HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering
Laboratory of Acoustics and Audio Signal Processing

Jussi Pekonen

Computationally Efficient Music Synthesis – Methods and Sound Design

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, June 1, 2007

Supervisor:	Professor Vesa Välimäki
Instructor:	Professor Vesa Välimäki

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Jussi Pekonen
Name of the thesis:	Computationally Efficient Music Synthesis – Methods and Sound Design
Date:	June 1, 2007 **Number of pages:** 80+xi
Department:	Electrical and Communications Engineering
Professorship:	S-89
Supervisor:	Professor Vesa Välimäki
Instructor:	Professor Vesa Välimäki

In this thesis, the design of a music synthesizer for systems suffering from limitations in computing power and memory capacity is presented. First, different possible synthesis techniques are reviewed and their applicability in computationally efficient music synthesis is discussed. In practice, the applicable techniques are limited to additive and source-filter synthesis, and, in special cases, to frequency modulation, wavetable and sampling synthesis.

Next, the design of the structures of the applicable techniques are presented in detail, and properties and design issues of these structures are discussed. A major implementation problem is raised in digital source-filter synthesis, where the use of classic waveforms, such as sawtooth wave, as the source signal is challenging due to aliasing caused by waveform discontinuities. Methods for existing bandlimited waveform synthesis are reviewed, and a new approach using polynomial bandlimited step function is presented in detail with design rules for the applicable polynomials. The approach is also tested with two different third-order polynomials. They reduce aliasing more at high frequencies, but at low frequencies their performance is worse than with the first-order polynomial. In addition, some commonly used sound effect algorithms are reviewed with respect to their applicability in computationally efficient music synthesis.

In many cases the sound synthesis system must be capable of producing music consisting of various different sounds ranging from real acoustic instruments to electronic instruments and sounds from nature. Therefore, the music synthesis system requires careful sound design. In this thesis, sound design rules for imitation of various sounds using the computationally efficient synthesis techniques are presented. In addition, the effects of the parameter variation for the design of sound variants are presented.

Keywords: sound synthesis, computer music, computational efficiency, synthesis algorithms, sound effects, sound design.

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Jussi Pekonen
Työn nimi:	Laskennallisesti tehokas musiikkisynteesi – Menetelmät ja äänisuunnittelu
Päivämäärä:	1. kesäkuuta 2007 **Sivuja:** 80+xi
Osasto:	Sähkö- ja tietoliikennetekniikka
Professuuri:	S-89
Työn valvoja:	Professori Vesa Välimäki
Työn ohjaajat:	Professori Vesa Välimäki

Tässä diplomityössä esitetään musiikkisyntetisaattorin suunnittelua systeemille, jonka laskentateho ja muistikapasiteetti ovat rajoitettuja. Ensiksi kerrataan mahdollisia synteesitekniikoita sekä arvioidaan niiden käyttökelpoisuutta laskennallisesti tehokkaassa musiikkisynteesissä. Käytännössä käyttökelpoiset tekniikat ovat lisäävä ja lähdesuodinsynteesit, ja erikoistapauksissa taajuusmodulaatio-, aaltotaulukko- ja samplaussynteesit.

Tämän jälkeen käyttökelpoisten tekniikoiden rakenteiden suunnittelua esitetään tarkemmin, sekä esitetään näiden rakenteiden ominaisuuksia ja suunnitteluongelmia. Suurin ongelma kohdataan digitaalisessa lähde-suodinsynteesissä, jossa klassisten aaltomuotojen, kuten saha-aallon käyttö lähdesignaalina on ongelmallista laskostumisen takia, joka johtuu aaltomuodossa olevista epäjatkuvuuksista. Olemassa olevia kaistarajoitettuja aaltomuotosynteesimenetelmiä kerrataan, ja polynomimuotoiseen kaistarajoitetuun askelfunktioon perustuvaa menetelmää esitellään tarkemmin antamalla suunnittelusääntöjä käyttökelpoisille polynomeille. Menetelmää testataan lisäksi kahdella kolmannen asteen polynomilla. Nämä polynomit vähentävät laskostumista korkeilla taajuuksilla enemmän verrattuna ensimmäisen asteen polynomiin, mutta pienillä taajuksilla ensimmäisen asteen polynomi tuottaa parempia tuloksia. Lisäksi kerrataan muita mahdollisia ääniefektialgoritmeja ja arvioidaan niiden käyttökelpoisuutta laskennallisesti tehokkaassa musiikkisynteesissä.

Useasti äänisynteesisysteemin täytyy pystyä generoimaan musiikkia, jossa käytetään monia erilaisia ääniä, jotka ulottuvat oikeista akustisista soittimista elektronisiin soittimiin ja luonnon ääniin. Siksi tällainen systeemi tarvitsee huolellista äänten suunnittelua. Tässä diplomityössä esitetään suunnittelusääntöjä erilaisten äänien imitoimiseksi. Lisäksi esitellään synteesimenetelmien parametrien vaikutus äänivarianttien suunnitteluun.

Avainsanat: äänisynteesi, tietokonemusiikki, laskennallinen tehokkuus, synteesialgoritmit, ääniefektit, äänisuunnittelu.

Acknowledgements

This Master's thesis has been done in the Laboratory of Acoustics and Audio Signal Processing at Helsinki University of Technology. The research was funded by VLSI Solution Oy, and was conducted during years 2006–2007.

I want to thank my supervisor and instructor Professor Vesa Välimäki for his guidance and support. I would also like to thank Dr. Teppo Karema and Mr. Tomi Valkonen for comments and Mr. Antti Huovilainen for fruitful discussions on this topic.

My gratitude also goes to the personnel of Acoustics Laboratory, who had made my three years in the laboratory much funnier than I ever expected. Without your enthusiastic attitude towards the essence of sound and practical pranks many things would have not happened.

I would like to thank my parents, Kaarina and Heikki, as well as my big sister Virpi and little brother Kari for their support through these years I have been wandering in this world.

Finally, I would like to thank my wife Susanna for her love and devotion. And my little daughter Kaisa, to whom I dedicate this work, hopefully you will enjoy the joys of sound at least as much as I do.

Otaniemi, June 1, 2007

Jussi Pekonen

Contents

List of Figures

List of Tables

List of Abbreviations

List of Symbols

Chapter 1

Introduction

1.1 Background

Despite the continuously increasing computing power of microprocessors, there are still applications where the efficiency of a real-time sound synthesis algorithm is a major issue. This issue is raised for example in gaming and mobile phone industry, where either the allocated computing time of the synthesis algorithm or the computing power is limited. In addition, the memory capacity of the system may be limited, and this limitation must also be taken into account.

Since many of the applications are generating polyphonic music, i.e., multiple sounds are played at same time, the limitations become even stricter. An example of this can be found in the generation of ring tones for mobile phones, which is based on some version of the Musical Instruments Digital Interface (MIDI) format [1]. Most commonly the MIDI version used is either General MIDI Lite (GM-LITE) or Scalable Polyphony MIDI (SP-MIDI). In GM-LITE the sound generating system must be capable to play 16 notes simultaneously, while in SP-MIDI the polyphony can be scaled depending on, e.g., the device model and overall power consumption of the device.

In MIDI specification, the sounds that a MIDI compatible system must be capable of producing are also defined [1]. For instance, in General MIDI Level 1 (GM-1) specification, which the SP-MIDI and GM-LITE are based on, there are 128 different melodic sounds in total. In addition, the specification defines one percussion set, containing 47 different percussion sounds. General MIDI Level 2 (GM-2) specification extends the GM-1 specification further by introducing mostly variants to the GM-1 sounds, defining 256 melodic sounds in total. GM-2 also extends the GM-1 percussion set and defines eight additional

sets.

All of the abovementioned MIDI specifications define the sound set requirements for the system, but the implementation of the sounds is left to a sound designer. The specifications do not specify the sound synthesis technique to be used, and therefore the designer can choose a method that suits best for the application. In addition, the MIDI specifications do not specify how the instruments should actually sound like, and the definition is therefore also left to the designer. However, many of the specified sounds are based on a real acoustical instrument, while some of the sounds are from the nature. In addition, some of the specified sounds have no real world counterparts, and the implementation is therefore left to the designers imagination. Therefore, the overall problem is to find the synthesis techniques, with which the desired sounds can be synthesized with minimal redundancy. In other words, the sounds that have real world counterparts should sound like their original models, and the other sounds should match the impression that the verbal description found from the specification gives.

When designing the sound synthesis for a system with restricted computing abilities, the design problem becomes more difficult. While some sounds can be implemented with good quality using only a certain synthesis technique, the implementation of some other sounds can be difficult or computationally inefficient with that specific technique. This forces the designer to implement some sounds with reduced quality, or the system must be capable of produce sounds with a few different synthesis techniques. Both of these issues, computationally efficient synthesis techniques and sound design, are addressed in this thesis.

1.2 Outline of the Thesis

This thesis is structured as follows. In Chapter 2, various sound synthesis techniques and their properties are reviewed. In addition, the use of the presented synthesis methods in computationally efficient music synthesis is discussed. Chapter 3 takes a closer look to the presented synthesis techniques by examining what methods their implementation requires. The properties and design issues of the methods are discussed in detail. In Chapter 4, design rules for sounds of different timbres are discussed. In addition, the effects of synthesis parameter variations are presented. Finally, Chapter 5 concludes this thesis and points the directions of possible future studies.

Chapter 2

Sound Synthesis Techniques

In this chapter, various sound synthesis methods are introduced. In addition, their properties and applicability in computationally efficient music synthesis are discussed. In Section 2.1, synthesis techniques based on table look-up are presented. Section 2.2 presents techniques based on spectral modelling. In Section 2.3, methods based on modulation of the signal are presented. Section 2.4 presents techniques based on physical modeling of the instruments. Section 2.5 concludes the presented methods and discusses their properties and applicability for practical computationally efficient music synthesis.

2.1 Table Look-up Synthesis

Techniques which involve reading data values from a memory are commonly called as table look-up synthesis. The data to be read is usually the values of the pressure function of the desired sound, and the data is called a wavetable. The sound is therefore produced by playing back the wavetable in appropriate way. Section 2.1.1 presents sampling, probably the simplest implementation of table look-up synthesis. In Section 2.1.2 a general form of table look-up synthesis, wavetable synthesis, is presented.

2.1.1 Sampling

The most intuitive sound synthesis method is to play back digital recordings, sample wavetables, from the memory. This synthesis technique is called sampling. The sampling devices, samplers, are designed to play these prerecorded sounds, shifted to the desired pitch. The sample wavetables contain thousands of individual cycles of the sound, thus producing a rich and time-varying sound. The length of the sample wavetable can be arbitrarily long,

limited only by the memory capacity [2]. Figure 2.1 presents high level block diagram of a sampling synthesizer.

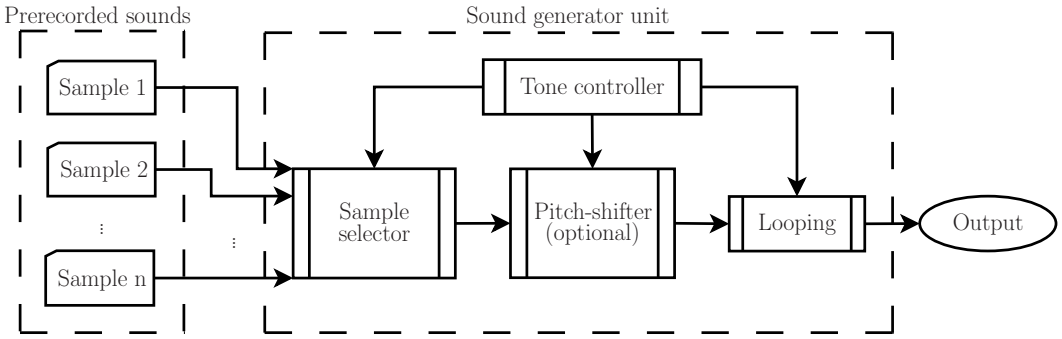


Figure 2.1: Block diagram of a sampling synthesizer.

When using sampling synthesis, issues of pitch-shifting, looping and data reduction must be addressed. By pitch-shifting a smaller set of sample wavetables can be used and the intermediate pitches are obtained by shifting the pitch of the nearest stored sample wavetable. Looping means extending seamlessly the duration of the sampled sound until the sound reaches its release phase. In data reduction, the memory size that a sample wavetable requires is reduced, or the data is compressed [2].

Pitch-shifting is implemented either by varying the clock frequency, i.e., the sampling frequency of the output of a digital-to-analog converter, or by keeping the sampling frequency constant through the whole system and by utilizing re-sampling of the signal. The first method can be implemented with only one converter, which must then operate at wide ranges, or with several different converters, which are used only at certain operation ranges. In the latter method, the sample-rate of the signal is altered in the digital domain, and it needs only one digital-to-analog converter. However, the re-sampling process adds little noise to the signal and it may cause aliasing if it is not carefully designed. In addition, the re-sampling shift also the spectrum variations such as vibrato and tremolo by the same shifting amount, which is often not desirable [2].

Looping is implemented using a special loop beginning and ending points, in between which the sound is played repetitively until the release phase is triggered. This is usually contolled by a musician via a musical keyboard, and the trigger to the release phase is given by releasing the key. The looping can be unidirectional, i.e., the loop is always played back from the beginning point to the ending point, or it can be bidirectional, i.e., the playback changes its direction at the looping points. By bidirectional looping a smooth loop is produced, in which vibrato and other variations in the signals are more realistic sounding [2].

In order to reduce the memory consumption, data reduction or compression of the sample wavetables is needed. In data reduction, non-essential data is omitted, in practice by reduction in sample resolution or quantization. In addition, perceptual analysis of the sounds can also be utilized to reduce, e.g., masked partials. In data compression, redundancies in the data are used in coding it [2].

The sound quality of sampling synthesis is excellent, if the memory consumption is not an issue. However, the sound quality is sort of mechanistic, and since the naturalness and realism are usually the judging criteria, this synthesis technique is probably not the best. In addition, the sound quality depends on the instrument to be recorded, since the playing style affects the resulting sound quality. However, also the recording technique affects the sound quality [2].

2.1.2 Wavetable Synthesis

Since most musical sound waves are repetitive, a fact that is reflected to the notions of frequency and pitch, an efficient synthesis method is to store the values of a single period of a tone into memory, to a so called wavetable. In order to produce the same tone, the stored wavetable is read in unidirectional loop sample by sample again and again. A sound synthesis technique implementing these procedures is called wavetable synthesis [2, 3, 4], and it can be understood as an extension of sampling synthesis. The block diagram of a wavetable synthesis is similar to block diagram of sampling synthesis. It can be obtained from Figure 2.1 by replacing samples with wavetables and using a wavetable selector instead of sample selector. In looping the wavetable is read through in whole, not in just some short range.

To produce tones of different pitch, the sample increment for the table look-up must be changed. The sample increment ΔN can be calculated by

$$\Delta N = f_0 \frac{L}{f_s}, \tag{2.1}$$

where f_0 is the fundamental frequency of the tone in Hertz, L is the length of the wavetable in samples, and f_s is the sampling frequency in Hertz [2]. Since the fundamental frequency f_0 can be arbitrary, the sample increment ΔN is not always an integer. There are several solutions to overcome this problem.

First, the sample increment can be truncated to its integer part, but this produces table-look-up noise. By using a larger wavetable the table-look-up noise can be decreased. On the other hand, the larger wavetable consumes more memory. Another solution to decrease the table-look-up noise is to round the increment instead of truncation. Secondly, by changing only the sampling frequency different pitches can be produced. This can be implemented

by varying the clock frequency of the output of the digital-to-analog converter or by re-sampling, as in sampling synthesis (see Section 2.1.1) [2].

The best solution to the non-integer sample increment is to interpolate the wavetable value at the obtained position. With interpolation smaller wavetables than without interpolation can be used to obtain the same sound quality. Yet, depending on the interpolation method, a good sound quality is obtained with a wavetable of different lengths [2, 3]. The simplest interpolation technique is linear interpolation, in which the value is assumed to be on the straight line between the adjacent samples of the wavetable. However, more sophisticated interpolation methods exist, such as sinc interpolation, in which no aliasing is produced [5, 6]. Interpolation can be implemented efficiently with fractional delay filters [7, 8, 9].

Although the wavetables are usually small in size, a large number of different wavetables can consume much memory. Therefore data reduction and compression issues must be addressed also in wavetable synthesis. Most commonly the data compression is implemented by differential coding, where only the difference between adjacent samples is stored [10, 11]. However, other coding methods have been proposed, e.g., a parametric representation of the wavetable instead of direct sample values. The parametrization can be implemented with a model where the signal is divided into transient, sinusoid and noise components [12].

In order to produce time-varying timbres, some modifications to the wavetable synthesis technique can be implemented. In wavetable crossfading, the synthesizer crossfades between two or more wavetables over the course of an event instead of scanning only one wavetable. In wavetable stacking, a set of wavetables are mixed with their corresponding envelope functions [2, 3]. Alternatively, the wavetable look-up can be from three-dimensional ”wave surfaces” [2], or the look-up can be an interpolation between two or more wavetables for different notes [3]. In addition, a combination of sampling and wavetable synthesis can be utilized. Sampling is used for the attack, and wavetable synthesis is used in the tone’s decay phase [13].

Wavetable synthesis with good sound quality is obtained by finding wavetable spectra and the associated amplitude envelopes which provide a close fit to an original time-varying spectrum. This can be done with Genetic Algorithm or with Principal Components Analysis methods [14, 15], or by grouping the harmonics of the signal into separate wavetables [16].

2.2 Spectral Modeling Synthesis

In spectral modeling, the sound synthesis is designed to match the spectrum of the reference signal at every time instant. Every signal can be decomposed to deterministic and stochastic components, which correspond to sinusoids and noise, respectively [17, 18]. Therefore it is possible to build a synthesis model where the signal is represented with periodic waveforms and an added noise component. Section 2.2.1 presents a synthesis method called additive synthesis, in which the deterministic component is modeled with a sum of sinusoids. In Section 2.2.2, a wavelet extension to additive synthesis, granular synthesis, is presented. Section 2.2.3 presents a synthesis method called source-filter synthesis, in which the deterministic component is modeled with a single filtered waveform.

2.2.1 Additive Synthesis

Additive synthesis, as its name suggests, is based on summation of sinusoidal components to generate a spectrally more complex waveform [2]. The output signal $y[n]$ of an additive synthesis sound generator can be represented by

$$y[n] = \sum_{k=1}^{M} A_k[n] \sin(\varphi_k[n]) + v[n], \tag{2.2}$$

where M is the number of signal components, $A_k[n]$ is time-varying amplitude of the k^{th} component, and $\varphi_k[n]$ is instantaneous phase of the k^{th} component. In addition, the generator may add colored noise $v[n]$ to the resulting signal. The instantaneous phase $\varphi_k[n]$ can be decomposed into two parts by

$$\varphi_k[n] = \omega_k[n] + \phi_k[n], \tag{2.3}$$

where $\omega_k[n]$ is possibly time-varying angular frequency of the k^{th} component and $\phi_k[n]$ is possibly time-varying phase shift of the k^{th} component.

From Equations (2.2) and (2.3) three control functions can be gathered; amplitude, frequency and phase. The impact of the components to the output signal are determined by the amplitude functions. With the frequency functions the timbre of the output signal can be varied. The phase is an issue, which can be significant depending on the context. A variation of a component phase does not usually affect the perceived sound, but it changes the visual appearance of the waveform. In short attacks and transients the perception of the phase relationships becomes more relevant [2]. In Figure 2.2, block diagram of an additive synthesis synthesizer is presented.

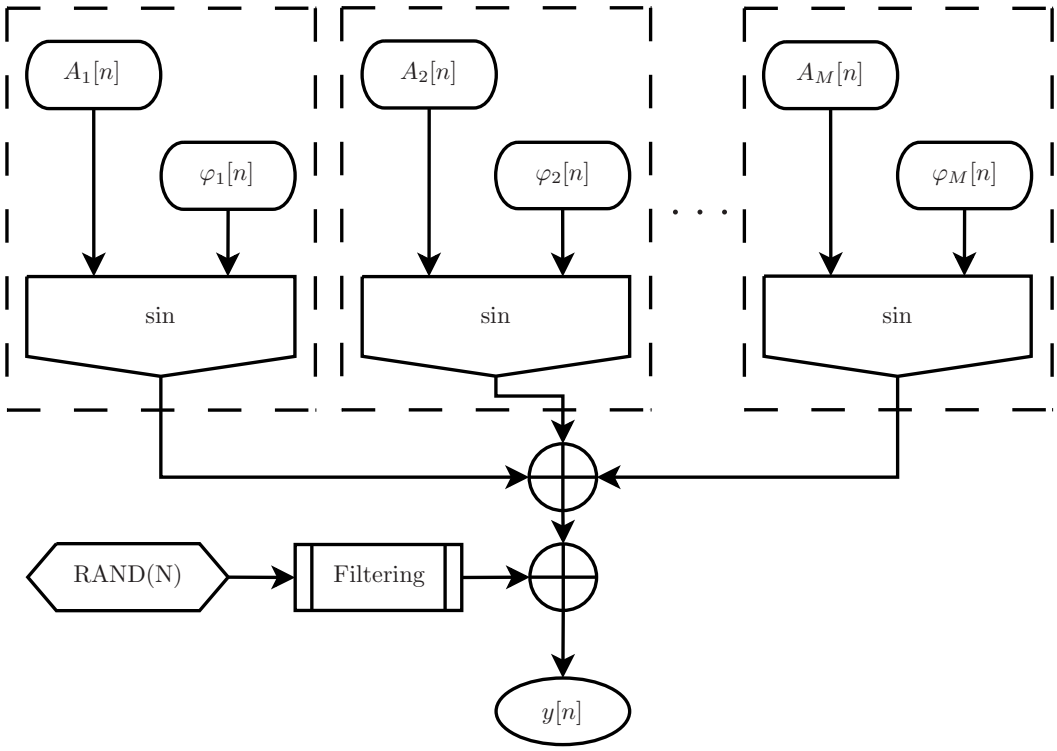


Figure 2.2: Block diagram of an additive synthesizer with filtered noise.

The control functions can be obtained with several procedures. First, the function shapes can be arbitrary, inspired by for instance sky lines of a town or by the shape of a mountain. Secondly, analysis can be applied to recorded instrument sounds. The analysis can be performed with Short-Time Fourier Transform (STFT) [2], phase vocoder [19, 20, 21], or McAulay-Quatieri algorithm [22]. In STFT, the components of the signal are mapped to peaks of short-time spectra from which they are picked to form the control functions of a signal component [2].

In phase vocoder, the signal is presented in multiple parallel channels each describing the signal in a specific frequency band, which allows predictable signal alternations, such as time scaling without pitch changes and pitch shifting without changing the temporal evolution of the signal [19, 20]. When altering the time scale of the sound, effect of ”phasiness” is introduced to the signal, and it can be overcome using phase synchronization in the altering stage [23]. Phase vocoder can be implemented efficiently using STFT [24].

McAulay-Quatieri algorithm is an extended version of phase vocoder with peak tracking. The signal is first windowed using a zero-phase window, and STFT is calculated in polar coordinates for the zero-phase signal. Peak detection is applied to the magnitude spectrum

to separate the most prominent signal components, and peak continuation tracking is applied to these peaks. If the frequency of a peak is within certain range, new amplitude, frequency and phase trajectory values for that peak are passed to the following frame. In the contrary case, zero amplitude, same frequency and shifted phase are passed forward. In addition, if there is no preceding trajectory for a component, a new trajectory is created [22].

One interesting method for implementing an additive synthesizer is Inverse FFT synthesis (FFT^{-1}) [25, 26]. The signal components are composed in frequency domain from consecutive STFT frames, from which the actual signal is constructed by calculating the inverse Fast Fourier Transform (FFT) of each frame, and the consecutive frames are attached to each other with overlap-add method. In addition, adding the colored noise component is straightforward in the frequency domain representation [18].

As from Equations (2.2) and (2.3) can be seen, the analysis data takes many times more memory space than the original analysis signal, and therefore the issue of data reduction must be addressed. In data reduction, the signal is first analyzed using one of the above-mentioned techniques, and then an algorithmic transformation is performed to the data in order to obtain a more compact representation. The amount of sinusoids can be decreased by performing Principal Components Analysis, in which the sinusoids are weighted according to their fit to the original signal [2]. On the other hand, the amplitude, frequency and phase trajectories can be smoothed using piece-wise approximations, in which the trajectories are replaced with piecewise curves which approximate the trend of the trajectories [2, 27]. Alternatively, the trajectories of adjacent frames can be interpolated in order to obtain smoother transitions. The analysis data is reduced by grouping the common transition paths of the sinusoids and by defining transition ramps between different spectra. However, this procedure has difficulties in processing the attacks of the sound [2].

2.2.2 Granular Synthesis

Granular synthesis is based on representing the sound signal with ”sound atoms” or grains. The grains can be basically anything, e.g. windowed or distorted sinusoids or even sample wavetables, and their durations can be anything between one millisecond and hundreds of milliseconds [28]. Therefore granular synthesis can be seen as a wavelet extension of additive synthesis, and the sinusoids of the additive synthesizer are replaced with the grains, i.e.,

$$y[n] = \sum_{k=1}^{M} A_k[n] g_k[n], \tag{2.4}$$

where $g_k[n]$ is the k^{th} grain component.

Depending on how the grains are obtained from the signals, the granular techniques can be divided into two categories, asynchronous and pitch synchronous. In asynchronous granular synthesis, the sound grains are scattered statistically over a region in time-frequency plane [28]. These regions, clouds, form the elementary units, and they have the several parameters: start time and the duration of the cloud, grain duration, density of grains, amplitude envelope and bandwidth of the cloud, waveform of each grain, and spatial distribution of the cloud. The grains of a cloud can have arbitrary waveforms, similar to each other or composing a random mixture, and the waveform of each single grain can be time-varying. The bandwidth of a grain is affected by its duration, and therefore a rich variety of timbres can be obtained by varying the grain durations. Asynchronous granular synthesis can produce effectively sound events, which can not be easily produced with musical instruments, but simulation of real world sounds are extremely hard to produce.

In pitch synchronous granular synthesis, the grains are obtained via STFT, and the signal to be analyzed is assumed to be nearly periodic. The fundamental frequency of the signal is estimated and the estimate is used in synchronous windowing with rectangular window, hence minimizing the side effects of windowing. After windowing, each grain is set to correspond one period of the signal, and from these grains the impulse responses corresponding prominent content in the frequency domain representation are derived. This estimation is obtained using Linear Predictive (LP) coding or interpolation of the frequency domain representation of a grain. The synthesized sound is obtained by driving a set of parallel Finite Impulse Response (FIR) filters having the analyzed impulse responses by a train of impulses with the period corresponding to the estimated fundamental frequency [28].

2.2.3 Source-Filter Synthesis

In source-filter synthesis, as the name suggests, the sound waveform is obtained by filtering a source signal, which can be generally be an arbitrary waveform or noise. Quite often source-filter synthesis is called subtractive synthesis, since in this synthesis method the source signal is usually a broadband signal or a harmonically rich waveform which is then filtered to obtain the desired sound, which is the opposite to additive synthesis approach. Source-filter synthesis has been used in speech synthesis for a long time, but it has musical applications too [29, 27].

In theory, the source signal can be arbitrary, but since any periodic waveform can be generated from an impulse train by applying appropriate filtering, the source-filter synthesis synthesizer block diagram can be simplified to be as simple as it is presented in Figure 2.3.

However, in many cases the impulse train generator is replaced with a simple waveform oscillator which produces a complex waveform, for example the sawtooth or rectangular pulse wave. By using these complex waveform oscillators, the filter can be of lower order and the whole synthesis structure becomes more simplified and efficient. In addition, the source signal can be composed of several different signals.

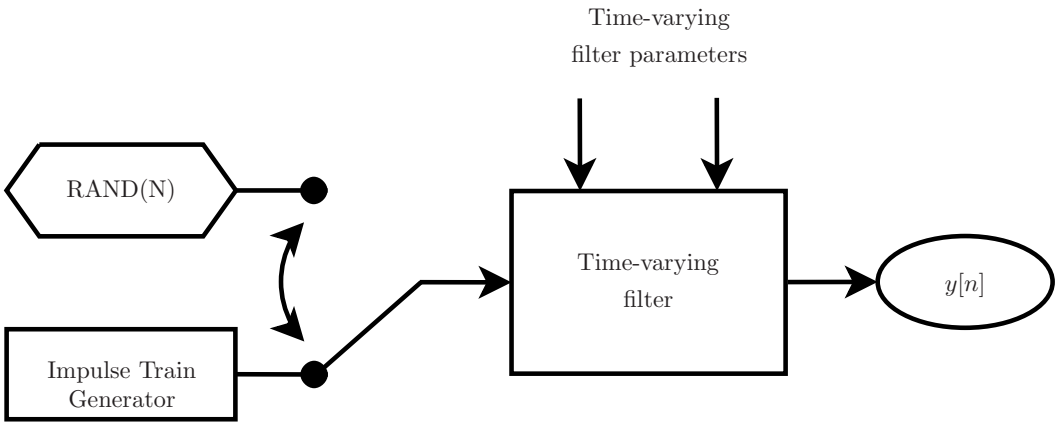


Figure 2.3: Block diagram of a source-filter synthesizer.

The use of source-filter synthesis requires two analysis stages to implement. First, a pitch detection must be utilized to determine the period of the waveform. If a fundamental frequency can be detected, the source is then set to be the impulse train with that specific frequency. In the opposite case, the noise source is used. The pitch detection can be implemented with various different methods, and it has been studied extensively. The methods can be divided into five categories, time-domain methods, autocorrelation-based methods, adaptive filtering methods, frequency-domain methods, and methods which utilize the models of human ear [2].

Secondly, the filter parameters must be determined. Quite often the filter is a recursive filter, and for such filters there are efficient filter coefficient estimation techniques. One of the most commonly used coefficient estimation techniques is to apply linear predictive (LP) analysis to the signal [30]. In LP, the basic idea is to match the magnitude response of an all-pole filter to the analysis signal. Opposed to STFT, where the exact magnitude and phase responses on a large number of equally spaced frequencies is produced, in LP only the magnitude spectrum envelope is in interest. In addition, since LP is a parametric method, it provides the best match in minimum-squared-error sense.

By choosing appropriate source signals and filters, a rich variety of individual sounds can be produced with source-filter synthesis. However, it is not a robust technique for producing

generic wideband audio signals. Yet, in many cases the source-filter synthesis method and its variants are efficient techniques, and therefore it is widely used.

2.3 Modulation Synthesis

In modulation synthesis, the desired sound is produced by performing a transformation to another signal. The transformation can in principle be arbitrary, but there are some simple special cases which are commonly used. One of these simple cases is amplitude modulation, presented in Section 2.3.1, in which the signals amplitude is varied by another signal. In Section 2.3.2 is presented another simple transformation called frequency modulation, in which the frequency of a signal is varied by another signal. Finally, Section 2.3.3 presents a general modulation synthesis technique called waveshaping synthesis, in which a signal is modified with a nonlinear shaping function.

2.3.1 Amplitude Modulation

Amplitude Modulation (AM) was first developed for radio transmission in the early 20^{th} century [31], and it is not conventionally expressed as an individual sound synthesis technique. However, it is widely used as an additional sound effect, and it can be used to create variants of an existing sound.

The basic idea of AM is that the amplitude a sound signal follows a unipolar modulator signal, i.e., the amplitude of the sound signal is multiplied with positive modulator value at each time instant. This can be expressed with

$$y[n] = (C[n] + m(n, \varphi_m[n]))w(n, \varphi_w[n]), \tag{2.5}$$

where $m(n, \varphi_m[n])$ is the modulator signal with instantaneous phase $\varphi_m[n]$, and $w(n, \varphi_w[n])$ is the original sound waveform with instantaneous phase $\varphi_w[n]$. $C[n]$ is possibly time-varying constant by which the modulation is kept unipolar. If the modulator signal is bipolar, i.e., it has both positive and negative values, the modulation technique is called Ring Modulation (RM) [2]. Block diagram of a AM/RM synthesizer is presented in Figure 2.4.

In AM, the components of $m(n, \varphi_m[n])$ spread around the components of the modulated waveform $w(n, \varphi_w[n])$, while in RM the components of the modulated waveform are missing, and their energies are transformed into the sidebands. This can be derived by investigating simple AM with sinusoids. In the simplest case, $C[n] = c$, $m(n, \varphi_m[n]) =$

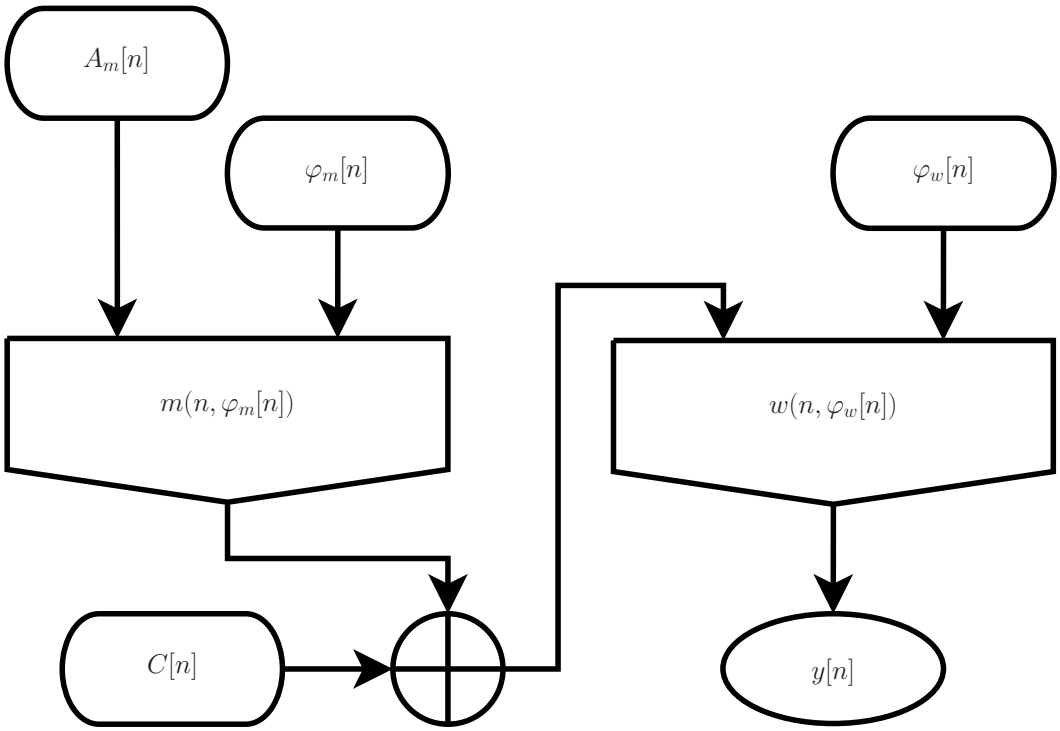


Figure 2.4: Block diagram of a AM/RM synthesizer.

$a\sin(\omega_m[n])$ and $w(n, \varphi_w[n]) = \sin(\omega_0[n])$, with which Equation (2.5) can be rewritten as

$$\begin{aligned} y[n] &= (c + a\sin(\omega_m[n]))\sin(\omega_0[n]) \\ &= c\sin(\omega_0[n]) + \frac{a}{2}(\sin(\omega_0[n] + \omega_m[n]) + \sin(\omega_0[n] - \omega_m[n])) \end{aligned}$$

using the properties of trigonometric functions. Since an arbitrary signal can be modeled with a sum of sinusoids (see Section 2.2), the derived result can be extended to any kinds of signals. In both AM and RM, the additional frequencies generated by the modulation are, in general, in inharmonic relation to the partials of the modulated signal. In addition, RM has characteristically metallic sound [32].

2.3.2 Frequency/Phase Modulation

Frequency Modulation (FM) was also first developed for radio transmission in mid 1920s [31]. FM was not applied to audio frequencies and sound synthesis purposes until late 1960s [33]. In FM synthesis, the instantaneous phase of a sound signal is varied with a modulator signal, i.e., the frequencies of the original waveform oscillate around their

nominal values along the modulator signal. Recalling Equation (2.3), the instantaneous phase $\varphi[n]$ of a waveform can be decomposed to frequency and phase parts. Therefore a related modulation technique called Phase Modulation (PM) can be understood as a special case of FM, or other way round. In practice, there is no other difference between FM and PM than the synthesis parameter values, since in the FM the signal phase is varied with the integral of the modulating signal [31].

The output of a FM/PM synthesizer is given by

$$y[n] = w(n, \varphi_0[n] + \beta[n]m(n, \varphi_m[n])), \tag{2.6}$$

where $\varphi_0[n]$ is the instantaneous nominal phase and $\beta[n]$ is possibly time-varying FM/PM deviation parameter. In Figure 2.5, block diagram of a FM/PM synthesizer is presented.

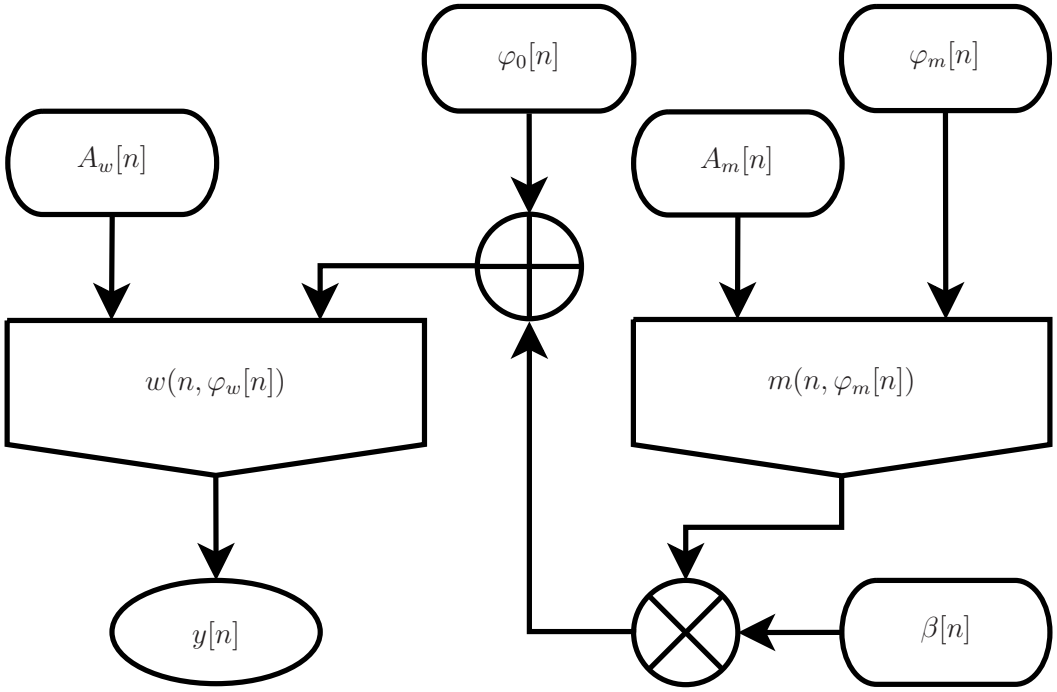


Figure 2.5: Block diagram of a FM/PM synthesizer.

With FM, rather complex audio spectra can be obtained using only two sinusoidal oscillators. In this case, when the phase shifts are omitted, Equation (2.6) becomes

$$y[n] = A[n]\sin(\omega_0[n] + \beta[n]\sin(\omega_m[n])),$$

which can be rewritten

$$y[n] = \sum_{k=-\infty}^{\infty} J_k(\beta[n])\sin(\omega_0[n] + k\omega_m[n]), \tag{2.7}$$

where $J_k(\tau)$ is the Bessel function of order k [33]. Since an arbitrary signal can be modeled with a sum of sinusoids (see Section 2.2), the derived result can be extended to any kinds of signals [34].

The simple FM synthesis of Figure 2.5 suffers from some control problems. From Equation (2.7) can be seen, that FM produces symmetric spectra around the frequencies of the modulated signal. However, the symmetries of the spectra can be controlled via additional control parameter [35]. In addition, by using two or more, simple or complex modulator signals, the resulting spectrum can be modified [36]. The resulting timbre is also affected by the phase relation between the modulator and the modulated signal [37] and the implementation of the FM algorithm [38].

In a simple FM synthesizer implemented as in Figure 2.5, the amplitude ratios of the newly generated signal components vary unevenly when the deviation parameter β is varied. This problem can be overcome by using feedback FM [39]. In a simple feedback FM synthesizer, the frequency of single oscillator is modulated according to its output. In two-oscillator feedback FM synthesizer, the feedback is used to drive the modulator oscillator.

2.3.3 Waveshaping Synthesis

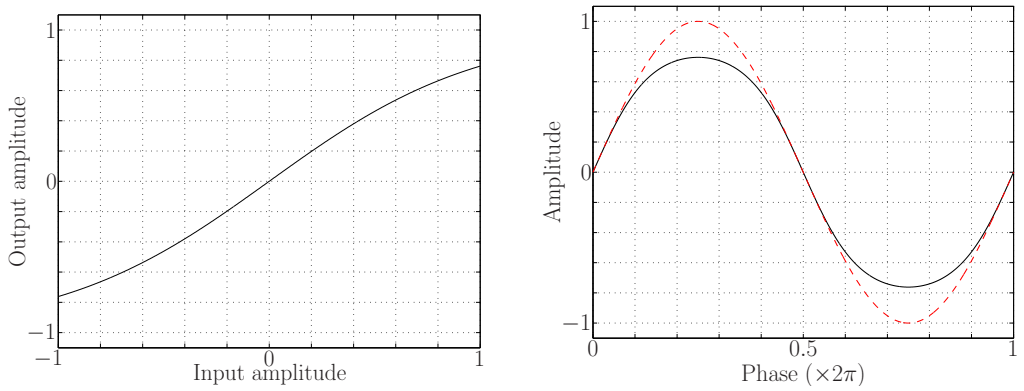
In waveshaping synthesis, a signal is modified with a nonlinear shaping function [40, 41]. The output of a waveshaping synthesizer is the instantaneous value of a source signal $w(n, \varphi_w[n])$ is processed with a shaping function $F(\tau)$, given by

$$y[n] = F(w(n, \varphi_w[n])). \tag{2.8}$$

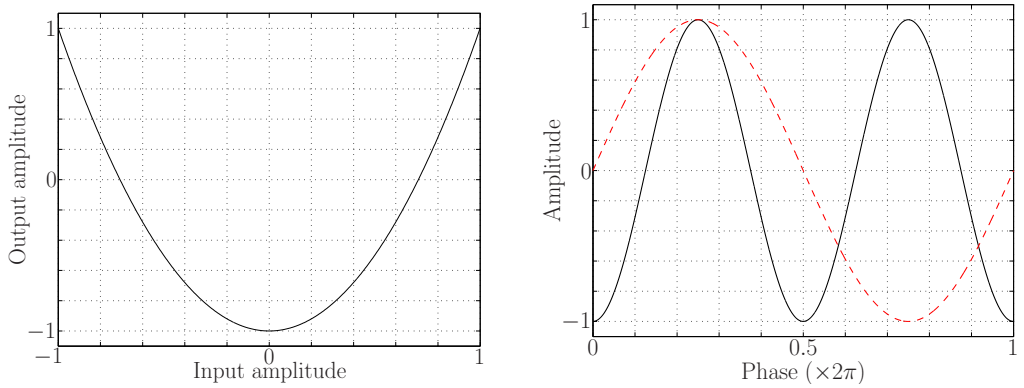
If the shaping function $F(\tau)$ is linear, it does not alter the source signal waveform but scales the amplitude. If the shaping function is nonlinear, it introduces new frequency components to the output with respect to the source signal. An example of a nonlinear shaping function is given by Equation (2.6), where $m(n, \varphi_m[n])$ is the source signal and the shaping function is $w(n, \varphi_w[n])$. In Figure 2.6, two examples of waveshaping functions and their respective outputs to sinusoid source signal, plotted with dashed line on the right pane, are presented.

The shaping function should not contain any discontinuities nor sharp edges, since they produce unlimited number of additional harmonic frequencies, which causes aliasing. Bandlimiting is also difficult when the source signal $w(n, \varphi_w[n])$ has a complex waveform. Therefore it is usually sinusoidal. In addition, the the signal obtained by waveshaping can be postprocessed, e.g., by another shaping function.

Most commonly the shaping function $F(\tau)$ is a polynomial, due to many useful properties of polynomials. First, any smooth curve can be approximated well with a polynomial,



(a) tanh function



(b) Second order Chebychev polynomial

Figure 2.6: Examples of waveshaping functions and their respective outputs to a sinusoid source signal.

and therefore huge amount of shaping functions can be represented with them. Secondly, the order of the polynomial defines the highest produced harmonic when the input signal is sinusoidal. Thirdly, the amplitude relations between the produced harmonics can be adjusted with the polynomial coefficients. In addition, dynamic behavior of the produced harmonics can be adjusted with the amplitude of the source sinusoid [40, 41].

Most commonly polynomial waveshaping is implemented with Chebychev polynomials. A Chebychev polynomial of order k, $T_k(x)$, can be calculated recursively with

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_k(x) &= 2xT_{k-1}(x) - T_{k-2}(x), \end{aligned}$$

and such polynomials have a property that makes them interesting. If $T_k(x)$ is applied to a

sinusoid of frequency f_0, the output is a sinusoid with frequency of kf_0, i.e.,

$$T_k(\cos(\alpha)) = \cos(k\alpha).$$

Therefore a desired shaping function can be obtained by combining a set of Chebychev polynomials of desired orders and desired weights, and the resulting signal can be maintained bandlimited [40, 41].

Chebychev polynomials are the most commonly used, but there are also other possible polynomial waveshaping functions. Interesting shaping functions can be obtained using cubic Bezier curves [42]. By modulating the control points of the Bezier curve, the resulting sound can mimic the sound of AM or FM synthesizer.

2.4 Synthesis by Physical Modeling

While all of the previously presented synthesis techniques attempt to model the produced sound, physical modeling synthesis techniques have totally opposite approach. Physical modeling synthesis attempts to imitate the sound production events of a musical instrument, e.g., string vibration. Physical modeling is based on either solving numerically formulas and differential equations derived from the physics of the structure of the instrument or designing a model which behaves as the original instrument [43, 44, 45]. In modal synthesis, presented in Section 2.4.1, the imitation of the sound production mechanism of an instrument is modeled with a rather small set of resonators which are excited by some signal to produce the desired sound. Section 2.4.2 presents digital waveguide modeling which attempts to duplicate the behavior of the instrument, and it implements a physical model of the sound production.

2.4.1 Modal Synthesis

Modal synthesis is based on assumption that any sound-producing object can be modeled with a set of resonators, which imitate the behavior, modes, of vibrating substructures of the object. The substructures are coupled and they can respond to external excitations. The substructures are typically bodies and bridges, bows, acoustic tubes, membranes and plates, and bells of a musical instrument. The modal synthesis algorithm attempts to simulate each of the substructures and their interconnections [46].

The modal data for a resonator is formed of the resonance frequencies, damping coefficients and the shape of the resonant modes. The modes are excited at a given point of the object by an external force, and the energy of the excitation is distributed to the modes depending

on the type of the excitation. Since a vibration pattern is a sum of infinite series of vibrating modes, computationally realizable synthesis requires a finite set of spatially divided points, in which the system is evaluated. If the number of points is N, N modes can be represented with a so called shape matrix $\mathbf{\Phi}$, which describes the relative displacements, i.e., the relative amplitude, of the points in each mode. The modes can be presented as second-order resonators in parallel [46].

The modal data for simple structures can be obtained analytically from the differential equations representing the vibration of the structure. However, for complex structures analytical approach is not possible, and therefore analysis from measurement experiments must be utilized. Once the modal data is obtained, it can be used in sound production in straightforward manner, since in modal synthesis all vibrating systems can be described using same equations, which describe the response to the excitation at a given point. The velocities of the modes can be calculated using the equations if all instantaneous excitations are known. However, only the excitation forces are usually known, and the forces implementing the nonlinear substructure coupling must be determined [46].

The resonator-based structure scheme can also be obtained by modeling acoustical systems with simple ideal mechanical elements, i.e., point masses, springs and dampers. With simplifications of treating each element one-dimensional and modeling interactions with an adjustable conditional link, the generated mass-spring network can be expressed with rather simple equations, which lead to structure resembling the structure generated by the modal synthesis [47].

2.4.2 Digital Waveguide Modeling

The fundamental idea of digital waveguides is to model the physics of the sound production of the acoustic waveguides existing in many acoustic instruments. The digital waveguide modeling resembles the finite-difference method, in which the sound production mechanism is modeled by discretizing and solving the differential wave equations representing the investigated system. Since the computation of the finite-difference is quite heavy, the digital waveguides offer a more efficient approach for the synthesis purposes [48, 49, 50, 51].

For instance, the differential wave equation for a one-dimensional vibration is given by

$$\frac{\partial^2 y}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2},$$

and the d'Alembert's solution to this equation is an arbitrary function of form

$$y(x,t) = y_l\left(t - \frac{x}{c}\right) \quad \text{or} \quad y(x,t) = y_r\left(t + \frac{x}{c}\right),$$

where the functions $y_l\left(t-\frac{x}{c}\right)$ and $y_r\left(t+\frac{x}{c}\right)$ can be interpreted as traveling waves going left and right, respectively. The general solution is a linear superposition of these two traveling waves. When the solution is discretized, the resulting signal can be interpreted as superposition of delayed input signal traveling in two directions, which gives the mathematical description of the digital waveguide [48, 49, 50, 51].

The digital waveguides can be extended to model the vibration of a two- of three-dimensional objects by performing similar analysis to the higher order wave equation. The higher order digital waveguides are practically sets of one-dimensional waveguides connected to each other with appropriate scattering criteria [52, 53, 54]. The digital waveguide approach also allows the model to be modified using rather simple additional processing in the waveguide, e.g., dispersion is implemented by inserting an allpass filter before each observation point. In addition, interaction between different parts of the system can be modeled using digital waveguides in series, e.g., the model of a sound box of a guitar can be connected to the model of vibrating string [55].

The problem of using two delay lines in the one-dimensional digital waveguide can be overcome by performing a formulation of the transfer functions describing the bidirectional digital waveguide. By rearranging the system using these transfer functions, a single-delay-loop version of the digital waveguide can be obtained [56]. When the bidirectional models of the instrument parts are transformed into single-delay-loop versions, the order of the parts can be changed without altering the resulting sound. This operation, commutation, means that for example the string model of the guitar can be excited using the response of the guitar body [57, 58]. By this the synthesis system can be simplified, and no additional excitation signal is needed to excite the system.

When the model of an instrument has been constructed, the model parameters must be derived in order to obtain natural sounding output of the model. This is usually performed by analyzing acoustic measurements, which are usually recorded anechoic in order to avoid the side effects of the recording environment, such as noise and room reverberation. The parameters are obtained by performing, e.g., sinusoidal modeling for the signal partials [59].

2.5 Practical Computationally Efficient Synthesis Techniques

The sound synthesis techniques reviewed in the previous sections have originally been developed for different types of purposes, and therefore they have different characteristics which define their applicability in computationally efficient music synthesis. While some

technique may produce the best sounding results, some other technique may be more efficient in the sound production. Therefore, the evaluation of these techniques is not straightforward, since both of these aspects, computational efficiency and the resulting sound quality, must be considered.

In Table 2.1, a summary of the applicability of the reviewed synthesis techniques in computationally efficient music synthesis is given. The evaluation is divided into two categories, implementation and parameter behavior[1] [60]. These categories are used to judge the overall rating of the synthesis technique from the viewpoint of computationally efficient sound synthesis. The computational efficiency (*Comp*) is determined directly by the number of operations required to produce single sample of the resulting sound. As stated in Chapter 1, the memory consumption (*Mem*) of the algorithms plays an important role for some implementations. The memory consumption is rated by the number of state variables in the synthesis technique implementation and the required memory space for the control data. In addition, the controllability (*Contr*) of the synthesis technique provides additional expressions and liveliness to the resulting sound. The rating of the controllability is determined by the system update rate.

Since some sounds can be implemented efficiently using only a certain synthesis technique while the implementation some other sound is difficult or computationally inefficient, there is a tradeoff between the sound quality and the number of synthesis techniques which must be implementable in that specific system. Therefore the evaluation of the synthesis techniques require inspection of the generality (*Gen*), which is ranked by the number of different kinds of sounds the technique can generate. Besides the generality, the usability of the synthesis parameters are also considered. This means that if a synthesis parameter is changed, it should produce a proportional and easily predictable change in the resulting sound, and the identity of the sound should retain. In addition, if the resulting change is barely audible or it is too drastic, the design of different sounds is hard and not intuitive at all. These aspects are judged by the intuitivity (*Int*) and linearity (*Lin*) of the parameter use, respectively. The synthesis parameter correspondence to some physical quantities of a real-world instrument is also judged (*Phys*).

By examining Table 2.1 one can conclude, that no one method is clearly better than some other. If the technique is computationally efficient, usually the parameter control is quite tricky. On the other hand, if the parameter control of the technique is easy, it usually is computationally inefficient. In addition, when the synthesis technique provides good generality and easy parameter control, it usually is quite memory intensive. Therefore,

[1] The rating of the parameter behavior is left out for table look-up synthesis techniques since they do not provide any parameters to control.

Table 2.1: Comparison of the reviewed synthesis methods, adapted from [61].

Synthesis	**Implementation**			**Parameter Behavior**				**Overall**
Technique	*Comp*	*Mem*	*Contr*	*Gen*	*Int*	*Phys*	*Lin*	**Rating**
TABLE LOOK-UP SYNTHESIS								
Sampling	Good	Poor	Good	Good	–	–	–	Poor
Wavetable	Fair	Poor	Good	Good	–	–	–	Poor
SPECTRAL MODELING SYNTHESIS								
Additive	Fair	Fair	Fair	Fair	Fair	Poor	Good	Fair
Granular	Fair	Poor	Poor	Good	Poor	Poor	Good	Poor
Source-Filter	Fair	Fair	Fair	Fair	Fair	Fair	Poor	Fair
MODULATION SYNTHESIS								
AM	Fair	Fair	Fair	Poor	Poor	Fair	Good	Poor
FM	Good	Good	Good	Good	Poor	Poor	Poor	Fair
Waveshaping	Fair	Good	Good	Fair	Fair	Poor	Fair	Fair
SYNTHESIS BY PHYSICAL MODELING								
Modal	Fair	Poor	Good	Fair	Fair	Good	Good	Poor
Waveguide	Fair	Poor	Fair	Fair	Good	Good	Good	Poor

there is a trade off between the controllability and computational efficiency and memory consumption.

Using these criteria, the physical modeling techniques have to be left out. In spite of good generality and sound quality, their memory consumption is rather high, which is one of the main factors in the determination of the applicability. In addition, the table look-up techniques and granular synthesis in general are not desirable, since their memory consumption is high. However, if some sounds can be problematic to generate using other methods, these may provide an intermediate solution. Especially a combined technique of wavetable synthesis and source-filter synthesis may provide quite efficient and good solution to these problematic sounds. If anything else can not be done, sampling of a single sound can be used.

Now, the practical synthesis techniques are limited to *additive synthesis*, *source-filter synthesis*, FM *synthesis* and *waveshaping synthesis*. From these, additive synthesis and source-filter synthesis provide a reasonable balance between the implementation and parameter control, and therefore they are quite good candidates. However, if the parameters of FM synthesizer can be adjusted correctly, it provides computationally the best approach. This,

in fact, is quite hard. The waveshaping synthesis in turn can be in general quite simple and efficient, but its computational load is dependent on the spectrum to be generated. If the spectrum contains only a small number of harmonic components, the computational load is quite low. If the spectrum is more complex and contains inharmonic components, the computational load is increased and is with complex spectra high. Therefore the waveshaping synthesis is rarely used.

However, the technique to be used is ultimately defined by the sound to be modeled. In addition, in many cases the sound design may use some additional synthesis technique as an effect for producing some characteristic phenomena of that specific sound. These two notes are illustrated more thoroughly in the design cases presented in Chapter 4.

Chapter 3

Design of a Synthesizer

In this chapter, the design of the synthesizers implementing the applicable synthesis techniques presented in Chapter 2 is discussed. In addition, the design problems confronted in the implementation design are examined and the respective solution candidates are presented. In Section 3.1, oscillator algorithms for generating the desired fundamental waveforms are presented. Section 3.2 presents the design of the filters used in the sound synthesis. Generation of time-varying control signals and their use in sound synthesis is discussed in Section 3.3. In Section 3.4, additional sound effects used in music synthesis are presented.

3.1 Oscillators

Recalling from Chapter 2, the synthesis methods require oscillators in order to produce the desired waveforms. The waveforms needed are sinusoids, random noise, and classic waveforms, such as the sawtooth, the rectangular pulse and the triangular pulse wave. In Section 3.1.1, algorithms for generating sinusoidal oscillation are presented. Section 3.1.2 discusses the problem of generating random noise signals. In Section 3.1.3, the issues of classic waveform synthesis are addressed.

3.1.1 Sinusoidal Generators

Most commonly the sinusoids are generated in a microprocessor with a special on-chip integrated wavetable, in which the values of a sine function are stored using rather high number of samples, and the desired value is interpolated from the table values. However, the sinusoids can also be generated algorithmically. The algorithmic approaches can be

categorized to polynomial form, direct form, coupled form, modified coupled form, and normalized waveguide form approaches.

In the polynomial form approach, the sinusoid is approximated using an appropriate polynomial, e.g., series or infinite product. The series form approximation can be based, for example, on Taylor series. The Taylor series of the sine and cosine functions near origin are given by [62]

$$\sin(\omega[n]) = \sum_{k=0}^{\infty}(-1)^k \frac{\omega[n]^{2k+1}}{(2k+1)!} \quad \text{and} \quad \cos(\omega[n]) = \sum_{k=0}^{\infty}(-1)^k \frac{\omega[n]^{2k}}{(2k)!}. \tag{3.1}$$

The sine and cosine functions can also be expressed as infinite products obtained by Weierstrass factorization [63], given by

$$\sin(\omega[n]) = \omega[n] \prod_{k=1}^{\infty}\left(1 - \frac{\omega[n]^2}{\pi^2 k^2}\right) \quad \text{and} \quad \cos(\omega[n]) = \prod_{k=1}^{\infty}\left(1 - \frac{\omega[n]^2}{\pi^2\left(k-\frac{1}{2}\right)^2}\right). \tag{3.2}$$

The approximation is obtained by calculating only finite amount of terms in order to make the realization computationally realizable. However, the accuracy of the approximation is reduced by the truncation at points further from the origin. For this reason, the polynomial form approach is not practically used.

In the direct form approach, the sinusoid generator algorithm is obtained using the properties of sine and cosine functions. The direct form oscillator can be expressed as

$$y[n] = 2\cos(\omega[n])y[n-1] - y[n-2], \tag{3.3}$$

where the desired waveform is replaced with $y[n]$. With initial states $y[-1] = -\sin(\omega[0])$ and $y[0] = 0$, this equation implements a sine oscillator, and with $y[-1] = \cos(\omega[0])$ and $y[0] = 1$ a cosine oscillator. If the frequency of the oscillator is changed via the coefficient $2\cos(\omega[n])$ after the oscillator has been running for a while, the amplitude of the oscillator will deviate from unity. This is due to the fact that the algorithm uses two previous values of the waveform which together set the amplitude of the oscillation, and it can be overcome by updating the state variables [64].

The coupled form is also obtained using the properties of sine and cosine functions, and it is given by

$$y_1[n] = \cos(\omega[n])y_1[n-1] + \sin(\omega[n])y_2[n-1] \quad \text{and} \tag{3.4}$$

$$y_2[n] = \cos(\omega[n])y_2[n-1] - \sin(\omega[n])y_1[n-1], \tag{3.5}$$

where $y_1[n]$ and $y_2[n]$ outputs sine and cosine, respectively, with initial states $y_1[0] = 0$ and $y_2[0] = 1$. This approach is also applicable in cases when the frequency of the sinusoid

is varied via the coefficients $\sin(\omega[n])$ and $\cos(\omega[n])$. The resulting waveform does not suffer from the amplitude deviation at the instant of frequency change, since it uses only the previous waveform values [64].

The coupled form oscillator can be modified in two steps. First, when the desired frequency is small with respect to the sampling frequency, the coefficient term $\cos(\omega[n])$ is close to unity, and it can be approximated to be one. With this approximation, so called first modified coupled form oscillator is obtained, in which the amplitude of the oscillation will clip. In addition, the approximation restricts the frequency to be less than a fourth of the sampling frequency [64].

The limited applicable frequency range of the first modified coupled form oscillator can be extended by using the current value of cosine in the calculation of sine. In addition, by changing the coupling coefficient slightly the applicable frequency range spans from 0 Hz to half of the sampling frequency. This second modified coupled form is given by

$$y_1[n] = 2\sin\left(\frac{\omega[n]}{2}\right) y_2[n] + y_1[n-1] \quad \text{and} \tag{3.6}$$

$$y_2[n] = y_2[n-1] - 2\sin\left(\frac{\omega[n]}{2}\right) y_1[n-1], \tag{3.7}$$

where $y_1[n]$ and $y_2[n]$ outputs sine and cosine, respectively, with initial states $y_1[0] = 0$ and $y_2[0] = 1$. As with the direct form, the amplitude of the oscillator will adjust itself to a new value when the frequency is changed [64].

In the normalized waveguide form, the derivation of the oscillator algorithm is based on a second-order waveguide filter derived from two lossless acoustic tube sections with different cross-sectional areas and appropriate ideal terminations and in which a standing wave is formed [65]. The frequency of the oscillation is determined by the cross-sectional areas, but at change of frequency the amplitude is also changed. This can be overcome using normalization in which the state variables are updated using an additional amplitude scale factor. The normalized waveguide form is given by

$$y_1[n] = \cos(\omega[n])y_1[n-1] + G[n](1+\cos(\omega[n]))y_2[n-1] \quad \text{and} \tag{3.8}$$

$$y_2[n] = (\cos(\omega[n]) - 1)y_1[n-1] + G[n]\cos(\omega[n])y_2[n-1], \tag{3.9}$$

where $y_1[n]$ and $y_2[n]$ outputs cosine and sine, respectively, with initial states $y_1[0] = 1$ and $y_2[0] = 0$. $G[n]$ is the amplitude scale factor, given by

$$G[n] = \frac{\tan\left(\frac{\omega[n]}{2}\right)}{\tan\left(\frac{\omega[n-1]}{2}\right)},$$

which deviates from unity only at the occurrence of frequency change.

All of the these algorithms require two state variables, except for the direct form in which both sine and cosine generation require two state variables each. All algorithms are robust in terms of coefficient quantization, and only the normalized waveguide form oscillators is always stable in terms of signal quantization. The direct form oscillator may become unstable, the coupled form oscillator may decay or clip depending on the polarity of the quantization error, and the modified coupled form oscillators need built-in saturation in order to avoid clipping [64]. The quantization also introduces some noise in all approaches, but it is quite low except in the direct form oscillator, and it can be decreased by utilizing error feedback [66, 67].

Alternatively, the sinusoid can be generated using a resonant filter with maximum resonance, with which the filter starts to oscillate. Some practical filters, which can be used to implement this approach, are presented in Section 3.2.

3.1.2 Random Number Generators

The random numbers are needed in generation of noise component used for instance in source-filter synthesis. The random numbers can be generated using either amplified noise generated by a physical process, such as thermal noise of the electronic components, or with special algorithms which approximate the properties of random numbers and produce periodic pseudorandom noise [68]. All algorithmic approaches can be expressed with an iterative equation

$$r[n] = g(r[n-k]), \tag{3.10}$$

where $r[n]$ is the random number sequence, and $g(\tau)$ is the iterative function according which the random numbers are calculated [69]. The random number sequence must be initialized by choosing an initial seed $r[0]$, which then sets the random sequence together with the iterative function $g(\tau)$.

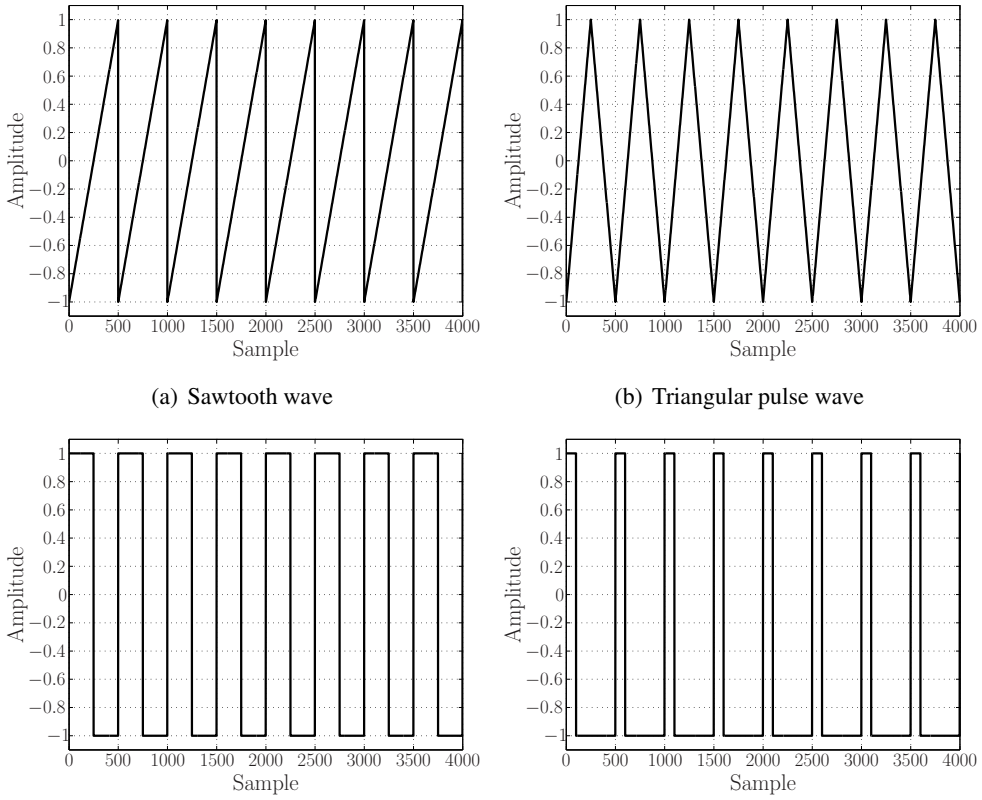
The iterative function $g(\tau)$ is often a linear function of the previous random number [69, 70] or it performs a bitwise operation to two previously outputted generated numbers [71]. In both approaches, the maximum period of the random numbers can be adjusted, and it is determined linearly on the function parameters. There exist extensions to these two fundamental approaches, such as nonlinear relation between adjacent random numbers [72] and bit twisting of the result of the bitwise operation [73, 74], and these extensions extend the maximum period of the random numbers.

All pseudorandom number generator algorithms are rather simple to implement, but their memory consumption varies. For instance, when the linear relation is used, the algorithms

requires only one multiplication and addition, and it requires only one state variable. On the other hand, the bitwise operations are extremely fast to implement, but a large output value buffer is needed in order to achieve long period. Therefore, when the memory consumption is the determining factor, the linear relation provides the most efficient solution.

3.1.3 Classic Waveform Synthesis

Any periodic waveform can be used in source-filter synthesis in order to produce a spectrum with harmonic structure. The shape of the source signal waveform determines the amplitude relations of the harmonics, and usually the source signal used is one of the spectrally rich classic waveforms generated with analog circuits. These classic waveforms are sawtooth wave, rectangular pulse wave and triangular pulse wave [75], and they are plotted in Figure 3.1.



(a) Sawtooth wave

(b) Triangular pulse wave

(c) Rectangular pulse wave, duty cycle 50%

(d) Rectangular pulse wave, duty cycle 20%

Figure 3.1: Classic waveforms used in source-filter synthesis.

Since the sawtooth and rectangular pulse waveforms contain discontinuities in their wave-

form, their spectral envelopes are approximately inversely proportional to the frequency, i.e., the amplitude of the harmonics decay 6 dB per octave. In the triangular pulse wave the discontinuity is in the derivative of the waveform, and hence the spectral envelope is approximately inversely proportional to the square of the frequency, i.e., the amplitude of the harmonics decay 12 dB per octave. Due to such low spectral tilts, the trivial sampling of the continuous time waveform causes harsh audible aliasing especially at high fundamental frequencies [76, 77, 78]. This is illustrated in Figure 3.2 with a sawtooth wave with high fundamental frequency of $0.09 \times f_s$. The harmonics below the half of the sampling frequency are indicated with markers.

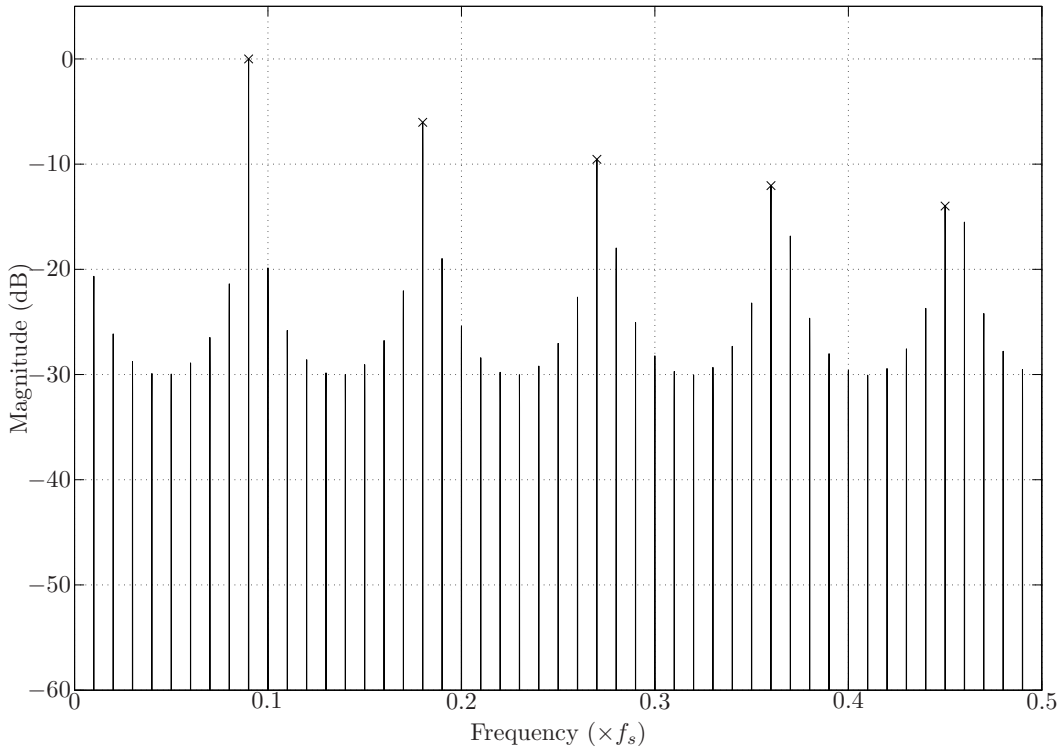


Figure 3.2: Spectrum of a trivially sampled sawtooth wave with fundamental frequency of $0.09 \times f_s$.

The aliased components in Figure 3.2 are clearly audible, and therefore the use of the classic waveforms in digital subtractive synthesis requires removal of the aliased components. This is implemented with a waveform synthesis algorithm, which should reduce the aliasing below an audible level. There are basically three approaches to this problem, which can be categorized as follows [79, 80, 81]:

1. Strictly bandlimited methods in which only the harmonics up to the half of the sam-

pling frequency are generated,

2. Quasi-bandlimited methods in which some aliasing is allowed mainly at high frequencies,

3. Alias-suppressing methods in which aliasing is allowed in the whole frequency band, but it is sufficiently suppressed at low and middle frequencies using spectral tilt modifications.

Next, these approaches are explained more thoroughly.

Strictly Bandlimited Waveform Synthesis

In strictly bandlimited waveform synthesis, only the harmonics up to the half of the sampling frequency are generated. This can be obtained by the means of additive synthesis or wavetable synthesis explained in Sections 2.2.1 and 2.1.2, respectively, or using closed form formulas which describe the bandlimited form of the waveform. In all of these approaches, the amplitudes and phases of the harmonics should remain the same as they are in the respective continuous-time waveforms.

The use of additive synthesis, or any related synthesis technique, is straightforward, since the amplitudes and phases can be generated as accurately as the implementation platform allows. The parameters are obtained from the Fourier series expansions describing the spectrum of the continuous-time waveforms [82]. However, the computational load of additive synthesis approach is inversely proportional to the fundamental frequency, i.e., the number of required oscillators increase as the fundamental frequency decreases. The computational load can be decreased by using a mixed approach of additive synthesis and trivial sampling, since at low fundamental frequencies the aliasing of trivial sampling approach is quite small. In this mixed approach, the method to be used is changed when the fundamental frequency passes a certain value. However, this requires additional bookkeeping, and therefore this approach provides practically no benefit at all.

Multirate additive synthesis technique, in which the sinusoidal oscillators are run at slow speed using decimation and interpolation operations [82], provides also better computational efficiency. However, if the fundamental frequency is time varying, the swapping of oscillators from one subband to another is inconvenience, and therefore the computational efficiency of this approach is in general not good.

Alternatively, the computational load can be decreased by reducing the number of sinusoidal oscillators using closed-form summation formulas, which describe a sum of finite

number of sine and cosine functions that are harmonically related [83, 84]. In practice, these formulas generate waveforms in which the amplitude of the sinusoidal components are not exactly the amplitudes of any classic waveform, but they can be approximated by applying appropriate filtering to the produced waveform. The computational load of this approach is less than in additive synthesis approach, but it requires a division per sample, which may lead to numerical errors especially when the denominator is close to zero.

Yet, in cases when the even harmonic components are in same or opposite phase compared to odd harmonics, the waveshaping synthesis technique presented in Section 2.3.3 may also provide savings in the computational load at high fundamental frequencies. By using Chebychev polynomials the weights of the k^{th} order polynomial is directly the amplitude of the k^{th} harmonic of the desired waveform. However, when the fundamental frequency is time-varying, the algorithm must be capable of scaling the number of harmonics to be produced, which leads to reduced efficiency.

The use of wavetable synthesis in the waveform generation is rather straightforward, since all the desired number of sinusoidal components can be calculated and stored in memory beforehand [4]. In addition, wavetable synthesis allows any waveshape to be stored, so a waveform of a certain analog synthesizer can be used. The use of wavetable synthesis reduces the computational load much, but it is traded off against high memory consumption. The memory consumption can be decreased using method presented in Section 2.1.2, or using group additive synthesis, in which the harmonics are distributed in different wavetables and the desired waveform is produced as a mix of the wavetables [85]. However, the group additive synthesis suffers from poor signal-to-noise ratio, which can be improved by reducing the peak amplitude of each wavetable [86].

The use of strictly bandlimited waveform synthesis is bound to the trade off between computational complexity and memory consumption. While the additive synthesis approach consumes only a small amount of memory, for general purpose it is computationally heavy. On the opposite, the wavetable synthesis approach is computationally extremely simple, but it consumes much memory. Therefore these approaches are practically not used in actual implementations, but they provide great support in the evaluation of other methods.

Quasi-Bandlimited Waveform Synthesis

In quasi-bandlimited waveform synthesis, some aliasing is allowed mainly at high frequencies where the human hearing is more insensitive. This is done by filtering the continuous-time waveform to attenuate the frequencies above the half of the sampling frequency to be used. The evaluation of the bandlimited waveform is not straightforward in general case,

but with geometric waveforms, such as the abovementioned classic waveforms, the evaluation can be done in closed form. The generation of the quasi-bandlimited classic waveforms can be divided into two approaches, integration of bandlimited impulses or summation of bandlimited step functions.

Bandlimited Impulse Train The approaches based on integration of bandlimited impulse train begin with the fact that the classic waveforms can be generated by applying an appropriate integration to an impulse train [87]. This can be seen from the derivative of the waveforms, or the second derivative in case of triangular pulse wave, since the discontinuities in the waveform are mapped as impulses in the derivative signal.

A bandlimited impulse train (BLIT) is obtained by lowpass filtering the continuous-time impulse train, which means that the impulses are replaced with impulse response of a lowpass filter [87]. In practice, the lowpass filter is taken to be an ideal lowpass filter with cutoff frequency f_c, which has impulse response given by

$$h_i(t) = \mathrm{sinc}(2f_c t) = \frac{\sin(2\pi f_c t)}{2\pi f_c t}. \tag{3.11}$$

Discrete time bandlimited impulse train is obtained by evaluating the bandlimited continuous-time train at time instants $\frac{n}{f_s}$, which yields

$$y[n] = \sum_{k=-\infty}^{\infty} \mathrm{sinc}\left(2\frac{f_c}{f_s}(n - kP)\right), \tag{3.12}$$

where P is the period in samples.

From Equation (3.12) can be seen, that the value of BLIT at each time instant requires summation of infinitely many sinc-function values, which is impossible in practice. In order to implement this, the summation must be expressed in some other form. An efficient approach is obtained by summing windowed sinc-functions (BLIT-SWS), in which the bandlimited impulse train is generated by summing the values of windowed sinc-functions, and from which the desired waveform is integrated [87].

However, the windowing causes the transition band of the filter to spread, which in turn causes errors in the bandlimitation of the impulse train. If the cutoff frequency is close to the half of the sampling frequency, the windowing causes some aliasing, which depends on the window type and its length. In addition, the amplitudes of the harmonics near the cutoff frequency will not retain exactly the amplitude values of the continuous-time waveform [87].

In addition, since the windowed sinc-functions have non-zero samples before the actual impulse instant, the algorithm must start outputting the appropriate values before the impulse

instant. This leads to additional control logic, implementation of which can be in general case computationally heavy. However, if the number of windowed sinc-functions to be integrated is limited, the control logic can be implemented more efficiently. However, the limitation restricts the highest possible fundamental frequency the system can produce without decreasing the alias reduction performance. This limit for the fundamental frequency is set by the window length and the number of windowed sinc-functions in use [87].

Bandlimited Step Function The methods based on summing of bandlimited step functions (BLEP) can be understood as an extension to BLIT. While in BLIT the waveform synthesis is based on integration of an impulse train, in BLEP the amplitude changes at the discontinuities are generated with bandlimited step functions. A bandlimited step function is obtained from a bandlimited impulse by integration and scaling the step function to saturate to unity. In other words, in BLEP the integration of the impulse train is performed in advance as compared to BLIT, and the resulting bandlimited step functions are then summed appropriately in order to produce the desired waveform [88].

Yet again in the ideal case, the resulting sum adds up an infinite number of integrated sinc-functions, which is not implementable. Integrated windowed sinc-functions can be use, and that approach sets same requirements to the control logic as the BLIT-SWS method. However, the look ahead of the control logic can be easily reduced in BLEP by performing phase reduction to the bandlimited impulse. The phase reduction changes the phase response of the impulse, but it retains the amplitude response. The phase reduction is performed for instance in Minimum-Phase BLEP (MINBLEP), where the integral of a minimum phase windowed sinc-function is used as the source signal for the waveform summation unit [88].

The look ahead in MINBLEP is reduced greatly compared to the approach without phase reduction [88], but it is not eliminated completely [81]. The look ahead would be eliminated completely only if a zero phase sinc-function is integrated, but the implementation of a zero-phase sinc-function is in general hard, since the algorithm should then require information of the signal period. However, the look ahead is only a few samples when the minimum phase sinc-function is used, and therefore it is usually neglected. This causes a slight delay in the resulting waveform, but since phase differences are usually inaudible, the minimum phase function practically eliminates the look ahead. In addition, the use of minimum phase function is more practical, since the control logic can be implemented more efficiently, since it does not require the information of the signal period.

Both the BLIT-SWS and the MINBLEP approaches reduce the aliasing greatly compared to the trivial sampling approach even when short windows are used, and the remaining

aliased components are almost inaudible. However, in BLIT-SWS approach the resulting waveform has quite much aliasing also at low frequencies. This should be noted, since the human hearing processes the frequencies of the audio band in logarithmic manner so that 1000 Hz is approximately in the middle of the perceived audio band. Therefore the aliased components at low frequencies are more clearly audible, and MINBLEP approach produces better results than BLIT-SWS. This is due to the integration before the sampling, which implements inherent 6 dB per octave lowpass filtering.

Despite good alias reduction performance, the implementation of both BLIT-SWS and MINBLEP approaches requires memory in which the bandlimited signals are stored. The memory consumption can be decreased by using shorter window, but it trades off with alias reduction performance. Therefore the use of these approaches requires a reasonable balance between the alias reduction performance and memory consumption. However, the memory requirement can be eliminated completely by evaluating the bandlimited transitions with a polynomial. This approach is called Polynomial Bandlimited Step function (POLYBLEP), in which the bandlimited impulse is first approximated using closed-form formulas, which is then integrated and used as the bandlimited step function [81].

Polynomial Bandlimited Step Function approximation The transitions at the discontinuity of a classic waveform are usually positioned between two sampling instants. In quasi-bandlimited waveform synthesis approaches, the neighbouring sample values of the transitions are practically manipulated in order to obtain the bandlimitation effect. In POLYBLEP, the fundamental idea is to perform the manipulation with piecewise continuous polynomials, with which the required residual, i.e., the difference from the trivially sampled waveform, can be calculated in closed form. Next, the requirements for the polynomials are given, and design rules for applicable polynomials are presented.

POLYBLEP is formulated in interval $t \in [-N, N]$ by piecewise continuous polynomials. Outside that interval the overall polynomial, $p(t)$, gets the value zero. The piecewise polynomials are designed to approximate the bandlimited impulse, i.e., the sinc function of Equation (3.11). This leads to the following criteria, which are used in the polynomial design:

1. $p(-t) = p(t)$,
2. $\max p(t) = p(0) = 1$.

The desired formulas for POLYBLEP residual calculation are obtained by integrating $p(t)$ and subtracting step function from the result.

In general, the polynomials can be of any form, but in order to achieve the best alias reduction performance, the polynomials should form as smoothly continuous integrated $p(t)$ as possible. The performance is reduced by discontinuities of $p(t)$, as a discontinuity at n^{th} derivative of $p(t)$ cause the aliased components decay $12 + 6n$ dB per octave. However, the decay of 18 dB per octave produces quite good performance, so the polynomials can be quite simple. In practice, the interval in which the polynomials are defined is wise to span few sampling instants, and the polynomials are defined in between the sampling instants. These two notes lead to computationally more efficient implementation of the algorithm.

The computationally simplest approach is to use one low order polynomial to alter the sample values before and after the transition, i.e., $N = 1/f_s$. In addition, the implementation of the PolyBLEP algorithm becomes more efficient when $f_c = f_s/2$ in Equation (3.11). When the polynomial is designed, one should remember that a n^{th} order polynomial requires $n + 1$ criteria, which define the coefficients of the polynomial. Since the polynomial values at zero and at the interval end point are given (see the overall design criteria above), the choice of the rest $n - 1$ criteria becomes a design issue of its own. This is illustrated with two third-order polynomials, which are based on spline and Lagrange interpolations.

In the third-order spline interpolation, all criteria for the polynomial are given at the end points of its definition interval. At the end points, the spline polynomial has the function values and the derivative of the polynomial has the derivative values of the function. Since the function to be approximated is the sinc function, the derivative values are given by

$$\frac{\mathrm{d}}{\mathrm{d}x}\mathrm{sinc}(x) = \frac{1}{x}(\cos(\pi x) - \mathrm{sinc}(x)),$$

which is zero at the origin. With these criteria, the overall polynomial is given by

$$p_{sp}(t) = \begin{cases} -t^3 - 2t^2 + 1, & -1 \leq t \leq 0, \\ t^3 - 2t^2 + 1, & 0 < t \leq 1, \\ 0, & \text{elsewhere}, \end{cases} \tag{3.13}$$

from which the polynomial bandlimited step function is obtained by integration, given by

$$P_{sp}(t) = \begin{cases} 0, & t < -1 \\ -\frac{3}{14}t^4 - \frac{4}{7}t^3 + \frac{6}{7}t + \frac{1}{2}, & -1 \leq t \leq 0, \\ \frac{3}{14}t^4 - \frac{4}{7}t^3 + \frac{6}{7}t + \frac{1}{2}, & 0 < t \leq 1, \\ 1, & t > 1, \end{cases} \tag{3.14}$$

where constants are due to continuity. By subtracting the unit step function, given by

$$\mu(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}$$

the POLYBLEP residual is obtained,

$$r_{sp}(t) = \begin{cases} -\frac{3}{14}t^4 - \frac{4}{7}t^3 + \frac{6}{7}t + \frac{1}{2}, & -1 \leq t \leq 0, \\ \frac{3}{14}t^4 - \frac{4}{7}t^3 + \frac{6}{7}t - \frac{1}{2}, & 0 < t \leq 1. \end{cases} \tag{3.15}$$

In Figure 3.3, the resulting polynomial is compared to the sinc function.

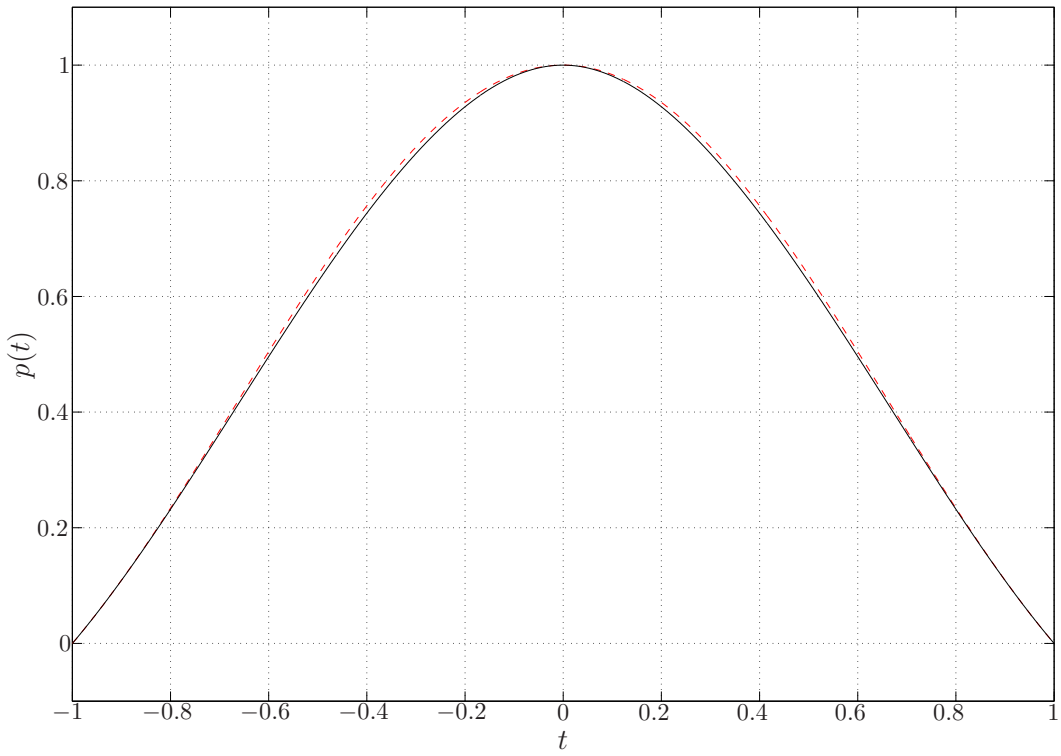


Figure 3.3: POLYBLEP approximation of sinc function (dashed line) with third-order spline (solid line).

The polynomial approximation can also be based on Lagrange interpolation polynomials. The practical Lagrange polynomials are of odd order, which yields more accurate approximation of the sinc function. In addition, if only the interval $[0, 1/f_s]$ is approximated, the desired polynomial is one of the polynomials obtained for the interpolation. For instance, the third-order Lagrange interpolation polynomial, which is applicable in POLYBLEP is given by

$$p_{l3}(t) = \begin{cases} -\frac{1}{2}t^3 - t^2 + \frac{1}{2}t + 1, & -1 \leq t \leq 0 \\ \frac{1}{2}t^3 - t^2 - \frac{1}{2}t + 1, & 0 < t \leq 1 \\ 0, & \text{elsewhere,} \end{cases} \tag{3.16}$$

which yields the polynomial bandlimited step function

$$P_{l3}(t) = \begin{cases} 0, & t < -1 \\ -\frac{3}{26}t^4 - \frac{4}{13}t^3 + \frac{3}{13}t^2 + \frac{24}{26}t + \frac{1}{2}, & -1 \leq t \leq 0 \\ \frac{3}{26}t^4 - \frac{4}{13}t^3 - \frac{3}{13}t^2 + \frac{24}{26}t + \frac{1}{2}, & 0 < t \leq 1 \\ 1, & t > 1. \end{cases} \tag{3.17}$$

The POLYBLEP residual is given by

$$r_{l3}(t) = \begin{cases} -\frac{3}{26}t^4 - \frac{4}{13}t^3 + \frac{3}{13}t^2 + \frac{24}{26}t + \frac{1}{2}, & -1 \leq t \leq 0 \\ \frac{3}{26}t^4 - \frac{4}{13}t^3 - \frac{3}{13}t^2 + \frac{24}{26}t - \frac{1}{2}, & 0 < t \leq 1. \end{cases} \tag{3.18}$$

In Figure 3.4, the resulting polynomial is compared to the sinc function.

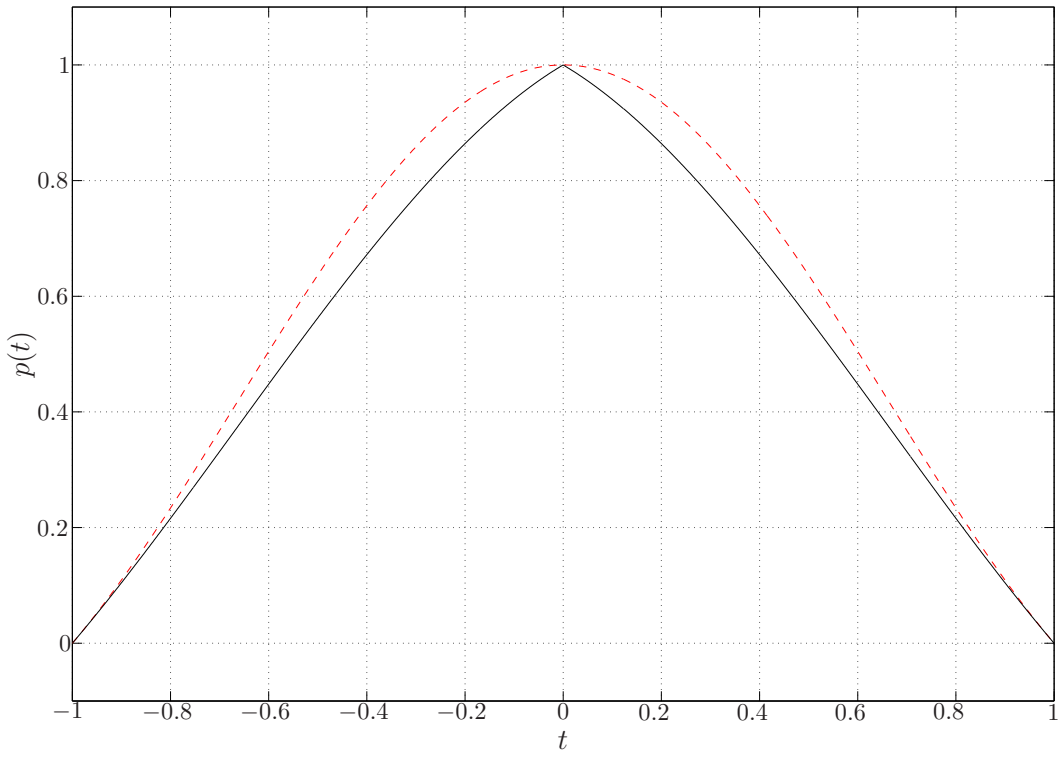


Figure 3.4: POLYBLEP approximation of sinc function (dashed line) with third-order Lagrange interpolation polynomial (solid line).

In Figure 3.5, comparison of the bandlimiting performance between the presented third-order polynomials and the first-order polynomial is presented with the magnitudes of the harmonics indicated with markers. The fundamental frequency used was $0.057 \times f_s$. As from the figure can be seen, the third-order polynomials decrease the aliasing slightly more at high frequencies compared to the first-order polynomial. However, they have more aliasing at low frequencies, where the human hearing is more sensitive. Yet, the remaining

aliased components are almost inaudible, so the use of these higher order polynomials require careful design. In addition, the use of more than two neighbouring samples may lead to better results.

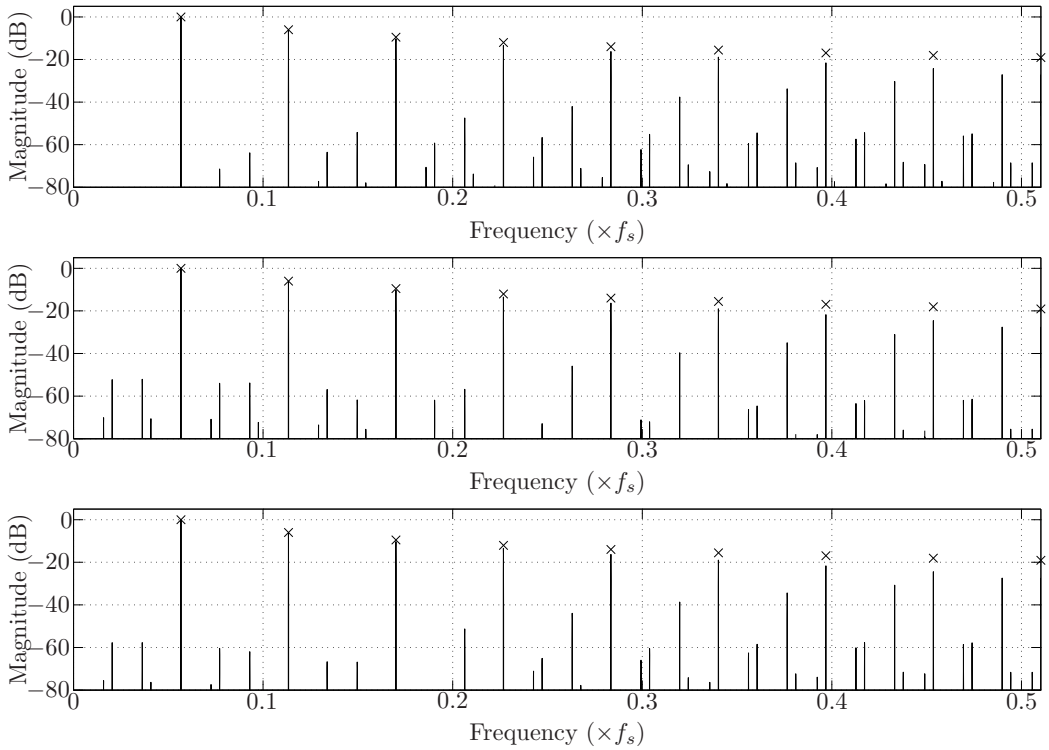


Figure 3.5: Comparison of the bandlimiting performance between first-order polynomial (top pane), third-order spline interpolation polynomial (middle pane), and third-order Lagrange interpolation polynomial (bottom pane).

Alias-Suppressing Waveform Synthesis

In alias-suppressing waveform synthesis, some aliasing is allowed in the whole frequency band, but it is sufficiently suppressed at low and middle frequencies. The alias-suppressing is obtained by rendering the slope of the spectrum to be steep before sampling and restoring the spectral tilt with a digital filter. The spectral tilt is modified by applying a exponential decay function to the harmonic amplitudes. However, all components above the half of the sampling frequency are reflected to the audio band, but their amplitudes are reduced with respect to the original amplitudes.

The simplest alias-suppressing approach is to use trivial sampling approach with quite high oversampling [25]. The oversampling reduces the aliasing at the audio band, since the

aliased components are reflected around higher frequency. However, in order to meet certain specifications of allowed aliasing, the oversampling factor must be large when the desired fundamental frequency is high. In addition, the alias rejection filters needed for the re-sampling to the desired sampling frequency should meet quite strict specifications, which leads to computationally inefficient implementation. Therefore the oversampling approach is not practical in computationally efficient music synthesis.

The spectral tilt modification can be done by filtering a heavily distorted sinusoid. For instance, the distortion can be obtained by full-wave rectifying a sine wave. The distortion adds harmonic components to the signal, and by filtering the amplitudes of the harmonics of the classic waveforms can be approximated. However, the filter must be capable of correct both the spectral tilt and suppress the aliased components. These are obtained by first-order infinite impulse response filter and any desired lowpass filter, respectively [89]. However, the lowpass filtering suppresses also the higher harmonics of the signal, which leads to only a rough approximation of the desired waveform.

One interesting and efficient approach in synthesis of the sawtooth wave is obtained by differentiating a piecewise parabolic waveform. In this approach, Differentiated Parabolic Waveform (DPW), the piecewise parabolic waveform is obtained by squaring the bipolar trivially sampled sawtooth waveform. When this parabolic waveform is differentiated, a waveform similar to the sawtooth wave is obtained which differentiates from the original only by one sample in each period. However, the amplitude of the resulting waveform depends on the fundamental frequency, and therefore the waveform must be scaled in order to generate a waveform with normalized amplitude. This scaling factor requires a division, which makes the implementation of this approach slightly tricky. In addition, this method may produce beating when the fundamental frequency is quite high, since the amplitudes of the first aliased components are not greatly decreased. This can be overcome by implementing a multirate version, where with decimation filter the amplitudes of the highest aliased components can be reduced [90]. The rectangular pulse wave can be generated with DPW by using two sawtooth waves with appropriate phase shift [80] or by filtering one sawtooth wave with a FIR comb filter [91]. The triangular pulse wave can be generated with slight modifications to the basic algorithm of DPW [80].

3.2 Filters

As concluded in Section 2.5, some of the practical synthesis techniques require filters with which the source signal is modified in order to produce the desired sound. The design of these filters differ from the traditional digital filters which are designed to fulfill given pass-

band and stopband specifications. The applicable filters are designed to provide easy control over the desired cutoff frequencies and resonances. Especially the filters used in classical analog synthesizers have been under intensive research due to their sonically pleasing characteristics. In Section 3.2.1, the design issues of musical resonant filters are presented, and some simple practical filter designs are also given. Section 3.2.2 presents the model of one classical analog synthesizer filter, the Moog ladder filter.

3.2.1 Musical Resonant Filters

The filters used in digital subtractive synthesis differ from the traditional filters used in digital signal processing in mainly three ways. First, the requirement for certain passband and stopband specifications is released, and the order of the filter is predetermined. Secondly, the filter is quickly time-varying, i.e., the filter parameters are changed at rapid rate. Thirdly, the applicable filter has a controllable resonant peak near its cutoff frequency, which is usually not desirable in digital filter design [79, 80].

These applicable musical resonant filters are usually of second or fourth order, and they are designed based the following criteria [79, 80]:

1. The coefficients of the filter should be updated fast, preferably on per-sample basis [92].
2. The cutoff frequency and the resonance controls should be separate, change in one should not affect the other.
3. When the parameters are within allowed ranges, the filter should be unconditionally stable.
4. With a certain resonance parameter, the filter should be capable of self-oscillating.
5. The filter should imitate the response of an existing analog resonant filter, and the characteristics of the analog filter should be emulated if possible.

The most obvious approach is to replace the components of the analog filter by their digital counterparts, which leads to digital state variable filter. An example of the result of this approach is given in Figure 3.6 [25]. With this filter, different kinds of filtering operations can be performed separately, as it outputs lowpass filtered input signal $x[n]$ with $y_1[n]$, highpass filtered with $y_2[n]$, bandpass filtered $y_3[n]$, and notch filtered $y_4[n]$. The mapping between the filter parameters and coefficients is simple, and they are given by

$$g = \frac{1}{Q} \quad \text{and} \quad f = 2\sin\left(\frac{\pi f_c}{f_s}\right). \tag{3.19}$$

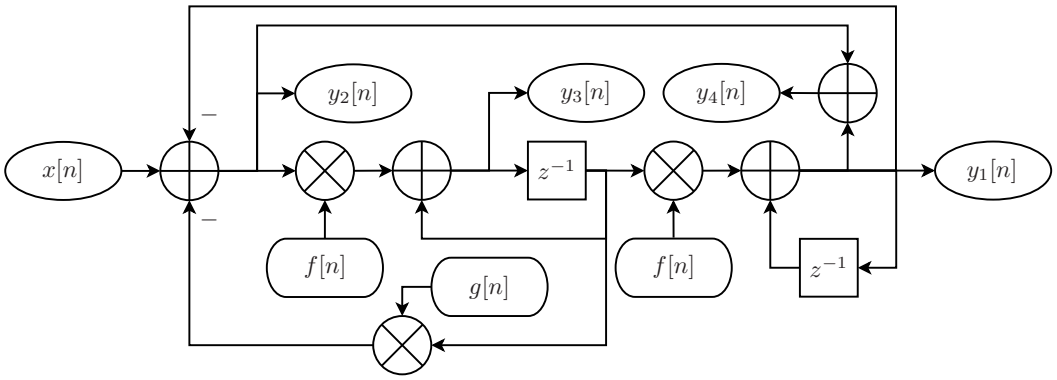


Figure 3.6: Flow diagram of a state variable filter, after [25].

The resonant filter design can be based on fact that the radius of a pole sets the height of the peak in filter magnitude response. In addition, the angle of the pole in turn sets the frequency of the peak [93]. Using appropriate approximations, a second-order all-pole resonator with peak frequency f_p and -3 dB bandwidth Δf is obtained,

$$H(z) = \frac{A_0}{1 - 2R\cos(\theta)z^{-1} + R^2 z^{-2}}, \tag{3.20}$$

where A_0 is the gain compensation factor,

$$R = 1 - \pi\frac{\Delta f}{f_s} \qquad \text{and}$$
$$\cos(\theta) = \frac{2R}{1 + R^2}\cos\left(2\pi\frac{f_p}{f_s}\right).$$

However, the bandwidth of this resonator becomes wider when the peak frequency is low. This can be compensated using an additional zero at zero frequency [94], when the filter becomes a bandpass filter.

Alternatively, the filter design can be based on digitization of an analog resonant filter by means of transformation from analog to digital domain. Different transformations provide different results, and an example of their use is given in Section 3.2.2 for the Moog Ladder filter.

3.2.2 Moog Ladder Filter

One of the most influential filters in the history of electronic music is the voltage-controlled filter designed by Robert Moog [95, 96]. This filter was constructed using a transistor ladder circuit, which consisted of four stages of a transistor pair and a capacitor. Each stage formed a nonlinear one-pole lowpass filter, and the resonance of the filter was controlled with a global negative feedback.

The discretization of the Moog Ladder filter with traditional transform techniques from analog to digital domain lead to delay-free loop, which requires insertion of a delay into the feedback loop in order to be implementable. Unfortunately, this leads to coupled control of cutoff frequency and resonance. This can be overcome by introducing an additional separation table, with which the parameters can be decoupled. Alternatively, the controls can almost be decoupled by introducing a zero at $z = -0.3$ for each lowpass stage. With this approach, the controls are separate at frequency range $[0, f_s/4]$ [97]. However, the cutoff frequency will then deviate from the desired value, and must be therefore compensated [80].

The nonlinearity of the filter has been modeled by analyzing one lowpass filter stage. It has turned out, that the nonlinearity can be implemented by using the $\tanh$-function. Each filter uses as input the $\tanh$ of the output of the previous stage, and this value is used by the previous stage in calculation of next sample [98]. This leads to five $\tanh$ calculations per sample, which may sometimes be computationally too heavy. In addition, oversampling by factor of at least two is required. The nonlinearity of each stage can be approximated with only one global nonlinearity, which decreases the computational load of the nonlinearity implementation [79, 80].

The Moog filter can be used as lowpass, highpass, bandpass and notch filter by mixing the outputs of each filter stage appropriately. In addition, the passband gain drop of the filter can be compensated by subtracting a fraction of the input signal from the feedback signal before the resonance parameter multiplier [79, 80].

3.3 Control Signals

Since many real world sound exhibit time-varying characteristics, the applicable sound synthesis techniques concluded in Section 2.5 will be inadequate without the possibility of time-varying implementation. In order to realize these time-varying characteristics, the sound synthesis techniques require additional control signals, with which the time-varying behavior of the real world counterparts can be modeled. The control signals can be categorized as envelopes, with which the continuous and smooth fluctuations of sound characteristics can be described, and low frequency oscillation, which in turn describe the repetitive fluctuations. The issues of envelope signal design are addressed in Section 3.3.1, and the use of low frequency oscillation is discussed in Section 3.3.2.

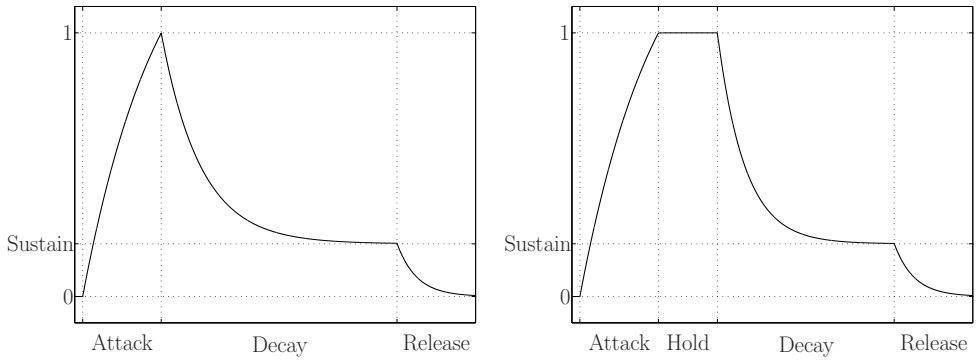
3.3.1 Envelope Generators

In numerous cases, the sound produced by an acoustical instrument contains continuously and smoothly time-varying sound characteristics, which make the timbre of the instrument unique and quite easily recognizable. In order to synthesize these instruments using the synthesis techniques practical for computationally efficient music synthesis, the time-varying behavior must be modeled somehow. This is done with envelope generators, which formulate the model of the characteristic fluctuations.

The simplest approach for modeling transitions from one value to another of the fluctuation is obtained with piecewise linear envelope [77, 78]. A piecewise linear envelope is computationally easy to implement, and the design problem is then to determine the control points between which the linear approximations are made. If these control points are placed densely, the model of the fluctuation becomes more accurate. However, the amount of control data is then increased, which increases the requirement for memory usage for the instrument definition data. Therefore, the balance between model accuracy and memory consumption must be carefully designed when the piecewise linear envelope is used.

In nature, almost all physical transitions between two stages exhibit exponential behavior, a practical approach is obtained by approximating the transitions with an envelope which contains exponential growths and decays [78]. In addition, the exponential behavior can be found also from the first sound synthesizers, which were constructed using analog electronics. This exponential envelope approach is also rather easy to implement, and control data requires also the information of the speed of each transition. However, the amount of control data is reduced greatly compared to the piecewise linear envelope approach, since the same accuracy can be obtained with more sparse control point positioning. Therefore, the exponential envelope approximation is better both in terms of accuracy and memory consumption.

In general, the fluctuations can be as complex as possible, when the modeling of them becomes a synthesis problem of its own. However, the fluctuations are usually quite simple, including only transitions from one value to another, which simplifies the modeling problem quite much. In such cases, the envelope generator can be designed to model transitions between values in range from zero to one, and by appropriate scaling any transitions can be modeled. Quite often the fluctuations can be decomposed into attack, decay, sustain, and release phases. Sometimes the fluctuation has after the attack phase a hold phase, during which the envelope holds its value. The respective envelopes are called Attack-Decay-Sustain-Release envelope (ADSR) and Attack-Hold-Decay-Sustain-Release envelope (AHDSR), and they are illustrated in Figure 3.7 using exponential approximation.



(a) ADSR (b) AHDSR

Figure 3.7: Commonly used envelope functions.

In addition, the modeling of continuous fluctuations may not always be possible using only one envelope generator. In such cases the overall envelope function can be composed of two or more envelopes. The decomposition of the overall envelope function may be a difficult task, and the composite envelopes are usually obtained by trial. However, if a similar overall envelope function is obtained form the analysis of various instruments, it would be good idea to implement a stand-alone envelope generator for that envelope function. Yet, one should carefully test the computational load of the generation of this new envelope function, and if its implementation is inefficient, a less accurate approximation should then be used.

3.3.2 Low Frequency Oscillators

The fluctuations of sound characteristics can also be repetitive, and the approximation of such fluctuations is computationally inefficient using the envelope functions discussed in Section 3.3.1. However, a highly efficient approach is to model the repetitive fluctuations with oscillators, which can be mixed with envelope generators in order to model the fluctuation in whole. Now, the design problem is to determine the type and the characteristics of the oscillator. The frequency of the oscillator is usually quite small, less than 20 Hz, a notion that leads to term Low Frequency Oscillator (LFO). The frequency variations of a LFO are usually extremely small, and therefore usually omitted, and therefore the amplitude of the oscillation is left to be determined.

The cycles found in nature are quite often sinusoidal, and therefore the sinusoidal LFOs are most commonly used in different sound synthesizers. However, any waveform can be in general be used, and in electronic instruments and synthetic sounds other waveforms are sometimes used. In addition, in many daily used electronic devices, such as telephones, the

produced sound may contain a LFO with, e.g., the rectangular pulse wave. Again, the overall LFO may be composed of several different oscillators. Therefore, the decomposition of the LFO can be performed using the sound analysis tools presented in Chapter 2.

3.4 Effects

The sound produced by real instruments is sometimes considered quite dull and dry. Therefore the sound is usually processed with additional sound effects, which brings liveliness to the plain instrument sound. There are numerous different effects designed for creating different kinds of expressions. However, the most commonly used effects are Chorus, Flanger, Phaser, Reverb, and Distortion. The design of Chorus, Flanger and Phaser effects is presented in Section 3.4.1. In Section 3.4.2, different algorithms for Reverb effect are presented. The use of Distortion effect is discussed in Section 3.4.3.

3.4.1 Chorus, Flanger and Phaser

With Chorus, an illusion of multiple simultaneous sounds is attempted to create [99]. The its simplest it means doubling, where a copy of the sound signal is added to the original. This can be repeated multiple times, with which a more realistic Chorus effect is obtained. In order to create the illusion of two or more sound playing at the same time, the sounds should not be exactly in synchrony. This leads to adding the copies with slightly delayed, which is practically the same as filtering the original with a set of FIR comb filters. If the delay is slowly time-varied, a more realistic sounding doubling effect is obtained. In addition, with the modulated delay the multiple parallel doubling units can be approximated by feeding back a fraction of the delayed signal, which reduces the number of needed state variables.

The Flanger was originally implemented using two tape players playing the same song in synchrony, and the flange of the other player was pressed in order to obtain a delayed version of the sound signal [99]. Therefore, the Flanger is essentially similar to doubling, but the interaction between the flange press and the tape player made the delay to be time-varying. This leads to a filter structure similar to the Chorus, and these two effects are usually implemented with the same filter by changing the filter coefficients.

The Phaser effect was created when the Flanger effect was tried to be imitated with electronic circuits. The notches of the filter resulting from the delay line were created using analog allpass filters, which lead to a slightly different sounding effect. In addition, the notches were usually spaced logarithmically in frequency, while in the Flanger effect they

are equally spaced. The digital Phaser is then implemented by using second-order allpass filters in cascade, and with each filter one notch is created [100].

Despite the computationally efficient filter structure, the implementation of the Chorus and Flanger effects require a rather long delay line, which becomes a problem in memory limited systems. On the other hand, the Phaser effect is slightly more complex in terms of number of operations, and the desired number of notches determine the number of required state variables. Therefore these effects are not used in systems with limited memory capacity.

3.4.2 Reverb

With Reverb, the interaction between the produced sound and the surrounding space is emulated. The room colors the spectrum of the sound by adding delayed versions of the sound that are reflected from the walls and object found in the space. The reflected sounds cause comb filtering, but since there are numerous almost simultaneous reflections coming from all directions, the resulting filter is more or less like an allpass filter. However, the impulse response of the space usually contains early and late reflections, from which the first ones are louder and rather sparsely spaced in time, while the latter ones are more quiet and really densely spaced. With this information, the Reverb algorithm can be simplified.

The Reverb algorithms can be divided into two categories, physical and perceptual [101, 102]. The algorithms of physical approach try to simulate exactly the propagation of the sound from the source to the listener. This can be done by filtering the sound with the binaural impulse response of the space, or by rendering the reflections from the walls and objects when the space does not exist. The binaural impulse responses are always measured from one point to a certain listening position with certain head positioning, and they are exactly applicable only for the measurement person, so they are not practically used. The latter method is computationally heavy, requiring several days of calculation even with a modern highly efficient computer. Therefore, also this approach is not applicable for computationally efficient purposes.

The perceptual Reverb algorithms try to model the room with a rather small set of parameters, which represent the space with perceptually relevant accuracy [101]. These algorithms are much simpler than those of physical approach, and ideally with one algorithm all spaces can be simulated. Usually the perceptual algorithms contain a set of interconnected comb and allpass filters, with which the reverberation is modeled [101, 102]. Since the reverberation is a complex structure, generation of close model of it requires large number of these elements. Therefore, the practical models for computationally efficient purposes contain

only one or two comb or allpass filters, which require only a short delay line. With these filters, the resulting effect sounds clearly artificial.

3.4.3 Distortion

Distortion is an important effect in some music styles, especially in guitar playing. The Distortion is obtained by processing the dry sound with a nonlinear process, in which new signal components are generated [102]. This is implemented by the same means as the waveshaping synthesis explained in Section 2.3.3, but now the input is a broadband signal produced by the instrument. This means that the problems of waveshaping synthesis are also found in Distortion generation.

First, the nonlinear function according to which the signal is distorted is usually smooth curve, which releases the problem of aliasing caused by the nonlinearity. Distortion that is most commonly rated the most pleasing is obtained from a guitar amplifier built with electronic tubes [102]. Therefore, the ideal function would simulate the distortion produced by an electronic tube. The nonlinear curve can be approximated with some trigonometric function such as $\tanh$, or with a polynomial.

Secondly, the broadband signal used as an input may cause aliasing, which causes the distorted signal sound like nothing like desired. The simplest solution for alias reduction is to perform the waveform shaping using a higher sampling frequency, and decimate back after the operation [102]. However, this increases the computational load of the algorithm, which may be not desirable. Therefore, in order to decrease to computational load, the algorithm should first lowpass filter the input signal, and perform the nonlinear shaping after the filtering [102].

Chapter 4

Sound Design

In this chapter, design rules for creating sound with different timbres are presented. In addition, the effect of the synthesis parameter variations in producing sound variants are presented. In Section 4.1, the use of computationally efficient sound synthesis techniques in imitation of different sounds are presented. Section 4.2 presents design rules for the sound variants, and their mapping to the parameters of the sound synthesis techniques generating the respective fundamental timbre is given.

4.1 Design Rules for Imitation of Different Timbres

The use of the sound synthesis techniques in music synthesis requires sound design, with which different timbres are obtained. The design of a sound differs from one technique to another, and the applicability of the techniques is ultimately determined by the sound to be imitated. Therefore, the imitation of a timbre can be implemented efficiently with a certain technique, while with the others it may be inefficient. Next, some practical design rules for different timbre categories are given.

In Section 4.1.1, design principles for creating timbres of string instruments are given. Section 4.1.2 presents design rules for imitation of pipe, brass and reed instruments. In Section 4.1.3, design principles for imitation of organ, mallet and bell sounds are given. Section 4.1.4 addressed the problem of creating electronic and synthetic sounds. In Section 4.1.5, design rules for drums and other percussion instruments are presented. Section 4.1.6 addresses the problem of designing sound effects.

4.1.1 String Instruments

The vibration of the string causes the spectrum of a string instrument have a harmonic structure, and the spectral envelope is determined by the string excitation, string material, and the instrument soundbox, which amplifies the sound generated by the string vibration. The spectral envelopes make the instrument sounds unique and quite easily separable. Therefore, the most prominent solution to the synthesis of string instruments is source-filter synthesis, in which harmonic spectrum is easily generated using periodic waveforms and the desired spectral envelope is easily controlled with the filter. The focus in the sound design of string instruments is in generating the desired spectral envelope.

The timbres of stringed instruments may be quite complex, and they often exhibit modulation either in frequency or amplitude, or in some cases in both. Good examples of such instruments are the violin and the acoustic piano. The violin timbre exhibits frequency modulation caused by the interaction between the bow and the string. Each acoustic piano tone is generated with two or three strings that are slightly mistuned with respect to each other, which leads to amplitude modulation of each harmonic separately. Therefore, the sound design of string instruments should also consider these instrument specific aspects.

4.1.2 Pipe, Brass and Reed Instruments

The sound production mechanism in pipe, brass and reed instruments is based on standing wave in a tube, and the standing wave has spectrum in which includes only odd harmonics. Therefore the source-filter synthesis is again the most prominent solution, and the design of these instruments is again focused on finding the spectral envelope of the instrument. The source signal can be chosen from all classical waveforms, with a rectangular pulse wave having a duty cycle of 50%, in order to obtain the desired spectral tilt.

However, in many of these instruments the sound includes much modulation due to turbulent airflow, and therefore the sound design requires the use of FM synthesis. In some cases, the frequency modulation is pronounced, and it would be better to use only FM synthesis in the sound generation. The parameter analysis of complex FM is quite hard, but when the correct parameters are obtained, the resulting sound imitates the desired one closely.

4.1.3 Organs, Mallets, and Bells

Organs, mallets and bells are good examples of instruments, which have simple spectra containing only a few components. Therefore the synthesis of these instruments is easily done by means of additive synthesis, and the design of these sounds is quite straightforward.

This is illustrated next with a marimba tone.

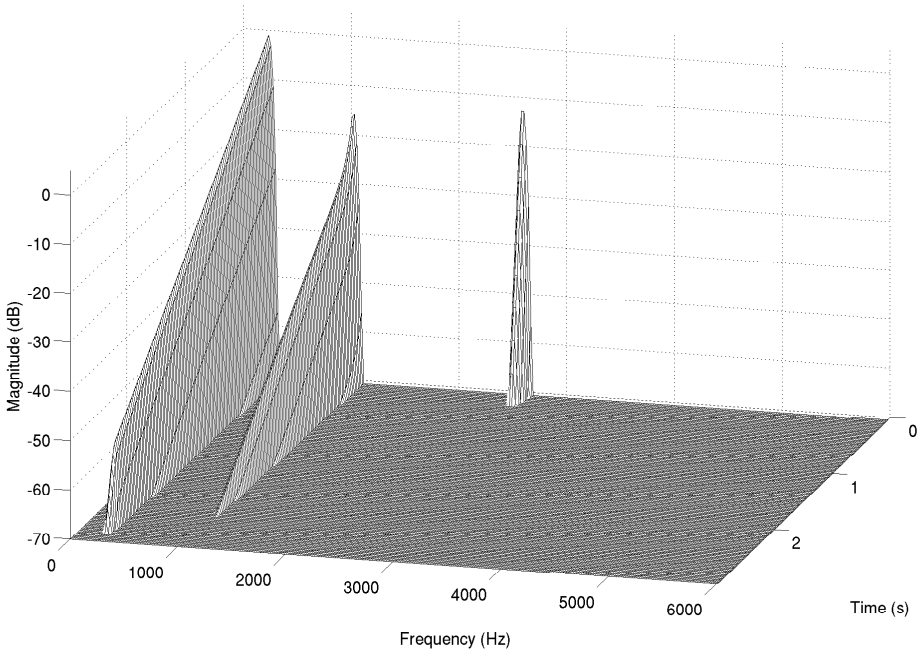


Figure 4.1: Spectrogram of a recorded marimba tone.

In Figure 4.1, the spectrogram of a marimba tone C5 is given. As from the figure can be seen, the tone has only three components, which are quite sparsely positioned. The two higher components are four and ten times higher than the fundamental frequency. The components have different weights in the resulting timbre, and the weights for the two higher components are tone dependent, as the higher pitched tone exhibits smaller weights. The fundamental frequency is the strongest, the highest component is the second strongest.

Each component has own amplitude envelope, which are piecewise linear on the dB scale. The amplitude envelopes of the higher components have a short attack, while the attack of the fundamental frequency is much longer. The fundamental frequency decays quite slow, and the highest component decays quickly. The second component has an envelope that is composed of two parts, slowly and fast decaying. The slowly decaying envelope is the same as the envelope of the fundamental frequency, and the fast decaying is slightly slower decaying than the envelope of the highest component. In addition, the decay rates for the two higher frequencies are also tone dependent, as a higher pitched tone exhibits faster decays.

4.1.4 Electronic Instruments and Synthetic Sounds

Electronic instruments and synthesis sounds form a heterogeneous group, where the sound production have been traditionally obtained by the synthesis technique used in the synthesizer. Mainly the technique is either sampling, wavetable, additive, subtractive or FM synthesis, but there are also synthesizers which exploit physical modeling. However, the most familiar electronic and synthetic sounds are generated using either subtractive or FM synthesis, due to the huge success of synthesizers exploiting these techniques. Therefore, no general sound design rules can be given for electronic instrument and synthetic sound imitation.

4.1.5 Percussion Instruments

Percussion instruments are as a group similar to electronic instruments and synthetic sounds as they produce a wide variety of sounds. The sounds range from simple clicks of wooden sticks to extremely complex cymbal sounds. Therefore, the sound design of these instruments can not be generalized into use of one synthesis technique. However, quite often the sounds have a noise-like spectrum with few additional sinusoidal components, and in these cases the design is based on finding the appropriate noise coloring filter and the parameters for the sinusoidal components. This is illustrated with a low tom tone.

In Figure 4.2, the spectrogram of a low tom tone is given. As from the figure can be seen, the tone is noise-like, and it contains few sinusoidal components which are low in frequency. The sinusoidal components are harmonically related, and the spectral tilt is steep. This can be obtained by using a triangular pulse wave. The noise shape of this tone can be obtained with a lowpass filter, the cutoff frequency of which is approximately twice the frequency of the triangular pulse wave.

The spectrum has a sharp resonance, and the higher frequencies are decaying faster than the lower frequencies. The decay characteristics can be obtained with an envelope, which is quickly decaying. In addition, the fundamental frequency of the triangular pulse wave has similar variation, so the frequency modulation is the same in both filter cutoff frequency and fundamental frequency. The overall amplitude of the tone has fast attack, and it is quickly decaying, which is typical for drum tones.

4.1.6 Sound Effects

Sound effects form yet another group in which the sounds can be similar or completely dissimilar. The sound effect used in music vary from sounds from the nature to human

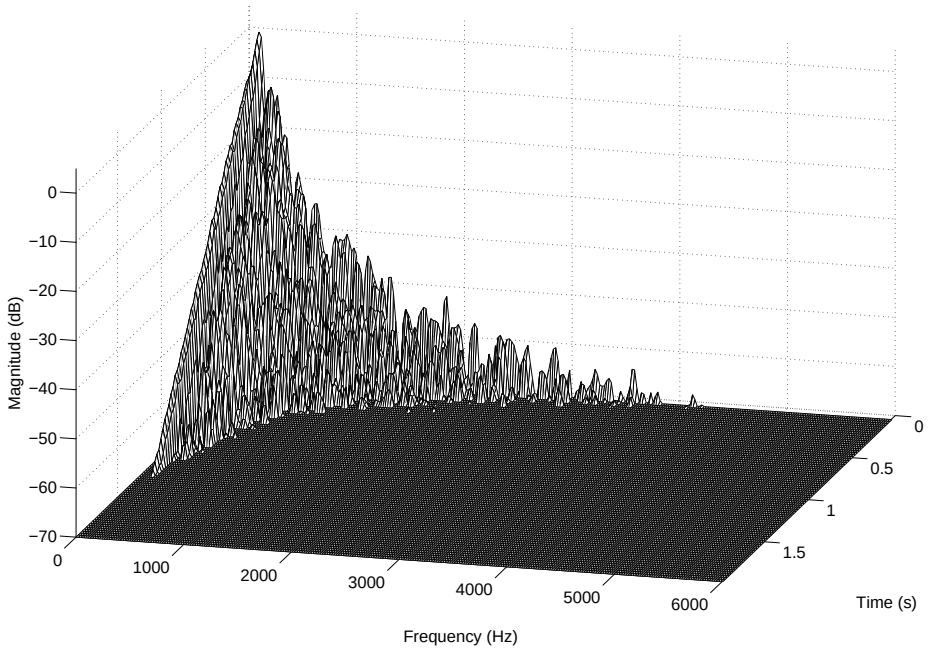


Figure 4.2: Spectrogram of a recorded low tom tone.

manufactured machines, so the generality of the sound design is minimal. However, many sounds have again a noise-like spectrum, which can be imitated with appropriately colored noise and additional tricks. In addition, they may contain single sinusoidal components, which are frequency modulated. Therefore, the focus of sound effect design is in finding the methods and tricks with which the desired timbres can be generated.

4.2 Parameter Variation in Generation of Sound Variants

The timbres found from synthesizers usually have quite many similar instrument definitions, which differ slightly from each other. For instance, the selection of piano sounds include the grand piano, the upright piano, the bright piano and various electric pianos to name few. Therefore the sound design is not just limited to finding the methods and parameters with which one instrument version can be generated. However, the sound variant design is usually reduced to simple parameter variations, with which clearly different but still similar sounds are obtained. This is illustrated next with a few commonly used instrument variants.

In Section 4.2.1, parameter variation rules for the generation of instrument variants with brighter timbres are given. Section 4.2.2 presents rules for generation of instruments with

wider timbre. Methods for increasing tone power is given in Section 4.2.3. In Section 4.2.4, methods for the generation of instrument variants with more reverberant timbres are presented.

4.2.1 Changing the Brightness

Brightness is a subjective measure of a sound, which is quite closely related to the spectral centroid, as a higher spectral centroid is perceived as brighter timbre. The spectral centroid can be shifted easily in additive and source-filter syntheses. In additive synthesis, the shift is done by changing the weight of the higher components, in practice by quite small amount in order to avoid too drastic changes. In source-filter synthesis, the same is obtained by shifting the cutoff frequency of the filter according to the desired change. The brightness control is illustrated next with two piano tones.

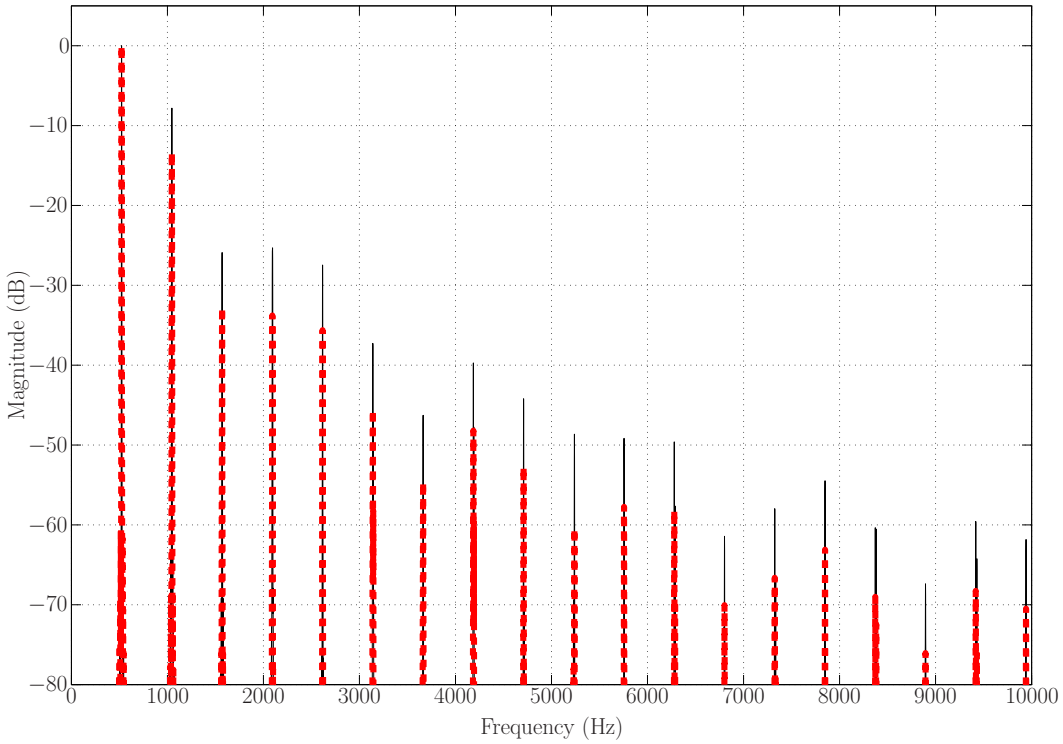


Figure 4.3: Comparison between synthesized Acoustic Grand Piano (dashed line) and Bright Acoustic Piano (solid line) spectra.

In Figure 4.3, the spectra of two piano tones played with an acoustic grand piano and a bright acoustic piano are presented. The grand piano tone is plotted with the dashed line in the front and the bright piano tone is given with the solid line in background. As can

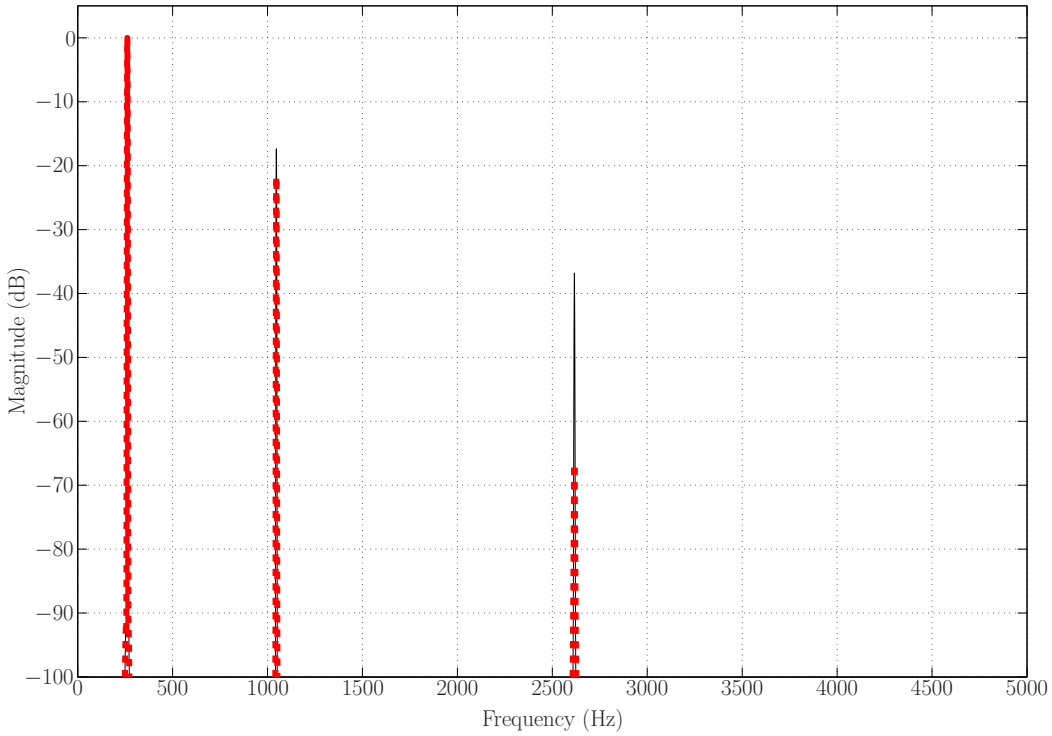


Figure 4.4: Comparison between synthesized marimba (dashed line) and wider marimba (solid line) spectra.

be seen in the figure, the higher harmonics of the bright piano tone are louder compared to the higher harmonics of the grand piano. This is obtained by doubling the cutoff frequency of the filter, and the resulting sound is clearly brighter but still recognizable to be a piano tone.

4.2.2 Widening of the Timbre

Wideness of a timbre is also a subjective measure, and it is related to the perceived sustain of the sound. However, the objective sustain of the sound is not changed at all, or it is changed only by a small amount. This can be obtained easily in additive and subtractive syntheses by increasing the decay rate of the higher components. Next, an example of widening of a marimba tone is given.

The spectra of a marimba and a wider marimba tone are given in Figure 4.4. The marimba tone is plotted with dashed line in the front, and the wider marimba tone is given with the solid line in the background. From the figure can be seen similar behavior as from Figure 4.3, and the wider timbres have similar characteristics as the brighter timbres. In

fact, a wider timbre can be obtained by increasing the brightness slightly. However, while in brighter timbres the decay rate of the higher components is the same as in the original, in wider timbres the higher components decay slightly slower. In Figure 4.4, the wider timbre is obtained by three times slower decay rates for the two higher components, and by increasing the weights of the second component by 35% and the third component by 7%.

4.2.3 Adding Power

The power of a tone is not trivially obtained by scaling the tone amplitude. It is related to energy concentration at the beginning of the tone, which can be obtained by increasing the hold time in the amplitude envelope. However, envelope must be carefully designed, since trivial addition of the hold phase to the envelope produces a sound that has electronic characteristics. This problem is addressed next with an example of a tom tone.

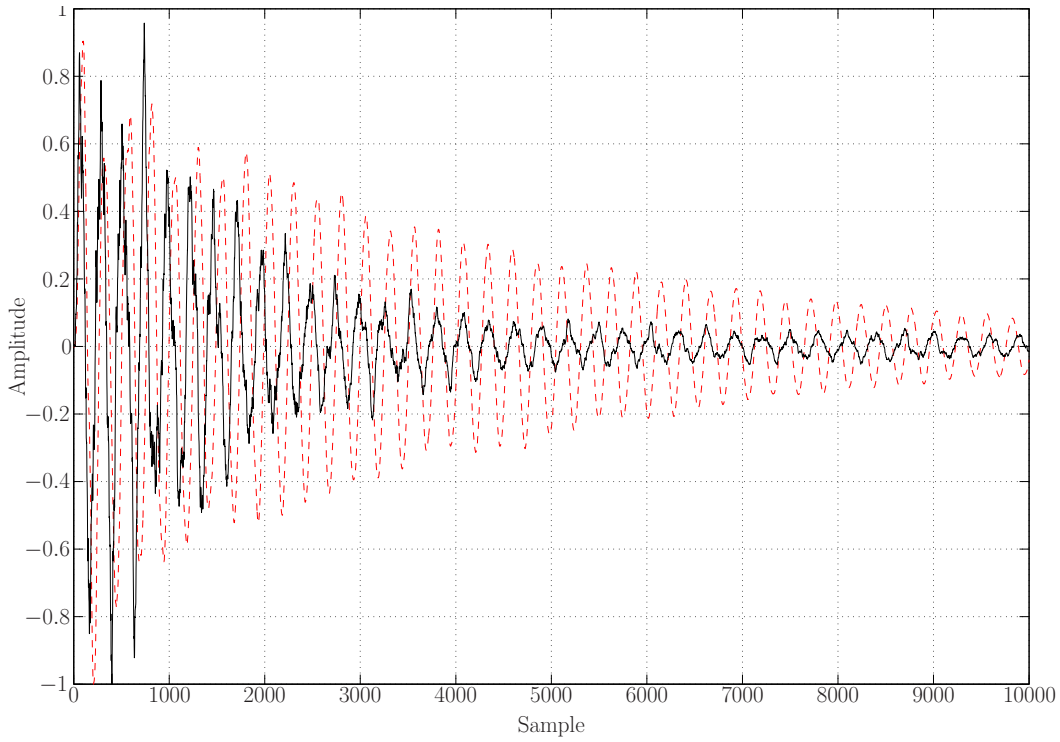


Figure 4.5: Comparison between synthesized tom (dashed line) and power tom (solid line) tones.

In Figure 4.5, time responses of tom and power tom tones are presented. The tom tone is plotted with dashed line and the power tom tone with solid line. Two observations can be made from the figure. First, the power tom tone decays faster than the tom tone. Secondly,

the power tom has more energy at the beginning of the tone. These two aspects are obtained by increasing a short hold phase after the attack, which is transformed to a decay phase by multiplying it with another envelope. The use of another envelope eliminates the electronic characteristics. The fast overall decay gives a sensation of more powerful sound, since now more energy is concentrated at the beginning of the tone.

4.2.4 Adding Reverberation

A reverberant sound is easily obtained by filtering the original sound with a reverb algorithm, but as in Section 3.4.2 is concluded, the use of traditional reverberation algorithms in computationally efficient music synthesis is problematic due to large delay lines required for good reverb modeling. Therefore, the generation of reverberant sounds requires tricks, and it is presented next with a tom tone.

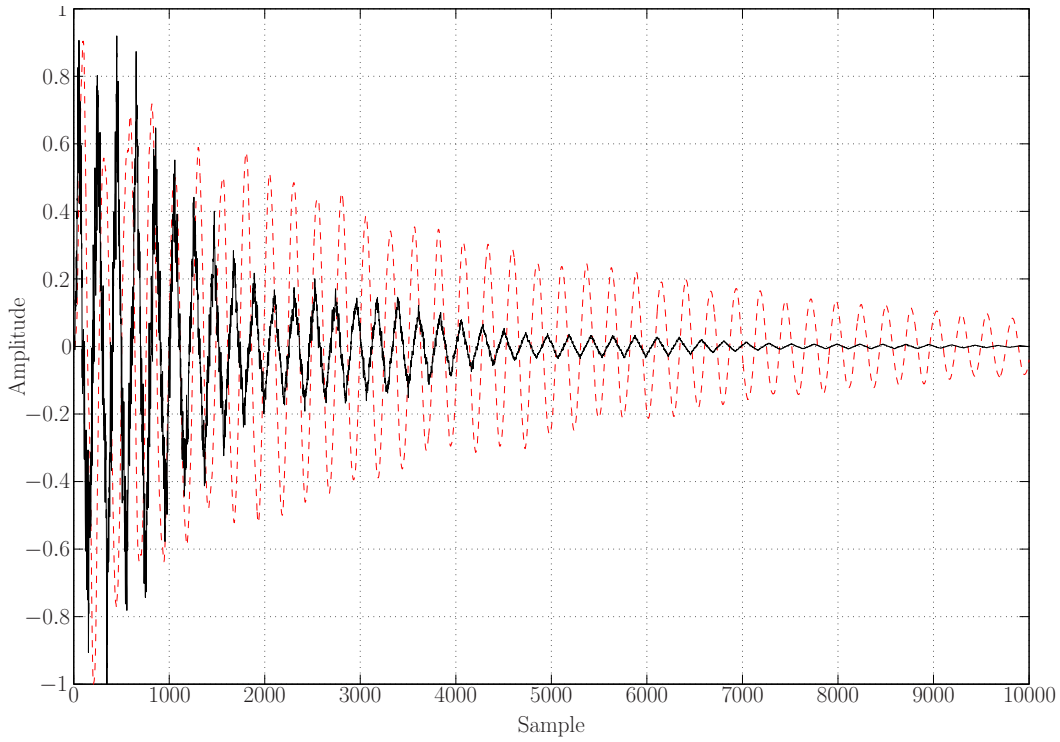


Figure 4.6: Comparison between synthesized tom (dashed line) and room tom (solid line) tones.

The time responses of tom and room tom tones are given in Figure 4.6. The tom tone is plotted with dashed line and the room tom tone with solid line. The room tom tone is again faster decaying, but the actual influence of the reverberation can be seen from its amplitude

behavior. The amplitude is modulated with a LFO, the frequency of which is quite high. In this case, the desired modulation is obtained with a 16 Hz triangular pulse wave, which varies the amplitude by 30%.

Chapter 5

Conclusions and Future Work

5.1 Main Results of This Thesis

The design of a music synthesizer for systems suffering from limited computing power and memory capacity was presented. Such systems are widely used in mobile devices, where power consumption of the device should be as small as possible. For example, these limitations can be found in the ring tone generation of mobile phones. Therefore, the synthesizer should be capable of generating sounds with simple calculations and minimal memory usage. Different possible synthesis methods are reviewed in this thesis, but with these limitations, the applicable sound synthesis techniques are limited to additive and source-filter syntheses, and in special cases to frequency modulation, wavetable and sampling syntheses. However, the synthesis technique to be used is ultimately determined by the sound to be imitated, and it may utilize a combination of applicable techniques.

The applicable synthesis techniques are obtained from interconnected oscillators and filters, which are often driven with some control signals. Sometimes the resulting sound is processed using a special sound effect, which transforms the often dull and dry timbre into more lively sound. Each of these synthesizer unit requires careful design, especially when the system sets limitations to the computations. Some existing methods for each unit is evaluated in this thesis with emphasis on computational efficiency and memory usage.

The largest design issue is raised in digital source-filter synthesis, where the use of classic waveforms, such as the sawtooth wave, is problematic due to aliasing caused by waveform discontinuities. A quasi-bandlimited waveform synthesis method based on the polynomial bandlimited step function approximation is presented in detail, and design rules for applicable polynomials are given. In addition, comparison between the first-order polynomial

and two different third-order polynomials are given. The third-order polynomials reduce aliasing more at high frequencies, but they exhibit more aliasing at low frequencies. Therefore, the use of higher order polynomials require careful design, and further analysis of the results are left for future work.

In addition, this thesis discusses the use of computationally efficient synthesis techniques in the imitation of different timbres, and presents design rules for the generation of sound variants and mapping of them to synthesis parameters. The design of a sound is based on the timbre characteristics, which can be modeled quite often with a single synthesis technique. However, in many cases the timbre is non-stationary, which yields the use of additional modulation techniques. In addition, the timbre may consist of special components, which must be generated separately. Variants of a sound can be generated by a small change in synthesis parameters of the original sound, and in many cases there are simple methods with which quite complex variants can be obtained without increasing the computational load.

5.2 Future Work

The most versatile sound synthesis technique in computationally efficient music synthesis is the source-filter synthesis, which offers only a mediocre sound quality. However, since the source-filter synthesis utilizes time-varying recursive filters and relatively simple source signals, which are also utilized in speech synthesis and coding, the methods derived for speech purposes could be used for the analysis of source-filter synthesis data. The filter coefficient estimation from speech data by using advanced LP techniques with emphasis on low and middle frequencies could be directly exploited. With these analysis techniques, the source-filter could model the timbre more accurately.

The excellence of speech coding is also based the efficient coding techniques for the residual, i.e., the source contribution. Therefore, the further development of source signal synthesis is justified. Especially further analysis of POLYBLEP waveform synthesizer is required. The reasonable balance between the polynomial type, order and computational complexity must be considered, and the bandlimiting results for different types and orders should be analyzed more. In addition, since many acoustical instruments have slightly inharmonic spectrum, the source-filter synthesis technique can not fully imitate these instruments. Therefore, the methods for inharmonicity control with the antialiasing waveform synthesis algorithms could be investigated.

Since most real world instruments include fluctuations of the tone characteristics, such

as fundamental frequency and amplitude, over time, further analysis of them using time-varying signal statistics would improve the liveliness of the synthesized tones, since the time-varying phenomena, such as beating and flutter, could be reproduced. By utilizing pseudo-random repetition of the variations, the synthesis of arbitrarily long tones in time could be possible by extending the tone decay with the repetition.

In all of the abovementioned research cases it would be necessary to conduct psychoacoustic modeling and listening test, with which evaluation of the algorithm development could be given. In addition, they would lead to the optimal usage of computing power for producing acoustic features that are perceptually relevant. Furthermore, the source-filter synthesis algorithm would be extended for accurate imitation of single musical tones, which leads to greatly improved sound quality comparable to sample-based techniques.

In addition, since the use of sound effects in music is common, the reduction of required delay line elements is required. This could be possible with a delay filter, which produce larger delays than the number of delay elements in the filter. However, the produced delay is non-uniform for all frequencies, especially when the desired delay is much larger than the filter order. Therefore, different delay filter designs could be evaluated and possible delay correction methods investigated.

Bibliography

[1] MIDI Manufacturers Association (MMA), *MIDI Specifications Home Page*. http://www.midi.org/about-midi/specshome.shtml, 2007. Checked June 1, 2007.

[2] C. Roads, *The Computer Music Tutorial*. The MIT Press, 1995.

[3] R. Bristow-Johnson, "Wavetable synthesis 101, a fundamental perspective," *Presented at the 101st AES Convention, Los Angeles, California, USA*, November 1996.

[4] D. C. Massie, "Wavetable sampling synthesis," in *Applications of Digital Signal Processing to Audio and Acoustics* (M. Kahrs and K. Brandenburg, eds.), ch. 8, pp. 311–342, Kluwer Academic, 1998.

[5] J. O. Smith and P. Gossett, "A flexible sampling-rate conversion method," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, San Diego, California, USA*, vol. 9, pp. 112–115, March 1984.

[6] J. O. Smith, *Digital Audio Resampling Home Page*. http://ccrma.stanford.edu/~jos/resample/resample.html, January 2002.

[7] T. I. Laakso, V. Välimäki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay – tools for fractional delay filter design," *IEEE Signal Processing Magazine*, vol. 13, pp. 30–60, January 1996.

[8] V. Välimäki and T. I. Laakso, "Principles of fractional delay filters," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, Istanbul, Turkey*, vol. 6, pp. 3870–3873, June 2000.

[9] V. Välimäki, M. Karjalainen, and T. I. Laakso, "Fractional delay digital filters," in *Proceedings of the 1993 International Symposium on Circuits and Systems, Chicago, Illinois, USA*, vol. 1, pp. 355–359, May 1993.

[10] R. C. Maher, "Compression and decompression of wavetable synthesis data," *Presented at the 115th AES Convention, New York, New York, USA*, October 2003.

[11] R. C. Maher, "Wavetable synthesis strategies for mobile devices," *Journal of the Audio Engineering Society*, vol. 53, pp. 205–212, March 2005.

[12] M. Szczerba, W. Oomen, and M. Klein Middelink, "Parametric audio coding based wavetable synthesis," *Presented at the 116th AES Convention, Berlin, Germany*, May 2004.

[13] J. Yuen and A. Horner, "Hybrid sampling-wavetable synthesis with genetic algorithms," *Journal of the Audio Engineering Society*, vol. 45, pp. 316–330, May 1997.

[14] A. Horner, J. Beauchamp, and L. Haken, "Methods for multiple wavetable synthesis of musical instrument tones," *Journal of the Audio Engineering Society*, vol. 41, pp. 336–356, May 1993.

[15] J. W. Beauchamp and A. Horner, "Wavetable interpolation synthesis based on time-variant spectral analysis of musical sounds," *Presented at the 98th AES Convention, Paris, France*, February 1995.

[16] A. Horner and L. Ayers, "Modeling acoustic wind instruments with contiguous group synthesis," *Journal of the Audio Engineering Society*, vol. 46, pp. 868–879, October 1998.

[17] X. Serra and J. O. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[18] X. Serra, "Musical sound modeling with sinusoids plus noise," in *Musical Signal Processing* (C. Roads, S. T. Pope, A. Piccialli, and G. De Poli, eds.), ch. 3, pp. 91–122, Swets & Zeitlinger, 1997.

[19] J. A. Moorer, "The use of the phase vocoder in computer music applications," *Journal of the Audio Engineering Society*, vol. 26, pp. 42–45, January/February 1978.

[20] M. Dolson, "The phase vocoder: A tutorial," *Computer Music Journal*, vol. 10, no. 4, pp. 14–27, 1986.

[21] J. W. Gordon and J. Strawn, "An introduction to the phase vocoder," in *Digital Audio Signal Processing – An Anthology* (J. Strawn, ed.), ch. 5, pp. 221–270, William Kauffmann Inc., 1985.

[22] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, pp. 744–754, August 1986.

[23] J. Laroche and M. Dolson, "Phase-vocoder: About this phasiness business," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, New York, USA*, October 1997.

[24] M. R. Portnoff, "Implementation of the digital phase vocoder using the fast Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, pp. 243–248, June 1976.

[25] H. Chamberlin, *Musical Applications of Microprocessors*. Hayden Books, 2nd ed., 1985.

[26] X. Rodet and P. Depalle, "Spectral envelopes and inverse FFT synthesis," *Presented at the 93rd AES Convention, San Francisco, California, USA*, October 1992.

[27] J. A. Moorer, "Signal processing aspects of computer music: A survey," in *Digital Audio Signal Processing – An Anthology* (J. Strawn, ed.), ch. 4, pp. 149–220, William Kauffmann Inc., 1985.

[28] C. Roads, "Granular synthesis of sound," in *Foundations of Computer Music* (C. Roads and J. Strawn, eds.), pp. 145–159, The MIT Press, 1985.

[29] S. C. Bass and T. W. Goeddel, "The efficient digital implementation of subtractive music synthesis," *IEEE Micro*, vol. 1, pp. 24–37, August 1981.

[30] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, pp. 561–580, April 1975.

[31] A. B. Carlson, P. B. Crilly, and J. C. Rutledge, *Communication Systems – An Introduction to Signals and Noise in Electrical Communication*. McGraw-Hill, 4th ed., 2002.

[32] J. Kleimola, "Design and implementation of a software sound synthesizer," Master's thesis, Helsinki University of Technology, Espoo, Finland, 2005. Available online at: http://www.acoustics.hut.fi/publications/.

[33] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Journal of the Audio Engineering Society*, vol. 21, pp. 526–534, September 1973.

[34] M. LeBrun, "A derivation of the spectrum of FM with a complex modulating wave," in *Foundations of Computer Music* (C. Roads and J. Strawn, eds.), pp. 65–67, The MIT Press, 1985.

[35] J.-P. Palamin, P. Palamin, and A. Ronveaux, "A method of generating and controlling musical asymmetrical spectra," *Journal of the Audio Engineering Society*, vol. 36, pp. 671–685, September 1988.

[36] B. T. G. Tan, S. L. Gan, S. M. Lim, and S. H. Tang, "Real-time implementation of double frequency modulation (DFM) synthesis," *Journal of the Audio Engineering Society*, vol. 42, pp. 918–926, November 1994.

[37] J. A. Bate, "The effect of modulator phase on timbres in FM synthesis," *Computer Music Journal*, vol. 14, no. 3, pp. 38–45, 1990.

[38] F. Holm, "Understanding FM implementations: A call for common standards," *Computer Music Journal*, vol. 16, no. 1, pp. 34–42, 1992.

[39] N. Tomisawa, "Tone production method for an electronic musical instrument." U.S. Patent 4,249,447.

[40] M. Le Brun, "Digital waveshaping synthesis," *Journal of the Audio Engineering Society*, vol. 27, pp. 250–266, April 1979.

[41] D. Arfib, "Digital synthesis of complex spectra by means of multiplication of nonlinear distorted sine waves," *Journal of the Audio Engineering Society*, vol. 27, pp. 757–768, October 1979.

[42] B. Lang, "Waveform synthesis using Bezier curves with control point modulation," *Presented at the 116th AES Convention, Berlin, Germany*, May 2004.

[43] V. Välimäki, J. Huopaniemi, M. Karjalainen, and Z. Jánosy, "Physical modeling of plucked string instruments with application to real-time sound synthesis," *Journal of the Audio Engineering Society*, vol. 44, pp. 331–353, May 1996.

[44] V. Välimäki, "Physics-based modeling of musical instruments," *Acta Acustica united with Acustica*, vol. 90, pp. 611–617, July/August 2004.

[45] V. Välimäki, J. Pakarinen, C. Erkut, and M. Karjalainen, "Discrete-time modelling of musical instruments," *Reports on Progress in Physics*, vol. 69, pp. 1–78, January 2006.

[46] J. M. Adrien and E. Ducasse, "Dynamic modeling of vibrating structures for sound synthesis, modal synthesis," in *Proceedings of the 7th AES International Conference, Toronto, Canada*, pp. 291–299, April 1989.

[47] C. Cadoz, A. Luciani, and J. Florens, "Responsive input devices and sound synthesis by simulation of instrumental mechanisms: The CORDIS system," *Computer Music Journal*, vol. 8, no. 3, pp. 60–73, 1984.

[48] J. O. Smith, "Music applications of digital waveguides," Tech. Rep. STAN-M-39, Center for Computer Research in Music and Acoustics, Stanford University, USA, 1987.

[49] J. O. Smith, "Physical modeling using digital waveguides," *Computer Music Journal*, vol. 16, no. 4, pp. 74–91, 1992.

[50] J. O. Smith, "Acoustic modeling using digital waveguides," in *Musical Signal Processing* (C. Roads, S. T. Pope, A. Piccialli, and G. De Poli, eds.), ch. 7, pp. 221–264, Swets & Zeitlinger, 1997.

[51] J. O. Smith, "Principles of digital waveguide models of musical instruments," in *Applications of Digital Signal Processing to Audio and Acoustics* (M. Kahrs and K. Brandenburg, eds.), ch. 10, pp. 417–466, Kluwer Academic, 1998.

[52] S. A. Van Duyne and J. O. Smith, "The 2-D digital waveguide mesh," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, New York, USA*, pp. 177–180, October 1993.

[53] S. A. Van Duyne and J. O. Smith, "Physical modeling with the 2-D digital waveguide mesh," in *Proceedings of International Computer Music Conference, Tokyo, Japan*, pp. 40–47, September 1993.

[54] S. A. Van Duyne and J. O. Smith, "The tetrahedral digital waveguide mesh," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, New York, USA*, pp. 234–237, October 1995.

[55] J. O. Smith and D. Rocchesso, *Aspects of Digital Waveguide Networks for Acoustic Modeling Applications*. Online paper: http://ccrma.stanford.edu/~jos/wgj/wgj.html, December 1997.

[56] M. Karjalainen, V. Välimäki, and T. Tolonen, "Plucked-string models: From the Karplus-Strong algorithm to digital waveguides and beyond," *Computer Music Journal*, vol. 22, no. 3, pp. 17–32, 1998.

[57] J. O. Smith, "Efficient synthesis of stringed musical instruments," in *Proceedings of International Computer Music Conference, Tokyo, Japan*, pp. 64–71, September 1993.

[58] M. Karjalainen, V. Välimäki, and Z. Jánosy, "Towards high-quality sound synthesis of the guitar and string instruments," in *Proceedings of International Computer Music Conference, Tokyo, Japan*, pp. 56–63, September 1993.

[59] T. Tolonen, "Model-based analysis and resynthesis of acoustic guitar tones," Master's thesis, Helsinki University of Technology, Espoo, Finland, 1998. Available online at: http://www.acoustics.hut.fi/publications/.

[60] D. A. Jaffe, "Ten criteria for evaluating synthesis techniques," *Computer Music Journal*, vol. 19, no. 1, pp. 76–87, 1995.

[61] T. Tolonen, V. Välimäki, and M. Karjalainen, "Evaluation of modern sound synthesis methods," Tech. Rep. 48, Laboratory of Acoustics and Audio Signal Processing, Helsinki University of Technology, Finland, 1998.

[62] E. Kreyszig, *Advanced Engineering Mathematics*. John Wiley & Sons, 8th ed., 1999.

[63] Wikipedia, *List of Trigonometric Identities*. http://en.wikipedia.org/wiki/List_of_trigonometric_identities, 2007. Checked June 1, 2007.

[64] J. Dattorro, "Effect design, part 3 oscillators: Sinusoids and pseudonoise," *Journal of the Audio Engineering Society*, vol. 50, pp. 115–146, March 2002.

[65] J. O. Smith and P. R. Cook, "The second-order digital waveguide oscillator," in *Proceedings of International Computer Music Conference, San Jose, California, USA*, pp. 150–153, October 1992.

[66] J. Dattorro, "The implementation of recursive digital filters for high-fidelity audio," *Journal of the Audio Engineering Society*, vol. 36, pp. 851–878, November 1988.

[67] A. I. Abu-El-Haija and M. M. Al-Ibrahim, "Improving performance of digital sinusoidal oscillator by means of error feedback circuits," *IEEE Transactions on Circuits and Systems*, vol. 33, pp. 373–380, April 1986.

[68] B. Hayes, "Randomness as a resource," *American Scientist*, vol. 89, pp. 300–304, July/August 2001.

[69] S. K. Park and K. W. Miller, "Random number generators: Good ones are hard to find," *Communications of the ACM*, vol. 31, pp. 1192–1201, October 1988.

[70] P. L'Ecuyer, "Tables of linear congruential generators of different sizes and good lattice structure," *Mathematics of Computation*, vol. 68, pp. 249–260, January 1999.

[71] T. G. Lewis and W. H. Payne, "Generalized feedback shift register pseudorandom number algorithm," *Journal of the ACM*, vol. 20, pp. 456–468, July 1973.

[72] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on Computing*, vol. 15, pp. 364–383, May 1986.

[73] M. Matsumoto and Y. Kurita, "Twisted GFSR generators," *ACM Transactions on Modeling and Computer Simulation*, vol. 2, pp. 179–194, July 1992.

[74] M. Matsumoto and Y. Kurita, "Twisted GFSR generators II," *ACM Transactions on Modeling and Computer Simulation*, vol. 4, pp. 254–266, July 1994.

[75] H. F. Olson, H. Belar, and J. Timmens, "Electronic music synthesis," *The Journal Of The Acoustical Society Of America*, vol. 32, pp. 311–319, March 1960.

[76] J. A. Moorer, "Signal processing aspects of computer music: A survey," *Proceedings of the IEEE*, vol. 65, pp. 1108–1137, August 1977.

[77] H. G. Alles, "Music synthesis using real time digital techniques," *Proceedings of the IEEE*, vol. 68, pp. 436–449, April 1980.

[78] J. W. Gordon, "System architectures for computer music," *ACM Computing Surveys*, vol. 17, pp. 191–233, June 1985.

[79] A. Huovilainen and V. Välimäki, "New approaches to digital subtractive synthesis," in *Proceesing of International Computer Music Conference, Barcelone, Spain*, pp. 399–402, September 2005.

[80] V. Välimäki and A. Huovilainen, "Oscillator and filter algorithms for virtual analog synthesis," *Computer Music Journal*, vol. 30, no. 2, pp. 19–31, 2006.

[81] V. Välimäki and A. Huovilainen, "Antialiasing oscillators in subtractive synthesis," *IEEE Signal Processing Magazine*, vol. 24, pp. 116–125, March 2007.

[82] A. Chaudhary, "Band-limited simulation of analog synthesizer modules by additive synthesis," *Presented at the 105th AES Convention, San Francisco, California, USA*, August 1998.

[83] G. Winham and K. Steiglitz, "Input generators for digital sound synthesis," *The Journal Of The Acoustical Society Of America*, vol. 47, pp. 665–666, February 1970.

[84] J. A. Moorer, "The synthesis of complex audio spectra by means of discrete summation formulas," *Journal of the Audio Engineering Society*, vol. 24, pp. 717–727, November 1976.

[85] P. Kleczkowski, "Group additive synthesis," *Computer Music Journal*, vol. 13, no. 1, pp. 12–20, 1989.

[86] A. Horner and S. Wun, "Low peak amplitudes for group additive synthesis," *Journal of the Audio Engineering Society*, vol. 53, pp. 475–484, June 2005.

[87] T. Stilson and J. O. Smith, "Alias-free digital synthesis of classic analog waveforms," in *Proceedings of International Computer Music Conference, Hong Kong, China*, pp. 332–335, August 1996.

[88] E. Brandt, "Hard sync without aliasing," in *Proceedings of International Computer Music Conference, Havana, Cuba*, pp. 365–368, September 2001.

[89] J. Lane, D. Hoory, E. Martinez, and P. Wang, "Modeling analog synthesis with DSPs," *Computer Music Journal*, vol. 21, no. 4, pp. 23–41, 1997.

[90] V. Välimäki, "Discrete-time synthesis of the sawtooth waveform with reduced aliasing," *IEEE Signal Processing Letters*, vol. 12, pp. 214–217, March 2005.

[91] D. Lowenfels, "Virtual analog synthesis with a time-varying comb filter," *Presented at the 115th AES Convention, New York, New York, USA*, September 2003.

[92] D. Rossum, "Making digital filters sound "analog"," in *Proceedings of International Computer Music Conference, San Jose, California, USA*, pp. 30–33, October 1992.

[93] K. Steiglitz, *A Digital Signal Processing Primer with Applications to Digital Audio and Computer Music*. Addison-Wesley, 1996.

[94] J. O. Smith and J. B. Angell, "A constant-gain digital resonator tuned by a single coefficient," *Computer Music Journal*, vol. 6, no. 4, pp. 36–40, 1982.

[95] R. A. Moog, "Voltage-controlled electronic music modules," *Journal of the Audio Engineering Society*, vol. 13, pp. 200–206, July 1965.

[96] R. A. Moog, "A voltage-controlled low-pass high-pass filter for audio signal processing," *Presented at the 17th AES Annual Meeting, New York, New York, USA*, October 1965.

[97] T. Stilson and J. O. Smith, "Analyzing the Moog VCF with considerations for digital implementation," in *Proceedings of International Computer Music Conference, Hong Kong, China*, pp. 398–401, August 1996.

[98] A. Huovilainen, "Non-linear digital implementation of the Moog ladder filter," in *Proceedings of the 7th International Conference on Digital Audio Effects, Naples, Italy*, pp. 61–64, October 2004.

[99] J. Dattorro, "Effect design, part 2: Delay line modulation and chorus," *Journal of the Audio Engineering Society*, vol. 45, pp. 764–788, October 1997.

[100] J. O. Smith, "An allpass approach to digital phasing and flanging," Tech. Rep. STAN-M-21, Center for Computer Research in Music and Acoustics, Stanford University, USA, 1982.

[101] W. G. Gardner, "Reverberation algorithms," in *Applications of Digital Signal Processing to Audio and Acoustics* (M. Kahrs and K. Brandenburg, eds.), ch. 3, pp. 85–132, Kluwer Academic, 1998.

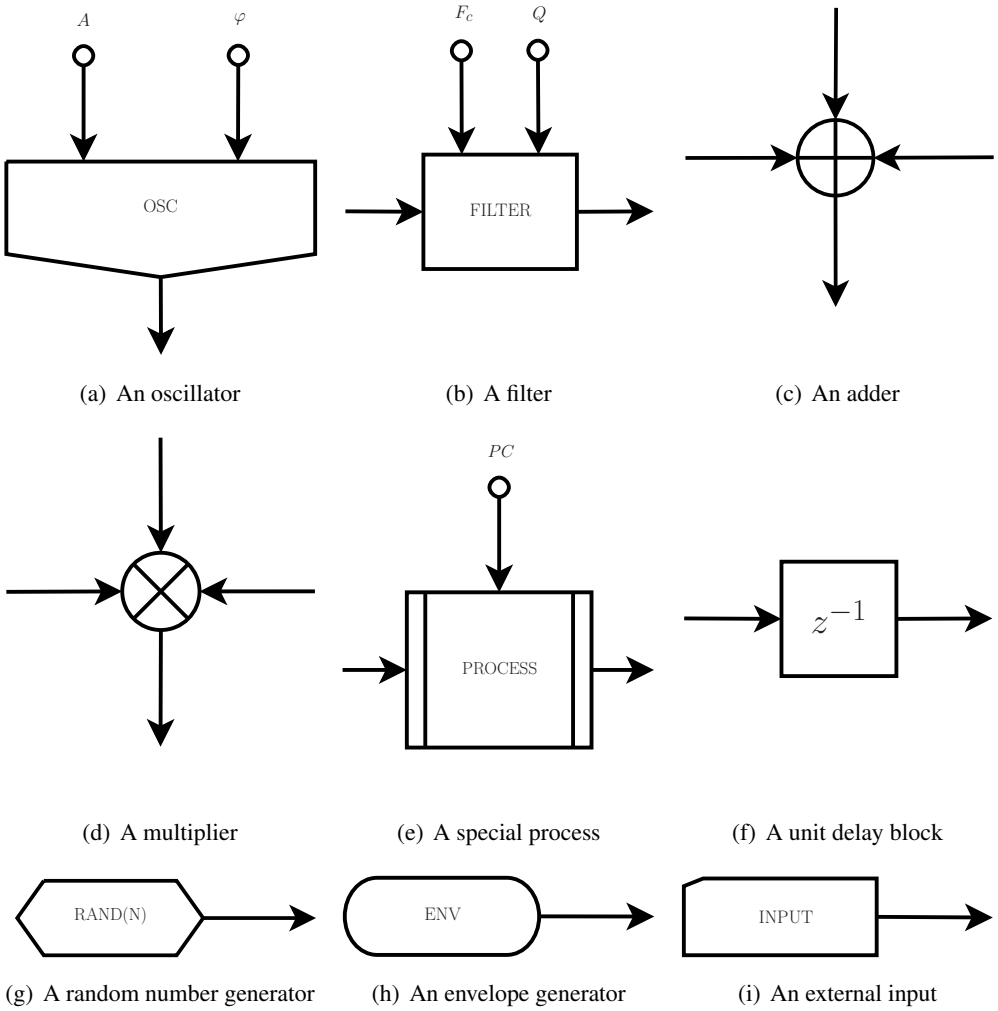
[102] U. Zölzer, ed., *DAFX – Digital Audio Effects*. John Wiley & Sons, 2002.

Appendix A

Synthesis System Notation

In Figure A.1, an adapted version of MUSIC block notation is presented, and this notation is used in this thesis. An oscillator can produce arbitrary waveforms, and the waveform is explicitly given in the oscillator block. The oscillator has two controllers, amplitude and instantaneous phase, A and φ, respectively. A filter can be of any type, and the filter type is explicitly given in the filter block. The filter can have two controllers, the cutoff frequencies and resonance parameters, F_c and Q, respectively.

An adder output the sum of its input, and a multiplier outputs the product of its inputs. A special process implements certain process to its input, and the process is possibly controlled with process control parameter PC. A random number generator produces white noise signal, either Gaussian or equally distributed. An envelope generator produces a time-varying function, and the function is either given explicitly in the envelope generator block. An external input converts an external signal, e.g., from memory into a signal processed the synthesis system.



(a) An oscillator (b) A filter (c) An adder

(d) A multiplier (e) A special process (f) A unit delay block

(g) A random number generator (h) An envelope generator (i) An external input

Figure A.1: Synthesis notation used in this thesis.

Appendix B

Summary of MIDI specifications

MIDI was first introduced as an interface protocol between different keyboard and sound module, but it has been expanded to allow transfer control messages between any sound generation devices. The control messages can be definitions of the instrument to be used, or control parameters which modify the timbre produced by the system. However, the control messages which are most commonly transferred are the instrument definitions, and they form the biggest sound design problem in creation of a MIDI compatible implementation. Depending on the version of MIDI specification, different requirements for simultaneous notes and overall timbre collection are set, and these are summarized next.

There exist four different MIDI specifications, which are used in different applications. In General MIDI Level 1 (GM-1), the implementation system must be capable of playing at least 24 simultaneous notes produced by 16 MIDI channels. The MIDI channels are allocated to different instruments, of which one can be percussion set. General MIDI Lite (GM-LITE) sets the simultaneous note limit to 16, and they are allocated to 15 melodic instruments and one percussion set. In Scalable Polyphony MIDI (SP-MIDI), the polyphony can be scaled depending, e.g., device model or power consumption, according to MIDI channel priorities. The channel definitions are the same as in GM-LITE. General MIDI Level 2 (GM-2) extends the GM-1 by requiring at least 32 simultaneous notes produced by 16 channels, of which all can be allocated to melodic instruments, and up to two to percussion sets.

GM-1 compatible system must provide 128 different melodic and 47 drum sounds, which are required also in GM-LITE and SP-MIDI specifications. GM-2 specification extends the sound sets of GM-1 by introducing 128 additional melodic sounds incorporated from Yamaha XG format, and extending the GM-1 percussion set with 14 additional sounds and introducing eight additional drum sets incorporated from Roland GS format. The melodic

sound and percussion sets of GM-2 compatible system are presented in Tables B.1 and B.2, respectively. The melodic sound for GM-1, GM-LITE and SP-MIDI compatible systems are obtained from GM-2 sounds at bank 00, and only percussion set Standard with keys 35–81 are used.

Table B.1: Melodic sound patches of the General MIDI Level 2 specification.

(a) Piano

Prog	Bank	Name
001	00	Acoustic Grand Piano
	01	Acoustic Grand Piano (wide)
	02	Acoustic Grand Piano (dark)
002	00	Bright Acoustic Piano
	01	Bright Acoustic Piano (wide)
003	00	Electric Grand Piano
	01	Electric Grand Piano (wide)
004	00	Honky-tonk Piano
	01	Honky-tonk Piano (wide)
005	00	Electric Piano 1
	01	Detuned Electric Piano 1
	02	Electric Piano 1 (velocity mix)
	03	60's Electric Piano
006	00	Electric Piano 2
	01	Detuned Electric Piano 2
	02	Electric Piano 2 (velocity mix)
	03	EP Legend
	04	EP Phase
007	00	Harpsichord
	01	Harpsichord (octave mix)
	02	Harpsichord (wide)
	03	Harpsichord (with key off)
008	00	Clavi
	01	Pulse Clavi

(b) Chromatic Percussion

Prog	Bank	Name
009	00	Celesta
010	00	Glockenspiel
011	00	Music Box
012	00	Vibraphone
	01	Vibraphone (wide)
013	00	Marimba
	01	Marimba (wide)
014	00	Xylophone
015	00	Tubular Bells
	01	Church Bells
	02	Carillon
016	00	Dulcimer

Table B.1: Melodic sound patches of the General MIDI Level 2 specification. Continued.

(c) Organ

Prog	Bank	Name
017	00	Drawbar Organ
	01	Detuned Drawbar Organ
	02	Italian 60's Organ
	03	Drawbar Organ 2
018	00	Percussive Organ
	01	Detuned Percussive Organ
	02	Percussive Organ 2
019	00	Rock Organ
020	00	Church Organ
	01	Church Organ (octave mix)
	02	Detuned Church Organ
021	00	Reed Organ
	01	Puff Organ
022	00	Accordion
	01	Accordion 2
023	00	Harmonica
024	00	Tango Accordion

(d) Guitar

Prog	Bank	Name
025	00	Acoustic Guitar (nylon)
	01	Ukulele
	02	Acoustic Guitar (nylon + key off)
	03	Acoustic Guitar (nylon 2)
026	00	Acoustic Guitar (steel)
	01	12-Strings Guitar
	02	Mandolin
	03	Steel Guitar with Body Sound
027	00	Electric Guitar (jazz)
	01	Electric Guitar (pedal steel)
028	00	Electric Guitar (clean)
	01	Electric Guitar (detuned clean)
	02	Mid Tone Guitar
029	00	Electric Guitar (muted)
	01	Electric Guitar (funky cutting)
	02	Electric Guitar (muted velo-sw)
	03	Jazz Man
030	00	Overdriven Guitar
	01	Guitar Pinch
031	00	Distortion Guitar
	01	Distortion Guitar (with feedback)
	02	Distorted Rhythm Guitar
032	00	Guitar Harmonics
	01	Guitar Feedback

Table B.1: Melodic sound patches of the General MIDI Level 2 specification. Continued.

(e) Bass

Prog	Bank	Name
033	00	Acoustic Bass
034	00	Electric Bass (finger)
	01	Finger Slap Bass
035	00	Electric Bass (pick)
036	00	Fretless Bass
037	00	Slap Bass 1
038	00	Slap Bass 2
039	00	Synth Bass 1
	01	Synth Bass (warm)
	02	Synth Bass 3 (resonance)
	03	Clavi Bass
	04	Hammer

Prog	Bank	Name
040	00	Synth Bass 2
	01	Synth Bass 4 (attack)
	02	Synth Bass (rubber)
	03	Attack Pulse

(f) Strings & Orchestral Instruments

Prog	Bank	Name
041	00	Violin
	01	Violin (slow attack)
042	00	Viola
043	00	Cello
044	00	Contrabass

Prog	Bank	Name
045	00	Tremolo Strings
046	00	Pizzicato Strings
047	00	Orchestral Harp
	01	Yang Chin
048	00	Timpani

(g) Ensemble

Prog	Bank	Name
049	00	String Ensembles 1
	01	Strings and Brass
	02	60s Strings
050	00	String Ensembles 2
051	00	Synth Strings 1
	01	Synth Strings 3
052	00	Synth Strings 2
053	00	Choir Aahs
	01	Choir Aahs 2

Prog	Bank	Name
054	00	Voice Oohs
	01	Humming
055	00	Synth Voice
	01	Analog Voice
056	00	Orchestral Hit
	01	Bass Hit Plus
	02	6th Hit
	03	Euro Hit

Table B.1: Melodic sound patches of the General MIDI Level 2 specification. Continued.

(h) Brass

Prog	Bank	Name
057	00	Trumpet
	01	Dark Trumpet Soft
058	00	Trombone
	01	Trombone 2
	02	Bright Trombone
059	00	Tuba
060	00	Muted Trumpet
	01	Muted Trumpet 2
061	00	French Horn
	01	French Horn 2 (warm)

Prog	Bank	Name
062	00	Brass Section
	01	Brass Section 2 (octave mix)
063	00	Synth Brass 1
	01	Synth Brass 3
	02	Analog Synth Brass 1
	03	Jump Brass
064	00	Synth Brass 2
	01	Synth Brass 4
	02	Analog Synth Brass 2

(i) Reed

Prog	Bank	Name
065	00	Soprano Sax
066	00	Alto Sax
067	00	Tenor Sax
068	00	Baritone Sax

Prog	Bank	Name
069	00	Oboe
070	00	English Horn
071	00	Bassoon
072	00	Clarinet

(j) Pipe

Prog	Bank	Name
073	00	Piccolo
074	00	Flute
075	00	Recorder
076	00	Pan Flute

Prog	Bank	Name
077	00	Blown Bottle
078	00	Shakuhachi
079	00	Whistle
080	00	Ocarina

(k) Synth Lead

Prog	Bank	Name
081	00	Lead 1 (square)
	01	Lead 1a (square 2)
	02	Lead 1b (sine)
082	00	Lead 2 (sawtooth)
	01	Lead 2a (sawtooth 2)
	02	Lead 2b (saw + pulse)
	03	Lead 2c (double sawtooth)
	04	Lead 2d (sequenced analog)

Prog	Bank	Name
083	00	Lead 3 (calliope)
084	00	Lead 4 (chiff)
085	00	Lead 5 (charang)
	01	Lead 5a (wire lead)
086	00	Lead 6 (voice)
087	00	Lead 7 (fifths)
088	00	Lead 8 (bass + lead)
	01	Lead 8a (soft wrl)

Table B.1: Melodic sound patches of the General MIDI Level 2 specification. Continued.

(l) Synth Pad

Prog	Bank	Name
089	00	Pad 1 (new age)
090	00	Pad 2 (warm)
	01	Pad 2a (sine pad)
091	00	Pad 3 (polysynth)
092	00	Pad 4 (choir)
	01	Pad 4a (itopia)
093	00	Pad 5 (bowed)
094	00	Pad 6 (metallic)
095	00	Pad 7 (halo)
096	00	Pad 8 (sweep)

(m) Synth SFX

Prog	Bank	Name
097	00	FX 1 (rain)
098	00	FX 2 (soundtrack)
099	00	FX 3 (crystal)
	01	FX 3a (synth mallet)
100	00	FX 4 (atmosphere)
101	00	FX 5 (brightness)
102	00	FX 6 (goblins)
103	00	FX 7 (echoes)
	01	FX 7a (echo bell)
	02	FX 7b (echo pan)
104	00	FX 8 (sci-fi)

(n) Ethnic Misc.

Prog	Bank	Name
105	00	Sitar
	01	Sitar 2 (bend)
106	00	Banjo
107	00	Shamisen
108	00	Koto
	01	Taisho Koto
109	00	Kalimba
110	00	Bag Pipe
111	00	Fiddle
112	00	Shanai

(o) Percussion

Prog	Bank	Name
113	00	Tinkle Bell
114	00	Agogo
115	00	Steel Drums
116	00	Woodblock
	01	Castanets
117	00	Taiko Drum
	01	Concert Bass Drum
118	00	Melodic Tom
	01	Melodic Tom 2
119	00	Synth Drum
	01	Rhythm Box Tom
	02	Electric Drum
120	00	Reverse Cymbal

Table B.1: Melodic sound patches of the General MIDI Level 2 specification. Continued.

(p) Sound FX

Prog	Bank	Name
121	00	Guitar Fret Noise
	01	Guitar Cutting Noise
	02	Acoustic Bass String Slap
122	00	Breath Noise
	01	Flute Key Click
123	00	Seashore
	01	Rain
	02	Thunder
	03	Wind
	04	Stream
	05	Bubble
124	00	Bird Tweet
	01	Dog
	02	Horse Gallop
	03	Bird Tweet 2
125	00	Telephone Ring
	01	Telephone Ring 2
	02	Door Creaking
	03	Door
	04	Scratch
	05	Wind Chime
126	00	Helicopter
	01	Car Engine
	02	Car Stop
	03	Car Pass
	04	Car Crash
	05	Siren
	06	Train
	07	Jetplane
	08	Starship
	09	Burst Noise
127	00	Applause
	01	Laughing
	02	Screaming
	03	Punch
	04	Heart Beat
	05	Footsteps
128	00	Gunshot
	01	Machine Gun
	02	Lasergun
	03	Explosion

Table B.2: Percussion sets of the General MIDI Level 2 specification.

(a) #01 Standard Set

Key	Name	Key	Name	Key	Name
27	High Q	48	High-Mid Tom	69	Cabasa
28	Slap	49	Crash Cymbal 1	70	Maracas
29	Scratch Push	50	High Tom	71	Short Whistle
30	Scratch Pull	51	Ride Cymbal 1	72	Long Whistle
31	Sticks	52	Chinese Cymbal	73	Short Guiro
32	Square Click	53	Ride Bell	74	Long Guiro
33	Metronome Click	54	Tambourine	75	Claves
34	Metronome Bell	55	Splash Cymbal	76	Hi Wood Block
35	Acoustic Bass Drum	56	Cowbell	77	Low Wood Block
36	Bass Drum 1	57	Crash Cymbal 2	78	Mute Cuica
37	Side Stick	58	Vibra-Slap	79	Open Cuica
38	Acoustic Snare	59	Ride Cymbal 2	80	Mute Triangle
39	Hand Clap	60	High Bongo	81	Open Triangle
40	Electric Snare	61	Low Bongo	82	Shaker
41	Low Floor Tom	62	Mute Hi Conga	83	Jingle Bell
42	Closed Hi-hat	63	Open Hi Conga	84	Bell Tree
43	High Floor Tom	64	Low Conga	85	Castanets
44	Pedal Hi-hat	65	High Timbale	86	Mute Surdo
45	Low Tom	66	Low Timbale	87	Open Surdo
46	Open Hi-hat	67	High Agogo		
47	Low-Mid Tom	68	Low Agogo		

(b) #09 Room Set

Key	Name	Key	Name	Key	Name
41	Room Low Tom 2	45	Room Mid Tom 2	48	Room Hi Tom 2
43	Room Low Tom 1	47	Room Mid Tom 1	50	Room Hi Tom 1

(c) #17 Power Set

Key	Name	Key	Name	Key	Name
36	Power Kick Drum	43	Power Low Tom 1	48	Power Hi Tom 2
38	Power Snare Drum	45	Power Mid Tom 2	50	Power Hi Tom 1
41	Power Low Tom 2	47	Power Mid Tom 1		

Table B.2: Percussion sets of the General MIDI Level 2 specification. Continued.

(d) #25 Electronic Set

Key	Name	Key	Name	Key	Name
36	Electric Bass Drum	43	Electric Low Tom 1	50	Electric Hi Tom 1
38	Electric Snare 1	45	Electric Mid Tom 2	52	Reverse Cymbal
40	Electric Snare 2	47	Electric Mid Tom 1		
41	Electric Low Tom 2	48	Electric Hi Tom 2		

(e) #26 Analog Set

Key	Name	Key	Name	Key	Name
36	Analog Bass Drum	45	Analog Mid Tom 2	62	Analog High Conga
37	Analog Rim Shot	46	Analog OHH	63	Analog Mid Conga
38	Analog Snare 1	47	Analog Mid Tom 1	64	Analog Hi Conga
41	Analog Low Tom 2	48	Analog Hi Tom 2	70	Analog Maracas
42	Analog CHH 1	49	Analog Cymbal	75	Analog Claves
43	Analog Low Tom 1	50	Analog Hi Tom 1		
44	Analog CHH 2	56	Analog Cowbell		

(f) #33 Jazz Set

Key	Name	Key	Name
35	Jazz Kick 2	36	Jazz Kick 1

(g) #41 Brush Set

Key	Name	Key	Name	Key	Name
35	Jazz Kick 2	38	Brush Tap	40	Brush Swirl
36	Jazz Kick 1	39	Brush Slap		

(h) #49 Orchestra Set

Key	Name	Key	Name	Key	Name
27	Closed Hi-hat 2	41	Timpani F	50	Timpani d
28	Pedal Hi-hat	42	Timpani F#	51	Timpani d#
29	Open Hi-hat 2	43	Timpani G	52	Timpani e
30	Ride Cymbal 1	44	Timpani G#	53	Timpani f
35	Concert BD 2	45	Timpani A	57	Concert Cymbal 2
36	Concert BD 1	46	Timpani A#	58	Concert Cymbal 1
38	Concert SD	47	Timpani B	88	Applause
39	Castanets	48	Timpani c		
40	Concert SD	49	Timpani c#		

Table B.2: Percussion sets of the General MIDI Level 2 specification. Continued.

(i) #57 SFX Set

Key	Name	Key	Name	Key	Name
39	High Q	55	Heart Beat	71	Starship
40	Slap	56	Footsteps 1	72	Gun Shot
41	Scratch Push	57	Footsteps 2	73	Machine Gun
42	Scratch Pull	58	Applause	74	Lasergun
43	Sticks	59	Door Creaking	75	Explosion
44	Square Click	60	Door	76	Dog
45	Metronome Click	61	Scratch	77	Horse-Gallop
46	Metronome Bell	62	Wind Chimes	78	Birds
47	Guitar Fret Noise	63	Car-Engine	79	Rain
48	Guitar Cutting Noise Up	64	Car-Stop	80	Thunder
49	Guitar Cutting Noise Down	65	Car-Pass	81	Wind
50	String Slap of Double Bass	66	Car-Crash	82	Seashore
51	Flute Key Click	67	Siren	83	Stream
52	Laughing	68	Train	84	Bubble
53	Scream	69	Jetplane		
54	Punch	70	Helicopter		