lay important groundwork for the rest of the discussion as ideas from and

contributions to the fields of generative algorithmic composition, computational aesthetics, music information dynamics, and digital signal processing are introduced.

Second, the dissertation investigates the use of a novel signal processing technique in which time-varying allpass filters are placed into feedback networks, producing synthesis structures capable of yielding interesting emergent sonic behaviors. Ideas from the field of computational aesthetics are employed to allow a large system built from these synthesis structures to become "aesthetically aware." Many theories about computational aesthetics center around a favorable balance between order and complexity in a stimulus - a successful artistic work is neither too orderly nor too complex. Using a model of human perception based on the "mere exposure" effect, which describes how listener appreciation and boredom change as they experience repeated exposure to a stimulus, the *AAS-4* system autonomously determines when and how to modify its own parameters to avoid repetitions that may lead to boredom in listeners.

The dissertation concludes with objective analysis of the generative system by considering the complexity of its output from an information-theoretic perspective. It was found that the generative audio system described here is capable of producing output with equivalent complexity to that of real-world musical examples. It is also shown that the level of complexity in the generated audio and real-world examples falls in-between the low complexity of silence and sinusoids and the maximal complexity of white noise, corresponding with the theories from computational aesthetics. Future directions of this work are also described. Two appendices describing related topics that would disrupt the flow of the dissertation are included.

Chapter 1

Introduction

1.1 Motivation

This dissertation is both a presentation of finalized results and a description of points along an ongoing investigation of generative music systems. Most, if not all, of my compositional and research work has, in some way, been related to ideas about generative music and the *generative audio system* - a new kind of generative music system, that combines synthesis and structure in the same generative process. This document traces the path that this work has followed, describing the significant research projects along with some of the fruitful detours encountered along the way. The work documented here includes research and contributions that combine the fields of digital signal processing, algorithmic composition and generative music, open-source hardware design, and music information retrieval.

Generative music is often broadly defined as music that arises from the specification of a system as a set of rules. After these rules have been specified, the system is left to work out the specifics of the music [1]. Generative techniques can be either manual or automated, and have been widely used in the production of a variety of art-forms [2]. Of particular interest to me is the field of live performance with generative electronic systems. I have long

been fascinated with the creative opportunities afforded in pieces and performances where a human performer (or performers) and an electronic system coexist as components of a larger cybernetic configuration, and exert mutual influence. In other words, the human responds to input from the electronics as much as the electronic system responds to input from the human. In these cases, the electronics are often at least partially generative - the performer may not specify low-level details about musical events, but instead may make high-level decisions about algorithmic parameters. This type of performance practice, involving a "push-pull" relationship between performer and instrument, has become extremely common in electro-acoustic music.[1]

In parallel with this interest in live and interactive generative music systems, I have long been interested in computational methods of musical analysis or evaluation. The work described in this dissertation applies ideas about computational aesthetics to the production of generative live electronic music. The project described in Chapter 8 attempts to replace the human component of the cybernetic system described in the previous paragraph with an aesthetically aware software component. Instead of relying on a human to make high-level decisions about musical parameters, the aesthetically aware software component makes these decisions based on the output of a computational model. A secondary motivating factor of this current research is related to what I perceive as a flaw in a large portion of the body of Music Information Retrieval (MIR) research. At the time of this writing, many advanced MIR techniques have been presented in the literature, ranging from automatic segmentation of musical audio signals [3] to automatic genre classification [4] to musical style modeling [5]. Unfortunately, these techniques rarely see application in creative works. When they do, they are often straightforward - such as segmentation according to an amplitude envelope follower as in [6]. Instead, these MIR techniques often seem to function as parts of the music

[1] By the author's count, at a recent (at the time of this writing) concert performance at the 2015 International Computer Music Conference, 75% of the pieces on one of the late-night concert programs featured a composer interacting with a (at least partially) generative system.

industry's push toward automation - forming components of recommendation systems or working as automatic transcriptionists [7, 8]. The goal here is to find a creative application of a particular advanced MIR technique. After all, what is the purpose of computer music research if not to make computer music?

1.2 Novel Contributions

This dissertation presents a number of novel contributions to the field of computer music. Chapter 4 and Appendix B describe a unique hybrid hardware and software generative music system. The system, called *Feld*, uses custom hardware devices to interface between a laptop computer and analog synthesis hardware. The hardware allows for computer control of the modulation and routing of the analog synthesizer, and the computer software generates parameter data and applies audio signal processing effects to the output of the analog synthesizer. Appendix B describes the original hardware devices used in this system, and Chapter 4 describes the overall system structure as well as the unique approach to sectional automation used. The *Feld* system was an initial attempt to use ideas from computational aesthetics in a real-time generative music system.

Chapter 5 describes the use of an information-theoretic measure obtained from a musical signal in a generative machine improvisation system. The PyOracle system is the first machine improvisation system to use the Audio Oracle algorithm, a modification of the Factor Oracle string-matching algorithm. PyOracle enables a machine improvisation system to learn musical structures from arbitrary, possibly multi-dimensional features extracted from audio signals. These structures can then be used to generate new variations on the original input signal. The chapter describes the implementation and use of the system, as well as a creative application of the system.

Chapters 6 and 7 are focused on the use of digital signal processing in so-called *generative audio systems*, and describe the use of time-varying all-pass filters as modulation

effects and in such generative systems. A novel technique is described in Chapter 6 that allows for modulation of only selected spectral components of a complex sound, while leaving the rest of the spectrum relatively unmodified. Chapter 7 addresses the issue of instability in time-varying all-pass filters, and a power-preserving formulation of the time-varying all-pass filter is presented. Finally, Chapter 8 describes a creative application in the form of a system, called *AAS-4*, that combines these generative feedback networks with a real-time measure of computational aesthetics. The output of the system consists of the signals obtained from a set of generative, time-varying feedback networks modulated by all-pass filters. The system uses an algorithm from the field of machine improvisation (Audio Oracle) to analyze these signals, "listening" for repetition. If the system determines that too much repetition has occurred, based on a simple model informed by research on perception and boredom, it attempts to produce novel output by making changes in its own parameters.

1.3 Dissertation Organization

This dissertation is organized into three main parts. This section will provide an overview of the structure of the document, along with a brief description of the contents of each chapter. Part I of the dissertation consists provides conceptual and technical background for the research to follow. The term "generative audio system" was introduced by the author in [9] and refers to a particular subset of generative music systems. Chapter 2 provides a definition and background of the "generative audio system". Chapter 3 describes theories of computational aesthetics and research on boredom and aesthetic preference that will be used throughout the chapters to follow.

Part II of the dissertation consists of Chapter 4 and Chapter 5, each of which describes a different generative music system. The first of these chapters describes the *Feld* system, a combined hardware and software system for the generation of live electronic music. The

second describes the *PyOracle* system, a library of code and a corresponding real-time system for machine improvisation and analysis of music information dynamics.

Part III consists of the final three chapters of the dissertation. The first two of these, Chapters 6 and 7 describe the use of time-varying all-pass filters in modulation effects and generative feedback networks. Stability issues in these time-varying filters are also addressed. Finally, Chapter 8 describes the *AAS-4* system, which combines the generative, all-pass filter-modulated feedback networks introduced in Chapter 7 with real-time analysis of music information dynamics. Finally, the dissertation also includes two appendices. The first of these describes my experiences at the David Tudor archives held at the Getty Research Library, while the second describes the custom hardware used in the *Feld* system.

Part I: Background

Chapter 2

Generative Systems

There are many names for algorithms and systems that autonomously produce music - *generative music*, *algorithmic composition*, *automated composition*, and *autonomous instruments*, to name a few [10, 11]. The work described in later chapters focuses on a newly recognized sub-category, referred to here as *generative audio systems*. The aim of this chapter is to introduce the generative audio system, the design of which is the ultimate aim of the current research, and to distinguish it from the more general and encompassing *generative music system*. Generative audio systems differ from other kinds of generative music systems in that they operate entirely at the sample level. In other words, the generative and synthesis processes are one and the same. Before entering into any detailed discussion of the current research, there is some background information which must be given on these systems. This chapter will give a conceptual and historical overview of generative audio systems as a way of providing background information for the chapters to follow.

Generative techniques are often used in art-making, and can include both computer-based and manual approaches to producing works through procedures that fall outside the control of the art maker [2]. Any generative music system depends on both sonic materials and the organizational principles which apply to those sounds. Generative audio systems additionally have a very tight coupling between these two elements, as will be elaborated

below. Historically, composers working with generative audio systems have shaped their works through either real-time or pre-composed manipulation of system parameters. In these cases, the performer serves as auditor, curator, and censor - emphasizing, highlighting, diminishing, or suppressing certain aspects of the system's output in order to produce a satisfying aesthetic experience. Although there is often an expressed desire to bring forth the "music in the machine," there is almost always a human shaping the aesthetic experience in time. This stands in contrast to the "pure computer-generated music" described by Nick Collins, where "the operation of the artwork is free of complicated human intervention during execution" [2].

This chapter will begin with a definition and overview of generative music systems and generative audio systems, with a particular focus on systems that employ feedback. Some important concepts that arise when working with feedback will be presented as a way to qualify some of the aspects and behaviors of these systems that are less easily analyzed. The final section contains a number of historical examples of composition using generative audio systems and feedback. The aim is to place the research in the following chapters into a historical and theoretical framework.

2.1 Generative Audio Systems

Composers have employed generative compositional techniques for generations. Although composer and computer music researcher Nick Collins makes a distinction between generative music and algorithmic composition - with the former being a real-time phenomenon, and the latter happening offline - it is important to describe early approaches to computer-based algorithmic composition [2]. Experiments in algorithmic composition have been on-going since the mid-1950s [11]. Many early examples of algorithmic composition, including landmark works by Hiller, Koenig, and Xenakis, used computer programs to generate material which was later converted into a traditionally notated score for performance

on acoustic instruments [2]. Though the concepts explored in these compositions were novel, often involving probability and stochastic processes, their presentation in familiar media (i.e. Xenakis' string quartet from the early 1960s *ST*/4, composed using stochastic a model of particle interactions) kept them somewhat rooted in the traditions of Western art music [12]. James Tenney's computer music compositions are well-known for being some of the first examples of fully computer-generated music. Tenney's model was to algorithmically generate a score, and then realize it through digital synthesis [11]. This score / orchestra paradigm is reflected in the design of computer music software like the *MUSIC N* family and *CSound* and possibly manifests itself through the distinction between control and signal rates in modern languages like Pure Data and Max [13, 14]. There are many more recent examples of computer-based generative music systems, many of which also produce their sonic output in real-time, such as Essl's *Lexicon-Sonate*, Eigenfeldt and Pasquier's *Kinetic Engine*, and Bown's work with recurrent neural networks, to name but a few [15, 16, 17].

In parallel - and perhaps, in contrast - with this ongoing creative research in computer music, another group of composers was investigating the use of electronic technology in live musical performance. The composer John Cage had perhaps the most significant influence on this group, notably for his pioneering live electronic music, but also for his insistence on embracing chance processes and his role in the creation of some of the first multi-media "happenings" [18]. Many of the composers used their own, homebrew equipment, and formed groups such as "Composers Inside Electronics" where ideas, circuitry, and music could be shared by practitioners [19]. An important concept for much of this work is the "circuit score" - a type of musical composition where the compositional logic is embedded in the same hardware that produces the sound. In such a piece, the hardware configuration alone is sufficient to produce the piece - without any further instructions to performers. This approach is nowhere more evident than in the music of David Tudor, a virtuoso pianist and Cage collaborator who later became a composer of live electronic music [20]. The idea of

the circuit score is closely related to that of the generative audio system, and the work of Tudor - perhaps the archetypical composer of this group - and others will be discussed in more detail below.

To reiterate: the phrase *generative audio system* is used to refer to a generative music system in which dynamic and surprising audio output is generated (with little or no human input) at the signal level - that is, the generative process is not based on symbolic manipulations of higher-level musical data, but instead functions directly through audio synthesis or processing. In other words, generative audio systems form a particular subset of generative music systems. In these systems, high-level musical details - such as form, dynamics, and timbre - emerge from the same algorithm or computational process that performs the audio generation itself.

2.1.1 Feedback Systems

In many cases, generative audio systems make use of *feedback*. A feedback system can be defined as one which transforms an input, the result of which then becomes the output and appears again at the input after a delay [21]. Feedback systems appear in engineering, nature, societies, and, of course, music. Feedback is commonly used in digital signal processing (DSP), for example in the realization of a filter with an infinite impulse response (IIR). This type of feedback is vital to the allpass filters discussed in Chapters 6 and 7, but I am also interested here in feedback that operates at a higher level. In an IIR filter, the feedback path includes simple multiplication and delay elements, with specific coefficient values selected to produce a particular frequency response in terms of magnitude, phase, or both. In the systems used to create the works discussed later in this section, the feedback path often includes complex and time-varying signal-processing blocks like filters, phase-shifters, and modulators.

Feedback systems can be characterized by the polarity of the feedback path, and the

feedback can be either negative or positive [22]. Negative feedback occurs when there is a inverse relationship between the input and output of a system. These systems tend toward equilibrium and stability. Positive feedback, on the other hand, occurs when changes in the system produce larger changes in the same direction. These systems are usually unstable. In [23], a distinction is also made between internal and external feedback. Systems with internal feedback route the output of the system back to the input directly, while those with external feedback allow the environment to affect the signal between output and input. Chapter 4 describes systems which use both internal and external feedback.

In addition to feedback polarity and the distinction between internal and external feedback, it will be useful to define some further characteristics of feedback systems: *non-linearity, iteration, coupling, self-organization*, and *complexity* [21]. These terms can be useful when discussing the behavior of complex, possibly time-varying, feedback systems (such as the ones described in Chapter 8) which may not be easily analyzed using traditional techniques.

The issue of *non-linearity* arises when working with feedback systems. Although non-linearity has a particular meaning in the context of DSP, Sanfillipo and Valle provide a slightly different, yet related, definition. The non-linearity in a feedback system is a result of "a process with circular causality. . . In such a configuration, effects are also causes, and there is a mutual relation between them" [21]. In other words, the order of components - even linear ones - in a feedback system, can be significant, and the relation between cause and effect can be disproportionate [23]. This is even more the case when traditionally non-linear or time-varying components are used [24].

Due to the *iterative* processing inherent in these systems, wherein a signal recursively passes through some form of processing block, they are often self-sustaining and capable of producing variations on initial conditions. As material recirculates through the system, transformations are applied again and again, producing generations of changes. This iterative

nature can also result in types of interrelations between musical parameters which may not occur in non-feedback systems [21]. As an example, consider the familiar Karplus-Strong string synthesis algorithm [25]. In this design, shown in block diagram form in Figure 2.1, a noise burst - applied at $X(z)$ in the figure - is used to excite a recirculating delay network. As the noise burst recirculates through the network, it is repeatedly smoothed by a low-pass filter. The result is a noise burst decaying to a sinusoid. The fundamental frequency of the output is dependent on the delay time n. A well-known side-effect of this algorithm in its basic form is that the decay time of a note is also dependent on n, as well as the magnitude response of the low-pass filter [25]. As such, in this particular feedback system, fundamental frequency is directly related to note duration - the parameters are interrelated due to the iterative nature of the network.

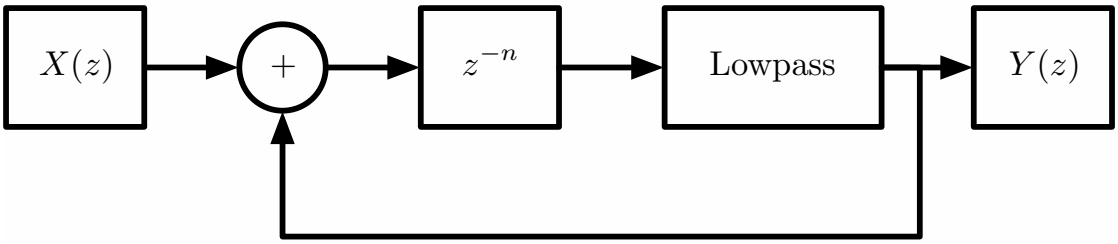

Figure 2.1: Block diagram of Karplus-Strong string synthesis algorithm.

The circular causality described in [21] - where causes and effects are often difficult to distinguish - leads to *coupling*, where components of a system are mutually influential. When this behavior extends across an entire system of components, the system will have a specific set of characteristic behaviors. All of the individual components are important to the overall output, and can have significant effects. Further, behaviors may arise which are not additive but instead synergistic [21]. In other words, the whole can be greater than (or radically different from) the sum of its parts. A basic signal-processing block, say a delay, can have important and surprising effects when integrated into a larger feedback system. Small changes in the network, for example swapping the order of two elements, can have

large effects on the output [23].

Many composers have been attracted to the property of *self-organization* which is often associated with feedback systems [26, 27, 28]. A system which is self-organizing is one that can autonomously shift from one state to another or from which can emerge high-level patterns based on the behavior between low-level components, without outside influence [21, 23]. In other words, self-organizing systems are characterized by the "emergence of coherent patterns at a *global* level out of *local* interactions between the elements of a system" [23]. The states of a self-organizing system can be characterized as either stable (characterized by static behavior) or unstable (continually shifting behavior). Self-organization - corresponding to stability - can be opposed with self-disorganization - instability - and they can be characterized in terms of their relative entropy or amount of randomness [21]. Self-organizing behaviors lead to a decrease in entropy, while self-disorganizing behaviors lead to an increase.

Finally, *complexity* has emerged as a phenomenon worthy of scientific investigation, and of relevance to feedback and generative audio systems [23]. A complex system is constructed of many small, simple processing units. From the interaction of these individual units, surprising and complicated results can emerge. Sanfillipo and Valle argue that since feedback is "a simple behavior that leads to unexpected results... it can be described in the framework of complexity." The unexpected results are often referred to as "emergent behavior" - and in the case of a generative audio system, the dynamic and surprising output "emerges" from the interactions between individual interconnected system components. The authors of [23] also point out the importance of distinguishing between complexity and total unpredictability. Complex systems are capable of both chaotic and non-chaotic behavior [23]. Chapter 4 of this dissertation describes a system constructed from many small processing blocks from which emerge complex gestural and timbral shapes and formal structures.

2.1.2 Historical Examples of Generative Audio Systems

The following discussion focuses on a set of examples of composition using feedback. Perhaps beginning with John Cage and David Tudor's collaborations in the 1950s and 60s, composers of electronic music have used feedback in a variety of ways, employing both analog and digital methods of sound production. Along with Tudor, Cage virtually invented the genre of live electronic music, and his piece *Imaginary Landscape No. 1* (1939) was one of the first pieces to use electronic equipment (primarily a turntable with a test-tone record) in performance [18]. While Cage is well-known for his systematic approach to acoustic and tape composition using chance processes, and later employing computer software as a compositional tool, he is particularly important to mention here for works such as *Cartridge Music* (1960) and *Variations II* (1961), both of which ask performers to perform using complex electro-acoustic systems capable of exerting their own influence on the performances [29]. After collaborating as performer (and in some cases, some might say co-composer) on many of Cage's works, Tudor went on to develop a unique approach to the composition and performance of live electronic music, and he became a key figure in the development of generative audio systems. Tudor's compositions often consisted of complex electronic networks either excited by pre-recorded material or connected into feedback networks [30]. The networks themselves became semi-automatic compositions and functioned using entirely analog signals. Like in most analog synthesizer music, there was little or no distinction between "control" and "audio" signals. Tudor's music was complex, bold, sometimes harsh, often alien, extremely rhythmic, and always evolving, and his work and approach has been a central inspiration to the pieces described in Chapter 4 and Chapter 8.

Tudor was a primary figure in the development of generative audio systems and is well-known for his use of feedback in composition. This interest began to emerge in his infamous realization of Cage's *Variations II* (1961) [29]. Cage designed a set of transparent

sheets with lines and dots printed on them. To make a performance score, a musician would indeterminately overlap these sheets, and measure the distances from lines to dots, with each distance giving a value for one parameter. Tudor, who was still focused on the piano as he developed an interest in live electronic music, decided to use an amplified piano as his instrument. Cage biographer James Pritchett describes the piano setup as being quite complex:

> The whole system presents a very complex interaction of its various parts. Adjusting the levels of the various microphone signals, the ways in which the cartridges are deployed in the piano, and the ways in which the piano is played will alter the behavior of the whole system. However, the system is so complex that its behavior can never be totally predicted: the amplification of the piano made it, to some degree, an uncontrollable instrument. Tudor's own characterization of it was that he 'could only hope to influence' the instrument – he could not predict the nature of the sounds that would result from a particular action. [29]

This embrace of complexity and unpredictability would become a defining characteristic of Tudor's approach to electronic composition. Beginning with his first acknowledged composition *Bandoneon!* (1966)[1], Tudor employed large networks of simple, often homemade signal processing modules as a means of obtaining complexity and dynamic behavior in electronic sound. The piece revolved around Tudor's performance using an accordion-like instrument called a bandoneon [31]. Microphones were placed on both ends of the instrument, and the signals were used to excite a network of signal processors. The performance consisted of Tudor playing the bandoneon, along with the results of the signal processing network: modulated or otherwise processed sounds derived from the bandoneon. The program note offers an important insight into Tudors developing aesthetic: "*Bandoneon!* uses no composing means; when activated it composes itself out of its own composite instrumental nature." [32]

This complexity would recur in Tudor's music throughout the rest of his career. The instrumentation for many of his later works consisted of a tabletop covered in homemade

[1] Also known as "Bandoneon Factorial."

electronic devices arranged with many interconnections and feedback loops. *Untitled* (1972), composed for a duo tour with John Cage, was based entirely around the principle of electronic feedback. The piece consisted of a pair of signal processing paths which were allowed to feed back and interfere with each other. This interference served to disrupt the static drone which often emerges from a saturated feedback system. Due to the non-linear nature of Tudor's electronic components, as Ron Kuivila writes,

> The feedback path becomes its own control signal, and the 'instrumentalization of sound' that began with the 9 Evenings [the concert series where *Bandoneon!* was performed] no longer requires the introduction of an external sound source rendered by a performer in a traditional manner. [20]

In these works, the activity of the composer changes from controlling and specifying to auditioning and suggesting. Due to the complexity of the "composite instrument" formed by the signal processing network, with its coupled components and iterative processing, it becomes extremely difficult to entirely predict the effect of a specific parameter change.

Though most of Tudor's work involved almost entirely analog equipment, one of his final compositions provides a hint of how this style of instrument building might have been extended into the digital realm. *Neural Synthesis* (1995) was composed in collaboration with engineers from Intel, and uses a small digital computer to control a custom neurally-inspired analog signal-routing matrix microchip [33]. The chip consisted of non-linear amplifiers which could be interconnected at will, including feedback routing and varying gain at connection points. In addition, a subset of these amplifiers could also behave as relaxation oscillators, introducing additional input to the routing matrix. Though the chip was designed for neural computation, Tudor used it as a central hub for a selection of his homebrew analog gear. This allowed him to access an enormous number of configurations in performance, and produced seemingly autonomous and evolving sound.[2] The composer again tasked himself with navigating the space of possible configurations, to locate those yielding interesting and

[2]Tudor's matrix approach was an important influence on the pucktronix.snake.corral, described in Appendix B.

musical results. Video artist Bill Viola, a former student of Tudor, suggests that Tudor saw his performance practice as being different than improvisation within the given electronic framework, instead it was "...a probing exploration of the internal junctions inside the system to see what might be hiding there." [34] Viola claims that Tudor saw his approach as being more systematic than improvisatory, however this distinction is probably more conceptual than practical and is perhaps more reflective of bias on Viola's part. In 2013, I was awarded a grant from the Getty Research Library to study their collection of Tudor's archives, and my first-hand experience with the complexity and idiosyncratic nature of his approach is described in Appendix A.

Roland Kayn (1933 - 2011) was a relatively unknown composer of electronic music. Despite his lack of renown, Kayn was a pioneer in what he called *cybernetic music*, borrowing from the theories of cybernetics, information theory, and aesthetics [35]. Cybernetics, introduced by Wiener, is the study of self-regulation and feedback [36]. Kayn was particularly influenced by Max Bense, who posed that aesthetics were related to a combination of predictability and surprise. Since the balance between surprise and predictability was so vital to his music, Kayn avoided involvement in computer music, as he believed that programming languages influenced composers to think mechanistically rather than artistically (though Thomas Patteson argues that this may have been due to an oversight or misunderstanding on Kayn's behalf) [35]. Instead, Kayn favored analog electronics, making use of the particular non-linearities of analog equipment and creating feedback systems which generated endless variations on previous material. As described above, cybernetic theories describe two forms of feedback, negative and positive. Negative feedback results in equilibrium and regulation. Positive feedback, on the other hand, results in jumps from one state to another. Kayn's electronic systems exploited positive feedback - resulting from interconnections between components - in order to produce musical interest.

Kayn produced his most famous cybernetic music while working at the Institute of Sonology in Utrecht [35]. Central to these compositions was the studio itself. In addition to an advanced modular synthesizer, the studio had a custom sequencer which could be used to control the entire studio. Kayn used this sequencer to produce musical situations where the final results - in terms of both sound material and system configuration - were unpredictable, due to their dependence on initial conditions. In essence, Kayn's compositional work consisted of designing an electronic configuration and a series of sequences which defined the settings of and interconnections between studio components. After these sequences were perfected, the system was set into motion and recorded to tape.

Iannis Xenakis is another composer who must be mentioned in any discussion of generative audio systems. Xenakis is well-known for incorporating stochastic methods into his acoustic and electro-acoustic compositions, and he viewed probability functions as a way of generating complexity without resorting to serial techniques [37, 38]. Starting in the early 1970s, Xenakis developed a series of software programs which applied stochastic processes to the generation of waveforms. These programs all relied on the manipulation of a break-point function, which was then used as a look-up table in a wave-table synthesis system. Xenakis applied stochastic processes to a variety of parameters of the break-point function, most significantly the location of the breakpoints themselves. In the simplest case, at each cycle of the waveform, the x and y values of each break-point are updated according to a constrained random walk.[3] The breakpoints are then interpolated and the break-point function is used in for wave-table synthesis [38]. This form of synthesis is conceived of entirely in the time domain, and is theoretically capable of evolving toward any possible waveform or sound - though the probability of arriving at a particular waveform is extremely low.

[3]These parameters and constraints were also often subjected to "secondary random walks" themselves, allowing them to evolve over time. The reader is encouraged to see Luque's "The Stochastic Synthesis of Iannis Xenakis" for a more detailed treatment of the various programs [38].

Xenakis apparently did often exert some high-level manual or probabilistic control over the way these synthesis algorithms were deployed in their final musical form [38]. Despite this, when a single instance of the generative process is considered, elements of pitch, timbre, dynamics, and possibly even rhythm are all directly related to the synthesis algorithm itself [37]. As Xenakis himself stated: "I am always trying to develop a program that can create the continuity of an entire piece" [38]. Though Xenakis' technique, unlike others discussed here, did not employ feedback, it is another important example of a generative audio system.

Research into generative audio systems, in particular those involving feedback, has continued. In [39], Christopher Burns documents his experiments using digital feedback networks in compositional settings. Burns describes the use of a soft-clipping function based on a guitar amplifier to ensure the stability of networks with non-standard topologies. This is similar to the way analog components will saturate rather than allow the signal amplitude to increase infinitely. Burns, writing from the perspective of a composer, argues that the harmonics introduced by the clipping function may be seen as a useful feature in a compositional context. Burns also describes the ways in which network topologies change the characteristics of the output. For example, if a network does not employ a change in sign in a feedback path, it will often produce DC - this is one possible end state of positive feedback. However, this tendency can be avoided by changing gains or delay lengths, effectively encouraging the system to jump to states other than DC. Burns' piece *Hero and Leander* (2003) makes use of a circular feedback system consisting of multiple sections based on standard wave-guides. The overall system gain and each of the delay times are controlled with envelopes, and the system is excited with a multi-channel sound-file. Instead of specifying musical details like rhythm, pitch, and timbre, the composer instead dealt directly with the parameters of delay times and gains.

Agostino Di Scipio's *Modes of Interference n.2* uses audio feedback modulated by saxophone and electronics [40]. Here, a saxophone body filters acoustic feedback, avoiding undesirable static feedback. In addition, a piece of software further modulates the sound. Finally, the performance space itself has an effect on the sounding result. Di Scipio designed a complex feedback system consisting of instrument, software, and architectural space, each of which has its own influence on the final sounding result. The composer provides a notated score for the performer, specifying the physical actions (fingerings, key-clicks, tonguing) to be performed using the saxophone. The sounding results are left unspecified. Similarly to Kayn's aesthetic, Di Scipio specifies part of the information necessary for the piece while leaving the rest up to the system (comprised of performer, saxophone, electronics, and environment) itself. A video of a performance given by Pedro Bettencourt demonstrates the subtle interactions between the acoustic instrument, acoustic space, and electronic processing [41]. Minuscule changes in physical location of the instrument, along with different fingerings and blowing techniques, produce unpredictable changes in the output.

Sanfillipo's *LIES (Live Interaction in Emergent Sound)* system was designed with the goal of producing surprising results without stochastic elements [23]. The piece *LIES (topology)*, a "real-time cybernetic music system," is constructed from multiple interacting feedback networks. These networks are described by the composer as comb filters with signal processing elements - such as ring modulation, frequency shifting, granulation, and reverberation - embedded in their feedback paths [23]. The parameters to these effects are chosen so that they produce output that will be perceptually merged and heard as as a single sound rather than as discrete events. The output of the system is played into a performance space, where it is further transformed by the room acoustics before being captured by microphones for recursive processing. The role of the performer is to change the coefficients of the feedback routing matrix and the parameters of the signal processing

components, as a way of "exploring the identities of the system and the perturbations that can push it towards unexpected behaviors" [23]. The output of the system is characterized by an extremely wide registral range, alternating high frequency tones with low rumbles and thumps. The influence of the performer can be heard in the way that registral changes mark sectional boundaries [42].

Finally, Holopainen's work on "feature-feedback" and "autonomous instruments" is very relevant to the current discussion. In feature-feedback systems, the output of a synthesis algorithm is subjected to some feature extraction [10]. The data retrieved from the feature extraction is then used to generate parameters for the synthesis algorithm. The feedback used here is not in the form of audio samples, but as feature descriptors extracted from the output of the system. Holopainen uses this model to create what he calls "autonomous instruments" - computer programs that "generate music algorithmically and without real-time interaction, from the waveform level up to the large scale form" [10]. Holopainen's goal is similar to the goal of the current research, in that he is interested in the development of generative audio systems, although his specific approach (the use of feature-feedback) is different. The audio examples provided by Holopainen demonstrate a wide array of sounds: chaotically modulated sinusoids, oscillating pitches, textures with extremely long transients, and slowly undulating tones with smoothly changing timbres [43].

There are a multitude of other contemporary approaches. It is clear, from the multitude of examples, that feedback is a compositional resource which suggests and encourages a variety of compositional approaches and can produce a diversity of musical material. For these reasons, many of the systems described in later chapters employ feedback in some form.

2.2 Discussion

This chapter presented background information on generative audio systems, providing context and motivation for further development and discussion. Section 2.1 defined the *generative audio system*. These are systems that autonomously generate music and are characterized by a tight coupling between signal generation and compositional logic. Many such systems, and in particular the systems discussed in later chapters, use some form of feedback as a way to achieve this type of behavior. Section 2.1.1 defined a few important characteristics of feedback systems: non-linearity, iteration, coupling, self-organization, and complexity. These terms provide some useful language for qualitatively describing the behavior of generative feedback systems which might be otherwise difficult to explain or quantify.

Some examples of pioneering composers and compositions were given in section 2.1.2, with a focus on analog systems but also including more recent computer music research. Composers like David Tudor, Roland Kayn, and others employed feedback in their generative audio systems. Many of their generative systems were designed to produce complex behaviors from the interaction of many smaller components. There is also a trend, exemplified in Tudor's performance practice, of systems which require human intervention to produce a meaningful musical experience. This is seen both in pieces requiring a human performer, like those of Di Scipio; and in those relying on pre-compositional decision-making, like those of Kayn or Burns.

The idea of the generative audio system is central to the current research. In the chapters to come, a variety of generative audio systems will be discussed. Chapter 4 will describe two systems which employ digital and electro-acoustic feedback as primary sound sources. Chapter 4 also describes a system which uses an idea similar to Holopainen's feature feedback to generate aspects of musical form. In Chapters 6 - 8, the use of time-varying allpass filters will be developed and their application as components in generative

audio systems will be discussed.

Chapter 3

Computational Aesthetics

Though the generative audio systems described in Chapter 2 are capable of producing interesting, time-varying output, they generally depend on control from a performer or composer. The role of this individual is to make decisions about the form and material of the work - in other words, to make aesthetic decisions about when and how to change the musical output. For a composer like David Tudor, this meant acting as a performer on stage, adjusting controls in real time, while for Roland Kayn, this meant shaping his music by composing specific system configurations and signal routings which would then unfold over time via an electronic sequencer. Due to this dependence on a performer or composer, their systems were not capable of autonomously making aesthetically-informed decisions. The hardware system used for Tudor's *Untitled*, for example, would be hard-pressed sound like the piece without input from the composer. If a system were capable of performing aesthetic self-evaluation, then it could structure its own output in a musically meaningful way.

This chapter will introduce research which attempts to formalize and explain aesthetic perception, in order to provide background for the specific music information theoretic measure described in Section 3.1. The aim of this chapter is to present ways in which the aesthetic experience of listening to a piece of music might be measured or quantified. Through computer analysis of a musical signal - either a symbolic representation or sampled

audio - it is possible to characterize aspects of the music according to their information content. This information content is related to the amount of novelty present in the signal at a given moment, and has been shown to be related to human affect and surprise or familiarity [44]. A particular measure (Music Information Rate) will be defined and described, and its potential use in a generative audio system will be explored [45].

Surprise and familiarity, concepts often associated with novelty, are also closely related to the idea of *expectations* - the listeners' beliefs about the future. Familiarity establishes and reinforces expectations, while surprise interferes with or contradicts these expectations. It has been suggested that a positive aesthetic experience is dependent on a proper balance between reinforcing and confounding expectations [46]. The experience of *boredom* has also been shown to be related to both novelty and repetition. The "mere exposure" effect, described below, causes subjects to rate a stimulus more positively as they are repeatedly exposed to it [47]. As the number of repetitions (or exposures) increases past a certain point, however, boredom begins to exert an effect and cause a less positive experience. This discussion aims to lay the groundwork for the self-reflective generative audio system described in Chapter 8.

The Oxford Dictionary defines aesthetics as "a set of principles concerned with the nature and appreciation of beauty, especially in art" [48]. In particular, the discussion will focus on the idea of aesthetic formalism, a view which assumes that the quality of an aesthetic object is a function of purely formal attributes - those which we can perceive directly [49]. Clearly, art is not solely a function of formal parameters, as many works of art make reference to real-world concepts, emotions, and events. However, this focus on formalism will be an advantage, as computers can more easily be programmed to extract formal features than the referential ones used in some approaches. The field of computational aesthetics, active since the early 20th century, forms an interesting parallel development to the study of self-regulating or self-organizing feedback systems described in Chapter 2.

Composers, like Roland Kayn, were interested in integrating formal aesthetic theories with their music, and in some cases interacted directly with some of the theorists working in the field of computational aesthetics [35]. Many theories of computational aesthetics are based on an assumption that the aesthetic worth of an art piece is based upon a balance between simplicity and complexity, familiarity and surprise, or order and entropy. This basic idea has been applied to a number of different media, and has been used for both predictive and generative applications [50].

Psychological research has also shown the influence of order, in the form of repeated exposures to a stimuli, on the affective response to that stimuli. It has been shown, across a variety of forms of stimuli, that repeated exposure creates a more positive affective response [51]. The more familiar a subject is with a stimuli, the more favorably they will rate it. In addition, theories on the interaction between repetition and boredom have been developed, which attempt to explain how affective response can change from increasingly positive to increasingly negative as the number of exposures increases past a certain threshold [47]. This chapter will provide an overview of some of the main theories in computational aesthetics, and discuss research on exposure, boredom, and affect.

3.0.1 Theories of Computational Aesthetics

Computational aesthetics is a branch of computer science research which attempts to formalize aesthetic appreciation [52]. Since the early part of the 20th century, theorists have proposed ways in which aspects of the aesthetic worth of a work of art might be objectively measured. George Birkhoff, in his 1933 book *Aesthetic Measure*, proposed a method for such a computation. Birkhoff proposed the ratio $M = O/C$, where M is the *aesthetic measure* of an object, and O and C its order and complexity, respectively [53]. Complexity is related to the amount of effort a subject must expend in order to understand an object, and order to the release of "unconscious tension" which comes from features like

symmetry and balance [50]. Birkhoff focused on purely formal aspects of the objects, and attempted to quantify their order and complexity mathematically, ignoring any symbolic meaning or reference [50]. Holopainen, in [10], provides a compact survey of ways of quantifying complexity, including *complexity as size*, *entropy*, and *algorithmic information content*. A further approach to measuring musical complexity through a measure of fractal dimension is described in [54]. One of the above mentioned methods, the *algorithmic information content* - also referred to as Kolmogorov complexity - has been used in studies of computational aesthetics applied to paintings [55]. The Kolmogorov complexity of a string is a measure of the size of the smallest program which can reproduce that string. In practice, this measure is usually estimated using compression algorithms, like .jpg and .png in the case of visual images [55]. The ratio of the length of the compressed string to that of the full, uncompressed version is called the compression ratio and the Kolmogorov complexity can be used to calculate to the amount of order in the string [55]. Though Birkhoff's specific formula has repeatedly been challenged and modified, the basic notion that order and complexity inform aesthetic experience has generally remained intact [56].

Leonard Meyer's *Emotion and Meaning in Music* (1956) is another important early work on aesthetics and musical meaning, and has particular relevance to the ideas of computational aesthetics [46]. David Huron describes Meyer's approach as a compromise between two basic positions: *absolutist*, which is a type of aesthetic formalism in which musical meaning is defined by the relationships of strictly musical materials, and *referentialist*, where musical emotion is created by reference to non-musical, real-world ideas [57]. Meyer theorizes that listeners hear new music through their previously existing, learned expectations. These expectations are generally in the form of stylistic norms, which can define things like acceptable harmonic progressions, theories of tuning, voice leading, and formal structure. As a listener hears a new work, he or she hears it in relation to these previously developed expectations. As the work deviates from these expectations, the listener will experience

these deviations as “emotional or affective stimuli” such as surprise, suspense, anxiety, or humor depending on the context [57].

The ideas of Birkhoff and Meyer have been applied to visual aesthetics [58], television and radio program structure [59, 60], humor [61] and music [62]. Section 3.1 will describe some ways in which researchers have attempted to apply the theories of Birkhoff and Meyer, among others, to the study of music.

3.0.2 Exposure, Affect, and Boredom

One common way of creating order in a musical work is through the use of repetition, and almost all forms of Western music employ repetition as an important structural element. By repeating material, such as the exposition of a sonata form which recurs in the conclusion or a rounded binary form, a composer can create a sense of familiarity in listeners. Familiarity, an effect of repeated exposure, has been shown to be related to affect, with a general finding that an individual who experiences repeated exposure to a stimuli will have a more positive affective response to the stimuli [51].

A meta-review, by Bornstein et al., collected and analyzed past experiments which tested and confirmed the original “mere exposure” theory using nonsense words, Chinese characters, photographs of faces, unfamiliar foods, and other stimuli of variable duration, complexity, and number of exposures [63]. In general, these studies were performed by presenting a particular stimulus to a subject. Variables which were manipulated in the studies include stimulus type, stimulus complexity, number of exposures, exposure sequence (in cases where subjects were exposed to multiple stimuli), exposure duration, and stimulus recognition (accomplished through priming procedures where the subjects were subliminally exposed to stimuli) [63]. The review presented some interesting findings. First, the order of stimuli is significant, with presentation of a diverse sequence of stimuli with interleaved repetitions leading to a more positive affect than a sequence of repetitions of a single

stimulus. Second, the "frequency-affect curve" is shallow, and tapers off after approximately 10 exposures. Third, exposure can be subliminal - the subject need not be aware that he or she is being exposed to the stimuli, suggesting that familiarity with the stimuli is not needed for the exposure effect.[1] Finally, the complexity of the stimuli can have a significant effect on subject response, with more complex stimuli found to lead to a more positive response. The findings can perhaps be summarized as follows: repeated exposures, up to about 10 in number, lead to a more positive affect. After 10 exposures, the influence on affect begins to taper of. Both stimulus complexity and the sequence of presentation (whether immediate repetition or interleaved) can increase this positive affect.

In theory, then, the ideal piece of art is simply a sequence of repetitions of material. Most people who have attended a concert, however, have probably had the experience of a piece that "goes on for too long." Clearly, there is more at play than "mere exposure." In fact, Bornstein et al. suggest that the findings summarized in their review point to the influence of *boredom* on the subjects' affective response [63]. In their own words, "when stimulus exposures are brief, limited in number, and interspersed among presentations of other stimuli. . . the exposure-affect relationship is enhanced" [63]. Stimulus complexity could also play a significant role, in that more complex stimuli remain interesting after more exposures than do simple ones. The effect of repetition eventually producing boredom can be seen as arising from an evolutionary origin, as new stimuli can be dangerous and are best avoided, but repeated unfruitful stimuli become boring and should also be avoided in favor of more rewarding ones [63].

Cohen defines musical boredom as a limit on "the tension that is created both by the expectation for change and by the uncertainty of change" [65]. When this limit is reached in a listener, boredom occurs. The impact of boredom on the exposure effect is sometimes explained through a two-factor model [47]. In this model, both familiarity and

[1] Although this has been recently contradicted in [64].

boredom can influence affective response. As familiarity increases, so does positive affect. However, boredom eventually begins to have an effect, and exerts a negative influence on the affective response. In [47], it was suggested that boredom can become an influencing factor in three ways. First, subjects may be "boredom-prone" and therefore less responsive to repeated exposure [66]. Second, uninteresting stimuli can encourage boredom - the study found that Welsh figures (simple images used in a non-verbal personality test) induced less of an exposure effect than more complex optical illusions. Finally, the combination of "interesting" and "uninteresting" exposure seems to have an effect. The authors of [47] found that if interesting and uninteresting stimuli are used together, instead of independently, the uninteresting stimuli induce less of an exposure effect than they would have otherwise.

It was found in [67] that increasing numbers of exposures to musical stimuli generally produced an inverted U-shaped affect curve - where affect initially became more positive but eventually became more negative - but that different types of stimuli and listening conditions could influence this result. As described above, the general assumption of the "mere exposure" effect is that as the number of exposures increases, a stimulus will be rated more positively. After a certain point, however, the affect was found to become more negative - corresponding to the appearance of boredom. This study used the notion of "ecological validity" - a spectrum measuring how close the stimuli is to real music - as a variable across three experiments [67]. The authors claim that ecological validity is equivalent to stimulus complexity. Exposure to synthetic tone sequences, the least ecologically valid stimuli, produced affective ratings that did not increase with exposure even as listeners became more familiar with the material. More complex stimuli, like 15-second excerpts of orchestral recordings, produced the inverted U-shaped curve described in previous studies. A distinction was also made between focused and incidental listening, with focused listening producing the familiar inverted U-shaped curve and incidental listening producing a linear increase in positive affect. Thus, if music is heard passively, as background sound, it will

become more and more liked. However, if listening is performed more actively, in a concert setting for example, then liking will follow an inverted U-shaped curve.

A related notion to that of exposure is processing fluency. Proponents of processing fluency theories claim that the subject's ability to process, or understand, a stimuli has an effect on their response. The better a subject is able to process the stimuli, the more positively they will respond [68]. This can be shown by using priming procedures, where a stimulus is subliminally previewed before it is consciously given [69]. A stimuli which is primed is generally better received. If repeated exposure is a form of priming, and therefore increases processing fluency, then we can easily connect the two theories. In [68], it is argued that there is a balance between complexity and simplicity which informs a subject's experience. When exposed to a simple stimuli, a subject will easily attribute their fluency to its simplicity. However, when a more complicated stimulus is used, the subject will instead attribute the fluency to the notion of "beauty". If complexity continues to increase, the subject will eventually experience a decrease in processing fluency, which ultimately leads to a decrease in positive response [68]. Again, we find an inverted U-shaped curve: as complexity increases, the affective response will increase (as the subject begins to attribute their fluency to "beauty") before again decreasing (as the stimulus complexity becomes too great for the subject to process it).

Both of these theories, mere exposure and processing fluency, relate affective response to repetition. The mere exposure effect shows that repeated exposure to a stimulus leads to a more positive response. The more easily a subject is able to process, or understand, a stimulus, the more positively they will respond. Both of these factors are countered by some notion of boredom. In both cases, boredom acts to negatively influence affective response. As exposure and fluency increase, boredom also increases, and at a certain point begins to exert a greater effect - "overriding" the influence of exposure or fluency.

3.0.3 Alternative Approaches to Musical Emotion

The goal of the discussion above is not to argue that expectation and repetition are the only factors in musical emotion and aesthetic appreciation. It is likely that there are many other elements at play when one has an emotional reaction to a piece of music. A more encompassing approach to musical emotions can be found in the Multiple Mechanism Theory proposed by Juslin and Vastfjall [70]. It is important to consider this theory as a means of understanding and contextualizing the potential shortcomings of the framework used in the rest of this dissertation. The Multiple Mechanism Theory centers around six psychological mechanisms that influence musical emotions, each of which as been studied independently:

- Brain stem reflexes: Related to auditory sensations (dissonance, loudness, speed) and reflexes - these result in basic emotions related to sounds or musical events which signify importance or urgency.

- Evaluative conditioning: Emotions can be associated with specific music because of repeated exposure paired with other positive or negative experiences. When this music is heard, it can cause the listener to recall the emotions associated with these past experiences.

- Emotional contagion: Music can induce emotions in listeners through a feedback process. In such a process, emotions perceived in music trigger muscular or psychological processes in the listener, ultimately leading to a similar emotional state.

- Visual imagery: Some musical emotions can be related to the imagining of visual stimuli while listening to music. These visual images may then trigger emotions in the listener.

- Episodic memory: Similar to evaluative conditioning. In this case, a piece of music

is associated with a particular memory or event. When the listener remembers this event, associated emotions can be evoked.

- Musical expectancy: Related to the establishment and violation of listener's expectations, as described in the discussion of the theories of Leonard Meyer.

The authors of [70] point out that the incorporation of multiple mechanisms in their theory allows for reconciliation between previously opposed views of musical emotion (like the absolutist and referentialist perspectives), as it is likely that these opposing perspectives are considering different psychological mechanisms. It also allows for the simultaneous activation of multiple mechanisms, perhaps explaining how music can produce complex, or mixed, emotional states.

Clearly, many of these mechanisms are dependent on subjective experience and individual memory. In addition, and perhaps most importantly to the current discussion, they may not be understood well enough to enable them to be modeled computationally. Therefore, the research in this dissertation will focus on the ideas about musical expectation, repetition, and boredom introduced above and related to the mechanism of musical expectancy.

3.1 Music Information Dynamics

More recently, researchers have used ideas from information theory to quantify Birkhoff's ideas of order and complexity [52, 10]. Many studies have used music as the aesthetic stimulus, and this has lead to the development of a field called Music Information Dynamics (MID). The "information" in MID refers to the *information content* proposed by Shannon [71]. Shannon's information theory provides two measurements which will become relevant to the discussion here [72]. The first, *information content*, refers to the number of bits needed to encode a piece of data. The second, *entropy*, quantifies the amount

of randomness or disorder in a signal. The measure described later in this section uses calculations of two forms of entropy in order to estimate the information content of a musical signal. Assuming that moments with high entropy - large amounts of randomness - are surprising to the listener, we can join these information-theoretical concepts with Meyer's theories of aesthetics. If we can model the time evolution of a signal according to its inner order and moments of complexity or novelty, then we can relate this model to ideas of aesthetic appreciation. In other words, as Meyer argued, a piece of music establishes its own inner set of rules and expectations, and - if we model the unfolding of the piece in terms of moments of conformity to or violation of these rules - we can potentially estimate the aesthetic content of the signal in terms of its order and complexity.

Many approaches to the calculation of MID use some type of statistical model in order to characterize the likelihood of current events given knowledge of past events. For example, the system described in [72] models both short-term and long-term musical memory. Both models operate on symbolic descriptions of music, using pitch as their main representation feature. The long-term model uses a database of 900 tonal melodies as training data, as a way to approximate what the authors call "typical human Western musical experience." The short-term model uses an identical algorithm to the long-term, but only learns from the current piece as it unfolds. By combining these two models, the probability of a given pitch occurring in a specific musical context can be calculated. Pitches with a high probability of occurrence will conform to a listener's expectations, while those with a low probability will violate those expectations. From this probability, it is possible to infer whether a given moment of music is surprising or familiar.

Other approaches, such as the IDyOM (Information Dynamics of Music Model) project [73] and the work of Abdallah and Plumbley [74], use Markov models to measure information content in symbolic representations of music. Analysis of minimalist music performed with these models has been shown to correspond well with human structural

analysis [74]. These representations work well with quantized, symbolic representations of musical works such as MIDI, but may not be able to deal well with timbral or microtonal / non-standard pitch material. In the case of a generative audio system, like those described later in this dissertation, much of the musical content may place an emphasis on timbre, and what pitch sequences there are may emphasize micro-tonality or continuous changes in pitch rather than discrete, equal-tempered intervals. The rest of this section will describe a measure of musical information which can be used with any extracted musical feature, and can be automatically calibrated to the characteristics of that particular feature.

3.1.1 Information Rate

Information Rate (IR) is a measure of the information content in a musical signal which operates directly on the audio signal, instead of a symbolic representation [45]. IR has been studied in relation to human judgments of emotional force and familiarity, making it particularly relevant for the goals of the current study [44]. IR quantifies to what extent the present moment can be "explained" by the past - and answers the question: "How much does the present repeat something that has been heard previously?" Recalling Meyer's ideas about musical expectation, IR describes whether the present moment of an audio signal conforms to or violates the expectations established in the prior portions of that signal, and can thus be related to aesthetic appreciation.

More specifically, IR measures the mutual information between the past x_{past} and the present x_n of a signal x:

$$IR(x_{past}, x_n) \quad = \quad H(x_n) - H(x_n|x_{past}), \tag{3.1}$$

where

$$x_{past} = \{x_1, x_2, \cdots x_{n-1}\}. \tag{3.2}$$

In (3.1), $H(x)$ represents the Shannon entropy of a variable x, representing the signal, and is given by

$$H(x) = -\Sigma P(x) \log_2 P(x), \tag{3.3}$$

where $P(x)$ is the probability distribution of that variable. $H(x|y)$ is the conditional entropy:

$$H(x|y) = -\Sigma P(x,y) \log_2 P(x|y). \tag{3.4}$$

A newer formulation of IR, introduced in [45], facilitates easier estimation from an audio signal, and is estimated using the equation

$$IR_{AO}(x_{past}, x_n) \quad = \quad C(x_n) - C(x_n|x_{past}), \tag{3.5}$$

where $C(x)$ is the length - measured in bits - of the output of a compression algorithm given the signal x as input. In other words, the IR at a given moment is the difference between two compressed versions of the input signal:

1. $C(x_n)$: A compressed version of x which does not consider the past of the signal - the "unconditional complexity."

2. $C(x_n|x_{past})$: A compressed version of x which does consider the past of the signal - the "conditional complexity."

Conveniently, the *Audio Oracle* (AO) algorithm can be used to model the signal and then create a compressed representation of this model. AO is an audio analysis method and representation of musical structure, and is based on the Factor Oracle (FO) string algorithm [75, 76]. AO has applications to both analysis and generation of audio [77]. During the construction of an AO, individual feature vectors (extracted from frames of an audio signal) are compared in order to detect similarities. The feature vectors are compared using a distance function, and if their measured distance is below a certain distance threshold, they

are considered to be identical for the purpose of representing the signal.

By using knowledge about the location and length of repeated patterns occurring throughout a signal, it is possible to obtain a lossy compression of the audio signal from the AO structure. Since it is possible for multiple suffixes to point to varying lengths of the same pattern, i.e. a musical motive or other recurring element which appears in multiple forms, the compressed version of the AO structure will only retain the longest of these repeated suffixes. The longest repeated suffix (LRS) of a state is the longest sequence of states leading directly up to state i that appears at least twice before i. The LRS of each frame is computed during the AO construction process. Compression of the AO is performed similarly to that presented in [78]:

1. First, the size of the coding alphabet is calculated. In AO, each transition from the first state, state 0, corresponds to the first appearance of a unique frame of audio. The number of unique frames / states is equal to the size of the coding alphabet. Therefore, the number of transitions from state 0 is equal to the size of the alphabet.

2. $C(x_n)$ can then be obtained by calculating $log_2(n)$ where n is the size of the alphabet from the previous step.

3. An array $K = [1]$ of encoding events is generated. For each frame i:

    - If no repetition of i is found, then the suffix link from i will point to frame 0. This is a new frame and it will be individually encoded. It is added to K.
    - Otherwise, if a suffix link to an earlier frame is found, and the *LRS* of that frame is less than the number of frames since the last encoding event, then the entire segment is encoded as a (length, position) pair.

The compressed version of the AO is the list of (length, position) pairs contained in K. The AO structure and its applications to machine improvisation and music analysis are explored further in Chapter 5.

Returning to (3.5), the IR can then be computed by calculating the difference between two entropies, H_0 and H_1, corresponding to $C(x_n)$ and $C(x_n|x_{past})$ [45]. These entropy values are calculated through an iterative process in which the locations and lengths of new states and encoded blocks are calculated, and stored as two intermediate functions. The first, C_{w0} counts the appearance of new states only, ignoring repetitions. The second, C_{w1} counts all encoded states. A third function, BL, stores the length of encoded blocks. The entropies H_0 and H_1 are calculated by taking the log of the cumulative sums of C_{w0} and C_{w1} respectively. H_1 is then normalized by the block lengths BL. The final IR result is then simply $H_0 - H_1$. The algorithm is shown in Algorithm 1.

Algorithm 1 Information Rate from Compressed AO

Require: *code* - compressed AO in the form of (length, position) pairs, and total signal length N.

1: Create arrays H_0 and H_1
2: **for** $i = 1$ to N **do**
3: $H_0 \leftarrow Log_2$(number of new states in *code* $(L == 0)$ up to i)
4: $H_1 \leftarrow \frac{Log_2(\text{number of all code-words up to } i)}{\text{length L of a block to which state } i \text{ belongs}}$
5: **end for**
6: **return** $IR = H_0 - H_1$

Since IR is related to the amount of information shared between the past and the present, i.e. how much of the present is a repetition of something which has come before, it can be seen as a measure of the balance between complexity and familiarity or reinforcement and violation of expectations. We can also reasonably assume a connection between IR and surprise. A low IR value corresponds to a high amount of novelty, and therefore surprise, while a high IR value corresponds to a repetition of something previously heard. As found in previous studies on computational aesthetics, such as Birkhoff's aesthetic measure, the most pleasing aesthetic experiences often rely on a careful balance between order and complexity.

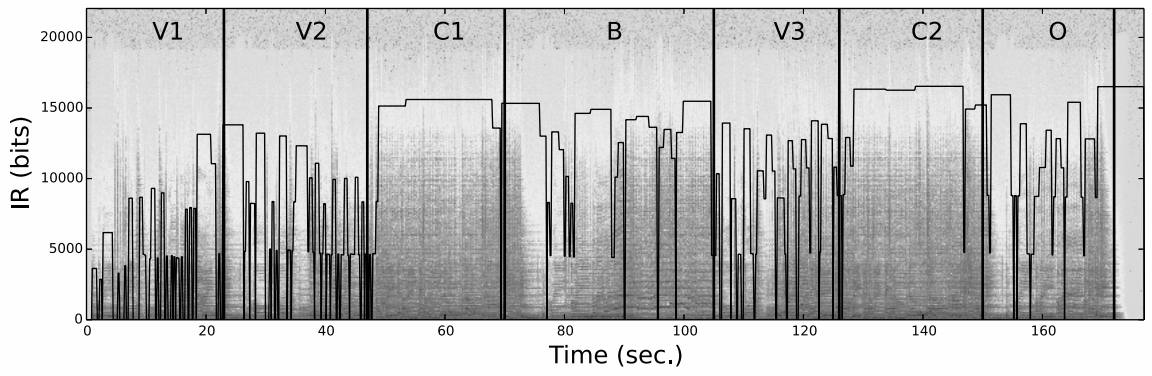

Figure 3.1: Music Information Rate function superimposed on spectrogram of Pink Floyd's *Arnold Layne*. IR was calculated using MFCC analysis frames.

3.1.2 IR Applied to a Musical Example

As an applied example of how an IR function can be related to real musical experience, consider the plot in Figure 3.1. This plot superimposes three pieces of information extracted from Pink Floyd's song *Arnold Layne* (1967), to compare IR with aspects of musical form. The first layer of the plot is a spectrogram of the audio signal, the second is the IR function, and the third is a high-level sectional analysis. The IR function was calculated using Mel-Frequency Cepstral Coefficient (MFCC) analysis frames, which give a concise description of the spectral envelope of the signal and are commonly used in speech and music processing [79].

This particular song was chosen for the variety of timbres produced by the instrumentation - electric guitar and bass, vocals, organ, and drums - and its clear formal structure. The sectional analysis is marked by vertical lines on the plot. Each section is labeled according to its function: verses with **V**, choruses with **C**, bridge material with **B**, and the outro with **O**. The overall IR value increases as the song progresses, which is expected: as more material is analyzed, it becomes more likely that each current moment can be explained as a repetition of something previously analyzed. Of particular interest is the way the different types of material are characterized by different behavior in the IR function. The verse material is extremely blocky, indicating a high amount of variety and novelty in the spectral envelope.

Contrasting with this, the choruses are much more stable, with both **C1** and **C2** essentially composed of single blocks - indicating very little change in the spectral envelope. Finally, the bridge **B** and outro **O** material are composed of slightly longer blocks than the verses, again indicating a higher amount of variety.

This analysis is consistent with the musical material during these sections. The verses are characterized by the gradual addition of various instruments: bass first, followed by drums, then organ, followed by arpeggiated guitar. This layering results in a continually changing spectral envelope. The choruses, on the other hand, feature the entire instrumentation, without any change in texture or envelope. The **B** material is characterized again by gradual layering, and expands into a brief instrumental passage in **B2**. Finally, the outro **O** material repeats vocal material heard earlier in the song, but with different layering and instrumental arrangement.

3.2 Discussion

This chapter provided an overview of the field of computational aesthetics. Many theories of computational aesthetics are influenced by Birkhoff's *aesthetic measure*, which describes the aesthetic value of a stimulus as the ratio between its order and complexity. Later researchers used information theoretic approaches to quantify the notions of order and complexity. Theories about how exposure and repetition influence aesthetic response were also introduced. First, the "mere exposure" effect was described, in which repeated exposure to a stimulus increases subjective affective response to that stimulus. This effect seems to be countered or balanced by a boredom effect, which begins to exert a negative affective response after a certain point. Second, the related idea of processing fluency was described. Processing fluency is tied to the ease with which a subject can understand or process a stimulus. It has been shown that greater processing fluency, which can be obtained through repetition or "priming," is tied with positive affective response. Greater ease of

processing leads to more positive response. There is again a countering influence, where the effect of stimulus complexity has a "U"-shaped curve. Too little complexity leads to fluency being attributed directly to the simplicity of the stimuli, while increasing complexity leads to fluency being attributed to the "beauty" or aesthetic merit of the stimuli and leads to a more positive affect. After a certain level of complexity, however, processing fluency will decrease, leading to a less positive overall response.

The field of Music Information Dynamics is particularly relevant to the ideas of computational aesthetics and exposure, and a particular approach - Information Rate - was described. IR is based on two measures of entropy in a musical signal, and considers the difference between the conditional complexity and unconditional complexity of the signal. These can be calculated by making two compressed versions of the signal: one which considers the past (conditional), and one which does not (unconditional). IR can be calculated from an audio signal in real-time, using the Audio Oracle algorithm, and gives a simple measure of to what extent the present moment in an audio signal can be related to past moments. Chapter 8 will describe an implementation of Audio Oracle, and its practical application to generative music.

Part II: Complexity and Computational Aesthetics

Chapter 4

Feedback and Complexity in Generative Systems

One of my primary compositional and research interests, both during my time at UCSD and before, has been the creation of systems which generate music with some level of autonomy. In particular, as laid out in Chapters 1 and 2, I am particularly interested in systems where sound synthesis and compositional logic are tightly coupled. My interest is in systems which generate audio through some type of synthesis, rather than those which deal with symbolic representations. This chapter will discuss a trio of generative systems and software instruments which function in a similar manner. These systems create a sense of autonomy and generativity in two ways: feedback and modular design. All three systems use some sort of feedback to create sonic material or structural parameters. They also all use modular design to facilitate the production of complex output, which - as described in Chapter 1 - is often thought to be a necessary component of a satisfying aesthetic experience.

The first section of this chapter will describe my use of digital and electro-acoustic feedback as a compositional resource, as evidenced in a pair of pieces, called *Gates No. 1* and *Gates No. 2*. Both pieces function as "improvisational instruments" - software programs which allow a performer to improvise within very specific constraints. In the case of these

pieces, formal aspects of the final composition are left almost entirely unspecified, while the character of the musical materials is determined by a specific and tightly constrained signal processing network. These pieces emphasize the use of feedback as a compositional resource, while leaving most of the organizational work up to the performer.

The second part of this chapter will discuss a generative system called *Feld*, which combines analog synthesis hardware with digital hardware control and signal processing to create a generative system. *Feld* does not require performer intervention, instead it constructs its own form in real-time, as a function of its own past output and a set of compositional algorithms. The complexity of the synthesis and signal processing network - as elaborated below - encourages a sense of emergent or self-organizing behavior. Little of the gestural or formal structure of the work is predetermined, instead arising from the momentary internal configuration of the network and compositional algorithms.

4.1 Feedback as Source Material - *Gates Nos. 1* and *2*

Digital and acoustic feedback have long been favored compositional materials of mine. As many composers have discovered, and as elaborated in Chapter 2, feedback systems are capable of generating interesting timbres with dynamic and surprising behaviors. In the following section, I will describe a pair of musical systems which make use of digital and electro-acoustic feedback. The systems are named *Gates No. 1* and *Gates No. 2*, after their methods of articulating rhythmic and phrase-level behavior from continuous feedback. In addition to the unpredictability of feedback, a desire to participate in free improvisation fueled my interest in leaving the final form of the output produced by the systems unfixed until the moment of performance. I will briefly describe the design of each system , as well as the ways in which compositional desires were intermingled with the dynamic results of the systems. Finally, I will discuss several flaws in the systems , and place them into the context of the research to be discussed later. It should be noted that the aim of this

section is not to describe the systems in great detail, but to give an overview of my own past involvement with feedback and generative audio systems, and to set up further discussion of digital feedback networks.

4.1.1 *Gates No. 1*

Gates No. 1 is a software improvisation instrument which uses of a pair of digital feedback networks as its sole source of musical material. An example network topology is shown below in Figure 4.1. The networks are loosely based on physical modeling wave-guides, and constructed out of two identical "rails". Each network is encouraged to resonate within a particular musical register, through particular settings of delay times and bandpass filtering. The delay times and filter center frequencies are allowed to vary randomly over time within predetermined ranges, ensuring that the output of each feedback network varies while remaining within the specified range. The ranges are left deliberately vague, corresponding to low, mid-range, and high frequencies. Each network is excited

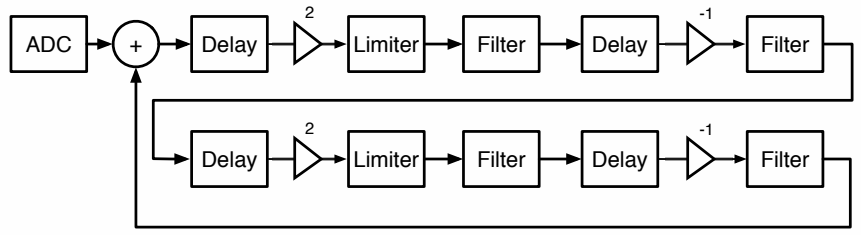

Figure 4.1: An example feedback loop from *Gates No. 1*. The loop contains time-varying delays, time-varying bandpass filters, and multipliers (ensuring the delay line never loses energy).

by input from the computer ADC or built-in microphone, ensuring variation and allowing the output of the system to re-appear at the input. (The goal is not really for the system to respond to audio input in any meaningful way, but simply to provide an interesting "random" excitation.) Since the feedback networks are time-varying, with changing delay times and filter center frequencies, it can be difficult to ensure stability. This problem is addressed in

a manner similar to that employed in [39], where a soft-clipping function is used both to ensure stability and for its interesting timbral characteristics. In *Gates No. 1*, stability is ensured using a look-ahead limiting function, ensuring that the amplitude over a window of samples stays below a specific threshold. In combination with a gain factor of two, shown in Figure 4.1, this structure forms a crude compressor, and boosts low amplitudes while constraining or reducing high ones.

Multiple points within each feedback network are used as tap locations, and the output of each tap is sent to one of a set of signal-processing modules. Each processing module applies a single, time-varying effect to the signal: amplitude modulation, ring modulation with either an oscillator or noise as the modulator, a delay-line based pitch-shifting effect, a chorus effect, and a very narrow bandpass filter. The parameters of the effects are modulated by reading from one of 50 pre-composed function tables. These functions are designed to be multi-purpose, and each may be used as the modulation source for a number of different effects over the course of a performance. The functions are collected into groups according to characteristics of their shapes - straight lines, curved lines, combinations of both, jagged contours, smooth contours, etc. Finally, each processing module also applies a randomly generated (with some constraints, discussed below) amplitude envelope to the input signal. This envelope serves to create rhythmic interest by gating the otherwise constant sound of the feedback network. The output of the processing modules is summed and passed through a reverberation and delay module, before being spatialized.

The composer/performer interacts with this system at a high level. Instead of specifying specific musical materials, a series of purposefully vague commands is used to trigger sounds. First, the performer selects which of the six processing modules should be activated. Then, he or she specifies a register and duration for the sound. As mentioned, the feedback networks are organized to produce low, mid-range, and high frequency sounds. When the performer specifies a register, the output of the corresponding feedback network

is used as input to the selected signal processing module. The duration parameter is sent to the amplitude envelope generation portion of the selected signal processing module. The amplitude envelopes can be short, medium-length, or long in duration. The processed output of the feedback network will be enveloped with a function designed according to the duration range specified. Finally, the group of modulation functions to use for effect parameter modulation is selected, and a particular function is chosen at random from that family and applied to the parameters of the processing unit. A performance consists of the performer triggering events by selecting these sets of parameters, and allowing the instrument to fill in the details. The challenge is to create a musically effective form with materials which can vary widely between performances and even within a single performance.

4.1.2 *Gates No. 2*

Gates No. 2 takes the strategy of processing the output of digital feedback networks and moves it into the electro-acoustic realm. The instrument was designed for performance at the eight-channel composition studio at the University of Wisconsin - Milwaukee, and uses eight discrete speakers and a single microphone placed in the center of the space. The microphone signal is digitized, split, and fed into eight identical signal processing channels, each of which is output directly to one of the speakers. Each channel has a gain of 20 (ensuring audible feedback), a set of time-varying filters (switchable, by the performer, to either bandpass or comb filters), a short delay intended to function as a phase shift rather than an audible echo, and a so-called "formant shifter," borrowed from David Tudor's *Untitled*, consisting of the product of the output of high-pass and low-pass filters with identical cutoff frequencies [80]. These signal processing modules are time-varying, and their automation is derived from analysis of the fundamental frequency and amplitude of the signal at the microphone at previous moments. The signal processing is designed to disrupt and animate the feedback which occurs between the microphone and speakers in order to avoid static,

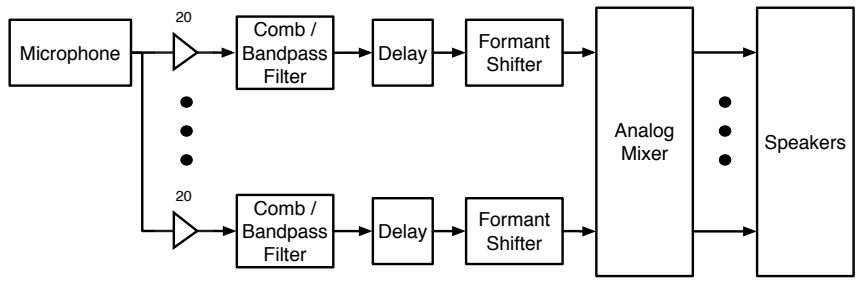

Figure 4.2: System structure of *Gates No. 2* showing both software and hardware components.

high-amplitude drones. The final output of each channel is fed through a time-varying low-pass filter before being output to a single speaker.

In performance, the final output gain of each channel (and therefore at each speaker) is controlled by a performer manipulating the faders of an analog mixer. The use of an analog mixer has the benefit of introducing a clipping mechanism to ensure that feedback does not grow unbounded or cause digital clipping. Through this simple interface, the performer has control over form, dynamics, and spatialization - all of which are interdependent. Depending on the output gain of a particular channel, it may produce sounds ranging from pure, sweeping tones to rhythmically articulated noise. The diverse material produced by the different output channels is blended at the microphone, and processed again. The microphone signal is also analyzed for fundamental frequency and amplitude, and then both the signal and the analysis are used in the generation of the next moments as the signal becomes material to be processed and the analysis informs the parameters of the processing modules. Although the performer has the ability to selectively audition certain channels, the output at any given moment is dependent on the past in a way which is impossible to fully predict.

4.1.3 Discussion and Further Work

Both *Gates No. 1* and *Gates No. 2* successfully use feedback as their sole source of sound material. Where *Gates No. 1* used fully digital feedback, *Gates No. 2* used a hybrid digital/analog feedback network. Both systems avoided static feedback by introducing disrupting signal processing elements - time-varying delays and filters, modulation and feedback. This resulted in dynamic and surprising sounds with characteristics unique to each system. Where *Gates No. 1* used a limiting/compression scheme to maintain system stability, *Gates No. 2* relied on the clipping inherent in the analog mixer circuitry.

Although the systems provide a satisfying performance experience, retaining distinct timbral and gestural identities while allowing flexibility and improvisation, they are not fully autonomous or generative audio systems. Due to their open-ended design, much of the final musical form is left undecided until the moment of performance. The systems do not function on their own - if left unmanaged, no music will emerge. Both systems need the input from a performer to trigger events. There is also a divide (particularly in *Gates No. 1*, and the use of pre-composed function tables) between pre-composed and emergent material. In a more autonomous generative audio system, musical materials - like rhythm, gestures, and motivic behavior - should be closely tied to the sound generation procedure itself, with musical form emerging from the synthesis algorithm employed. The next section describes another system which employs a different method of feedback as a method of creating formal autonomy.

4.2 Generativity and Complexity - the *Feld* system

The software systems described in the previous section employed digital and electro-acoustic feedback as a means of generating surprising and dynamic material. This approach was successful in that it created works with unique and unified sounds and gestures. However,

these works always required some human intervention to audition, disrupt, or otherwise frame the generated sounds in a larger musical context. In order to create a system from which interesting music might arise spontaneously - that is, without the influence of a human performer - it seemed necessary to design a system with a higher level of complexity and to introduce opportunities for other types of indeterminacy than were previously employed.

This section will describe my next attempt at an autonomous generative system. The system, called *Feld*, couples a custom hardware analog synthesizer with a complex DSP network, drives synthesis and signal processing with a suite of compositional algorithms, and performs feature extraction on its own output in order to inform the development of the work. There are many theories about ways in which the computational worth of an art object might be objectively measured, but almost all of them involve a ratio between order and complexity. During the design of *Feld*, special care was taken to ensure that this balance lead to a satisfying aesthetic experience. I will describe below the ways in which this balance was designed into the system.

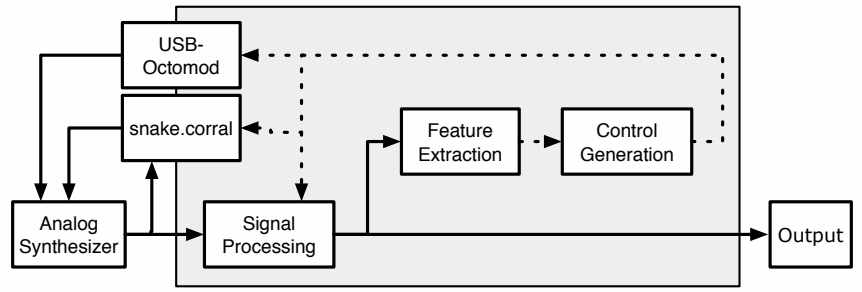

Figure 4.3: A high-level block diagram of the Feld system feedback loop. The laptop controls the synthesis hardware while also analyzing and processing the output of the hardware. The gray box indicates software components. Solid lines indicate audio signals, dashed ones indicate control signals.

4.2.1 System Structure

The *Feld* system consists of both hardware and software components, arranged into a feedback loop. The structure is shown in block diagram form in Figure 4.3. This section

will describe the hardware and software components and arrangement of the system.

4.2.1.1 Hardware

The hardware used in *Feld* is entirely hand-made, and includes both custom designs by the author and circuit boards purchased from small designers around the globe. In particular, I developed two custom devices - the *USB-Octomod* and *snake.corral* - which were used to interface between the computer and hardware systems [81]. An additional custom device, the *tabulaRasa*, was also designed and employed in the system. The motivations and design of these devices are described in greater detail in Appendix B, so as not to disrupt the narrative flow of this chapter. In addition to these custom devices, the *Feld* system uses conventional oscillators, low-frequency oscillators (LFO), a state-variable filter, voltage-controlled amplifiers, and a ring modulator.

4.2.1.2 Software

The system also makes use of a software patch (implemented in Max/MSP) which handles spatialization, signal processing, and hardware control. Much of the gestural and timbral information in the work is generated or modified through the software. The patch communicates with an external eight-channel audio interface as well as the USB-Octomod and snake.corral. This section will describe the software, starting with components which generate control signals (control-rate components), and then discussing the signal processing.

4.2.1.2.1 Control-Rate Components There are two major control-rate components to the *Feld* system, each contained in a sub-patch. The first allows the composer to specify a base duration and a set of proportions which are used to determine many elements of the form and sectionality of the piece. The proportions are made into a fractal using one level of recursion - the set of proportions is reflected on both large and small scales. In other

words, the proportions manifest in both the progression from section to section and on the average duration over groups of sections. The sub-patch also contains a timer which steps through the durations and transmits a pulse to other areas of the *Feld* patch at the onset of each section.

The other major control-rate component generates and assigns parameters to DSP modules elsewhere in the patch, and also generates and transmits data to the USB-Octomod and snake.corral hardware. The DSP parameters generated in this sub-patch will be discussed in greater detail in the next section.

The first function of this sub-patch interfaces with the snake.corral matrix. This code generates an evolving set of matrix patch configurations. The algorithm was inspired by John Cage's use of charts to generate slowly evolving indeterminate material during the composition of *Music of Changes* (1952) [18]. At each sectional boundary, two preset configurations are selected at random - one for each matrix in the snake.corral. At the same time, there is a probability that a selected preset will be replaced. This allows for the system to have a limited number of possible configurations at a particular time, while also allowing for the set of configurations to change over time. A balance between repetition and novelty is created, where some configurations will be repeated while new configurations will gradually emerge. After a configuration is selected, it is transmitted to the snake.corral hardware over a USB serial connection.

The second function of the sub-patch is to generate eight modulation signals that are sent to the synthesis hardware via the USB-Octomod. The eight modulation signals are constructed from the output of a set of six control-rate modulation generators:

- Random Ramp / Step - Generates smooth / instantaneous transitions between values selected from a uniform distribution.
- Random Walk / Random Walk Slider - A standard random walk, with instantaneous / ramped transitions between values.

- Cycle - Iterates through a list of values in sequence, with a small probability that a given value will be replaced after it is used.

- Bag - Randomly draws from a set of values, with a small probability that a given value will be replaced after it is used.

Each modulation output channel has an independent set of six generators. There are two parameters for the modulation generators, “Switching Rate” and “Modulation Rate.” The “Modulation Rate” parameter determines the speed with which new values are drawn from each of these generators. Each of the eight final modulation signals is constructed by an algorithm which occasionally switches between the output of these generators, at a rate controlled by the “Switching Rate” parameter. The composite output is then transmitted to the USB-Octomod via a USB serial connection.

4.2.1.2.2 Signal Processing Components Four channels of audio are used as input to the software system, and are taken directly from four matrix outputs on the snake.corral. The simplicity of the small hardware modular system can cause it to function with only a few basic rhythmic or gestural motives at a time. *Feld* includes a few strategies for avoiding monotony which may occur. For example, a single LFO may be patched to many modules simultaneously, causing similar modulation patterns to be heard across multiple output channels. The *Feld* patch applies an independent delay - randomized at each sectional boundary - to each of the four input audio channels, creating polyphonic textures and canonic effects. Each channel is heavily processed, as described below, so these canonic effects are rarely heard as such. Instead, particular features (a change in LFO speed, for example) of the input signals will appear in short succession, but with large amounts of variation. Listeners hear relation, but not repetition or imitation.

Each channel then passes through its own signal-processing network. Each of the four networks is constructed of four “nodes,” each consisting of a subset of 14 individual

effect modules, including amplitude modulation, comb filtering, delay, single-side-band modulation, delay-line frequency modulation, soft clipping, and rhythmic gating, to name a few. An example node is shown in Figure 4.4. The four nodes are reordered at sectional boundaries, to ensure timbral and gestural diversity between sections. In one section, an amplitude modulated signal may be processed by a comb filter, while in the next a comb filtered signal is processed by the AM module. As described in [21], this change in order of operations can have a significant effect on the output, since many of the effects are non-linear and all are time-varying. Effect parameters are automated in the same way as the USB-Octomod channels, by switching between independent parameter generation algorithms.

Within each node, the modules are connected in networks, many of which have multiple branches. These branches can be automatically switched on or off using rhythmic gating modules. Each section of the generated formal structure of the system's output uses only a subset of available effect modules, bypassing the rest. Even if a section reuses a network topology, the resulting sounds will vary in interesting ways. Each active module is also occasionally bypassed within each section according to a rhythm generation module, which again switches between multiple rhythm generators, adding polyphonic timbral variation to the signal as effects are switched in and out of the signal path. The clean input to the processing network is then blended with the processed version, with the weighting between the two signals determined randomly for each section. The combination is passed through a delay effect and an automated stereo panner. Finally, the stereo signal is passed through a reverberator and out to the DAC and a signal analysis module.

4.2.1.2.3 Analysis Components The analysis module computes three perceptual descriptors extracted from the output signals. The three descriptors are "loudness," "brightness," and "noisiness," and are calculated using Tristan Jehan's `analyzer~` object[1]. During each

[1] Available at http://web.media.mit.edu/ tristan/maxmsp.html

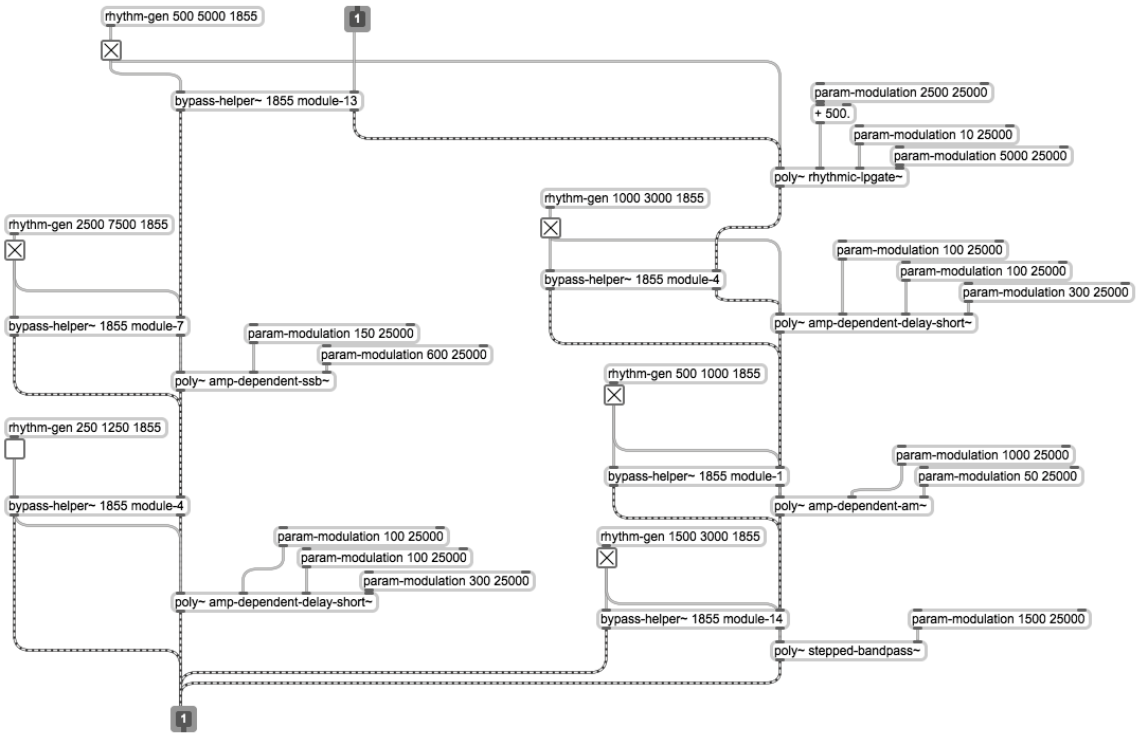

Figure 4.4: Screenshot showing an example *Feld* signal processing node. The top object, `bypass-helper~`, divides the network into two branches. Each branch passes through a few different effects modules, which are bypassed according to a rhythm generator, and modulated independently. Finally, both branches are summed and output to the next node.

section, a running average of each descriptor is constructed. At the end of the section, the difference between the current section and the previous is calculated. These difference values are scaled and assigned - in a rather loose manner - to the “Switching Rate” and “Modulation Rate” parameters used in determining the USB-Octomod modulation signals for the next section. An inverse relationship is used, so that small contrasts between features across sections result in more rapid and varied modulation of the hardware. The system favors contrast, and attempts to introduce high levels of activity when the level of sectional contrast is low. The result is a form of negative feedback, and the system tends toward self-regulation and settles in a state in which sectional contrast is emphasized.

4.2.2 Order and Complexity in the *Feld* system

As mentioned in Section 4.2, the *Feld* system attempts to create balance between order and complexity in a few ways. This approach is based on the assumption that the aesthetic worth of a stimulus is related to the amount of order and complexity in that stimulus. Perhaps the most famous example of this is Birkhoff's *aesthetic measure* [53], given as

$$M = \frac{order}{complexity}, \tag{4.1}$$

but these theories have also been applied more loosely by composers of electronic music [35] and computer art [82]. Later researchers have found that there is an inverted 'U'-shaped curve in aesthetic preference. In other words, listeners prefer neither too much complexity nor too much order - the "best" experience is somewhere between [83]. This section will describe the ways in which complexity is constrained or otherwise given order in the system design, sound sources, and large-scale form of *Feld*.

First, sound synthesis is performed using a modular synthesizer which, by nature, is limited by the number and type of modules in the system. Rather than the "unlimited" sonic resources available through computer synthesis, *Feld* creates complexity through very limited means. Complex routings and modulation help to produce surprising output, but the system itself provides a significant constraint and sense of order.

Second, the software signal processing chains (described in Section 4.2.1.2.2) consist of many distinct algorithms, but can only be arranged in a few ways. Only a subset of the algorithms is used during a section of the piece. Complexity arises from the variety and number of possible algorithms, and their continually automated parameters. Conversely, the limited number of arrangements along with the rule that only a few algorithms are used during each section help to constrain the complexity and give the listener a sense of order.

Third, the modular synthesizer is automatically re-patched according to a limited

but evolving group of presets. By constraining the synthesis system to a small number of possible configurations, particular timbres and behaviors will reappear, creating a sense of order, repetition, and familiarity. On the other hand, the fact that presets from the group are gradually replaced introduces an amount of complexity and surprise.

Fourth, both the hardware system and the software DSP modules are given a set of modulation control signals. These signals are generated with a variety of random algorithms, but are constrained with higher-level parameters like "Switching Rate" and "Modulation Rate" (described in Section 4.2.1.2.1) and minimum/maximum values which control the content and rate of change of their output.

Finally, the self-analysis portion of the system described in Section 4.2.1.2.3 attempts to enforce contrast between order and complexity. If the previous section was judged to be too similar to the current, then the system attempts to create more diverse material for the next section. By introducing higher levels of parameter modulation and rhythmic activity, the system is more likely to find new and surprising or contrasting material.

4.2.3 Performance and Presentation

The *Feld* system was designed, in part, to experiment with long-duration works which evolve in a continual fashion and avoid repetitions. Each performance of *Feld* begins with a small number of pre-compositional decisions (described in greater detail below). After these decisions are made, the system is set into motion and the music evolves from those initial conditions. The hope is that these constraints, along with the framework of compositional assumptions provided by the system itself, will provide enough order to keep the complexity of the output from completely overwhelming listeners. This first part of this section will describe the pre-performance decisions and their impact on the longer work. The second will discuss the motivation behind the long duration format of the work.

4.2.4 Performance Decisions

There are two primary decisions which must be made before *Feld* can be performed. First, a patch must be chosen for the modular system. Generally, most of the synthesis modules are connected in some way to the inputs and outputs of the pucktronix.snake.corral, allowing for significant reconfiguration of the system to be performed instantaneously at sectional boundaries during the piece. Some module connections may be left "hard-wired" in order to lend some unique characteristics to a given patch. For example, the user may choose to connect the modulation rate control inputs of an LFO to each of the oscillators, and then connect each of the LFOs to the outputs of the switching matrix. In this way, the system is unable to modulate the oscillators directly, and can only affect their frequency by modulating the LFOs. These more specific configurations help to lend a unique sound and behavior to a patch. The second decision which must be made is a set of durational proportions and a base duration, as described above. These proportions are automatically transformed (with reference to the base duration) into a longer, fractal set of specific durations. These durations are used to designate sections within the longer work. These two decisions provide some control over the output of *Feld.* By embracing constraints and allowing for these simple pre-performance decisions to be made, the composer is able to balance the sheer complexity of the possible output - resulting from the reconfigurability of the system, the density of textures and events, and the use of multiple simultaneous compositional algorithms - with an amount of simplicity and order.

4.2.5 Listening Experience

As mentioned, a specific aesthetic and compositional interest for this work was to experiment with longer durations. During my time teaching an undergraduate course on American experimental music, I became interested in the way the listening experience changes over the course of a long duration. For example, while listening to a long string

trio composed of slowly unfolding three-note chords, the listeners' interest may first be focused on ideas about musical progression, defined by pitch, harmony, and rhythm. Over time, however, these elements tend to recede in importance while details of timbre and space become more prominent. Small changes in parameters eventually take on a greater significance.

My aim with *Feld* was to create a situation in which the listener is forced to attune to minute musical details on a local scale. The piece is a series of exciting and dynamic moments, organized into a simple form characterized by a long duration segmented into a series of contrasting sections. By limiting musical progress, enforcing contrasts, and prioritizing complexity, the piece encourages attention to moment-to-moment gestural and timbral complexity, while discouraging focus on form and progression.

Feld was recorded and presented as a set of three hour-long recordings.[2] Each recording was composed of a single "take" - the direct recorded output of the system. No editing was used, aside from trimming the ends and adding a fade at the beginning and end of each recording.

4.3 Conclusion

Both the *Gates* and *Feld* systems make use of feedback in the production of generative music. Where the *Gates* systems used digital and electro-acoustic feedback as their source of sound materials, *Feld* used a less direct form of feedback - analyzing its own output, and using the analysis results to inform future musical material. Both types of feedback provide interesting opportunities for generative systems.

In terms of autonomy, the *Gates* systems are significantly more dependent on a human performer than is *Feld*. Both *Gates No. 1* and *Gates No. 2* rely on a performer to directly trigger sonic events and also to provide a large-scale shape to the music. This is

[2] Available at `http://music.gregsurges.com/album/feld`

not the case in *Feld* - instead, the user provides a set of numbers which are then used to determine the durations of sections throughout the work. Following that limited interaction, the musical work unfolds according to the algorithmic procedures defined in the software. While *Feld* makes an attempt to pay attention to its own output, this ability is rudimentary at best and makes no real attempt to relate the signal analysis to human perception or expectation.

Unfortunately, both *Gates* and *Feld* fail to achieve any tight coupling between the generative procedure and the synthesis algorithm. A true generative audio system would make little to no distinction between the two - the synthesis algorithm itself would produce interesting and dynamic variations. While both systems are capable of producing compelling music over moderate to long durations, there is room for further development. Whereas the *Gates* systems function more as a instruments than autonomous performers, *Feld* primarily functions under the control of a set of pre-designed compositional algorithms with minimal adaptation to its previous output. Both fall short of being fully autonomous systems which make music according to principles of human expectation and interest. The problems described here will be addressed in greater detail in Chapter 8.

Chapter 5

Machine Learning and Computational Aesthetics in the PyOracle System

5.1 Introduction

This chapter describes PyOracle, a new machine improvisation and analysis system in the family of software built using the Factor Oracle (FO) and Audio Oracle (AO) algorithms [84, 85]. PyOracle was introduced by the author in [77], and is the first software to use AO - a graph structure built using features derived from input audio in the context of machine improvisation. The AO algorithm was described in Chapter 3. AO is useful for both analysis and generative purposes, and these uses will be described in this chapter. Though PyOracle is not a generative *audio* system, it will be useful to describe its applications to generative music and analysis.

The PyOracle project can be divided into two parts: PyOracle Analyzer and PyOracle Improviser. PyOracle Analyzer is a library written in the Python programming language for AO analysis of music and calculation of music information dynamics [86]. PyOracle Analyzer also provides some generative music functions, but is primarily intended for offline work. PyOracle Improviser embeds PyOracle Analyzer into the Max/MSP programming

environment, enabling real-time analysis of audio input and generation of new musical content based on that input. PyOracle Improviser is conceptualized as a software improvising partner which learns elements of a performer's style and generates a real-time audio accompaniment. In this context, the AO algorithm is used to determine the repetition structure of an improvisation as it unfolds. This repetition structure is then used to recombine the original audio material into new, but related material.

The PyOracle system makes use of ideas from the field of Music Information Dynamics, in order to determine the best model of an arbitrary input signal. A measure called Music Information Rate (IR) is used to measure the amounts of complexity and repetition in the signal over time, and can be used to find the ideal AO model.

PyOracle Improviser provides some unique features for enabling composition and structured improvisations. Constraints, probabilities, and other parameters can be modified in real-time or according to a predefined script. This chapter will discuss the use of PyOracle Improviser in a compositional context, and suggest some directions for further work in this area. The chapter will conclude with a description of an implementation of the AO algorithm as an external object for Pure Data and Max/MSP.

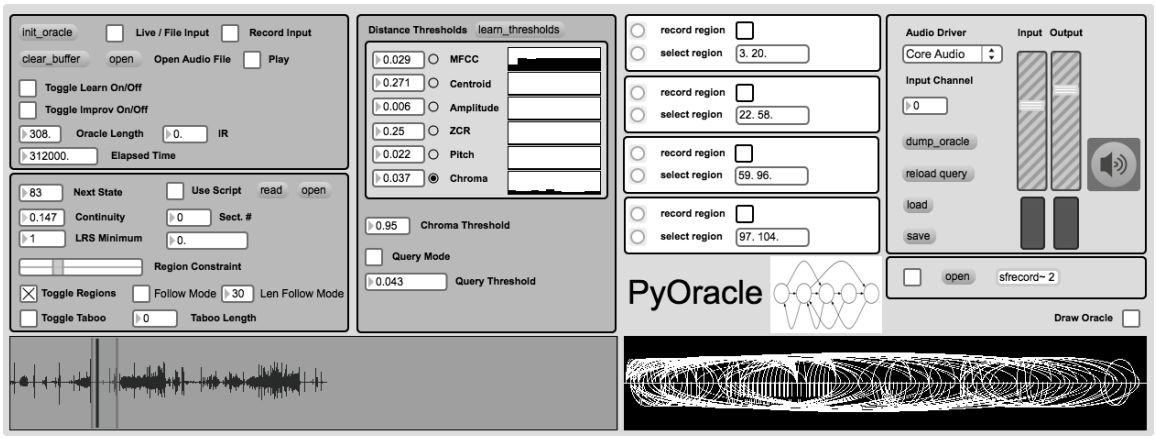

Figure 5.1: The PyOracle Improviser interface.

5.2 Machine Improvisation

Machine improvisation is a field of research wherein computer programs are designed to function as improvisational soloists or partners. In a situation where the software functions as part of a duo with a human performer, the software often receives musical input from the performer in the form of MIDI or audio, and responds appropriately according to some stylistic model or other algorithm. An important example is George Lewis's *Voyager* (1988), described by the composer, in [87] as a "virtual improvising orchestra."The *Voyager* software receives MIDI input derived from an instrumental audio signal, analyzes it, and produces output with one of a set of many timbres. *Voyager* is capable of both producing variations on input material and generating completely new material.

The Continuator project uses variable-length Markov chains to generate new continuations from a MIDI input stream [88]. The MIDI stream is parsed and a tree structure is constructed. As further MIDI input arrives, the tree is traversed to find continuations of the input. If no continuation is found, the next event is randomly chosen. The use of a weighted fitness function enables control over the "sensitivity" of the machine improvisation to the current musical context of the input. The authors also present some interesting ideas for structured improvisations using the Continuator system.

The Jambot, a flexible improvising system, was introduced in [89]. The Jambot attempts to combine "imitative" and "intelligent" behaviors, using a confidence measure to trigger switching between the two behaviors. The authors demonstrate this confidence-based switching method with the example of a beat-tracking behavior. If the system has a high confidence in its beat-tracking ability, given the current musical context, it will function more "intelligently" - producing more novel material. On the other hand, if the system is unconfident in its current model of the beat, it will switch to an "imitative" behavior until it regains confidence.

Many previous machine improvisation systems use symbolic music representations,

such as MIDI, which can have some advantages, but can also fail to capture significant musical information, such as timbre. Additionally, by their very nature, these symbolic representations require quantization of musical features. For many features, such as those related to the harmonic structure of a sound, it is often unclear how best to quantize the features - a "one size fits all" approach will often fail to accurately represent many sounds. The Audio Oracle algorithm, described in the following section, attempts to address these problems.

5.3 Audio Oracle

Audio Oracle (AO) is an audio analysis method and a representation of musical structure, and was introduced in Chapter 3. The description of the AO structure will be expanded upon here, focusing on its application to real time machine improvisation. AO extends previous methods which used Lempel-Ziv and Probabilistic Suffix Trees, described in [75]. The AO algorithm is based on a string matching algorithm called Factor Oracle (FO), extending it to audio signals [76]. AO accepts an audio signal stream as input, transforms it into a time-indexed sequence of feature vectors (calculated from a moving window on the time-domain signal), and submits these vectors to pattern analysis that attempts to detect repeating sub-sequences or factors in the audio stream. Mathematically speaking, the AO generalizes the FO to partial or imperfect matching by operating over metric spaces instead of a symbolic domain. In other words, AO does not require the same kind of quantization of input that FO does. One of the main challenges in producing an AO analysis is determining the level of similarity needed for detecting approximate repetition. This can be done using ideas from information theory applied to the structure of an AO. The "listening mechanism" of the AO tunes itself to the differences in the acoustic signal so as to produce a optimal representation that is the most informative in terms of its prediction properties. This "tuning" process will be described in greater detail below. Like the FO-based method, AO generates

an automaton containing pointers to different locations in the audio data that satisfy certain similarity criteria. The resulting automaton is passed later to an oracle compression module that is used for the estimation of Information Dynamics, as will be described in the following section.

The structure of an AO segment is shown in Figure 5.2. Similar to the Factor Oracle algorithm, forward transitions (upper arcs) correspond to states that can produce *similar patterns* with alternative continuations by continuing forward, and suffix links (lower arcs) correspond to states that share the *largest similar sub-clip* in their past when going backward.

During construction, the oracle is updated in an incremental manner, allowing for real-time construction. The algorithm iteratively traverses the previously learned oracle structure, jumping along suffix links, in order to add new links. A more detailed treatment of the AO construction algorithm is given in [75].

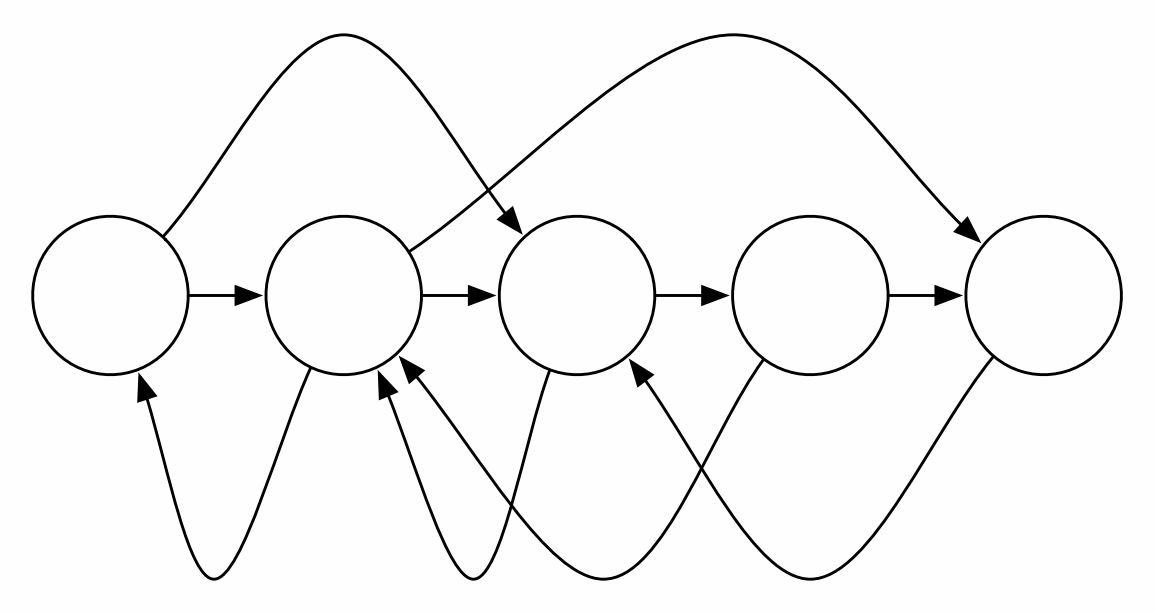

Figure 5.2: An example of an Audio Oracle structure.

5.4 Computational Aesthetic Theories

One of the novel properties of PyOracle Analyzer is that it performs analysis of music in terms of signal complexity and repetition structure during the learning stage undertaken in preparation for improvisation. A link between statistical properties of the signal and human perception, was initially established in [44], and motivates the current approach to using AO for composition.

5.4.1 Music Information Dynamics and Information Rate

Music Information Dynamics is a field of study that considers the evolution of the information content of music [74, 72]. The changes in the information content of a piece of music over time are often assumed to be related to structures captured by cognitive processes such as the forming, validation, and violation of musical expectations [46]. In particular, a measure called Information Rate (IR) has been studied in relation to human judgments of emotional force and familiarity [44]. The process of obtaining the IR function of an AO analysis was detailed in Chapter 3.

Recalling the AO algorithm described above, we can use the notion of balancing complexity and familiarity to solve the problem of how to determine the best model of the signal. The ideal model should represent the complexity of the signal, while also capturing repetitions that occur. If we consider IR to be related to the amount of information shared between the past and the present, i.e. how much of the present is a repetition of something which has come before, we can relate IR to the balance between complexity and familiarity. We can also draw a connection between IR and surprise. A low IR value corresponds to a high amount of novelty, and therefore surprise, while a high IR value corresponds to a repetition of something previously heard. As found in the previous studies on computational aesthetics, the most pleasing aesthetic experiences seem to rely on a balance between order and complexity. This has a practical application in determining the best distance threshold

to use during AO construction.

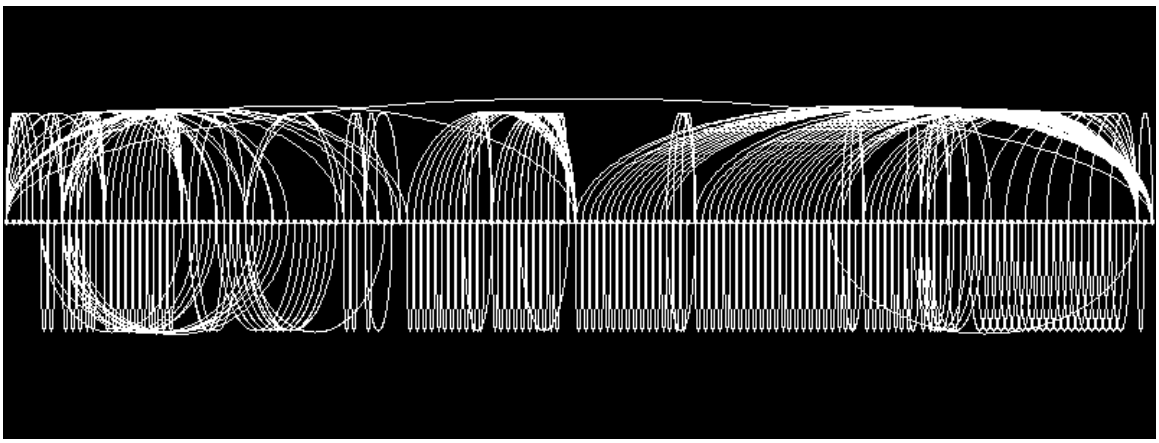

Figure 5.3: The ideal AO model. Note the balance between occurrences of transitions (upper arcs) and suffixes (low arcs).

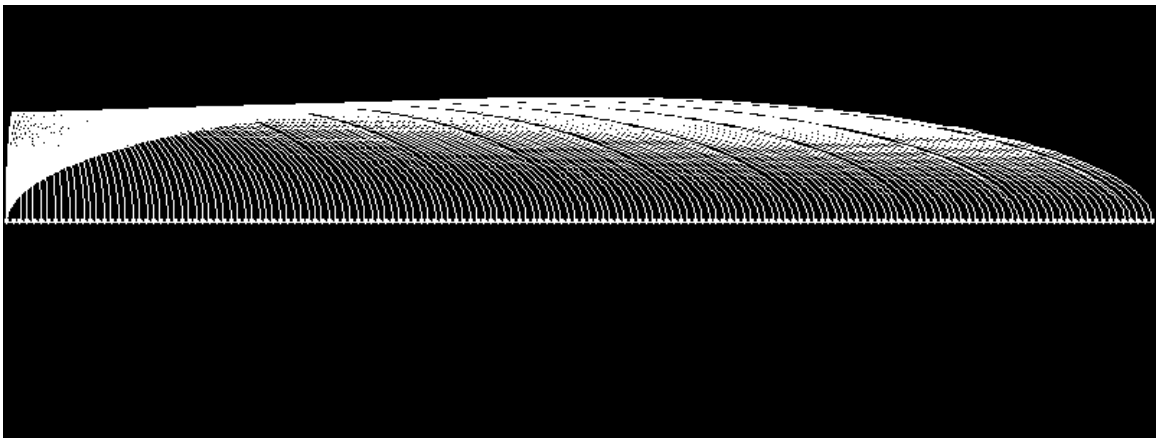

Figure 5.4: An AO with the distance threshold set too low. Each state is considered unique.

5.4.2 Tuning the AO Algorithm

A distance threshold is used to detect similar states during the construction of the AO. If the distance between two states is found to be lower than the threshold, the states are determined to be similar. For a more intuitive understanding of this distance threshold, IR, and how they relate to the AO structure, consider Figures 5.3 - 5.5. These figures

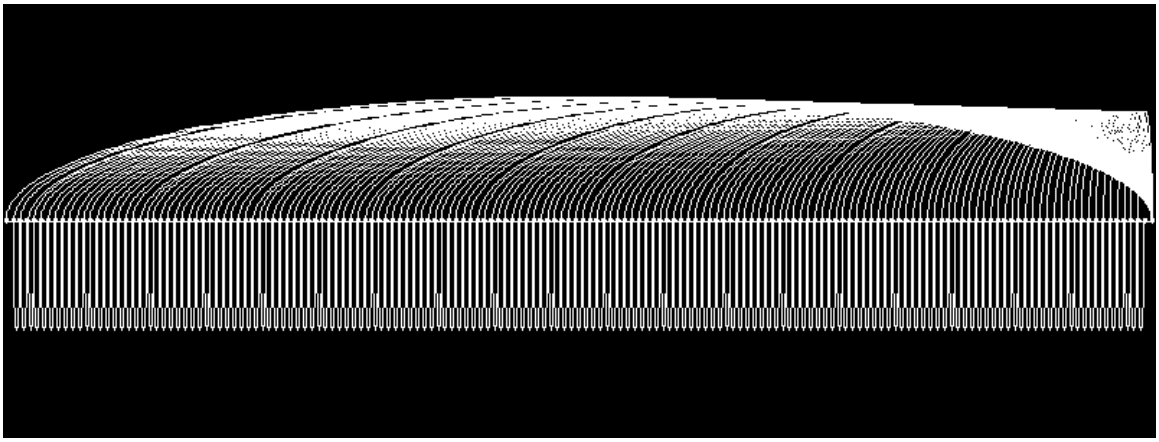

Figure 5.5: An AO with the distance threshold set too high. All states are considered repetitions.

demonstrate the differences between an oracle constructed using a well-chosen distance threshold and those constructed where the threshold is too low or too high. The audio signal analyzed here is the complete fourth movement of Prokofiev's *Visions Fugitives*, a short composition for solo piano. The first oracle, shown in Figure 5.3, is considered the optimal oracle. It is assumed that a good AO analysis will capture as much as possible of the mutual information between the past and present of the signal, and therefore an analysis with a high total IR is preferred. Through an iterative process, the total IRs obtained from using each of a range of possible distance thresholds are compared, and the particular threshold which maximizes total IR across the IR function is selected. Figure 5.6 shows the total IR as a

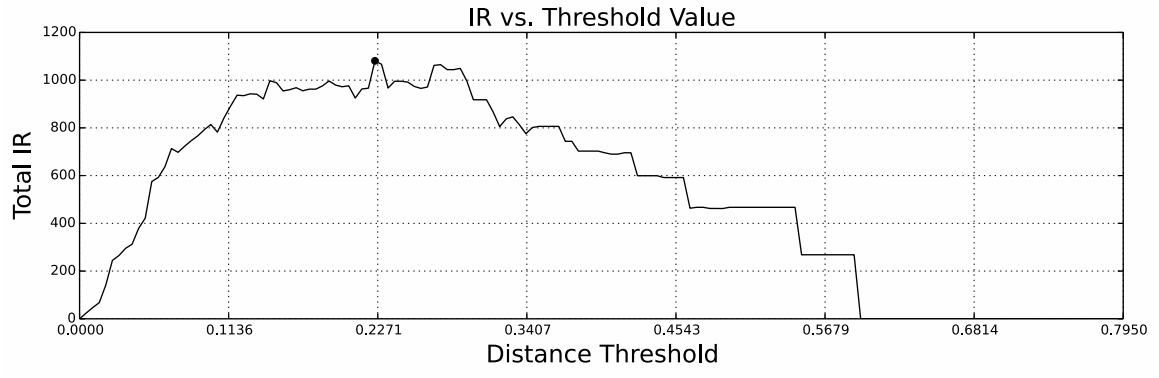

Figure 5.6: Total IR as a function of Distance Threshold for Prokofiev piano work.

function of the distance threshold. The peak of this function yields the optimal AO, though it is often worth studying oracles formed using secondary peaks as well. In the case of the Prokofiev work, the optimal distance threshold was found to be approximately 0.22, yielding a total IR of 1081 bits. Figure 5.4 shows an oracle where the distance threshold was too low. In this case, no similarity between any frames was found, and all frames were considered to be new. Each frame has only a link from state 0, indicating that this is its first appearance in the sequence. The algorithm has determined the input to be a "random" sequence, where all frames are unrelated and no repetitions or patterns occur. Conversely, Figure 5.5 shows an oracle where the distance threshold was set too high, resulting in dissimilar frames being lumped together, and producing an oracle where all frames were considered to be repetitions of the first. Each frame has a suffix link to the previous frame, indicating this repetition.[1]

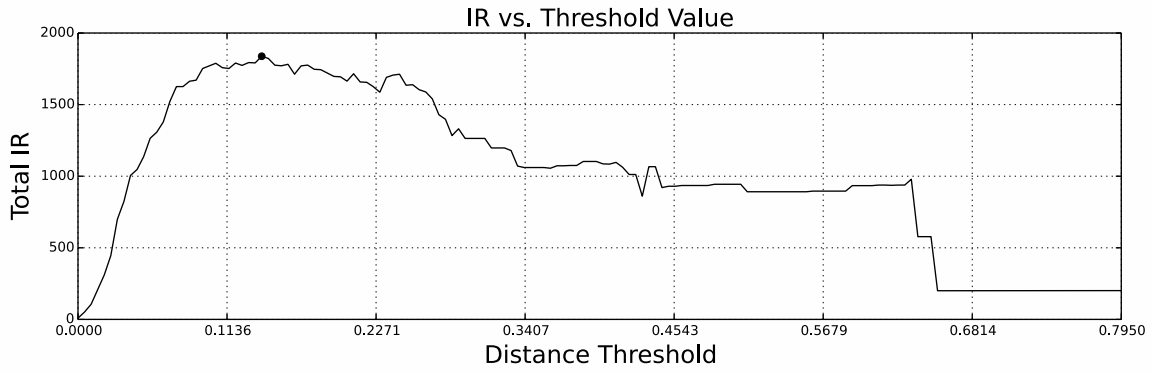

Figure 5.7: Total IR vs. Distance Threshold for Shakuhachi recording.

To understand how this threshold selection process "tunes" the oracle to its input, consider the difference between Figures 5.6 and 5.7. As described above, Figure 5.6 shows the total IR as a function of the distance threshold for the Prokofiev piano work. Figure 5.7 shows a similar plot, but for a recording of a solo shakuhachi. The underlying feature for both plots was Mel-Frequency Cepstral Coefficients (MFCCs). The optimal threshold for the Prokofiev was 0.22, while that of the shakuhachi performance was 0.14. This corresponds

[1]Note that in these diagrams, suffix links to the 0th frame, present when a new frame is detected, are omitted for visual clarity.

to the oracle adjusting to the more subtle timbral variation in the shakuhachi recording. A lower distance threshold was required to detect meaningful distinctions between frames with smaller variations. Both the PyOracle system described here and the *AAS-4* system described in Chapter 8 use an implementation of this training process.

5.5 Music Generation with PyOracle Improviser

AO can be used in a generative context, to produce real-time variations on the learned musical structure. PyOracle Improviser aims to facilitate experimentation and help simplify composition with this algorithm. In order to understand how one might interact with PyOracle Improviser, it is helpful to consider the basic parameters of the system. A pointer is used to navigate the AO structure, navigating linearly forward with probability p and jumping forward or backward along a suffix link with probability $1 - p$. This parameter is referred to as the *continuity* control. As the pointer moves, audio corresponding to the current oracle state is played. Each new frame is cross-faded with the previous, in order to avoid discontinuities in the output signal. Forward linear movement corresponds to playback of the original audio, while jumps forward or backward produce new variations on the original material. Since suffix links connect states which share similar context, the transitions between these jump points will be musically smooth. The length of this shared context, the *longest repeated suffix* (*LRS*), can also be used to constrain navigation. Setting a minimum *LRS* allows jumps to occur only when a certain length of context is shared between states. This parameter allows the user to control the smoothness of the jumps, with shorter contexts producing more abrupt jumps. If a sufficiently long shared context is not found, the system defaults to a linear continuation to the next frame. The oracle navigation can also be limited to only certain areas of the data structure, in effect restricting the musical generation to specific materials. This is accomplished through specifying a *region* of contiguous states. Transitions or jumps which would cause the navigation to fall outside this region are ignored.

Finally, a *taboo list* can be applied. This functions as a circular buffer of flexible length, into which the index of each played frame is placed. During navigation, if the taboo list is active, jumps or transitions which would lead to playback of a frame already present in the taboo list are ignored. This feature helps to eliminate excessive repetition and loop-like behavior that can arise during oracle navigation.

PyOracle Improviser uses the IR tuning process described in the previous section to independently determine the best distance threshold for each of 6 signal features. PyOracle Improviser uses MFCCs, spectral centroid, RMS amplitude, zero-crossing rate, pitch, and chroma estimates. These features were chosen because they relate to important musical parameters. MFCCs and the spectral centroid provide information about musical timbre and brightness, RMS amplitude corresponds to signal energy and musical dynamics, zero-crossing rate is related to both the fundamental frequency and the noisiness of the signal, and pitch and chroma estimates relate to melodic or harmonic materials used. Oracles built on different features will often have different structures, and so it is important to determine each threshold independently. The training process is simple: the human improviser plays a short (approximately 30 seconds to 1 minute) improvisation, aiming to create an excerpted version of their full improvisation. During this short improvisation, feature vectors are stored. After the improvisation, oracles are built on each feature, iterating over a range of possible distance thresholds. Finally, as described above, the threshold which maximizes the total IR of the sequence for each feature is retained, and used to build the oracle for that feature during the full improvisation.

5.5.1 Composition and Improvisation using PyOracle Improviser

PyOracle Improviser also provides several new features to facilitate composing and designing structured improvisations using AO. The primary means for creating a reproducible musical structure is through a scripting mechanism. Any of the parameters

described in the previous section - such as *continuity*, *LRS*, and *regions* - can be scripted using a simple time-line-based system. Sections based on different musical materials are easily defined using *regions*, and areas of greater or lesser musical "fragmentation" can be defined using the *continuity* parameter.

Several different modes of interaction are implemented in PyOracle Improviser. In the *standard* mode, oracle navigation and interaction are defined according to the above parameters, and modified according to a script or in real-time by a human machine operator. The *query* mode is a novel introduction, and deserves special mention here. A similar idea has been explored in the SoMax system, developed by Bonasse-Gahot, in which input MIDI data is used to query a corpus of symbolic musical material [90]. A new accompaniment is generated driven by the input but using materials found in the corpus. In PyOracle Improviser, this notion is extended to operate on the audio signal itself, and without relying on a pre-analyzed corpus. In query mode, the pitch content (chroma) of the input signal is captured and used as a guide for oracle navigation. When query mode is active, each AO navigation jump triggers a three-part search process:

1. The first phase of the search compares past Oracle feature frames to the most recently received input (from the human musician). The indices of those frames which are determined to be similar enough, according to a query distance threshold, are stored.

2. Next, the algorithm iteratively backtracks along suffix links starting from the current state k, and collects states which are connected via suffix, reverse suffix, and transition links.

3. Finally, the next state is determined with a random choice made from the intersection of the set of states with similar features and the set of connected states. If there are no states in common, the oracle simply advances to the next state $k+1$. The function described returns the index of the next state for navigation.

It was found that the query distance threshold should be set slightly higher than the ideal distance threshold for the chroma feature. When set in this manner, less precise matches will be found, which may have produced a less ideal oracle structure but are still similar enough to be musically meaningful. The final mode is **follow** mode. Like the mode of the same name found in OMax, this mode creates a sliding window which advances automatically and constrains oracle navigation to only the material most recently played by the human improviser.

Figure 5.8: Peter Farrar and Ollie Bown sound-checking *Nomos ex Machina.* Photo courtesy Ben Carey.

5.5.2 Case Study: *Nomos ex Machina*

Nomos ex Machina is a structured improvisation which was composed for the Musical Metacreation Weekend, held in June 2013.[2] A graphical score indicating the structure of the improvisation is shown in Figure 5.9. This score is intended for a human performer to read, and is accompanied by a script file which automates PyOracle Improviser. The total duration is approximately eight minutes, and is broken into eight sections, two of which are repeated. The human improviser's part is indicated on the upper line, while the oracle behavior is indicated on the lower line. For the human, the sections in the score indicate time-brackets within which he or she is free to play. Within each section, the performer should aim to produce distinct and focused musical material. This emphasis on focused material is important when considering the way the script automates the PyOracle Improviser. In terms of the software, each section is primarily demarcated by the use of the *region* parameter described above. The region for each section corresponds to the material played during a previous section or sections. The piece unfolds as a series of introductions and interactions of new material that is layered with previously heard material recombined via the oracle. As mentioned previously, PyOracle Improviser builds oracles on multiple audio features simultaneously. In the case of the two repeated sections, sections five and six, the same structure is performed twice, but with a change in oracle feature to shift the emphasis from pitch-based material to timbral material. The instrumentalist is asked to mirror the shift in emphasis. Finally, the last sections of the piece use the query mode. At this point in the performance, there is a variety of material for the query algorithm to choose from, which helps to ensure better matches.

This method of structuring composition and improvisation presents some unique challenges to the human performer. Due to the strictly-timed progression of sections in the piece, the performer must develop materials which can be established or developed within

[2] http://www.isea2013.org/events/mume2013/

a specific duration. Additionally, the material associated with each section will need to function well when layered with previous and/or future materials.

During rehearsals, a few performers expressed a desire for a more flexible timing system. The automatic nature of the timing mechanism may feel inhibiting or limiting to a certain playing style. Additionally, it was found that if the timing shifts during performance (i.e. if the performer begins counting a section before the software), it is possible that previously defined regions will contain erroneous silence or unintended sounds. One potential future project is to explore methods for adding flexibility or variability to the PyOracle Improviser score system. A few options could be considered:

- Enabling performers to advance through sections manually, via a MIDI foot-pedal or other controller. This would increase flexibility, at the cost of additional burden on the performer.

- Using some form of score-following mechanism to advance sections according to the musical performance. This would add significant design overhead during the composition of a new piece, as the score-following mechanism and its mapping to parameters would need to be specified.

- Automatic section changes, according to some compositional algorithm. This would not necessarily increase flexibility, but would provide variety in performance.

5.6 The `ao` Object

The `ao` object was designed to address a pair of problems with the structure and implementation of PyOracle, in order to obtain more modularity and better performance in real-time settings. The first problem was related to computation speed. PyOracle is implemented in Python, a high-level programming language used for a variety of purposes.

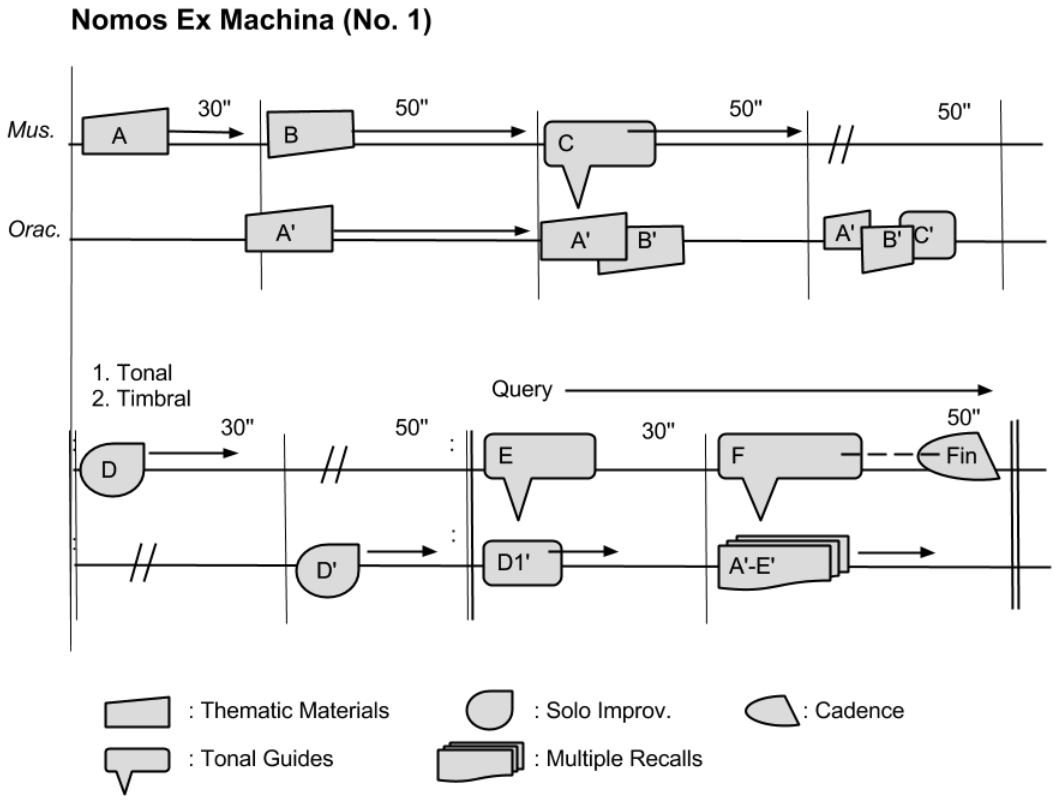

Figure 5.9: Score for *Nomos ex Machina*.

Python was chosen for the way it facilitates rapid development of systems. However, Python trades the efficiency of lower-level languages like C for the development speed enabled by its high-level design. This trade-off is acceptable for offline AO and IR calculations, and therefore was acceptable for the PyOracle library. The PyOracle Improviser system used a combination of Python scripts and Max patches to enable real-time machine improvisation. The Max components handled audio feature extraction and playback, and transmitted feature vectors to the Python scripts via the OSC communication protocol [91]. This design was practical, in that it enabled heavy code reuse from the PyOracle Python library - in fact, most of the PyOracle Improviser scripts are simply wrappers around PyOracle functions. However, the slower speed of the components programmed in Python created a performance bottleneck. As a piece progresses, the AO structure built from it grows proportionally longer.

As the AO structure grows, each new feature vector must be compared to more and more previous vectors - the AO algorithm ensures that not all vectors are compared, but this still became a problem. Eventually, the AO algorithm implemented in Python was not able to keep up with the output of new feature vectors from Max, resulting in irregular time-steps or skipped feature vectors.

The second problem that `ao` addresses is the complexity added by the separation of AO calculation and audio computation. A user needs to ensure that both the Max patch and Python scripts are running and synchronized, and this seemed to be daunting for less technical users. Implementing `ao` as a PD/Max external solved this problem, while also enabling easier integration of AO with a variety of systems, as both PD and Max are widely used in a large number of computer music applications.

5.6.1 Features of the `ao` Object

The `ao` external object provides a simple interface that allows users to use AO for a variety of purposes. The help patch, shown in Figure 5.10, demonstrates how to configure and use AO in both generative and purely analytical contexts. While this section will focus primarily on those aspects of the `ao` external of relevance to IR analysis, the object implements the generative and machine improvisation features described above.

5.6.1.1 Training

By design, `ao` does not perform its own feature extraction. This enables users to use any kind of features desired, and from any source. In order to use `ao`, the object must be trained. As mentioned in Section 8.3.1, the similarity of different feature vectors is measured according to a threshold θ. If the difference between the two vectors is below this threshold, they are considered similar. Of course, as described in Chapter 5, appropriate values of θ will vary with different types of features and different signals. For example, if the feature

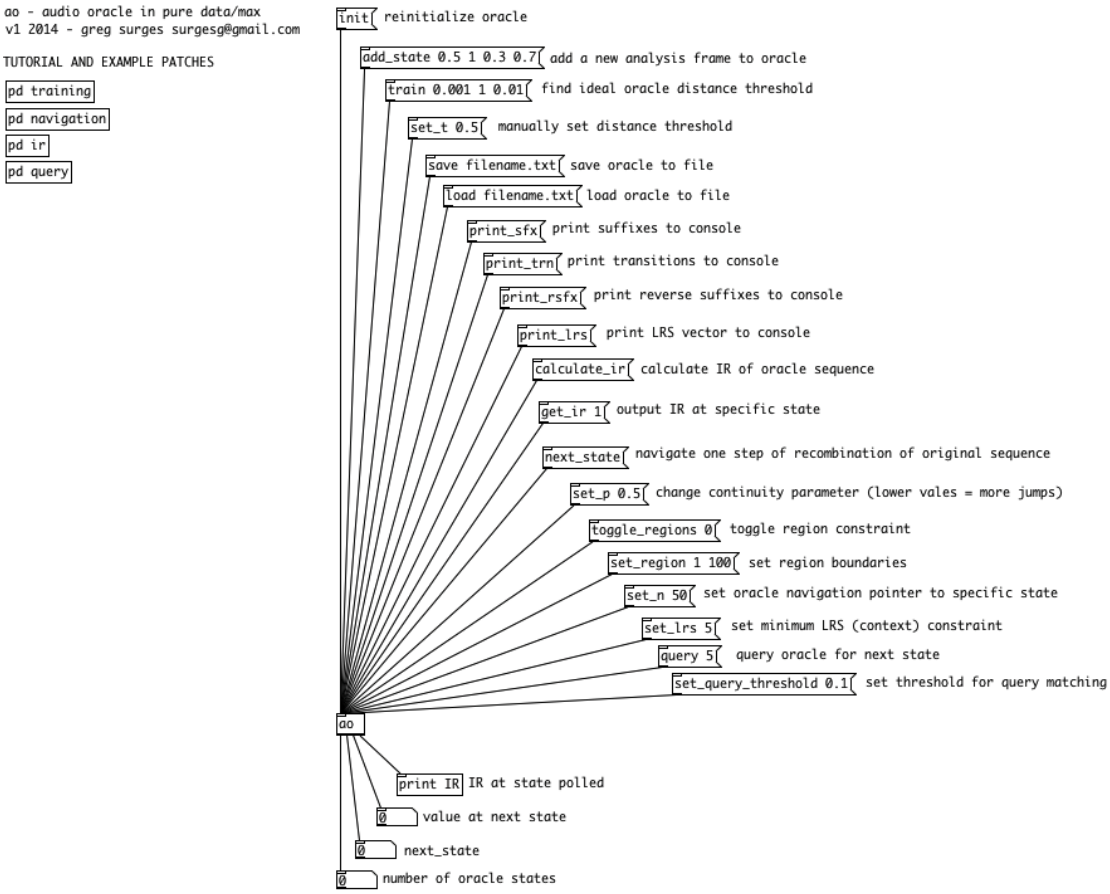

Figure 5.10: The `ao` object help patch, showing all messages.

being used is a measure of signal brightness, then a signal with large jumps in brightness will need a much different threshold than one with very small fluctuations. By setting θ appropriately, the AO algorithm avoids arbitrary quantization schemes, and instead adapts to the particular signal being analyzed.

The `ao` object implements the training scheme introduced in [77] and described above. By iterating over a range of possible thresholds, and choosing the one which maximizes the total IR over the course of the entire signal, the ideal threshold can be found which will produce the best model of the signal. Practically, this is achieved by giving the `ao` object a series of feature vectors as training material. This initial material should cover a wide range. After the initial feature vectors are captured, the threshold θ can be determined with the `train` message, which has arguments indicating a start point, end point, and step size for the range of thresholds to be considered. Since, as described in [77], we want a model that captures both repetitions and novelty, we choose the θ that maximizes the total IR over the signal. After this ideal θ is determined, the AO can be reset for capture of the actual signal.

5.6.1.2 AO Construction

After training, feature vectors are extracted from the audio signal and passed to `ao` as arguments to an `add_state` message. As new vectors are added, the total length of the AO is sent to the leftmost outlet of the `ao` object. Four additional messages can be used to inspect the state of the AO data structure as it is constructed. The messages `print_sfx`, `print_trn`, `print_rsfx`, and `print_lrs`, print their respective data to the PD or Max window.

5.6.1.3 IR Calculation

During AO construction, it is also possible to calculate the IR of the signal. Since this can be somewhat computationally intensive, IR calculation is decoupled from AO

calculation. A secondary message `calculate_ir` causes the IR to be calculated for the current AO structure and stored in an array. It is necessary to recalculate the IR for the entire AO as the structure expands, as the repetition structure of the signal will likely change as new feature vectors are added. After the IR has been calculated, individual indices can be obtained with the `get_ir` message.[3]

5.7 Conclusion

This chapter described musical applications of the Audio Oracle (AO) algorithm. PyOracle Analyzer and PyOracle Improviser use AO to enable music analysis and machine improvisation. Unlike many other machine improvisation programs, PyOracle Improviser does not use a symbolic or quantized representation of the musical events, but instead learns directly from musical signal features. The quality of an AO representation depends on the distance threshold used, and Music Information Rate (IR) can be used to determine the best threshold. Since IR is easy to derive from the AO structure, it is convenient to employ music information dynamics considerations in music composition and improvisation. This chapter described a procedure for predicting the best AO distance threshold for a given improvisation, by learning an excerpt of similar material. PyOracle Improviser presents a group of features enabling reproducible compositions or structured improvisations. In addition to a set of modes defining specific modes of interaction, a scripting function has been implemented and used in a composition titled *Nomos ex Machina*. The script enables the creation of compositional structures by automating parameters and constraints during performance. Though some performers have found found the timing mechanism to be too rigid, this chapter presented some possible methods and future directions for introducing flexibility into these compositional structures. Finally, an implementation of AO for the

[3] Although it may be more desirable to output the current IR function as a single list, this design was a compromise, as Max does not allow for messages longer than 256 atoms to be output from an object.

Pure Data and Max/MSP environments was described. This implementation improved on the original in terms of computational speed and software modularity. The `ao` object will be used in Chapter 8.

Although PyOracle Improviser is not a generative audio system, it is useful to understand prior uses of AO in a generative context. This chapter laid groundwork for discussion to come. Chapter 8 will describe a new application of AO and IR calculation in a generative audio system, as well as an improved implementation of the AO algorithm itself.

5.8 Acknowledgements

Thanks to Ollie Bown and Peter Farrar for their rehearsals and performance during the MuME Weekend 2013. The material in this chapter was originally published in:
Surges, G. and Dubnov, S. "Feature Selection and Composition Using PyOracle." *Proceedings, Workshop on Musical Metacreation, Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.* Boston, MA. 2013. The dissertation author was the primary investigator and author of this paper.

Part III: Generative Audio Systems

Chapter 6

Phase Distortion Using Time-Varying All-pass Filters

The next three chapters describe a technique for applying vibrato or phase distortion to specific spectral components of an input signal using a cascade of time-varying, parametric second-order all-pass filters. This chapter describes the basic technique and provides some examples of how the technique might be applied to instrumental sounds. Chapter 7 addresses stability issues in the time-varying second-order all-pass filter, and presents the use of these filters in generative unitary gain feedback networks. Finally, Chapter 8 describes a generative audio system that makes use of these feedback networks in combination with ideas about computational aesthetics described in Chapters 3 and 5.

As mentioned above, this chapter describes the time-varying, second-order parametric all-pass filter. The parametric second-order all-pass filter offers control over the position and slope of the transition region of the phase response, and this control can be used to tune a phase distortion effect to a specific frequency range. First, the phase response of a cascade of first-order filters is presented, and then is related to that of the parametric second-order all-pass. Instead of directly modulating coefficients, the focus here is on modulating the filter parameters which control the placement and size of the transition region. This makes

it possible to apply phase distortion to specific spectral bands, independently of others. Time-varying parameters and the resulting time-varying phase response are derived for the second-order case, and examples are provided which demonstrate the frequency-selective phase distortion effect in the context of processing of instrumental sounds.

6.1 An Overview of Synthesis Applications of All-pass Filters

All-pass filters are a fundamental synthesis building-block, and have many applications in computer music. All-pass filters have been studied with applications to both synthesis and effects processing, often in cascaded form or with modulation of the filter coefficients. In [92], the dispersive effects of a cascade of first-order all-pass filters are exploited to produce a frequency-dependent delay effect, called a spectral delay filter. Kleimola et al., in [93], propose the use of a cascade of filters, with audio-rate modulation of coefficients, to obtain complex AM- and FM-like spectra, with applications to synthesis, physical modeling, and effect processing. The dispersive effects of cascaded all-pass filters have been used in physical modeling of piano strings [94] and spring reverberators [95]. Finally, Lazzarini et al. describe the use of a first-order all-pass filter in phase distortion synthesis [96], where the authors modulate the filter coefficient with a modulation function designed to create a desired time-varying phase shift.

As described in the introduction to this chapter, this work is part of a larger investigation into the use of time-varying filters in generative feedback systems. By making the phase response of the feedback system time-varying, it is possible to avoid the static timbres characteristic of some feedback systems, and introduce more dynamic musical behaviors. Though the applications discussed in this chapter do not involve feedback systems, the techniques introduced here are a first step toward that goal. The chapters to follow explore

the use of all-pass filters in feedback systems.

6.2 The First-Order All-pass Filter

It is well known that all-pass filters have unity gain at all frequencies. They are, therefore, frequently used in situations where a frequency-dependent phase shift is desirable, but where the amplitudes of spectral components should be left unmodified.

A first-order all-pass filter, given by

$$H_1(z) = \frac{c + z^{-1}}{1 + cz^{-1}}, \tag{6.1}$$

has a pole-zero pair, with the pole located at $-c$ and the zero at $-1/c$, and the constraint that $|c| < 1$ for stability. Because of the stability constraint, the pole lies inside the unit circle, while the reciprocal zero lies outside the unit circle. If the coefficient c is real-valued, both pole and zero will lie on the real axis, with $|c|$ controlling the spacing of the pole-zero pair from the unit circle—a magnitude of c close to 1 positions the pole and its reciprocal zero closer to (as well as more equidistant from) the unit circle (see Fig. 6.1 for example of $c = 0.9$ and $c = -0.6$).

The magnitude of (6.1), is given by:

$$|H_1(\omega)| = \left| \frac{c + e^{-j\omega T}}{1 + ce^{-j\omega T}} \right| = 1, \tag{6.2}$$

where f_s is the sampling rate and $T = 1/f_s$ is the sampling period. A sampling rate of $f_s = 44100$ Hz is used throughout this chapter. It is shown in [97] that the phase of (6.1) is given by

$$\angle H_1(\omega) = -\omega + 2\tan^{-1}\left(\frac{c\sin(\omega)}{1 + c\cos(\omega)}\right). \tag{6.3}$$

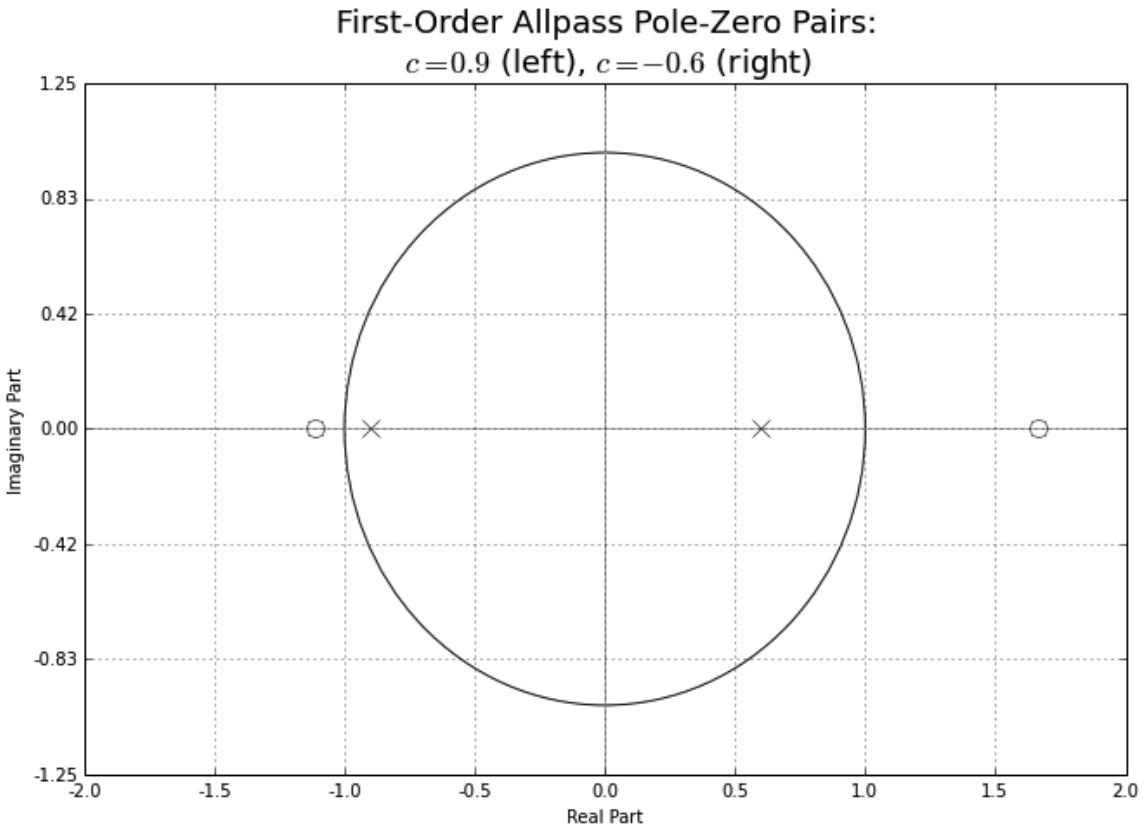

Figure 6.1: The position of the pole-zero pair for the first-order all-pass filter (6.1). A coefficient of $c = 0.9$ yields the pair on the real axis to the left and $c = -0.6$ yields the pair to the right.

Since the denominator inside the $\tan^{-1}(\cdot)$ is always positive (for $|c| < 1$), and the numerator can change sign, the contribution due to the $\tan^{-1}(\cdot)$ term has a possible range of $\pm\pi/2$ rad (and $\pm\pi$ rad for $2\tan^{-1}(\cdot)$), and thus contributes an oscillation around the linear-phase term [98]. The result, as shown in Figure 6.2, is a phase response that is monotonically decreasing, with an overall decrease of 2π rad as ω increases by 2π rad / sample.

Rearranging (6.3) yields the following expression for the coefficient c:

$$c = -\frac{\tan\left(\dfrac{\angle H_1 + \omega}{2}\right)}{\tan\left(\dfrac{\angle H_1 + \omega}{2}\right)\cos(\omega) - \sin(\omega)}, \tag{6.4}$$

which, for $\angle H_1 = -\pi/2$, reduces to

$$c = \frac{\tan(\omega/2) - 1}{\tan(\omega/2) + 1}. \tag{6.5}$$

That is, the behavior of the phase response can be controlled to some extent using (6.5), by

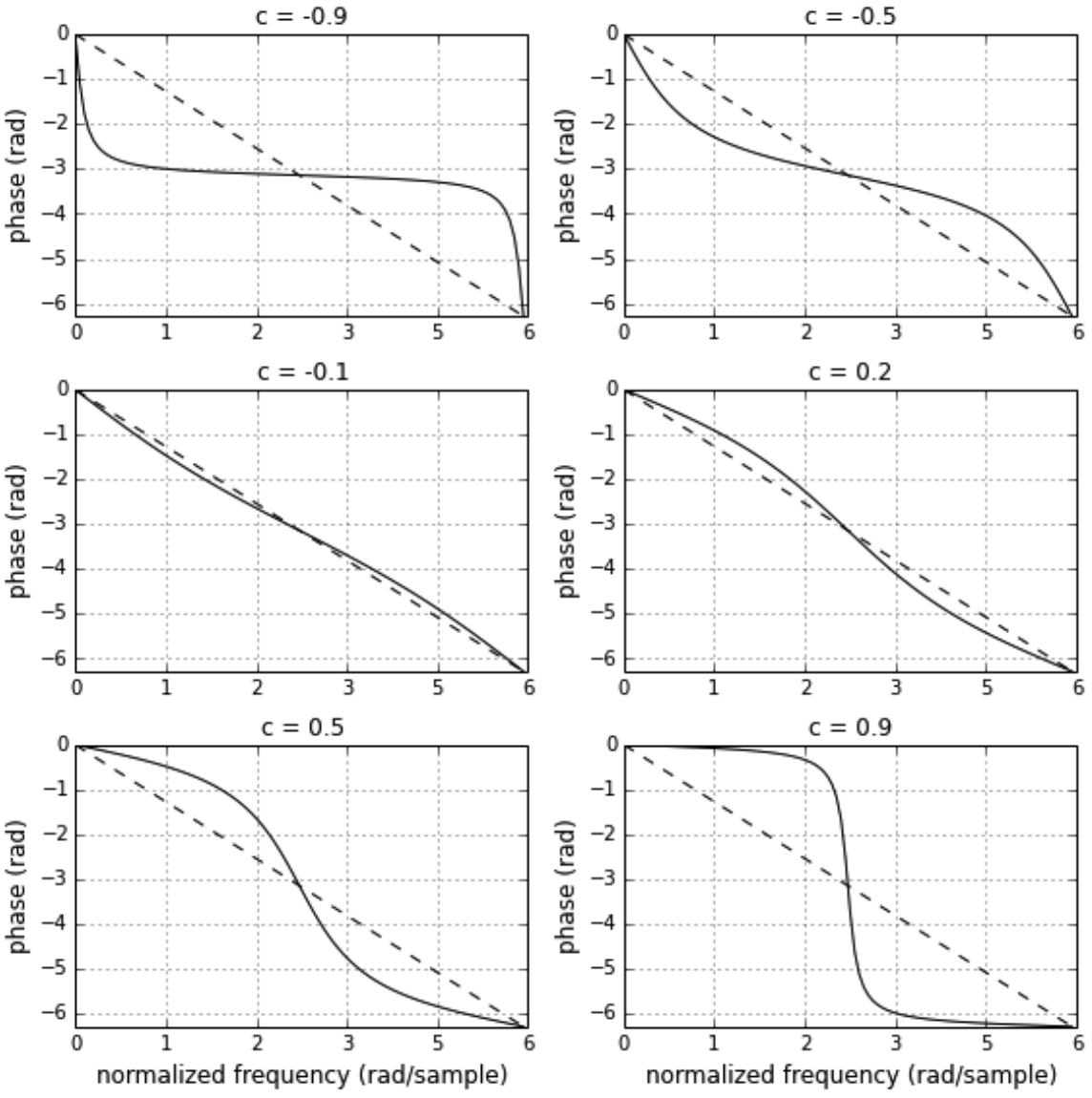

Figure 6.2: The phase response (curved line), monotonically decreasing by 2π with an increase in ω of 2π, is shown with the linear-phase term - ω from (6.3).

specifying the angular frequency

$$\omega = 2\pi \frac{f_{\pi/2}}{f_s} \text{ rad/sample}, \tag{6.6}$$

where $f_{\pi/2}$ is the frequency in Hz at which 90° ($\pi/2$) phase shift is reached.

A general higher-order all-pass filter can be made by cascading several first-order all-pass sections

$$H_k(z) = \prod_{k=0}^{K} \frac{z^{-1} - a_k^*}{1 - a_k z^{-1}}, \tag{6.7}$$

where if the all-pass filter has real coefficients, for each complex root a_k, there must be a corresponding complex conjugate root a_k^*, making the phase anti-symmetric about $\omega = 0$ [98]. The phase response for the overall filter is the sum of the phases for each section, and is given by

$$\angle H_k(\omega) = -K\omega - 2\sum_{k=1}^{K} \tan^{-1}\left(\frac{R_k \sin(\omega - \theta_k)}{1 - R_k \cos(\omega - \theta_k)}\right), \tag{6.8}$$

for $a_k = R_k e^{j\theta_k}$. In the following, a special second-order case is considered.

6.3 The Second-Order All-pass Filter and its Cascade

Though there is some control over the phase response of the first-order filter by using (6.5) to specify the frequency $f_{\pi/2}$ at which the phase response is -90°, it is possible to obtain greater control using a special case of the second-order all-pass filter, for which there is an additional "bandwidth" parameter [99].

The transfer function of a second-order all-pass filter may be expressed using (6.7) for $k = 2$, but a more convenient formulation is given in [99]:

$$H_2(z) = \frac{-c + d(1-c)z^{-1} + z^{-2}}{1 + d(1-c)z^{-1} - cz^{-2}}, \tag{6.9}$$

which allows for specification of coefficients

$$d = -\cos\left(\frac{2\pi f_\pi}{f_s}\right) \tag{6.10}$$

and

$$c = \frac{\tan(\pi f_b/f_s) - 1}{\tan(\pi f_b/f_s) + 1} \tag{6.11}$$

according to the frequency f_π (in Hz) at which the phase response is -180° (or $-\pi$), and a bandwidth of the phase transition region f_b.

Figure 6.3 shows the effects of f_π and f_b on the phase response. Adjusting f_π and f_b allows for both placement of the frequency point at which a 180° phase shift is reached, and control over the slope of the phase transition region.

The effect of the f_π parameter can also be seen in Figure 6.4 which shows how it affects the angle of the two pole-zero pairs on the unit circle. The bandwidth f_b controls the distance of the pole and zero to the unit circle.

6.4 Phase Distortion with the Second-Order All-pass

Through careful tuning of a cascade of these second-order all-pass filters, given in (6.9), it is possible to apply a time-varying phase distortion effect to a specific band of the spectrum. This section will describe how the effect is obtained, the effects of various relevant parameters on the output, and provide a basic example.

6.4.1 A time-varying all-pass filter

In order to make (6.9) time varying, it is necessary to redefine coefficients d and c as functions of time. In this work, rather than modulating the coefficients directly, it is the

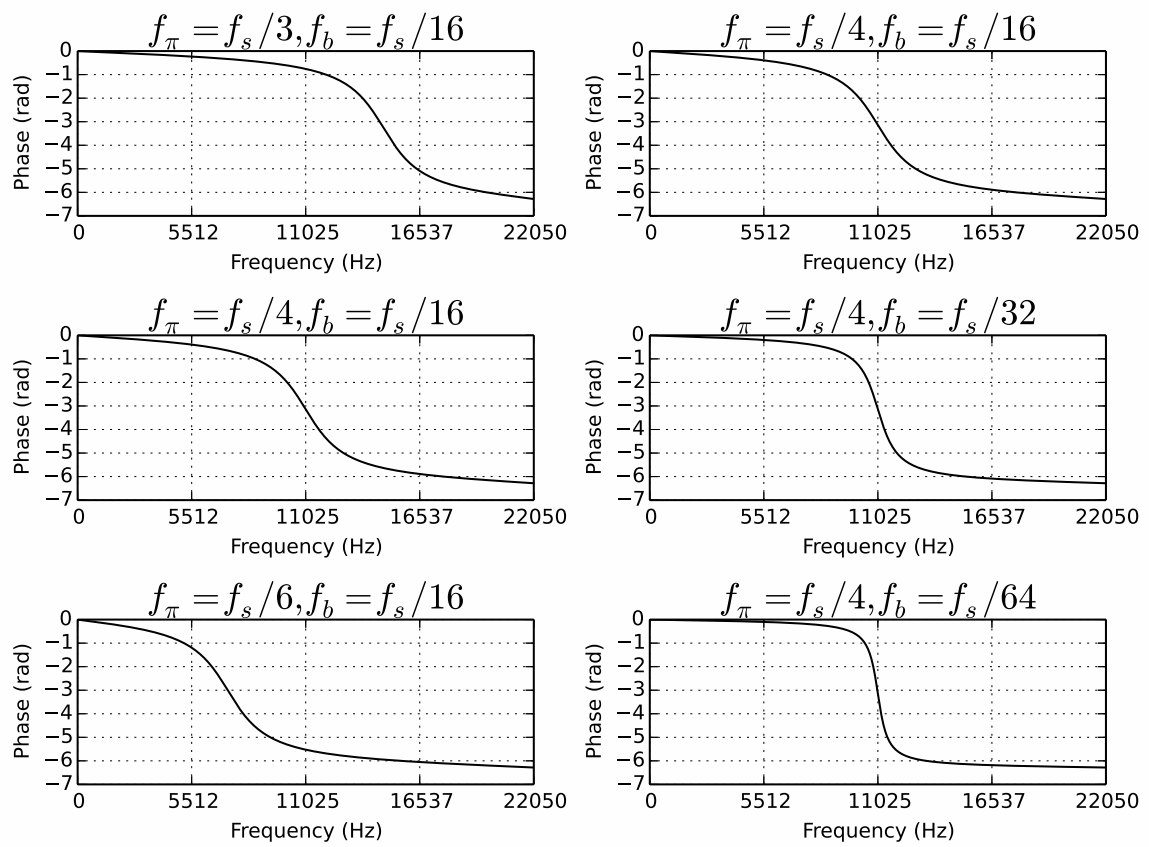

Figure 6.3: Effects of f_π and f_b on the phase response of the second-order all-pass. In the left column, f_π changes while f_b remains constant. In the right, f_b changes while f_π remains constant.

parameter f_π that is made time varying:

$$\tilde{f}_\pi(n) = f_\pi + M\cos\left(\frac{2\pi f_m n}{f_s}\right), \tag{6.12}$$

where $\tilde{}$ indicates a function made time varying, f_π is as previously defined, M is the depth of modulation, f_m is the modulation frequency, and n is the discrete time index. Here, $\tilde{f}_\pi$ is modulated sinusoidally (though this is not a requirement), and can be seen as an FM signal with f_π being the carrier frequency (which it will be subsequently called when referred to in the time-varying case).

The coefficient d from (6.10) is then replaced with

$$\tilde{d}(n) = -\cos\left(\frac{2\pi\tilde{f}_\pi(n)}{f_s}\right), \tag{6.13}$$

yielding the time-varying filter's difference equation

$$\begin{aligned} y(n) &= -cx(n) + \tilde{d}(n)(1-c)x(n-1) + x(n-2) \\ &\quad -\tilde{d}(n)(1-c)y(n-1) + cy(n-2). \end{aligned} \tag{6.14}$$

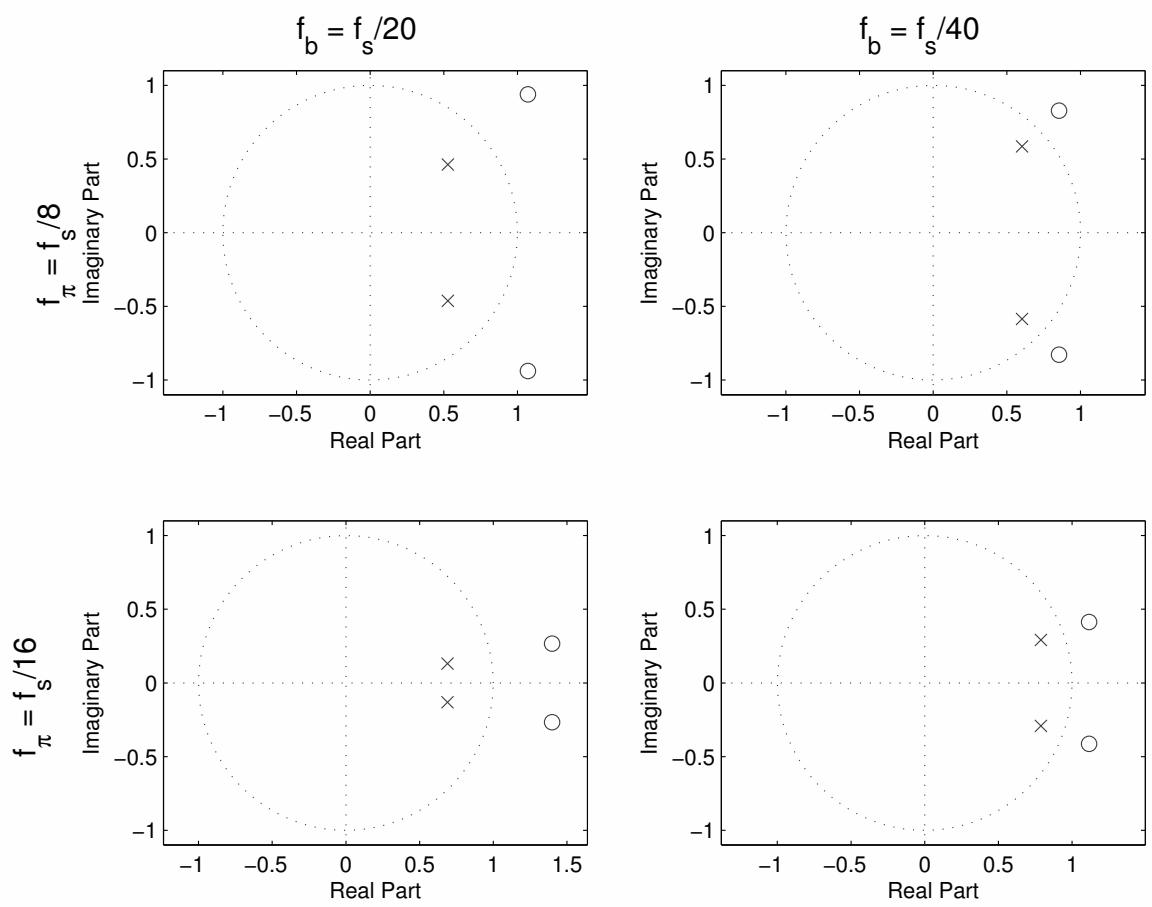

Figure 6.4: The position of the poles and zeros for the second-order all-pass filter described by (6.9). Holding the parameter f_π constant for each row, and f_b constant for each column, we can see how the former adjusts the angle of the two pole-zero pairs, while the latter controls their distance to the unit circle.

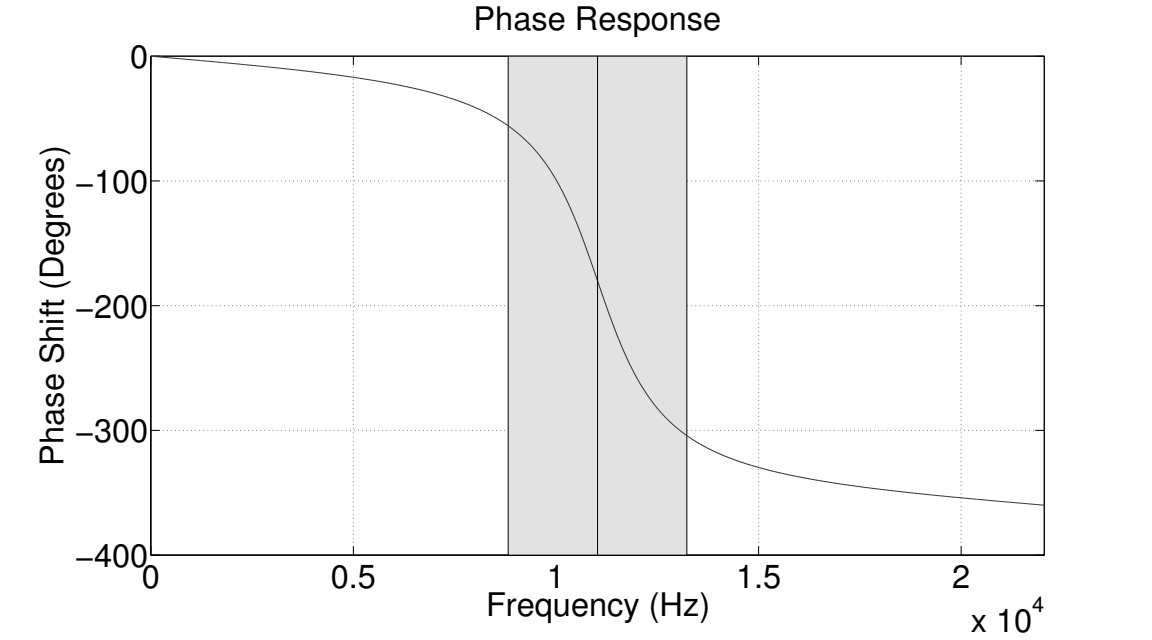

Figure 6.5: Effect of modulation depth and transition region on phase distortion. Vertical shaded region indicates range of $\tilde{f}_\pi(n)$, i.e. $f_\pi - M \leq \tilde{f}_\pi(n) \leq f_\pi + M$.

Expressing (6.9) as the difference equation in (6.14) follows the example given in [93], in which the output of a time-varying filter $y(n)$ is seen as a combination of delayed versions of input $x(n)$ which are ring modulated with sinusoidally-varying coefficients. Here, in contrast, coefficient *parameters* are sinusoidally modulated, yielding time-varying coefficients that are FM signals (see 6.13). Following (6.8) for the phase of the general all-pass, it can be easily be shown that the time-varying second-order all-pass has a family of phase responses which depends on frequency ω and time n, and is given by

$$\theta_A(\omega, n) = -2\omega + 2\tan^{-1}\left[\frac{\tilde{d}(n)(1-c)\sin(\omega) - c\sin(2\omega)}{1+\tilde{d}(n)(1-c)\cos(\omega) - c\cos(2\omega)}\right] \tag{6.15}$$

To gain some intuition for the effect of this new time-varying parameter $\tilde{f}_\pi(n)$, consider the effect of the parameters f_π, f_b, and M on this family of phase responses. Figure

6.5 illustrates an example of the phase response for $f_\pi = f_s/4$ Hz, $f_b = f_s/40$ Hz, and $M = f_s/10$. Recalling (6.12), we can imagine the transition region centered on $\tilde{f}_\pi(n)$ being shifted up and down in frequency - left and right in Figure 6.5 - at a rate corresponding to f_m. The upper and lower limits of this shift are provided by M, and indicated by the shaded box in Figure 6.5. Spectral components in the shaded region will experience significant time-varying phase shift as f_π is modulated, while components outside of that region will experience relatively less. Components below the transition region will be delayed by a small and relatively stable amount, while those above the transition region will be delayed by a larger, but still relatively stable amount. By placing f_π at some frequency of interest, and tuning f_b and M to generate the appropriate transition region, we can apply phase distortion to components which fall into the transition region, while leaving others (relatively) unmodified.

Given a desired frequency deviation $|\tilde{f}_\pi(n)| - f_\pi$, there is a dependency between M and f_m. This can be explained by the interaction between the modulation frequency f_m and modulation index M in determining the instantaneous frequency of an FM signal. Assuming a constant modulation index M, the instantaneous frequency[1] of $\tilde{f}_\pi$ is given by:

$$\tilde{f}_\pi(n) = f_\pi - M f_m \sin(2\pi f_m n + \phi_m) \tag{6.16}$$

By substituting a cosine modulation function (which allows us to disregard time, since the cosine will begin at maximum deviation), and rearranging (6.16) to solve for M, we obtain:

$$M = \frac{|\tilde{f}_\pi| - f_\pi}{f_m} \tag{6.17}$$

where $|\tilde{f}_\pi|$ is the desired peak frequency deviation, and f_π and f_m are the carrier frequency

[1] `http://musicweb.ucsd.edu/~trsmyth/modulation/Modulation_Index_cont.html`

and modulation frequency as defined above. Equation (6.12) then becomes:

$$\tilde{f}_{\pi}(n) = f_{\pi} - M f_m \cos\left(\frac{2\pi f_m n}{f_s}\right) \tag{6.18}$$

As shown in (6.8), a cascade of identical all-pass filters will produce an overall phase response that is the sum of the phases of each section. In addition, the composite filter will have a phase response with a similar curve to that of a single section, with the only difference being a greater range between minimum and maximum delay (the range increasing by a factor of K, the cascade length). This is an important consideration, as the cascade length corresponds to the maximum possible amount of phase distortion. It is often necessary to adjust the cascade length to obtain the desired amount of distortion. In the following discussion, K is the cascade length in terms of second-order filters.

The bandwidth of a modulated cascade is similar to that of FM synthesis, but with a different dependency on the modulation index M. Whereas for classical FM synthesis, the bandwidth can be approximated by

$$BW_{fm} = 2(M+1)f_m, \tag{6.19}$$

the maximum bandwidth of a modulated second-order all-pass is more closely approximated by:

$$BW_{ap} = 2(M+2)f_m \tag{6.20}$$

Thus, given an input signal with a single component with frequency f_0, the resulting spectrum will consist of

$$f_0 \pm k f_m, \text{ where } k = \{0, 1, \ldots, M+2\}. \tag{6.21}$$

The cascade length K has a small effect on the overall bandwidth, by introducing additional,

smaller amplitude sidebands. It has been shown that cascades of all-pass filters can become unstable due to numerical error when larger values of K are used [100], so K generally should not be used as a bandwidth parameter. This and other sources of instability are addressed in Chapter 7. Both f_b and K affect the relative amplitudes of f_0 and the generated side-bands. In the case of a single second-order all-pass filter, as f_b decreases, the amplitudes of the side-bands increase while that of f_0 decreases. As K increases, the amplitudes are affected in a more complex manner, due to the recursive processing at each stage in the all-pass cascade. If each all-pass stage is modulated at the same rate, the output of each stage will contain side-bands at the same frequencies. The generated side-bands sum to produce more complex spectra, with slightly larger bandwidths, as further side-bands are generated around components introduced by earlier all-pass stages. It is important to note that the above measures only apply when input components fall into the transition region controlled by f_b. Input components which fall to either side of this region will be affected to a lesser extent or hardly at all, as described above.

In summary, the parameters of the modulated second-order filter cascade are as follows:

- f_π: modulated filter "carrier" frequency - controls placement of frequency band affected by modulation.
- f_b: filter "bandwidth" - affects width of frequency band affected by modulation, and amplitudes of side-bands.
- f_m: modulation frequency - controls spacing of side-bands at audio rate / speed of vibrato at sub-audio rates.
- M: modulation depth - controls maximum bandwidth of spectrum at audio rate / depth of vibrato at sub-audio rates.

- K: filter cascade length - affects amount of phase distortion applied to frequencies in phase transition region, also affects overall bandwidth. Large K can lead to instability.

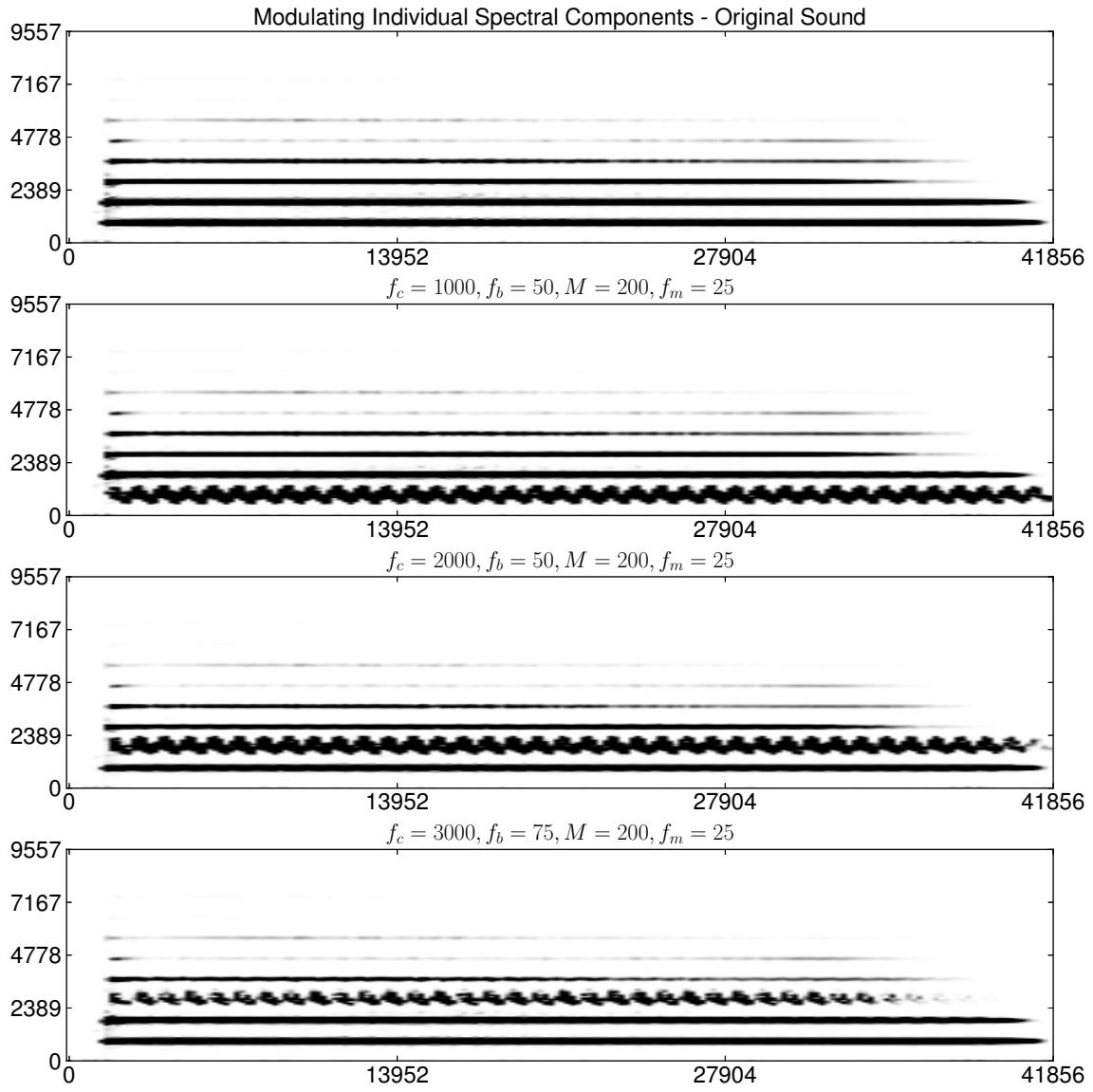

Figure 6.6: Modulating individual components of a clarinet tone. From top to bottom: spectrograms of the original signal and processed versions in which frequency modulation was applied to the fundamental, first, and second harmonics, respectively. In all cases, $\tilde{f}_\pi(n) = f_\pi - M f_m \cos(2\pi f_m n/f_s)$, $K = 10$.

6.4.2 Simple Application to a Clarinet Sample

To demonstrate the frequency-selective nature of this effect, consider a clarinet tone, which contains primarily odd harmonics. Figure 6.6 shows the effect of a modulated all-pass cascade on the spectrum. The all-pass cascade has been tuned to affect only specific harmonics of the clarinet sound. The carrier frequency f_π is set to the frequency of the ith harmonic and the bandwidth f_b is set to 200 Hz. This creates a narrow transition region with a steep slope centered around the frequency of the harmonic. The carrier f_π is modulated by a 25 Hz sinusoid.

6.5 Applications

As a basic signal processing effect, it is possible to use this technique to animate the spectra of steady-state tones (as in the clarinet example of above). As shown previously in Figure 6.6, specific spectral components can be modulated independently. This effect could be useful to add interest to otherwise static timbres, perhaps as a post-processing stage applied to common "analog" waveforms. Here we discuss other musical applications.

6.5.1 Modulation at Sub-Audio Rates

As described above, through careful tuning of the filter parameters, it is possible to apply a frequency modulation effect to specific frequency ranges independently of others. With sub-audio coefficient modulation rates, this produces a selective vibrato effect. Various partials can be modulated independently of the rest, as illustrated in Figure 6.7. A recording of a clarinet improvisation was processed with a cascade of second-order all-pass filters, adding a $f_m = 2$ Hz vibrato to a selected portion of the spectrum. The filter carrier frequency f_π was set to 3674 Hz (the visual midpoint of the spectrum), and a wide filter bandwidth $f_b = 800$ was used in order to affect a range of frequencies. The modulation depth M was

set to 300, producing a 600 Hz swing for $\tilde{f}_\pi(n)$ (6.17), and it was necessary to use a filter cascade of length $K = 15$ in order to obtain the amount of phase distortion necessary to produce dramatic changes in frequency. Referring back to Figures 6.3 and 6.5, we see how f_b affects the slope of the phase transition region. As f_b is increased, producing a more shallow slope, the amount of phase distortion applied to any particular frequency component will decrease. By increasing K, we can compensate for this by increasing the maximum phase delay of the system - and consequently the phase delay applied to any given input component.

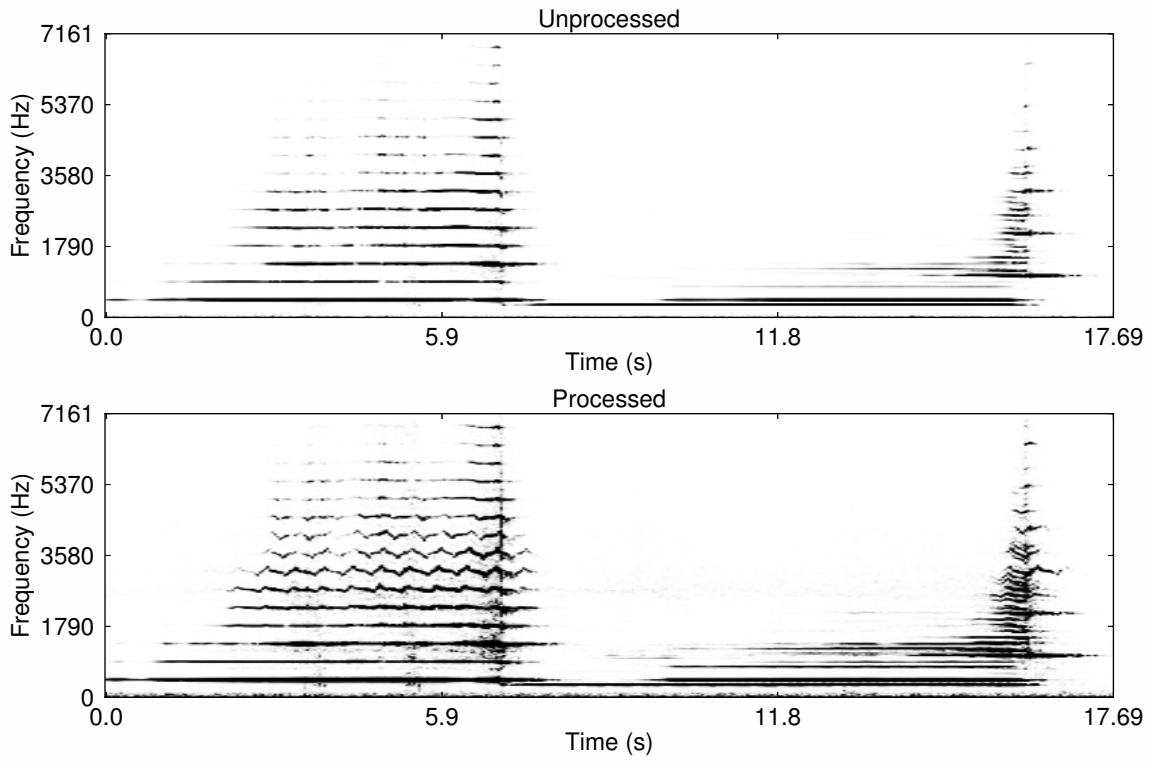

Figure 6.7: Clarinet passage with spectral modulation of selected harmonics for $f_\pi = 3674$ Hz, $M = 300, f_b = 800$ Hz, $f_m = 2$ Hz. Length of all-pass cascade $K = 15$.

6.5.2 Audio-Rate Phase Distortion

In addition to sub-audio modulation, it is possible to modulate the filter parameters at audio rates. As described in Section 6.4, this has the effect of building FM-like side-bands

around component frequencies present in the original signal because of the ring-modulation and FM terms in (6.14), and therefore can produce very rich spectra. Figure 6.8 provides a spectrogram of a clarinet performance through a single cascade of identical second-order all-pass filters, which are driven by a fundamental frequency estimator. The carrier frequency f_π is set to the estimated fundamental.

In this case, the upper harmonics of the sound are left relatively unmodified, while the lower components (those nearest to the estimated fundamental frequency) are significantly modulated. FM-like side-bands appear around at 100 Hz intervals around the distorted frequencies. The amplitude envelopes of new components follow those of the originals. Temporal aspects are also preserved, but a smearing effect is added. There is a possible trade-off that may need to be considered as greater cascade lengths produce a more pronounced "smearing" effect.

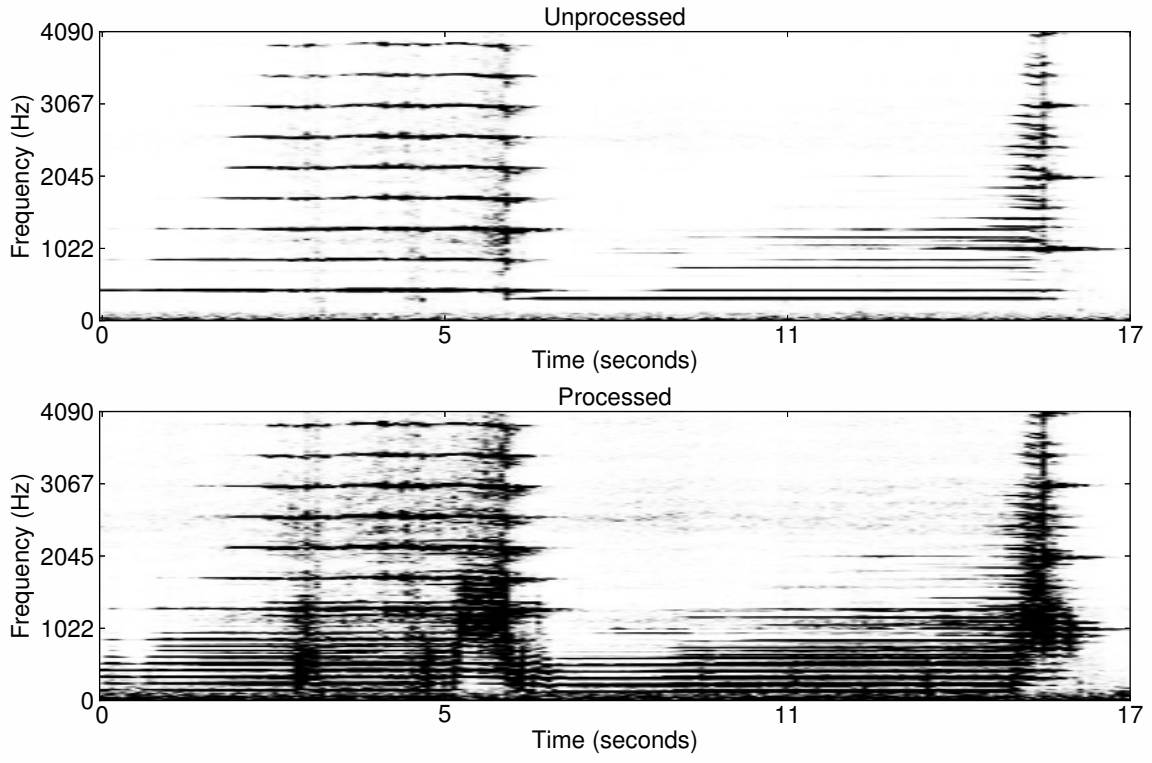

Figure 6.8: Clarinet passage with $\tilde{f}_\pi(n)$ driven by fundamental frequency estimator. Here, $f_m = 100$ Hz, $M = 1, f_b = 500$ Hz, and f_π is the estimated fundamental. Cascade length is $K = 5$.

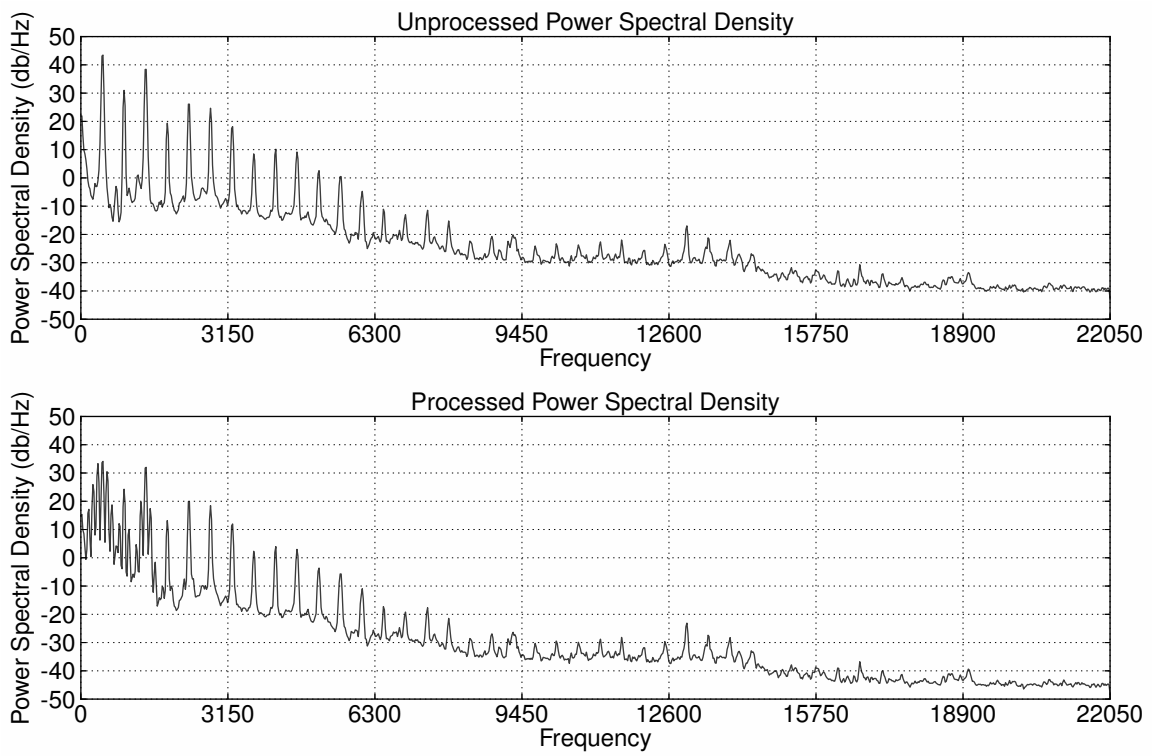

Figure 6.9: Example of audio-rate modulation of $\tilde{f}_\pi(n)$ on clarinet tone. Parameters used are the same as those in Figure 6.8.

Figure 6.9 provides another view of this effect on a portion of the clarinet passage used above. The excerpt used is approximately the first sonority of Figure 6.8 (approximately the first 4 seconds). Here, we see spectra of both the distorted and un-distorted clarinet tone. The estimated fundamental frequency and a few of the surrounding components exhibit significant side-bands, while the remainder of the spectrum is left relatively unmodified. The effect of the interaction between M and f_b is to control the range of affected components (see Figure 6.5). The spectrum of the modulated tone is dense, and contains sub-harmonics not present in the original signal. The overall spectral envelope follows that of the original tone.

6.6 Conclusion

Cascaded and coefficient-modulated second-order all-pass filters have useful applications in de-tuning and phase distortion applications. The second-order all-pass provides a greater amount of control over the transition region of the phase response than does the first. This property can be used to apply phase distortion to only specific frequency ranges. The parametric second-order all-pass filter was presented, along with a means of making the parameters - and therefore the coefficients - time-varying. This allows for the use of the filter as a frequency-selective phase distortion effect, and a simple example of this effect applied to a clarinet tone was provided. Some more realistic example applications were also provided. The first example demonstrated a low-frequency de-tuning effect applied to the upper harmonics of a recorded clarinet passage, and the second applied a high-frequency phase distortion to the same passage. Unfortunately, these filters are susceptible to instabilities when made time-varying. This problem is addressed, and a solution is presented in the form of a unitary rotation matrix, in Chapter 7.

These filters have also been studied as components in self-oscillating feedback systems, where they provide a method of avoiding static timbres without introducing unwanted gain or attenuation into the system. By introducing a time-varying, frequency-dependent phase shift, the system function changes over time, thus producing dynamic and evolving sonic behavior. This is described in more detail in the following chapter.

This technique could also be extended to modulation of the filter bandwidth parameter f_b, the use of non-sinusoidal parameter modulation functions, and the use of second-order all-pass cascades as a synthesis technique - whether driven by a sinusoid or some other signal. Finally, multiple cascades with different time-varying parameters could be used in series, applying differing amounts of phase distortion to various spectral bands. Chapter 8 describes a system which uses some of these techniques.

6.7 Acknowledgements

The material presented in this chapter was originally published in:

Surges, G. and Smyth, T. “Spectral Modulation Using Second-Order All-pass Filters.” *Proceedings, 10th Sound and Music Computing Conference.* Stockholm, Sweden. 2013.

The dissertation author was the primary investigator and author of this paper.

Chapter 7

Generative Feedback Networks Using Time-Varying All-pass Filters

7.1 Introduction

Chapter 6 described some problems that can arise when traditional, time- invariant filter structures are made time-varying. Filters which are stable when their coefficients are time-invariant can quickly experience rapid growth in output power when those coefficients are allowed to vary. This is shown analytically, and a method is given for predicting stability of a filter with a known set of parameters. Next, a power-preserving second-order all-pass filter is given and parameterized. This filter is then related to previous parametric second-order all-pass structures, focusing on its behavior when made time-varying.

These power-preserving all-pass filters are being studied for their use in feedback networks and *generative audio systems*. As described in Chapter 2, such systems are capable of producing dynamic and surprising audio output, with little or no human input and function entirely at the signal rate. The same process which governs high-level musical details should also generate the signal in which those details are embodied. This paper will explore the use of all-pass filters in such systems. In the discussion below, time-invariant and time-varying

all-pass filters are used in feedback loops, and their parameters are modulated in a variety of ways. These systems are shown to have the potential to produce dynamic, generative behavior.

7.2 Synthesis Applications of All-pass Filters

7.2.1 The First- and Second-order All-pass Filters

All-pass filters are well known for applying a frequency-dependent delay to an input signal, while leaving its magnitude spectrum unmodified. The first and second-order all-pass filters were introduced in Chapter 6 and will be briefly covered here. The first-order all-pass filter has a difference equation given by

$$y(n) = ax(n) + x(n-1) - ay(n-1), \tag{7.1}$$

where $|a| < 1$ for stability and for the filter to impart unity gain at all frequencies (the characteristic after which the filter is named). The phase behavior can be controlled to some extent by specifying a frequency $f_{\pi/2}$ (in Hz) at which $\pi/2$ phase shift is reached (the frequency at which the angle of the filter's frequency response is $-\pi/2$) and setting the all-pass filter coefficient to

$$a = \frac{\tan(\pi f_{\pi/2}/f_s) - 1}{\tan(\pi f_{\pi/2}/f_s) + 1}, \tag{7.2}$$

where f_s is the sampling rate used.

Though (7.2) offers some ability to specify phase behavior, additional control is

afforded in the case of the second-order all-pass filter,

$$\begin{aligned} y(n) &= -cx(n) + d(1-c)x(n-1) + x(n-2) - \\ &\quad d(1-c)y(n-1) + cy(n-2), \end{aligned} \tag{7.3}$$

where c and d are set according to the desired "bandwidth" f_b of the phase transition region and the frequency f_π at which the phase response is $-\pi$:

$$d = -\cos\left(\frac{2\pi f_\pi}{f_s}\right) \quad \text{and} \quad c = \frac{\tan(\pi f_b/f_s) - 1}{\tan(\pi f_b/f_s) + 1}. \tag{7.4}$$

Adjusting f_π and f_b allows for both placement of the frequency point at which a phase shift of π is reached and control over the slope of the phase transition region. Figure 6.3 shows the effects of f_π and f_b on the phase response.

7.2.2 Frequency-selective Phase Distortion

In Chapter 6, a technique was described in which a user may apply a vibrato or phase-distortion effect to particular frequency bands of a complex sound, leaving the rest of the spectrum relatively unmodified. The technique involved sinusoidally modulating the parameter f_π,

$$\tilde{f}_\pi(n) = f_\pi + M\cos\left(\frac{2\pi f_m n}{f_s}\right), \tag{7.5}$$

where $\tilde{\cdot}$ indicates a function made time varying, M is the depth of modulation, and f_m is the modulation frequency. The result is a second-order all-pass similar to (7.3), but made time varying,

$$\begin{aligned} y(n) &= -cx(n) + \tilde{d}(n)(1-c)x(n-1) + x(n-1) - \\ &\quad \tilde{d}(n)(1-c)y(n-1) + cy(n-2), \end{aligned} \tag{7.6}$$

where

$$\tilde{d}(n) = -\cos\left(\frac{2\pi\tilde{f}_{\pi}(n)}{f_s}\right). \tag{7.7}$$

By applying carefully designed modulation to the center frequency f_{π}, it is possible to apply a frequency-selective phase distortion effect. Due to the non-linearity of the phase response, certain frequency bands (those nearest to f_{π}) are modulated while others are left unmodified.

The results introduced in [97] and described in Chapter 6 showed very promising musical potential, but use of a "time-invariant" filter caused expected instabilities. The work presented here addresses these issues so that the original synthesis technique may be used more reliably and its musical potential further explored.

7.3 Stability Analysis of Time-Varying All-pass Filters

7.3.1 Instability of the Time-Invariant All-pass Filter

It is well-known that systems with constant parameters can become unstable when those parameters are made time varying. In [101], it is shown that if the first-order all-pass filter (7.1) is made time-varying,

$$y(n) = a(n)x(n) + x(n-1) - a(n)y(n-1), \tag{7.8}$$

the filter coefficient condition $|a(n)| \leq \varepsilon < 1$ is not sufficient to ensure energy preservation. It is demonstrated that when $x(n)$ is an impulse and filter coefficients $a(n) = \varepsilon(-1)^n$, the

output energy is

$$\begin{aligned} ||y(\cdot)||^2 &= a(0)^2 + (1 - a(0)a(1))^2 \left(1 + \sum_{n=2}^{\infty} \prod_{i=2}^{n} a(i)^2\right) \\ &= \varepsilon^2 + (1+\varepsilon^2)^2 \sum_{n=0}^{\infty} \varepsilon^{2n} \\ &= \frac{1+3\varepsilon^2}{1-\varepsilon^2}, \end{aligned} \tag{7.9}$$

which is greater than the input for any choice of $\varepsilon < 1$ [101]. A solution is suggested in terms of a wave digital one-port and a corresponding orthogonal matrix formulation that ensures energy preservation (to be discussed in Section 7.4).

7.3.2 Stability Analysis

Though the above example shows that stability cannot be ensured, it is not the case that all time-varying coefficients will render (7.6) unstable. Because the coefficients in (7.6) are modulated sinusoidally, a technique for determining stability of a *periodic* time-varying system may be used by representing the system as a state space matrix [24]. It is necessary to represent only the recursive part of (7.6) as a system of equations which, in matrix form, becomes

$$\begin{bmatrix} y_1(n) \\ y_2(n) \end{bmatrix} = \begin{bmatrix} -d(n)(1-c) & c \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} y_1(n-1) \\ y_2(n-1) \end{bmatrix}, \tag{7.10}$$

or

$$\mathbf{y}(n) = \mathbf{A}(n)\mathbf{y}(n-1), \tag{7.11}$$

where $\mathbf{y}(n)$ and $\mathbf{y}(n-1)$ are state vectors of the system at time sample n and $n-1$, respectively, and $\mathbf{A}(n)$ is the time-varying coefficient matrix. The filter state vector at time sample

k may be determined by

$$\begin{aligned}\mathbf{y}(k) &= \mathbf{A}(k)\mathbf{y}(k-1) \\ &= \mathbf{A}(k)\mathbf{A}(k-1)\mathbf{y}(k-2) \\ &= \ldots \\ &= \prod_{n=k}^{1}\mathbf{A}(n)\mathbf{y}(0),\end{aligned} \tag{7.12}$$

for known initial conditions $\mathbf{y}(0)$. Though stability can be calculated by assessing the behavior of the state vector's norm,

$$||\mathbf{y}(k)||^2 = y_1^2(k) + y_2^2(k) + \ldots + y_n^2(k), \tag{7.13}$$

a simplified method can be used if filter coefficients change periodically. For a periodically linear time-variant system, the system monodromy matrix,

$$\mathbf{C}(N,0) = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \prod_{i=N}^{1}\mathbf{A}(n). \tag{7.14}$$

connects arbitrary states of the system separated by N, the period of variation, in samples, of $d(n)$. The system is stable if all eigenvalues $|\lambda_1, \lambda_2, \ldots, \lambda_n|$ of $\mathbf{C}(N,0)$ are less than or equal to 1, giving the following limits which ensure stability:

$$\begin{aligned} 1 - C_{11} - C_{22} + \det(\mathbf{C}(N,0)) &\geq 0 \\ 1 + C_{11} + C_{22} + \det(\mathbf{C}(N,0)) &\geq 0 \\ |\det(\mathbf{C}(N,0))| &\leq 1, \end{aligned} \tag{7.15}$$

where

$$\det[C(N,0)] = \prod_{i=N}^{1} c = c^N. \tag{7.16}$$

By evaluating the conditions in (7.15), it is possible to determine whether a given set of filter parameters f_m, f_π, M, f_b, will produce a stable, periodic time-varying filter. Figure 7.1 shows the complexity of the interactions between the four parameters. The interdependence of these parameters makes it difficult to intuitively understand which combinations will produce instability, rendering this version of the filter very dangerous for real-time use where a variety of settings may be used.

7.4 A Power-Preserving All-pass Filter

Though it is possible to analyze stability and determine which combinations of parameters f_π, f_b, M, and f_m will produce a system in which power is preserved, there is an alternate power-preserving form of the first- and second-order all-pass filters. Fortunately, for the application considered here, the power-preserving all-pass filter exhibits comparable phase response characteristics, and can be parameterized in nearly the same way as the time-invariant case.

Ensuring the output power of a filter is equal to the input power,

$$\sum_{n=0}^{N-1} x^2(n) = \sum_{n=0}^{N-1} y^2(n), \tag{7.17}$$

can be accomplished by using a power-preserving rotation matrix such as

$$\begin{bmatrix} \cos(r) & -\sin(r) \\ \sin(r) & \cos(r) \end{bmatrix}. \tag{7.18}$$

If multiple input signals to a system $x_1(n), \ldots, x_K(n)$ are considered coordinates in K-space,

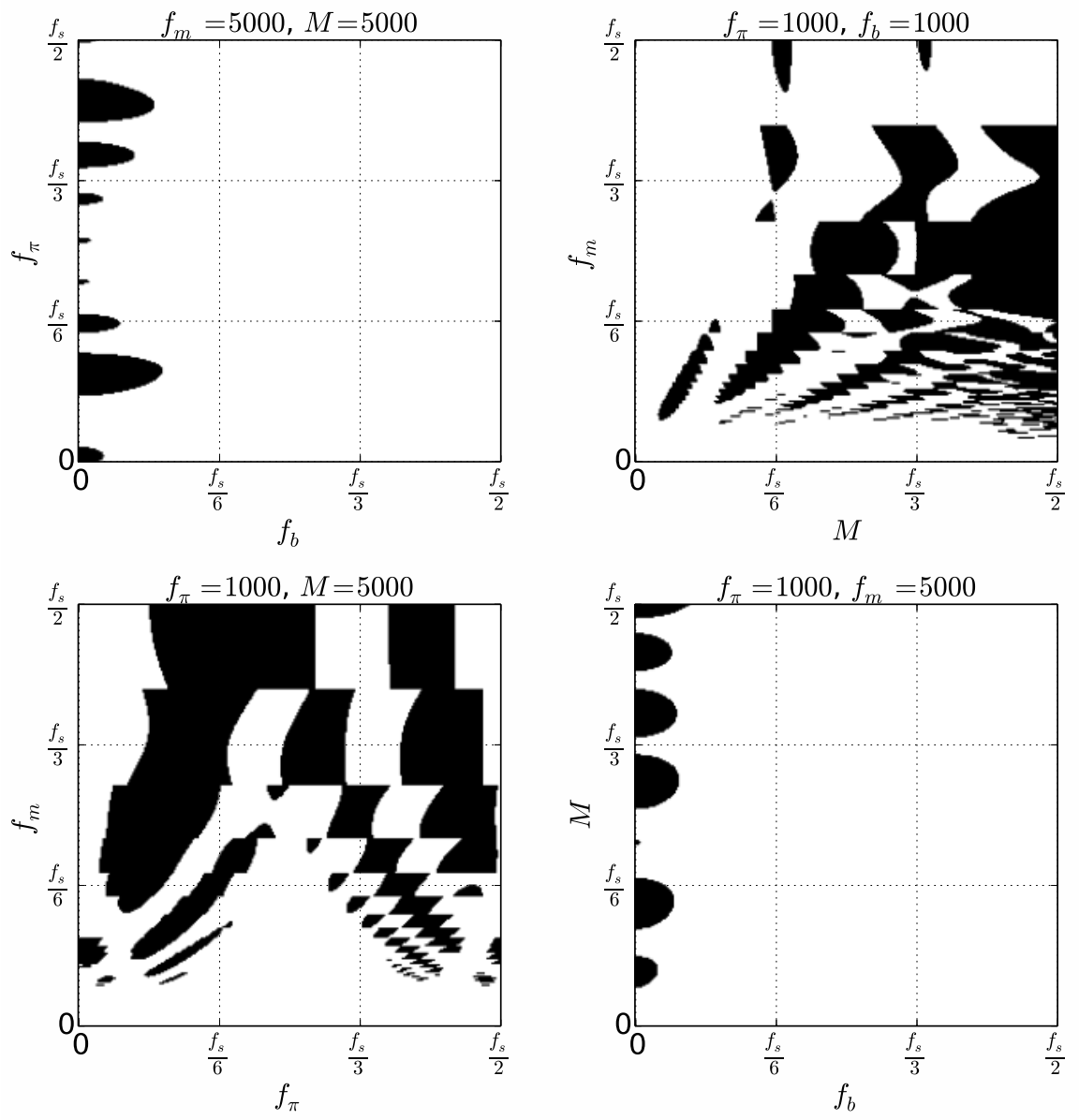

Figure 7.1: Influence on stability of the interaction between pairs of parameters. Each subplot allows two parameters to vary, while the others are fixed. White indicates a stable combination.

then a rotation is an operation that, by definition, will preserve the length, or correspondingly, the magnitude of the input [102]. This solution is very similar to that presented in [101], in which an orthogonal scattering matrix formulation of a wave digital one-port is used to ensure power preservation, however the use here of sinusoids as matrix elements instead of square root functions is preferred because it enables generation of both positive and negative

values.

The second-order all-pass filter is obtained using a pair of power-preserving rotation matrices, yielding

$$\begin{bmatrix} y(n) \\ z_1(n) \\ z_2(n) \end{bmatrix} = \begin{bmatrix} x(n) \\ z_1(n-1) \\ z_2(n-1) \end{bmatrix} \mathbf{AB}, \tag{7.19}$$

where

$$\mathbf{A} = \begin{bmatrix} \cos(r_1) & -\sin(r_1) & 0 \\ \sin(r_1) & \cos(r_1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.20}$$

and

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(r_2) & -\sin(r_2) \\ 0 & \sin(r_2) & \cos(r_2) \end{bmatrix}. \tag{7.21}$$

With some algebraic manipulation, the equivalent difference equation can be obtained

$$\begin{aligned} y(n) &= b_0 x(n) - b_1 x(n-1) + x(n-2) + \\ &\quad b_1 y(n-1) - b_0 y(n-2), \end{aligned} \tag{7.22}$$

where

$$b_0 = \cos(r_1) \tag{7.23}$$

$$b_1 = \cos(r_2)(1+\cos(r_2)). \tag{7.24}$$

If (7.22) is made equal to the time-invariant all-pass in (7.3), coefficients r_1 and r_2 can be

expressed in terms of c and d,

$$r_1 = \arccos(-c) \tag{7.25}$$

$$r_2 = \arccos(-d). \tag{7.26}$$

and thus in terms of the corresponding desired control parameters f_π and f_b as given by (7.4).

7.4.1 Differences in Output

Figures 7.2-7.5 show the difference in output between the original time-invariant version of the filter and the new power-preserving one. Figure 7.2 shows the effect of an instantaneous change in f_π on both filters. The time-invariant form experiences a large jump in amplitude at the time where f_π changes, while the power-preserving form experiences a jump in phase. Figure 7.3 shows the difference in output between the two filters when they are made time-varying with the same sinusoidal modulation parameters. For the most part, the output is similar, but the time-invariant form experiences larger spikes in amplitude. These spikes are much smaller in the power-preserving output. As implied by the waveforms in Figure 7.3, the output of the two filters is very similar in terms of harmonic content. In fact, as evidenced by the analysis in Figure 7.4, the outputs of the two filters share the same harmonics, only differing slightly in their amplitudes. Figure 7.5 shows the output of both filter types with parameters chosen so that the time-invariant form will become unstable. The power-preserving matrix form maintains stability, while the time-invariant form produces output which grows rapidly.

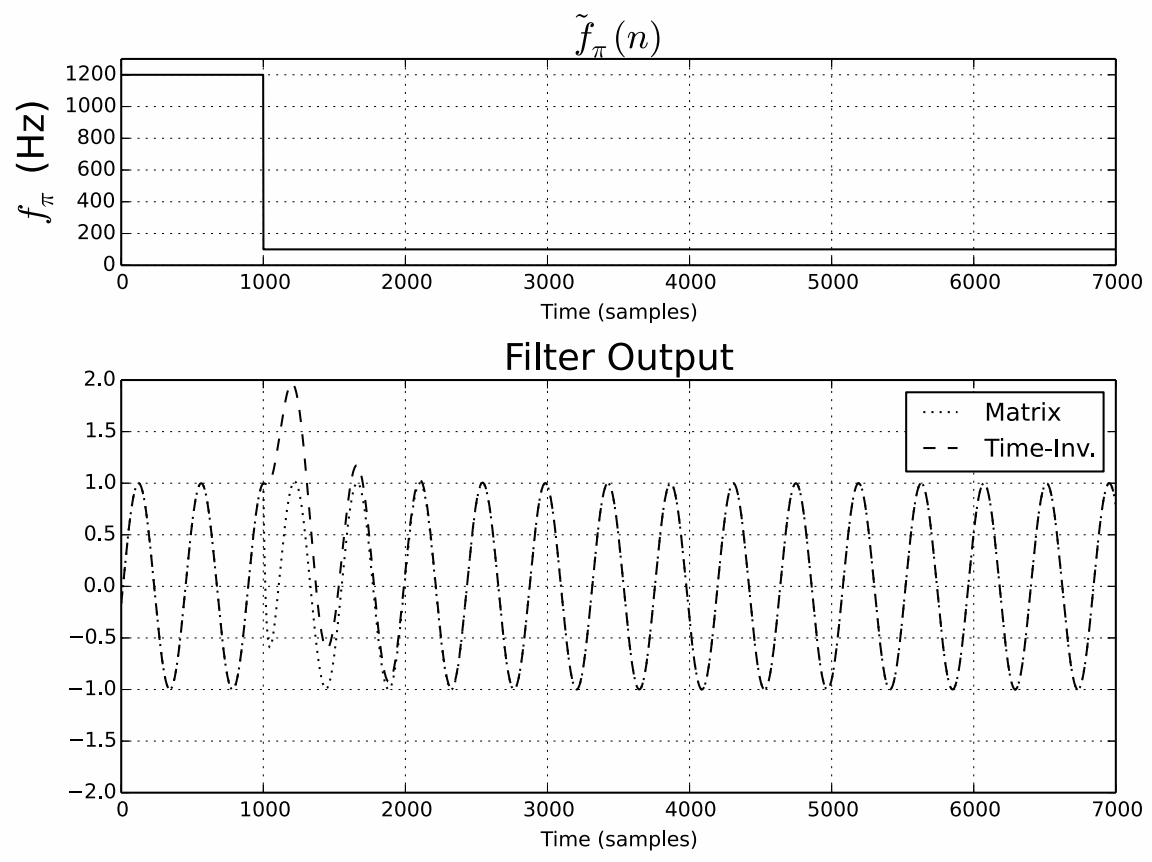

Figure 7.2: Effects of instantaneous change in f_π for time-invariant and power-preserving filter types.

7.5 All-pass Filters in Feedback Networks

As discussed in the introduction, the aim of this project is to investigate the use of all-pass filters in generative audio systems. It is thus useful to look at the behavior and potential sound production of various configurations of all-pass filters in unity gain feedback networks. First, the time-invariant transfer function is examined, followed by a qualitative study of the spectra generated by the corresponding time-varying system. Finally, an application will be given, in which time-varying all-pass filter is used to create a generative oscillator.

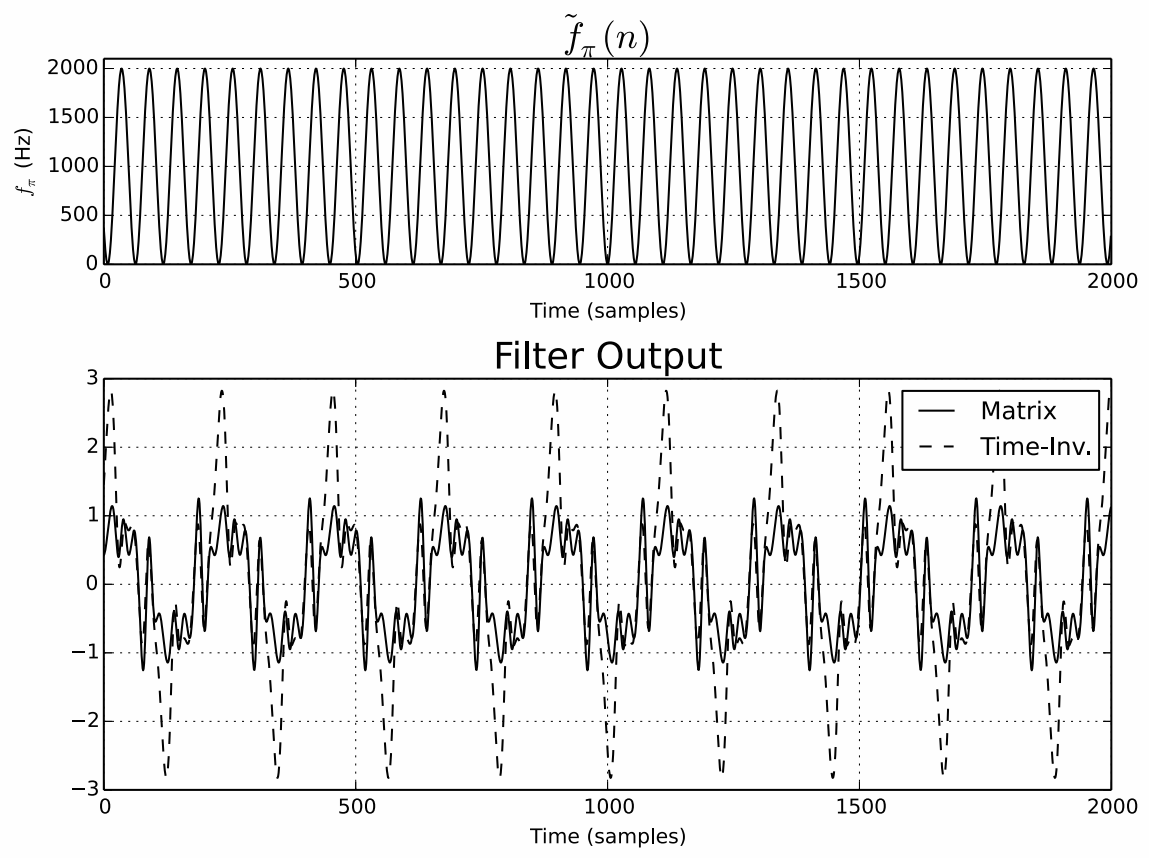

Figure 7.3: Output of time-invariant and power-preserving filter types for identical modulation parameters. Parameters were chosen so that both filters would remain stable.

7.5.1 Transfer Function Analysis of Time-invariant Feedback Systems

In the examples in this section, if the systems are excited with an impulse, they behave as oscillators and may be left to oscillate indefinitely. The aim of this section is to gain insight into the character of the generative oscillator, as well as how they respond to control parameters f_π and f_b.

The output of a feedback system, like the one shown in Figure 7.6, comprised of a cascade of N second-order all-pass filters given by (7.29) and a feedback delay of T, may be expressed

$$Y(z) = \left(X(z) + Y(z)z^{-T}\right) H_1(z)^N, \tag{7.27}$$

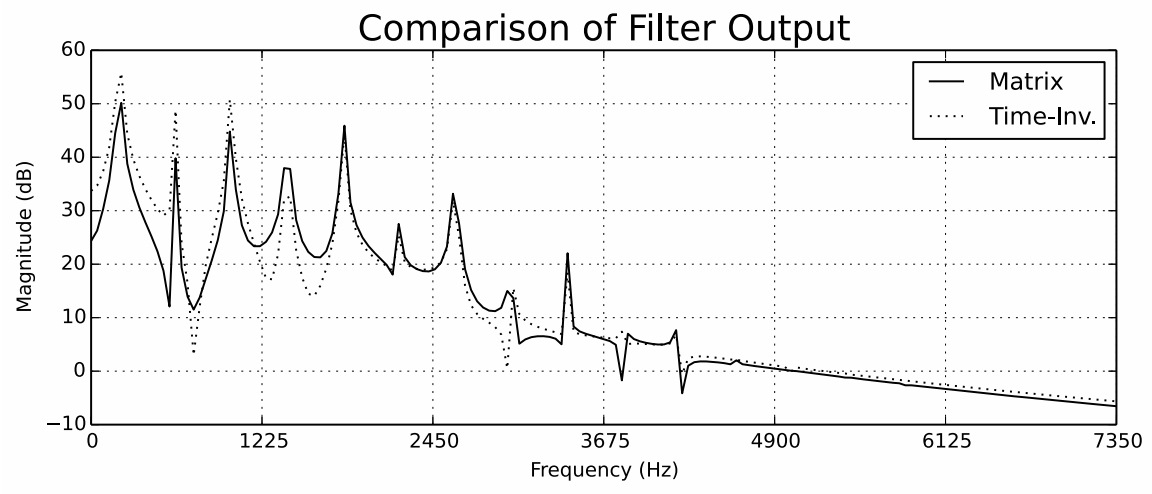

Figure 7.4: Magnitude spectrum of output of time-invariant and power-preserving filter types for identical modulation parameters. Parameters were chosen so that time-invariant filter would become unstable.

yielding a system transfer function of

$$H_2(z,N) = \frac{Y(z)}{X(z)} = \frac{H_1^N(z)}{1 - H_1^N(z)z^{-T}}. \tag{7.28}$$

where

$$H_1(z) = \frac{b_0 - b_1 z^{-1} + z^{-2}}{1 - b_1 z^{-1} + b_0 z^{-2}}, \tag{7.29}$$

is the transfer function of the power-preserving second-order all-pass filter whose difference equation is given in (7.3).

It is clear that since $H_2(z,N)$ is not an all-pass filter, the overall effects of control parameters f_π and f_b are no longer what they were for $H_1(z)$ and warrant further exploration of their behavior in their new context.

When $N = 1$ and $T = 1$, the system transfer function reduces to

$$H_2(z,1) = \frac{b_0 - b_1 z^{-1} + z^{-2}}{1 - (2b_0 + b_1)z^{-1} + (2b_0 + b_1)z^{-2} - z^{-3}}. \tag{7.30}$$

Representing H_2 in this way has the advantage of allowing observation of its poles and zeros

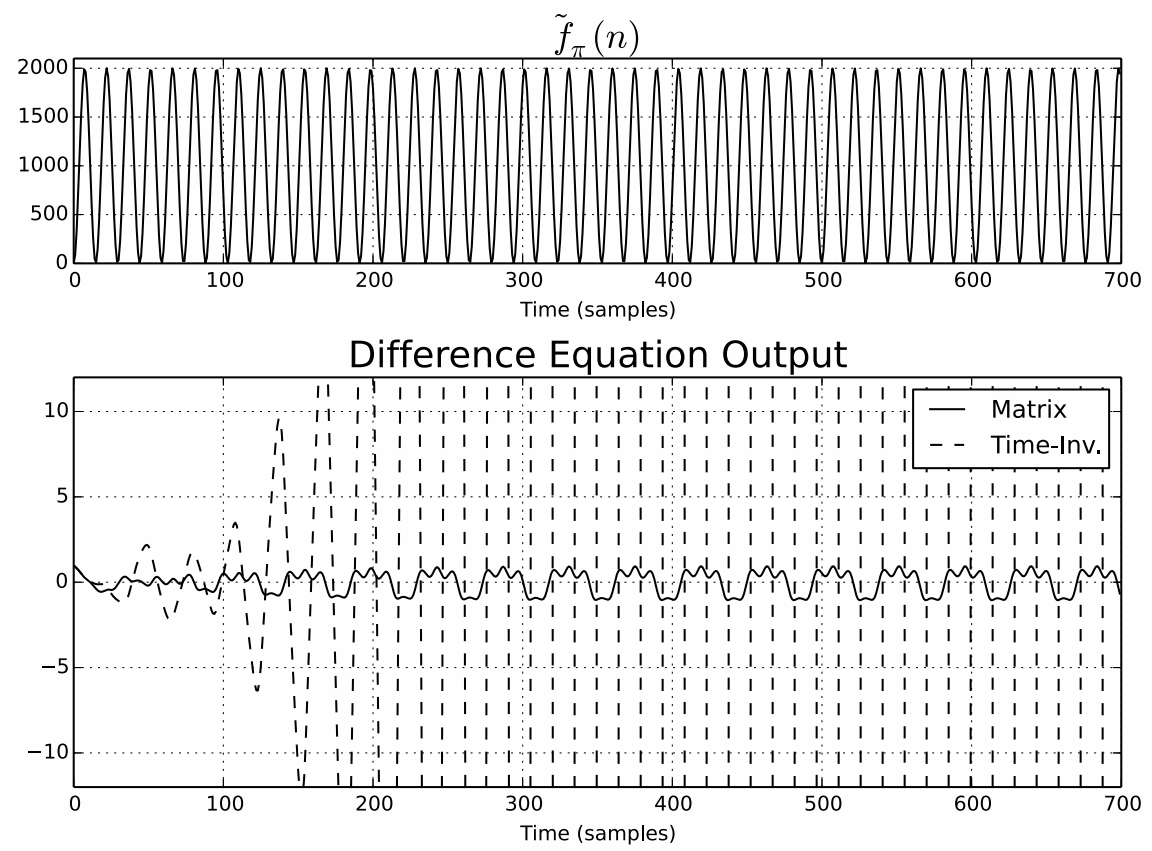

Figure 7.5: Output of time-invariant and power-preserving filter types for identical modulation parameters. Parameters were chosen so that time-invariant filter would become unstable.

(shown in Figure 7.7 for f_π constant at 11025 Hz and various values of f_b between 500 and 4000 Hz). As might be expected of a unitary feedback system, all three poles - two complex and one real (at DC) - are directly on the unit circle. A decrease in f_b corresponds to a movement of the two zeros downward toward the unit circle, and the two complex poles along the unit circle toward a point of pole-zero convergence.

The behavior of the poles and zeros has the effect, shown in Figure 7.8, of a resonant peak in the amplitude response being shifted somewhat downward in frequency, with a notable decrease in bandwidth. Accordingly, as shown in Figure 7.9, keeping f_b constant while increasing f_π results in a magnitude response characterized by a peak that loosely follows f_π, but with a bandwidth that decreases slightly with an increase in frequency. It

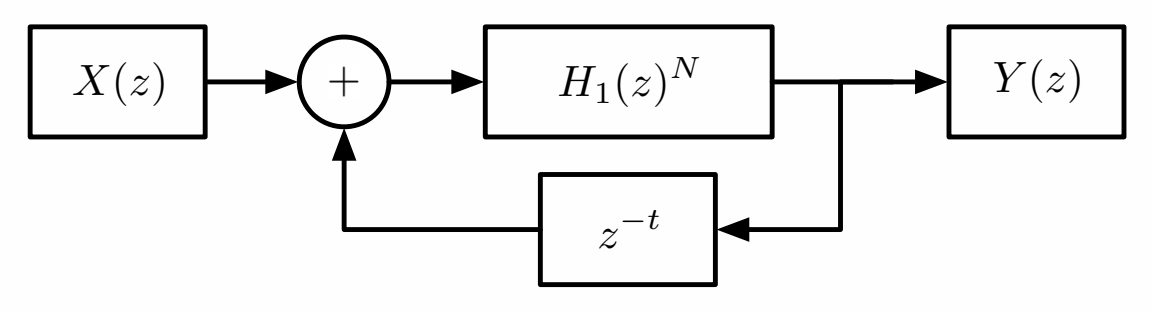

Figure 7.6: Block diagram of all-pass filter cascade in feedback configuration.

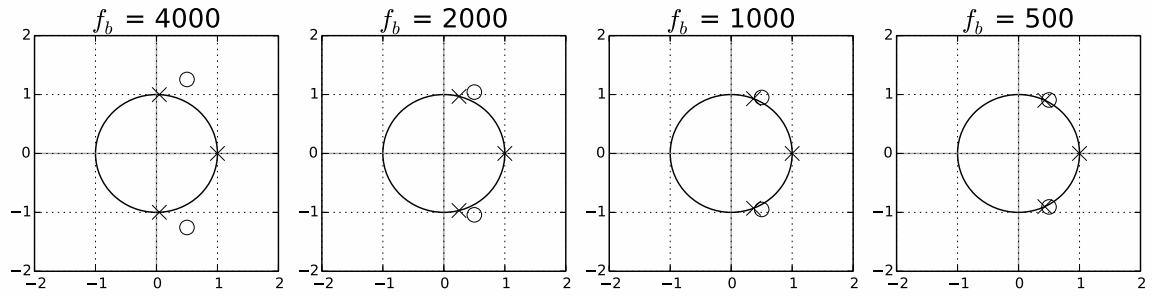

Figure 7.7: Example pole-zero plots as f_b is varied while f_π is kept fixed, demonstrating effect of f_b on zero location.

is clear, therefore, that both parameters affect the position and width of the peak, with the bandwidth being more significantly influenced by f_b, and the position (frequency) more significantly influenced by f_π.

Finally, cascading multiple all-pass filters (increasing N) has the effect of increasing the order of H_2, resulting in the introduction of an additional peak. Thus, as shown in Figure 7.10, a cascade of N filters within H_2 results in an amplitude response with N peaks (excluding DC).

7.5.2 Spectra of Time-varying Feedback Systems

The following discussion considers the effect of various modulation parameters on the spectra of the output of a time-varying feedback system. The parameters are made time-varying as shown in Section 7.2.2.

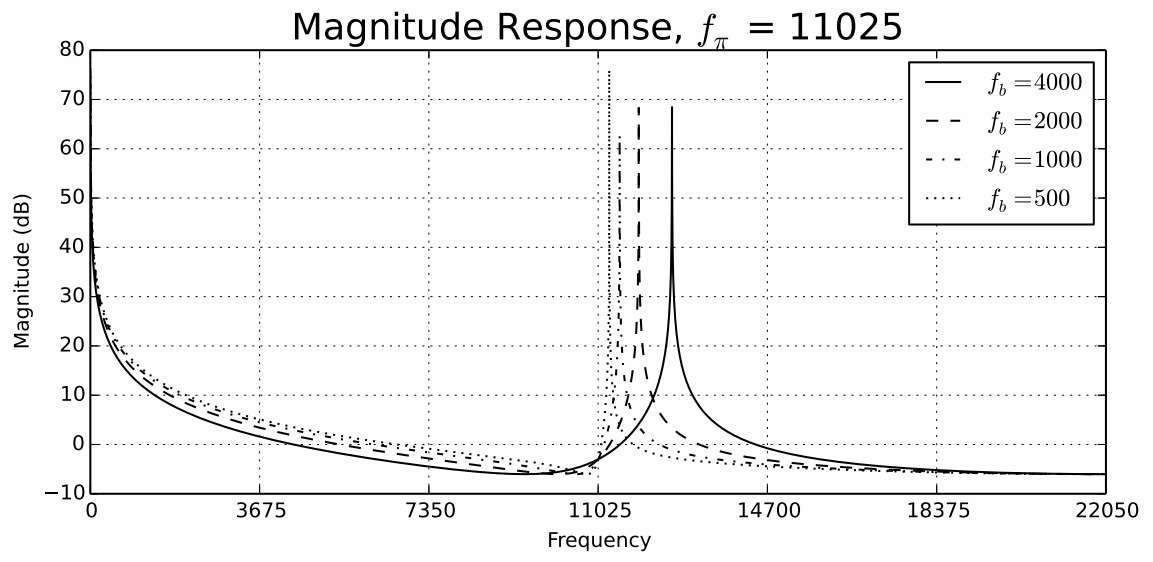

Figure 7.8: Example magnitude responses as f_b is varied while f_π is kept fixed, demonstrating decreasing peak location.

The effect of each of the parameters f_π, f_b, f_m, and M on the output spectra is relatively clear, and in most cases is related to the effect on the non-feedback systems described in [97]:

- f_m - Controls spacing of side-bands. Side-bands are spaced at integer multiples of a central frequency (which depends on f_π and f_b as discussed above). Figure 7.11 shows this effect.

- f_π - Affects center frequency of side-bands. As f_π increases, the center frequency increases, though not in a 1 : 1 relationship. Figure 7.12 shows this effect.

- f_b - Affects center frequency of side-bands. As f_b decreases, the center frequency decreases, becoming closer to f_π. Additionally, the bandwidth of the output is increased slightly with increasing f_b. Figure 7.13 shows this effect.

- M - Affects bandwidth of output. As M increases, bandwidth increases. Figure 7.13 shows this effect. Side-bands can alias around DC or the Nyquist frequency back into the spectrum.

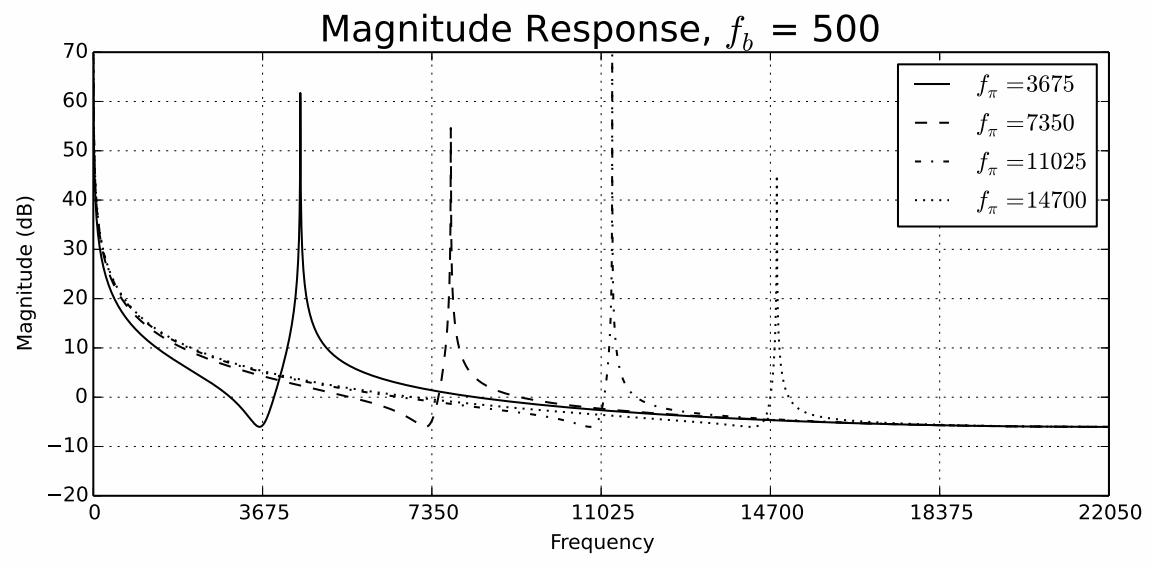

Figure 7.9: Example magnitude responses as f_π is varied while f_b is kept fixed, demonstrating increase in peak location.

Finally, as an important note, consider the side-bands present just above DC in Figures 7.11 - 7.14. These are components built around the DC term, introduced by the real pole of the feedback system.

7.5.3 A Self-Modulating All-pass Feedback Network

The final network investigated here is one in which the sinusoidal modulation of (7.5) is replaced with a modulation signal derived from a point in the network itself. In this configuration, similar to one posed in [103], the delayed output of the all-pass filter is used to modulate the f_π parameter. A block diagram of this system is shown below in Figure 7.15.

In order to use such a configuration, the delayed output sample $y(n-1)$ must be scaled and biased in order to occupy the same range as the parameter to be modulated. For example, if we desire to modulate f_π with $y(n-1)$, then (7.5) is replaced with

$$\tilde{f}_\pi(n) = b + sy(n-1), \tag{7.31}$$

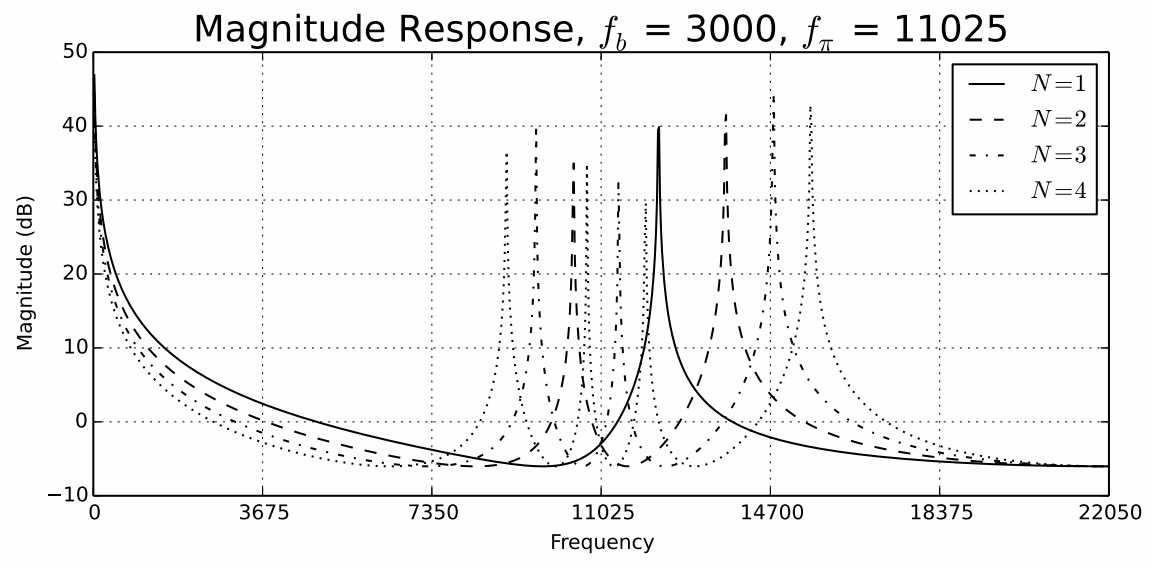

Figure 7.10: Example magnitude responses as N is varied while f_π and f_b are kept fixed, demonstrating centering around f_π and increasing flatness.

where s and b are user-defined scaling and bias factors, respectively.

In general, s and b should scale and offset $y(n-1)$ so that it falls into the range expected for $\tilde{f}_\pi(n)$. However, this is not strictly necessary for stability, as the cosine term in (7.7) will wrap any values into the range $-1 <= \tilde{d}(n) <= 1$. Thus, interesting results may be obtained by using other values for s and b.

7.5.3.1 Qualities of Musical Feedback Systems

A configuration like the one proposed in this section is very sensitive to particular combinations of s and b. It can therefore be difficult to assign meaningful parameters to such a system. Sanfilippo et al. give some terminology which is useful when describing the behavior of musical feedback systems like this one [21]. First, the system is *iterative* - it is self-sustaining and produces variations on initial conditions. Second, there is *coupling* between components of the system. All components of the system are of equal importance, and this equality can lead to a specific set of characteristic behaviors. Finally, the system is also capable of *self-organization*, in that the output tends to oscillate between two or more

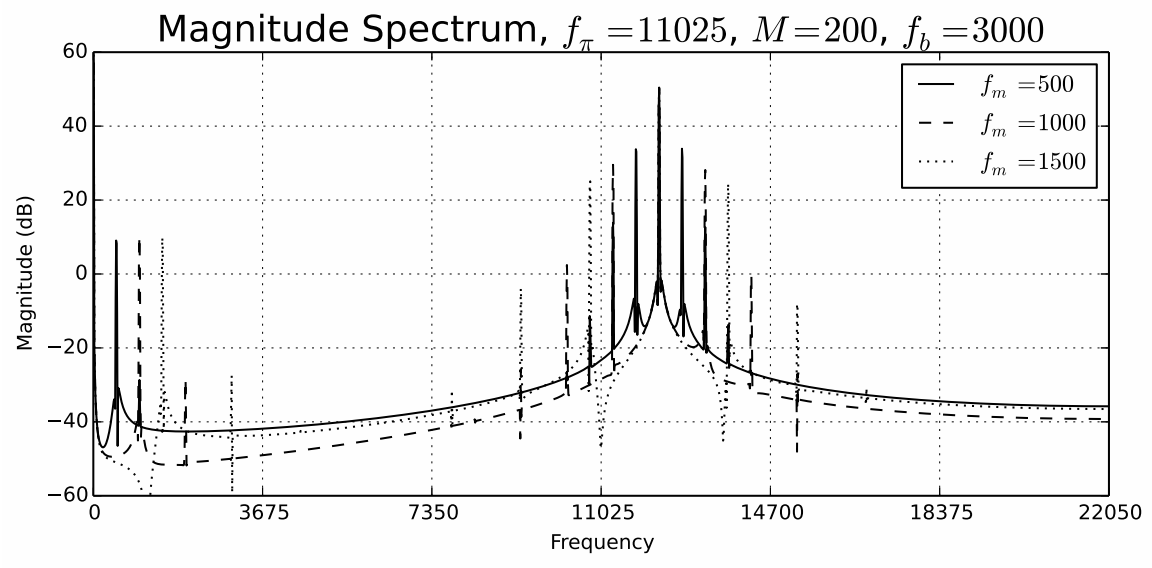

Figure 7.11: Output spectra of all-pass feedback network with time-varying parameters. The parameter f_m changes, while the others are fixed.

distinct types of behaviors. Due to the *circularity* of the time-varying feedback system, in which "effects are also causes," there is also an interaction between different musical attributes - loudness, pitch, and timbre are interrelated.

This inter-relatedness is easy to see when considering the feedback system described here: as the magnitude of the system's output at $y(n)$ increases, this causes a greater range of values for $\tilde{f}_\pi(n)$ - equivalent to increasing M in the case of sinusoidal modulation - and therefore a greater output bandwidth. Higher frequencies at the output - related to both pitch and timbre - cause a faster rate of change in $y(n-1)$. This is equivalent to increasing f_m and causes wider harmonic spacing around spectral components.

Practically, these systems often produce very high-amplitude output, which often has a DC offset, due to the unit-radius pole at DC. Therefore, it is often necessarily to scale and apply a high-pass or DC-blocking filter to the output.

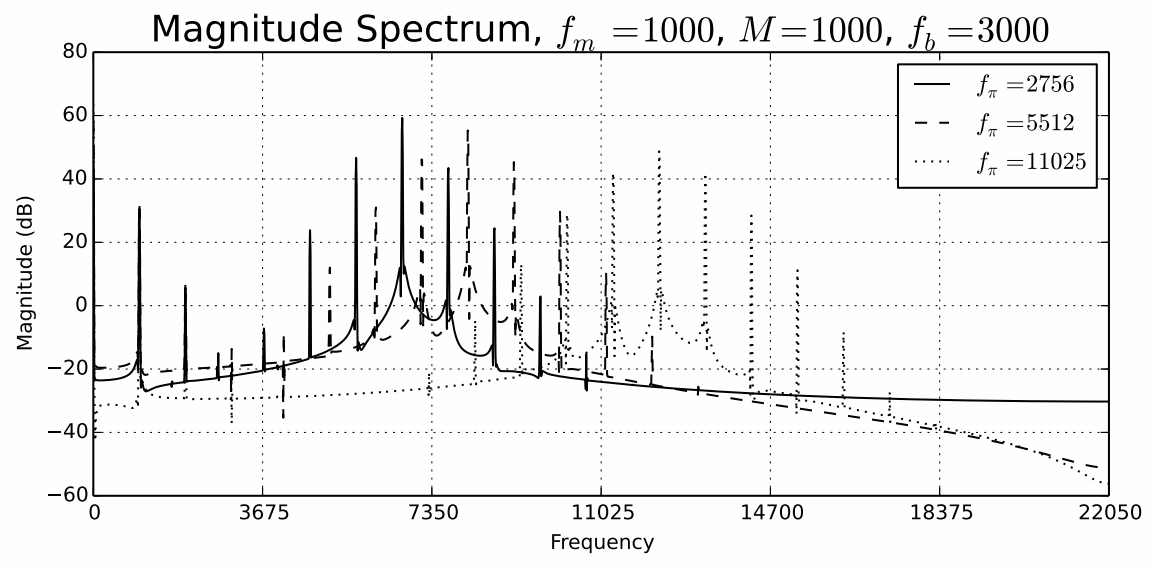

Figure 7.12: Output spectra of all-pass feedback network with time-varying parameters. The parameter f_π changes, while the others are fixed.

7.5.3.2 Sample Output

Consider a system where $\tilde{f}_\pi(n)$ is defined as

$$\tilde{f}_\pi(n) = 3333 + 1173y(n-1). \tag{7.32}$$

An example of the output of this system is shown in Figure 7.16. The characteristic output of this system consists of rhythmic "chirps" of increasing frequency, alternating with steady tones and short bursts of noise. The chirps are grouped into perceptible units by their duration. Most of the steady tones are short, such as those from 0.0" to 1.5", however others are longer, like those beginning at about 1.9". The rhythms are quasi-periodic, never repeating exactly - the sense is more of a series of iterative variations, rather than repetitions. Occasional interjections of noisy, over-modulated material occur, further challenging the sense of periodicity. There is a semi-periodic oscillation between three types of material: the rapid chirps, the steady tones, and the noise. Depending on the particular values of s and b, this self-organizing behavior can continue indefinitely or eventually settle into a

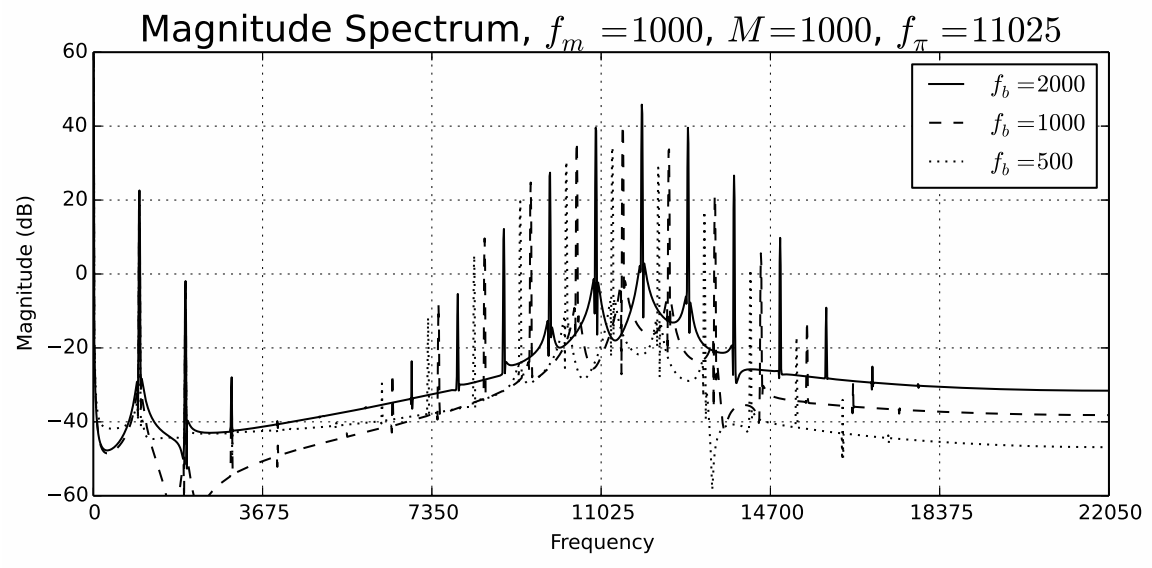

Figure 7.13: Output spectra of all-pass feedback network with time-varying parameters. The bandwidth parameter f_b changes, while the others are fixed. $M = 1000$, $f_\pi = 1000$, $f_m = 1000$.

more repetitive mode. It should be noted that the system is extremely dependent on initial conditions, and a given set of *s* and *b* will not necessarily produce the exact same results each time they are recalled.

Figure 7.17 shows another view of the characteristic behavior of this system. The values of f_b and *b* are the same as those used in Figure 7.16, while *s* has changed slightly (from 1173 to 1167). The change to *s* affects the rhythmic behavior of the system, as evidenced by the much shorter duration of the steady tones, and the increased prominence of noise. The harmonic structure changes as well, with the presence of more widely-spaced bands of emphasis (two are visible at approximately 800 and 2000 Hz), instead of the tightly spaced, clearly defined harmonics in Figure 7.16.

7.6 Conclusion

Time-varying all-pass filters have a variety of applications in signal processing and generative music. This chapter described issues of stability in time-varying all-pass filters

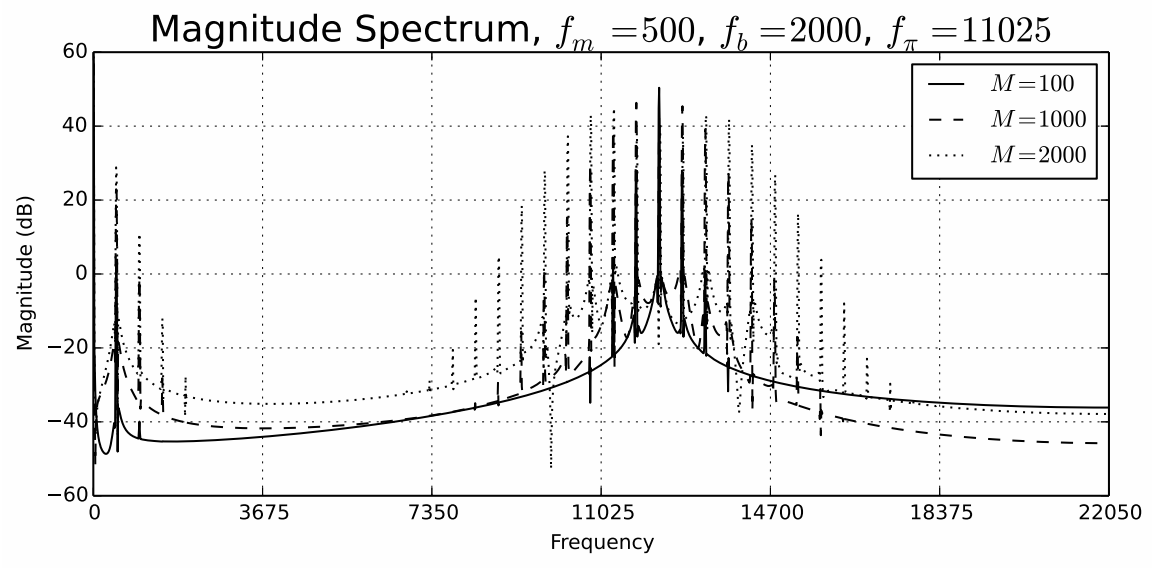

Figure 7.14: Output spectra of all-pass feedback network with time-varying parameters. The parameter *M* changes, while the others are fixed.

and the application of these filters to generative audio systems. Unfortunately, commonly used all-pass filter formulations are susceptible to instability when their coefficients are made time-varying. Two methods of showing this instability were used, though neither is useful in a real-time situation. Instead, a power-preserving matrix form of the second-order all-pass was presented. This version of the filter is stable when its coefficients are modulated.

The filter was used to extend a technique for applying frequency-selective phase distortion to signals, by incorporating a time-varying all-pass filter into a unity gain feedback network. These networks were studied in terms of their static frequency responses and their output spectra when the all-pass component was made time-varying.

Finally, an application was given which demonstrated the potential usefulness of the power-preserving all-pass filter in a generative audio system. A self-modulating feedback network was designed, in which a system's parameters were modulated by its own output. This type of system is capable of interesting, generative behavior, and will be studied further in larger compositional contexts. Possible expansions include more complex modulation schemes (for example. modulating *s* or *b* with signals from points in the feedback network),

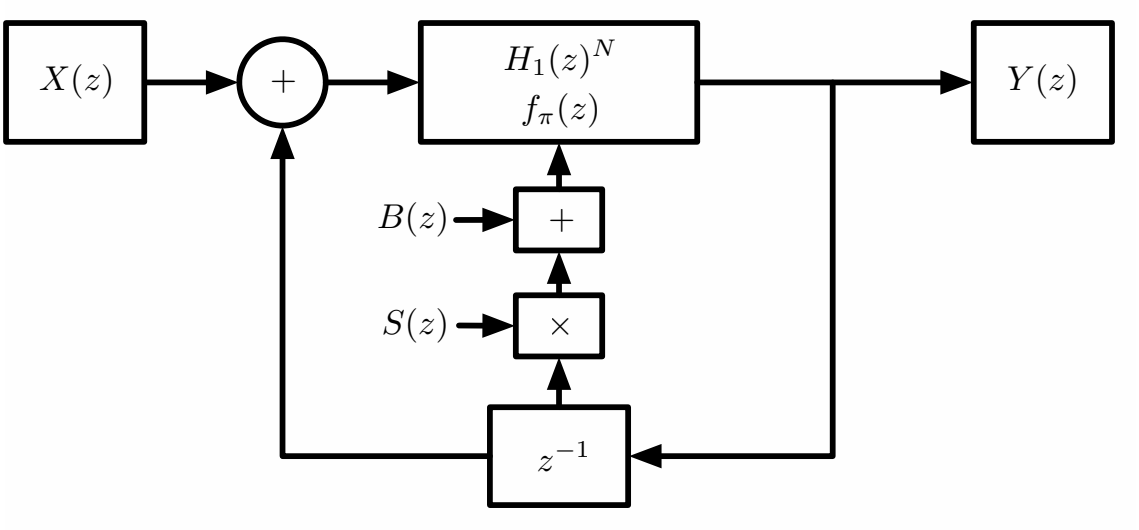

Figure 7.15: A high-level block diagram of a self-modulating all-pass feedback network.

or building feedback networks using multiple all-pass filters (or cascades) with differing modulation parameters.

7.7 Acknowledgements

The material in this chapter was originally published in:

Surges, G., and Smyth, T. 2015. "Generative Feedback Networks Using Time-Varying All-pass Filters." *Proceedings, International Computer Music Conference.* Denton, TX. 2015. The dissertation author was the primary investigator and author of this paper. This paper was awarded "Best Paper" at the 2015 International Computer Music Conference, and the 2015 Erickson Prize for Excellence Graduate Research by the Department of Music at UC San Diego.

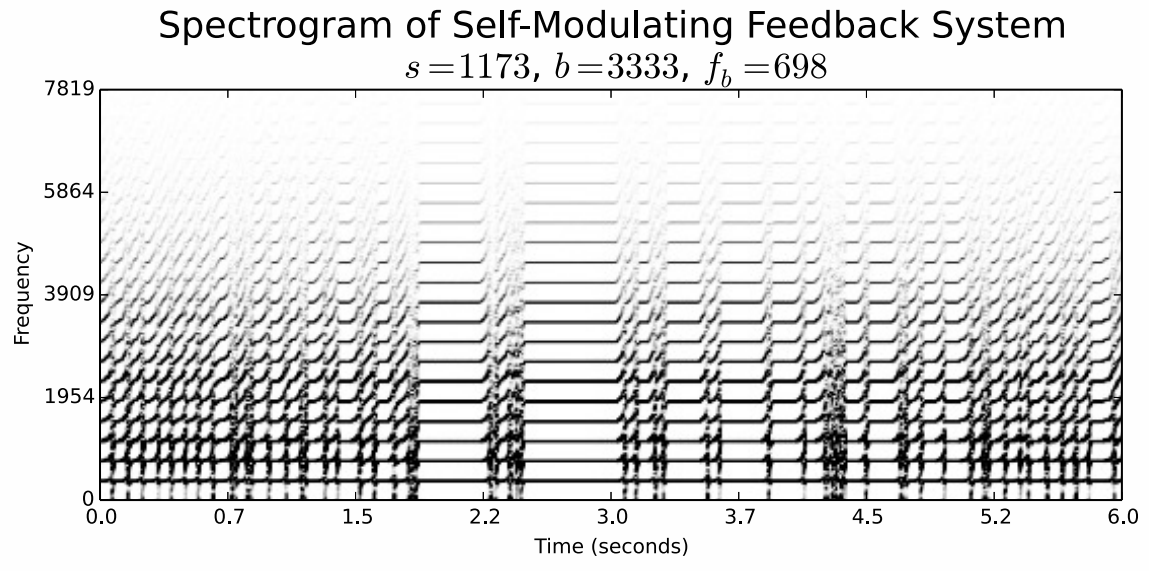

Figure 7.16: Spectrogram showing self-organizing behavior of self-modulating feedback system.

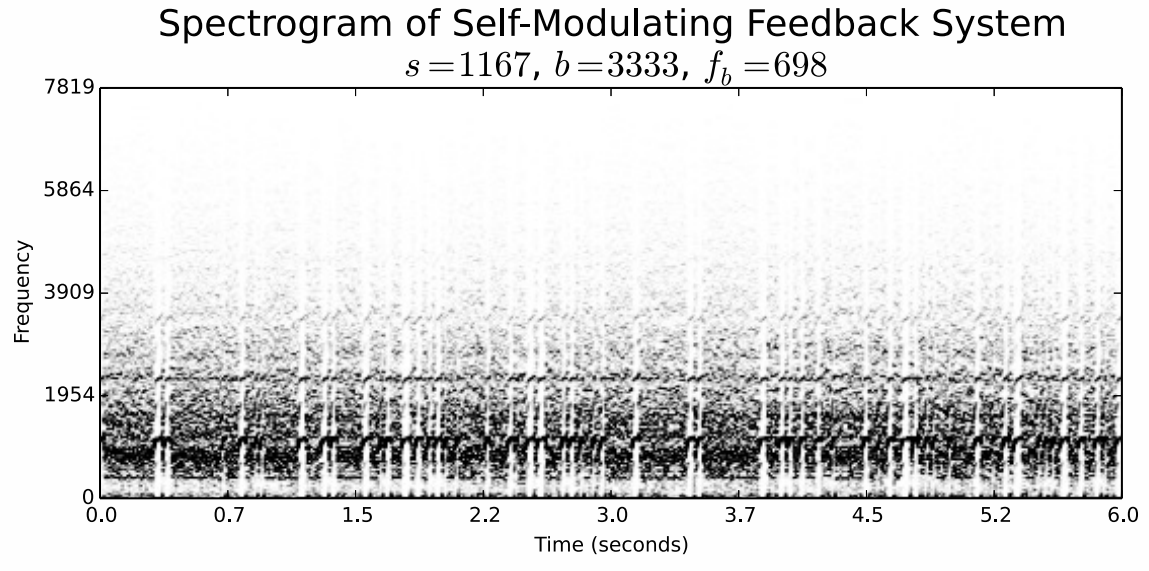

Figure 7.17: Spectrogram showing self-organizing behavior of self-modulating feedback system.

Chapter 8

Evaluation of a Generative Audio System

8.1 Introduction and Motivation

The discussion will now turn toward practical application and evaluation of the ideas of generative audio systems and computational aesthetics discussed in previous sections. This chapter describes a creative application in the form of a system that couples a generative audio system component with a real-time computational aesthetic analysis component. This analysis component allows the system to analyze its own output and is used to maintain a balance between order and complexity or repetition and novelty. The result is a system called *AAS-4*. In order to determine the usefulness of such a system, both the generative audio component and the computational aesthetic analysis component are evaluated in terms of the complexity of their output.

Chapter 2 introduced the generative audio system, and provided historical examples of systems that were capable of generating interesting sonic behavior. Most of these systems required the influence of a performer or composer. From a computational aesthetics standpoint, the role of this individual is to make the decisions which determine the aesthetic

worth of the music by manipulating the balance between order and complexity. Though David Tudor's electronic systems may have produced endlessly-sustaining variations on particular textures, it is unrealistic to expect that the systems would have known its subjective qualitative state (i.e. whether it had become too boring or, conversely, too dynamic, exciting or lively). The systems had no way of compensating for such an imbalance.Likewise, my *Feld* system (described previously in Chapter 4) exhibits a related problem: the system is capable of producing sectional contrast, but it relies on arbitrary timing mechanisms to produce sectional durations rather than any notion of aesthetics or balance.

Theories of computational aesthetics, along with psychological research on exposure and processing fluency were presented in Chapter 3. These theories point toward the role of repetition (order) and novelty (complexity) as significant factors in aesthetic experience. Many researchers in the field of computational aesthetics have posed and tested theories demonstrating the importance of a balance between these two factors, and the theories have been reinforced by psychological studies on the "mere exposure" effect. Research in the related field of Music Information Dynamics has produced multiple methods of calculating the information content of a musical signal. Since Music Information Rate (IR), described in Chapter 3, gives a measure of how repetitive the signal is at each moment, i.e. it indicates whether the current moment is a repetition of something from earlier, it has potential application in enabling a generative audio system to analyze its own output, and maintain some balance between order and complexity. Through this "self-listening" process, a system could make aesthetically informed judgments about when to make musical changes without relying on the influence or control of a human performer or composer.

The research aim of the *AAS-4* system described in this chapter is two-fold. The first part is the creative application of the use of self-modulating all-pass feedback networks, like those described in Chapter 7, in a generative audio system. The second is to make the system fully autonomous, so that no human input will be required either before or during

performance. This is achieved by using IR to enable the system to gain knowledge about its own output. At the time of this writing, this represents a novel use of computational aesthetic theories and analysis in real-time musical generation.

First, the self-modulating all-pass feedback network used in the *AAS-4* system will be described. The output of this generative audio system is passed through an additional signal-processing network, which will be described next. The network topology and modulation signal routing will be discussed, and example output from the system will be given. A novel adaptive parameter switching technique informed by computational aesthetics will be described. This technique uses real-time IR calculation to determine when system parameters should be switched, in order to minimize over-exposure or boredom in a listener. By determining when sonic material has become too repetitive, the *AAS-4* system can change the parameters of the generative audio system component to inject novelty into the musical output.

Following these descriptions of system components, both the sound generation capabilities and the adaptive switching technique will be evaluated. This evaluation will take the form of a computational analysis of signal complexity. IR will be used alongside a measure of Kolmogorov complexity to evaluate the utility of the system components.

8.2 The Generative Audio System Component

8.2.1 Network Topology

The generative audio system component of the *AAS-4* system consists of a configuration of all-pass filters in a feedback network similar to the one described in Chapter 7. The system is composed of two identical all-pass feedback networks, one of which is shown in Figure 8.1. Each network consists of four all-pass cascades, with each constructed of identical second-order all-pass filters. Within each network, the all-pass cascades are divided

into two pairs. In Figure 8.1, the top two cascades (labeled $H_1(z)$ and $H_2(z)$) form one pair, while the bottom two (labeled $H_3(z)$ and $H_4(z)$) form the pair. Each pair has a feedback path from its output to its input.

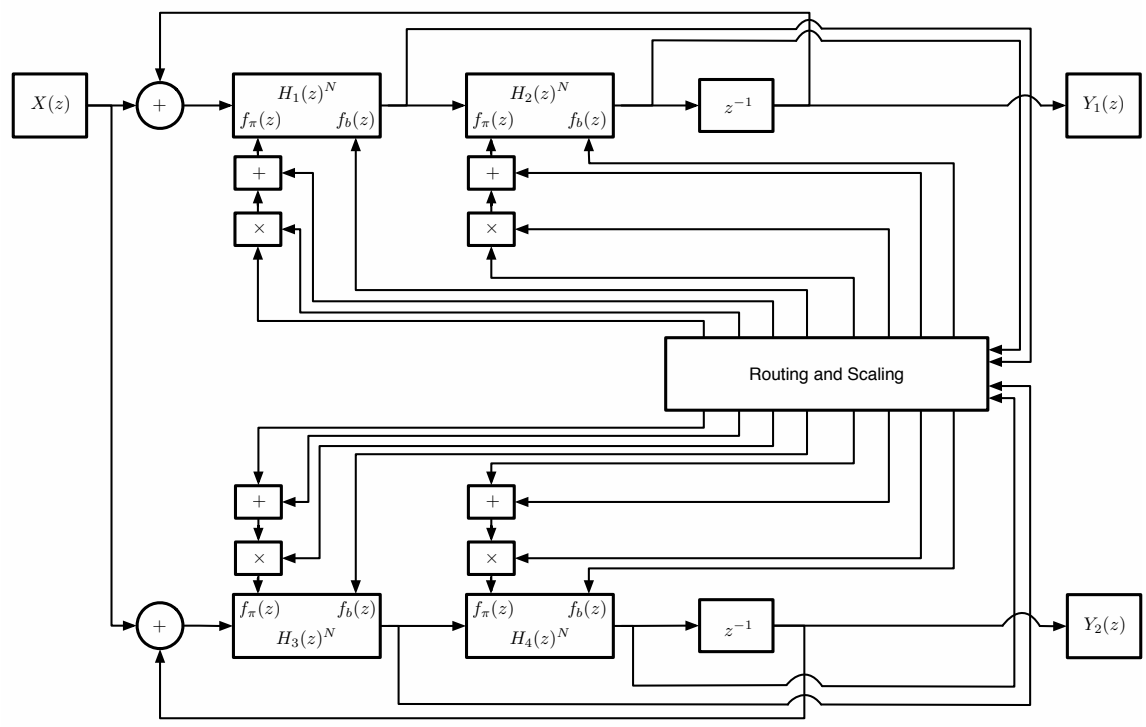

Figure 8.1: The *AAS-4* feedback network topology. $H_1(z)$ through $H_4(z)$ are time-varying all-pass filter cascades.

8.2.2 Modulation Signal Generation

The modulation signals for each all-pass cascade, labeled $f_\pi(z)$ and $f_b(z)$ in Figure 8.1, derived from other points in the feedback network. The cascades do not drive each other directly, but instead are coupled in such a way that the output of each all-pass cascade can be transformed into a set of modulation signals for the others.

In order to map these output signals to parameters, the output of each all-pass pair is subjected to routing and scaling. In the figure, the box labeled "Routing and Scaling" represents this process. Each of the parameters of the all-pass cascades - *s* (represented

by the X box in Figure 8.1), b (the $+$ box), and $f_b(n)$, along with the modulation source signal itself $m(n)$ - is derived from this routing and scaling process. For each parameter, the source signal $m(n)$ is chosen from the set of individual all-pass outputs. This source modulation signal is then scaled and biased in order to convert it from the range $\{-1,1\}$ to a range suitable for the parameter it will modulate. Following scaling and biasing, the signal is passed through a one-pole low-pass filter with a cutoff frequency parameter f_{lp} in order reduce high-frequency modulation components. In this manner, the output from any of the four all-pass cascades can be mapped to any of the modulation signals $\{b(n)$, $s(n)$, $f_b(n)$, $m(n)\}$ of any of the all-pass filters in the network at the next time step. Reasonable ranges for the scaling, biasing, and filtering parameters were determined empirically, favoring those which produced rich results while avoiding static behavior. These bounds were found to be $0 < s \leq 2000$, $0 < b \leq 8000$, and $0 < f_{lp} \leq 400$.[1]

8.2.3 Output of the Generative Audio System

Figure 8.2 and Figure 8.3 show spectrograms of example output of this configuration. The top plot in both figures shows the full recording, while the bottom plot shows a shorter excerpt demonstrating details of the behavior. The parameters were randomly chosen within the ranges described above. Both excerpts are characterized by the presence of a few prominent pitches or harmonics, with varying amounts of "jitter" around those frequencies.

The first example, shown in Figure 8.2, features a pair of prominent spectral components which rapidly oscillate around a pair of central frequencies. As shown in the bottom

[1] From an implementation standpoint, care must be taken to ensure that the software routing of these modulation signals does not introduce unwanted delays, as this can radically alter the behavior of the output signals. During the implementation, it was found that Max/MSP introduced a delay equal to the signal buffer size when a feedback loop was suspected. This behavior is not particularly well documented, but is discussed on the Cycling 74 forums: `https://cycling74.com/forums/topic/send-and-receive-not-sync/`. PD, on the other hand, allows for more consistent behavior and explicit control over sub-patch block size, and so proved to be the better choice for the implementation of this particular structure. Placing the `send~` and `receive~` objects in sub-patches helps enforce the correct behavior, as explained in the PD help patch `G05.execution.order.pd`

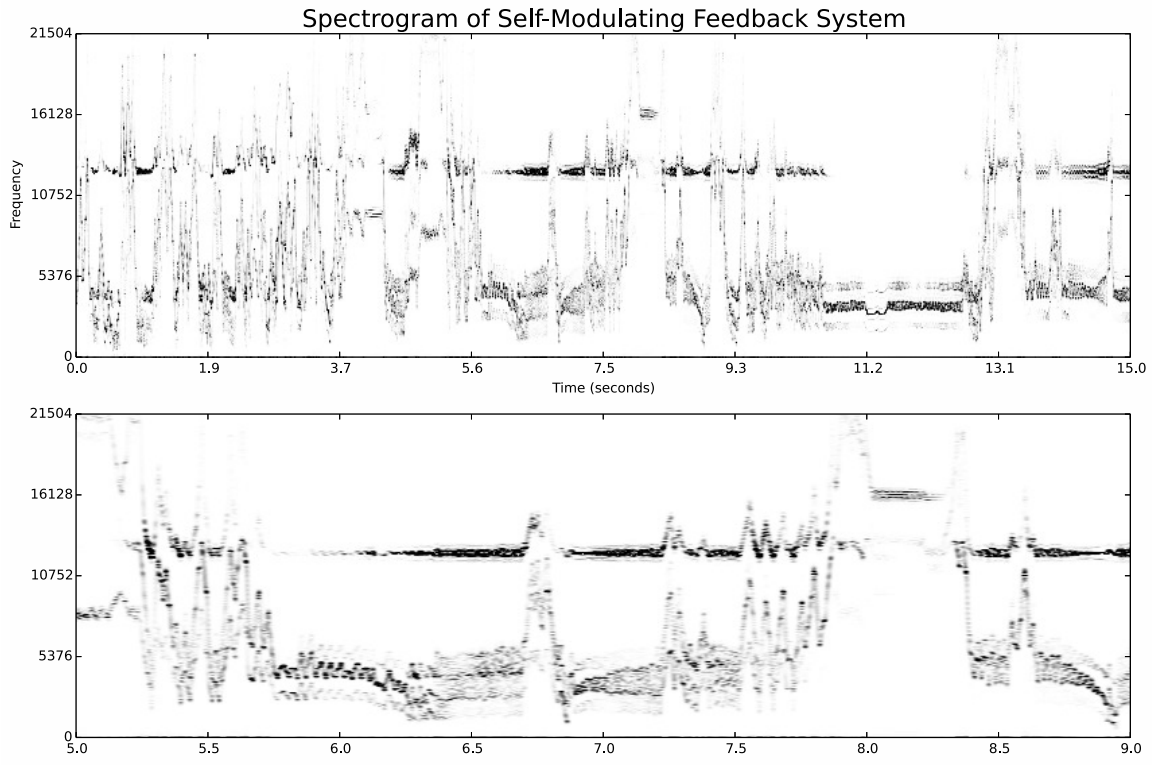

Figure 8.2: An example spectrogram from *AAS-4* all-pass filter feedback network. Bottom plot shows higher time resolution excerpt from top plot.

plot, the audio-rate modulation of all-pass filter parameters causes these signal components experience rapid frequency modulation between larger jumps in frequency. The temporal behavior is chaotic and unpredictable, but has a sense of uniformity and exhibits motivic, self-organizing behavior. There are areas of high activity (from 0 - 3.7 seconds, for example) as well as areas of low activity (from 11 seconds to just before 13 seconds), giving the material a sense of pacing and phrase structure.

The second example, shown in Figure 8.3, is less rhythmically active and features a set of four prominent spectral components. These components experience simultaneous moments of oscillation around central frequencies, but with differing modulation depths. Rhythmically, the oscillations are organized into clusters of semi-periodic rhythms of differing tempi. For example, contrast the rapid oscillation at around 3 seconds with the slower, more “spiky” ones from 5 - 7 seconds. Finally, at the end of the example (around

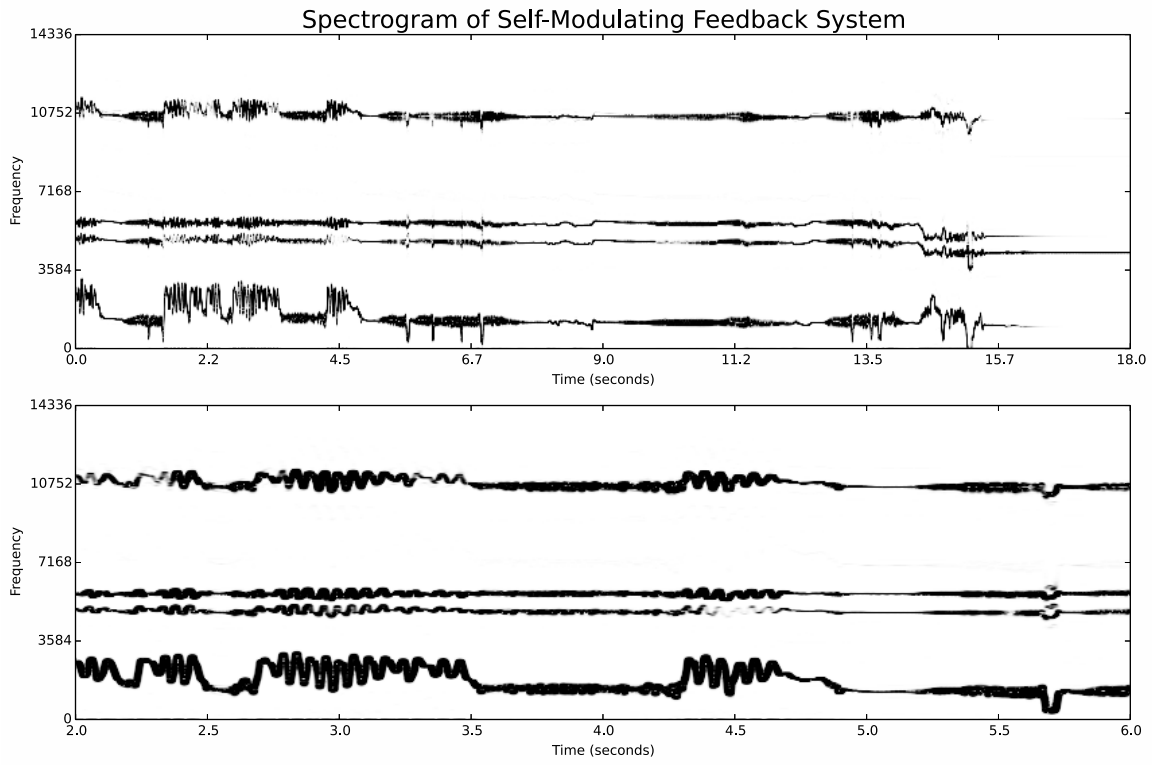

Figure 8.3: Another example spectrogram from *AAS-4* all-pass filter feedback network. Bottom plot shows higher time resolution excerpt of top plot.

15.7 seconds), the system enters another stable state characterized by the presence of a single, steady tone. Many parameter sets will produce such emergent behavior, where a particular transient state may continue for a significant amount of time before abruptly switching to another.

The *AAS-4* system derives four output channels from the two feedback networks described here - each network provides a pair of output channels. Though the two outputs from each network are similar, they are different enough to function as separate voices. These output channels are each then sent to a separate signal processing chain for further rhythmic and timbral modification, which helps to reinforce the sense of independent voices. Before describing these effect chains, the computational aesthetic analysis component of the system must be described.

8.3 The Computational Aesthetic Analysis Component

It has been demonstrated, in Figures 8.2 and 8.3 for example, that the generative audio system configuration described above is capable of producing dynamic audio output. Although a given set of parameters will produce audio output that evolves over time, the range of textures, timbres, and gestures produced is limited. In order to produce a more varied output, the system needs a means of changing its own parameters. This section describes a method of determining when to reconfigure the generative audio system with a parameter change, in order to maintain a balance between order and complexity in its output. The desired balance is informed by the "mere exposure" effect and the interactions between repetition, affect, and boredom described in Chapter 3.

Chapters 3 and 5 showed how the Audio Oracle (AO) algorithm can be used to perform real-time Music Information Rate (IR) analysis of an audio signal. IR fits nicely with the theories of computational aesthetics described in Chapter 3, as it attempts to quantify the balance of order and complexity in the signal. Moments where the IR function is high are those which can be understood as a repetition of previous material, while those where the IR function is low are those with a high amount of novelty. This section will describe the use of the AO algorithm and a real-time measurement of IR in the *AAS-4* system.

8.3.1 Audio Oracle Analysis

To briefly recapitulate the discussion in Section 3.1.1, the AO algorithm is able to model an audio signal in terms of its repetition structure. The resulting data structure resembles Figure 5.2. Each node represents a single audio analysis frame - a feature vector. The lower arcs ("suffix links") connect repeating patterns and the upper arcs ("transitions") represent alternative continuations of patterns. An AO is built in a sequential fashion, from input feature vectors extracted from an audio signal. As each new feature vector is received as input, the AO algorithm recursively checks for similar vectors which have been previously

received, moving backward in time through the data structure. Feature vectors connected by suffix links are compared, and if their difference is below a threshold θ, they are considered to be similar. The resulting structures, like the one in Figure 5.2, represent the location of these repetitions through suffix links. The AO construction process is described in greater detail in [75].

8.3.2 Compositional Applications of IR Analysis

In the *AAS-4* system, the `ao` object described in Chapter 5 is used to model the interaction between repetition and boredom in a listener. As the studies summarized in Chapter 3 show, as a listener experiences repeated exposures to a sound stimulus, their affective response to the sound becomes more positive [47]. At a certain point (after approximately 10 exposures), however, boredom begins to have an effect, and the affect becomes more negative. The IR function is used as a measure of repetition over time, which - for the purposes of the current discussion - is considered to be related to repeated exposures to a stimulus. This model is then used by an adaptive timer to determine when the parameters of the generative audio system should be changed, in order to produce new musical material.

8.3.2.1 An Adaptive Timing Mechanism

Each of the four output channels of the generative audio system is sent to an independent AO analysis module. The analysis uses Mel-Frequency Cepstral Coefficients as its feature. As more and more signal is analyzed, the AO structure and IR function change. It is therefore necessary to periodically recalculate the IR function. In this case, a 500 ms rate of recalculation was chosen and each 500 ms long segment of music is considered to be a single exposure event. The resulting IR analysis is normalized to the range 0 - 1.

At each IR update, the most recent point in the IR function is used as the next sample in a secondary signal sent to an adaptive timer. This resulting signal is smoothed

with a low-pass filter, in order to avoid rapid jumps caused by the "blocky" character of IR functions. After filtering, the signal is compared to a threshold value τ, that determines whether the current moment is similar enough to some past moment to count as a meaningful repetition. The threshold value can be set to an decimal value in the range 0 - 1. If the smoothed IR signal crosses the threshold, as a result of repetition detected in the signal, a boredom counter is incremented. The boredom counter stores the number of sequential repeated exposures. As the repetitions continue, the counter continues to increment. When 10 repeated exposures have occurred, the adaptive timer determines that the listener is beginning to become bored, and a change in the generative audio system's parameters is triggered. However, if the repetitions stop before this threshold count is reached, the counter resets, as the adaptive timer assumes that the listener is no longer bored.

It was found that setting the threshold $\tau = 0.3$ produced a good compromise in the system's sensitivity to novelty. Theoretically, any IR value above 0 should be regarded as a repetition, but in practice it was found that short repetitions (producing a low, but non-zero IR value) were functionally equivalent to a new frame. In other words, a repetition consisting of two signal frames totaling about half of a second will probably not be perceived as such by a listener. As such, it should not be counted as a repetition by the adaptive timer.

In tracking the number of repeated exposures, the computational aesthetic analysis models the interaction between exposure and boredom. This is based on the theory, described in Chapter 3, that exposure and boredom are interrelated, and that boredom begins after around 10 exposures [47]. The adaptive timer simulates this process. Small numbers of repetitions are allowed to occur; but after 10 sequential appearances of repeated material, further repetitions trigger a change in the configuration of the synthesis network. In other words, when the current audio output has been a repetition of previous material for long enough, the generative audio system is reorganized in a way that is conducive to generation of novel material. The boredom trigger is sent, via the OpenSoundControl protocol, to a

Python script which runs in parallel with the PD synthesis patch.[2]

This Python script, `make_params.py`, handles the 64 generative audio system parameters, corresponding to routing, scaling, biasing, and filtering parameters for each of the four output channels. Multiple sets of parameters are stored simultaneously. These parameters are randomly assigned at script start-up, and constrained within ranges described above. Each time a boredom trigger received from the adaptive timer, one of the stored parameter sets is retrieved. One-third of the time, the parameter set is used as-is; the other two-thirds of the time, four of the 64 parameter values in that set are replaced. A parameter set produces related material each time it is recalled, while also varying slowly over time. The location of this adaptive switching method in the context of the larger system is shown in Figure 8.4.

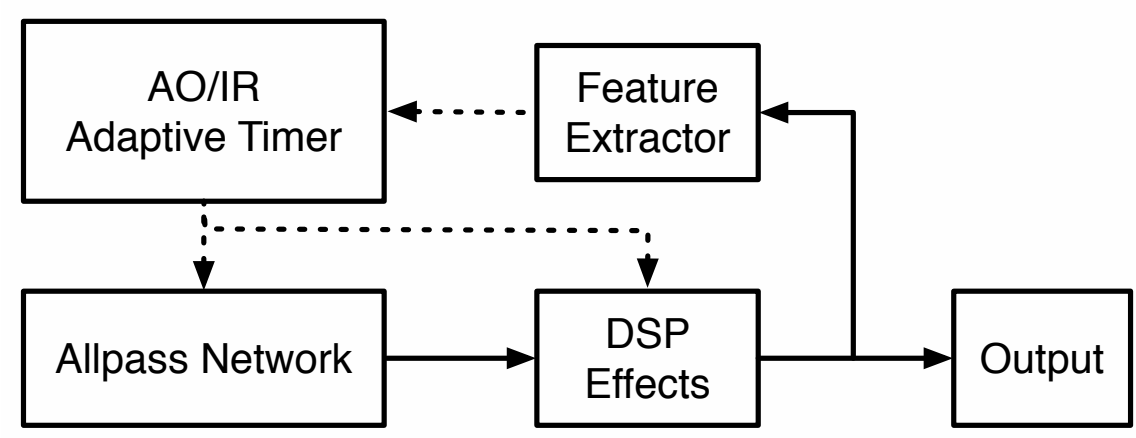

Figure 8.4: The signal processing network, showing the location and configuration of the adaptive, IR-based parameter switching mechanism. Dotted lines indicate control messages.

8.4 Additional Signal Processing

As mentioned above, the four generative audio system output channels are sent to separate signal processing chains, for further rhythmic and timbral modification. The effects

[2] In this case, computational demands are extremely low, so Python is an apt solution.

chains here are similar to those used in *Feld*. These modules are automated according to the boredom trigger derived from the IR function. When a trigger is received, each signal processing module has a probability of becoming active, otherwise it will be bypassed for that section. After a trigger is received, the parameters for active modules are ramped to new, randomly chosen settings. The signal processing modules include an amplitude dependent gate, designed to dramatically expand the dynamic range of the generative audio system output; pitch-shifting, for expansion of the registral range of the output; two recirculating delay lines, one with short times and one with long; ambisonic spatialization, for dynamic panning and easy expansion to multi-channel output formats; and artificial reverberation for a sense of depth and to emphasize the percussive aspects of the generative audio system output.

This modular, switchable design was inspired by the one used in the *Feld* system described in Chapter 4 and provides a computationally efficient and flexible method for applying timbral and temporal modifications to the input signal. The all-pass network used here provides a richer source of material than does the analog synthesizer of *Feld*, and the desire was to showcase this material with slight accentuation, rather than large modifications. Ultimately, it was decided to use fewer processing modules, so that the sounds from the generative audio system would remain relatively "pure" and their inherent timbres kept intact.

8.5 Evaluation

The following section will describe an approach to the objective evaluation of both the output of the generative audio system used in the the *AAS-4* system and the usefulness of the adaptive, IR-based timing method (called the adaptive timing method here). These techniques will be evaluated using a measure of signal complexity obtained during IR calculation, as well as by calculating the total IR of the generated audio signals. In addition,

the Kolmogorov complexity of the signals will be estimated and used to measure the amount of order in each. Prior research using both of these approaches was discussed in Chapter 3.

Holopainen argues that generative audio systems (which he refers to as "autonomous instruments") are best evaluated using an objective measure of complexity [10]. Since it is straight-forward for a system to obtain simple or trivial output and behavior, like a sinusoid, the ability to produce complexity is the hallmark of a sophisticated generative audio system. Many of the theories about computational aesthetics discussed in Chapter 3 consider complexity to be a primary component of aesthetic experience. As Streich argues in [104], complexity is related to the amount of effort a listener must expend in order to understand the music, and therefore can also be related to the ideas about processing fluency and affect found in Chapter 3.

The adaptive timing method will be evaluated for its ability to maintain a balance between order and complexity. The method should produce a signal with an amount of complexity which, as a minimum requirement, falls in-between white noise (maximal complexity) and silence (minimal complexity). The method should also produce output which is more complex than other methods of determining when to make parameter changes. A higher amount of complexity can be interpreted as an indication that the system is limiting the amount of repetition in favor of more variety in its output.

It must be emphasized that the evaluation here is focused on analysis of objective characteristics of the system's output. The aim is not to consider notions of computational creativity, intent, or machine musicianship. As Ariza writes, in his article on the misapplication of techniques like the Turing Test to generative music systems and the evaluation of computational creativity, "generative music systems, as systems within problem-seeking domains, likewise have no criteria for testing correct answers [105]." Machine-embodied artistic intent is not necessarily a requirement for musical experience, as the listener can provide the necessary intent [105]. The goal here is to evaluate those aspects of the system

that can be objectively evaluated.

8.5.1 Audio Oracle Evaluation of Synthesis Output Complexity

If the generative audio system described in this chapter is to be used to generate music, then it should produce sounds and structures with a complexity level similar to that of real music. The *AAS-4* system generates order by creating repetitive materials as it recalls particular collections of parameter settings. By mostly utilizing recurring settings which produce complex and evolving textures and timbres, the system will automatically create a level of balance between order and complexity. This section will describe a method of objectively measuring the complexity of the system's output using IR calculations. This complexity will then be compared to that measured from other musical examples.

In [10], a measure called spread of features (SOF) was used to quantify the complexity of an audio signal. The SOF consists of the average spread over a set of features extracted from the signal. The features used were zero-crossing rate (ZCR), voicing, spectral entropy, inverse crest factor, spectral flux, and spectral centroid. This measure was applied to several audio signals, including a sinusoid, pink noise, a chirp signal, speech, Xenakis' *S.709*, and Reich's *Pendulum Music*. It was found that the sinusoidal signal and pink noise had the lowest complexity, while music and speech had the highest.

Rather than the SOF, a measure of complexity related to IR calculation will be used here. Equation 3.5, in Chapter 3, showed how IR is calculated from the difference between two measures of complexity: the unconditional complexity (or entropy) $C(x)$ and the conditional complexity $C(x_n|x_{past})$. The unconditional complexity measures the total complexity of a signal, and the order in that signal can be measured from the difference between the unconditional and conditional complexities [45]. Another way to understand the unconditional complexity is that it describes the size of the "encoding alphabet" used to compress a signal, and is measured in bits. Each unique frame of audio in the signal

increases the size of this alphabet, corresponding to an increase in overall signal complexity. Therefore, we can use $C(x)$, the unconditional complexity, as a convenient measure of the total complexity of a signal. Since the complexity increases with the length of the signal, as described in Algorithm 1, the final value in the array H_0, obtained during the IR calculation, can be used that as a measure of the total signal complexity.

Tables 8.1 and 8.2 show the results of applying this form of complexity analysis to a variety of signals. All results were calculated by building an AO structure from a 20 second long clip of 44.1k audio, using feature vectors extracted from blocks of 2048 samples with a 50% overlap. The features used were spectral centroid, MFCC, and zero-crossing rate, calculated using the Bregman Audio-Visual Information Toolbox.[3] The signals used were silence, white noise, a 440 Hz sinusoid, a sinusoidal chirp, an excerpt from David Tudor's *Neural Network Plus* [106], and sample output from the generative audio system component of the *AAS-4* system. The output of the generative audio system was recorded before the additional signal processing effects were applied, and the parameter set (the collection of scaling and biasing terms described above) did not vary over the duration of the excerpt. There are two forms of complexity analysis performed here. Table 8.1 shows the results obtained using the first, the automatic IR threshold calculation process described in Section 5.6.1.1 and Chapter 5. This results in an ideal quantization scheme and representation tuned to each individual signal.

The silence, white noise, and sinusoidal signals were found to have low levels of complexity, while the chirp and musical examples have higher levels.[4] While it is expected that the white noise would have the highest complexity - as it has extremely high entropy - the complexity is calculated using feature extractors operating on blocks of samples, causing

[3] `http://digitalmusics.dartmouth.edu/~mcasey/bregman/`

[4] Since the continually increasing frequency of the sinusoidal chirp covers the entire range of the zero-crossing rate feature, and directly impacts the spectral centroid and envelope, it is not surprising that the complexity of this feature rivals that of the musical examples - there are simply more values to encode than a static sinusoid. The centroid measure is clearly erroneous for the silent signal as well.

Table 8.1: Complexity results for frame size of 2048 samples and automatic Audio Oracle threshold detection. Highest two values for each feature in bold. Spectral centroid measure for silent excerpt is assumed to be a false reading.

Feature	Silence	Sinusoid $C(x)$	Noise $C(x)$	Chirp $C(x)$	Synth $C(x)$	Tudor $C(x)$
Centroid	*6.492*	5.907	6.459	6.15	**8.214**	**7.907**
MFCC	1	3.7	4.954	6.322	**6.409**	**6.34**
ZCR	1	2.585	3.7	**5.492**	5.542	**6.426**
Total	8.492	12.192	15.133	17.964	**20.165**	**20.673**

the attributes of the noise signal to be averaged over time. Summing the complexities across features yields similar results, with the musical excerpt yielding the highest complexity, and the synthesized material in close second. The results shown in Table 8.1 are also plotted in Figure 8.5. This figure shows the increase in complexity from test signals to generative audio system output and actual music. These results show that, in terms of complexity, the output of the generative audio system is comparable with real music.

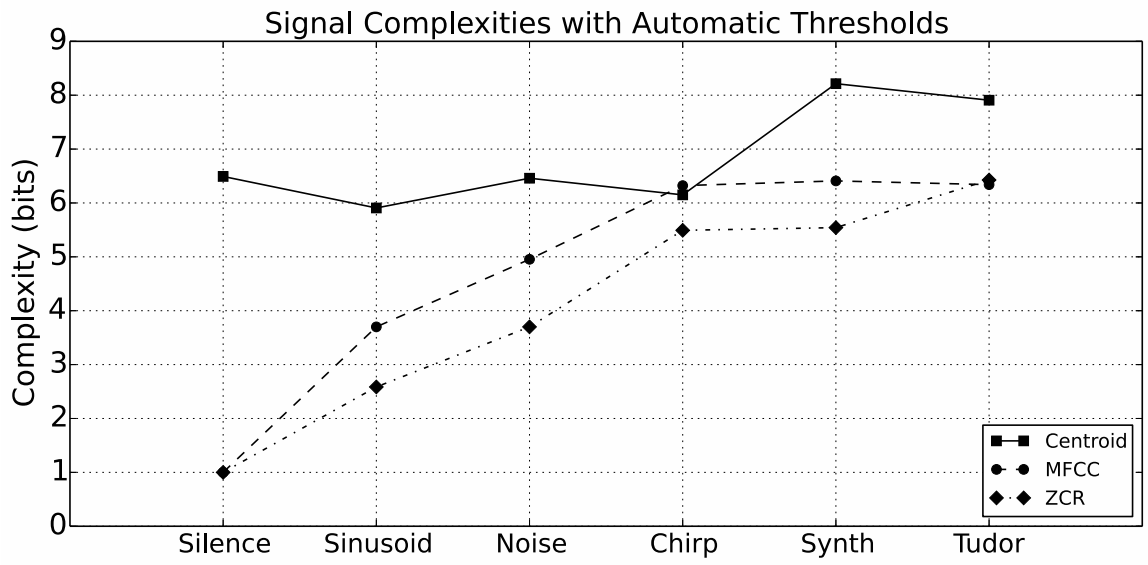

Figure 8.5: A plot of the data shown in Table 8.1, measuring the signal complexity using automatic threshold detection. Signal names are shown along the x-axis, signal complexities along the y-axis. Each line shows the change in complexity for a signal feature across the six signals.

It might be argued that allowing the complexity of each signal to be calculated according to the measured ideal threshold for that particular signal makes it difficult to

Table 8.2: Complexity results for frame size of 2048 samples and manual threshold θ setting. Highest two values for each feature in bold. Spectral centroid of silent excerpt is assumed to be a false reading.

Feature	θ	Silence	Sinusoid $C(x)$	Noise $C(x)$	Chirp $C(x)$	Synth $C(x)$	Tudor $C(x)$
Centroid	10	*7.401*	4.17	7.516	7.492	**8.994**	**8.916**
MFCC	0.2	1	2.322	6.109	5.615	**7.901**	**7.295**
ZCR	1	1	2.585	6.304	5.492	**8.484**	**7.972**
Total	n/a	9.401	9.077	19.929	18.599	**25.379**	**24.183**

compare complexity between signals. Since the signal features are not quantized according to the same scheme, they may not be compatible measurements for comparison. The results in Table 8.2 address this argument. Instead of using automatic threshold detection, like in table 8.1, thresholds were chosen manually for each feature and kept uniform across signals. Centroid, measured in Hz, uses a threshold of 10 (Hz); MFCC, with coefficient values ranging from 0-1, uses a threshold of 0.2; and zero-crossing rate, measured in number of crossings per block, uses a threshold of 1.

The results of these measurements are similar to those in Table 8.1, but the musical signals are more consistently the most complex. Across all three features, the synthesized excerpt and the audio recording are more complex than all other signals, and are close together in complexity. The results in Table 8.2 are plotted in Figure 8.6. Like the previous example, the general increase in objective complexity as the signals move from silence to sinusoids to musical signals is clearly visible. Again, the generative audio system output and the musical example are very close in complexity, with a marked drop-off when comparing them to the chirp signal.

8.5.2 Kolmogorov Complexity Analysis of Synthesis Output

In [55], image compression algorithms are used to estimate the complexity and a measure of the aesthetic value of paintings. Using a compression algorithm results in an estimate of a measure of complexity called Kolmogorov complexity (referenced in

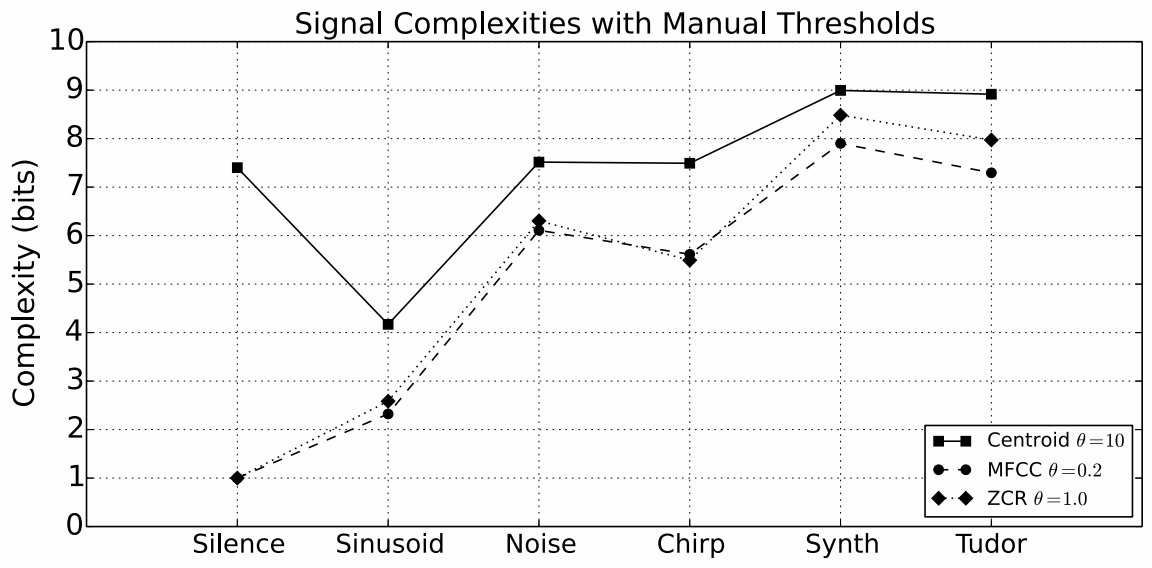

Figure 8.6: A plot of the data shown in Table 8.2, measuring the signal complexity using manual thresholds. Signal names are shown along the x-axis, signal complexities along the y-axis. Each line shows the change in complexity for a signal feature across the five signals.

Chapter 3), which corresponds to the size of the most compressed version of a string. According to Rigau, the order in an image can be measured by calculating the normalized difference between the maximum image size and the Kolmogorov complexity of the image:

$$M_K = \frac{NH_{rgb} - K}{NH_{rgb}}, \tag{8.1}$$

where N is the number of pixels, K is the Kolmogorov complexity, and H_{rgb} is the entropy of all pixels [55]. In [55], each pixel is represented by three values - corresponding to red, green, and blue channels, each of which is stored in 8 bits (256 values). The total entropy H_{rgb} of the pixels is the sum of entropies for each color channel, computed from probability distributions estimated for those channels:

$$H_{rgb} = H(X_r) + H(X_g) + H(X_b), \tag{8.2}$$

where X_r, X_g, and X_b are the probability distributions of the color channels.

Table 8.3: Results of calculating order M_K from Kolmogorov complexity K estimated using FLAC algorithm on default settings.

File	Input Size (bytes)	K (bytes)	Compression Ratio	H_s	M_K
Noise	1764044	1774356	1.006	0.997	-0.009
Silence	1764044	324621	0.184	0.662	0.822
Sine	1764044	578098	0.328	0.957	0.658
Chirp	1764044	542726	0.308	0.985	0.688
Synth	1764044	1229650	0.697	0.970	0.281
Tudor	1764044	1025554	0.582	0.895	0.350

This method of obtaining a measure of order can be applied to an audio signal as well. Although the .PNG and .JPG algorithms used in [55] work well for image data, they are not appropriate for audio signals. The FLAC compression algorithm, a method of loss-less audio compression based on linear prediction, can be used to measure the compressability of audio signals [107]. The FLAC audio compression was chosen for its high-quality performance and ease of use. In this case, (8.1) becomes:

$$M_K = \frac{NH_s - K}{NH_s}, \tag{8.3}$$

where H_s is the entropy of a time series of 16-bit audio samples, and N is the number of samples. In the case of audio, K can be estimated using an audio compression algorithm. In the examples here, K corresponds to the size in bytes of the output of the FLAC compressor using default compression settings.

The results of computing the order M_K for the sample audio files are shown in Table 8.3. As expected, the noise file has the lowest amount of order, with a value of $M_k = -0.009$.[5] The silent signal has the highest order, with $M_K = 0.822$. It is assumed that K is inflated here, due to header information added by the FLAC algorithm, resulting in M_K lower than 1. Both the sine and chirp signals have relatively high amounts of order: 0.658 and 0.688 respectively. The musical examples, on the other hand, are more complex. The

[5]The fact that M_K is negative is an expected result of constant information added by the compression algorithm and is addressed in [55].

synthesis output example has an order of 0.281 while the excerpt of Tudor's composition has 0.350. As expected, both musical examples fall between the high complexity of noise and the low complexity of the sinusoid. The results are plotted in Figure 8.7. The discrepancy between white noise complexity measurements obtained from the AO complexity estimation method and the Kolmogorov complexity is also expected. Whereas the AO method uses feature vectors obtained across multiple samples, thus obtaining averaged results, the Kolmogorov complexity considers the signal as a time-series of discrete, uncorrelated samples.

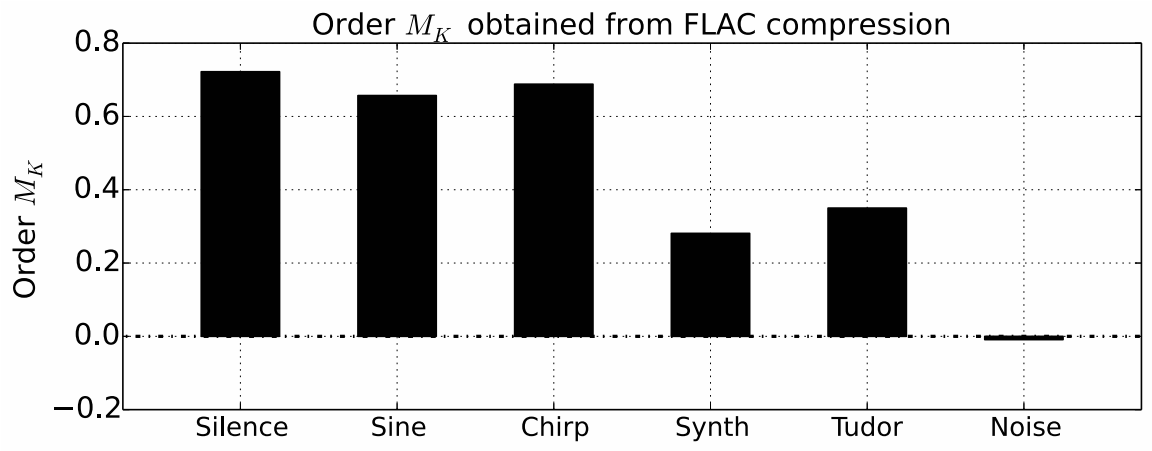

Figure 8.7: A plot of the data shown in table 8.3, showing signal order M_K obtained using FLAC compression to estimate the Kolmogorov complexity K. Signal names are shown along the x-axis, signal orders along the y-axis.

Clearly, this objective analysis is not the whole picture, and it is not the author's aim to disregard subjective aesthetic analysis. However, the results here suggest that the levels of complexity found in various features extracted from the output of the generative audio system are comparable to those extracted from real-world music. Therefore, the use of such a generative audio system in a compositional context is merited. Order can be relatively easily obtained through repetition, but complexity is less easily obtained. Starting with a system that is capable of generating complexity is advantageous, and it is likely that the output of such a system will suggest possibilities for its own further development as

particular characteristics of the output are either refined and emphasized or modified and suppressed.

8.5.3 AO Evaluation of the IR-based Adaptive Timer

The following section will describe a similar experiment designed to determine whether the adaptive, IR-based timer described above is a useful method of balancing order and complexity. In order to do so, it is necessary to have some method of objective analysis in order to enable comparison between the proposed method and alternatives. This section and the following will describe the results of subjecting the adaptive timing method, along with two alternative methods of parameter change, to evaluation using the same computational aesthetic measures employed in Section 8.5.1.

The output of the generative audio system using the adaptive timer to trigger parameter changes was compared with that of two other systems, each using a different timing mechanism. The first of these other methods changed the state of the generative audio system at random intervals falling in the range of 10 - 40 seconds, a range chosen to approximately correspond with timings obtained from the adaptive method (ranging from 5 - 53 seconds when measured).[6] The second method changed parameters periodically, with a change occurring every 20 seconds, again chosen to fall within the range produced by the adaptive method. These comparisons were performed using the output from a single channel of the generative audio system. Unlike the results in Section 8.5.1, the parameters of the networks used for this comparison were allowed to vary over time. Ten 60-second long excerpts were produced using each timing mechanism, and all excerpts were subjected to the same analysis.

First, the unconditional complexity $C(x)$ was measured from the excerpts using the IR obtained from the AO algorithm. Table 8.4 contains the analysis using composite frame

[6]These timings are entirely dependent on the material being analyzed, and can vary widely with different input.

Table 8.4: Complexity results for frame size of 8192 samples (approx 186 ms.) and manual threshold $\theta = 0.2$. $C(x)$ and IR measured in bits. IR files are those with adaptive parameter switching times according to IR calculation, Rand are those with random parameter switching times, and Per are those with periodic parameter switching. Averages and standard deviations shown for Rand do not include Rand6 and Rand8.

File	$C(x)$	IR	File	$C(x)$	IR	File	$C(x)$	IR
IR1	5.807	624.359	Rand1	3.585	1953.819	Per1	4.322	1775.586
IR2	4.087	1711.905	Rand2	2.233	1187.442	Per2	3	915.801
IR3	6.285	897.058	Rand3	4.7	2109.934	Per3	2.585	1181.659
IR4	5.044	1984.54	Rand4	5.459	1797.302	Per4	4.322	1130.935
IR5	6.524	2221.12	Rand5	3.807	1158.316	Per5	3.807	1447.329
IR6	6.6	2519.806	Rand6	1	0	Per6	3.7	1265.207
IR7	5.426	2409.216	Rand7	3.7	1036.1	Per7	5.087	2079.605
IR8	6.066	2396.459	Rand8	1	0	Per8	6.022	2140.55
IR9	6.066	1966.365	Rand9	2.585	468.351	Per9	4.322	2163.642
IR10	6.087	1928.264	Rand10	3.459	586.363	Per10	4.954	2153.052
AVG	5.799	1865.909	**AVG**	3.827	1287.302	**AVG**	4.212	1625.337
STDEV	0.765	637.905	**STDEV**	0.881	580.052	**STDEV**	1.01	490.888

sizes of 8192 samples, or approximately 186 ms at a sample rate of 44100. The AO distance threshold $\theta = 0.2$ was set manually, to ensure that each sound-file was analyzed with the same level of quantization. The feature used was again MFCCs, chosen to focus the analysis on the spectral envelope and resulting timbre of the signal. The adaptive, IR-based method produced output with the highest average complexity, at 5.799 bits, while the periodic and random methods are second and third with 4.212 and 3.827 respectively.[7]

It is also useful to consider the total IR as a measure of aesthetic balance. Recalling Algorithm 1, we see that the two required complexities are calculated at time i as follows:

$$C_0(i) = Log_2(\text{number of new encoding events up to } i) \tag{8.4}$$

$$C_1(i) = \frac{Log_2(\text{number of all code-words up to } i)}{\text{length L of a block to which state } i \text{ belongs}} \tag{8.5}$$

Since the IR at time i is calculated by $C_0(i) - C_1(i)$, in order for a signal to have a high total

[7]The analyses of Rand6 and Rand8 are likely flawed, as they found no distinction between signal frames. When these values are removed, the average $C(x)$ for the random method becomes 3.827. This does not change the ordering of the methods in terms of complexity.

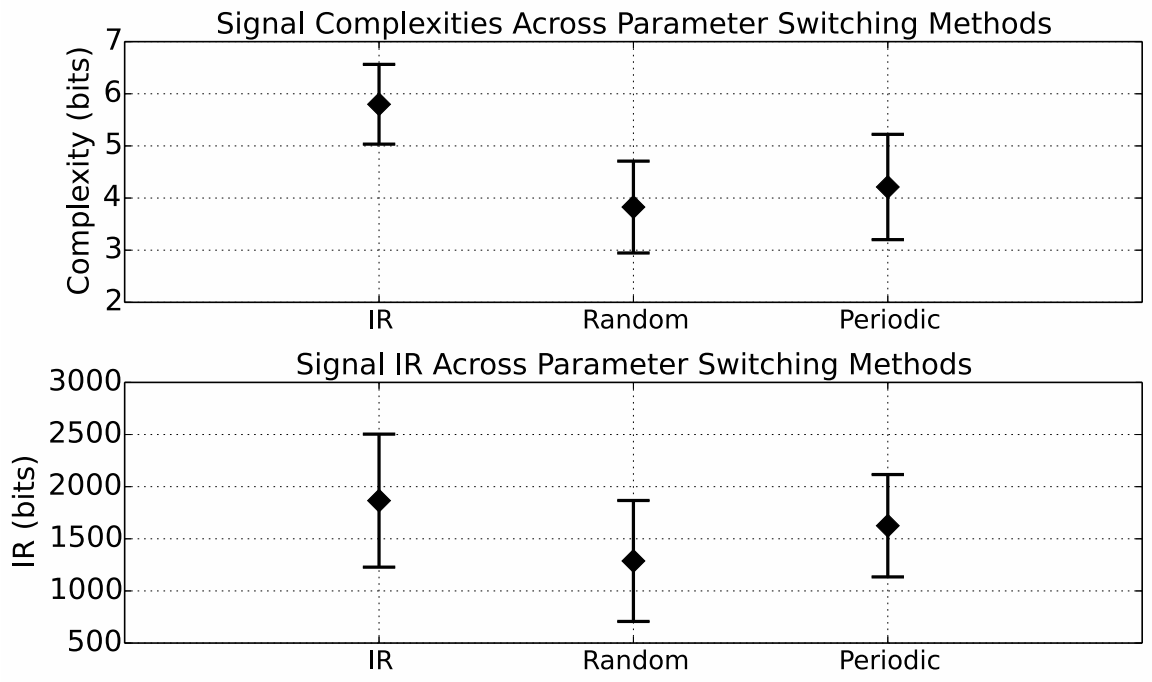

Figure 8.8: A plot of the data shown in table 8.4. Top plot shows signal complexity averages and standard deviations for three parameter switching methods. Bottom plot shows signal IR averages and standard deviations.

IR, C_1 should be small relative to C_0 much of the time. This can happen in two ways:

- Due to C_0 being large, as a result of many unique signal frames being encoded.
- Due to C_1 being small, resulting from either a low number of encoding events or a high amount of repetition (corresponding to large block length L).

For a particular point in the IR function to have a high value, the portion of the signal leading up to that point must have contained many novel signal frames - resulting in high C_0 - and the current moment must be a repetition of a significant portion of the previous signal - resulting in low C_1. Therefore, a signal with high total IR will have both a large number of unique frames and a large number of long repeated patterns - representing the good balance of order and complexity necessary from the perspective of computational aesthetics. Considering the total IR data from Table 8.4 and Figure 8.8, we see that the adaptive timer again produces the best results. The average total IR is highest, at 1865.909

bits, while the next closest value (1625.337 bits) is produced by the periodic method. The random method produces the lowest total IR, with 1287.302 bits.

8.5.4 Kolmogorov Complexity Analysis of the Adaptive Timer

The three timing methods can also be studied in terms of the Kolmogorov complexity K and resulting order M_K of their output. These results are presented in Table 8.5. The adaptive timer produces a slightly more complex output, with order of $M_K = 0.400$, than the random and periodic timers, which have order $M_K = 0.496$ and $M_K = 0.507$ respectively. The musical excerpts all fall between the maximally complex noise signal and the orderly sinusoid and chirp signals, the orders M_K of which are presented in the table. The use of Kolmogorov complexity to estimate the order of these signals confirms the results described above using the complexity $C(x)$ obtained from the AO IR analysis. The results in the table are summarized in Figure 8.9. The figure also includes the order M_K obtained from the noise, sinusoid, chirp, and silence test signals, demonstrating the relationship of the generative audio system output to the minimally and maximally complex silence and noise signals. All three musical examples fall in the middle of this range, with the adaptive method almost directly centered between noise and silence.

These comparisons were performed to quantify the effectiveness of the adaptive method in balancing order and complexity by limiting excessive repetition. Although both the random and periodic methods produced a variety of materials - and sometimes quite interesting musical output - neither made use of any information about their behavior. Textures and materials were often sustained long after their novelty and interest had worn off, producing awkward and inconsistent pacing. On the other hand, the adaptive method was able to measure when a particular type of material had been sustained long enough to lose its interest and become boring, and change the system state as a way to produce new material.

Table 8.5: Results of computing Kolmogorov complexity K and order M_K for excerpts using three different parameter switching methods.

Sound-file	Input Size (b)	K (b)	Ratio	H_s	M_K
Noise	5292044	5306728	1.003	0.999	-0.004
Chirp	5292044	1585448	0.3	0.950	0.685
Sine	5292044	1695062	0.32	0.986	0.675
Silence	5292044	813805	0.154	0.609	0.748
IR1	5291948	3185168	0.602	0.905	0.335
IR2	5291948	3065857	0.579	0.882	0.343
IR3	5291948	3305459	0.625	0.936	0.333
IR4	5291948	2340320	0.442	0.808	0.453
IR5	5291948	2725154	0.515	0.850	0.394
IR6	5291948	2406725	0.455	0.841	0.459
IR7	5291948	2720092	0.514	0.897	0.427
IR8	5291948	2929195	0.554	0.952	0.419
IR9	5291948	2476396	0.468	0.829	0.435
IR10	5291948	2819908	0.533	0.897	0.406
AVG	N/A	N/A	0.529	0.880	0.400
STDEV	N/A	N/A	0.062	0.047	0.048
Rand1	5291948	1862520	0.352	0.782	0.550
Rand2	5291948	2241522	0.424	0.758	0.441
Rand3	5291948	2150544	0.406	0.924	0.560
Rand4	5291948	2600336	0.491	0.893	0.450
Rand5	5291948	1651848	0.312	0.732	0.574
Rand6	5291948	2081129	0.393	0.844	0.534
Rand7	5291948	2341865	0.443	0.812	0.455
Rand8	5291948	2548187	0.482	0.921	0.477
Rand9	5291948	2133653	0.403	0.892	0.548
Rand10	5291948	3124111	0.59	0.941	0.372
AVG	N/A	N/A	0.430	0.849	0.496
STDEV	N/A	N/A	0.078	0.075	0.066
Per1	5291948	2540452	0.48	0.911	0.472
Per2	5291948	1453177	0.464	0.836	0.671
Per3	5291948	2565938	0.485	0.920	0.473
Per4	5291948	2429163	0.459	0.906	0.493
Per5	5291948	2571816	0.486	0.906	0.463
Per6	5291948	1911595	0.361	0.686	0.473
Per7	5291948	2146030	0.406	0.835	0.514
Per8	5291948	2427124	0.459	0.885	0.482
Per9	5291948	2163453	0.409	0.817	0.500
Per10	5291948	1954289	0.369	0.776	0.524
AVG	N/A	N/A	0.438	0.848	0.507
STDEV	N/A	N/A	0.048	0.075	0.061

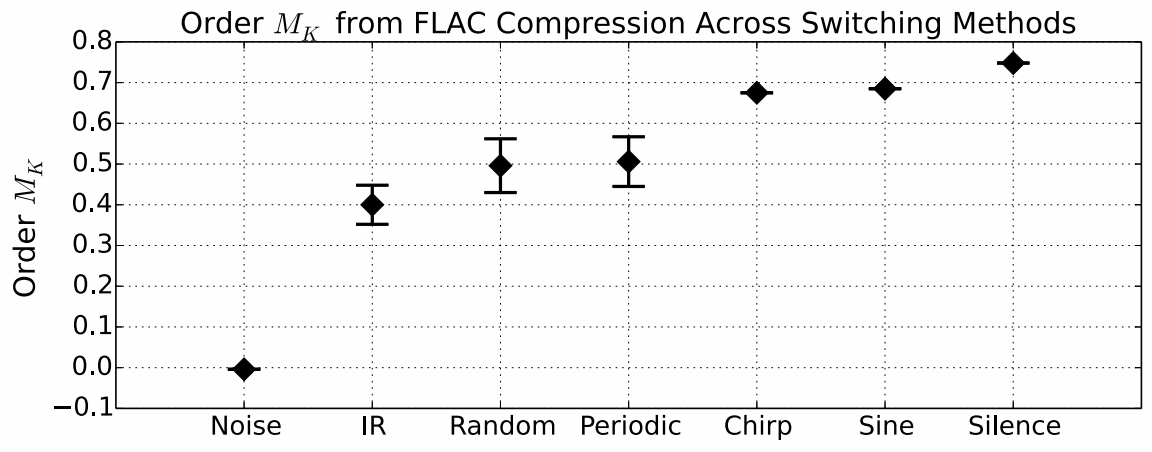

Figure 8.9: A plot of the data shown in table 8.5. The plot shows signal order M_K averages and standard deviations for three parameter switching methods. Also shows order for noise, sinusoid, and chirp test sound-files.

8.5.5 Summary and Discussion

These results demonstrate the usefulness of the generative audio system in producing complex musical audio output and the effectiveness of the adaptive timing mechanism in maintaining a good balance between order and complexity.

Two analysis techniques were used to measure the order and complexity of the various signals. The first analysis technique built an AO structure from each audio signal. After this structure was created, it was compressed and the unconditional complexity $C(x)$ - corresponding to the number of unique signal frames used in compression - was obtained. The complexity was measured from a variety of signals, including silence, noise, sinusoids, generative audio system output, and music. It was shown that the generative audio system output and the musical excerpt were the most complex, although this is partially due to the averaging imparted onto the noise signal by the feature extraction algorithms used.[8] The silence, sinusoid, noise, and chirp were found to be less complex than the musical signal, with silence being the least complex. This held true for both manual and automatic AO threshold detection. These results show that the all-pass network used here is a useful generative audio

[8] In reality, and as confirmed by later analysis using the Kolmogorov complexity, the noise signal is actually the most complex.

system. The system is capable of producing output with a level of complexity close to that of real-world music.

When using the AO algorithm to evaluate the usefulness of the adaptive timing method, the average unconditional complexity $C(x)$ was measured across three groups of 10 excerpts - one group for each timing method (adaptive, random, or periodic). It was found that the complexity of the signals created using the adaptive method was higher than that of either the random or periodic method. In the case of these excerpts, which exhibit emergent formal characteristics and feature multiple types of potentially repeated musical material, it was useful to compare the total IR calculated across each excerpt. A higher total IR results from an excerpt with a good balance between novelty and repetition. It was found that the signals made using the adaptive timing method had a higher total IR than those made using the random and periodic methods, and therefore had a better balance between order and complexity.

The second analysis technique used an estimate of Kolmogorov complexity in order to characterize the amount of order in an audio signal. The Kolmogorov complexity, defined as the length of the shortest program which can generate a string, can be estimated using a compression algorithm. In this case, it was estimated using the FLAC audio compression algorithm, a loss-less compression technique designed specifically for audio. From this, the amount of order M_K in a signal can be characterized. This method was first used to measure the amount of order in the set of test signals. It was shown that silence and noise signals have maximal and minimal amounts of order, respectively. The sinusoid and chirp signals also had high amounts of order. The musical excerpt and synthesis audio were in between these two bounds of complexity, demonstrating a balance between order and complexity.

The Kolmogorov complexity was also used to measure the average amount of order in musical excerpts created using the three timing techniques studied. Again, it was found that the musical excerpts fell in between the complexity of noise and simplicity of silence.

The adaptive timer produced slightly more complex material than the random and periodic methods; resulting from the adaptive method detecting material that has been allowed to repeat for too long, and changing its own parameters to inject novelty into the audio stream.

While it is possible that either reducing the lower bounds on the random timing method or shortening the duration of the periodic method could increase the complexity of the signals they generate, the advantage of the IR-based method is its adaptiveness. Rather than simply maximizing the amount of variety in the signals it generates - which could conceivably be obtained by simply changing parameter sets at a rapid pace - this method seeks to minimize the duration over which materials are allowed to repeat past a specific, perceptually-informed threshold. If a particular set of parameters produces an interesting, varied, and evolving sound, then this method will allow it to be heard in a prolonged fashion, whereas a random or periodic timer might switch too soon. Conversely, if a set of parameters produces dull, unchanging material, the adaptive system will detect this and adjust, while the random or periodic method may allow it to continue unchecked.

8.6 Conclusion

The *AAS-4* system described in this chapter provides a creative application of concepts from computational aesthetics to the construction of a generative audio system. The system uses a series of all-pass filters arranged in a complex, unitary feedback network. The filters are made time-varying, by allowing their output signals to become modulation signals for use elsewhere in the feedback network. As shown, this system is capable of producing audio with a complexity level similar to other generative audio systems.

In order to produce a more compelling musical experience, the parameters controlling the modulation signals can be changed over time. Music information rate can be used to measure the amount of order in the systems output signal. An adaptive timing method can then be developed, where the system makes changes only after it has determined that the

signal has been redundant for a significant amount of time. It was shown that the adaptive switching method produced greater complexity than switching at either random or periodic intervals. This approach is supported by the theories on “mere exposure” and boredom discussed in Chapter 3.

Chapter 9

Conclusion

The research in this dissertation centered around the idea of the *generative audio system*, a term introduced in Chapter 2 to describe a subcategory of generative music systems which function primarily at the signal level. Generative audio systems are capable of producing surprising and idiosyncratic musical output, and have been used (in some form or another) by many composers and musicians throughout the history of electronic music. This conclusion will recapitulate the research described earlier, and provide a few areas where it might be further developed.

9.1 Recapitulation

The first part of the dissertation introduced important concepts in the fields of generative music and computational aesthetics. Chapter 2 introduced the generative audio system and provided a variety of historical and contemporary examples. Many of these examples used feedback as a method of producing generative material. The chapter described some of the important attributes of feedback systems: non-linearity, iteration, coupling, self-organization, and complexity [21]. These attributes have been attractive to many composers, and form an important framework for the discussion and evaluation of feedback systems.

The field of cybernetics provides another important distinction between feedback systems: polarity. Systems can use either positive or negative feedback, and the character of their output depends on the form chosen. Positive feedback is particularly interesting in generative systems, as it can lead to unstable systems that jump from one state to another.

The examples of previous generative audio systems rely on a performer or composer to make determinations about the output of the system, either in performance or as compositional work. The systems are not capable of making their own formal structures. A truly autonomous system needs to be able to evaluate its own output to determine how the shape of that output should change over time, therefore a method of computational aesthetic evaluation is needed. Chapter 3 introduced the field of computational aesthetics. Many theories about computational aesthetics involve the balance between order and complexity in a stimulus. The ideal aesthetic work is not too complex, and also not too simple. Researchers have attempted to produce methods of quantifying the complexity in stimuli. The discussion focused on an idea from the field of Music Information Dynamics, called Music Information Rate (IR). IR gives a measure of the balance between order and complexity in an audio signal, and was chosen for its ease of use and efficiency. Chapter 3 also described research on the effect of repetition on subjective affect and boredom. In order to develop an understanding of how the balance of order and complexity might shape a listener's reaction to a piece of music as it unfolds, ideas from perceptual research were introduced. In particular, the "mere exposure" effect describes how subject affect changes as repeated exposures to a stimulus are experienced. As the number of exposures increases, the subject's affective response to the stimulus will become more positive - familiarity leads to "liking." However, research also shows that after a certain number of repetitions, affect will begin to become more negative again. It is theorized that this is a result of the influence of boredom on affect.

The next part of the dissertation described a group of experimental systems that built

on the ideas laid out in the previous chapters. These creative applications were presented in parallel with related research contributions in the areas of sound synthesis, generative music, and computational aesthetics.

The first such system, *Feld*, was described in Chapter 4. The chapter began by describing some early experiments that used digital or elecro-acoustic audio feedback as a means of producing interesting, generative behavior. These systems, called the *Gates* series, were conceived of as software instruments. As such, they were not fully autonomous; instead, they relied on control from a human performer to give their output a musical shape. The *Feld* system was an attempt to rectify this situation, by developing a fully autonomous generative system. The system obtained sonic complexity through interactions between many smaller components - including custom hardware devices described in Appendix B - and attempted to balance order and complexity through a simple self-analysis process. *Feld* can function fully autonomously, only needing a performer to trigger the beginning and ending of a performance. The *Feld* system is capable of generating interesting musical gestures and textures with a large amount of variety. Using the simple measure of computational aesthetics, *Feld* attempts to balance order and complexity from section to section. Unfortunately, *Feld* perhaps falls short of producing sophisticated formal structures. Additionally, the design of *Feld* employed a variety of compositional algorithms to produce gestural material. Although these algorithms generate their output in real-time, they can still be thought of as the result of a composer exerting influence on the final output. This decoupling of sonic generation and formal generation is antithetical to the concept of a generative audio system.

Chapter 5 described the next system, called *PyOracle*. *PyOracle* was the first real-time system to make use of the Audio Oracle (AO) algorithm to enable machine improvisation and computational analysis of musical audio signals. IR was reviewed in this chapter, and its relevance in computational aesthetics was explored. The *PyOracle* software

provides some facilities for composers to specify formal structures for performances. The most notable example of this is the script method, where a composer can specify a set of time-varying constraints on the generative process. This leads to the ability to pre-compose a formal structure, and have it filled in with improvised material during performance. However, *PyOracle* makes no attempt at producing content or form generatively. Rather than producing new materials, *PyOracle* instead recycles previously recorded materials into a new form. The system is therefore more useful as a model of how music information dynamics might be used in a true generative audio system. In particular, the AO algorithm was extracted from the *PyOracle* system and used as a standalone PD or Max/MSP object.

The final section describes the system that is most clearly a generative audio system, the *AAS-4* system. The system functions as a synthesis and creative application of ideas explored in the previous chapters. The *AAS-4* system makes use of time-varying all-pass filters in feedback networks as its sound-generating mechanism. A novel sound-processing technique using these filters was described in Chapter 6. By making a second-order all-pass filter time-varying, and carefully choosing the modulation parameters, it is possible to apply a frequency-dependent vibrato or frequency modulation effect. This effect was explored and the parameters were studied.

Unfortunately, making the parameters of the second-order all-pass filter time-varying can lead to instability in some implementations. Chapter 7 provided a new implementation that avoids these instabilities. After describing this new filter, the chapter described its use in feedback systems. The effects of the filter parameters on the spectra of these feedback systems were studied. Finally, a self-modulating all-pass network was described. In such a network, the parameters of an all-pass filter (placed in a feedback loop) depend on the previous output of the filter. These networks are capable of complex generative behavior, and exhibit the characteristics of feedback systems (iteration, self-organization, etc.) mentioned in Chapter 2.

The *AAS-4* system makes use of a set of complicated self-modulating all-pass networks. This generative audio system forms the core of its audio synthesis capabilities. The synthesis network produces a wide variety of textures, timbres, and gestures, and has no compositional logic built-in. The higher-level characteristics of the output are entirely a function of sample-level calculations. The *AAS-4* system also makes use of the AO algorithm to enable itself to perform real-time computational aesthetic evaluation on its output. This allows the system to determine when its output has become too repetitive, and to adjust the parameters of the generative audio system to produce novel output. It was shown that this system is capable of producing musical output with an objective complexity level similar to that of other musical excerpts.

9.2 Future Work

Both the use of time-varying all-pass filters for synthesis and the IR-based computational aesthetic evaluation technique merit further exploration. The feedback network used in *AAS-4* is a very specific one, and there exist many other possible topologies which may produce other types of sounds and behaviors. The self-modulating allpass network described in Chapter 7 could also be developed in isolation, with more attention paid to understanding the interaction of its parameters. Additionally, the use of the time-varying all-pass in sound effects processing was demonstrated, but further creative applications of these techniques could be developed.

The IR-based method of detecting repetition described in Chapter 8 is not confined to use with all-pass filter systems and so could be used in a variety of musical contexts. In particular, the field of machine improvisation seems to be a likely area - perhaps as an extension to the *PyOracle* system. It would be useful for such a system to be able to detect repetition in the input from a human improviser and manipulate its own output accordingly - either reinforcing or disrupting the sense of stasis. If the human performer has been

playing repetitive materials, the system could become “bored” and spur the performer on by producing an interjection. Alternatively, the system could reinforce the sense of repetition by playing similar materials. Additionally, a method of combining data extracted from multiple signal features into a single measure of repetition would be useful. Clearly, a human listener tracks multiple features simultaneously, and it would be advantageous to have a system capable of the same. This seems to be currently unsolved. The system could either track multiple features independently, and allow the AO and IR models to interact, or some sort of “meta-feature” could be designed - either manually, or through machine learning.

This dissertation described a generative audio system which has a level of aesthetic awareness, and is therefore capable of producing meaningful output without the use of arbitrary timing mechanisms. The research described within combined DSP and music information retrieval techniques into a creative application. A system like *AAS-4* is of relevance for composers, sound designers, and researchers with a need for an autonomous sound generation system.

Appendix A

Report on the David Tudor Archives at the Getty Center

A.1 Background and Goals

David Tudor is perhaps best known as a performer of the experimental or avant-garde music of composers like John Cage, Karlheinz Stockhausen, Morton Feldman, and Christian Wolff. After an extremely successful career as a pianist, in which he worked as interpreter of others music, Tudor became increasingly focused on his own compositions of electronic music. Tudor essentially gave up his piano performance career in favor of a style of live electronic music which relied heavily on his own home-made tools. Unfortunately, after Tudor's death, this music has not been regularly performed. I assumed that this was largely due to the fact that traditional scores for the compositions do not exist. However, the Getty holds Tudor's archives, including many of his hand-drawn schematics and performance documentation. Instead of a traditionally notated musical score, Tudor would design a particular configuration of equipment for a given piece, and then create a "block diagram" representing the physical orientation and interconnections between sound-generating or transforming devices. My hope was that I could use these block diagrams, along with other

information about specific devices (schematics, written descriptions, or photographs) to recreate a few significant works as a performance using software rather than hardware. The pieces I chose to focus on were *Pulsers*, *Toneburst*, and *Untitled*.

A.2 Findings

On my first visit to the Special Collections area of the Getty Research Library, my main aim was to gain a broad overview of the types of material held there, and to understand how they might relate to the specific compositions I was interested in. In particular, I focused on materials in `Series 1.A. Tudor, early, 1940s-1994`. I requested box 3 and box 4, containing materials related to specific pieces and unidentified diagrams, worksheets, and notes. Box 3 contains folders related to both *Pulsers*, *Toneburst*, and *Untitled*, in addition to a large number of Tudor's other electronic compositions.

My initial findings were promising, as the files contained a variety of schematic drawings showing the development of the compositions. In addition, there were various drafts of performance and liner notes, in which Tudor described the functioning of his systems (in very non-technical terms). The system used in *Untitled* was divided into two drawings - one indicating the generative portion of the system, and the other the performance processing portion. I photographed many of the relevant materials, in the hope that further study would allow me to recreate the block diagram scores as software. I also studied some of the unidentified materials in box 4, hoping that they might fill in any gaps that I might find in the previous materials.

After my visit, I spent time trying to decipher Tudor's drawings. The block diagrams consist of various icons, representing specific pieces of hardware musical equipment (in many cases custom-made), interconnected by lines indicating cables. In no case does Tudor provide a "legend" or "key" to help decipher these notations. Many of the icons were familiar, using customary iconography for amplifiers (a triangle) or ring-modulators (an

"X" - indicating multiplication). Unfortunately, some of the most important icons were non-standard. For example, the diagram for *Pulsers* contains a variety of amplifiers, filters (lowpass, bandpass, and highpass), and mixers. Unfortunately, none of these components generates sound on its own, and the component which seems to be the key sound-generator (represented by a pair of boxes containing EEG-like pulses) is unknown. There is a professionaly designed equipment layout for *Toneburst* - created in 1994, as Tudor's health began to decline - which indicates the use of specific "off-the-shelf" electronic components. Guitar effects pedals, cassette tape players, and mixers are specified in detail, but again other components are left unspecified. Tudor specifies two equalizers, but provides no information about the type, brand, or settings for the equalizers. He also indicates a "spectrum transfer" box, which is not described elsewhere.

I made note of these missing pieces of information and planned a second visit. This time, I studied the materials in boxes from `Series III. Electronics, 1950s - 1990s.` These boxes contain various electronic documentation, and I hoped that I might find descriptions or other information about the missing parts of his compositions. I also examined boxes containing photographs of Tudor's "table top setups" - his typical method of performance.

Unfortunately, I did not find what I had hoped. The electronic documentation consisted mainly of Tudor's collection of manuals, ads, and receipts pertaining to some of his equipment. There was also a selection of magazine articles related to electronics, audio, and other more esoteric topics (brainwave modification through sound, for example). Tudor had not sorted these materials in any way, and it was impossible to assign papers to specific compositions or times. Though the photograph collection does contain some interesting and potentially useful images of hardware configurations, no photos corresponded to the pieces I was interested in, instead focusing on later pieces like *Laser Focus*.

A.3 Conclusion

My research at the Getty has opened up large questions, both pertaining to Tudor's music, and the preservation of electronic music in general. Tudor must have known that his documentation was vague and incomplete. Did he intend that his works would pass with him, or was he perhaps interested in a kind of performance indeterminacy like that in the music of John Cage? It is difficult to know, as any change in the equipment used in these pieces would create a potentially enormous change in the sound and character of the musical output. If Tudor did intend that his works might carry on, composers could stand to consider ways in which they could ensure the preservation of their works. Particularly in such technology-mediated composition, it can be difficult or impossible to recreate certain compositions. As older technologies age and are replaced with new ones, music which relies on those technologies can also become obsolete. Composers who work with hardware, as Tudor did, should document their work in a way that it can be understood and reproduced in the future.

Appendix B

The Hardware Used in *Feld*

B.1 Overview

Some of the earliest experiments in real-time computer music involved interfacing a computer with separate sound-generating hardware [108]. At the time (ca. 1970), computer hardware was prohibitively expensive and incapable of real-time performance, while analog synthesis hardware was responsive and comparatively affordable. In more recent years, this situation has changed radically. Laptop computers capable of complex real-time sound processing can be had for a few hundred dollars, while modular analog synthesizers are generally the domain of boutique small-run manufacturers [109]. Perhaps in response to this, a do-it-yourself (DIY) movement has emerged, centered around internet forums and mailing-lists [110]. The DIYers, some borrowing from the hardware-hacking tradition of David Tudor, Nicholas Collins, and others, tend to embrace experimentation without much concern for the commercial viability of ideas. Many musicians working in this area are also fluent in one or more computer music programming languages, many of which are freely available and/or open-source. It is common practice for electronic musicians to perform using only a laptop running custom performance software. The basic motivation behind the projects described below was to integrate these two areas of electroacoustic music-making,

and to stimulate further exploration in this direction.

The USB-Octomod , tabulaRasa, and pucktronix.snake.corral were developed concurrently with the authors efforts to establish an affordable DIY hardware performance setup. Budget constraints mandated that this was a slow-moving effort, with periods of musical experimentation punctuated by soldering sessions. The three devices discussed here were all designed to meet a particular musical need felt by the author, and reflect an attempt to provide maximum flexibility with a minimum number of components. The devices each take a different approach to combining computer music software with DIY hardware. The USB-Octomod is an 8-channel control-voltage interface that allows a computer to be interfaced with a modular synthesis system. The tabulaRasa is a table-lookup oscillator that allows the user to design and edit custom waveforms using a PC software application. Finally, the pucktronix.snake.corral is a dual 8 x 8 matrix routing device for analog signals, with a computer control interface allowing for arbitrarily complex and rapid switching and automation.

B.2 Related Works

Other control-voltage/computer interfaces exist, including the GROOVE system and several MIDI-CV systems, both DIY and commercially manufactured [111]. Mark of the Unicorn's VOLTA and Expert Sleepers Silent Way represent another approach, each using a DC-coupled audio interface to directly output voltages [112, 113]. Potential downsides of these approaches include the low resolution of the MIDI protocol, and the requirement to dedicate audio output channels to control-voltage generation (assuming one already has access to a DC-coupled audio interface).

Several hardware table-lookup oscillators also exist in modular format, with the Synthesis Technology E350 Morphing Terrarium being a notable example. The E350 is commercially available as a pre-made module, and has the ability to morph between several

waveforms using a proprietary algorithm [114]. The module comes with a built-in set of 192 non-modifiable waveforms, stored permanently in memory. Though the user has an amount of control over the blending of the waveforms, the waveforms themselves remain fixed and designed acccording to the desires of the manufacturers.

Finally, there is at least one other computer-controlled routing matrix, the 4ms Pedals Bend Matrix [115]. This device uses similar hardware components to the pucktronix.snake.corral but relies on MIDI and physical pushbuttons for control over matrix connection points. 4ms has undertaken great efforts toward making the Bend Matrix a playable, musical instrument, through the use of firmware supporting presets, automation routines, and multiple I/O configurations. Unfortunately, if the user desires a more customized configuration than provided, he or she has to use MIDI to program the device. In addition, the additional hardware required for pushbutton control and preset storage adds to the overall cost of the device.

B.3 The Devices

B.3.1 Common Features

The devices discussed below all use a simple microcontroller as their primary electronic component. The USB-Octomod and pucktronix.snake.corral use a Teensy2.0, a breadboard-compatible microcontroller, programmed with the Arduino environment running on OS X, Windows, and Linux [116]. The Teensy 2.0 uses an Atmel ATmega32U4 processor, with 32k of flash memory, 25 digital I/O pins, and 12 analog input pins. The Teensy also contains an integrated USB port for both firmware uploading and serial communication with a PC. The Teensy 2.0 was chosen for its small footprint, built-in USB hardware, and low price relative to Arduino-branded options. The tabulaRasa uses a simplified Arduino configuration. Using an ATmega328, along with a crystal and a few other passive

components, it is possible to create a breadboarduino - a device capable of running Arduino code that can be incorporated into another circuit. This method was used in the tabulaRasa because no serial communication with the PC was needed after programming, and because of the low hardware cost associated with such a minimal circuit.

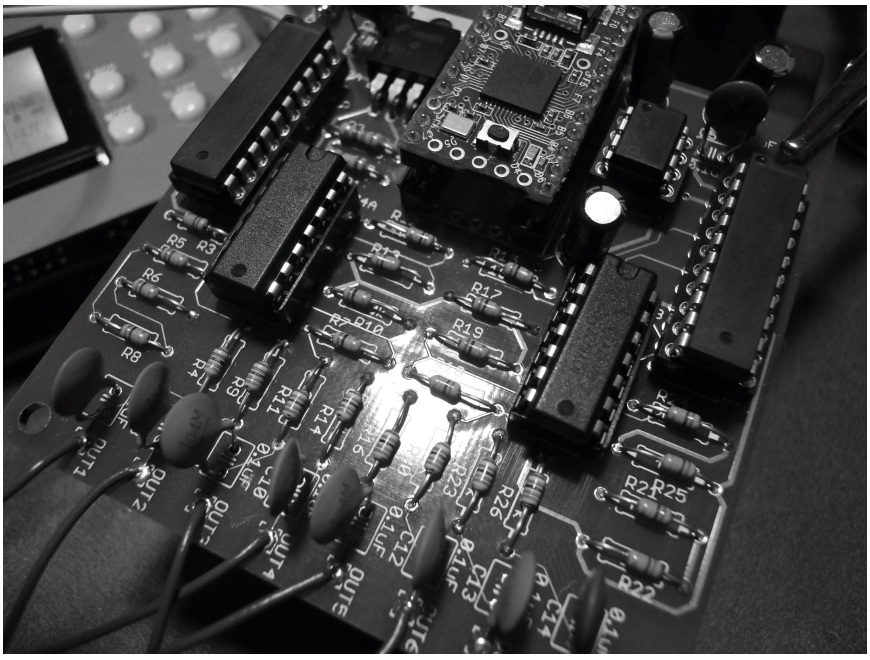

Figure B.1: USB-Octomod ciruit board assembly showing Teensy 2.0

The free, open-source Arduino programming language was used to develop the microcontroller code for all three projects [117]. Arduino provides a library of functions which enable simplified access to the ATmega328s I/O pins and other functionality, while allowing more experienced users to use the C programming language for lower-level hardware control. The popularity of the Arduino among hobbyists and experimenters provides the benefit of a large body of freely available example code, forum discussions, and tutorials. Both the USB-Octomod and tabulaRasa use the Processing language for their PC-side interfaces [118]. Processing is primarily intended for programming visual and interactive art, but provides libraries for GUI design, serial communication, and sound generation. Finally, the pucktronix.snake.corral uses a script written in the Python programming language to receive

data and coordinate with the snake.corral hardware via the PC serial port. Efforts are made to provide the software as both compiled binaries and source code, and modifications or additions are encouraged. The projects are all hosted at https://bitbucket.org/pucktronix/

B.3.2 USB-Octomod

The USB-Octomod is an 8-channel control-voltage interface employing a minimal hardware part count. The device uses a Teensy 2.0 and a pair of 4-channel 10-bit DAC ICs as the backbone of the voltage-generation circuitry. Although higher resolution DACs are available (at higher cost), experimentation showed that 10 bits was enough, when combined with an RC low-pass filter, to produce a wide range of discrete output voltages and to allow for smooth sweeps from one value to another. The device draws power directly from the USB bus, and is designed to provide output voltages in the range of +/- 5 volts corresponding to a 10-octave range in a typical Volt/octave synthesis system. Since the USB-Octomod uses external DACs, the user is not required to use a DC-coupled audio interface, or dedicate output channels to control-voltages. The USB-Octomod uses the OpenSoundControl protocol (OSC) to enable a modular software design [91]. A lightweight Processing GUI application (the host) intercepts OSC messages, buffers and converts them, and finally sends them to the USB serial port. The host also provides the option to select a custom OSC port, and allows the user to direct data to a specific serial device. Received OSC data is visualized with a set of sliders. In order to send data to the host application, the user assembles an OSC message formatted `/dac ch1 ch2 ch3 ch4 ch5 ch6 ch7 ch8` - replacing the `ch1-8` placeholders with an integer value 0 - 1023.

The separation of data generation from transmission is an important feature of the device. The communication structure is shown in Figure B.2. By decoupling the host software, which handles the technicalities of serial communication, from the OSC generation algorithm, the device allows a musician to use his/her preferred software environment for

the musically relevant OSC generation tasks. Users have developed Max and PD patches, templates for the iPad touchOSC application, and ChucK programs all designed to generate data and send it to the host application. The USB-Octomod can be thought of as 8 extremely versatile modulation sources. Each output can be driven by any number of unique processes, generated in real-time or pre-composed.

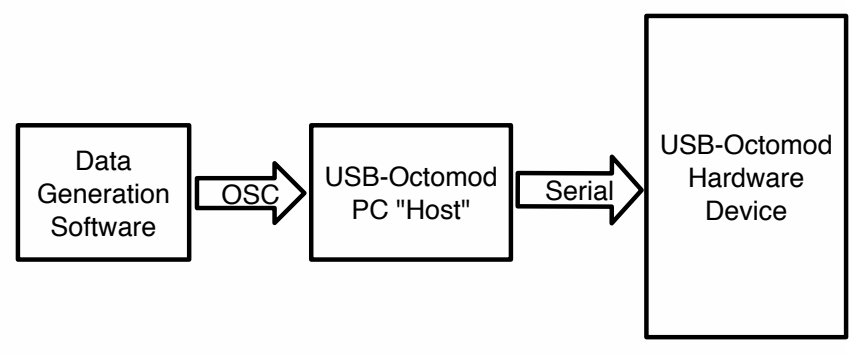

Figure B.2: Illustration of USB-Octomod Control Flow

B.3.3 tabulaRasa

The tabulaRasa generates audio signals using a table-lookup oscillator algorithm, and is designed to integrate directly into a modular synthesis system without necessitating that a PC be present during performance. The device also requires either a +/- 12V or +/- 15V power source (modular synthesis standards), as there is no USB connection present. An ATmega328 reads waveform data from an SD memory card held in a socket mounted to the circuit board, and uses a modified version of Adrian Freeds table-lookup oscillator code to generate a PWM signal at one of the digital output pins [119]. This signal is converted to a continuous waveform by an external RC lowpass filter. Due to the limited memory on the ATmega328, waveforms are stored as 256 byte arrays. The SD card is used as storage, and two waveforms are read into RAM at a time. Modifications to the basic table-lookup algorithm allow for interpolation (blend) between the two waveforms in RAM, allowing for continuous timbral variation. Six of the ATmega328s analog inputs are used for controlling

synthesis parameters in real-time. Each of the following parameters has both a potentiometer and control-voltage input: oscillator frequency (V/octave), amount of interpolation between waveform pairs, and selection of which waveform pair is currently stored in RAM.

Figure B.3: tabulaRasa waveform design interface

The tabulaRasa software presents a GUI (shown in Figure B.3) which allows the user to design breakpoint-based waveforms in several ways, apply various interpolations to the waveforms, and write them to an SD card in a format which can be read by the hardware. The software also provides a real-time audio preview of the waveform, and allows the created waveforms to be saved into any of 32 slots. The slots are organized into 16 pairs, which correspond to the interpolation pairs mentioned in describing the hardware. The ability to design and load arbitrary waveforms gives the device a large amount of sonic flexibility lacking in devices with a fixed bank of wavetables stored in permanent memory.

B.3.4 pucktronix.snake.corral

pucktronix.snake.corral is a computer-controlled dual 8 x 8 analog signal routing matrix. Two independent matrices are presented, each with 8 inputs and 8 outputs. Within each matrix, any input (or summed combination of inputs) can be routed to any output. The device can switch and route any type of analog signal within the range of +/- 5V. The main electronic components of the pucktronix.snake.corral are a Teensy 2.0 and a pair of Zarlink MT8816 analog switching matrix ICs. The MT8816 is a bidirectional 8 x 16 matrix with minimal signal bleed. Like the USB-Octomod, the pucktronix.snake.corral is powered from the USB bus.

The pucktronix.snake.corral decouples the control and transmission components of the software. Like the USB-Octomod, the device communicates with a PC through a light-weight software application - here, a script written in the Python programming language. Instead of demanding that the user employ a particular programming language or compositional environment, the script listens for OSC messages and converts them into serial data which is communicated to the hardware. The OSC protocol contains four pieces of data: a flag selecting which of the two MT8816 ICs is being addressed, the x-address of the switch being addressed, the y-address of the switch being addressed, and the state (open/closed) of that switch. A Max/MSP patch which allows the user to define and switch between presets and/or apply various algorithmic rhythmic effects to the switching matrices has also been developed.

Using the pucktronix.snake.corral, a modest number of synthesis modules can be used to create interesting rhythmic and timbral variety. The ability to rapidly switch or reconfigure a large number of signal connections enables a level of rhythmic complexity which is difficult to obtain through other means. Sharp cuts between disparate types of musical material are made possible, and patches can be stored and quickly recalled.

B.4 Further Work and Evaluation of Hardware

Ignoring, for a moment, the creative application of these devices, there are some possible improvements which could allow for greater flexibility. The utility of the USB-Octomod could easily be extended by the addition of analog inputs. A situation in which a computer receives and transforms voltages from an analog synthesizer in real-time, acting as an extremely flexible processing module, could be extremely musically rewarding. Further exploration of the device in combination with GUI or alternate control methods is also necessary.

The tabulaRasa could be improved by replacing the ATmega328 with a chip capable of running at a higher clock rate. The current implementation (PWM running at 16Mhz) is quite susceptible to aliasing. A chip with a larger amount of RAM would also enable a 2D crossfade between 4 waveforms - something initially planned for the tabulaRasa but discarded due to memory constraints.

The pucktronix.snake.corral would benefit from a more flexible range of input voltages. Currently, voltages greater than +/- 5V are clipped. This protects the internal circuitry of the MT8816 chips, but could be changed. From a musical standpoint, having a variable gain at each switch-point would allow for more variety and control over routing configurations - especially useful in a situation where feedback paths are allowed.

The relative success of these projects suggests that further experimentation with affordable microcontrollers in musical applications would be fruitful. While basic chips like the ATmega328 are perhaps not powerful enough for real-time DSP, others, like the dsPIC family, seem to merit exploration in this area [120]. Additionally, the processors used here are more than powerful enough for control-based applications running at lower rates, while also providing analog input from a variety of sensor devices.

Aside from issues of technological feasability, the emergence of the Arduino and associated projects, along with the DIY and open-source software movements, create a

fertile environment for experimentation. Computer musicians, hardware hackers, and composers can design, rapidly prototype, and fabricate previously non-existent or commercially unviable hardware devices. The hardware needs of the artist are no longer subject to the whims of corporate manufacturers. Instead of bending a premade device to the specific needs of individual musical practice, we (artists and musicians) can now create exactly the device needed for a given piece, performance, or purpose.

B.5 Acknowledgements

The material in this chapter was originally published in:
Surges, G. “DIY Hybrid Analog/Digital Modular Synthesis.” *Proceedings, 12th International Conference on New Interfaces for Musical Expression.* Ann Arbor, MI. 2012. The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] B. Eno, "Generative Music - Brian Eno - In Motion Magazine," 1996. [Online]. Available: http://www.inmotionmagazine.com/eno1.html

[2] N. Collins, "The Analysis of Generative Music Programs," *Organised Sound*, vol. 13, no. 3, pp. 237 – 248, 2008.

[3] J. Foote, "Automatic Audio Segmentation Using a Measure of Audio Novelty," *2000 IEEE International Conference on Multimedia and Expo, 2000.*, pp. 452 – 455, 2000.

[4] G. Tzanetakis and P. Cook, "Musical Genre Classification of Audio Signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, 2002.

[5] D. Cope, *Virtual Music: Computer Synthesis of Musical Style*. Boston, MA: The MIT Press, 2004.

[6] B. Carey, "Designing for Cumulative Interactivity : The _ derivations System," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2012, pp. 495–498.

[7] P. Cano, M. Koppenberger, and N. Wack, "An Industrial-strength Content-based Music Recommendation System," *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[8] M. Plumbley and S. Abdallah, "Automatic Music Transcription and Audio Source Separation," *Cybernetics and Systems: An International Journal*, vol. 33, no. 6, pp. 603 – 627, 2002.

[9] G. Surges, T. Smyth, and M. Puckette, "Generative Feedback Networks Using Time-Varying Allpass Filters," in *Proceedings of the 2015 International Computer Music Conference*, 2015.

[10] R. Holopainen, "Self-organised Sound with Autonomous Instruments: Aesthetics and experiments," Doctoral Dissertation, University of Oslo, 2012.

[11] C. Ames, "Automated Composition in Retrospect: 1956-1986," *Leonardo*, vol. 20, no. 2, pp. 169 – 185, 1987.

[12] P. Griffiths, "Xenakis: Logic and Disorder," *The Musical Times*, vol. 116, no. 1586, pp. 329 – 331, 1975.

[13] C. Roads, *The Computer Music Tutorial*. Boston, MA: The MIT Press, 1996.

[14] M. Puckette, "Max at Seventeen," *Computer Music Journal*, vol. 26, no. 4, pp. 31–43, 2002.

[15] K. Essl, "Lexikon-Sonate: An Interactive Realtime Composition for Computer-Controlled Piano," *Proceedings of the 2nd Brazilian Symposium on Computer Music*, 1995.

[16] A. Eigenfeldt and P. Pasquier, "A Realtime Generative Music System using Autonomous Melody, Harmony, and Rhythm Agents," *Proceedings of the 12th Generative Art Conference GA*, 2009.

[17] O. Bown, "Experiments in Modular Design for the Creative Composition of Live Algorithms," *Computer Music Journal*, vol. 35, no. 3, pp. 73–85, 2011.

[18] J. Pritchett, *The Music of John Cage*. Cambridge, UK.: Cambridge University Press, 1996.

[19] J. Driscoll and M. Rogalsky, "David Tudor's Rainforest: An Evolving Exploration of Resonance," *Leonardo Music Journal*, vol. 14, pp. 25 – 30, 2004.

[20] R. Kuivila, "Open Sources: Words, Circuits and the Notation-realization Relation in the Music of David Tudor," *Leonardo Music Journal*, vol. 14, pp. 17 – 23, 2004.

[21] D. Sanfilippo and A. Valle, "Feedback Systems: An Analytical Framework," *Computer Music Journal*, vol. 37, no. 2, pp. 12–27, 2013.

[22] W. Ashby, *An Introduction to Cybernetics*. London: Chapman and Hall, 1956.

[23] D. Sanfilippo, "Turning Perturbation Into Emergent Sound, and Sound into Perturbation," *Interference: A Journal of Audio Culture*, no. 3, 2013. [Online]. Available: http://www.interferencejournal.com/articles/noise/turning-perturbation-into-emergent-sound

[24] M. Cherniakov, *An Introduction to Parametric Digital Filters and Oscillators*. Hoboken, NJ: Wiley Publishing, 2004.

[25] K. Karplus and A. Strong, "Digital Synthesis of Plucked-string and Drum Timbres," *Computer Music Journal*, vol. 7, no. 2, pp. 43 – 55, 1983.

[26] T. Blackwell, "Swarming and Music," in *Evolutionary Computer Music*, E. Miranda and J. Biles, Eds. London: Springer, 2007, ch. 9, pp. 194 – 217.

[27] E. Miranda, "At the Crossroads of Evolutionary Computation and Music: Self-programming Synthesizers, Swarm Orchestras and the Origins of Melody," *Evolutionary Computation*, vol. 12, no. 2, pp. 137 – 158, 2004.

[28] P. Beyls, "Musical Morphologies from Self-organizing Systems," *Journal of New Music Research*, vol. 19, no. 2-3, pp. 205–218, 1990.

[29] J. Pritchett, "David Tudor as Composer/Performer in Cage's Variations II." *Leonardo music journal*, vol. 14, pp. 11 – 16, 2004.

[30] N. Collins, "Composers Inside Electronics: Music after David Tudor," *Leonardo Music Journal*, vol. 14, pp. 1 – 3, 2004.

[31] J. Goldman, "The Buttons on Pandora's Box: David Tudor and the Bandoneon," *American Music*, vol. 30, no. 1, pp. 30–60, 2012.

[32] B. Duffie and D. Tudor, "Presenting David Tudor: A Conversation with Bruce Duffie," 1986. [Online]. Available: http://www.bruceduffie.com/tudor3.html

[33] F. Warthman, "Album Notes to David Tudor's 'Neural Synthesis'," 1996. [Online]. Available: http://www.lovely.com/albumnotes/notes1602.html

[34] B. Viola, "David Tudor: The Delicate Art of Falling," *Leonardo Music Journal*, vol. 14, pp. 49–56, 2004.

[35] T. Patteson, "The Time of Roland Kayn's Cybernetic Music," in *Sonic Acts Festival XIV: Travelling Time*, Amsterdam, 2012, pp. 46 – 57.

[36] N. Wiener, *Cybernetics, or Control and Communication in the Animal and the Machine*. Boston, MA: The MIT Press, 1965.

[37] J. Harley, "The Electroacoustic Music of Iannis Xenakis," *Computer Music Journal*, vol. 26, no. 1, pp. 33–57, 2002.

[38] S. Luque, "The Stochastic Synthesis of Iannis Xenakis," *Leonardo Music Journal*, vol. 19, pp. 77–84, 2009.

[39] C. Burns, "Emergent Behavior from Idiosyncratic Feedback Networks," *Proceedings of the 2003 International Computer Music Conference*, 2003.

[40] P. S. P. Bittencourt, "The Performance of Agostino Di Scipio's Modes of Interference n. 2: A Collaborative Balance," *Contemporary Music Review*, vol. 33, no. 1, pp. 46–58, 2014.

[41] P. S. Bittencourt, "Modes of Interference 2 (2006) - Agostino Di Scipio - YouTube," 2013. [Online]. Available: https://www.youtube.com/watch?v=fHbV9fSxYoA

[42] D. Sanfilippo, "LIES (topology/nodes) 1.0 Live at Padua's Conservatory - 9th july 2011 - 8th SMC Conference," 2011. [Online]. Available: https://soundcloud.com/dario-sanfilippo/live-at-paduas-conservatory

[43] R. Holopainen, "Sound Examples," 2014. [Online]. Available: http://ristoid.net/research/sndex.html

[44] S. Dubnov, S. McAdams, and R. Reynolds, "Structural and Affective Aspects of Music from Statistical Audio Signal Analysis," *Journal of the American Society for Information Science and Technology*, vol. 57, pp. 1526–1536, 2006.

[45] S. Dubnov, G. Assayag, and A. Cont, "Audio Oracle Analysis of Musical Information Rate," *Fifth IEEE International Conference on Semantic Computing (ICSC)*, pp. 567–571, 2011.

[46] L. Meyer, *Emotion and Meaning in Music*. Chicago, IL: University of Chicago Press, 1956.

[47] R. Bornstein, A. Kale, and K. Cornell, "Boredom as a Limiting Condition on the Mere Exposure Effect," *Journal of Personality and Social Psychology*, vol. 58, no. 5, pp. 791–800, 1990.

[48] O. E. D. Online, "Oxford English Dictionary Online," 2010. [Online]. Available: http://dictionary.oed.com

[49] J. Shelley, "The Concept of the Aesthetic," 2009. [Online]. Available: http://plato.stanford.edu/entries/aesthetic-concept/#ConAes

[50] P. Galanter, "Computational Aesthetic Evaluation: Past and Future," in *Computers and Creativity*, J. McCormack and M. D'Inverno, Eds. Berlin: Springer, 2012, pp. 255–293.

[51] R. Zajonc, "Attitudinal Effects of Mere Exposure," *Journal of Personality and Social Psychology*, vol. 9, no. 2, pp. 1–27, 1968.

[52] F. Hoenig, "Defining Computational Aesthetics," in *Proceedings of the First Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, 2005, pp. 13–18.

[53] G. D. Birkhoff, *Aesthetic Measure*. Cambridge, MA: Harvard University Press, 1933.

[54] D. Borgo, *Sync or Swarm: Improvising Music in a Complex Age*. New York: Continuum, 2005. [Online]. Available: https://scholar.google.com/scholar?q=sync+or+swarm&btnG=&hl=en&as_sdt=0%2C5#2

[55] J. Rigau, M. Feixas, and M. Sbert, "Conceptualizing Birkhoff's Aesthetic Measure using Shannon Entropy and Kolmogorov Complexity," in *Proceedings of the Third Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, 2007, pp. 105–112.

[56] F. Boselie and E. Leeuwenberg, "Birkhoff Revisited: Beauty as a Function of Effect and Means," *The American Journal of Psychology*, vol. 98, no. 1, pp. 1–39, 1985.

[57] D. Huron, "Music and Emotion - Notes on Leonard Meyer." [Online]. Available: http://www.musiccog.ohio-state.edu/Music829D/Notes/Meyer1.html

[58] J. Rigau, M. Feixas, and M. Sbert, "Informational Aesthetics Measures," *IEEE Computer Graphics and Applications*, vol. 28, no. 2, pp. 24–34, 2008.

[59] R. Krull, J. Watt, and L. Lichty, "Entropy and Structure: Two Measures of Complexity in Television Programs," *Communication Research*, vol. 4, no. 1, pp. 61 – 86, 1977.

[60] R. Potter and J. Choi, "The Effects of Auditory Structural Complexity on Attitudes, Attention, Arousal, and Memory," *Media Psychology*, vol. 8, no. 4, pp. 395 – 419, 2006.

[61] P. Rozin, A. Rozin, B. Appel, and C. Wachtel, "Documenting and Explaining the Common AAB Pattern in Music and Humor: Establishing and Breaking Expectations," *Emotion*, vol. 6, no. 3, pp. 349 – 355, 2006.

[62] D. Hargreaves, "The Effects of Repetition on Liking for Music," *Journal of Research in Music Education*, vol. 32, no. 1, pp. 35–47, 1984.

[63] R. Bornstein, "Exposure and Affect: Overview and Meta-analysis of Research, 1968 - 1987," *Psychological Bulletin*, vol. 106, no. 2, pp. 265 – 289, 1989.

[64] D. de Zilva, L. Vu, B. R. Newell, and J. Pearson, "Exposure Is Not Enough: Suppressing Stimuli from Awareness Can Abolish the Mere Exposure Effect," *PloS ONE*, vol. 8, no. 10, 2013.

[65] D. Cohen, "Palestrina Counterpoint: A Musical Expression of Unexcited Speech," *Journal of Music Theory*, vol. 15, no. 1/2, pp. 84 – 111, 1971.

[66] R. Farmer and N. Sundberg, "Boredom Proneness - The Development and Correlates of a New Scale," *Journal of Personality Assessment*, vol. 50, no. 1, pp. 4 – 17, 1986.

[67] K. Szpunar, "Liking and Memory for Musical Stimuli as a Function of Exposure," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 30, no. 2, pp. 370 – 381, 2004.

[68] R. Reber, N. Schwarz, and P. Winkielman, "Processing Fluency and Aesthetic Pleasure: Is Beauty in the Perceiver's Processing Experience?" *Personality and Social Psychology Review*, vol. 8, no. 4, pp. 364 – 382, 2004.

[69] R. Reber, P. Winkielman, and N. Schwarz, "Effects of Perceptual Fluency on Affective Judgments," *Psychological Science*, vol. 9, no. 1, pp. 45 – 48, 1998.

[70] P. Juslin and D. Västfjäll, "Emotional Responses to Music: The Need to Consider Underlying Mechanisms," *Behavioral and Brain Sciences*, vol. 31, no. 5, pp. 559 – 575, 2008.

[71] C. Shannon, "A Mathematical Theory of Communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3 – 55, 2001.

[72] K. Potter, G. Wiggins, and M. Pearce, "Towards Greater Objectivity in Music Theory: Information-dynamic Analysis of Minimalist Music," *Musicae Scientiae*, vol. 11, no. 2, pp. 295 – 324, 2007.

[73] M. T. M. Pearce and G. G. A. Wiggins, "Auditory Expectation: The Information Dynamics of Music Perception and Cognition," *Topics in Cognitive Science*, vol. 4, no. 4, pp. 625–652, 2012.

[74] S. Abdallah and M. Plumbley, "Information Dynamics: Patterns of Expectation and Surprise in the Perception of Music," *Connection Science*, vol. 21, no. 2/3, pp. 89–117, 2009.

[75] S. Dubnov, G. Assayag, and A. Cont, "Audio Oracle: A New Algorithm for Fast Learning of Audio Structures," in *Proceedings of the 2007 International Computer Music Conference*, Copenhagen, 2007.

[76] C. Allauzen, M. Crochemore, and M. Raffinot, "Factor Oracle: A New Structure for Pattern Matching," *Lecture Notes in Computer Science*, vol. 1725, pp. 1–16, 1999.

[77] G. Surges and S. Dubnov, "Feature Selection and Composition Using PyOracle," in *Proceedings, Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, Boston, MA, 2013. [Online]. Available: http://www.aaai.org/ocs/index.php/AIIDE/AIIDE13/paper/viewPDFInterstitial/7452/7685

[78] A. Lefebvre and T. Lecroq, "Compror: Compression with a Factor Oracle," in *Proceedings of the 2001 Data Compression Conference*. IEEE Computer Society, 2001, p. 502. [Online]. Available: http://dl.acm.org/citation.cfm?id=882454.875065

[79] B. Logan, "Mel Frequency Cepstral Coefficients for Music Modeling," in *Proceedings of the 2000 International Symposium on Music Information Retrieval*, vol. 28, 2000.

[80] Composers inside Electronics, "David Tudor Exhibition." [Online]. Available: http://composers-inside-electronics.net/dtudor/legacy/untitled.html

[81] G. Surges, "DIY Hybrid Analog/Digital Modular Synthesis," in *Proceedings of the 2012 Conference on New Interfaces for Musical Expression*, Ann Arbor, MI, 2012.

[82] F. Nake, "Order in Complexity," *Leonardo Electronic Almanac*, vol. 17, no. 1, Aug. 2010.

[83] K. Burns, "Atoms of EVE: A Bayesian Basis for Aesthetic Analysis of Style in Sketching," *AIE EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, vol. 20, no. 3, pp. 185–199, 2006.

[84] G. Assayag, G. Bloch, and M. Chemillier, "Omax Brothers: A Dynamic Topology of Agents for Improvization Learning," in *Proceedings of the 1st ACM Workshop on Audio and music Computing Multimedia*, 2006, pp. 125 – 132.

[85] A. Francois, I. Schankler, and E. Chew, "Mimi4x: An Interactive Audiovisual Installation for High-level Structural Improvisation," *International Journal of Arts and Technology*, vol. 6, no. 2, pp. 138–151, 2013.

[86] Python Software Foundation, "Welcome to Python.org." [Online]. Available: https://www.python.org/

[87] G. Lewis, "Too Many Notes: Computers, Complexity and Culture in Voyager," *Leonardo Music Journal*, vol. 10, pp. 33 – 39, 2000.

[88] F. Pachet, "The Continuator: Musical Interaction With Style," *Journal of New Music Research*, vol. 31, pp. 333–341, 2002.

[89] T. Gifford and A. R. Brown, "Beyond Reflexivity: Mediating Between Imitative and Intelligent Action in an Interactive Music System," in *25th BCS Conference on Human-Computer Interaction*, 2011.

[90] IRCAM, "omax:examples [OMax]." [Online]. Available: http://repmus.ircam.fr/omax/examples

[91] M. Wright, "Open Sound Control: An Enabling Technology for Musical Networking," *Organised Sound*, vol. 10, no. 3, p. 193, 2005.

[92] V. Välimäki, J. Abel, and J. Smith, "Spectral Delay Filters," *Journal of the Audio Engineering Society*, vol. 57, no. 7/8, pp. 521–531, 2009.

[93] J. Kleimola and J. Pekonen, "Sound Synthesis Using an Allpass Filter Chain with Audio-rate Coefficient Modulation," in *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, Como, Italy, 2009.

[94] J. Rauhala and V. Valimaki, "Tunable Dispersion Filter Design for Piano Synthesis," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 253–256, 2006.

[95] J. Abel and J. Smith, "Robust Design of Very High-order Allpass Dispersion Filters," in *Proceedings of the 6th International Conference on Digital Audio Effects (DAFX-06)*, 2006.

[96] V. Lazzarini and J. Timoney, "Adaptive Phase Distortion Synthesis," in *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, Como, Italy., 2009, pp. 1–8.

[97] G. Surges and T. Smyth, "Spectral Distortion Using Second-Order Allpass Filters," in *Proceedings of the 2013 Sound and Music Computing Conference*, Stockholm, Sweden, 2013.

[98] P. Kabal, "Minimum-phase & all-pass filters," Department of Electrical and Computer Engineering, McGill University., Montreal, Quebec., Tech. Rep., 2011. [Online]. Available: http://bellatrix.ece.mcgill.ca/documents/Reports/2011/KabalR2011a.pdf

[99] D. Arfib, F. Keiler, and U. Zölzer, *DAFx - Digital Audio Effects*. Chichester: J. Wiley & Sons, 2002.

[100] J. Abel, D. Berners, S. Costello, and J. S. III, "Spring Reverb Emulation Using Dispersive Allpass Filters in a Waveguide Structure," in *Audio Engineering Society Convention 121*, 2006.

[101] S. Bilbao, "Time-varying Generalizations of All-pass Filters," *IEEE Signal Processing Letters*, vol. 12, no. 5, pp. 376–379, 2005.

[102] M. Puckette, *The Theory and Technique of Electronic Music*. River Edge, NJ: World Scientific Publishing Company, 2007.

[103] ——, "Infuriating Nonlinear Reverberator," in *Proceedings of the 2011 International Computer Music Conference*, Huddersfield, UK, 2011.

[104] S. Streich, "Automatic Characterization of Music Complexity: A Multifaceted Approach," Doctoral Dissertation, Universitat Pompeu Fabra, 2006.

[105] C. Ariza, "The Interrogator as Critic : The Turing Test and the Evaluation of Generative Music Systems," *Computer Music Journal*, vol. 33, no. 2, pp. 48–70, 2009.

[106] D. Tudor, *The Art of David Tudor (1963-1992), Vol. 7*. Sound Recording. Anthology of Recorded Music, Inc., 2013.

[107] J. Coalson and Xiph.org, "FLAC - Free Lossless Audio Codec," 2014. [Online]. Available: https://xiph.org/flac/index.html

[108] M. Mathews and F. Moore, "GROOVE - A Program to Compose, Store, and Edit Functions of Time," *Communications of the ACM*, vol. 13, no. 12, pp. 715 – 721, 1970.

[109] Wessex Analogue, "Wiard." [Online]. Available: http://www.wiard.com/

[110] Muffwiggler, "MUFFWIGGLER." [Online]. Available: https://www.muffwiggler.com/

[111] PAiA Corporation USA, “PAiA - MIDI to CV Converter.” [Online]. Available: http://www.paia.com/midi2cv.asp

[112] MOTU Inc., “MOTU.com - What is Volta?” [Online]. Available: http://www.motu.com/products/software/volta

[113] Expert Sleepers Ltd., “Expert Sleepers - Silent Way.” [Online]. Available: http://www.expert-sleepers.co.uk/silentway.html

[114] Synthesis Technology, “Synthesis Technology - Eurorack E350.” [Online]. Available: http://synthtech.com/eurorack/E350/

[115] 4ms Company, “Bend Matrix.” [Online]. Available: http://www.4mspedals.com/bendmatrix.php

[116] PJRC, “Teensy USB Development Board.” [Online]. Available: https://www.pjrc.com/teensy/

[117] Arduino, “Arduino - Home.” [Online]. Available: https://www.arduino.cc/

[118] B. Fry and C. Reas, “Processing.org.” [Online]. Available: https://processing.org/

[119] A. Freed, “Arduino sketch for high frequency precision sine wave tone sound synthesis — Adrian Freed.” [Online]. Available: http://adrianfreed.com/content/arduino-sketch-high-frequency-precision-sine-wave-tone-sound-synthesis

[120] Microchip Technology, “16-bit PIC Microcontrollers - Overview — Microchip Technology Inc. — Microchip Technology Inc.” [Online]. Available: http://www.microchip.com/pagehandler/en-us/family/16bit/