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

Marko E. Takanen

Automated system level testing of a software audio platform

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

Espoo, February 17th, 2005

Supervisor: Professor Vesa Välimäki
Instructor: Jarmo Hiipakka, M.Sc. (Tech.)

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Marko E. Takanen
Name of the thesis:	Automated system level testing of a software audio platform
Date:	February 17th, 2005 **Number of pages:** 73
Department:	Electrical and Communications Engineering
Professorship:	S-89 Acoustics and Audio Signal Processing
Supervisor:	Professor Vesa Välimäki
Instructor:	Jarmo Hiipakka, M.Sc. (Tech.)

Mobile computing devices have great current and future potential in new audio processing applications. The Entertainment Audio Platform is a software package that enables such applications. This thesis describes the design and implementation of the methods required for automated testing of mixing, sampling rate conversion, and dynamic range controller functionality of the software.

Literature from both software and conventional audio testing is reviewed in the thesis, and relevant methods are selected for the task at hand. Test automation, widely described in the software engineering field, is applied to testing audio processing functionality. Especially, equivalence partitioning is applied to the case selection problem and test oracles are used for automated verification. Various signal analysis methods are used from the audio engineering field. Among others, the coherent sampling method is used for maximizing the measurement accuracy.

Testing is designed as system-level, or end-to-end testing using the application programming interface of the platform. This means that the individual algorithms operate in their natural context. In addition, testing is parameterized in order to allow flexible and extensive test case definition with moderate effort needed to set up the tests. This poses requirements on the implementation of the analysis functionality used in testing.

This thesis brings together testing methodologies from both software and audio engineering fields. As the main result, the thesis presents methods and implementations required in automated audio software testing.

Keywords: audio software, audio measurement technique, automated testing, black-box testing

Tekijä: Marko E. Takanen

Työn nimi: Automated system level testing of a software audio platform

Päivämäärä: 17. helmikuuta 2005 **Sivuja:** 73

Osasto: Sähkö- ja tietoliikennetekniikka

Professuuri: S-89 Akustiikka ja äänenkäsittelytekniikka

Valvoja: professori Vesa Välimäki

Ohjaaja: dipl.ins. Jarmo Hiipakka

Mobiililaitteilla on suuri nykyinen ja tuleva potentiaali uudenlaisille äänenkäsittely-sovelluksille. Entertainment Audio Platform on ohjelmistoalusta, joka mahdollistaa tämän tapaiset sovellukset. Tämä työ käsittelee menetelmiä, joita vaaditaan äänimiksauksen ohjauksen, näytetaajuusmuunnosten sekä signaalin dynamiikan kontrolloijan automatisoidussa testauksessa.

Työssä perehdytään kirjallisuuden avulla sekä ohjelmistojen että perinteisten audio-laitteiden testaukseen. Kirjallisuudesta poimitaan sopivia menetelmiä kuhunkin tehtävään. Testauksen automatisointia, jota on käsitelty laajasti ohjelmisto-tekniikassa, sovelletaan audiokäsittelytoiminnallisuuden testaamiseen. Erityisesti syöte- ja tulosarvojen ekvivalenssiluokittelua sovelletaan testitapausten valinnassa, sekä testioraakkeleja käytetään järjestelmän ulostulosignaalien tarkastuksen automatisointiin. Audiotekniikan alalta sovelletaan useita signaalin analysointi-menetelmiä. Muun muassa koherenttia näytteistystä käytetään parhaan mittaustark-kuuden saavuttamiseksi.

Testaus on suunniteltu tehtäväksi järjestelmätasolla läpi koko järjestelmän käyttäen alustan tarjoamaa sovellusohjelmointirajapintaa. Tällöin yksittäiset algoritmit toimivat lopullisessa yhteydessään. Lisäksi testaus on parametrisoitu, jotta mahdollistetaan joustava ja kattava testitapausten määrittely kohtuullisella vaivalla. Tämä asettaa vaatimuksia testin analysointitoiminnallisuuden toteutukselle.

Työ kokoaa sekä ohjelmistotekniikan että audiotekniikan metodiikkaa. Päätulok-sena esitetään menetelmät ja käytännön toteutukset, joita tarvitaan audio-ohjelmistojen automatisoidussa testauksessa.

Avainsanat: audio-ohjelmistot, audiomittaustekniikka, automatisoitu testaus, mustalaatikkotestaus

Acknowledgements

This Master's thesis has been done for Nokia Research Center between the end of year 2003 and the beginning of year 2005.

First, I want to thank the Multimedia Laboratory of Nokia Research Center and especially my superior Samu Kaajas for providing the possibility of doing this work. I thank my colleagues for encouragement and support. Also, thanks belong to Ole Kirkeby for proof-reading the abstract. Many thanks to my academic supervisor Professor Vesa Välimäki for the scientific and linguistic comments. Enormous thanks go to my instructor Jarmo Hiipakka for inspiring discussions, support and valuable comments in both scientific and linguistic issues.

I thank my family and relatives, especially my father Arto Takanen and my mother Raili Takanen for the support throughout the studies.

Finally, and deepest I thank my intended wife Heini Kaistinen for love during weak times on the writing period.

Espoo, February 17, 2005

Marko Takanen

Contents

List of Abbreviations

API	Application Programming Interface
DFT	Discrete Fourier Transform
DTFT	Discrete-Time Fourier Transform
EAP	Entertainment Audio Platform
FR	Frequency Response
IR	Impulse Response
LTI	Linear and Time-Invariant (system)
SRC	Sampling Rate Conversion
SST	Swept-Sine Technique
SUT	System Under Test
THD+N	Total Harmonic Distortion and Noise

List of Notations

A	amplitude
f	frequency (in Hertz)
f_{test}	frequency of a test sinusoid
F_S	sampling frequency
i	general purpose index
J	number of sinusoid periods (integer)
k	Discrete Fourier Transform bin index
K	Discrete Fourier Transform length
L	interpolation factor
M	decimation factor
n	discrete-time sequence index
N	number of samples (general purpose)
t	time
T	sampling period
ω	frequency (in radians)

Chapter 1

Introduction

The interest in audio features of mobile phones has increased in recent years. It was not long ago when ring tones, in addition to speech, were the only sounds heard from a mobile phone. Today, playback of high quality audio is available in the latest models from several manufacturers. However, few have heard, e.g., music played simultaneously with a ring tone. Reason for that might be that the devices have been lacking audio mixing — functionality obviously found in every personal computer (PC). One can foresee similar entertainment audio capabilities that are commonplace in PC environment today, appear in mobile devices in the near future. A candidate that enables such functionalities is Entertainment Audio Platform (EAP) developed at Nokia Research Center.

The purpose of this work is to design and implement automated tests for some of the core functionalities in EAP. The motivation for automated testing comes from the fact that manual testing takes a lot of time. Those familiar with software development know that development usually takes place in multiple phases, i.e., releases, and is iterative in nature. Each release typically has a set of new features and a set of revisited or optimized versions of the old functionality. The new functionality obviously requires new tests. Can we trust that the old functionality also works correctly? Obviously, it must be tested again. Thus, the testing need increases continuously with new functionalities. If testing is automated, the time needed for testing a given functionality during the whole development process is essentially the time needed to implement the tests for it.

Test automation starts to be commonplace in software development. However, it is difficult to find references where it is applied to audio testing. It might be explained with an observation made in this work that audio test automation does not require dedicated methodology. It can be achieved by combining existing methods from both engineering fields.

It should be noted at this point, that the results of this thesis are not in measuring the performance of the EAP platform but in developing the methods required for automated testing of its functionality.

The outline of this thesis is as follows. In Chapter 2, the tested audio processing functionalities of EAP are introduced and the necessary theoretical background required in designing the tests is studied. In addition, a slightly more comprehensive review of the operation and applications of audio mixing, sampling rate conversions, and dynamic range compression is presented. In Chapter 3, the methodology used in software testing is reviewed. Chapter 4 presents existing audio testing methodology and describes how testing automation can be applied in audio software testing. In Chapter 5, the theory of audio signal analysis that is needed in the practical test measurements is studied. In addition to the review of the existing methods, a new technique is proposed for sampling rate converter measurements. Chapter 6 describes author's own work in the test design and implementation of test cases. Chapter 7 concludes the thesis.

Chapter 2

Overview of the system under test

The system under test (SUT) is the Entertainment Audio Platform (EAP), an audio processing software platform developed at Nokia Research Center. The system features a configurable software framework for running various audio signal processing components. The main functionality of the system is interactive mixing of audio with the support for several input sampling rates. In addition, EAP supports various audio effects.

The targeted use case of EAP is a system master mixer as a part of an operating system, providing the system with mixing of audio streams from several applications to a single stream that is fed to the audio device. Applications using the system mixer services are, e.g., games, interactive music players, and web browsers with Synchronized Multimedia Integration Language (SMIL) support. As any software platform, EAP provides its clients with a high-level application programming interface (API). The interface provides an access to audio resources, essentially feeding the audio data to the system and for controlling the processing.

In the following sections, EAP core functionalities and the corresponding algorithms are described. The EAP software architecture is not presented in this thesis and the internal software modules are covered only in detail that is required in test case design. Actually, considering the chosen testing strategy, detailed information of software internals is not even needed. This will be justified in more detail in Chapter 3.

2.1 General concepts

The audio data flow, which feeds the mixers and on the other hand is the output of a mixer, is called an *audio stream.* A stream consists of one or more *audio channel*s. Each channel is able to transfer spatially monophonic sound. As a distinction from Musical Instrument Digital Interface (MIDI) channels, a single audio channel can

contain more than one type of sound. However, once two or more sounds are mixed into a single channel, it is generally not possible to separate them.

The definition of stream includes an assumption that an entire audio file is not loaded in the device memory in any phase but the data is processed as it becomes available. The stream may originate from a local file or another system. In addition, the source that generates the stream or its intermediate data format is not specified. It can be a natural or synthetic source and the data can be in compressed or uncompressed format. The only assumption is that the stream is brought to digital sampled (16-bit pulse code modulated) audio prior to feeding it to the system. EAP is targeted for generic audio processing. In the audio signal processing sense, this means that no assumptions can be made of the characteristics of the signal. Finally, as is the case with any mixing console, EAP is only responsible for processing the audio data and sending it to the audio device, not actually generating it.

2.2 Mixer

Interactive audio mixing is the most essential functionality of the EAP audio engine. The mixer supports several controls that can be applied in real time during the mixing. The logical structure of a stereo mixer is illustrated in Figure 2.1. The input stream of the mixer contains *N* audio channels. Output of a stereo mixer always has two channels. There are three controls available for each channel of an input stream: level, muting and panning. These input stream specific *channel controls* are applied independently of other channels. In addition to the channel controls, there are equivalent *stream controls* for the down-mixed stereo signal.

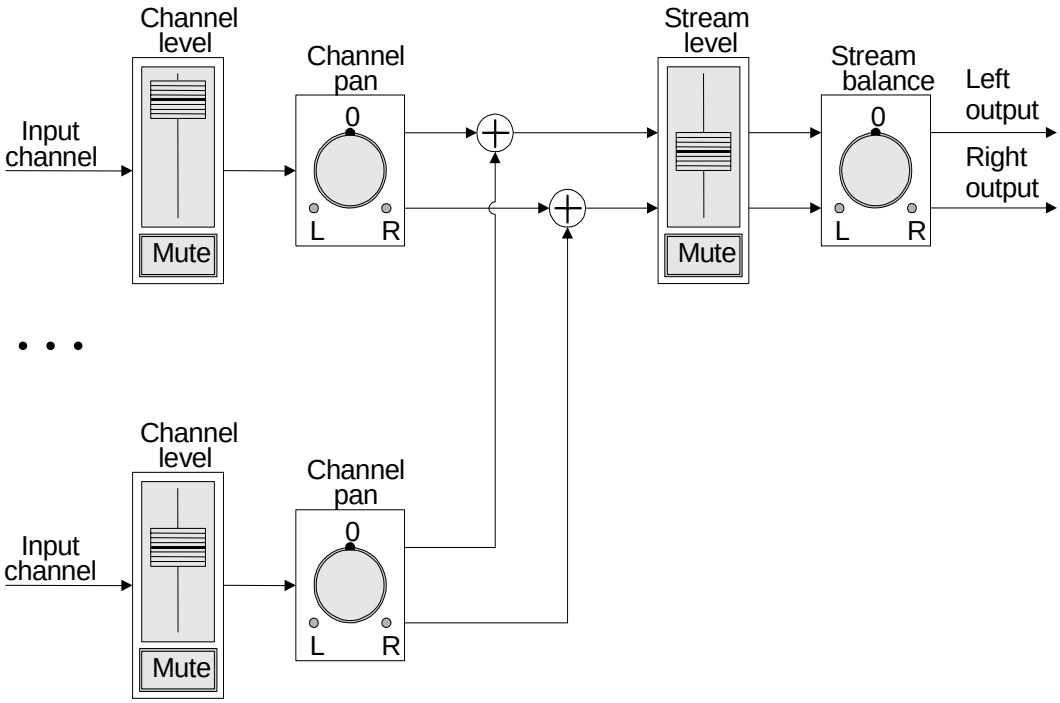


Figure 2.1: Logical structure of an N-input stereo mixer with independent channel controls and separate controls for mixed down stereo signal.

2.2.1 Signal mixing

Mixing two or more signals together is a linear operation. The simplest case, mixing of two signals, can be illustrated with a circuit of Figure 2.2. Input signals x_1 and x_2 are first multiplied by independent gain coefficients g_1 and g_2, and then summed together. In general, the output of the mixer can be written as

$$y[n] = \sum_{i=1}^{N} g_i x_i[n], \tag{2.1}$$

where *N* is the number of input signals. Setting one of the gain coefficients equal to zero, implements muting for the corresponding input signal.

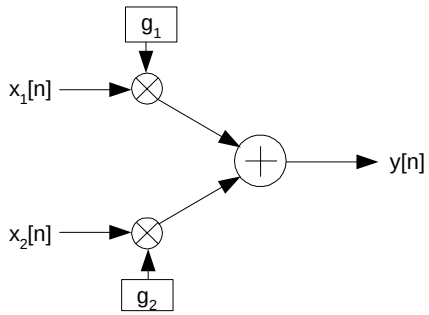


Figure 2.2: Circuit for mixing two signals.

2.2.2 Panning

As monophonic channels are brought to a stereo mixer, they can be distributed into particular positions in the stereo image. Traditionally, this has been done with a simple electronic circuit called panoramic potentiometer or *panpot* (Nisbett, 1979, p. 72). In panning, the signal from a mono channel is divided into the left and right channels of a stereo output. The gain proportion between the left and right channel signals determines the location of the source in the final stereo image. Usually, the user desires that the overall level is kept constant when the image of the source is moved from one position to another. The most standard panning rule for stereo loudspeaker reproduction is the equal power panning law (DLS-2, 1999, p. 22). If the desired panning is P and it is normalized to interval [–1.0, +1.0], where –1.0 represents far left panning and +1.0 far right, equations

$$\begin{aligned} g_L &= \cos\left(\pi(P+1)/4\right) \\ g_R &= \sin\left(\pi(P+1)/4\right) \end{aligned} \tag{2.2}$$

give linear left and right channel gains, respectively. Equal power panning law is also known as trigonometric panning law, as it weights the channel gains with trigonometric functions.

The panning law currently used in EAP is what could be called a “linear panning law.” It is used because it seems to work well for headphone playback. Using the same notation and normalization as in equation (2.2), the linear gains are given by

$$\begin{aligned} g_L &= 1/2 - P/2 \\ g_R &= 1/2 + P/2 \end{aligned} \tag{2.3}$$

Equal power panning law and linear panning law are illustrated in Figure 2.3.

Default panning for a channel is in the center (P = 0). Left and right channel gains of equation (2.3) are equal and thus the source appears in the middle of the stereo image. In case of linear panning law, both gains are 0.5. Consequently, the amplitude of a monophonic signal is halved in panning. This feature has its implications in testing. The two-channel input stream is handled as a special case. To preserve the original stereophonic image, the first channel of a stereo stream is panned to the far left and the second channel to the right.

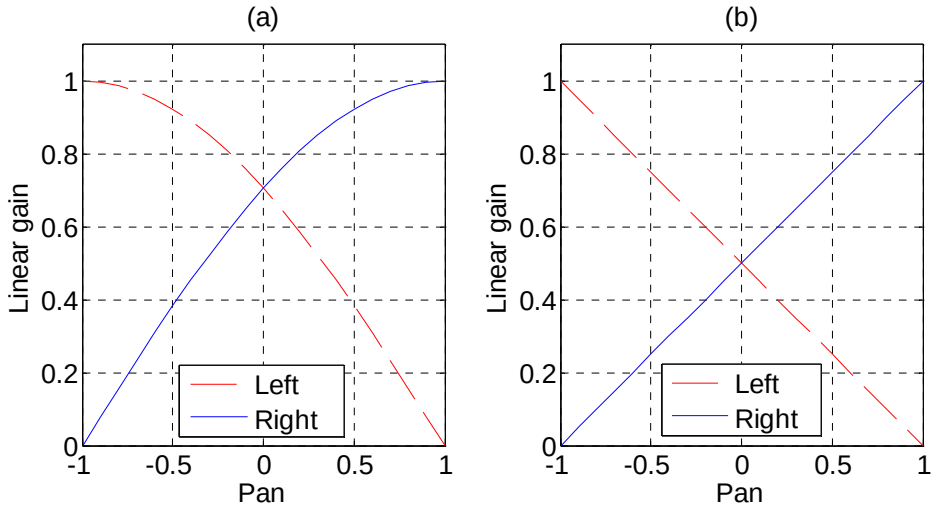


Figure 2.3: a) Equal power panning law and b) linear panning law.

2.2.3 Stream controls

When all channel specific controls have been applied and the two-channel output has been formed, channels are mixed down. For the panning to be retained as desired, left and right channels of each panpot are mixed to the left and right output channels, respectively.

After that, the controls that are specific to the down-mixed stereo signal are applied. The stream controls are otherwise equivalent to the channel controls but instead of panning, a balance control is used for adjusting the level balance between final left and right outputs. The balance curves are slightly different from the ones used in panning. If balance setting is at center, both gains equal one. When it is adjusted towards one of the ends, the gain of the opposite output channel is decreased similarly as in panning but the gain of the other channel is kept constant.

2.2.4 Gain interpolations

All controls except muting are applied with smooth level interpolation, or fade, from the current value to a new value with a desired transition duration. For muting, long fade is usually not desirable, thus only a very short fixed transition is used to prevent discontinuities when the muting setting is changed. To produce a smooth level transition, a gain signal generator is needed. There are three signal generators for each input channel of a mixer. An exponential signal source is used to implement the level interpolations as linear fades on dB scale. For the pan and balance fades, one linear signal generator is needed for both output channels. Together they provide the two gains of equation (2.3) that mix the input channel to the left and right output.

2.2.5 Discussion

What was presented above, and is used in EAP, is traditional stereo mixing. It would be quite straightforward to generalize the functionality to multi-channel mixers that rely on amplitude mixing. Essentially, the only difference is the greater number of

output channels and thus panning and balance controls with higher degrees of freedom. Examples of speaker-systems that take advantage of multi-channel mixing are a 5.1-system and Ambisonic (Leese, 1998).

Another, more recently introduced system designed for reproduction of surround-sound is Vector Base Amplitude Panning (VBAP). It allows an arbitrary placement of virtual source on an active triangle spanned by the three loudspeakers that are nearest to it (Pulkki, 1997). An advantage of VBAP over Ambisonic system is that it more easily suits arbitrary loudspeaker configurations.

2.3 Sampling Rate Conversions

As the input streams for the mixer may originate from various sources, it is reasonable that they may have different sampling rates. Before two streams can be mixed together, they have to have a common sampling rate. EAP supports several input sampling rates. The first family of converters is: 8 to 48 kHz, 16 to 48 kHz, 24 kHz to 48 kHz, and 32 to 48 kHz. As the greatest common factor is eight, they are called the 8-kHz family of converters. In addition, the following conversions are supported: 11.025 to 48 kHz, 22.05 to 48 kHz, and 44.1 to 48 kHz. They are called the 44.1-kHz family of converters as the sampling rate conversion is implemented in several stages where the last stage is always a conversion from 44.1 kHz to 48 kHz.

A procedure that brings sampling rate of a signal to a different rate is called sampling rate conversion (SRC). If the sampling rate is increased, the procedure is called *upsampling*. In case the sampling rate is decreased, it is called *downsampling*. There are several methods available for the conversions. The basic operations are interpolation and decimation (Oppenheim *et al*., 1999). Combining interpolation and decimation is a classical approach for conversions by a non-integer factor. Before reviewing the basic operations and conversion by a rational factor, an analog interpretation of sampling rate conversion is studied. Unless otherwise stated, the theory of operation is after Oppenheim *et al*. (pp. 150-178, 1999).

2.3.1 Analog interpretation

Sampling of continuous-time signal $x_c(t)$ is a periodic operation that results in a sequence of samples

$$x[n] = x_c(nT), \qquad -\infty < n < \infty \tag{2.4}$$

where T is the sampling interval and its reciprocal, $F_S = 1/T$, is the sampling frequency. Signal x[n] is a discrete-time representation of $x_c(t)$, i.e., it exists only for the integer multiples of T. As human beings are not able to receive discrete-time signals, it has to be converted back to continuous-time signal in the end. For the reconstruction of the continuous-time signal to be unique, $x_c(t)$ has to be bandlimited to $\pm F_S/2$, i.e., *Nyquist frequency* before sampling. This condition is a

straight implication of Nyquist-Shannon's sampling theorem and it is assumed from now on.

If one wants to change the sampling rate of *x*[n], essentially the task is to find a new discrete-time representation of the underlying continuous-time signal of the form

$$x'[n] = x_c(nT'), \tag{2.5}$$

where sampling period $T' \neq T$ (Oppenheim *et al*., 1999). The most intuitive approach to obtain $x'[n]$ from $x[n]$ is to reconstruct the continuous-time signal $x_c(t)$ via discrete-to-continuous time (D/C) conversion and then resample it with the new sampling period *T'*, i.e., perform continuous-to-discrete time (C/D) conversion[1]. However, this is impractical for the purpose.

It is not necessary that $x[n]$ be obtained directly by sampling the continuous-time signal. Nyquist-Shannon's sampling theorem guarantees that the underlying continuous time signal $\hat{x}_c(t)$ can be uniquely reconstructed from the discrete time signal via

$$\hat{x}_c = \sum_{n=-\infty}^{\infty} x[n]h(t-nT), \tag{2.6}$$

where

$$h(t) = \mathrm{sinc}(F_S t) = \frac{\sin(\pi F_S t)}{\pi F_S t}. \tag{2.7}$$

Equation (2.6) is a convolution of the input signal and the impulse response of an ideal lowpass filter and this reconstruction operation is called *bandlimited interpolation* (Smith and Gossett, 1984; Smith, 2004a).

2.3.2 Interpolation

In interpolation, the sampling rate of a signal is increased by an integer factor *L*. For example, in case of a conversion from 16 kHz to 48 kHz *L* equals three. If $x[n]$ represents the input signal, the output is given by

$$y[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise.} \end{cases} \tag{2.8}$$

In the analog interpretation, equation (2.8) corresponds to the substitution $T' = T/L$ in equation (2.5), i.e., shortening the sampling interval of $x_c(t)$ to the *L*:th fraction of its original length. This operation is called sampling rate *expansion*, and the system

[1] C/D conversion is otherwise similar to an analog-to-digital conversion (ADC) but in C/D conversion the amplitude of the signal is not quantized.

that performs the operation is referred as an *expander*[2]. In practice, expansion is carried out by inserting $L - 1$ zeros between successive samples in the signal. After expansion, every L:th sample in $y[n]$ is an exact copy of an input sample and the samples in between are plain zeros.

Writing the impulse trail of equation (2.8) in series form, gives

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - kL]. \tag{2.9}$$

The discrete-time Fourier transform[3] (DTFT) of equation (2.9) yields the frequency domain representation of $y[n]$:

$$\begin{aligned} Y(e^{j\omega}) &= \sum_{n=-\infty}^{\infty}\left(\sum_{k=-\infty}^{\infty} x[k]\delta[n - kL]\right)e^{-j\omega n} \\ &= \sum_{k=-\infty}^{\infty} x[k]e^{-j\omega Lk} = X(e^{j\omega L}), \end{aligned} \tag{2.10}$$

which reveals that the spectrum of $y[n]$ has L-fold repetition of the spectrum of the input signal $x[n]$.

Figure 2.4 shows the repetition of the spectrum in frequency domain. Figure 2.4 a), presents the discrete-time Fourier transform of the sequence $x[n]$ on a normalized frequency scale, where $\Omega_N = \pi$ is the Nyquist frequency. Figure 2.4 b) shows $Y(e^{j\omega})$ according to equation (2.10), with $L = 3$. If these spectral images are not treated in any way, they will be audible. To get rid of the images, the output of the expander is filtered with a lowpass filter that ideally meets the following specification for its transfer function

$$H(e^{j\omega}) = \begin{cases} L, & |\omega| \le \pi / L \\ 0, & \text{otherwise.} \end{cases} \tag{2.11}$$

The example of Figure 2.4 requires a lowpass filter with a gain of 3 and cutoff frequency $\omega = \pi/3$ as depicted in Figure 2.4 c). Finally, Figure 2.4 d) shows the scaled and image filtered response.

[2] Not to be confused with the signal dynamics related expander term in section 2.4.
[3] Distinction is made between the discrete-time Fourier transform (DTFT) and the discrete Fourier transform (DFT). In DTFT, only the time variable is discrete; the frequency ω is continuous, while in (DFT) both time and frequency variables are discrete.

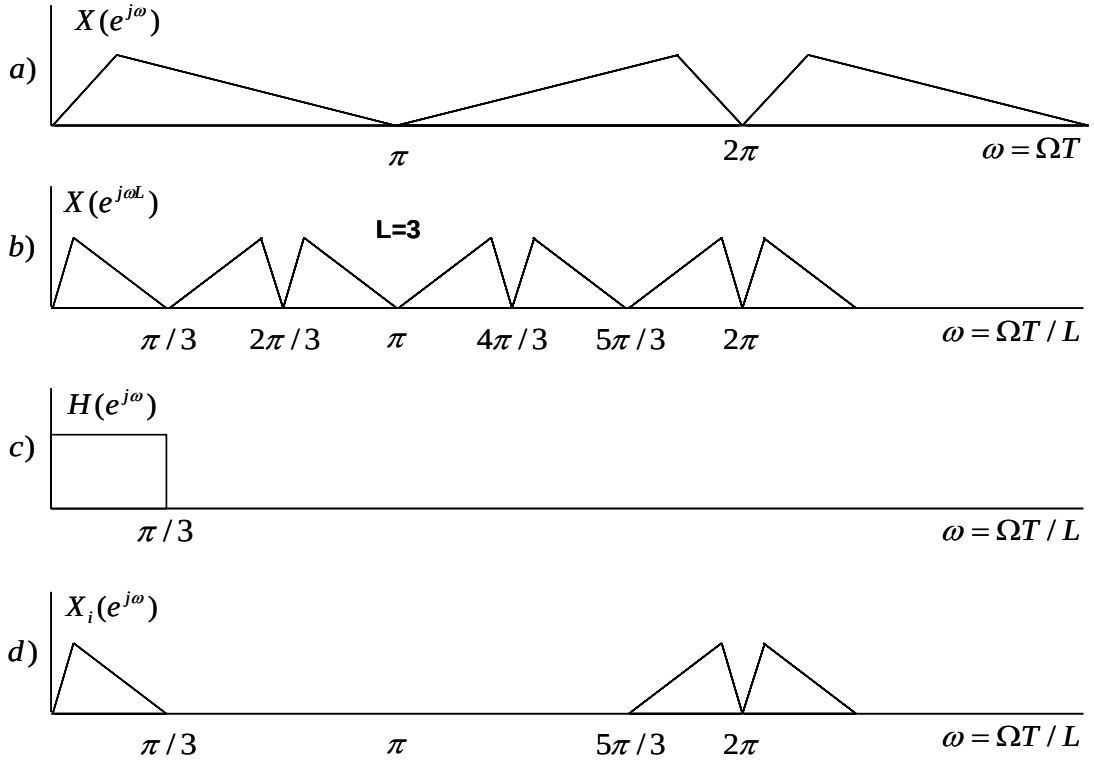


Figure 2.4: Frequency domain illustration of interpolation by a factor of three.

The filter specified by the equation (2.11) is called an *anti-imaging filter*, as it is responsible of removing the spectral images. Another commonly used name is interpolation filter since it interpolates the values for the zero samples that were inserted between the original samples and thus smoothes out the signal. The filter is scaled by factor *L* in order to maintain unity gain in the passband. It compensates for the signal power that was lost in the filtering of spectrum images. A system that implements interpolation is called an *interpolator* and is presented in Figure 2.5.

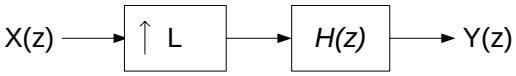


Figure 2.5: Interpolator is a system that increases the sampling rate by L. It consists of a sampling rate expander and an anti-image filter.

2.3.3 Decimation

The opposite operation to sampling rate interpolation is sampling rate *decimation*. In decimation, the sampling rate is lowered by an integer factor *M*. Discarding $M-1$ consecutive samples after every *M* samples in the signal carries out the sampling rate *compression*. For instance, in decimation from 48 kHz to 24 kHz *M* equals two and a

compressor throws out one sample after every second input samples[4]. The operation is analogous with interpolation and therefore its derivation is omitted.

A fundamental difference is that the signal has to be filtered with a lowpass filter that ideally satisfies the following specification for its transfer function

$$H(e^{j\omega}) = \begin{cases} 1, & |\omega| \le \pi / M \\ 0, & \text{otherwise.} \end{cases} \tag{2.12}$$

before the compression. Otherwise, *aliasing* will occur because in compression all frequency components above π / M fold down. If they have energy, aliased components overlap in frequency with the input signal $x[n]$ and it is not possible to recover the original signal from its decimated version. The lowpass filter of equation 2.12 is called *anti-aliasing* filter, as its purpose is to prevent aliasing.

A system that decreases the sampling rate by an integer factor is called a *decimator*. It consists of an anti-alias filter and a sampling rate compressor as depicted in Figure 2.6.

Time-variance of sampling rate conversions

Linearity and time-invariance are important system properties because the signal processing theory of linear and time-invariant (LTI) systems is well established. A system is linear if it satisfies scaling and superposition properties. It is time-invariant (shift-invariant) system, if the following holds true for any time-shift *N*: the time-shifted output of the system is equal to the output of the system for a time-shifted input (Oppenheim *et al.*, 1999, p. 20).

The time-invariance property can be easily tested for the compressor. In Figure 2.7 a), ten samples of an arbitrary signal are plotted. Figure 2.7 b), depicts the output of the compressor when the input is first shifted by one sample and then compressed by a factor of two. It is clearly different from the signal in Figure 2.7 c) that is the output when the input is first compressed by a factor of two and then shifted by one sample. By this counter-example, decimator is not a time-invariant system.

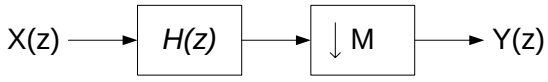


Figure 2.6: System diagram of decimator.

[4] In case $M = 2$, one could say more fluently, “throw out every second sample” but it is dangerous mnemonic as, e.g., in case $M = 4$ it becomes “throw out every third sample” Wrong! Instead, “keep every fourth” is right.

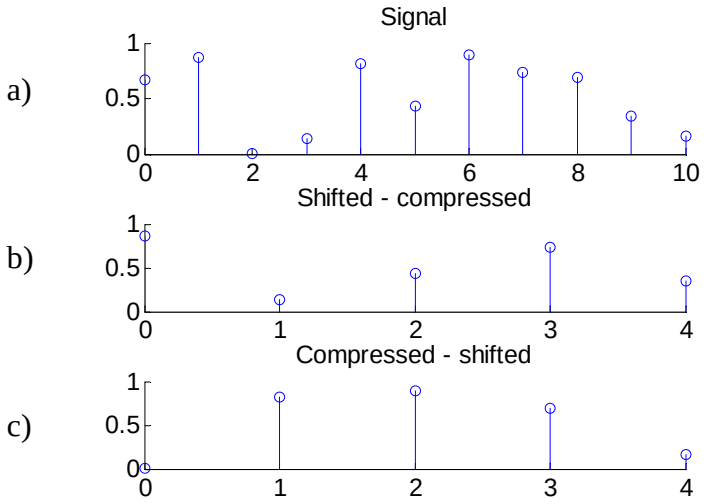


Figure 2.7: Time-variance in a compressor. a) The input signal, b) output when first shifted then compressed c) output when first compressed then shifted.

Strictly speaking, any discrete time operation that produces different number of samples compared to its input is time-variant. Thus, the interpolator is time-variant system, too. This has to be considered when selecting the analysis method for sampling rate converters.

2.3.4 Sampling rate conversion by a rational factor

A classical approach for conversions by a rational factor L / M is to use integer ratio conversions in multiple stages as illustrated in Figure 2.8. The interpolator is cascaded with the decimator. Since both the interpolation and decimation filters operate at the same sampling rate, they can be combined in a single filter. From equations (2.10) and (2.11), it follows that the combined filter $H(e^{j\omega})$ should ideally have the following specification

$$H(e^{j\omega}) = \begin{cases} L, & 0 \le |\omega| \le \min\{\pi / M, \pi / L\} \\ 0, & \text{otherwise.} \end{cases} \tag{2.13}$$

Conversion by the factor 3/2 is illustrated in Figure 2.9 in frequency domain. Figure 2.9 a) presents the periodic magnitude spectrum of the input sequence $x[n]$. Figure 2.9 b) shows the expanded signal $Y_e(e^{j\omega})$ according to equation (2.10), with L = 3. Until now, the spectrums are similar to the ones seen in Figure 2.4 a) and b). If compression by a factor of M = 2 is done at this point, all images are stretched by a factor of two and mirrored around the new Nyquist frequency, i.e., $\omega = 2\pi/3$. The resulting spectrum in Figure 2.9 c) has overlapping images and the signal is severely

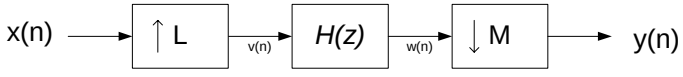


Figure 2.8 Sampling rate conversion by a rational factor L/M

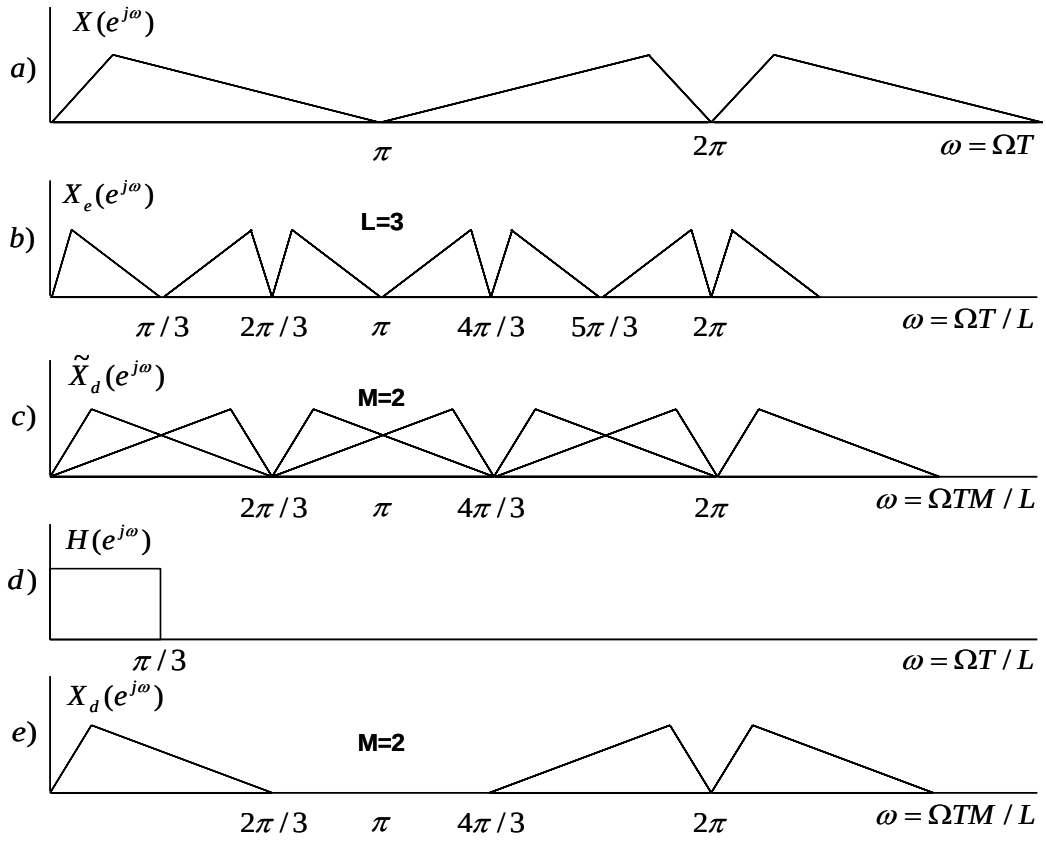


Figure 2.9: (a) – (c) Sampling rate conversion by a rational factor with aliasing. (d) – (e) with prefiltering to avoid the aliasing.

aliased. However, if $Y_e(e^{j\omega})$ is filtered with an ideal anti-aliasing filter that has its cutoff frequency at $\omega = \pi/3$ as illustrated in Figure 2.9 d) and then compressed by a factor of two, aliasing is avoided.

Conceptually, a sampling rate converter that operates by a non-integer factor always contains a decimator. Thus, conversion by a rational factor is not time-invariant. This has implications in testing.

2.3.5 Implementation in EAP

Sampling rate conversions are usually implemented as a polyphase filter structure. Polyphase representation allows computationally efficient implementations of interpolation and decimation filters (Vaidyanathan, 1993, p. 120). The sampling rate conversion algorithm used in EAP is derived from the bandlimited interpolation technique that was originally described by Smith and Gossett (1984) and further explained in Digital Resampling Tutorial (Smith, 2004a). It relies on the analog interpretation of sampling rate conversion presented earlier.

The impulse response (IR) of an interpolation filter is obtained by sampling the ideal low-pass filter, i.e., the sinc function given by equation (2.7). The number of samples used per zero crossing determines the cut-off frequency of the lowpass filter. As the

sinc function ranges from $-\infty$ to ∞ it is shortened by means of windowing. The Kaiser window is used as it results the best stop-band attenuation with given filter length, it has one parameter that is used to trade-off between stop-band attenuation and the transition bandwidth. The resulting IR is convolved with the input signal according to equation (2.6), which carries out the filtering.

If the rate conversion factor is fixed, the sampling rate conversion algorithm reduces to regular polyphase implementation.

In the bandlimited interpolation technique, it is possible to reconstruct the signal in arbitrary continuous times from the discrete-time samples of the input signal. The main difference in the EAP algorithm is that the filter coefficients are pre-calculated for the necessary sampling rates only, rather than interpolated from large array of generic interpolator coefficients. This implies that time-varying resampling, i.e., the possibility to change sampling rate smoothly from one rate to another, is not present. The reason for supporting only selected rates is the strict memory requirements posed to the system.

2.4 Dynamic Range Controller

The dynamic range of natural sounds is huge. The human ear can accept sound pressures in the range $10^{-5} - 10^{2}$ Pa, approximately (Karjalainen, 1999, p. 18). In terms of dynamic range, i.e., the ratio of the maximum peak signal level to the inherent noise level, this corresponds to 130 dB. Since the very first attempts of recording and reproducing of sound, there have been technical challenges to handle such enormous dynamics.

Dynamic range controller (DRC) is a device that reduces or increases the dynamic variations of a signal by monitoring the incoming signal level and applying a time-varying gain to the signal according to predefined rules. Depending on the operation mode of DRC, different names are used. When the dynamic range is reduced, the device is called a *compressor*. If the dynamics are increased, the device is known as an *expander*. Extreme cases of compressor and expander are called *limiter* and *noise gate*, respectively. (Nisbett, 1979, p. 351)

The characteristics of the DRC can be defined with static and dynamic parameters. Static parameters describe the transfer function for steady state signals. Attack and release times are the parameters that control dynamic behavior. Their contribution is essential during signal transients.

2.4.1 Applications

To reproduce sound ideally, the signal level has to be kept between the noise and distortion levels through the entire audio chain from the sound source to the listener at all times. Prior to the digital era, the transmission media was the weakest link. Inexpensive consumer equipment was not able to achieve signal-to-noise ratios

(SNR) higher than 50 dB (Nisbett, 1979, p. 351). On the bottom end of the dynamic range the problem was the high noise level of transmission media, e.g., shellac records, and on the top end the vulnerability of AM transmitter power-amplifiers to excessive signal levels (Nisbett, 1979, p.358). As the signal level of a radio or TV program varies greatly from speech to musical material, human mixer operators were responsible for keeping the variation in signal level of the program in the required range. Because sudden level adjustments within, for instance, a musical piece are perceived annoying, the operator had to prepare for loud and quiet parts in good time before the peaks with a smooth level change. Further problems were caused by unexpected signal peaks in the program, which were totally beyond the control of the operator. The need for an automated system was evident.

Modern day applications and reasoning for the need of a DRC in a mobile device are as follows. In listening situations where the background noise level is high, the useful dynamic range is limited. With a DRC, loudness of a signal can be increased without clipping it. It has artistic purposes, e.g., in vocal and drum sound processing. A known disadvantage of DRC is that excessive compression results in a signal with (musically) dull dynamics. As the dynamics processing is done dynamically in the device, it is possible in ideal listening conditions to turn off the compression. In addition, EAP has several preset settings for the extent of compression for different program material. An important situation also arises after mixing stages, where signal may be wider than 16 bits. Then the limiter functionality is exploited to squeeze the signal to 16-bit dynamics.

2.4.2 Static parameters

The static characteristics of DRC are defined as a relationship between input and output levels on decibel scale (Zölzer 1998):

$$Y[dB] = f(X[dB]). \tag{2.14}$$

The function *f* can, e.g., be a piecewise linear curve. Figure 2.10 depicts the input-output relations for typical operation modes: a) compression, b) expansion, c) limiting, and d) noise gating. Figures are conventionally plotted such that the horizontal axis represents the DRC input signal level (on dB scale) and the vertical axis is the output signal level. The diagonal dash line in the plots represents the linear mode, in which the DRC does not alter the dynamics at all. Each mode in the Figure 2.10 has a section in its curve that has a slope unequal to one. On that part the signal dynamics are altered.

For instance, in case of compression, the dynamic range of the output signal is 10 dB smaller than the dynamic range of the input signal. The parameters that characterize each operation mode are summarized in Table 2.1. It lists the static parameters and their abbreviations as used in Figure 2.10.

The *compression ratio* (CR), as well as the other ratios ER, LR, and NR, are defined by the ratio of the input level change ΔL_I to the output level change ΔL_O as given by

Table 2.1: Static DRC parameters; their abbreviations, value range and parameter values used in Figure 2.10

Abbrev.	Parameter	Value range	Value
CT	Compression Threshold	CR > 1	-20 dB
CR	Compression Ratio	CF ≥ 0	2:1
CF	Compression Factor		10 dB
ET	Expansion Threshold	ER < 1	-30 dB
ER	Expansion Ratio	EF ≥ 0	1:1.5
EF	Expansion Factor		10 dB
LT	Limiting Threshold	LR >> 1	-20 dB
LR	Limiting Ratio		∞
NT	Noise-gating Threshold	0 ≤ NR << 1	-30
NR	Noise-gating Ratio		0

$$R = \frac{\Delta L_I}{\Delta L_O}. \tag{2.15}$$

Conventionally, CR and LR are normalized so that the required input level change is proportional to a one-decibel change in the output level, and thus expressed in the form *x*:1, where *x* is a real number greater than one. For expansion and noise gating, ER and NR are scaled so that the one-decibel change in the input level is related to the change in the output level and written in the form 1:*x* (McNally, 1984). For instance, in Table 2.1, the expansion ratio $ER = 20/30$ is denoted 1:1.5. The limiter and the noise gate employ the extreme values for the compression ratio (LR=∞, NR=0) in the example curves. It should be mentioned that finite and nonzero ratios are used in certain applications.

Threshold values (CT, ET, LT, NT) define the input level that triggers the wanted action. For compressor and limiter, CT and LT define the lower limits of compression and limiting, while ET and NT define the upper limits of expansion and noise gating, respectively. In case of compressor, for example, compression only activates when the input level exceeds the compression threshold (CT). Below CT the signal is not compressed.

Below CT, as seen in Figure 2.10 a), the output level is constantly 10 dB higher than the input level. This constant gain is called the *compression factor* (CF). It tells how much the non-compressed part, i.e., the section of the curve that has 1:1 compression ratio, is amplified. A corresponding gain factor can be recognized in the expander (EF), but the limiter and the noise gate lack it completely.

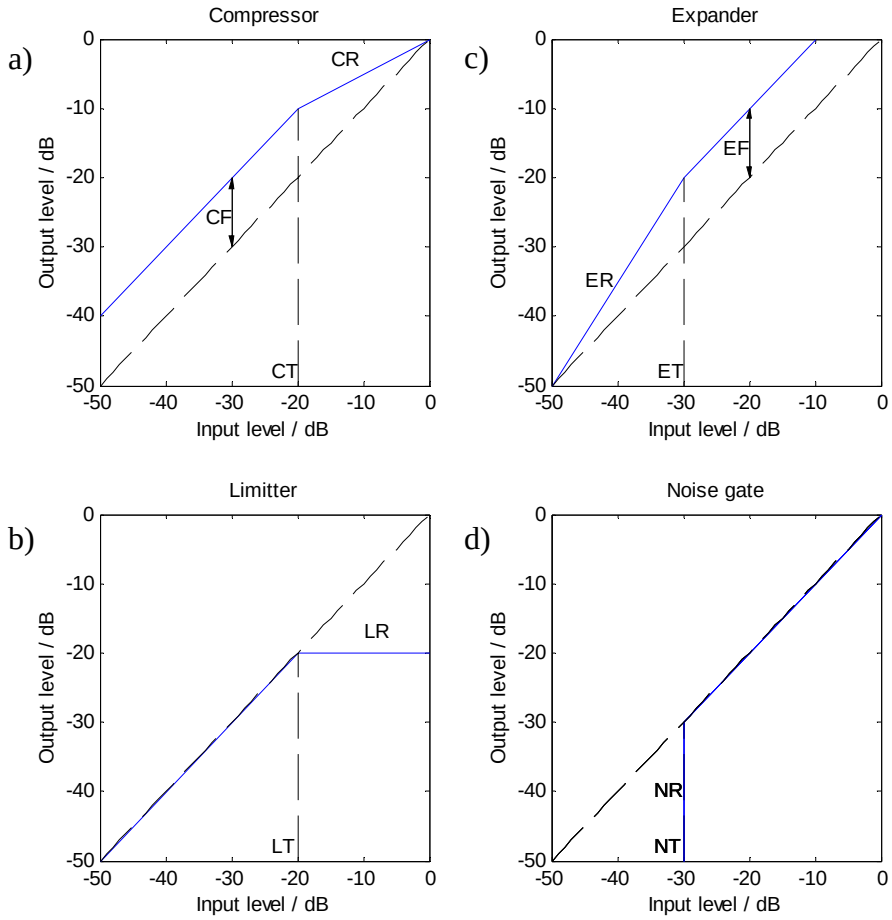


Figure 2.10: DRC operation modes and their essential static parameters.

In EAP, several operation modes can be combined into a single DRC curve. The turning points of this curve can be defined quite freely with the following restrictions:

- The first point along the line is always at (-100, -100) dB.
- The number of user-definable values is confined to five points.
- One of the user-defined points has to be for 0 dB input level.
- After the last user-defined point, the output level is restricted to its maximum given value.

2.4.3 Dynamic Parameters

The *attack time* is defined as the time required for the gain to reach its final value after the input level has exceeded the threshold value. Careful consideration is needed when the value is selected. If the attack time is too long, the gain is not able to react to a level change from the very beginning. With typical input signals, e.g., music and speech, this means emphasis on the beginnings of notes and syllables. In

case that the attack time is too short, the gain compensation becomes too sudden, which means that the system responds to signal peaks rather than energy or loudness. In the worst case, it results in a discontinuity in the signal that is heard as a click. The *release time* affects how fast the output level changes as the input level has decreased below the threshold. (Blesser *et al*., 1968)

2.4.4 Implementation in EAP

The EAP full-band DRC algorithm is depicted in Figure 2.11. Both the input and output signal are stereo. The incoming signal is delayed using a look-ahead delay line that gives the algorithm the necessary possibility to smoothly react to the changes in the incoming signal. After the delay, the signals are multiplied with a common time-varying gain. The gain is calculated from the current level of the incoming signal, taking into account the compression curve. An essential feature of the algorithm is that the most complex part, i.e., applying the compression curve, is done at a lower sampling rate than the audio sampling rate. This gives significant performance benefits, and allows for a fairly complex calculation of the compression. Another computational efficiency aspect in the algorithm that affects testing is that the estimate of incoming signal level is not a true RMS energy. As presented in Figure 2.11, the level estimate is obtained from the absolute values of the current audio samples on both of the channels rather than squared ones.

The DRC presented does not utilize the spectral information of a signal. It alters the amplitudes of all frequency components uniformly. With a multi-band DRC, it is possible to alter the signal dynamics in a more complex way. It divides the full frequency into sub-bands and amplifies each band separately.

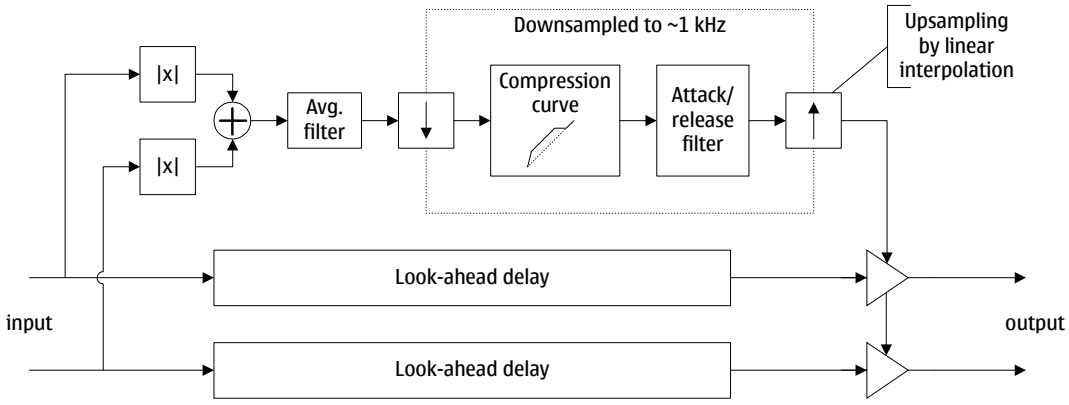


Figure 2.11: Block diagram of EAP full-band DRC.

Chapter 3

Software testing

To begin with, G.J. Myers makes the definition of the word “testing” in software development clear in his pioneering textbook *The Art of Software Testing* (Myers, 1979). He brings out a very essential psychological aspect in software testing. As human beings tend to be very goal oriented, establishing the proper goal has an important psychological influence. Compare the two definitions: “Testing is the process of executing a program with the intent of finding errors,” and “Testing is the process of establishing confidence that a program does what it is supposed to do.” In the former, the target of the tester is to find as many programming errors as possible. She is not satisfied with the results until she has found several bugs from the program under test. On the other hand, if the test designer has adopted the latter definition it is more probable that she chooses tests the program is likely to pass.

It is clear that the tester who adopts the first orientation in test case design will add more value to the program than the tester who is using the latter definition as her guideline. In fact, following the latter definition the tester should prove that the program is error-free, which is theoretically and practically impossible, except in the simplest cases. However, most people are mentally more into making new objects than ripping them apart, which implies that destructive thinking behind the first definition can be very difficult to achieve. The higher the testing abstraction level the more the tester’s goal tends to incline towards the latter orientation.

3.1 Testing as a part of software development process

The development of new software starts with the idea and preliminary study of a problem. Requirement specification is written to formalize the problem and customer’s requirements. In addition to designing the system itself, careful planning is needed to meet economical targets. After planning, follow programming and testing of the system. When the system is ready or the dead line of the project is at

hand, the system is released to the customer. After the release, installation and maintenance of the system is usually needed, too.

In order to avoid chaos, above-mentioned steps are usually carried out according to some software process model. The big-bang model is the least process-oriented but it gives very little control (Patton, 2001, p. 31). The most commonly used software process model is the waterfall model (Patton, 2001, p. 34). Its variation is a spiral model (Patton, 2001, p. 34). The V-model is the most testing oriented software development process (Haikala and Märijärvi, 2002, p. 286). Aspects of V-model are studied in more detail in section 3.3.

The testing process

Several phases can be recognized in the testing process. First comes test planning. Testing of software is a demanding and labor consuming process. In order to achieve effective results testing has to be planned carefully. Other reasons for test planning are to make the testing organized and repeatable, improve the tracking of failed and passed test cases and in certain industries provide means for proving what was tested (Patton, 2001). In test planning decisions are made about what is actually tested, and what is not tested.

Then the tests are designed for the planned functionality. In the test design phase, solutions are found for testing the planned functionalities and verifying the output. After that, test cases are specified. The test case specification contains detailed descriptions of test parameters, e.g., input signals, and the criteria that are used in the verification. When the tests are implemented and the program is ready, the tests are executed. Finally, a test report is produced. It has at least a summary of test run and detailed descriptions of failed test cases. All phases should be documented. The required content of each document is specified in the IEEE standard (IEEE-829, 1998).

3.2 Testing strategies

It is quite easy to convince oneself that complete testing of a program is theoretically impossible, except for the trivial ones. Complete testing would require that the tester executes the program with all possible inputs and with program's internal states are not economically reasonable. One test case would be required for each execution path. There are two main strategies available to the test case selection problem, black-box testing and white-box testing.

3.2.1 Black-box testing

In *black-box testing*, the system is fed a known input x and the system response is y. The tester compares the response to the specified output $f(x)$. If y equals $f(x)$ the test is passed, otherwise a flaw is present in the program. When testing a program on its interfaces, knowledge of the internal structure of the program or programming

solutions is not used. The tester is only interested in finding situations in which the program does not behave according to its specifications. To find these situations, the tester feeds the system with inputs and compares the outputs of the system to the specified correct outputs, hence the name data-driven testing (Myers, 1979, p. 8).

Using this technique, the tester cannot make any assumptions about the program's internals. In theory, the only solution to show that no errors are present in the program is to execute it with every possible input (Myers, 1979, p. 8).

Equivalence partitioning

A method called *equivalence partitioning* can be used to reduce the number of test cases to a reasonable, but effective set (Patton, 2001, p. 68). It provides means for systematically selecting input values that will most probably reveal errors and ignore the redundant ones. The approach is to group similar inputs, similar outputs and similar operation of the software into equivalent partitions. The test cases derived from a given partition, test the same thing or reveal the same bug. Important equivalence partitions can be found from the following input data classes: boundary conditions, sub-boundary conditions, nulls, and invalid data.

Boundary conditions are the situations at the edge of the planned operational limits of the software (Patton, 2001, p. 68). They exist in every program that performs operations on a range of numbers but they are the most obvious to find, as they should be documented in the program's specification. *Sub-boundary conditions* are conditions that take place on the internal boundaries of the program. Although the end-user of the program is not aware of these boundaries, they still need to be checked by the tester. Finding them requires some knowledge of the program's internal operation but not necessarily access to the implementation. For instance, in real-time audio software many algorithms are designed such that they process the audio streams in blocks of samples rather than sample by sample. A sub-boundary condition related to the block size used in the processing can create a situation for a bug.

Once the boundaries have been identified, testing them is straightforward. In addition to testing with the last possible valid input, it is worth testing with a valid input just inside the boundary and with an invalid input just outside the boundary. Boundary condition testing is based on the binary nature of software and on the assumption that if the program behaves correctly on the limit of its operation, it most likely functions correctly on typical input values, too.

Nulls mean situations when no input was passed to the program or when it operates on default values. *Invalid data* situations occur when the program is used with incorrect inputs. Resolving the two latter conditions requires input data checking in the program. In testing, the expected result is that the program is not able to perform the operation on illegal data. Thus, testing invalid data is inherently testing-to-fail. In error situations, program should return an error code. Invalid data should not be able to crash the program.

3.2.2 Other testing strategies

The testing strategy where the tester derives the test cases by studying the program's logic and internal structure is known as *white-box testing*, or logic-driven testing. To assure that no errors are present in the program the tester has to execute every statement in the program with every possible combination of inputs and program's internal states (Myers, 1979, p. 9). White-box testing requires understanding of and thus access to program's implementation.

A testing strategy that utilizes the best aspects of black-box testing and white-box testing is called gray-box testing. It takes advantage of program's specifications and implementation principles (Haikala and Märijärvi, 2001, p. 289).

3.3 The V-model of software testing

According to the V-model, testing is planned and actually done on several levels. Figure 3.1 illustrates how the planning of testing and verification of results is directly linked to the planning and documentation of the system. At the lowest level is module testing, which corresponds to the specifications of software modules. Second level is integration testing that corresponds to the architecture specification. The third level is system testing that corresponds to the functional specification of the system. The highest level of testing is acceptance testing that actually validates the system against to end user requirements.

The V-model strongly emphasizes the importance of testing as testing is planned right from the beginning of the process. Each level of testing can be planned right after the corresponding specification is written (Haikala and Märijärvi 2002, p. 286). In the following, each of the testing levels is briefly described.

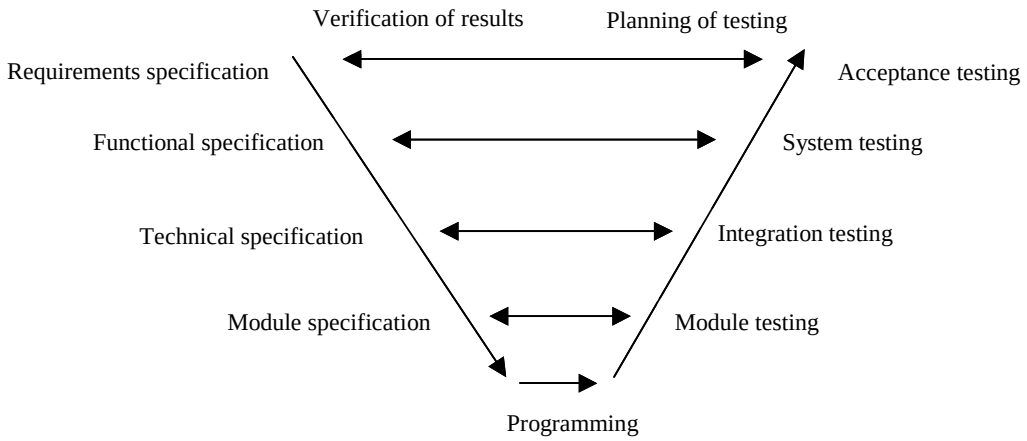


Figure 3.1:The V-model of testing (adapted from Haikala and Märijärvi 2002, p. 287)

3.3.1 Module testing

Testing of a program should be started from its modules. A module is a software component that ideally has the following attributes (Beizer, 1990):

- It is the work of one programmer
- It has a documented specification that includes, in minimum, input definition, functional definition, output definition, and interface definition.
- A module can be compiled, executed, and tested separately from other modules, except for sub-modules it might call.

The tester and the programmer of a module is usually but not necessarily the same person. Module testing should reach 100 % coverage of code. Therefore, white-box testing is used as the testing strategy. Additional code is usually required when testing a module that is intended to interact with other modules. The pieces of software that provide, e.g., the initialization of the module under test are called *test beds* (Haikala and Märijärvi, 2002, p. 289).

3.3.2 Integration testing

Once the individual modules have been thoroughly tested, they are integrated to groups of modules and sub-systems. Integration testing concentrates on testing the interfaces between the modules. Results of testing are verified against the technical specification. Integration testing of modules can be done either constructively (bottom up) during the module integration or structurally (top down) after the integration (Haikala and Märijärvi, 2002, p. 288).

3.3.3 System testing and acceptance testing

In system testing, an entire software system, is tested end-to-end to discover common system bugs, e.g., resource loss, synchronization problems, and shared file conflicts. On that level, the program logic is so complicated that logic-driven testing is not feasible. Therefore, black-box testing strategy is commonly used. The software will be published only after a formal acceptance test on the target hardware. After this phase, the customer accepts or rejects the software.

If module and interface testing are for some reason not done carefully, all kind of bugs are potential on the system level. Essentially, the difference to the lower testing is that testing is done through the same API the end-user of the system has access to. This very interpretation reflects to the title of this thesis.

3.4 Test automation

In test execution automation, or *test automation*, for short, tests are executed without human intervention. Test automation requires a system that is able to read and

interpret the specified test and execute the program according to it. During the program execution, the test system has to be able to receive the data the program outputs. When the program has exited, the system must be able to validate the output and return an unambiguous result whether the test passed or failed.

The first two tasks can be automated easily. The test execution is usually implemented with test drivers that run tests with pre-generated scripts. For output capturing, many tools exist. Often, the program itself is able to write its output to a file. Specifying the test cases for automated execution requires a formal language. The ETSI-standardized language used to write test specifications for automated test is the testing and test control notation (TTCN-3, 2005).

The verification task needed in the test automation is in general not a trivial one. Until recently, a person has always been needed to determine, whether the SUT behaves correctly in a given test case. However, human assessment of software behavior has two main drawbacks: cost and accuracy (Baresi and Young, 2001). Verifying results of hundreds of tests manually not only takes a lot of time but also is very prone to errors. An automated approach has been proposed by Baresi and Young (2001). An automated *test oracle* is some method that is able to check whether the SUT has behaved correctly on a particular execution.

Test oracles are ideally general in the sense that pre-computed input-output pairs or earlier versions of the SUT are not needed (Baresi and Young, 2001). The first property allows rapid increase in test coverage via input parameter variation. The independency of earlier version of SUT is essential when that does not exist. Moreover, the most clever test oracles are also able to derive the test cases (Memon *et al*. 2000). For that a formal specification of program's behavior is needed. It should be noted that the composition task of the state transition diagram or another system model needed for such a formal specification may be a bigger effort than the manual derivation of test cases.

Chapter 4

Audio testing and audio quality

Traditionally, audio testing has been done with a set of measurements that evaluate the performance of an audio device. In previous chapter, testing was defined as the process of executing a program with the intent of finding errors. In this chapter, the meaning is quite the opposite, i.e., the process of establishing confidence that a system does what it is supposed to do.

The difference comes from the fact that traditionally audio systems have been sound reproduction systems whereas software systems have always had tasks that are more complex. A software system either works correctly or does not. The acceptability of an audio system is not so binary in nature. A measure for that exists: audio quality.

4.1 Parametric vs. bit-exact testing

In bit-exact testing, the output of the SUT is compared to a pre-generated reference signal bit-by-bit. The reference signal must be generated with a system that is considered correct. In software technology, such a system is called a reference implementation. The test criterion is same for all tests: the signals have to be identical up to the least significant bit. In order to compare the reference signal to the device output bit-exactly, careful synchronization of signals is required. This requirement implies that bit-exact testing cannot be considered for algorithms that are time-variant, if proper synchronization cannot be guaranteed. Bit-exact testing has been used, e.g., in testing of low-bit rate speech codec implementations (3GPP, 2002).

In *parametric testing*, the test criterion is parametric. Estimates of the parameters under interest are analyzed from the output signal of the SUT. Typical parameters are, e.g., average amplitude level or minimum attenuation required in the filter stopband. As mentioned in section 3.4, the test coverage can be conveniently increased by input parameter variation. In practice, this requires testing be done

parametrically. Otherwise, the number of input-output pairs increases drastically. In this thesis, all designed tests are parametric. In addition, usage of error tolerances is possible. *Error tolerance* is an allowed variation from the required parameter value.

The only advantage of bit-exact testing over parametric testing is that the test signals can be chosen freely. Natural sound, e.g., music can be used as a test signal. This is advantage in case of passive functionality. However, in case of interactive functionality, the test signal selection is only half of the task. Control parameter variation is independent of the test signal selection. Hence, another synchronization issue arises from the timing of control adjustments.

4.2 Audio quality

Sound quality is the common term used for referring quality of sound in general. Depending on the point of view, the specific term varies, e.g., speech quality, sound quality of a concert hall, and noise quality. This section concerns sound quality in audio technology, i.e., *audio quality*. Audio quality can be divided into subjective and objective quality and the methods further into conventional and perceptual audio quality methods. Typically, an audio quality measure tries to describe the performance of the audio device with a single scalar valued parameter.

4.2.1 Conventional objective methods

In ideal sound reproduction, the sound chain is an LTI system and its transfer function is $H(e^{j\omega})=1$, i.e., the system does not modify the sound (Karjalainen, 1999, p. 47). In real world systems, distortions always exist. In conventional objective methods, audio quality is determined using physical signal representations, e.g., frequency and amplitude.

Classical system characteristics measured from the audio system are frequency response function and various distortion measures, e.g., signal to noise ratio (SNR) and total harmonic distortion (THD). A commercial device capable of such measurements in both analog and digital domain is available from Audio Precision (2005).

Conventional audio design does not take advantage of perceptual constraints that could produce computational or transmission savings. The requirements stated for the devices are often perceptually exaggerated. Therefore, conventional methods are considered to measure the quality of the system rather than audio quality.

4.2.2 Perceptual audio quality

The requirements stated for perceptually motivated audio quality take into consideration psychoacoustical phenomena, i.e., what kind of impairment introduced by the SUT can be heard, and what cannot be heard. Conventionally, subjective audio quality has been measured with listening tests. A representative group of

listeners called a *listener panel* is selected. From individual opinions of each listener, audio quality can be measured using statistical methods (Karjalainen, 1999, p. 200).

Formal listening tests are laborious, expensive and not suitable for continuous monitoring of audio quality (BS-1387, 1999). Relevant objective quality measures can be obtained with human hearing models. They model the sensitivity to, e.g., level difference, frequency difference, and distortions. In addition, different perceptual constraints, such as the frequency masking phenomenon, are considered. Beerends *et al*. (1992) have proposed a method that is able to estimate the masking effect. ITU-R standard specifies a method for objective measurements of perceived audio quality (BS-1387, 1999). Both methods have applications on evaluation of low-bit rate audio codecs.

Common to all methods is that, the physical signal representations, e.g., frequency and amplitude of the input and SUT output signals are mapped onto a psychoacoustical representation, e.g., pitch and loudness. The mappings permit calculating the perceptual degradation introduced by the SUT, from which the perceptual audio quality can be predicted to a certain extent.

Figure 4.1 depicts the general concept of an objective audio quality measurement. Test signal is fed into SUT. The signal captured from the SUT output is called a *signal under test*. Using a perceptual model, the objective measurement method calculates an audio quality estimate from the test signal and signal under test. It estimates the extent of perceptual degradation the system under test causes to that test signal.

Perceptual models are out of the scope of the practical work presented in this thesis. The A-weighting described below is the only effort towards psychoacoustically motivated testing.

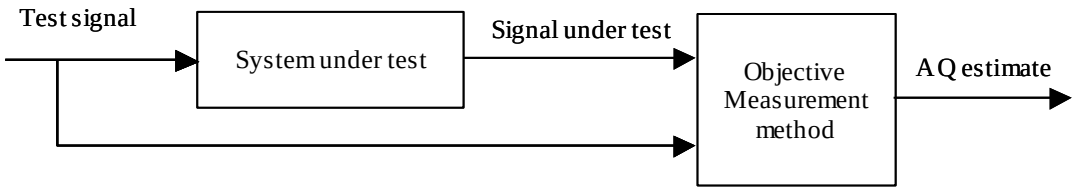


*Figure 4.1 General concept for perceptual audio quality measurements. (*Adapted from *BS 1387-1, 2000)*

4.2.3 A-weighting

The human hearing is less sensitive at low and high frequencies than in the upper midrange, and the perceived loudness level as a function of frequency depends on the sound pressure level (SPL). Fletcher and Munson have determined this dependency in equal loudness curves (Rossing, 1990, p. 92). The A-weighting tries to combine these curves into a single filter. It has been criticized for oversimplifying the hearing, as it totally ignores the loudness level's dependency of the sound intensity (Elliot,

2003). The approximation is closer to the true loudness level at low levels. In this thesis, A-weighting is applied to the THD+N measurements of the rate conversions. There, it is justified as the weighted signal mainly consist of low-level noise.

The A-weighting function is defined in an IEC standard and given by:

$$A(f) = 20\log\frac{f_4^{\,2} f^{\,4}}{\left(f^{\,2} + f_1^{\,2}\right)\sqrt{\left(f^{\,2} + f_2^{\,2}\right)\left(f^{\,2} + f_3^{\,2}\right)}\left(f^{\,2} + f_4^{\,2}\right)} - A_{1000}, \quad (4.1)$$

where A_{1000} is a normalization constant, in decibels, that scales the filter gain to 0 dB at 1 kHz (IEC 61672-1, 2003). Approximate values for frequencies f_1 to f_4 in equation (5.10) are 20,60 Hz, 107,7 Hz, 737,9 Hz, and 12,194 kHz, respectively. Figure 4.2 illustrates the magnitude spectrum of the A-weighting filter on audio band.

4.3 Automated audio testing

Few publications on automated audio testing exist. The oldest reference is probably an article by Roberts (1968), in which he describes an automated system that is capable of doing several performance tests for audio amplifiers. The test system was used in production line testing and it might be operated by "non-technical personnel". For that, the system was able to report the test result with "go" and "no go" lights. Cabot has proposed an automated measurement procedure for loudspeaker small signal parameter calculation (1986) and an automated measurement technique for DRC dynamic parameters (1987). In either article, Cabot does not describe the

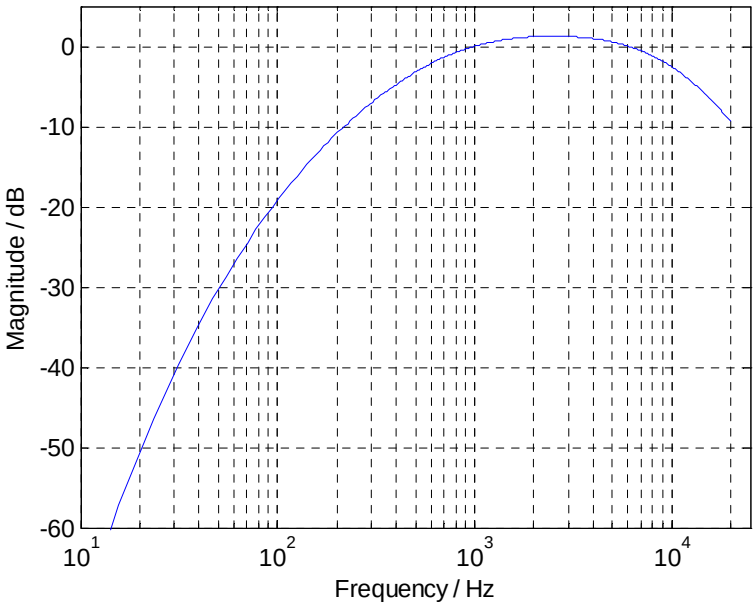


Figure 4.2 The magnitude spectrum of A-weighting filter.

validation task needed in testing. In a more recent study, Groeper *et al.* (1991) have proposed an automated FFT-based distortion measurement method for loudspeaker cones. In their approach, measured parameter values are validated against the values of respective parameters measured from a known good reference unit.

Parametric automated audio testing of a single functionality consists of three phases: parameter selection, parameter extraction, and verification. The parameter selection phase includes test parameter variation. In interface testing, parameter selection can by done by means of equivalence partitioning (discussed in section 3.2.1). Parameter extraction is done by means of audio signal analysis that is the topic of the next chapter. For verification of parameter values, the approach used in this thesis is described below.

4.3.1 Test oracles

Automated test oracles were introduced in section 3.4. The automated test oracle in audio testing is a method that is able to determine the correctness of the signal under test in terms of test parameters. A schematic presentation of the oracle is depicted in Figure 4.3. Oracle extracts a specific signal characteristic from the signal under test with methods of audio signal analysis. In order to verify the parameter values, it receives the test criterion. In case of simplified oracle, it consists of the expected parameter values and error tolerance. In some cases, it is worth implementing a clever oracle that is able to derive the expected parameter values itself. This requires a model of the (sub-) system under test that is able to compute the expected parameter values from the test signal.

In practice, test oracles are not generic in the sense that one oracle was able to verify all functionalities of the SUT. Instead, designated oracles are used for most of the functionality.

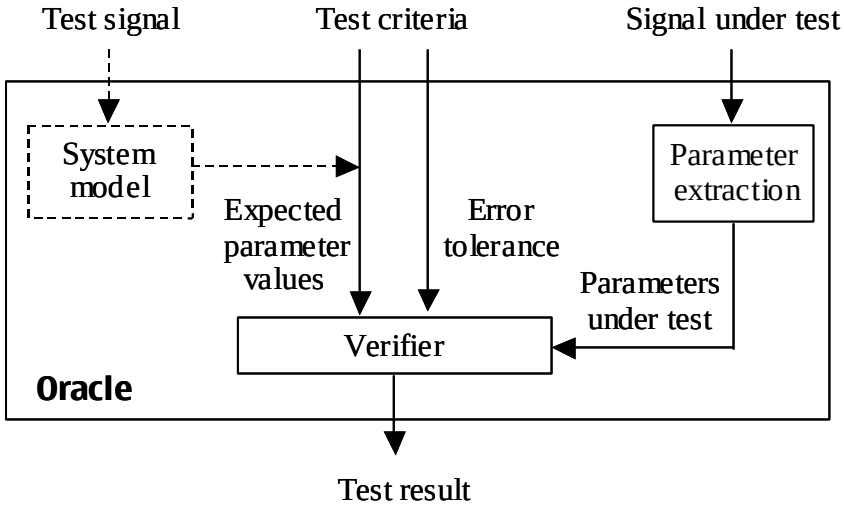


Figure 4.3: Test oracle in audio software testing (adapted from Memon et al. 2000).

4.4 DUT framework

To give an overall picture of the EAP testing, a high-level description of the test framework is given. The test framework used in EAP testing is called DUT. It automates the test execution. In the following, the test instruments are first shortly described and then the operation flow in test execution is discussed.

4.4.1 DUT instruments

As EAP is targeted for system service rather than stand-alone mixer application, it is implemented as dynamically linked libraries (DLL) and therefore does not have any executable front-end or user interface. Stribodus is a program that offers the front-end and a script based user interface for the tester to execute any of the EAP API level functionalities. In a Stribodus script, the EAP control commands that are needed in the test are defined. Essentially, the script interface makes testing easier and less prone to errors as the tester is able to execute versatile tasks without writing a designated program for each scenario.

In order to allow flexible parameter variation, Stribodus control scripts are generated dynamically. As a basis for generation, is a selection of parametrical script templates needed in the tests. The script generator assigns the values for the parameters specified in the test case.

Test oracles needed in the verification are implemented in Matlab®. Matlab is well established among audio developers as a simulation environment and it has extensive selection of built-in tools for signal processing. Therefore it is a natural choice for an analysis environment. Parameters always include at least the filename of the signal under test and the test criteria. Estimates of the parameters under test are computed and they are compared against the test criteria. Each analysis script is responsible of returning the test result (pass/failed) and a detailed report of the analysis. The signal analysis needed in the tests is described in Chapter 6. Appendix B gives an example implementation.

4.4.2 Operation flow in test execution

Figure 4.3 depicts the most essential DUT components. The test controller is the main module of the DUT framework. It controls the test instruments and test execution. In addition to the instruments, all necessary data needed in the tests must be available. Essentially, test configuration contain the test case specifications, test signals stored in pre-generated files, and Stribodus templates in their own database.

® Matlab is a registered trademark of Mathworks Inc. URL: http://www.mathworks.com/products/matlab/

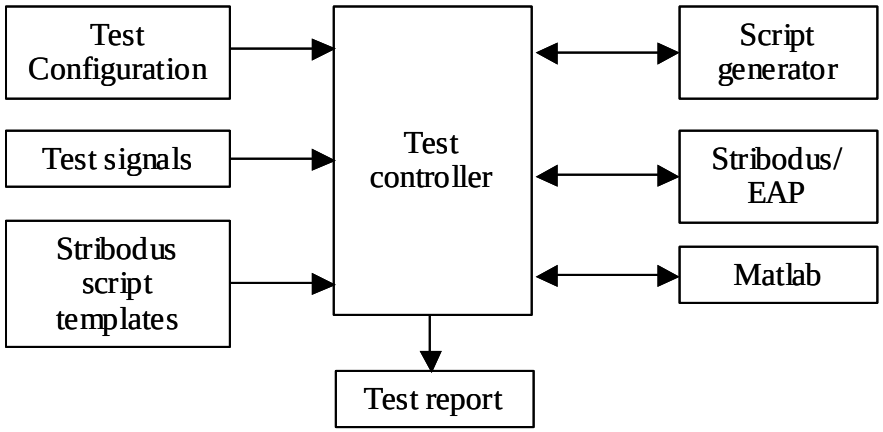


Figure 4.4:DUT test framework.

The operation flow in testing can be described as follows.

1. Read test configuration data.
2. Generate a Stribodus script.
3. Process input files with EAP and apply the controls defined in the script.
4. Copy test parameters to the Matlab workspace.
5. Compute estimates of the test parameters in Matlab and verify parameters against the test criteria.
6. Report test result.
7. Loop through all test cases, write test report and send result to developers.

Chapter 5

Audio signal analysis for testing

Audio signal analysis is required in various audio processing tasks. In feature analysis of musical sounds, e.g., pitch can be extracted. Physics based sound synthesis requires analysis of natural sounds in order to model the sound source. Audio coding exploits perceptual constraints in hearing. Signal analysis is needed to find the perceptually redundant information in audio signal. Automated audio testing requires signal analysis in extracting various parameters from the output signal of the SUT. This chapter presents audio signal analysis techniques that are needed in the measurements.

Measurements can be divided into time-domain measurements and frequency-domain measurements. Theory and definitions of time-domain measurements are examined first. Before going to the details of frequency domain measurements practical limitations of the discrete Fourier transform (DFT) are studied. The theory is presented to the extent it is needed in practical measurements.

5.1 Level measurements

The most basic audio signal analysis can be done on the time-domain representation of a signal, i.e., by inspecting the signal waveform. For simple signals, the waveform is a useful representation to get an idea of temporal features of the signal. If the signal is more complex, the pure waveform data is quite obscure as such. Useful parameters that characterize the signal in the time-domain are signal peak level, energy level, and amplitude envelope.

5.1.1 Amplitude level

Amplitude of a signal is a parameter that tells the signal strength. It can be measured at a certain point of time that is referred as *instantaneous amplitude* or it can be the *average amplitude* over a longer period.

Instantaneous amplitude is a useful parameter for simple periodic signals. In case of non-periodic signals, instantaneous amplitude is less informative, as it changes arbitrarily over time. In practical measurements, the average amplitude is often a more useful parameter. It averages absolute values of a signal over an *N*-length window:

$$A_{avg} = \frac{1}{N}\sum_{n=1}^{N}\left|x[n]\right|. \tag{5.1}$$

If expressed on a linear scale, the amplitude is cumbersome. A well-established quantity in signal processing is *level*. Being measured on logarithmic scale, level is more manageable and corresponds better to human perception. To obtain amplitude level, amplitude *A* is expressed in decibels

$$L = 20\log_{10}\left(\frac{A}{A_{ref}}\right), \tag{5.2}$$

where A_{ref} is a reference amplitude. Decibel is ratio of two numbers; it always requires a reference value. In digital domain, a standard reference value is *full-scale amplitude*, which is the amplitude of a 997-Hz sine wave whose positive peak value reaches the positive digital full scale (AES-17, 1998, p. 5). With signed two's complement integers the full-scale amplitude is $2^{15}-1=32767$.

Expressing the signal amplitude relative to the full-scale amplitude defines *decibels, full-scale* (dB FS) (AES-17, 1998, p. 5). With the fixed reference value, level becomes an absolute quantity similarly to, say, sound pressure level. Although rarely seen in other audio applications but in audio measurements, dB FS is a convenient unit when expressing digital signal levels. The amplitude level of a signal is an essential parameter when verifying mixing. In addition, it is very useful in measurements of other features, e.g., DRC static parameters.

5.1.2 Energy level

Signal energy can be defined in several ways. Mathematically the most correct estimate of signal energy is a root-mean-square (RMS) energy value. It is computed over some time window by first squaring the signal, then averaging it over the frame and finally taking square root, as given by:

$$E = \sqrt{\frac{1}{N}\sum_{n=1}^{N}x^2[n]}, \tag{5.3}$$

where *N* is the length of the time frame and x[n] is the input signal (Cabot, 1999). To obtain the energy level, RMS energy is expressed in decibels with equation (5.2), full-scale amplitude as the reference.

5.1.3 Amplitude envelope

Amplitude envelope is a temporal representation of signal amplitude. In envelope extraction, the signal is smoothed out such that only the overall variation of the original signal is left. Amplitude envelope is a useful feature when verifying mixer gain ramps. There are several techniques available for computing the amplitude envelope of a signal. A traditional DSP oriented approach is to use a simple circuit that consists of a half-wave rectifier and a leaky integrator. Leaky integrator is a low-pass filter that is implemented with a first order IIR filter having its feedback coefficient slightly less than one (Smith, 2005). Being computationally very efficient, the technique suits best real-time extraction of the envelope. It yields an average envelope rather than peak envelope.

Another technique that is better suited to off-line extraction of an amplitude envelope is to obtain an intermediate signal representation called *analytic signal*, from which amplitude envelope is readily available. Analytic signal is a complex-valued signal that lacks negative frequency components (Oppenheim *et al.*, 1999, pp. 775–810). It can be obtained via a Hilbert transform that uses filtering techniques to remove the negative frequencies.

A pure transformation based approach has been proposed by Marple (1998). It assumes that the signal to be analyzed is finite-length and real-valued. The conditions are obviously met for computer stored audio signals. The procedure begins with the computation of an *N*-point discrete Fourier transform (DFT) (see also section 5.2) of the signal under investigation. Then the cancellation of negative frequencies is done by manipulating the frequency bins. Finally, an *N*-point inverse DFT is computed yielding a complex discrete-time analytic signal. Details of the operation are presented in Marple's article (Marple, 1998).

The resulting complex analytic signal is of the form

$$z[\mathrm{n}] = x[\mathrm{n}] + jy[\mathrm{n}]. \tag{5.4}$$

Using the polar form of complex signals, i.e., writing $z[\mathrm{n}]$ in terms of magnitude $A[\mathrm{n}]$ and phase $\varphi[\mathrm{n}]$, the analytic signal can be expressed as

$$z[\mathrm{n}] = A[\mathrm{n}]e^{j\varphi[n]}, \tag{5.5}$$

where $A[\mathrm{n}]$ is the amplitude envelope of the original signal (Smith, 2004b).

A few problems are expected in practical analysis. The transformation introduces the Gibbs phenomenon if the signal contains radical discontinuities (Oppenheim *et al.* 1999, p. 468). They tend to provoke ringing effects on the envelope corners. However, in testing the problems introduced by discontinuities can be prevented by careful selection of test signals. At low amplitude levels, the finite word-length effects become dominating; they are seen as a ripple in the envelope. This is a more severe problem as the ripple interferes with the measurement, and cannot easily be

avoided in practical tests. Using relaxed error tolerances below a certain signal level is required.

Although the technique presented is theoretically complicated, the whole procedure can be implemented very conveniently in Matlab using a couple of built-in functions. In addition, the transformation method was found to give accurate estimations of amplitude envelopes in this application.

5.2 On the analysis of periodic signals using the Discrete Fourier Transform

The discrete Fourier Transform (DFT) is an essential tool in several digital signal-processing applications, including filtering and spectral analysis. When performing any spectral analysis with DFT, it is necessary to understand its operation, capabilities, and limitations. Measurements that require high precision in the frequency domain are impossible to perform without this understanding. Such measurements in this thesis are, e.g., the accuracy of SRC and the distortion measurements, in which the test signal cannot be separated from distortion components if the signal energy is spread on nearby frequency bins. Hofbauer (2004) has presented an extensive survey of methods for high accuracy measurement of sinusoids in harmonic signals.

5.2.1 Spectral accuracy and spectral leakage

To obtain accurate measurements on periodic signals with the DFT, care must be taken. This is due to the discrete nature of the DFT, which will only give an exact answer if there are an integer number of signal periods under the window of interest. The discrete Fourier transform actually finds the frequency components of a signal, which consists of endlessly repeated copies of the sampled signal. In other words, DFT assumes that the waveform being analyzed is periodic with a period that is equal to the length of the data record seen by the DFT (Cabot, 1999). If the frequency of the sinusoid being analyzed does not exactly coincide with one of the available frequency bins, some of the energy is spread between nearby bins. This also implicates that the obtained frequency, amplitude, and phase estimates are never accurate. The terms used of this phenomenon vary in the literature. Oppenheim *et al.* (1999, p. 703) refers to it with the term *spectral sampling*. A more commonly used and more felicitous term is *spectral leakage*.

Figure 5.1 illustrates a usual situation when DFT is used in frequency analysis without knowledge of its operation. Signal under investigation is a sinusoid with a period of 200 samples. In order to compute a 256-point DFT, a 256 samples long data record is chosen arbitrarily from the signal. Once one has fixed the length of the DFT to 256 samples, it is assumed that the period of the signal is 256 samples long. The DFT interprets the sampled signal to repeat endlessly but the beginning and the end of the record do not meet with the same value and slope (refer Figure 5.1 b)).

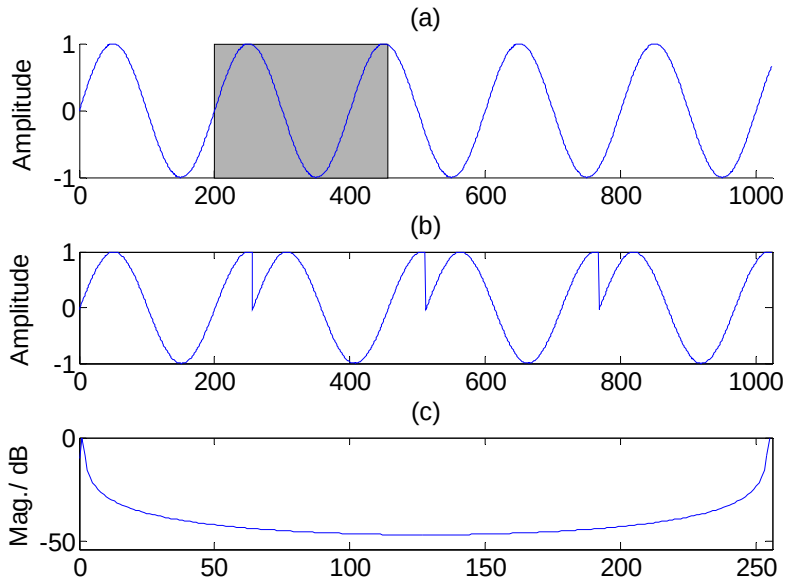


Figure 5.1: A careless computation of DFT of a periodic signal. a) A sinusoid that has a period of 200 samples. The gray rectangle represents a 256-point DFT window. b) The periodic signal as seen by the DFT. b) The magnitude of DFT that is influenced with the spectral leakage.

The discontinuity will result in spectral leakage in the magnitude spectrum. This is seen in Figure 5.1 c). In case of a single sinusoid, the magnitude of DFT should consist of two peaks. Now, the energy of the input signal is spread over all frequencies, which disturbs the measurement by possibly hiding the nearby frequency components. Another observation is that the signal-to-noise ratio is less than 50 dB. In an ideal measurement, over 300 dB SNR can be observed. In that case, the noise only constitutes of the rounding errors in the DFT computation with the Matlab precision. Refer, e.g., magnitude spectrum of Figure 5.2 for such high precision measurement.

The spectral leakage can be avoided by understanding the periodic nature of the DFT. In testing circumstances, if the test signal can be composed of one or more sinusoids whose frequencies are in integer relationships, it is possible to choose the frequencies such that they coincide with the DFT bins. This possibility is heavily exploited in a DFT-technique called *coherent sampling* that is studied in more detail in section 5.2.2.

In practice, situations that are more common are those when the length of the signal period and DFT length cannot be synchronized in the analysis. Another situation is analysis of non-periodic signals. Fortunately, there are a couple of techniques available for those situations, too. They are *windowing* and *zero-padding*. They do not yield as accurate results as coherent sampling but a significant improvement can be achieved compared to direct application of DFT.

5.2.2 Coherent sampling

In coherent sampling, the frequency of the test signal is chosen such that it coincides with one of the DFT bins in the analysis. Let *K* be the DFT window length and *J* the number of sinusoid periods that fits exactly in *K* samples. Coherent sampling occurs when the following relationship is met for integer *K* and *J*

$$\frac{f_{test}}{f_S} = \frac{J}{K}, \tag{5.6}$$

where f_{test} is the frequency of the test sine at f_S output sampling frequency (IEEE-1241, 2001). When proper ratios are used, the DFT yields perfect results in terms of frequency and amplitude accuracy with the absence of spectral leakage. To find the test frequencies, the sampling frequency and the DFT length are fixed first. The test frequency can then be selected letting *J* be any integer. The upper limit is $J = K / 2$ that gives the Nyquist frequency.

Despite the coherence relationship of equation (5.6) will work for any integer *J*, it has been reported that certain values provide better results (IEEE-1241, 2001). The frequency of a sine wave is optimum if there are *K* distinct phases in the DFT data record that are uniformly distributed between 0 and 2π radians. Those frequencies are found from equation (5.6), when *J* is chosen such that it is mutually prime with *K*. Although originally used in the context of Analog-to-Digital Converter (ADC) measurements, the criterion is relevant in the measurements of sampling rate converters, too.

Figure 5.2 depicts a coherently sampled DFT. As the length of the DFT is now exactly the same as the length of two periods in sine, the DFT window repeats without discontinuities. The resulting SNR is only limited by computational accuracy. The position of the DFT window does not affect the repetition of the window. If information about the phase response of the system is not needed, no synchronization is required in measurement.

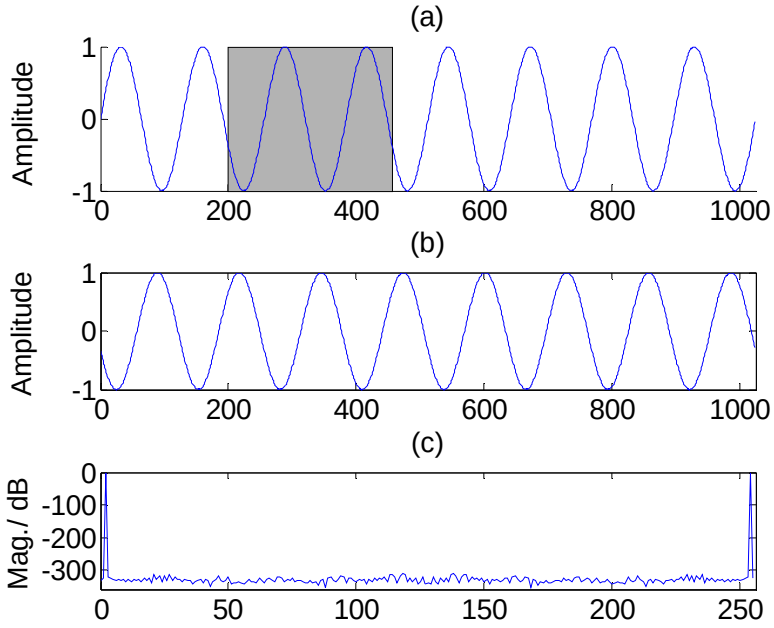


Figure 5.2: Coherent sampling of a periodic waveform. In a), sinusoid that has a period of 128 samples. The gray rectangle represents the 256-point DFT window. In b), the periodic signal as seen by DFT. In c), the obtained magnitude of DFT with great spectral resolution.

5.2.3 Windowing and zero-padding

For situations when the conditions of coherent sampling cannot be met, there are windowing and zero-padding. As they are techniques that are more common in digital signal processing than the coherent sampling, their properties are only reviewed shortly.

Windowing

In windowing, smooth fade-in and fade-out are applied to the signal frame to remove the discontinuities (that were seen in Figure 5.1) at frame ends. In practice, windowing is carried out by multiplying the frame by the mathematical expression that defines the window. There is a wide selection of window functions available. The frequency characteristic of a window is a continuous spectrum consisting of a main lobe and several side lobes. The width of the main-lobe relates to the frequency and amplitude precision of a window, while the amplitudes of the side-lobes are proportional to the amount of frequency leakage (Oppenheim *et al.*, 1999, p. 701). By default, DFT uses rectangular window function. It has the narrowest main-lobe width for the given window length but, on the other hand, the worst side-lobe attenuation of all the commonly used windows (Oppenheim *et al.*, 1999, p. 701). From windowing point of view, the accuracy and leaklessnes of coherent sampling can be explained with the fact that only the main lobe of the rectangular window is effective; the side lobes of the window are totally suppressed at bin frequencies (Dallas, 2002).

The most important windowing function in EAP testing is the Kaiser window. It has two parameters, β and *N*, which can be used to trade between main-lobe width and relative side-lobe amplitude (Oppenheim *et al*., 1999, p. 701). The length of the window is defined by *N* and the shape of the window by β. Other windows that are suitable for spectrum analysis have been extensively studied by Harris (1978).

Window selection is always a trade-off between width of the central lobe and suppression of side lobes. A disadvantage of windowing is that it reduces the ability to resolve signal components that are closely spaced in frequency (Oppenheim *et al*., 1999, p. 701).

Zero-padding

Alternatively, the signal frame can be extended with a sequence of zeros, which is known as *zero-padding*. DFT is then computed over the extended sequence. Padding a signal with sequence of zeros makes the frequency spacing finer because it changes the spacing of the frequency bins. Thus, with padding one is able to obtain a more accurate estimation of the frequency of a single sinusoid. However, it does not affect the frequency resolution, i.e., ability to resolve two sinusoidal components that are closely situated in frequency (Oppenheim *et al*., 1999, p. 711). Padding only interpolates the already sampled spectrum. If there were two sinusoidal components between the two frequency bins in a spectrum obtained without zero-padding, padding would not reveal them either. In order to increase the frequency resolution, the only solution is to increase the length of the signal frame.

Zero-padding has a practical application in time critical systems as it can be used to speed up DFT computation. Even if the signal is made longer with added zeros, the computation is quickened. This is due to the nature of many Fast Fourier Transform (FFT) algorithms that require the length of the sequence to be a power of two. Zero-padding may also be necessary when magnitude spectra from two different signals are needed with the same resolution. This is the case, e.g., when comparing a system's output to a reference spectrum; a case often encountered in testing.

Zero-padding is by no means a general solution to the spectral leaking problem but when used in conjunction with windowing, the accuracy of DFT can be improved with controlled spectral leakage.

5.3 Existing methods for frequency response function measurements

The frequency response (FR), function or its time-domain counterpart impulse response (IR) function, is the most important characteristic feature of a system (Lahti, 1995, p. 91). The frequency response function is a complex function given by

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})}, \tag{5.7}$$

where $Y(e^{j\omega})$ is the DTFT of the output and $X(e^{j\omega})$ is the DTFT of the input. If the impulse response function $h[n]$ of the system is known, the frequency response function is obtained with the discrete-time Fourier transform

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n] e^{-j\omega n}. \tag{5.8}$$

There are many different approaches to identify the FR function. In case the identification of the frequency response function of a system is done by measurements, methods mainly vary by the signals that are used to excite the system. Common to all of them is that excitation is a wide-band signal, i.e., it contains all frequencies on the frequency band under interest. Moreover, in all methods, the task is to find an inverse function $X^{-1}(e^{j\omega})$ that is needed to deconvolve the input signal from the output, leaving the frequency response function. The inverse function exists for all excitation signals, but in general, finding one is not a trivial task.

Frequency response function measurements are needed in the evaluation of anti-aliasing filters in sampling rate conversions. When selecting a measurement method for the purpose, it must be kept in mind that time-invariance cannot be assumed, as was shown in section 2.3.3. In the following, conventional techniques for impulse response measurements are first shortly reviewed. Then a more recent swept-sine technique is studied.

5.3.1 Conventional methods

The most prominent conventional methods for impulse response measurements are presented in this section. They employ impulses, pseudo-random noise, and pure tones as their excitation signals.

Impulses

From the impulse response point of view, an impulse is the most intuitive excitation. The impulse measurement method does not require finding an inverse function, as the captured response already is the desired IR (Müller and Massarani, 2001). The frequency response function is then obtained with DFT. The method has some practical limitations in acoustical measurements. Physical constraints in loudspeakers prevent the reproduction of perfect impulses. In addition, SNR from a single measurement is poor due to the very low amount of energy an impulse is able to bring to the system. However, in low noise digital systems, those problems do not exist and the impulse measurement is valid.

The impulse measurement method could be used in measurements of an anti-imaging filter of sampling rate conversions that interpolate by an integer factor. In these

cases, the impulse response characterizes the system fully even if SRC is inherently a time-variant operation.

Maximum-length sequences

In an MLS measurement, a sequence of periodic pseudorandom noise called maximum-length sequence is used as an excitation. The impulse response of a system is obtained by a cross-correlation between input and the measured output signals (Borish and Angell, 1983).

Compared to the impulse measurement method, better SNR can be achieved with the MLS technique because for given maximum amplitude, more energy is brought to the system (Borish and Angell, 1983). For this reason, MLS technique has been used often in measurements where noise is present, e.g., acoustical measurements. However, in the digital era, as the noise level in the measurement data storing has dropped dramatically, the internal SNR of the technique has been criticized (Müller and Massarani, 2001).

In addition, the MLS technique assumes perfect linearity and time-invariance of the system (Farina, 2000). Thus, it is not suitable for measurements of time-variant systems. Another drawback of the MLS is that it requires perfect synchronization of excitation signal and system output (Farina, 2000).

Stepped sinusoids

In the stepped sinusoids method, a pure tone is used as an excitation to the system. As a steady sine is able to measure a single frequency at a time, several sub-measurements are required. The system is excited over the desired frequency band by increasing frequency step by step. Removing the excitation from the system response yields the frequency response at the test frequency. The test sine can be removed by means of filtering and rectifying the fundamental or by numerically canceling the fundamental from the DFT of the response with coherent sampling (Müller and Massarani, 2001). When the desired frequency band is measured, the FR of the system is obtained by averaging the FR’s from the sub-measurements.

The measured frequency response function is inherently discrete. Unless smoothness of the system’s actual FR can be assumed or for some other reason a high resolution is required, the frequency distance between two successive sines has to be selected short enough. In that case, determining the FR over a wide band, e.g., the whole audio band can be exhaustingly time consuming (Müller and Massarani, 2001).

The stepped sinusoids method provides the best SNR of all known methods (Müller and Massarani, 2001). This is due to the possibility to use coherent sampling in the individual measurement of each sine. Another advantage of the method over the previously presented ones is that the distortion generated by the system can be evaluated from the same measurement.

5.3.2 Swept-sine technique

The swept-sine technique (SST), originally introduced by Farina (2000), uses logarithmic sweep as an excitation. With SST, it is possible to separate the non-linear distortion from the system's impulse response. More accurately, in addition to the IR corresponding to the linear part of the system is obtained, each IR corresponding to the harmonic distortion orders of interest can be selectively separated from the same measurement. This is because the SST produces a two-sided nonsymmetrical impulse response: on the right side of the impulse is the linear response of the system while the distortion is pushed to the left side.

In order to obtain the two-sided impulse response the input signal is deconvolved from the system output. For this purpose, an inverse filter that packs the input sweep into a delayed Dirac's delta function has to be generated. The inverse filter is a time reversed and equalized version of the input sweep. Its generation has been described in detail by Stan *et al.* (2002). Once the inverse filter has been computed, the deconvolution can be obtained by linearly convolving the output of the measured system with this inverse filter. According to Farina (2000), using linear convolution instead of circular one used in earlier methods is the trick that separates the distortion from the linear impulse response.

It should be noted that the SST method as used by Farina (2000) is not able to isolate distortion components that in frequency appear below the original signal. In fact, distortion sweeps always show above the excitation signal in the spectrograms presented in the article. Therefore, SST is not able to show aliased components, and direct application to SRC testing is not possible.

The requirement for the possibility to separate the distortions of different harmonics is that the sine is swept logarithmically. The distortion products become detached from the linear IR also with linearly swept sine, but it is not possible to distinct different harmonics as they collapse horizontally on the left side of the IR. In a logarithmic sweep, the instantaneous frequency is made to vary exponentially with time.

Logarithmic sweeps with the desired edge conditions are obtained from the equation

$$x(t) = \sin\left(\frac{T\omega_1}{\ln(\omega_2 / \omega_1)} \left(e^{(t/T)\ln(\omega_2/\omega_1)} - 1 \right) \right), \tag{5.9}$$

where ω_1 is the initial frequency and ω_2 is final the frequency of the sweep of duration T (Stan *et al.*, 2002). The magnitude spectrum of a logarithmic sweep is not flat; it decreases 3 dB per octave. This is due to the fact that frequency increases by a fixed factor per time unit, e.g., doubles each second. Every octave shares the same energy but as the bandwidth of an octave increases, the magnitude of each frequency component decreases (Müller and Massarani, 2001).

It is essential to pad the sweep with zeros. Without zeros, the captured IR will have attenuated images of the sweep close to the main impulse that interfere with the measurement. The number of zeros inserted in the beginning and the end of the sweep is proportional to the ill-sweeps distance from the center.

5.3.3 Discussion

Existing methods for frequency response function measurements were reviewed in the previous sub-sections. With the exception of the stepped sine method and the swept-sine technique, they are only applicable to LTI systems. In addition, all of them are formulated in the literature such that the measurement is carried out at a fixed sampling rate. In testing of SRC lowpass filters, this is not the case since the test signal and the signal under test are inherently at different sampling rates.

With the SST, the frequency response measurement is potentially achievable in multirate systems. The generation of the inverse filter needed in the deconvolution is an analytic operation but generalizing it for the measurements of a multirate system is not trivial. Therefore, a dedicated measurement method was developed.

5.4 Frequency response function measurements in sampling rate conversions

In the thesis, the frequency response functions are only computed in the evaluation of anti-imaging and anti-aliasing filters used in the sampling rate conversions. The requirements are stated for passband width, its flatness (the maximum ripple allowed), stopband start, and stopband attenuation. All those should be verified from the computed magnitude response. In the method used in this thesis, SST is simplified such that the inverse filter is not needed.

5.4.1 Approach

The idea is to generate an *equalization spectrum* that is able to deconvolve the excitation signal from the system output and thus yield the frequency response function of the filter. In subsection 2.3.2, it was shown that sampling rate expansion by a factor of L produces $L - 1$ additional images of the input spectrum. If we are able to generate the sampling rate expanded signal, given by the equation (2.9), from the test signal, or alternatively from its discrete-time Fourier transformed counterpart $Y(e^{j\omega})$ according to equation (2.10), the required equalization is available. After all, the rate-expanded signal is the signal that the lowpass filter actually sees.

As in SST, the excitation signal used is a logarithmic sweep ranging from 0 Hz to the Nyquist frequency of the input sampling rate. For optimum results, the test sweep should be padded with zeros, as is the case with SST. In the analysis, the equalization spectrum is generated by replicating the DFT of the test sweep. Replicated spectrum corresponds to the spectrum of the expanded signal that by definition is not filtered. This equalization spectrum is used to deconvolve the input of the system output. The

convolution is realized in frequency domain by dividing the system output spectrum by the equalization spectrum.

5.4.2 Equalization spectrum computation

The equalization spectrum for conversions by factor L / M, that also suits for conversions with integer factor, then M equals one, is generated from the excitation signal in the following manner. To get the reference spectrum, an N_F-point DFT is computed of the test signal, where N_F equals N_E (the length of the excitation signal in samples) rounded up so that it is divisible by M. The DFT outcome is repeated L / M times to produce the spectral images as would be seen from the sampling rate expansion. The length of the replicated spectrum becomes N_R. An equally long DFT is computed of the captured system output signal.

In case of interpolation by factor L, it is possible to generate the equalization spectrum in time-domain. The test sweep is first expanded by inserting $L - 1$ zero samples between each input sample. Then an $N_E \times L$ –point DFT is computed of the expanded signal as well as of the system output signal. With certain ratios, this approach permits a computation of considerable shorter DFT of the system output. However, for conversions with non-integer ratios, the time-domain expansion is not possible, as it would require insertion of a non-integer number of zeros between the samples.

5.4.3 Discussion

The generation technique of equalization spectrums that are able to deconvolve the test sweep from the captured output sweep was described above. Theoretically correct equalization spectrums are only available for the integer factor interpolation filters. In conversions by a non-integer factor, the technique gives suggestive estimates of the system FR function but they are not exact. If expansion and compression were done without filtering, the spectrum should have overlapping images, as seen in Figure 2.9 c). The summation of images depends on the phase relationships between the aliased images and the input spectrum that in practice cannot be controlled accurately. In principle, it is possible to generate such signal by doing the expansion followed by the compression without any filtering in between. However, using the spectrum computed of that kind of signal as an equalization spectrum was found out resulting totally misleading responses.

This technique, or any method that uses wideband signal as an excitation, is not able to reveal aliasing. This is because the aliased components are inevitably buried under the excitation as they fold down. A steady sinusoidal is the only suitable excitation in testing the attenuation of aliased components.

A suitable end-to-end measurement technique for the evaluation of filter frequency response function in upsampling was developed. The FR function is obtained from a single measurement and it contains the spectral information up to Nyquist frequency of the target sampling rate. The technique has a limitation that it gives exact

estimates of the frequency response functions in the conversions by integer factors only.

5.5 Distortion measurements

Distortion measurements are probably the most common measurements of audio devices. Cabot (1992) has presented different non-linear distortions and compared existing measurement techniques.

In a total harmonic distortion (THD) measurement, the system is excited with a single pure tone. If the system is linear, the magnitude spectrum of the system output has a single impulse on the test frequency. In case of a non-linear system, there will be seen additional signal components on the integer multiples of the test frequency, i.e. on harmonic frequencies of the fundamental. According to the definition, the measured RMS energy proportion of the harmonic components to that of the fundamental component is total harmonic distortion (Cabot, 1992).

5.5.1 THD+N

Alternatively, the test frequency is removed from the captured system output and the remainder is measured. At low levels of harmonic distortion, the noise level of the system contributes to the THD measure (Cabot, 1992). To emphasize the presence of contribution of the system noise, the distortion measures obtained with this approach are called THD+N. If the magnitude bins of the discrete Fourier transformed output signal are H_1, H_2, …, H_N, THD+N in per cents can be computed from the equation:

$$\%THD+N = \sqrt{\frac{H_1^{\,2} + H_2^{\,2} + ... + H_{N/2+1}^{\,2}}{H_1^{\,2} + H_2^{\,2} + ... + H_k^{\,2} + ... + H_{N/2+1}^{\,2}}} \times 100\% \qquad (5.10)$$

As can be seen from the equation (5.10), the only difference between the nominator and the denominator is that the noise and distortion term lacks the input signal component. Both levels include all harmonic, inharmonic and noise components.

It should be noted that each DFT bin contains all energy between two harmonics, not just the energy of strictly harmonic components. Furthermore, the contribution of all bins is included in the sum, except for the dividend, where the missing H_k refers to the removed fundamental frequency of the output signal. Using the coherent sampling presented in subsection 5.2.2, the energy of the input signal will be in one frequency bin. As the test frequency is known, the input signal can be discarded conveniently.

THD+N can be measured as a function of frequency or amplitude (AES17-1998). In order to measure it as a function of frequency, the test frequency is varied. It is recommended that the whole frequency range of the filter should be tested

(AES-17, 1998). The frequencies of the test signals are chosen roughly in octave intervals up to the passband end frequency of the anti-imaging or anti-aliasing filter under test.

Chapter 6

Design and implementation of test cases

When the signal is analyzed visually, e.g., by plotting the waveform data or magnitude spectrum, features can be recognized from surprisingly complex systems. As the goal in this work was to automate the test execution, visual analysis was not feasible. When testing is automated, parameter extraction and verification become complicated tasks for seemingly simple functionality. Surely, several parameters can be estimated from a single measurement. However, it is essential that each test case focus on a single functionality at the time. It not only makes the parameter extraction easier but also has an advantage that a failing of a test case directly indicates where the problems are. This can be achieved by isolating the feature, turning off all unnecessarily processing, and keeping test signals as simple and deterministic as possible. These aspects were taken into consideration in test design.

The tested functionalities were mixing, sampling rate conversions, and dynamic range compression as presented in Chapter 2. Depending on the functionality under test, the number of test cases varies greatly. In testing of interactive API level functionality, e.g., mixer controls, parameters had to be varied extensively. On the other hand, in case of passive functionality, e.g., sampling rate converters, it was possible to limit the number of test cases to the supported conversion ratios. Of course, the test signal set must then contain all frequencies.

As discussed in Section 4.3, automated testing of given functionality requires successful completion of three stages: test parameter selection (including variation), parameter estimation from the signal under test, and verification. Aspects from all stages are considered in each of the following sections but the emphasis varies a bit. In mixer control tests, the focus is on test parameter variation. In the testing of anti-image filters of sampling rate converters, parameter estimation is highlighted. A case study of verification with a test oracle is given in the context of DRC tests.

6.1 Tests for the mixer

EAP mixer supports channel controls and stream controls, as described in Section 2.2. The channel controls are level, panning, and muting setting. The stream controls are level, balance, and muting. Although they are semantically different controls, they can be tested quite similarly. In the end, they all affect the signal gain only. From analysis point of view, the only difference is that level control is inherently logarithmic whereas panning and balance are linear controls. Muting control is a special case and it was tested on a logarithmic scale but it might have been analyzed on a linear scale as well. In the first sub-section, the focus of testing is on the mixer control mechanism and the corresponding static gain calculations. The transitions from one value to another are considered in the second sub-section.

6.1.1 Mixer controls

The following criterion was considered common for all controls. The control should change the signal value exactly (with certain tolerances) as was requested. For instance, if a 6-dB level adjustment is requested, the change in the signal should be exactly 6 dB. Additionally, for level and muting the control should have an effect on the instance it was requested to, and only to that. For example, if a channel level change is requested to channel 1 of a stream it should affect the level of the channel 1 of the stream only. For panning and balance that are stereophonic controls, the testing of the latter requirement was considered too laborious from measurement point of view and was ignored.

Assumptions

It was assumed that the mixer controls are frequency independent, i.e., the output of the mixer does not depend on the frequency of the input signal. This assumption is considered reasonable as all controls affect the signal gain only.

Test design and instrumentation

Tests were designed such that they measure the change in the signal between two time instances rather than the absolute values. Change measurement was considered more convenient because the expected result need not depend on the input level of the test signal. It can even be justified psycho-acoustically as the human ear is more sensitive to detect changes than absolute values. In practice, the change measurement was realized as follows. To get a reference, test signal is first played back for a certain time with the control in its initial state. Then without any interruption in playback, the control change is requested. To get the changed value in the output signal, the signal is yet played for a certain time. With this approach, the stored signals have to be at least twice as long as with the absolute value measurement. However, this was not considered a critical issue. As the change measurement applies the control twice during the execution of SUT, it might even reveal system state related errors. For instance, if the muting control had an implementation failure that once channel is muted it stays muted, the change measurement would tell it.

In order to test whether level and muting controls are applied to the correct instance, two-channel streams are used in the tests. In channel control tests, both channels of the stream contain the same signal but the adjustment is requested for one of the channels only. In stream control tests, two stereo streams are used. The other stream has the test signal in its left channel and the other in its right channel. The adjustment is requested for one of these streams only. With this setup, the signal in one of the EAP output the channels should always remain intact. Assuming the frequency independency of mixing controls, the test waveform can be chosen quite freely. A full-scale 200-Hz sine that has a duration of 2 seconds at 48 kHz sampling rate was used.

Parameter estimation and verification

Both the signal analysis and verification is straightforward in these tests. RMS energy is estimated from both SUT output channels before and after the control change using equations (5.2) and (5.3). The analyzed data is extracted with 100-ms rectangular windows.

In case of level and muting controls, the signal energy values are converted to decibels relative to the amplitude of a full-scale sine, and signed level changes are computed for both channels. In the verification phase, the measured level change is compared against the required level change. In the case of the muting setting, the required level change is $-\infty$ for mute on and ∞ for mute off control. Test case is passed if the measured level change does not deviate from the required value more than the specified error tolerance. Typical error tolerance in these tests is 0.001 dB.

For panning and balance, the linear signal gains that would be required for the measured change in the channel energies are estimated. The verification is done similarly as for level and muting.

Parameter variation

With black-box testing approach, the only means of assuring that no errors are present in any of the mixer control mechanisms and their corresponding signal gain calculations is to derive so many test cases that they would set all possible control parameters values – not forgetting the illegal values. As stated earlier it is not reasonable even try to. Using equivalence partitioning presented in subsection 3.2.1, it is possible to derive an equally effective set of tests. In order to find those sets for each of the controls, similar inputs and similar outputs were partitioned. Especially, boundary and sub-boundary conditions, and invalid data conditions were considered.

As an example, Figure 6.1 depicts input and output relations for a) panpot and b) balance controls. Both controls have boundary conditions on both ends of their input value ranges. For balance, there is an additional sub-boundary condition at input value zero. Deriving a test case for each of the boundaries and just above and below them (e.g. 0.995 and 1.005 in case of boundary condition at 1.0) results six test cases for panpot and nine for the balance. Rest of the possible inputs are partitioned into

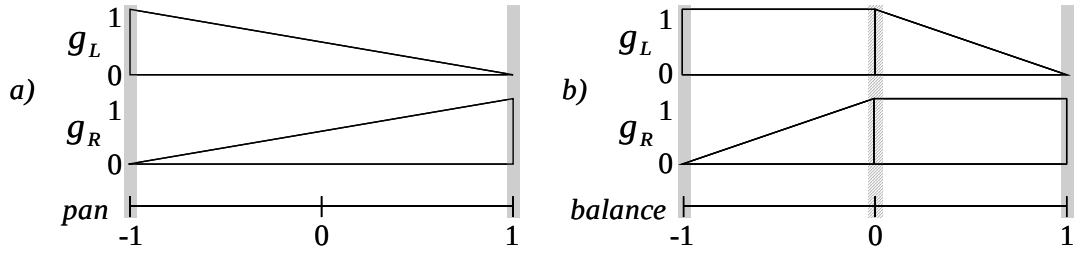


Figure 6.1 a) Boundary conditions in panpot. b)Boundary conditions (solid bars) and sub-boundary conditions (dashed bar) in balance control.

two classes that represent "typical" positive and negative input values. In addition, remembering that the tests were designed to measure the change in the gain, which means that each point is tested using two different reference points, yields in total 16 test cases for panpot and 21 test cases for balance.

6.1.2 Gain signal generators

The gain signal generators that are responsible of producing smooth level, pan, and balance transitions were tested. The focus is on verification of correct gain function, ramp duration, and on the smoothness of the realized ramp. In addition, the correct level change is required and they should be tested from both output channels. Panning and balance ramps should be linear and level ramps should be logarithmic. The shape of the ramp is verified from the amplitude envelope of the signal.

Assumptions

Gain signal generators are a part of the same mixer controls already tested, with the addition of only the time dependency. Therefore, it can be assumed that each control affects the requested instance and the generators are frequency independent.

Test design and instrumentation

Linear and logarithmic gain signal generators are tested by ramping panning, balance and level controls from one value to another with different ramp durations. In all cases, the response consists of three sections: onset, ramp, and set-value.

A 200-Hz full-scale sine at 48 kHz sampling rate was used as the test signal. The duration of the test signal depends on the desired transition duration; it is always about two seconds longer than the tested ramp to leave sufficient onsets and set-values.

Parameter estimation and verification

Amplitude envelope of a signal is computed with the technique presented in subsection 5.1.3. In Matlab, it is obtained with the following line of two nested

commands `y=abs(hilbert(x_w))`, where `x_w` is the channel output half windowed from both ends to suppress the discontinuation effects. In case of panning and balance ramps, the values are normalized to the range [0, 1]. Level ramps are first converted to decibel scale and then normalized to 0 dB. Once level ramps have been brought to a logarithmic scale, they appear linear. Panning and balance ramps appear inherently linear on a linear scale. With this scaling, level ramps and panning ramps can be inspected further similarly on linear fashion.

Initial and final gains or levels are first estimated from the long time average values of the onset and the set-value of the envelope. The beginning and the end of the ramp are detected. A line is fitted to the ramp section of the response in a least-squares sense and the limits for the ripple on each section of the response are derived. Test case is passed if the measured amplitude envelope does not violate the limits of error tolerance in any section.

In Figure 6.2 is plotted a) captured EAP channel data and b) estimated amplitude envelope with error tolerance margins in a level ramp test. The amplitude envelope appears precisely on high signal levels but on low levels, the quantization distortion interferes with the measurement, which requires the error tolerance to be substantially higher.

Parameter variation

The boundary conditions of the gain signal generators are readily available as specified minimum and maximum ramp lengths. A more interesting measure of algorithm performance can be obtained from the sub-boundary conditions. They are determined by the slope of the ramp and are tested when the slope of the ramp approaches zero and $\pm\infty$. In practice, as the former would require an infinitely long ramp and the latter an infinite level change, such slopes are not realizable. Table 6.1

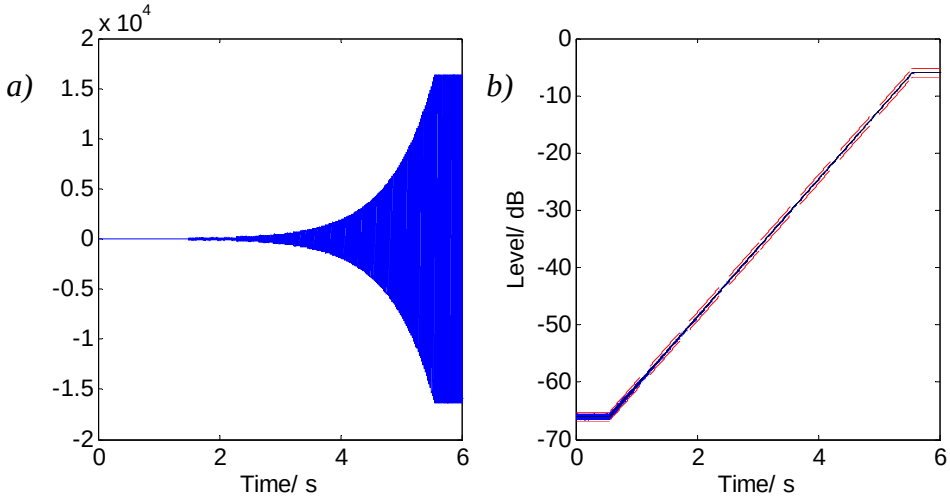


Figure 6.2: A 60-dB level ramp in 5 seconds. A) measured channel data.b) corresponding amplitude envelope and error tolerance margins (dashed lines) on a dB scale.

Table 6.1: The values used to test the sub-boundary conditions in gain signal generators.

Slope	Level change	Pan / balance	Ramp duration
<< 1	0.1 dB	-0.01 -> 0.01	5 s
>> 1	60 dB	-1.0 -> 1.0	100 ms

presents the selected parameter values for testing sub-boundary conditions with level ramps and panning and balance ramps.

6.2 Tests for the sampling rate converters

EAP audio output is typically at 48 kHz sampling rate. In addition to the main sampling rate, it supports several additional input sampling rates as presented in Section 2.3. All supported sampling rate conversions are tested in terms of conversion accuracy, anti-imaging and anti-aliasing filtering, and overall distortion.

The equivalence partitioning approach does not suit very naturally the test case selection problem in case of passive audio functionality because the properties of the test signal are the only parameters that can be varied. Surely, each conversion can be considered its own equivalence class and for each of them there are boundary conditions at frequencies 0 Hz and Nyquist frequency but finding the sub-boundary conditions is not as evident as in case of interactive functionality.

Magnitude response tests for anti-imaging filters were done with a sweep measurement. The rest of the measurements, that are, conversion accuracy, anti-aliasing, and THD+N measurements utilize coherent sampling technique. Since the last-mentioned tests have common test method, they share a considerable number of other aspects, too. Test signals, assumptions, test design and instrumentation, and parameter variation are common for all these tests and thus are described only once.

6.2.1 Test signal generation for the measurements using coherent sampling

Sinusoids are used as the test signals in most of the SRC tests. Being the most accurate method in frequency domain measurements of periodic signals coherent sampling (sub-section 5.2.2) was selected as a measurement technique.

In order to find the test frequencies, the DFT length K is selected first. In case of 8-kHz family of converters, $K = 6156$ is used in the analysis. The value was selected such that it is divisible by interpolation factors $L = 2$, 3, and 6. For 44.1-kHz family, $K = 5120$ is used. In this case, K is divisible by both the common decimation factor $M = 147$ and the interpolation factors $L = 160$, 320, and 640 related to the rate conversions from 44.1 kHz, 22.05 kHz, and 11.025 kHz, respectively.

After that, the test frequencies can be selected. The following standard guidelines were followed in the test frequency selection. 1) Frequencies were selected to cover the whole passband of the anti-imaging (anti-aliasing) filter under test in octave intervals as recommended in the AES standard (AES-17, 1998). 2) The IEEE guideline of optimal test frequency selection in coherent sampling was followed (IEEE-1241, 2001). The optimal frequencies are used when *J* has no common factors with *K*. That being said exact octave intervals are not always possible, but that was not considered critical. Additionally, a few exceptions were made in 2). The fundamental frequency and first harmonic (i.e., *J* equals 1 and 2) were included in order to get test signal coverage also in lower frequencies.

Finally, the actual test signals can be computed. Applying equation (5.6) to the sinusoidal equation gives a formula for the test signal generation

$$x[\mathrm{n}] = A\cos(2\pi\, t[\mathrm{n}]\frac{J}{K_{in}}F_S), \tag{6.1}$$

where $t[\mathrm{n}]$ is the discrete time vector, F_S the input sampling rate, and $K_{in} = KL/M$ is the length of the DFT window as seen at the input sampling rate, e.g., K_{in} = 1026 at 8 kHz input rate.

The divisibility of *K* by *L* and *M* is not mandatory for most of the tests. However, it was found out that when it is followed, perfect accuracy is also guarantied for the expected images and aliased components. This is illustrated in Figure 6.3. The rate conversion was computed with the Matlab built-in `resample(x,L,M)` function. In Figure 6.3 b) is plotted the magnitude spectrum of a sinusoid interpolated by a factor of 6. The DFT length was chosen as *K* = 8198, that is not divisible by 6. Being a power-of-two number this selection is otherwise desirable but only the fundamental frequency (i.e., the test signal) fits in the DFT. The frequencies of the image components do not meet the condition for coherent sampling and thus appear inaccurate in both frequency and magnitude and are affected by the spectral leakage. In Figure 6.3 a), *K* is selected such that K_{in} is also an integer. Accurate frequency and amplitude are now observed in image components, too.

Table A.1 in Appendix A summarizes the test frequencies used in the tests that employ the coherent sampling technique. In the same table, are also given passband and stopband frequency specifications for each filter. Frequencies of the test signals cover only the passband of the filter. Extending test frequencies up to Nyquist frequency of the input sampling rate is not justifiable, as the transition band of the filter is not specified in the requirements.

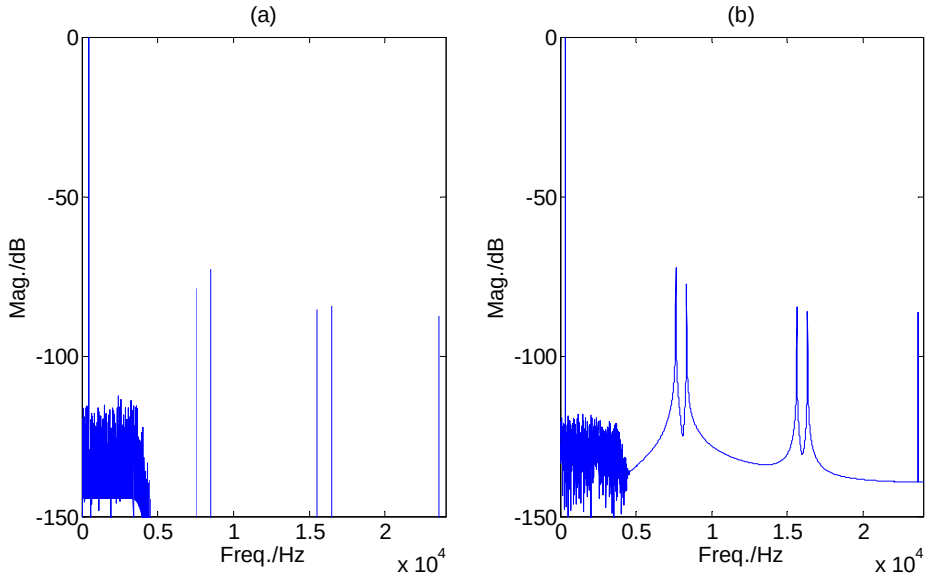


Figure 6.3: The effect of DFT length (K) selection in a sampling rate conversion from 8 kHz to 48 kHz (L = 6) done with Matlab built-in resample function when a) K = 6156 is divisible 6, and b) K = 8192 is not divisible by 6. In both cases, the frequency of test sine is 475.6 Hz (J=61).

6.2.2 Rate conversion accuracy

The fundamental properties of a signal should not change in a sampling rate conversion. In case of sinusoids, it means that the frequency and the amplitude should remain unchanged in the rate conversion. Deviations from constant amplitude implicate that the filter passband is not flat. In rate conversion accuracy tests, the immutability of frequency and amplitude of sinusoids were tested. The amplitude accuracy was assumed independent of the amplitude of the input signal. Thus, the amplitude of the test signal was not varied in the tests.

Test design and instrumentation

Rate conversion accuracy was tested with the coherent sampling technique. Each conversion is tested with several sinusoids that have their frequencies as specified in Table A.1. A one-channel stream having one of those sinusoids as a source is opened at correct input sampling rate and played for two seconds at a time. As the frequency and the amplitude of the test sinusoid are known it is straightforward to verify the measured values against them.

Testing the frequency accuracy of a given converter with several frequencies is somewhat exaggeration. Expressed in samples the frequency immutability means that the outcome of rate conversion by L/M should have $N \times L/M$ samples, where N is the number of samples in the input signal. It does not depend on the frequency of the input signal.

Parameter estimation and verification

A DFT of length *K* is computed of the output signal and scaled to dB FS. The highest peak in the magnitude spectrum is interpreted as the test signal. Its bin index *k* and amplitude *A* are determined. If *k* equals to *J*, the frequency is exact and if the determined amplitude deviates from – 6.0206 dB FS level[5] less than the required error tolerance the rate conversion is considered accurate at the test signal and test is passed.

It is noted that the tester cannot claim that a given converter is amplitude accurate by a single passed test case. All test cases for the converter must be passed in order to be able to say that with reasonable confidence. However, this is not a big problem in automated testing. As all test cases are run the judgment is pronounced in the summary of results available in the test report.

6.2.3 Anti-imaging filtering

Upsampling requires a lowpass filter that removes the spectral images (refer subsection 2.3.2 for details). Magnitude responses of the anti-imaging and anti-aliasing filters were tested in terms of the width and the flatness of the passband, stopband start, and the stopband attenuation. The flatness of the passband is specified as a maximum ripple allowed. All input rates were tested.

Test design and instrumentation

The equalization spectrum technique, described in section 5.4 was used. The excitation signal is a logarithmic sweep ranging from 0 Hz to the Nyquist frequency of the input sampling rate at 0-dB FS level. Logarithmic sweeps according to equation (5.9) were generated with Matlab built-in function `chirp`. The length of the sweep was chosen 8 seconds, which gives very high resolution for the frequency response function. The sweep was padded with 0.5 seconds of zeros to the both ends.

Parameter estimation and verification

The magnitude spectrum of the filter is computed using the equalization spectrum technique described in section 5.4.

Upper and lower limit lines for the passband ripple and a line for the minimum attenuation required in the stopband were defined. The magnitude of the convolved filter response is verified against these lines. Both the lower and the upper limit lines begin from the passband start frequency. The lower limit ripple line reaches the specified passband end frequency while the upper limit line extends up to the stopband start frequency.

[5] The value is one half in decibels and is resulted when a one-channel full-scale sine is panned center. Refer section 2.2.2 for details.

In Figure 6.4, is plotted the obtained magnitude spectrum and the lines according to required filter specifications. Magnitude of the filter frequency response function is checked against the defined lines. If no violations are found the test is passed. In case of failure, the test is able to report the realized width for passband the start frequency for the stopband.

6.2.4 Anti-aliasing filtering

As presented in subsection 2.3.4, upsampling by a rational factor is prone to aliasing artifacts. If the anti-alias filtering is not done correctly, aliased components can be heard on audio band. Criterion in the tests is that the magnitude response of the EAP output should only have one frequency peak (i.e., the test signal) above the stopband attenuation requirement of the anti-aliasing filter under test. However, aliased components are allowed above the audio band, e.g., 20 kHz. This is taken on note with the upper frequency limit.

Test design and instrumentation

Each conversion that operates on rational factor is tested with several sinusoids that have their frequencies as specified in Table A.1. Coherent sampling technique (subsection 5.2.2) is used in the measurements.

Parameter estimation and verification

Magnitude of DFT of length K is computed of the output signal and scaled to dB FS. The magnitude spectrum is searched for peaks H_k that are above the minimum

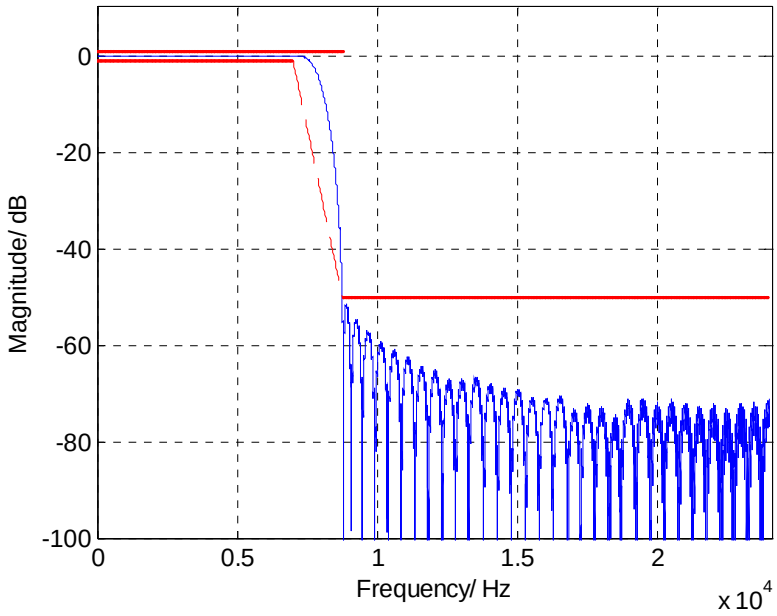


Figure 6.4: The magnitude response of the anti-imaging filter in a conversion from 16 kHz to 48 kHz. Note: The error tolerance for passband ripple is exaggerated for clarity.

stopband attenuation requirement. Components above 20 kHz are ignored. If more than one peak is found the test is failed and the frequencies corresponding to k are reported.

In Figure 6.5, is plotted the magnitude spectrum of the SUT output in SRC from 32 kHz to 48 kHz. With 14,4 kHz test frequency, the strongest spurious components produced by the sampling rate converter are an image at 17,6 kHz and an aliased component at 1,6 kHz. Amplitude levels of all distortion components are below the required 60-dB attenuation.

6.2.5 THD+N

The overall distortion produced by SRC was evaluated with the Total Harmonic Distortion and Noise (THD+N) measurement described in section 5.5.1. The THD+N was evaluated as a function of frequency.

Test design and instrumentation

Each conversion is tested with several 0-dB FS level sinusoids that have their frequencies as specified in Table A.1. Coherent sampling technique (subsection 5.2.2) was used in the measurements. For better perceptual relevance, the distortion

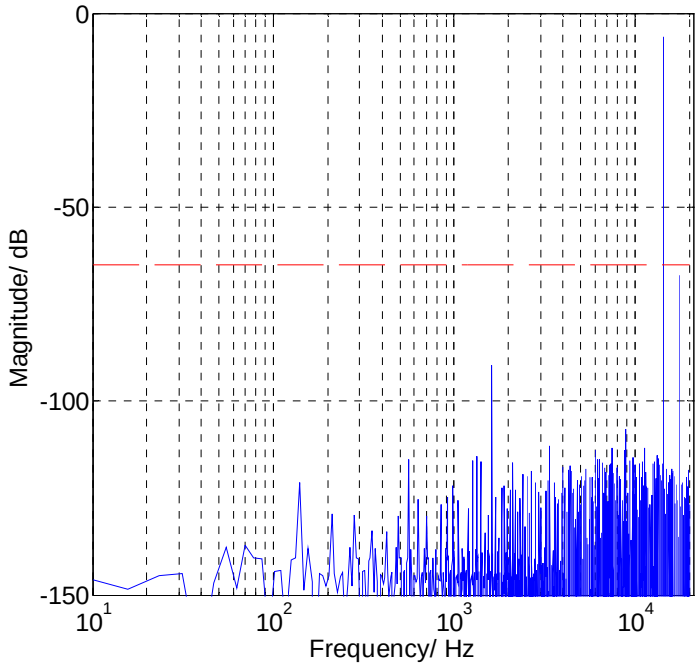


Figure 6.5: Test for anti-aliasing filter in 32 to 48 kHz SRC. The highest peak at 14,4 kHz is the test frequency. The Dashed line at –60 dB level indicates the attenuation required of aliased components.

and noise components were weighted with a standard A-weighting filter.

Parameter estimation and verification

At first, the magnitude of the DFT of length K is computed of the output signal. Components above 20 kHz are discarded from the spectrum. The resulted signal is the divider of equation (5.10). To get the distortion and noise, the test signal is cancelled from the spectrum. As the frequency (J) of test sinusoid is known, the cancellation is obtained when H_J is set to zero.

The distortion and noise signal is filtered with A-weighting function. As the distortion and noise components are already in a frequency domain, the weighting is most conveniently carried out in frequency domain. The frequency response function of the A-weighting filter (equation (4.1)) is evaluated at the DFT-points. The weighting is obtained as the distortion and noise signal is multiplied pointwise by the weighting function.

In Figure 6.6, is plotted the A-weighted distortion and noise in rate conversion from 44.1 kHz to 48 kHz. The magnitudes of the distortion components are well below –100 dB FS level. A-weighted THD+N according to equation (5.10) is obtained when the A-weighted distortion and noise is divided by the above-computed divider. The test is passed if the computed THD+N does not exceed the allowed value.

6.3 Tests for DRC static parameters

Dynamic range controller was tested. The focus of testing is on static parameters, especially the compression curve. Test oracle is used in the verification. It is assumed that processing is frequency independent, i.e., DRC boosts all frequencies equally.

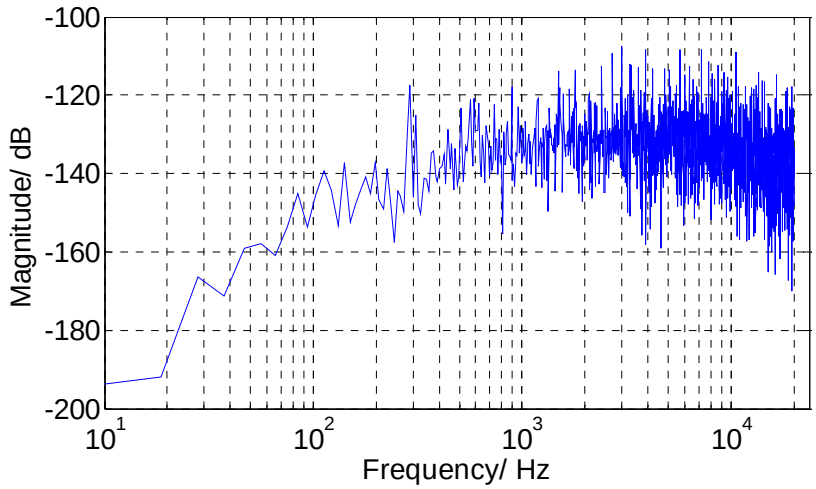


Figure 6.6: A-weighted distortion and noise originating from a 1,2 kHz sinusoid in a rate conversion from 44.1 to 48 kHz.

Being a full-band DRC this assumption is relevant.

Test design and instrumentation

In order to simulate dynamic variations of natural sounds, the input signal is made having ascending and descending level steps. Actually, the test signal has constant level but the steps are generated with (already tested) EAP level controls. This is possible as the mixing takes place before the DRC processing. To make analysis simpler it is essential that the DRC output reaches its static condition before the next input level change takes place. Thus, the lengths of the level steps were selected at least as long as the attack and release times of tested preset. The values for the compression curve turning points are read from the same configuration file that EAP uses. Thus, new test cases for later added compression curves can be derived without need for calculating expected output levels for each curve.

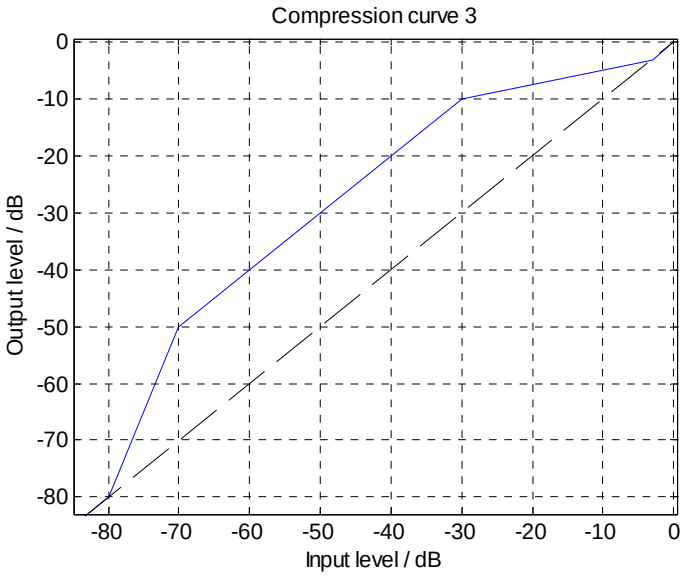
Parameter estimation and verification

Parameter under test is a signal level after each level step. Level steps are detected from EAP output and measurement points are selected at the end of each step. Same measurement points are used throughout the analysis. Level estimation is computed using equations (5.2) and (5.3).

Expected parameter values are computed with a simplified DRC model. It models those signal processing blocks in EAP DRC algorithm (subsection 2.4.4) that contribute to the static parameter behavior. They are input signal level estimation, compression curve computation, and gain multiplication. The test signal and the level adjustments applied in the test are fed to the model. Expected output levels corresponding to the predefined compression curve are obtained. Error, i.e., the difference of expected and realized levels is computed and a verifier compares it against test case specific error tolerance. If measured error in each level step is less than the tolerance, the test case is considered passed. Implementation of analysis and its essential helper functions are given in the code listings available in Appendix 0.

Figure 6.7 a) illustrates compression curve used in a DRC test. It has ET – 70 dB, CT – 30 dB, and CF 20 dB. With the presented curve, DRC does not perform any noise-gating or limiting. Correctness of CT is tested with an input level sequence – 35 dB, – 20 dB, – 15 dB, – 20 dB, – 35 dB. The corresponding output energy level sequence and error tolerances are plotted in Figure 6.7 b). In order to give overall view of the test, the amplitude envelope of the output signal is aligned with energy levels.

a)



b)

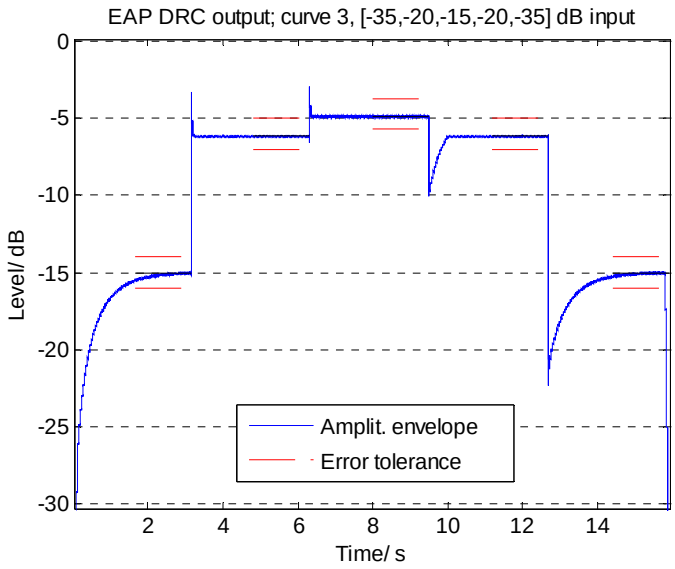


Figure 6.7: Figure plots from a DRC test. a) Compression curve b) EAP output and the error tolerances. Note: for the clarity, the energy averaging windows are presented significantly longer and the error tolerance is doubled.

Parameter variation

Input levels for a single test case are chosen around one of the compression curve turning points. Each curve has four turning points in maximum. All turning points of various compression curves are tested.

Chapter 7

Conclusions and future work

The purpose of this work was to design and implement automated tests for the core functionality in the EAP audio platform.

In order to familiarize the reader with the object to be tested, core functionalities of EAP were presented. In addition to the operation and applications, the necessary theoretical background of audio mixing, sampling rate conversions, and dynamic range compression was studied. To solve the addressed problem from software point of view, the conventional software testing methodology was reviewed. The most important findings from software testing literature were the systematic methods for test case selection problem and the concept of automated test oracles.

On the other hand, audio testing and audio quality aspect had to be considered. The existing methodology in audio testing was found to be audio reproduction system oriented. On the audio quality side, measurement methods based on the physical signal representation and on the hand, perceptually motivated parameters were shortly reviewed. Recent research results strongly recommend the latter approach. However, as the focus in EAP testing is in functional testing, and more importantly, as the current quality requirements stated for EAP are measured in Hertz and decibels, the conventional quality measurement methods were considered sufficient for the time being.

To find sufficient methods for functionality testing, audio signal analysis techniques were studied. The necessary methods needed in testing the mixing and DRC functionalities were otherwise basic but a clever technique for amplitude envelope computation came by. In case of mixer controls, essential part of the functional testing was the control parameter variation. The equivalence partitioning method commonly used in software testing was found valuable for that. However, in case of passive functionality, e.g., sampling rate conversion, where the test frequency is the only parameter that is varied, traditional “octave step recommendations” were found more natural.

The most challenging part of the work, in audio signal processing sense, was to find suitable measurement techniques for the sampling rate converters. As measured end-to-end, testing the anti-imaging and anti-aliasing filters was not easy. In the measurements for anti-aliasing, it turned out that the classical stepped sine method was the only applicable method. In case of rate conversion accuracy and THD+N measurements, the discrete Fourier transform was found to give the most accurate results when used with coherent sampling — a practical technique found from the literature concerning analog-to-digital converter measurements.

For system frequency response function measurement, several techniques were considered but apart from the stepped sine technique, none of them directly suited to a multirate system. To work out the problem, a sine sweep based measurement method was derived from the upsampling theory. In the method, a reference spectrum computed from the test sweep is used to deconvolve the test sweep from the system output. The derived equalization spectrum method has its limitations but precise magnitude spectrums are obtained in interpolations by an integer factor. It took a while to have confidence in the method as it first showed strange high-energy peaks in the magnitude spectrums measured from EAP. After a careful inspection, it turned out that the peaks actually originated from a minor rounding mistake in the EAP code. As a bug was found that any other test hardly would have found, developing the method was worth the effort.

Unfortunately, also the test system itself is exposed to bugs. In fact, they lurk behind every corner: in compilation time and runtime configurations of EAP, in the test design, in test case specification, and in the Matlab analysis code. Before a single real bug can be found, all bugs on the way have to be eliminated. One way to prevent bugs in the test parameter variation phase is to use automated test oracles. Test oracle concept is getting attention in software testing and it was experimented in DRC tests. As the input-output pairs need not be computed in the test case specification phase, it has an advantage in testing complicated systems. Given the input parameters and error tolerance, the oracle is able verify the functionality.

The expected disadvantage of automated audio testing is that only objective evaluation is possible. Unless perceptual models are used, it is difficult to find methods that acquire subjectively relevant measures. One of the tasks in the future is to consider perceptually motivated quality measures. An interesting task that was actually close to its resolution during the thesis work would be to generalize the swept sine technique for sampling rate converter measurements.

To conclude, testing EAP through the API was found to be possible but it cannot be considered a trivial task. Testing algorithms separately on a lower level would be more efficient. However, it does not dispense with system level testing. The end-user of the system listens to a black box, too!

References

3GPP, 2002, “Adaptive Multi-Rate (AMR) Wideband speech codec test sequences”, Technical Specification, 3GPP, Available online at http://www.arib.or.jp/IMT-2000/V430Dec04/5_Appendix/Rel5/26/26174-540.pdf

AES-17, 1998, “AES Standard Method for Digital Audio Engineering – Measurement of Digital Audio Equipment”, Audio Engineering Society, 1998, Available on-line at http://www.aes.org/standards/b_pub/aes17-1998.pdf, Referenced 20.11.2004.

Audio Precision, 2005, “Audio Precision 2700 Series Instrument Specifications”, Product Specification, Available on-line at http://audioprecision.com/bin/2700_series_specs.pdf, Referenced 12.2.2005

Baresi, L., Young, M., 2001, “Test Oracles”, Technical Report, Dept. of Computer and Information Science, Univ. of Oregon, August 2001, Available on-line at: http://www.cs.uoregon.edu/~michal/pubs/oracles.pdf Referenced 10.2.2005.

Beerends, J., Stemerdink, J., 1992, “A Perceptual Audio Quality Measure Based on a Psychoacoustic Sound Representation”, *Journal of AES*, Vol. 40, No 12, pp. 963-978

Beizer, B., 1990, Software Testing Techniques. 2nd ed., Van Nostrand Reinhold Co. New York, International Thomson Computer Press. pp. 550.

Blesser, B., Baeder, K., 1968, “A New Approach to Dynamic Range Compression for Audio Systems”, *AES 35th convention*, Oct., 1968, New York. Preprint No. 602 (K-10), pp. 1-10

Borish, J., Angell, J., 1983, “An Efficient Algorithm for Measuring the Impulse Response Using Pseudorandom Noise”, *Journal of AES*, Vol. 31 No. 7/8, pp. 478-488.

BS-1387, 1999, “Method for objective measurements of perceived audio quality”, ITU-R, pp. 100.

Cabot, R., 1986, “Automated Measurements of Loudspeaker Small Signal Parameters”, *Preprint 2402,* AES Convention 81, Los Angeles, Nov. 12-16, pp. 9.

Cabot, R., 1987, “Automated Measurement of Compressors and Expanders”, *Preprint 2513*, AES Convention 83, New York, Oct. 16-19, pp. 7

Cabot, R., 1992, “Comparison of Nonlinear Distortion Measurement Methods”, *Proceedings of the AES 11th International Conference: AES Test & Measurement Conference*, pp. 53 – 65

Cabot, R., 1999, “Fundamentals of Modern Audio Measurement”, *Journal of AES*, Volume 47, Number 9, pp. 738-744, 746-762

Dallas Semiconductor, “Coherent Sampling vs. Window Sampling” Maxim application note, March 29, 2002, available on-line at http://pdfserv.maxim-ic.com/en/an/AN1040.pdf, Referenced 31.12.2004.

DLS-2, 1999, “Downloadable Sounds Specification Level 2”, Version 1.0c, MIDI Manufacturers Association, pp. 85.

Elliot, R., 2003, ” ‘A’ Weighting Filter For Audio Measurements”, Available on-line at http://sound.westhost.com/project17.htm, Referenced 4.2.2005

Farina, A., 2000, “Simultaneous Measurement Of Impulse Response And Distortion With A Swept-sine Technique”, *110th AES Convention*, Paris 18- 22, February 2000, Preprint 5093, pp. 23.

Groeper, G., Blanchard, M., Brummett, T., Bailey, J., 1991, “A Reliable Method of Loudspeaker Rub and Buzz Testing Using Automated FFT Response and Distortion Techniques”, *Preprint 3161*, AES Convention 91, New York, Oct. 4-8, pp. 63.

Haikala, I., Märijärvi, J., 2002, ”*Ohjelmistotuotanto*”, 8th ed., Talentum Media Oy, pp. 430.

Harris, F., 1978, “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform.” *Proceedings of the IEEE*, Vol. 66, No. 1, pp. 51-83.

Heyser, R., “Acoustical Measurements by Time Delay Spectrometry”, *Journal of AES*, Vol. 15, No. 4, October 1967, pp. 370-382.

Hofbauer, K., 2004, “Estimating Frequency and Amplitude of Sinusoids in Harmonic Signals - A Survey and the Use of Shifted Fourier Transforms”, *Diploma*

Thesis, Graz Univ. of Technology, Available on-line at http://www.konrad-hofbauer.de/papers/web_hofbauer.pdf, Referenced 6.2.2005, pp. 111

IEC 61672-1, 2003, "Electroacoustics – Sound level meters, Part 1: Specifications", CENELEC, pp. 43.

IEEE 829, 1998, "Standard for software test documentation", The Institute of Electrical and Electronics Engineers, pp. 41.

IEEE 1241, 2001, "IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters", The Institute of Electrical and Electronics Engineers, Inc. pp. 98.

Karjalainen, M., 1999, "*Kommunikaatioakustiikka*", korjattu esipainos, The Report Series of Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing, pp. 237

Lahti, T., 1995, "*Akustinen mittaustekniikka*", 2. ed., The report series of Helsinki Laboratory of Acoustics and Audio Signal Processing, Helsinki University of Technology, pp. 152.

Leese, M., "Ambisonic Surround Sound FAQ" Available online at http://members.tripod.com/martin_leese/Ambisonic/faq_latest.html, Referenced 4.10.2004.

Marple, S., 1998, "Computing the Discrete-Time 'Analytic' Signal Via FFT," *IEEE Transactions On Signal Processing*, Vol. 47, No. 9, pp. 2600-2603.

McNally, G., 1984, "Dynamic Range Control of Digital Audio Signals", Journal. of AES, Vol. 32 No. 5 pp. 316-327.

Memon, A., Pollack, M., Soffa, M., 2000, "Automated Test Oracles for GUIs", ACM SIGSOFT Software Engineering Notes, Vol. 25, No. 6, pp. 30 – 39, Available on-line at http://www.cs.pitt.edu/~soffa/research/SE/FSE2000.pdf, Referenced 12.2.2005.

Müller S., Massarani, P., 2001, "Transfer-Function Measurement with Sweeps", *Journal of AES*, Vol. 49, No. 6, pp. 443-471.

Myers, G., 1979, "*The Art of Software Testing*", John Wiley & Sons, Inc., pp. 170.

Nisbett, A., 1979, "*The Technique Of The Studio Sound*", 4th ed., Focal Press, pp. 560.

Oppenheim, A., Schafer, R., Buck, J., 1999, "*Discrete-time signal Processing*", 2nd ed., Prentice-Hall Inc., pp. 870.

Patton, R., 2001, "*Software Testing*", SAMS Publishing, pp. 389.

Pulkki, V., 1997, "Virtual Sound Source Positioning Using Vector Base Amplitude Panning", *Journal of AES*, Vol. 45, No. 6, pp. 456 - 466.

Roberts, R., 1968, "High Speed Automated Test Set", *Preprint 573*, AES Convention 34, Apr. 29 – May 2, pp. 4.

Rossing, T., "*The Science of Sound*", 2nd ed., Addison-Wesley, pp. 686.

Smith, J., 2004a, "Digital Audio Resampling Home Page", website, Available on-line at http://ccrma-www.stanford.edu/~jos/resample/, Referenced 10.2.2005

Smith, J., 2004b, "Analytic Signals and Hilbert Transform Filters", in Mathematics of the Discrete Fourier Transform (DFT), with Music and Audio Applications, Available on-line at http://www-ccrma.stanford.edu/~jos/mdft/Analytic_Signals_Hilbert_Transform.html.html, Referenced 22.12.2004.

Smith, J., 2005, "First-Order Discrete-Time Wave-Variable Conversion Filters", in Choice of Wave Variables in Digital Waveguide Models, Available on-line at http://ccrma.stanford.edu/~jos/VariableChoice/First_Order_Discrete_Time_Wave_Variable_Conversion.html, Referenced 10.2.2005

Smith, J., Gossett, P., 1984, "A Flexible sampling-rate conversion method", *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, San Diego, vol. 2, (New York), IEEE Press, pp. 19.4.1–19.4.2.

Stan, G., Embrechts, J., Archambeau, D., 2002, "Comparison of Different Impulse Response Measurement Techniques", *Journal of AE*S, Vol. 50, No. 4, pp. 249-262.

TTCN-3, 2005, "The Testing and Test Control Notation", TTCN-3 website, Available on-line at http://www.etsi.org/ptcc/ptccttcn3.htm, Referenced 10.2.2005.

Vaidyanathan, P., 1993, "*Multirate systems and filter banks*", Prentice Hall, pp. 900.

Zölzer, U., 1998, "*Digital Audio Signal Processing*", John Wiley & Sons Ltd., pp. 290.

Appendix A

Test frequencies

Table A.1 gives the test frequencies that were used in the tests for sampling rate converters. The frequencies are rounded to the four most significant digits.

Table A.1: Filter specifications and test frequencies in sampling rate conversions.

Input rate (kHz)	Passband end (kHz)	Stopband start (kHz)	Test frequencies (Hz)
8	3.5	4.4	7.797, 15.59, 23.39, 54.58, 132.6, 241.7, 475.6, 1006, 2004, 3485
11.025	4.5	6.0	9.375, 18.75, 28.13, 65.63, 159.4, 290.6, 571.9, 1209, 2409, 4490
16	7.0	8.8	7.797, 15.59, 23.39, 54.58, 132.6, 241.7, 475.6, 1006, 2004, 4000, 6994
22.05	9.0	12.1	9.375, 18.75, 28.13, 65.63, 159.4, 290.6, 571.9, 1209, 2409, 4809, 8990
24	10.0	13.2	7.797, 15.59, 23.39, 54.58, 132.6, 241.7, 475.6, 1006, 2004, 4000, 7977, 9988
32	14.4	17.6	7.797, 15.59, 23.39, 54.58, 132.6, 241.7, 475.6, 1006, 2004, 4000, 7977, 1.439×10^4
44.1	18.0	26.4	9.375, 18.75, 28.13, 65.63, 159.4, 290.6, 571.9, 1209, 2409, 4809, 9590, 1799×10^4

Appendix B

Implementation

This appendix introduces Matlab source code for selected part of the work.

B.1 Analysis of DRC

B.1.1 test_drc.m

Test_drc.m is the Matlab entry point in the analysis part of test cases for full-band DRC as defined in Section 6.3. Script takes filenames of the test signal and the signal under test; DRC control parameters, and test criteria from Matlab workspace. It reads signals from files and passes them and the criteria to a test oracle that verifies the signal under test. Finally it reports the results and handles figure plotting if requested.

```
%test_drc(data_file_name,input_file_name, sample_rate_out, ...
%         number_of_output_channels, step_dur_seconds, ...
%         in_level1,in_level2,in_level3,in_level4,in_level5, ...
%         ccurve_inlevels, ccurve_outlevels, preset_id, ...
%         error_tolerance_db_pp, figure_plotting)
%
% Analysis script for full-band DRC.
%
% Uses find_drc_mspoints.m (not listed), drc.m, rawread.m (not
% listed), cat_report.m (not listed), and standard Matlab functions

% initialize output variables
ok = 0;
report = '';
tolerance = '';

% assign input values to short variables
fs_out = sample_rate_out;
numch = number_of_output_channels;
steplen = step_dur_seconds*fs_out;
etol_db = error_tolerance_db_pp /2;
curve_in = ccurve_inlevels;
curve_out = ccurve_outlevels;

% collect level adjustments into array
numsteps = 5;
```

```
inlevels = zeros(1,numsteps);
inlevels(1) = in_level1;
inlevels(2) = in_level2;
inlevels(3) = in_level3;
inlevels(4) = in_level4;
inlevels(5) = in_level5;

% read EAP numch-channel output from raw audio file
[eap_output, numsamples] = rawread(data_file_name,numch);
% strip leading and trailing zeros from the output
eap_out=stripzeros(eap_output,numch);

% read test signal from raw audio file
[test_signal, i] = rawread(input_file_name,numch);

% compute amplitude envelope from left channel (for visualization)
peak_envel=20*log10(abs(hilbert(eap_out(1,:)))/(2^15-1));

wlen= 24000; % length of the measurement window

% pick measurement points from the amplitude envelope
ms_points = find_drc_mspoints(peak_envel,fs_out, ...
                              numsteps,steplen,wlen);

% verify
[ok,levels] = drc(test_signal, eap_out, inlevels, ms_points, ...
                  curve_in, curve_out, etol_db, figure_plotting);

levels_realized = levels(1,:);
levels_expected = levels(2,:);
error = levels(3,:);
tolerance = [num2str(error_tolerance_db_pp) ' dB pp'];

% report results
report=cat_report(report,['DRC preset ' num2str(preset_id) '.\n']);
report=cat_report(report,['Compression curve in levels: ' ...
                          num2str(curve_in) ' dB.\n']);
report=cat_report(report,['Compression curve out levels: ' ...
                          num2str(curve_out) ' dB.\n']);
report=cat_report(report,['EAP input levels: ' ...
                          num2str(inlevels) ' dB.\n']);
report=cat_report(report,['Expected output levels: ' ...
                          num2str(levels_expected) ' dB.\n']);
report=cat_report(report,['Measured output levels: ' ...
                          num2str(levels_realized) ' dB.\n']);
report=cat_report(report,['Error (Lexpected - Lmeasured): ' ...
                          num2str(error) ' dB.\n']);
if ok == 1
    report=cat_report(report,['Analysis passed.']);
else
    report=cat_report(report, ['Analysis failed.']);
end

% for debug mode, plot measured and expected levels
if figure_plotting == 1
    figure;

    % plot amplitude envelope of the output to get the overall view
    Ny = length(peak_envel);
    time_axis=linspace(0, Ny/fs_out, Ny);
    plot(time_axis, peak_envel);
    hold on;

    % scale energy levels to peak amplitude levels
    dBFS_offset1 = 3.9318; % offset for energy level of a FS sine
```

```
    levels_realized = levels_realized + dBFS_offset1;
    levels_expected = levels_expected + dBFS_offset1;

    % compute time axis, realized level lines and error tolerance
    % lines for each level step
    t=ones(1,2);
    ms_points_sec = ms_points / 48000;

    t1 = linspace(ms_points_sec(1,1)-1,ms_points_sec(2,1),2);
    level1=t*levels_realized(1);
    max1 = t*(levels_expected(1)+etol_db);
    min1 = t*(levels_expected(1)-etol_db);

    t2 = linspace(ms_points_sec(1,2)-1,ms_points_sec(2,2),2);
    level2=t*levels_realized(2);
    max2 = t*(levels_expected(2)+etol_db);
    min2 = t*(levels_expected(2)-etol_db);

    t3 = linspace(ms_points_sec(1,3)-1,ms_points_sec(2,3),2);
    level3=t*levels_realized(3);
    max3 = t*(levels_expected(3)+etol_db);
    min3 = t*(levels_expected(3)-etol_db);

    t4 = linspace(ms_points_sec(1,4)-1,ms_points_sec(2,4),2);
    level4=t*levels_realized(4);
    max4 = t*(levels_expected(4)+etol_db);
    min4 = t*(levels_expected(4)-etol_db);

    t5 = linspace(ms_points_sec(1,5)-1,ms_points_sec(2,5),2);
    level5=t*levels_realized(5);
    max5 = t*(levels_expected(5)+etol_db);
    min5 = t*(levels_expected(5)-etol_db);

    % plot error tolerances and estimated output level
    plot(t1,max1,'r--',t2,max2,'r--',t3,max3,'r--',t4,max4,'r--',...
         t5,max5,'r--');
    plot(t1,min1,'r--',t2,min2,'r--',t3,min3,'r--',t4,min4,'r--',...
         t5,min5,'r--');
    plot(t1,level1,'k',t2,level2,'k',t3,level3,'k',t4,level4,'k',...
         t5,level5,'k');
    hold off;

    title([ 'EAP DRC output; curve ' num2str(preset_id) ...
            ', [' num2str(inlevels) '] dB input ']);
    xlabel('Time/ s');
    ylabel('Level/ dB');
    legend('Amplit. envelope','Error tolerance',0); %0=best position
end
```

B.1.2 drc.m

Drc.m is the test oracle in full-band DRC tests. It estimates energy levels of system output *y* in given windows and verifies them against the expected values. Expected levels are obtained from the input signal *x* with a simplified DRC model.

```
function [ok, levels] = drc(x, y, level_adjust, ms_points, ...
                            ccurve_in, ccurve_out, etol_db, ...
                            figure_plotting)
% Test oracle for DRC.
%
% Uses compressor.m and standard Matlab functions

Ny=length(y);
```

```
num_steps = length(level_adjust);

% measure realized levels at the end of each level step
levels_real=zeros(2,num_steps);

for i = 1:num_steps
    % calculate energy estimate and normalize level to 0 dB
    levels_real(1,i)=20*log10( ...
        mean(sqrt(y(1,ms_points(1,i):ms_points(2,i)).^2))/(2^15-1));
    levels_real(2,i)=20*log10(...
        mean(sqrt(y(2,ms_points(1,i):ms_points(2,i)).^2))/(2^15-1));
end

% Compute expected output levels for each input level with a
% simplified DRC model.
levels_exp=zeros(2,num_steps);

for i=1:num_steps
    % boost input signal at measurements points according to level
    % adjustments
    step_level_linear = 10 ^(level_adjust(i)/20);
    x_a = x_(:,ms_points(1,i): ms_points(2,i)).* step_level_linear;

    % input level estimation
    x_a_summed = sum(abs(x_a),1);
    inlevel = 20*log10(mean(x_a_summed) ./ (2^15 -1))

    % compute output level from input level and compression curve
    outlevel = compressor(inlevel,ccurve_in,ccurve_out, ...
                          figure_plotting)
    % compute drc gain
    drc_gain_log = outlevel - inlevel;
    drc_gain = 10 ^ (drc_gain_log/20)

    % apply linear drc gain to input signal
    y_exp = x_a .* drc_gain;

    % compute expected levels
    levels_exp(1,i)= 20*log10(mean(sqrt(y_exp(1,:).^2)')/(2^15-1))';
    levels_exp(2,i)= 20*log10(mean(sqrt(y_exp(2,:).^2)')/(2^15-1))';
end

% create matrix of realized, expected and error levels
levels=zeros(3,length(level_adjust));
levels(1,:)=levels_real(1,:);
levels(2,:)=levels_exp(1,:);
levels(3,:)=levels_exp(1,:)-levels_real(1,:);

% verification of results
ok = 1;
error = levels(3,:);
for i = 1:steps
    if abs(error(i)) > etol_db
        ok = 0;
    end
end
```

B.1.3 compressor.m

Calculates the output level *y* for the input level *x* according to compression curve relation f(curve_in, curve_out).

```
function y = compressor(x, curve_in, curve_out, figure_plotting)

% add -100 db point in the curves
curve_in = [-100,curve_in];
curve_out = [-100,curve_out];

N=length(curve_in);

% find linear section of a compression curve for given input level
for i=1:N-1
    if x > curve_in(i) & x <= curve_in(i+1)
        % compute the output level from the line equation y=y0+k*x
        a=(x-curve_in(i))/(curve_in(i+1)-curve_in(i));
        b=curve_out(i+1)-curve_out(i);
        y=curve_out(i)+a*b;
        break; % level found, no need to seek further
    end
end

% for debug mode, plot compression curve
if figure_plotting
    plot(curve_in,curve_out)
    hold on;
    plot([-100,0],[-100,0],'k--');
    hold off
    title('Compression curve 3');
    xlabel('Input level / dB');
    ylabel('Output level / dB');
    grid on;
end
```