The Musical Spectrogram: A Tool for the Time-Frequency Analysis of Musical Signals

by

Anthony Antoniou
B.Eng., University of Victoria, Canada, 1995

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of
Electrical and Computer Engineering

We accept this thesis as conforming
to the required standard

[redacted]

Dr. D. J. Shpak, Supervisor (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. P. Agathoklis, Supervisor (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. K. F. Li, Departmental Member (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. P. F. Driessen, Departmental Member (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. W. A. Schloss, Outside Member (School of Music, Faculty of Fine Arts)

[redacted]

Dr. R. Illner, External Examiner

Supervisors: Dr. D. J. Shpak , Dr. Pan Agathoklis

ABSTRACT

The frequency domain of signals is traditionally examined through techniques based on the Fourier transform: These techniques give a linear view of the frequency spectrum in Hertz, and are ideal for the numerical analysis and manipulation of spectra. In this work, we propose a scheme suited to the analysis of musical signals: The Musical Spectrogram.

A four-stage system, the Musical Spectrogram consists of a tuning module, a 9 octave-band filter bank, a 12 semitone-band filter bank array and an output system. The spectrum of a musical signal is first decomposed into octave subbands, and each is separated into musical scale semitone subbands and plotted. New insights are offered by exploring the musical spectrograms of a number of test signals, and the commercial potentials of this new analysis tool are explored.

Simulation results of a Musical Spectrogram system are presented. A complete MATLAB implementation of this system is tested to prove the function and form of the system. A limited real-time implementation of this system on the Analog Devices ADSP-2181 fixed-point digital signal processor is also presented which underscores the many difficulties inherent in development for such platforms, and details several novel solutions devised to address them. The numerous advantages and properties of the Musical Spectrogram system are also presented and discussed, and a complete real-time software version is put forward as future work.

Examiners:

[redacted]

Dr. D. J. Shpak, Supervisor (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. P. Agathoklis, Supervisor (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. K. F. Li, Departmental Member (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. P. F. Driessen, Departmental Member (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. W. A. Schloss, Outside Member (School of Music, Faculty of Fine Arts)

[redacted]

Dr. R. Illner, External Examiner

List of Figures

List of Tables

Acknowledgments

I would like to thank the many people who have made this dissertation possible. First of all, I thank my supervisors, Dr. Pan Agathoklis and Dr. Dale Shpak of the Department of Electrical and Computer Engineering. Their supervision, support, understanding and (most of all) patience during the many years that have elapsed between this work's inception and its final completion have been constant. I would also like to thank my supervisory committee members, Dr. Kin Li, Dr. Peter Driessen and Dr. Andy Schloss for agreeing to review my work on such a compressed timescale. After this many years in the making, I suppose it is only natural that the final convergence should seem practically instantaneous. I appreciate their understanding.

A great debt of thanks goes out to my father, Dr. Andreas Antoniou, whose support and advice has been continuous throughout my life - particularly over the last few months. He has always been there for me, and I shall never forget it.

Finally, I would like to thank my fiancee Lacey and my daughter Amanda. Without their moral support, love and consideration I would never have been able to pick up the fallen pieces and forge this work back together again after the passing of my mother in 1996. A broken will is the hardest thing to mend, and thanks to the love of my family, my determination to succeed has been restored.

Completing this thesis means I can move on to the challenges ahead. Thank you all, for making it possible!

List of Abbreviations

ALU - Arithmetic Logic Unit

CD - Compact Disc

CWT - Continuous Wavelet Transform

dB - Decibels

DFT - Discrete Fourier Transform

DSB - Dyadic Semitone Band

DSP - Digital Signal Processor

DWT - Discrete Wavelet Transform

FFT - Fast Fourier Transform

FIR - Finite Impulse Response

Hz - Hertz

IIR - Infinite-duration Impulse Response

kHz - kiloHertz

MAC - Multiplier/Accumulator

MIDI - Musical Insrument Digital Interface

OPM - Octave Processing Module

PC - Personal Computer

QMF - Quadrature Mirror-image Filter

SPM - Semitone Processing Module

STFT - Short-Time Fourier Transform

VLSI - Very Large Scale Integration

Chapter 1

Introduction

1.1 General

The proliferation of increasingly powerful desk-top computers has made possible affordable audio editing software tools that yield excellent results in real-time. As the technology of personal computers has advanced, the capabilities of digital multi-track recording and processing software have kept pace. Today's computer based hard-disk digital recording systems have surpassed the quality of magnetic tape systems hundreds of times their cost[1]. In addition they offer additional analysis and processing functions far beyond the capabilities of such older analog equipment. Other advances in the technology of music have seen the emergence of sophisticated digital audio processing algorithms that filter and process signals based on musical parameters such as key, scale, or tempo. Over the last decade, personal computer technology has advanced to the point that affordable "digital audio workstation" PCs are now available that can perform professional quality multi-track hard disk recording while running multiple instances of complex audio software. In short, musical analysis and processing algorithms have never been so commercially feasible or as applicable as they are today.

---

1. Analog multi-track recording systems can cost hundreds of thousands of dollars, wheras a fully equipped hard-disk recordingcomputer system can be had for under $10,000.

1.2 Motivation

Signal analysis is fundamentally scientific in nature. Pioneered by the mathematician Jean Baptiste Joseph Fourier and his transform [1], this discipline employs mathematical transforms which yield frequency-domain representations of time-domain signals that may be examined to gain insight into the signal's vital characteristics. This thesis describes the *Musical Spectrogram*, a signal analysis method based on an understanding of human auditory perception and designed to transform audio signals to a more intuitive and usable form for musicians and their applications.

While the intuitive visualization of musical signals remains the primary application of this method, it is also well suited for use as a front end processor for other algorithms which apply knowledge of harmonic structure and music theory in a human-like fashion to perform polyphonic pitch detection, chord recognition, instrument identification and musical scene analysis.

For most musical applications, the sampled signal describing sound pressure levels reaching the ear is not an appropriate representation for analysis. When plotted on an amplitude vs. time axis, this low level signal representation yields global attack and decay envelopes and amplitude maxima and minima. Figure 1.1 illustrates a simple musical signal as score notation, a MIDI event piano roll, and finally as a time-domain "wave" signal created from the MIDI events when performed by a synthesizer. The visual inspection of the sampled wave does not reveal which notes or how many are being played at any one time, nor can the observer derive the nature of any instrument(s) performing the part.

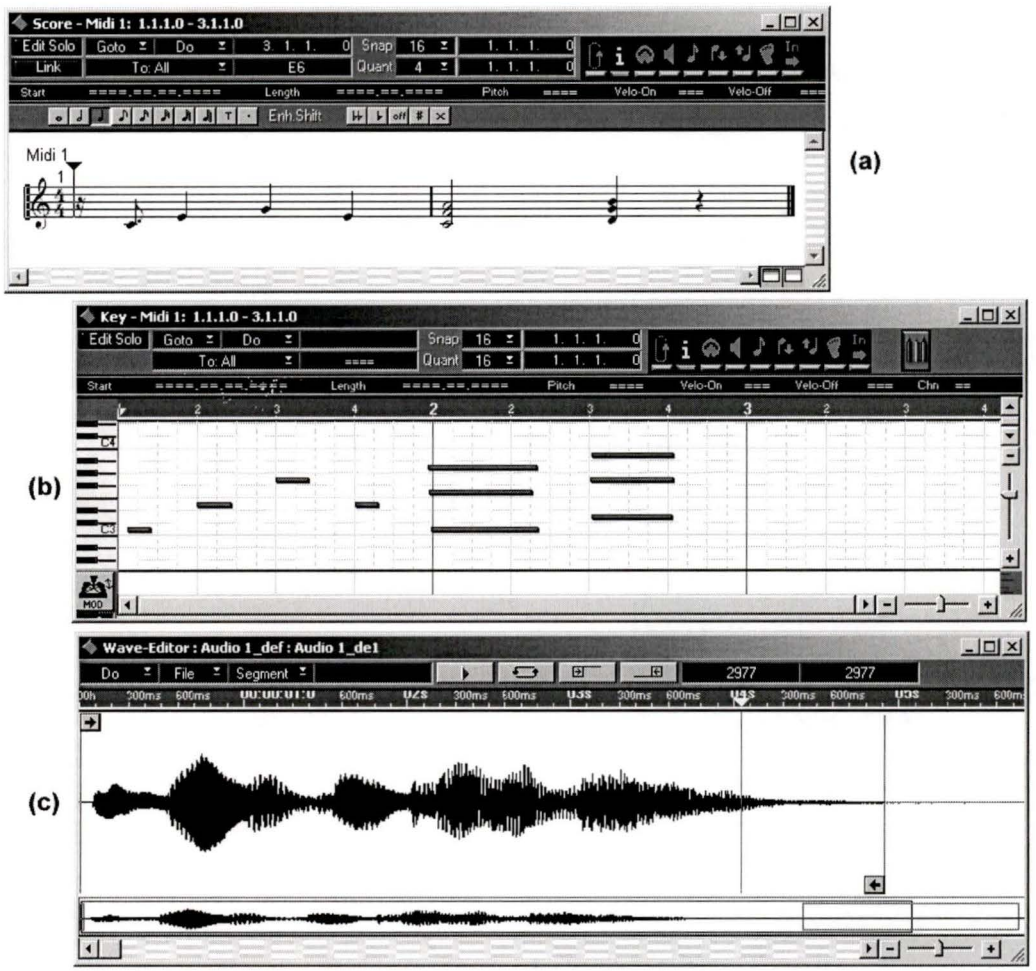


Figure 1.1 A passage of music represented as

(a) music score, (b) MIDI piano roll, and (c) as a sampled wave.

Human auditory perception can be classified as a sequence of representations from "low" to high" where low-level representations are roughly approximate to the sound signal reaching the ear, and high-level representations are those to which we have cognitive access. An example of a high-level representation would be: "A heavy metal guitar solo in a pentatonic scale plays over analog synthesizer chords in the key of A, while reverberant drums beating 4/4 time and a bass guitar riding the eighth notes back

up the sound." [2]. Between these two extremes are a multitude of "mid-level" representations which, as far as the mechanisms of the human brain are concerned, the scientific community currently knows very little about.

Despite difficulties in directly determining the mid-level representations at work within the human brain, a number of mid-level representations have been developed within the field of music signal processing in the last two decades. These are now summarized for the benefit of the reader.

Ellis and Rosenthal [2] presented a comprehensive tutorial paper on Mid-level representations for auditory scene analysis that details the following desirable properties in auditory mid-level representations:

- Sound Source Separation - The capability of discerning different sound sources.
- Invertibility - The property that the original signal may be recovered from the mid-level representation.
- Component Reduction - The reduction of many harmonic components to a single sound, much like the human ear which reduces sets of grouped harmonics into sources with a fundamental frequency and logical harmonic partials.
- Abstract salience of attributes - The features made explicit by a representation should approach the perceptual attributes of our final result.
- Physiological plausibility - To model and understand the auditory system it is important to respect the functional physiological knowledge derived from experimentation.

An analytic framework is put forward in this work in which auditory mid-level representations are classified according to three conceptual axes:

- The choice between fixed and variable bandwidth of the initial frequency analysis.
- Discreteness, corresponding to the degree in which the representation is structured.

- The dimensionality of the representation - Some posess a third dimension in addition to time and frequency.

Any given representation may be assigned a position on a cube as shown in Figure 1.2.

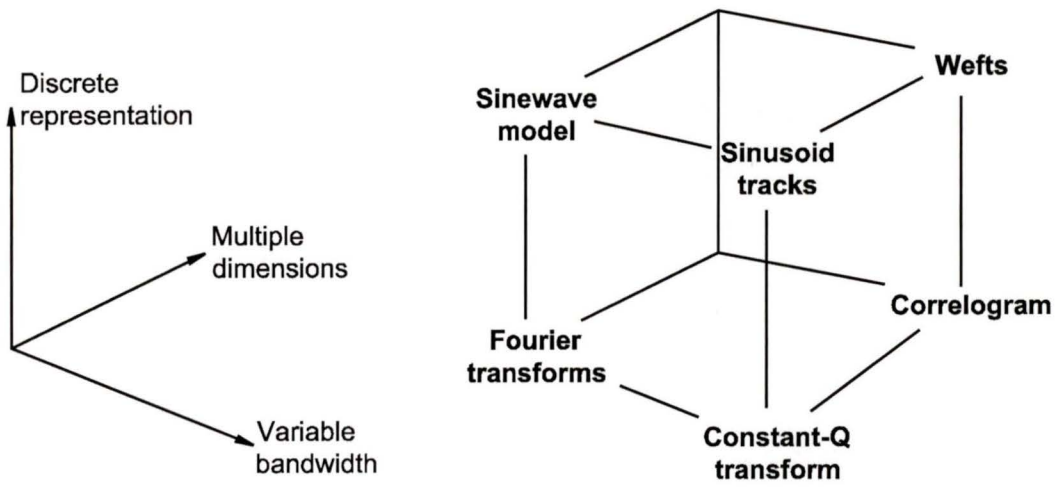


Figure 1.2 Analytic framework cube.

The fast and short-time Fourier transforms (FFT and DSTFT) are located in the front lower-left corner of the framework cube. Used to efficiently compute spectrograms of musical signals, Fourier techniques are limited by their fixed time-frequency resolution and were applied to the analysis and synthesis of musical signals by Piszczalski in 1981 [3], and by Strawn in 1985 [4]. The fixed time-frequency difficulty is overcome by moving along the variable bandwidth axis to the front lower-right of the framework cube, which is occupied by the constant-Q transform. Implemented as a bank of filters whose bandwidths vary in proportion to their center frequency, this technique yields an analysis quantitatively similar to that of the inner ear. Notable work employing this technique was done by Izmirli in 1999 [5] and by Nawab et. al. in 2001 [6]. Both of the preceding two representations are limited in that no higher level structure is derived from the signal. The representations on the top face of the framework cube are higher up the discrete representation axis than the FFT and constant-Q trans-

form and therefore consist of discrete analysis objects in addition to these two transforms. The front upper-left corner of the framework cube is occupied by sinusoidal modeling, in which music signals are modeled as a sum of evolving sinusoids (called partials) and combined with an additive noise component. These partials are based on frame-by-frame Fourier transformations. Notable examples of work applying this technique were published by Serra in 1997 [7] and by Virtanen and Klapuri in 2000 [8]. The front upper-right corner of the framework cube is occupied by the sinusoid tracks representation, which is employs the constant-Q transform to generate contiguous regions of local energy maxima in time-frequency. Sinusoid tracks were introduced by Ellis in 1993 [9]. Along the back face of the framework cube are representations that involve another dimension beyond the time and space. The back lower-right corner is occupied by the correlogram which was made popular by Slaney and Lyon in 1990 [10]. Based upon a pitch detector that mimics the human auditory perceptual system, the correlogram is computed by finding the short-time windowed autocorrelation of each output of a cochlear (inner-ear) model, and works for a wide range of pitch effects. In addition, it is robust against a wide variety of distortions, unlike analysis based on Fourier and constant-Q analyses. The third dimension in the correlogram is the lag time of short-time autocorrelations applied to the energy envelopes of each band. At the back upper-right corner is the weft representation which was put forth by Ellis and Rosenthal in 1995 [2], which forms regions of spectral dominance that evolve over time and serve as trajectories for the salient parameters of the correlogram.

The Musical Spectrogram presented in this dissertation is a new mid-level signal representation based upon the wavelet transform. It is easily generated from a low-level wave signal and yields a musically meaningful display of the signal's spectral energy across a multi-octave range subdivided into the 12 tone equal-tempered western music scale. From this mid-level representation, higher level information such as pitch, key, chordal and musical scene data can be derived via further information processing. While such algorithms are beyond the scope of this thesis, it is worthwhile considering

them when evaluating the utility of the Musical Spectrogram. Unlike many of the mid-level representations cited above, the Musical Spectrogram is only an analysis tool and does not explicitly seek to reconstruct the original signal from the derived parameters. Consequently, many of the requirements such as an orthogonal wavelet basis for perfect reconstruction (Section 2.4.1 on page 34) or linear-phase response (Section 1.3.1 on page 10) may be safely ignored, thus facilitating implementation. More advanced versions of the Musical Spectrogram will support signal resynthesis, but these fall under the category of future work.

1.3 Digital Filters

Filters are used to modify the frequency spectrum of a signal. While analog filters perform this function electronically, digital filters use finite-precision arithmetic operations such as multiplication and addition on an input sequence of digital samples (numbers) to achieve the desired spectral reshaping or modification. Not only are digital filters more reliable, stable and easily modified than their analog counterparts, the resources to implement them as real-time software are readily available on modern PC workstation computers.

There are four stages in the design of a digital filter:

1. Approximation

2. Realization

3. Analysis of computational error

4. Implementation

Complete coverage of the filter design process is detailed in [13]. Additional excellent sources of filter design and analysis theory are [14]-[16]. The approximation stage is driven by the application requirements of the filter, and involves determining the desired amplitude, phase and time-domain response specifications and generating a transfer function to meet these criteria. The realization stage is where the transfer function is converted into a related network topology of functional units such as unit delays, adders and multipliers. Many realization methods have been proposed which lead to structures of varying complexity and characteristics, some of which seek to minimize the number of components required, while others minimize the sensitivity of the filter network to the numerical errors introduced by finite-precision arithmetic.

The third stage of digital filter design involves examining the filter topology generated in the realization stage and determining its vulnerability to finite-precision errors.

Although filter coefficients generated during the approximation step are determined to a high degree of precision, affordable digital hardware typically operates with a finite precision dependent on the length of the registers used to store numbers. The type of number system used and the type of arithmetic supported (fixed-point or floating point) must be considered at this stage of the design. In cases where the filter is to be implemented in terms of hardware, all filter coefficients must be quantized to the precision of the target system and all changes (errors) to the proposed filter's transfer function (and therefore its amplitude and phase response) need to be examined fully to ensure the design will remain within specifications once implemented. Another important aspect of finite-precision target environments is that signals passing through them are also quantized, and considerable care must be taken to ensure that the resulting quantization noise has negligible impact on the performance of the filter. Thorough steps must also be taken to ensure that numerical overflows or underflows do not occur at any points within the digital filter network if it is to be implemented in a fixed-precision hardware environment. On modern PC audio workstations, however, software based processes enjoy high resolution floating-point numerical representations and operations, which frees this implementation from many of the error sensitivity issues detailed above.

Once analysis of the computational error stage is complete, the design may move to the final stage of implementation. The filter can be implemented in software to run on a general purpose processor such as a PC or Unix system, in firmware to run on a specialized DSP chip, or in dedicated hardware as a custom VLSI chip. There is also the trade-off between real-time and nonreal-time implementations to be considered within the software environment: Real-time implementations must be optimized for speed, while nonreal-time implementations have no such requirements.

In general, the time-domain relationship between the input and output sequences of a general linear time-invariant digital filter is given by

$$y(iT) = \sum_{j=0}^{n1} a_j x(iT - jT) - \sum_{m=1}^{n2} b_m y(iT - mT) \tag{1.1}$$

where T is the period between samples.

1.3.1 FIR Filters

1.3.1.1 FIR Time-Domain Representation

If all of the b_m coefficients in the second term of Equation 1.1 are zero, there is no feedback path in the filter's topology and the time-domain relationship simplifies to

$$y(iT) = \sum_{j=0}^{n1} a_j x(iT - jT) \tag{1.2}$$

meaning the filter is nonrecursive. Filters of this type can be shown to have a finite-duration impulse response and are therefore known as FIR filters.

The transfer function of a causal FIR filter is given as

$$H(z) = \sum_{n=0}^{N-1} h(nT) z^{-n} \tag{1.3}$$

An FIR filter with an impulse response of length N can have linear phase and constant group delay provided its impulse response is symmetrical about the midpoint between samples $(N-2)/2$ and $N/2$ for even N, or about sample $(N-1)/2$ for odd N. These linear-phase filters are advantageous in that they do not introduce phase distortion into the signals they process and that their symmetrical coefficient characteristic may be exploited to reduce the number of multiplications required by the filter. Because there is no feedback loop in FIR filters (by definition), they cannot become unstable and are usually insensitive to quantization errors. In addition, their realizations

are simple, regular, and easily implemented as high-level language software, DSP assembly code firmware or VLSI hardware.

On the downside, the lack of feedback in FIR filters means that their $N-1$ poles are fixed at the origin of the z plane: High-selectivity can only be achieved with large values of N, along with the associated penalty of increased processing delay and implementation cost. However, the implementation efficiency of FIR filters can be improved by using fast-Fourier transforms, owing to their finite duration impulse response.

1.3.1.2 FIR Design Techniques

The approximation problem for FIR filters can be solved by using windowed Fourier series or numerical-analysis formulas such as the *Stirling* central-difference formula.

Although these closed-form solutions are easy to apply and involve low orders of computation to derive, they lead to suboptimal designs in which the efficiency and speed of filter operation could be improved. Optimization techniques [13] based on the second method of Remez [17] address this issue: From an initial FIR design achieved by closed-form approximation, an error function is formulated for the desired filter and then minimized with an iterative procedure known as the Remez exchange algorithm [18] that weighted-Chebyshev optimizes a linear combination of cosine functions that control the error in different bands of interest. When the algorithm converges the error function becomes equiripple, as in other types of Chebyshev solutions. A more robust Remez method that is capable of designing multiband filters and other filter types is presented in [19].

1.3.2 IIR Filters

1.3.2.1 IIR Time-Domain Representation

When at least one of the b_m coefficients in the second term of Equation 1.1 is not zero, this indicates a feedback path in the filter's topology. The output of this filter is computed from the input samples and previous output samples, which makes the filter recursive. Although recursive filters that have finite-duration impulse response can be realized, no practical advantage appears to be gained in using these filters instead of nonrecursive ones. Consequently, for most practical purposes, recursive filters have an infinite-duration impulse response and are also known as IIR filters.

The transfer function of a causal IIR filter obtained by using the z transform is

$$H(z) = \frac{\sum_{n=0}^{N} a_n z^n}{\sum_{m=0}^{M} b_m z^n} \tag{1.4}$$

where $N \leq M$.

For stability to be assured, the poles (roots of the denominator) must lie within the unit circle of the z plane. Although this restriction is neccessary, it should be noted that there is otherwise considerable freedom in the placement of the poles. This fact can be exploited to yield high selectivitities that are impossible to achieve in FIR filters of equivalent order because their poles are restricted to the origin of the z plane. On average, comparisons have found that FIR filters require 5 to 10 times the filter order to achieve the same selectivity as IIR filters of similar order (p. 305 in [13]). In one case, an FIR filter of order 52 was found to be as selective as an IIR filter of order 8.

The increased selectivity of IIR filters does not come without a cost, however. As will be shown in the next section, only one of the five major analog filter approximations places any contraints on the group delay characteristic, which means that most IIR filters based on these analog prototypes have a varying group delay and consequently introduce phase distortion into the signals they process. While phase corrective solutions do exist for IIR designs [20], these involve the design of a phase equalizer filter of equivalent order which can more than double the cost and processing delay of the IIR filter. For high-selectivity applications where the delay characteristic is of secondary importance, IIR designs are the correct choice.

1.3.2.2 IIR Design Techniques

The approximation problem in IIR digital filters is handled by converting one of five of the most frequently used analog-filter approximations into corresponding discrete-time transfer functions. Each of these approximations brings the following specific characteristics to IIR digital filters based on them:

1. Butterworth: Monotonic passband and stopband, the simplest approximation.
2. Chebyshev: Equiripple passband but monotonic stopband.
3. Inverse-Chebyshev: Equiripple stopband but lopsided passband.
4. Elliptic: Equiripple passband and stopband, the most efficient approximation.
5. Bessel: Monotonic passband and stopband but an approximately constant group delay.

While these approximations lead to complete solutions of the transfer function in closed-form and lead to very efficient and precise designs as a consequence, they are only applicable to the design of filters with piecewise-constant amplitude responses. Filters with arbitrarily shaped passband and stopband characteristics are typically

designed through optimization methods in which a discrete-time transfer function is assumed and an error function is formulated based on the differences between the actual and the desired amplitude and/or phase response. When this error function is iteratively minimized with respect to the transfer function coefficients, the resulting amplitude and/or phase response of the filter approaches the desired results. This technique is covered in numerous texts, but [13] is recommended to the interested reader. Novel optimization methods for such filters are introduced in [21].

1.4 Multirate Systems and Filter Banks

Traditional single-rate digital signal processing systems are schematically represented as interconnections of computational blocks such as multipliers, adders, and delay elements. In Figure 1.3, a canonical second-order recursive digital filter implementation is schematically represented as a network of 2 unit delays, 2 adder and 5 multipliers.

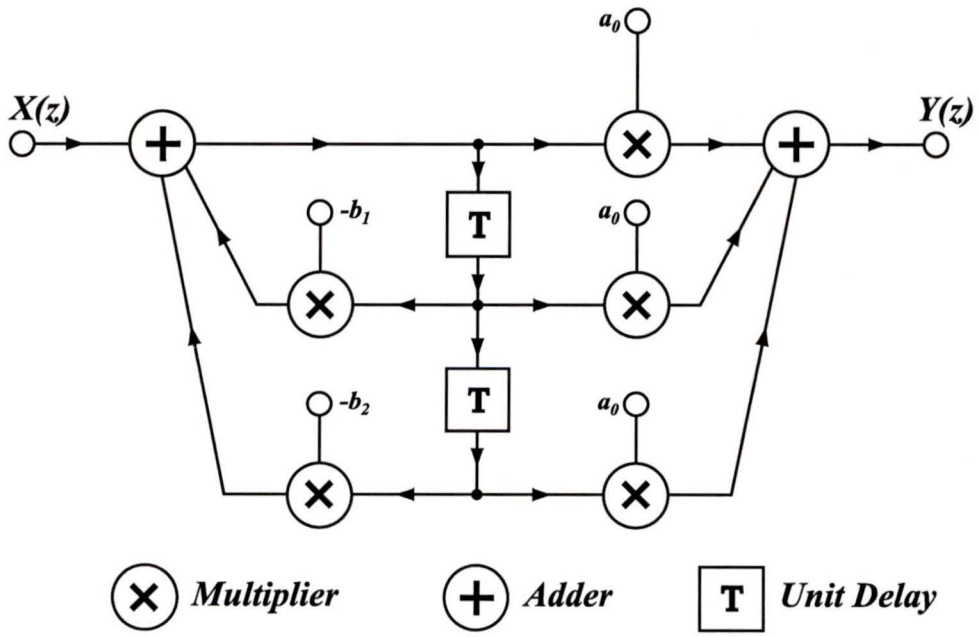


Figure 1.3 A canonical second order IIR digital filter implementation.

Multirate signal processing systems feature two more building blocks in addition to those shown in Figure 1.3. These change the sampling rates of signals that pass through them and are called the *M-fold downsampler* and the *L-fold upsampler.* They are also represented by the symbols shown in Figure 1.4.

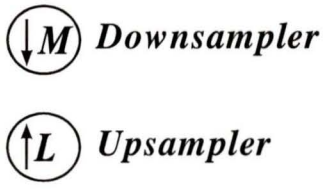


Figure 1.4 Schematic symbols for downsampler and upsampler blocks.

A downsampler is a device that reduces the sampling rate of a signal by an integer factor of M, while an upsampler is a device that increases the sampling rate of a signal by an integer factor of L. Complete coverage of multirate signal processing can be found in [13], [22] and also in the foundational text [23]. Relevant topics are now covered here.

1.4.1 Downsamplers

The M-fold *downsampler* takes input sequence $x(n)$ and produces output sequence

$$y_D(n) = x(Mn) \tag{1.5}$$

where M is an integer. Only samples that are multiples of M are allowed to pass through the downsampler. This process divides the sampling rate by a factor of M and therefore multiplies the sampling period T by M. An example of a continuous signal, its sampled representation and the resulting signal after downsampling by a factor of 2 are shown in Figure 1.5.

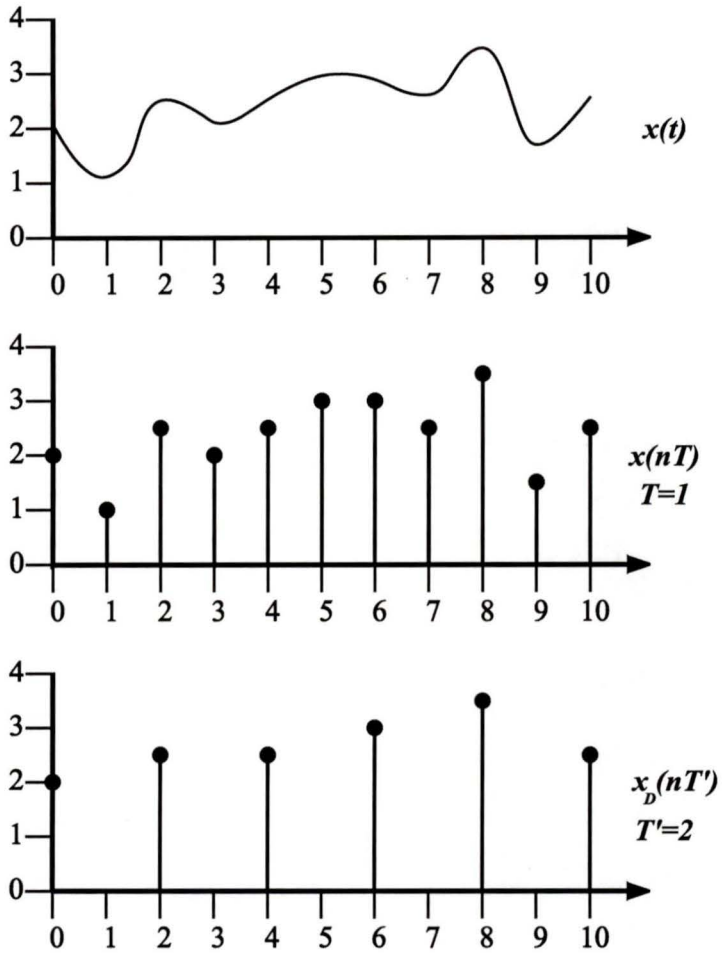


Figure 1.5 A downsampled signal when M = 2.

A bandlimiting filter must preprocess any signal prior to downsampling in order to avoid signal degradation in the form of *aliasing*. This filter is called a *decimation filter*, and should accompany the downsampler (unless the signal is already bandlimited) before it can be correctly called a decimator. This requirement is now explained.

1.4.2 The Sampling Theorem

Shannon's sampling theorem [24] states that when a continuous time signal is periodically measured (or sampled) at sampling rate f_s, every frequency component of the original signal is periodically replicated over the entire frequency axis with period given by f_s.

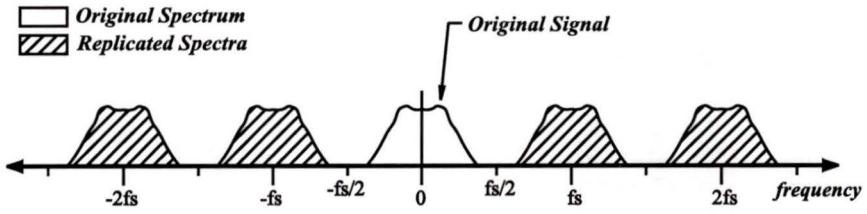


Figure 1.6 Spectrum replication caused by sampling.

Figure 1.6 demonstrates how the replicated spectra will not overlap the original spectrum provided it is bandlimited within 0 and $f_s/2$. The sampling theorem states that for accurate representation of a signal $x(t)$ by its time samples $x(nT)$, two conditions must be met:

- The signal $x(t)$ must be bandlimited: Its frequency spectrum may only contain frequencies up to maximum frequency f_{max}.
- The sampling rate f_s must be chosen to be at least twice the maximum frequency f_{max}: $f_s \geq 2f_{max}$. This is also referred to as the *sampling theorem rule.*

If these conditions are not met, the spectral replicas will overlap resulting in aliasing, where spectral components above $f_s/2$ are reflected into the baseband spectrum resulting in distortion.

The minimum sampling rate allowed by the sampling theorem is therefore $f_s = 2f_{max}$. Whenever a signal is downsampled, its sampling rate f_s is reduced, which lowers the Nyquist frequency $f_s/2$. Any signal must therefore be bandlimited

within this lowered Nyquist interval prior to downsampling to ensure its frequency spectrum is not overlapped (or aliased) by the spectral replicas generated as a consequence. Figure 1.7 illustrates the aliasing introduced when $f_s' = f_s/2$.

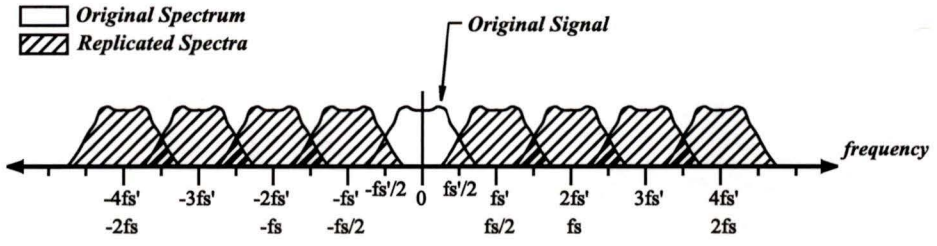


Figure 1.7 Aliasing caused by downsampling a non-bandlimited signal.

1.4.3 Decimators

Because the sampling theorem requires that a signal to be bandlimited prior to downsampling, a downsampler will always be preceded by a lowpass filter to meet this requirement. Such a combination of processing elements is referred to as a *decimator* and is illustrated in Figure 1.8.

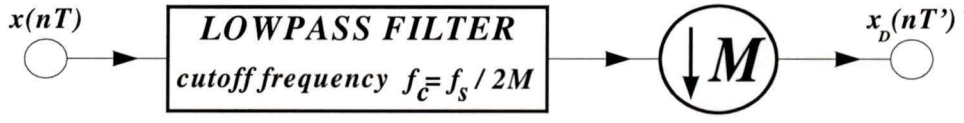


Figure 1.8 An M-fold decimator.

1.4.4 Upsamplers

The L-fold upsampler takes an input sequence $x(n)$ and produces the output sequence

$$y_E(n) = \begin{cases} x(n/L) & \text{if n is an integer multiple of L} \\ 0 & \text{Otherwise} \end{cases} \tag{1.6}$$

where L is an integer.

All samples of the signal $x(n)$ are therefore present in the expanded signal $y_E(n)$, and are separated by zero-valued samples. For this process to be correctly called interpolation, the upsampler must also be followed by a lowpass filter to replace the zero-valued samples with interpolated values. Figure 1.9 demonstrates a signal $x(nT)$ that is upsampled by a factor of two to yield the signal $x_U(nT')$ and then lowpass filtered to produce interpolated signal $x_I(nT')$.

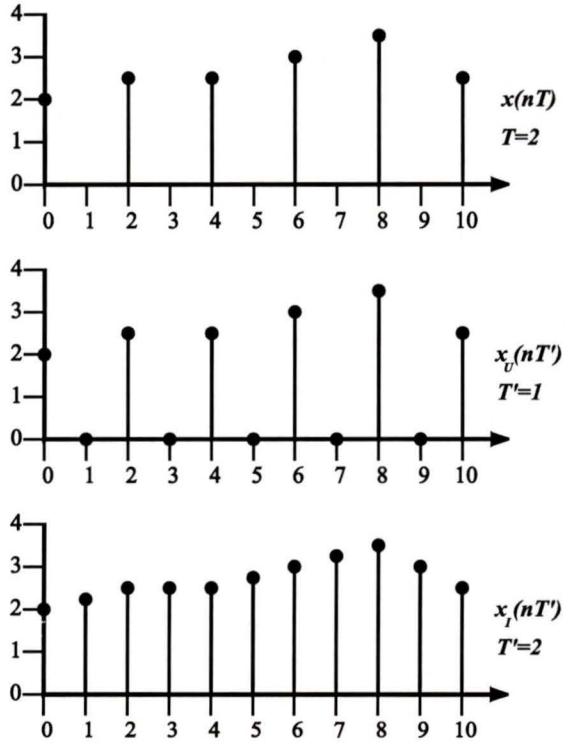


Figure 1.9 Upsampling and interpolation for L = 2.

1.4.5 Interpolators

One way to view the requirement of lowpass filtering in the time domain is to consider that the abrupt transitions between all zero valued samples and their neighbors

manifest as unintended high-frequency components in the upsampled signal. This is another example of aliasing, and is visually demonstrated in Figure 1.10, where the bandlimited and sampled signal of Figure 1.6 is upsampled by a factor of 2.

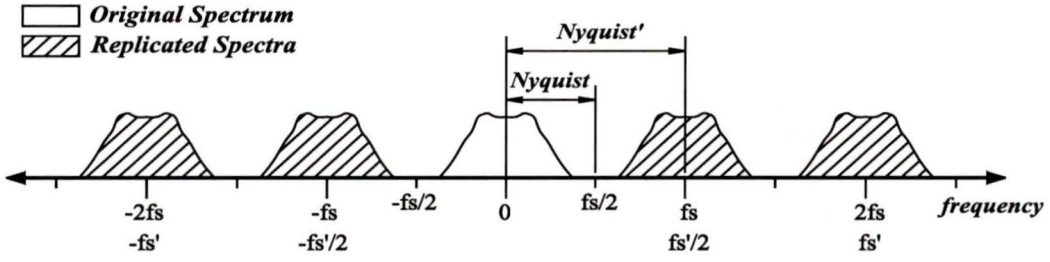


Figure 1.10 Aliasing caused by upsampling a signal.

As can be seen, doubling the sample rate ($f_s' = 2f_s$) of the signal also doubles the Nyquist interval which includes the lower half of the spectral replica centered around $(f_s')/2$. Unless the upsampler is followed by an *interpolation filter* to remove the unwanted replica, the signal will remain aliased. Such a combination of processing elements is referred to as a *interpolator* and is illustrated in Figure 1.11.

Figure 1.11 An L-fold interpolator.

1.4.6 Rational Sampling Frequency Conversion

There are many applications where it is necessary to change the sampling rate of a signal by a non-integer factor. In such cases, a cascade arrangement of an L-fold interpolator followed by an M-fold decimator will implement a rational sampling rate conversion of $R = L/M$.

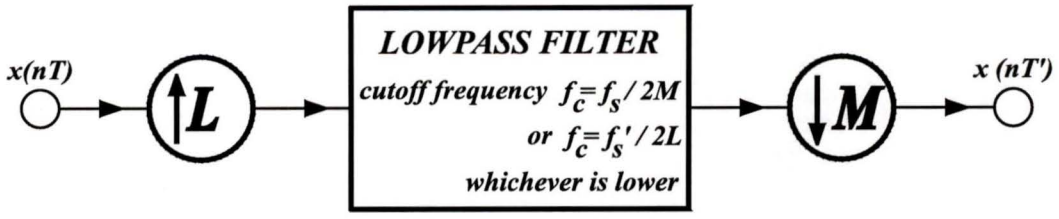


Figure 1.12 L/M rational sampling rate converter.

When an interpolator and decimator are combined in this way, one of the included filters will have a higher cutoff frequency and may be eliminated from the design as shown in Figure 1.12. A more efficient implementation can be found in [22].

1.4.7 Digital Filter Banks

Digital filter banks are networks of digital filters with a common input or output. Both cases are shown here:

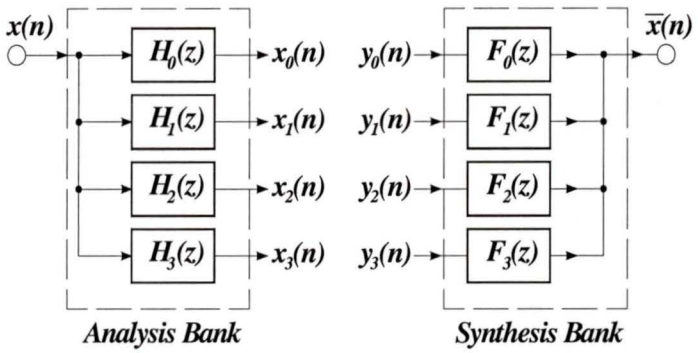


Figure 1.13 Analysis and synthesis digital filter banks.

The *analysis bank* is composed of four digital filters characterized by the transfer functions $H_0(z)$ to $H_3(z)$ (for the sake of simplicity, these filters will be represented by the symbols H0 to H3). Typically, analysis filters are designed to have mutually exclusive passbands. This implements a filter bank that splits an incoming signal $x(n)$ into 4

subband signals $x_0(n)$ to $x_3(n)$ which may be processed, coded and transmitted independently depending on the application. The *synthesis bank* shown above is composed of four digital filters F0 to F3. These synthesis filters are application dependent, but typically bandlimit their respective subband signals $y_0(n)$ to $y_3(n)$ to their prescribed frequency ranges before they are combined into the output signal $\bar{x}(n)$.

Filter banks are not restricted to single-rate digital signal processing. In fact, most modern analysis/synthesis filter bank systems reduce the bandwidth and processing requirements of each subband signal through decimation in analysis and interpolation during synthesis. These techniques and their application to the Musical Spectrogram are covered in more detail in the next chapter.

1.5 Musical Signal Analysis

It is important to distinguish the analysis of musical signals from the science of signal analysis itself. Signal analysis can be described as the mathematical study and modification of continuous and discrete variables of all dimensions, while musical signal analysis involves the application of signal analysis techniques to the study of musical sound. On first glance, it is tempting to see the latter as merely a subset of the former, but this is not the whole story, as will be shown.

As a science, signal analysis has many of its roots in the 16th century when the scientists of that time were attempting to create mathematical descriptions of the physical phenomena around them: Early curve fitting and interpolation algorithms provided ways to bring recorded data into the realm of mathematics for analysis and processing, which in turn hastened the development of the calculus in the 17th century. In 1807, the science of signal analysis entered a new stage with the introduction of the Fourier series. Within the Fourier series, a signal is broken down into sinusoids of different frequencies, and thereby transformed from the time-domain to the frequency-domain.

Many of Fourier's contemporaries (Laplace, Lagrange and Poisson to name only a few) found the concept of expanding functions into trigonometric series extremely difficult to accept, and actively resisted its introduction [25]. As a consequence, this mathematical gateway to the frequency-domain went mostly unappreciated by 19th century scientists. In the twentieth century, the Fourier transform gained acceptance and was in common use by physicists and mathematicians by the end of the Thirties. The signal processing community's complete adoption of Fourier analysis occured when the fast-Fourier transform (FFT) was put forth in 1965 [26]. This highly efficient numerical algorithm made the analysis of analog and digital signals in the frequency-domain a practical computational reality, and Fourier analysis became commonplace. Over the four decades since, the science of signal analysis has driven the development of real-time digital signal processing applications and accelerated communications and media technology in countless ways.

Musical signal analysis also has a history: One that is older and more subjective, but with a legacy no less profound than that of the science of signal analysis. Every musical instrument ever made owes its design technology to the analysis of musical sound, as does every aspect of music theory. It follows therefore that formal music itself would not exist without the analysis of musical sound, for its composition is based on music theory and its tonal library on the designed properties of musical instruments. The elements of musical signals will now be detailed, along with theory and formulae that represent them.

1.5.1 Musical Pitch

Pitch is the musical term analogous to the scientific frequency of a periodic sound. Music is made from notes, and the primary note parameter is pitch. An established musical standard is *A-440* (or *concert tuning*), which equates the frequency 440 Hz to the musical note A located above Middle C on the piano keyboard, as shown by Figure 1.14.

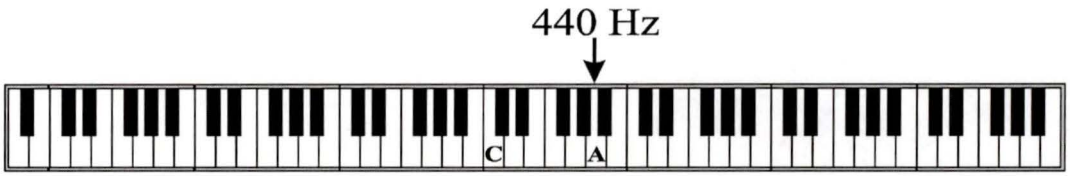


Figure 1.14 A-440 on the piano keyboard.

Human auditory perception is not linear: Amplitudes are perceived in terms of decibels and pitches in terms of octaves, both of which are logarithmic in nature [27].

The relationship between octaves and frequency is defined in terms of powers of 2. When midddle A is tuned to 440 Hz, the A note one octave higher corresponds to 880 Hz, while the A note one octave lower is 220 Hz. The octave is the common pitch interval of every human musical system, including the Western musical scales, the Asian, the Indian, the Arabic and Persian scales, as well as the tribal music of aboriginal cultures.

1.5.2 Fundamental Frequency and Harmonics

While periodic musical sounds are considered to have a singular pitch, they are actually composed of more than one harmonically related frequency. Called *harmonics*, these frequency components are multiples of the fundamental frequency f_0 that vary in amplitude and phase. Sounds with intense high-frequency harmonics sound bright or shrill to the ear, while those with dominant low-frequency harmonics sound dull and dark by comparison. Although the *pitch* of a periodic musical sound depends on its fundamental frequency, the *timbre* of the sound depends on the number and characteristics of the harmonics accompanying the fundamental. According to the terminology of music theory, harmonics are often referred to as the *overtones* of a sound. When the fundamental frequency component is included with the harmonics for consideration, they are referred to as the *harmonic partials* of the sound, where the fundamental is the

1st harmonic partial and so on.

Pitch perception is a psychoacoustic phenomenon and it is not always directly related to fundamental frequencies. For example, there are sounds whose harmonic structure suggests a fundamental that is not apparent, and in these cases, the ear "hears" a pitch corresponding to this missing component. This is referred to as *virtual pitch*, and is considered to be the main mechanism for pitch perception by several well respected researchers [29]. Further evidence supporting this theory comes from the observation that there is an entire class of sounds that are not periodic by any scientific measure but are still perceived to have a pitch. Bell tones, bands of noise and twanging sounds are some examples of this class. The human ear can apparently analyze all of the harmonic partials within a mixture of diverse sources, group the harmonically related partials into separate sources and then assign a pitch to the logical fundamental for each source. When a fundamental is present but no harmonics exist (as in sine waves, for instance), human pitch perception is deprived of harmonic cues and the ear is less certain to place the pitch and more likely to confuse which octave it belongs to.

It is natural that the octave serves as the fundamental interval of music. If a given tone has a fundamental frequency *f*, then its harmonics are *2f*, *3f*, *4f* and higher. Meanwhile, the same tone one octave higher at frequency *2f* has harmonics at *4f*, *6f*, *8f* and higher. Every harmonic partial in the higher tone also exists in the lower tone, and they are therefore perceived as virtually identical by the human ear.

1.5.3 Consonant Intervals, Scales and Tunings

While the octave is the fundamental musical interval, the partitioning of this interval into *tones* is what defines a scale. For a scale to be useful, the tone intervals must be defined to facilitate the construction of harmonic structures whose frequency ratios are pleasing to the ear. Such intervals are said to be *consonant* intervals. The ancient Greek conviction that ratios govern beauty and science has roots in the work of

Pythagoras, who was one of the first scientists of musical sound. Pythagoras studied the length and pitches of plucked strings and found that all the consonant intervals to be ratios of whole numbers. Some of these consonant intervals according to the modern 12 semitone scale, tuning and terminology are shown in Figure 1.15.

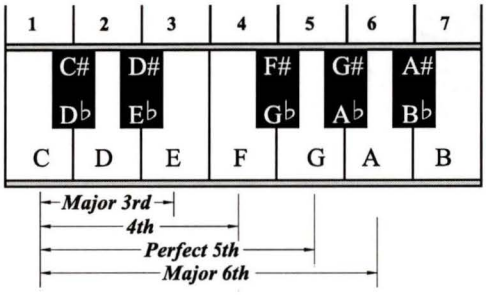


Figure 1.15 Consonant intervals of C according to the modern scale.

Pythagoras determined that 3/2 and 4/3 ratios of string length produced the most consonant intervals (these are perfect fifths and fourths according to modern musical terminology, as shown in Figure 1.15). He then devised a 7-tone scale to facilitate the production of these favored intervals. Unfortunately, ideal or perfect intervals with frequency ratios defined by small integers such as 3/2, 4/3, 5/3, 5/4, 6/5 and 8/5 cannot all be exactly realized in any one scale or system of tuning, and Pythagorean tuning did not have the capacity to produce major thirds (5/4). This led to the development of later tunings that favor certain consonant intervals at the expense of others. *Ptolemaic* (or *just*) tuning favors major thirds (5/4) and major sixths (5/3), at the expense of one of the fifths, while *mean-tone* tuning favors major thirds. The fifths in mean-tone tuning are still flattened, but this error is distributed better than Ptolemaic tuning allows.

Each of these three tunings implement a 7 note scale built from tones of varying width that favor certain intervals but discriminate against others. Such scales may only be used in specific musical keys, and even then certain note combinations are unacceptable because the harmonics produced are sufficiently out of tune as to sound dreadful.

1.5.4 The Equal Tempered Scale

An ideal musical scale must satisfy two opposing criteria:

- True intonation: Notes exist that match the exact thirds, fourths, etc.
- Free modulation: Music may be transposed between all keys without any change to its harmonic properties.

Pythagorean and Ptolemaic tunings address the first criteria at the expense of the second and were used from ancient Greece until the late 15th century when Mean-tone tuning appeared. Mean-tone tuning also favors true intonation over free modulation, but its emphasis is on thirds instead of fifths. According to [30], Mean-tone tuning encouraged composers to use major and minor thirds, to avoid the open perfect fifths commonplace in Medieval music, and to stay within the first four steps of the central key. Mean-tone tuning changed the rules of composition for the Baroque and Renaissance periods and consequently redefined the music as well.

A new tuning was developed during Johann Sebastian Bach's lifetime (1685-1750) that allowed composers and musicians to work in all 12 keys and did not make certain triads of notes unplayable. The *well-tempered* tuning was defined for free modulation and instrumental compatibility, but also retained minute concessions to intonation. Each key in well-tempered tuning therefore has its own character, and Bach capitalized on these differences when he wrote *The Well-Tempered Clavier* in all 24 major and minor keys [30]. Well-tempered tuning was used during the 18th and 19th centuries in various forms, although gradually the intervals became more equal in tuning and less concessionary to intonation.

By the 20th-century, progressively accurate tuning techniques based on frequency measurement led to the *equal tempered* tuning. This scale is fully optimized for free modulation and makes no special concessions any particular interval, consonant or otherwise. The equal tempered scale has been the Western standard ever since, although

there are those who argue that it renders the color of every musical key as uniform grey.

The equal tempered scale divides each octave into 12 semitone notes of logarithmically equal width, and the 7 tones inherited from earlier tunings are built from them. Given a semitone at frequency f_0 and its next higher octave at $2f_0$, the center frequencies of the equal tempered semitones between them are given in Figure 1.16.

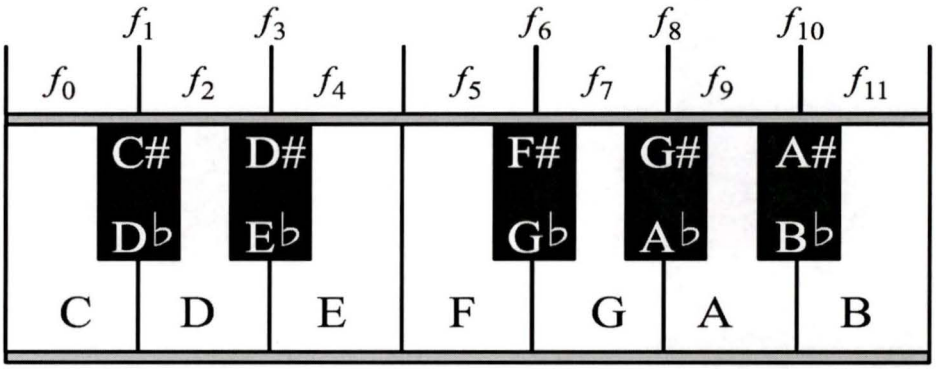


Figure 1.16 The equal-tempered 12-tone scale.

Frequencies f_0 to f_{11} are given as:

$$f_i = f_0 2^{\frac{i}{12}} \text{ where } i = [1,11] \tag{1.7}$$

Note that these frequencies denote the *center* of each semitone. The equal tempered scale subdivides each semitone into 100 cents which are symmetrical about the center frequency f_i such that:

$$f_i = \begin{cases} f_{(i-1)} + 50 \text{ cents} \\ f_{(i+1)} - 50 \text{ cents} \end{cases} \tag{1.8}$$

Each octave is therefore divided into 1200 cents.

1.5.5 The Frequency Spectrum of Music

While the frequency spectrum of electromagnetic signals extends up into the Tera-

Hertz range, audio signals occupy the baseband defined by the limits of human hearing from about 20 Hz to around 16 kHz. The functional upper frequency limit of commercial music is set at 22.05 kHz by the Compact Disc sampling rate of 44.1 kHz. Signal analysis based on musical note information has a funadamental frequency range of interest one fifth the size of the CD signal baseband. Figure 1.17 (below) shows the useful frequency range of the piano and other extreme range instruments when the equal tempered scale is tuned to A-440.

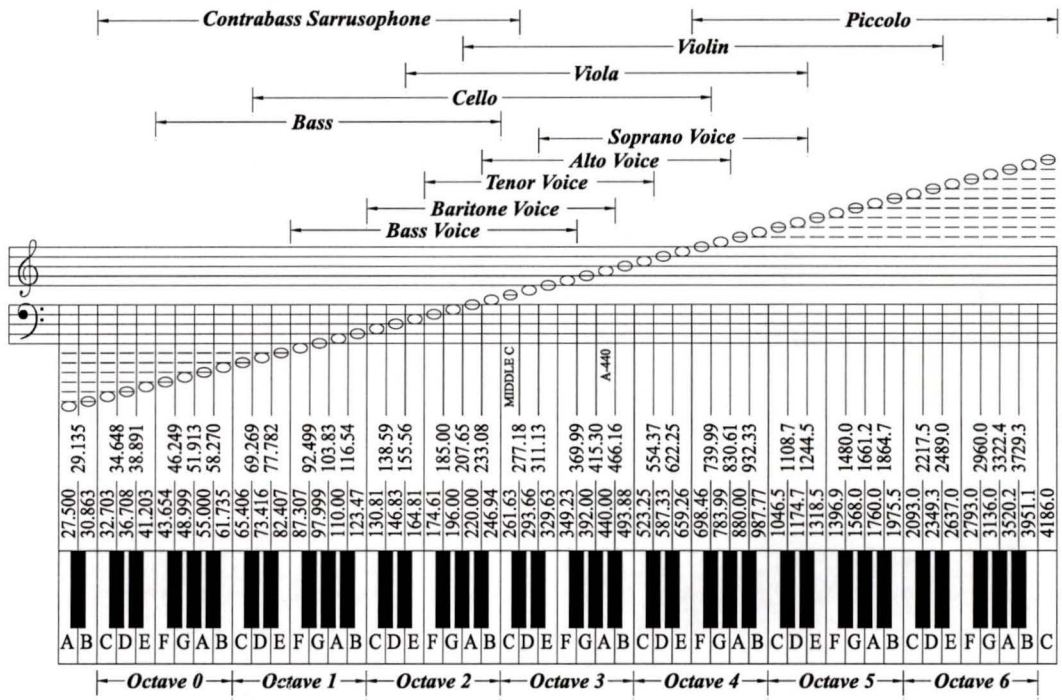


Figure 1.17 The piano keyboard: Frequency intervals and instrumental ranges.

The highest note playable by piano and piccolo corresponds to 4.186 kHz, which suggests an upper frequency bound only 20 percent that of the whole audio spectrum. This figure only charts the fundamental frequencies produced by these instruments, however and not the harmonics that are also produced. These issues are now explored.

1.5.6 Harmonic Partials of Interest

The lowest note of the piano has a fundamental frequency of 27.5 Hz, which means there could be up to 500 audible harmonic partials associated with this single note. While comprehensive study of the partials required by the human ear is beyond the scope of this work, an intuitive grasp on the subject may be gained by examining the harmonics of three different piano notes spanning the keyboard (see Figure 1.18). Each DFT plotted in Figure 1.18 was computed from the first 16384 samples of each representative sound. At 44.1 kHz, the DFT analysis window therefore spans the first 372 milliseconds, covering the so-called *attack* of each note. Each x-axis grid is set for each DFT plot to match the harmonic partial frequencies for the analyzed note, and the following is apparent:

- C0 ($f_0 = 32.703\text{Hz}$) has notable overtones at $f_0 - f_5$, with lesser overtones clustered higher at progressively smaller amplitudes. Paradoxically, there is no f_0 detected here, which undescores the importance of harmonics other than the fundamental for piano tones in this frequency range.
- C3 or middle C ($f_0 = 261.63\text{Hz}$) has notable overtones at $f_0 - f_6$ .with the last clear overtone at f_{14} .
- C7 ($f_0 = 4168.00\text{Hz}$) exists almost entirely at the fundamental.

It can be seen from this limited data that low pitched piano sounds are composed of more harmonic partials than midrange piano sounds, while the highest pitched piano sounds have only one or two harmonic partials. This perception is supported by studies of the response characteristics of the human cochlea's basilar membrane (summarized in [27] and [10]), which find that human hearing is unable to distinguish between harmonic partials separated by more than a *critical bandwidth*. The critical bandwidth begins at 100 Hz and gets progressively wider up the piano keyboard scale to high frequencies (see Figure 1.19). It is natural that the piano's tonal structure fits the model of human hearing. Like all musical instruments, the piano was designed for human enjoyment. In summary, harmonic partials are very important to the perceptions of pitch and

timbre at lower frequencies. This importance diminishes for progressively higher pitched sounds, to the point where only the fundamental is relevant at the highest notes.

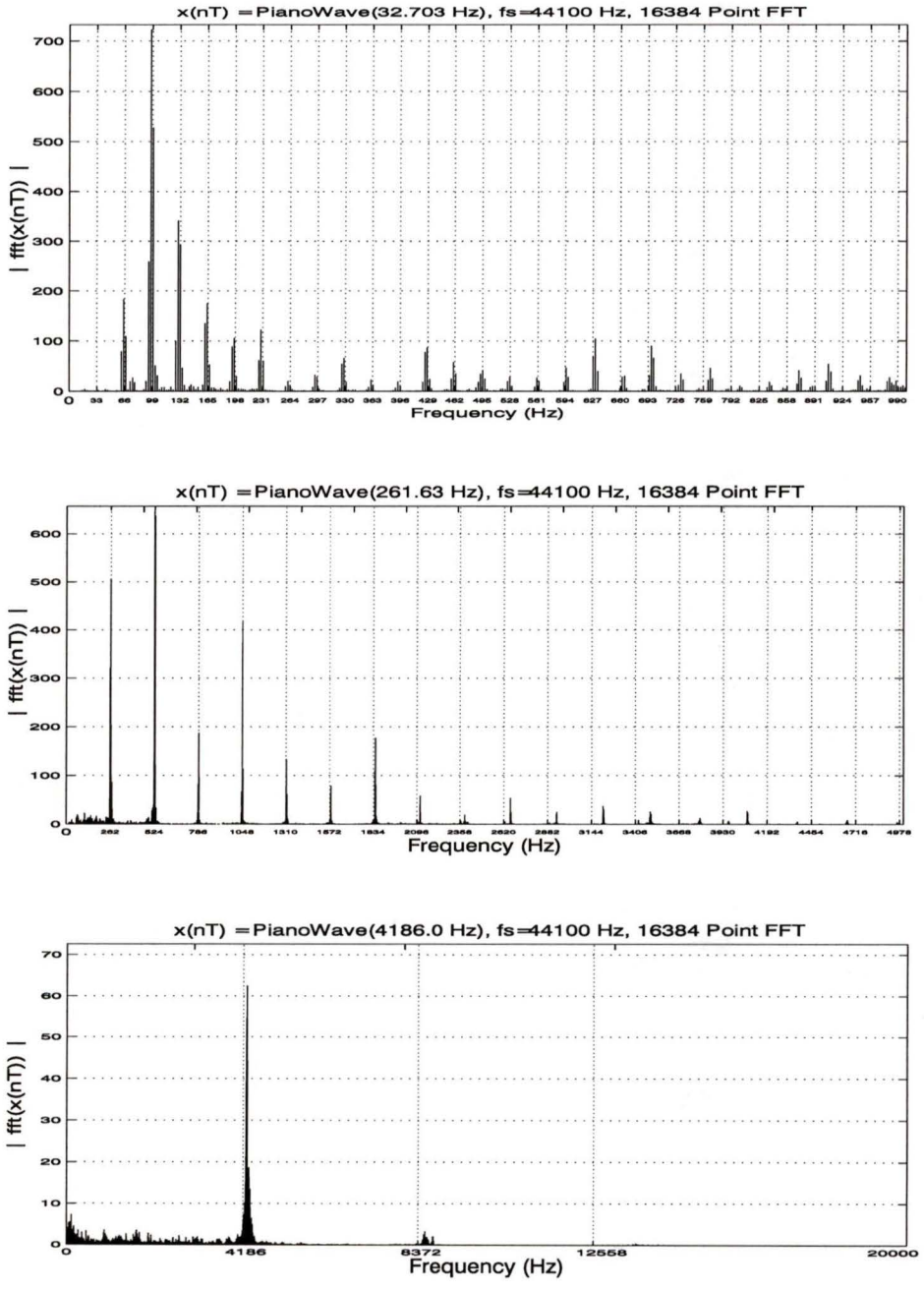


Figure 1.18 The harmonic analysis of C0, C3 and C7.

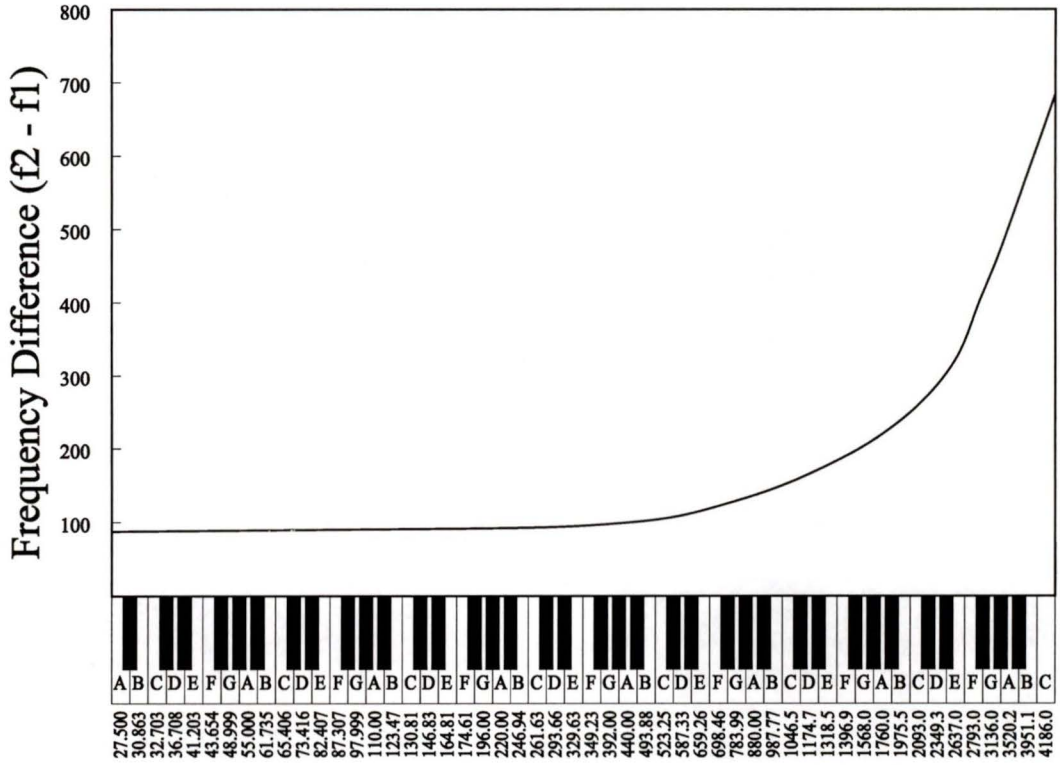


Figure 1.19 The critical bandwidth of human hearing.

1.5.7 Nonharmonic Partials

There exist spectral components related to the fundamental frequency by non-integer factors, and these are called *nonharmonic partials*. While nonharmonic partials are the main constituents of bell, chime and gong sounds, they also make up a noticeable part of certain periodic musical signals and must always be considered by musical analysis systems. Any system that resolves musical signals into their harmonic partials will also resolve any nonharmonic partials present, so care must be taken when the partials are sorted and grouped by higher level pitch and timbral detection algorithms. Musical sounds are not guaranteed to have only harmonic components.

1.5.8 Stochastic Components

Musical analysis-synthesis systems based on sinusoidal modeling [8], [33] and [34], represent signals as the sum of a deterministic component (consisting of all harmonic and nonharmonic partials), and a stochastic component which is a broadband noise-like process. The presence of such a component in music signals is not surprising since the breath noise in woodwinds, the bow noise in strings, and other fast attack transients with noise-like characteristics are common in musical sounds. Since the energy of these components is distributed noise-like across a wide frequency spectrum, their contributions to any one frequency band or harmonic are safely ignored by analysis systems like the Musical Spectrogram.

1.6 Musical Spectrogram Applications

The primary goal of mid-level music signal representation is to build a computational model that can understand musical audio signals in a human-like fashion. The motivations is to produce music technologies that are intuitive for musicians and therefore facilitate the creative process. The Musical Spectrogram represents the frequency content of music signals as an array of time varying subband signals, each of which maps to a semitone of the equal tempered scale; this suggests a number of musical applications which are now described.

1.6.1 Wave Editor Visualization

An entire class of music software exists for editing and applying effects to digital signal data. *Wave editors* allow the user to select blocks of sample data from audio and music tracks for selective processing such as pitch correction, dynamics compression, equalization etc. As shown in Figure 1.1 on page 3, wave data does not provide the

user with many cues as to its musical content other than amplitude envelopes, and the Musical Spectrogram would provide a musical perspective to such applications in the form of a scrolling view track synchronized with the currently displayed sample data.

1.6.2 Partial Synthesizer Plug-in Instruments

Partial synthesis is an analysis/synthesis system based on sinusoidal modeling that uses the Musical Spectrogram as its front end processor. Partials detected by successive analysis frames are grouped into trajectories which drive sinusoidal generators to create synthetic sounds based on the original. Unlike sinusoidal modeling, which seeks to recreate the original signal, partial synthesis is meant to be an audio synthesis and special effect engine. The primary application for such an engine would be use in *plug-in instruments*, which are software based music applets that plug into digital multi-track recording and MIDI sequencing programs such as **Cubase** [35] and **Sonar** [36].

1.6.3 Musical Pitch Detection Software

While robust pitch detection methods are common for monophonic musical signals, polyphonic signals have proved very difficult to process reliably. Many promising polyphonic pitch detector algorithms have been put forward that use human pitch perception models as their front end [32], and the Musical Spectrogram is also of this type. Since it produces analysis data that corresponds to musical terminology, the Musical Spectrogram is also compatible with *blackboard* software algorithms [32] that integrate various forms knowledge for the purpose of solving ill-posed problems. For handling the ill-posed problem of polyphonic pitch detection, many forms of musical and harmonic knowledge must be applied heuristically to the analysis data. It is therefore a great advantage that no translation is required between the Musical Spectrogram and blackboard components of any proposed polyphonic pitch detection system.

1.6.4 Automatic Transcription

The goal of producing an automatic transcription system that extracts note pitches and onset/offset times from any polyphonic music signal has been pursued by musicians and engineers for over 25 years. Should a working polyphonic pitch detector be developed with the Musical Spectrogram, one of the major applications would be transcription software capable generating of musical scores and MIDI sequence data from individual instrument recordings. Such software would detect key, scale, chord progression and tempo information from the notes being scored, providing automatic classification services for music catalogues and features for generating simpler forms of musical notation such as guitar tablature and so-called *fake books*. Such documents allow musicians with poor sight-reading skills to play popular music without any form of preparation, and are in great demand.

1.6.5 Musical Scene Analysis

Musical scene analysis involves the separation of musical signals into their constituent instrumental sources such as bass, piano, strings and drums. This problem is even more intractable than polyphonic pitch detection in that sophisticated timbral analysis is required to recognize and separate instrumental sounds that span similar frequency ranges. Some commercial uses of such musical scene analysis algorithms are automatic content classification for musical catalogues, multi-instrumental automatic transcription systems, and as intelligent front ends for advanced CD mastering and live performance equalization systems.

1.7 Thesis Organization

The design and implementation of the Musical Spectrogram requires diverse knowledge from the fields of engineering, musical theory and psychoacoustics. Chapter 1 provided a complete overview of these theories and facts while explaining both the histories of signal analysis and of musical sound.

Chapter 2 develops the background of wavelet theory sufficently to introduce dyadic multi-rate filter bank designs that implement cost-effective implementations of the discrete wavelet transform. With this design, audio signals may be decomposed into octave sub-bands that are analogous to the logarithmic frequency spacing of the human auditory system.

In Chapter 3, the architecture of the Musical Spectrogram is detailed. There are four stages to the design:

1. Tuning resampler,
2. 9 octave band filter bank,
3. 12 semitone band filter bank array, and
4. threshold detector

Each of these stages is proposed and then developed into a functional design.

Unlike the Fourier spectrogram, which plots signal spectra along a linear frequency scaled x-axis, the Musical Spectrogram plots signal spectra according to an equal-tempered 12-tone musically scaled y-axis. This makes it a powerful analysis tool for musicians who lack the sufficient scientific background to understand or easily appreciate the analysis tools available to scientists and engineers. The Musical Spectrogram is also very efficient. The design is *critically sampled*, which means that every stream of data processed and output by this algorithm is at the lowest possible sampling rate permitted by the sampling theorem. This maximizes the processing speed and minimizes the buffering requirements without any degradation in analysis quality. The architecture of the Musical Spectrogram is also conceptually elegant. A total of three digital filter designs provide the customized bandlimiting and separation required to separate a musical audio signal into 108 perfectly tuned semitone signals.

The main disadvantage inherent in the Musical Spectrogram proposed here is that it does not have a synthesis component. Any parameters derived by this analysis can not be amalgamated to reconstruct the original audio. This is a consequence of the use of elliptic filters in the Musical Spectrogram design. This was deemed to be an acceptable trade-off, given the far greater selectivity and efficiency of this filter type.

Chapter 4 covers the functional implementation details of the Muscial Spectrogram. The first implementation is complete MATLAB simulation which serves as the

proof-of-concept and also as the complete implementation of an offline batch processing system. A full suite of test signals were applied to the MATLAB Musical Spectrogram and their results plotted and discussed. Many of the musical theory introduced in chapter 1 are found to be relevant to the interpretation of these experimental results, and therefore they are discussed and possible solutions and enhancements to the Musical Spectrogram are reviewed.

The second implementation of the Musical Spectrogram is a fixed-point DSP hardware implementation on the Analog Devices ADSP-2181 Fixed-point processor. Although incomplete, this second implementation provided many examples of sample-processing optimizations of the Musical Spectrogram that were neccessary given the practical constraints of this real-time system. Among the new theories developed for this implementation are a filter-coefficient scaling algorithm that groups transfer function poles by their proximity and reorders the stages of processing in such a way that signal overflow and underflow cannot occur. Since fixed-point DSPs are cheap and universally used, this single innovation is of considerable value.

Chapter 5 summarizes the more important results presented here, and suggests a new implementation of the Musical Spectrogram as a real-time software module running within the graphical control language environment called Infinity. The benefits and capabilities of audio software 'plugins' are also presented here and a plugin version of the Musical Spectrogram is proposed.

Chapter 2

The Discrete Wavelet Transform

2.1 Introduction

The Fourier transform computes analysis coefficients which are inner products of the signal with sinusoidal basis functions of infinite duration. This analysis is based on the assumption that the analyzed signal is stationary, and therefore has statistical properties that do not change over time [37].

A version of the Fourier transform adapted for nonstationary signals was introduced by Gabor in 1946 called the short-time Fourier transform (STFT) [38], which introduces a time dependent parameter τ. The STFT depends on a window function $g(t)$ of limited extent over which the nonstationary signal is assumed to be stationary. This limitation makes the STFT inappropriate for musical signals, as they are typically composed of nonstationary low-frequency components of long duration (bass notes, for example) and nonstationary high-frequency components of short duration (such as attack transients and percussion).

A more recent transform was introduced in 1989 whose window function $g(t)$ varies in width and amplitude across the frequency axis. The *continuous wavelet transform* (or CWT) [42] employs a scale factor a in addition to time parameter τ. This transform is well suited to frequency-dependent nonstationary signals.

Discretization of the CWT a and τ parameters yields the *discrete wavelet trans-*

form (DWT), which is applicable to digitally sampled signals. While direct computation of the DWT is possible, there are much more efficient implementations based in the construction of iterated filter banks. These discrete-time case methodologies were developed separately and independently of wavelet theory, but lead naturally to DWT designs which are critically sampled (i.e. require only a minimum number of samples) and easily implemented as *DWT filter banks*. DWT filter banks are specialized quadrature mirror-image filter (QMF) multiresolution filter banks with dyadic tree structures. A DWT filter bank provides octave decomposition within the Musical Spectrogram; a general treatment of wavelet theory supporting the DWT is therefore presented. The ordering and development of these formulae and theories is based on tutorial papers [39] and [40].

2.2 Nonstationary Signal Analysis

The Fourier transform [1] computes analysis coefficients $X(f)$ as inner products of the signal with complex sinusoidal basis functions of *infinite* duration:

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-2j\pi ft}dt \tag{2.1}$$

Since the basis functions employed by the Fourier transform are infinite in duration, they are not well suited for the representation of nonstationary or statistically time-dependent signals. Any abrupt changes in the analyzed signal are erroneously spread out on the entire frequency axis in $X(f)$. One way to modify the Fourier transform to allow the meaningful analysis of nonstationary signals is to introduce time dependency to the transform without sacrificing linearity. This is achieved by introducing a "local in time" parameter t which governs a windowing operation on the transformation process, allowing multiple "local" Fourier transforms to be evaluated across the frequency axis. Each of the local Fourier transforms treats the signal as stationary for the length

of the window about the τ parameter.

2.3 Scale Versus Frequency

2.3.1 STFT: Fixed Resolution Analysis

The modified Fourier transform described above is formally known as the ***short-time Fourier transform (STFT)*** [38], and is given by:

$$STFT(\tau, f) = \int_{-\infty}^{\infty} x(t)g^*(t-\tau)e^{-2j\pi ft}dt \tag{2.2}$$

Some inspection of the STFT formula reveals that the window function $g(t)$ is shifted by the τ parameter to allow a time-localized Fourier transformation to be performed centered at time location τ. The parameter f is similar to the Fourier frequency and many properties of the traditional Fourier transform hold true for the STFT but the general STFT characteristics are dependent on the choice of window function $g(t)$. There are two alternate ways to view the STFT's effect on input $x(t)$:

- A succession of Fourier transforms of a windowed signal, or
- a modulated analysis filter bank.

Both of these views are illustrated in Figure 2.1.

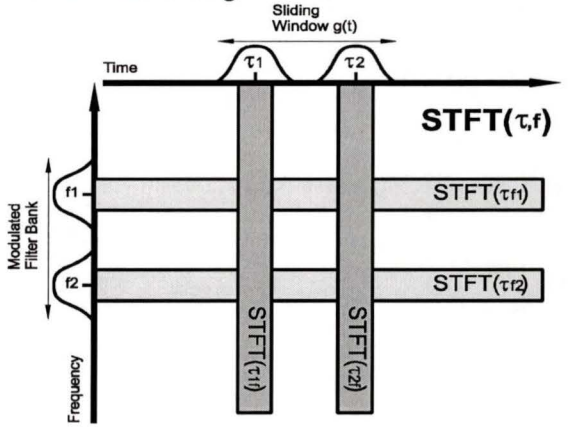


Figure 2.1 The two alternate views of the STFT.

When the time parameter τ is held constant and the frequency parameter f is varied, the STFT appears as a traditional Fourier transform of a windowed version of $x(t)$ centered about time τ (illustrated as the vertical bars $STFT(\tau 1f)$ and $STFT(\tau 2f)$ in Fig. 1). Conversely, when τ is varied and f is held constant, the STFT appears as a bandpass filter modulated to frequency f (the horizontal bars $STFT(\tau f1)$ and $STFT(\tau f2)$) [38]. From this dual interpretation of the STFT, a fundamental drawback related to its time and frequency resolution can be developed. Given a window function $g(t)$ and its corresponding Fourier transform $G(f)$, the RMS bandwidth of the window's transfer function Δf is given by

$$\Delta f^2 = \frac{\int_{-\infty}^{\infty} f^2 |G(f)|^2 df}{\int_{-\infty}^{\infty} |G(f)|^2 df} \tag{2.3}$$

while the window's spread in time is given by

$$\Delta t^2 = \frac{\int_{-\infty}^{\infty} t^2 |g(t)|^2 dt}{\int_{-\infty}^{\infty} |g(t)|^2 dt} \tag{2.4}$$

From these expressions, it can be seen that the frequency resolution of the STFT is given by Δf and that the STFT's time resolution is given by Δt. It follows intuitively then, that any two frequencies closer than Δf to each other in the frequency domain or closer than Δt to each other in the time domain will ***not*** be distinguishable from each other.

2.3.2 Bounded Time-Bandwidth Product

Heisenberg's uncertainty principle states that given a particle's uncertainty of its x-position Δx and the uncertainty of its momentum Δp, the product of these two variables can *never* be less than a number of the order of Planck's constant, i.e.

$$\Delta x \Delta p \geq \frac{h_{bar}}{2} \tag{2.5}$$

Specifically, the uncertainty principle implies a trade-off between Δx and Δp, in that one quantity cannot be reduced without a corresponding gain in the other. The uncertainty principle applies to the time-frequency resolution of the STFT in the following way:

$$\Delta t \Delta f \geq \frac{1}{4\pi} \tag{2.6}$$

Since the time-bandwidth product $\Delta t \Delta f$ is bounded from below, Heisenberg's uncertainty principle states that the time resolution of the STFT is only enhanced at the expense of its frequency resolution, and vice-versa.

Once a window is specified for the STFT, the time-frequency resolution is fixed over the entire time-frequency plane since the same window is merely shifted for use at all frequencies in this transform. In order to minimize the time-bandwidth product $\Delta t \Delta f$, Gaussian windows are often chosen since they meet Equation (2.6) with equality [38].

2.3.3 CWT: A Multiresolution Analysis

Although perfectly acceptable in some cases, the fixed time-frequency resolution of the STFT is a significant drawback. This is because most practical signals are composed of high-frequency components of short duration and low-frequency components of long duration. When analysis is viewed in terms of a filter bank, it is understood that the successively higher-frequency filters require successively higher time resolutions. To meet this requirement, the frequency bandwidth Δf is defined to be proportional to f:

$$\frac{\Delta f}{f} = c \tag{2.7}$$

where c is a constant. This requirement results in a transform which can be viewed as a filter bank with constant relative bandwidth ("constant-Q") in which the bands of the analysis filters are evenly spaced in a logarithmic scale along the frequency axis. Human auditory perception also follows a log-2 spacing, so this is a fortuitous development towards a transform adapted to the analysis of musical signals.

When the requirement of Equation (2.7) is satisfied, Δt and Δf still satisfy the Heisenberg inequality of Equation (2.6), but the analysis now exhibits a time resolution that improves at higher frequencies and a frequency resolution that improves at lower frequencies. The transform under development must do more than simply shift the frequency spectrum of the window function $g(t)$ along the frequency axis, however. A changing frequency bandwidth proportional to the analysis frequency requires that the

time-domain window function *contract* at higher frequencies and *dilate* at lower frequencies.

The CWT implements this property by defining a *mother wavelet* $h(t)$, which is shifted, compressed and dilated by scaling to create the individual window functions $h_{a,\tau}(t)$ for each analysis filter evaluated by the transform

$$h_{a,\tau}(t)\frac{1}{\sqrt{|a|}}h\left(\frac{t-\tau}{a}\right) \tag{2.8}$$

The complete CWT [42] is therefore given by

$$CWT_x(a,\tau) = \frac{1}{\sqrt{|a|}}\int_{-\infty}^{\infty} x(t)h^*\left(\frac{t-\tau}{a}\right)dt \tag{2.9}$$

2.4 Wavelet Analysis and Synthesis

Wavelet analysis via the CWT formula of Equation (2.9) results in a set of wavelet coefficients which indicate how close the signal $x(t)$ is to the set of *basis functions* defined by the mother wavelet $h(t)$. Any signal can therfore be represented as a decomposition of wavelets of constant shape but differing size and amplitude. The flexibility inherent in the CWT is that the choice of mother wavelet (and therefore set of basis functions) is up to the user of the transform. The criteria of this choice is now discussed.

2.4.1 Orthogonal Basis Functions

The wavelets $h_{a,\tau}(t)$ generated by the mother wavelet $h(t)$ must behave like an ***orthogonal basis*** in order for the CWT to be effective for most analysis and resynthesis tasks. Generally speaking, the decomposition of a signal onto an orthogonal basis guarantees that two fundamental conditions have been met:

1. The signal's content is perfectly partitioned into a set of sub-signals (basis functions) without any data redundancy between them, and

2. all of the sub-signals must be included in the resynthesis (or inverse-transform) operation for the original signal to be perfectly reconstructed.

Since the CWT is defined for continuously varying a and τ parameters, the functions $h_{a,\tau}(t)$ are in fact very redundant, and so the first condition cannot be fulfilled by any choice of mother wavelet. Fortunately, any basis generated by the CWT is said to *behave* as an orthogonal basis if the following energy reconstruction formula holds true:

$$x(t) = c \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} CWT(a,\tau) h_{a,\tau}(t) \frac{da\,dt}{a^2} \qquad \begin{array}{c}\text{where c is a constant, and} \\ a > 0\end{array} \tag{2.10}$$

This reconstruction formula is always satisfied whenever the mother wavelet $h(t)$ is of finite energy and has a bandpass characteristic in the frequency domain. Furthermore, if the mother wavelet is further assumed to be sufficiently regular, the energy reconstruction condition simplifies further to

$$\int_{-\infty}^{\infty} h(t)dt = 0 \tag{2.11}$$

It is important not to confuse the perfect reconstruction of a signal's energy described in Equation (2.10) and Equation (2.11) with the perfect reconstruction of the signal as a whole that is guaranteed by a *true* orthogonal basis: The generation of these will be covered in the case of the DWT in following sections.

2.5 Discrete Wavelet Transform

When two scales $a_0 < a_1$ roughly correspond to two frequencies $f_0 > f_1$, the wavelet coefficients at scale a_1 can be downsampled at $(f_0/f_1)^{th}$ the rate of the coeffi-

cients at scale a_0, according to the sampling theorem rule. The parameters a and τ are therefore discretized according to the dyadic sampling grid shown in Figure 2.2 [43].

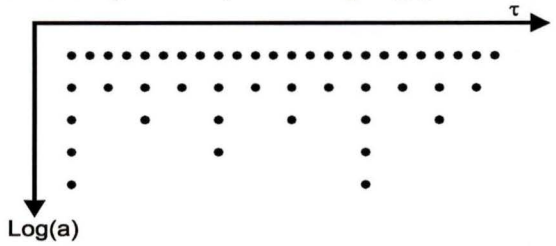


Figure 2.2 Dyadic sampling grid of the DWT.

The discretization of a and τ of the DWT are mathematically given by:

$$\begin{aligned} a &= a_0^{\,m} \\ b &= n(a_0^{\,m})T \end{aligned} \tag{2.12}$$

Where m and n are integers and T is the sampling period. This definition leads to wavelets defined by

$$h_{m,n}(t) = \frac{1}{\sqrt{a}} h\left(\frac{1}{a}t - b\right) \tag{2.13}$$

with corresponding wavelet coefficients

$$c_{m,n} = \int_{-\infty}^{\infty} x(t) h^*_{m,n}(t) dt \tag{2.14}$$

If $a_0 \approx 1$, and T is sufficiently small, the resulting DWT will approximate the CWT nearly perfectly, and the CWT's non-restrictive conditions on the mother wavelet (Equation (2.11)) will apply. On the other hand, if a_0 is chosen to be a more useful value such as 2 (to allow octave by octave computation of the wavelet coefficients), the sampling is said to be sparse, and a true orthonormal basis is only possible by careful choice of the mother wavelet.

2.5.1 Signal Scale, Decimation and Interpolation

In the context of wavelet analysis, the scale parameter ranges from small to large, where smaller scales of analysis mean contracted wavelets (and consequently greater signal detail) in the time domain, and larger scales mean dilated wavelets operating on *subsampled* signals (global views of signal subbands).

The effects of halfband filtering, decimation and interpolation on the signal's scale and resolution are as follows:

1. Halfband filtering results in the same number of samples representing half of the original frequency bandwidth. Scale is unchanged but resolution is halved.

2. Halfband filtering followed by subsampling by two (2-fold decimation) results in half the original number of samples representing half of the original frequency bandwidth. Scale is doubled but resolution is still halved.

3. Upsampling by two followed by halfband filtering (2-fold interpolation) results in twice the original number of samples representing half of the original frequency bandwidth. Scale is halved but resolution is unchanged (The upsampling process doubles the resolution, but the halfband filtering halves it.).

These effects and the half-band decimation property shown above are exploited in ***multiresolution pyramid*** [44] and ***subband coding*** [45] schemes to produce efficient digital filter structures naturally suited to implementing DWT analysis.

2.6 Multiresolution Pyramid

Consider a discrete-time signal $x(n)$ which is downsampled by 2 and halfband filtered. The signal's scale is doubled by the downsampling process, while its resolution is halved by the lowpass filtering process. The resulting signal $y(n)$ is given by

$$y(n) = \sum_{k=-\infty}^{\infty} g(k)x(2n-k) \tag{2.15}$$

where $g(n)$ is the impulse response of the halfband filter.

In order to obtain approximation $a(n)$ of the original signal from $y(n)$, we must first upsample it by a factor of two and then halfband filter the upsampled signal to remove any aliasing introduced. This signal is therefore given by

$$a(n) = \sum_{k=-\infty}^{\infty} g'(k)y'(n-k) \tag{2.16}$$

where $g'(n)$ is the impulse response of the second halfband filter and $y'(n)$ is the expanded version of $y(n)$. If both halfband filters characterized by impulse responses $g(n)$ and $g'(n)$ [1] are assumed to be ideal (i.e. passband amplitude=1 and stopband amplitude=0), $a(n)$ would be a perfect halfband lowpass approximation to $x(n)$.

Once the lowpass halfband approximation of $x(n)$ has been computed, a difference signal $d(n)$ can be simply derived by subtracting the sample values of $a(n)$ from the sample values of $x(n)$ as

$$d(n) = x(n) - a(n) \tag{2.17}$$

This results in a decomposition of $x(n)$ into a halfband approximation $a(n)$ and an additional detail signal $d(n)$ which can be added to reconstruct $x(n)$.

This scheme is relevant to the implementation of the DWT because of two notable features:

---

1. These filters are henceforth represented by symbols g and g', respectively

1. $a(n)$ is synthesized from the downsampled signal $y(n)$, which has double the scale and half the resolution of the original (hence the term ***multiresolution***).

2. $d(n)$ can be downsampled by a factor of two without penalty since it is a *highpass* halfband approximation of $x(n)$, resulting in a second double scale / half resolution signal.

The multiresolution scheme described does not downsample $d(n)$, so one stage of this scheme leads to both a half-rate low-resolution signal and a full rate difference signal, increasing the number of samples by 50%. Figure 2.3 gives an example of one stage of this scheme.

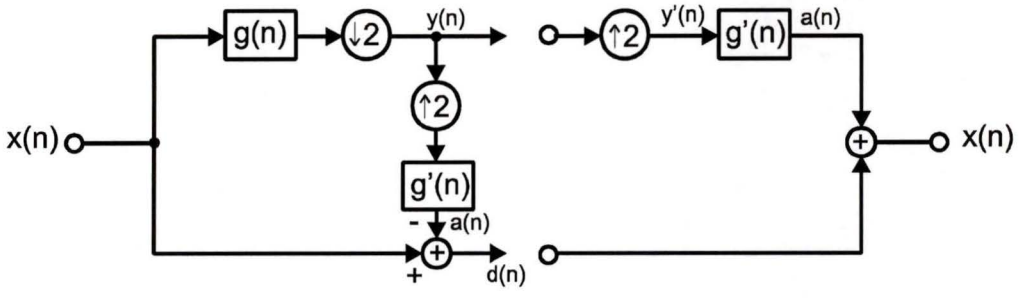


Figure 2.3 One stage of multiresolution scheme.

This scheme can be iterated on $y(n)$ to create hierarchically shorter subband signals. Such multistage schemes are called signal or image ***pyramids*** [44] because of this successive stacking of shorter signals.

2.7 Subband Coding via the QMF Filter Bank

Although the multiresolution pyramid scheme lends itself to implementation of the DWT, it is characterized by a redundancy in the sample data brought about by the fact that the difference signal $d(n)$ is not sampled at the lowest possible frequency by the sampling theorem rule. We now examine a scheme that has no such flaws.

Subband Coding schemes [45] obtain a lowpass halfband downsampled version of $x(n)$ exactly as the multiresolution pyramid does, but the highpass halfband component of the signal is derived instead by a highpass filter h characterized by impulse response $h(n)$ in parallel with lowpass filter g characterized by impulse response $g(n)$. The highpass halfband signal is then downsampled by two, exploiting its lower bandwidth according to the sampling theorem rule. Figure 2.4 gives an example of one stage of this scheme.

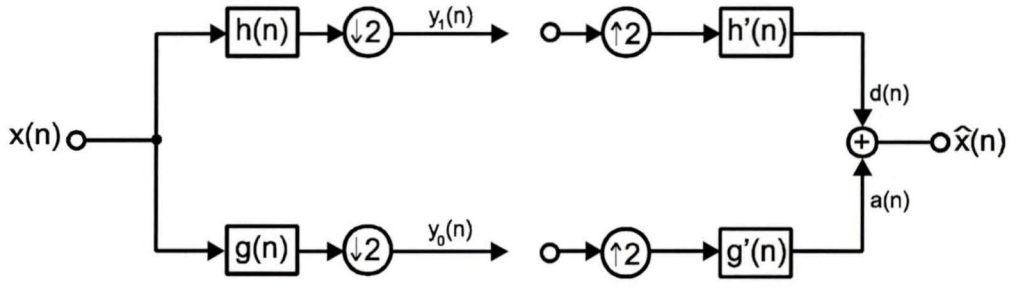


Figure 2.4 Single stage subband coding scheme.

If the assumption is made that h and g are ideal halfband filters (highpass and lowpass respectively), the single stage subband coding scheme is one step of a DWT decomposition since the original signal is decomposed into two subsignals at twice its scale. In addition, the ideal characteristics of these halfband filters ensure that the DWT implemented is *identical* to the CWT.

Since ideal filters do not exist in practice, there exist restrictions on the filters h, g, h' and g' that must be met before the output signal $\hat{x}(n)$ can be identical to $x(n)$. Filter bank structures which meet this criteria are said to exhibit the ***perfect reconstruction*** property.

A popular type of subband coding structure with the potential for perfect reconstruction is the quadrature mirror-image filter bank, which is identical to the subband coding scheme of Figure 2.4 with the following time-domain restrictions on filters h, g, h' and g':

$$h(n) = (-1)^n g(n)$$
$$h'(n) = -[(-1)^n g(n)]$$
$$g'(n) = (-1)^n h(n)$$

In the z-domain, these restrictions impose the following constraints on the transfer functions of filters h, g, h' and g':

$$\begin{aligned} H(z) &= G(-z) \\ H'(z) &= -G(-z) \\ G'(z) &= H(-z) \end{aligned} \tag{2.18}$$

In Equation 2.18, $H(z)$ is the transfer function of a lowpass halfband filter with even length N and a symmetrical impulse response.

2.8 DWT Filter Bank

Having established that a single stage subband coding scheme such as the two-channel QMF filter bank (with perfect reconstruction) implements one step of a wavelet decomposition, we would like to extend the scheme to an arbitrary number of steps in such a way that implements the complete DWT [22]. Such a multistage structure is possible by iterating the two-channel QMF structure on the lower halfband signal as shown in Figure 2.5 :

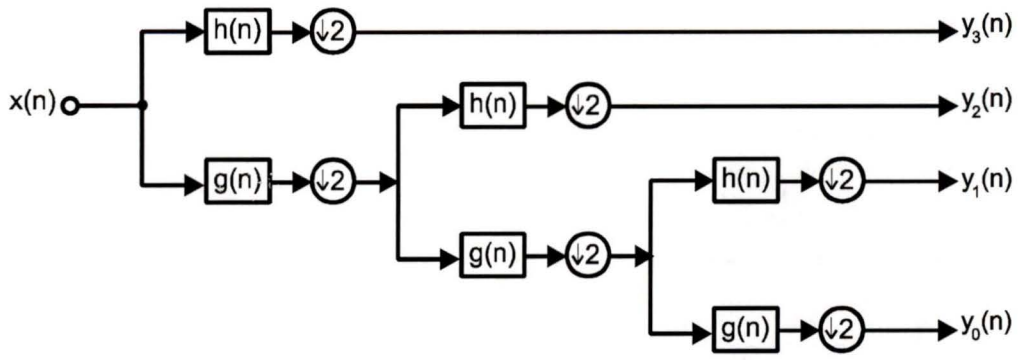


Figure 2.5 Dyadic tree QMF filter bank: 4 octave wavelet decomposition.

The first iteration of this structure creates a new low-band that corresponds to the lower quarter of the frequency spectrum of $x(n)$; each further iteration is characterized by a low-band which has a width that is half of the previous iteration's lowband, and a high-band which corresponds to the difference between the previous iteration's low-band and the current one (effectively a passband). The frequency bands that result from a structure such as the dyadic tree structured QMF filter bank of Figure 2.5 correspond to those of the wavelet transform, and appear as shown by Figure 2.6.

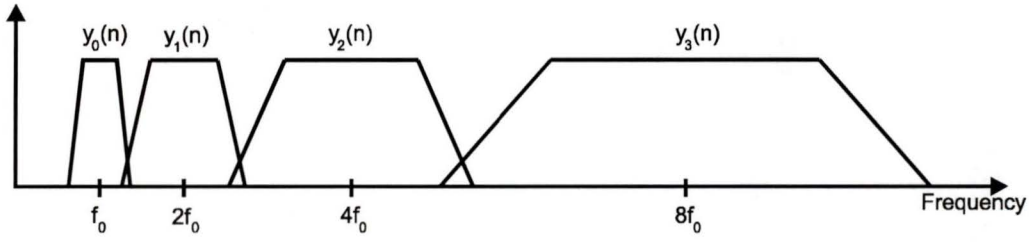


Figure 2.6 Frequency resolution of 4 octave wavelet decomposition.

The dyadic-tree structured QMF filter bank is a remarkably efficient computational structure. However, the complexity of the structure is linear in the number of input samples with a constant factor dependent on the length of the filter g. This is true *regardless of the depth of the tree.* The total complexity of this structure is related to

the number of operations per input sample C_0 in the following way:

$$C_{total} = C_0 + \frac{C_0}{2} + \frac{C_0}{4} + ... < 2C_0 \tag{2.19}$$

Equation 2.19 illustrates the efficiency of the DWT algorithm implemented by this structure; the delay associated with this structure grows exponentially as the tree depth is increased.

2.9 Iterated Filters and Regularity

The fundamental difference between the DWT algorithm of the dyadic tree QMF structure and the CWT is that the former scales the signal $x(n)$ for convolution with a filter whose impulse response $h(n)$ is constant for all stages, while the latter scales the mother wavelet $h(t)$ for windowing of a constant scale signal $x(t)$. Since the CWT's wavelets are exact scaled versions of one another, it therefore follows that the octave band filters that arise from the cascading of successive lowpass and highpass filter components in the DWT are required to be scaled versions of each other. Although the octave band filters are not really scaled versions of one another, the QMF structure can converge to this result provided the condition of ***regularity*** is met by the prototype filter h, which guarantees that the DWT implemented by the structure is the basis for the construction of continuous time compactly supported wavelet bases [43].

One of the multiresolution signal flow noble identities [22] states that signal decimation by a factor of two followed by a filter g characterized by transfer function $G(z)$ is equivalent to filtering the signal by a corresponding filter with transfer function $G(z^2)$ followed by 2-fold downsampling (see Figure 2.7).

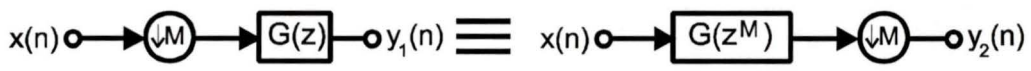


Figure 2.7 The decimation noble identity.

Consider the signal path that results in the lowest octave band filter arising from dyadic tree QMF structure with a depth of three. Successive applications of the identity on this signal path result in the 3-step simplification shown by Figure 2.8.

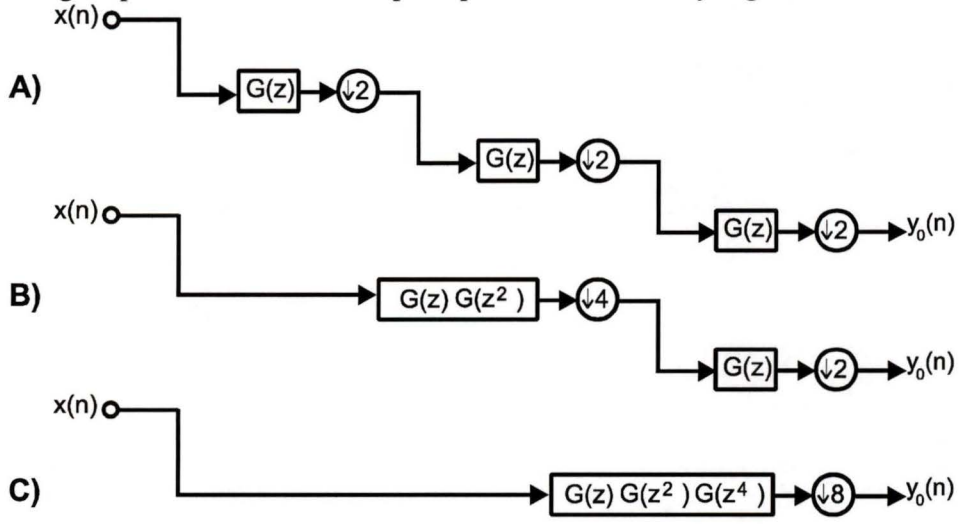


Figure 2.8 Simplification of the lowest octave band signal path.

By induction, the lowest octave band filter's transfer function $G_i(z)$ can be defined for any arbitrary tree-depth i as given by

$$G_i(z) = \prod_{k=0}^{i-1} G(z^{2^k}) \tag{2.20}$$

followed by a decimation of 2^i. This definition also applies to the time domain in that the impulse response of the filter is given by $g_i(n)$, which is the inverse-z transform of its transfer function $G_i(z)$.

2.9.1 Convergence Properties of $G_i(z)$

As the depth of any dyadic tree QMF filter bank infinitely increases, the impulse

response $g_i(n)$ of the lowest octave band filter becomes infinitely long. This increasing set of impulse response values plays a key role in ascertaining whether or not the octave band filters are scaled versions of one another, and therefore whether or not the filter is *regular.*

Since the property of *self-similarity* is easily ascertained from continuous-time signals, a continuous-time mapping of the impulse response $g_i(n)$ is developed as follows:

1. A staircase function $f_i(x)$ is defined for each i, in which the step-width is $1/2^i$, and
2. the value of each sample in $g_i(n)$ is mapped sequentially to each step of $f_i(x)$.

The continuous-time impulse response $f_i(x)$ is therefore supported on the interval $[0, N-1]$, where N is the length of the "prototype" filter g. It can be seen that this interval is constant for all values of i, since the larger number of impulse response values for deeper tree structures are mapped to smaller step-widths in the corresponding definition of $f_i(x)$. As the depth (i) of the tree goes to infinity, $f_i(x)$ will do one of three things:

1. It will converge to a continuous function with compact support $g_c(x)$,
2. it will converge to finitely discontinuous or fractal function, or
3. it will fail to converge at all.

In order to demonstrate how this convergence depends on the choice of filter impulse response $g(n)$, consider a MATLAB function *Reg_Chek* (See Appendix 1) written to compute and plot the continuous-time impulse response function $f_i(x)$ for succeeding values of i.

When this program is called to compute and plot $f_i(x)$ from $i = 1 \ldots 6$ for $g(n) = \begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix}$, the convergence of $f_i(x)$ to a smooth continuous function $g_c(x)$ is remarkably swift, as shown by Figure 2.9.

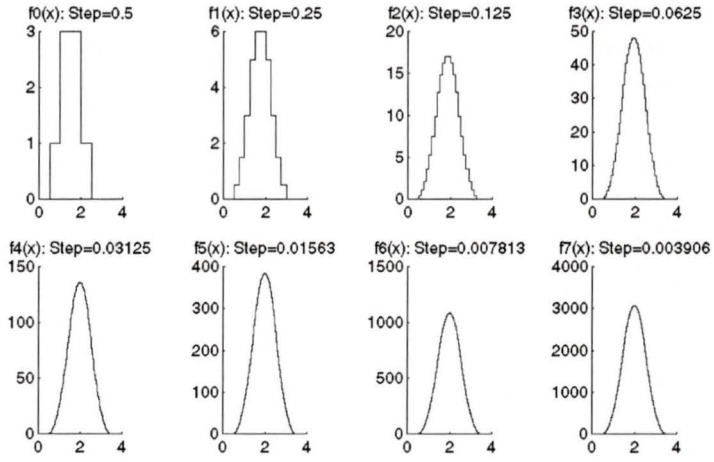


Figure 2.9 Convergence of $f_i(x)$

The function $g_c(x)$ is said to be regular because it is not only continuous, but is continuously differentiable at least once. In contrast to $g(n)$'s convergence to $g_c(x)$, consider $f_i(x)$ from $i = 1 \ldots 6$ for $g(n) = \begin{bmatrix} -1 & 3 & 3 & -1 \end{bmatrix}$ as shown by Figure 2.10.

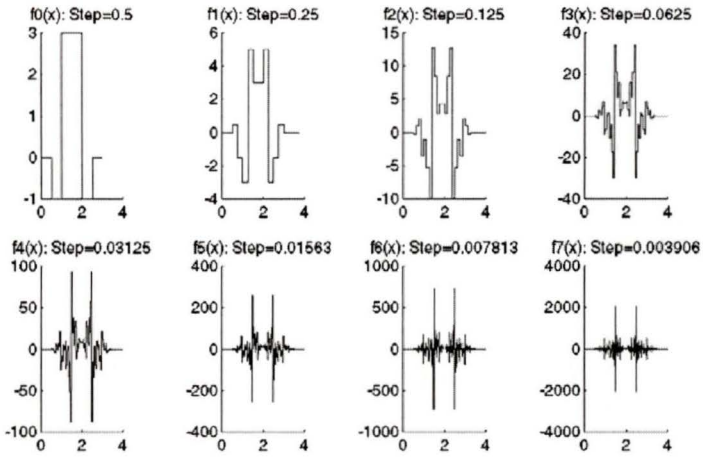


Figure 2.10 Divergence of $f_i(x)$.

$f_i(x)$ does not converge to a regular function at all. Instead, $f_i(x)$ diverges into fractal behaviour. In terms of filter theory, the regularity criterion may be viewed as a flatness condition on the spectrum of the frequency response of filter g at the Nyquist frequency $f_s/2$ - where the degree of flatness is defined by the number of zeros of the derivative $\frac{d}{d\omega}F(e^{j\omega})$. FIR and IIR Filters which meet this condition are characterized by having an arbitrary number of zeros at π, and as a consequence are of even order[1].

2.10 Iterated Filter Scaling Functions and Wavelets

Whenever $f_i(x)$ converges to a continuous function $g_c(x)$, it is referred to as a ***scaling function*** because it is used to go from a finer scale to a coarser one. Since this scaling function is the product of the iterated lowpass filters of the dyadic binary QMF filter bank, it satisfies the following two-scale difference equation:

---

1. Odd length filters have zeros at π. The order of the filter is *N-1*, where *N* is the length.

$$g_c(x) = \sum_{n=-\infty}^{\infty} g(n) g_c(2x-n) \tag{2.21}$$

In effect, this equation states that the impulse response of the filter can be convolved with the scaling function at twice its scale to yield the scaling function at its own scale. Self-similarity of the scaling function is satisfied.

The analysis of the higher octave band filters of the DWT for regularity and self-similarity are similar to the lowest octave band filter with one exception. The last filter in the iterated structure is highpass rather than lowpass (see Figure 2.5), which results in the continuous function $h_c(x)$ which satisfies the following two-scale difference equation:

$$h_c(x) = \sum_{n=-\infty}^{\infty} h(n) g_c(2x-n) \tag{2.22}$$

The function $h_c(x)$ is related to itself in different scales through convolution with the scaling function $g_c(x)$ and is referred to as the ***wavelet function*** of the DWT.

If the impulse responses $g(n)$ and $h(n)$ form an orthonormal set, then the continuous functions $g_c(x)$ and $h_c(x)$ also form an orthonormal set [43]. These continuous functions both satisfy two scale difference equations, so the set of all scaled and translated versions of the wavelet ($h_c(2^{-m}x-n)$ where m and n are integers) form an orthonormal basis for the set of square integrable functions.

2.11 Conclusions

The wavelet transform is a fully invertable transform that is ideally suited to the analysis of nonstationary signals. Musical signals are nonstationary in nature, and are therefore prime candidates for wavelet analysis. The multiple resolutions of the wavelet

transform allows large scales of analysis to be applied to low-frequency signal components of long duration while also allowing small scales of analysis to be applied to high-frequency components of short duration. The subband decomposition applied by the DWT results in an octave by octave separation of frequency spectra, which corresponds to the octaves of the musical scale. This chapter has shown how regular filters in QMF perfect reconstruction filter banks may be used to generate orthonormal wavelet bases. The converse operation is also possible, and is detailed in [43] and [46].

Chapter 3

The Musical Spectrogram

3.1 Introduction

The objective of the Musical Spectrogram is to visualize signal spectra in a musically meaningful way. Music is composed of pitched events (or notes) which occur in time and at varying amplitudes (volumes). A well known example of a musically meaningful visualization is the musical score notation as shown in Figure 3.1. This visualization of music signals conveys pitch data concisely while the timing data is subject to the defined performance tempo and time signature of the piece. The volumes of these notes are left entirely to the interpretation of the reader based on global volume cues such as *piano*, *forte*, etc.

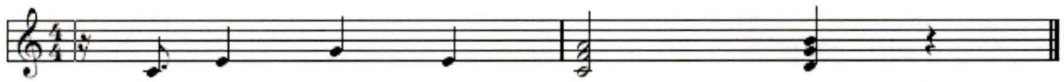


Figure 3.1 Musical score notation.

A more modern and precise form of music visualization is MIDI piano-roll, which is based on the punched-paper rolls used to control the mechanical player-pianos of the early twentieth century. In this visualization, note events are represented as rectangular shapes whose placement and length determine the pitch of each note along with it's start, sustain and end times. The same passage of music shown in Figure 3.1 is shown as MIDI piano roll in Figure 3.1.

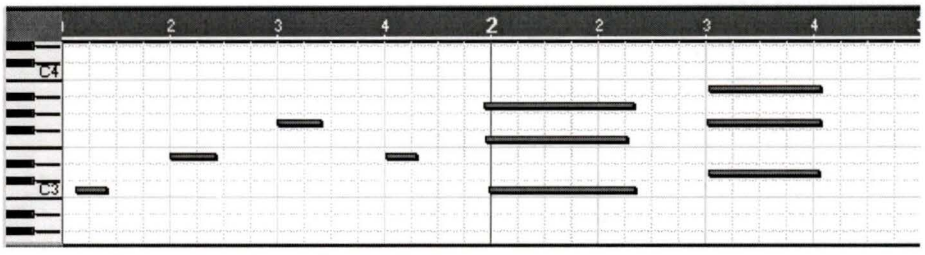


Figure 3.2 MIDI Piano-roll notation.

Piano-roll notation is an innovation of the MIDI sequencer software industry, and as such its form varies between manufacturers. Common to all versions are the representation of notes as rectangular shapes and the definition of a master clock that time-stamps each event. Many other parameters such as volume are also available for inspection in this visualization, but they require interaction with the software displaying the piano-roll.

The Musical Spectrogram visualizes audio data in a manner similar to MIDI piano-roll. The y-axis measures out the semitones of the equal-tempered musical scale while the x-axis demarcates time. Unlike the piano-roll, however, events are not merely rectangles. Instead, they are plotted signals of varying amplitude that correspond to the musical signal's spectral energy at the pitch and time (x,y) coordinates. This amplitude can be scaled and plotted vertically within the semitone band height, or it can be plotted as a color or greyscale-mapping in the z-axis depending on the application.

Deriving the visualization described above from audio signal wave data is not a trivial task. It involves a great deal of careful signal filtering and sampling-rate adjustment, as will be shown. This chapter presents and develops each of the following four stages of this process:

1. Tuning Resampler
2. 9-OBF Bank

3. 12-SBF Bank Array
4. Output System

In its entirety, the Musical Spectrogram is an analysis tool that processes digital audio signals (such as instrumental recordings, for instance) to produce time varying-spectral plots meaningful to those with music theory knowlege and training. The Musical Spectrogram is both a *joint analysis* [34] and a mid-level [2] signal representation, and is a fresh perspective on musical signals.

3.2 Musical Spectrogram Architecture

The four stages of processing involved in the Musical Spectrogram are sequential in application. The high-level architecture is shown in Figure 3.3.

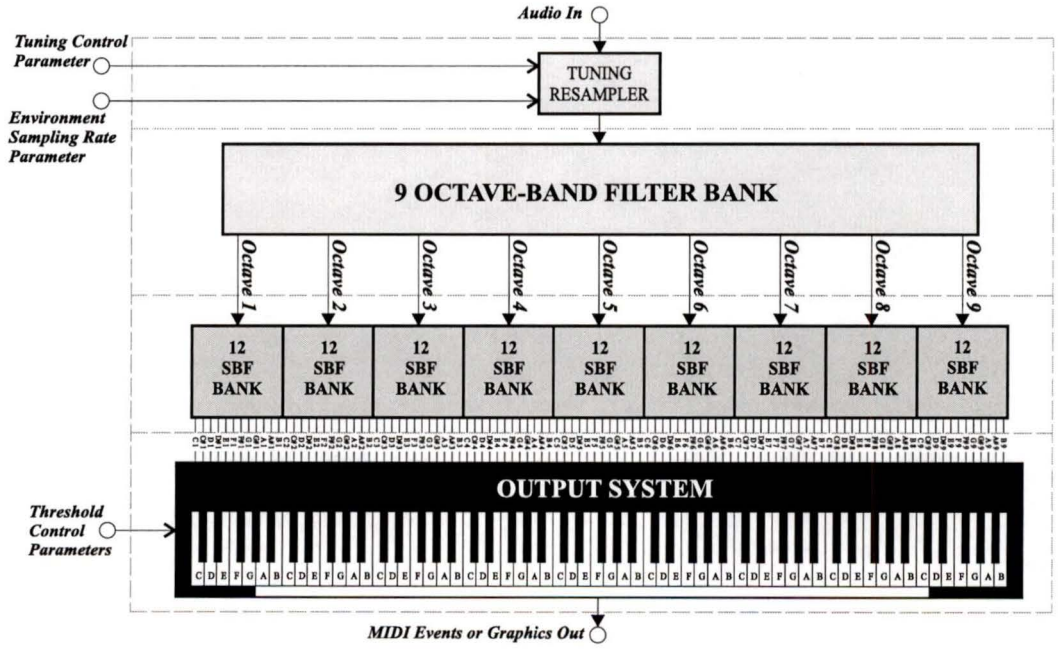


Figure 3.3 The Musical Spectrogram system architecture.

Stage 1 (the tuning resampler) adjusts the incoming musical signal to the rate required for correct tuning of the algorithm. Stage 2 (the 9-OBF Bank) separates the

resampled musical signal into 9 octave-band signals. Stage 3 (the 12-SBF Bank Array) separates each of the 9 octave-band signals into 12 semitone-band signals spaced according to the equal-tempered scale. Finally, stage 4 (the Output System) processes and plots each of the resulting 108 semitone-band signals as either a time-dependent amplitude or as a gate signal triggered whenever a specified threshold value has been exceeded.

3.3 Audio Filter Design Criteria

Each stage of the Musical Spectrogram is composed of digital filters varying in form and function. Although these filters differ in design, their common purpose is the processing of musical audio signals which exist within the parameters of human hearing. The following relaxed criteria therefore apply to the Musical Spectrogram's filters:

- Selectivities are not required to be greater than the frequency resolution of the equal-tempered musical scale.
- Since the Musical Spectrogram lacks a synthesis component, its outputs are not meant to be heard. This means that a linear phase response is not of critical importance.
- Filter orders (and processing delays) must be small enough to ensure real-time operation with acceptable latency under 10 ms [46].

These relaxed criteria encourage the use of IIR recursive filter designs in many cases where FIR designs would otherwise be desirable.

3.4 Tuning Resampler

From a practical perspective, the Musical Spectrogram must be tunable to a reference pitch other than A-440 and operate within DSP environments that support variable sampling rates. For example, typical electronic guitar tuners support the range A-436 to A-445 Hz, while common DSP hardware and software processing environments support a

subset of the following sampling rates: 8000, 9600, 11025, 12000, 16000, 22050, 24000, 32000, 44100, 48000, 88200 and 96000 Hz. The Musical Spectrogram therefore has two parameters called *environment sampling rate* (f_E) and *reference A* (f_A). These variables are required by the tuning resampler in order for it to correctly filter and downsample incoming audio to the *system sampling rate* (f_s) required by the succeeding stages of the Musical Spectrogram algorithm.

The Musical Spectrogram system sampling rate is calculated from the reference A frequency as follows:

$$f_s = 2f_A 2^{i/24} \tag{3.1}$$

where i is the number of equal-tempered *quarter-tones* from f_A, the reference A.

In Equation 3.1, negative values of i yield frequencies below f_A, while positive values of i return frequencies above it. Although quarter-tones are not explicitly used in the equal-tempered tuning, they correspond to the boundary frequencies between notes of this scale. According to Equation 3.2, even values of i yield the center frequencies of equal-tempered semitone notes, while odd values of i return the boundary frequencies of equal-tempered semitones. Figure 3.4 illustrates the quarter-tone frequencies across the Musical Spectrogram's 9 octave range.

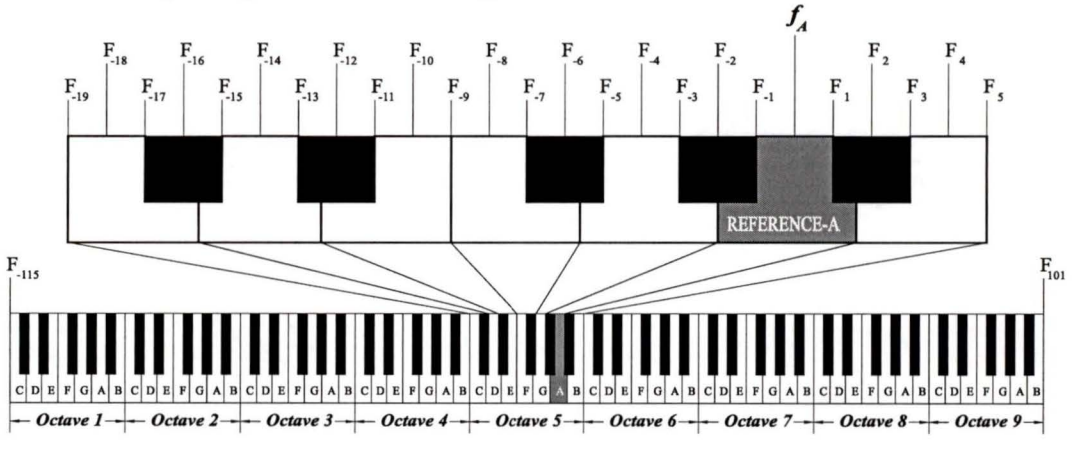


Figure 3.4 Quarter-tone frequencies across 9 octaves.

Table 3.1 lists the standard reference tunings A-436 to A-445 and the system sampling rates associated with them.

Table 3.1. Reference Tunings and System Sampling Rates.

Reference A f_A (Hz)	**System Sampling Rate f_s (Hz)**
432.768	16000.00
436	16119.48
440	16267.37
445	16452.22

A direct observation of Table 3.1 is that any host DSP environment must support the minimum standard sampling frequency of 22.05 kHz to avoid signal aliasing within the Musical Spectrogram. A lower cost musical spectrogram may be designed by reducing the number of octaves to 8, which reduces the required sampling rates to half of those in Table 3.1. The trade-off inherent with this cost reduction is that the first harmonic partials of all notes within the highest octave of the piano keyboard are then outside the analysis range of the 8 octave musical spectrogram, and this is usually not acceptable.

Given standard environment sampling rates of 22.05 kHz and beyond, the tuning resampler must apply fractional decimation to the incoming audio signal. The sampling ratio is computed from the system and environment sampling rates as

$$R = \frac{f_s}{f_E} \tag{3.2}$$

Some representative values of R are given in Table 3.2.

Table 3.2. Fractional Decimation Ratios.

Reference A	**22050.00 Hz**	**24000.00 Hz**	**32000.00 Hz**	**44100.00 Hz**
436	0.7310421769	0.671645	0.50373375	0.365521088
440	0.7377490237	0.6778069154	0.50835519	0.368874512
445	0.7461325353	0.6855092668	0.51413195	0.373066268

When the ratios of Table 3.2 are examined as rational numbers in Table 3.3, it can be seen that sample rate conversion techiques as described in Section 1.4.6 will not be very efficient to implement because the upsampling and downsampling factors required are simply too large.

Table 3.3. Upsampling / Downsampling Ratios.

Reference A	**22050.00 Hz**	**24000.00 Hz**	**32000.00 Hz**	**44100.00 Hz**
436	617 / 844	45 / 67	135 / 268	670 / 1833
427	159 / 217	241 / 358	155 / 307	159 / 434
438	553 / 753	307 / 455	293 / 579	300 / 817
439	251 / 341	681 / 1007	317 / 625	152 / 413
440	256 / 347	284 / 419	213 / 419	128 / 347
441	437 / 591	125 / 184	134 / 263	166 / 449
442	229 / 309	399 / 586	383 / 750	229 / 618
443	283 / 381	101 / 148	386 / 719	270 / 727
444	705 / 947	145 / 212	336 / 655	386 / 1037
445	241 / 323	194 / 283	382 / 743	241 / 646

Fortunately, an effective alternative method of fractional sample rate conversion exists for ratios such as these. This technique is now discussed.

3.4.1 Quartic Interpolation

A low-cost fractional sampling-rate conversion can be achieved through a quartic interpolator based on the Stirling central-difference formula [41]. According to this formula, if a fourth-order polynomial passes through 5 equally spaced points as shown in Figure 3.5, the interpolated sample at $t = (n - \alpha)T$ can be expressed as

$$\hat{x}(n-\alpha) = C_2 x(n-2) + C_1 x(n-1) + C_0 x(n) + C_{-1} x(n+1) + C_{-2} x(n+2) \quad (3.3)$$

where

$$\begin{bmatrix} C_{-2} \\ C_{-1} \\ C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{24}(\alpha+1)\alpha(\alpha-1)(\alpha-2) \\ -\frac{1}{6}(\alpha+2)\alpha(\alpha-1)(\alpha-2) \\ \frac{1}{4}(\alpha+2)(\alpha+1)(\alpha-1)(\alpha-2) \\ -\frac{1}{6}(\alpha+2)(\alpha+1)\alpha(\alpha-2) \\ \frac{1}{24}(\alpha+2)(\alpha+1)\alpha(\alpha-1) \end{bmatrix} \tag{3.4}$$

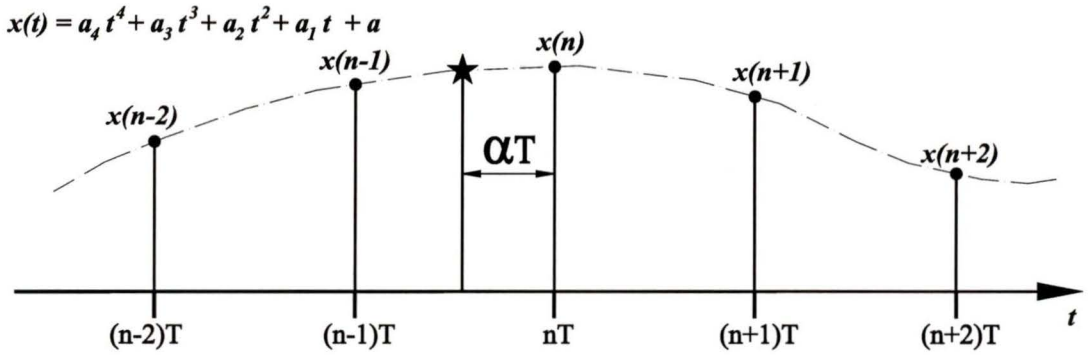


Figure 3.5 Quartic interpolation.

The delay parameter α is calculated for each output sample based on the difference between the input and output sampling grids. Figure 3.6 illustrates 3 values of α that result from fractional sampling-rate conversion ratio of $R = 8/3$. The next step of interpolation (at time $t = (n+1)T$) will see α calculated twice, and so on. New C_i coefficients are calculated according to Equation 3.4 for each value of α, and the out-

put sample is calculated as the vector product of the C_i and input samples.

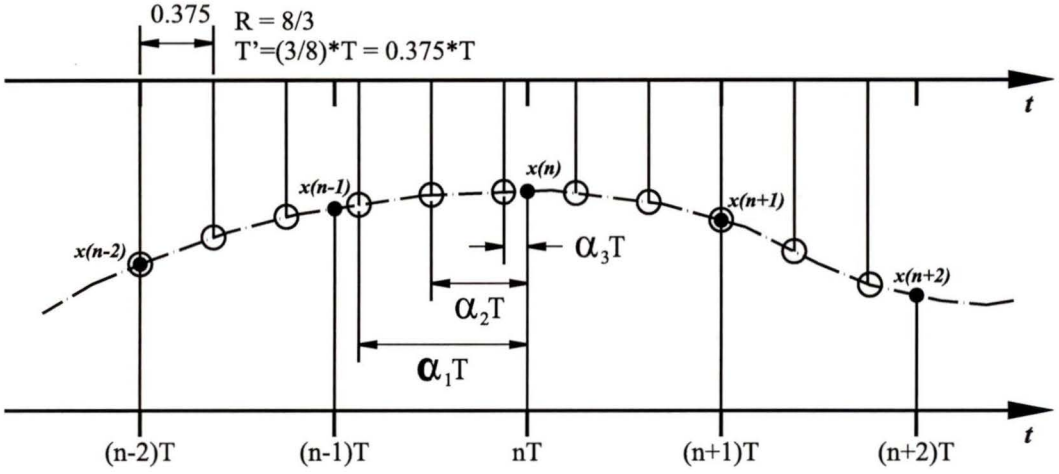


Figure 3.6 Calculation of α for 3 output samples.

Quartic interpolation based on Stirling's central-difference formula implements a fourth-order time-dependent FIR filter that performs rational sampling rate conversion. Although this filter is deemed to be an interpolator by numerical analysis theory, it is not a true interpolator by multi-rate signal processing standards because

- It operates as a decimator when the fractional sample-rate conversion factor is less than unity, and
- It does not bandlimit signals prior to decimation.

All signals must therefore be bandlimited to be within the baseband[1] of the output sampling rate before they can be processed by this numerical interpolator for fractional decimation (see Figure 3.7).

1. The baseband is defined as the frequency range between $-f_s/2$ and $f_s/2$.

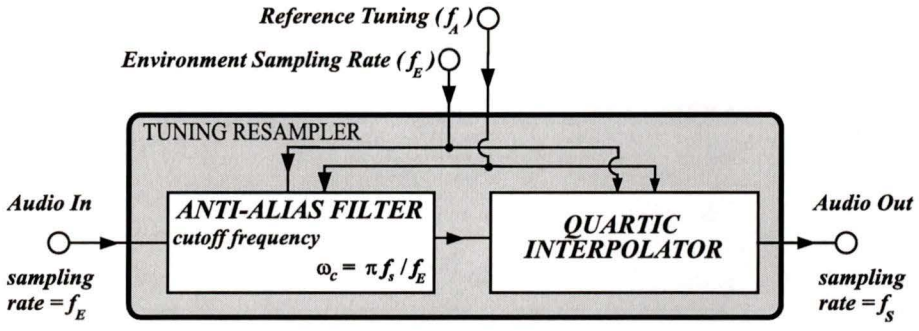


Figure 3.7 Tuning resampler architecture.

3.4.2 Anti-Alias Filter Design

As discussed in the previous section, an anti-alias filter is required to bandlimit the incoming audio signal prior to its fractional decimation. Assuming a normalized sampling frequency $\omega_s = 2\pi$, the cutoff frequency ω_c of this filter is dependent on the ratio of the Musical Spectrogram system sampling rate f_s over the environment sampling rate f_E and is given by

$$\omega_c = \pi R \tag{3.5}$$

where R is given by Equation 3.2.

A lowpass filter design that can self-adjust to this cutoff frequency is achieved through the application of the *Constantinides lowpass to lowpass transformation* [42] (see also p. 243 of [7]). This technique changes any normalized halfband lowpass filter characterized by the transfer function $H_0(z)$ into a denormalized anti-alias filter characterized by the transfer function $H_A(\bar{z})$ according to the transformation

$$H_A(\bar{z}) = H_0(z)\Big|_{z = \frac{\bar{z} - \alpha}{1 - \alpha\bar{z}}} \tag{3.6}$$

where

$$\alpha = \frac{\sin\{[(\pi/2) - \omega_c]/2\}}{\sin\{[(\pi/2) + \omega_c]/2\}} \tag{3.7}$$

This transformation is applicable to both FIR and IIR designs, and can be computed directly when the transfer function $H_A(\bar{z})$ is expressed in terms of the original coefficients of $H_0(z)$. For a second-order transfer function

$$H_0(z) = \frac{A + Bz + Cz^2}{D + Ez + Fz^2} \tag{3.8}$$

the transformed transfer function can be reorganized algebraically to yield

$$H_A(\bar{z}) = \frac{[A - B\alpha + C\alpha^2] + [-2A\alpha + B(1+\alpha) - 2C\alpha]\bar{z} + [A\alpha^2 - B\alpha + C]\bar{z}^2}{[D - E\alpha + F\alpha^2] + [-2D\alpha + E(1+\alpha) - 2F\alpha]\bar{z} + [D\alpha^2 - E\alpha + F]\bar{z}^2} \tag{3.9}$$

where α is given by Equation 3.7. With patience, higher-order transfer functions can be also derived in the same way.

The tuning resampler's initialization task list is as follows:

- Read the environment sampling rate f_E and reference-A f_A parameters.
- Calculate the Music Spectrogram system sampling frequency f_s from Equation 3.1.
- Calculate the fractional decimation ratio R from Equation 3.2. and the anti-alias filter's normalized cutoff frequency ω_c from Equation 3.5.
- Initialize the Stirling interpolator with R.
- Transform the $h0$ normalized halfband lowpass filter coefficients into the required hA anti-alias filter coefficients via Equation 3.7 and coefficient formula derived in the same manner as Equation 3.8.

The design of the $h0$ normalized lowpass halfband filter is discussed and developed in Section 3.5.2. where issues such as filter type, order and specification are addressed in the context of the 9-OBF Bank requirements.

3.5 Nine Octave-Band Filter Bank

The second stage of the Musical Spectrogram is based on a DWT filter bank as described in Section 2.8. This filter bank is composed of 9 iterations of the two-channel QMF structure shown in Figure 2.5. For the sake of organization, the iterated components of the QMF structure are now defined within the *octave processing module* (OPM) illustrated in Figure 3.8. The OPM incorporates both the highpass and lowpass halfband filters (denoted as **h1** and **h0**, respectively), as well as the 2-fold downsamplers follow them. The 9-octave band filter (9-OBF) bank architecture employing the OPM is shown in Figure 3.9. The *octave 0* output of the 9-OBF bank is not used by the Musical Spectrogram because it contains frequency spectra below the analysis range (0 - 15.87 Hz, when reference A is 440).

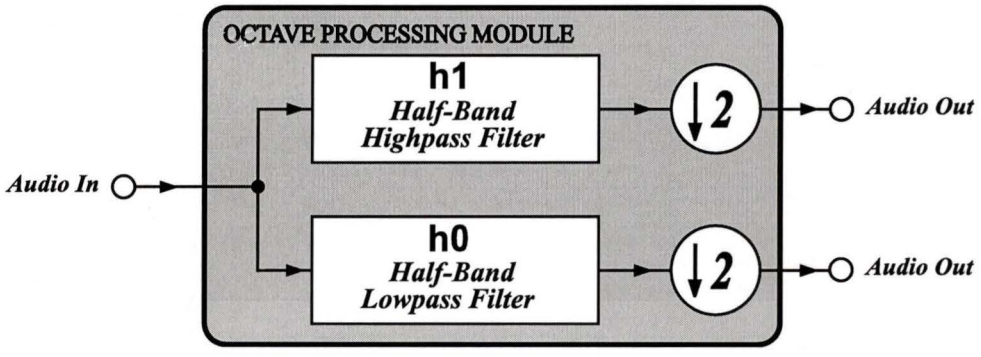


Figure 3.8 Octave processing module (OPM) of the 9-OBF bank.

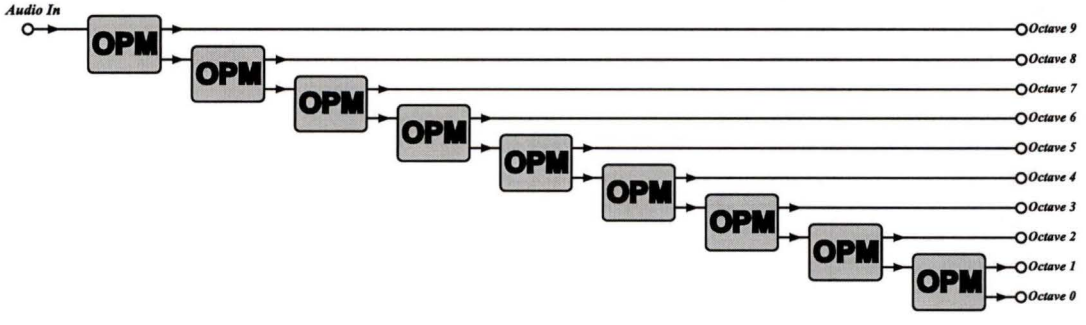


Figure 3.9 9-OBF bank architecture.

3.5.1 Filter Transition-Band Width

In Section 3.4, the boundary frequencies of each equal-tempered semitone according to the Musical Spectrogram were defined, calculated and illustrated. The practical transitional range between adjacent semitones is now explained as a consequence of the filter designs employed by the 9-OBF bank and 12-SBF banks. Figure 3.10 shows the loss characteristic of three ideal filters in a filter bank designed to separate 3 semitones. The transition bands are 33 cents wide in this scaled illustration.

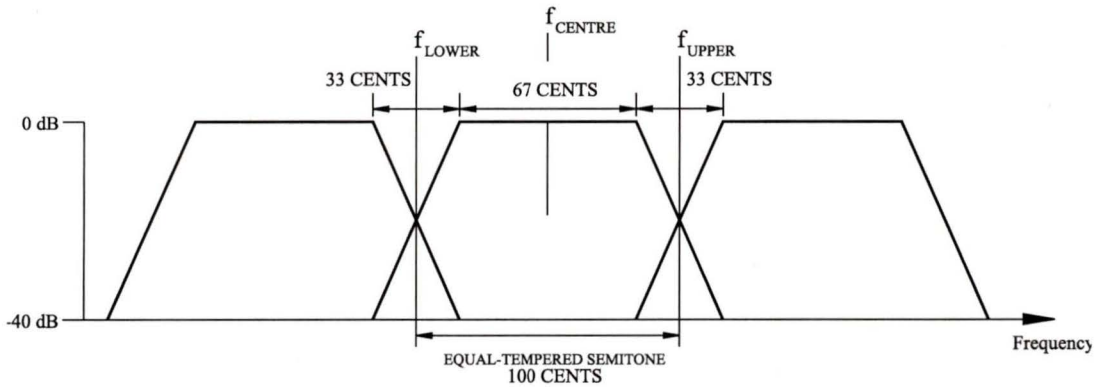


Figure 3.10 Loss characteristic of ideal filter bank with 33-cent transition bands.

Any spectral content falling within the transition bands of this filter bank will be attenuated during the analysis. The 33-cent transition bands shown in Figure 3.10 occupy almost one third of the entire spectrum, which is simply too wide for meaningful analysis. The prescribed specification technique of filter design allows the specification of arbitrarily narrow transition bands at the cost of higher filter orders, and this illustrates the trade-off between filter selectivity and implementation cost.

3.5.2 Halfband Filter Type

For the 9-OBF bank to guarantee perfect reconstruction, the impulse response of halfband filter *h0* must display the property of self-similarity (see Section 2.9.1), which can be viewed as a flatness condition in its spectrum at the Nyquist frequency. Filters

meeting this condition are characterized by having an arbitrary number of zeros at π, and are FIR filters of even order. Unfortunately, such filters are not very selective and do not meet the selectivity requirements of the Musical Spectrogram. Figure 3.11 illustrates the amplitude responses of 4 candidate halfband filter types:

- Daubechies FIR filter, self-similar, zeros at π, $N = 54$
- Kaiser windowed FIR filter, self-similar, zeros at π, $N = 54$
- Chebyshev IIR filter, zeros at π, $N = 20$
- Elliptic IIR filter, $N=8$

While the Daubechies filter is an orthogonal wavelet filter, its selectivity is quite poor. A Kaiser windowed FIR filter of the same order based on the Fourier series is found to have superior selectivity and better passband characteristics without sacrificing the self-similarity property required for the perfect reconstruction of the DWT. Two IIR filter frequency responses (Chebyshev and elliptic), are also shown in Figure 3.11 and demonstrate greatly increased selectivity at a significantly lower order. Both of these IIR filters were designed using the *prescribed specifications* technique described in Chap. 8 of [7] to yield 66 and 33 cent transition bands respectively (centered around $\pi/2$). Although the Chebyshev filter cannot be tested for self similarity because it has an impulse response of infinite duration, it does have zeros at π, which is an important criterion for a perfect reconstruction DWT. The most selective and efficient of all filter types is the elliptic. Although it does not meet the DWT's perfect reconstruction criteria, this filter type clearly offers the best amplitude response.

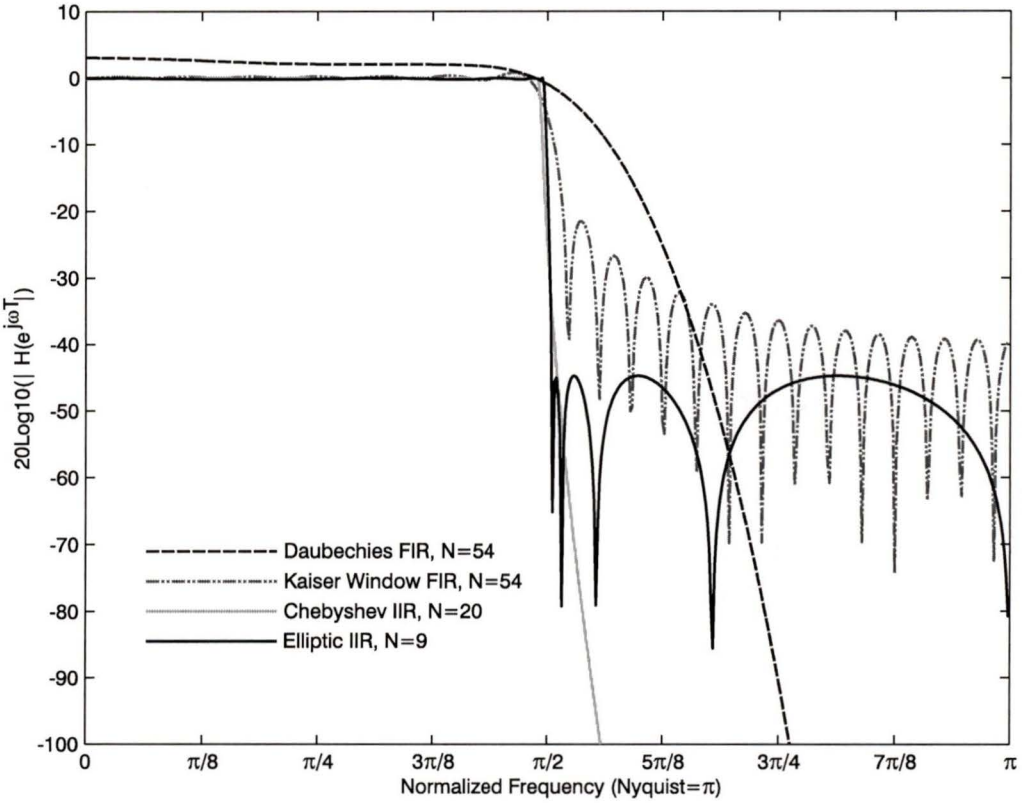


Figure 3.11 Candidate halfband filter reponses.

If perfect reconstruction is not imposed on the Musical Spectrogram, the elliptic IIR type is the best choice. On the other hand, if perfect reconstruction is imposed, the Kaiser windowed FIR type is the most selective of the guaranteed self-similar alternatives. The Chebyshev IIR type does warrant further investigation to determine whether it guarantees the perfect reconstruction criterion, for it would be superior to the Kaiser windowed FIR type if this were found to be true.

As a dedicated analysis system, the Musical Spectrogram does not include a synthesis component, and perfect reconstruction is not a concern. While future designs including synthesis stages will require perfect reconstruction filters, the Musical Spectrogram proposed here is not so constrained and uses elliptic IIR filters instead.

When a halfband lowpass elliptic filter with a 1-cent transition band (and a 99-cent wide passband) is specified, a 14th order lowpass halfband elliptic design $h0$ is produced with:

- 1-cent transition bands
- Passband ripple of 0.25 dB
- Minimum stopband attenuation of 41.74 dB

This filter's loss characteristic is plotted in Figure 3.12. The characteristics of the previous candidate filters are also shown as light grey traces for comparison purposes. this filter is the final choice for the OPM lowpass halfband filter $h0$.

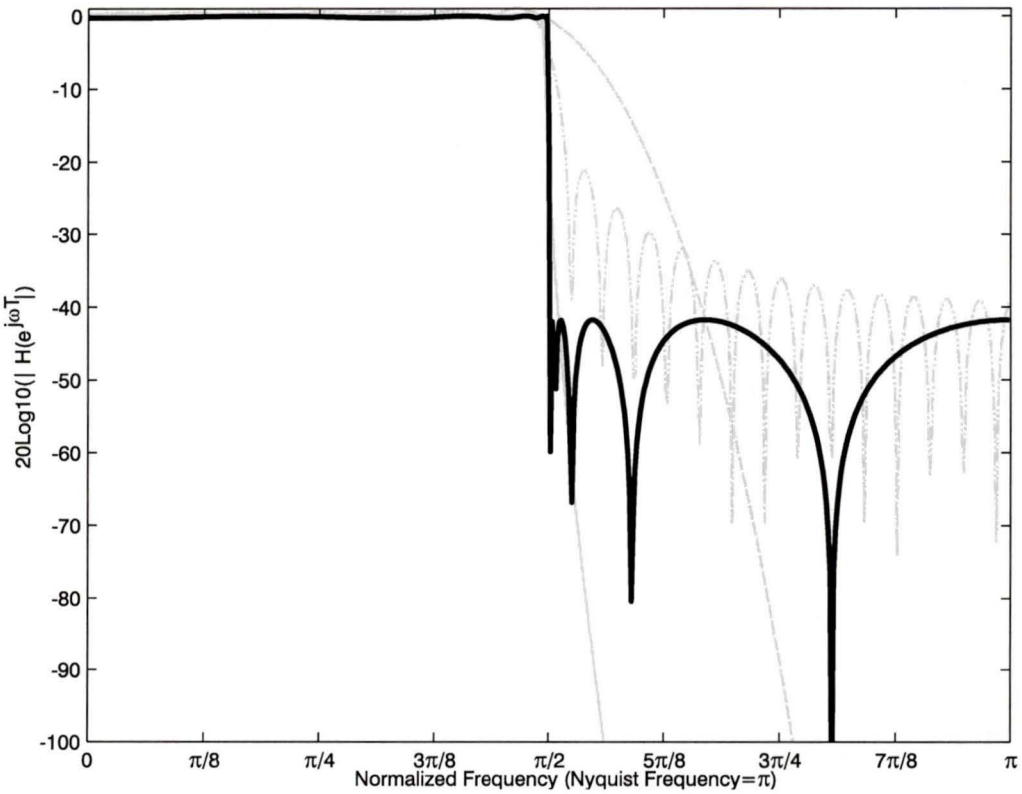


Figure 3.12 The final halfband lowpass elliptic design $h0$.

3.5.3 Lowpass to Highpass Transformation

Equation 2.18 defined how an FIR lowpass halfband filter h characterized by transfer function $H(z)$ yielded a QMF design g characterized by transfer function $G(z)$ when the simple substitution $z = -\bar{z}$ was applied. This substitution was in fact an application of the Constantinides *lowpass to highpass transformation* (see p. 243 of [7]), where the normalized and transformed cutoff frequencies are both equal to $f_s/4$. This transformation is properly given by

$$z = -\frac{1}{z} \tag{3.10}$$

The reason that Equation 2.8 and Equation 3.10 do not agree is because Equation 3.10 when applied to an FIR filter produces a non-causal design. Such non-causal FIR filter transfer functions must be multiplied by a factor of z^{-N}, where N is the order of the transfer function to convert them to a causal design with an identical amplitude response.[1]

When Equation 3.10 is applied to the transfer function of IIR filter $h0$, it yields the filter design $h1$, which is a halfband highpass filter sharing the transition band, passband ripple and minimum stopband attenuation characteristics of $h0$.[2] The loss characteristics of both filters are plotted in Figure 3.13, and are considered excellent choices for the Musical Spectrogram's OPM module because of their superb selectivity.

1. $z^{-N}\Big|_{z = e^{j\omega T}} = e^{-j\omega NT}$, and $\overline{\left|e^{-j\omega NT}\right|} = 1$.

2. Note that IIR filters are not rendered non-causal by the Constantinides lowpass to highpass transformation because the positive powers of z introduced into the transfer function's numerator are cancelled by the positive powers of z similarly introduced into its denominator. No correction is required as in the FIR case.

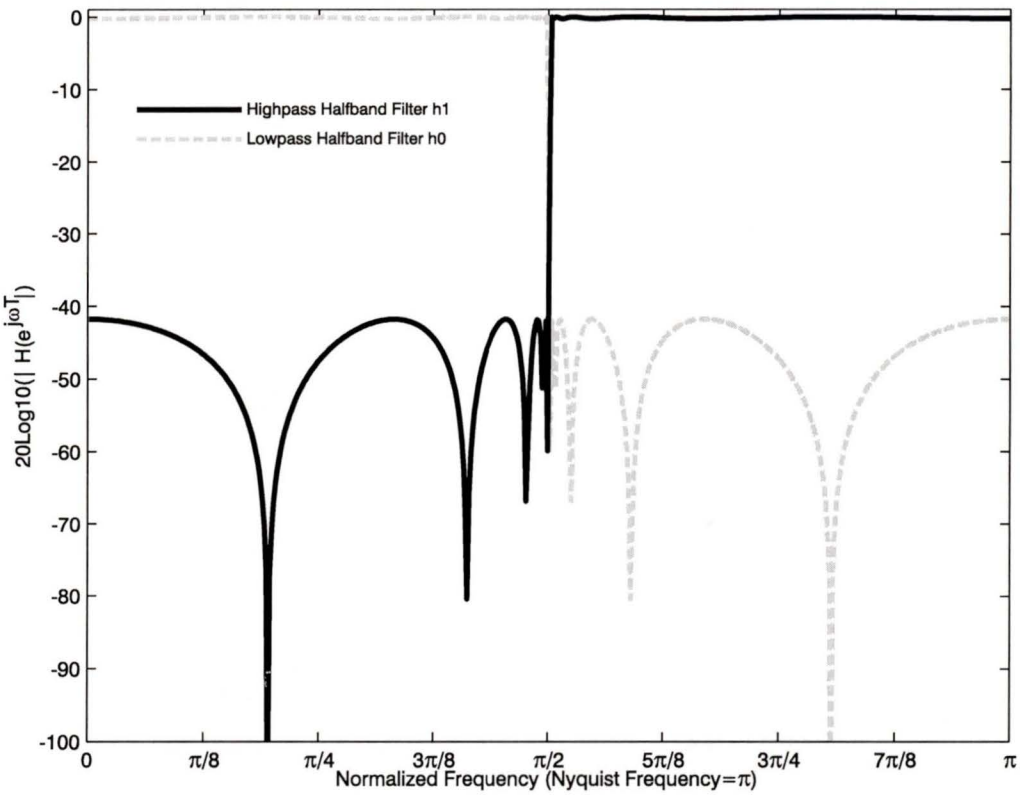


Figure 3.13 Lowpass and highpass halfband filters $h0$ and $h1$.

3.5.4 Analysis-Only OPM

The main consequence of the definition of $h0$ and $h1$ filters as elliptic IIR designs is that the Musical Spectrogram does not include a synthesis bank. This deviation from the DWT architecture means that the OPM design must be examined to ensure no dependencies on these removed components exist.

Examination of the OPM design's adherence to the rules of alias-free decimation and interpolation (see Section 1.4.3 and Section 1.4.5, respectively) reveals that a correctable form of aliasing is introduced when signals are decimated at the output of highpass filter $h1$ (see Figure 3.8). This aliasing is shown in Figure 3.14, which illustrates

how a spectral image of the highpassed signal is generated in the $(0, f_s'/2)$ interval by the 2-fold downsampler that follows $h1$.

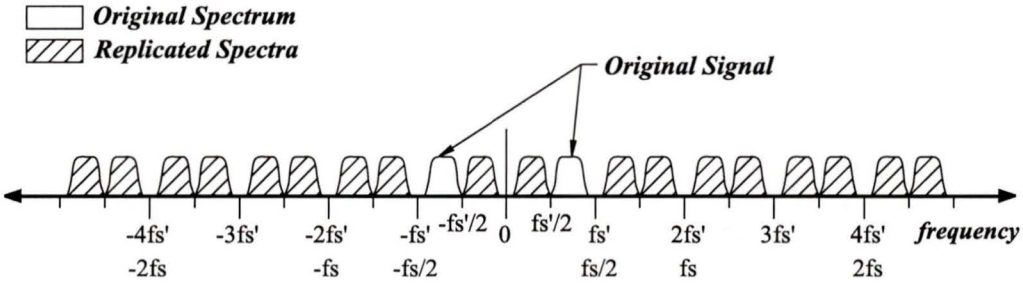


Figure 3.14 Highpass filter output signal aliasing.

In the DWT implementations employing full QMF analysis and synthesis banks, this aliasing is cancelled out. Since there is no synthesis component to the Musical Spectrogram (which is not a DWT either because of its choice of filters), this aliasing would be a problem as long as the 2-fold downsampler is allowed to remain after the *h1* filter. Fortunately, this downsampler has no use in an analysis-only filter bank system, and is therefore removed from the design. Figure 3.8 shows this change in the OPM's architecture.

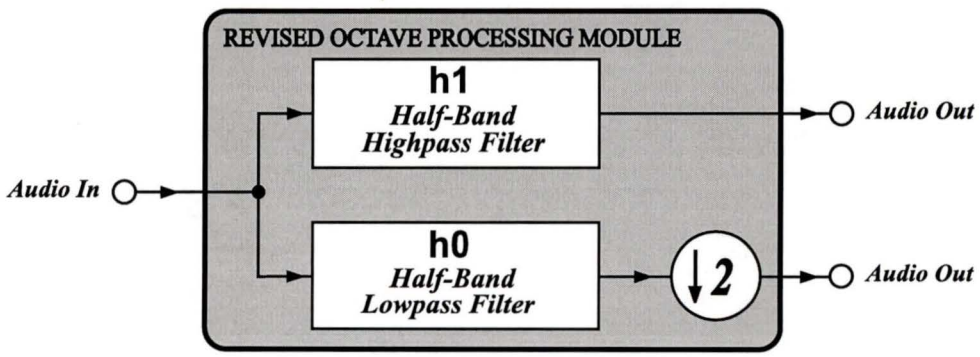


Figure 3.15 Revised OPM.Twelve Semitone-Band Filter Bank

3.6 Twelve Semitone Band Filter Bank

The next stage of the Musical Spectrogram processing is the *12 semitone-band filter* (12-SBF) bank, which is a novel adaptation of the techniques and structures previously defined and discussed in this chapter. Just as the 9-OBF bank was organized into octave processing OPM modules, the functional elements of the 12-SBF bank are defined within a *Semitone Processing Module* (SPM) as shown in Figure 3.16.

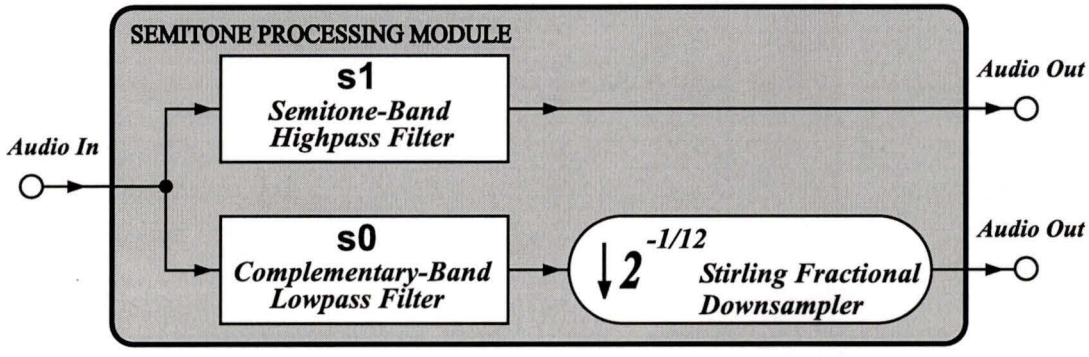


Figure 3.16 Semitone processing module (SPM) of the 12-SBF bank.

The SPM incorporates a highpass and lowpass filter (denoted as **s1** and **s0**, respectively), which separate the upper semitone from the rest of the octave signal. The 12-SBF bank architecture employing the SPM is shown in Figure 3.17.

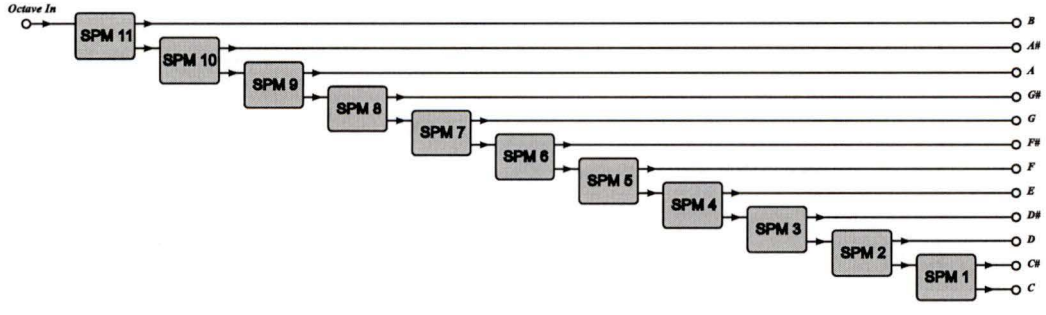


Figure 3.17 12-SBF Bank Architecture.

Although the 12-SBF bank resembles a DWT filter bank, it must be stressed that

this design is for analysis purposes only. The mathematics and filter theory required to develop a complementary perfect reconstruction 12-SBF synthesis bank are well beyond the scope of this design and are left for future work.

3.6.1 Semitone and Complementary Band Filters

The $s1$ filter is designed to separate the upper semitone band from the rest of the octave, and has a normalized cutoff frequency defined as

$$\omega_c = \pi 2^{-1/12} \tag{3.11}$$

The prescribed specifications method of IIR filter design yields 11th order elliptic filters with the common characteristics of 1-cent transition bands, a passband ripple of 0.25 dB and a minimum stopband attenuation of 45.9 dB. These loss characteristics of these designs are plotted in Figure 3.18.

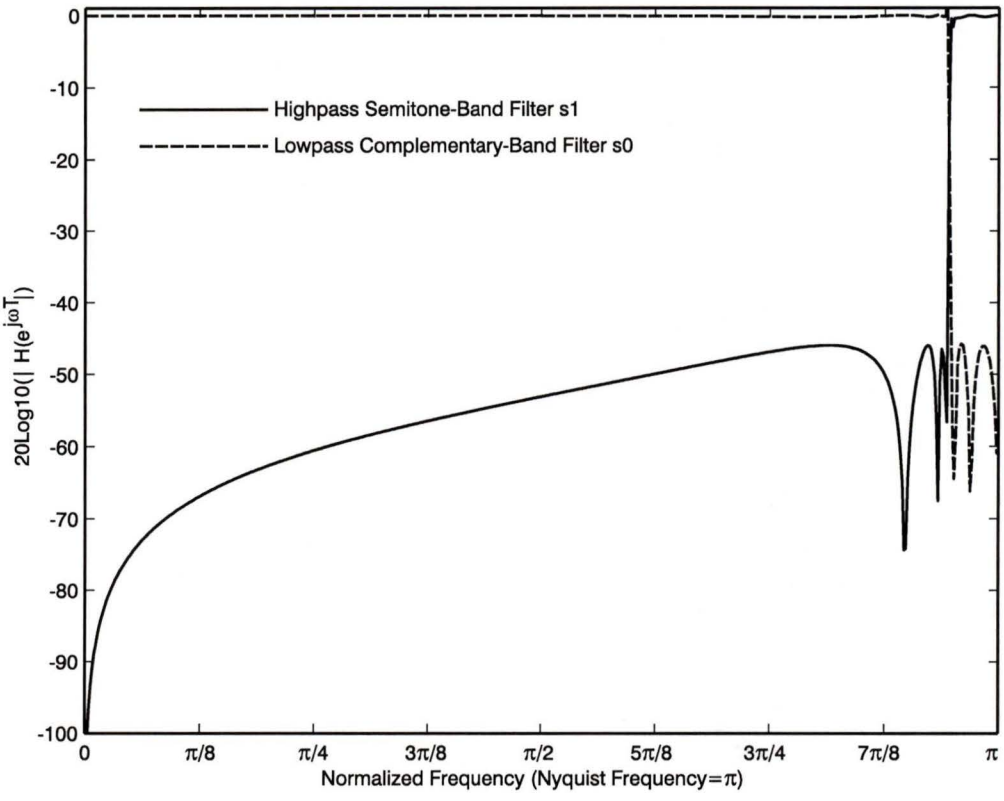


Figure 3.18 s1 and s0 filters.

3.6.2 Stirling Fractional Downsampler

The Stirling fractional downsampler is a specialized version of the quartic interpolator described in Section 3.4.1. Unlike this general interpolator, however, the fractional downsampler has a fixed sampling-rate conversion ratio R_D given by

$$R_D = 2^{(-1/12)} = 0.94387431... \tag{3.12}$$

The fixed ratio allows design optimizations that were not feasible for the generalized quartic interpolator. Analysis of R_D shows that it is approximated within 6 digits by the ratio $1564/1657$, which permits the precomputation of the first 1564 values of α

and associated coefficients C_i to be recycled indefinitely. Examination of this technique's approximation error reveals that α periodically "slips" from 0.00027958 to 0.0 every 1564 samples. This minute error is cumulative and calculations project that 1 hour of downsampling a signal from 16.57 kHz to 15.528 kHz results in a combined loss of 9.9933 samples. Such a loss of samples would be imperceptible to human hearing if it was to occur over one second, let alone the hour it requires to accumulate.

The precomputed data structures required by this technique are now described. Summed values of α are stored in a 1-by-1564 vector

$$S_i = i2^{1/12} \tag{3.13}$$

where $i = 1$ to 1563. Delta vector Δ is computed as

$$\Delta_i = \lfloor S_{(i+1)} \rfloor - \lfloor S_{(i)} \rfloor \tag{3.14}$$

and the actual values of α are

$$A = S - \lfloor S \rfloor \tag{3.15}$$

The quartic coefficients for every α are stored in a 5-by-1564 matrix C where each column is given as

$$C_j = \begin{bmatrix} \frac{1}{24}(A_j+1)A_j(A_j-1)(A_j-2) \\ -\frac{1}{6}(A_j+2)A_j(A_j-1)(A_j-2) \\ \frac{1}{4}(A_j+2)(A_j+1)(A_j-1)(A_j-2) \\ -\frac{1}{6}(A_j+2)(A_j+1)A_j(A_j-2) \\ \frac{1}{24}(A_j+2)(A_j+1)A_j(A_j-1) \end{bmatrix} \tag{3.16}$$

Only the delta vector Δ and coefficient matrix C are required by the fractional downsampler during run-time. Each instance of the fractional downsampler uses a local counter variable c_o that resets to 1 upon reaching 1564. Each block of input samples is then stepped through by index counters I_i and I_o that keep track of the current input sample $x(n)$ and output sample $\hat{x}(n-\alpha)$ respectively. Block processing occurs through the following algorithm:

1. c_o is used as a column index of C to obtain the 5 coefficients to compute the current output sample.
2. The current 5 input sample values around index I_i are obtained. This index points to the third of five (sample $x(n)$) so an offset of -2 is required.
3. The vector dot product of the current coefficients and sample values is computed and written to the output sample block at index I_o.
4. I_o is incremented by 1, I_i is incremented by the value stored at $\Delta(c_o)$
5. c_o is incremented by 1 or reset if it has reached 1563.

At initialization, index counters I_i and I_o are set to 3 to account for the interpolator's non-causal impulse response.

This algorithm is also easily adapted to sample processing in real-time DSP systems. In this case the each arriving sample backshifts the input buffer and triggers the-first three events listed above, followed by a check of the value stored at $\Delta(c_o)$ to determine if more than one input sample should be indexed. In this case, a value of two will cause the setting of a local flag variable called *StopProc* that tells the downsampler to backshift the input buffer on the next sample and clear the flag without computing an output sample. This StopProc flag is also used for real-time control, as will be seen in Section .

The Musical Spectrogram employs 99 fractional downsamplers and all share a

single global of the C matrix and Δ vector. At a storage cost of about 95 words per fractional downsampler, the calculation of output samples is reduced to a 5-element dot-product vector multiplication (or 5 multiply and accumulate instructions in DSP hardware, depending on the implementation platform).

3.6.3 Implementation: Control of the 9-OBF Bank and 12-SBF Banks

As with any multi-rate system, the timing and control considerations of the 9-OBF bank require special attention when applied to sample processing systems such as real-time DSP environments. Such systems are typically interrupt driven, where each arriving sample or block of samples trigger processing. For the 9-OBF bank, this process is initiated when each new sample arrives at the input shown in Figure 3.19. An integer counter called *ActiveStages* is maintained that runs from 1 to 128. This variable provides complex control sequencing to the system of OPM modules by optimal assignment of each processing state to a numerical value whose bit pattern can be determined with a minimal number of tests.

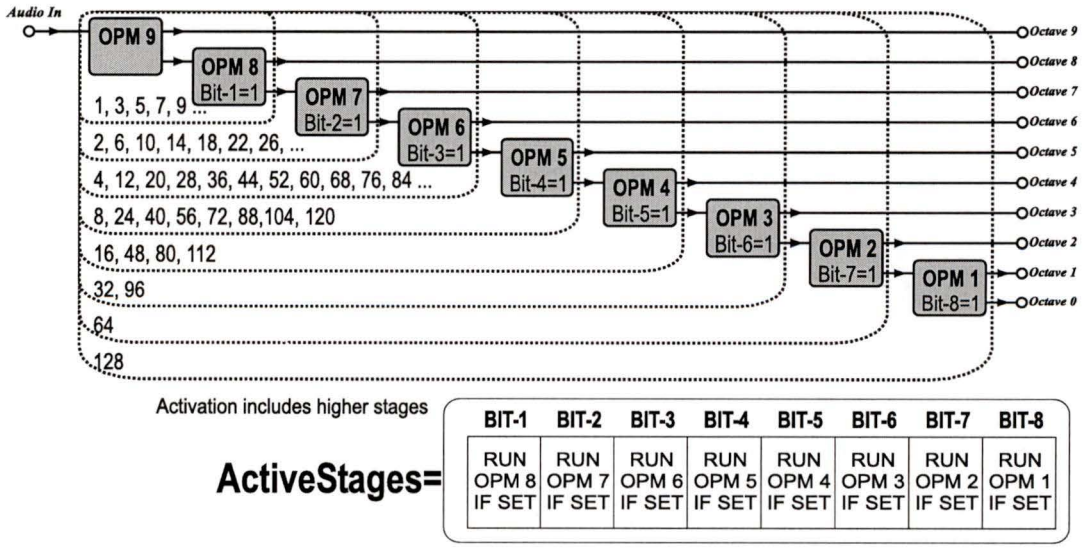


Figure 3.19 9-OBF bank sample-processing control system.

Every sample arriving at the input is always processed by OPM 9. If the first bit in the ActiveStages variable is set (ActiveStages=1, 3, 5, ...), then OPM 8 is also activated, and so on. When one of the control bits in ActiveStages is tested and found clear, processing is complete and the ActiveStages variable is either incremented or reset to 1 when it has reached 128.

The output sampling rates of each semitone output are now examined. Figure 3.20 shows each output semitone signal and its fractional sampling rate relative to f_s. The actual output signal sampling rate upper and lower limits are summarized in Table 3.4.

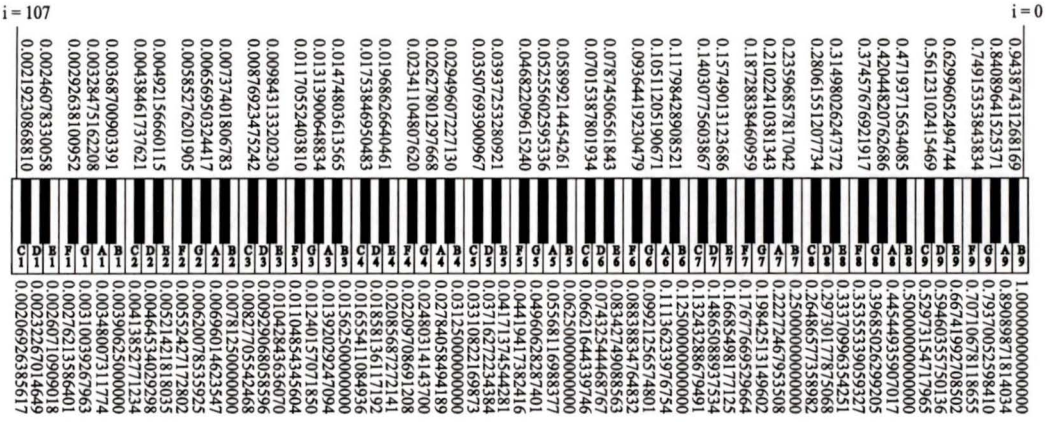


Figure 3.20 Output signal sampling rates as fractions of f_s.

Table 3.4. Output Signal Sampling Rates at Different System Tunings.

Output Signal	**A-436** **fs = 16119.48**	**A-440** **fs = 16267.37**	**A-445** **fs = 16452.22**
B9	16119.48 Hz	16267.37 Hz	16452.22 Hz
C1	33.36 Hz	33. 66 Hz	34.04 Hz

Examination of the sampling rates of Figure 3.20 show that each of the semitones operates at twice the rate of that one octave lower, which, in turn operates at twice the

rate of that a further octave below. In terms of timing, this observation indicates a need for organized planning to avoid processing bottlenecks and resource conflicts in real-time implementations.

Fortunately, the real-time control strategy is readily extended to handle each of the 9 12-SBF banks. This approach has each 12-SBF bank directly connected to a corresponding OPM output so that the *ActiveStages* control variable triggers scheduled OPM modules *and* the 12-SBF bank units connected to their outputs. Once the first module (SPM 11) of any 12-SBF bank unit has been delivered a sample and activated, processing proceeds recursively for $N = 11$ down to 1 according to the following algorithm[1]:

1. SPM *(N)* checks to see if its *StopProc* flag was set by the last execution.
2. If StopProc is set, the flag is cleared and processing is complete.
3. If StopProc is not set, SPM *(N)* executes once and writes the highband output sample to the semitone output and the lowband output sample to the input of SPM *(N-1)*.
4. SPM *(N-1)* is activated.

3.7 Single-Rate 12-SBF Bank

Although the 12-SBF bank described in the preceding section has many positive features, there are aspects of the design that raise complications on implementation. Specifically, the fractional downsampler does not have a uniform unity gain over its operational frequency range and attenuates the input signal's spectrum noticeably throughout the upper half of the Nyquist interval. When 10 such fractional downsamplers are cascaded (in the case of the C# output signal, for instance), the total signal attenuation is quite severe. Since this attenuation is frequency dependent and varies

1. Please refer to Figure 3.17 for illustration if required.

depending on the changing values of α, the design of a corrective filter with an inverse loss characteristic requires adaptive filter design methodologies. These are beyond the scope of this dissertation and are therefore left as future work.

An alternative version of the 12-SBF bank is now introduced that operates at a single sampling rate and does not require any form of fractional downsampling. This version is straightforward to implement, although it requires more data storage and processing resources than the multi-rate 12-SBF Bank.

The single-rate 12-SBF bank is a classic single-rate filter bank that employs 12 IIR elliptic filter designs with 40-cent transition bands and a stopband attenuation of 60 dB. Each of these filters are specific to a particular semitone within the octave under analysis, and are named accordingly. Figure 3.21 plots the loss characteristics of each of these filters.

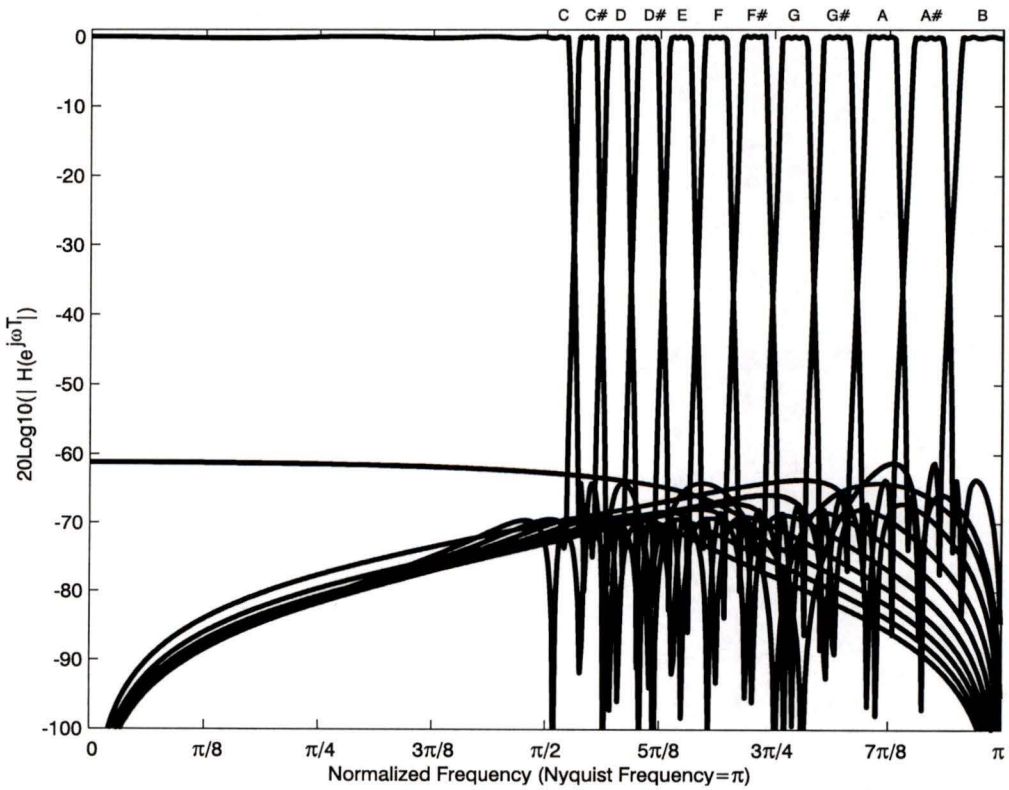


Figure 3.21 Loss characteristics of the 12 filters in single-rate 12-SBF bank.

The *C* filter is an 11th order lowpass design, the *C#* to *A#* filters are all 10th order bandpass design and the *B* filter is a 6th order highpass design. The orders of these filters were kept relatively low by allowing the 1-cent transition band requirement of the *h0* and *h1* filters to expand to 40 cents. Only the central 60% of any semitone signal will be passed by these filters, but this compromise is neccessary to keep the filter designs practical.

3.8 The Output System

The output system of the Musical Spectrogram processes the 108 multirate output signals of the 12-SBF bank array into display traces. The most basic form of output system is an x-y plot of each semitone signal with every signal trace offset to provide separation between tones. This seismograph-style of output system can take the form of a offline-process and display plot such as Figure 3.22, or it can be implemented as a real-time scrolling plot as shown by Figure 3.23, (This is a screen capture of the Musical Spectrogram's DSP host application, which is briefly discussed in the next chapter).

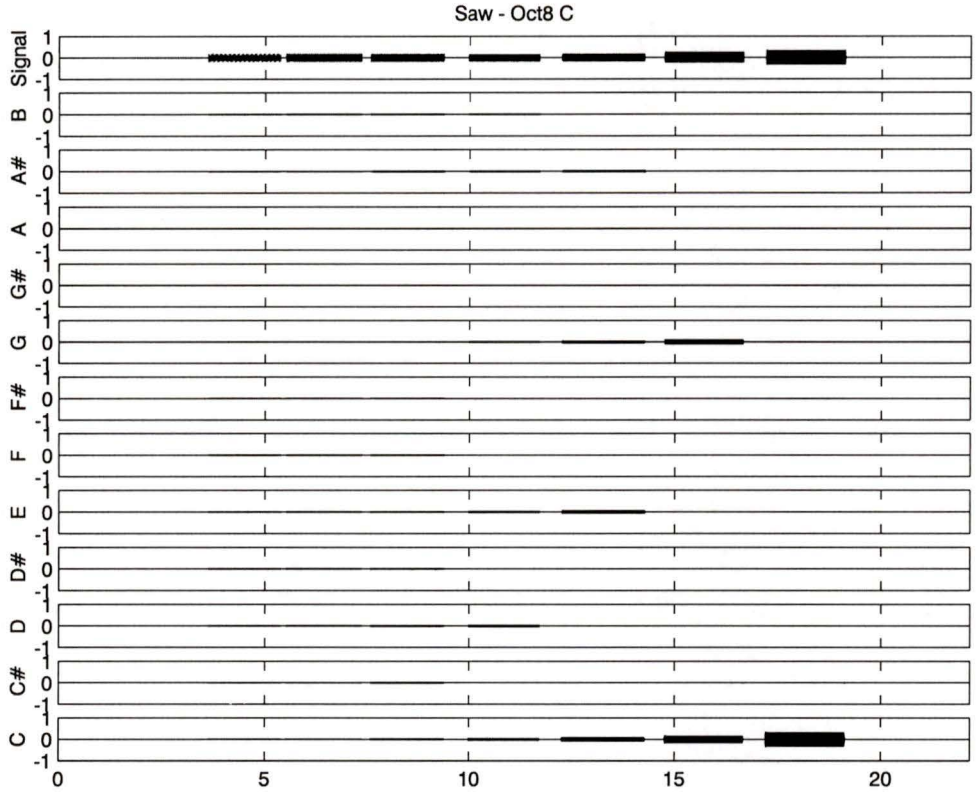


Figure 3.22 Offline-processed output of a 12-SBF bank.

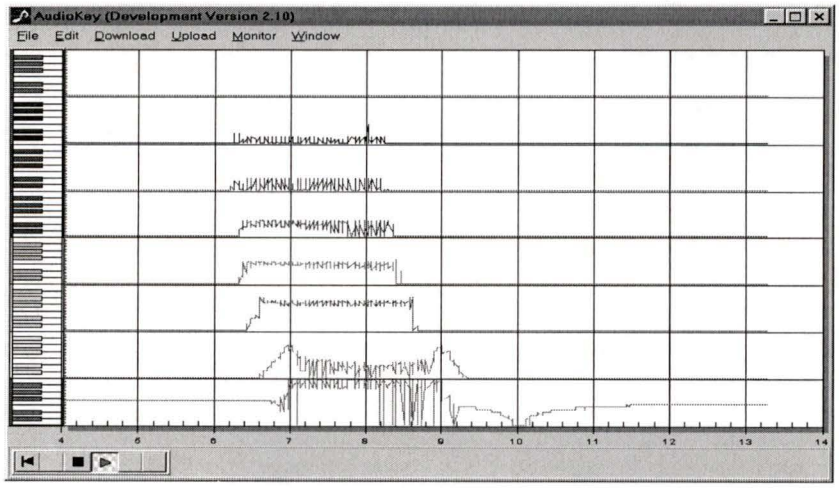


Figure 3.23 Real-time output of an 8-OBF bank.

Another compelling form of output system involves the conversion of semitone output data into MIDI piano-roll information for recording and display within music sequencing software applications such as Cubase [29] and Sonar [30]. Although this form of output is beyond the scope of this work, it is very relevant to the Musical Spectrogram's intended application.

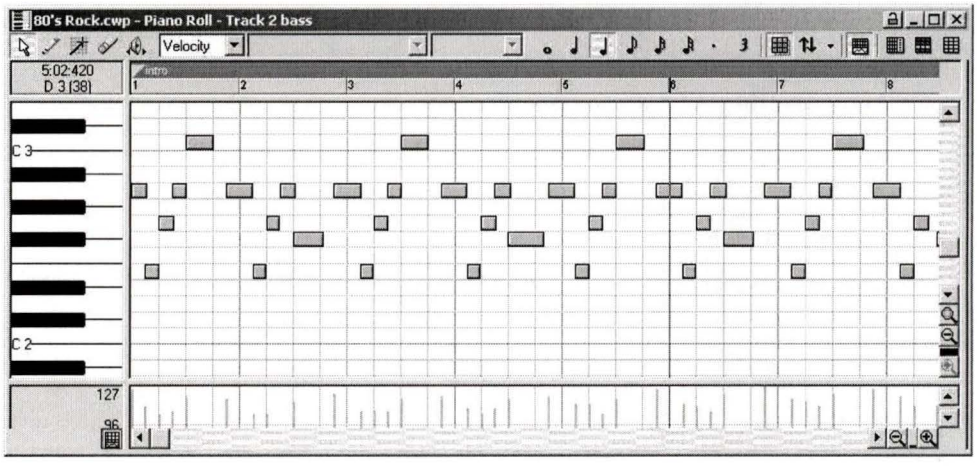


Figure 3.24 Display of MIDI note events within Cakewalk Sonar [30].

3.9 Conclusions

Unlike the Fourier spectrogram, which plots signal spectra along a linear frequency scaled x-axis, the Musical Spectrogram plots signal spectra according to an equal-tempered 12-tone musical scaled y-axis. This makes it a powerful analysis tool for musicians who lack the sufficient scientific background to understand or easily appreciate the analysis tools available to scientists and engineers. The Musical Spectrogram is very efficient. The design is *critically sampled*, which means that every stream of data processed and output by this algorithm is at the lowest possible sampling rate permitted by the sampling theorem (See Figure 3.17). This maximizes the processing speed and minimizes the buffering requirements without any degradation in signal quality. The

architecture of the Musical Spectrogram is also conceptually elegant. A total of three digital filter designs provide the customized bandlimiting and separation required to separate a musical audio signal into 108 perfectly tuned semitone signals.

Chapter 4

Musical Spectrogram Implementations

4.1 Introduction

The implementation of the Musical Spectrogram can take one of three forms

1. Offline file-processor software,
2. real-time DSP audio hardware/firmware, and
3. real-time audio software.

This chapter describes the first of these implementations, a system of off-line file processors written for MATLAB. A real-time DSP implementation was undertaken as part of this work, but as the results will show, the Musical Spectrogram proved to be too computationally intensive to permit a fully functional version on the target hardware. Regardless of these difficulties, the work undertaken on the DSP platform underscored the many issues involved with real-time fixed-point implementations and these are introduced and discussed in this chapter. A real-time audio software version was not undertaken as part of this work, but it is recommended in the future work section of Chapter 5.

4.2 MATLAB Implementation

MATLAB m-files were written, debugged and tested to implement the various stages of the Musical Spectrogram. These files were:

- SRI, the Stirling rational interpolator.
- MS_9OBF, the 9-octave-band filter bank.
- MS_OPM, the octave processing module.
- MS_12SBF, the 12-semitone-band filter bank.
- MS_SPM, the semitone processing module.
- MS_9OBFplot, for plotting the 9-OBF output.
- MS_12SBFplot, for plotting the 12-SBF output.
- MS_12SBPF, the 12 semitone bandpass filter bank.

These files can be found in Appendix I: MATLAB Files. Test signals were created by a software synthesizer and saved as audio wave files. These files were subsequently imported into MATLAB and passed to the MS_9OBFplot program, whose octave outputs were passed to the MS-12SBFplot program. The resulting MATLAB figures are shown in the following section.

4.2.1 Results

The MATLAB implementation was tested with a battery of musical audio signals:

- Ascending sine wave notes spaced in octaves from 1 to 9.
- Ascending sine wave notes spaced chromatically (semitone steps) within each octave 1 to 9.
- Ascending sawtooth wave notes spaced in octaves 1 to 9.

As will be shown, the results demonstrate the Musical Spectrogram's ability to discriminate between notes played anywhere in the 9 octave range supported by this implementation, *provided* the waveform is not composed of many harmonics beyond the fundamental frequency. Such harmonics are also detected by the Musical Spectrogram, which makes determination of the original note increasingly difficult as the number of

dominant harmonics increases.

4.2.2 Sine Wave Results

Composed of only a fundamental frequency, these test signals demonstrate basic functionality in that the fundamental is easily detected by the Musical Spectrogram and plotted without ambiguity as the source of each note. Figure 4.1 illustrates the time and frequency characteristics of a sine wave at frequency 440 Hz. The frequency characteristic at the right shows that the wave is a pure sinusoid with a fundamental but no harmonics.

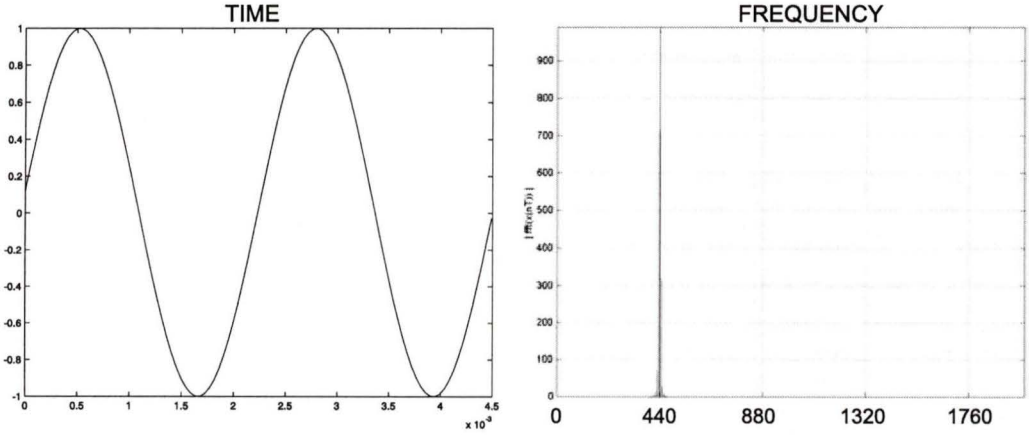


Figure 4.1 Sine wave characteristics.

Figures 4.2 to 4.22 illustrate the effectiveness of the Muscial Spectrogram for sine wave signals. At the head of each figure is a sub-plot labeled *Signal*, which illustrates the wave of the unprocessed musical sound.

Figure 4.2 plots an ascending sequence of 'A' notes in each of the 9 octaves covered by the 9-OBF Bank. As can be seen, each octave of this note is correctly separated.

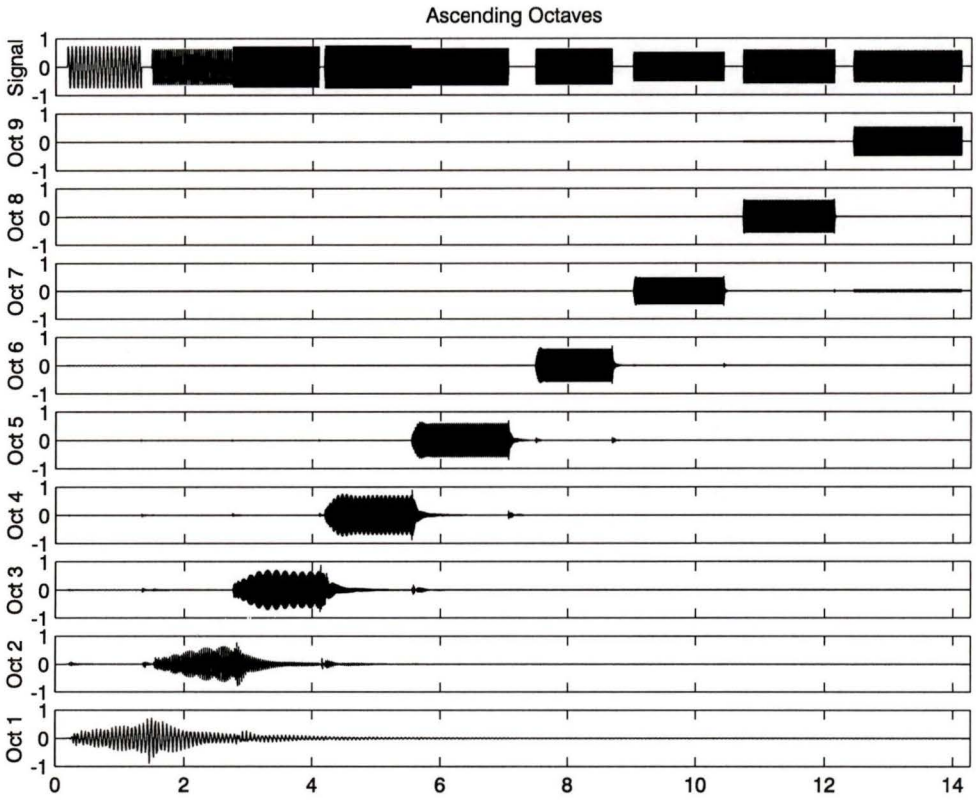


Figure 4.2 Ascending notes across all octaves (sine waves).

Figure 4.3 shows the time/frequency plot of a chromatic progression in octave 2. This progression begins on 'C' and ascends by semitones to 'B'.

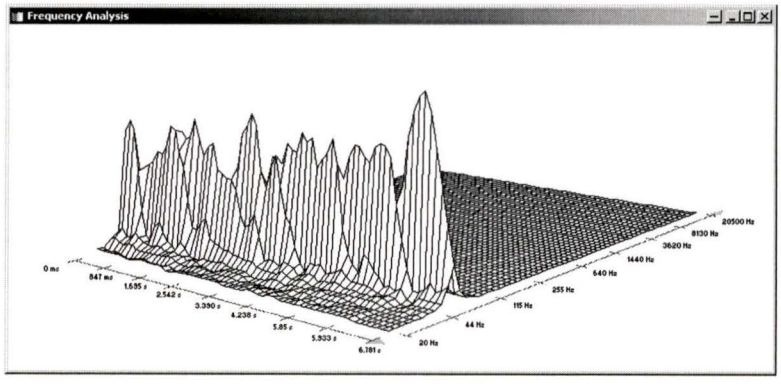


Figure 4.3 Chromatic progression in octave 2.

The 9-OBF bank detects the octave of this sequence in Figure 4.4.

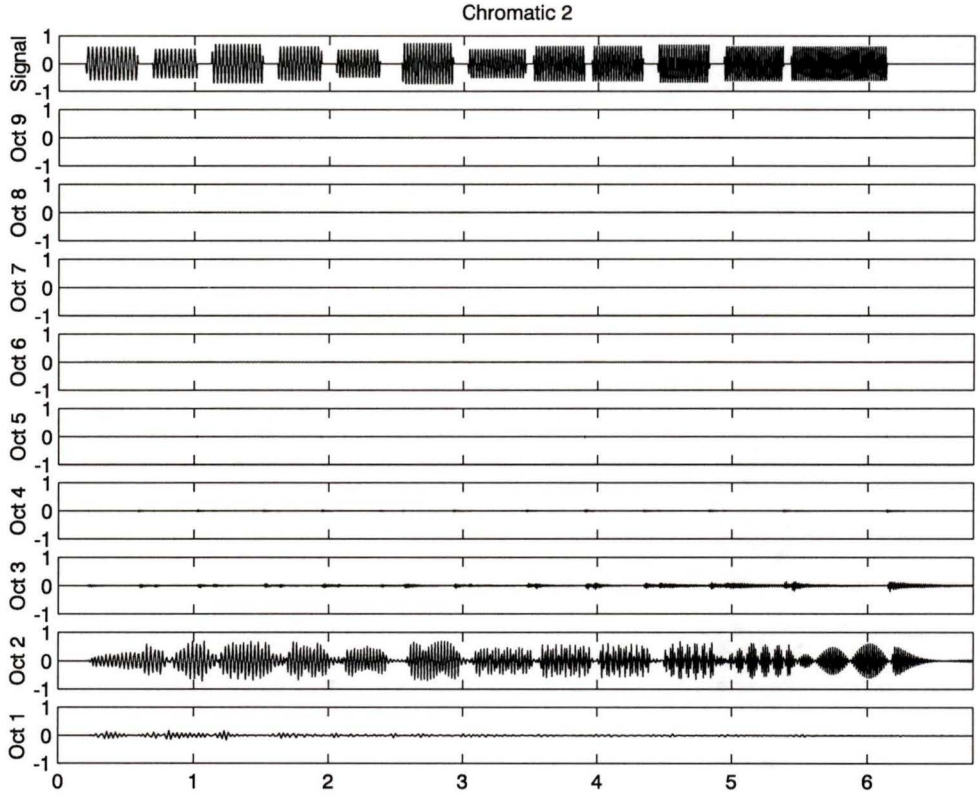


Figure 4.4 Chromatic sequence detected within octave 2.

Figure 4.5 shows how the 12-SBF bank has separated each semitone in this sequence into the correct note band.

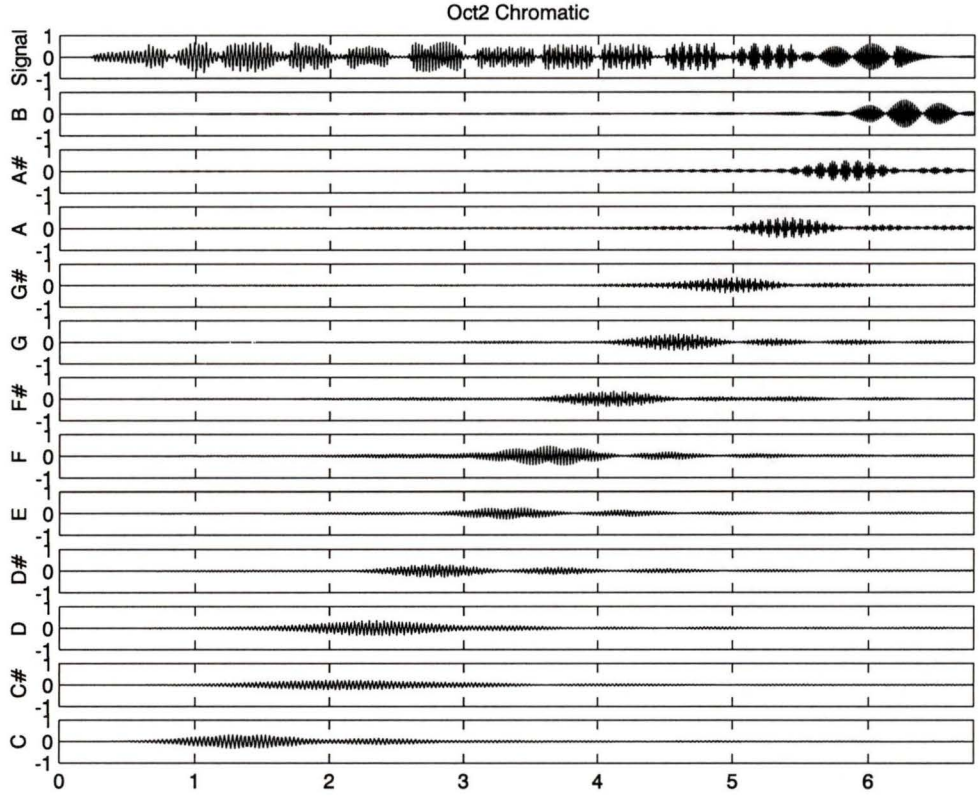


Figure 4.5 Individual semitones in octave 2 chromatic separated.

Figures 4.5 to 4.22 repeat the demonstration of octave 2 for octaves 3-8. Octave 9 is not shown because it is only included in the Musical Spectrogram as a means of detecting harmonics of notes rather than fundamentals. As the frequency (and sampling frequency) of the signals increases, the plotted signal definition improves, which is the expected result at higher octaves bands.

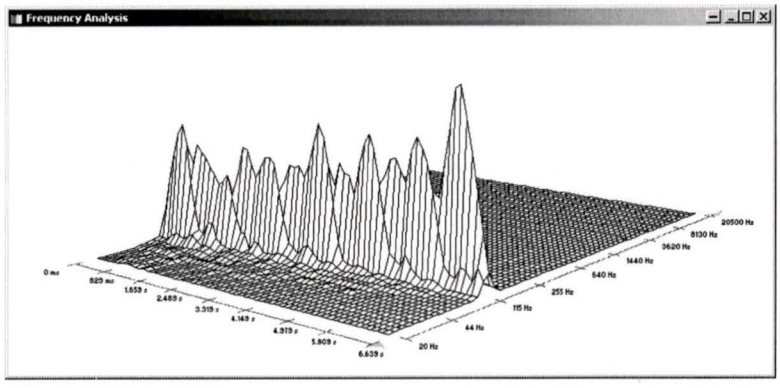


Figure 4.6 Chromatic progression in octave 3.

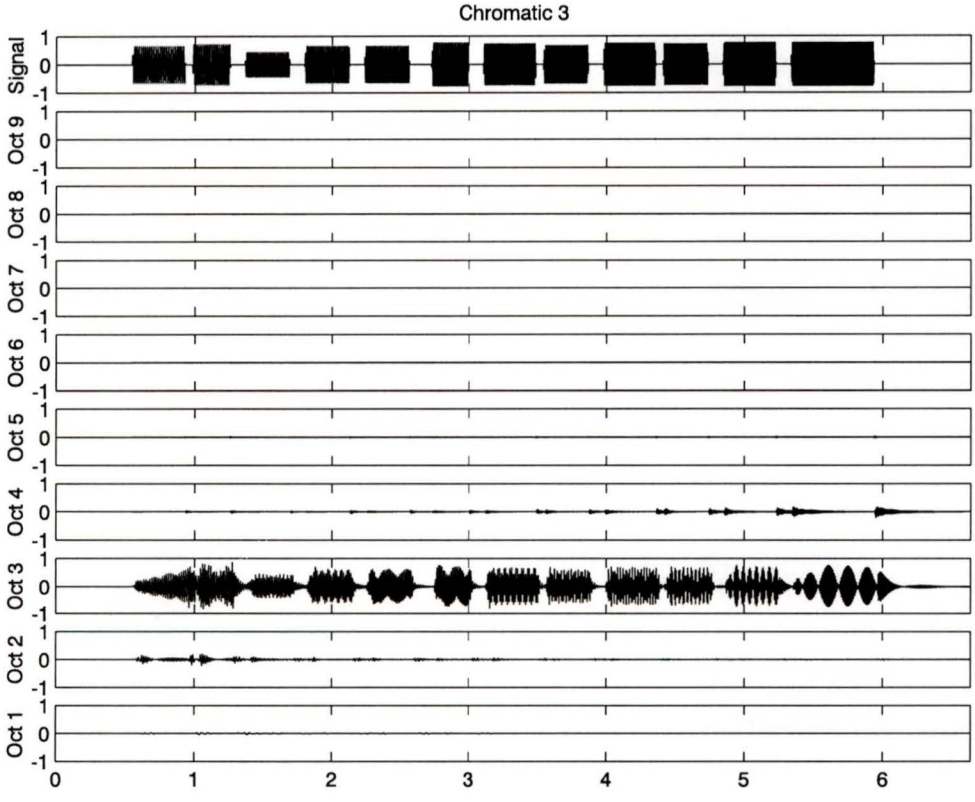


Figure 4.7 Chromatic sequence detected within octave 3.

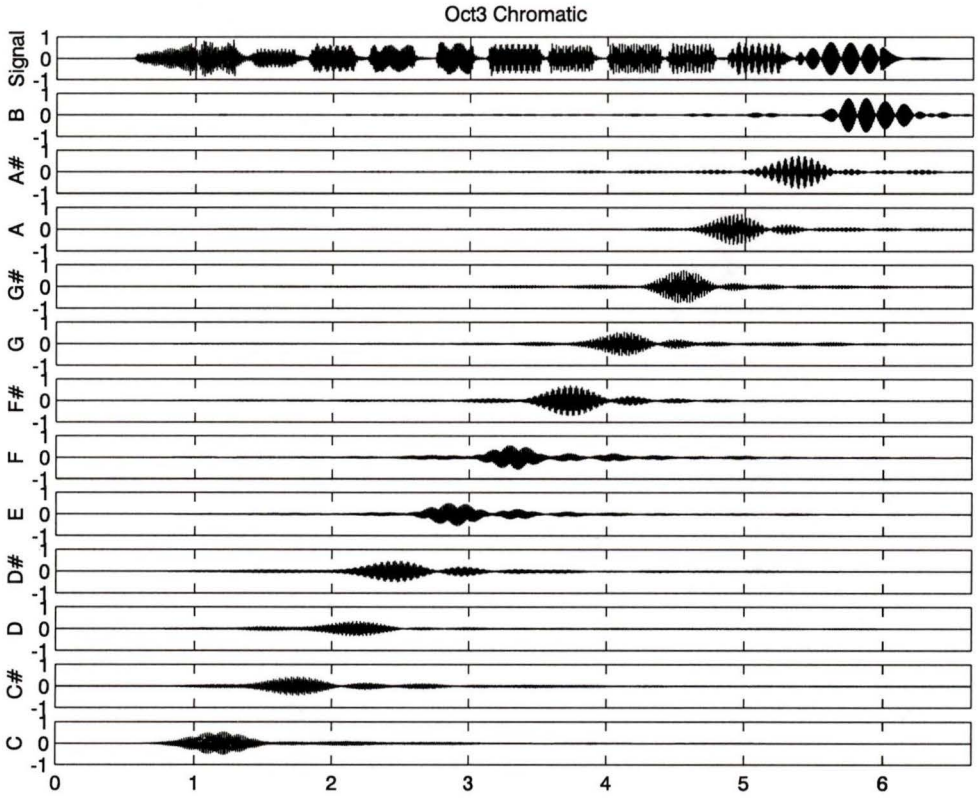


Figure 4.8 Individual semitones in octave 3 chromatic separated.

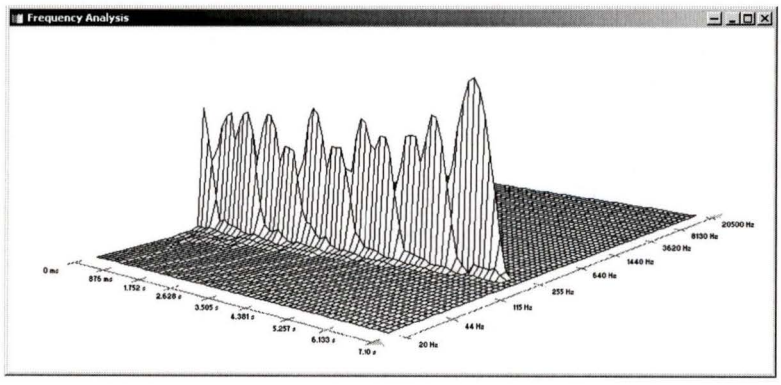


Figure 4.9 Chromatic progression in octave 4.

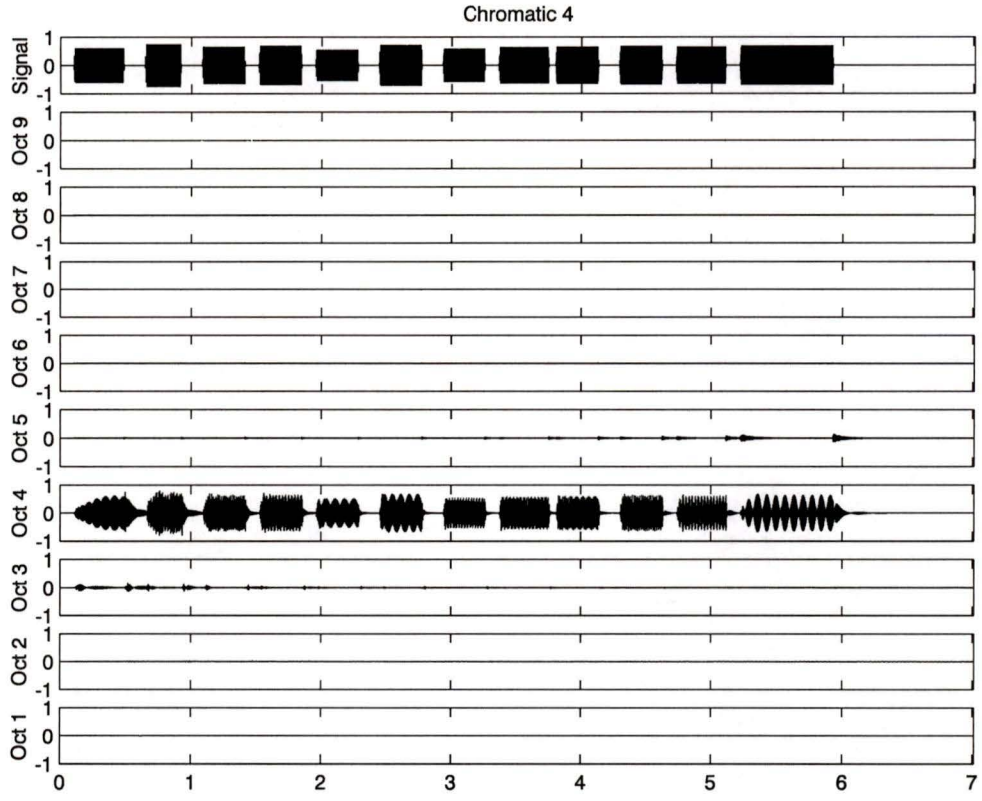


Figure 4.10 Chromatic sequence detected within octave 4.

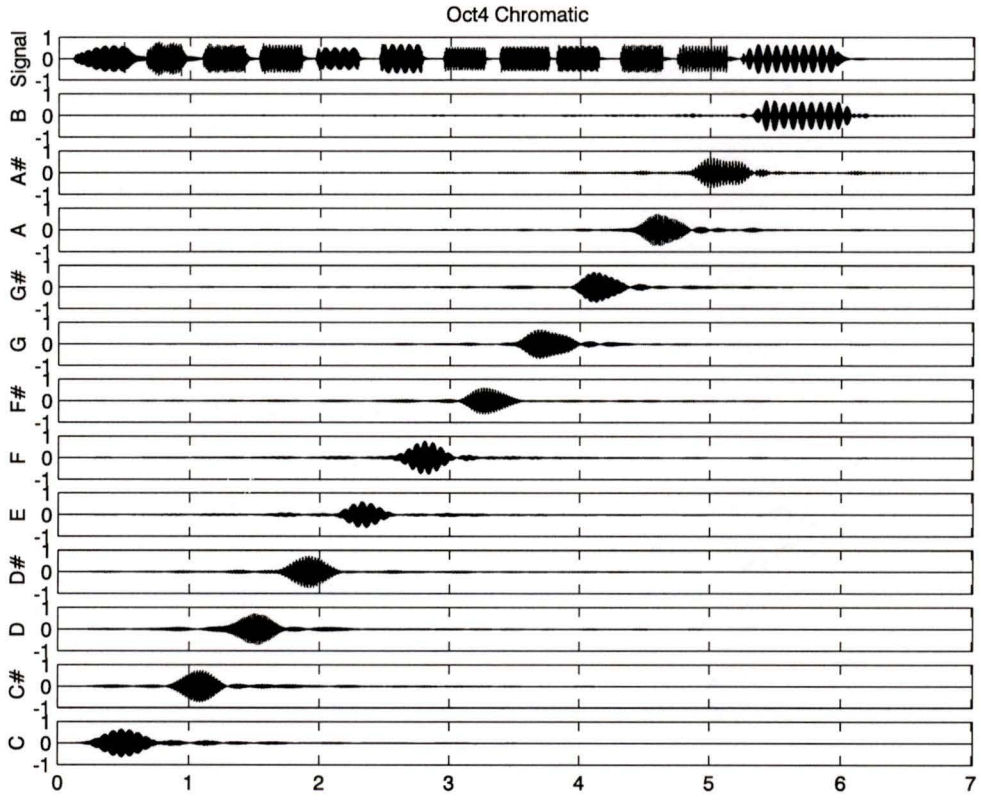


Figure 4.11 Individual semitones in octave 4 chromatic separated.

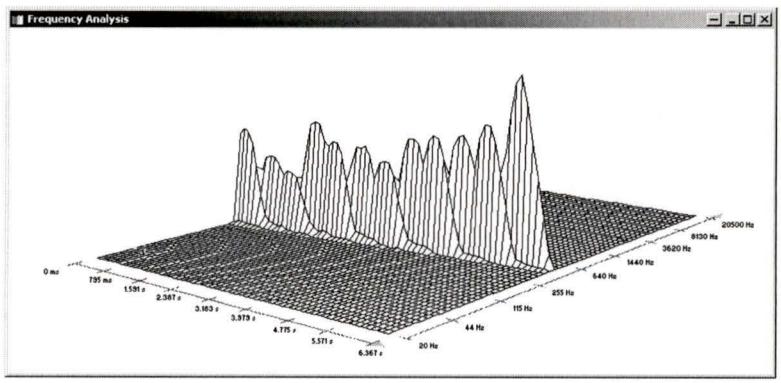


Figure 4.12 Chromatic progression in octave 5.

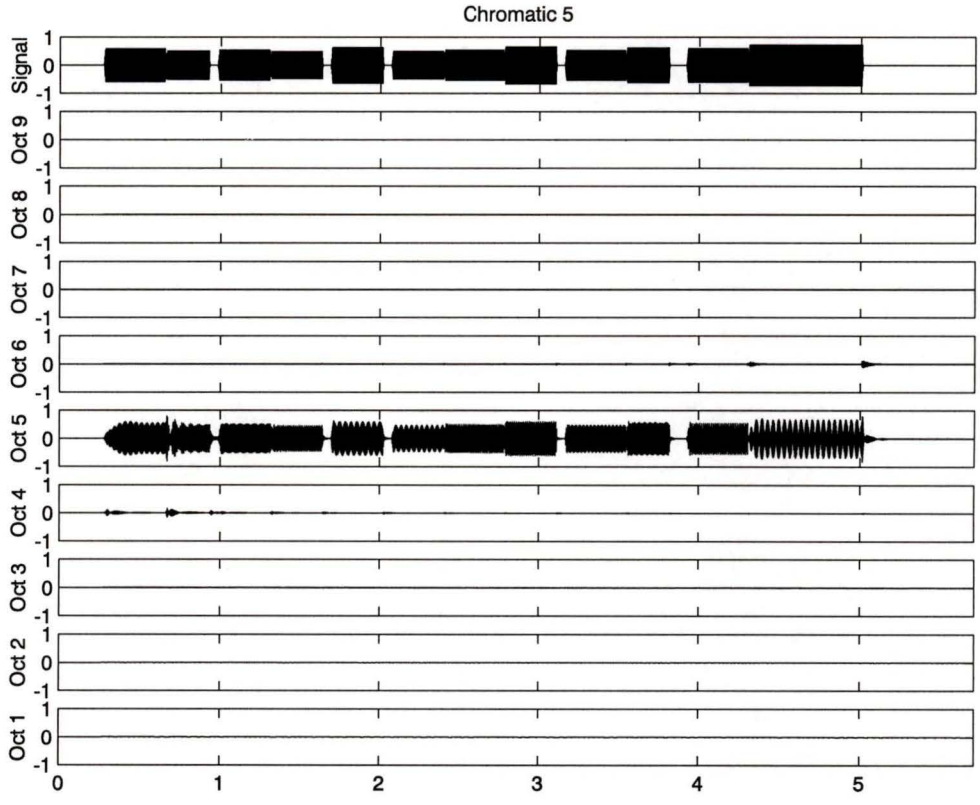


Figure 4.13 Chromatic sequence detected within octave 5.

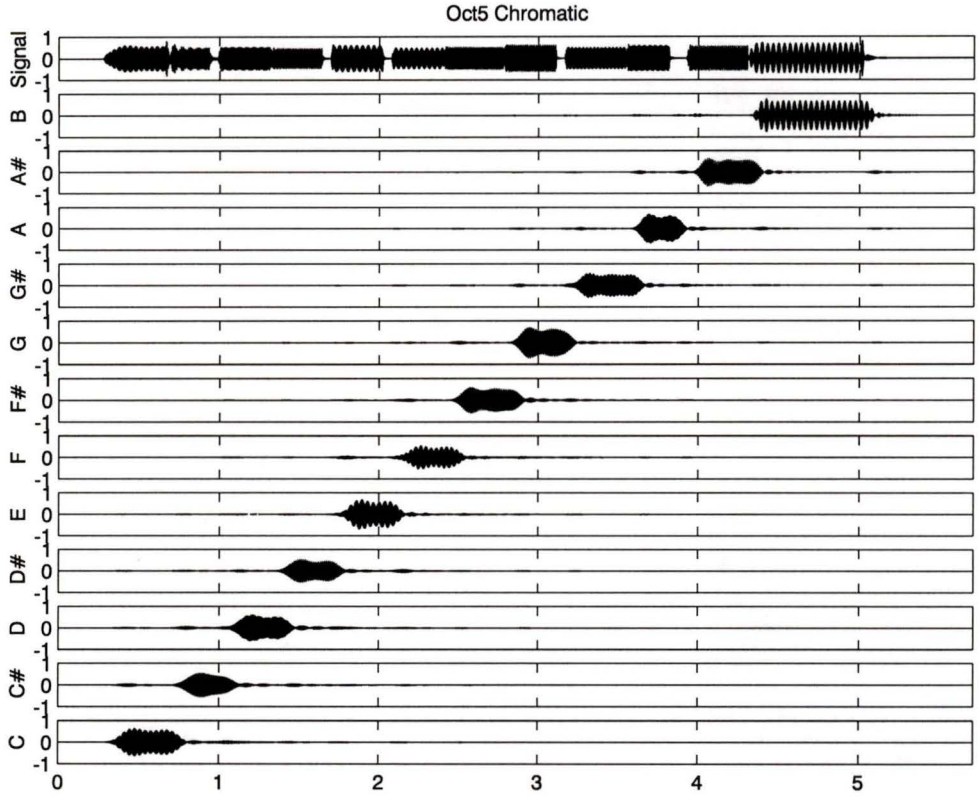


Figure 4.14 Individual semitones in octave 5 chromatic separated.

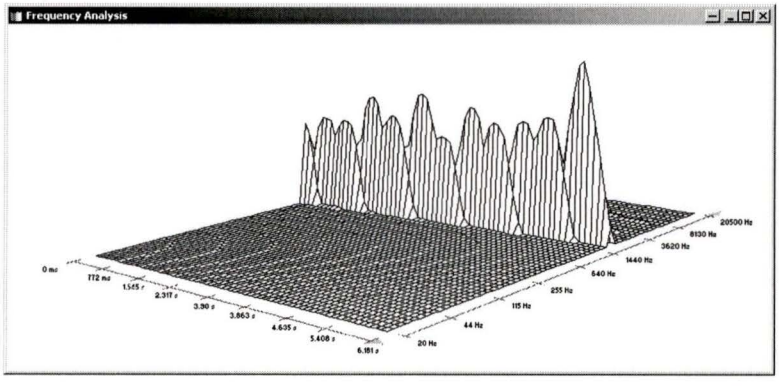


Figure 4.15 Chromatic progression in octave 6.

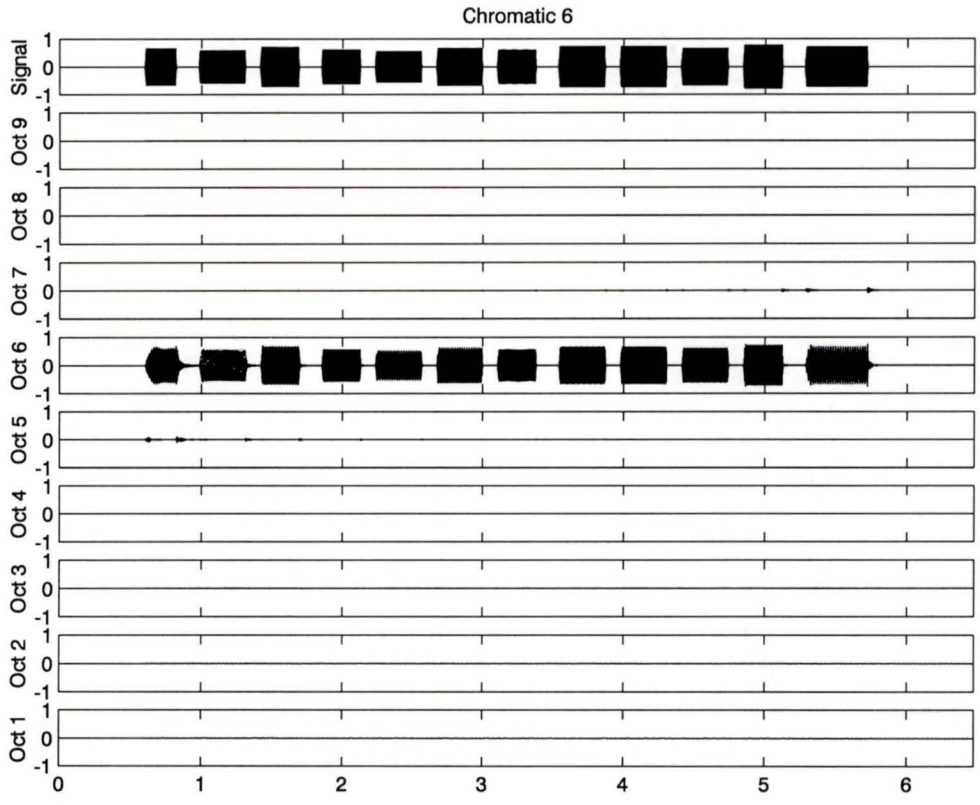


Figure 4.16 Chromatic sequence detected within octave 6.

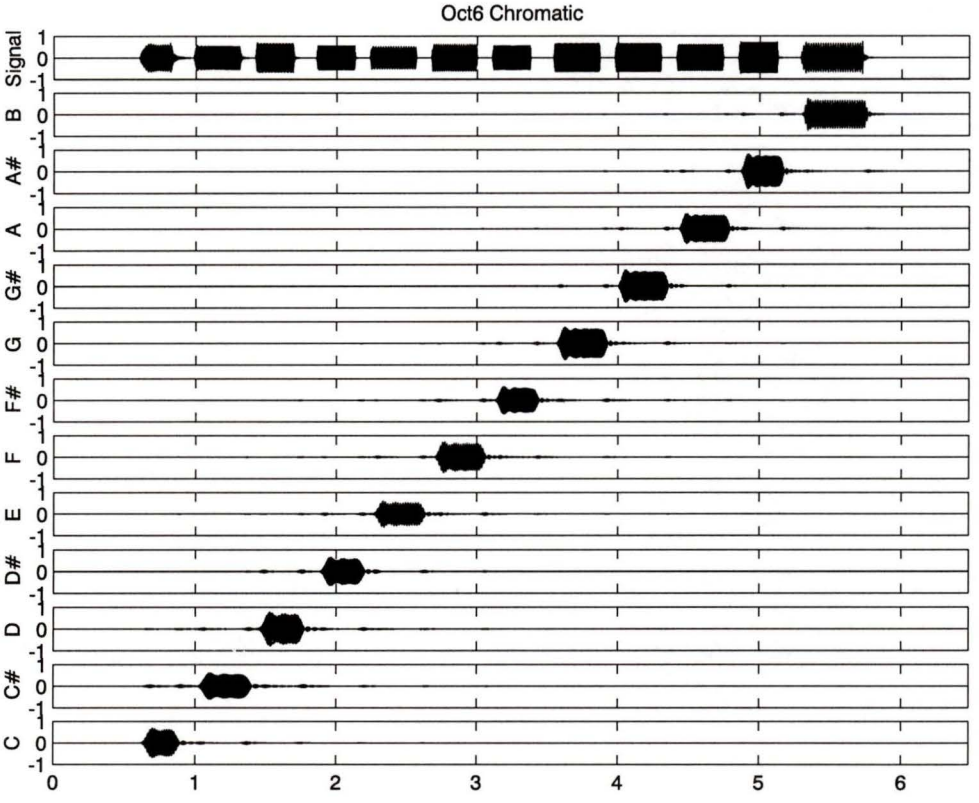


Figure 4.17 Individual semitones in octave 6 chromatic separated.

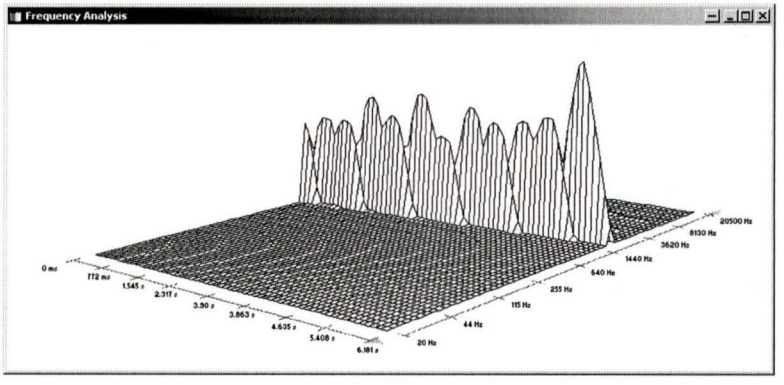


Figure 4.18 Chromatic progression in octave 7.

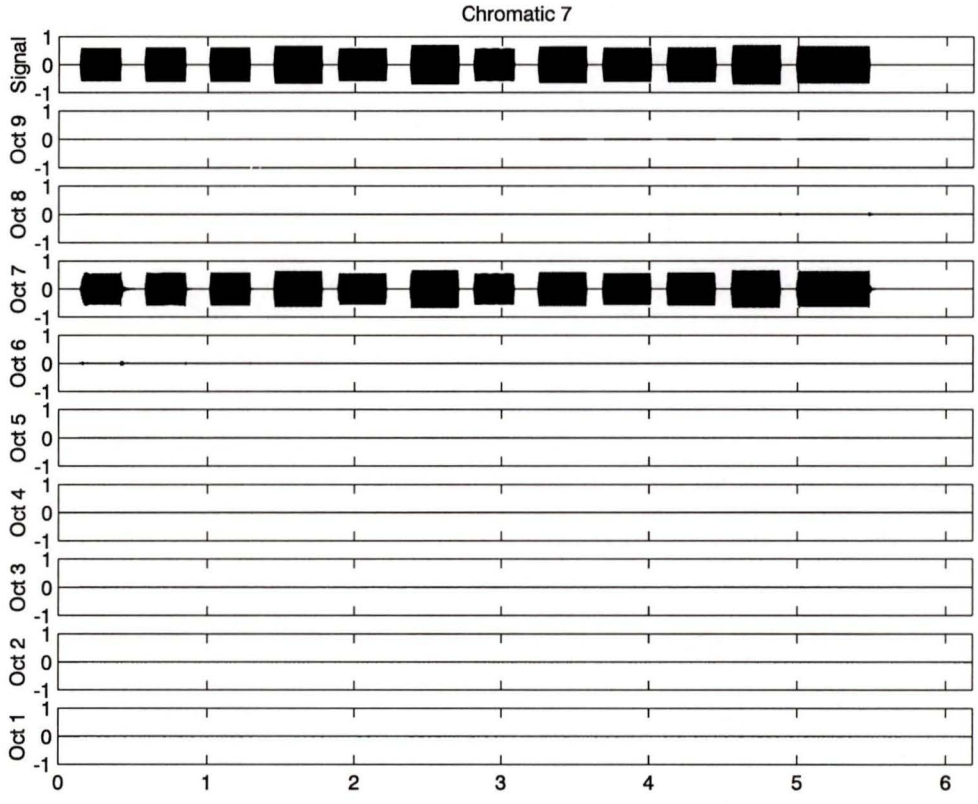


Figure 4.19 Chromatic sequence detected within octave 7.

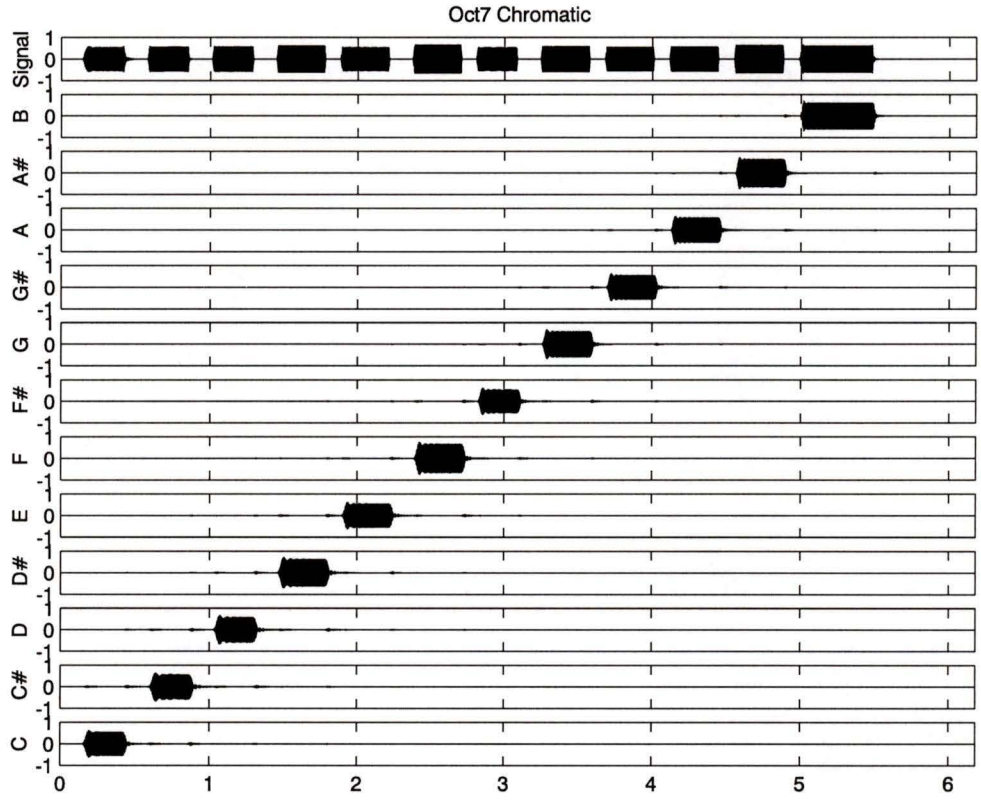


Figure 4.20 Individual semitones in octave 7 chromatic separated.

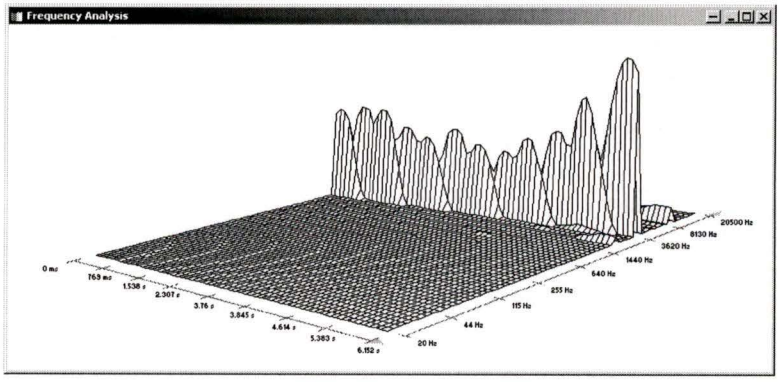


Figure 4.21 Chromatic progression in octave 8.

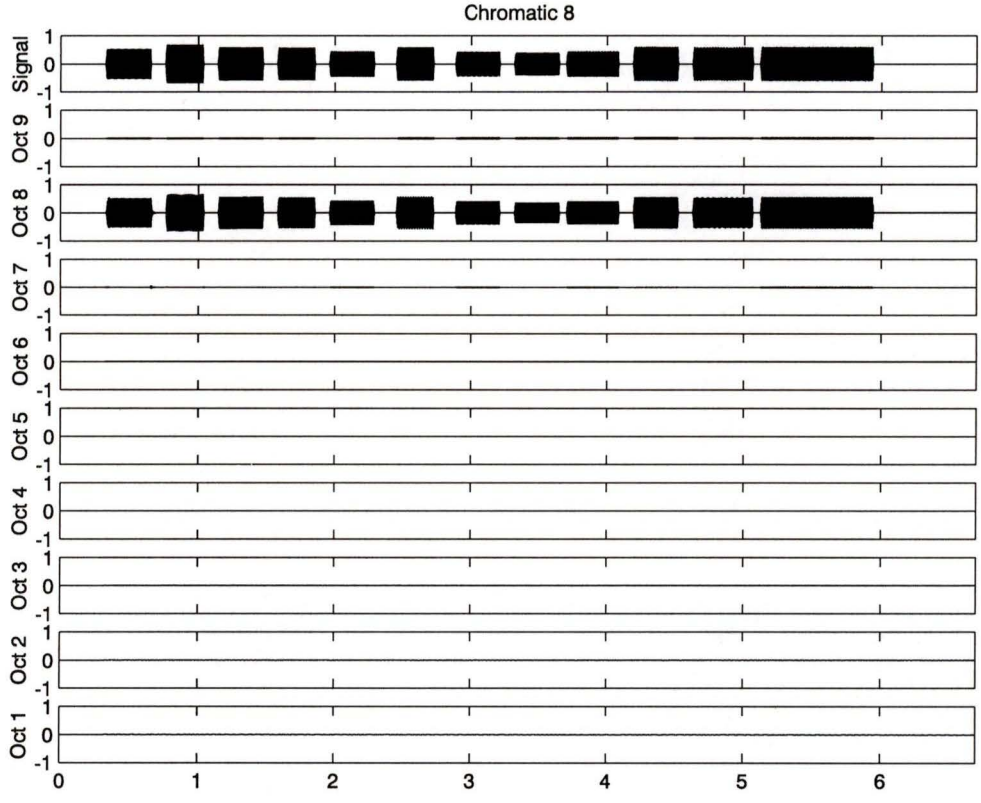


Figure 4.22 Chromatic sequence detected within octave 8.

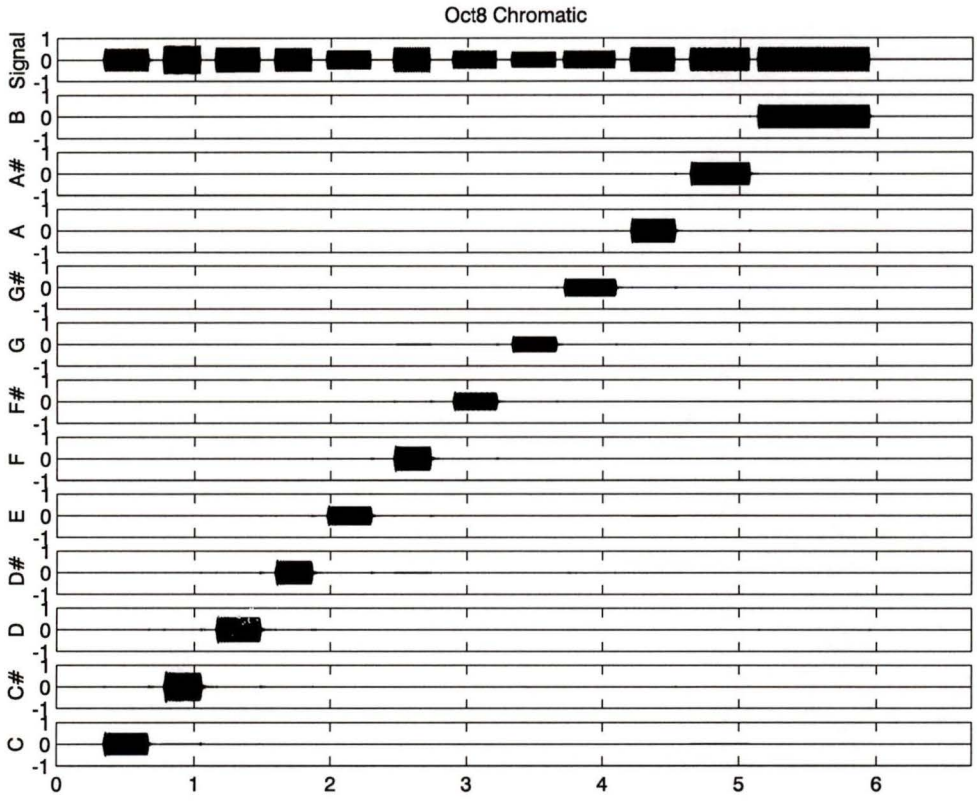


Figure 4.23 Individual semitones in octave 8 chromatic separated.

4.2.3 Sawtooth Wave Results

Unlike sine waves, sawtooth waves are characterized by a fundamental and an infinite number of harmonics. Figure 4.24 illustrates the time and frequency characteristics of a sawtooth wave at frequency A-440 Hz, and as can be seen, harmonics of diminishing strength exist at higher multiples of the fundamental frequency. This renders the Musical Spectrogram less effective as a note detector, since ambiguities are present in the analyses presented by Figures 4.25 to 4.34. It can nonetheless be seen that the Musical Spectrogram correctly separates and plots the harmonics of each note played.

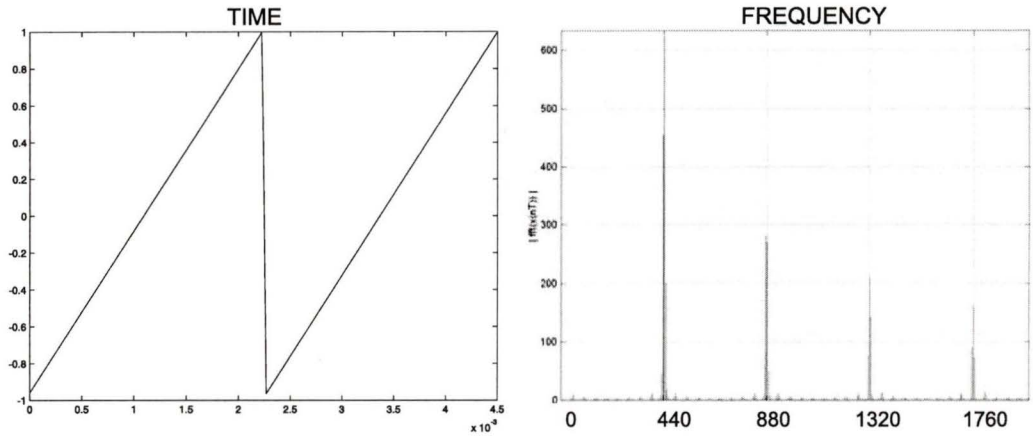


Figure 4.24 Sawtooth wave characteristics.

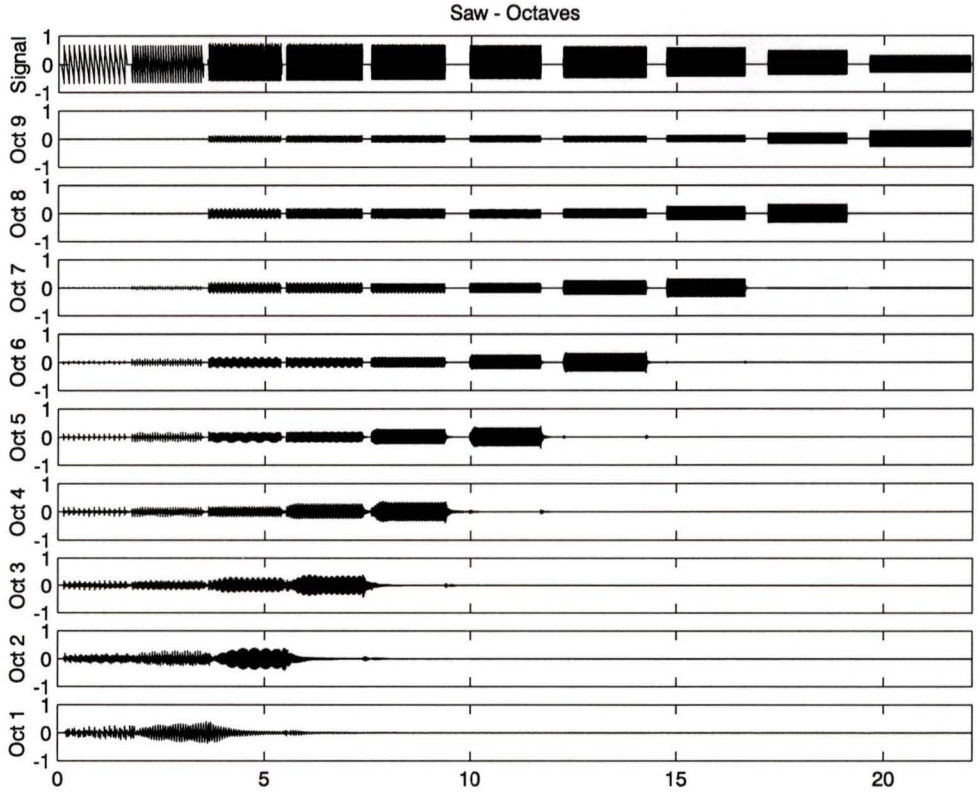


Figure 4.25 Ascending notes across all octaves (sawtooth waves).

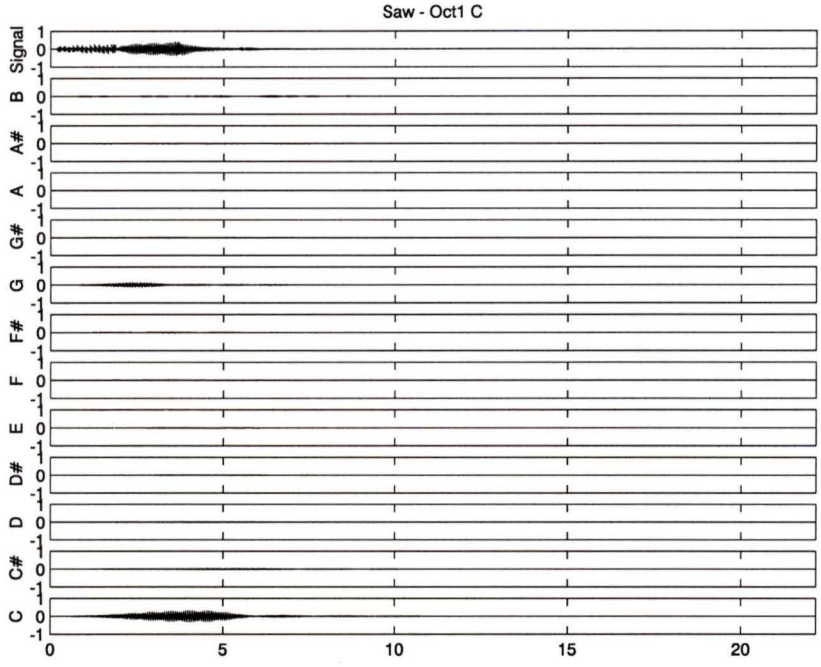


Figure 4.26 Octave 1 'C' note.

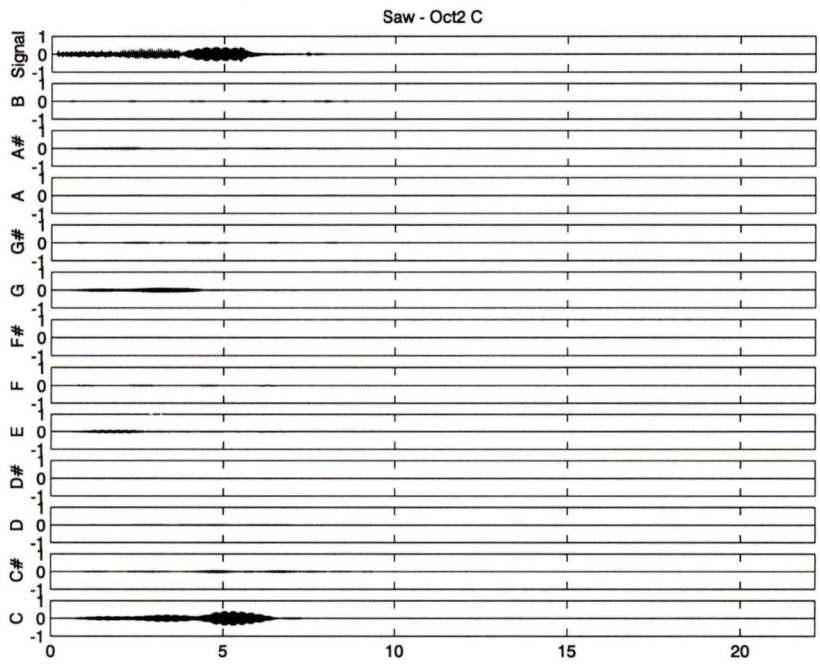


Figure 4.27 Octave 2 'C' note and harmonic from octave 1.

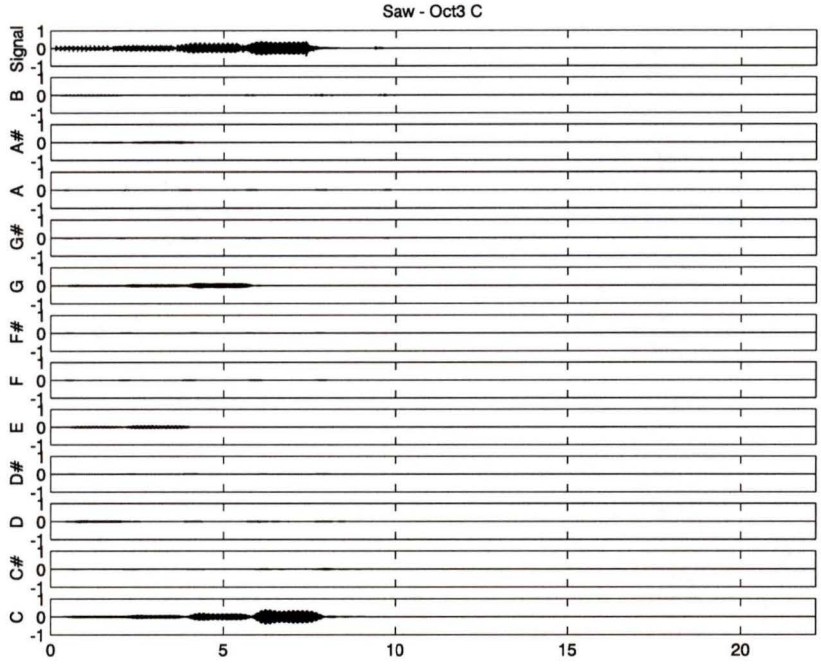


Figure 4.28 Octave 3 'C' note and harmonics from octaves 1 and 2.

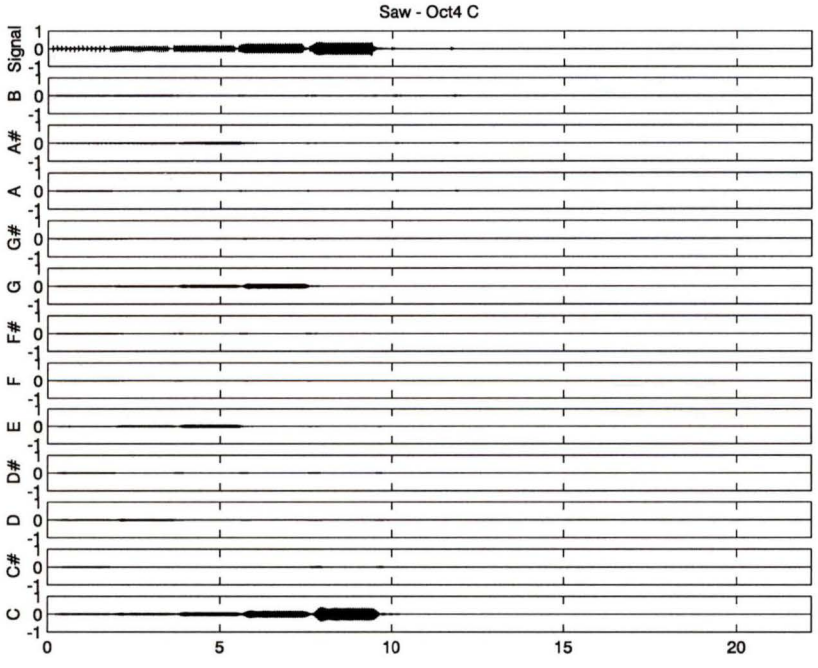


Figure 4.29 Octave 4 'C' note and harmonics from octaves 1, 2 and 3.

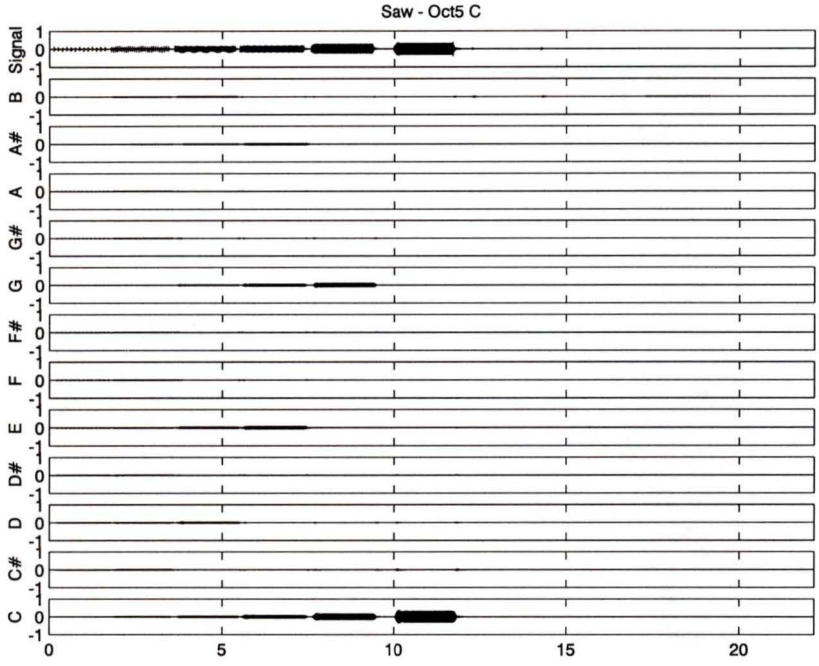


Figure 4.30 Octave 5 'C' note and harmonics from octaves 1, 2, 3 and 4.

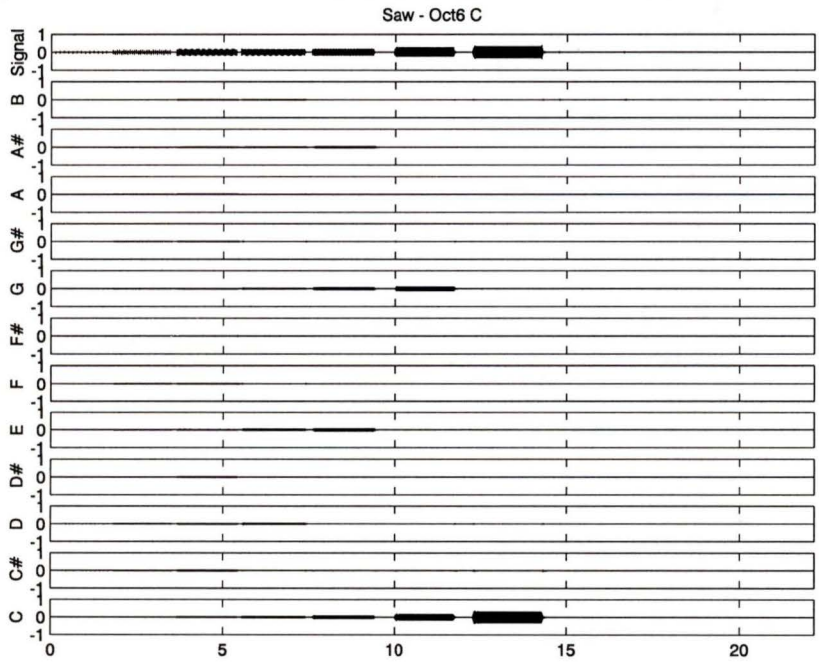


Figure 4.31 Octave 6 'C' note and harmonics from octaves 1, 2, 3, 4 and 5.

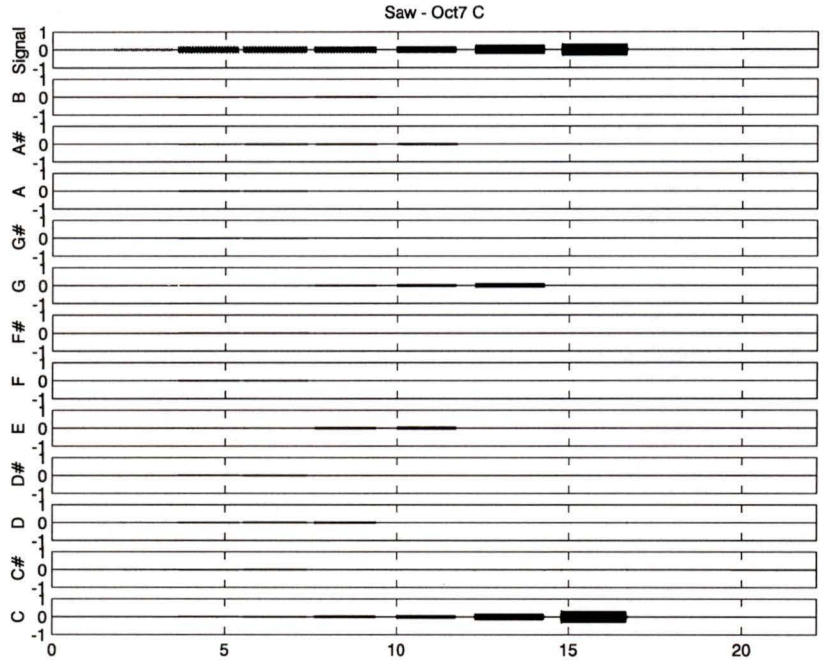


Figure 4.32 Octave 7 'C' note and harmonics from octaves 2, 3, 4, 5 and 6.

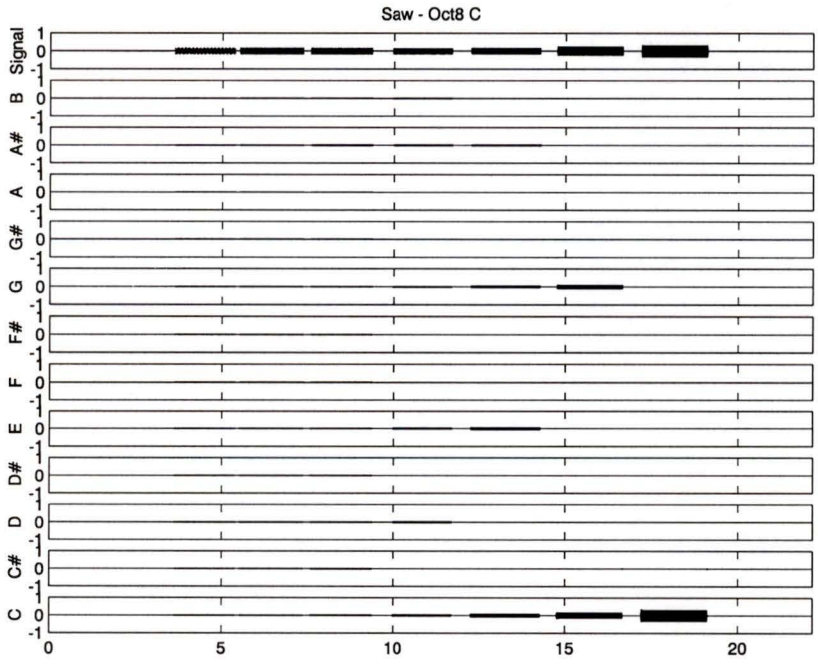


Figure 4.33 Octave 8 'C' note and harmonics from octaves 2, 3, 4, 5, 6 and 7.

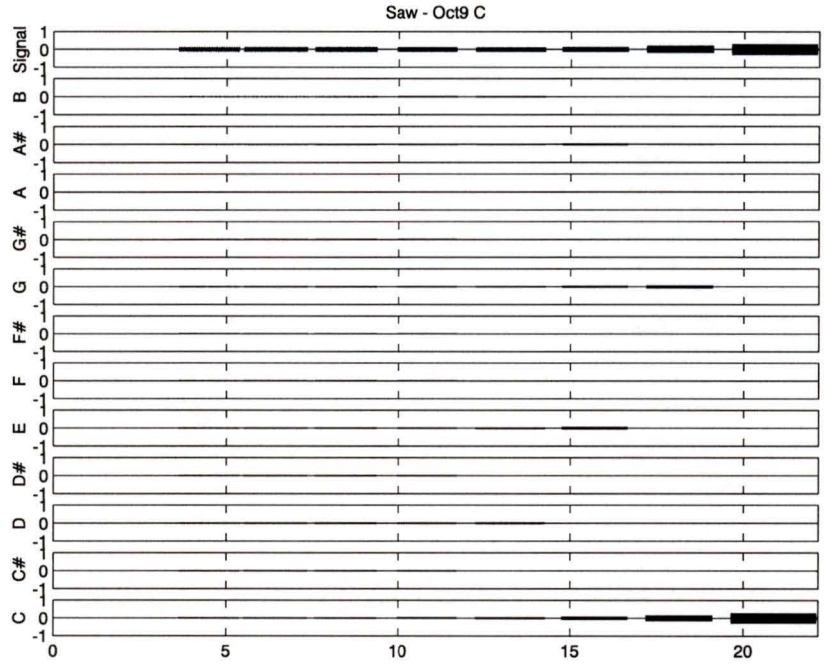


Figure 4.34 Octave 9 'C' note and harmonics from octaves 2, 3, 4, 5, 6, 7 and 8.

4.2.4 Polyphonic Test Results

In order to evaluate the Musical Spectrogram's polyphonic pitch detection, a 7 second passage of polyphonic music was performed on a sine wave oscillator and processed. Figure 4.35 shows the output of the 9-OBF bank, in which it is clear that the polyphonic passage consists of notes within octaves 5, 6 and 7. Figure 4.36 shows the MIDI notes that were played in this passage, which can be compared to Figure 4.37, which shows the combined output plots produced by 12-SBF bank when each of these identified octave signals were processed. As can be seen, the accuracy of the Musical Spectrogram is 100% with regards to the detection of each note's pitch. The transient response of the system seems stable, but further testing is recommended for future work, as is a rigorous examination of the timing accuracy of the Musical Spectrogram.

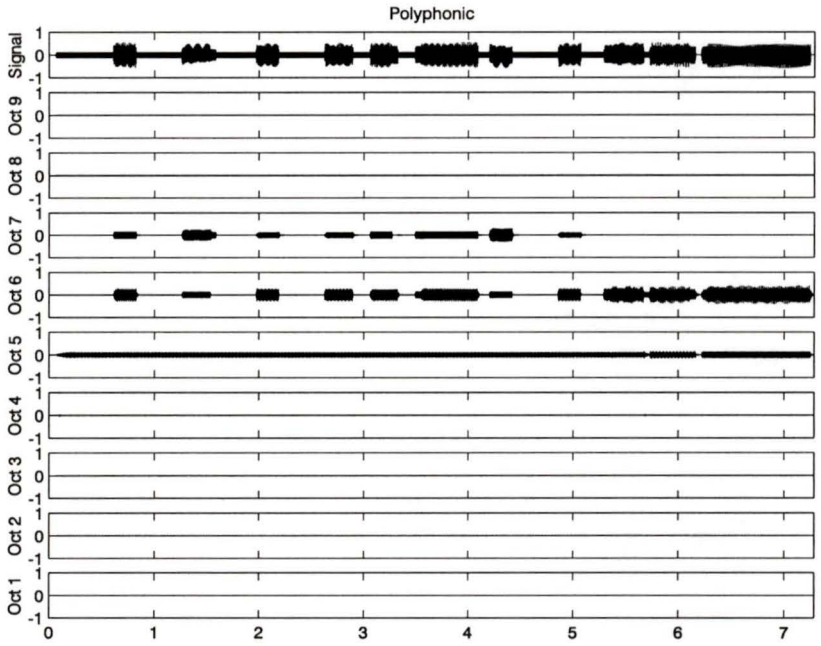


Figure 4.35 Polyphonic music separated into octaves by 9-OBF bank.

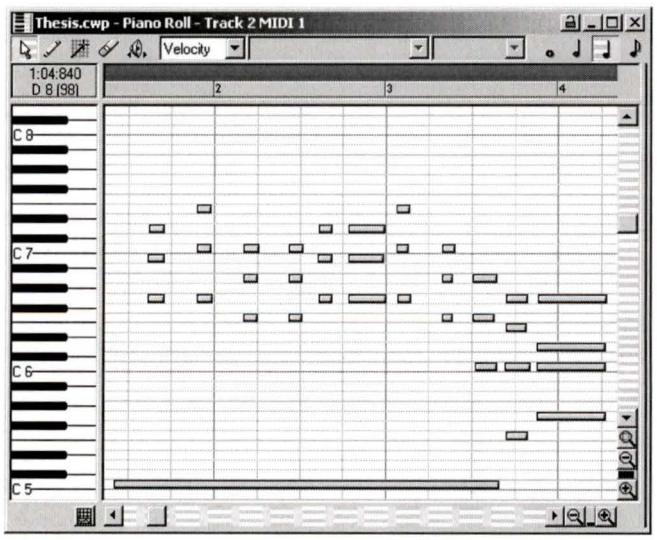


Figure 4.36 Polyphonic passage as MIDI piano roll.

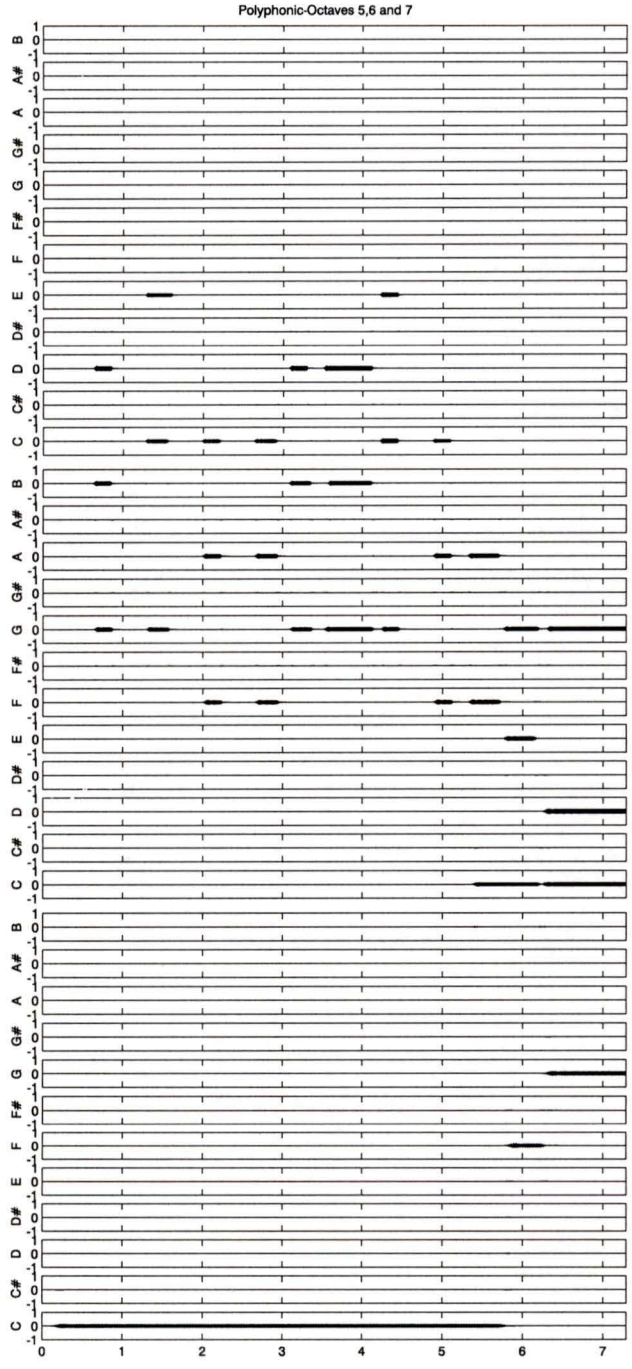


Figure 4.37 Polyphonic music separated into semitones by 12-SBF banks.

4.3 DSP Implementation Issues

A real-time implementation of the Musical Spectrogram was written for the Analog Devices ADSP-2181 fixed-point processor. While this implementation's technical features are not consistent with the MATLAB implementation detailed here, it served as an important phase of the algorithm's development. In addition, the many difficulties encountered served to underscore the issues involved with fixed-point DSP development.

4.3.1 ADSP 2181 EZ-Kit Lite

The Analog Devices EZ-Kit Lite is an inexpensive evaluation board system for the ADSP-2181 processor. This evaluation board connects via serial cable to a host PC running the Windows operating system and includes PC software that allows the editing, compilation, and downloading of DSP programs to the EZ-Kit for execution. Various programs relevant to the Musical Spectrogram were developed with this system:

- 5-octave.dsp, a 5-octave real-time DWT with audible output.
- Wavelet7.dsp, a 7-octave real-time DWT with serial link.
- AudioKey.dsp, an 8-octave limited real-time Musical Spectrogram with companion host monitor program for Windows.

Although this platform provided a very educational experience, it was not adequate for running a real-time Musical Spectrogram for the following reasons:

- Universal asynchronous receiver/transmitter (UART) chip support for the serial port was not included. All UART functionality was performed in software by the DSP which robbed the processor of cycles and resources while communicating to the host.

- The transfer rate of the serial connection was fixed at 9600 baud which was insufficient bandwidth for transmitting semitone signal data in real-time without implementing additional compression schemes.
- No in-circuit emulation capabilities are included in the EZ-Kit Lite and its simulator was too primitive. Better tools were required to debug an algorithm as large and sophisticated as the Musical Spectrogram.[1]
- Insufficient speed and resources to implement the 108 high order IIR filters used by the 12-SBF bank array.

The resulting DSP programs were limited in numerous ways. Low-ordered FIR filter design substitutions were used where high-order IIR filters were called for, the sampling rate and octaves under analysis were reduced, and output plotting capabilities were only achieved for the octave decompositions. Regardless, many important lessons were learned in the process of developing these real-time programs that were applied to the MATLAB implementation.

4.3.2 The Limitations of Fixed-Point Representation

The 1.15 fixed point notation limits the numerical range of data to between -1 and $1 - 2^{-15}$ $(-1, 0.99996948)$. This means that filter coefficients and signal data must be scaled within this range to avoid critical processing errors. Every mathematical operation that occurs within a fixed-point system is subject to the dangers of underflow and overflow errors, so great care must be taken to ensure they do not occur.

4.3.3 IIR Filter Fixed-Point Implementation Strategies

The implementation of high-order recursive IIR filters was handled by structuring the filter into biquadratic (*biquad*) stages whose sensitivity to coefficient quantization is much lower than direct forms based on their N^{th} order rational polynomial transfer

1. Tools of this caliber have now been released, but unfortunately too late for any of this work to have benefited from them. Future work is recommended.

functions. These structures are then connected in cascade realization to realize the complete filter. In fixed-point architectures such as the ADSP-2181, such a strategy was essential and was coupled with the techniques described in the following sections.

4.3.3.1 Constraints on Filter Coefficient Size

Filter designs yielded by the prescribed specifications method are characterized by filter coefficients that are not limited to any particular range. Before such designs may be implemented on a fixed-point system they must be scaled to prevent overflow of the multiplier/accumulator (MAC) unit of the processor's artihmentic logic unit (ALU) during processing and then compensated for when the signal is converted to analog output. Accordingly, the transfer functions of each biquad stage were scaled to yield values that could be represented in the ADSP-2181's 1.15 fixed-point environment.

4.3.3.2 Signal Scaling Between Second-Order Stages

In addition to the filter coefficient scaling that must occur, signal data passing through fixed-point filter implementations must also be scaled so that each stage's output signal does not exceed the limit of fixed-point representation. This involves the computation of an L_2 norm of each filter's magnitude response and applying the inverse of this value as a signal scale factor prior to processing.

4.3.3.3 Matched Pole and Zero Technique

The signal scaling described in the previous section can yield scale factors that are too large for fixed-point representation. A technique that reorganizes the poles and zeroes of the filter transfer function into close-proximity pairs was applied to yield biquad stages whose unscaled gain was as close to unity as possible. This assured that individual signal scale factors computed were minimized and eliminated where possible.

4.3.3.4 Recursive Filter Pole Quantization and Stability

The feedback loop inherent in recursive IIR filter designs is characterized by transfer function poles that are usually extremely close to the unit circle on the z-plane. These poles can be shifted beyond this design boundary when quantized into fixed-point representation, which would render the filter unstable. To assure this did not occur, the quantized transfer function was tested in MATLAB and found to be both stable and to yield the required frequency response before it was approved.

4.3.4 Benefits of the DSP Implementation

While the DSP implementation work did not yield a complete or fully functional version of the Musical Spectrogram, it was invaluable in the verification of the underlying theories. In addition, very marketable DSP and C++ windows application programming skills and experience were attained in the process.

4.4 Conclusions

The Musical Spectrogram was sucessfully implemented and tested as an off-line file processor within MATLAB. Test results obtained from this implementation proved that the Musical Spectrogram is an effective and musically meaningful tool for viewing the spectral content of music signals, and that it also is capable of polyphonic pitch detection when the harmonic structure of the source is not complex. As the first stage of processing for more complex mid-level musical representations, its performance is exceptional.

Much effort was invested into a DSP implementation but for various unexpected technical difficulties this implementation was not completed. However, this approach could be pursued in the future as part of a product development on a faster, more integrated DSP platform, and is recommended for future work.

Chapter 5

Conclusions

5.1 Results

Consisting of a tuning module, a 9 octave-band filter bank, a 12 semitone-band filter bank array and an output system, the Musical Spectrogram was introduced and developed. New insights were offered by exploring the musical spectrograms of numerous test signals, and the commercial potentials of this new analysis tool were explored.

The Musical Spectrogram is an efficient multi-rate algorithm. The design is *critically sampled*, which means that every stream of data processed and output by this algorithm is at the lowest possible sampling rate permitted by the sampling theorem. This maximizes the processing speed and minimizes the buffering requirements without practical degradation of the signal quality. The architecture of the Musical Spectrogram is also conceptually elegant. A total of three digital filter designs provide the customized bandlimiting and separation required to separate a musical audio signal into 108 perfectly tuned semitone signals.

The main disadvantage inherent in the proposed Musical Spectrogram is the lack of a synthesis component. Any parameters derived by this analysis cannot therefore be amalgamated to reconstruct the original audio. This is a consequence of the use of elliptic filters in the Musical Spectrogram design, which was deemed to be an acceptable trade-off, given the far greater selectivity and efficiency of this filter type.

A complete MATLAB implementation of the Musical Spectrogram was written

and tested to prove the function and form of this proposed system. A limited real-time implementation of the Musical Spectrogram was also developed for the Analog Devices ADSP-2181 fixed-point digital signal processor platform, which underscored many of the difficulties inherent in development for such low-cost platforms. The numerous advantages and properties of the Musical Spectrogram system were also presented and discussed.

5.2 Recommendations for Further Research

When the Musical Spectrogram was first conceived in 1995, real-time audio processing systems were still proprietary and expensive. Audio DSP hardware was complex and rarely found in consumer technology. Affordable personal computers were barely able to perform CD sampling-rate recording and playback without hardware accelerator cards.

Seven years later, so much has changed. Real-time DSP consumer audio systems are commonplace in the form of MP3 players, cellular phones and countless other products, and audio DSP hardware is available in affordable evaluation board systems bundled with visual high-level language support and in multi-processor accelerator card development systems for under $1000.00. Personal computers have become not only powerful enough to be digital hard disk recording systems, they are also capable of simulating multiple instances of hardware products while performing to these tasks.

5.2.1 Software Plugins

Today's personal computers are fast enough to run demanding algorithms like the Musical Spectrogram as real-time software. They are even capable of running very large real-time music production applications such as **Cubase** [35] and **Sonar** [36] that "host" smaller programs that "plug" into them through a software interface. The Musical Spectrogram is an ideal candidate for further development as a plugin, so the

advantages of this type of program are now described.

Plugin developers concentrate on the processing engine and user interface of their application while the host program handles everything else. Plugins may be universally run within any compatible host program, which also enhances their market value. Currently, there are three competing plugin standards on the Microsoft Windows PC platform: *DirectX*, *DXi* and *VSTi*. Figure 5.1 shows the user interface panel of a DirectX plugin included with Sonar that applies reverberation effects to digital audio signals.

Figure 5.1 DirectX reverb effect plugin by Cakewalk.

Plugins are not restricted to audio effects processing. They can even be musical instruments in their own right. In these cases, the plugin accepts MIDI events and other control parameters from the host and returns the audio signals generated by the plugin. Figure 5.2 shows the LM-9, which is a VSTi drum machine plugin included with

Cubase.

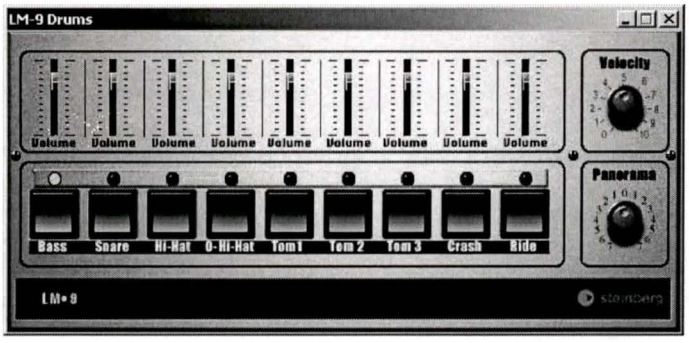


Figure 5.2 VSTi drum machine plugin by Steinberg.

The functionality of plugins goes beyond simple musical instruments such as the LM-9. Authoring applications now exist that allow users to build their own plugins from module libraries. Figure 5.3 shows a DXi plugin called *Tassman* [50], which is a user configurable modular synthesizer. Users of Tassman's authoring application can build custom synthesizers from the included module library and load these devices as plugins within any DXi host.

Figure 5.3 DXi Tassman modular synthesizer plugin by AAS.

Even more compelling to DSP engineers and electronic musicians is a new type of plugin authoring application called the *virtual device studio*. Applications of this class allow the user to build software devices of great complexity from an included graphical library of processing elements and then run these as plugins. One such application is *Infinity* [51], which includes a library of over 380 processing objects which can be interconnected in a variety of ways to create synthesizers, audio effects, sam-

plers or any device imaginable. These virtual devices may then be run as stand-alone applications or as a hosted plugin. Figure 5.4 shows an instrument that has been created from Infinity objects within the Infinity virtual device studio application.

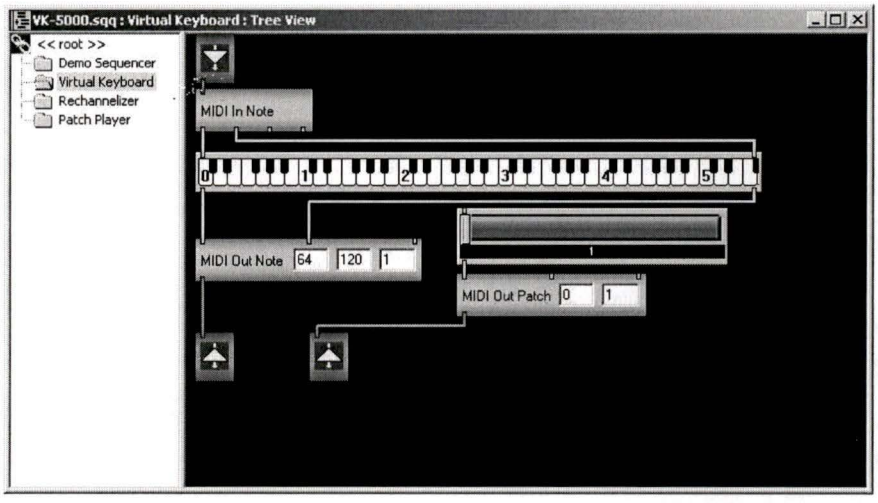


Figure 5.4 Infinity objects connected within a device.

New Infinity objects can be created by third party developers using Sound Quest's Infinity software developer kit (SDK). This SDK provides resources and program code that allow those proficient in C and C++ programming to write and compile their own objects for use within Infinity. The author of this dissertation plans to implement the Musical Spectrogram as an Infinity object. This proposed object will appear much like the illustration of Figure 5.5.

Figure 5.5 Proposed Musical Spectrogram Infinity object.

The rectangular port on the upper left corner of this object is the audio signal input, the port on the upper right is the threshold trigger parameter input, and the port on the lower left is the MIDI event output. This is only one possible object definition based on the Musical Spectrogram. 108 audio outputs could also be included in the

object to allow individual semitone signal processing to occur, for example. Implementation of the Musical Spectrogram as an Infinity object would permit extensive real-time testing of the algorithm, and would make the development and testing of larger systems based on this algorithm much easier as well.

With regards to the Musical Spectrogram itself, further work is planned towards the development of an analysis/synthesis version that would provide the basis for perceptual coding, bandwidth compression and musically intuitive audio effects processing. No trivial task, this development effort would require not only significant research into the design of filters that balance the perfect reconstruction criteria of the DWT filter bank implementation with the high selectivity requirements of the Musical Spectrogram application, it would also require serious study of the 12-SBF bank to determine if perfect reconstruction is even possible for such a filter bank.

References

[1] J. B. J. Fourier, ''Theorie Analytique de la Chaleur'', *Ouevres de Fourier*, tome premier, G.Darboux, Ed., Paris: Gauthiers-Villars, 1888.

[2] D. Ellis and D. Rosenthal, ''Mid-level representations for computational auditory scene analysis'', *International Joint Conference of Artificial Intelligence*, (Montreal, Quebec, 1995).

[3] M. Piszczalski et. al. ''Performed music: Analysis, synthesis and display by computer'', *J. Audio Eng Soc.*, vol. 29, 1/2, pp 38-45, 1981.

[4] J. Strawn, ''Analysis and synthesis of musical transitions using the discrete short-time Fourier transform'', *J. Audio Eng Soc.*, vol. 35, 1/2, pp 3-13, 1985.

[5] O. Izmirli, ''A hierarchichal constant Q transform for partial tracking in musical signals'', *Proceedings of 2nd COST G-6 Workshop on Digital Audio Effects*, (NTNU, Trondheim, 1999).

[6] S. H. Nawab, A. Ayyash and R. Wotiz, ''Identification of musical chords using constant-Q spectra'', *Proceedings of IEEE-ICASSP,* (Salt Lake City, Utah, 2001).

[7] X. Serra, ''Musical sound modeling with sinusoids plus noise'', *Musical Signal Processing*, Swets and Zeitlinger Publishers, 1997.

[8] T. Virtanen and A. Klapuri, ''Separation of harmonic sound sources using sinusiodal modeling'', *Proceedings of IEEE-ICASSP*, vol. 2, pp. 765-768, 2000.

[9] D. Ellis, ''Hierarchic models of hearing for sound spearation and reconstruction'', *Proceedings of IEEE Workshop on Apps. of Sig. Proc. to Acous. and Audio*, (Mohonk, October 1993).

[10] M. Slaney and R. Lyon ''A perceptual pitch detector'', *Proceedings of IEEE-ICASSP*, vol. 1, pp. 357-360, 1990.

[11] M. Goto, ''A robust predominant-F0 estimation method for real-time detection of melody and bass lines in CD recordings'', *Proceedings of IEEE-ICASSP*, vol. 2, pp. 765-768, 2000.

[12] D. Preis and V.C. Georgopoulos, ''Wigner distribution representation and analysis of audio signals: An illustrated tutorial review'', *J. Audio Eng Soc.*, vol. 47, no. 12, pp 1043-1053, 1999.

[13] A. Antoniou, *Digital Filters: Analysis, Design and Applications*, 2nd Edition, McGraw-Hill, NY, 1993.

[14] L. Jackson, *Digital Filters and Signal Processing*, Kluwer, Boston, 1986.

[15] J. Proakis and D. Manolakis, *Digital Signal Processing*, 3rd Edition, Prentice Hall, New Jersey, 1996.

[16] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*, 2nd Edition, Prentice Hall, New Jersey, 1998.

[17] E. Remes, *General Computational Methods for Tchebycheff Approximation*, Kiev, 1957 (Atomic Energy Comission Translation 4491, pp. 1-85).

[18] A. Antoniou, ''New improved method for the design of weighted-Chebyshev, nonrecursive digital filters'', *IEEE Trans. Circuits Syst.,* vol. CAS-30, pp.740-750, October 1983.

[19] D. Shpak and A. Antoniou, ''A generalized Remez method for the design of FIR digital filters'', *IEEE Trans. Circuits Syst.,* vol. CAS-37, pp.161-174, February 1990.

[20] C. Charalambous and A. Antoniou, ''Equalisation of recursive digital filters'', *IEEE Proc.,* vol. 127, pt. G, pp. 219-225, October 1980.

[21] D. Shpak et al., ''Filter Design Toolbox'',The MathWorks, Inc., 2002.

[22] P. Vaidyanathan, *Multirate Systems and Filter Banks*, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[23] R. Crochiere and L. Rabiner, *Multirate Digital Signal Processing*, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1983.

[24] H. Nyquist, ''Certain factors affecting telegraph speed'', *Bell Systems Technical Journal,* vol. 3, pp. 324, April 1924.

[25] J. O'Connor and E. Robinson, ''Jean Baptiste Joseph Fourier'', *http://www-groups.dcs.st-and.ac.uk/~history/Mathematicians/Fourier.html,* (Appendix II).

[26] J. Cooley and J. Tukey, ''An algorithm for the machine calculation of complex Fourier series'', *Mathematics of Computation*, vol. 19, pp. 297-301, 1965.

[27] J. Pierce, *The Science of Musical Sound,* pp. 17-19 and pp. 115-117, W.H. Freeman and Company, NY, 1992.

[28] D. Bartlett, ''The mathematics of tuning'', *http://www.hlalapansi.demon.co.uk/Acoustics/MusicMaths/MusicMaths.html,* (Appendix II).

[29] E. Terhardt, ''Virtual pitch'', *http://www.mmk.e-technik.tu-muenchen.de/persons/ter/top/virtualp.html,* (Appendix II).

[30] K. Gann, ''An introduction to historical tunings'', *http://home.earthlink.net/~kgann/histune.html,* (Appendix II).

[31] E. Terhardt, ''Dominant spectral region'', *http://www.mmk.e-technik.tu-muenchen.de/persons/ter/top/dominant.html,* (Appendix II).

[32] K. Martin, ''Automatic transcription of simple polyphonic music: Robust front end processing'', *M.I.T. Media Laboratory Perceptual Computing Section Technical Report no. 399*, 1996.

[33] M. Goodwin and M. Vetterli, ''Time-frequency signal models for music analysis, transformation and synthesis'', *Proceedings of IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, (Paris, France, 1996).

[34] A. Eronen and A. Klapuri, ''Musical instrument recognition using cepstral coeffiecients and temporal features'', *Proceedings of IEEE-ICASSP*, vol. 2, pp. 753-756, 2000.

[35] ''Cubase VST 5.1'', Music recording and editing software, Steinberg Soft- und Hardware GmbH, 2001.

[36] ''Sonar 1.3'', Music recording and editing software, Cakewalk Music Software, 2001.

[37] P. Peebles, Jr., *Probability, Random Variables, and Random Signal Principles*, pp. 168-173, 3rd Edition, McGraw-Hill, NY, 1993.

[38] D. Gabor, ''Theory of communication'', *J. of the IEE*, vol. 93, pp. 429-457, 1946.

[39] O. Rioul and M. Vetterli, ''Wavelets and signal processing'', *IEEE Signal Proc. Magazine*, no. 1053-5888, pp. 14-38, October 1991.

[40] S. Qian and D. Chen, ''Joint analysis'', *IEEE Signal Proc. Magazine*, pp. 52-67, March 1999.

[41] J. Allen and L. Rabiner, ''A unified approach to short-time Fourier analysis and synthesis'', *Proceedings. of the IEEE*, vol. 65, no. 11, pp. 1558-1564, 1977.

[42] A. Grossmann, R. Kronland-Martinet, and J. Morlet, ''Reading and understanding continuous wavelet transforms'', *Proc. Int. Conf. Marseille, France,* Dec. 14-18, 1989.

[43] I. Daubechies, ''Orthonormal bases of compactly supported wavelets'' *Commun. Pure Appl. Math.,* vol. 41, no.7: pp. 901-996, 1988.

[44] P. Burt and E. Adelson, ''The Laplacian pyramid as a compact image code'', *IEEE Trans. on Com.,* vol. 31, no.4: pp. 532-540, April 1983.

[45] R. Crochiere, S. Weber and J. Flanagan, ''Digital coding of speech in subbands'', *Bell Syst. Tech. J.,* vol. 55, no.7: pp. 1069-1085, October 1976.

[46] S. Mallat, ''A theory for multiresolution signal decomposition: The wavelet representation'', *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 11, no.7: pp. 674-693, July 1989.

[47] G. Liu and C. Wei, ''A new variable fractional sample delay filter with nonlinear interpolation'', *IEEE Trans. Circuits Syst.-II: Analog and Digital Signal Processing,* vol. 39, no. 2, pp. 123-126, February 1992.

[48] A. G. Constantinides, ''Spectral transformations for digital filters'', *Proceedings. of the IEE*, vol. 117, pp. 1585-1590, August 1970.

[49] D. Koya, *Aural Phase Distortion Detection*, M.Sc. thesis, Department of Music Engineering, University of Miami, May 2000.

[50] ''Tassman Software Synthesizer'', Modular synthesizer builder and plugin software, Applied Acoustics Systems, http://www.applied-acoustics.com/ tassman.htm, 2002.

[51] ''Infinity 2.04'', Virtual device studio and plugin software, Sound Quest Music Software, http://www.squest.com, 2002.

[52] E. Brandt and R. Dannenberg, ''Low-latency music software using off-the-shelf operating systems'', *Proceedings of the International Music Conference*, pp. 137-141, 1998.

[53] R. Aarts and R.T. Dekkers, ''A real-time speech-music discriminator'', *J. Audio Eng Soc.*, vol. 47, no. 9, pp 720-725, 1999.

[54] M. Bobrek and D. Koch, ''Music signal segmentation using tree-structured filter banks'', *J. Audio Eng Soc.*, vol. 46, no. 5, pp 412-427, 1998.

[55] A. P. Klapuri, ''Multipitch estimation and sound separation by the spectral smoothness principle'', *Proceedings of IEEE-ICASSP,* (Salt Lake City, Utah, 2001).

[56] M. Goto, ''A predominant-F0 estimation method for CD recordings: MAP estimation using EM algorithm for adaptive tone models'' *Proceedings of IEEE-ICASSP,* (Salt Lake City, Utah, 2001).

[57] A. P. Klapuri, T. Virtanen and J. Holm, ''Robust multipitch estimation for the analysis and manipulation of polyphonic music signals'', *Proceedings of COST-G6 Conference on Digital Audio Effects,* (Verona, Italy, 2000).

[58] W. Kuhn ''A real-time pitch recognition algorithm for music applications'', *Computer Music Journal,* vol. 14, no. 3, pp. 60-71, Fall 1990.

[59] K. Nayebi, T. Barnwell III and M. Smith, ''The design of perfect reconstruction nonuniform band filter banks'', *Proceedings of IEEE-ICASSP,* pp. 1781-1784, 1991.

[60] J. Kovacevic and Martin Vetterli, ''Perfect reconstruction filter banks with rational sampling rate changes'', *Proceedings of IEEE-ICASSP,* pp. 1785-1788, 1991.

Appendix I: MATLAB Programs

Appendix I contains lisitings of the MATLAB m-files that make up the Musical Spectrogram implementation covered in Chapter 4. Listings of the m-files that were used to prepare filter coefficients for implementation as fixed point DSP data are also included.

```
function [C,S]=BiquadPrep(A);
%
% BQ_PREP- Biquad Preparation Program
% reads a matlab coefficient matrix and H0 value and
% returns a coefficient buffer and Scales Buffer.
%                    by Tony Antoniou, 05 August, 1998
%
% [C,S]=bq_prep(A);
%
% Where A is a multistage filter coefficient matrix of the form
%    [a01 a11 a21 b01 b11 b21;
%     a02 a12 a22 b02 b12 b22;
%       ... ...          ;
%     a0n a1n a2n b0n b1n b2n ],
% and H0 is the filter's multiplier constant.
%
% C is a vector of the form
% [a01 a11 a21 b01 b11 ... a0n a1n a2n b0n b1n]
%
% and S is a vector of the form
% [s0, s1, s2, s3 ... s(n-1)]
%
% usage: [C,S]=bq_pr_2(A);
global stages width S w i Stage_i MaxGain C j;

[stages,width] = size(A);

S = ones(stages,1);
w=(pi/2):0.003:pi;

for i=1:stages
   Stage_i = A(1:i,:);
   MaxGain  = coe_zmax(Stage_i,w,0);
   S(i) = 1/MaxGain;
end

C = zeros(stages*5,1);
j =1;

for i=1:stages
   C(j) = A(i,1)*S(i);
   j=j+1;
   C(j) = A(i,2)*S(i);
   j=j+1;
   C(j) = A(i,3)*S(i);
   j=j+1;
   C(j) = -A(i,4);
   j=j+1;
   C(j) = -A(i,5)/2;
```

```
    j=j+1;
end
```

```
function [b,a]=dffmatlab(A,H0);
%
% DFFMATLAB- Digital Filter File Matlab converter
% reads an unformatted *.coe file from D-Filter and
% converts a matlab coefficient matrix into
% polynomial vectors for matlab's FREQZ routine.
% Standard filter form in matlab.
%                    by Tony Antoniou, 17 March, 2002
%
% [b,a]=dff2tf(A,H0);
%
% Where A is a multistage filter coefficient matrix of the form
%    [a01 a11 a21 b01 b11 b21;
%     a02 a12 a22 b02 b12 b22;
%       ... ...          ;
%     a0n a1n a2n b0n b1n b2n ],
% H0 is the filter's multiplier constant.
%
% b is the numerator polynomial and
% a is the denominator polynomial
%
% usage: [b, a]=dff2tf(A,H0);

[Bs,As]= sos2tf(A);

b = fliplr(Bs);
a = fliplr(As);

b = b * H0;

if a(1) == 0
   b = b(2:end);
   a = a(2:end);
end
```

```
function [A,H0]=dffread(filename);
%
% DFFREAD- Digital Filter File Read
% reads an unformatted *.coe file from D-Filter and
% returns a matlab coefficient matrix.
%                    by Tony Antoniou, 19 March, 1998
%             Revised by Tony Antoniou, 18 March, 2002
%
% [A, H0]=dffread('C_raw.coe');
%
% Where 'C_raw.coe' is an unformatted coefficient text file
% produced by the D-Filter program.
%
% A is a multistage filter coefficient matrix of the form
%    [a01 a11 a21 b01 b11 b21;
%     a02 a12 a22 b02 b12 b22;
%       ... ...          ;
%     a0n a1n a2n b0n b1n b2n ],
% and H0 is the filter's multiplier constant.
%
% usage: [A, H0]=dffread('C_raw.coe');

[fid, message]=fopen(filename,'r');

if fid== -1
   error(message);
end

status=fseek(fid,0,'bof');

twochars=fscanf(fid,'%s',2);

order=fscanf(fid,'%d',1);

H0=fscanf(fid,'%f',1);

A=zeros(order,6);

for i = 1:order
   numord=fscanf(fid,'%f',1);
   denord=fscanf(fid,'%f',1);

   if (numord == 1)
      temp=fscanf(fid,'%f', 4)';
      A(i,1:2)=temp(1,1:2);
      A(i,3)=0.0;
      A(i,4:5)=temp(1,3:4);
      A(i,6)=0.0;
   else
```

```
function a=FilterbankPrepare;
%
% FB_Pr_2- Prepares the 12 tone filterbank's Filter coefficients for
% use with the Biquad routine
%                          by Tony Antoniou, 12 August, 1998
%
% usage: fb_pr_2;

w=(pi/2):0.01:pi;
figure;

format long
echo on

diary FinalFilterBank.txt

%------ C FILTER DESIGN ------------------------------------------------
[A, H0]=dffread('Cn.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
Cn = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[Cn,CnS] =BiquadPrep(Cn)              % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_Cn = H0 / prod(CnS)                 % Calculate final scale factor

vec_zpl2(Cn,S_Cn,'b-',w);              % Plot the spectrum of the design
hold on;

%------ C# FILTER DESIGN -----------------------------------------------
[A, H0]=dffread('Cs.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
Cs = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[CsV,CsS] =BiquadPrep(Cs)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_Cs = H0 / prod(CsS)                 % Calculate final scale factor

vec_zpl2(CsV,S_Cs,'r-',w);               % Plot the spectrum of the design

%------ D FILTER DESIGN ------------------------------------------------
[A, H0]=dffread('Dn.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
Dn = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[DnV,DnS] =BiquadPrep(Dn)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_Dn = H0 / prod(DnS)                 % Calculate final scale factor

vec_zpl2(DnV,S_Dn,'b-',w);             % Plot the spectrum of the design
```

```
%------ D# FILTER DESIGN ------------------------------------------------
[A, H0]=dffread('Ds.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
Ds = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[DsV,DsS] =BiquadPrep(Ds)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_Ds = H0 / prod(DsS)                 % Calculate final scale factor

vec_zpl2(DsV,S_Ds,'r-',w);             % Plot the spectrum of the design

%------ E FILTER DESIGN ------------------------------------------------
[A, H0]=dffread('En.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
En = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[EnV,EnS] =BiquadPrep(En)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_En = H0 / prod(EnS)                 % Calculate final scale factor

vec_zpl2(EnV,S_En,'b-',w);             % Plot the spectrum of the design

%------ F FILTER DESIGN ------------------------------------------------
[A, H0]=dffread('Fn.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
Fn = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[FnV,FnS] =BiquadPrep(Fn)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_Fn = H0 / prod(FnS)                 % Calculate final scale factor

vec_zpl2(FnV,S_Fn,'b-',w);             % Plot the spectrum of the design

%------ F# FILTER DESIGN ------------------------------------------------
[A, H0]=dffread('Fs.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
Fs = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[FsV,FsS] =BiquadPrep(Fs)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_Fs = H0 / prod(FsS)                 % Calculate final scale factor

vec_zpl2(FsV,S_Fs,'r-',w);             % Plot the spectrum of the design

%------ G FILTER DESIGN ------------------------------------------------
[A, H0]=dffread('Gn.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
```

```
Gn = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[GnV,GnS] =BiquadPrep(Gn)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_Gn = H0 / prod(GnS)                 % Calculate final scale factor

vec_zpl2(GnV,S_Gn,'b-',w);             % Plot the spectrum of the design

%------ G# FILTER DESIGN ------------------------------------------------
[A, H0]=dffread('Gs.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
Gs = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[GsV,GsS] =BiquadPrep(Gs)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_Gs = H0 / prod(GsS)                 % Calculate final scale factor

vec_zpl2(GsV,S_Gs,'r-',w);             % Plot the spectrum of the design

%------ A FILTER DESIGN -------------------------------------------------
[A, H0]=dffread('An.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
An = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[AnV,AnS] =BiquadPrep(An)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_An = H0 / prod(AnS)                 % Calculate final scale factor

vec_zpl2(AnV,S_An,'b-',w);             % Plot the spectrum of the design

%------ A# FILTER DESIGN ------------------------------------------------
[A, H0]=dffread('As.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
As = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[AsV,AsS] =BiquadPrep(As)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_As = H0 / prod(AsS)                 % Calculate final scale factor

vec_zpl2(AsV,S_As,'r-',w);             % Plot the spectrum of the design

%------ B FILTER DESIGN -------------------------------------------------
[A, H0]=dffread('Bn.coe')   % Read in the *.coe file from D-Filter

[Poles,Zeros]= MultiStageZplot(A,0);  % Compute unpaired Poles & Zeros
Bn = SecondOrderRootPair(Poles,Zeros) % Pair up the Poles & Zeros
[BnV,BnS] =BiquadPrep(Bn)             % Convert to DSP vector & Compute
                                      % 2nd order stage scaling
S_Bn = H0 / prod(BnS)                 % Calculate final scale factor
```

```
function [bn,as,an,gs,gn,fs,fn,en,ds,dn,cs,cn]...
=MS_12SBF(octaveinput,SB,SA,CB,CA);
% MS_12SBF: A 12 tone DSB implementation employing filter banks.
%
%USAGE: [bn,as,an,gs,gn,fs,fn,en,ds,dn,cs,cn]=MS_DSB(octaveinput,SB,SA,CB,C↙
A);
%
%                                   Resurrected & updated February 12, 2002

[low_11,bn]=MS_SPM(octaveinput,SB,SA,CB,CA);
[low_10,as]=MS_SPM(low_11,SB,SA,CB,CA);
[low_9,an] =MS_SPM(low_10,SB,SA,CB,CA);
[low_8,gs] =MS_SPM(low_9,SB,SA,CB,CA);
[low_7,gn] =MS_SPM(low_8,SB,SA,CB,CA);
[low_6,fs] =MS_SPM(low_7,SB,SA,CB,CA);
[low_5,fn] =MS_SPM(low_6,SB,SA,CB,CA);
[low_4,en] =MS_SPM(low_5,SB,SA,CB,CA);
[low_3,ds] =MS_SPM(low_4,SB,SA,CB,CA);
[low_2,dn] =MS_SPM(low_3,SB,SA,CB,CA);
[cn,cs]    =MS_SPM(low_2,SB,SA,CB,CA);
```

```
function [oct9,oct8,oct7,oct6,oct5,oct4,oct3,oct2,oct1]...
=MS_9OBF(signalinput,h0_B,h0_A);
% MS_9OBF: A Nine octave DWT implementation employing a QMF filter bank.
%
%USAGE: [oct9,oct8,oct7,oct6,oct5,oct4,oct3,oct2,oct1]=MS_DWT(signalinput,h↙
0_B,h0_A);
%
%                                   Resurrected & updated February 12, 2002

[low_9,oct9]=MS_OPM(signalinput,h0_B,h0_A);
[low_8,oct8]=MS_OPM(low_9,h0_B,h0_A);
[low_7,oct7]=MS_OPM(low_8,h0_B,h0_A);
[low_6,oct6]=MS_OPM(low_7,h0_B,h0_A);
[low_5,oct5]=MS_OPM(low_6,h0_B,h0_A);
[low_4,oct4]=MS_OPM(low_5,h0_B,h0_A);
[low_3,oct3]=MS_OPM(low_4,h0_B,h0_A);
[low_2,oct2]=MS_OPM(low_3,h0_B,h0_A);
[oct0, oct1]=MS_OPM(low_2,h0_B,h0_A);
```

```
function [outlow,outhi] = MS_OPM(insignal,B,A)
% MS_OPM: The Octave Processing Module of the 9-Octave DWT
% structure.
%
%   insignal is the processed input signal
%   B is the lowpass halfband filter's Numerator Coeffients
%   A is the lowpass halfband filter's Denominator Coefficients
%
%   The low output is downsampled by a factor of 2.
%
%   USAGE: [out_low,out_hi]=MS_OPM(insignal,B,A);
%

hi =filter(QMF(fliplr(B),0),QMF(fliplr(A),0), insignal);   % Filter highpas↙
s signal
outhi =  hi;                                % Decimate highpass signal

low=filter(B,A, insignal);                  % Filter lowpass signal

outlow = low(1:2:end);              % Decimate lowpass signal

% length(outlow)
```

```
function [outlow,outhi] = MS_SPM(insignal,SB,SA,CB,CA)
% MS_SPM: The Semitone Processing Module of the 12-Tone DSB
% structure.
%
%   insignal is the processed input signal
%   SB is the semitone filter's Numerator Coeffients
%   SA is the semitone filter's Denominator Coefficients
%   CB is the complementary filter's Numerator Coeffients
%   CA is the complementary filter's Denominator Coefficients
%
%   The low output is downsampled by a factor of 2.
%
%   USAGE: [out_low,out_hi]=MS_SPM(insignal,SB,SA,CB,CA);
%

hi =filter(SB,SA, insignal);      % Filter highpass signal
outhi =  hi;                      % Output highpass signal

low=filter(CB,CA, insignal);      % Filter lowpass signal
outlow = SRDQ_Mod(low);           % Rational Downsample signal
```

```
function [Poles,Zeros]= MultiStageZplot(A,PlotFlag);

% ZPlot utility for 2nd order multistage coefficient matrices following
% Antoniou's filter conventions.    by Tony Antoniou, 8 September, 1998
%
% A is a multistage filter coefficient matrix of the form
%    [a01 a11 a21 b01 b11 b21;
%     a02 a12 a22 b02 b12 b22;
%       ... ...          ;
%     a0n a1n a2n b0n b1n b2n ], and
%
% PlotFlag is a boolean switch parameter: 0 means no plot
%                                         1 means plot
%
% This program computes and plots the poles and zeros of
% each stage of a Multistage 2nd order D-Filter coef matrix
%
% Usage: MultiStageZplot(A);

[Stages,Col] = size(A);

n = ceil(Stages/2);
Poles = zeros(Stages*2,1);
Zeros = Poles;
count =1;

for i=1:Stages

   tpoles = roots(fliplr(A(i,4:6)));
   tzeros = roots(fliplr(A(i,1:3)));

   if (PlotFlag == 1)
      subplot(2,n,i);
      zplane(tpoles,tzeros);
   end

   Poles(count) = tpoles(1);
   Zeros(count) = tzeros(1);
   count = count+1;
   Poles(count) = tpoles(2);
   Zeros(count) = tzeros(2);
   count = count+1;

   title(['Stage ', num2str(i)]);
end;
```

```
function [oct9,oct8,oct7,oct6,oct5,oct4,oct3,oct2,oct1]...
=qmf_9(wavefile,h0_B,h0_A,ttl);
% qmf_8: An Eight octave DWT implementation employing a QMF filter bank.
%
%USAGE: [oct9,oct8,oct7,oct6,oct5,oct4,oct3,oct2,oct1]=qmf_9('8_Climb.wav',↙
h0_b,h0_a,'8 Climb');
%
%                                        Resurrected & updated February 12, ↙
2002


Baa =  [ 0.17407181754116,
   1.60547559482943,
   7.19144336906369,
   20.53617982400905,
   41.47067736908829,
  62.23858294112011,
  71.08850540413357,
  62.23858294112013,
  41.47067736908829,
  20.53617982400905,
   7.19144336906369,
   1.60547559482943,
   0.17407181754116 ];

Aaa = [ 1.00000000000000,
   6.13533052564281,
  19.70639953817681,
  41.51677909875608,
  63.22155672941503,
  72.48499738412531,
  63.99373084676789,
  43.70085632395421,
  23.03377764791505,
   9.20576871418643,
   2.73655996568792,
   0.56628827415689,
   0.07512404931233 ];


[inputsignal,Fs,NBITS]=WAVREAD(wavefile);

if Fs ~= 22050
   fprintf( 'NO NO NO - WRONG SAMPLING RATE\n');
end

sound(inputsignal,Fs);

aainput = filter(Baa,Aaa,inputsignal);  % AntiAlias it!
```

```
input = SRI(0.73774920634921,aainput);  % Resample to 16267.37.

figure
Fs = 16267.37;

SoundLength = length(input)/Fs;

tvec=0:1/Fs:((length(input)-1)/Fs);
subplot(10,1,1)
plot(tvec,input);
axis([0 SoundLength -1 1]);
title(ttl);
ylabel('Signal');
set(gca, 'XTickLabel',' ');


[low_9,oct9]=MS_OPM(input,h0_B,h0_A);
   tvec=0:1/Fs:((length(oct9)-1)/Fs);
   subplot(10,1,2)
   plot(tvec,oct9);
   axis([0 SoundLength -1 1]);
   ylabel('Oct 9');
   set(gca, 'XTickLabel',' ');

Fs8=Fs/2;
[low_8,oct8]=MS_OPM(low_9,h0_B,h0_A);
   tvec=0:1/Fs8:((length(oct8)-1)/Fs8);
   subplot(10,1,3)
   plot(tvec,oct8);
   axis([0 SoundLength -1 1]);
   ylabel('Oct 8');
   set(gca, 'XTickLabel',' ');

Fs7=Fs8/2;
[low_7,oct7]=MS_OPM(low_8,h0_B,h0_A);
   tvec=0:1/Fs7:((length(oct7)-1)/Fs7);
   subplot(10,1,4)
   plot(tvec,oct7);
   axis([0 SoundLength -1 1]);
   ylabel('Oct 7');
   set(gca, 'XTickLabel',' ');

Fs6=Fs7/2;
[low_6,oct6]=MS_OPM(low_7,h0_B,h0_A);
   tvec=0:1/Fs6:((length(oct6)-1)/Fs6);
   subplot(10,1,5)
   plot(tvec,oct6);
   axis([0 SoundLength -1 1]);
```

```
   ylabel('Oct 6');
   set(gca, 'XTickLabel',' ');

Fs5=Fs6/2;
[low_5,oct5]=MS_OPM(low_6,h0_B,h0_A);
tvec=0:1/Fs5:((length(oct5)-1)/Fs5);
   subplot(10,1,6)
   plot(tvec,oct5);
   axis([0 SoundLength -1 1]);
   ylabel('Oct 5');
   set(gca, 'XTickLabel',' ');

Fs4=Fs5/2;
[low_4,oct4]=MS_OPM(low_5,h0_B,h0_A);
tvec=0:1/Fs4:((length(oct4)-1)/Fs4);
   subplot(10,1,7)
   plot(tvec,oct4);
   axis([0 SoundLength -1 1]);
   ylabel('Oct 4');
   set(gca, 'XTickLabel',' ');

Fs3=Fs4/2;
[low_3,oct3]=MS_OPM(low_4,h0_B,h0_A);
   tvec=0:1/Fs3:((length(oct3)-1)/Fs3);
   subplot(10,1,8)
   plot(tvec,oct3);
   axis([0 SoundLength -1 1]);
   ylabel('Oct 3');
   set(gca, 'XTickLabel',' ');


Fs2=Fs3/2;
[low_2,oct2]=MS_OPM(low_3,h0_B,h0_A);
tvec=0:1/Fs2:((length(oct2)-1)/Fs2);
   subplot(10,1,9)
   plot(tvec,oct2);
   axis([0 SoundLength -1 1]);
   ylabel('Oct 2');
   set(gca, 'XTickLabel',' ');


Fs1=Fs2/2;
[oct0, oct1]=MS_OPM(low_2,h0_B,h0_A);
   tvec=0:1/Fs1:((length(oct1)-1)/Fs1);
   subplot(10,1,10)
   plot(tvec,oct1);
   axis([0 SoundLength -1 1]);
   ylabel('Oct 1');
```

```
% Fs0=Fs1/2;
% tvec=0:1/Fs0:((length(oct0)-1)/Fs0);
% subplot(11,1,11)
% plot(tvec,oct0);
% axis([0 SoundLength -1 1]);
% ylabel('Oct 0');
```

```
function reg_chek(g,n);

% FIR Filter Regularity Checker Function: by Tony Antoniou, 14 January, 199↙
6
%
% usage: reg_chek(g,n)
%    g:  vector describing the impulse response of FIR filter [ A0 ... AN ]
%    n:  The depth of QMF tree to plot to
%
% This function was implemented to demonstrate the convergence of regular f↙
ilters
% by computing the continuous-time impulse response function fi(x) for each↙
 value
% of i. Effectively, we compute and plot the lowest octave band filter's im↙
pulse
% response for QMF trees of depth 1,2,3...n, and observe whether the functi↙
on is
% converging to a smooth, self-similar function at arbitrarily high tree de↙
pths.

gi = [0 g 0];
fi = gi*sqrt(2);
rows=ceil(n/4);                          % Compute the number of rows in the su↙
bplot
   step_width = 1/2;
   x= 0 : step_width : (step_width*size(gi,2));    % Define step vector
   subplot(rows,4,1);                              % Activate sub plot windo↙
w
   stairs(x,[fi 0]);                               % perform Stairplot
   title(['f0(x): Step=0.5']);                     % Title first subplot
   set(gca,'Box','off','XLim',[0 size(g,2)]);      % Make the axes pretty!


for depth = 2:n                          % For each depth of tree...

   % Here's a trick: By upsampling gi(n) by two, we double the order of z i↙
n Gi(z)
   % i.e. G(z)G(z^2)...-> G(z^2)G(z^4)... Then we need only to convolve it ↙
with g(n)

   gtemp=zeros(1,2*size(gi,2));      %  First we build zero vector of 2x le↙
ngth
   baron=1;
   for count=1:size(gi,2);           %  Now we loop through entries of gi(n↙
)
      gtemp(baron) = gi(count);      %  copying to every other entry in gte↙
mp(n)
      baron = baron + 2;
   end;                              %  End of upsampling loop
```

```
   gi=conv(g,gtemp);                    % Now all that remains is to convolve ↙
g(n)
   fi=gi*2^(depth/2);                   % Scale the value of fi(x)=gi(x)*2^(i/↙
2)

   step_width = 1/2^(depth);
   x= 0 : step_width : (step_width*size(gi,2));       % Define step vector
   subplot(rows,4,depth);                             % Activate sub plot w↙
indow
   stairs(x,[ fi 0]);                                 % perform Stairplot o↙
f fi(x)
   title(['f',num2str(depth-1),'(x): Step=',num2str(step_width)]);  % Title↙
 subplot
   set(gca,'Box','off','XLim',[0 size(g,2)]);         % Make the axes prett↙
y!

end                                % End of this depth's impulse response pl↙
ot
```

```
function A= SecondOrderRootPair(P,Z);

% 2nd order multistage coefficient matrix generator following
% Antoniou's filter conventions.    by Tony Antoniou, 8 September, 1998
%
% This program takes the poles and zeros of any even order digital
% filter and optimally groups them into second order stages in such
% a way that each stage approaches unity gain.
%
% P is a Column Vector of all filter poles
% Z is a Column Vector of all filter zeros
%
% Returns A, a multistage filter coefficient matrix of the form
%    [a01 a11 a21 b01 b11 b21;
%     a02 a12 a22 b02 b12 b22;
%       ... ...          ;
%     a0n a1n a2n b0n b1n b2n ],
%
%
% Usage: A= SecondOrderRootPair(P,Z);

Rows = length(P);

Poles = reshape(P,2,Rows/2);
Zeros = reshape(Z,2,Rows/2);

PoleCol = Poles(2,:)';
ZeroCol = Zeros(2,:)';
PoleCol = flipud(sort(PoleCol));

PairList = zeros(length(PoleCol),1);
ZeroList = ZeroCol;

for i = 1:length(PoleCol)
   anglediffs = abs(angle(ZeroList) - angle(PoleCol(i)));
   best = min(anglediffs);
   PairList(i) = find(abs(angle(ZeroCol) - angle(PoleCol(i)))==best);
   temp = zeros((length(ZeroList)-1),1);
   count = 1;
   for j = 1 : length(ZeroList)
      if (ZeroList(j) ~= ZeroCol(PairList(i)))
         temp(count) = ZeroList(j);
         count = count+1;
      end
   end
   ZeroList = temp;
end

% Now to reassemble the poles and zeroes into 2nd order stages
```

```
% in the sequence defined by the ordering of PoleCol and the
% PairList's index data into ZeroCol

A = zeros(Rows/2,6);

for i = 1:Rows/2;
   % Reconstitute the ith Stage's Zeros into Numerator Coefficients
   RootNi= [ZeroCol(PairList(i)), conj(ZeroCol(PairList(i)))]';
   Numi = poly(RootNi);

   % Reconstitute the ith Stage's Poles into Denominator Coefficients
   RootPi= [PoleCol(i), conj(PoleCol(i))]';
   Deni = poly(RootPi);

   % Copy these values into the ith row of A: a0i a1i a2i b0i b1i b2i;
   A(i,1) = Numi(3); A(i,2) = Numi(2); A(i,3) = Numi(1);
   A(i,4) = Deni(3); A(i,5) = Deni(2); A(i,6) = Deni(1);
end
```

```
Fs9=Fs10/(2^(1/12));
[low_8,gs]=MS_SPM(low_9,SB,SA,CB,CA);
   tvec=0:1/Fs9:((length(gs)-1)/Fs9);
   subplot(13,1,5)
   plot(tvec,gs);
   axis([0 SoundLength -1 1]);
   ylabel('G#');
   set(gca, 'XTickLabel',' ');

Fs8=Fs9/(2^(1/12));
[low_7,gn]=MS_SPM(low_8,SB,SA,CB,CA);
tvec=0:1/Fs8:((length(gn)-1)/Fs8);
   subplot(13,1,6)
   plot(tvec,gn);
   axis([0 SoundLength -1 1]);
   ylabel('G');
   set(gca, 'XTickLabel',' ');

Fs7=Fs8/(2^(1/12));
[low_6,fs]=MS_SPM(low_7,SB,SA,CB,CA);
tvec=0:1/Fs7:((length(fs)-1)/Fs7);
   subplot(13,1,7)
   plot(tvec,fs);
   axis([0 SoundLength -1 1]);
   ylabel('F#');
   set(gca, 'XTickLabel',' ');

Fs6=Fs7/(2^(1/12));
[low_5,fn]=MS_SPM(low_6,SB,SA,CB,CA);
   tvec=0:1/Fs6:((length(fn)-1)/Fs6);
   subplot(13,1,8)
   plot(tvec,fn);
   axis([0 SoundLength -1 1]);
   ylabel('F');
   set(gca, 'XTickLabel',' ');


Fs5=Fs6/(2^(1/12));
[low_4,en]=MS_SPM(low_5,SB,SA,CB,CA);
tvec=0:1/Fs5:((length(en)-1)/Fs5);
   subplot(13,1,9)
   plot(tvec,en);
   axis([0 SoundLength -1 1]);
   ylabel('E');
   set(gca, 'XTickLabel',' ');


Fs4=Fs5/(2^(1/12));
```

```
[low_3,ds]=MS_SPM(low_4,SB,SA,CB,CA);
   tvec=0:1/Fs4:((length(ds)-1)/Fs4);
   subplot(13,1,10)
   plot(tvec,ds);
   axis([0 SoundLength -1 1]);
   ylabel('D#');
   set(gca, 'XTickLabel',' ');



Fs3=Fs4/(2^(1/12));
[low_2,dn]=MS_SPM(low_3,SB,SA,CB,CA);
   tvec=0:1/Fs3:((length(dn)-1)/Fs3);
   subplot(13,1,11)
   plot(tvec,dn);
   axis([0 SoundLength -1 1]);
   ylabel('D');
   set(gca, 'XTickLabel',' ');



Fs2=Fs3/(2^(1/12));
[cn,cs]=MS_SPM(low_2,SB,SA,CB,CA);
   tvec=0:1/Fs2:((length(cs)-1)/Fs2);
   subplot(13,1,12)
   plot(tvec,cs);
   axis([0 SoundLength -1 1]);
   ylabel('C#');
   set(gca, 'XTickLabel',' ');


   Fs1=Fs2/(2^(1/12));
   tvec=0:1/Fs1:((length(cn)-1)/Fs1);
   subplot(13,1,13)
   plot(tvec,cn);
   axis([0 SoundLength -1 1]);
   ylabel('C');


%  Fs0=Fs1/2;
%  tvec=0:1/Fs0:((length(oct0)-1)/Fs0);
%  subplot(11,1,11)
%  plot(tvec,oct0);
%  axis([0 SoundLength -1 1]);
%  ylabel('Oct 0');
```

```
subplot(13,1,5)
plot(tvec,gs);
axis([0 SoundLength -1 1]);
ylabel('G#');
set(gca, 'XTickLabel',' ');

gn =filter(Bgn,Agn, input);
subplot(13,1,6)
plot(tvec,gn);
axis([0 SoundLength -1 1]);
ylabel('G');
set(gca, 'XTickLabel',' ');

fs =filter(Bfs,Afs, input);
subplot(13,1,7)
plot(tvec,fs);
axis([0 SoundLength -1 1]);
ylabel('F#');
set(gca, 'XTickLabel',' ');

fn =filter(Bfn,Afn, input);
subplot(13,1,8)
plot(tvec,fn);
axis([0 SoundLength -1 1]);
ylabel('F');
set(gca, 'XTickLabel',' ');

en =filter(Ben,Aen, input);
subplot(13,1,9)
plot(tvec,en);
axis([0 SoundLength -1 1]);
ylabel('E');
set(gca, 'XTickLabel',' ');

ds =filter(Bds,Ads, input);
subplot(13,1,10)
plot(tvec,ds);
axis([0 SoundLength -1 1]);
ylabel('D#');
set(gca, 'XTickLabel',' ');


dn =filter(Bdn,Adn, input);
subplot(13,1,11)
plot(tvec,dn);
axis([0 SoundLength -1 1]);
ylabel('D');
set(gca, 'XTickLabel',' ');
```

```
function [bn,as,an,gs,gn,fs,fn,en,ds,dn,cs,cn]=semi_BP_12(input,iFs,ttl);

% qmf_8: An Eight octave DWT implementation employing a QMF filter bank.
%
%USAGE: [bn,as,an,gs,gn,fs,fn,en,ds,dn,cs,cn]=semi_BP_12(Oct5,1016.71,'Oct5↙
 Chromatic');
%
%                                        Resurrected & updated February 12, ↙
2002

figure


load BandFilters;



SoundLength = length(input)/iFs;

tvec=0:1/iFs:((length(input)-1)/iFs);
subplot(13,1,1)
plot(tvec,input);
axis([0 SoundLength -1 1]);
title(ttl);
ylabel('Signal');
set(gca, 'XTickLabel',' ');

   bn =filter(Bbn,Abn, input);
   subplot(13,1,2)
   plot(tvec,bn);
   axis([0 SoundLength -1 1]);
   ylabel('B');
   set(gca, 'XTickLabel',' ');

   as =filter(Bas,Aas, input);
   subplot(13,1,3)
   plot(tvec,as);
   axis([0 SoundLength -1 1]);
   ylabel('A#');
   set(gca, 'XTickLabel',' ');

   an =filter(Ban,Aan, input);
   subplot(13,1,4)
   plot(tvec,an);
   axis([0 SoundLength -1 1]);
   ylabel('A');
   set(gca, 'XTickLabel',' ');

   gs =filter(Bgs,Ags, input);
```

```
cs =filter(Bcs,Acs, input);
subplot(13,1,12)
plot(tvec,cs);
axis([0 SoundLength -1 1]);
ylabel('C#');
set(gca, 'XTickLabel',' ');


cn =filter(Bcn,Acn, input);
subplot(13,1,13)
plot(tvec,cn);
axis([0 SoundLength -1 1]);
ylabel('C');


%  Fs0=Fs1/2;
%  tvec=0:1/Fs0:((length(oct0)-1)/Fs0);
%  subplot(11,1,11)
%  plot(tvec,oct0);
%  axis([0 SoundLength -1 1]);
%  ylabel('Oct 0');
```

```
function RSV= SRD(InputVector);

% Stirling Rational Downsampler, based on Quadratic Interpolator
% takes InputVector and returns resampled vector RSV
%                                by Tony Antoniou, 23 March, 2002
%
% Usage: [A,B,C,RSV]= SRD(InputVector);

InputVector = InputVector';
AlphaStepSize = 1/(2^(1/12));

LengthOfInputVector = length(InputVector);
LengthOfRSV = floor(LengthOfInputVector*(1/AlphaStepSize));

RSV = zeros(1, LengthOfRSV);

% Initialize AlphaSum_S
MaxAlpha     = AlphaStepSize*(LengthOfRSV - 1);
AlphaSum_S   = 0 : AlphaStepSize : MaxAlpha;
Index_B      = floor(AlphaSum_S);
Alpha_A      = AlphaSum_S - Index_B;
Incrementor  = diff(Index_B);

SampleIndex = 3;
OutputIndex = 3;
AlphaIndex =  1;

while (SampleIndex <= (length(InputVector) - 2))
        alpha = Alpha_A(AlphaIndex);

        % compute Coefficients now
        B1  = (alpha-1)*(alpha-2);
        B2  = (alpha+1)*(alpha+2);
        Cn2 = 1/24*(alpha+1)*alpha*B1;
        Cn1 = -1/6*(alpha+2)*alpha*B1;
        C0  = 1/4*B2*B1;
        Cp1 = -1/6*B2*alpha*(alpha-2);
        Cp2 = 1/24*B2*alpha*(alpha-1);

        Samples = InputVector((SampleIndex - 2) : (SampleIndex + 2));
        RSV(OutputIndex) = sum( [Cn2 Cn1 C0 Cp1 Cp2] .* Samples );
        OutputIndex = OutputIndex+1;
        SampleIndex = SampleIndex + Incrementor(AlphaIndex);
        AlphaIndex = AlphaIndex + 1;
end

RSV=RSV';
```

```
function RSV = SRD_Mod(InputVector);

% Stirling Rational Downsampler, based on Quadratic Interpolator
% takes InputVector and returns resampled vector RSV
% With additonal lookup mod.
%                                by Tony Antoniou, 23 March, 2002
%
% Usage: RSV = SRD_Mod(InputVector);

[rw,cl] = size(InputVector);
if cl == 1
   InputVector = InputVector';
end

AlphaStepSize = 2^(1/12);

LengthOfInputVector = length(InputVector);
LengthOfRSV = floor(LengthOfInputVector/AlphaStepSize);

RSV = zeros(1, LengthOfRSV);

% Initialize AlphaSum_S
AlphaSum_S   = 0 : AlphaStepSize : AlphaStepSize*1563;
Index_B      = floor(AlphaSum_S);
Alpha_A      = AlphaSum_S - Index_B;
Incrementor  = diff(Index_B);

SampleIndex = 3;
OutputIndex = 3;
AlphaIndex =  1;

%Build C Matrix

C= zeros(5,1564);
C(1,:) = 1/24*(Alpha_A+1).*Alpha_A.*(Alpha_A-1).*(Alpha_A-2);
C(2,:) = -1/6*(Alpha_A+2).*Alpha_A.*(Alpha_A-1).*(Alpha_A-2);
C(3,:) = 1/4*(Alpha_A+1).*(Alpha_A+2).*(Alpha_A-1).*(Alpha_A-2);
C(4,:) = -1/6*(Alpha_A+1).*(Alpha_A+2).*Alpha_A.*(Alpha_A-2);
C(5,:) = 1/24*(Alpha_A+1).*(Alpha_A+2).*Alpha_A.*(Alpha_A-1);

while (SampleIndex <= (length(InputVector) - 3))
   Samples = InputVector((SampleIndex - 2) : (SampleIndex + 2));
   RSV(OutputIndex) = Samples * C(:,AlphaIndex);
   OutputIndex = OutputIndex+1;
   SampleIndex = SampleIndex + Incrementor(AlphaIndex);
   if (AlphaIndex == 1563)
      AlphaIndex = 1;
   else
       AlphaIndex = AlphaIndex + 1;
```

```
    end

end

if c1 == 1
    RSV = RSV';
end



%size(RSV);
%A = Alpha_A;
%B = Incrementor;
%S = AlphaSum_S;
```

```
function RSV= SRI(ratio,InputVector);

% Stirling Rational Downsampler, based on Quadratic Interpolator
% takes InputVector and returns resampled vector RSV
%                                by Tony Antoniou, 23 March, 2002
%
% Usage: RSV= SRI(ratio,InputVector);

[rw,cl] = size(InputVector);
if cl == 1
   InputVector = InputVector';
end

AlphaStepSize = 1/ratio;

LengthOfInputVector = length(InputVector);
LengthOfRSV = floor(LengthOfInputVector*(1/AlphaStepSize));

RSV = zeros(1, LengthOfRSV);

% Initialize AlphaSum_S
MaxAlpha     = AlphaStepSize*(LengthOfRSV - 1);
AlphaSum_S   = 0 : AlphaStepSize : MaxAlpha;
Index_B      = floor(AlphaSum_S);
Alpha_A      = AlphaSum_S - Index_B;
Incrementor  = diff(Index_B);

SampleIndex = 3;
OutputIndex = 3;
AlphaIndex =  1;

while (SampleIndex <= (length(InputVector) - 1))
        alpha = Alpha_A(AlphaIndex);

        % compute Coefficients now
        Cn1 = 1/2*alpha*(alpha-1);
        C0  = -(alpha-1)*(alpha+1);
        Cp1 = 1/2*alpha*(alpha+1);

        Samples = InputVector((SampleIndex - 1) : (SampleIndex + 1));
        RSV(OutputIndex) = sum( [Cn1 C0 Cp1] .* Samples );
        OutputIndex = OutputIndex+1;
        SampleIndex = SampleIndex + Incrementor(AlphaIndex);
        AlphaIndex = AlphaIndex + 1;
end

if cl == 1
    RSV = RSV';
end
```

```
function [bn,as,an,gs,gn,fs,fn,en,ds,dn,cs,cn]=semi_12(input,iFs,SB,SA,CB,C↙
A,ttl);

% qmf_8: An Eight octave DWT implementation employing a QMF filter bank.
%
%USAGE: [bn,as,an,gs,gn,fs,fn,en,ds,dn,cs,cn]=semi_12(oct5,2033,SB,SA,CB,CA↙
, 'Chrom.');
%
%                                          Resurrected & updated February 12, ↙
2002

figure

SoundLength = length(input)/iFs;

tvec=0:1/iFs:((length(input)-1)/iFs);
subplot(13,1,1)
plot(tvec,input);
axis([0 SoundLength -1 1]);
title(ttl);
ylabel('Signal');
set(gca, 'XTickLabel',' ');

[low_11,bn]=MS_SPM(input,SB,SA,CB,CA);
%%   bn =filter(CB,CA, input);
   tvec=0:1/iFs:((length(input)-1)/iFs);
   subplot(13,1,2)
   plot(tvec,bn);
   axis([0 SoundLength -1 1]);
   ylabel('B');
   set(gca, 'XTickLabel',' ');

Fs11=iFs/(2^(1/12));
[low_10,as]=MS_SPM(low_11,SB,SA,CB,CA);
   tvec=0:1/Fs11:((length(as)-1)/Fs11);
   subplot(13,1,3)
   plot(tvec,as);
   axis([0 SoundLength -1 1]);
   ylabel('A#');
   set(gca, 'XTickLabel',' ');

Fs10=Fs11/(2^(1/12));
[low_9,an]=MS_SPM(low_10,SB,SA,CB,CA);
   tvec=0:1/Fs10:((length(an)-1)/Fs10);
   subplot(13,1,4)
   plot(tvec,an);
   axis([0 SoundLength -1 1]);
   ylabel('A');
   set(gca, 'XTickLabel',' ');
```

Appendix II: Web Page Citations

Appendix II contains hard-copy of the web pages cited by this dissertation. Since the world-wide-web does not have the permanence of official publication, these pages are included here to ensure their availability.

An Introduction to Historical Tunings

By Kyle Gann

1. Tuning in Pre-20th Century Europe

Those who attack equal temperament, the tuning of our modern pianos - as I do on my Just Intonation Explained page - seem to be attacking the great European musical tradition itself. After all, the music of Bach, Mozart, Beethoven, et al, was written for 12 equally-spaced pitches to the octave, right? And if we change our tuning, that music would no longer be playable as it was intended to be heard, right?

Dead wrong.

Equal temperament - the bland, equal spacing of the 12 pitches of the octave - is pretty much a 20th-century phenomenon. It was known about in Europe as early as the early 17th century, and in China much earlier. But it wasn't used, because the consensus was that it sounded awful: out of tune and characterless. During the 19th century (for reasons we'll discuss later), keyboard tuning drifted closer and closer to equal temperament over the protest of many of the more sensitive musicians. Not until 1917 was a method devised for tuning exact equal temperament.

So how was earlier European music tuned? What are we missing when we hear older music played in 20th-century equal temperament?

(Return to top menu)

2. Meantone Tuning

Let's start with Europe's most successful tuning, if endurance can be equated with success. Meantone tuning appeared sometime around the late 15th century, and was used widely through the early 18th century. In fact, it survived in pockets of resistance, especially in the tuning of English organs, all the way through the 19th century. No other tuning has survived in the west for 400 years. Let's see what meantone offered.

Every elegant tuning has a generating principle. The generating principle behind meantone was that it was more important to preserve the consonance of the major thirds (C to E, F to A, G to B) than it was to preserve the purity of the perfect fifths (C to G, F to C, G to D). There are acoustical reasons for this, namely - though I wouldn't want to go into the math involved - that the notes in a slightly out-of-tune third, being closer together than those in a fifth, create faster and more disturbing beats than those in a slightly out-of-tune fifth. (I can confirm this from experience with my own Steinway grand, which I keep tuned to an 18th-century tuning.) The aesthetic motivation for meantone was that composers had fallen in love with the sweetness of the major third, and were trying to get away from the medieval austerity of open perfect fifths.

In a purely consonant major third, the two strings vibrate at a frequency ratio of 5 to 4. For example, if

A
vibrates at
440 cycles per second,

then

C#
vibrates at
550 cycles per second.

Or if G vibrates at 100 cycles per second, then B vibrates at 125, and so on. (If you'd like this explained in more detail, visit my Just Intonation Explained page.) The size of a pure 5:4 major third is 386.3 cents, a cent being one 1200th of an octave, or one 100th of a half-step. Since an octave is 1200 cents, by definition, it is easy to see that three pure major thirds (3 x 386.3 cents = 1158.9) do not equal an octave. That's the whole problem of keyboard tuning, where you're limited to 12 steps per octave. Where do you put the gaps in your chains of perfect major thirds?

A pure perfect fifth is a 3 to 2 frequency ratio; if

A

vibrates at
440 cycles per second,

then

E
vibrates at
660 cycles per second.

A pure perfect fifth should be 702 cents wide, which is just about 7/12 of an octave; our current equal-tempered tuning accomodates perfect fifths (at 700 cents) within 2 cents, which is closer than most people can distinguish, but the thirds (at 400 cents) are way off, and form audible

beats that are ugly once you're sensitized to hear them.

Let's look at the meantone solution. There was no one invariable meantone tuning; before the 20th century, tuning was an art, not a science, and each tuner had his own method of tuning according to his own taste. The following is a chart of a meantone tuning defined in 1523 by Pietro Aaron.

Pitch:	C	C#	D	Eb	E	F	F#	G	G#	A	A#	B	C
Cents:	0	76.0	193.2	310.3	386.3	503.4	579.5	696.8	772.6	889.7	1006.8	1082.9	1200

(I adapt this chart, and ones following below, from an invaluable book, the bible of historical keyboard tuning: Owen Jorgensen's Tuning: Containing the Perfection of Eighteenth-Century Temperament, the Lost Art of Nineteenth-Century Temperament, and the Science of Equal Temperament, Michigan State University Press, 1991.) Now let's look at the sizes of the major thirds and perfect fifths on each pitch:

Major third	**Cents**	**Perfect Fifth**	**Cents**
C - E	386.3	C - G	696.8
Db - F	427.4	Db - Ab	696.6
D - F#	386.3	D - A	696.5
Eb - G	386.5	Eb - Bb	696.5
E - G#	386.3	E - B	696.6
F - A	386.3	F - C	696.6
F# - A#	427.3	F# - C#	696.5
G - B	386.1	G - D	696.4
Ab - C	427.4	Ab - Eb	737.7
A - C#	386.3	A - E	696.6
Bb - D	386.4	Bb - F	696.6
B - D#	427.4	B - F#	696.6

A major third and perfect fifth on the same pitch, of course, make up a major triad, the most common chord in European music from 1500 to 1900 - the meantone era. Let's look at what kind of major triads we have in meantone tuning.

The major thirds that are about 386 cents wide will be sweet, consonant, attractive. Eight pitches have virtually perfect major thirds on them - all except Db, F#, Ab, and B, whose major thirds are all about 427 cents. A third of 427 cents sounds like this: WAWAWAWAWAWAWAWA...!!! and is unusable for normal musical purposes. (Trust me on this.) All of the fifths are about 696 cents except for one, that on Ab, which is 737 cents and sounds terrible. The fifths would sound better at 702 cents, but at 696 or 697 you don't really notice the difference, especially if the chord is filled in with that perfect major third to smooth over the discrepancy. This is where the practice originated in European music of never having an open fifth sounding by itself without a third filling it in: the spare perfect fifth isn't quite consonant, and that fact becomes obvious if the third isn't there.

So meantone tuning gives us eight usable major triads: on C, D, Eb, E, F, G, A, and Bb. If you're writing a piece in meantone, those are the major triads you have available. Look through some 16th-century keyboard music: how many F#-major and Ab-major triads do you see? Probably none, and if you do see some, it means the composer was counting on a meantone tuning centered around some pitch other than C. If you want to use I, IV, and V chords in your piece, you can write in the keys of C, D, F, G, A, or Bb major. If you're writing in A major, you can't go to the V/V chord (B major), because it sounds awful. Renaissance and early Baroque music tends to be in a few keys grouped (in the circle of fifths) around C, usually C, F, G, D, Bb, or A. Ever wonder why Palestrina and Orlando Gibbons and Heinrich Schutz didn't get around to composing in F# major or Ab major? They couldn't, it sounded terrible in their tuning. (There were a few purely vocal early works that went through triads in diverse keys, such as Josquin's motet Absalon fili mi and Di Lasso's Prophetiae sybyllarum, the tuning and even notation of which have been subjects of much 20th-century controversy.)

Before we leave the subject of meantone, lets look at the available minor triads:

Minor third	Cents	Perfect Fifth	Cents
C - Eb	310.3	C - G	696.8
C# - E	310.3	C# - G#	696.6
D - F	310.2	D - A	696.5
Eb - Gb	269.2	Eb - Bb	696.5
E - G	310.5	E - B	696.6
F - Ab	269.2	F - C	696.6
F# - A	310.2	F# - C#	696.5
G - Bb	310.0	G - D	696.4
G# - B	310.3	G# - D#	737.7
A - C	310.3	A - E	696.6
Bb - Db	269.2	Bb - F	696.6
B - D	310.3	B - F#	696.6

A pure minor third is supposed to have a frequency ratio of 6:5. For example, if C# vibrates at 550 cycles per second, E should vibrate at 660. A 6:5 ratio interval is 315.64 cents wide. None of the minor thirds in this meantone are quite that wide, but most of them are 310 cents, which is, pardon the expression, close enough for jazz. (Actually, a narrow 7/6 minor third, often used by La Monte Young, is 266.8 cents, invitingly close to that 269; but 7/6 is an interval that was never recognized by European theory, though used in jazz and Arabic music among others.) Therefore the minor triads on C, C#, D, E, F#, G, A, and B are acceptable. (Not the one on G#, despite its OK minor third, because it has that wildly beating fifth.) If you think about it, these triads define the relative minor of the major keys implied by the major triads above:

Major:	C	D	Eb	E	F	G	A	Bb
Minor:	A	B	C	C#	D	E	F#	G

These 16 triads, 8 major and 8 minor, constitute the harmonic vocabulary of Renaissance and early Baroque music. Don't believe me? Look through a 16th- or 17th-century keyboard collection, such as the Fitzwilliam Virginal Book.

One important keyboard work from the early 17th century (a real masterpiece, in fact) is Orlando Gibbons's Lord Salisbury Pavane. It's in A minor. If you look at it (it's in the Historical Anthology of Music), Gibbons several times goes to the major triads on F, G, and C (which are in A natural minor), E (in the harmonic major), and D (not in A minor). He never, however, uses a B major (V/V) or F# major (V/ii) triad, even though V/V is not rare and V/ii not unthinkable in a minor key. He avoids them because they don't really exist in the tuning of his harpsichord. Had Gibbons begun in the key of C minor, he would have had to write a different piece, because instead of moving from A minor to F major, he would have had to move from C minor to Ab major, and Ab major, strictly speaking, didn't exist on his harpsichord.

Because it determines what sounds good, tuning has a pervasive influence on compositional tendencies. Every piece of pitched music is the expression of a tuning. Meantone encouraged composers to use major and minor triads, to avoid open perfect fifths without thirds, and to not stray more than three or four steps in the circle of fifths away from a central key. Renaissance and early Baroque music played in meantone sounds seductively sweet and attractive. By playing it in modern equal temperament, we do violence to its essential nature. Perhaps that's why this repertoire is no longer often heard. It's been painted over with the ugly gray of equal temperament.

One last point: Why is it called meantone? Because it splits the difference on where to place certain pitches. If C and E are tuned as a perfect major third of 386 cents, D should be tuned at 204 cents (9/8) for the key of C, but at 182 cents (10/9) for the key of D. Tuned at 193, D is right in the middle, halfway between C and E, and haflway between the two points it needs to be in for the various common keys; 193 is the mean between182 and 204. Meantone temperament sacrificed the seconds, which were mainly melodic intervals rather than harmonic ones anyway, to achieve beautiful thirds.

(Return to top menu)

3. Werckmeister III and Bach's W.T.C.

If you are or were ever a college music student, you probably read, or were told, that Johann Sebastian Bach wrote his collection of preludes and fugues The Well-Tempered Clavier in all 24 major and minor keys in order to demonstrate equal tempered tuning.

If so, you were misinformed.

Bach did not use equal temperament. In fact, in his day there was no way to tune strings to equal temperament, because there were no devices to measure frequency. They had no scientific method to achieve real equal-ness; they could only approximate.

Bach was, however, interested in a tuning that would allow him the possibility of working in all

12 keys, that did not make certain triads off-limits. He was a master of counterpoint, and chafed and fumed when the music in his head demanded a triad on A-flat and the harpsichord in front of him couldn't play it in tune. (In fact, he used to torment his organ tuner by playing sour Ab-major triads when the old man came in to work.) So he was glad to see tuners develop a tuning that, today, is known as well temperament. Back then, they did call it equal temperament - not because the 12 pitches were equally spaced, but because you could play equally well in all keys. Each key, however, was a little different, and Bach wrote The Well-Tempered Clavier in all 24 major and minor keys in order to *capitalize* on those differences, not because the differences didn't exist.

In any case (according to Jorgensen), the error that Bach wrote the W.T.C. in order to take advantage of what *we* call equal temperament crept into the 1893 Grove Dictionary, and has since been uncritically taught as fact to millions of budding musicians. Lord knows how long it will take to get that error out of the universities. It's still in all kinds of reference books.

The theorist who came up with the easiest way to tune the kind of well temperament Bach needed was the German organist Andreas Werckmeister (1645-1706), whose most famous tuning, dating from 1691, is known as Werckmeister III. A table for Werckmeister III is as follows:

Pitch:	C	C#	D	Eb	E	F	F#	G	G#	A	A#	B	C
Cents:	0	90.225	192.18	294.135	390.225	498.045	588.27	696.09	792.18	888.27	996.09	1092.18	1200

Notice that we've moved considerably closer to equal temperament; no pitch is more than 12 cents off. The following perfect fifths are 3/2 ratios of 701.955 cents each: Gb - Db - Ab - Eb - Bb - F - C, as well as A - E - B. The Pythagorean comma is distributed among the remaining fourths, C - G - D - A and B - F#, each of which is 696.09 cents. Let's look at the triads we now have on each pitch, organized for clarity's sake following the circle of fifths:

Major third	**Cents**	**Perfect Fifth**	**Cents**	**Minor third**	**Cents**
C - E	390.225	C - G	696.09	C - Eb	294.135
G - B	396.09	G - D	696.09	G - Bb	300.0
D - F#	396.09	D - A	696.09	D - F	305.865
A - C#	401.955	A - E	701.955	A - C	311.73
E - G#	401.955	E - B	701.955	E - G	305.865
B - D#	401.955	B - F#	696.09	B - D	300.0
F# - A#	407.82	F# - C#	701.955	F# - A	300.0
Db - F	407.82	Db - Ab	701.955	C# - E	300.0
Ab - C	407.82	Ab - Eb	701.955	G# - B	300.0
Eb - G	401.955	Eb - Bb	701.955	Eb - Gb	294.135
Bb - D	396.09	Bb - F	701.955	Bb - Db	294.135
F - A	390.225	F - C	701.955	F - Ab	294.135

As you look down the columns, you can get an idea of the quality of each triad. Note that no

perfect fifth is narrower than 696 cents, nor wider than 702; this is what renders all 12 (or 24 keys) usable. The closest major thirds to perfect are C-E and F-A. G-B, D-F#, and Bb-D are each 396.09 cents, still sweeter than equal temperament. A-C#, E-G#, and Eb-G are around 401 cents, close to equal temperament; they therefore have a rather bland, neutral quality. The major thirds on F#, Db, and Ab are 408 cents wide, the same size as in Pythagorean tuning (for which, see below), and not very attractive. Again, the best minor triads are grouped around A minor, with the minor third A-C, at 312 cents, coming closest to the optimum of 316 cents.

So what is the effect of Werckmeister III? Can the ear really hear a difference from equal temperament?

I've done experiments with students at Bard and Bucknell, playing preludes from the W.T.C. in different keys on a sampled piano tuned to Werckmeister III; say, playing the C major prelude in B, C, and D (computer-sequenced, so that the quality of the transposed performances wasn't a factor). Especially at Bard, **the students could invariably pick which was the appropriate key for each prelude.** In keys with poor consonances, like F# major, Bach will pass quickly by the major third, and the slight touches of dissonance give the prelude a bright, sparkly air. In more consonant keys, as in the C major prelude, the tonality is much more mellow, and Bach can afford to dwell on the tonic triad. Each key has a different color (as opposed to the uniform color of all keys in equal termperament), and even (or especially!) the unpracticed ear can hear appropriate and inappropriate correspondences between the character of each prelude and the color of each key. Of course, there are preludes that sound fine in more than one key; but it's disconcerting to move a prelude to a distant key, such as from Bb to B, or C# minor to Eb minor.

Playing Bach's Well-Tempered Clavier in today's equal temperament is like exhibiting Rembrandt paintings with wax paper taped over them.

(Return to top menu)

4. Young's Well Temperament and Classical-Era Music

I keep my own grand piano tuned to Thomas Young's well temperament of 1799. Some synthesizers offer an alternate temperament called Velotti-Young; the Young referred to is Thomas Young (not, of course, La Monte). Jorgensen considers Young's Well Temperament to be the most elegant well temperament, with a fluid variety of tonal colors and a symmetry that matches the piano keyboard: all intervals are symmetrical around D and G# - that is, D-F# and D-Bb are the same size, G#-F# and Ab-Bb the same size, and so on. The chart is as follows:

Pitch:	C	C#	D	Eb	E	F	F#	G	G#	A	A#	B	C
Cents:	0	93.9	195.8	297.8	391.7	499.9	591.9	697.9	795.8	893.8	999.8	1091.8	1200

This is even closer to equal temperament; even so, when I switched to it, my piano tuner had to return twice within two months before it began to stabilize. (You'd be surprised how exactly your piano's soundboard can remember a 6-cent difference.) Let's look at the quality of the triads:

Major third	Cents	Perfect Fifth	Cents	Minor third	Cents
C - E	391.7	C - G	697.9	C - Eb	297.8
G - B	393.9	G - D	697.9	G - Bb	301.9
D - F#	396.1	D - A	698	D - F	304.1
A - C#	400.1	A - E	697.9	A - C	306.2
E - G#	404.1	E - B	700.1	E - G	310.3
B - D#	406	B - F#	700.1	B - D	304
F# - A#	407.9	F# - C#	702	F# - A	301.9
Db - F	406	Db - Ab	701.9	C# - E	297.8
Ab - C	404.2	Ab - Eb	702	G# - B	296
Eb - G	400.1	Eb - Bb	702	Eb - Gb	294.1
Bb - D	396	Bb - F	700.1	Bb - Db	294.1
F - A	393.9	F - C	700.1	F - Ab	295.9

This is a subtle tuning, quite usable in all keys, and the differences from equal temperament are more evident to the pianist playing in it than to the listener. The best major thirds are grouped in the circle of fifths around C-E, whereas the perfect fifths become more perfect in the black keys, which all have fifths of 702 cents. This gives the keys related to C a sweet, gentle quality, the black-note keys an austere, noble quality, and middle keys like Eb and A a neutral, ambiguous quality.

Certain keys are warmer than others; F# minor, for instance, imparts a lush quality to the slow movement of the *Hammerklavier* Sonata. Db major is surprising, almost too harsh, and if I happen to play Db and F alone on the keyboard the buzzy beats make me jump as though I had played a wrong note. I'm surprised, when I play the slow movement of Beethoven's *Appassionata* Sonata that he chose this bright key for such a mellow movement. (It makes me wonder if his deafness made him forget about the varying qualities of the keys.)

Nineteenth-century musicians used to argue about what colors the various keys represented; whether Eb major was gold, for example, and D major red. Twentieth-century musicians have dismissed such arguments as sentimental nonsense, but when you play 19th-century music in well temperament, you begin to hear the differences of color. Is it far-fetched to suggest that Mozart and Beethoven wrote keyboard music with certain key-colors in mind, and that we miss subtle but pervasive qualities in the music when we homogenize it into equal temperament?

(Return to top menu)

5. A Word about Pythagorean Tuning

Before the advent of meantone tuning, the French academy at Notre Dame (13th and 14th centuries) decreed that only a series of perfect fifths could make up a scale; their ratio was 3/2, and 3, after all, connoted the Trinity. Thus the Pythagorean scale is a just-intonation scale on a series of perfect fifths, all the ratio numbers powers of either 3 or 2:

Pitch:	C	C#	D	Eb	E	F	F#	G	G#	A	A#	B	C
Ratios:	1/1	2187/2048	9/8	32/27	81/64	4/3	729/512	3/2	128/81	27/16	16/9	243/128	2/1
Cents:	0	113.7	203.9	294.1	407.8	498	611.7	702	792.2	905.9	996.1	1109.8	1200

This was an appropriate scale for a music in which perfect fifths and fourths were the overwhelmingly dominant sonority, and in which the pitches C#, F#, and G# hardly appeared if at all. Though used, the thirds were theoretical dissonances, and therefore avoided at final cadences: the major third, 81/64, was 408 cents wide, and the minor third, 32/27, 294 cents. As Margo Schulter has convincingly written me, however, those wide thirds do provide a compelling pull to the perfect fifths they usually resolve outward to; that is, in a cadence typical of Guillaume de Machaut (c. 1300-1377), a D and F# 408 cents apart will move outwardly to C and G. Gradually, especially under the English influence of John Dunstable and others, the thirds began to be redefined as 5-related intervals, 5/4 and 6/5, precipitating the necessity of meantone tuning and a revolution in musical style that led to the Renaissance. Since equal temperament has close-to-perfect fifths (700 cents compared to a perfect 702), much music written in Pythogorean tuning doesn't fare too badly in equal temperament. The Hilliard Ensemble observes Pythagorean tuning in its recordings of the Machaut Notre Dame Mass (Hyperion) and the organum of Perotin (ECM).

(Return to top menu)

6. Conclusion

I wish I could offer a wider disography of recordings in historical tunings. Luckily, the first recording of Beethoven in well temperament has just appeared: "Beethoven in the Termperaments," with pianist Enid Katahn and piano tuner Edward Foote (Gasparo Discs). The disc offers the Moonlight Sonata, the Waldstein, and the Pathetique in a late-18th-century temperament that brings out subtle color differences among the keys. And Francis Markey has written from Uppsala, Sweden, to tell me that many recordings of Renaissance-and-earlier brass music are played in just intonation. The same is more or less true, of course, of many string quartet recordings and old-fashioned barbershop quartets. Then, there are the Hilliard Ensemble recordings in Pythagorean tuning given above.

It may be that some of the many original-practice harpsichord recordings and European organ recordings use meantone. I haven't run into any that document their tuning. This whole subject has been so well hidden by the universities and musical authorities that very few musicians even realize how arbitrary, recent, and misleading our current universal tuning is. I was introduced to the subject by pianist Phillip Bush, who played a concert in New York a few years ago featuring Renaissance music in meantone and Beethoven in well temperament. For those further interested, I highly recommend Owen Jorgensen's four-inch thick Tuning compendium (Michigan State University Press, 1991). And I hope this will spark some interest that will lead to further experiments in reclaiming the original beauty of Europe's musical past.

(Return to top menu)

Return to the Just intonation Explained page.

If you find any of this not clearly enough expressed, e-mail me at **kgann@earthlink.net**

return to the home page

Dominant spectral region

The spectral-frequency band in which the ear can receive acoustic signals extends from about 20 to 16000 Hz - at least for a young and normal-hearing person. When a sound stimulus is reduced to a narrow frequency band, one can be sure that it will be appreciated by the auditory system even when that band occurs near the low or the high end of the auditory range of frequencies (provided, of course, that the signal's intensity exceeds the threshold of hearing). However, when a broad-band signal is presented that covers the entire frequency band of audition, there is an intermediate frequency region through which the majority of information is acquired by the auditory system. There is a *dominant spectral-frequency region.*

The position and the extension of the dominant region can roughly be inferred from existing systems for audio communication. The frequency band of the telephone channel was for economical reasons confined to 300-3400 Hz, and this, of course, was based on the observation that it suffices for high-intelligibility speech communication. The AM radio channel, extending from about 100 Hz to 4000 Hz, was for decades regarded more or less acceptable even for transmission of music.

Of course, one condition for those frequency bands to be sufficient is that the sound signals to be transmitted include essential information in these bands. However, as those channels are useless if they don't transmit essential information *to the ear*, it is the auditory system that determines what is essential. This is why the layout of the telephone- and AM-radio channels tell us something about the auditory system's dominant spectral region.

For instance, transmission of the frequency band below 300 Hz obviously is not essential for speech communication. This is not trivial, as the oscillation frequency of the human voice (both of men and women) essentially is below 300 Hz. If perception of the voice pitch were dependent on the presence of the lower harmonics of the speech signal, this could not work (see topic virtual pitch). On the other hand, transmission of the first two or three formants of speech *is* essential. These formants indeed are included in the telephone frequency band.

More information about the dominant spectral region comes from a number of aspects of audio communication. French & Steinberg (1947a) and Pollack (1948a) have measured speech intelligibility of high-pass and low-pass limited channels, respectively. French and Steinberg found that filtering out the frequency band *beyond* about 1800 Hz (low-pass) had the same reducing effect on the intelligibility of monosyllables as did filtering out the band *below* 1800 Hz (high-pass). This can be regarded as an experimental verification of the notion that consonants are about as important as the first two formants of vowels, as the former essentially are represented above about 1800 Hz, the latter, below. Moreover it is interesting to note that the frequency 1800 Hz, in the projection of the frequency scale on the length of the cochear partition, roughly corresponds to the middle of the the latter's extension.

On first sight, these observations suggest that 1800 Hz might be the center of the dominant region - at least where intelligibility of monosyllables is concerned. However, while in a sense this may appear reasonable, in another sense it is not. At least, 1800 Hz does not appear to be the "most important" frequency. From their data on intelligibility of monosyllables French & Steinberg deduced an "importance function", i.e., according to the criterion of how drastically the so-called *articulation index* is affected by a small change of the filter's cut-off frequency. The articulation index is defined

as the negative logarithm of the relative number of errors made in syllable recognition, and such is a measure of the transmitted information. The "importance function" such obtained by French & Steinberg as a function of frequency has a maximum at about 700 Hz and decays on both sides almost symmetrically on a log-frequency scale.

So, at least with respect to intelligibility of speech, 700 Hz appears to be the "most important" or "most dominant" frequency. Relating this to the bands used by the telephone- and AM-radio channels, one may notice that 700 Hz is not too far off the geometric mean of the respective cut-off frequencies. These means are about 630 and 1000 Hz, respectively. Thus one can say that on a logarithmic frequency scale the "most important" frequency (700 Hz) lays in the middle between the lower and upper cut-off frequencies - which intuitively appears to make sense, indeed.

Another aspect is perception of virtual pitch. Schouten (1962a), Plomp (1967a), Ritsma (1967a), and Yost (1982a) explored in independent types of experiments the question, from which spectral region the mechanism that creates virtual pitch (residue pitch) picks up the necessary information, when the Fourier spectrum of the stimulus extends from a low fundamental frequency up to several kHz. From their results one can infer that this, roughly, is the frequency region extending from about 400 to 2000 Hz. As the geometric mean of these values is about 890 Hz, which is not too far off 700 Hz, one may conclude that the auditory system's dominant region for speech intelligibility may be more or less identical with that for the formation of virtual pitch, which is a remarkable coincidence [55], [56], [104] p. 349-351.

When we worked out the algorithm for calculating virtual pitch and its relative prominence, a quantitative specification of an "importance" or "dominance" function was required. We designed that function according to the criterion that the algorithm's predictions of the strike note of bells should be optimal [55], [56]. This appeared appropriate, as finding the strike note of bells from the partials is a challenge to any pitch model in which the relative dominance of partials plays a prominent role [65]. As a result, a weighting function for the spectral pitches was found that agrees well with the "importance" function for the intelligibility of speech found by French & Steinberg (1947a). In particular, the frequency of the maximum of spectral-pitch weighting at 700 Hz was found to be optimal.

It will thus appear that the "importance" and "weighting" functions just discussed characterize a universal property of the auditory system. This view is supported by the finding of Bilsen & Raatgever (1973a), that in binaural lateralization a spectral dominance region is involved that resembles the one described above.

Author: Ernst Terhardt terhardt@ei.tum.de - Feb. 20, 2000

---

main page

Jean Baptiste Joseph Fourier

Born: 21 March 1768 in Auxerre, Bourgogne, France
Died: 16 May 1830 in Paris, France

Click the picture above
to see four larger pictures

Show birthplace location

Previous (Chronologically) Next Biographies Index
Previous (Alphabetically) Next Main index

Joseph Fourier's father was a tailor in Auxerre. After the death of his first wife, with whom he had three children, he remarried and Joseph was the ninth of the twelve children of this second marriage. Joseph's mother died went he was nine years old and his father died the following year.

His first schooling was at Pallais's school, run by the music master from the cathedral. There Joseph studied Latin and French and showed great promise. He proceeded in 1780 to the Ecole Royale Militaire of Auxerre where at first he showed talents for literature but very soon, by the age of thirteen, mathematics became his real interest. By the age of 14 he had completed a study of the six volumes of Bézout's *Cours de mathematique.* In 1783 he received the first prize for his study of Bossut's *Méchanique en général.*

In 1787 Fourier decided to train for the priesthood and entered the Benedictine abbey of St Benoit-sur-Loire. His interest in mathematics continued, however, and he corresponded with C L Bonard, the professor of mathematics at Auxerre. Fourier was unsure if he was making the right decision in training for the priesthood. He submitted a paper on algebra to Montucla in Paris and his letters to Bonard suggest that he really wanted to make a major impact in mathematics. In one letter Fourier wrote

> *Yesterday was my* 21*st birthday, at that age* Newton *and* Pascal *had already acquired many claims to immortality.*

Fourier did not take his religious vows. Having left St Benoit in 1789, he visited Paris and read a paper on algebraic equations at the Académie Royale des Sciences. In 1790 he became a teacher at the Benedictine college, Ecole Royale Militaire of Auxerre, where he had studied. Up until this time there had been a conflict inside Fourier about whether he should follow a religious life or one of

mathematical research. However in 1793 a third element was added to this conflict when he became involved in politics and joined the local Revolutionary Committee. As he wrote:-

> *As the natural ideas of equality developed it was possible to conceive the sublime hope of establishing among us a free government exempt from kings and priests, and to free from this double yoke the long-usurped soil of Europe. I readily became enamoured of this cause, in my opinion the greatest and most beautiful which any nation has ever undertaken.*

Certainly Fourier was unhappy about the Terror which resulted from the French Revolution and he attempted to resign from the committee. However this proved impossible and Fourier was now firmly entangled with the Revolution and unable to withdraw. The revolution was a complicated affair with many factions, with broadly similar aims, violently opposed to each other. Fourier defended members of one faction while in Orléans. A letter describing events relates:-

> *Citizen Fourier, a young man full of intelligence, eloquence and zeal, was sent to Loiret. ... It seems that Fourier ... got up on certain popular platforms. He can talk very well and if he put forward the views of the Society of Auxerre he has done nothing blameworthy...*

This incident was to have serious consequences but after it Fourier returned to Auxerre and continued to work on the revolutionary committee and continued to teach at the College. In July 1794 he was arrested, the charges relating to the Orléans incident, and he was imprisoned. Fourier feared the he would go to the guillotine but, after Robespierre himself went to the guillotine, political changes resulted in Fourier being freed.

Later in 1794 Fourier was nominated to study at the Ecole Normale in Paris. This institution had been set up for training teachers and it was intended to serve as a model for other teacher-training schools. The school opened in January 1795 and Fourier was certainly the most able of the pupils whose abilities ranged widely. He was taught by Lagrange, who Fourier described as

> *the first among European men of science,*

and also by Laplace, who Fourier rated less highly, and by Monge who Fourier described as

> *having a loud voice and is active, ingenious and very learned.*

Fourier began teaching at the Collège de France and, having excellent relations with Lagrange, Laplace and Monge, began further mathematical research. He was appointed to a position at the Ecole Centrale des Travaux Publiques, the school being under the direction of Lazare Carnot and Gaspard Monge, which was soon to be renamed Ecole Polytechnique. However, repercussions of his earlier arrest remained and he was arrested again imprisoned. His release has been put down to a variety of different causes, pleas by his pupils, pleas by Lagrange, Laplace or Monge or a change in the political climate. In fact all three may have played a part.

By 1 September 1795 Fourier was back teaching at the Ecole Polytechnique. In 1797 he succeeded Lagrange in being appointed to the chair of analysis and mechanics. He was renowned as an outstanding lecturer but he does not appear to have undertaken original research during this time.

In 1798 Fourier joined Napoleon's army in its invasion of Egypt as scientific adviser. Monge and Malus were also part of the expeditionary force. The expedition was at first a great success. Malta was occupied on 10 June 1798, Alexandria taken by storm on 1 July, and the delta of the Nile quickly taken. However, on 1 August 1798 the French fleet was completely destroyed by Nelson's fleet in the Battle of the Nile, so that Napoleon found himself confined to the land that he was occupying. Fourier acted as an administrator as French type political institutions and administration was set up. In particular he helped establish educational facilities in Egypt and carried out archaeological explorations.

While in Cairo Fourier helped found the Cairo Institute and was one of the twelve members of the mathematics division, the others included Monge, Malus and Napoleon Bonaparte. Fourier was elected secretary to the Institute, a position he continued to hold during the entire French occupation of Egypt. Fourier was also put in charge of collating the scientific and literary discoveries made during the time in Egypt.

Napoleon abandoned his army and returned to Paris in 1799, he soon held absolute power in France. Fourier returned to France in 1801 with the remains of the expeditionary force and resumed his post as Professor of Analysis at the Ecole Polytechnique. However Napoleon had other ideas about how Fourier might serve him and wrote:-

> *... the Prefect of the Department of Isère having recently died, I would like to express my confidence in citizen Fourier by appointing him to this place.*

Fourier was not happy at the prospect of leaving the academic world and Paris but could not refuse Napoleon's request. He went to Grenoble where his duties as Prefect were many and varied. His two greatest achievements in this administrative position was overseeing the operation to drain the swamps of Bourgoin and to oversee the construction of a new highway from Grenoble to Turin. He also spent much time working on the *Description of Egypt* which was not completed until 1810 when Napoleon made changes, rewriting history in places, to it before publication. By the time a second edition appeared every reference to Napoleon would have been removed.

It was during his time in Grenoble that Fourier did his important mathematical work on the theory of heat. His work on the topic began around 1804 and by 1807 he had completed his important memoir *On the Propagation of Heat in Solid Bodies.* The memoir was read to the Paris Institute on 21 December 1807 and a committee consisting of Lagrange, Laplace, Monge and Lacroix was set up to report on the work. Now this memoir is very highly regarded but at the time it caused controversy.

There were two reasons for the committee to feel unhappy with the work. The first objection, made by Lagrange and Laplace in 1808, was to Fourier's expansions of functions as trigonometrical series, what we now call Fourier series. Further clarification by Fourier still failed to convince them. As is pointed out in [4]:-

> *All these are written with such exemplary clarity - from a logical as opposed to calligraphic point of view - that their inability to persuade* Laplace *and* Lagrange *... provides a good index of the originality of Fourier's views.*

The second objection was made by Biot against Fourier's derivation of the equations of transfer of heat. Fourier had not made reference to Biot's 1804 paper on this topic but Biot's paper is certainly

incorrect. Laplace, and later Poisson, had similar objections.

The Institute set as a prize competition subject the propagation of heat in solid bodies for the 1811 mathematics prize. Fourier submitted his 1807 memoir together with additional work on the cooling of infinite solids and terrestrial and radiant heat. Only one other entry was received and the committee set up to decide on the award of the prize, Lagrange, Laplace, Malus, Haüy and Legendre, awarded Fourier the prize. The report was not however completely favourable and states:-

> *... the manner in which the author arrives at these equations is not exempt of difficulties and that his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.*

With this rather mixed report there was no move in Paris to publish Fourier's work.

When Napoleon was defeated and on his way to exile in Elba, his route should have been through Grenoble. Fourier managed to avoid this difficult confrontation by sending word that it would be dangerous for Napoleon. When he learnt of Napoleon's escape from Elba and that he was marching towards Grenoble with an army, Fourier was extremely worried. He tried to persuade the people of Grenoble to oppose Napoleon and give their allegiance to the King. However as Napoleon marched into the town Fourier left in haste.

Napoleon was angry with Fourier who he had hoped would welcome his return. Fourier was able to talk his way into favour with both sides and Napoleon made him Prefect of the Rhône. However Fourier soon resigned on receiving orders, possibly from Carnot, that the was to remove all administrators with royalist sympathies. He could not have completely fallen out with Napoleon and Carnot, however, for on 10 June 1815, Napoleon awarded him a pension of 6000 francs, payable from 1 July. However Napoleon was defeated on 1 July and Fourier did not receive any money. He returned to Paris.

Fourier was elected to the Académie des Sciences in 1817. In 1822 Delambre, who was the Secretary to the mathematical section of the Académie des Sciences, died and Fourier together with Biot and Arago applied for the post. After Arago withdrew the election gave Fourier an easy win. Shortly after Fourier became Secretary, the Academy published his prize winning essay *Théorie analytique de la chaleur* in 1822. This was not a piece of political manoeuvring by Fourier however since Delambre had arranged for the printing before he died.

During Fourier's eight last years in Paris he resumed his mathematical researches and published a number of papers, some in pure mathematics while some were on applied mathematical topics. His life was not without problems however since his theory of heat still provoked controversy. Biot claimed priority over Fourier, a claim which Fourier had little difficulty showing to be false. Poisson, however, attacked both Fourier's mathematical techniques and also claimed to have an alternative theory. Fourier wrote *Historical Précis* as a reply to these claims but, although the work was shown to various mathematicians, it was never published.

Fourier's views on the claims of Biot and Poisson are given in the following, see [4]:-

> *Having contested the various results* [Biot *and* Poisson] *now recognise that they are exact but they protest that they have invented another method of expounding them and that this method is excellent and the true one. If they had illuminated this branch of*

> *physics by important and general views and had greatly perfected the analysis of partial differential equations, if they had established a principal element of the theory of heat by fine experiments ... they would have the right to judge my work and to correct it. I would submit with much pleasure .. But one does not extend the bounds of science by presenting, in a form said to be different, results which one has not found oneself and, above all, by forestalling the true author in publication.*

Fourier's work provided the impetus for later work on trigonometric series and the theory of functions of a real variable.

Article by: *J J O'Connor* and *E F Robertson*

Click on this link to see a list of the Glossary entries for this page

List of References (24 books/articles) **Some Quotations** (5)

A Poster of Joseph Fourier **Mathematicians born in the same country**

Cross-references to History Topics

1. Topology enters mathematics
2. orbits and gravitation
3. An overview of the history of mathematics

Other references in MacTutor

1. Chronology: 1800 to 1810
2. Chronology: 1820 to 1830

Honours awarded to Joseph Fourier
(Click a link below for the full list of mathematicians honoured in this way)

Fellow of the Royal Society Elected 1823

Lunar features **Crater Fourier**

Commemorated on the Eiffel Tower

Other Web sites

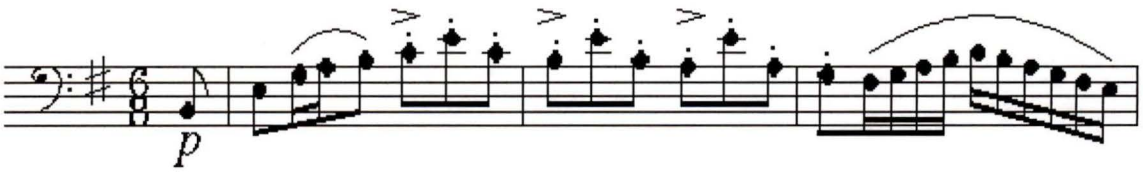
1. Rouse Ball
2. Encyclopaedia Britannica

Previous (Chronologically) Next Biographies Index
Previous (Alphabetically) Next Main index
History Topics Societies, honours, etc. Famous curves
Time lines Birthplace maps Chronology Search Form
Glossary index Quotations index Poster index
Mathematicians of the day Anniversaries for the year

JOC/EFR January 1997

School_of_Mathematics_and_Statistics
University_of_St_Andrews,_Scotland

The URL of this page is:
http://www-history.mcs.st-andrews.ac.uk/history/Mathematicians/Fourier.html



The Mathematics of Tuning and Temperament
With audio examples

Contents

---

1. Introduction

Almost all Western music is based upon the diatonic scale, having seven notes within an octave. The octave is the fundamental interval for which the notes are related in frequency exactly by the ratio of 2:1. This simple ratio means that to the ear they sound as if they are almost "the same note". Virtually all modern Western music divides the octave into 12 equal semi-tones on which the music is based. The system is a compromise, and the octave has not always been divided this way. Over the ages there have been four principle tuning systems in wide usage for Western music. This paper takes a look at these four systems with illustrative audio clips.

This description of the mathematical basis is very abridged. It is not intended to be a definitve work on the subject, but rather a practical illustration of the various aspects of different systems. Various audio samples are included to illustrate a few of the key points. These are encoded as MPEG Layer 3 audio at 16kbps. For more detailed information on the subject the reader is referred to the many excellent references, historical, musical and mathermatical, that may be found.

---

2. General Mathematical Basis

For pure monophonic (one note at a time) musical melody, it does not matter a great deal

what scale or tuning method is used. However, could you imagine modern music with only one note being played at a time? Almost all modern music is polyphonic (many notes sounding simultaneously). For the music to sound consonant (in tune) the different notes being sounded simultaneously need to be related in a way which is pleasing to the ear. The relationship between the frequencies of notes of the important musical intervals is what makes them sound pleasing. The important ratios are as follows:

- 3:2, perfect fifth
- 4:3, fourth
- 5:3, major sixth
- 5:4, major third
- 6:5, minor third
- 8:5, minor sixth

All four of the main tuning systems considered here were devised around some or most of these intervals.

---

3. The Principle Tuning Systems

NOTE: The values given as "cents" represent the intervals where the octave is divided into 1200 cents. This is the modern way of representing pitch in such a way that each semi-tone comprises 100 cents.

3.1 Pythagorean Tuning

Western music is generally considered to have started from Pythagoras, the ancient Greek. Pythagoras devised a system based on mathematical principles. He defined the scale around the ratios of the fifth, being in the ratio 3:2 exactly, and the fourth being 4:3 exactly. The difference between these two was then 9:8, which he defined as the tone, or whole step. He then divided the octave so that there were the seven notes, just as we have today, but to get the mathematics to add up he was left with two semi-tones which he defined as 256:243.

Thus the Pythagorean scale has the following intervals:

Cumulative Intervals:	1	9:8	81:64	4:3	3:2	27:16	243:128	2
Note:	C	D	E	F	G	A	B	C
Incremental Intervals:		9:8	9:8	256:243	9:8	9:8	9:8	256:243
Cents:		204	204	90	204	204	204	90

It is interesting to note that Pythagoras did not recognise the major third, which is distictly sharp at 81:64 compared with the ideal of 5:4.

The chromatic Pythagorean scale is formed by inserting semitones equal to 114 cents in such a way as to keep all perfect fifths true, except for the interval G#-E*b*, which needs to be adjusted so that the intervals add up mathematically. This difference is known as the "comma of Didymus".

3.2 Ptolemaic Tuning

Ptolemaic Tuning is generally referred to as "just intonation". To create perfect major third intervals, this system alters one of the fifths, D-A. (And thus the major sixths are also perfect, except for F-D.) This makes the triad D-F-A quite unusable, although the others are perfectly in tune. The Just Intonation scale employs two different sized tones in the ratios 9:8 and 10:9, and thus it can hardly be considered satisfactory even for purely melodic music.

Cumulative Intervals:	1	9:8	5:4	4:3	3:2	5:3	15:8	2
Note:	C	D	E	F	G	A	B	C
Incremental Intervals:		9:8	10:9	16:15	9:8	10:9	9:8	16:15
Cents:		204	182	112	204	182	204	112

The above comments hold true for scales and intervals based solely on the "white notes". Adding the "black notes" to give a full chromatic Just Intonation scale creates more perfect fifths, but major thirds and sixths which are not true. In fact the result is **three** different sizes of semi-tones. (16:15, 135:128 and 256:243.) It is thought that this system, although considerably debated, was not used much.

3.3 Mean-tone Temperament

In this scheme the major thirds are made exact. This results in the fifths becoming slightly flattened but in such a way that the error of the Ptolemaic system is spread out over four fifths. This reduces the dissonance and makes the fifths more acceptable. The whole tones are also all equal in size, being half the major third. Melodically the Mean-tone scale is more acceptable than the Ptolemaic scale.

Cumulative Intervals:	1		5:4					2
Note:	C	D	E	F	G	A	B	C
Cents:		193	193	117	193	193	193	117

The chromatic Mean-tone scale has semi-tones of two very different sizes: wide 117 cent semi-tones in the diatonic scale, with 76 cent semi-tones balancing the whole-tone interval of 193 cets. Mean-tone termperament was designed for keyboard instruments and it was an acceptable compromise as long as the "black notes" beyond E *b* or G# were not used. The G#:E*b* fifth was so bad as to be unusable - it was often given the name "the wolf".

3.4 Equal Temperament

In equal temperament the octave is divided into 12 equal semi-tones each of 100 cents. The intervals are then built from the semi-tones. For example a fifth is 7 semi-tones and a third 5. The Equal temperament scale is universally used today in Western music.

Cumulative Intervals:	1							2
Note:	C	D	E	F	G	A	B	C
Cents:		200	200	100	200	200	200	100

The chromatic Equal temperament scale with all semi-tones = 100 cents means that perfectly tuned intervals have been totally eliminated. However, the mistuning on fifths is only 2 cents and on thirds 14 cents which the ear does not appear to mind. The big

advantage is that all keys are equally usable.

---

4. Harmonic Comparison

The differences between the tuning systems are most evident in polyphonic music, particularly when chords and intervals incorporating the "black notes" are used. To illustrate this samples of the basic two-note chords for some of the 5th, 3rd and 6th intervals have been constructed. Each of the audio samples consists of a sequence of the chord based on: Pythagorean, Just intonation, Mean-tone temperament and Equal intonation played directly one after the other.

4.1 Perfect Fifths

The basic diatonic 5th interval is the Perfect 5th C:G. The deviation from perfect tuning is 0, 0, -5, -2 cents respectively.

In all systems the 5ths are perfectly tuned, or very nearly so, with the exception of two intervals. The Perfect 5th D:A is -22 cents in just intonation, and the enharmonic fifth 5th G#:E*b* (the Wolf) is -24, -2, +35, -2 cents out respectively.

4.2 Major Thirds

The basic diatonic 3rd interval is the Major 3rd C:E. The deviation from perfect tuning is +22, 0, 0, +14 cents respectively. This illustrates the fact that the Major third was not recognised in Pythagorean tuning.

Apart from the three Major third intervals contained within the diatonic scale, all other thirds become sharp in Just intonation, and some of the enharmonic intervals become unusable in Mean-tone tuning as well. An example of this is the Major 3rd C#:F which has mis-tuning errors of -2, +20, +42, +14 cents respectively.

4.3 Major Sixths

The basic diatonic 6th interval is the Major 6th C:A. The deviation from perfect tuning is +22, 0, +6, +16 cents respectively.

In the Just intonation system all sixths incorporating one or more of the notes not in the diatonic scale and the sixth F:D become sharp by 22 cents. Some of the enharmonic intervals become unusable in Mean-tone temperament. An example is the Major 6th C#:B*b* which has tuning errors of -2, +20, +49, +16 cents respectively.

---

5. Summary of sample sounds

Here is a complete list of audio clips available:

Pythagorean scale
Just intonation scale

Mean-tone scale
Equal temperament scale
Chromatic Pythagorean scale
Chromatic Just intonation scale
Chromatic Mean-tone scale
Chromatic Equal temperament scale
Perfect fifth C:G
Perfect fifth D:A
Perfect fifth G#:E*b*
Major third C:E
Major third C#:F
Major sixth C:A
Major sixth C#:B*b*

Acoustics, sound and music Home page.

Virtual pitch

When you listen to a male speaking voice emanating from your hi-fi stereo set while sound reproduction of the bass range is well enhanced, you will hear the lowest Fourier component of the uttered vowels, i.e., the so-called fundamental, rising and falling in pitch, just as if it were a boosted bass tone of music.It indeed *is* a bass tone, as the oscillation frequency of a typical adult male speaking voice extends from about 70 to 150 Hz. This is one of the rare cases in which you may consciously perceive the fundamental - and its pitch - of a male speaking voice. Especially for the reproduction of speech it is neither necessary nor desirable to boost the low-frequency spectral components, because intelligibility suffers from it, and the voice pitch (intonation) is very well perceived without acoustic reproduction of the fundamental. This is the reason why, e.g., the telephone channel does not distort the pitch of speech although transmission is confined to the frequency range from about 300 to about 3400 Hz. When, in the above "experiment", you suppress bass reproduction using the equalizer of your stereo set, you will notice that the fundamental gets inaudible, while the speaker's voice pitch continues to be well reproduced.

The kind of pitch of the fundamental that you may hear if the fundamental indeed is strong enough, is the pitch of a sine tone; it is of the spectral pitch type. The pitch that you *ordinarily* hear, however, is not dependent on the fundamental being audible; it is by the auditory system extracted from a range of the Fourier spectrum that extends *above* the fundamental. The latter type of pitch is termed *virtual pitch* [14] , [17].

From the above example it can be concluded that, as an attribute of auditory sensation, virtual pitch is fundamentally different in type from spectral pitch. This conclusion is strongly suggested by the fact that one can hear both types of pitch at a time, having the same height. Evidently, it is possible to communicate one and the same pitch (in terms of pitch *height*) through two drastically different perceptual "channels": Spectral pitch is communicated *immediately*, i.e. by a Fourier component's frequency, while virtual pitch is communicated by providing to the auditory system information about the oscillation frequency of a complex signal that is implied in the Fourier spectrum as a whole.

Moreover, the above example illustrates that in real life perception of virtual pitch is far from being an exotic, rare phenomenon or even a kind of illusion. On the contrary, perception of virtual pitch is the rule rather than an exception.

In the acoustic laboratory, the above example can be mirrored by the following demonstration. A harmonic complex tone is generated that only includes, e.g., the 5th, 6th, 7th, and 8th harmonics of 200 Hz. (Such type of sound was by Schouten [1940a] termed a "residue".) That sound (especially when presented with a *low* intensity) elicits a virtual pitch corresponding to 200 Hz. Now, while listening to that "residue", another sound shall be switched on, namely, a 200-Hz sine tone. The latter is heard as another sound object, i.e., it can be distinguished from the "residue". Each of the two objects has its own pitch, i.e., independently of the other. Yet the two pitches are equal. One hears two different *types of pitch* (i.e., virtual and spectral) at a time, and it does not matter that they are equal in height.

While from a considerable number of observations it is evident that spectral pitch must be explained by a kind of "place theory" (see the discussion in topics diplacusis binauralis and spectral pitch), it is likewise evident that virtual pitch requires another explanation. So, spectral pitch and virtual pitch

differ not only in the aforementioned phenomenological aspects; they also must be theoretically explained by basically different mechanisms. So, as the basic outline of explaining spectral pitch is apparent, possible mechanisms of virtual-pitch formation remain to be discussed, as follows.

An explanation of virtual pitch that on first sight is highly suggestive is the "time-domain solution", i.e., measurement of the period of the tone signal. (Which, of course, requires that the signal actually *is* periodic.) That kind of solution was suggested already in the 19th Century, in particular by Seebeck (1841a). Schouten (1940c, 1970a) strongly promoted that solution. However, already in the 1950s and 1960s new observations had indicated that the time-domain model in its original design was not adequate.

Indeed, there are quite a number of obstacles against time-domain modeling of virtual pitch, such as the following.

- To elicit virtual pitch, a sound signal does not need to be strictly periodic.Real-life examples of such type of sound are the tones of the piano, and that of bells. Thus, when pitch is strictly made dependent on periodicity, an essential point is missed.
- The Fourier components of a signal that elicits virtual pitch do not need to be present at the same ear; they can be distributed to the two ears (Houtsma & Goldstein 1972a). This implies that virtual pitch can be perceived although neither of the two ear signals includes any periodicity that would correspond to the pitch perceived.
- To elicit virtual pitch, the pertinent Fourier components do not even need to be *simultaneously* present (Hall & Peters 1981a). This implies that there does not exist any periodicity in the ear signal that would correspond to the pitch perceived.
- It is hard to see how a time-domain model should be able to account for the phenomena of pitch shifts and octave stretch.
- Pitch (i.e., both spectral pitch and virtual pitch) is a highly *robust* phenomenon, i.e., it is little or not at all affected by drastic linear distortion and/or additional sound. By contrast, the structure of a sound signal is heavily dependent on both linear distortion and superimposed additional sound.
- Pitch (i.e., both spectral pitch and virtual pitch) is in particular robust with respect to changes of the *phases* of Fourier components (which, strictly, was already addressed by the reference to linear distortion). By contrast, the structure of any time signal is heavily dependent on component phases.
- Pitch is *multiple*, and multiple sound objects can elicit pertinent pitches. So it is not sufficient to measure the period - or the like - of an isolated tone. Rather, *pitch determination* and *sound-object segregation* must go hand in hand.

In spite of these difficulties with explaining virtual pitch in the time domain, a considerable number of attempts of that type have been described in the literature until present days. However, I am not aware of a satisfactory solution.

Returning to the notion that two separate solutions are required for the explanation of spectral pitch and virtual pitch, respectively, up to this point the question remained open as to *how* separate are these two mechanisms. The above time-domain model of virtual pitch - if it actually worked - were in fact totally separate from, i.e., independent of, the spectral-pitch mechanism. The two models would operate "in parallel". There is, however, the alternative that they operate "in series", more precisely, in hierarchical steps of processing. This indeed is how the virtual-pitch theory works.

The virtual-pitch theory [14], [17], [18], which I worked out in 1969/70, explicitly and strictly

pursues the latter kind of approach. Virtual pitch is conceptualized as a percept of "higher order" than spectral pitch [10]. The relationship between virtual pitch and spectral pitch is conceptualized as analogous to the relationship between virtual and primary visual contour. Just like visual virtual contours (or, as they unfortunately have also been termed, "illusory" contours) are based on visual primary contours, so are virtual pitches based on spectral pitches.

Formation of virtual pitch can essentially be said to be a process of *subharmonic matching*: The tonal aspects of any sound are primarily represented by a set of spectral pitches, and pertinent virtual pitches are "inferred" on the basis of the presumption that in any case they must be subharmonic to the spectral pitches.

For example, when three Fourier components with the frequencies 600, 800, and 1000 Hz are heard, the auditory system is wise enough to know that "in real life nothing is just what it appears to be". That is, not only the spectral pitches corresponding to 600, 800, and 1000 Hz are apprehended, but it is supposed that these components are likely to be harmonics of a complex tone the lower harmonics of which have been attenuated or even removed by linear distortion of the sound path. So it is advisable to look out for the fundamental pitch of that prospective complex tone. This is done by assuming subharmonic virtual pitches of each of the spectral pitches to be candidates of the unknown fundamental pitch. In terms of frequency, those subharmonics are simply obtained by dividing every component frequency by each of the integer numbers 1, 2, 3, and so on, until about 12. Then the test of subharmonic match comes into play. Obviously, in the above example, a first "full match" is obtained at the fundamental frequency 200 Hz, as this is the 3rd subharmonic of 600 Hz, the fourth subharmonic of 800 Hz, and the fifth subharmonic of 1000 Hz. By definition, such a match amplifies the relative prominence which is assigned to any of the virtual-pitch candidates, such that the above "full match" will yield a well pronounced virtual pitch at 200 Hz. However, another "full match" is obtained at 100 Hz, and since this also is an oscillation frequency that frequently occurs in real life, it is considered as an alternative virtual pitch that competes in prominence with the former one. This is how both multiplicity of virtual pitch, and pitch-commonality (see topics tone affinity, octave equivalence) of harmonic complex tones come about.

This simple example should also suffice to illustrate that the formation of virtual pitch basically cannot fail for *any* type of sound, provided that a few spectral pitches are elicited. (Strictly, even just one spectral pitch can be sufficient as a cue to virtual pitch, see below.) That is, any type of sound, no matter how it was created, will stimulate the auditory virtual-pitch mechanism to look out for subharmonic pitch matches, and this will yield a number of virtual pitches that are more or less prominent. This is how the theory explains not only the pitch of harmonic complex tones of any kind, but also explains phenomena such as the strike note of bells, the root of musical chords, and the "acoustic bass" of the organ. Moreover, as virtual pitch is dependent on spectral pitches, any kind of pitch deviations and/or pitch shifts must be reflected in the resulting virtual pitches, i.e., in a manner that is predetermined by the process of subharmonic pitch matching. The theory can predict pitch shift effects of virtual pitch of a size and kind that indeed have been observed [9], [11], [13] (see also topic diplacusis binauralis).

So, the virtual pitch mechanism deals with both "harmonic" and "inharmonic" sounds as well, though internally it strictly sticks to the presumption that each and every virtual-pitch candidate must be a subharmonic of a spectral pitch. Why is this so?

The first part of the answer is that this is the method by which the auditory system detects *periodicity*, i.e., even of *multiple* sound objects. Periodicity of a sound signal is invariably represented by *harmonicity* of its Fourier components. Thus, any harmonic series of spectral pitches that can be

found in a given set indicates that there is a periodic sound signal included in the entire complex sound.

The second part of the answer refers to the next question which immediately is elicited by the first, namely: why is detection of periodicity so important to the auditory system? The answer is that periodicity is a strong clue to a sound object. In real life, periodicity of an ear signal (i.e., the sound signal at the eardrum) is highly unlikely to occur when that ear signal is composed of the sound waves that emenate from two or more independent sources. Correspondingly, harmonicity of frequencies (and so, of spectral pitches) that originate from independent sound sources is highly unlikely to occur in real life. Hence, any harmonic series of spectral pitches that occur in a complex spectral-pitch pattern can safely be interpreted as emerging from a periodic portion of the auditory stimulus and can be ascribed to one particular sound source. The virtual pitch which is inferred from that harmonic series by subharmonic matching plays the role of a label that is attached to that periodic sound object. This is how the virtual-pitch theory accounts for both pitch determination and sound-object segregation.

These notions include an answer to yet another question, namely, why does the auditory system create virtual pitches at all? The answer is that virtual pitch is a label that characterizes a certain type of sound objects, namely periodic ones, including "supposedly" periodic ones. And segregation and identification of sound objects naturally is one of the fundamental functions of any hearing organ.

All this implies that the auditory system possesses "knowledge" about (sub)harmonic pitch intervals. That knowledge obviously is available to the system, i.e., as a kind of template on which the intervals between subharmonic pitches are engraved. As suggested by the existence of stretch of harmonic pitch intervals, that template appears to be at least *affected*, perhaps even *entirely acquired*, from natural samples of periodic sound signals. For Man, the most obvious and most important type of periodic sound is speech, i.e., its voiced elements. Acquisition of the template very probably happens in earliest, i.e., even pre-natal life [18], [22].

The virtual-pitch theory, after its foundation in [17], [18], was further elaborated to quantitatively account not only for the virtual and spectral pitches as such, but also to predict their relative prominence. Besides a number of other factors, the prominence of pitches is dependent on the frequency region in which the Fourier components occur (see topic dominant spectral region) [22], [38], [55], [56]. The entire process of pitch evaluation sketched above can be automatically carried out on a computer.

The virtual-pitch theory hardly needs to be defended in view of its success in explaining and quantitatively predicting a great variety of pitch phenomena. However, there is one aspect which deserves consideration. According to the virtual-pitch theory in its current design, there cannot occur any virtual pitch without existence of at least one spectral pitch. (That indeed one spectral pitch can be sufficient, was demonstrated by Houtgast 1976a). Occasionally it was claimed that (virtual) pitch can be observed in sounds which could not be conceived as eliciting any spectral pitch. To my knowledge, however, so far no direct experimental proof was offered for the total absence of spectral pitch in those test sounds. As I have pointed out on various occasions, it is almost impossible to find *any* sound that does not elicit even the faintest, and perhaps temporary, spectral pitch ([104] p. 323, see also topic spectral pitch). It may well be impossible to strictly prove the total absence of spectral pitch in any sound. Thus, the above evidence appears questionable, and disproving on that ground the assumption that for formation of virtual pitch at least one spectral pitch is required, may turn out to be impossible, as well.

However, one should not entirely ignore the possibility that the auditory percept of "rattling", or *roughness*, that ordinarily is evoked by a periodic sound signal - provided that the period is no less than about 3 ms - may suffice to evoke a faint and poorly defined sensation of virtual pitch (i.e., without existence of any spectral pitch). In my original layout of the virtual-pitch theory [17], [18] I have included this factor, though I assumed that roughness as a clue to virtual pitch may become useful only in combination with at least one spectral pitch. If there are any cases at all in which roughness could be proven to be sufficient as a clue to virtual pitch (i.e., without existence of any spectral pitch), I can conceive of such a virtual pitch only as a faint and poorly defined sensation. Moreover, the information on periodicity included in roughness can be available only for a single, isolated periodic stimulus. Auditory segregation of multiple sound signals on that basis does not appear to be conceivable.

In summary: When the votes provided by experimental evidence are counted and evaluated in a "democratic" manner, the hierarchical approach (spectral pitch determines virtual pitch) wins by a vast majority. Evidence in favor of the idea that virtual pitch emerges from an immediate analysis of an audio signal's time structure is rare and weak.

Author: Ernst Terhardt terhardt@ei.tum.de - Feb. 22, 2000

---

main page

VITA

Surname: Antoniou Given Names: Anthony

Place of Birth: London, England

Educational Institutions Attended:

University of Victoria 1989 to 2002

Degrees Awarded:

B.Eng. University of Victoria 1995

Honours and Awards:

Publications:

Partial Copyright License

I hereby grant the right to lend my thesis (or dissertation) to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis/Dissertation:

Author 

Anthony Antoniou, B. Eng.

April 2, 2002