Digital Audio Restoration of Gramophone Records

by

Christoph Frank Stallmann

Submitted in partial fulfillment of the requirements for the degree
Masters of Science (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

January 2015

Publication data:

Christoph Frank Stallmann. Digital Audio Restoration of Gramophone Records. Masters of Science, University of Pretoria, Department of Computer Science, Pretoria, South Africa, January 2015.

Digital Audio Restoration of Gramophone Records

by

Christoph Frank Stallmann
E-mail: cstallmann@cs.up.ac.za

Abstract

Gramophones were the main audio recording medium for more than seven decades and regained widespread popularity over the past few years. Being an analogue storage medium, gramophone records are subject to distortions caused by scratches, dust particles, high temperatures, excessive playback and other noise induced by mishandling the record. Due to the early adoption of the compact disc and other digital audio mediums, most research to reduce the noise on gramophone records focused on physical improvements such as the enhancements of turntable hardware, amelioration of the record material or advances through better record cutting techniques. Comparatively little research has been conducted to digitally filter and reconstruct distorted gramophone recordings.

This thesis provides an extensive analysis on the digital detection and reconstruction of noise in gramophone audio signals distorted by scratches. The ability to approximated audio signals was examined though an empirical analysis of different polynomials and time series models. The investigated polynomials include the standard, Fourier, Newton, Lagrange, Hermite, osculating and piecewise polynomials. Experiments were also conducted by applying autoregressive, moving average and heteroskedasticity models, such as the AR, MA, ARMA, ARIMA, ARCH and GARCH models. In addition, different variants of an artificial neural network were tested and compared to the time series models. Noise detection was performed using algorithms based on the standard score, median absolute deviation, Mahalanobis distance, nearest neighbour, mean absolute spectral deviation and the absolute predictive deviation method. The reconstruction process

employed the examined polynomials and models and also considered adjacent window, mirroring window, nearest neighbour, similarity, Lanczos and cosine interpolation.

The detection and reconstruction algorithms were benchmarked using a dataset of 800 songs from eight different genres. Simulations were conducted using artificially generated and real gramophone noise. The algorithms were compared according to their detection and reconstruction accuracy, the computational time needed and the tradeoff between the accuracy and time.

Empirical analysis showed that the highest noise detection accuracy was achieved with the absolute predictive deviation using an ARIMA model. The predictive outlier detector employing a Jordan simple recurrent artificial neural network was most efficient by achieving the best detection rate in a limited timespan. It was also found that the artificial neural networks reconstructed the audio signals more accurately than the other interpolation algorithms. The AR model was most efficient by achieving the best tradeoff between the execution time and the interpolation error.

Keywords: gramophone, audio reconstruction, noise detection, interpolation, polynomials, time series modelling, neural networks.

Supervisor : Prof. Andries P. Engelbrecht
Department : Department of Computer Science
Degree : Masters of Science

"I never made one of my discoveries through the process of rational thinking."

Albert Einstein

Acknowledgements

I would like to express my gratitude to the following people for their assistance during the production of this thesis:

- Professor A. P. Engelbrecht, my supervisor, for his insight, guidance and support;
- All CIRG members for their help and input;
- My family and friends who kept asking when I would finish my thesis;
- The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

From the early days of mankind, people were obsessed with preserving information for generations to come. Starting from early cave paintings and the formulation of alphabets, man started to develop means by which the current knowledge could be retained for future use. For centuries the only information that could be preserved was subject to human interpretation, such as sculpturing, painting or transcribing important events. The first breakthrough in capturing and preserving physical quantities through an analogue medium was with the invention of photography in the early 19^{th} century. Recording sound on the other hand proved to be more difficult. The first advances in this field were made by Édouard-Léon Scott de Martinville, printer and bookseller by trade, who was fascinated with the mechanic means for recording vocal sounds. Scott delivered the designs of his device, the *phonautograph*, to the French Academy of Sciences in 1857 [80]. A number of working devices were built with the primary intend to record sound and provide a visual representation, rather than reproducing the sound wave. Three decades later the poet Émile-Hortensius-Charles Cros invented the *paleophone*, the first device that would have been able to both record and playback sound. He submitted the designs to the French Academy of Sciences in 1877 [14]. Due to the meagre income as a poet, Cors was never able to produce a working model and eventually gave his ideas and designs to public domain free of charge. The same year Thomas Edison designed and produced the *phonograph*, the first working device that was able to record and playback sound [211]. His invention was so unexpected to the public that it earned him the nickname *The*

Wizard and propelled his scientific career within weeks. The original phonograph used a tinfoil cylinder, resulting in poor sound quality and limiting the number of playbacks to a few times. The device got more attention for practical use when Graham Bell improved it by amongst other things making use of a wax-covered cardboard cylinder as recording medium, which became known as the *graphophone*. The real breakthrough of sound recordings for industrial and personal use occurred when Emile Berliner marketed his *gramophone* in 1889, the forerunner of the modern turntable [264]. Berliner replaced the wax cylinders with a flat round disc which had an inferior sound quality, but was able to hold a four minute recording compared to the two minutes of a wax cylinder. The first gramophone records were made out of gutta-percha, a vulcanized natural latex. In the early 20^{th} century, some records were made from metal in order to increase the durability, but were discontinued due to the increased production costs and poor sound quality. Gramophone records were then produced from shellac, but with the increase in record sales and advancements in chemistry, the material was replaced by polyvinyl chloride, which led to the adoption of the widely used name *vinyl records*.

Due to the limited durability of record material, the audio fidelity decreases with each playback. To prevent the wearing-down of records, William Heine developed the first optical turntable in 1976, the *laserphone*, which used a helium-neon laser to read records without physical contact [123]. The research was extended by Robert Reis during his postgraduate studies in 1987 [212], which contributed to the development of the first commercial laser turntable, the *Finial LT-1*. The Finial project was later acquired by Birmingham Sound Reproducers and improved versions of the turntable are still produced and sold by ELP today. Between 2003 and 2006 researchers at the Lawrence Berkeley National Laboratories started the *IRENE* project which takes photographs of records and then reconstructs the sound wave by using image processing techniques [76]. The original project used a two-dimensional camera and was able to reconstruct physically broken records. Later enhancements used confocal scanning microscopy with a three-dimensional camera to improve the reproduced sound quality by being able to scan surfaces down to 200 µm [77].

Gramophone records served as the main recoding medium for more than six decades, replacing the phonograph cylinders in the 1920s. Although the compact cassette (CC)

was introduced in the 1970s, record sales remained stable, until the introduction of the compact disc (CD) in the late 1980s. In 2013 the Nielsen Company reported the highest vinyl record sales in the United States since it was discontinued as main music medium in 1993 [213]. Approximately six million units were sold in the Unites States only, an increase of 33% from 2012 and holding a market share above 2%. It was however reported that the Nielsen Company's statistics were an underestimate and that only 15% of the sales were logged, since most sold records do not have a bar code which is used to track sales [244].

1.1 Motivation

Gramophone records were discontinued as main recording medium in the late 1980s. Prior to this time only little research was conducted on digitally processing gramophone audio with the aim of improving the sound quality by mitigating the problems of durability, noise and an inferior listening experience. Most research focused on the analogue sound reproduction by improving the turntable mechanics and record material. With the introduction of the CD, the sound storage moved from analogue to digital, eliminating most of the noisy distortions of gramophone records. The majority of recordings prior to the 1960s are only available on gramophone, are rarely in mint condition and are often heavily damaged due to excessive use. Many of these recordings are still digitised today, both for archiving purposes and to make them publicly available. In addition, numerous private collectors and audiophiles are digitizing their music collection, which mostly consists of old and damaged records. The digitization requires a tedious manual process of removing the noise from the recording. Although some automation exists, the available algorithms are not specifically designed and fine-tuned for gramophone records. Additionally, most automated systems process the entire audio, therefore also adjusting parts of the sound wave that are not affected by noise.

1.2 Objectives

The main objective of this thesis is to create an accurate and efficient automated system for the detection and removal of distortions caused by scratches on gramophone records. This is achieved through the following sub-objectives:

- Detecting the noise prior to removing it in order to ensure that only distorted parts are subjected to the noise removal algorithms.
- Determining the ability of various polynomials and time series models to accurately approximate a music signal.
- Utilising the same polynomials and models to detect and correct distortions through time series prediction and interpolation.
- Determining an accurate algorithm for the detection of disruptions caused by scratches, by comparing outlier detectors commonly used in time series analysis.
- Determining an accurate interpolation algorithm to remove the previously detected noise, by comparing and benchmarking various interpolators used in time series analysis.
- Measuring the ability of adaptive systems such as an artificial neural network to learn temporal characteristics of music and comparing it to the traditional time series models.
- Correcting distorted music so that the human ear is unable to distinguish between the original and corrected recordings.

1.3 Contributions

The research in this thesis makes the following contributions to knowledge in the fields of time series analysis and audio processing:

- Provides an extensive benchmark of 22 different models that can be used for the interpolation of audio data.

- Demonstrates how certain models, which are commonly employed in financial markets with low model orders, can be adapted to use a higher order for the accurate approximations of more stable audio signals.
- Lays out the fundamental analysis and benchmarking that can be used for future research on the digitization of gramophone records and other fields of audio processing.
- Exemplifies the advantages of an adaptive neural network over traditional time series models for processing audio signals.
- Illustrates how the characteristics of different music genres can influence the noise detection and reconstruction process and provides recommendations for choosing an algorithm and accompanying parameters for certain genres.

In addition to the theoretical contributions, all algorithms were implemented as an open source project and two external libraries which are available for public use [234].

1.4 Restoration Approach

The audio restoration approach adopted by this thesis is outlined as follows:

1. **Audio Decoding and Preprocessing:** The audio data is decoded from a direct input audio signal or a file and prepared for the refurbishment phase.
2. **Noise Detection:** The noise in the sound wave is detected using an outlier detection algorithm. The detection is achieved by analysing sub-populations of the signal and determining the degree of divergence between different sub-populations and individuals within a given sub-population. The process is divided as follows:
   (a) **Noise Mapping:** A real map is generated, indicating the level of disturbance for each individual sample in the sound wave by comparing each sample to its surrounding values.

(b) **Noise Masking:** Using the previously generated noise map and a given threshold, a binary mask is generated to indicate if a sample contains noise and must be flagged as an outlier, or if no distortions are present and the sample is an inlier.

3. **Noise Removal:** The previously generated noise mask, where each sample in the sound wave is marked as either an outlier or an inlier, is utilized to determine which parts of the signal contain noise and must be refurbished. The process is divided as follows:

   (a) **Sample Approximation:** Given the samples preceding and succeeding a noisy segment, an interpolation algorithm is used to approximate the original non-noisy samples.

   (b) **Sample Replacement:** The noisy samples are replaced by the values of the approximated samples.

4. **Evaluation and Audio Encoding:** The restoration quality is determined and the audio signal is encoded and save to a file.

1.5 Thesis Outline

The remainder of this thesis is organized as follows:

- Chapter 2 provides a brief overview of digital audio storage, comprised of audio formats, codecs, sample rates and sizes, endianness and channels. The chapter continues with the gramophone mechanics, the dynamic range and frequency response of records. Some basic concepts of digital signal processing are highlighted which includes variances, convolution, correlations and least square regression.

- Chapter 3 discusses a number of polynomials and models used to approximate discrete functions. The underlying model mathematics are explained and some insight into the model order and parameter selection is provided.

- Chapter 4 explains a number of noise detection algorithms, including a predictive outlier identification that employed the models from chapter 3 to predict an audio signal.
- Chapter 5 examines a number of reconstruction algorithms, including models from chapter 3 that can be approximated for an audio signal and utilized for interpolation.
- Chapter 6 highlights the research methodology, provides an overview of the test dataset, explains how noise is introduced in the audio signals and describes the performance and execution time measurements for the examined algorithms.
- Chapter 7 presents and discusses the empirical results for all the algorithms described in chapters 3, 4 and 5. The chapter also highlights the optimal structure, model orders and parameters for the different algorithms.
- Chapter 8 concludes the thesis, provides a summary of the conducted research and highlights some important directions for future work.

Additional details are provided in the appendices and are organized as follows:

- Appendix A provides a list of the test data used for the empirical analysis in order to allow future researchers to replicate the results from this thesis.
- Appendix B lists the optimal parameter configurations for all examined algorithms.
- Appendix C serves as an extension to chapter 7, containing a detailed report on the noise detection results.
- Appendix D expands chapter 7 with more elaborate results generated by the interpolation algorithms.
- Appendix E provides a summary of the acronyms used in this thesis.
- Appendix F lists the important mathematical symbols and notations used throughout the chapters.
- Appendix G contains a list of publications derived from the research of this thesis.

Chapter 2

Gramophones and Audio Processing

The purpose of this chapter is to provide a background for the fundamentals of digital signal processing with a specific reference to the digital processing of audio recorded from gramophone records. Technical aspects of storing audio in a digital format is given, followed by an overview of the mechanics, dynamic range and frequency response of gramophone records and turntables. The basic mathematics used in digital signal processing is provided, which will be utilized by the algorithms in chapter 3, 4 and 5.

2.1 Digital Audio Formats

Computers use pulse code modulation (PCM) to represent and sample discrete values from a continuous signal. PCM samples the amplitude of a signal at uniform intervals and stores it as the nearest quantity in a predefined range. The quantization levels vary as a function of the amplitude, which are typically determined using the μ-law or A-law algorithms [191]. If the quantization levels are linearly uniform, it is known as linear pulse code modulation (LPCM). PCM is often used to describe LPCM data. Alternatives to PCM include pulse width modulation (PWM) and pulse position modulation (PPM) which are, however, rarely used in audio coding.

In order to work with a discrete signal on a software level, a hardware device that is capable of converting a continuous (analogue) signal to and from a discrete (digital) signal is needed. A device handling the forward conversion is called an analogue-to-digital

(ADC) and the backward conversion device is known as a digital-to-analogue (DAC). The ADC has two components, namely the sampler and quantizer which is illustrated in figure 2.1. The sampler captures the continuous signal, $f(t)$, at certain intervals, known as sample and hold in electronics, which is then used by the quantizer to generate a digital signal, $f_q(n)$, using discrete quantities.

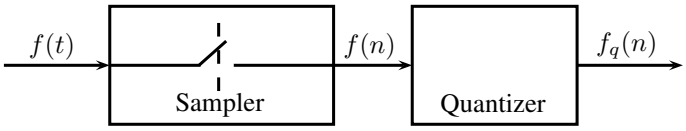


Figure 2.1: The sampler and quantizer of an ADC.

The rest of this section will discuss various factors that influence the structure, quality and storage size of digital audio data encoded with PCM. These factors include the codec, sample rate, sample type, sample size, endianness and the number of channels, each discussed in a subsection below.

2.1.1 Audio Codecs

On a software level an audio codec is a set of algorithms and procedures needed to convert a set of discrete samples into a compressed format (encoding) and vice versa by means of decompression (decoding). The main purpose of a codec is to reduce the storage size compared to the raw samples with limited or no reduction in fidelity and to add headers containing essential metadata needed to decode and play back the audio. Most audio codecs divide the original signal into chunks. If all chunks follow the same format and convention, only a single header is added to the audio file. If chunks have a different format, headers have to be added to each individual chunk. Although the latter approach has an increased storage size due to the additional headers, it allows each chunk to be compressed optimally according to the sample values which will reduce the overall file size.

Audio codecs can be classified into lossless and lossy compressions. Lossless compression allows the original signal to be fully reconstructed, whereas lossy compressions only approximates the original signal but has a smaller storage size.

One of the most well-known and oldest audio formats is the Waveform Audio File Format (WAVE or WAV) [181] which is part of Microsoft's and IBM's Resource Interchange File Format (RIFF) [182]. Although not an audio codec itself, WAVE can make use of other codecs to compress individual chunks. Most WAVE files do not use any compression, where samples are stored in raw PCM as a single sequential chunk with a single header, allowing lossless data storage. Drawbacks of WAVE files include the limitation of only two channels (extensions allow for multi-channel data), big file sizes (can be reduced by making use of other codecs) and a maximum file size of 4GB due to its use of 32 bit unsigned integers (an extension of the format makes use of 64 bit integers).

Lossless codecs include the Free Lossless Audio Codec (FLAC) [93], Apple Lossless Audio Codec (ALAC) [10] and the MPEG-4 Audio Lossless Coding (ALS) [163]. Some of the most well-known lossy codecs include MPEG-1 and MPEG-2 Audio Layer III (MP3) [117], Advanced Audio Coding (AAC) [207], Windows Media Audio (WMA) [183] and Ogg Vorbis (OGG) [92].

2.1.2 Sample Rate

The sample rate is the frequency at which PCM samples the amplitude and is determined by the number of samples per second measured in Hertz (Hz). The process of the sampler in figure 2.1 to convert a continuous signal $f(t)$ to a discrete signal $f(n)$ can be expressed mathematically as

$$f(n\nu) = f(t)\delta(t - n\nu) \tag{2.1}$$

where ν is the sample interval and δ the Dirac delta function defined as

$$\delta(t) = \begin{cases} 1 & \text{for } t = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

The sample rate affects the frequency response of the reproducing system. Higher rates have a higher frequency response, but also lead to larger file sizes.

Theorem 1: Nyquist-Shannon *A signal can be completely determined by a series of points that are sampled at a frequency at least twice the maximum frequency contained in the signal.*

Neither music instruments nor human speech exceed frequencies of 20 kHz, with most of them rarely exceeding 10 kHz [230]. The human ear is capable of observing frequencies between 20 Hz and 20 kHz [127], with most adults not able to hear frequencies above 15 kHz [100]. Taking the Nyquist-Shannon theorem into account, a signal with a maximum frequency ν can be fully reconstructed from a set of discrete points sampled at a frequency greater or equal to 2ν. The optimal frequency range for sampling music is therefore $2 \times 20kHz = 40kHz$. The sample rate is typically chosen somewhat higher than $40kHz$ to accommodate low-pass filters which prevent aliasing having a finite slope. Oversampling occurs when sampling rates are significantly higher than what is suggested by the Nyquist-Shannon theorem.

A rate of 44.1 kHz is the standard used to sample CD audio, downloadable music (MP3) and low quality videos (Video and Super Video CD) where both the video and audio data is sampled at 44.1 kHz. Another widely used sampling rate utilized by professional equipment is at 48 kHz, used for digital TV, DVD, mixing consoles and digital recording devices. 96 kHz and 192 kHz started to be used with the introduction of DVD-Audio, HD DVD and Blu-ray discs. Higher sampling rates can relax the low-pass filter design requirements in traditional DACs and ADCs, but are less important with modern converters. Although sampling rates above 50 kHz mostly do not improve the listening experience of humans, ultrasonic frequencies interact and modulate the audible part of the frequency spectrum, known as intermodulation.

2.1.3 Sample Type and Size

Besides the sample rate, the sample size or bit depth determines the quality of the PCM data. The quantizer in figure 2.1 takes the measurement from the sampler and generates quantities in a range defined by the bit depth. The sample size is the number of bits that are used to store a single value with ranges typically chosen as a power of two, that is 2^n. Although a large sample size increases the dynamic range of the system, it also

increases the file size. A linear increase in the sample size has an exponential increase in the dynamic range.

The signal-to-noise ratio (SNR) is the measurement of power between the desired signal and the background noise, defined as

$$\text{SNR} = \frac{\mathcal{P}_{signal}}{\mathcal{P}_{noise}} = \left(\frac{\mathcal{A}_{signal}}{\mathcal{A}_{noise}}\right)^2 \tag{2.3}$$

where $\mathcal{P}$ is the average power and $\mathcal{A}$ is the root mean square (RMS) amplitude. Since audio systems have a very wide dynamic range, the SNR is often expressed as a logarithmic decibel (dB) scale, calculated as

$$\text{SNR}_{\text{dB}} = 10\log_{10}\left(\frac{\mathcal{P}_{signal}}{\mathcal{P}_{noise}}\right) \text{dB} \tag{2.4}$$

The RMS of a signal y with n samples is given as

$$\text{RMS} = \sqrt{\sum_{i=1}^{n} \frac{y_i^2}{n}} \tag{2.5}$$

The crest factor $\mathcal{C}$ is the peak amplitude $\hat{\mathcal{A}}$ divided by the RMS of the signal. The peak-to-average-power ratio (PAPR) is the squared crest factor, defined as

$$\text{PAPR} = \mathcal{C}^2 = \left(\frac{\hat{\mathcal{A}}}{\text{RMS}}\right)^2 \tag{2.6}$$

The sample size limits the SNR to a maximum level which is determined by the quantization error. The ADC introduces a quantization error when digitizing the analogue voltage input, which is ideally a uniform distribution between $\pm\frac{1}{2}$ of the least significant bit. The average signal-to-quantization-noise ratio (SQNR) is calculated from the bit depth b [96] and measured in decibels as follows:

$$\text{SQNR}_{\text{avg}} \approx 6.02b + 4.77 - 10\log_{10}(\text{PAPR}) \text{ dB} \tag{2.7}$$

If the input is a sinusoid, the PAPR is equal to two, simplifying equation (2.7) to

$$\text{SQNR}_{\text{sin}} \approx 6.02b + 1.76 \text{ dB} \tag{2.8}$$

Using equation (2.8) above, the SQNR can be calculated for different sample sizes. Table 2.1 lists commonly used integer sample sizes with the corresponding unique values for the samples and the SQNR for the given bit depth.

Table 2.1: The SQNR and corresponding number of possible values for different sample sizes.

Bit Depth	Possible Unique Values	SQNR (dB)
4	16	25.84
8	256	49.92
16	65536	98.08
20	1048576	122.16
24	16777216	146.24
32	4294967296	194.40
64	18446744073709551616	387.04

Due to the limitations in modern integrated circuit design, most ADCs rarely exceed a SNR of 120 dB which is at a sample size just under 20 bits [237, 266]. Due to this limitation, most audio, including CDs, downloadable MP3s and DVDs, is sampled at 16 bits. Some DVDs adopted a 20 bit format with Blu-rays and HD audio making use of 24 bits. Bit depths beyond 24 bits are rarely utilized and mostly limited to professional recording equipment.

The sample type is directly related to the bit depth. Samples can be stored as integers (signed or unsigned) or decimal values. Most formats and codecs make use of unsigned 16 bit integers with a limited few codecs allowing decimal values to be stored. When digitally processing audio, the bit depth is dependent on the underlying hardware, operating system and programming language, and is typically restricted to primitive data types, namely *char* (8 bit), *short* (16 bit), *int* (32 bit), *long* (64 bit) and the decimal types *float* (32 bit) and *double* (64 bit). Digitally processing any bit depths in between the previously listed ones, either requires a storage overhead, since not all bits in a memory block are utilized, or requires the use of bit shifting, which reduces processing performance and is not available in all programming languages.

2.1.4 Endianness

Endianness or the bit order is the organization of 8 bit units (bytes) in memory. Although it does not have an affect on the audio quality, it is important for the direction of reading, writing and the processing of samples in memory. Computer memory is divided into units, each with a unique address, where the smallest assignable unit is 8 bits. Primitive data types (8, 16, 32 and 64 bits) are generally stored in consecutive memory addresses. The endianness determines in which order the individual bytes are store in a consecutive memory block. Big-endian systems store the most significant byte in the smallest address and use the last address for the least significant byte. Little-endian systems handle storage the opposite way, with the most and least significant bytes stored in the largest and smallest address respectively.

Little-endianness is known as the Intel convention, since the Intel x86 and x86-64 processors make use of this format. Big-endianness is used less often and known as the Motorola convention for being used on the Motorola 6800 and 68k processors. Bi-endian systems allow the switch between these two formats and amongst others include the ARMv3 (and above), PowerPC and Alpha architectures.

2.1.5 Channels

Channels refer to the individual recorded signals contained in a single audio track. Although very similar, the signals from different channels sampled at the same time during a multichannel recording still differ, since they were obtained from different sources in the same environment. The sources are typically microphones stationed at various places during a recording session. Signals contain more characteristics of the sound wave the closer they are placed the originating source.

Traditionally a single channel (mono) was used for recording audio. Music from modern sources are typically distributed with two channels (stereo). Although some music releases are available as multichannel data, surround sound systems using six or eight channels are mostly used for audio accompanying video data (Dolby Digital or DTS audio on DVD and Blu-ray discs). Multichannel data is typically stored as interleaving samples.

2.2 Gramophones

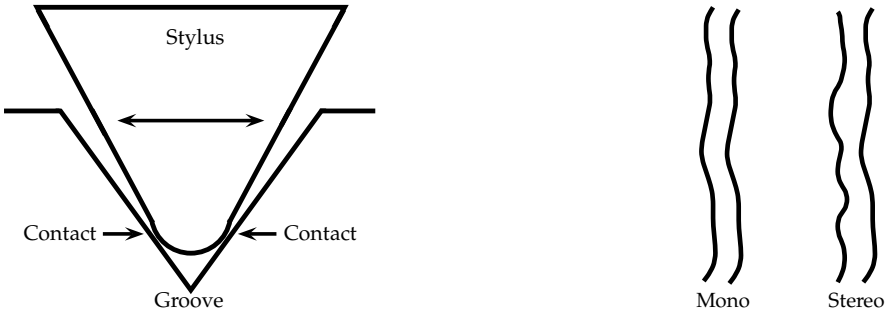
This section discusses the basic concepts of gramophone records, turntables, the dynamic range and frequency response of gramophone systems, and gives a broad overview of different types of noise affecting record playback.

2.2.1 Gramophone Mechanics

Gramophone records are flat discs with continuous grooves on both sides of the disc. The diameter of the grooves typically range from 20 µm to 80 µm and run from the periphery to the center of the disc. A cartridge with a needle is mounted at the end of the turntable arm that runs through the grooves, generating a sound wave by swivelling from side to side as illustrated in figure 2.2(a). Traditionally, these needles were made of metal which reduced the fidelity of the records and produced a poor sound quality. Modern cartridges have a more sophisticated stylus with a hardened point, typically a small diamond or sapphire. Although cartridges are only a small part of turntables, they are the component that influences the sound quality the most, with some selling for as much as $20,000 [228].

Standard gramophone records are produced in three different sizes, with diameters of 12 in (30.48 cm), 10 in (25.4 cm) and 7 in (17.78 cm). Playback of the records are done at rotational speeds of $33\frac{1}{3}$ rpm, 45 rpm and 78 rpm with 78 rpm records rarely being issued since the 1950s. The combination of the diameter and rational speed determines the time capacity and are generally divided into long-play (LP), typically between 30 to 50 minutes, extended-play (EP) holding about 25 minutes and the single-play (SP) with a typical capacity of under 10 minutes.

Monophonic records were the standard until the mid 1960s when the improvements of turntables replaced them with stereophonic records. Figure 2.2(b) illustrates that mono records have the same contours on both sides of the grooves, but differ on stereo records to accommodate the reproduction of two sound waves. In the early 1970s quadrophonic records were introduced and although without a commercial success, it provided the initial research for modern surround sound systems. Quadrophonic records did not have a separate source for each channel like modern multichannel audio, but appeared to



(a) The stylus swivels between the sides of the groove to generate a sound wave.

(b) The mono groove contours are the same on both sides, but differ on stereo records.

Figure 2.2: The workings and grooves of gramophone records.

have an ordinary stereo recording. Separate carriers with different frequency ranges were added to both sides of the groove and during playback the left and right channels were subtracted and combined to generate four channels of audio.

2.2.2 Dynamic Range of Gramophones

Since gramophone records are an analogue medium, the dynamic range is theoretically infinite. In practice, however, the dynamic range is physically limited by the groove width. The dynamic range improves with an increase in the groove width, but at the same time the playback duration of the record will decrease. A wider dynamic range that keeps the original duration can be accomplished by increasing both the groove width and placing grooves closer to each other. The drawback of this approach is that the gap between parallel grooves are smaller which can lead to an accelerated sound quality reduction, since the gap can completely disappear through extensive playback.

Between the 1960s and 1980s research was conducted to increase the dynamic range and reduce the inner-groove distortions. These methods improved disc cutting equipment and special playback equipment which was mostly incompatible with standard gramophones. The most notable projects in this field includes the Dynagroove, CBS DisComputer, Teldec Direct Metal Mastering and the dbx-encoded records. Due to the drawbacks and the small market niche, the projects had limited commercial success and were quickly

discontinued.

Since gramophones are analogue and therefore can reproduce a continuous signal which is only limited by the accuracy of the disc cutting equipment, little research has been done since the 1980s to improve the dynamic range. Audiophiles often argue that the sound quality of an analogue medium is better due to the infinite number of values of a continuous function, but when sampling at the optimal rate and taking the Nyquist-Shannon theorem into account, the human ear will not be able to distinguish between the quality of modern digital and analogue mediums.

2.2.3 Frequency Response of Gramophones

Gramophone records that are acoustically recorded have a frequency response from 168 to 2000 Hz [177]. Electronic recordings improved the range from 100 to 5000 Hz. The record itself has little effect on the frequency response, whereas the playback equipment, especially the stylus cartridge, has the main impact that determines the frequency response. With the introduction of quadrophonic records such as the Compatible Discrete 4 (CD-4), sub-carriers used special frequency modulation-phase modulation-single sideband frequency modulation (FM-PM-SSBFM) to increase the maximum frequencies to 45 kHz. Inexpensive modern cartridges typically have a minimum frequency response close to that of the human ear at 20 Hz to 20 kHz, but often go up to 50 kHz. Expensive cartridges have a response from 20 Hz to 100 kHz [228]. Since human speech and instruments do not exceed this range, modern gramophones, turntables and cartridges are able to capture and reproduce most frequencies present in natural speech and music.

2.2.4 Gramophone Noise and Distortions

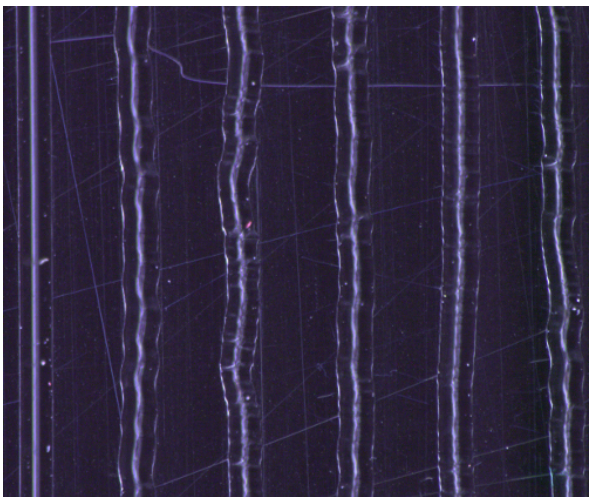
Noise from gramophone records can be caused by various external factors, such as scratches, dust particles, extensive playback, sunlight and heat, improper storage, mechanical imbalances and room vibrations. Figure 2.3(a) is a magnification of an used record with some minor scratches which will have little effect on the reproduced sound wave. Dust particles and a microscopic thread can be seen in figure 2.3(b). Figures 2.3(c) and 2.3(d) show a wide scratch which will cause a substantial disruption during playback. Noise,

such as small dust particles, mechanical or room vibrations has a minor impact on the sound wave, often only on the subsonic frequencies, which makes the noise difficult to detect. Since these frequencies can not be observed by the human ear, they will not be further discussed in this thesis. Other kinds of noise such as extensive playback or the warping of the record as a result of sunlight, has a serious impact on the entire sound wave. Since almost all samples are disrupted, it is difficult to mathematically reconstruct this kind of noise. In addition, playing these records can sometimes damage the turntable and stylus and are therefore often discarded by their owners. This thesis focuses on the noise that is caused by scratches, often referred to as *crackles* and *pops*. Scratches causes most of the observable disruptions during gramophone playback and since only a part of the sound wave is affected, can be mathematically reconstructed with the unaffected samples.

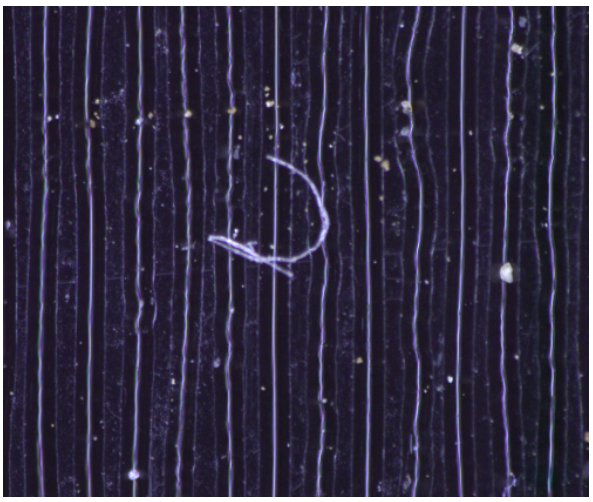
2.3 Digital Signal Processing Basics

Digital signal processing (DSP) is the field of mathematics that deals with the manipulation of discrete time or discrete frequency signals that were often obtained from continuous signals in order to improve, modify or extract information from the signal. This section discusses some basic concepts and mathematics of DSP which are being used by various models and algorithms in chapters 3, 4 and 5.

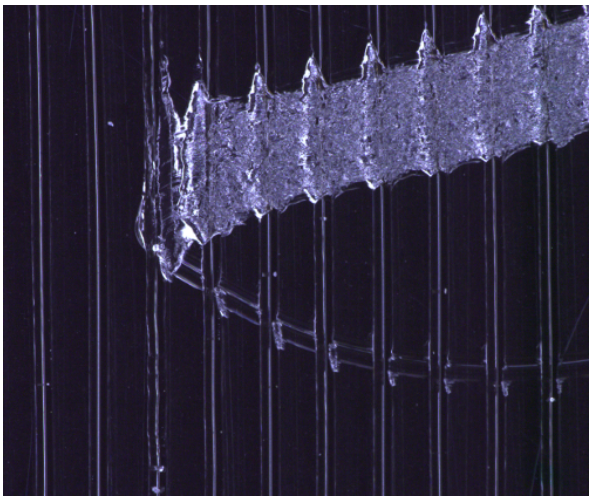
In statistics a sample refers to a set of observations collected from a larger population. In audio processing a sample refers to a single observation holding a value within a discrete range. Throughout this thesis a sample, observation or data point denotes a single discrete value and a collection will be referred to as a sample set. In statistics it is also important to distinguish between the population and sample mean, variance, standard deviation and all related concepts. When working with the population, all observations from the signal are used. In contrast, the sample mean and variance only use observations from a subset of the population. Since it makes little sense to use the entire population when processing a song, the mean, variance and related concepts are assumed to be calculated from a sample subset and not the entire signal. A discrete sample set is denoted as y with n samples which are represented by the sample values y_i



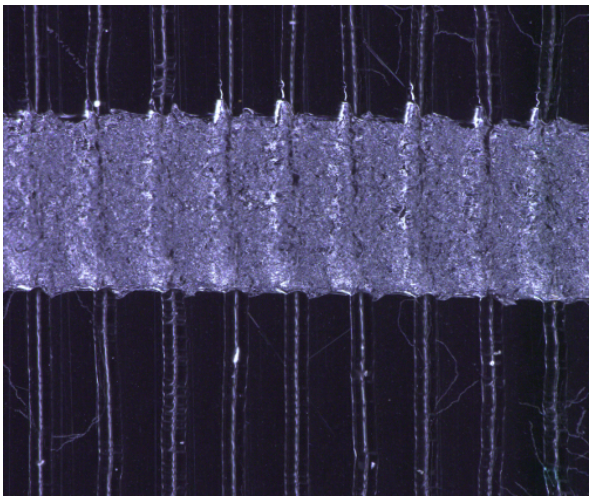
(a) A clean record with some minor scratches across the grooves.



(b) A record with dust particles and a small thread.



(c) A record with a scratch only affecting some grooves.



(d) A record with a large scratch affecting multiple grooves.

(e) The cross section of a record.

Figure 2.3: Magnified gramophone records distorted by scratches and dust particles.

for $i \in \{1, 2, \ldots, n\}$. x_i refers to the time delay of the i^{th} sample.

2.3.1 Time and Frequency Domains

An audio time-domain signal stored as PCM is plotted with the amplitude of the sound wave against the time. The time-domain therefore illustrates how the signal's values change over time. For instance, a three minute song encoded at 44.1 kHz will contain 7938000 samples in each channel, where the samples are quantized approximately every 22.7 µs.

A time-domain signal can also be represented in the frequency-domain, known as a frequency spectrum. Values are usually plotted as either amplitude or phase against frequencies. The Fourier transform (FT) can be used to translate a integrable time-domain function, $f(x)$, into the frequency-domain. The FT is defined as

$$\mathfrak{f}(\xi) = \int_{-\infty}^{\infty} f(x)\ e^{-2j\pi x\xi}\, dx \tag{2.9}$$

where $\mathfrak{f}(\xi)$ is the amplitude or phase of the signal at a given frequency ξ. The frequency spectrum can be translated back into the time-domain, known as the Fourier inversion theorem [94], which was proven by Titchmarsh more than a century later [242]. The inverse Fourier transform (IFT) is defined as

$$f(x) = \int_{-\infty}^{\infty} \mathfrak{f}(\xi)\ e^{2j\pi\xi x}\, d\xi \tag{2.10}$$

The discrete Fourier transform (DFT) is used to transform the discrete series y with a finite number of samples, n, into the frequency-domain as follows:

$$\mathfrak{f}_\xi = \sum_{i=1}^{n} y_i e^{-2j\pi\xi i/n} \tag{2.11}$$

Since the time-domain signal is discrete, the resulting spectrum also contains discrete frequencies. The discrete finite frequency spectrum of the DFT stands in contrast to the discrete time Fourier transform (DTFT) which produces a continues periodic frequency spectrum from an discrete infinite input signal. DFTs are computationally expensive. The fast Fourier transform (FFT) is an algorithm to efficiently compute the DFT and its

inverse by factorizing the DFT matrix into a product of sparse factors. Although the idea of FFTs already existed from 1805, it did not gain widespread acceptance until 1965 with the Cooley-Tukey algorithm [46] during the popularization of computers.

2.3.2 Least Squares Regression

Least squares regression approximates the unknowns of an overdetermined system, that is a system with more equations than unknowns. Attributed to Gauss [102], a least square fit aims to find the set of coefficients that will minimize the sum of squared errors between the observed and approximated values. Least squares regression can be categorized into linear least squares (LLS) and nonlinear least squares (NLS) regression. LLS, also referred to as ordinary least squares, has a closed-form solution, meaning that the system can be solved in a finite number of operations. LLS is globally concave and will converge to a unique solution. NLS on the other hand has no closed-form solution, requiring an iterative process to improve the current solution. Non-convergence is a common problem in NLS, since there might be multiple minima for the sum of squares. LLS is discussed below, but NLS is left for further reading [15].

A system of linear equations can be expressed in matrix form as follows:

$$\mathbf{X}\boldsymbol{\alpha} = \mathbf{y} \tag{2.12}$$

where $\mathbf{X}$ is a matrix that holds the expressions, $\mathbf{y}$ a vector that contains the solutions to the expressions and $\boldsymbol{\alpha}$ the unknown coefficients that have to be approximated. For a discrete function with a set of n data points (x_i, y_i) for $i \in \{1, 2, \ldots, n\}$, matrix $\mathbf{X}$ is typically constructed with a routine that uses the x_i values and vector $\mathbf{y}$ with the y_i values. The objective function for the regression with a linear combination of the coefficients measures the difference between the observed and approximated values, defined as

$$\mathcal{S}(\boldsymbol{\alpha}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\alpha}\|^2 = \sum_{i=1}^{n} \left[y_i - \sum_{j=1}^{d} X_{ij}\alpha_j \right]^2 \tag{2.13}$$

where d is the number of coefficients, corresponding to the number of columns in $\mathbf{X}$. In a later publication by Gauss [103], it was suggested that the Laplace mean absolute error should be used instead of the sum of squared errors, defined as

$$\mathcal{M}(\boldsymbol{\alpha}) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \sum_{j=1}^{d} X_{ij}\alpha_j\right| \tag{2.14}$$

Since $\mathcal{M}(\boldsymbol{\alpha})$ requires an additional division operation and the absolute value operation is slower than calculating the squares by multiplication, $\mathcal{S}(\boldsymbol{\alpha})$ is typically used as objective function for LLS in modern computer systems. The objective of LLS is to find the set of parameters $\check{\boldsymbol{\alpha}}$ that best fit the given data, therefore minimizing the sum of squares. This objective can be formulated as

$$\check{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \mathcal{S}(\boldsymbol{\alpha}) \tag{2.15}$$

The minimization problem can then be solved by

$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})\check{\boldsymbol{\alpha}} = \mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{2.16}$$

where $\mathbf{X}^{\mathrm{T}}$ is the transpose of matrix $\mathbf{X}$. In order to solve the coefficients, equation (2.16) is rearranged as

$$\check{\boldsymbol{\alpha}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{2.17}$$

which is easily computed by matrix transposition, inversion and multiplication. Equation (2.17) is also expressed as

$$\check{\boldsymbol{\alpha}} = \mathbf{X}^{+}\mathbf{y} \tag{2.18}$$

where $\mathbf{X}^{+}$ is the Moore-Penrose pseudoinverse of $\mathbf{X}$, which is a generalization of the inverse matrix, calculated using singular value decomposition (SVD) [24, 186, 205]. Inverting a matrix with normal equations is computationally expensive. A more efficient approach can be used when the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ from equation (2.17) is positive definite and well-conditioned. In such a case, the normal equations are directly solved with a Cholesky decomposition [18], defined as

$$\mathbf{C}^{\mathrm{T}}\mathbf{C}\check{\boldsymbol{\alpha}} = \mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{2.19}$$

where $\mathbf{C}$ is an upper triangular matrix. The first step in the Cholesky decomposition is to solve $\boldsymbol{\omega}$ by forward substitution, in other words

$$\mathbf{C}^{\mathrm{T}}\boldsymbol{\omega} = \mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{2.20}$$

The second step uses $\boldsymbol{\omega}$ to solve $\check{\boldsymbol{\alpha}}$ through backward substitution, that is

$$\mathbf{C}\check{\boldsymbol{\alpha}} = \boldsymbol{\omega} \tag{2.21}$$

Besides the normal equations, other alternatives exists for solving the least squares problem, such as QR decomposition. QR decomposition is the basis of the QR transform, an eigenvalue algorithm independently developed by Francis [97, 98] and Kublanovskaya [155]. It is an orthogonal method that decomposes the matrix $\mathbf{X}$ into a product

$$\mathbf{X} = \mathbf{QR} \tag{2.22}$$

where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ an upper triangular matrix [8]. Although QR decomposition is computationally more expensive than normal equations, it is numerically more stable since it is not required to evaluate $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. QR decomposition may still fail if the matrix $\mathbf{X}$ is nearly rank deficient, meaning that $\mathbf{X}$ has a rank close to the largest possible value [257]. Another orthogonal alternative is SVD where $\mathbf{X}$ is decomposed into a product,

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{*} \tag{2.23}$$

where $\mathbf{U}$ is an orthogonal matrix, $\boldsymbol{\Sigma}$ a rectangular diagonal matrix with a non-negative diagonal, and $\mathbf{V}^{*}$ the conjugate transpose of an orthogonal matrix $\mathbf{V}$ [116]. SVD is even more computationally expensive than the QR approach, but is numerically stable and can handle rank deficiency [159].

2.3.3 Expected Value

The expected value is the weighted average of all possible values, where each value is multiplied by the probability of the event occurring. Essentially, the expected value can be seen as the calculated average of the outcome of an experiment repeated indefinitely,

where each outcome is assigned a probability of occurrence. The expected value of a signal y is defined as

$$\begin{aligned} \mathrm{E}[y] &= \frac{y_1 p_1 + y_2 p_2 + \cdots + y_n p_n}{p_1 + p_2 + \cdots + p_n} \\ &= \frac{\sum_{i=1}^{n} y_i p_i}{\sum_{i=1}^{n} p_i} \end{aligned} \tag{2.24}$$

where p_i are the probabilities. Since the probabilities should add up to one, the equation is simplified to

$$\begin{aligned} \mathrm{E}[y] &= y_1 p_1 + y_2 p_2 + \cdots + y_n p_n \\ &= \sum_{i=1}^{n} y_i p_i \end{aligned} \tag{2.25}$$

The expected value of a $m \times n$ matrix $\mathbf{Y}$ is written as

$$\mathrm{E}\left[\mathbf{Y}\right] = \begin{bmatrix} \mathrm{E}\left[Y_{1,1}\right] & \mathrm{E}\left[Y_{1,2}\right] & \cdots & \mathrm{E}\left[Y_{1,n}\right] \\ \mathrm{E}\left[Y_{2,1}\right] & \mathrm{E}\left[Y_{2,2}\right] & \cdots & \mathrm{E}\left[Y_{2,n}\right] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}\left[Y_{m,1}\right] & \mathrm{E}\left[Y_{m,2}\right] & \cdots & \mathrm{E}\left[Y_{m,n}\right] \end{bmatrix} \tag{2.26}$$

2.3.4 Variance

Variance is the measurement of how much the samples are spread out. The population variance is defined as

$$\begin{aligned} \acute{\sigma}^2 &= \mathrm{E}\left[(y - \mathrm{E}\left[y\right])^2\right] \\ &= \mathrm{E}\left[y^2\right] - \left(\mathrm{E}\left[y\right]\right)^2 \end{aligned} \tag{2.27}$$

The expected value of y is represented by the population mean μ_y of y. Using a subset of the population y with a known sample mean $\bar{y}$, the sample variance is calculated using

$$\acute{\sigma}_y^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{2.28}$$

Equation (2.28) is biased, since it does not correspond to the degree of freedom in the vector of residuals for the sample set. Applying Bessel's correction [148], the unbiased sample variance is given by

$$\operatorname{var}(y) = \sigma_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{2.29}$$

The unbiased sample variance is used in this thesis when referring to the variance, except if otherwise stated.

2.3.5 Covariance

Covariance is the measurement of the correlation strength between multiple random variables. Given two random variates y and z, the population covariance is defined as

$$\begin{aligned}\operatorname{cov}(y,z) &= \mathrm{E}\left[(y - \mathrm{E}[y])(z - \mathrm{E}[z])\right] \\ &= \mathrm{E}[yz] - \mathrm{E}[y]\,\mathrm{E}[z]\end{aligned} \tag{2.30}$$

If the covariance $\operatorname{cov}(y,z)$ is positive, z increases with an increase in y. If the covariance is negative, z decreases as y increases. As with the variance, there is a biased and unbiased covariance. The unbiased sample covariance is used in this thesis, except if otherwise stated. When replacing the expectation of y and z with their respective sample means, $\bar{y}$ and $\bar{z}$, the unbiased sample covariance is calculated by

$$\operatorname{cov}(y,z) = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})(z_i - \bar{z}) \tag{2.31}$$

where y and z both have n number of samples. A special case exists when calculating the covariance with the same random variate. If y and z are equal, the covariance simplifies to the variance of y as follows:

$$\operatorname{cov}(y,y) = \operatorname{cov}(y) = \mathrm{E}\left[y^2\right] - (\mathrm{E}[y])^2 = \operatorname{var}(y) \tag{2.32}$$

The covariance matrix of random vectors $\mathbf{y}$ and $\mathbf{z}$ is calculated by

$$\begin{aligned}\operatorname{cov}(\mathbf{y},\mathbf{z}) &= \mathrm{E}\left[(\mathbf{y}-\mathrm{E}[\mathbf{y}])\,(\mathbf{z}-\mathrm{E}[\mathbf{z}])^{\mathrm{T}}\right] \\ &= \mathrm{E}\left[\mathbf{y}\mathbf{z}^{\mathrm{T}}\right]-\mathrm{E}[\mathbf{y}]\,\mathrm{E}[\mathbf{z}]^{\mathrm{T}}\end{aligned} \tag{2.33}$$

The relationship between the vectors $\mathbf{y}$ and $\mathbf{z}$ is more explicitly represented as the population covariance matrix, given as

$$\operatorname{cov}(\mathbf{y},\mathbf{z}) = \begin{bmatrix} \mathrm{E}[(y_1-\mu_y)\,(z_1-\mu_z)] & \cdots & \mathrm{E}[(y_1-\mu_y)\,(z_m-\mu_z)] \\ \mathrm{E}[(y_2-\mu_y)\,(z_1-\mu_z)] & \cdots & \mathrm{E}[(y_2-\mu_y)\,(z_m-\mu_z)] \\ \vdots & \ddots & \vdots \\ \mathrm{E}[(y_n-\mu_y)\,(z_1-\mu_z)] & \cdots & \mathrm{E}[(y_n-\mu_y)\,(z_m-\mu_z)] \end{bmatrix} \tag{2.34}$$

with n and m the size of vectors $\mathbf{y}$ and $\mathbf{z}$ respectively. The mean vector is often referred to as the centroid and the covariance matrix as the dispersion.

2.3.6 Autocovariance

Autocovariance is the covariance of a random variate with a time-shifted version of itself. If two samples are selected at time t and s respectively and the variate expectation is the population mean at the given times, the population autocovariance is defined as

$$\operatorname{acov}(y_t, y_s) = \mathrm{E}[(y_t-\mu_t)\,(y_s-\mu_s)] \tag{2.35}$$

The corresponding unbiased sample autocovariance is calculated using

$$\operatorname{acov}(y_t, y_s) = \frac{1}{n-1}\sum_{i=1}^{n}(y_{t+i}-\bar{y}_t)\,(y_{s+i}-\bar{y}_s) \tag{2.36}$$

where $\bar{y}_t$ and $\bar{y}_s$ are the sample means of y at time delay t and s respectively. The covariance matrix in equation (2.34) for a time-shifted version of vector $\mathbf{y}$ has a diagonal equal to the variance of $\mathbf{y}$ at the intervals specified by the diagonal. This matrix is known as the variance-covariance matrix and is computed using the variances and covariances as follows:

$$\text{cov}(\mathbf{y}) = \begin{bmatrix} \text{var}(y_1) & \text{cov}(y_1, y_2) & \cdots & \text{cov}(y_1, y_n) \\ \text{cov}(y_2, y_1) & \text{var}(y_2) & \cdots & \text{cov}(y_2, y_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(y_n, y_1) & \text{cov}(y_n, y_2) & \cdots & \text{var}(y_n) \end{bmatrix} \tag{2.37}$$

Since it makes little sense in DSP to compute the matrix in equation (2.34) with two different independent vectors, the terms covariance and variance-covariance matrix are used interchangeably when computed on a time-shifted version of the same signal. If the joint probability distribution does not change if the time is shifted by τ, y is a stochastic process with $\mu_t = \mu_s$. In the case of a stochastic process with $\tau = s - t$, equation (2.35) simplifies to

$$\begin{aligned} \text{acov}(y_t, y_{t-\tau}) &= \text{E}\left[(x_t - \mu_y)\,(y_{t-\tau} - \mu_y)\right] \\ &= \text{E}\left[y_t y_{t-\tau}\right] - \mu_y^2 \end{aligned} \tag{2.38}$$

The unbiased sample autocovariance given in equation (2.36) is used in this thesis, except if otherwise stated.

2.3.7 Standard Deviation

The standard deviation is the measurement of how much the samples variate from the mean and is calculated as the square root of the population variance, defined as

$$\sigma = \sqrt{\text{E}\left[(y - \mu_y)^2\right]} \tag{2.39}$$

The uncorrected sample standard deviation is calculated using

$$\acute{\sigma}_y = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.40}$$

Applying Bessel's correction, equation (2.40) is rewritten as the corrected sample standard deviation, given as

$$\sigma_y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.41}$$

The bias of the variance in equation (2.41) was removed, but the bias of the square root still exists. According to Jensen's inequality [144], since the square root is a concave function, additional adjustments have to be applied for an unbiased estimation that depends on the distribution of the sample set. The difference between the corrected and uncorrected standard deviation is only substantial for small sample sets and becomes less significant as n increases. An unbiased version that takes Jensen's inequality into account has an even less significant influence on the estimation. Since the unbiased standard deviation requires the underlying distribution of the sample set to be known, which is difficult to determine for a random processes, it is rarely used in DSP. The corrected sample standard deviation is used in this thesis when referring to the standard deviation, except if otherwise stated.

2.3.8 Convolution

Convolution is an integral that measures the amount of overlap when a function g is shifted over another function f. Shifting g over f, which are both objects in the algebra of Schwartz functions in $\mathbb{R}^n$, will produce a third function from the product of g and f that represents the overlapping area between the two functions. The convolution of f and g over a finite range $[0, t]$ is expressed as an integral transform as follows:

$$(f * g)(t) = \int_0^t f(\tau)g(t-\tau)d\tau \tag{2.42}$$

Using discrete functions y and z, the discrete convolution is defined as

$$\text{con}(y, z) = (y * z)_t = \sum_{\tau=0}^{n} y_\tau z_{t-\tau} \tag{2.43}$$

The convolution in equation (2.43) assumes causality. A system is said to be causal if its outputs are only depended on the current and past inputs and do not depend on future inputs. Due to the forward processing of audio signals, this thesis utilizes causal convolution, except if stated otherwise.

The symmetric properties of convolution allow the time delay τ to be applied to either one of the functions. Since both y and z have a finite number of samples, the convolution can only be calculated if $t \geq 0$. Entries outside the respective ranges of y and z are zero.

The convolution will therefore also be zero if $t < 0$, since one of the signals will have a value of zero at the given time delay τ.

2.3.9 Crosscorrelation

Crosscorrelation is the measurement of similarity between two functions and is similar in nature to convolution. The discrete crosscorrelation is given as

$$(f \star g)_t = \sum_{\tau=-\infty}^{\infty} y_\tau^* z_{t-\tau} \tag{2.44}$$

where y^* represents the complex conjugate of y. Crosscorrelation is used to determine the time delay between two similar signals. The signals are aligned by iteratively increasing the delay until the crosscorrelation is maximized. The time delay τ between y and z is calculated in number of samples as follows:

$$\tau = \arg\max_t (y \star z)_t \tag{2.45}$$

The Pearson product-moment correlation coefficient, or simply the correlation coefficient, is the measure of the linear correlation between two variables. The correlation coefficient is the commonly used to determine how closely y matches z and is calculated by

$$\begin{aligned} \operatorname{cor}(y,z) &= \frac{\mathrm{E}\left[(y-\mu_y)(z-\mu_z)\right]}{\sqrt{\mathrm{E}\left[(y-\mu_y)^2\right]}\sqrt{\mathrm{E}\left[(z-\mu_z)^2\right]}} \\ &= \frac{\sum_{i=1}^{n}(y_i-\mu_y)(z_i-\mu_z)}{\sqrt{\sum_{i=1}^{n}(y_i-\mu_y)^2\sum_{i=1}^{n}(z_i-\mu_z)^2}} \end{aligned} \tag{2.46}$$

Equation (2.46) is the normalized covariance, that is the covariance between y and z divided by the product of their standard deviations. Like convolution and covariance, crosscorrelation is symmetric, making $\operatorname{corr}(y,z)$ equivalent to $\operatorname{corr}(z,y)$.

The crosscorrelation matrix of vector $\mathbf{y}$ is represented in matrix form as

$$\text{cor}(\mathbf{y}) = \begin{bmatrix} 1 & \text{cov}(y_1, y_2) & \cdots & \text{cov}(y_1, y_n) \\ \text{cov}(y_2, y_1) & 1 & \cdots & \text{cov}(y_2, y_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(y_n, y_1) & \text{cov}(y_n, y_2) & \cdots & 1 \end{bmatrix} \tag{2.47}$$

which is the variance-covariance matrix in equation (2.37) with the variance on the diagonal equal to one.

2.3.10 Autocorrelation

Autocorrelation is the crosscorrelation of a signal with itself at a given time delay. Given the sample mean μ_t and μ_s at time delay t and s respectively, the biased autocorrelation of signal y with n data points is defined as

$$\text{acor}(y_s, y_t) = \frac{\text{E}\left[(y_t - \mu_t)\,(y_s - \mu_s)\right]}{\sqrt{\text{E}\left[(y_t - \mu_t)^2\right]}\sqrt{\text{E}\left[(y_s - \mu_s)^2\right]}} \tag{2.48}$$

If y is stochastic where the mean and standard deviation are time-independent, the autocorrelation simplifies to the normalized autocovariance, given as

$$\begin{aligned} \text{acor}(y_t, y_{t-\tau}) &= \frac{\text{E}\left[(y_t - \mu_y)\,(y_{t-\tau} - \mu_y)\right]}{\text{E}\left[(y - \mu_y)^2\right]} \\ &= \frac{\text{acov}(y_t, y_{t-\tau})}{\text{var}(y)} \end{aligned} \tag{2.49}$$

Although equation (2.48) is a biased estimator, it is still asymptotically unbiased. Alternatively, the autocorrelation can be calculated by

$$\text{acor}(y) = \frac{\sum_{i=1}^{n-t}(x_i - \mu)(y_{i+t} - \mu)}{(n-t)} \tag{2.50}$$

which has a lower bias than equation (2.48), but is susceptible to a higher mean square error [143].

2.3.11 Partial Autocorrelation

Partial autocorrelation was introduced for the order identification of Box-Jenkins models [29]. Partial autocorrelation is the autocorrelation between observation y_i and y_{i+t} with the linear dependency from intervening sample y_{i+1} through to y_{i+t-1} removed. Unlike the normal autocorrelation, partial autocorrelation is not accounted for, for lags 1 through to $t-1$. The partial autocorrelation sequence can be solved by calculating the autocorrelations, inserting the autocorrelations into the Yule-Walker equations and then determining the best linear projection by solving the system of Yule-Walker equations through LLS regression. In practice, the application of the Levinson-Durbin recursion [60, 161] is more efficient and is calculated with two recursive equations where the first partial autocorrelation is equal to the first autocorrelation, that is

$$\text{pac}_{1,1}(y) = \text{acor}_1(y) \tag{2.51}$$

Furthermore, the partial autocorrelation for $i \in \{1, 2, \ldots, t-1\}$ using the Levinson-Durbin approach is calculated using

$$\text{pac}_{t,i}(y) = \text{pac}_{t-1,i}(y) - \text{pac}_{t,t}(y)\text{pac}_{t-1,t-i}(y) \tag{2.52}$$

The rest of the partial autocorrelations for $t > 2$ are determined by

$$\text{pac}_{t,t}(y) = \frac{\text{acor}_t(y) - \sum_{i=1}^{t-1} \text{pac}_{t-1,i}(y)\text{acor}_{t-i}(y)}{1 - \sum_{i=1}^{t-1} \text{pac}_{t-1,i}(y)\text{acor}_i(y)} \tag{2.53}$$

2.3.12 Pearson Correlation Coefficient

The Pearson correlation coefficient (PCC) is a statistical measurement of the linear dependency between two variables x and y [101, 204]. The linear relationship is in $[-1, 1]$, where zero indicates that no linear correlation exists. A PCC of positive one describes a perfect linear correlation, such that if x increases, y increases as well. A PCC of negative one implies that if x increases, y decreases. The sample PCC is defined as

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.54}$$

for a sample size of n and $\bar{x}$ and $\bar{y}$ the sample mean of x and y respectively. The confidence intervals for Pearson's r is based on Fisher's r-to-z transform [83, 85], defined as

$$z = \frac{1}{2} \ln \left(\frac{1+r}{1-r} \right) \tag{2.55}$$

If the pairs (x_i, y_i) are independent variables and (x, y) has a bivariant normal distribution, then z is approximately normally distributed with a mean

$$\mu_z = \frac{1}{2} \ln \left(\frac{1+\rho}{1-\rho} \right) \tag{2.56}$$

where ρ is the population correlation of which r is an estimate. Under a bivariant normal distribution, z has a standard error of

$$se_z = \frac{1}{\sqrt{n-3}} \tag{2.57}$$

The 95% confidence levels in the z-space are then calculated using

$$c_z = z \pm 1.96 se_z \tag{2.58}$$

To determine the confidence levels in the r-space, the inverse of the z-transformed is used as follows:

$$c_r = \frac{e^{2c_z} - 1}{e^{2c_z} + 1} \tag{2.59}$$

2.4 Summary

This chapter provided a broad overview of how audio data is stored in a digital format with reference to codecs, sample rates, sample sizes, endianess and the number of channels. A clarification on the mechanics, playback, dynamic range and frequency response of gramophone records and turntables was given, as well as the noise causing disruptions during playback. The difference between the time- and frequency-domain was highlighted and the mathematical basis of LLS was given. Fundamental DSP mathematics were discussed, including variance, convolution, correlation and Pearson's correlation coefficient.

Chapter 3

Signal Modelling

Signal modelling is the process of mathematical representing a signal, $f(x)$, as a model, $m(x)$, with a set of parameters, such that

$$m(x) \cong f(x) \tag{3.1}$$

Complex signals such as audio data are difficult to model perfectly and the parameters are therefore only an approximation of the true signal. Depending on the characteristics of the signal, the nature of the mathematical model and the chosen parameters, certain parts of a fluctuating signal might be modelled with high accuracy. However, using the same model on a different part of the signal might result in a very poor approximation. The process of modelling a discrete signal typically involves the selection of a model with a set of coefficients in such a way that it runs through all given discrete points as closely as possible. Amongst other things, models are used for signal compression, prediction, interpolation, pattern recognition, and the characteristics analysis of signals.

This chapter outlines the basics of a number of functions that can be used to model audio data. Linear models include the standard, Fourier, Newton and Lagrange polynomials which can also be employed using osculating, Hermite and piecewise polynomials. Models that can be applied in a linear and non-linear fashion include the autoregressive, moving average, autoregressive moving average, autoregressive integrated moving average, autoregressive conditional heteroskedasticity, generalized autoregressive conditional heteroskedasticity models and artificial neural networks. The theoretical basis

for the given models is presented in this chapter, with their application for extrapolation given in chapter 4 and interpolation given in chapter 5.

3.1 Standard Polynomials

A polynomial is a mathematical expression of a set of terms, each term consisting of a variable and a coefficient. In the standard form a polynomial is the sum of terms where the variables only have non-negative integer exponents. A standard polynomial (STP) is expressed as

$$m_{stp}(x) = \alpha_d x^d + \alpha_{d-1} x^{d-1} + \cdots + \alpha_2 x^2 + \alpha_1 x + \alpha_0 = \sum_{i=0}^{d} \alpha_i x^i \tag{3.2}$$

where x represent the variable of interest, α_i the coefficients, and d the degree of the polynomial. For audio data, x represents the time delay of the samples y. A unique polynomial is guaranteed for the given set of points according to the unisolvence theorem [147].

Theorem 2: Unisolvence *Given any set of $n+1$ points (x_i, y_i), a unique polynomial m with a degree of n or lower exists such that $m(x_i) = y_i$ for $i \in \{0, 1, \ldots, n\}$ with no two x_i the same.*

Considering the unisolvence theorem, the degree d must at least be n to ensure a unique polynomial. For larger datasets, however, it is impractical to have a polynomial of degree n, due to the computational time needed to eventuate the polynomial and estimate the coefficients. It is therefore often sufficient to choose a lower degree polynomial and approximate the function $f(x)$ to a certain degree of accuracy. Although the unisolvence theorem states that a discrete function can be modelled perfectly with an appropriate degree, it does not guarantee that other intermediate points in $[0, n]$ that were not used for the polynomial approximation, or any exterior points outside $[0, n]$ can be accurately calculated. In practice the difference between the function $f(x)$ and its approximation $m(x)$ can be very large, especially outside the interval $[0, n]$. The difference between the function and its approximation is formally defined by Weierstrass' approximation

theorem [258].

Theorem 3: Weierstrass Approximation *Any continuous function f defined on the real interval $[a, b]$, for every $\varepsilon > 0$ there exists a polynomial function m on $[a, b]$ such that $|f(x) - m(x)| < \varepsilon$ for $x \in [a, b]$.*

Taking Weierstrass' theorem into account, a function on the bounded interval $[a, b]$ can be uniformly approximated to any degree of accuracy desired by the polynomial. The theorem can be proven with Bernstein polynomials [20]. The error term ε between a function and its approximation given a set of discrete points is calculated as the sum of absolute deviations as follows:

$$\varepsilon = |f(x) - m(x)| = \sum_{i=0}^{n} |f(x_i) - m(x_i)| \tag{3.3}$$

The aim of the modelling process is to reduce the error ε by finding the degree d that will approximate m as closely to f as possible. The polynomial will converge to the continuous function on $[a, b]$ as the degree heads to infinity. The objective is therefore to reduce the maximum difference between f and m until it reaches zero, which is formally defined as

$$\lim_{d \to \infty} \left[\max_{a \leq x \leq b} \Big| f(x) - m(x) \Big| \right] = 0 \tag{3.4}$$

From this objective it might seem like a good approach to choose a high degree polynomial in order to reduce the error. However, high degree polynomials for certain functions can overfit, and the polynomials start oscillating at the edges of the interval, which is known as the Runge phenomenon [218]. Runge found that when modelling the function $f(x) = (1 + 25x^2)^{-1}$ with equidistant data points over the interval $[-1, 1]$, the approximation error increases as the degree of the polynomial increases. Figure 3.1 illustrates the phenomenon with polynomials of degree four and six osculating at the edges.

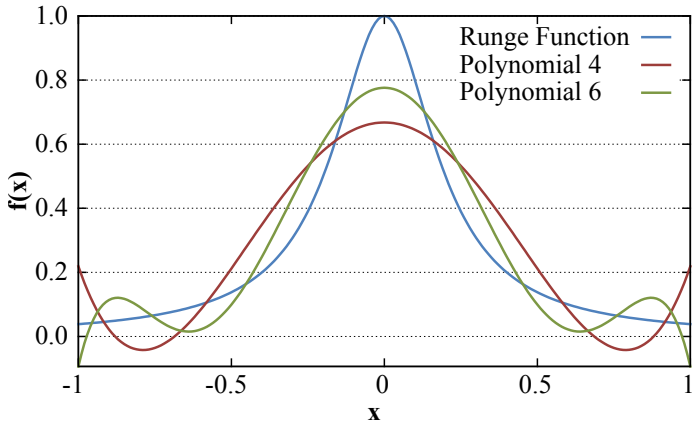


Figure 3.1: The Runge phenomenon with polynomials osculating at the edges of the interval.

Runge's phenomenon can be mitigated by selecting points which are more densely distributed towards the edges of the interval such as Chebyshev nodes, fitting a lower degree polynomial, or using piecewise polynomials [90, 91].

According to the unisolvence theorem a system of linear equations adhering to $m(x_i) = y_i$ with a set of coefficients α_i can be constructed. Equation (3.2) is expressed in matrix form as

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{d-1} & x_0^d \\ 1 & x_1 & x_1^2 & \cdots & x_1^{d-1} & x_1^d \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n-1} & x_{n-1}^2 & \cdots & x_{n-1}^{d-1} & x_{n-1}^d \\ 1 & x_n & x_n^2 & \cdots & x_n^{d-1} & x_n^d \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{d-1} \\ \alpha_d \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \tag{3.5}$$

where the left matrix represents a Vandermonde matrix which can be proven to be non-singular [173]. Since $x_i - x_j$ for $i, j \in \{0, 1, \ldots, n\}$ is never zero for the $n + 1$ distinct data points, the determinant of the matrix can never be zero, making the matrix non-singular and therefore invertible. The matrix can be solved using LLS regression. In the case of a linear polynomial running through exactly two points, the calculation is simplified to

$$m_{stp}(x) = y_0 + \frac{(y_1 - y_0)(x - x_0)}{x_1 - x_0} \tag{3.6}$$

which removes the burden of solving the equation with a computationally expensive LLS fit.

3.2 Fourier Polynomials

In 1807, Joseph Fourier introduced a series for solving the heat equation of metal plates [95], which later became known as the Fourier series. The idea was to model the complex partial differentiable equation as a superposition of simpler oscillating sine and cosine functions. Fourier's theorem for cosine series can be applied to even functions. A function, $f(x)$, is said to be even if it is symmetric around the y-axis, that is, $f(x) = f(-x)$.

Theorem 4: Fourier Cosine Series *Given an even function $f(x)$ with a period p, the function can be expressed as an infinite sum of cosine functions, such that*

$$f(x) = \frac{\alpha_0}{2} + \sum_{i=1}^{\infty} \alpha_i \cos\left(\frac{i\pi x}{p}\right) \tag{3.7}$$

The cosine model's coefficients α_i are calculated with Euler's formula using an integral on the interval $[-p, p]$ as follows:

$$\alpha_0 = \frac{1}{p}\int_{-p}^{p} f(x)dx \tag{3.8}$$

$$\alpha_i = \frac{1}{p}\int_{-p}^{p} f(x)\cos\left(\frac{i\pi x}{p}\right)dx \tag{3.9}$$

Fourier formulated a similar theorem for odd functions using sine series. A function, $f(x)$, is said to be odd if it is origin symmetric, that is, $-f(x) = f(-x)$.

Theorem 5: Fourier Sine Series *Given an odd function $f(x)$ with a period p, the function can be expressed as an infinite sum of sine functions, such that*

$$f(x) = \sum_{i=1}^{\infty} \beta_i \sin\left(\frac{i\pi x}{p}\right) \tag{3.10}$$

The sine coefficients β_i are calculated in a similar fashion to the cosine coefficients:

$$\beta_i = \frac{1}{p}\int_{-p}^{p} f(x)\sin\left(\frac{i\pi x}{p}\right)dx \tag{3.11}$$

A combination of Fourier's cosine and sine theorems is known as the Fourier series or the Fourier polynomial (FOP).

Theorem 6: Fourier Series *Given any periodic piecewise continuous function $f(x)$ with a period p, the function can be written as an infinite sum of sine and cosine waves, such that*

$$f(x) = \frac{\alpha_0}{2} + \sum_{i=1}^{\infty}\left[\alpha_i\cos\left(\frac{i\pi x}{p}\right) + \beta_i\sin\left(\frac{i\pi x}{p}\right)\right] \tag{3.12}$$

A function $f(x)$ is said to be piecewise continuous on the interval $[a, b]$ for values $x_0, \cdots, x_n$ where $x_0 = a$ and $x_n = b$, such that $f(x)$ is continuous on the open intervals (x_{i-1}, x_i) and there exists a one-sided left limit $f(x^-)$ and right limit $f(x^+)$ such that

$$f(x^-) = \lim_{x\to {x_i}^-} f(x) \tag{3.13}$$

$$f(x^+) = \lim_{x\to {x_{i-1}}^+} f(x) \tag{3.14}$$

for $i \in [1, n]$. A piecewise continuous function may therefore have jump discontinuities at x_i for $i \in [1, n-1]$. Since an infinite series of functions is impossible to use in practise, the function can only be estimated to a certain degree. The discrete Fourier polynomial can therefore be approximated with a finite sum d of sine and cosine functions with a period of one as follows:

$$m_{fop}(x) = \frac{\alpha_0}{2} + \sum_{i=1}^{d}\left[\alpha_i\cos\left(i\pi x\right) + \beta_i\sin\left(i\pi x\right)\right] \tag{3.15}$$

For any set of n data points, there exists a trigonometric polynomial that satisfies equation (3.12) as long as there are no more points than coefficients, therefore $n \leq 2d+1$. There exists a unique solution if and only if the number of coefficients is equal to the number of data points, that is $n = 2d+1$. If $n > 2d+1$ there might be a solution, depending on the data points.

Theorem 7: Fourier Series Pointwise Convergence *The Fourier series converges pointwise to $f(x)$ on the interval (a, b), if $f(x)$ is continuous and $f'(x)$ is piecewise continuous on the interval $[a, b]$, such that the sequence of Fourier partial sums converges to the left and right limit $\frac{1}{2}(f(x^-) + (f(x^+))$.*

The Fourier series of $f(x)$ will therefore converge to the left and right limit's average if the periodic extension has a jump discontinuity at x. If the function is periodic with bounded variation, meaning that the total variation of $f(x)$ is finite, the Fourier series will converge everywhere.

In addition to pointwise convergence, it can be proven by Jackson's theorem that the partial sum of the Fourier series will converge uniformly to $f(x)$ [141]. The absolute convergence of Fourier series was proven by Wiener's $1/f$ theorem [263]. The Riesz-Fischer theorem shows that the Fourier series converges to $f(x)$ in the norm of the function space L^2, which was proven independently by Riesz [216] and Fischer [81]. Carleson showed that any continuous function's Fourier expansion in L^2 converges almost everywhere [40].

Using equation (3.15), a system of linear equations can be constructed and solved by a LLS fit. The equations are expressed in matrix form as

$$\begin{bmatrix} \frac{1}{2} & \cos(\pi x_0) & \cdots & \cos(d\pi x_0) & \sin(\pi x_0) & \cdots & \sin(d\pi x_0) \\ \frac{1}{2} & \cos(\pi x_1) & \cdots & \cos(d\pi x_1) & \sin(\pi x_1) & \cdots & \sin(d\pi x_1) \\ \frac{1}{2} & \cos(\pi x_2) & \cdots & \cos(d\pi x_2) & \sin(\pi x_2) & \cdots & \sin(d\pi x_2) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2} & \cos(\pi x_{n-1}) & \cdots & \cos(d\pi x_{n-1}) & \sin(\pi x_{n-1}) & \cdots & \sin(d\pi x_{n-1}) \\ \frac{1}{2} & \cos(\pi x_n) & \cdots & \cos(d\pi x_n) & \sin(\pi x_n) & \cdots & \sin(d\pi x_n) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_d \\ \beta_1 \\ \vdots \\ \beta_d \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix} \tag{3.16}$$

3.3 Newton Polynomials

In his early years, Isaac Newton formulated a polynomial of least degree that coincides at all points of a finite dataset [190]. Given $n+1$ data points (x_i, y_i), the Newton polynomial (NEP) is defined as

$$m_{nep}(x) = \sum_{i=0}^{n} \alpha_i h_i(x) \tag{3.17}$$

where $h_i(x)$ is the i^{th} Newton basis polynomial given as

$$h_i(x) = \prod_{j=0}^{i-1} (x - x_i) \tag{3.18}$$

Newton's divided differences polynomial is the polynomial where the coefficients α_i are calculated with divided differences, given as

$$\alpha_i = [y_0, \ldots, y_i] \tag{3.19}$$

The divided differences is a recursive division of the $n + 1$ points. When using the divided differences for the coefficients, the Newton polynomial is equivalent to the Taylor polynomial by replacing the instantaneous rates of change with finite differences [240]. The forward divided differences process is defined as

$$[y_i] = y_i \qquad i \in \{0, \ldots, n\} \tag{3.20}$$

$$[y_i, \ldots, y_{i+j}] = \frac{[y_{i+1}, \ldots, y_{i+j-1}] - [y_i, \ldots, y_{i+j}]}{x_{i+j} - x_i} \qquad \begin{matrix} i \in \{0, \ldots, n-j\} \\ j \in \{1, \ldots, n\} \end{matrix} \tag{3.21}$$

The backward divided differences can also be used for calculating the coefficients, where the differences are determined in a reversed recursive process by replacing $i+j$ in equation (3.21) with $i-j$. New data points can easily be added to the polynomial by placing them at the right side for the forward formula and at the left side for the backward formula. However, the accuracy of an estimated point is depended on its distance to the middle of the x values of the data points. Hence, when increasing the degree by adding points to the end, the accuracy can only be increased at the end and nowhere else. Gauss mitigated this problem by altering the addition process to add points to the left and right, thereby keeping the points centred around the same place [261]. Bessel and Stirling provided similar approaches for keeping the points centred around a specific middle [235, 261]. Stirling's formula works best when the evaluation point falls in a region where it is needed less. Bessel's formula works best when the evaluation point lies between two other

points somewhere in the middle. When working with large datasets, the mean accuracy difference between the Gauss, Bessel and Stirling formulas is statistically insignificant.

Since the Newton basis polynomial is the product of time delay differences, when the x values are in the range $[0, 1]$, the resulting product $h_i(x)$ can be very small. Decimal precision of most programming languages is only accurate to the 15^{th} or 16^{th} digit. Multiplying smaller numbers will result in zero, therefore making the entire Newton polynomial constant at zero. If high degree Newton polynomials are used with time delays smaller than one, the precision should be accommodated with multi-precision libraries such as the GNU Multiple Precision Arithmetic Library (GMP) [111].

Another approach to constructing a Newton polynomial is to estimate the coefficients instead of calculating them from the divided differences. The Newton basis polynomial in equation (3.18) can be used to construct a system of linear equations with a lower triangular matrix as follows:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & x_1 - x_0 & 0 & \cdots & 0 \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x_0 & (x_n - x_0)(x_n - x_1) & \cdots & \prod_{j=0}^{n-1}(x_n - x_j) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{3.22}$$

The coefficients α_i can then be estimated by solving the equations with a LLS fit.

3.4 Lagrange Polynomials

In 1795, Joseph Louis Lagrange formulated a polynomial of least degrees that fits all given points of a finite dataset [156]. Although attributed to Lagrange, the formula was previously discovered by Waring [255] and Euler [75] in 1779 and 1783 respectively. The Lagrange polynomial (LAP) for $n + 1$ data points (x_i, y_i) is defined as

$$m_{lap}(x) = \sum_{i=0}^{n} y_i l_i(x) \tag{3.23}$$

with $l_i(x)$ the i^{th} Lagrange basis polynomial given as

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \tag{3.24}$$

The Lagrange basis polynomial is always well defined since no x_i is the same, ensuring that the division is never zero. The Lagrange polynomial constructed with equidistant points is subject to the Runge phenomena. The divergence tends to increase as more points are added to the polynomial and can be eliminated by constructing the polynomial with points at Chebyshev nodes.

The Lagrange polynomial of least degree is unique, making it equivalent to the Newton polynomial where the coefficients are calculated by divided differences. The Lagrange polynomial has to be recalculated every time new points are added, making the Newton polynomial easier and more efficient to use when continuously adding points to the set. An alternative to the recalculation problem is to use the barycentric form of the Lagrange basis polynomial, defined as

$$l_i(x) = \frac{\prod_{i=0}^{n}(x - x_i)}{(x - x_i)\prod_{j=0, j \neq i}^{n} x_i - x_j} \tag{3.25}$$

Like with Netwon, Lagrange polynomials can be zero with time delays smaller than one. Multi-precision arithmetic has been employed to mitigate this problem.

3.5 Hermite Polynomials

In 1878, Charles Hermite introduced a polynomial closely related to the Newton and Lagrange polynomials [125]. Besides calculating a polynomial for $n + 1$ points, Hermite also considered the derivatives at these points. The Hermite polynomial (HEP) using the first derivative is defined as

$$m_{hep}(x) = \sum_{i=0}^{n} h_i(x) f(x_i) + \sum_{i=0}^{n} \overline{h}_i(x) f'(x_i) \tag{3.26}$$

where $h_i(x)$ and $\overline{h}_i(x)$ are the first and second fundamental Hermite polynomials, calculated as follows:

$$h_i(x) = [1 - 2l_i'(x_i)(x - x_i)]\,[l_i(x)]^2 \tag{3.27}$$

$$\overline{h}_i(x) = (x - x_i)\,[l_i(x)]^2 \tag{3.28}$$

$l_i(x)$ is the i^{th} Lagrange basis polynomial given in equation (3.24) and $l_i'(x_i)$ is the derivative of the Lagrange basis polynomial at point x_i. The fundamental Hermite polynomials have the following properties:

$$h_i(x_j) = \delta_{ij} \qquad h_i'(x_j) = 0 \tag{3.29}$$

$$\overline{h}_i{}'(x_j) = \delta_{ij} \qquad \overline{h}_i(x_j) = 0 \tag{3.30}$$

where δ_{ij} is the Kronecker delta such that $\delta_{ij} = 0$ if $i \neq j$ and $\delta_{ij} = 1$ if $i = j$.

Since Lagrange polynomials are computationally expensive to differentiate and evaluate, $h_i(x)$ and $\overline{h}_i(x)$ can be calculated by a more efficient Newton divided differences table [150], where each entry for two identical points are replaced by the slope, $f'(x)$, at the common point. The table can then be written in matrix form and solved in the traditional way or by using the more efficient approach of Mehdi and Hajarian which rely on the eigenvalues of the matrix [51].

In the original publication by Hermite, Lagrange polynomials were used as the fundamental polynomials. This approach can be used for any polynomial function as long as the derivatives of the function are known. To distinguish between them, Hermite polynomials with Lagrange fundamental polynomials will be simply referred to as *Hermite polynomials*, whereas if a different fundamental polynomial is used, it will be denoted as *osculating polynomials*. This thesis will investigate the osculating standard polynomial (OSP) and the osculating Fourier polynomial (OFP).

An osculating polynomial constructed from polynomials in standard form from equation (3.2) takes on the following system of linear equations:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{d-1} & x_0^d \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{d-1} & x_n^d \\ 0 & 1 & 2x_0 & \cdots & (d-1)x_0^{d-2} & dx_0^{d-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & 2x_n & \cdots & (d-1)x_n^{d-2} & dx_n^{d-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \\ \vdots \\ \\ \alpha_{d-1} \\ \alpha_d \end{bmatrix} = \begin{bmatrix} f(x_0) \\ \vdots \\ f(x_n) \\ f'(x_0) \\ \vdots \\ f'(x_n) \end{bmatrix} \tag{3.31}$$

The given matrix only contains the equations of the first derivatives. Higher derivatives can be added in a similar fashion. The first half of the right value vector is easily obtained by the y values of the data points. The second half of the vector requires the calculation of the derivatives of the function at the given points. This is not a problem with continuous functions that have known derivatives, but can be difficult to obtain for discrete functions. Numerical differentiation has to be applied to the data points of the discrete functions to estimate the derivatives at the given points. A common approach to this problem is using finite differences based on the central formula of Gauss, Bessel or Stirling [235, 261]. The generic central formula for higher order derivatives of $n+1$ points is defined as

$$\delta_s^n[f](x) = \sum_{i=0}^{n} (-1)^i \binom{n}{i} f\left(x + \left(\frac{n}{2} - i\right) s\right) \tag{3.32}$$

where s is the spacing between the data points. $\binom{n}{i}$ are binomial coefficients, which are the number of different ways i unordered outcomes can be picked from n possibilities. $\binom{n}{i}$ is the coefficient of the term x^i in the expansion of the binomial power $(1+x)^n$. The Pascal triangle is made up of ordered binomial coefficients with rows for successive values of n and $i \in \{0, 1, \ldots, n\}$. The binomial coefficients are calculated as

$$\binom{n}{i} = \prod_{j=1}^{i} \frac{n-i+j}{j} \tag{3.33}$$

When n is odd in equation (3.32), dividing by two will result in a fraction, causing a change to the interval of discretization. This problem can be mitigated by taking the average of the previous and next central differences, that is $\delta^n[f](x - \frac{s}{2})$ and $\delta^n[f](x + \frac{s}{2})$.

If a discrete function was derived from a continuous function, numerical differentiation will often be inaccurate, since it is only an estimation of the true derivative. Depending

on the function and the nature of the data points, this may cause Hermite and osculating polynomials to be inaccurate at certain points of the discrete function.

3.6 Splines

Splines are a set of piecewise polynomials where the derivatives at the endpoints of neighbouring polynomials are equal. Spline interpolation is not subject to Runge's phenomenon during high-degree interpolation, since a separate polynomial is estimated between every two neighbouring data points. Additionally, the approximation error can be kept small even for low-degree splines. Given a set of $n+1$ data points, n number of splines are constructed, one between every neighbouring data point pair. The splines are created as follows:

$$m_s(x) = \begin{cases} s_1(x) & \text{for} \quad x_0 \leq x < x_1 \\ s_2(x) & \text{for} \quad x_1 \leq x < x_2 \\ \quad \vdots & \\ s_{n-1}(x) & \text{for} \quad x_{n-2} \leq x < x_{n-1} \\ s_n(x) & \text{for} \quad x_{n-1} \leq x < x_n \end{cases} \tag{3.34}$$

The individual splines can therefore be constructed using any other kind of polynomial function, as long as the derivatives can be calculated. This thesis will focus on standard polynomial splines (SPS) and Fourier polynomial splines (FPS). To ensure a smooth connection between neighbouring splines, the derivatives at the interior data points x_i have to be continuous [253], that is,

$$s_i'(x_i) = s_{i+1}'(x_i) \tag{3.35}$$

for the first derivatives with $i \in \{1, 2, \ldots, n\}$. As the degree of the individual splines increases, higher order derivatives also have to be continuous. If the splines each have c number of coefficients, a total of $n \times c$ coefficients have to be estimated and therefore $n \times c$ equations have to be obtained to solve the system. $2 \times n$ equations are calculated for each spline using the data points to the left and right. Spline s_i will therefore result

in two equations obtained from point x_i and x_{i+1}. Another $(d-1)\times(n-1)$ equations are calculated from the derivatives at the interior points, where d is the degree of the splines. To make up the additional equations, constraints are imposed on the derivatives at the endpoints by incorporating the *free endpoint condition* [23]. The derivatives at the free endpoints are set to zero, starting from the highest continuous derivative. Lower derivatives are set to zero until enough equations were obtained. When such constraints are imposed, the function is called a natural spline [42].

Constructing the splines with polynomials in standard form, the system of linear equations is represented in matrix form as follows:

$$
\begin{array}{c}
\\
r_a \\
\\
\vdots \\
\\
r_b \\
r_c \\
\vdots \\
r_d \\
r_e \\
\vdots \\
r_f \\
r_g \\
\\
\vdots \\
\\
r_h
\end{array}
\begin{bmatrix}
1 & x_0 & \cdots & x_0^d & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
1 & x_1 & \cdots & x_1^d & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 & x_{n-1} & \cdots & x_{n-1}^d \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & 1 & x_n & \cdots & x_n^d \\
0 & 1 & \cdots & dx_0^{d-1} & 0 & -1 & \cdots & -dx_1^{d-1} & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 1 & \cdots & dx_{n-1}^{d-1} & 0 & -1 & \cdots & -dx_n^{d-1} \\
0 & 0 & \cdots & d!x_0 & 0 & 0 & \cdots & -d!x_1 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & d!x_{n-1} & 0 & 0 & \cdots & -d!x_n \\
0 & 0 & \cdots & d!x_0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & d!x_n \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 1 & \cdots & dx_0^{d-1} & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & dx_n^{d-1}
\end{bmatrix}
\begin{bmatrix}
\alpha_{1,0} \\ \alpha_{1,1} \\ \vdots \\ \alpha_{1,d} \\ \alpha_{2,0} \\ \alpha_{2,1} \\ \vdots \\ \alpha_{2,d} \\ \\ \\ \vdots \\ \\ \\ \alpha_{n,0} \\ \alpha_{n,1} \\ \vdots \\ \alpha_{n,d}
\end{bmatrix}
=
\begin{bmatrix}
y_0 \\ y_1 \\ \vdots \\ y_n \\ 0 \\ 0 \\ \\ \\ \\ \\ \vdots \\ \\ \\ \\ \\ 0 \\ 0
\end{bmatrix}
\tag{3.36}
$$

where $\alpha_{i,j}$ is the j^{th} coefficient of the i^{th} spline. Rows r_a to r_b hold the equations for the spline polynomials. These equations always come in pairs of two, one for the left and one for the right point of the spline. Rows r_c to r_d contain the equations for the first derivatives of the interior points. Rows r_e to r_f hold higher derivatives which are

only required for cubic splines and higher. The last equations in rows r_h to r_h hold the constraints on the derivatives at the endpoints and are added sequentially until the matrix reaches $n \times c$ rows.

In practice cubic splines or lower are mostly used, since higher degree splines tend to overfit the model and reduce the approximation accuracy for intermediate points. Cho provided a theoretical justification for this observation and concluded that cubic splines are sufficient for most practical applications [43]. If the splines are constructed with first degree standard polynomials, they are equivalent to linear polynomials in equation (3.2). However, since the derivatives and free endpoint conditions are included in the matrix, the LLS spline approximation will estimate slightly different coefficients compared to the system in equation (3.5).

3.7 Autoregressive Model

The autoregressive (AR) model, also known as the maximum entropy model, is an infinite impulse response filter that models a random process where the generated output is linearly depended on the previous values in the process. Since the model keeps track of the feedback, therefore retaining memory, it can generate internal dynamics. Given y_i as a sequential series of $n+1$ data points, the AR model of order p predicts the value of a point at time delay t with the previous values of the series, defined as

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \alpha_i y_{t-i} \tag{3.37}$$

where c is a constant, typically considered to be zero, ε_t the white noise error term, almost always considered to be Gaussian white noise, and α_i the autoregressive coefficients for the model. A common approach in time series analysis is to subtract the temporal mean from time series y before feeding it into the AR model. It was found that this approach is not advisable with sample windows of short durations, since the temporal mean is often not a true representation of the series' mean and can vary greatly among subsets of the series [55]. The series y is assumed to be linear, stationary and has a zero mean. A stationary series is stochastic where the joint probability distribution does not

change with a progression in time. If the series does not have a zero mean, an additional parameter α_0 is added to the front of the summation in equation (3.37).

One of the most widely used methods for estimating the AR model coefficients is solving the Yule-Walker equations with a LLS regression. The Yule-Walker equations [251, 270] for the AR model coefficients, α_i, are defined as

$$\gamma_t = \sum_{i=1}^{p} \alpha_i \gamma_{t-i} + \sigma_\varepsilon^2 \delta_{t,0} \tag{3.38}$$

where γ_t is the autocovariance function of y_t given in equation (2.36), σ_ε^2 the variance of the noise ε and $\delta_{t,0}$ the Kronecker delta function. For a time delay τ, multiplying equation (3.37) with $x_{t-\tau}$ and normalizing the expectation values will result in a set of linear equations where γ_τ is said to be the autocorrelation coefficient [29]. Since the last part of equation (3.38) will always be zero if $t \neq 0$ due to the Kronecker delta function, the Yule-Walker equations for $t > 0$ are represented in matrix form as

$$\begin{bmatrix} 1 & \gamma_1 & \gamma_2 & \gamma_3 & \cdots & \gamma_{p-1} \\ \gamma_1 & 1 & \gamma_1 & \gamma_2 & \cdots & \gamma_{p-2} \\ \gamma_2 & \gamma_1 & 1 & \gamma_1 & \cdots & \gamma_{p-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_{p-1} & \gamma_{p-2} & \gamma_{p-3} & \gamma_{p-4} & \cdots & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_p \end{bmatrix} \tag{3.39}$$

Besides the LLS method, a number of alternative approaches exist for the estimation of the AR model coefficients. One such alternative constructs a system using two sets of Yule-Walker equations for forward and backward prediction and approximates the parameters using the Burg method [36]. The estimation is computationally more expensive than using only forward prediction with LLS and the resulting autocovariances may also be different. This method is commonly used in maximum entropy spectral estimation [28]. It was also found that the Yule-Walker approach may produce incorrect coefficients if the function is near periodical and that the Burg method should be used in such a case [50]. Another route calculates the coefficients based on the autocorrelations in equation (2.50) and autocovariances in equation (2.36) that are estimated separately using conventional estimates [33]. The maximum likelihood estimation can also be used, which is well-defined for normal distribution problems, but can be computationally expensive and unsuitable

or even unsolvable in a number of situations [140, 176]. The Markov chain Monte Carlo method [106, 120, 180] can also be used for the AR coefficient estimation.

3.8 Moving Average Model

The moving average is a statistical calculation where a series of averages are generated from subsets of the full dataset. The moving average is a finite impulse response filter which continuously updates the average as the window of interest moves across the dataset. Although the concept of moving averages was used in mathematics for decades, it was only popularized at the beginning of the 20$^{\text{th}}$ century by Hooker [131] and Yule [268]. The moving averages with a window size of n samples over a series y is the new sequence a_t of unweighed means defined as

$$a_t = \frac{1}{n} \sum_{i=t}^{t+n-1} y_i \tag{3.40}$$

Adaptations of the algorithm were published, which include the cumulative-sum, exponential, variable, weighted and modified moving averages [57, 148]. A study by Slutsky on applying the moving average on random events lead to the formulation of what later became known as the moving average (MA) model [229]. This approach is a finite impulse response filter with some additional interpretation added where univariate time series are modelled with white noise terms. Slutzky [229] and Yule [269] independently discovered that the moving summation of random data series oscillates when no such fluctuation exists in the original observation. This characteristic later became known as the Slutzky-Yule effect. The MA model predicts the value of a data point with a time delay t as follows:

$$y_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \beta_i \varepsilon_{t-i} \tag{3.41}$$

where μ is the mean of the series, typically assumed to be zero, β_i the model coefficients of order q and $\varepsilon_t, \ldots, \varepsilon_{t-q}$ the white noise error terms. The error terms are assumed to be independent and identically distributed random variables, meaning that all random variables are mutually independent and are subject to the same probability distribution.

Given a normal distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2, the error terms in the MA model are sampled with zero mean $\mathcal{N}(0, \sigma^2)$, and will therefore be Gaussian white noise. The Gaussian white noise is typically generated from the mean and the variance in equation (2.29). A well-known method for sampling pseudo-random numbers from a probability distribution is the Smirnov transform, more commonly known as inverse transform sampling [54]. Given the cumulative distribution function [35], a number u is uniformly sampled in the interval $[0, 1]$. The largest number r from the distribution of interest $p(x)$ is then returned such that $p(-\infty < x < r) \leq u$. Although the generation from a discrete distribution is more efficient than from a continuous distribution, the Smirnov transform is considered computationally inefficient since it requires a complete approximation of the cumulative distribution function, regardless of the number of samples [107]. Similar concerns about the efficiency of the transform have been brought forward by other publications [110, 265].

A more efficient approach was proposed by Box and Muller where independent pairs of pseudo-random numbers are generated from a normal distribution [30]. The Box-Muller transform can be expressed in basic form where two uniformly distributed independent random numbers u and v from the interval $(0, 1]$ are mapped to two standard, normally distributed samples z_1 and z_2 by making use of the sine and cosine functions. The Box-Muller transform in basic form is defined as

$$\begin{aligned} z_1 &= \sqrt{-2\ln(u)}\cos(2\pi v) \\ z_2 &= \sqrt{-2\ln(u)}\sin(2\pi v) \end{aligned} \tag{3.42}$$

Marsaglia and Bray proposed to express the Box-Muller transform in polar form to increase computational efficiency by removing the sine and cosine functions [172]. The Marsaglia polar transform, often just referred to as the polar method, was first standardized by Bell [17] and later modified by Knop [151]. Given two independently generated random numbers u and v which are uniformly distributed over the interval $[-1, 1]$, set $s = u^2 + v^2$, $\cos(2\pi v) = \frac{u}{\sqrt{s}}$ and $\sin(2\pi v) = \frac{v}{\sqrt{s}}$. If $s \geq 1$ or $s = 0$, the random numbers are discarded and two new values are generated for u and v. Once s is in the interval $(0, 1)$, equation (3.42) is changed as follows:

$$z_1 = \sqrt{-2\ln(s)}\left(\frac{u}{\sqrt{s}}\right) = u\sqrt{\frac{-2\ln(s)}{s}}$$
$$z_2 = \sqrt{-2\ln(s)}\left(\frac{v}{\sqrt{s}}\right) = v\sqrt{\frac{-2\ln(s)}{s}} \tag{3.43}$$

Both the basic and polar form generate two random numbers, even if only one term is needed at a time. In order to increase performance, certain parts such as $\sqrt{-2\ln(u)}$ in the basic form are computed once when generating z_1 and then simply recalled when generating the second sample z_2, rather than recalculated it. The polar form is a rejection sampling algorithm. Although $1 - \frac{\pi}{4} \approx 21.5\%$ of all randomly generated sample pairs are rejected, since s falls outside the interval $(0, 1)$, the polar form is still more efficient than the basic form. A multiplication and trigonometric operation in the basic form is replaced by only a single division operation in the polar form, making it faster on most machines.

In order to generate Gaussian white noise for the MA model in equation (3.41), the Gaussian error terms have to be subjected to the observed data series y. This can be done by multiplying the generated samples z_1 and z_2 by the variance σ^2 of the series y.

Since the lagged error terms of the MA model are not observable, the model parameters can not be estimated by means of linear fitting, such as LLS regression. The estimation has to be done by means of nonlinear fitting, often applied in an iterative fashion by making use of a nonlinear optimization algorithm. One such widely used algorithm for estimating the parameters of a model is the maximum likelihood estimation (MLE) method. The MLE tries to find the set of model coefficients that will most likely produce the observed values. Popularized by Fisher over a period of ten years [82, 83, 84, 85, 86], MLE is based on previous work done by Newton, Laplace and most notably Edgeworth [62]. The probability density function (PDF), $\mathcal{P}(y|\beta)$, gives the probability that the dataset y will be observed when using a set of q coefficients $\beta = (\beta_1, \beta_2, \ldots, \beta_q)$. If the n data points in the set y are statistically independent from each other, the PDF of the set can be expressed as the product of the PDFs of the individual data points as follows:

$$\mathcal{P}(y_1, y_2, \ldots, y_n|\beta) = \prod_{i=1}^{n} \mathcal{P}(y_i|\beta) \tag{3.44}$$

The PDF requires the density of a continuous random variable in order to determine the relative likelihood. This is achieved using a specific distribution, such as a Bernoulli or Poisson distribution. The PDF of the MA model is determined by a normal distribution, since the error terms in the model are generated using a Gaussian distribution. Given the definition of a normal distribution,

$$p(x, \sigma, \mu) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.45}$$

where σ is the standard deviation and μ the mean of the series, the PDF for the normal distribution relies on two parameters, σ and μ, and can be expressed as

$$\mathcal{P}(y_1, y_2, \ldots, y_n|\sigma, \mu) = \prod_{i=1}^{n} \frac{1}{\sigma_i\sqrt{2\pi}} \exp\left(-\frac{(y_i-\mu)^2}{2\sigma_i^2}\right) \tag{3.46}$$

The PDF shows that some data has a higher probability than others for a set of coefficients β. The likelihood function is the inverse problem of the PDF, namely finding a single PDF that is most likely to have produced the given data. The parameters are therefore switched around, giving the likelihood function as

$$\mathcal{L}(\beta|y) = \mathcal{P}(y|\beta) \tag{3.47}$$

Once the data is available and the corresponding model's likelihood function was determined, the goal is to find the set of parameters that will maximize the likelihood function $\mathcal{L}(\beta|y)$. The natural logarithm of the likelihood function is often used due to computational convenience. This does not pose a problem, since $\mathcal{L}(\beta|y)$ and $\ln \mathcal{L}(\beta|y)$ are monotonically related to each other and will always result in the same maximum likelihood estimate when maximizing either one of them [188]. The logarithm of the likelihood, commonly referred to as log-likelihood, is written as

$$\ell(\beta|y) = \ln \mathcal{L}(\beta|y) = \sum_{i=1}^{n} \ln \mathcal{P}(y_i|\beta) \tag{3.48}$$

Given the MA model in equation (3.41) with the PDF for a normal distribution in equation (3.46), $\mathbf{C}$ as the variance-covariance matrix from equation (2.37), $\boldsymbol{\mu}$ the mean vector and $\mathbf{y}$ the vector of observations, the likelihood of the MA model is expressed as

$$\mathcal{L}(\beta|\mathbf{y}) = (2\pi)^{-\frac{n}{2}} |\mathbf{C}|^{-\frac{1}{2}} \exp\left(\frac{(\mathbf{y}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{y}-\boldsymbol{\mu})}{-2}\right) \tag{3.49}$$

Triangularly factorizing $\mathbf{C}$ simplifies the likelihood of the MA model to

$$\mathcal{L}(\beta|\mathbf{y}) = (2\pi)^{-\frac{n}{2}} \left(\prod_{i=1}^{n}\theta_i\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\frac{\varphi_i^2}{\theta_i}\right) \tag{3.50}$$

where

$$\theta_i = \sigma^2 \frac{1+\beta^2+\beta^4+\cdots+\beta^{2i}}{1+\beta^2+\beta^4+\cdots+\beta^{2(i-1)}} = \frac{\sigma^2\sum_{j=0}^{i}\beta^{2j}}{\sum_{j=1}^{i}\beta^{2(j-1)}} \tag{3.51}$$

and

$$\varphi_i = y_i - \mu - \frac{\beta(1+\beta^2+\beta^4+\cdots+\beta^{2i})}{1+\beta^2+\beta^4+\cdots+\beta^{2(i-1)}}\beta_{i-1} = y_i - \mu - \frac{\beta_{i-1}\beta\sum_{j=0}^{i}\beta^{2j}}{\sum_{j=1}^{i}\beta^{2(j-1)}} \tag{3.52}$$

To improve computational efficiency, the sum of the numerator and denominator in equations (3.51) and (3.52) only has to be calculated once for i, stored temporarily and then used to calculate both θ_i and φ_i. Using equation (3.50), the log-likelihood of the MA model is defined as

$$\ell(\beta|y) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\sum_{i=1}^{n}\ln(\theta_i) - \frac{1}{2}\sum_{i=1}^{n}\frac{\theta_i^2}{\varphi_i} \tag{3.53}$$

Maximizing the likelihood function in equation (3.47) is commonly known as exact maximum likelihood (EML) or unconditional maximum likelihood. Based on the conditional testing by Rasch [210], instead of maximizing the likelihood directly, the maximization is done on a conditional distribution with respect to an increasing number of incidental parameters [7], called conditional maximum likelihood (CML). Given the conditional probability $p(y|x)$ where the probability of y is based on x, the unconditional likelihood from equation (3.47) changes to the conditional likelihood as follows:

$$\mathcal{L}(\beta; x|y) = \mathcal{P}(x|y;\beta) \tag{3.54}$$

Given the first and second data points from y, the conditional density $f(y_2|y_1;\beta)$ is only useful if an initial assumption about the values of interest, which are the white error terms in the MA model, are made. Assume that the first error term in the MA model is known to be zero, that is $\varepsilon_0 = 0$. The sequence of error terms ε are calculated from the observations x by iteratively calculating the next error term by rewriting the MA model of order q in equation (3.41) as

$$\varepsilon_t = y_t - \mu - \beta_1\varepsilon_{t-1} - \beta_2\varepsilon_{t-2} - \ldots - \beta_i\varepsilon_{t-q} = y_t - \mu - \sum_{i=1}^{q}\beta_i\varepsilon_{t-i} \tag{3.55}$$

With the assumption that $\varepsilon_0 = 0$ and using equation (3.55) above, the conditional PDF for the i^{th} observation is defined as

$$\mathcal{P}(y_i|\varepsilon_{i-1};\beta) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\varepsilon_i^2}{2\sigma^2}\right) \tag{3.56}$$

The conditional log-likelihood of the MA model is therefore

$$\ell_c(\boldsymbol{\beta}|\mathbf{y}) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2}\sum_{i=1}^{n}\frac{\varepsilon_i^2}{\sigma^2} \tag{3.57}$$

Although CML is in general slightly more efficient than EML when the spread between parameters decreases, information loss during parameter estimation almost always occurs [63]. In practice, however, the information loss is so marginal that CML is considered a sound practice.

Generalizing the parameters to allow β to be a vector $\boldsymbol{\beta}$, $\hat{\boldsymbol{\beta}}$ is the set of parameters that will maximize the log-likelihood, that is,

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta}|\mathbf{y}) \tag{3.58}$$

$\hat{\boldsymbol{\beta}}$ can generally be determined in three different ways, namely exhaustive search, analytical methods, and numerical optimization. If it is known that $\hat{\boldsymbol{\beta}}$ lies within a certain subspace, an exhaustive search is conducted over the subspace by repeatedly approximating the model until the value is found that has the largest likelihood. Although easy to implement and a guarantee to find the appropriate value for $\hat{\boldsymbol{\beta}}$, depending on the granularity of the search, it is in most cases not practical. An exhaustive search is computationally

expensive and rises exponentially with each additional model parameter. It is also often difficult to determine in which subspace $\hat{\boldsymbol{\beta}}$ will fall. The subspace will determine the outer bounds for the exhaustive search.

The second option for determining $\hat{\boldsymbol{\beta}}$ is to use an analytical approach. Assuming that coefficients $\hat{\boldsymbol{\beta}}$ exist that will maximize the log-likelihood, if $\ell(\boldsymbol{\beta}|\mathbf{y})$ is differentiable, it has to satisfy the partial differential likelihood equation for each coefficient in the set $\boldsymbol{\beta}$, in other words,

$$\frac{\partial \ell(\boldsymbol{\beta}|\mathbf{y})}{\partial \hat{\boldsymbol{\beta}}_i} = 0 \tag{3.59}$$

where $\hat{\boldsymbol{\beta}}_i$ are the i^{th} model coefficients that maximizes the log-likelihood. For the log-likelihood of a normal distribution in equation (3.46), the function has to be partially differentiated for both parameters, that is,

$$\frac{\partial \ell(\boldsymbol{\sigma}|\mathbf{y})}{\partial \hat{\boldsymbol{\sigma}}_i} = 0 \tag{3.60}$$

$$\frac{\partial \ell(\boldsymbol{\mu}|\mathbf{y})}{\partial \hat{\boldsymbol{\mu}}_i} = 0 \tag{3.61}$$

Since the first derivative does not indicate whether $\ell(\boldsymbol{\beta}|\mathbf{y})$ is a minimum or maximum, a second condition has to be satisfied to ensure that the maximum is chosen, that is a convex log-likelihood function. This can be done by checking whether or not the second derivative is negative, that is

$$\frac{\partial^2 \ell(\boldsymbol{\beta}|\mathbf{y})}{\partial \hat{\boldsymbol{\beta}}_i^2} < 0 \tag{3.62}$$

A more accurate way of determining convexity makes use of Hessian matrices [109]. Given the Hessian matrix as

$$\mathbf{H}_{ij}(\boldsymbol{\beta}) = \frac{\partial^2 \ell(\boldsymbol{\beta}|\mathbf{y})}{\partial \boldsymbol{\beta}_i \partial \boldsymbol{\beta}_j} = \sum_{k=1}^{n} \frac{\partial^2 \ell_k(\boldsymbol{\beta}|y_k)}{\partial \boldsymbol{\beta}_i \partial \boldsymbol{\beta}_j} \tag{3.63}$$

for both $i, j \in \{1, 2, \ldots, q\}$, the log-likelihood function will be convex if the determinate of the matrix $\mathbf{H}_{ij}(\boldsymbol{\beta})$ is negative definite, that is

$$\mathbf{v}^{\mathrm{T}}\mathbf{H}_{ij}(\boldsymbol{\beta})\mathbf{v} < 0 \tag{3.64}$$

for a real-numbered vector $\mathbf{v}$.

If there is no analytical solution, that is the likelihood equation in (3.59) can not be solved, a numerical approach can be used to determine $\hat{\boldsymbol{\beta}}$. The gradient is systematically followed until the maximum is found, where the gradient vector, $\mathbf{g}(\boldsymbol{\beta})$, containing the first derivatives of the log-likelihood function are calculated as follows:

$$\mathbf{g}_i(\boldsymbol{\beta}) = \frac{\partial \ell(\boldsymbol{\beta}|\mathbf{y})}{\partial \boldsymbol{\beta}_i} = \sum_{k=1}^{n} \frac{\partial \ell_k(\boldsymbol{\beta}|y_k)}{\partial \boldsymbol{\beta}_i} \tag{3.65}$$

Gradient-based methods that iteratively search for the maximum are generalised as given in algorithm 1.

Algorithm 1 A gradient-based algorithm for finding the maximum.

determine the convergence tolerance
determine the maximum number of iterations
determine the initial coefficients $\boldsymbol{\beta}_0 \in \boldsymbol{\beta}$
repeat
  if $\mathbf{g}(\boldsymbol{\beta}_0)$ meets the convergence tolerance **then**
    return $\boldsymbol{\beta}_0$
  else
    compute the direction vector $\mathbf{H}(\mathbf{g}(\beta_0))$;
    set $\boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 + \mathbf{H}(\mathbf{g}(\boldsymbol{\beta}_0))$;
    change $\boldsymbol{\beta}_0 = \boldsymbol{\beta}_1$;
  end if
until current iteration exceeds the maximum number of iterations

Two parameters are specified, namely a threshold the gradient vector $\mathbf{g}(\boldsymbol{\beta})$ should converge to and the maximum number of iterations in case of non-convergence. This approach is followed by a number of algorithms, which mainly differ in the way the direction vector $\mathbf{H}(\mathbf{g}(\boldsymbol{\beta}_0))$ is calculated, such that $\ell(\boldsymbol{\beta}_1) > \ell(\boldsymbol{\beta}_0)$ to ensure that the maximum is eventually found, unless the maximum number of iterations is reached.

One such gradient-based optimization is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm which was independently discovered by Broyden [34], Fletcher [88], Goldfarb [115] and Shanno [222]. The algorithm is a quasi-Newton method that iteratively solves unconstrained nonlinear optimization problems. BFGS constructs a Hessian matrix from the second derivatives and is essentially the same as the Davidon-Fletcher-Powell (DFP) formula [48, 89], but instead of approximating the Hessian matrix, $\mathbf{H}_i(\boldsymbol{\beta})$, the inverse of the matrix $\mathbf{H}_i(\boldsymbol{\beta})^{-1}$ is used. DFP has to solve a system of linear equations to determine the gradient direction, whereas the BFGS is computationally more efficient, since the direction, which is defined as $-\mathbf{H}_i(\boldsymbol{\beta})\mathbf{g}_i(\boldsymbol{\beta})$, is determined by a simple matrix-vector multiplication. $\boldsymbol{\vartheta}_i$ and $\boldsymbol{\varrho}_i$ are the differences between consecutive parameters and gradient vectors respectively, and are defined as

$$\begin{aligned} \boldsymbol{\vartheta}_i &= \boldsymbol{\beta}_i - \boldsymbol{\beta}_{i-1} \\ \boldsymbol{\varrho}_i &= \mathbf{g}_i - \mathbf{g}_{i-1} \end{aligned} \tag{3.66}$$

The BFGS formula for updating the Hessian matrix $\mathbf{H}_i$ at stage i is given as

$$\mathbf{H}_{i+1} = \mathbf{H}_i + \frac{\boldsymbol{\varrho}_i \boldsymbol{\varrho}_i^{\mathrm{T}}}{\boldsymbol{\varrho}_i^{\mathrm{T}} \boldsymbol{\vartheta}_i} - \frac{\mathbf{H}_i \boldsymbol{\vartheta}_i \boldsymbol{\vartheta}_i^{\mathrm{T}} \mathbf{H}_i}{\boldsymbol{\vartheta}_i^{\mathrm{T}} \mathbf{H}_i \boldsymbol{\vartheta}_i} \tag{3.67}$$

If no initial estimation for $\mathbf{H}_0$ is available, it is initialized with the identity matrix. The first step in updating the Hessian matrix will therefore be equivalent to a gradient descent. Convergence can be checked where the gradient approaches zero, that is

$$\lim_{i \to \infty} |\mathbf{g}_i| = 0 \tag{3.68}$$

Typically a tolerance is chosen to asses the convergence. Convergence is not required for BFGS, unless the function being optimized has a quadratic Taylor series expansion [240] near an optimum.

The inverse of the matrix $\mathbf{H}_{i+1}$ is updated by storing the information of previous calculations and then using the stored information recursively to update the inverse matrix $\mathbf{H}_{i+1}^{-1}$, an idea originally proposed by Bathe and Cimento [16], Matthies [175] and Nocedal [200]. The inverse is calculated by applying the Sherman-Morrison formula [223] to equation (3.67) as follows:

$$\mathbf{H}_{i+1}^{-1} = \left(\mathbf{I} - \frac{\boldsymbol{\vartheta}_i \boldsymbol{\varrho}_i^{\mathrm{T}}}{\boldsymbol{\varrho}_i^{\mathrm{T}} \boldsymbol{\vartheta}_i}\right) \mathbf{H}_i^{-1} \left(\mathbf{I} - \frac{\boldsymbol{\varrho}_i \boldsymbol{\vartheta}_i^{\mathrm{T}}}{\boldsymbol{\varrho}_i^{\mathrm{T}} \boldsymbol{\vartheta}_i}\right) + \frac{\boldsymbol{\vartheta}_i \boldsymbol{\vartheta}_i^{\mathrm{T}}}{\boldsymbol{\varrho}^{\mathrm{T}} \boldsymbol{\vartheta}_i} \tag{3.69}$$

where $\mathbf{I}$ is the identify matrix. In order to avoid temporary matrices and making the computation more efficient, if $\mathbf{H}_{i-1}^{-1}$ is a symmetric matrix, equation (3.70) is computed using

$$\mathbf{H}_{i+1}^{-1} = \mathbf{H}_i^{-1} + \frac{\left(\boldsymbol{\vartheta}_i^{\mathrm{T}} \boldsymbol{\varrho}_i + \boldsymbol{\varrho}_i^T \mathbf{H}_i^{-1} \boldsymbol{\varrho}_i\right) \left(\boldsymbol{\vartheta}_i \boldsymbol{\vartheta}_i^{\mathrm{T}}\right)}{\left(\boldsymbol{\vartheta}_i^{\mathrm{T}} \boldsymbol{\varrho}_i\right)^2} - \frac{\mathbf{H}_i^{-1} \boldsymbol{\varrho}_i \boldsymbol{\vartheta}_i^{\mathrm{T}} + \boldsymbol{\vartheta}_i \boldsymbol{\varrho}_i^{\mathrm{T}} \mathbf{H}_i^{-1}}{\boldsymbol{\vartheta}_i^{\mathrm{T}} \boldsymbol{\varrho}_i} \tag{3.70}$$

Other implementations for calculating the inverse exist [38, 53, 189]. A common extension to BFGS is the limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) which only uses a predefined amount of physical memory [200]. The main problem with BFGS is that, as the number of coefficients q increases, the Hessian matrix grows in both directions, which can lead to large memory consumption. Instead of storing the entire inverse Hessian matrix of size $q \times q$, L-BFGS only has a couple of vectors of size q which implicitly store the approximation values. L-BFGS keeps a history of k previous updates to the matrix, discarding any update memory prior to k. Additionally, the search direction is calculated in a number of steps that are linear in q and k. Due to the linear amount of memory needed, L-BFGS is well suited for approximating a very large number of parameters. A variant of the algorithm, the limited memory Broyden-Fletcher-Goldfarb-Shanno bounded (L-BFGS-B), adds lower and upper bounds constraints on the values of the parameters in L-BFGS [273]. The online limited memory Broyden-Fletcher-Goldfarb-Shanno (O-LBFGS) is an online version based on the idea of L-BFGS-B for both BFGS and L-BFGS [187].

BFGS is a robust maximizer that can be applied in many different situations. The Newton-Raphson method [209] is an alternative maximizer which is not as robust as BFGS, but may converge much faster than BFGS for certain problems, especially where the maximand is quadratic [124]. The Newton-Raphson method updates the parameter by multiplying it with the negative of the inverted Hessian matrix $\mathbf{H}$, where $\mathbf{H}$ is determined analytically [219]. The Newton-Raphson method is defined as

$$\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i + \lambda_i(-\mathbf{H}_i^{-1})\mathbf{g}_i \tag{3.71}$$

where λ_i is the step size that ensures and increase in $\ell(\boldsymbol{\beta}|\mathbf{y})$ during each iteration. Calculating a Hessian matrix is computationally expensive and an analytical Hessian is rarely available. The Newton-Raphson method can also fail to find an increase in $\ell(\boldsymbol{\beta}|\mathbf{y})$ if the function is convex at $\boldsymbol{\beta}_i$, will move in the opposite direction to the slope of the log-likelihood function and make $-\mathbf{H}_i^{-1}$ positive definite [11]. If the function is evaluated at a point where the Hessian is not negative definite, the Newton-Raphson method will fail. Since BFGS is more robust in most cases, it can be used as an initial maximzer. If BFGS fails or does not converge according to the specified tolerance and maximum number of iterations, the Newton-Raphson method can be applied as a second option.

The Berndt-Hall-Hall-Hausman (BHHH) algorithm is an extension to the Newton-Raphson method which uses an information identity in the numerical search for the log-likelihood maximum [19]. BHHH is defined as the iterative procedure,

$$\begin{aligned} \boldsymbol{\beta}_{i+1} &= \boldsymbol{\beta}_i + \lambda_i \mathbf{r}_i \\ \mathbf{r}_i &= -\mathbf{H}_i^{-1}\mathbf{g}_i \\ -\mathbf{H}_i &= \sum_{j=1}^{n} \mathbf{g}_j \mathbf{g}_j^{\mathrm{T}} \\ \mathbf{g}_i &= \sum_{j=1}^{n} \mathbf{g}_j \end{aligned} \tag{3.72}$$

The identify information indicates that the asymptotic variance-covariance matrix of a maximum likelihood estimator is equal to the variance-covariance matrix of the gradient of the likelihood function [19]. According to the central limit theorem [126], the asymptotic distribution of $\hat{\boldsymbol{\beta}}$ is multivariate normally distributed with a mean vector $\bar{\boldsymbol{\beta}}$ and has a variance matrix that is equal to the inverse of the negative Hessian matrix. The variance matrix of $\hat{\boldsymbol{\beta}}$ is expressed as

$$\mathrm{var}(\hat{\boldsymbol{\beta}}) = \left(-\mathrm{E}\left[\mathbf{H}(\bar{\boldsymbol{\beta}})\right]\right)^{-1} \tag{3.73}$$

The variance-covariance matrix can therefore be estimated as the inverse of the outer product of gradients (OPG) [12], therefore

$$\text{var}(\hat{\boldsymbol{\beta}}) = \frac{n}{n-1}\left[\sum_{j=1}^{n} \mathbf{g}_j(\hat{\boldsymbol{\beta}})\mathbf{g}_j(\hat{\boldsymbol{\beta}})^{\mathrm{T}}\right]^{-1} \tag{3.74}$$

The numerical optimization procedure of the BHHH algorithm is summarized in algorithm 2.

Algorithm 2 The Berndt-Hall-Hall-Hausman algorithm.

determine the convergence tolerance
determine an initial vector of parameters $\boldsymbol{\beta}_i$
repeat
  calculate $\mathbf{H}(\boldsymbol{\beta}_i)$ by the OPG
  calculate the direction vector $\mathbf{r}_i = [-\mathbf{H}(\boldsymbol{\beta}_i)]^{-1}$
  set $\lambda = 0$
  repeat
    increment λ by one
    calculate the new vector $\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i + \lambda \mathbf{r}_i$
  until $f(\boldsymbol{\beta}_i + \lambda \mathbf{r}_i) \leq f(\boldsymbol{\beta}_i + (\lambda - 1)\mathbf{r}_i)$
until convergence tolerance reached

The convergence tolerance is set to a value just above zero, commonly 0.0001 [11, 12]. BHHH is often used in conjunction with CML.

3.9 Autoregressive Moving Average Model

The autoregressive moving average (ARMA) model is a combination of the AR and MA models. Using Fourier and Laurent series with statistical interference, Whittle proposed the ARMA model in his PhD thesis [262]. The model was later popularized by Box and Jenkins who described a method for determining the model orders and an iterative method for estimating the model coefficients [29]. The ARMA model is defined as

$$y_t = c + \varepsilon_t + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=1}^{q} \beta_i \varepsilon_{t-i} \tag{3.75}$$

where p and q are the AR and MA model orders respectively. If $p = 0$, the ARMA model reduces to the MA model; equally, if $q = 0$ the ARMA model reduces to the AR model. ARMA(p, q) is a common notation to indicate the orders for the AR and MA parts. The ARMA model parameters are typically fitted using the MLE approach.

It is important to select the correct model and the values for the model orders p and q, since it directly influences the prediction accuracy. Choosing an appropriate model for the given data is known as *model selection* and choosing the order that will best fit the given data is known as *model order selection*. The orders should be large enough to model the complexity of the given data, but on the other hand should be as small as possible to eliminate possible overfitting of the function. A good practice is to choose the smallest values for p and q that provide an acceptable fit and is as close as possible to the true observations [206]. There are three approaches for choosing the ARMA model orders, namely fixed orders, the autocorrelation function and information criteria. The first approach uses fixed values for p and q. Although easy to implement and without any computational overhead, the fixed values might not always represent the best model order for all sample subsets.

The second approach makes use of the autocorrelation function (ACF) and the partial autocorrelation function (PACF) in equations (2.50) and (2.51) to (2.53) respectively [142]. Once the ACF and PACF are calculated, they can be plotted to determine if the AR, MA or ARMA model should be used. A confidence level is added to the plot to determine the optimal order of the selected model. The confidence intervals are typically set at 95% [271]. The confidence interval for PACF is the same for all lags and is defined as the standard error with 95% of the PACF sequence falling inside the confidence interval as follows:

$$\mathrm{SE}_{\mathrm{pacf}} = \frac{\pm 1.96}{\sqrt{n}} \tag{3.76}$$

where n is the number of observations. Note that the 1.96 is often rounded up to 2 in academic literature, which is not an accurate representation of the 95% confidence interval. Additionally, equation (3.76) is also often used for the confidence interval of the ACF. Since equation (3.76) is a test for randomness and ACF, unlike PACF, is not calculated recursively, determining the standard error in this way may not provide an

accurate confidence interval for ACF. The 95% ACF confidence should be calculated by using Bartlett's formula:

$$\mathrm{SE}_{\mathrm{acf}} = \pm 1.96 \sqrt{\frac{1 + 2\sum_{i=1}^{t} \mathrm{acor}_i^2}{n}} \tag{3.77}$$

where acor_i is the autocorrelation in equation (2.50). The standard error therefore uses the autocorrelations of all previous lags. Since the first autocorrelation has no previous lags, acor_1 is set to zero. The AR model order p is chosen as the last PACF lag that falls outside the confidence interval, $\mathrm{SE}_{\mathrm{pacf}}$. Similarly, the MA model order q is the last ACF lag confidence interval, $\mathrm{SE}_{\mathrm{acf}}$. More specifically, the model type and order is selected according to the rules given in table 3.1.

Table 3.1: The model selection rules of AR, MA and ARMA models using ACF and PACF.

Model	**ACF**	**PACF**
AR(p)	Tails off gradually	Cuts off after p lags
MA(q)	Cuts off after q lags	Tails off gradually
ARMA(p,q)	Tails off gradually	Tails off gradually
AR(p) or MA(q)	Cuts off after q lags	Cuts off after p lags
Neither (random process)	No spike outside confidence	No spike outside confidence

The third approach for selecting the model orders is using information criteria, which determine how well a model with a certain order fits the observed dataset. The best known criteria for linear models is the coefficient of determination R^2, defined as

$$\mathrm{R}^2 = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.78}$$

where the residual sum of squares (RSS) is calculated with $\tilde{x}_i$ as the predicted value of x_i and the total sum of squares (TSS) with a sample mean $\bar{x}$ over n observations. R^2 does not penalize the inclusion of additional parameters. Since the overuse of parameters mostly does not increase the estimation accuracy much, compared to the decrease in computational efficiency, the lowest model order with an acceptable fit should be selected. The adjusted R^2 which penalizes additional parameters is given as

$$R_a^2 = 1 - \frac{\mathrm{RSS}/(n-m)}{\mathrm{TSS}/(n-1)} = 1 - \frac{(n-1)\sum_{i=1}^{n}(x_i - \tilde{x}_i)^2}{(n-m)\sum_{i=1}^{n}(x_i - \mu_x)^2} \tag{3.79}$$

where m is the number of parameters. Since R^2 can only be used for linear models, in this case the AR model, a different criterion is needed for MA and ARMA model order selection. The Akaike information criterion (AIC) measures the relative model quality with respect to a given set of parameters [2]. AIC is defined as

$$\mathrm{AIC} = 2m - 2\ell(\hat{\beta}) \tag{3.80}$$

where m is the number of independently adjusted parameters and $\hat{\beta}$ the set of parameters that maximises the log-likelihood function $\ell(\hat{\beta})$. A model is continuously evaluated with different sets of parameters. The set of parameters for the model that has the lowest AIC value is selected as best solution for the current sample set. The AIC for models estimated by least squares is calculate as

$$\mathrm{AIC}_{\mathrm{LS}} = n(1 + \ln(2\pi) - \ln(n)) + n\ln(\mathrm{RSS}) + 2m \tag{3.81}$$

Although AIC favours parsimony, it does not go far enough with the penalization of additional parameters. If there are two models with m and $m-1$ parameters respectively, taking the null hypothesis $m = 0$ into account, AIC will select the parsimonious model for large datasets about 16% of the time [49]. Schwarz proposed the Bayesian information criterion (BIC) which puts a greater emphasis on the penalty for additional parameters [221]. BIC is defined as

$$\mathrm{BIC} = m\ln(n) - 2\ell(\hat{\beta}) \tag{3.82}$$

Although BIC reduces the chances of overfitting, it is not derived from the principles of information like AIC. Additionally, BIC has a prior distribution of $\frac{1}{c}$, where c is the number of models being evaluated, which should rather be a decreasing function of the number of parameters m. These limitations contributed to the proposal of a corrected version of AIC, known as the Akaike information criterion corrected (AICC), which penalizes additional parameters more, but still retains the benefits of AIC. AICC is calculated as

$$\text{AICC} = \text{AIC} + \frac{2m(m+1)}{n-m-1} = 2m + \frac{2m(m+1)}{n-m-1} - 2\ell(\hat{\beta}) \tag{3.83}$$

Simulations indicated that AICC has a computational advantage over BIC [37]. A number of other information criteria exist that can be used for model order selection. The Hannan-Quinn information criterion (HQIC) uses the law of iterated logarithms to penalize additional parameters [119]. The deviance information criterion (DIC) uses the mean model-level deviance and the model complexity to estimate the expected loss for a certain set of parameters [232]. The focused information criterion (FIC) directs its attention directly to the parameter of primary interest, rather than following the approach of AIC, BIC and DIC which assess the overall quality of candidate models [45, 129]. Model selection using information criteria is not a reliable approach and has gained widespread discussion in academic literature. The success of various information criteria choosing the correct model greatly depends on the characteristics of the data, the presence and degree of outliers, the set of candidate models to choose from and the number of samples used to determine the model. Benchmarking showed that on average only 62% [164] and 63.8% [13] of the time the correct AR model was selected for various information criteria for a set of 100 samples or lower. Similar only 61.3% [3] of the time the correct MA model was selected for 100 samples or lower.

Other approaches for model selection using particle swarm optimization [250, 250], genetic algorithms [184, 201] and artificial neural networks [1, 4] were also proposed. The latter three approaches usually take longer to find the optimal parameters than to solve the model itself, often without a major increase in accuracy for the given parameters. Therefore, for continuous model selection and evaluation, a fixed suboptimal model approach is advisable if computationally power is limited.

3.10 Autoregressive Integrated Moving Average Model

The autoregressive integrated moving average (ARIMA) model is a generalization of the ARMA model which is applied if the observed data shows characteristics of

non-stationarity, such as seasonality, trends and cycles [29]. A differencing operation is added as an initial step to the ARMA model to remove possible non-stationarity. The ARMA model in equation (3.75) can also be expressed in terms of the lag operator as

$$\alpha(L)y_t = \beta(L)\varepsilon_t \tag{3.84}$$

where $\alpha(L)$ is said to be the lag polynomial of the AR model given as

$$\alpha(L) = 1 - \sum_{i=1}^{p} \alpha_i L^i \tag{3.85}$$

and $\beta(L)$ the lag polynomial of the MA part as

$$\beta(L) = 1 + \sum_{i=1}^{q} \beta_i L^i \tag{3.86}$$

The lag or backshift operator produces the previous values of a time series y with $n+1$ data points as follows:

$$Ly_t = y_{t-1} \tag{3.87}$$

for $t \in \{1, 2, \ldots, n\}$. The operator is expressed for any lag i as

$$L^i y_t = y_{t-i} \tag{3.88}$$

The first difference operator is the difference between the point of interest and its previous point, defined as

$$\Delta y_t = y_t - y_{t-1} = (1 - L)y_t \tag{3.89}$$

This is generalized to any Δ^d for positive integers d as follows:

$$\Delta^d y_t = (1 - L)^d y_t \tag{3.90}$$

The ARIMA model is expressed by expanding equation (3.84) and incorporating the difference operator as follows:

$$\left(1-\sum_{i=1}^{p}\alpha_i L^i\right)(1-L)^d y_t = \left(1+\sum_{i=1}^{q}\beta_i L^i\right)\varepsilon_t \tag{3.91}$$

where p is the AR order, q the MA order and d the order of integration. The ARMA model assumes a zero mean for the observed data, which may not always be the case if a series is processed in subsets that have a non-zero mean. The ARMA model in equation (3.84) is expressed as a collection of parametrized terms as

$$y_t = \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p} + \varepsilon_t + \varepsilon_1\beta_{t-1} + \cdots + \varepsilon_q\beta_{t-q} \tag{3.92}$$

Incorporating μ as the non-zero mean of y_t changes the model in equation (3.92) to

$$z_t = \alpha_1 z_{t-1} + \cdots + \alpha_p z_{t-p} + \varepsilon_t + \varepsilon_1\beta_{t-1} + \cdots + \varepsilon_q\beta_{t-q} \tag{3.93}$$

where $y_t = z_t + \mu$, or more explicitly by subtracting the non-zero mean from y_t, that is $z_t = y_t - \mu$. A non-zero mean can be incorporated into the ARIMA model as follows:

$$\left(1-\sum_{i=1}^{p}\alpha_i L^i\right)(1-L)^d y_t = \left(1+\sum_{i=1}^{q}\beta_i L^i\right)\varepsilon_t + \tilde{\mu} \tag{3.94}$$

where $\tilde{\mu}$ are the AR parametrized non-zero mean terms calculated by expressing equation (3.93) in terms of y_t instead of z_t, such that

$$\tilde{\mu} = \mu - \sum_{i=1}^{p}\alpha_i\mu \tag{3.95}$$

ARIMA(p, d, q) is typically used to indicate an ARIMA model with specific orders. An ARIMA model with $d = 0$ reduces it to the ARMA model.

3.11 Autoregressive Conditional Heteroskedasticity

ARMA models are the conditional expectation of a process with a conditional variance that stays constant for past observations. This means that ARMA models use the same conditional variance, even if the latest observations indicate a higher or lower variation. The autoregressive conditional heteroskedasticity (ARCH) model was developed by Engel

for financial markets that show periods of low volatility followed by periods of high volatility and vice versa [69] and earned him the Nobel Prize in Economic Sciences in 2003. Music data shows similar characteristics where the start of refrains, verses or beats produce sudden bursts caused by *upbeats*, leading to an increase in variance. These bursts are then often followed by an unaccented stroke, temporarily decreasing the variance of the sound wave. However, music signals typically oscillate less than most financial markets and have, therefore, a comparatively lower variance.

A variable is said to be heteroskedastic if subsets of its population have different variabilities. The variability is quantified using a statistical dispersion, such as the standard deviation or variance. ARCH achieves nonconstant conditional variance by calculating the variance of the current error term ε_t as a function of the error terms ε_{t-i} in the previous i time periods. Therefore the forecasting is done on the error variance at time t, compared to the AR model which does its prediction directly on the previously observed values. The ARCH process for a zero mean series is defined as

$$y_t = \sigma_t \varepsilon_t \tag{3.96}$$

where ε_t is Gaussian white noise and σ_t is the conditional variance, modelled by an AR process as

$$\sigma_t = \sqrt{\alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2} \tag{3.97}$$

The ARCH model is typically denoted as ARCH(q), where q is the order of the ARCH model. In order to ensure that the variance is always nonnegative, $\alpha_i \geq 0$ must hold true for $i \in \{0, 1, \ldots, q\}$. If $\alpha_i = 0$ for $i \in \{1, 2, \ldots, q\}$, then $\alpha_0 > 0$. If the sum of the coefficients α_i is less than one, the ARCH process is weakly stationary and will have a constant unconditional variance, that is

$$\sigma_t = \sqrt{\frac{\alpha_0}{1 - \sum_{i=1}^{q} \alpha_i}} \tag{3.98}$$

Since ARCH makes use of an AR model, the coefficients α_i can be estimated using LLS with Yule-Walker equations. Since the distribution of ε_{t-i}^2 is naturally not normal, the

Yule-Walker approach does not provide an accurate estimation for the model coefficients, but can be used to set the initial values for the coefficients. An iterative approach, such as MLE, is then used to refine the coefficients in order to find a better fit. Similar to the ARMA model, the PACF or various information criteria can be used to determine the ARCH model order. Engel proposed another method for determining heteroskedasticity and if ARCH is the correct model for the observed data or not [69]. Under the null hypothesis, the model is a standard dynamic regression model with ε_t as Gaussian white noise, which can be tested with the Lagrange multiplier test. The alternative hypothesis is that ε_t is not Gaussian white noise, but an ARCH process. Engel later extended his method using the statistic TR^2 where T are the residuals and R^2, given in equation (3.78), is calculated from the regression of ε_t^2 [70]. Under the null hypothesis with no ARCH errors, TR^2 follows a $\mathcal{X}^2$ distribution with q degrees of freedom. If TR^2 is greater than the chi-square the null hypothesis is rejected, meaning that the process has an ARCH effect. Other related approaches for testing heteroskedasticity includes the White test [260], the one-sided ARCH effect test [52], and the locally most mean powerful based score test [160].

3.12 Generalized Autoregressive Conditional Heteroskedasticity

The generalized autoregressive conditional heteroskedasticity (GARCH) is a generalization of the ARCH model proposed by Bollerslev which also uses the weighted average of past squared residuals without the declining weights ever reaching zero [27]. Instead of assuming an AR model in equation (3.97), GARCH uses an ARMA model for the error variance as follows:

$$\sigma_t = \sqrt{\alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2} \tag{3.99}$$

The notation GARCH(p, q) is used for GARCH models with an ARCH degree of q and a GARCH degree of p. Since GARCH makes use of the ARMA model for the error

variance, the model can not be estimated using LLS regression, but has to follow the same estimation approach used by ARMA, such as the MLE.

The Engel test can be used to test for ARCH errors. In order to test for GARCH errors, the autocorrelations of ε are calculated using equation (2.50). Using the Ljung-Box Q-test [165], the first k autocorrelations are evaluated to determine if they have a collectivity small magnitude. The Q-test is defined as

$$\mathcal{Q} = n(n+2)\sum_{i=1}^{k}\frac{r_i^2(\varepsilon)}{n-i} \tag{3.100}$$

where n is the number of samples and $r_i(\varepsilon)$ the sample autocorrelation of ε at lag i. Under the null hypothesis, the Q-test statistic is asymptotically $\mathcal{X}^2(k-p-q)$ distributed. The null hypothesis is rejected at level θ, indicating that GARCH errors exist, if $\mathcal{Q}$ exceeds the $(1-\theta)$-quantile of the $\mathcal{X}^2(k-p-q)$ distribution. The significance level θ is typically chosen at 5% [99].

3.13 Artificial Neural Networks

An artificial neural network (ANN) is a model of a biological neural network with a network structure that consists of a number of layered artificial neurons that are connected with artificial synapses. This section discusses the basics of ANNs, the workings of an artificial neuron, different activation functions, the configuration of various ANN architectures and how an ANN is trained.

3.13.1 Artificial Neurons

An artificial neuron (AN) or perceptron is the fundamental functional component of an ANN. An AN takes one or more input signals and produces an output signal using a mathematical function, also referred to as an activation function. Figure 3.2 depicts the process of an AN.

The first task of an AN is to calculate the net input by combining the input signals s_i with the weights w_i for each signal. A typical approach is to use the weighted sum of the inputs,

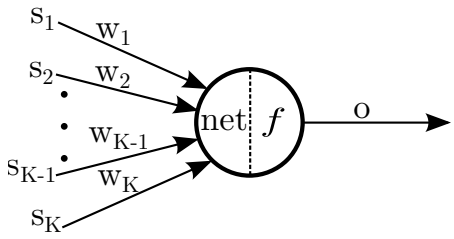


Figure 3.2: An artificial neuron.

$$net = \sum_{i=1}^{I} w_i s_i \tag{3.101}$$

An alternative approach is to calculate the net input using the product of exponential inputs [61], defined as

$$net = \prod_{i=1}^{I} s_i^{w_i} \tag{3.102}$$

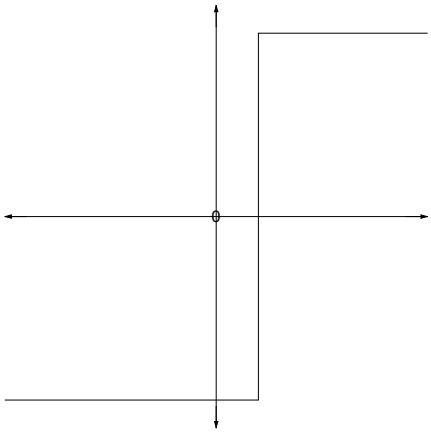
The second task of the AN is to pass the net input through an activation function f in order to produce the output signal o. A threshold θ is often added to the AN. In order to simplify training, the threshold is added as an additional input with a value of -1, referred to as a bias unit. The bias changes the net input in equation (3.101) to

$$net = \sum_{i=1}^{I} w_i s_i - \theta \tag{3.103}$$

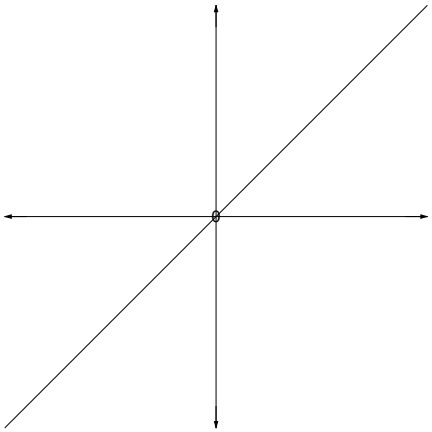
where $\theta = w_{I+1} s_{I+1}$. The bias can be added to the product of exponential inputs in a similar fashion.

3.13.2 Activation Functions

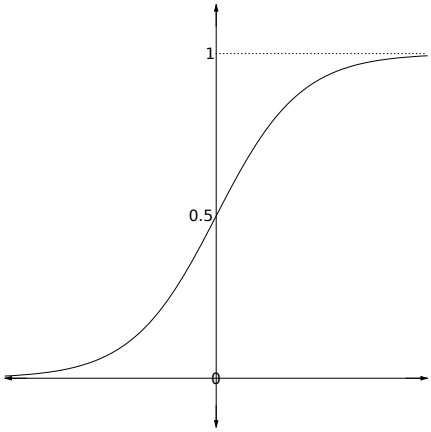
The net input and bias must be passed through an activation function in order to determine the output signal o. The weights of the inputs and the type of activation function determine the strength of the AN output. Activation functions typically produce outputs in the range of $[0, 1]$ or $[-1, 1]$. Figure 3.3 illustrates some widely used activation functions.



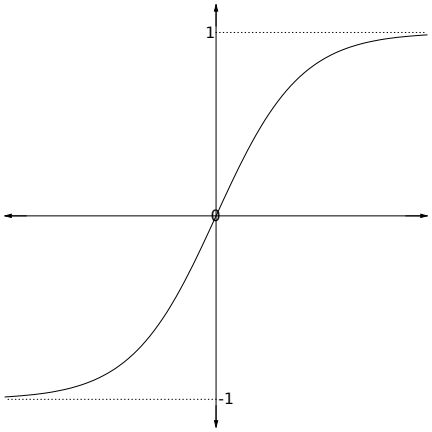
(a) The step activation function.



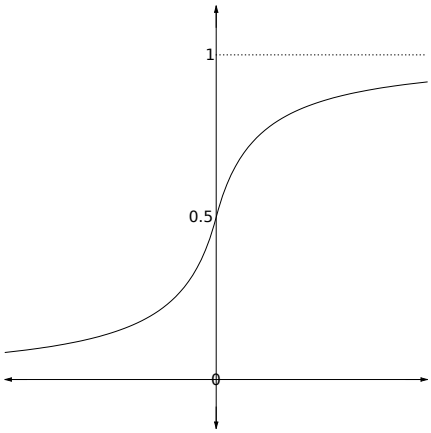
(b) The linear activation function.



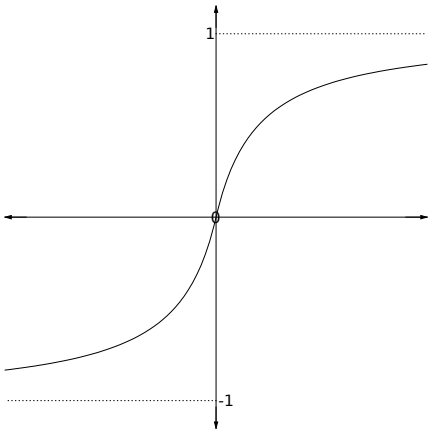
(c) The sigmoid activation function.



(d) The hyperbolic tangent activation function.



(e) The Elliot activation function.



(f) The symmetric Elliot activation function.

Figure 3.3: Various artificial neural network activation functions.

The step activation function produces a binary output of either α or β as follows:

$$f(net-\theta) = \begin{cases} \alpha & \text{for } net \geq \theta \\ \beta & \text{otherwise} \end{cases} \tag{3.104}$$

The linear activation function is unbounded and produces a modulated output with gradient α as follows:

$$f(net-\theta) = \alpha(net-\theta) \tag{3.105}$$

The more frequently used sigmoid activation function produces an output in the range of $(0,1)$ using

$$f(net-\theta) = \frac{1}{1+e^{-\alpha(net-\theta)}} \tag{3.106}$$

where α controls the steepness of the function and is usually set to one. In order to produce an output in $(-1,1)$ with the sigmoid function, a symmetric version known as the hyperbolic tangent function can be used as follows:

$$f(net-\theta) = \frac{e^{\alpha(net-\theta)} - e^{-\alpha(net-\theta)}}{e^{\alpha(net-\theta)} + e^{-\alpha(net-\theta)}} \tag{3.107}$$

Since the exponential function has to be evaluated at least twice, the function can be approximated more efficiently with

$$f(net-\theta) = \frac{2}{1+e^{-\alpha(net-\theta)}} - 1 \tag{3.108}$$

Even with a single exponential operation, the sigmoid function can be computationally expensive when evaluated numerous times. Elliot proposed an activation function that produces similar results to the sigmoid function, but is more efficient [65]. The Elliot function produces outputs in $(0,1)$, and is defined as

$$f(net-\theta) = \frac{\frac{net-\theta}{2}}{1+|net-\theta|} + \frac{1}{2} \tag{3.109}$$

The symmetric function in $(-1,1)$ is even more efficient to calculate, and is defined as

$$f(net - \theta) = \frac{net - \theta}{1 + |net - \theta|} \tag{3.110}$$

Even though the Elliot function is faster than the sigmoid function to calculate, it reaches its extremes more slowly. The output error will therefore in general be greater, requiring more training iterations to reach the desired error. This problem was observed by benchmarking the sigmoid and Elliot functions on a backpropagation ANN [226].

3.13.3 Architecture

By itself, an AN is capable of learning linearly separable functions if the summation unit is used [68]. In order to realize complex non-linearly separable functions, ANs are combined into a network architecture. A widely used architecture is the feed forward artificial neural network (FFANN), consisting of an input, one or more hidden, and an output layer. Each layer contains one or more ANs which are fully connected to all ANs in the previous and next layer. Figure 7.22 illustrates a FFANN with a single hidden layer.

The complexity of the FFANN is increased by adding additional ANs to layers or by adding additional hidden layers. It was proven that a single hidden layer with a sufficient number of ANs can be used to model any continuous function [25, 132, 133].

Discrete values from the signal being processed are provided to the ANs in the input layer. The inputs are then propagated to all connected ANs in the hidden layer. The signal is passed through the activation function and the resulting $f(net - \theta)$ is then forwarded to the next hidden layer until the signal finally reaches the output layer. The forward propagation to calculate the outputs of a FFANN with a single hidden layer is formalized as

$$\begin{aligned} o_k &= f_{o_k}\left(\sum_{j=1}^{J+1} w_{j,k} f_{h_j}\left(net_{h_j}\right)\right) \\ &= f_{o_k}\left(\sum_{j=1}^{J+1} w_{j,k} f_{h_j}\left(\sum_{i=1}^{I+1} v_{i,j} g_{i,p}\right)\right) \end{aligned} \tag{3.111}$$

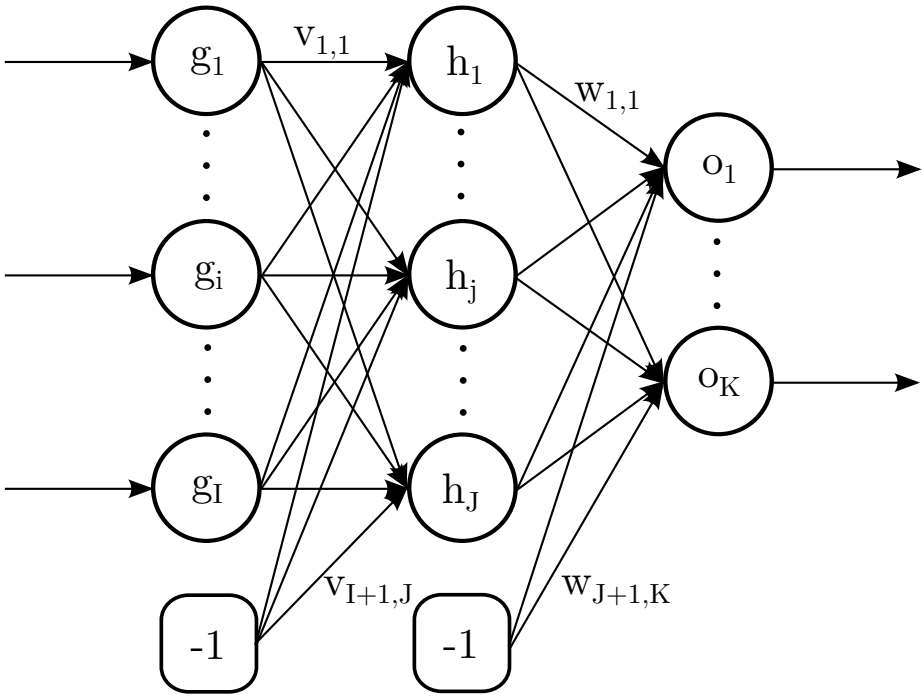


Figure 3.4: A feed forward neural network.

where g_i are the ANs in the input, h_j the ANs in the hidden layer and o_k the ANs in the output layer as illustrated in figure 7.22. $v_{i,j}$ and $w_{j,k}$ are the connections between the input and hidden, and hidden and output units respectively. f_{o_k} and f_{h_j} are the activation functions for specific units on the output and hidden layer. A FFANN can therefore be regarded as a non-linear mapping of an input signal to a desired output signal.

Another architecture, the time delay artificial neural network (TDANN) [157], is often used for the processing of time series signals. On initialization, the TDANN has a value for only the first input AN, all other inputs are set to zero. Once the first pattern was propagated through the TDANN and a second pattern is available, all input values are shifted ahead by one. Therefore, a time delay t is introduced with a total of τ previous shifts. The time delay is formalized as

$$g_k(t-\tau) = g_k(t-\tau+1) \tag{3.112}$$

for $\tau \in \{1, 2, \ldots, T\}$. Hence, a total of T patterns are used during training. The forward propagation to calculate the outputs in equation (3.111) changes to

$$o_k = f_{o_k}\left(\sum_{j=1}^{J+1} w_{j,k} f_{h_j}\left(\left[\sum_{i=1}^{I}\sum_{t=1}^{T} v_{i(t),j} g_{i(t)}\right] + v_{I+1,j} g_{I+1}\right)\right) \tag{3.113}$$

Feedback connections can also be added to the network, allowing the FFANN to learn the temporal characteristics of the input signal. These networks are known as simple recurrent artificial neural network (SRANN). If one for instance tries to predict the next few samples of a time series, one sample at a time, the output of the previous forward propagation of the hidden layer or output layer can be used as an additional input neuron for the next propagation. A feedback connection is therefore established between the output and input layer, which is known as a Jordan SRANN [146]. The output of a Jordan SRANN is calculated as

$$o_k = f_{o_k}\left(\sum_{j=1}^{J+1} w_{j,k} f_{h,j}\left(\sum_{i=1}^{I+K+1} v_{i,j} g_i\right)\right) \tag{3.114}$$

where an additional K ANs are added to the input layer which are linked with a feedback connection from the K output ANs. An alternative approach to the Jordan SRANN, is to create feedback connections between the hidden and input layers by duplicating the activation of the hidden ANs and adding them as additional ANs to the input layer. This architecture is known as the Elman SRANN [66]. A novel approach in time series is to combine TDANN and SRANN into a simple recurrent time delay artificial neural network (SRTDANN) in order to learn the temporal characteristics of the latest samples in the series [104, 108, 149].

Choosing the correct size and structure for the ANN is a problematic process. Finding the optimal structure that can accurately model the given signals requires the optimization of the number of neurons and hidden layers which can be computationally expensive. Overfitting may occur during training when the network size is too large and training continues for too long with a dataset that contains noise. Overfitting occurs when the ANN

structure has an excess degree of freedom, that is an excess number of weights, allowing the ANN to remember noisy inputs, that would otherwise have been discarded more quickly. Different approaches exist to mitigate the problem of choosing the network size. Pruning removes unfavourable ANs and weights in order to improve the generalization [67, 185]. Unfavourable ANs and weights are those that have no or a statically insignificant effect on the ANN's error rate and only increase the computational time to evaluate and train the ANN. A cascade artificial neural network (CANN) starts with the simplest architecture, only consisting of direct connections between the input and output layers [79]. A set of ANs are trained separably and the most promising of the candidate ANs are then added to the ANN. These candidate units are systematically added in order to improve the output accuracy. All input units are connected to all hidden units, where the output of the hidden units serve as the inputs to all succeeding hidden and output ANs. Although CANN relieves the responsibility of choosing the network size, it can be computationally expensive to train when a large number of candidate ANs are maintained and trained separably.

3.13.4 Learning

By itself, an ANN is not capable of adapting to the characteristics of the input signal. A training algorithm is required to adjust the weights of the ANN in order to capture the input characteristics and reduce the output error. The process of weight adjustments is known as *learning*. Three major paradigms are employed for ANN training, namely supervised, unsupervised and reinforcement learning.

- **Supervised learning:** Supervised learning is used for pattern recognition, classification problems and regression. Supervised learning requires prior knowledge of the problem by making use of a dataset containing the inputs and desired outputs. The ANN outputs are calculated based on the inputs and compared to the desired outputs. The aim is to reduce the output error by adjusting the weights of the ANN, which can be done using gradient descent. A widely used algorithm is the backward propagation of errors, or backpropagation for short, which utilizes gradient descent to find a local minimum. Backpropagation passes the error signal from

the output layer, through the hidden layers and uses the error signal to adjust the weights [217, 259]. Backpropagation requires the activation and error functions to be differentiable. Backpropagation can be applied in three different modes, namely incremental, stochastic and batch training. During incremental or online training, the weights of the ANN are updated after each presentation of a new training pattern. Incremental backpropagation may be beneficial to TDANN, since the last weight updates where done with the most recent patterns. Stochastic training also updates the weights after each pattern propagation, but randomly selects the pattern from the set instead of a sequential selection. Similar to incremental training, stochastic training may reduce the error on the single pattern, yet increase the error on the entire training set. However, an increase in the number of training patterns reduces the influence of the weights' adjustment of a single outlier pattern on the error of the entire training set. Batch training presents all patterns to the ANN before adjusting the weights. The summation of the weight updates of all patterns are used for training instead of updating the weights for each individual pattern. Batch learning reduces the risk of an ANN unlearning what was learned in previous steps, which can also be achieved though a learning momentum in stochastic training by adding a fraction of the previous weight update to the current one. A variation of the backpropagation algorithm is *quickprop* which is loosely based on Newton's method [78]. Quickprop requires the second order derivatives of the error function and tries to approximate the error surface with a quadratic polynomial (parabola). In general, quickprop trains faster than the standard backpropagation algorithm, since it attempts to use a single step to jump from the current gradient position directly into the minimum of the parabola. Resilient backpropagation, or *Rprop* for short, is another widely used learning algorithm that adds a penalization when the previous weights adjustment was too large, causing the ANN to jump over local minima [214]. If the partial derivative of a weight changes sign compared to the previous iteration, Rprop decreases the weight update by a factor η^- for $\eta^- < 1$. If the sign of the partial derivative does not change, the weight update is increased by a factor η^+ for $\eta^+ > 1$. Riedmiller and Braun suggested η^- to be fixed at 0.5 and η^+ at 1.2 [215]. An improved version of Rprop, known as iRprop, decreases the training

time and was shown to outperform the standard Rprop and quickprop algorithms [138]. Both, Rprop and iRprop, have two variants each. Rprop$^+$ and iRprop$^+$ make use of weight-backtracking, whereas Rprop$^-$ and iRprop$^-$ omit weight-backtracking [137]. Other supervised learning algorithms include the Widrow-Hoff rule [178], the generalised delta rule [22] and particle swarm optimization (PSO) [118, 246].

- **Unsupervised learning:** Unsupervised learning only requires a set of inputs, but no desired outputs. The ANN attempts to find patterns without any feedback from error signals or rewards. Approaches in this category include clustering [58], compression [231], classification [167] and self organizing maps (SOM) [152].

- **Reinforcement learning:** Unlike supervised learning, reinforcement learning does not rely on a set of input-output pairs for training. Reinforcement learning is typically used for sequential decision making and control tasks. The ANN interacts with the environment and is penalized with a negative signal or rewarded with a positive signal based on its performance. The weights are updated through the rewards and penalties until it produces favourable outputs. Popular reinforcement learning algorithms include Q-learning [256] and methods of temporal differences [236].

3.14 Summary

This chapter discussed a number of polynomials and models that can be used to approximate a function. The polynomials include the standard, Fourier, Newton, Hermite, Lagrange polynomials and splines. More advanced modelling of audio signals can be through the AR, MA, ARMA, ARIMA, ARCH or GARCH models. ANNs provide an additional level of complexity, by allowing the model to learn from previous mistakes and adapt accordingly in order to produce more accurate results. These polynomials and models can be used to detect and reconstruct noise in audio signals which is discussed in the upcoming chapters 4 and 5.

Chapter 4

Noise Detection

Noise in gramophone recordings is a result of frequent use and mishandling of the record. The noise can be caused by scratches, dust particles, long exposure to sunlight or other sources of heat, extensive playback, rumble caused by the turntable mechanics and vibrations from the speakers or other external factors in the room. Although all these aspects contribute towards the degradation of the reproduced audio, scratches cause the most audible disruptions as a result of a high deviation of the signal's amplitude. Individual samples that are affected by scratches diverge from the surrounding unaffected parts of the signal. In order to reconstruct the disrupted samples, they first have to be identified and flagged as disrupted, a process know as noise or outlier detection.

Outlier detection methods are broadly classified into time and frequency domain approaches [114, 248]. Frequency domain techniques discover abnormalities in certain frequencies. This is often achieved by comparing the frequency amplitudes calculated from smaller time delay sample windows to those of larger parts of the signal, thereby correlating temporal with population frequencies. Time domain techniques analyse the displacement of an individual or a group of samples from the time delay signal with relation to the non-corrupted parts of the signal. The noise causes impulse disturbances in the sound wave, referred to as outliers in statistical analysis and considered to be isolated disruptions in the samples' amplitudes. These variances have a unity length and a certain probability of occurrence. In practice, however, noise from gramophone recordings are often not isolated, with disturbances occurring close to each other. If only

a single sample is affected by the noise, the outlier is univariate. If outliers occur in a joint combination of consecutive samples, they are multivariate.

Some outliers from noisy gramophone recordings, such as those caused by dust particles and turntable rumble, are not easily detectable. These outliers only cause a minor interference, making it difficult to distinguish the noise from the signal. The typical approach to this problem is to apply a smoothing filter over the entire signal. Various techniques were introduced, such as smoothing Kalman filters [56, 192], Monte Carlo Bayesian filtering [166], an Ephraim and Malah noise suppressor [39], and two-pass split-window filtering [71]. Although smoothing filters are able to reduce much of the noise, they also sift the parts of the signal that do not contain outliers, leading to the adjustment of non-corrupted samples. Additionally, audiophiles often listen to records because of the unique listing experience caused by imperfections of the gramophone medium and playback devices. This thesis therefore focuses only on multivariate outliers that cause notable disruptions, and other types of minor noise is left for future research.

Since gramophone records have been around for many years, numerous approaches were published over the years that attempt to reduce the noise caused by scratches. Niedźwiecki proposed a method where mono records were played back with a stereo turntable, generating two identical signals from one groove [192]. Since most scratches only affect one side of the groove, the two signals are correlated with each other to detect disturbances. This method is problematic when scratches affect both sides of the groove and will be unusable if stereo recordings are used, which is the case with most records.

In his more recent research, Niedźwiecki used bidirectional processing in the time domain to eliminate impulse disturbances of gramophone recordings [194]. This technique relies on unidirectional detection techniques that process the signal from start to end, and then in reverse order, from the last to the first sample. By adapting the existing forward-time algorithms to also include backward-time detection, it was found that the detection accuracy was increased, comparable to state-of-the-art commercial audio-based outlier detectors. This approach can only be conducted on prediction-based algorithms, such as the AR model and also requires the availability of the entire signal before commencing the processing, making it unsuitable for real-time noise detection.

Niedźwiecki and Ciołek also proposed a model-based predictor that matches highly

repetitive click patterns to a set of previously generated noise templates [196]. The main problem with pattern matching is that the click templates must be generated prior to the detection phase and noise that does not match any of the given templates will go undetected. The empirical tests were conducted on only five gramophone recordings with a subjective perceptual evaluation of 20 participants.

Another recent research project makes use of multiple copies of the same gramophone record to refurbish the audio signal, with the hope that the scratches and damages do not occur in the same place on the different copies [233]. The recorded signals are then aligned to detect and eliminate outliers. Although this approach has a very high restoration accuracy, it is infeasible in practice. In most cases it is difficult to obtain multiple copies of the same record, especially for the older archived ones. Additionally, damages that are similar and occur in the same place on all copies, such as the beginning of a record where the needle enters the groove, are not detected and corrected.

Czyzewski proposed a FFT-based click detector by subtracting the frequency spectrum of the impulse-related part of the signal from the whole spectral representation of the signal, therefore, highlighting the frequencies of the signal segments which are affected by noise [47]. Czyzewski's research also proposed the training of an ANN to learn two classes of noise in order to detect outliers. This ANN approach requires extensive training prior the the detection phase and certain classes of noise may go undetected.

Most of these approaches are very limited, since they require special playback equipment, multiple records, or have some restrictions on how the data should be processed. This chapter discuses a number of generic outlier detection algorithms without these kind of restrictions. The algorithms include the standard score, median absolute deviation, Mahalanobis distance, nearest neighbour deviation, mean absolute spectral deviation, and absolute predictive deviation. Each of these algorithms produces a per-sample noise map which is then analysed with a direct, mean and maximum thresholding technique to produce a noise mask.

4.1 Noise Mapping

This section discusses a number of algorithms to detect outliers and to generate a noise map. The noise map has an entry for each sample in the audio signal. Each of these entries are in the range of $[0, k]$, where 0 indicates that the sample is an inlier and not affected by any noise and k indicates that the sample represents pure noise. The maximum value for k is determined by the individual algorithms. The algorithms are applied in a moving window fashion on series y_i, generating a value for the noise map m at time delay t as follows:

$$m_t = d\left((y_i)_{i=t-u}^{t+v}\right) \tag{4.1}$$

where d is the noise detection algorithm and u and v the number of samples to the left and right of y_t respectively. The values for u and v are determined by the individual algorithms, but always adhere to $(u + v) = (n - 1)$, where n is the size of the moving window. If n is even and the detection algorithms require t to be centred between u and v, the right-hand side of the moving window is reduced by one sample, that is $v = (u - 1)$.

The purpose of this section is to provide a theoretical overview of different outlier detection methods. The standard score, median absolute deviation, Mahalanobis, nearest neighbour, mean absolute spectral deviation, and absolute predictive deviation outlier detection algorithms will be discussed in the subsections to follow.

4.1.1 Standard Score

The standard score (SS), also know as the z-score or standardized variable, is a statistical measurement of the relationship between a single observation and its population mean. A positive score indicates that the data of interest is above the mean, with a negative value indicating that the data is below the mean. The standard score can be calculated for the entire population or only a sample subset. Since the mean of audio data is typically very close to zero, it makes little sense to use the population's score for outlier detection and this thesis therefore only focuses on the sample standard score. Given the data point y_t for series y at time delay t, the sample standard score is calculated as

$$d_{ss}(y_t) = \frac{y_t - \mu}{\sigma} \tag{4.2}$$

where μ is the mean and σ the standard deviation of y as given in equation (2.41). If y follows a normal distribution, that is $y \sim \mathcal{N}(\mu, \sigma^2)$, then the standard score follows a standard normal distribution such that $\frac{y_t - \mu}{\sigma} \sim \mathcal{N}(0, 1)$. A rule of thumb for a set of n samples is to mark them as outliers if their absolute standard score is 2.5 or greater for $n \leq 80$ and 3 or greater for $n > 80$ [249]. It was however shown that the absolute maximum possible score is dependent on the sample count and can be calculated as $\frac{n-1}{\sqrt{n}}$ [224]. The standard score has a limitation in that the standard deviation is exaggerated when a few or even a single extreme outlier is present in the set, which is especially problematic for very small datasets. Moderate outliers can therefore go undetected in the presence of extreme outliers.

4.1.2 Median Absolute Deviation

The mean and standard deviation are greatly influenced by a few extreme values in the population. The median absolute deviation (MAD) detection replaces the mean with the median of the population, which reduces the risk of a single extreme value affecting the outcome of the score [162]. Like the standard score, the MAD score can also be calculated on the entire population or a subset. This thesis only discusses the sample MAD score. The median of absolute deviations is calculated by

$$mad_y = \text{median}(|y_i - \tilde{y}|)_{i=1}^{n} \tag{4.3}$$

where $\tilde{y}$ is the median of the sample subset y. A MAD score at time delay t is defined as

$$d_{mad}(y_t) = \frac{c(y_t - \tilde{y})}{mad_y} \tag{4.4}$$

where c is a constant. Iglewicz and Hoaglin suggested using 0.6745 as a value for c and labelling samples as outliers if the MAD score exceeds 3.5 [139]. This threshold was estimated by simulations where a tabulated proportion of random normal observations were labelled as outliers. It is computationally expensive to calculate the median. The

calculation can be accelerated with more efficient approaches, such as quickselect, which is based on quicksort [130], or successive binning [241].

4.1.3 Mahalanobis Distance

Mahalanobis introduced a relative measure to determine the distance from a data point to a common point [169]. The Mahalanobis distance (MHD) accounts for the covariance between variables and accommodates variances in different directions. It differs from the Euclidean distance in that it is scale-invariant and therefore does not change when the scales of length are multiplied by a common factor. A function $f(x)$ is said to be scale-invariant for a scale factor λ if

$$f(\lambda x) = \lambda^{\Delta} f(x) \tag{4.5}$$

for some exponent Δ. The Mahalanobis distance is equal to the Euclidean distance under a standard normal distribution [64]. Given a vector $\mathbf{y}$ of n multivariate independent random data points and $\boldsymbol{\mu}$ as a vector of size n containing the means of the independent variables, the Mahalanobis distance is defined as

$$d_{mhd}(\mathbf{y}) = \sqrt{(\mathbf{y} - \boldsymbol{\mu})^{\mathrm{T}} \mathbf{C}^{-1} (\mathbf{y} - \boldsymbol{\mu})} \tag{4.6}$$

where $\mathbf{C}^{-1}$ represents the inverse of the covariance matrix in equation (2.34). Since $\mathbf{C}$ contains the covariances with itself on the diagonal, $\mathbf{C}$ can also be expressed as the variance-covariance matrix in equation (2.37). The Mahalanobis distance reduces to the Euclidean distance if the covariance matrix is equal to the identify matrix [26]. If the covariance matrix is diagonal, the distance measurement is called a normalized Euclidean distance [171], defined as

$$d_{mhd}(\mathbf{y}) = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \mu_i)^2}{\sigma_i^2}} \tag{4.7}$$

where σ_i and μ_i are the standard deviation and mean at points y_i respectively.

4.1.4 Nearest Neighbour Deviation

Similar to other nearest neighbour (NN) based algorithms, outliers can be detected by calculating the deviation of a subset of k samples from a larger data set, which is commonly referred to as k nearest neighbour (kNN) outlier detection. Calculating the deviation for continuous attributes is often done using the Euclidean distance between vectors of attributes [21, 267, 272], but other means for determining the deviation exists, such as the Mahalanobis, Kullback-Leibler and Hamming distances [254]. If the data is multivariate, the distance is calculated for each individual attribute and then combined to create a distance for all multivariate attributes [238].

The nearest neighbour deviation (NND) algorithms are grouped into two main categories. The first approach determines a score for a kNN global anomaly and is calculated using the distance to the k^{th} neighbour [208]. Alternatively, the mean distance of k nearest neighbours to the point of interest can be used as the score [9]. Given the point of interest y_t at time delay t, the global kNN score for a sequential dataset is calculated by using $\frac{k}{2}$ samples on both sides of y_t as follows:

$$d_{nnd}(y_t) = \frac{1}{k}\left(\sum_{i=1}^{\frac{k}{2}} |y_t - y_i| + \sum_{j=\frac{k}{2}+2}^{k+1} |y_t - y_j|\right) \tag{4.8}$$

To ensure that both sides of y_t contribute equally, the window size k should be an even number. Using the mean distance instead of the distance to the k^{th} neighbour is more robust with regards to statistical fluctuation and often the preferred method [6].

The second category of NN anomaly detectors make use of the local outlier factor (LOF). LOF finds outliers by calculating the local deviation of a point with respect to its k nearest neighbours [32]. The distance between the point of interest and its neighbours is used to estimate the local density. The local density of the given point is then compared to the local density of its neighbours. Points in regions with a local reachability density of approximately one indicate a similar density with respect to its neighbours and are therefore not considered outliers. Denser regions with values below one are also considered inliers. Regions with substantially lower densities, that are values greater than one, are considered outliers. LOF shares the concept of *reachability distance* and *core distance* with clustering algorithms such as the ordering points to identify the clustering structure

(OPTICS) [74] and density-based spatial clustering of applications with noise (DBSCAN) [31]. A benefit of LOF is that, due to calculating the local instead of the global deviation, NN distances will be more prominent for local outliers that would otherwise not be detected through global approaches [6, 32]. In addition, LOF can easily be generalized to work in various problem areas, such as detecting outliers in video streaming, spatial data and authorship networks [220].

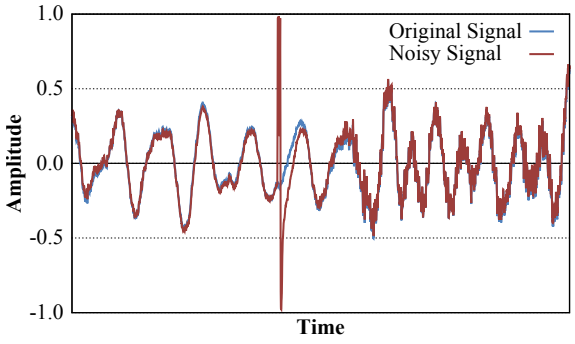
Parametrization poses a problem with local distances in LOF: when choosing the threshold to determine which LOF value will be marked as outlier, the threshold might work well for one dataset, but poorly for another. This drawback limits LOF when the same parameters are applied to a number of datasets which are coherently different in nature. A number of extensions to LOF have been proposed in order to reduce the previously mentioned and other drawbacks, but still maintaining LOF's advantages. The feature bagging for outlier detection (FBOD) runs LOF on multiple instances, combining the NN distances in higher dimensions to increase the detection accuracy [158]. The local outlier probability (LoOP) calculates the final distance of neighbours as a probability, allowing data instances to be compared with other instances in the same or other sets [153]. LoOP also uses inexpensive local statistics to reduce the sensitivity when choosing parameter k. The interpreting and unifying outlier scores (IUOS), an improvement of LoOP, proposes the normalization of the LOF score to the range $[0, 1]$ by using statistical scaling [154]. The connectivity based outlier factor (COF) was introduced to handle anomalies from spherical density patterns, such as straight lines, with the idea that outliers do not always have a lower density [239]. The influenced outlierness (INFLO) method increases the detection accuracy by taking clusters with varying densities that are near each other into account [145]. The local correlation integral (LOCI) eliminates the crucial parameter k by using the density of an instance which is proportional to the number of objects within a specified radius. The radius starts from the minimum and increases to incorporate all instances of the dataset [203].

Benchmarking with optimal parameters between global kNN, LOF, COF, LoOP, INFLO and LOCI showed that the kNN global score on average performs the best over a number of datasets, with LOF and its extensions only performing slightly better on individual datasets [6].

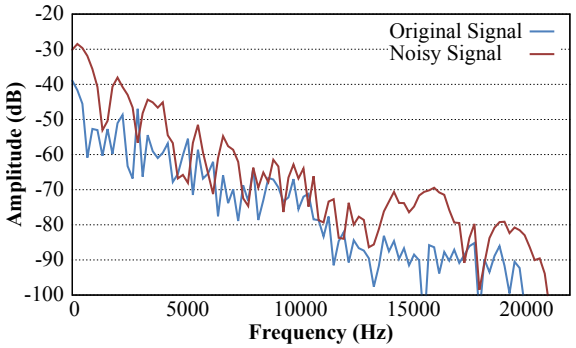
4.1.5 Mean Absolute Spectral Deviation

Noise in the time domain causes sudden disruptions for a set of samples. These disruptions can be prominent in the time domain, but can still be hard to detect if the surrounding signal has a certain degree of randomness, making it difficult to distinguish between inliers and outliers. Transforming the signal into the frequency domain moves the problem from detecting which samples are disrupted to determining which frequencies are affected by the disruptions. Most excitations occur in the very low and higher frequencies. Since speech and instruments mostly do not exceed the 10000 Hz range, noise in the higher frequencies (above 10000 Hz) are prominent and therefore easier to detect.

Figure 4.1 shows an example of an original and the corresponding noisy samples in the time and frequency domains for a signal obtained from a gramophone with a scratch. The frequency spectrum was acquired with a Hamming window computed over 4096 samples. Although lower frequencies between 100 Hz and 4000 Hz have a clear excitation from the noise, more noticeable disturbances are observed in the higher frequencies between 12000 Hz and 20000 Hz.



(a) The sound wave of the signals in the time domain.



(b) The spectrum of the signals in the frequency domain.

Figure 4.1: The difference between the original and noisy signals.

Outliers often cause a phase and amplitude shift in the Fourier series, making it suitable to apply spectral methods on the frequency domain to identify anomalies. An algorithm was proposed by Shittu and Shangodoyin that makes use of MLE to estimate the parameters of a Fourier model in order to determine the variance between the

approximation and the actual values [225]. It was found that the algorithm performed well for three datasets, but very poorly for two others. Another proposition focuses on the use of warped linear prediction on the frequency domain of audio data by using bilinear conformal mapping [72]. It was found that by properly tuning the parameters, the outliers can be emphasized in the higher frequencies with a better detection accuracy than the traditional warped linear prediction. A comparison was done on the effects of using outlier interval detection on the time domain by applying a variance algorithm, versus the use of outlier interval detection on the frequency domain by replacing the variance with a nearest neighbour algorithm [135]. A kNN algorithm was used to determine the distances between the interval's frequency vector and the k nearest frequency vectors. The three mentioned algorithms all have a slight increase in the detection accuracy compared to working in the time domain, but have either an unacceptable increase in the time complexity or only work well for special datasets.

By using a moving window, time domain outliers can be detected by analysing the surrounding frequency spectrum. The Euclidean distance between the amplitudes of neighbouring frequencies is used to calculate the mean absolute deviation among the entire frequency spectrum for a window at a certain time delay. Given a sample window y, a set of frequencies $\mathfrak{f}$ can be calculated using the DFT in equation (2.11). The resolution of the DFT depends on the window size and is also influenced by the windowing function. Using the set of frequencies $\mathfrak{f}$, the median absolute spectral deviation (MASD) for a window size of n samples is calculated using

$$d_{masd}(\mathfrak{f}) = \frac{1}{n-1} \sum_{i=un}^{vn} |\mathfrak{f}_{i-1} - \mathfrak{f}_i| \tag{4.9}$$

where u and v are additional parameters in $[0, 1]$ which control the range of frequencies considered to be affected the most by noise. If u and v are at the extremes, that is zero and one respectively, the entire spectrum is used.

Since noise mostly only disrupts a certain frequency range, the mean absolute deviation for those frequencies is greater than those of non-noisy data. The frequency deviation score for the original and noisy signals from the example in figure 4.1 is given in figure 4.2. Note that the values are not given in decibels, but represent the deviation between unprocessed Fourier values.

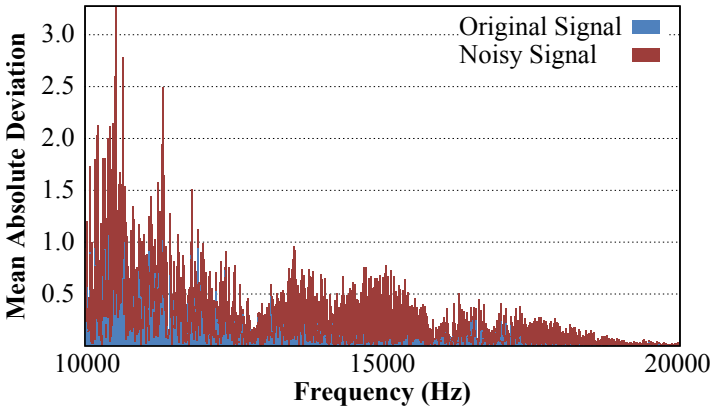


Figure 4.2: The deviation between the original and noisy signals.

The frequency deviation score, calculated using equation (4.9), for the given example's noise signal has a clear excitation in the higher frequencies with a score of 3.56, 66% higher than the original signal's score at 2.15.

4.1.6 Absolute Predictive Deviation

The absolute predictive deviation (APD) outlier detection makes use of a forecasting model to determine the next values in a time series and if these predicted values deviate from the observed values with a certain degree, they are marked as outliers. Predictive outlier detection has gained widespread academic coverage, using models such as AR [193] and ARMA models [243], multilayer perceptrons [128], nearest cluster prediction [128], and support vector regression (SVR) [168]. Given a predictive model m with a lag of n points that predicts the the next value at time delay $t+1$ in the series y, outliers are detected with the absolute deviation as follows:

$$d_{apd}(y) = |y_{t+1} - m(y_{t-n+1}, \ldots, y_t)| \tag{4.10}$$

An alternative approach makes use of the Mahalanobis distance to determine the deviation from the original signal [121]. If the absolute deviation in equation (4.10) exceeds a given threshold, the sample is flagged as an outlier. This approach is sound for univariate outliers, but can skew the model estimation for multivariate outliers, depending on the

characteristics of the input data. If y_{t+1} was flagged as an outlier, the observed value at $t+1$ should not be used for future model estimations, that is for estimations at $y_{t+2}, \ldots, y_{t+r}$, where r is the number of sequential points that contain noise. The problem is mitigated by utilising one of two alternative approaches. The first approach depicted in algorithm 3 makes use of recurrent prediction where outliers at $t+1$ are replaced with their predicted value before estimating the next model at $t+2$.

Algorithm 3 The recurrent algorithm for the absolute predictive deviation.

```
set y as the time series of observations
set p as an empty set of predictions
determine n as the optimal lag length for the model
set i = n
while i <= size of y do
    estimate model with y_{i-n}, ..., y_{i-1}
    set p_i to the model prediction at time i
    if |y_i - p_i| > threshold then
        set y_i = p_i
    end if
    increment i
end while
```

The second approach, batch prediction, is shown in algorithm 4. A model is estimated once for the surrounding samples and then used to predict all r sequential outliers. Although batch prediction is computationally less expensive than recurrent processing, since the model has to be estimated only once for an entire batch of sequential outliers, it relies on the model's ability to accurately predict up to r points. If the model is able to accurately predict enough samples into the future, batch prediction is advised, otherwise recurrent prediction should be used. In the case that a models predicts the first point in a multivariate outlier set inaccurately, all $r-1$ successive predictions will deviate even more from the intended values. In such a case the detection accuracy is very low and the model parameters must be adapted to ensure accurate predictions. If adjusting the model parameters does not improve the prediction accuracy, the model is deemed unfit

for predictive outlier detection.

Algorithm 4 The batch algorithm for the absolute predictive deviation.

```
set y as the time series of observations
set p as an empty set of predictions
determine n as the optimal lag length for the model
set i = n
while i <= size of y do
    estimate model with y_{i-n}, ..., y_{i-1}
    set p_i to the model prediction at time i
    set r = 0
    while |y_{i+r} - p_{i+r}| > threshold do
        set p_{i+r} to the model prediction at time i + r
        increment r
    end while
    increment i
end while
```

Another problem with predictive outlier detection is that the first n samples can not be tested for noise, since there is not enough data available for the model estimation. This is not a major problem when processing audio data, because even for large n, this will evaluate to only a few milliseconds than can not be processed. Alternatively, the model can be simplified to use less samples for the first prediction, reducing the number of unprocessed samples but not completely mitigating the problem. Another more formal approach that does not require a less accurate model for the first samples, is to process the first n samples in reversed order by making use of the second n samples.

In his master's thesis, Cheboli compared different outlier detection methods by benchmarking them against 19 datasets from different unrelated fields, such as NASA valve data, disk failures, power usage and enhanced vegetation data [41]. It was shown that predictive outlier detection with AR and SVR models are on average inferior to proximity based algorithms such as kNN. However, the detection accuracy of the benchmarked algorithms greatly depend on the characteristics of the data, such as volatility, and do

not necessarily provide a true representation of music data which is less volatile for high sample rates, compared to the datasets used by Cheboli.

The notation APD-m is used in this thesis, where m represents the model used for the prediction.

4.2 Noise Masking

The noise detection algorithms described in section 4.1 generate a noise map with a score for each sample in the observed signal. The scores are in $[0, k]$, where k is determined by the individual detection algorithms. The noise map indicates how much a sample is affected by noise, with a value of zero indicating no noise at all and k as pure noise. A threshold process generates a noise mask where each sample is either flagged as an inlier or an outlier. Once the noise mask is generated, a noise removal algorithm reconstructs those samples that were previously flagged as outliers by the masking threshold. Three common approaches to apply a threshold during the noise masking process are discussed next.

The first approach uses a threshold in the standard manner, where samples above a specific value are flagged as outliers. Given a threshold θ and a noise map η, the noise mask is generated as follows:

$$\breve{\eta}_i = \begin{cases} 1 & \text{for } \eta_i \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

where integer i ranges over all samples in the dataset. The other two approaches, mean and maximum thresholding, are often used in conjunction with nearest neighbour algorithms [122], but can be adapted to work with other outlier detection algorithms as well. The mean threshold is applied as

$$\breve{\eta}_i = \begin{cases} 1 & \text{for } \eta_i \geq \frac{\mu}{1-\theta} \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

where μ is the mean of η. The mean threshold approach is beneficial if a mutual value for θ must be chosen amongst a set of different noise detection algorithms. The third

approach uses the maximum value of the noise map, that is

$$\breve{\eta}_i = \begin{cases} 1 & \text{for } \eta_i \geq \hat{\eta}\theta \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

where $\hat{\eta}$ is the maximum value in η. The maximum threshold approach is beneficial if a global value is chosen for θ and the noise characteristics of different signals vary greatly. The detection accuracy may, however, be very low for this approach if a single sample was significantly affected by noise with the rest of the samples only being influenced moderately. The first approach that uses a standard threshold, has a performance advantage over the other two approaches, since neither the mean $\bar{\eta}$ nor the maximum $\hat{\eta}$ has to be computed.

4.3 Summary

This chapter discussed a number of algorithms used to detect outliers in a dataset, including a standard score, median absolute deviation, Mahalanobis, nearest neighbour and mean absolute spectral deviation. A predictive outlier method was proposed using the models from chapter 3. The standard, mean and maximum thresholding techniques are used to transform a real noise map into a binary mask.

Chapter 5

Noise Reconstruction

Interpolation is the process of reconstructing a function, curve or other geometric object by using a finite number of known values. The word *interpolation* is derived from the Latin word *interpolare* which means "to refurbish" or "to add something new". From as early as 300 BC, interpolation was used by Babylonian, Chinese and Arabian astronomers to fill gaps in calendars, often with formulas equivalent to later work done by Gregory and Newton [179]. The first reference to interpolation in the context of mathematics was in 1655 by Wallis in his book on infinitesimal arithmetic [252]. In the past century, interpolation theory has been adopted in different fields, ranging from pure mathematics in numerical integration and differentiation to more practical uses such as image resampling, audio filtering or network transmissions.

Vaseghi and Frayling-Cork proposed an adapted spectral subtraction algorithm to eliminate white noise from gramophone recordings [247]. In addition, a restoration method was discussed in the case that several copies of the same record are available. Using multiple copy restoration caused major problems, such as the time misalignment of two or more signals and signal delays as a result of fluctuations in the speed of the playback devices. In addition, multiple records are rarely available, making the solution infeasible in most practical circumstances.

Godsill and Rayner proposed a simple reconstruction algorithm using a truncated sinc function [113]. Although the authors concluded that the audio restoration was of high quality, neither an empirical analysis nor a qualitative or quantitative evaluation of

the algorithm was performed.

Godsill and Rayner also compared a frequency-based interpolation algorithm to the reconstruction of an AR model [112]. It was found that reconstructing a signal with its estimated FFT was more accurate than the AR model. The research provided only two isolated examples without any qualitative or quantitative measurement of the algorithm's performance.

Niedźwiecki and Ciołek proposed an restoration solution for stereo audio utilizing a vector autoregressive (VAR) model where the model parameters are tracked online using the stability-preserving Whittle-Wiggins-Robinson algorithm with exponential data weighting [195]. The experiments were conducted using only five artificially corrupted audio recordings. It was found that the VAR model outperformed the AR model by a small margin for all five test cases. It was also shown that if the stereo channels are analysed and processed jointly, the restoration improves. The research was later extended by Niedźwiecki et. al. where the online tracking of model coefficients was conducted using a weighted least squares algorithm [197].

The purpose of this chapter is to discusses and evaluate some fundamental interpolation algorithms used to reconstruct segments of noisy samples that were previously flagged by the outlier detection algorithms in chapter 4. The algorithms are categorised into duplication, trigonometric and model interpolation. Duplication techniques follow a copy-and-paste principle where gaps in the sound wave are replaced by sample segments retrieved from elsewhere in the signal. Duplication interpolation includes adjacent windows, mirroring windows, nearest neighbour and similarity interpolation. Lanczos and cosine interpolation are part of the trigonometric methods which utilize a trigonometric function to approximate the missing samples. Model interpolation employs any of the models presented in chapter 3 to estimate the temporal characteristics of the signal and use the coefficients to reconstruct sample gaps.

5.1 Duplication Approaches

A number of early approaches for reconstructing music and other sources of audio rely on the principle of re-occurrence. Assuming that the pre- or succeeding sequence of samples

share some characteristics with samples to be interpolated, this category of algorithms reconstruct missing samples by duplicating them from the surrounding data. Duplication algorithms typically make use of equivalent sources to reconstruct the audio, where at least one of the sources is not subjected to noise at each given time delay [5, 233]. Since in practice multiple copies are mostly not available, a more generic technique is required by copying samples from different parts of the same source. Niedźwiecki and Cisowski proposed a smart copying algorithm that copies a similar fragment from the preceding or succeeding samples using an AR model with mixed excitation and a Kalman filter [199]. The reconstruction accuracy was not determined mathematically, but by a small group of people that had to distinguish the reconstructed signal from the original audio. The smart copying algorithm was also applied to gramophone records in Cisowski's PhD thesis which provided similar performance results compared to the clean audio processing [44]. This section discusses four primitive interpolation techniques using sample duplication, namely adjacent window, mirroring window, nearest neighbour and similarity interpolation.

5.1.1 Adjacent Window Interpolation

During adjacent window interpolation (AWI), a gap of size n is interpolated at time delay t by simply copying the preceding n samples from the signal y, that is,

$$y_{t+i} = y_{t-n+i} \tag{5.1}$$

for integer $i \in \{0, 1, \ldots, n-1\}$. This approach relies on the idea that if a certain combination of samples exists, there is a likelihood that they might be repeated. The interpolation accuracy is improved by using bidirectional processing and taking the average between the forward and backward interpolation process, in other words,

$$y_{t+i} = \frac{y_{t-n+i} + y_{t+n+i}}{2} \tag{5.2}$$

5.1.2 Mirroring Window Interpolation

Volatile signals that are interpolated with the adjacent windows can cause a sudden jump between sample y_t and y_{t-1} and sample y_{t+n-1} and y_{t+n}, that is, where the gap of missing

samples starts and ends respectively. This sample jump is prominent for a large n, since using a forward adjacent window will only interpolate the gap with the historical data and not consider the future direction of the signal. By mirroring the samples during mirroring window interpolation (MWI), the signal is smoothed between the first and last sample of the gap as follows:

$$y_{t+i} = y_{t-1-i} \tag{5.3}$$

for integer $i \in \{0, 1, \ldots, n-1\}$. Similarly, the average between the forward and backward interpolation increases the accuracy:

$$y_{t+i} = \frac{y_{t-1-i} + y_{t-1+2n-i}}{2} \tag{5.4}$$

5.1.3 Nearest Neighbour Interpolation

The nearest neighbour interpolation (NNI) interpolates a point by choosing the value of the closest neighbouring point in the Euclidean space. NNI for a sequential dataset at time delay t is defined as

$$y_t = \sum_{i=t-k}^{t+k} h(t - i\triangle_t) y_i \tag{5.5}$$

where k is the number of samples to consider at both sides of y_t and h is the rectangular function defined as

$$h(t) = \begin{cases} 1 & \text{for } -\frac{1}{2} \leq \frac{t}{\triangle_t} < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

For a sequential dataset of n samples, NNI always chooses point y_{t-1} for $t \leq \frac{n}{2}$ and point y_{t+n} $t > \frac{n}{2}$. Since NNI was originally intended for resampling two-dimensional grid data, the reconstructed samples may have a steep jump in the middle of the gap if there is a steep gradient between the last sample before and the first sample after the gap. In order for a smoother interpolation between the samples on the left and right side of the gap, instead of simply using the nearest point in a dataset with k values, the missing samples

are interpolated with the mean of the nearest k samples. Depending on the position of the interpolated sample, a different number of samples from the left and right of the gap is considered to calculate the mean. The mean can also be replaced with the median, which is statistically more accurate for volatile signals or if the signal still contains outliers that were not flagged by the outlier detection process in chapter 4. However, as n increases, the interpolation accuracy advantage of the median over the mean will decrease, and will only increase the computational time.

5.1.4 Similarity Interpolation

Duplication-based interpolation algorithms can produce an inaccurate reconstruction if the interpolation gap shares little characteristics with the preceding and successive samples. A more accurate approach through similarity interpolation (SI) involves the search for a sequence of samples that are similar to the samples on each side of the gap. This can be done by constructing a set of vectors $\mathbf{d}_i$ by calculating the deviation between the amplitudes of neighbouring samples in a moving window as follows:

$$\mathbf{d}_i = [(y_i - y_{i+1}), (y_{i+1} - y_{i+2}), \ldots, (y_{i+n-1} - y_{i+n})] \tag{5.7}$$

where y is the series of observed samples with a moving window size of $n+1$. Given u as the number of previous and v as the number of future samples surrounding the gap, a total of $(u+v-n)$ vectors are calculated when the window moves over y. For a gap starting at time delay t, the last deviation vector just before the gap is denoted as

$$\tilde{\mathbf{d}} = \mathbf{d}_{t-n} = [(y_{t-n} - y_{t-n+1}), (y_{t-n+1} - y_{t-n+2}), \ldots, (y_{t-2} - y_{t-1})] \tag{5.8}$$

In addition, for every vector $\mathbf{d}_i$, except for $i = t-n$, the difference between the last sample in the current and the first sample in the next window is calculated as

$$r_i = y_{i+n} - y_{i+n+1} \tag{5.9}$$

The goal of similarity interpolation is to find the vector in $\mathbf{d}_i$ that shares most of its characteristics with $\tilde{\mathbf{d}}$. Note that, by using the amplitude deviation, the algorithm will find a sequence of samples that are similar in direction and gradient and not necessarily

similar in amplitude. This ensures that sample sequences with a similar waveform are also considered and not only sequences that have similar amplitudes. The similarity between a vector $\mathbf{d}$ and the desired vector $\tilde{\mathbf{d}}$ is calculated as the sum of squared differences:

$$s(\mathbf{d}) = \sum_{i=0}^{n-1} (\tilde{d}_i - d_i)^2 \tag{5.10}$$

The vector which has the lowest score is then chosen, as follows:

$$\check{\mathbf{d}} = \arg\min_i \ s(\mathbf{d}_i) \tag{5.11}$$

Once the most similar vector was found, the corresponding deviation r_i is added to the last sample before the gap, that is

$$y_t = y_{t-1} + r_i \tag{5.12}$$

The process is applied iteratively until all missing samples are interpolated. Similar to AWI and MWI, in order to increase the accuracy, SI can be applied in a forward and reversed order by taking the average r_i in both directions. Processing in both directions will only improve the interpolation accuracy slightly, but double the computational time.

5.2 Trigonometric Approaches

Trigonometric functions can be fitted between two points to create a smooth adjacency used for interpolation. This section discusses two basic trigonometric approaches, namely Lanczos and cosine interpolation.

5.2.1 Lanczos Interpolation

Lanczos interpolation (LI), named after Cornelius Lanczos, is a smoothing interpolation technique based on the sinc function [59]. The sinc function is the normalized sine function [105] defined as

$$\text{sinc}(x) = \frac{\sin(x)}{x} \tag{5.13}$$

The Lanczos interpolation is defined as

$$l(x) = \sum_{i=\lfloor x \rfloor - n+1}^{\lfloor x \rfloor + n} y_i L(x-i) \tag{5.14}$$

where $\lfloor x \rfloor$ is the floor function of x, n the number of samples to consider on both sides of x and $L(x)$ the Lanczos kernel. The Lanczos kernel is a dilated sinc function used to window another sinc function as follows:

$$L(x) = \begin{cases} \text{sinc}(x)\text{sinc}(\frac{x}{n}) & \text{for} -n < x < n \\ 0 & \text{otherwise} \end{cases} \tag{5.15}$$

Lanczos interpolation is typically used for resampling where the interpolation is applied to gaps between equidistant samples. If non-equidistant gaps are interpolated, all values of x in the Lanczos kernel are scaled to the range $[0, 1]$ and the resulting interpolate is divided by the sum of the Lanczos kernel.

5.2.2 Cosine Interpolation

A continuous trigonometric function like cosine can be used to smoothly interpolate between two points. Given a gap of n missing samples starting at time delay t, the cosine interpolation (CI) is defined as

$$c(x) = y_{t-1}(1 - h(x)) + y_{t+n}h(x) \tag{5.16}$$

where $h(x)$ is calculated with cosine as follows:

$$h(x) = \frac{1 - \cos\left(\frac{\pi(x+1)}{n+1}\right)}{2} \tag{5.17}$$

The cosine operation can be replaced with any other smoothing function $f(x)$, as long as the function adheres to

$$f(0) = 1 \quad \text{and} \quad f'(x) < 0 \quad x \in (0, 1) \tag{5.18}$$

Alternatively, if the function has the properties

$$f(0) = 0 \quad \text{and} \quad f'(x) > 0 \quad x \in (0, 1) \tag{5.19}$$

the points y_{t-1} and y_{t+n} in equation (5.16) have to be swapped around.

5.3 Model Interpolation

Model interpolation is a widely used technique for reconstructing gaps of missing samples. A model is estimated to fit a set of given data points as accurately as possible. The estimated model is then used to determine the values of the missing points. Any of the models discussed in chapter 3 can be used for this type of interpolation. The reconstruction accuracy of the model depends on the characteristics of the observed points and the chosen parameters for the model.

5.4 Summary

This chapter discussed interpolation techniques using duplication approaches such as adjacent and mirroring windows, nearest neighbour and similarity interpolation. Two trigonometric approaches, namely Lanczos and cosine interpolation were considered. In addition the models in chapter 3 can be estimated in order to approximate the missing samples.

Chapter 6

Research Methodology

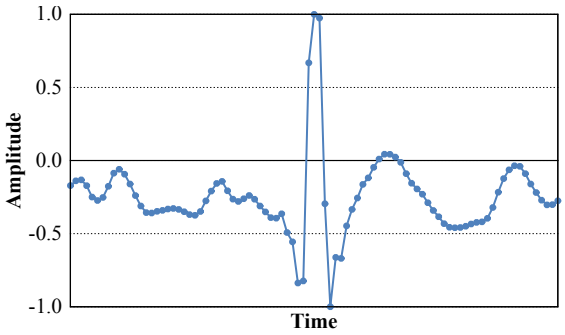
This chapter discusses the methodology, test data collection and procedures followed to prepare, benchmark and analyse the ability of the proposed system to detect and reconstruct noise from gramophone recordings. The characteristics of gramophone noise is discussed, followed by a description of the data used during testing. A short report on the audio format, benchmarking system, performance measurement and parameter optimization process is also given.

6.1 Noise

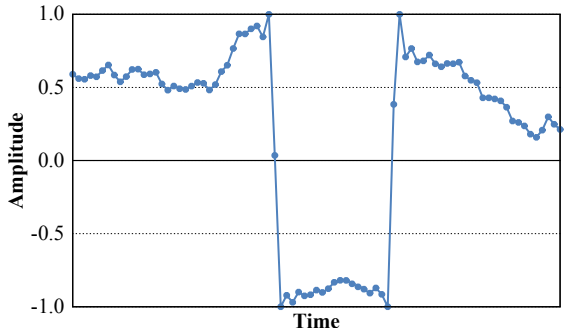
A single stereo song encoded at 44.1 kHz with a duration of four minutes contains approximately 21 million samples. In order to validate the ability of the outlier detection algorithms, the noise in the recorded songs must be manually identified and correlated with the output of the algorithms. Manually identifying the noisy samples, even for a single song of 21 million samples, is a tedious and time-consuming process and it is therefore impractical to create a large dataset where the noise was manually flagged. A typical approach in audio processing is to generate artificial disruptions in clean audio data with Gaussian white noise [73, 134, 202]. Besides Gaussian white noise, Niedźwiecki [198] also suggested using positive pulses with a constant magnitude and a mixture of white noise and impulses.

Figure 6.1 shows four examples of noise disruptions typically observed in gramophone

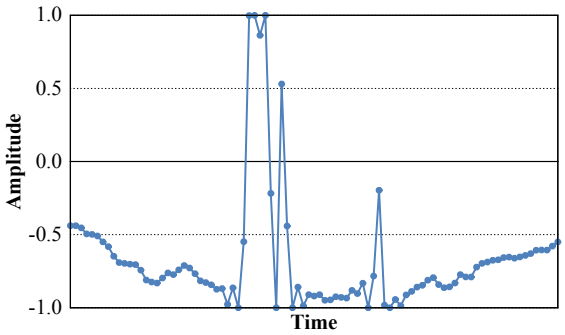
recordings. These examples were specifically chosen, since they represent the overall characteristics of the majority of noise caused by scratches. As can be seen from figure 6.1, it is insufficient to simply use artificially generated Gaussian white noise to represent the disruptions caused by scratches on gramophone records, since is not a true representation of the observed noise which typically follows one of several patterns.



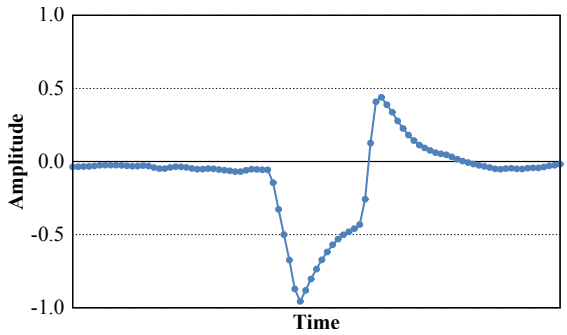
(a) A distortion with a positive pulse affecting 7 samples.



(b) A distortion with a negative pulse affecting 23 samples.



(c) A distortion with multiple positive pulses affecting 10 samples.



(d) A distortion with a negative followed by a positive pulse affecting 21 samples.

Figure 6.1: The sound wave of typical noise pulses from a distorted gramophone record.

Four different types of artificial noise are used for the empirical tests. Unlike Niedźwiecki [198], the four different types of noise are applied with both positive and negative pulses, since scratches affect the amplitudes in both directions. Positive pulses are shown in figure 6.1(a) and 6.1(c) with negative pulses given in figure 6.1(b) and 6.1(d). The patterns of artificial noise used in the experiments are categorised as follows:

- **Single bursts:** A single burst with a sudden positive increase or negative decrease in the amplitude as shown in 6.1(a) and 6.1(b).
- **Dual bursts:** A single burst with a sudden positive increase or negative decrease, immediately followed by another burst in the opposite direction.
- **Dual gradual bursts:** Similar to the dual bursts, this type of noise has two peaks with one positive and one negative pulse. Instead of the two peaks following each other immediately, a more gradual increase or decrease between the peaks are followed as shown in figure 6.1(d).
- **Oscillating bursts:** This type of noise follows a more irregular rhythm with sudden consecutive positive and negative bursts. As shown in figure 6.1(c), short positive and negative pulses follow each other, creating a zigzag-like pattern.

These four noise patterns are not directly applied to the audio signals, but are first subjected to a Gaussian white noise process. Adding a certain degree of randomness to the noise patterns ensures that the outlier detection algorithms and the corresponding parameters are generic enough to detect different variations of the noise and not only a small group of constant patterns.

Scratches typically affect no more than 30 sequential samples. For the empirical tests, noise with a duration of up to 50 samples are generated to also accommodate slightly larger scratches. Each of the four artificial noise patterns are applied to the 50 different noise lengths and subjected to white noise, generating a large set of different noise variations. The artificially generated noise of each pattern is uniformly distributed over noise durations between one and 50 samples. Hence, a generated noise sequence of type p_a with a duration of n_a has the same probability of occurrence than any other noise pattern p_b with a duration of n_b for $n_a, n_b \in \{1, 2, \ldots, 50\}$. Although scratches can exceed 50 samples, they are very rare in practice and were omitted in order not to skew the detection performance with scarce anomalies.

Besides using artificially generated noise, another dataset with real noise from gramophone recordings is also used. Since real noise requires the tedious task of manually flagging the outliers beforehand, the dataset is considerably smaller than the artificially generated noise dataset.

A standard four minute stereo song encoded at 44.1 kHz was artificially disrupted with a total of 529200 noisy samples which is equivalent to approximately 2.5% of all samples in the signal. Real gramophone recordings had fewer disruptions with less than 1.5% of samples being distorted.

6.2 Test Data

The test data is divided amongst eight different music genres, in order to accommodate various characteristics of different types of music. The genres are categorised as follows:

- **Classical:** Classical music is typically composed of the string, brass, percussion, and woodwind families of instruments and also includes opera. The genre is subdivided into medieval, renaissance, baroque, classical, modern, and contemporary, according to the era of composition. Classical music typically has a very narrow dynamic range with calm or even silent fragments. An example of a classical sound wave is given in figure 6.2(a).

- **Country:** Country music is characterised by simple harmonies accompanied by mostly string instruments such as banjos, acoustic guitars, fiddles, and other instruments such as harmonicas and accordions. Country can be divided into folk, swing, blues, boogie, and gospel. Country music has a lower dynamic range than most other genres. An example of a country sound wave is given in figure 6.2(b).

- **Electronic:** Electronic music encapsulates music generated with electronic instruments such as electric guitars, keyboards, theremins, and sound synthesizers. A large part of this genre is directly created through computer software. Subgenres include electro, techno, dance, trance, and house. The genre is characterised by a very wide dynamic range with a large number of samples reaching the extremes. An example of an electronic sound wave is given in figure 6.2(c).

- **Jazz:** Jazz is typically composed using drums, guitars, pianos, and brass instruments, most notably the trumpet and saxophone. Although not one of the main genres, jazz was included due to the higher amplitudes from brass instruments

in a mostly narrow dynamic range. Subgenres include soul, bop, latin jazz, and often overlaps with blues and swing. An example of a jazz sound wave is given in figure 6.2(d).

- **Metal:** Metal is characterised by loud and distorted sound generated by drums, distorted guitars, dense bass and vigorous vocals. Although a subgenre of rock, metal was specifically included due to the very wide dynamic range and sound distortions that are similar to the noise caused by gramophone scratches. The genre can be divided into heavy metal, death metal, nu metal, punk, and grunge. An example of a metal sound wave is given in figure 6.2(e).

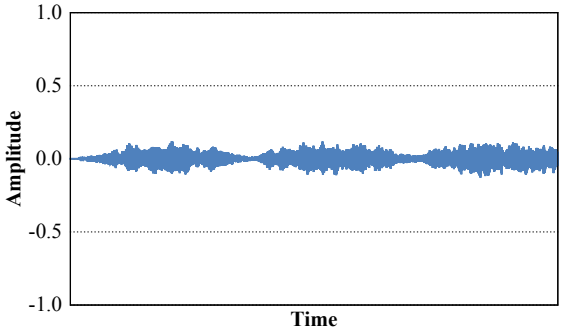
- **Pop:** Originally derived from rock, dance and country music, pop or popular music employs repeated choruses, melodic tunes, and hooks composed with a wide range of instruments. Pop music typically has a medium dynamic range with higher amplitudes caused by electric guitars, electronic beats or high-pitched vocals often repeated in refrains. An example of a pop sound wave is given in figure 6.2(f).

- **Reggae:** A genre that originates from Jamaica, reggae mainly uses guitars, hand drums, and traditional Jamaican and African instruments. This genre was included, since reggae is characterised by emphasizing the last note in a beat. This stands in direct contrast to almost all western music which emphasizes the first note in a beat. Other genres that fall in this category includes ska and rocksteady. An example of a reggae sound wave is given in figure 6.2(g).

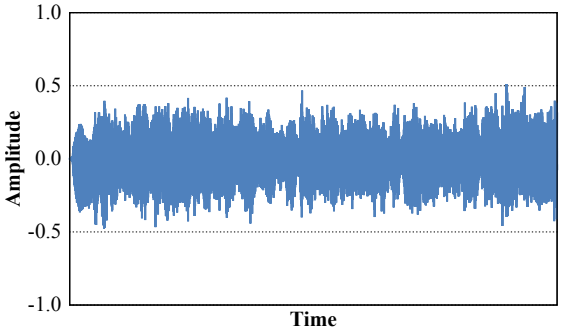
- **Rock:** Also known as rock and roll, this genre is typically composed with electric and acoustic guitars, bass and drums. Rock has a wide dynamic range, but in general still narrower than metal and electronic music ranges. Subgenres include classic, hard, punk, and alternative rock. An example of a rock sound wave is given in figure 6.2(h).

Although other songs in the same genre may deviate from the sound waves in figure 6.2, it illustrates the general pattern most songs in the given genres follow.

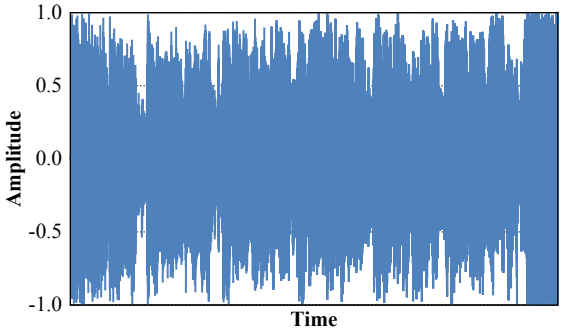
The test data consists of 100 songs from each genre. For a total of 800 songs, the dataset has a size of 23.2 GB with a duration of approximately 14 hours and more than



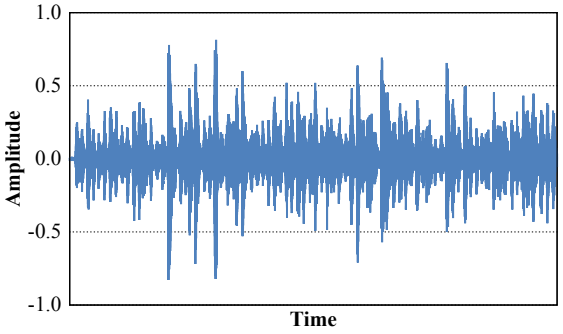
(a) The sound wave of a classical song.



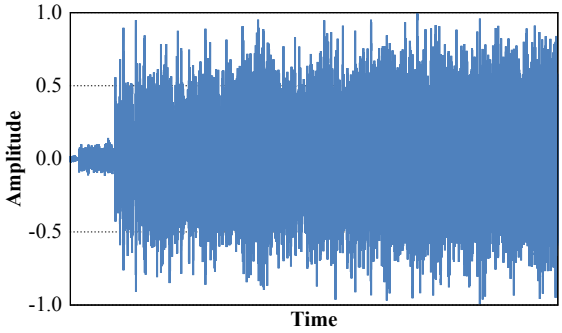
(b) The sound wave of a country song.



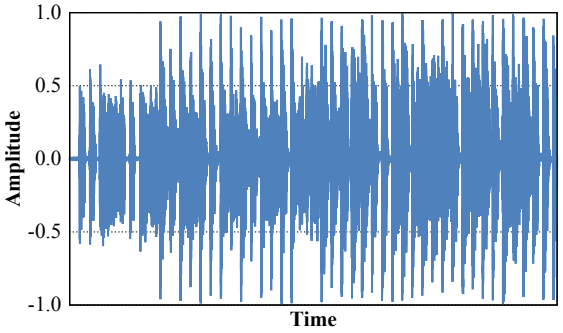
(c) The sound wave of an electronic song.



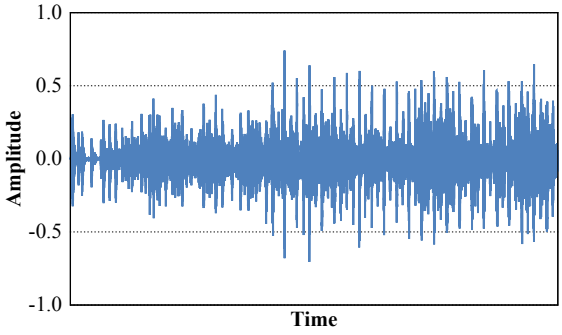
(d) The sound wave of a jazz song.



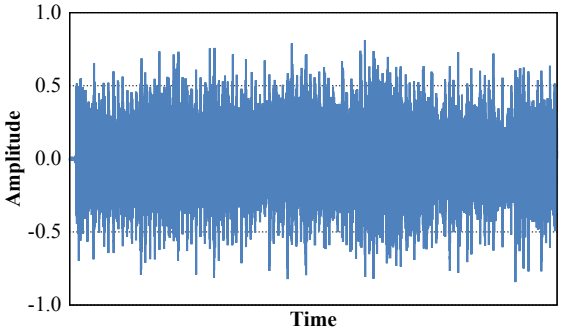
(e) The sound wave of a metal song.



(f) The sound wave of a pop song.



(g) The sound wave of a reggae song.



(h) The sound wave of a rock song.

Figure 6.2: The sound waves of songs from eight different genres.

2 billion (2.2×10^9) samples. The entire list of songs used during testing is given in appendix A. The test data is encoded as follows:

- **Codec:** FLAC (Free Lossless Audio Codec)
- **Sample rate:** 44100 Hz
- **Sample type:** Integer
- **Sample size:** 16 bits
- **Channels:** 2 (Stereo)

In addition to the 800-track set, a number of gramophone records were also tested. In order to evaluate the algorithms' performance, the reconstructed signal must be compared to a target signal. For this reason, gramophone records in a sealed mint condition were obtained, meaning the records were new and never played before. All records were recorded in mint condition to generate a set of target signals. The records were then physical damaged and rerecorded to create a set of distorted signals. The distorted signals were reconstructed and correlated against the target signals. The interpolation process will produce similar results to the non-gramophone test set, since the reconstruction process is identical in both cases. However, since artificial noise was used, the noise detection process for gramophone records may deviate from the results obtained from the non-gramophone test set. Since sealed gramophone records are difficult to obtain, are often very expensive, and require the noise to be manually flagged, the gramophone test set only consists of eight records. The gramophone records have a total of 83 songs with a duration of more than six hours and a total size of 9.1 GB. The list of gramophone records used for benchmarking can be found in appendix A under section A.9.

6.3 Performance Measurement

All benchmarking and tests were performed on the following machine:

- **Processor:** Intel Core i7 2600 (3.4 GHz with 8 threads)

- **Memory:** 16 GB (DDR3 at 1333 MHz)
- **Operating System:** Ubuntu 14.04 (64 bit)

Although a multi-threaded processor was used, all algorithms were executed with a single thread to ensure consistency in the computational time between the different algorithms. This section discusses the performance measurement for the noise detection and interpolation algorithms, the assessment of the computational time and the tradeoff between the accuracy and time.

6.3.1 Outlier Detection Performance

A common approach to measure the ability of an algorithm to detect outliers is to calculate the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). The TP and TN are the number of correctly identified outliers and inliers respectively, whereas the FP and FN are the number of incorrectly flagged inliers and outliers respectively. To make sense of these values, the sensitivity (SEN) which is the ability to identify outliers, and the specificity (SPE) which is the ability to identify inliers, can be calculated [170]. The SEN and SPE are defined as

$$\begin{aligned} \text{SEN} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ \text{SPE} &= \frac{\text{TN}}{\text{TN} + \text{FP}} \end{aligned} \tag{6.1}$$

Although SEN and SPE on their own are good indications of how well in- and outliers were identified, a single measurement is needed to determine the detection performance.

The binary classification performance can be measured as the proportion of true identifications to the population size, which is known as the statistical accuracy and is calculated using

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6.2}$$

The statistical accuracy has an inflated performance when calculated with imbalanced datasets. If the SNR of the dataset is very low, the accuracy favours an increase in

negatives (TN and FN) more than a decrease in positives (TP and FP). To reduce this problem, a balanced accuracy is often used on imbalanced datasets instead, which is defined as

$$\text{BACC} = \frac{\text{TP}}{2(\text{TP}+\text{FN})} + \frac{\text{TN}}{2(\text{TN}+\text{FP})} \tag{6.3}$$

The balanced accuracy is equivalent to the average of the sensitivity and specificity. Both accuracy and balanced accuracy are still subject to the accuracy paradox which states that a model may have a greater predictive power than another one even though the accuracy is lower [245]. If the minority class is more important, the paradox is avoided by using the G or F1 score measurement, defined as

$$\begin{aligned} \text{G} &= \frac{\text{TP}}{\sqrt{(\text{TP}+\text{FN})(\text{TP}+\text{FP})}} \\ \text{F1} &= \frac{2\text{TP}}{2\text{TP}+\text{FP}+\text{FN}} \end{aligned} \tag{6.4}$$

The G-measurement is the geometric mean of the precision and recall. F1 is the harmonic mean and produces results very similar to G. Both these measurements correctly drop to zero as the specificity reaches one and the sensitivity reaches zero. However, both measurements result in a high sensitivity even if the specificity is very low. This problem can be corrected by penalizing false identifications more. The Matthews correlation coefficient (MCC), is a balanced measure even if the classes are unbalanced and in essence is the correlation coefficient between the observed and predicted binary classifications [174]. The MCC is defined as

$$\text{MCC} = \frac{\text{TP}\times\text{TN} - \text{FP}\times\text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}} \tag{6.5}$$

Unlike the other measurements, the MCC can have negative values. Negative one indicates a total disagreement with the observation, zero a random prediction and positive one a perfect prediction. The maximum MCC is close to the intersection of the sensitivity and specificity and drops to zero if either one of them heads to zero.

6.3.2 Reconstruction Performance

In order to determine the ability of algorithms to reconstruct missing samples, a performance measurement is needed. Such a performance measurement relies on the availability of the target samples to be interpolated or predicted, which does not pose a problem in a controlled experimental environment. However, if the target signal is unknown, for instance if only a single damaged gramophone record is available, the only measurement is through human cognition by listening and evaluating the reconstructed signal. If the target sample values are known, the mean squared error (MSE) can be used to calculate the difference between the observations and the model's interpolated or predicted samples. The MSE is defined as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2 \tag{6.6}$$

for a set of n samples, where y_i is the observed value and $\tilde{y}_i$ is the estimated value. The root mean squared error (RMSE) allows the error to be compared with the absolute sample deviation and is calculated using

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{6.7}$$

The RMSE is scale dependent and can therefore only be used to determine the accuracy between different models for a particular variable [136]. The normalized root mean squared error (NRMSE) takes the range of the variables into account. For audio samples this range lies in $[-1, 1]$. The NRMSE is defined as

$$\text{NRMSE} = \frac{\text{RMSE}}{\hat{y} - \check{y}} \tag{6.8}$$

where $\hat{y}$ is the maximum and $\check{y}$ the minimum of the observed values y. Most musical signals come close to the edges of the interval $[-1, 1]$ at some stage and the NRMSE will therefore be close to the RMSE divided by two for the entire signal. The NRMSE is used as the interpolation performance measurement in this thesis, since unlike RMSE, NRMSE accurately compares the performance of small sample windows with different sample ranges.

An assumption was made during the reconstruction phase for the dataset corrupted by artificial noise that all noise was correctly flagged during the detection phase. Hence, the results for the reconstruction of samples are not influenced by possible mistakes made by the outlier detection algorithm. The real gramophone recordings were refurbished using the previously generated noise mask. Therefore, the real gramophone audio reconstruction is affected by mistakes made by the outlier detector and will therefore have a higher error compared to the artificially disrupted dataset.

6.3.3 Computational Speed

The computational time in this thesis is calculated as the number of seconds required to process one second of audio data, denoted as s\s. Therefore, the computational time should be interpreted as a speed, measuring the time required by an algorithm to process a certain number of samples, where both the required time and the number of processed samples are measured in seconds. Note that all tests were conducted with audio channels encoded at 44.1 kHz. Hence, one second of stereo data contains $(2 \times 44100) = 88200$ samples. Other sample rates will result in different execution speeds, but will follow the same trend.

A score of 1 s\s or lower indicates that the algorithm can be executed in real-time. Scores greater than 1 s\s indicate that it takes longer to process the data than the actual duration of the audio signal. Algorithms with an computational speed greater than 1 s\s can still be applied in real-time on a multi-core processor if the algorithm's calculation can be executed over multiple threads.

6.3.4 Tradeoff

Based on the concept of the scoring metric in [227], in order to evaluate the tradeoff between the algorithms' performance and the required execution speed, the speed-accuracy tradeoff (SAT) is used as follows:

$$\mathrm{SAT} = \left(\frac{\kappa}{\hat{\kappa} - \check{\kappa}} + \frac{\tau}{\hat{\tau} - \check{\tau}} \right)^{-1} \tag{6.9}$$

where τ is the computational speed measured in s\s. $\hat{\tau}$ and $\check{\tau}$ are the execution speeds

of the fastest and slowest algorithm respectively. Since a lower NRMSE indicates a better performance, κ for the interpolation SAT is equal to the NRMSE. However, since a higher MCC indicates a better performance, κ is equal to $(1 - \text{MCC})$ for the outlier detection SAT. $\hat{\kappa}$ and $\check{\kappa}$ are the accuracies of the best and worst performing algorithms respectively. $\hat{\kappa}$ and $\check{\kappa}$ have bounds which lie in $[0, 1]$ for the NRMSE and $[-1, 1]$ for the MCC. However, since equation (6.9) does not have an upper limit for $\hat{\tau}$, a few or even a single algorithm with a computational speed significantly higher than the mean execution speed, skews the SAT scores with the deviation $(\hat{\tau} - \check{\tau})$. Algorithms that take longer to execute are therefore not penalized appropriately. The problem is mitigated by specifying a fixed upper speed limit for $\hat{\tau}$. The SAT in this thesis utilizes the real-time limit of 1 s\s. A higher SAT score indicates a more efficient tradeoff between the accuracy and the execution speed.

6.4 Parameter Optimization

Factorial design, as introduced by the statistician Fisher [87], was utilized for all parameter optimization. Factorial design fully searches all possible discrete values in the parameter space, and will therefore also evaluate values in areas of the search space that are evidently moving away from the intended objective. More specifically, fractional factorial design was used, where a carefully selected subset of the parameter space was searched. This subset was chosen by iteratively increasing the parameters, such as the window size and polynomial orders, until the results stabilized or a clear trend was observed in the opposite direction of the objective. The full list of optimal parameters is given in appendix B. A total of 80 songs, ten in each genre, were used for the parameter optimization.

6.5 Summary

The methodology and procedures followed during the empirical analysis was discussed in this chapter. A description of artificially generated and real gramophone noise was given, followed by an overview of the test dataset. The performance measurement of the outlier detection and reconstruction algorithms, the execution speed and tradoff was presented,

followed by a discussion about the optimization process.

Chapter 7

Empirical Analysis

This chapter provides the empirical analysis of the reconstruction system. The test dataset is anayalsed according to its mean, standard deviation and the Pearson correlation coefficient. The polynomials and models from chapter 3 are then benchmarked to determine their ability to interpolate and predict music signals. The capabilities of the proposed system are examined by testing and evaluating the various outlier detection algorithms from chapter 4 and the reconstruction algorithms from chapter 5. The noise detection and reconstruction algorithms are compared using artificially generated noise, followed by an analyses of the performance using real gramophone noise. A more detailed report on the results of the outlier detection and reconstruction algorithms is given in appendix C and D respectively.

7.1 Data Analysis

Table 7.1 shows the average sample mean, standard deviation and PCC over the entire test dataset of 800 songs, calculated using a sample size of 32. All genres had a sample mean close to zero. The sample standard deviation shows that electronic and metal music had the highest volatility, whereas classical, country and jazz signals were more stable. The PCC was calculated with two consecutive windows x and y as defined in equation (2.54). Hence, the PCCs in table 7.1 show the positive and negative linear dependencies between a sample window x and the sample sequence y that immediately followed x. The

95% PCC confidence intervals are given in brackets next to the correlation coefficients. Classical, country and jazz music had a strong linear dependency. With an increase in volatility, notable in electronic and metal signals, the linearity decreased, although the linear relationship was still moderate at approximately ± 0.5 to ± 0.6.

Table 7.1: The mean, standard deviation and Pearson correlation coefficient of the dataset.

Genre	Sample μ	Sample σ	Positive Sample PCC	Negative Sample PCC
Classical	-0.0005	0.0101	0.7609 [0.426, 0.912]	-0.7281 [-0.364, -0.899]
Country	-0.0008	0.0294	0.6708 [0.262, 0.875]	-0.6829 [-0.283, -0.881]
Electro	0.0001	0.0543	0.4931 [-0.004, 0.795]	-0.5079 [-0.016, -0.802]
Jazz	-0.0002	0.0271	0.6886 [0.293, 0.883]	-0.6445 [-0.219, -0.864]
Metal	-0.0002	0.0526	0.5352 [0.054, 0.815]	-0.6109 [-0.165, -0.849]
Pop	-0.0001	0.0459	0.5271 [0.043, 0.811]	-0.5491 [-0.073, -0.821]
Reggae	-0.0003	0.0428	0.5638 [0.094, 0.828]	-0.5845 [-0.125, -0.838]
Rock	-0.0016	0.0319	0.5888 [0.131, 0.839]	-0.6252 [-0.188, -0.856]
Average	-0.0004	0.0368	0.6035 [0.154, 0.846]	-0.6166 [-0.174, -0.852]

7.2 Model Analysis

This section empirically analyses and discusses the ability of the polynomials and models in chapter 3 to accurately interpolate and predict a music signal. The interpolation NRMSE determined the reconstruction accuracy of missing or noisy samples for each model. The forecasting quality was evaluated for the predictive noise detection algorithm discussed in section 4.1.6 and was also measured using the NRMSE. The figures in this section that are given in pairs illustrate the model fitness for the observed values for a given parameter configuration on the left-hand side. The corresponding graphs on the right-hand side show the interpolation or prediction fitness for different parameters. The rectangular boxes in the top legend of the graphs represent the minimum NRMSE with the optimal parameters given in brackets. These optimal parameters are given in the measurement of the x-axis in tow-dimensional graphs, and the x -and y-axis' measurement in three-dimensional surface graphs. The interpolation and prediction results for each model in this section represent the overall performance of reconstructing or forecasting

up to 50 samples. The polynomials and models can, however, interpolate or predict any number of samples without changes to their implementation, but the results were omitted since longer noise durations rarely occur on gramophones.

7.2.1 Standard Polynomials

This section analyses the interpolation and prediction ability of standard polynomials as discussed in section 3.1. The polynomials are also applied in an osculating fashion and by using splines which were examined in section 3.5 and section 3.6 respectively.

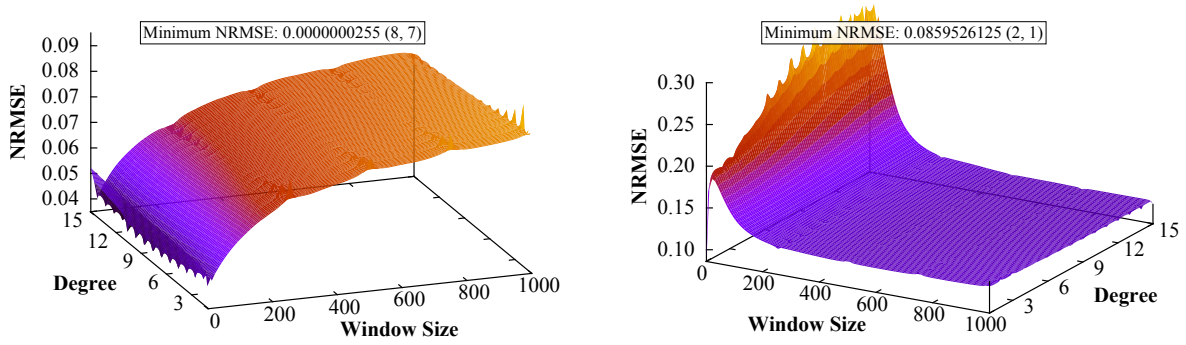
Standard Polynomial Interpolation

The standard polynomial was able to accurately fit 100 data points or less, but an increase in the window size lead to a slight surge in the NRMSE as illustrated in figure 7.1(a). The performance of the interpolation shows an opposite trend as depicted in figure 7.1(b). As the window size increased, the NRMSE of the observed and interpolated values converged close to an error of 0.1. However, for smaller window sizes, the higher deviation between the model and interpolation NRMSE indicates that the polynomial was overfitted for the given data. The degree of the polynomial had a lesser impact on the interpolation accuracy. Low degree polynomials performed better with smaller window sizes, but as the number of samples increased, higher degree polynomials achieved better results, although the improvement was statistically insignificant.

Taking the unisolvence theorem 2 into account, at least $n + 1$ points are needed to accurately approximate a polynomial of degree n. Hence, the peak in figure 7.1(b) was the result of high degree polynomials that were approximated with too few data points. This trend is also observable in figure 7.1(a), however, to a lesser extend. All the polynomials computed over small window sizes achieved the lowest model NRMSE with a polynomial order of exactly one less than the window size. The interpolation performed best with a linear polynomial running through only two points.

Standard polynomials that were applied in an osculating fashion resulted in a similar model trend for the observed values as shown in figure 7.2(a).

The best model accuracy was also achieved using a polynomial order of seven, computed over eight samples. The influence of the number of derivatives on the model accuracy is



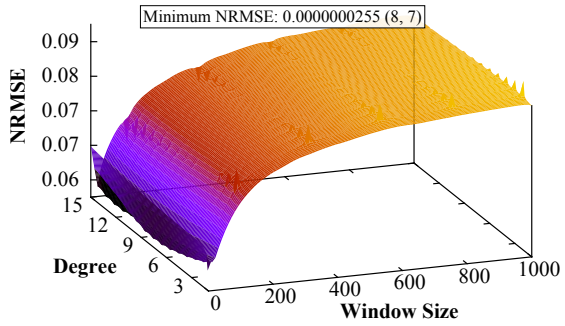
(a) The model accuracy for different window sizes and degrees.

(b) The interpolation accuracy for different window sizes and degrees.

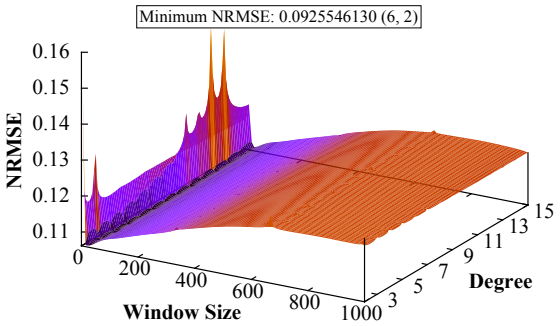
Figure 7.1: The accuracy of standard polynomials for different parameter configurations.

given in figure 7.2(c). Lower derivatives performed better, indicating that the derivatives should be omitted and that osculating polynomials are not appropriate for reconstructing music signals. The interpolation accuracy of osculating polynomials is given in figure 7.2(b) with the corresponding derivatives in figure 7.2(d). The NRMSE of the derivatives was calculated as the average error over all window sizes. Similar to figure 7.1(b), small window sizes resulted in a high NRMSE due to a lack of data points to accurately approximate the polynomial. Quadratic osculating polynomials interpolated most accurately with a window size of six samples. Although osculating polynomials performed relatively well, omitting the derivatives increased the interpolation accuracy. Note that the worst interpolation NRMSE of osculating polynomials was still twice as low as the worst case of standard polynomials. Due to the inclusion of derivatives, the chances of overfitting osculating polynomials, especially for smaller window sizes, were reduced.

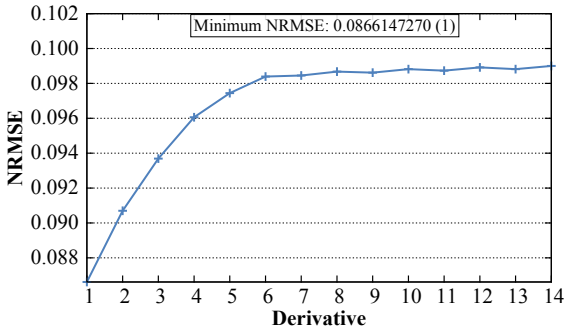
The accuracy of splines that were constructed with standard polynomials had a good fit for larger window sizes as illustrated in figure 7.3(a). The best model and interpolation accuracy was achieved through linear splines. Figure 7.3(b) shows that smaller window sizes were preferred for spline interpolation. Although the reconstruction error of splines was slightly lower than that of standard polynomials, spline matrices grew larger through the incorporation of derivatives and the free endpoint conditions which increased the computational speed without adding a statistical significant improvement to the interpolation process.



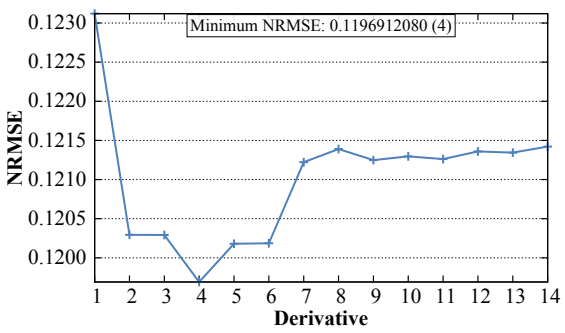
(a) The model accuracy for different window sizes and degrees.



(b) The interpolation accuracy for different window sizes and degrees.



(c) The model accuracy for different derivatives.



(d) The interpolation accuracy for different derivatives.

Figure 7.2: The accuracy of osculating standard polynomials for different parameter configurations.

Standard Polynomial Prediction

Figure 7.4(a) shows that low degree polynomials achieved a better prediction than high degree polynomials. Since prediction clearly favoured linear polynomials, figure 7.4(b) was added to illustrate the trend more clearly. A linear polynomial approximated with 48 samples was able to most accurately forecast music signals.

Osculating polynomials had a lower interpolation accuracy than standard polynomials, but when applied for forecasting purposes, the prediction error of osculating polynomials was notably lower. The prediction NRMSE reached a minimum using quadratic osculating polynomials computed over 48 samples as depicted in figure 7.5(a). The corresponding

affect of the derivatives on the forecast accuracy is given in figure 7.5(b), calculated as the average over all polynomial orders and window sizes. Higher order derivatives performed better during forecasting and stabilized after the fourth derivative.

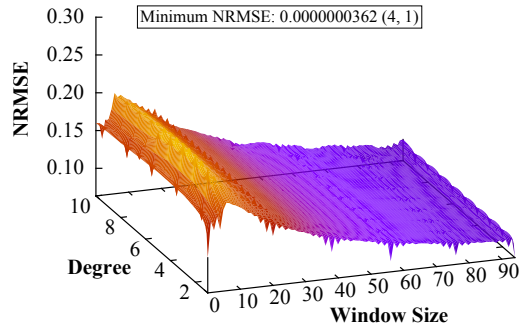
Due to the free endpoint condition that is applied to the first and last samples of a window, splines often have a poor prediction accuracy. In addition, forecasting one sample at a time with splines is computationally expensive, is infeasible for prediction-based outlier detection and was therefore omitted from the results.

7.2.2 Fourier Polynomials

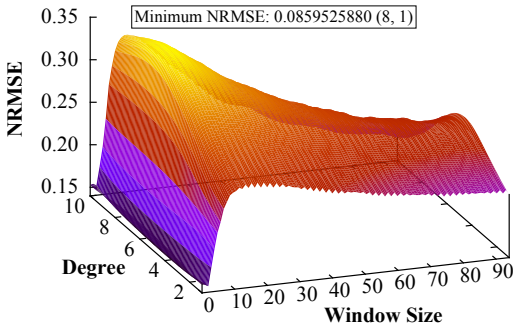
The interpolation and prediction performance of Fourier polynomials from section 3.2 is provided in this section. The Fourier polynomials are also employed in an osculating fashion as described in section 3.5 and by using splines discussed section 3.6.

Fourier Polynomial Interpolation

Both the model and interpolation accuracy of Fourier polynomials quickly decreased with an increase in the polynomial order as depicted in figure 7.6. In both cases a Fourier polynomial of degree one achieved the best results. The influence of the window size played a smaller role, with 350 samples or lower performing slightly better than larger window sizes.

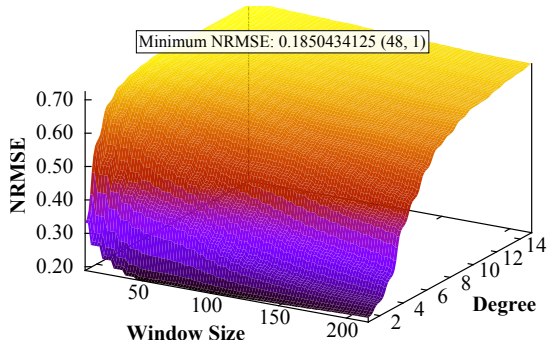


(a) The model accuracy for different window sizes and degrees.

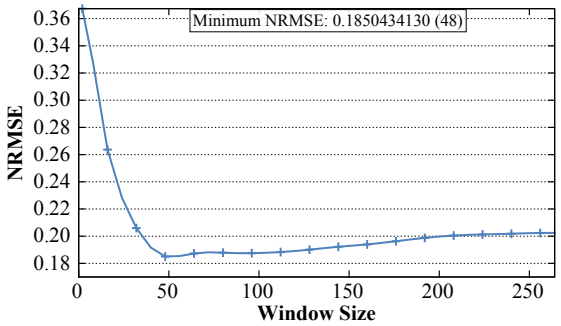


(b) The interpolation accuracy for different window sizes and degrees.

Figure 7.3: The accuracy of standard polynomial splines for different parameter configurations.

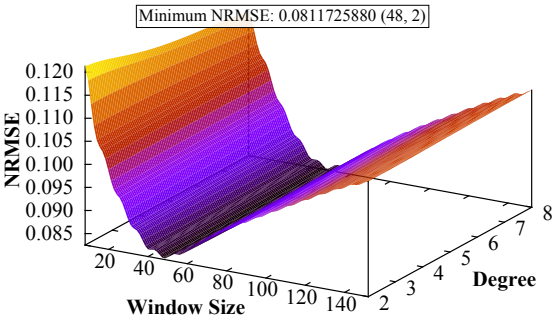


(a) The prediction accuracy for different window sizes and degrees.

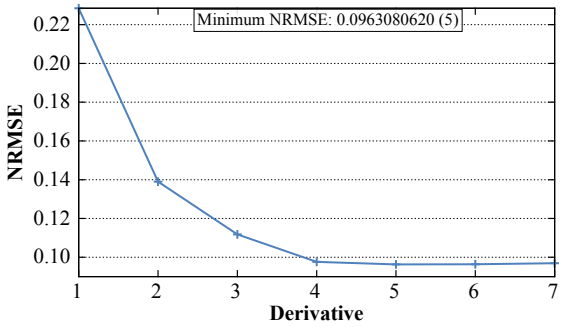


(b) The prediction accuracy of linear standard polynomials for different window sizes.

Figure 7.4: The prediction accuracy of standard polynomials for different parameter configurations.



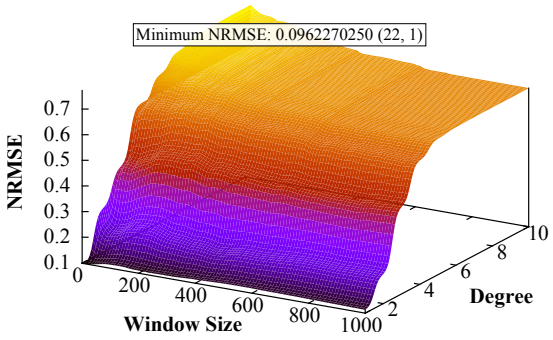
(a) The prediction accuracy for different window sizes and degrees.



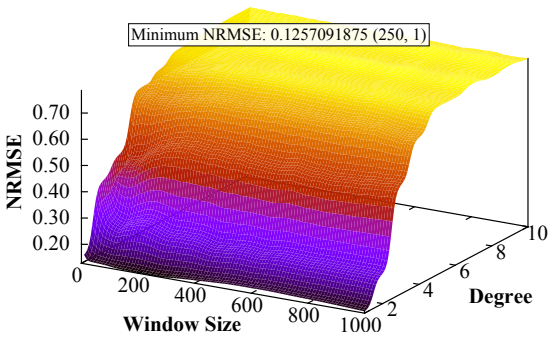
(b) The prediction accuracy for different derivatives.

Figure 7.5: The prediction accuracy of osculating standard polynomials for different parameter configurations.

Using osculating Fourier polynomials showed more promising results for higher orders. Like with standard polynomials, the high anomalies for high degree polynomials with small window sizes in figure 7.7(c) and 7.7(d) was caused by a lack of data points to accurately approximate the coefficients. The accuracy deviation between different model orders was reduced and stabilized at window sizes of 200 samples and greater. Figure 7.7(e) and 7.7(f) show a decrease in the NRMSE as the number of derivatives increased. The best

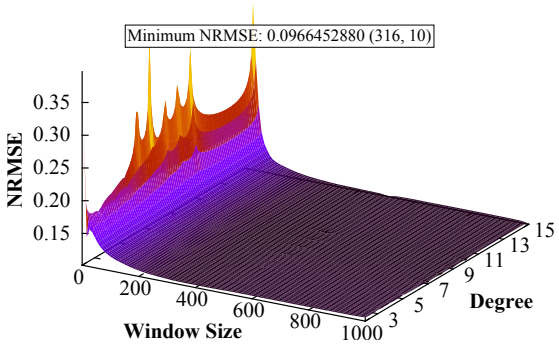


(a) The model accuracy for different window sizes and degrees.

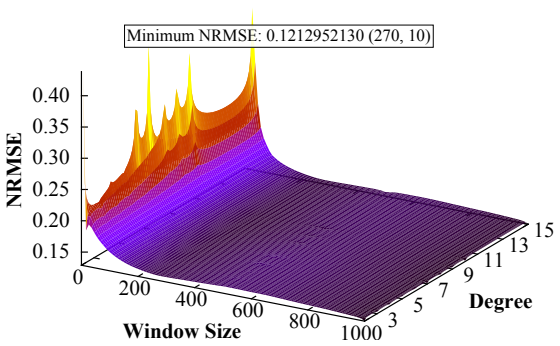


(b) The interpolation accuracy for different window sizes and degrees.

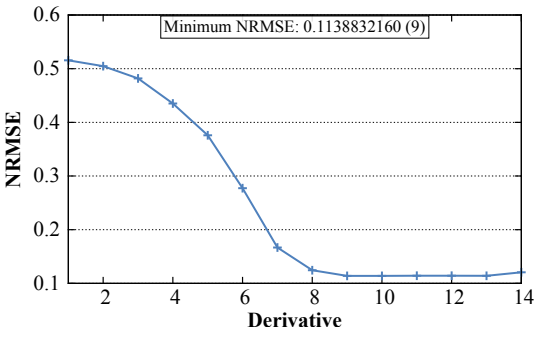
Figure 7.6: The accuracy of Fourier polynomials for different parameter configurations.



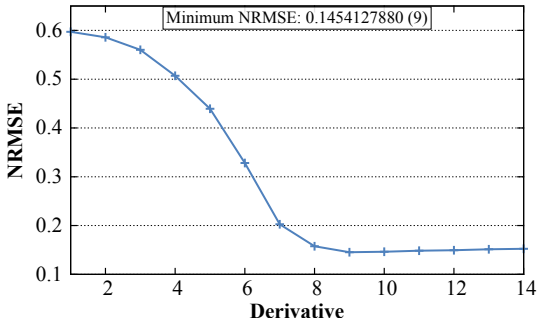
(c) The model accuracy for different window sizes and degrees.



(d) The interpolation accuracy for different window sizes and degrees.



(e) The model accuracy for different derivatives.

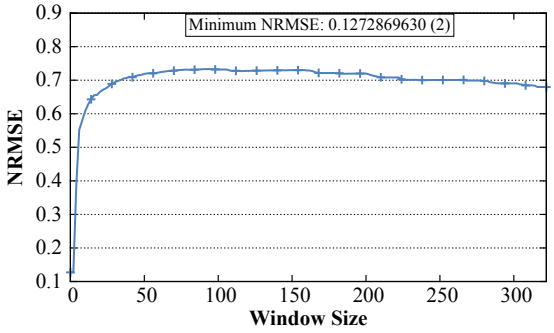


(f) The interpolation accuracy for different derivatives.

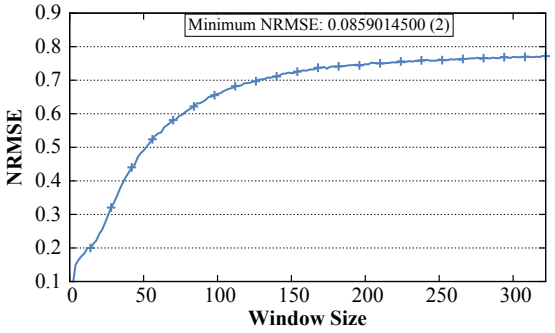
Figure 7.7: The accuracy of osculating Fourier polynomials for different parameter configurations.

parameters for both the observed and interpolated data points were at a degree of ten and the corresponding maximum derivative order of nine. Unlike standard polynomials, Fourier interpolation clearly benefited when applied in an osculating fashion.

Fourier splines can only be created with an order of one. For every increase in the order, two additional coefficients are added, one for the sine and one for the cosine waves. Due to the double increase in coefficients, not enough equations can be constructed to approximate higher degree Fourier splines. Only Fourier splines of degree one have enough equations if the free endpoint condition is applied to both the first and last spline. The NRMSE for the observed values in figure 7.8(a) and the interpolated values in figure 7.8(b) show that a window size of only two samples had the best fit. Although the model accuracy was slightly lower than that of Fourier and osculating Fourier polynomials, Fourier splines had a substantial increase in their interpolation accuracy, with a NRMSE reduction of approximately 0.04.



(a) The model accuracy for different window sizes and degrees.



(b) The interpolation accuracy for different window sizes and degrees.

Figure 7.8: The accuracy of Fourier splines for different parameter configurations.

Fourier Polynomial Prediction

Although Fourier polynomials were inferior to standard polynomials for interpolation purposes, Fourier polynomials performed better during forecasting. The best prediction accuracy was achieved over a window size of 64 samples with ten sine and cosine waves as depicted in figure 7.9. Higher degree Fourier polynomials predicted more accurately,

however, there was little statistical difference between the forecast of Fourier polynomials of different orders. In addition, higher order polynomials took longer to approximate. For instance, a tenth degree Fourier polynomial with a window size of 64 samples took approximately ten times longer to estimate than a first degree polynomial, but had an insignificant error reduction of only 0.01. The long execution time of high degree Fourier polynomials make them impractical for noise detection purposes and a first degree polynomial was therefore utilized for the outlier detection.

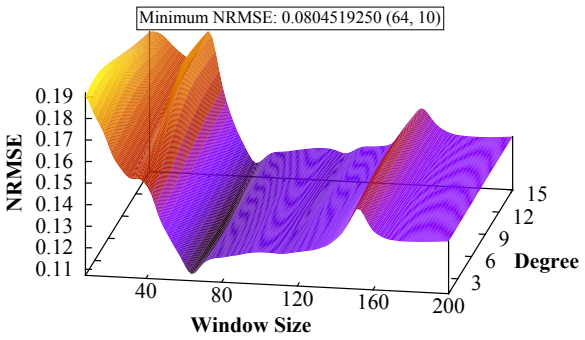
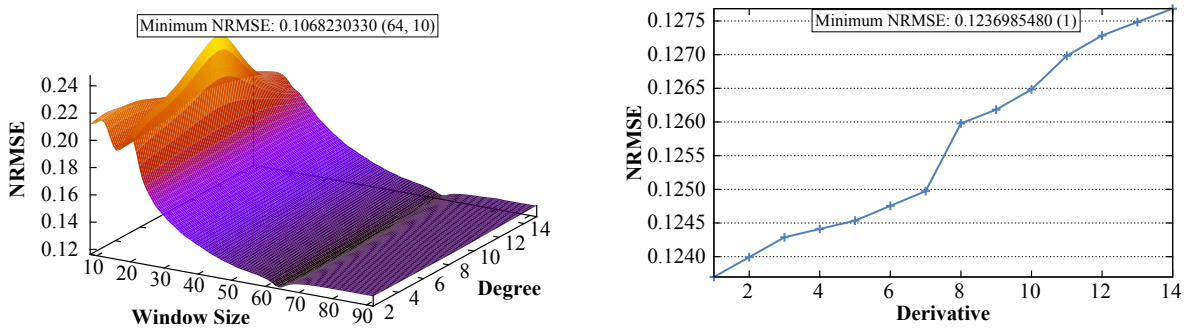


Figure 7.9: The prediction accuracy of Fourier polynomials for different parameter configurations.

Figure 7.10(a) shows the affect of the window size and the degree of osculating Fourier polynomials on the prediction accuracy. Although interpolation slightly benefited from adding derivatives to the Fourier polynomial, the forecast accuracy decreased with osculating Fourier polynomials. The influence of the derivatives on the prediction is given in figure 7.10(b). Osculating Fourier prediction had an opposite trend compared to the interpolation, where higher derivatives reduced the forecasting accuracy. Since higher order osculating polynomials are computationally expensive and add little improvement to the prediction accuracy, a quadratic osculating polynomial with the first derivative was employed for the predictive noise detection.

Like standard polynomial splines, Fourier splines are inaccurate for forecasting purposes, are computationally expensive and were therefore omitted from this study.



(a) The prediction accuracy for different window sizes and degrees.

(b) The prediction accuracy for different derivatives.

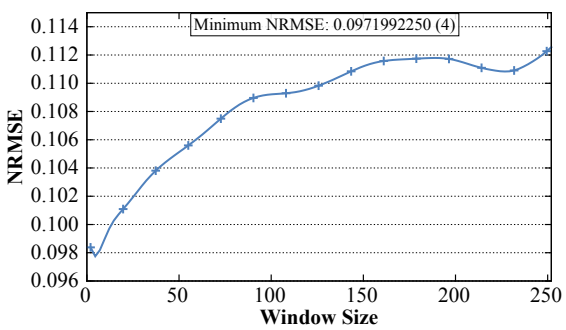
Figure 7.10: The prediction accuracy of osculating Fourier polynomials for different parameter configurations.

7.2.3 Hermite, Lagrange and Newton Polynomials

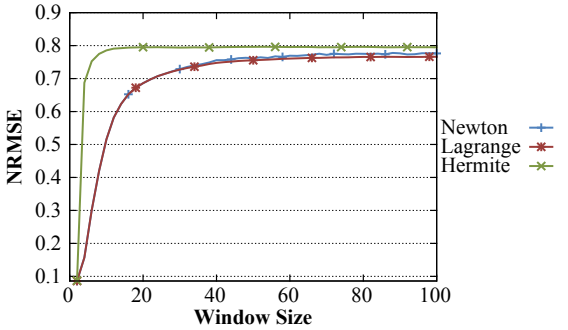
This section analyses the ability of Newton, Lagrange and Hermite polynomials to interpolate and predict audio signals as discussed in section 3.3, section 3.4 and section 3.5 respectively.

Hermite, Lagrange and Newton Polynomial Interpolation

Hermite, Lagrange and Newton polynomials followed a similar trend with the best interpolation accuracy achieved with a polynomial running trough only two points as depicted in figure 7.11(b). Since Lagrange and Newton polynomials are equivalent, Newton polynomials were estimated using a LLS fit to determine if coefficient approximation improves the interpolation process. The approximated Newton polynomials achieved almost identical results to the Lagrange polynomials for window sizes up to 32 samples. For larger window sizes, Newton polynomials had a minor increase in the NRMSE compared to Lagrange polynomials. The model accuracy of Newton polynomials in figure 7.11(a) indicates that the observed data points were fitted well, but employing the model for interpolation resulted in a poor performance. Due to the nature of the Lagrange and Newton polynomials in equations (3.23) and (3.17) respectively, achieving their best performance with two samples makes the constructed polynomials equivalent to linear



(a) The model accuracy of Newton polynomials for different window sizes.



(b) The interpolation accuracy for different window sizes.

Figure 7.11: The accuracy of Hermite, Lagrange and Newton polynomials for different parameter configurations.

polynomials in equation (3.6). Newton polynomials deviated from the results of linear polynomials, since they were approximated with LLS regression and not computed in the traditional way using divided differences. The quick increase to a NRMSE of almost 0.8 shows that Hermite, Lagrange and Newton polynomials are inadequate to interpolate music signals.

Hermite, Lagrange and Newton Polynomial Prediction

Similar to the interpolation, predicting with Hermite, Lagrange and Newton polynomials was inaccurate as shown in figure 7.12. Lagrange and Newton polynomials performed best for a window size of two samples, but their error quickly increased when computed over more samples.

7.2.4 Autoregressive and Moving Average Models

This section discusses the capacity to which the AR, MA, ARMA and ARIMA models were able to interpolate and predict music signals as discussed in section 3.7, 3.8, 3.9 and section 3.10 respectively. The AR model coefficients were approximated using a LLS fit, whereas the MA, ARMA and ARIMA models were estimated with CML. The BHHH algorithm was utilized for the CML, where BFGS served as a fallback if the characteristics

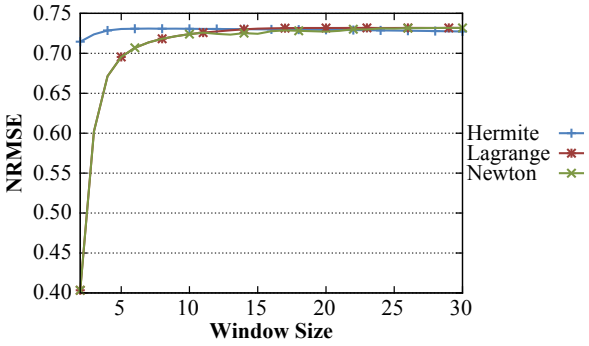


Figure 7.12: The prediction accuracy of Hermite, Lagrange and Newton polynomials for different window sizes.

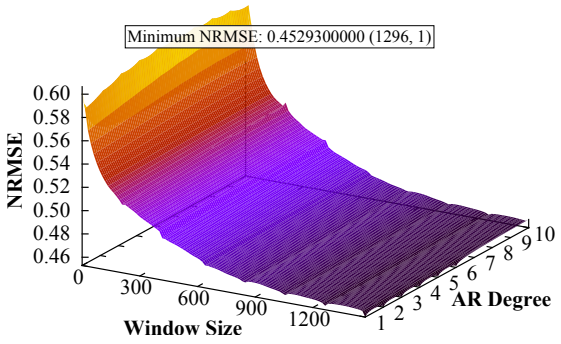
of the input data, the model type and order resulted in a poor approximation using the BHHH algorithm.

Autoregressive and Moving Average Model Interpolation

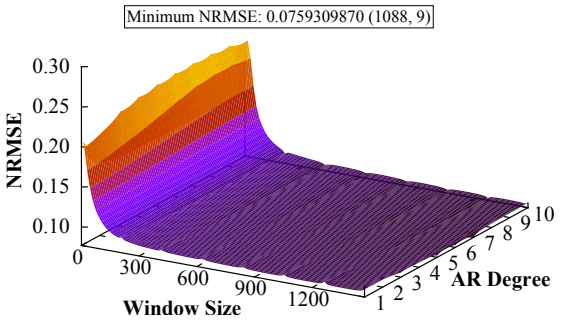
Compared to the interpolation performance of the polynomials priorly discussed in this chapter, the AR models performed notably better. Due to the nature of the AR process, the model fitness had a very high NRMSE as shown in figure 7.13(a). However, the interpolation accuracy in figure 7.13(b) was substantially better and reached the optimal configuration with the AR(9) model computed over 1088 samples. The AR model struggled to fit a small number of data points, but as the window size increased, the linear dependency of its previous values decreased both the model and the interpolation error.

The MA model for the observed values in figure 7.14(a) shows a similar trend to the AR model. However, due to the Guassian white noise, the MA interpolation accuracy in figure 7.14(b) shows a different trend, where smaller window sizes performed better. Increasing the order of the MA model added little to no improvement when employed on its own. The MA(1) model performed best, reaching its minimum NRMSE at a window size of four samples.

Figure 7.15 shows the relationship between the AR and the MA orders in an ARMA model. An increase in the AR order decreased the model fitness for the observed

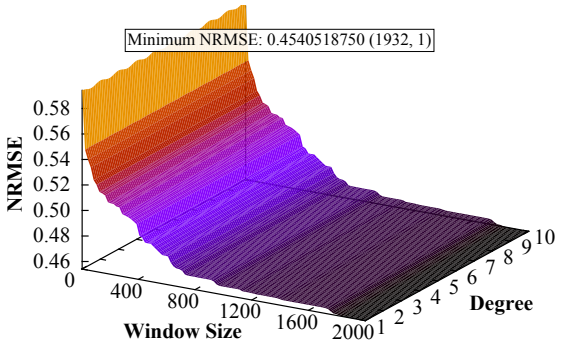


(a) The model accuracy for different window sizes and degrees.

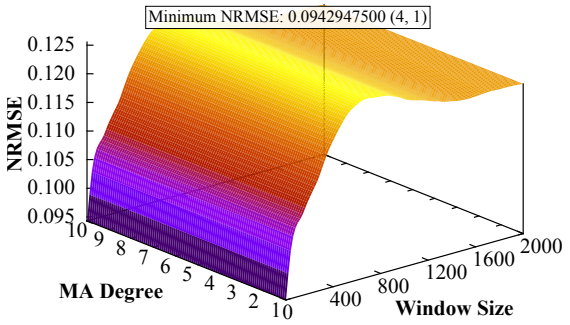


(b) The interpolation accuracy for different window sizes and degrees.

Figure 7.13: The accuracy of AR models for different parameter configurations.



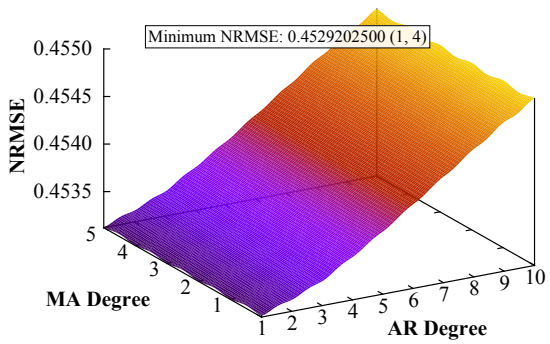
(a) The model accuracy for different window sizes and degrees.



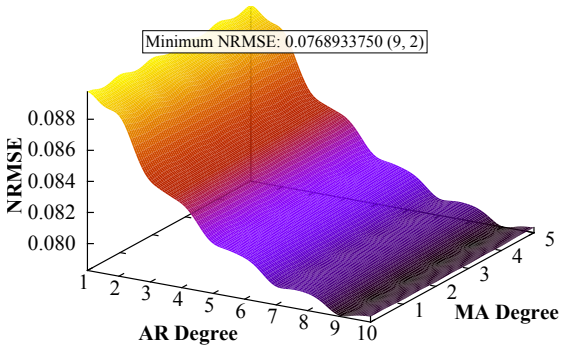
(b) The interpolation accuracy for different window sizes and degrees.

Figure 7.14: The accuracy of MA models for different parameter configurations.

values, but improved the interpolation accuracy. The MA component added little to no improvement for both the observed and interpolated models. The ARMA(9,2) model performed best for all window sizes and reached its minimum error when calculated over 1440 samples. The addition of an integrated component to the ARMA model is illustrated in figure 7.16. The model for the observed values in figure 7.16(a) shows a similar trend to the AR, MA and ARMA models. Figure 7.16(b) reveals that the addition of an iterated part did not improve the model's interpolation accuracy. The ARIMA(9,1,4) model achieved the bet results with a window size of 1312 samples.

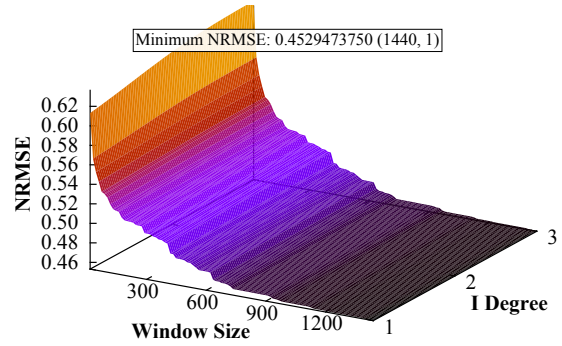


(a) The model accuracy for different degrees.

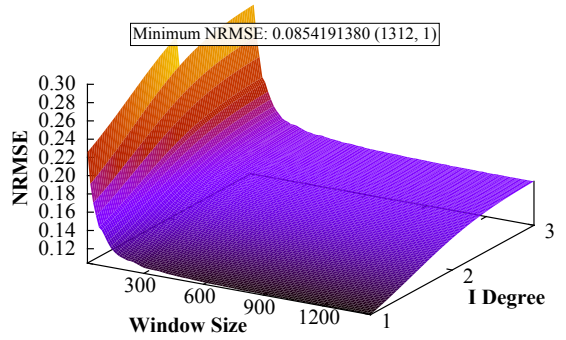


(b) The interpolation accuracy for different degrees.

Figure 7.15: The accuracy of ARMA models for different parameter configurations.



(a) The model accuracy for different window sizes and degrees.



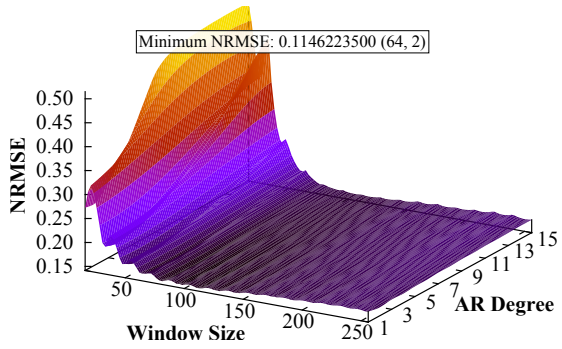
(b) The interpolation accuracy for different window sizes and degrees.

Figure 7.16: The accuracy of ARIMA models for different parameter configurations.

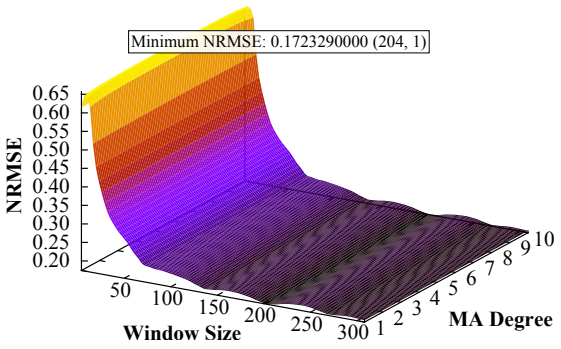
Autoregressive and Moving Average Model Prediction

The prediction accuracy of the AR and MA models is illustrated in figure 7.17(a) and figure 7.17(b) respectively. Both models followed a similar trend resulting in a low forecast accuracy when computed over window sizes smaller than 50 samples. The best prediction was achieved with an AR(2) and MA(1) model using a window of 64 and 204 samples respectively.

The relation between the AR and MA orders in the ARMA model is given in figure 7.18(a). The ARMA(2,2) model predicted most accurately, however, with a slight decline



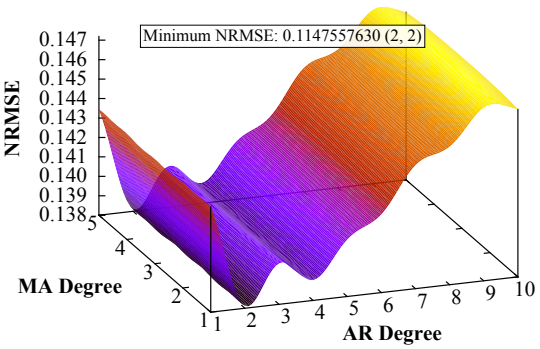
(a) The prediction accuracy of AR models for different window sizes and degrees.



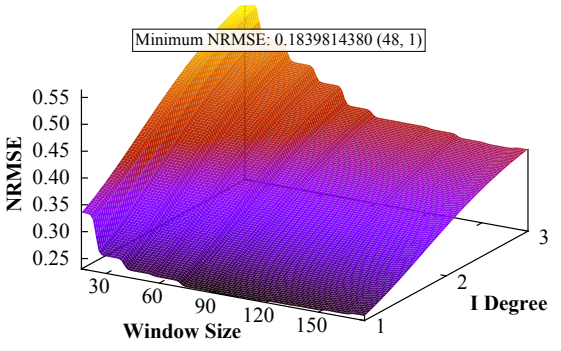
(b) The prediction accuracy of MA models for different window sizes and degrees.

Figure 7.17: The prediction accuracy of AR and MA models for different parameter configurations.

compared to the AR(2) model. Figure 7.18(b) shows the effect of extending the ARMA model with an integrated part. Forecasting quickly deteriorated with an increase in the integrated order. The ARIMA(2,1,1) model with a window of 48 samples predicted most accurately.



(a) The prediction accuracy of ARMA models for different degrees.



(b) The prediction accuracy of ARIMA models for different window sizes and degrees.

Figure 7.18: The prediction accuracy of ARMA and ARIMA models for different parameter configurations.

Model Order Selection

Table 7.2 shows the AR, ARMA and ARIMA models' interpolation performance for different model order selection criteria as discussed in section 3.9. The optimized parameters were used for the fixed models, that is a window size of 1088, 1440 and 1312 for the AR(9), ARMA(9,2) and ARIMA(9,1,4) models respectively. Although the models had a minor error reduction for most model selection criteria, the substantial increase in the computational speed does not justify the utilization of these criteria. The AR model had a five fold increase in the execution speed when utilizing a model selection criteria, whereas the ARIMA model had an even bigger increase of 706. Since the employment of selection criteria is practically infeasible for processing audio signals, a fixed model order was used for the time series models. Using ARCH and GARCH models in conjunction with a model selection criteria had a similar trend to the results in table 7.2.

Table 7.2: The influence of model selection criteria on the interpolation performance of the AR, ARMA and ARIMA models.

Criteria	AR		ARMA		ARIMA	
	NRMSE	Speed (s\s)	NRMSE	Speed (s\s)	NRMSE	Speed (s\s)
Fixed	0.071107	**0.074627**	0.071045	**0.520056**	0.076689	**1.444043**
$\mathbf{R^2}$	0.103022	0.359764	-	-	-	-
$\mathbf{R_a^2}$	0.103022	0.362720	-	-	-	-
ACF	0.074018	2.207516	0.074053	2.268634	0.077763	2.330745
MSE	**0.069094**	0.403360	**0.069483**	77.95319	**0.075127**	319.6081
AIC	0.069226	0.402776	0.069588	248.1559	0.075232	1017.439
AICC	0.069243	0.401304	0.069586	248.3064	0.075230	1018.056
BIC	0.069515	0.399242	0.069563	248.3685	0.075207	1018.311
HQIC	0.069364	0.399615	0.069512	248.2319	0.075156	1017.750

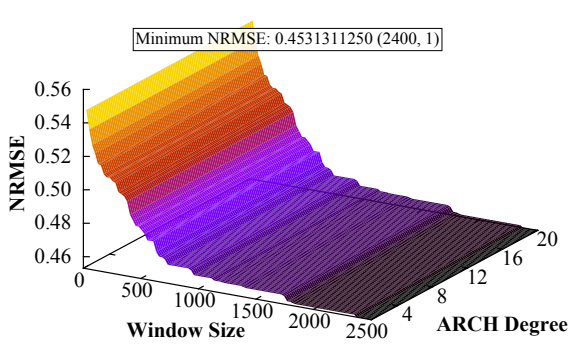
7.2.5 Heteroscedasticity Models

This section provides an overview of the heteroscedasticity models, ARCH and GARCH, as examined in section 3.11 and 3.12 respectively. The interpolation and prediction

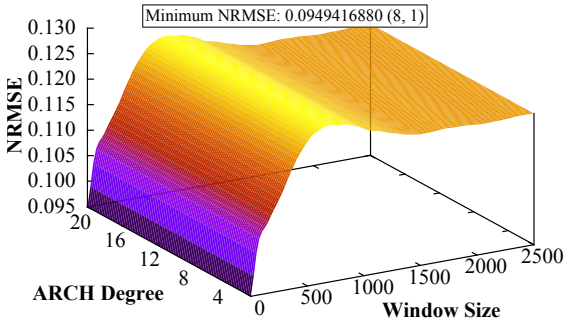
accuracy of both heteroscedasticity models for music signals were evaluated. The ARCH and GARCH models were both estimated using CML.

Heteroscedasticity Model Interpolation

The model and interpolation accuracy of the ARCH model is depicted in figure 7.19. An increase in the window size resulted in a better model fitness, whereas the interpolation benefited more from smaller window sizes. Both the model and interpolation performed best with a ARCH(1) model. Since the ARCH process models the time series using Gaussian white noise, the results were similar to that of the MA model in figure 7.14.



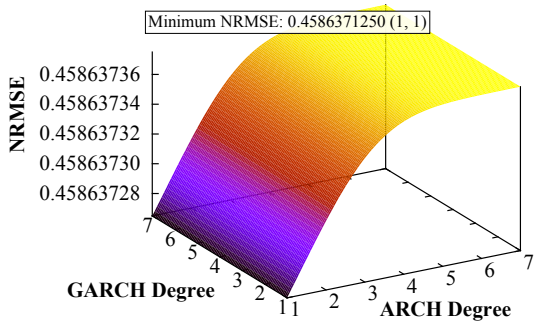
(a) The model accuracy for different degrees.



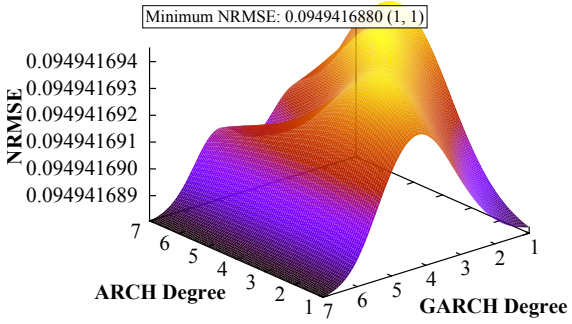
(b) The interpolation accuracy for different degrees.

Figure 7.19: The accuracy of ARCH models for different parameter configurations.

The relationship between the ARCH and GARCH orders of the GARCH model is illustrated in figure 7.20. Figure 7.20(a) shows that the GARCH order had little effect on the model fitness, whereas an increase in the ARCH order lead to an increase in the model NRMSE. The GARCH(1,1) model performed best. The error deviation with different ARCH and GARCH orders were so marginal that only a distinction in the eighth decimal digit or the NRMSE could be observed. Although the ARCH and GARCH models had an acceptable interpolation accuracy, the error increased compared to the AR and ARMA models. This decline indicates that employing the AR and ARMA models in a heteroscedasticity fashion is not suitable for reconstructing audio data.



(a) The model accuracy for different degrees.

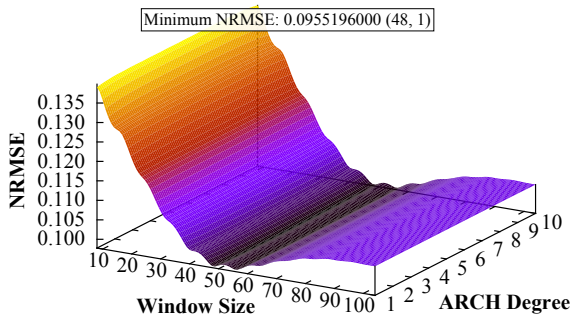


(b) The interpolation accuracy for different degrees.

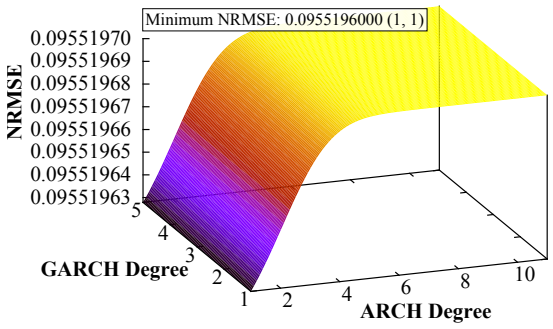
Figure 7.20: The accuracy of GARCH models for different parameter configurations.

Heteroscedasticity Model Prediction

Figure 7.21(a) shows the forecasting error of the ARCH model with different window sizes and degrees. The ARCH(1) model achieved the best prediction with 48 samples. Figure 7.21(b) shows the relationship between the ARCH and GARCH degrees. The addition of a GARCH component to the model did not increase the prediction accuracy and the GARCH(1,1) model prediction was close to that of the ARCH(1) model. Since



(a) The prediction accuracy of ARCH models for different window sizes and degrees.



(b) The prediction accuracy of GARCH models for different degrees.

Figure 7.21: The prediction accuracy of the ARCH and GARCH models for different parameter configurations.

both these models were originally designed to predict instead of interpolate signals, the forecasting accuracy compared to the AR, MA, ARMA and ARIMA models was notably better, whereas the interpolation performed worse.

7.2.6 Artificial Neural Network Model

This section discusses the implementation and parameters of various ANNs as introduced in section 3.13 and their ability to interpolate and forecast music signals.

Artificial Neural Network Model Interpolation

Given K as the sample length of the largest gap in the signal and k as the length of the current gap to be interpolated for $1 \leq k \leq K$, different ANNs were utilized for the reconstruction process as follows:

- **Forward incremental TDANN (FI-TDANN):** A TDANN was trained by sequentially moving a window one sample at a time over the input signal. Incremental training updated the weights using only historical data. The TDANN had K output neurons. Therefore, if a gap of k samples had to be reconstructed, the TDANN predicted a series of K samples and only used the first k outputs for the reconstruction.

- **Bidirectional incremental TDANN (BI-TDANN):** A FI-TDANN only operated on historical data without considering the future direction of the signal. Two separate FI-TDANNs were incrementally trained, one processing the signal from start to end, the other one from end to start. The gap was then reconstructed by taking the average of the output of both FI-TDANNs.

- **Forward incremental SRTDANN (FI-SRTDANN):** A Jordan SRTDANN network was trained incrementally on historic data. The SRTDANN had only one output neuron which was linked with a recurrent connection as an additional neuron to the input layer. The SRTDANN recurrently predicted one sample at a time until the entire gap was reconstructed.

- **Bidirectional incremental SRTDANN (BI-SRTDANN):** Similar to the FI-SRTDANN, two distinct Jordan SRTDANNs were trained on the signal in both directions. The average output of both SRTDANNs was used to interpolate the samples.

- **Forward separate batch TDANN (FSB-TDANN):** A training set was created with the historical data of the input signal. The weights of the TDANN were updated once for all training patterns during an epoch using a batch training algorithm. A total of K TDANNs were maintained, that is, one TDANN for each possible gap size. Each TDANN had a different number of output neurons, equal to the number of samples in the gap. If a gap of k samples was encountered, the corresponding k^{th} TDANN was selected from the set, trained and then utilized to reconstruct the k missing samples.

- **Bidirectional separate batch TDANN (BSB-TDANN):** The signal was processed in both directions using a set of FSB-TDANNs and the average of the reconstruction process was used to interpolate the gap. No additional TDANNs had to be maintained, since one of the K TDANNs from the forward process was simply reused for the backward interpolation.

- **Forward complete batch TDANN (FCB-TDANN):** Training patterns were generated using the historical samples of the signal and the weights were updated using a batch training algorithm. The TDANN had K output neurons. If a gap of size k had to be interpolated, only the first k outputs from the K output neurons were used.

- **Bidirectional complete batch TDANN (BCB-TDANN):** Two separate FCB-TDANNs were trained, one for the forward interpolation and one for the backward interpolation of the signal. The average output of both FCB-TDANNs was used to reconstruct the gap.

- **Interpolation batch TDANN (IB-TDANN):** This TDANN had two sets of inputs which were k samples apart. The first set of inputs consisted of the consecutive samples preceding the gap, the second set of the sample succeeding the gap. A total

of K IB-TDANNs were batch trained, one for each possible gap duration. When a gap of size k was reconstructed, the k^{th} IB-TDANN was selected from the set, trained with the patterns and then utilized to interpolate the missing samples.

Incremental training for the FI-TDANN, BI-TDANN, FI-SRTDANN and the BI-SRTDANN kept the training time low by only presenting each sample pattern once during the weight updates. Batch training for the FSB-TDANN, BSB-TDANN, FCB-TDANN, BCB-TDANN and the IB-TDANN on the other hand presented each training pattern z times to the TDANN during the weight updates, where z is the number of training epochs. Batch training was, therefore, unacceptably slow when processing a single song consisting of millions of training patterns. In addition, a music signal changes slowly, meaning that most training patterns were very similar when moving a window one sample at a time over the input signal. In order to reduce the computational time of batch training, the number of training patterns for each interpolation was limited to a maximum value v. Training patterns were also accumulated by delaying the patterns by τ samples instead of one sample. Both v and τ were determined empirically. The lowest value for v and the highest value for τ was chosen in such a way that the ANN's output error was not increased compared to the ANN's performance for τ equal to one and v equal to the maximum number of patterns preceding the interpolation gap. Therefore, the parameters v and τ reduced the training time without affecting the output error of the ANNs.

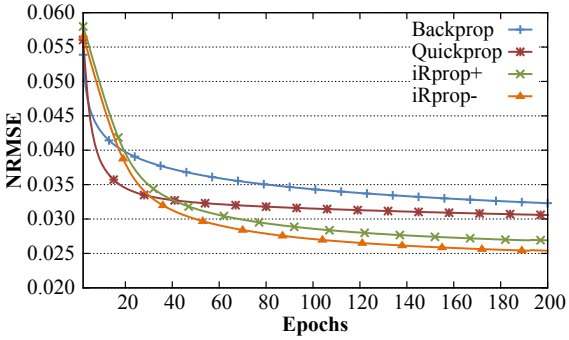
Table 7.3 lists the optimal parameters for each ANN which were obtained through factorial design. The layers' structure is given in the format g-o, where g represents the number of input neurons and o the number of output neurons. Parameter optimization was conducted on an input layer of up to 1536 neurons with up to three hidden layers each containing a maximum of 512 neurons. It was found that all TDANNs performed best without any hidden layers, which were therefore a set of perceptrons. The PCCs in table 7.1 support these findings, indicating that there is a strong linear dependency between the inputs and outputs of the perceptrons. Since most music signals are relatively stable, a linear combination of the input samples generally achieves a better interpolation accuracy than a nonlinear input composition.

Table 7.3: The optimal parameter configurations for interpolating ANNs.

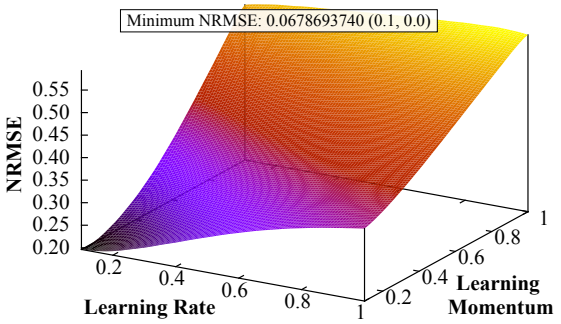
ANN Type	Layer Structure	Training Algorithm	Learning Rate	Activation Function	Training Epochs	Training Patterns
FI-TDANN	832-K	Incr. Backprop.	0.1	Sym. Elliot	1	-
BI-TDANN	832-K	Incr. Backprop.	0.1	Sym. Elliot	1	-
FI-SRTDANN	961-1	Incr. Backprop.	0.1	Hyp. Tangent	1	-
BI-SRTDANN	961-1	Incr. Backprop.	0.1	Hyp. Tangent	1	-
FSB-TDANN	224-k	iRprop$^-$	-	Sym. Elliot	50	256-6
BSB-TDANN	224-k	iRprop$^-$	-	Sym. Elliot	50	256-6
FCB-TDANN	220-K	iRprop$^-$	-	Sym. Elliot	50	256-8
BCB-TDANN	220-K	iRprop$^-$	-	Sym. Elliot	50	256-8
IB-TDANN	256-k	iRprop$^-$	-	Sym. Elliot	50	768-8

Figure 7.22a shows the average training NRMSE for the FSB-TDANN, FCB-TDANN and IB-TDANN using different training algorithms. The detailed training algorithm results for the individual ANNs are given in appendix D in figure D.1. Standard batch backpropagation and quickprop performed better for short training durations. However, iRprop$^-$ had a lower error from epoch 17 onwards, whereas iRprop$^+$ took longer and only outperformed quickprop from epoch 41 onwards. iRprop$^+$ and iRprop$^-$ produced similar results and the deviation in their error increased as training continued over more epochs. iRprop$^-$ performed best overall and was therefore utilized as the training algorithm for the batch trained ANNs.

Figure 7.22b illustrates the affect of the learning rate and learning momentum on the interpolation error of the incrementally trained ANNs. The results for the individual ANNs are given in appendix D in figure D.2. The interpolation error grew with an increase in the learning rate and momentum. The autocovariance of music signals is small, meaning the signal changes little over time. Hence, incremental training performed better with lower learning rates, where smaller consecutive weight updates corresponded to the slow changes in the input signal. Learning at a rate of 0.1 without a momentum proved most effective and more than halved the ANNs output error compared to higher learning rates above 0.4. Since Rprop is an adaptive training algorithm, it does not require a learning rate or momentum.



(a) The average training NRMSE of the batch trained ANNs for different training algorithms.



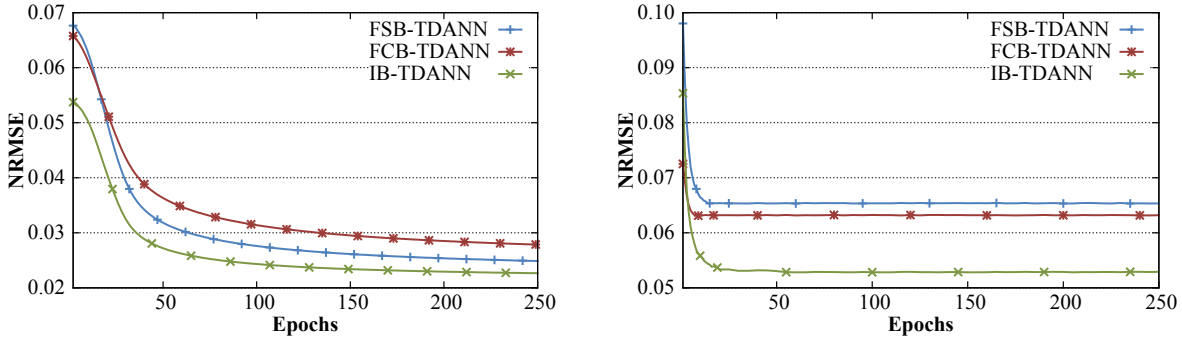
(b) The average interpolation NRMSE of the incrementally trained ANNs for different learning rates and momentums.

Figure 7.22: The average output NRMSE of the ANNs for different parameter configurations.

Various activation functions were tested for each ANN, including the symmetric Elliot function, the hyperbolic tangent function, a bounded linear function, the symmetric Gaussian function, and the symmetric sine and cosine functions. The symmetric Elliot activation function produced the best results for all ANNs, except for the SRTDANNs which benefited more from the hyperbolic tangent activation function. Besides producing better results in most cases, the Elliot function was slightly faster to compute than the tanh function. The detailed results of the interpolation NRMSE and the execution speed using different activation functions is given in appendix D in table D.1 and table D.2 respectively.

Figure 7.23(a) shows the training error of the batch trained ANNs over a number of epochs and figure 7.23(b) the corresponding interpolation output error. Although the training error only stabilized around the 100^{th} epoch, the interpolation error decreased more quickly. The interpolation error of the FSB-TDANN steadied after the 16^{th} epoch, the FCB-TDANN after the 10^{th} epoch and the IB-TDANN after the 34^{th} epoch. Since the interpolation error only decreased marginally between the 40^{th} and 250^{th} epochs, the maximum number of epochs for all batch trained TDANNs was set to 50.

The last column in table 7.3 is given in the format v-τ. All batch trained ANNs, except the IB-TDANN, performed best with a maximum of 256 training patterns. These training patterns were collected either 6 or 8 samples apart.



(a) The ANN training error over an increasing number of training epochs.

(b) The ANN interpolation error over an increasing number of training epochs.

Figure 7.23: The batch ANN training and interpolation errors when trained over an increasing number of epochs.

Artificial Neural Network Model Prediction

Two different ANNs were utilized for prediction purposes, with the optimal parameter configurations given in table 7.4. The TDANN predicted all K samples at once, whereas the SRTDANN predicted one sample at a time, recurrently feeding the output back into the ANN as an additional neuron in the input layer. Both ANNs were trained incrementally using a learning rate of 0.1. The TDANN utilized the symmetric Elliot activation function, whereas the SRTDANN applied the hyperbolic tangent activation function. Similar to the interpolating TDANNs, the forecasting was more accurate using a linear combination of the preceding samples and the prediction ANNs therefore performed best without any hidden layers.

Table 7.4: The optimal parameter configurations for predicting ANNs.

ANN Type	Layer Structure	Training Algorithm	Learning Rate	Activation Function	Training Epochs
TDANN	12-K	Incr. Backprop.	0.1	Sym. Elliot	1
SRTDANN	17-1	Incr. Backprop.	0.1	Hyp. Tangent	1

Table 7.24 shows the forecasting error of both predictive ANNs when predicting further into the future. The TDANN and SRTDANN had a slight decrease in the NRMSE when

predicting up to 23 and up to 20 samples respectively. Forecasting further than 23 samples shows a quick increase in the error. Both ANNs had a average prediction NRMSE above 0.1 calculated over all gap sizes.

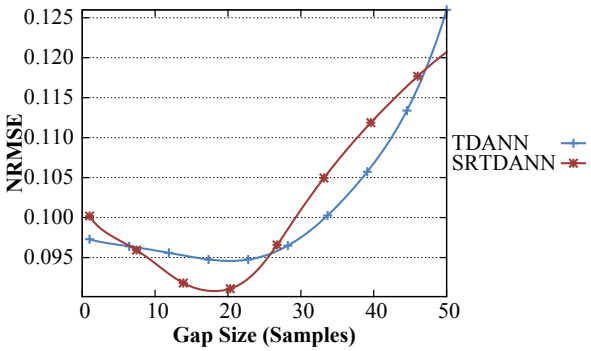


Figure 7.24: The ANN forecasting error for predicting further into the future.

7.3 Noise Detection Analysis

This section discusses the outlier detection algorithms from chapter 4. The parameters of the algorithms are analysed, followed by an inspection of the three examined noise masking techniques. Finally, the algorithms are compared according to their detection accuracy, computational speed and the tradeoff between the accuracy and speed.

7.3.1 Noise Detection Algorithm Parameters

The outlier detection performance of the SS for an increasing window size is illustrated in figure 7.25. The highest detection accuracy was achieved when the SS is computed over a window of 1868 samples. Windows with more than 1900 samples saw a steady decline in the MCC. Since the mean and standard deviation of the sequence headed towards zero as the window size increased beyond 1900, a division with an increasingly small value in equation (4.2) resulted in an amplified SS which on the other hand resulted in a lower MCC. The SS had an acceptable detection rate, even when computed over very few samples.

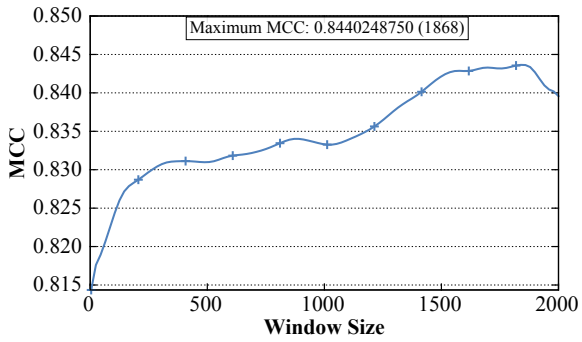


Figure 7.25: The noise detection accuracy of the SS for different window sizes.

The MAD for different window sizes is given in figure 7.26. Since MAD used the median instead of the mean, the chances of the score heading towards zero with an increasing window size were reduced. MAD stabilized at a window size of 136 samples and reached its best detection accuracy when calculated over 1600 samples.

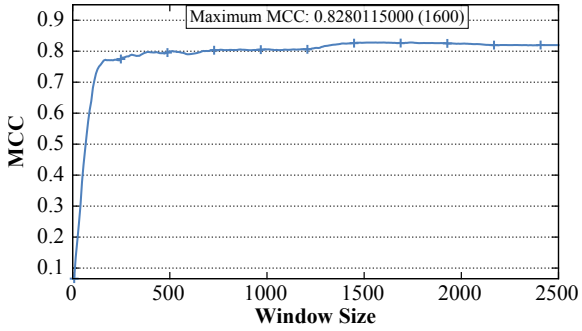


Figure 7.26: The noise detection accuracy of the MAD for different window sizes.

Figure 7.27 shows the outlier detection accuracy of the MHD for different window sizes. The MHD detection accuracy stabilized around a window size of 980 samples and reached its best performance at 2248 samples. Both the MAD and MHD performed poorly when computed over a small number of data points.

The detection performance of the NND algorithm using a global outlier factor is illustrated in figure 7.28. Even for a very small k of 100 samples or less, the algorithm preformed well. The best detection was achieved with k equal to 324. Larger windows

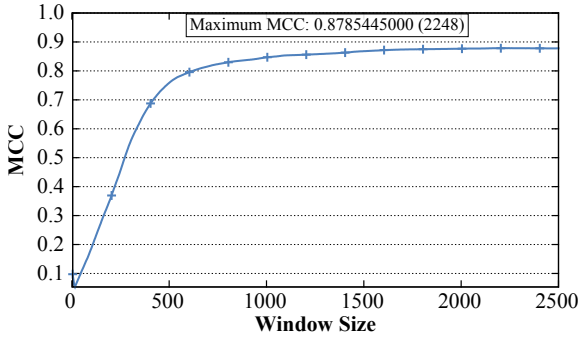


Figure 7.27: The noise detection accuracy of the MHD for different window sizes.

provided a continuous stable detection accuracy.

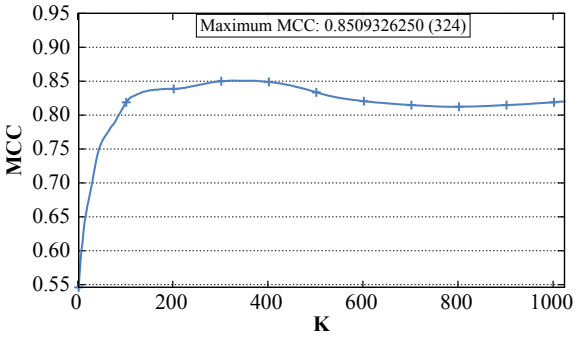


Figure 7.28: The noise detection accuracy of the NND for different window sizes.

Figure 7.29 shows the MASD outlier detection performance through the observation of changes in the frequency spectrum. Although the observation of frequencies beyond 15000 Hz resulted in an acceptable detection rate, lower frequencies were on average slightly more affected by noise. The best performance was achieved by observing frequencies between 1102 Hz and 5512 Hz which reached a sensitivity of approximately 0.432 and a specificity of 0.999. Other parts of the spectrum resulted in a sensitivity as high as 0.973 but with a reduced specificity of almost 0.8, indicating that it was difficult to distinguish the signal from the noise in those frequencies.

Table 7.5 shows the performance of the APD using the batch prediction algorithm 4 and the recurrent prediction algorithm 3. Recurrent prediction performed notably

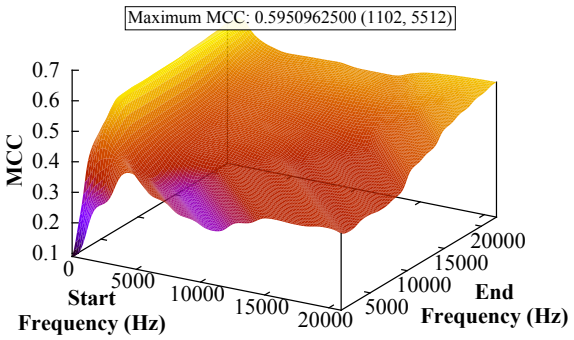


Figure 7.29: The noise detection accuracy of the MASD for different window sizes.

better for all models, indicating that the models were able to more accurately predict a single sample rather than a sequence of samples. The recurrent approach was therefore employed for the predictive outlier detection.

Table 7.5: The comparison between the batch and recurrent approaches for predictive outlier detection expressed in terms of the MCC.

Algorithm	Batch	Recurrent
APD-STP	0.649120	**0.817205**
APD-OSP	0.781792	**0.811854**
APD-FOP	0.575251	**0.820939**
APD-OFP	0.556653	**0.820760**
APD-NEP	0.541140	**0.809282**
APD-LAP	0.567202	**0.812545**
APD-HEP	0.581386	**0.818658**
APD-AR	0.796796	**0.822206**
APD-MA	0.791946	**0.772551**
APD-ARMA	0.796948	**0.832109**
APD-ARIM	0.729526	**0.833718**
APD-ARCH	0.771116	**0.823137**
APD-GARCH	0.771116	**0.823137**
APD-TDANN	0.765105	**0.778625**
APD-SRTDANN	0.770974	**0.816484**
Average	0.696405	**0.814214**

The full list of optimal parameters for all noise detection algorithms is given in appendix B in table B.1.

7.3.2 Noise Masking Comparison

Section 4.2 in chapter 4 introduced three alternative approaches for creating a noise mask from a noise map using a threshold. The masking techniques were benchmarked over all 800 songs and the results, expressed in terms of the MCC, are given in table 7.6. The APD represents the average over all forecasting models.

Table 7.6: The noise detection MCC performance of different noise masking techniques.

Algorithm	Standard Threshold	Mean Threshold	Maximum Threshold
SS	0.792384	**0.798677**	0.798666
MAD	**0.740720**	0.740155	0.740161
MHD	0.798867	**0.808759**	0.808748
NND	0.753863	**0.760939**	0.760933
MASD	**0.558229**	0.543735	0.543727
APD	**0.801338**	0.784376	0.784353
Average	**0.740900**	0.739440	0.739431

The standard thresholding technique performed best on average. Since additional time is required to calculate the mean and maximum without improving the detection rate significantly, the standard thresholding technique was chosen for the noise masking process.

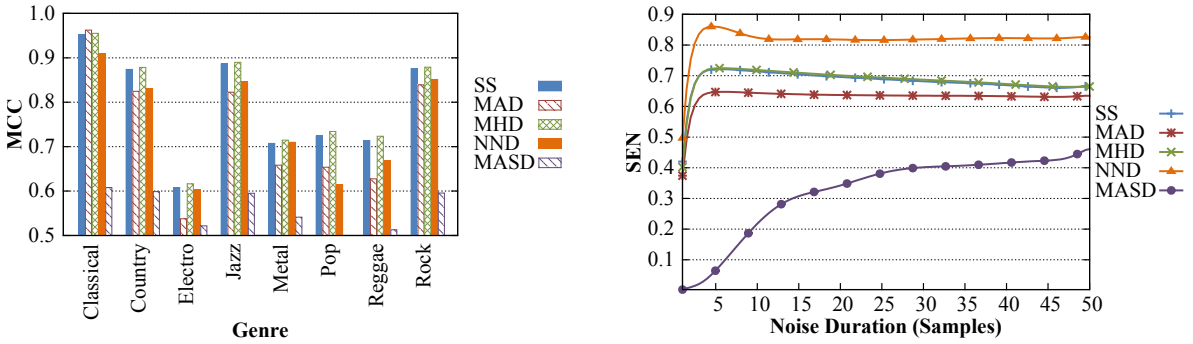
7.3.3 Noise Detection Algorithm Comparison

Figure 7.30(a) shows the noise detection performance of the proximity and spectral algorithms over different genres. The MAD performed best for classical music, but was inferior to the MHD for all other genres. The SS had a good detection accuracy, but still lagged behind the MHD. The spectral algorithm, MASD, performed considerably worse than the proximity approaches. Figure 7.30(b) illustrates the sensitivity for an increasing multivariate outlier duration. The NND had the highest sensitivity over all noise durations, but due to a reduced specificity, had a lower MCC compared to the SS

and MHD. MASD struggled to identify short noise sequences. Although not shown in the graph, MASD had a steady increase in the sensitivity for multivariate outlier durations longer than 50 samples and outperformed SS, MAD and MHD for noise durations beyond 93 samples. MASD's poor performance indicates that short noise bursts do not disrupt the frequencies to an extend that is observable by the algorithm. Since gramophone distortions rarely exceed 50 samples, MASD is deemed unfit for outlier detection and should only be employed in the rare occasion of a very long disruption.

Figure 7.31(a) gives the per-genre detection performance for the APD algorithm using polynomials. Although the APD-HEP detected outliers most accurately in classical music, it performed very poorly for all other genres. The other polynomials had a similar detection rate, with APD-NEP and APD-LAP performing slightly better for the more volatile electro, metal, pop and reggae signals. Figure 7.31(b) highlights the APD polynomial algorithms' sensitivity for an increasing noise duration. Approximately half of all single outlying samples were not identified and a stable detection rate was achieved with multivariate outliers longer than 15 samples. The FOP and OFP reached the overall best detection accuracy amongst the polynomials over all noise durations.

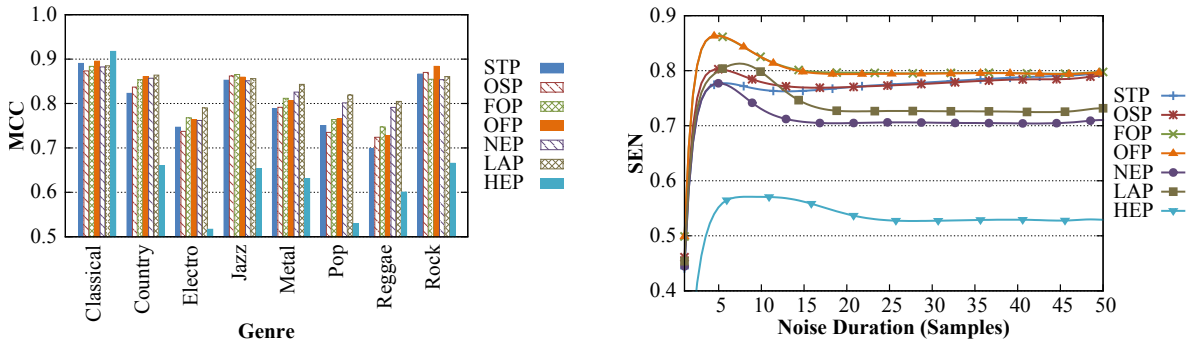
The noise detection performance for the predictive time series and ANN algorithms is given in figure 7.32(a) and figure 7.32(b). The AR, ARMA and ARIMA models achieved a



(a) The outlier detection accuracy for different genres.

(b) The outlier detection sensitivity for different noise durations.

Figure 7.30: The outlier detection accuracy of the proximity and spectral algorithms for different genres and noise durations.



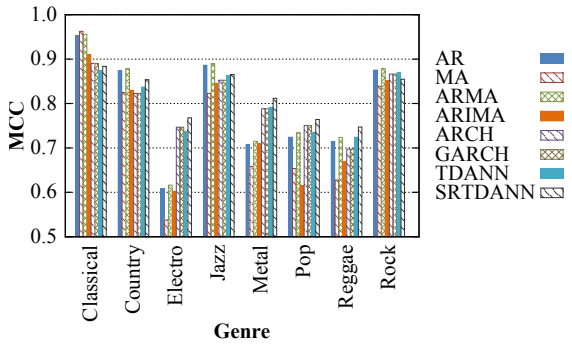
(a) The outlier detection accuracy for different genres.

(b) The outlier detection sensitivity for different noise durations.

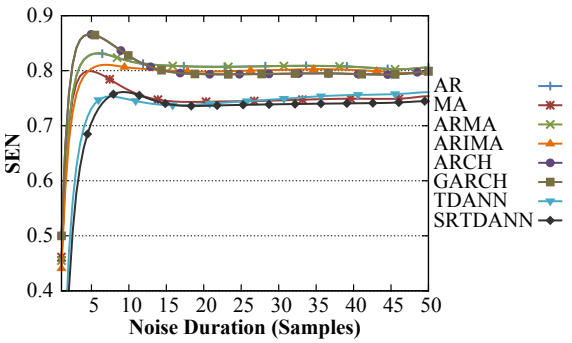
Figure 7.31: The outlier detection accuracy of the predictive polynomial-based algorithms for different genres and noise durations.

good detection accuracy for classical, country and jazz music. However, the volatile genres performed better under the ARCH and the GARCH models, since they are intended for highly irregular financial markets rather than stable signals. The APD-SRTDANN performed even better for more volatile series, achieving the best detection rate in electro, metal, pop and reggae. The models had a similar sensitivity trend over different noise durations as depicted in figure 7.32(b). More than half of the single outliers were not flagged, with a considerable sensitivity increase from durations of two samples onwards.

Table 7.7 shows the overall performance of the outlier detection algorithms. Note that the results in table 7.7 were calculated over the entire dataset of 800 songs and, therefore, slightly deviate from the results in table 7.5 which were computed from the test set of 80 songs. The NND had the highest sensitivity and was therefore able to identify most of the outliers. The MHD flagged the fewest inliers as noisy, which is indicated by the high specificity. The overall best detection accuracy between the correctly identified inliers and outliers was achieved by the ARIMA prediction, followed by the APD-AR and APD-ARMA algorithms. The fifth column in table 7.7 lists the sample standard deviation of the MCC over the entire dataset. A higher standard deviation indicates that noise in some tracks or parts of a track were easier to identify than in other songs. A lower standard deviation on the other hand stipulates that the detection accuracy across all songs is more consistent. The APD-SRTDANN achieved the most uniform MCC



(a) The outlier detection accuracy for different genres.



(b) The outlier detection sensitivity for different noise durations.

Figure 7.32: The outlier detection accuracy of the predictive model-based algorithms for different genres and noise durations.

over the entire dataset. Only five of the detection algorithms were able to execute in real-time using a single thread, where the APD-TDANN had the fastest computational speed. The TDANNs had a very low execution speed, since they did not have any hidden layers, had a small number of input neurons and were trained incrementally. The tradeoff between the detection accuracy and the computational speed is given in the last column. The APD-SRTDANN was most efficient by achieving a good detection rate in real-time, followed by the APD-TDANN.

Detailed reports on the detection rate for different noise durations, the performance over different genres and a full list of the MCC's sample standard deviation is given in appendix C.

7.4 Noise Reconstruction Analysis

This section empirically analyses the ability of the models and algorithms in chapter 3 and 5 to accurately reconstruct a sample gap that was previously flagged by a noise detection algorithm. The parameters of the NNI, SI and LI are analysed, followed by a comparison of all interpolation algorithms according to their reconstruction accuracy, computational speed and the tradoff between the accuracy and speed.

Table 7.7: The sensitivity, specificity, detection accuracy, detection accuracy's standard deviation, computational speed and tradeoff of the outlier detection algorithms.

Algorithm	SEN	SPE	MCC	MCC σ	Speed (s\s)	SAT
SS	0.681131	0.999762	0.792384	0.172186	5.466633	0.140497
MAD	0.636569	0.999370	0.740720	0.194504	19.16386	0.042976
MHD	0.685187	**0.999817**	0.798867	0.168352	15.47347	0.053309
NND	**0.824754**	0.993025	0.753863	0.174360	1.690628	0.350316
MASD	0.384057	0.998339	0.558229	0.083418	0.506437	0.459383
APD-STP	0.784893	0.997493	0.801771	0.124437	13.80646	0.059506
APD-OSP	0.783452	0.997617	0.803745	0.121825	30.85764	0.027268
APD-FOP	0.802095	0.997499	0.818400	0.116617	24.69167	0.033975
APD-OFP	0.800950	0.997428	0.820194	0.120362	68.22695	0.012473
APD-NEP	0.712309	0.999245	0.808044	0.074389	7.177062	0.110436
APD-LAP	0.736020	0.999149	0.800283	0.070196	0.770631	0.619005
APD-HEP	0.535626	0.998066	0.646568	0.137766	0.496814	0.541018
APD-AR	0.811191	0.998019	0.835521	0.111351	4.269161	0.179634
APD-MA	0.752519	0.995856	0.746852	0.170093	24.76732	0.033582
APD-ARMA	0.822540	0.997529	0.834558	0.108479	26.64355	0.031595
APD-ARIMA	0.803464	0.998261	**0.836681**	0.107817	11.49651	0.071495
APD-ARCH	0.801142	0.997946	0.829901	0.107119	14.58489	0.056782
APD-GARCH	0.801142	0.997946	0.829901	0.107119	14.58489	0.056782
APD-TDANN	0.753882	0.997744	0.780434	0.113581	**0.142083**	1.048069
APD-SRTDANN	0.744223	0.999272	0.827215	**0.055086**	0.163704	**1.232533**

7.4.1 Noise Reconstruction Algorithm Parameters

The interpolation accuracy of the NNI algorithm discussed in section 5.1.3 is illustrated in figure 7.33. For each interpolated sample, the standard NNI approach selected the value from the observations that is closest to the point of interest. The standard NNI interpolation accuracy was always constant, since only two samples were considered, in other words, the data point just before the interpolation gap and the point just after the gap. The other two approaches depicted in figure 7.33 used the mean and median of the kNN respectively. The mean and median approaches followed a smooth interpolation sample sequence for large gaps. However, the standard NNI approach performed notably better for smaller gaps, which on the other hand resulted in overall lower NRMSE than

the mean and median techniques.

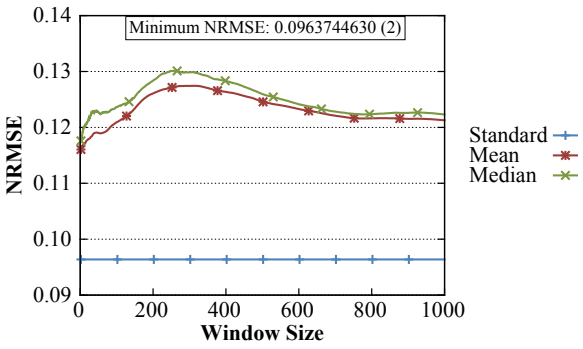


Figure 7.33: The interpolation accuracy of the NNI for different window sizes.

The SI algorithm from section 5.1.4 searches for a sequence of n samples in a larger window that are similar to the samples preceding and succeeding the gap to be interpolated. Figure 7.34 illustrates the parameter space of the SI with the best performance achieved by searching for 182 similar sequential points in a window consisting of 284 samples.

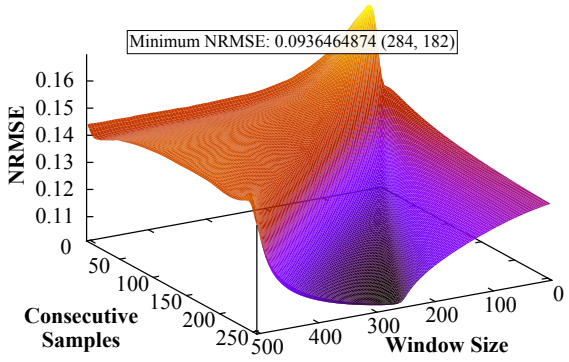


Figure 7.34: The interpolation accuracy of SI for different window sizes.

The LI is depicted in figure 7.35 and achieved the best interpolation with a window of 28 samples. The Lanczos kernel in equation (5.15) evaluates to zero for a small n, since x falls outside the interval $[-n, n]$. The zero Lancos kernel, therefore, had a low interpolation accuracy for windows smaller than 28 samples. Similarly, the accuracy also decreased with larger window sizes. A larger window resulted in a zero Lancozs kernel, since an increased number of points in equation (5.14) causes $x - i$ to exceed n.

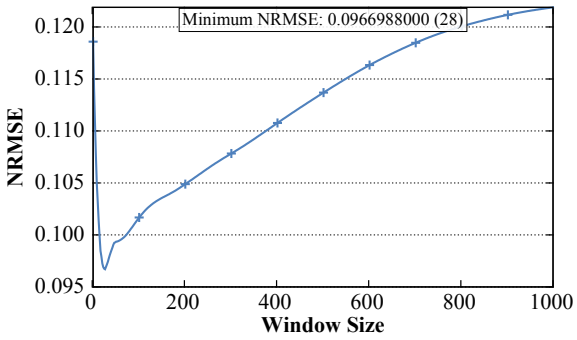


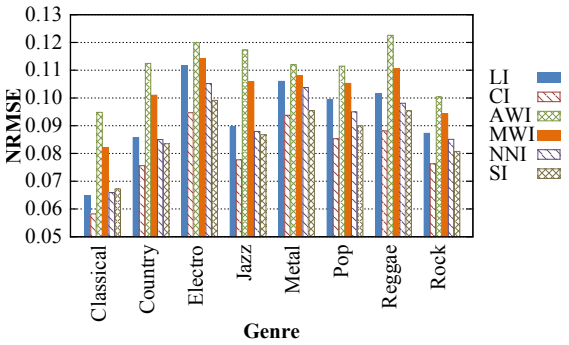
Figure 7.35: The interpolation accuracy of LI for different window sizes.

The full list of optimal parameters for all interpolation algorithms is given in appendix B in table B.2.

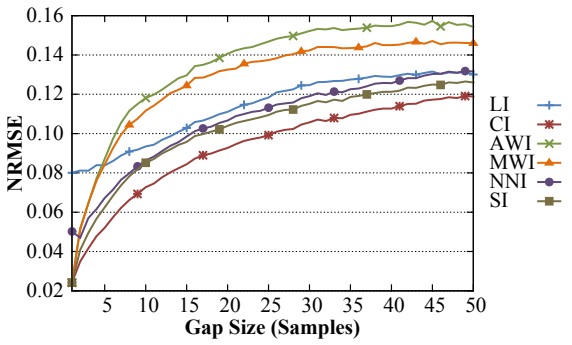
7.4.2 Noise Reconstruction Algorithm Comparison

The reconstruction performance of the duplication and trigonometric interpolation algorithms for different genres is given in figure 7.36(a). CI performed best, whereas AWI had the highest NRMSE over all genres. The corresponding results over an increasing gap size is given in figure 7.36(b). CI achieved the most accurate interpolation over all durations. AWI and MWI performed the worst overall, only outperforming LI for gaps of five samples and smaller.

Figure 7.37(a) shows the interpolation accuracy of the examined polynomials over different genres. The accuracy of standard polynomials decreased when applied in an osculating fashion. Amongst various spline interpolation orders, linear SPS performed best, therefore making the results equivalent to the STP. In contrast, Fourier polynomials benefited when derivatives were included. Although OFP only had minor improvements, Fourier splines had a substantial reduction in the NRMSE, making it the best performing polynomial in classical music. Figure 7.37(b) shows the results of the polynomials for different gap durations. All polynomials had a similar interpolation trend, except FOP and OFP which showed an opposite trend where smaller gaps were more difficult to interpolate than larger ones. When estimated over small gaps, low degree Fourier polynomials had a high frequency which caused the interpolated signal's amplitude to



(a) The interpolation accuracy for different genres.

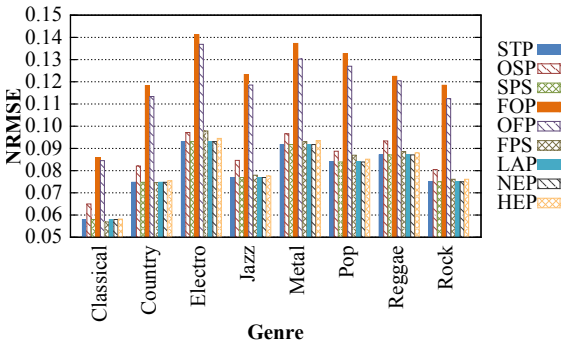


(b) The interpolation accuracy for different gap sizes.

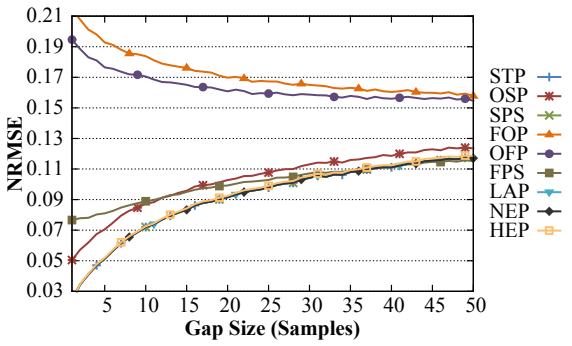
Figure 7.36: The interpolation accuracy of the duplication and trigonometric interpolation algorithms for different genres and gap sizes.

oscillate between the last sample before and first sample after the gap. As the gap size increased, the frequency of the sine and cosine waves decreased, providing a smoother interpolation.

The performance of the time series models for different genres is given in figure 7.38(a). The ARCH and GARCH models had the highest NRMSE, followed by the MA model. The ARMA model had the best reconstruction, although the improvement over the AR model were statistically insignificant. Figure 7.38(b) shows similar reconstruction trends



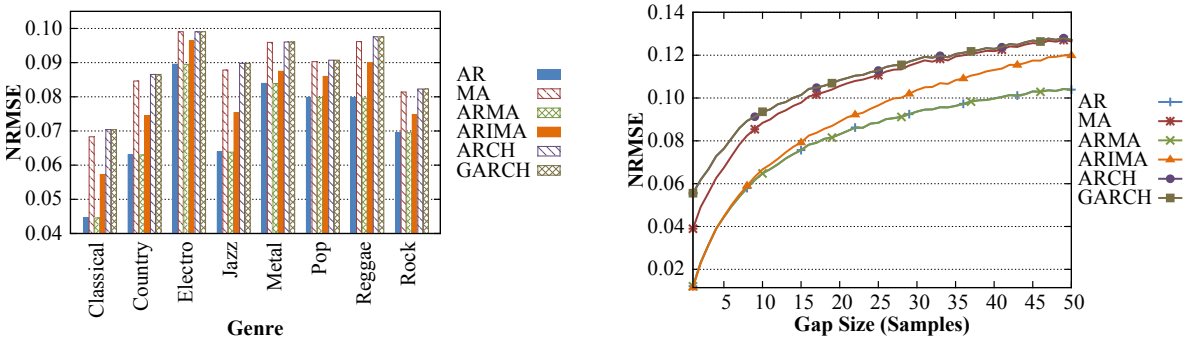
(a) The interpolation accuracy for different genres.



(b) The interpolation accuracy for different gap sizes.

Figure 7.37: The interpolation accuracy of the polynomials for different genres and gap sizes.

for an increasing gap size. The ARIMA model's accuracy was close to that of the AR and ARMA models for durations shorter than ten samples, but started to diverge thereafter. The MA performed better than the heteroskedasticity models, but their interpolation accuracies converged as the gap size increased.



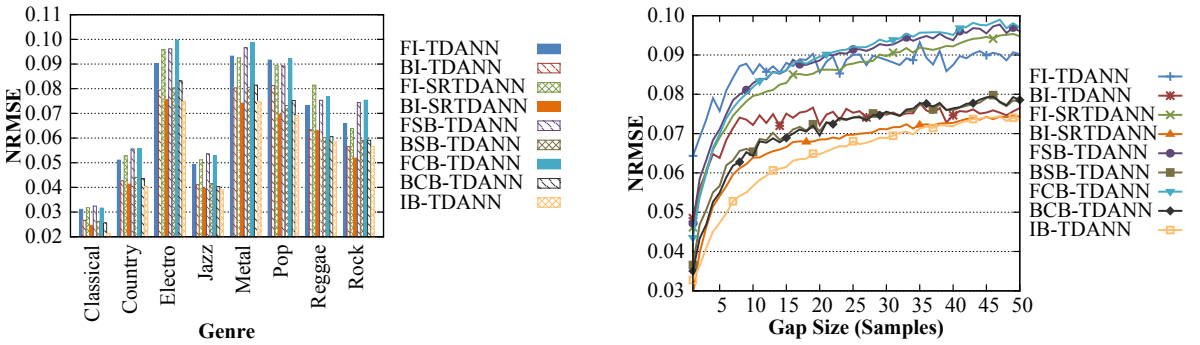
(a) The interpolation accuracy for different genres.

(b) The interpolation accuracy for different gap sizes.

Figure 7.38: The interpolation accuracy of the time series models for different genres and gap sizes.

The ANN reconstruction performance for the examined genres is given in figure 7.39(a). Classical music was almost perfectly refurbished with all ANNs achieving a NRMSE below 0.036. The IB-TDANN performed best over all genres, except for metal and rock music which had a better reconstruction with the BI-SRTDANN. The interpolation accuracy for an increasing gap size is given in figure 7.39(b). The ANNs followed a similar trend, with the BI-SRTDANN performing best over gaps of 44, 46, and 47 samples and the IB-TDANN having the best reconstruction accuracy for all other gap sizes. However, compared to the other interpolation algorithms, the autoregressive models were able to reconstruct gaps smaller than seven samples more accurately. More specifically, the ARIMA model had the lowest NRMSE for durations of up to four samples. The ARMA model performed best for gap sizes of six and seven samples, whereas the BI-SRTDANN and the IB-TDANN had the lowest NRMSE thereafter.

Table 7.8 lists the overall interpolation accuracy, the NRMSE's standard deviations, the computational speed and tradeoff between the various reconstruction algorithms. The IB-TDANN had the lowest NRMSE over all genres, followed by the BI-SRTDANN



(a) The interpolation accuracy for different genres.

(b) The interpolation accuracy for different gap sizes.

Figure 7.39: The interpolation accuracy the ANNs for different genres and gap sizes.

and BSB-TDANN. The AR model had the lowest sample standard deviation, indicating a more consistent interpolation accuracy over all songs in the dataset. Although the FI-SRTDANN had an acceptable NRMSE, some tracks were notably better refurbished than others which is illustrated by the highest standard deviation amongst all algorithms. The CI had the fastest computational speed, whereas the AR model was most efficient by achieving the best tradeoff between the interpolation error and the execution speed. The ANNs took considerably longer than the other algorithms to process the signals and were therefore not able to execute in real-time. However, the computational speed is not that crucial if a gramophone collection is digitized, since the interpolation accuracy is more important and the reconstruction process has to be executed only once. The computational speed of the SRTDANNs was much lower than the other ANNs, since they only had one output neuron, were trained incrementally and updating the weights was therefore much faster.

A detailed report on the reconstruction accuracy for different gap sizes, the interpolation performance over the examined genres and the standard deviation of the NRMSE is given in appendix D.

Table 7.8: The reconstruction accuracy, sample standard deviation, computational time and tradeoff of the interpolation algorithms.

Algorithm	NRMSE	NRMSE σ	Speed (s\s)	SAT
AWI	0.111371	0.027830	0.049794	0.593457
MWI	0.104806	0.025612	0.051691	0.628663
NNI	0.090748	0.027156	0.027313	0.735624
SI	0.087269	0.024434	0.054413	0.748390
LI	0.093215	0.026271	0.027908	0.716238
CI	0.081218	0.025435	**0.027128**	0.820088
STP	0.080057	0.024845	0.031490	0.828606
OSP	0.086014	0.025207	0.034523	0.770803
SPS	0.080057	0.024845	0.038588	0.823627
FOP	0.122412	0.031732	0.068523	0.535829
OFP	0.117923	0.029610	5.325502	0.138812
FPS	0.081493	0.023598	0.033035	0.813341
NEP	0.080058	0.024845	0.027329	0.831552
LAP	0.080057	0.024845	0.031609	0.828522
HEP	0.081066	0.025381	0.027557	0.821287
AR	0.071764	**0.023160**	0.092778	**0.870951**
MA	0.087952	0.024475	0.029541	0.757201
ARMA	0.071709	0.023178	2.435243	0.281283
ARIMA	0.080201	0.023740	6.808781	0.122321
ARCH	0.089057	0.024445	0.062245	0.729670
GARCH	0.089057	0.024445	0.062378	0.729598
FI-TDANN	0.068145	0.029736	64.46232	0.014868
BI-TDANN	0.058917	0.025292	129.0510	0.007490
FI-SRTDANN	0.069853	0.033467	2.663952	0.265745
BI-SRTDANN	0.054953	0.025194	5.237380	0.161561
FSB-TDANN	0.071659	0.029915	52.02565	0.018339
BSB-TDANN	0.058147	0.023480	104.2376	0.009259
FCB-TDANN	0.072788	0.032014	89.62463	0.010731
BCB-TDANN	0.058637	0.024168	179.7895	0.005386
IB-TDANN	**0.054247**	0.025009	100.0583	0.009648

7.5 Gramophone Analysis

The results in section 7.3 were obtained by distorting music with artificial noise. The models and algorithms were also tested on real distorted gramophone recordings. The outlier detection comparison of the results between the artificially generated and real gramophone noise is given in table 7.9. The two values between brackets in the second column of table 7.9 are the standard deviations from table 7.7 subtracted from and added to the detection MCC respectively. All algorithms performed slightly worse when tested on real gramophone recordings. However, the difference between the detection accuracy using artificial and real gramophone noise is statistically insignificant, since the gramophone's MCC falls within the standard deviation range given in the second column. The artificially generated noise discussed in section 6.1 is therefore considered a good estimation of real gramophone distortions.

Table 7.9: The performance comparison of the outlier detection algorithms between artificially generated and real gramophone noise.

Algorithm	Artificial Noise (MCC)	Gramophone Noise (MCC)
SS	0.792384 [0.620, 0.965]	0.744230
MAD	0.740720 [0.546, 0.935]	0.710120
MHD	0.798867 [0.631, 0.967]	0.754100
NND	0.753863 [0.580, 0.928]	0.710010
MASD	0.558229 [0.474, 0.642]	0.500120
APD-ARIMA	0.836681 [0.729, 0.944]	0.797480

Only the outlier detection results are influenced differently by artificial and real gramophone noise. The interpolation process on the other hand is unaffected by different types of noise, since outliers are completely discarded and treated as a gap of missing samples.

7.6 Summary

This chapter presented the empirical results of the outlier detection and interpolation algorithms. The polynomials, time series models and ANNs were analysed according to

their interpolation and prediction abilities. The optimal parameter configuration for all algorithms was discussed. The outlier detection processes were compared according to their SEN, SPE and MCC. The signal reconstruction was evaluated using the NRMSE. In addition, the execution speed of the algorithms was measured and a tradeoff between the accuracy and speed was calculated in order to rank the algorithms according to efficiency. The noise detection was also benchmarked using artificially generated and real gramophone noise.

Chapter 8

Conclusions

This thesis investigated and empirically analysed a number of algorithms and models to detect and reconstruct noise in gramophone recordings. This chapter presents the conclusion to the thesis by highlighting the key findings and best performing algorithms and models used to detect and reconstruct the noise. Future work and potential extensions to the research is provided in the last section.

8.1 Summary

Various polynomials and models were investigated in chapter 3 in order to determine their ability to accurately model music signals. Once the coefficients were approximated, the models were used to interpolate a gap of missing samples or predict future samples. The examined polynomials include the standard, Fourier, Hermite, Newton, and Lagrange polynomials. The standard and Fourier polynomials were also applied in an osculating fashion and by making use of splines in order to determine how the inclusion of derivatives will affect the model accuracy. The more advanced autoregressive, moving average, and their combinations, the ARMA and ARIMA models, were also tested. The heteroskedasticity models, ARCH and GARCH, and a variety of artificial neural networks also formed part of the comparative study.

Chapter 4 inspected a number of algorithms that are typically used in statistics and DSP to detect outliers in signals or sample sequences. The outlier detectors include

the standard score, median absolute deviation, Mahalanobis, nearest neighbour and the absolute mean spectral deviation algorithm. The models from chapter 3 were used in the absolute predictive deviation algorithm and compared to the other outlier detectors.

Chapter 5 utilized the models from chapter 3 to reconstruct the samples that were previously flagged as noise by the outlier detectors. Besides the models, duplication techniques were tested, including adjacent window, mirroring window, nearest neighbour and similarity interpolation. Two trigonometric methods, Lanczos and cosine interpolation, determined the affect of sine and cosine waves on the reconstruction process.

The parameters of the models and algorithms were optimized using fractional factorial design. A test set of 800 songs covering eight major genres was used to benchmark the noise detection and reconstruction accuracy. Both artificially generated and real gramophone noise was used during benchmarking. The noise detection algorithms were compared using their sensitivity, specificity and the Matthews correlation coefficient. The interpolation performance was measured using the normalized root mean squared error. The algorithms' computational time was evaluated and the tradeoff between the execution time and the accuracy determined the most efficient algorithm.

It was found that APD-ARIMA had the best noise detection accuracy with a MCC of 0.8367. The NND and MHD reached the highest sensitivity and specificity at 0.8248 and 0.9998 respectively. The APD-TDANN had the lowest execution time at 0.1420 s\s and the APD-SRTDANN achieved the best tradeoff between the detection accuracy and computational time with a SAT score of 1.2325.

During reconstruction, the IB-TDANN interpolated most accurately with a NRMSE of 0.0542. The ANNs performed overall better than the polynomials, time series models, duplication and trigonometric approaches. CI was the fastest to compute at 0.0271 s\s. The AR model achieved the best tradoff between the interpolation NRMSE and time with a SAT score of 0.8710.

8.2 Future Work

Although this research encapsulated a large number of different polynomials, models and algorithms to detect and reconstruct noise from gramophone records, there are still

numerous areas that require additional and more in-depth research. Potential future research and extensions to this study are listed below:

- This study only investigated a few outlier detection algorithms. Although some of the algorithms have a good detection accuracy, additional research in more advanced outlier detection algorithms, such as spectral methods, has to be conducted.
- This thesis only investigated the reconstruction of major disruptions caused by scratches. Smaller disruptions and deviations cannot be eliminated with the algorithms examined here. A different set of mathematical procedures, such as smoothing filters, can be utilized to reduce these kind of distortions.
- Although multi-channel audio data can be reconstructed with the system proposed in this study, the reconstruction process relies solely on the samples of a single channel at a time. An extension to the examined models, especially artificial neural networks might benefit from incorporating and combining data from multiple channels.
- This thesis focused on the reconstruction of music from gramophone records. However, the research can be directly applied to music and speech from any sources. The research can also be applied to other audio mediums, such as compact cassettes, eight-tracks or even deteriorated CDs. The examined algorithms may also prove beneficial in other signal processing domains, such as online media streaming, voice over IP, digital radio transmission and video and audio conversion, resampling and refurbishment.
- Scratches on gramophone records typically disrupt less than 50 samples. If the studied interpolators are applied to other fields of audio and speech processing, gap sizes larger than 50 samples may also need to be probed. The characteristics of the noise may also be significantly different compared to gramophone noise and should be considered in future studies that do not focus on gramophone audio.
- Only the AR, MA, ARMA, ARIMA, ARCH and GARCH models were considered in this study. Numerous extensions and improvements to these models exist, which might prove beneficial to music data.

- A total of 800 songs in eight different genres were used during benchmarking. Clear divergence between different genres and even different songs in the same genre were observed. Future research is needed to investigate which characteristics in music and speech influences the reconstruction process. These characteristics can then be automatically identified and classified so that the corresponding reconstruction algorithm can be adjusted to better accommodate these features.

- The optimal model and algorithm parameters over all genres were used in this study, ensuring that an acceptable noise detection and reconstruction accuracy was achieved for different types of music signals. A more accurate system can be developed that makes use of separate parameter configurations to optimize the reconstruction of each individual genre.

- An in-depth perceptual evaluation of the audio restoration process should be conducted. Some subjective listening has been done with a small group of participants, but a more formal and elaborate analysis with a larger test group will provide better and more accurate feedback.

Bibliography

[1] A.M. Aibinu, M.J.E. Salami, and A.A. Shafie. Artificial Neural Network Based Autoregressive Modeling Technique with Application in Voice Activity Detection. *Journal of Engineering Applications of Artificial Intelligence*, 25(6):1265–1276, 2012.

[2] Hirotugu Akaike. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974.

[3] Ali Hussein Al-Marshadi. Improving the Order Selection of Moving Average Time Series Model. *African Journal of Mathematics and Computer Science Research*, 5(5):102–106, 2012.

[4] Khaled E. Al-Qawasmi, Adnan M. Al-Smadi, and Alaa Al-Hamami. Artificial Neural Network-Based Algorithm for ARMA Model Order Estimation. In *Networked Digital Technologies*, volume 88 of *Communications in Computer and Information Science*, pages 184–192. Springer, 2010.

[5] John S. Allen. Some New Possibilities in Audio Restoration. *The Association for Recorded Sound Collections Journal*, 21(1):39–44, 1990.

[6] Mennatallah Amer and Markus Goldstein. Nearest-Neighbor and Clustering based Anomaly Detection Algorithms for RapidMiner. In *Proceedings of the RapidMiner Community Meeting and Conferernce*, pages 1–12, Herzogenrath, Germany, August 2012. Shaker Verlag.

[7] Erling Bernhard Andersen. Asymptotic Properties of Conditional Maximum-Likelihood Estimators. *Journal of the Royal Statistical Society*, 32(2):283–301, 1970.

[8] E. Anderson, Z. Bai, and J. Dongarra. Generalized QR Factorization and its Applications. *Linear Algebra and its Applications*, 162–164:243–271, 1992.

[9] Fabrizio Angiulli and Clara Pizzuti. Fast Outlier Detection in High Dimensional Spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 15–26, London, United Kingdom, 2002.

[10] Apple. Apple Lossless Audio Codec. http://alac.macosforge.org/, October 2011. Accessed: 2014-10-08.

[11] Josip Arnerić and Rozga A. Numerical Optimization within Vector of Parameters Estimation in Volatility Models. In *Proceedings of World Academy of Science, Engineering and Technolog*, volume 37, pages 632–636, January 2009.

[12] Josip Arnerić, Zoran Babić, and Blanka Škrabić. Maximization of the Likelihood Function in Financial Time Series Models. In *Proceedings of the International Scientific Conference Contemporary Challenges of Theory and Practice in Economics*, pages 1–12, Belgrade, Serbia, 2007.

[13] Zahid Asghar and Irum Abid. Performance of lag length selection criteria in three different situations. Technical report, MPRA, University Library of Munich, Germany, 2007.

[14] Jacques Attali. *Noise: The Political Economy of Music*. Theory and History of Literature. Manchester University Press, Manchester, United Kingdom, 1985.

[15] Douglas M. Bates and Donald G. Watts. *Nonlinear Regression Analysis and its Applications*. Wiley Series in Probability and Mathematical Statistics. Wiley, New York City, New York, United States of America, 1988.

[16] Klaus Jürgen Bathe and Arthur P. Cimento. Some Practical Procedures for the Solution of Nonlinear Finite Element Equations. *Computer Methods in Applied Mechanics and Engineering*, 22(1):59–85, 1980.

[17] James R. Bell. Algorithm 334: Normal Random Deviates. *Communications of the ACM*, 11(7):498–498, July 1968.

[18] Commandant Benoit. Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés à un système d'équations linéaires en nombre inférieur à celui des inconnues. Application de la méthode à la résolution d'un système défini d'équations linéaires. *Bulletin Géodésique*, 2(1):67–77, 1924.

[19] E. K. Berndt, B. H. Hall, R. E. Hall, and Jerry A. Hausman. Estimation and Inference in Nonlinear Structural Models. In *Annals of Economic and Social Measurement*, volume 3, pages 653–665. National Bureau of Economic Research, Inc, January 1974.

[20] Sergei N. Bernstein. Sur l'ordre de la meilleure approximation des fonctions continues par les polynômes de degré donné. *Mémoires de l'Académie Royale de Belgique*, 4:1–103, 1912.

[21] Kanishka Bhaduri and Bryan L. Matthews. Algorithms for Speeding up Distance-based Outlier Detection. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 859–867, San Diego, California, United States of America, August 2011.

[22] Nicolò Cesa Bianchi, Philip M. Long, and Manfred K. Warmuth. Worst-case Quadratic Loss Bounds for a Generalization of the Widrow-Hoff Rule. In *Proceedings of Computational Learning Theory*, pages 429–438. ACM, 1993.

[23] Garrett Birkhoff and Carl R. de Boor. *Piecewise Polynomial Interpolation and Approximation*. Approximation of Functions. Elsevier Applied Science, Amsterdam, Netherlands, 1965.

[24] Arne Bjerhammar. Application of Calculus of Matrices to Method of Least Squares with Special Reference to Geodetic Calculations. *Transactions of the Royal Institute of Technology Stockholm*, 49:1–86, 1951.

[25] Edward K. Blum and Leong Kwan Li. Approximation Theory and Feedforward Networks. *Neural Networks*, 4(4):511–515, 1991.

[26] Lorant Bodis. *Quantification of Spectral Similarity: Towards Automatic Spectra Verification*. PhD thesis, Eidgenössische Technische Hochschule Zürich, Zürich, Switzerland, 2007.

[27] Tim Bollerslev. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

[28] Robert Bos, Stijn de Waele, and Piet M. T. Broersen. Autoregressive Spectral Estimation by Application of the Burg Algorithm to Irregularly Sampled Data. *IEEE Transactions on Instrumentation and Measurement*, 51(6):1289–1294, December 2002.

[29] George Edward Pelham Box and Gwilym Meirion Jenkins. *Time Series Analysis: Forecasting and Control*. Holden-Day Series in Time Series Analysis. Holden-Day, San Francisco, California, United States of America, 1970.

[30] George Edward Pelham Box and Mervin Edgar Muller. A Note on the Generation of Random Normal Deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, June 1958.

[31] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. OPTICS-OF: Identifying Local Outliers. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*, pages 262–270, Prague, Czech Republic, September 1999. Springer.

[32] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. LOF: Identifying Density-based Local Outliers. *ACM SIGMOD Record*, 29(2):93–104, May 2000.

[33] Piet M. T. Broersen. The Quality of Lagged Products and Autoregressive Yule-Walker Models as Autocorrelation Estimates. *IEEE Transactions on Instrumentation and Measurement*, 58(11):3867–3873, 2009.

[34] Charles George Broyden. The Convergence of a Class of Double-rank Minimization Algorithms. *IMA Journal of Applied Mathematics*, 6(1):76–90, March 1970.

[35] Elodie Brunel and Fabienne Comte. Cumulative Distribution Function Estimation under Interval Censoring Case 1. *Electronic Journal of Statistics*, 3:1–24, 2009.

[36] John Parker Burg. A New Analysis Technique for Time Series Data. In Donald G. Childers, editor, *Modern Spectrum Analysis, NATO Advanced Study Institute of Signal Processing with emphasis on Underwater Acoustics*. IEEE Press, New York City, New York, United States of America, 1968.

[37] Kenneth P. Burnham and David R. Anderson. Multimodel Inference: Understanding AIC and BIC in Model Selection. *Sociological Methods and Research*, 33(2):261–304, 2004.

[38] Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. Representations of quasi-Newton Matrices and Their Use in Limited Memory Methods. *Mathematical Programming*, 63(2):129–156, January 1994.

[39] Olivier Cappé. Elimination of the Musical Noise Phenomenon with the Ephraim and Malah Noise Suppressor. *IEEE Transactions on Speech and Audio Processing*, 2(2):345–349, 1994.

[40] Lennart Carleson. On Convergence and Growth of Partial Sums of Fourier Series. *Acta Mathematica*, 116(1):135–157, 1966.

[41] Deepthi Cheboli. Anomaly Detection of Time Series. Master's thesis, University of Minnesota, Minneapolis, Minnesota, United States of America, May 2010.

[42] Ward Cheney and David Kincaid. *Numerical Mathematics and Computing*. International Thomson Publishing, Bethesda, Maryland, United States of America, 4 edition, 1998.

[43] Jaerin Cho. *Optimal Design in Regression and Spline Smoothing*. PhD thesis, Queen's University, Kingston, Ontario, Canada, July 2007.

[44] Krzysztof Cisowski. *Application of Adaptive Filtering to Reconstruction of Audio Signals*. PhD thesis, Gdańsk University of Technology, Gdańsk, Poland, 2000.

[45] Gerda Claeskens and Nils L. Hjort. The Focused Information Criterion. *Journal of the American Statistical Association*, 98(464):900–945, 2003.

[46] James Cooley and John Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90):297–301, 1965.

[47] Andrzej Czyzewski. Some Methods for Detection and Interpolation of Impulsive Distortions in Old Audio Recordings. In *IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 139–142, October 1995.

[48] William Cooper Davidon. *Variable Metric Method for Minimization*. AEC Research and Development Report. Argonne National Laboratory, Lemont, Illinois, United States of America, 1959.

[49] Russell Davidson and James G. MacKinnon. *Estimation and Inference in Econometrics*. Oxford University Press, New York City, New York, United States of America, 1993.

[50] M. J. L. de Hoon, T. H. J. J. van der Hagen, H. Schoonewelle, and H. van Dam. Why yule-walker should not be used for autoregressive modelling. *Annals of Nuclear Energy*, 23(15):1219–1228, 1996.

[51] Mehdi Dehghan and Masoud Hajarian. Determination of a Matrix Function Using the Divided Difference Method of Newton and the Interpolation Technique of Hermite. *Journal of Computational and Applied Mathematics*, 231:67–81, 2009.

[52] Antonis Demos and Enrique Sentana. Testing for GARCH Effects: A One-sided Approach. *Journal of Econometrics*, 86(1):97–127, 1998.

[53] John E. Dennis and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1996.

[54] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer, New York City, New York, United States of America, 1 edition, 1986.

[55] Mingzhou Ding, Steven L. Bressler, Weiming Yang, and Hualou Liang. Short-window Spectral Analysis of Cortical Event-related Potentials by Adaptive Multivariate Autoregressive Modeling: Data Preprocessing, Model Validation, and Variability Assessment. *Biological Cybernetics*, 83(1):35–45, 2000.

[56] Gerhard Doblinger. Smoothing of Noisy AR Signals Using an Adaptive Kalman Filter. In *European Signal Processing Conference*, volume 2, pages 781–784, Island of Rhodes, Greece, September 1998.

[57] Clif Droke. *Moving Averages Simplified*. Marketplace Books, 1 edition, 2001.

[58] K. L. Du. Clustering: A Neural Network Approach. *Neural Networks*, 23(1):89–107, 2010.

[59] Claude E. Duchon. Lanczos Filtering in One and Two Dimensions. *Journal of Applied Meteorology*, 18(8):1016–1022, 1979.

[60] James Durbin. The Fitting of Time-series Models. *Review of the International Statistical Institute*, 28(3):233–243, 1960.

[61] R. Durbin and D. E. Rumelhart. Product Units: A Computationally Powerful and Biologically Plausible Extension to Backpropagation Networks. *Neural Computation*, 1:133–142, 1989.

[62] Francis Ysidro Edgeworth. On the Probable Errors of Frequency-Constants. *Journal of the Royal Statistical Society*, 71(3):651–678, 1908.

[63] Theo J. H. M. Eggen. On the loss of information in conditional maximum likelihood estimation of item parameters. *Psychometrika*, 65(3):337–362, 2000.

[64] Joakim Ekström. Mahalanobis' Distance Beyond Normal Distributions. In *UCLA Department of Statistics*, Los Angeles, California, United States of America, September 2011. University of California.

[65] David L. Elliot. A Better Activation Function for Artificial Neural Networks. Technical report, Institute for Systems Research, 1993. T.R. 93-8.

[66] Jeffrey L. Elman. Distributed Representations, Simple Recurrent Networks, and Grammatical Structure. *Machine Learning*, 7(2-3):195–225, 1991.

[67] Andries P. Engelbrecht. A New Pruning Heuristic Based on Variance Analysis of Sensitivity Information. *Transactions on Neural Networks*, 12(6):1386–1399, 2001.

[68] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, 2 edition, 2007.

[69] Robert Fry Engle. Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50(4):987–1007, July 1982.

[70] Robert Fry Engle. Wald, Likelihood Ratio, and Lagrange Multiplier Tests in Econometrics. In *Handbook of Econometrics*, volume 2 of *Handbook of Econometrics*, chapter 13, pages 775–826. Elsevier, 1984.

[71] Paulo A. A. Esquef, Luiz W. P. Biscainho, Vesa Välimäki, and Matti Karjalainen. Removal of Long Pulses from Audio Signals Using Two-pass Split-window Filtering. In *Audio Engineering Society Convention*, Munich, Germany, May 2002.

[72] Paulo A. A. Esquef, Matti Karjalainen, and Vesa Välimäki. Detection of Clicks in Audio Signals Using Warped Linear Prediction. In *International Conference on Digital Signal Processing*, volume 2, pages 1085–1088, Santorini, Greece, July 2002.

[73] Paulo A. A. Esquef and Guilherme S. Welter. Audio De-thumping using Huang's Emperical Mode Decomposition. In *International Conference on Digital Audio Effects*, volume 11, pages 401–408, Paris, France, 2011.

[74] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, United States of America, August 1996. AAAI Press.

[75] Leonhard Euler. De Eximio usu Methodi Interpolationum in Serierum Doctrina. *Academia Imperialis Scientiarum*, 1:157–210, 1783.

[76] Vitaliy Fadeyev and Carl Haber. Reconstruction of Mechanically Recorded Sound by Image Processing. *Journal of the Audio Engineering Society*, 51(12):1172–1185, 2003.

[77] Vitaliy Fadeyev, Carl Haber, Christian Maul, John W. Mcbride, and Mitchell Golden. Reconstruction of Recorded Sound from an Edison Cylinder Using Three-Dimensional Noncontact Optical Surface Metrology. *Journal of the Audio Engineering Society*, 53(6):485–508, 2005.

[78] Scott E. Fahlman. Faster-Learning Variations on Back-Propagation: An Empirical Study. In *Proceedings of the Connectionist Models Summer School*, pages 38–51. Morgan-Kaufmann, 1988.

[79] Scott E. Fahlman and Christian Lebiere. The Cascade-Correlation Learning Architecture. In *Advances in Neural Information Processing Systems 2*, pages 524–532. Morgan Kaufmann, 1990.

[80] Patrick Feaster. The Phonautographic Manuscripts of Édouard-Léon Scott de Martinville. http://www.firstsounds.org/publications/articles/Phonautographic-Manuscripts.pdf, December 2009. Accessed: 2014-04-16.

[81] Ernst Fischer. Sur la convergence en moyenne. In *Comptes rendus de l'Académie des sciences*, volume 144, pages 1022–1024, 1907.

[82] Ronald Aylmer Fisher. On an Absolute Criterion for Fitting Frequency Curves. *Messenger of Mathematics*, 41:155–160, 1912.

[83] Ronald Aylmer Fisher. Frequency Distribution of the Values of the Correlation Coefficient in Samples from an Indefinitely Large Population. *Biometrika*, 10(4):507–521, 1915.

[84] Ronald Aylmer Fisher. A Mathematical Examination of the Methods of Determining the Accuracy of an Observation by the Mean Error, and by the Mean Square Error. *Monthly Notices Roy. Astronom. Soc.*, 80:758–770, 1920.

[85] Ronald Aylmer Fisher. On the Probable Error of a Coefficient of Correlation Deduced from a Small Sample. *Metron*, 1:3–32, 1921.

[86] Ronald Aylmer Fisher. The Goodness of Fit of Regression Formulae, and the Distribution of Regression Coefficients. *Journal of the Royal Statistical Society*, 85:597–612, 1922.

[87] Ronald Aylmer Fisher. *The Design of Experiments*. Oliver and Boyd, Edinburgh, United Kingdom, 1935.

[88] Roger Fletcher. A New Approach to Variable Metric Algorithms. *The Computer Journal*, 13(3):317–322, March 1970.

[89] Roger Fletcher and Michael James David Powell. A Rapidly Convergent Descent Method for Minimization. *The Computer Journal*, 6(2):163–168, 1963.

[90] Bengt Fornberg, Tobin A. Driscoli, Grady Wright, and Richard Charles. Observations on the Behavior of Radial Basis Function Approximations near Boundaries. *Computers & Mathematics with Applications*, 43(3):473–490, February 2002.

[91] Bengt Fornberg and Julia Zuev. The Runge Phenomenon and Spatially Variable Shape Parameters in RBF Interpolation. *Computers & Mathematics with Applications*, 54(3):379–398, August 2007.

[92] Xiph Foundation. The Ogg container format. http://www.xiph.org/ogg/, 2013. Accessed: 2014-10-08.

[93] Xiph Foundation. Free Lossless Audio Codec. https://xiph.org/flac/, July 2014. Accessed: 2014-10-08.

[94] Jean Baptiste Joseph Fourier. *Théorie Analytique De La Chaleur*. 1822.

[95] Joseph J.B. Fourier. Mémoire sur la propagation de la chaleur dans les corps solides. In *Nouveau Bulletin des sciences par la Société philomatique de Paris*, volume 6, pages 112–116, Paris, France, December 1807.

[96] Brian L. Fox. Analysis and Dynamic Range Enhancment of the Analog-to-digital Interface in Multimode Radio Receivers. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, United States of America, February 1997.

[97] John G. F. Francis. The QR Transformation A Unitary Analogue to the LR Transformation - Part 1. *The Computer Journal*, 4(3):265–271, 1961.

[98] John G. F. Francis. The QR Transformation - Part 2. *The Computer Journal*, 4(4):332–345, 1962.

[99] Christian Francq and Jean-Michel Zakoian. *GARCH Models: Structure, Statistical Inference and Financial Applications*. Wiley, Hoboken, New York, United States of America, 2010.

[100] Bruce Fries and Marty Fries. *Digital Audio Essentials*. O'Reilly Series. O'Reilly, 2005.

[101] Francis Galton. Regression Towards Mediocrity in Hereditary Stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263, 1886.

[102] Carl Friedrich Gauss. *Theoria Motus Corporum Coelestium In Sectionibus Conicis Solem Ambientium*. 1809.

[103] Carl Friedrich Gauss. *Theoria Combinationis Observationum Erroribus Minimis Obnoxiae*. 1823.

[104] Hongwei Ge, Wenli Du, Feng Qian, and Yanchun Liang. A Novel Time-delay Recurrent Neural Network and Application for Identifying and Controlling Nonlinear Systems. In *International Conference on Natural Computation*, volume 1, pages 44–48, 2007.

[105] William B. Gearhart and Harris S. Shultz. The Function sin x/x. *The College Mathematics Journal*, 21(2):90–99, March 1990.

[106] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[107] James E. Gentle. *Random Number Generation and Monte Carlo Methods*. Springer, 2 edition, 2004. pp. 103.

[108] C. Lee Giles, Steve Lawrence, and Ah Chung Tsoi. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine Learning*, 44(1-2):161–183, 2001.

[109] Jeff Gill and Gary King. What to do When Your Hessian is Not Invertible: Alternatives to Model Respecification in Nonlinear Estimation. *Sociological Methods and Research*, 32(1):54–87, August 2004.

[110] Geof H. Givens and Jennifer A. Hoeting. *Computational Statistics*. Wiley Series in Computational Statistics. Wiley, 2 edition, 2012. pp. 155.

[111] GMP. The GNU Multiple Precision Arithmetic Library. https://gmplib.org, 2014. Accessed: 2014-10-14.

[112] S. J. Godsill and P. J. W. Rayner. Frequency-based interpolation of sampled signals with applications in audio restoration. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 209–212, April 1993.

[113] S. J. Godsill and P. J. W. Rayner. The Restoration of Pitch Variation Defects in Gramophone Recordings. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 148–151, October 1993.

[114] Simon J. Godsill and Peter J.W. Rayner. *Digital Audio Restoration - A Statistical Model-Based Approach*. Springer, London, United Kingdom, 1 edition, 1998.

[115] Donald Goldfarb. A Family of Variable-Metric Methods Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26, January 1970.

[116] G.H. Golub and C. Reinsch. Singular Value Decomposition and Least Squares Solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[117] B. Grill and S. Quackenbush. MPEG-2 Audio. http://mpeg.chiariglione.org/standards/mpeg-2/audio, October 2005. Accessed: 2014-10-08.

[118] Venu G. Gudise and Ganesh K. Venayagamoorthy. Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 110–117, 2003.

[119] E. J. Hannan and B. G. Quinn. The Determination of the Order of an Autoregression. *Journal of the Royal Statistical Society*, 41(2):190–195, 1979.

[120] W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[121] M. C. Hau and H. Tong. A Practical Method for Outlier Detection in Autoregressive Time Series Modelling. *Stochastic Hydrology and Hydraulics*, 3(4):241–260, 1989.

[122] Ville Hautamäki, Ismo Kärkkäinen, and Pasi Fränti. Outlier Detection Using k-Nearest Neighbour Graph. In *International Conference on Pattern Recognition*, volume 3, pages 430–433, Washington, United States of America, 2004. IEEE Computer Society.

[123] William K. Heine. A Laser Scanning Phonograph Record Player. In *Audio Engineering Society Convention 57*, May 1977.

[124] Philipp Hennig. Fast Probabilistic Optimization from Noisy Gradients. In *Proceedings of the International Conference on Machine Learning*, volume 28, pages 62–70, 2013.

[125] Charles Hermite. Sur la Formule d'Interpolation de Lagrange. *Journal für die Reine und Angewandte Mathematik*, 84(1):70–79, 1878.

[126] C.C. Heyde. On the Central Limit Theorem for Stationary Processes. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 30(4):315–320, 1974.

[127] Marko Hiipakka. Measurement Apparatus and Modelling Techniques of Ear Canal Acoustics. Master's thesis, Helsinki University of Technology, Helsinki, Finland, November 2008.

[128] David J. Hill and Barbara S. Minsker. Anomaly Detection in Streaming Environmental Sensor Data: A Data-driven Modeling Approach. *Environmental Modelling and Software*, 25(9):1014–1022, 2010.

[129] Nils L. Hjort and Gerda Claeskens. Frequentist Model Average Estimators. *Journal of the American Statistical Association*, 98(464):879–899, December 2003.

[130] Charles Antony Richard Hoare. Algorithm 65: Find. *Communications of the ACM*, 4(7):321–322, July 1961.

[131] Reginald Hawthorn Hooker. Correlation of the Marriage Rate with Trade. *Journal of the Royal Statistical Society*, 64:485–492, 1901.

[132] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2(5):359–366, 1989.

[133] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Universal Approximation of an Unknown Mapping and its Derivatives using Multilayer Feedforward Networks. *Neural Networks*, 3(5):551–560, 1990.

[134] Jamie Howarth and Patrick J. Wolfe. Correction of Wow and Flutter Effects in Analog Tape Transfers. In *Audio Engineering Society Convetion*, volume 117, San Francisco, Californis, United States of America, 2011.

[135] Wei Hu and Junpeng Bao. The Outlier Interval Detection Algorithms on Astronautical Time Series Data. *Journal of Mathematical Problems in Engineering*, pages 1–6, 2013.

[136] Rob J. Hyndman and Anne B. Koehler. Another Look at Measures of Forecast Accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

[137] Christian Igel and Michael Hüsken. Improving the Rprop Learning Algorithm. In *International Symposium on Neural Computation*, pages 115–121, 2000.

[138] Christian Igel and Michael Hüsken. Empirical Evaluation of the Improved Rprop Learning Algorithm. *Neurocomputing*, 50:105–123, 2003.

[139] Boris Iglewicz and David Caster Hoaglin. *How to Detect and Handle Outliers Front Cover*. ASQ Quality Press, Milwaukee, Wisconsin, United States of America, 1 edition, 1993.

[140] Michael Jachan, Gerald Matz, and Franz Hlawatsch. Least-Squares and Maximum-Likelihood Tfar Parameter Estimation for Nonstationary Processes. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 492–495, Toulouse, France, May 2006.

[141] Dunham Jackson. The Theory of Approximation. *American Mathematical Society Colloquium Publications*, 11:77–255, 1930.

[142] Sanjiv Jaggia. Forecasting with ARMA Models. *Case Studies in Business, Industry, and Government Statistics*, 4(1):59–65, 2010.

[143] Gwilym M. Jenkins and Donald G. Watts. *Spectral Analysis and its Applications*. Holden-Day series in Time Series Analysis. Holden-Day, San Francisco, California, United States of America, 1968.

[144] Johan Ludwig William Valdemar Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(1):175–193, 1906.

[145] Wen Jin, Anthony K. H. Tung, Jiawei Han, and Wei Wang. Ranking Outliers Using Symmetric Neighborhood Relationship. In *Proceedings of the Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 577–593, Singapore, 2006. Springer.

[146] Michael I. Jordan. Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In *Proceedings of the Cognitive Science Society*, pages 531–546, 1986.

[147] Ryan Kastner, Anup Hosangadi, and Farzan Fallah. *Arithmetic Optimization Techniques for Hardware and Software Design*. Cambridge University Press, Cambridge, United Kingdom, June 2010.

[148] J. F. Kenney and E. S. Keeping. *Mathematics of Statistics*. Van Nostrand, 3 edition, 1964.

[149] Sung-Suk Kim. Time-delay Recurrent Neural Network for Temporal Correlations and Prediction. *Neurocomputing*, 20(1-3):253–263, 1998.

[150] D. Kincaid and W. Cheney. *Numerical Analysis*. Approximation of Functions. Brooks Cole, Pacific Grove, California, United States of America, 2 edition, 1996.

[151] R. Knop. Remark on Algorithm 334 [G5]: Normal Random Deviates. *Communications of the ACM*, 12(5):281–281, May 1969.

[152] Teuvo Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps*. Springer Verlag, Secaucus, New Jersey, United States of America, 3 edition, 2001.

[153] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. LoOP: Local Outlier Probabilities. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 1649–1652, New York City, New York, United States of America, 2009. Association for Computing Machinery.

[154] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Interpreting and Unifying Outlier Scores. In *Society for Industrial and Applied Mathematics Conference on Data Mining*, pages 13–24, Mesa, Arizona, United States of America, April 2011. Omnipress.

[155] Vera Nikolaevna Kublanovskaya. On Some Algorithms for the Solution of the Complete Eigenvalue Problem. *USSR Computational Mathematics and Mathematical Physics*, 1(3):637–657, 1961.

[156] Joseph Louis Lagrange. Lecons Elémentaires sur les Mathématiques, Données à l'Ecole Normale en. *Journal de l'école Polytechnique*, 7:183–287, 1795.

[157] Kevin J. Lang, Alex H. Waibel, and Geoffrey E. Hinton. A Time-delay Neural Network Architecture for Isolated Word Recognition. *Neural Networks*, 3(1):23–43, 1990.

[158] Aleksandar Lazarevic and Vipin Kumar. Feature Bagging for Outlier Detection. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 157–166, New York City, New York, United States of America, 2005. Association for Computing Machinery.

[159] Do Q. Lee. Numerically Efficient Methods for Solving Least Squares Problems. University of Chicago REU Research Experiences, May 2012.

[160] John H.H. Lee and Maxwell Leslie King. A Locally Most Mean Powerful Based Score Test for ARCH and GARCH Regression Disturbances. *Journal of Business and Economic Statistics*, 11(1):17–27, 1993.

[161] Norman Levinson. The Wiener RMS (Root Mean Square) Error Criterion in Filter Design and Prediction. *Journal of Mathematical Physics*, 25(4):261–278, 1947.

[162] Christophe Leys, Ley Christophe, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting Outliers: Do not use Standard Deviation Around the Mean, use Absolute Deviation Around the Median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[163] Tilman Liebchen. An Introduction to MPEG-4 Audio Lossless Coding. In *IEEE ICASSP 2004*, Montreal, Canada, May 2004.

[164] Venus Khim-Sen Liew. On autoregressive order selection criteria. Technical report, EconWPA, 2004.

[165] G.M. Ljung and G.E.P. Box. On a Measure of Lack of Fit in Time Series Models. *Biometrika*, 65(2):297–303, 1978.

[166] Marco J. Lombardi and Simon J. Godsill. Monte Carlo Bayesian Filtering and Smoothing for TVAR Signals in Symmetric α-stable Noise. In *European Signal Processing Conference*, pages 865–872, Vienna, Austria, September 2004.

[167] Hongjun Lu, Rudy Setiono, and Huan Liu. Effective Data Mining Using Neural Networks. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):957–961, 1996.

[168] Junshui Ma and Simon Perkins. Online Novelty Detection on Temporal Sequences. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–618. Association for Computing Machinery, 2003.

[169] Prasanta Chandra Mahalanobis. On the Generalised Distance in Statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936.

[170] H. R. Marateb, M. Rojas-Martínez, M. A. Mañanas Villanueva, and R. Merletti. Robust Outlier Detection in High-density Surface Electromyographic Signals. In *IEEE International Conference on Engineering in Medicine and Biology Society*, pages 4850–4853, August 2010.

[171] Maged Marghany and Mazlan Hashim. Comparison Between Mahalanobis Classification and Neural Network for Oil Spill Detection Using RADARSAT-1 SAR Data. *International Journal of the Physical Sciences*, 6(3):566–576, 2011.

[172] George Marsaglia and T. A. Bray. A Convenient Method for Generating Normal Variables. *SIAM Review*, 6(3):260–264, July 1964.

[173] Buck W. Marshall, Coley A. Raymond, and Robbins P. David. A Generalized Vandermonde Determinant. *Journal of Algebraic Combinatorics*, 1(2):105–109, September 1992.

[174] Brian W. Matthews. Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme. *Biochimica et Biophysica Acta Protein Structure*, 405(2):442–451, 1975.

[175] Hermann Matthies and Gilbert Strang. The Solution of Nonlinear Finite Element Equations. *International Journal for Numerical Methods in Engineering*, 14(11):1613–1626, 1979.

[176] José Alberto Mauricio. Exact Maximum Likelihood Estimation of Partially Nonstationary Vector ARMA Models. *Computational Statistics and Data Analysis*, 50(12):3644–3662, August 2006.

[177] Scott McCarrey and Lesley A. Wright. *Perspectives on the Performance of French Piano Music*. Ashgate Publishing Company, 2014.

[178] James L. McClelland and David E. Rumelhart. Training Hidden Units: The Generalised Delta Rule. In *Explorations in Parallel Distributed Processing*, volume 3, pages 121–137. MIT Press, 1998.

[179] Erik Meijering. A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing. In *Proceedings of the IEEE*, volume 90, pages 319–342, March 2002.

[180] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[181] Microsoft. Multiple channel audio data and WAVE files. http://msdn.microsoft.com/en-us/library/windows/hardware/dn653308(v=vs.85).aspx, March 2007. Accessed: 2014-10-08.

[182] Microsoft. Resource Interchange File Format Services. http://msdn.microsoft.com/en-us/library/windows/desktop/dd798636(v=vs.85).aspx, 2014. Accessed: 2014-10-08.

[183] Microsoft. Windows Media Audio Encoder. http://msdn.microsoft.com/en-us/library/ff819498(v=vs.85).aspx, 2014. Accessed: 2014-10-08.

[184] Tommaso Minerva and Irene Poli. Building ARMA Models with Genetic Algorithms. In *Applications of Evolutionary Computing*, volume 2037, pages 335–342. Springer Berlin Heidelberg, 2001.

[185] John Moody and Joachim Utans. Architecture Selection Strategies for Neural Networks: Application to Corporate Bond Rating Prediction. In *Neural Networks in the Capital Markets*, 1995.

[186] Eliakim Hastings Moore. On the Reciprocal of the General Algebraic Matrix. *Bulletin of the American Mathematical Society*, 26(9):394–395, 1920.

[187] José Luis Morales and Jorge Nocedal. Remark on "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-scale Bound Constrained Optimization". *ACM Transactions on Mathematical Software*, 38(1):7:1–7:4, December 2011.

[188] In Jae Myung. Tutorial on Maximum Likelihood Estimation. *Journal of Mathematical Psychology*, 47(1):90–100, 2003.

[189] John Lawrence Nazareth. Conjugate Gradient Methods Less Dependent on Conjugacy. *SIAM Review*, 28(4):501–511, December 1986.

[190] Isaac Newton and Derek Thomas Whiteside. *The Mathematical Papers of Isaac Newton*, volume 1 of *The Mathematical Papers of Sir Isaac Newton*. Cambridge University Press, 2008.

[191] Lucien Ngalamou, Marcus George, and Leary Myers. Hardware Implementation of the PCM Codec for VOIP Telephony. *International Journal of Engineering Science and Technology*, 2(10):5069–5079, 2010.

[192] Maciej Niedźwiecki. Elimination of Clicks and Background Noise from Archive Gramophone Recordings Using the Two Track Mono Approach. In *European Signal Processing Conference*, pages 1749–1752, Trieste, Italy, 1996.

[193] Maciej Niedźwiecki and Marcin Ciołek. Elimination of Clicks from Archive Speech Signals Using Sparse Autoregressive Modeling. In *Proceedings of the 20th European Signal Processing Conference*, pages 2615–2619, August 2012.

[194] Maciej Niedźwiecki and Marcin Ciołek. Elimination of Impulsive Disturbances from Archive Audio Signals Using Bidirectional Processing. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):1046–1059, May 2013.

[195] Maciej Niedźwiecki and Marcin Ciołek. Elimination of Impulsive Disturbances from Stereo Audio Recordings. In *European Signal Processing Conference*, pages 66–70, September 2014.

[196] Maciej Niedźwiecki and Marcin Ciołek. Localization of Impulsive Disturbances in Archive Audio Signals using Predictive Matched Filtering. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2888–2892, May 2014.

[197] Maciej Niedźwiecki, Marcin Ciołek, and Krzysztof Cisowski. Elimination of Impulsive Disturbances From Stereo Audio Recordings Using Vector Autoregressive Modeling and Variable-order Kalman Filtering. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(6):970–981, June 2015.

[198] Maciej Niedźwiecki and Krzysztof Cisowski. Adaptive Scheme for Elimination of Background Noise and Impulsive Disturbances from Audio Signals. In *Quatrozieme Colloque*, pages 519–522, Juan-les-Pins, France, 1993.

[199] Maciej Niedźwiecki and Krzysztof Cisowski. Smart Copying - A New Approach to Reconstruction of Audio Signals. *IEEE Transactions on Signal Processing*, 49(10):2272–2282, October 2001.

[200] Jorge Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[201] Chorng-Shyong Ong, Jih-Jeng Huang, and Gwo-Hshiung Tzeng. Model Identification of ARIMA Family using Genetic Algorithms. *Applied Mathematics and Computation*, 164(3):885–912, 2005.

[202] Laurent Oudre. Automatic Detection and Removal of Impulsive Noise in Audio Signals. *Image Processing On Line*, 2014.

[203] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. LOCI: Fast Outlier Detection Using the Local Correlation Integral. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 315–326, March 2003.

[204] Karl Pearson. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London*, 58(347):240–242, 1895.

[205] Roger Penrose. A Generalized Inverse for Matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51:406–413, July 1955.

[206] Dimitris N. Politis, Joseph P. Romano, and Michael Wolf. Inference for Autocorrelations in the Possible Presence of a Unit Root. *Journal of Time Series Analysis*, 25(2):251–263, March 2004.

[207] S. Quackenbush. MPEG Advanced Audio Coding. http://mpeg.chiariglione.org/standards/mpeg-2/advanced-audio-coding, October 2005. Accessed: 2014-10-08.

[208] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. *ACM SIGMOD Record*, 29(2):427–438, May 2000.

[209] Joseph Raphson. *Analysis Aequationum Universalis*. London, United Kingdom, 1690.

[210] George Rasch. On General Laws and the Meaning of Measurement in Psychology. In *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, volume 4, pages 321–333, Berkeley, California, United State of America, 1961. University of California Press.

[211] Oliver Read and Walter L. Welch. *From Tin Foil to Stereo: Evolution of the Phonograph*. H. W. Sams, Indianapolis, Indiana, United States of America, 2 edition, 1976.

[212] Robert S. Reis. An Optical Turntable. Master's thesis, Stanford University, Stanford, California, United States of America, 1987.

[213] Felix Richter. The LP is Back! http://www.statista.com/chart/1465/vinyl-lp-sales-in-the-us, January 2014. Accessed: 2014-04-16.

[214] Martin Riedmiller and Heinrich Braun. RPROP - A Fast Adaptive Learning Algorithm. In *International Symposium on Computer and Information Sciences*, pages 279–285, 1992.

[215] Martin Riedmiller and Heinrich Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

[216] Frigyes Riesz. Sur la convergence en moyenne. In *Sur les systémes orthogonaux de fonctions*, volume 144, pages 615–619, 1907.

[217] David E. Rumelhart, Geoffrey E. Hintont, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986.

[218] Carl Runge and Rudolf Mehmke. Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten. In *Zeitschrift für Mathematik und Physik*, volume 46 of *Americana*, pages 224–243, November 1901.

[219] Henrik Sangö. Modeling Electricity Prices in the German Market. Master's thesis, Göteborg University, Göteborg, Sweden, 2008.

[220] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. Local Outlier Detection Reconsidered: a Generalized View on Locality with Applications to Spatial, Video, and Network Outlier Detection. *Data Mining and Knowledge Discovery*, 28(1):190–237, 2014.

[221] Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2):461–464, 1978.

[222] David F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656, July 1970.

[223] Jack Sherman and Winifred J. Morrison. Adjustment of an Inverse Matrix Corresponding to a Change in One Element of a Given Matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, March 1950.

[224] Ronald E. Shiffler. Maximum Z Scores and Outliers. *The American Statistician*, 42(1):79–80, February 1988.

[225] O.I. Shittu and D.K. Shangodoyin. Detection of Outliers in Time Series Data: A Frequency Domain Approach. *Asian Journal of Scientific Research*, 1(2):130–137, 2008.

[226] P. Sibi, S. Allwyn Jones, and Siddarth P. Analysis of Different Activation Functions Using Back Propagation Neural Networks. *Journal of Theoretical and Applied Information Technology*, 47(3):1264–1268, 2013.

[227] Stelios Sidiroglou-Douskos, Sasa Misailovic, Henry Hoffmann, and Martin Rinard. Managing Performance vs Accuracy Trade-offs with Loop Perforation. In *Proceedings of the ACM SIGSOFT Symposium and the European Conference on Foundations of Software Engineering*, pages 124–134, New York, United States of America, 2011. ACM.

[228] Paragon Sight and Sound. Koetsu Cartridge Guide 2012. http://www.sonarecoeli.com/pdf/koetsu.pdf, 2012. Accessed: 2014-04-25.

[229] Eugen Slutzky. The Summation of Random Causes as the Source of Cyclic Processes. *Econometrica*, 5(2):105–146, 1927.

[230] W. B. Snow. Audible Frequency Ranges of Music, Speech and Noise. *Bell System Technical Journal*, 10(4):616–627, 1931.

[231] B. Solaiman and E. P. Maillard. Image Compression using HLVQ Neural Network. In *Acoustics, Speech, and Signal Processing*, volume 5, pages 3447–3450, May 1995.

[232] David J. Spiegelhalter, Nicola G. Best, Bradley P. Carlin, and Angelika van der Linde. Bayesian Measures of Model Complexity and Fit. *Journal of the Royal Statistical Society*, 64(4):583–639, 2002.

[233] Pablo Sprechmann, Alexander M. Bronstein, Jean-Michel Morel, and Guillermo Sapiro. Audio Restoration from Multiple Copies. In *International Conference on Acoustics, Speech and Signal Processing*, pages 878–882, Vancouver, Canada, May 2013. IEEE.

[234] Christoph Frank Stallmann. Visore Project. https://github.com/visore, October 2014. Accessed: 2014-10-08.

[235] J. Stirling. *Methodus Differentialis Sive Tractatus de Summation et Interpolation Serierum Infinitarium*. 1730.

[236] Richard S. Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3(1):9–44, 1988.

[237] Mamoru Tamba. A Hybrid A/D Converter with 120dB SNR and -125dB THD. In *IEEE International Test Conference*, pages 1–9, October 2008.

[238] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, Boston, Massachusetts, United States of America, 1 edition, 2005.

[239] Jian Tang, Zhixiang Chen, Ada Fu, and David Cheung. Enhancing Effectiveness of Outlier Detections for Low Density Patterns. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Taipei, Taiwan, May 2002.

[240] Brook Taylor. *Methodus Incrementorum Directa et Inversa*. Impensis Gulielmi Innys, 1717.

[241] Ryan J. Tibshirani. Fast Computation of the Median by Successive Binning. *Cornell University Computing Research Repository*, 2008.

[242] Edward Charles Titchmarsh. *Introduction to the Theory of Fourier Integrals*. Clarendon Press, 1937.

[243] Ruey S. Tsay. Outliers, Level Shifts, and Variance Changes in Time Series. *Journal of Forecasting*, 7(1):1–20, 1988.

[244] Complete Music Update. SoundScan may be under reporting US vinyl sales. http://www.thecmuwebsite.com/article/soundscan-may-be-under-reporting-us-vinyl-sales, October 2011. Accessed: 2014-04-16.

[245] Francisco J. Valverde-Albacete and Carmen Peláez-Moreno. 100% Classification Accuracy Considered Harmful: The Normalized Information Transfer Factor Explains the Accuracy Paradox. *PLoS ONE*, 9(1), January 2014.

[246] Frans van den Bergh and Andries Petrus Engelbrecht. Cooperative Learning in Neural Networks using Particle Swarm Optimizers. *South African Computer Journal*, 26:84–90, 2000.

[247] S. V. Vaseghi and R. Frayling-Cork. Restoration of Old Gramophone Recordings. *Journal of the Audio Engineering Society*, 40(10):791–801, October 1992.

[248] Saeed V. Vaseghi. *Advanced Digital Signal Processing and Noise Reduction*. Wiley, Hoboken, New Jersey, United States of America, 4 edition, January 2009.

[249] Singh Vijendra and Pathak Shivani. Robust Outlier Detection Technique in Data Mining: A Univariate Approach. *Cornell University Computing Research Repository*, 2014.

[250] Mark S. Voss and Xin Feng. ARMA Model Selection Using Particle Swarm Optimization and AIC Criteria. In *Triennial World Congress of the International Federation of Automatic Control*, volume 15, Barcelona, Spain, 2002.

[251] Gilbert Thomas Walker. On Periodicity in Series of Related Terms. *Proceedings of the Royal Society of London*, 131:518–532, 1931.

[252] John Wallis. *Arithmetica Infinitorum*. Olms Verlag, Hildesheim, Germany, 1972.

[253] J. L. Wals, J.X. Ahlberg, and E. N. Nilsson. Best Approximation Properties of the Spline Fit. *Journal of Applied Mathematics and Mechanics*, 11:225–234, 1962.

[254] Janett Walters-Williams and Yan Li. Comparative Study of Distance Functions for Nearest Neighbors. In *Advanced Techniques in Computing Sciences and Software Engineering*, pages 79–84. Springer, 2010.

[255] Edward Waring. Problems concerning Interpolations. *Philisophical Transactions of the Royal Society of London*, 69:59–67, 1779.

[256] Christopher J. C. H. Watkins and Peter Dayan. Q-Learning. *Machine Learning*, 8(3-4):279–292, 1992.

[257] Per-Åke Wedin. On the Almost Rank Deficient Case of the Least Squares Problem. *BIT Numerical Mathematics*, 13(3):344–354, 1973.

[258] Karl T. Weierstrass. Über die Analytische Darstellbarkeit Sogenannter Willkürlicher Funktionen einer Reellen Veränderlichen. In *Sitzungsbericht der Akademie zu Berlin*, pages 633–639, Berlin, Germany, 1885.

[259] Paul J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, Massachusetts, United States of America, 1974.

[260] Halbert White. A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity. *Econometrica*, 48(4):817–38, May 1980.

[261] Edmund Taylor Whittaker and G. Robinson. *The Calculus of Observations: An Introduction to Numerical Analysis*. Dover Publications, 4 edition, 1967.

[262] Peter Whittle. *Hypothesis Testing in Time Series Analysis*. PhD thesis, Uppsala University, Uppsala, Sweden, 1951.

[263] Nobert Wiener. *The Fourier Integral and Certain of its Applications*. Cambridge University Press, Cambridge, United Kingdom, 1 edition, 1933.

[264] Raymond R. Wile. Etching the Human Voice: The Berliner Invention of the Gramophone. *Association for Recorded Sound Collections Journal*, 21(1):2–22, 1990.

[265] Daniel S. Wilks. *Statistical Methods in the Atmospheric Sciences*, volume 91 of *International Geophysics*. Academic Press, 2 edition, 2005. pp. 122.

[266] Liu Yan, Hua Siliang, Wang Donghui, and Hou Chaohuan. A 100dB-SNR mixed CT/DT audio-band sigma delta ADC. In *IEEE International Conference on Solid-State and Integrated Circuit Technology*, pages 199–201, November 2010.

[267] Peng Yang and Biao Huang. KNN Based Outlier Detection Algorithm in Large Dataset. In *IEEE International Workshop on Geoscience and Remote Sensing*, volume 1, pages 611–613, Shanghai, China, December 2008.

[268] George Udny Yule. The Applications of the Method of Correlation to Social and Economic Statistics. *Journal of the Royal Statistical Society*, 72:721–730, 1909.

[269] George Udny Yule. Why do we sometimes get Nonsense Correlations between Time-Series? *Journal of the Royal Statistical Society*, 89(1):1–64, 1926.

[270] George Udny Yule. On a Method of Investigating Periodicities in Disturbed Series with Special Reference to Wolfer's Sunpot Numbers. *Philosophical Transactions of the Royal Society of London*, 226:267–298, 1927.

[271] Jerrold H. Zar. *Biostatistical Analysis*. Prentice-Hall, Upper Saddle River, New York, United States of America, 5 edition, 2007.

[272] Manqi Zhao and Venkatesh Saligrama. Anomaly Detection with Score Functions Based on Nearest Neighbor Graphs. In *Neural Information Processing Systems*, pages 2250–2258, Vancouver, Canada, December 2009.

[273] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-scale Bound-constrained Optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, December 1997.

Appendix A

Empirical Data

This appendix provides an overview of the test data used during the empirical analysis with the purpose of providing enough information to replicate the test results. The data is divided into eight major genres, namely, classical, country, electro, jazz, metal, pop, reggae and rock. Each genre consists of 100 songs which were accumulated from a number of chart lists, specifically the United States and United Kingdom Billboard album and singles charts and various Rolling Stones Magazine charts. The last section of this appendix lists the gramophone albums and tracks that formed part of the second test data set.

A.1 Classical Test Dataset

Classical music is typically composed of the string, brass, percussion, and woodwind families of instruments and also includes opera. The genre is subdivided into medieval, renaissance, baroque, classical, modern, and contemporary, according to the era of composition. Classical music typically has a very narrow dynamic range with calm or even silent fragments. The complete classical test dataset used for the empirical analysis is given in table A.1.

Table A.1: The classical test dataset.

Title	Artist	Duration
23. Klavierkonzert (K. 488) - Adagio	Wolfgang Amadeus Mozart	03:17
23. Klavierkonzert (K. 488) - Allegro	Wolfgang Amadeus Mozart	11:07

Title	Artist	Duration
1812 Overture	Pyotr Ilyich Tchaikovsky	06:02
Adagio For Strings	Samuel Osborne Barber	10:01
Adagio In Sol Minore	Tomaso Giovanni Albinoni	09:52
Aida - Marcia Trionfale	Giuseppe Fortunino Francesco Verdi	06:14
An Der Schönen Blauen Donau (Op. 314)	Johann Strauss	10:11
Ave Maria	Charles-François Gounod	02:32
Ave Maria	Giulio Caccini	05:59
Ave Verum Corpus (K. 618)	Wolfgang Amadeus Mozart	03:25
Boléro	Maurice Ravel	14:51
Brandenburgisches Konzert Nr. 1 (BWV 1046) - Allegro	Johann Sebastian Bach	04:43
Cantique De Jean Racine (Op. 11)	Gabriel Urbain Fauré	06:46
Carmen - Habanera	Georges Bizet	04:37
Carmina Burana - O Fortuna	Carl Orff	05:20
Cavalleria Rusticana - Intermezzo	Pietro Mascagni	03:36
C'era Una Volta Il West (Once Upon A Time In The West)	Ennio Morricone	05:05
Concerto Pour Piano No. 1 (Op. 11) - Romance	Frédéric François Chopin	04:08
Concerto Pour Piano No. 2 (Op. 21)	Frédéric François Chopin	09:37
Concerto Pour Violon No. 1 (Op. 6)	Niccolò Paganini	04:58
Concierto De Aranjuez - Adagio	Joaquín Rodrigo Vidre	11:31
Dank Sei Dir, Herr	Aafje Heynis	04:25
Danse Macabre	Charles Camille Saint-Saëns	07:11
Die Zauberflöte (K. 620) - Der Hölle Rache Kocht In Meinem Herzen	Wolfgang Amadeus Mozart	03:13
Die Zauberflöte (K. 620) - Der Vogelfänger Bin Ich Ja	Wolfgang Amadeus Mozart	02:52
Die Zauberflöte (K. 620) - Overture	Wolfgang Amadeus Mozart	06:35
Doppelkonzert Für Zwei Violinen (BWV 1043)	Johann Sebastian Bach	07:16
Eine Kleine Nachtmusik (K. 525)	Wolfgang Amadeus Mozart	06:33
Exsultate, Jubilate (K. 165)	Wolfgang Amadeus Mozart	05:05
Finlandia (Op. 26)	Jean Sibelius	09:04
Für Elise	Ludwig van Beethoven	02:53
Gymnopédie No.1	Erik Alfred Leslie Satie	03:37
Jesu, Der Du Meine Seele (BWV 78) - Wir Eilen Mit Schwachen	Johann Sebastian Bach	05:19
Jesus Bleibet Meine Freude (BWV 147)	Johann Sebastian Bach	03:49
Kanon In D	Johann Pachelbel	04:25
Klarinettenkonzert (K. 622) - Adagio	Wolfgang Amadeus Mozart	07:01
Klarinettenkonzert (K. 622) - Rondo	Wolfgang Amadeus Mozart	09:29
Konzert Für Flöte, Harfe Und Orchester (K. 299) - Allegro	Wolfgang Amadeus Mozart	10:30
Krönungsmesse (K. 317) - Agnus Dei	Wolfgang Amadeus Mozart	03:56
La Donna Mobile Rigoletto	Giuseppe Fortunino Francesco Verdi	02:23
La Traviata - Libiamo Ne' Lieti Calici	Giuseppe Fortunino Francesco Verdi	03:15
Lakmé - Duo Des Fleurs	Clément Philibert Léo Delibes	05:05
Le Nozze Di Figaro (K. 492) - Voi, Che Sapete Che Cosa E Amor	Wolfgang Amadeus Mozart	02:49
Le Quattro Stagioni (Op. 8, RV 269) - La Primavera	Antonio Lucio Vivaldi	10:51
Le Quattro Stagioni (Op. 8, RV 293) - l'Autunno	Antonio Lucio Vivaldi	11:08
Les Pcheurs De Perles - Au Fond Du Temple Saint	Georges Bizet	06:13
Má Vlast - Vltava	Bedrich Smetana	12:51
Matthäus Passion (BWV 244) - Erbarme Dich	Johann Sebastian Bach	07:12
Matthäus Passion (BWV 244) - Kommt, Ihr Töchter	Johann Sebastian Bach	09:07
Matthäus Passion (BWV 244) - Wir Setzen Uns Mit Tränen Nieder	Johann Sebastian Bach	06:02
Méditation	Jules émile Frédéric Massenet	05:44
Mesícku Na Nebi Hlubokém	Antonín Leopold Dvorák	05:57
Messiah (HWV 56) - For Unto Us A Child Is Born	Georg Friederich Händel	04:02
Messiah (HWV 56) - Hallelujah	Georg Friederich Händel	04:22
Miserere (Psalm 51)	Gregorio Allegri	05:45
Mondscheinsonate (Op. 27)	Ludwig van Beethoven	06:07
Nabucco - Va, Pensiero	Giuseppe Fortunino Francesco Verdi	05:29
New World Symphony (Op. 95) - Largo	Antonín Dvorák	04:39
Norma - Casta Diva	Vincenzo Salvatore Carmelo Bellini	05:39
Orchestersuite Nr. 2 (BWV 1067) - Badinerie	Johann Sebastian Bach	01:26
Orchestersuite Nr. 3 (BWV 1068) - Air	Johann Sebastian Bach	05:05
Orfeo Ed Euridice	Christoph Willibald Gluck	03:59
Orfeo Ed Euridice - Dance Of The Blessed Spirits	Christoph Willibald Ritter von Glück	06:52
Panis Angelicus	César Franck	04:05
Peer Gynt Suite No. 1 (Op. 46) - Morgenstemning	Edvard Hagerup Grieg	04:01

Title	Artist	Duration
Piano Concerto No. 1 (Op. 23)	Pyotr Ilyich Tchaikovsky	09:00
Piano Concerto No. 2 (Op. 18) - Adagio Sostenuto	Sergej Vassiljevitsj Rachmaninoff	11:59
Piano Concerto No. 2 (Op. 18) - Moderato	Sergej Vassiljevitsj Rachmaninoff	11:08
Piano Concerto No. 5 (Op. 73) - Adagio Un Poco Mosso	Ludwig van Beethoven	09:33
Piano Concerto No. 5 (Op. 73) - Rondo	Ludwig van Beethoven	12:11
Piano Concerto No. 21 (K. 467)	Wolfgang Amadeus Mozart	07:01
Requiem (K. 626) - Dies Irae	Wolfgang Amadeus Mozart	01:48
Requiem (K. 626) - Domine Jesu Christe	Wolfgang Amadeus Mozart	04:15
Requiem (K. 626) - Introitus	Wolfgang Amadeus Mozart	05:57
Requiem (K. 626) - Kyrie Eleison	Wolfgang Amadeus Mozart	02:50
Requiem (K. 626) - Lacrimosa	Wolfgang Amadeus Mozart	03:09
Requiem (Op. 48) - In Paradisum	Gabriel Urbain Fauré	03:49
Requiem (Op. 48) - Pie Jesu	Gabriel Urbain Fauré	03:21
Rhapsody In Blue	George Gershwin	16:30
Romeo And Juliet Suite No. 2 (Op. 64b)	Sergei Sergeyevich Prokofiev	05:21
Serse (HWV 40) - Ombra Mai Fu	Georg Friederich Händel	03:19
Solomon (HWV 67) - The Arrival Of The Queen Of Sheba	Georg Friederich Händel	02:59
Spartacus - Adagio Of Spartacus And Phrygia	Aram Ilich Khachaturian	09:54
Stabat Mater	Giovanni Battista Pergolesi	04:18
Swan Lake (Op. 20)	Pyotr Ilyich Tchaikovsky	03:14
Symphony No. 5	Gustav Mahler	06:42
Symphony No. 5 (Op. 67)	Ludwig van Beethoven	07:06
Symphony No. 6 (Op. 68)	Ludwig van Beethoven	10:34
Symphony No. 7 (Op. 92)	Ludwig van Beethoven	14:17
Symphony No. 9 (Op. 125)	Ludwig van Beethoven	12:59
Symphony No. 40 (K. 550) - Molto Allegro	Wolfgang Amadeus Mozart	07:41
The Blue Danube	Johann Strauss	05:44
Toccata E Fuga (BWV 565)	Johann Sebastian Bach	09:23
Turandot - Nessun Dorma	Giacomo Antonio Domenico Michele	03:22
Vesperae De Dominica (K. 321) - Laudate Dominum	Wolfgang Amadeus Mozart	05:48
Violinkonzert Nr. 1 (Op. 26) - Adagio	Max Christian Friedrich Bruch	08:42
Violinkonzert Nr. 1 (Op. 26) - Allegro Moderato	Max Christian Friedrich Bruch	08:36
Wachet Auf, Ruft Uns Die Stimme (BWV 140)	Johann Sebastian Bach	03:30
Wassermusik (HWV 348-350)	Georg Friederich Händel	04:46
Weihnachtsoratorium (BWV 248) - Jauchzet, Frohlocket	Johann Sebastian Bach	08:19

A.2 Country Test Dataset

Country music is characterised by simple harmonies accompanied by mostly string instruments such as banjos, acoustic guitars, fiddles, and other instruments such as harmonicas and accordions. Country can be divided into folk, swing, blues, boogie, and gospel. The genre has a lower dynamic range than most other genres. The complete country test dataset used for the empirical analysis is given in table A.2.

Table A.2: The country test dataset.

Title	Artist	Duration
A Boy Named Sue	Johnny Cash	03:40
Alabam	Cowboy Copas	02:17
Almost Persuaded	David Houston	03:01
Always Late	Lefty Frizzell	03:08
Always On My Mind	Willie Nelson	03:34

Title	Artist	Duration
Back In The Saddle Again	Gene Autry	02:43
Boot Scootin' Boogie	Brooks and Dunn	03:18
Bouquet Of Roses	Eddy Arnold	02:22
Cattle Call	Eddy Arnold	02:44
Chattanoogie Shoe Shine Boy	Red Foley	02:45
Coal Miner's Daughter	Loretta Lynn	03:01
Convoy	CW McCall	03:48
Cool Water	Sons Of The Pioneers	02:46
Crazy	Patsy Cline	02:43
Crazy Arms	Ray Price	02:34
Detroit City	Bobby Bare	02:46
Don't It Make Your Brown Eyes	Crystal Gayle	02:31
Don't Worry	Marty Robbins	03:11
El Paso	Marty Robbins	04:23
Elvira	The Oak Ridge Boys	03:45
Every Which Way But Loose	Eddie Rabbitt	02:54
Family Tradition	Hank Williams Jr	04:01
Flowers On The Wall	Statler Brothers	02:26
Folsom Prison Blues	Johnny Cash	02:50
For The Good Times	Ray Price	03:49
Forever And Ever Amen	Randy Travis	03:35
Four Walls	Jim Reeves	02:52
Friends In Low Places	Garth Brooks	04:20
Golden Ring	Tammy Wynette	03:06
Gone	Ferlin Husky	02:25
Good Hearted Woman	Waylon Jennings and Willie Nelson	03:02
Guitar Polka	Al Dexter	02:42
He Stopped Loving Her Today	George Jones	03:18
Heartbreak Hotel	Elvis Presley	02:06
He'll Have To Go	Jim Reeves	02:23
Hello Darlin	Conway Twitty	02:26
Hello Walls	Faron Young	02:21
Here Comes My Baby	Dottie West	02:31
I Ain't Never	Mel Tillis	02:08
I Believe In You	Don Williams	04:09
I Can't Stop Loving You	Ray Charles	04:13
I Walk The Line	Johnny Cash	02:44
I Want To Be A Cowboy's Sweetheart	Patsy Montana	03:08
I Was Country	Barbara Mandrell	03:40
I Will Always Love You	Dolly Parton	03:06
I'll Hold You In My Heart	Eddy Arnold	02:44
I'm Movin On	Hank Snow	02:50
I'm So Lonesome I Could Cry	Hank Williams	02:49
In The Jailhouse Now	Jimmie Rodgers	03:18
In The Jailhouse Now	Webb Pierce	02:03
It Was Almost Like A Song	Ronnie Milsap	03:35
It Wasn't God Who Made Honky Tonk Angels	Kitty Wells	02:33
I've Forgotten More Than You'll Ever Know	The Davis Sisters	02:59
I've Got A Tiger By The Tail	Buck Owens	02:14
Just Someone I Used To Know	Porter Wagoner and Dolly Parton	02:24
King Of The Road	Roger Miller	02:21
Kiss An Angel Good Morning	Charley Pride	02:04
Lookin' For Love	Johnny Lee	03:31
Loose Talk	Carl Smith	02:32
Louisiana Woman Mississippi Man	Loretta Lynn and Conway Twitty	02:24
Love Without End Amen	George Strait	03:03
Lovesick Blues	Hank Williams Sr	02:45
Luckenbach Texas	Waylon Jennings	03:16
Mountain Music	Alabama	04:13
My Hang Up Is You	Freddie Hart	02:06
New San Antonio Rose	Bob Wills	02:39
Oh Lonesome Me	Don Gibson	02:32
Okie From Muskogie	Merle Haggard	02:40

Title	Artist	Duration
Once A Day	Connie Smith	02:19
Please Help Me I'm Falling	Hank Locklin	02:53
Rhinestone Cowboy	Glen Campbell	03:14
Room Full Of Roses	Mickey Gilley	02:47
Rose Garden	Lynn Anderson	02:58
Silver Wings	Merle Haggard	02:46
Singing The Blues	Marty Robbins	02:28
Sixteen Tons	Tennessee Ernie Ford	02:39
Slipping Around	Jimmy Wakely and Margaret Whiting	02:14
Slow Poke	Pee Wee King	02:54
Smoke, Smoke, Smoke	Tex Williams	03:00
So Round, So Firm, So Fully Packed	Merle Travis	03:00
Stand By Your Man	Tammy Wynette	02:42
Still	Bill Anderson	02:56
The Battle Of New Orleans	Johnny Horton	02:35
The Dance	Garth Brooks	03:58
The Devil Went Down To Georgia	Charlie Daniels Band	03:35
The Gambler	Kenny Rogers	03:30
The Green, Green Grass Of Home	Porter Wagoner	02:24
The Most Beautiful Girl	Charlie Rich	02:41
The Prisoner's Song	Vernon Dalhart	03:05
The Three Bells	The Browns	02:53
The Wild Side Of Life	Hank Thompson	02:45
There Goes My Everything	Jack Greene	03:08
Wabash Cannonball	Roy Acuff	02:32
Walk On By	Leroy Vandyke	02:22
Walking The Floor Over You	Ernest Tubb	02:37
What's Your Mama's Name	Tanya Tucker	03:13
White Lightning	George Jones	02:30
Why Not Me	The Judds	03:31
Will The Circle Be Unbroken	Carter Family	03:48
Young Love	Sonny James	02:31

A.3 Electronic Test Dataset

Electronic music encapsulates music generated with electronic instruments such as electric guitars, keyboards, theremins, and sound synthesizers. A large part of this genre is directly created through computer software. Subgenres include electro, techno, dance, trance, and house. The genre is characterised by a very wide dynamic range with a large number of samples reaching the extremes. The complete electronic test dataset used for the empirical analysis is given in table A.3.

Table A.3: The electronic test dataset.

Title	Artist	Duration
18 mne uzhe	Resource and Reflex	05:27
Adagio For Strings	Tiesto	07:24
Always And Forever	DJ Splash	03:45
Anima Libera	DJ Raaban	03:05
Another World	DJ Shog	03:54
Are You Ready	Pakito	03:28

Title	Artist	Duration
Baila With Me	Ravekorr and Sander	05:56
Bass Is Kicking	DJ Splash and B-Bass and DJ MDS	03:43
Be Alive	Stian K	04:09
Better Off Alone	Alice Deejay	03:36
Bounce	Tune Up	03:29
Can You Feel It	Dj Raaban	03:18
Can't You See	Nikita and Lance	03:31
Castle In The Sky	DJ Satomi	03:15
Club Inferno	Scarf	03:31
Cold As Ice	Starsplash	04:41
Colors Of The Rainbow	Tune Up and Italobrothers	03:29
Come Home	Zoe	04:52
Cuttin' Deep	Darren Styles	02:50
Dam Dadi Doo	Nightcore	02:43
Dance With My Pants	Crazy Rockers	01:24
Day After Day	Millennium	07:49
Days Goes By	DJ Splash	03:08
Days Goes By	DJ Splash	03:08
DJ Number 1	Kompulsor	03:19
Don't Stop Till You Get Enough	Eliess	05:55
Ecuador	Sash!	03:35
Encanto	Danijay	04:46
End Of Summer	Zone Breaker	03:55
Every Single Day	Benassi Bros and Dhany	03:42
Fable	Robert Miles	03:54
Feel So Blue	Secondtunez	04:01
Flying High	DC-10	03:38
Flying High	DJ Splash	02:53
Freedom	DJ Mangoo	03:51
Frozen Flame	Jekyll and Hyde	03:34
Funeral Song	Vr00z	04:48
Hardstyle My Style	Starsplash	03:40
Have You Ever Been Mellow	Tune Up	03:55
I Adore	DJ Breeza	03:08
I Miss You	Basshunter	03:48
I Touch Myself	Jan and Scarlet Wayne	03:43
If I Could Be You	DJ Dean	03:19
I'm Just More	Vibeout	05:43
I'm Your Basscreator	Basshunter	05:25
In The Name Of Love	Jet Set	03:39
It's A Dream	East Clubbers	04:46
Le	Settler Project	03:26
Lektion 1	Bassrockerz	02:37
L'Esperanza	Topmodelz	03:09
Like i Love you	The Hitmen	03:27
Little Star	Lazard	06:10
Love You Till I Die	Alextc	03:43
Magic Summer Night	Klubbingman	03:27
Makes Me Go Mmmm	Jazmine	03:29
Me So Horny	Dj Porny	03:09
Milky Way	Cyrus And The Joker	10:01
Monkey Island	Banana Inc	04:15
More More More	East Clubbers	03:36
More Than Words	Rave Allstars	03:36
Move Your Hands Up	Clubraiders	06:09
Nasty Girl	Renegade Masterz	03:49
New Life	DJ Splash	03:08
Now You're Gone	Basshunter	02:35
Nu Ska Ni Fa Hora Gott Folk	DJ Bass Rocker	03:57
Paradise on E	B0UNC3	03:24
Pina Colada Boy	Baby Alice	05:18
Place To Be	Redwing	03:23
Por Que No	United Beats	04:43

Title	Artist	Duration
Professional Party People	Basshunter	03:09
Project Well	Beattraax	03:48
Rainbow	DJ Sixty	02:56
Rave Heaven	Dave McCullen	03:34
Rave Techno	DJ Mangoo	03:25
Ravers fantasy	Tune Up	03:30
Ride On Time	Dancetech and Tune Up	04:07
Secret of Love	Barcera	03:42
Secret Of Love	DJ Bobo and Sandra	03:16
September 28th	Atlantis	04:36
Show Me Your Love	Bassrockerz	03:14
Sommaren är här	Secondtunez	03:05
Summertime	Sandra Gee	03:56
Sunrise	Angel City	03:43
Surrender	Bassrockerz and Elena	03:04
Sweet Love	DJ Cargo and D-Verse	06:46
Sweetest Ass In The World	Alex C and Y-Ass	03:17
Tarzan Boy	Denise Lopez	03:11
Techno Rocker	Base Attack	03:31
Tell Me Why	Supermode	02:50
The Logical Song	Rave Allstars	03:23
The Logical Song	Scooter	03:52
The Power Of Pleasure	DJ Carpi	04:24
This Is My Life	DJ Splash	03:28
Time 4 Dance	Clubnature	02:37
Tune	2 Playaz	03:27
When Love Becomes A Lie	Liz Kay	03:06
When The Rain Begins To Fall	Age Pee	03:43
You	Special D	03:01
Youre My Angel	Styles and Breeze	05:13
You're The One That I Want	Deep Spirit	03:55

A.4 Jazz Test Dataset

Jazz is typically composed using drums, guitars, pianos, and brass instruments, most notably the trumpet and saxophone. Although not one of the main genres, jazz was included due to the higher amplitudes from brass instruments in a mostly narrow dynamic range. Subgenres include soul, bop, latin jazz, and often overlaps with blues and swing. The complete jazz test dataset used for the empirical analysis is given in table A.4.

Table A.4: The jazz test dataset.

Title	Artist	Duration
A Night In Tunisia	Ronnie Scott	03:42
A Quiet Gass	Henry Mancini	03:06
Aint Misbehavin	Sidney Bechet	02:54
Among My Souvenirs	Louis Armstrong	02:45
As Time Goes By	Tony Bennett	03:15
Autumn In New York	Dexter Gordon	06:33
Baby It's Cold Outside	Buddy Clark	02:22
Baby It's Cold Outside	Dinah Shore	02:22
Begin the Beguine	Artie Shaw	03:18

Title	Artist	Duration
Between the Devil and the Deep Blue Sea	Django Reinhardt	02:57
Bewitched,Bothered and Bewildered	Eddie Lockjaw Davis	03:42
Black Coffee	Ed Bentley	06:18
Blackbird	Dave Valentin	06:25
Buona Sera	Louis Prima	02:59
Contrasts	Jimmy Dorsey	03:01
Crazy Rhythm	Benny Carter	03:27
Cry Me a River	Julie London	02:50
Dragnet	Ray Anthony	02:49
Dream a Little Dream of Me	Frankie Laine	02:49
Drummin' Man	Gene Krupa	03:04
East of the Sun	Ted Heath	03:34
Embraceable You	Sarah Vaughan	04:52
Fever	Peggy Lee	03:20
Fly Me to the Moon	Cy Coleman	02:54
Frenesi	The American Patrol Orchestra	02:50
Giant Steps	Candido Camero	05:38
God Bless the Child	Billie Holiday	03:08
Greenbacks	Ray Charles	02:56
Harlem Nocturne	Jonny Cooper Orchestra	03:17
Here There And Everywhere	David Benoit	04:07
Honeysuckle Rose	Count Basie	03:03
I Wanna Be Loved	Dinah Washington	02:53
I Wanna Be Loved By You	Marilyn Monroe	02:59
In Other Words	Nancy Wilson	02:54
In the Mood	Glenn Miller	03:37
Is You Or Is You Ain't My Baby	Diana Krall	04:58
It Could Happen to You	June Christy	02:01
I've Got You Under My Skin	Stan Getz	03:19
Jeepers Creepers	Maynard Ferguson	02:35
Jumpin' at the Woodside	Count Basie	03:13
La Mer	Henry Cuesta	02:35
Let Me Off Uptown	Gene Krupa	03:04
Let's Do It, Let's Fall In Love	Eartha Kitt	03:10
Love For Sale	Cannonball Adderley	07:09
Lullaby In Rhythm	June Christy	02:37
Lullaby of Birdland	Lionel Hampton	04:17
Lullaby of Birdland	Mel Torme	04:51
Mack the Knife	Lol Williams Band	03:44
Mack the Knife	Louis Armstrong	03:26
Manhattan Serenade	Dinah Shore	03:09
Moon River	Art Blakey	05:12
Mr Bojangles	Sammy Davis Jr	05:46
My Baby Just Cares For Me	Nina Simone	03:33
My Funny Valentine	Chet Baker	02:22
My Funny Valentine	Sammy Davis Jr	03:08
My Melancholy Baby	Mildred Bailey	02:53
My Old Flame	Libbie Jo Snyder	06:48
Off Minor	Thelonious Monk	05:15
Old Devil Moon	Lena Horne	02:35
On the Sunny Side of the Street	The Memphis Belle Orchestra	03:25
On the Sunnyside of the Street	Dizzy Gillespie	05:41
Ornithology	Charlie Parker	03:05
Patterns	Ahmad Jamal	06:16
Peggy's Blue Skylight	Charles Mingus	05:20
Pennies from Heaven	Jimmy Rushing	05:26
Put Off	Milt Jackson	05:37
Rags to Riches	Tony Bennett	02:51
Route 66	Nat King Cole	03:03
S Wonderful	Mel Powell	04:16
S' Wonderful	Quincy Jones	03:15
Satin Doll	Jo Jones	05:54
Sentimental Journey	The Merry Macs	03:03

Title	Artist	Duration
Sing Me a Swing Song	Ella Fitzgerald	02:34
Sing Sing Sing	Benny Goodman	05:05
Six Flats Unfurnished	Benny Goodman	03:18
Skyliner	Charlie Barnet	03:01
Smoke Gets In Your Eyes	Lawrence Welk	02:20
So Long Eric	Charles Mingus	04:35
So What	Miles Davis	09:24
Softly, As In A Morning Sunrise	Dave Weckl	05:01
Stairway to the Stars	Bill Evans	04:51
Stolen Moments	New York Voices	06:46
Strange Fruit	Billie Holiday	03:18
Stranger on the Shore	Acker Bilk	03:09
Strangers In the Night	Billy Vaughn	01:48
Summertime	Ella Fitzgerald	04:56
Take Five	Dave Brubeck Quartet	03:24
Take the A Train	Duke Ellington	03:01
That Old Feeling	Gerry Mulligan	06:00
That's My Desire	Frankie Laine	02:52
The Entertainer	Chris Barber's Jazz Band	03:52
The Entertainer	Scott Joplin	03:37
The Girl From Ipanema	Stan Getz and Joan Gilberto	05:22
The Good Earth	Woody Herman	02:35
The Lady Is a Tramp	Erroll Garner	03:38
Toot-Toot Tootsie	Sonny Rollins	04:23
Tuxedo Junction	Al Hirt	02:43
Venus De Milo	Miles Davis	03:12
What a Difference a Day Makes	Dinah Washington	02:33
Yardbird Suite	The Modern Jazz Quartet	05:15

A.5 Metal Test Dataset

Metal is characterised by loud and distorted sound generated by drums, distorted guitars, dense bass and vigorous vocals. Although a subgenre of rock, metal was specifically included due to the very wide dynamic range and sound distortions that are similar to the noise caused by gramophone scratches. The genre can be divided into heavy metal, death metal, nu metal, punk, and grunge. The complete metal test dataset used for the empirical analysis is given in table A.5.

Table A.5: The metal test dataset.

Title	Artist	Duration
A Change of Seasons	Dream Theater	23:09
A Question of Heaven	Iced Earth	07:40
And the Mirror Cracked	Disillusion	08:28
Angry Again	Megadeth	03:47
Annihilation of the Wicked	Nile	08:36
Battles in the North	Immortal	04:12
Blood Red Skies	Judas Priest	07:49
Bombtrack	Rage Against the Machine	04:04
Cemetery Gates	Pantera	07:02
Chemo	Garden Wall	34:06

Title	Artist	Duration
Chromatic Chimera	Unexpect	05:52
Civil War	Guns N' Roses	07:42
Close to a World Below	Immolation	08:19
Close to the Edge	Yes	18:43
Code Anticode	Gordian Knot	06:47
Cold Hate, Warm Blood	Cryptopsy	03:55
Control and Resistance	Watchtower	06:59
Decrystallizing Reason	Emperor	06:23
Deep Inside	Lemur Voice	09:29
Demise and Vestige	Zero Hour	15:48
Disrespectfully Yours	Sieges Even	03:51
Dream of a New Day	Richie Kotzen	03:18
Elastic	Meshuggah	15:31
Evidence of the Unseen	Zero Hour	08:44
Exospacial Psionic Aura	Behold The Arctopus	07:31
Expect the Unexpected	Control Denied	07:18
Faceless One	Hate Eternal	04:39
Faceless Ones	Gorguts	03:50
Fire and Ice	Yngwie Malmsteen	04:31
Follow Me	Savatage	05:12
For the Love of God	Steve Vai	06:03
From Bellatrix To Betelgeuse	Sadist	04:44
Gridzone	Nocturnus	06:07
Halo	Machine Head	09:03
Helpless Corpses Enactment	Sleepytime Gorilla Museum	05:57
Higher Ground	Red Hot Chili Peppers	03:22
Hiking Up Feel Good Mountain	Upsilon Acrux	07:47
House of Nadir	Twisted into Form	06:40
I Was Made For Loving You	Kiss	04:30
Ice Age	Ice Age	11:09
Insect	Spiral Architect	05:54
Jump	Van Halen	04:26
La Idea de Borde	Coprofago	02:41
Leal Souvenir	Jonas Hellborg	10:55
Living in Fear	Dali's Dilemma	07:44
Madness Remains	MASSACRA	05:43
Monitoring the Mind	Neuraxis	03:42
Morax	Arcturus	04:21
Mourning Palace	Dimmu Borgir	05:12
Move Through Me	In Flames	03:08
Mutha	Extreme	04:52
Nevermore	Beneath The Massacre	04:05
November Rain	Guns N' Roses	08:57
One	Metallica	07:26
Perpetual Burn	Jason Becker	03:30
Piece Of Time	Atheist	04:33
Piste 2	Atrox	06:44
Polars	Textures	18:25
Prognosis	Enchant	07:24
Pseudo	Cephalic Carnage	05:55
Pull Me Under	Dream Theater	08:15
Queen of Winter, Throned	Cradle of Filth	10:28
Rearviewmirror	Pearl Jam	04:44
Relentless Beating	Cannibal Corpse	02:17
Remind The Differences	Anon Vin	04:32
Rise of the Iridescent Behemoth	The Flying Luttenbachers	20:15
Rock America	Danger Danger	04:54
Salome's Dance	Eldritch	05:27
Shadowkings	Paradise Lost	04:42
Suicide Nation	At the Gates	03:35
Symposium of Sickness	Carcass	06:57
Test of Wills	Power of Omens	19:58
Textures	Cynic	04:42

Title	Artist	Duration
The Chaos Theory	Continuo Renacer	07:40
The Conquering	Satyricon	07:28
The Edge of Forever	Symphony X	08:59
The God That Failed	Metallica	05:08
The Last Baron	Mastodon	13:01
The One Brooding Warning	Dark Tranquility	04:15
The Patient	Tool	07:14
The Pig Keepers Daughter	Psyopus	03:35
The Trooper	Iron Maiden	04:11
The Truth Will Set You Free	The Flower Kings	31:02
The Unknowable and Defeating Glow	Canvas Solaris	10:20
The Waking Hours	Sieges Even	04:43
There Goes the Neighborhood	Body Count	05:50
Thorns On My Grave	Emperor	05:56
Tornado of Souls	Megadeth	05:22
Universal Mind	Liquid Tension Experiment	07:53
Virtual Emotions	Martyr	04:08
Warp Zone	Martyr	03:04
Where Dead Angels Lie	Dissection	05:52
Wind Of Change	Scorpions	05:13
Wishmaster	Nightwish	04:24
With Strength I Burn	Emperor	08:18
Yankee Rose	David Lee Roth	03:51
You And Me	Mayadome	05:18
Youth Gone Wild	Skid Row	03:18
Zambra	Hacride	06:48
Zos Kia Cultus	Behemoth	05:33

A.6 Pop Test Dataset

Originally derived from rock, dance and country music, pop or popular music employs repeated choruses, melodic tunes, and hooks composed with a wide range of instruments. Pop music typically has a medium dynamic range with higher amplitudes caused by electric guitars, electronic beats or high-pitched vocals often repeated in refrains. The complete pop test dataset used for the empirical analysis is given in table A.6.

Table A.6: The pop test dataset.

Title	Artist	Duration
2 Become 1	Spice Girls	04:05
4 Minutes	Madonna and Justin Timberlake	04:04
7 Things	Miley Cyrus	03:25
A Public Affair	Jessica Simpson	03:22
As Long As You Love Me	Backstreet Boys	03:38
Baby One More Time	Britney Spears	03:32
Bailamos	Enrique Iglesias	03:33
Barbie Girl	Aqua	03:18
Be With You	Enrique Iglesias	03:41
Beat It	Michael Jackson	04:18
Because Of You	98 Degrees	03:55
Billie Jean	Michael Jackson	04:54
Black Or White	Michael Jackson	03:20

Title	Artist	Duration
Boombastic	Shaggy	04:09
Boyfriend	Ashlee Simpson	03:01
Break The Ice	Britney Spears	03:14
Bye Bye	Mariah Carey	04:18
Bye Bye Bye	NSYNC	03:21
Come On Over	Christina Aguilera	03:25
Cool	Gwen Stefani	03:09
Cry Me A River	Justin Timberlake and Timbaland	04:49
Dirrty	Christina Aguilera	04:46
Don't Stop the Music	Rihanna	04:27
Don't Stop 'Til You Get Enough	Michael Jackson	03:56
Escape	Enrique Iglesias	03:29
Every Morning	Sugar Ray	03:42
Everybody	Backstreet Boys	04:48
Fly	Sugar Ray and Super Cat	04:54
Genie In A Bottle	Christina Aguilera	03:39
Give Me Just One Night	98 Degrees	03:24
God Must Have Spent A Little More Time On You	NSYNC	04:43
Hollaback Girl	Gwen Stefani	03:20
Hot N Cold	Katy Perry	03:40
Hung Up	Madonna	03:24
I Kissed A Girl	Katy Perry	03:00
I Want It That Way	Backstreet Boys	03:33
I Want You	Savage Garden	03:52
I Want You Back	NSYNC	03:23
I'm a Slave 4 U	Britney Spears	03:24
I'm Too Sexy	Right Said Fred	02:52
It's Gonna Be Me	NSYNC	03:13
Just Dance	Lady Gaga	04:03
L.O.V.E.	Ashlee Simpson	02:34
LaLa	Ashlee Simpson	03:45
Larger Than Life	Backstreet Boys	03:55
Let's Take A Ride	Justin Timberlake and Pharrell Williams	04:44
Livin' La Vida Loca	Ricky Martin	04:04
Lovefool	The Cardigans	03:14
Milkshake	Kelis	03:04
Mmm'Bop	Hanson	04:01
More Than That	Backstreet Boys	03:44
Music	Madonna	03:44
My Happy Ending	Avril Lavigne	04:03
Never Again	Justin Timberlake and Brian McKnight	04:34
Never Again	Kelly Clarkson	03:35
No Diggity	Blackstreet	04:13
One Week	Barenaked Ladies	02:49
Oops I Did It Again	Britney Spears	03:32
Pop	NSYNC	02:55
Quit Playing Games	Backstreet Boys	03:59
Ray of Light	Madonna	05:21
Rich Girl	Gwen Stefani	03:57
Rock Your Body	Justin Timberlake and Pharrell Williams	04:28
Say You'll Be There	Spice Girls	03:58
Scatman	Scatman John	03:34
See You Again	Miley Cyrus	03:11
Seorita	Justin Timberlake and Pharrell Williams	04:55
SexyBack	Justin Timberlake	03:14
Shake It Off	Taylor Swift	03:39
Shape of My Heart	Backstreet Boys	03:52
Show Me The Meaning Of Being Lonely	Backstreet Boys	03:55
Since U Been Gone	Kelly Clarkson	03:09
Smile	Lily Allen	03:19
Smooth Criminal	Michael Jackson	04:17
Someday	Sugar Ray	04:05
Sometimes	Britney Spears	04:07

Title	Artist	Duration
Spice Up Your Life	Spice Girls	02:56
Take It From Here	Justin Timberlake and Pharrell Williams	06:15
Tearin' Up My Heart	NSYNC	03:32
The Way You Make Me Feel	Michael Jackson	04:58
This I Promise You	NSYNC	04:45
Thong Song	Sysqo	04:10
Thriller	Michael Jackson	05:12
To The Moon And Back	Savage Garden	05:41
Touch My Body	Mariah Carey	03:24
Toxic	Britney Spears	03:22
Truly, Madly, Deeply	Savage Garden	04:41
Unbelieveable	EMF	03:31
Waiting For Tonight	Jennifer Lopez	04:07
Waking Up In Vegas	Katy Perry	03:19
Wannabe	Spice Girls	02:55
What A Girl Wants	Christina Aguilera	03:35
What Goes Around Comes Around	Justin Timberlake	07:30
What Is Love	Haddaway	04:30
What You Waiting For	Gwen Stefani	03:42
When It's Over	Sugar Ray	03:39
Who Do You Think You Are	Spice Girls	03:46
Womanizer	Britney Spears	03:44
You Drive Me Crazy	Britney Spears	03:20
You Rock My World	Michael Jackson	04:27

A.7 Reggae Test Dataset

A genre that originates from Jamaica, reggae mainly uses guitars, hand drums, and traditional Jamaican and African instruments. This genre was included, since reggae is characterised by emphasizing the last note in a beat. This stands in direct contrast to almost all western music which emphasizes the first note in a beat. Other genres that fall in this category includes ska and rocksteady. The complete reggae test dataset used for the empirical analysis is given in table A.7.

Table A.7: The reggae test dataset.

Title	Artist	Duration
Blaze	Junior Kelly	04:06
Not I	Junior Kelly	04:22
Africa Prepare	Sizzla	03:23
Badman Place	Busy Signal and Movado	03:17
Be Free	Pressure	03:18
Be Strong	Sizzla	03:37
Blaze Fire Blaze	Sizzla	03:34
Blind To You Haters	Collie Buddz	03:51
Blood Again	Richie Spice	03:50
Born To Be Wild	Damian Marley	04:01
Burn Down The System	Collie Buddz	03:35
Call Pon Dem	Chezidek	03:23
Call Up Jah	Alborosie	03:22
Catch A Fire	Damian Marley	04:52

Title	Artist	Duration
Come Around	Collie Buddz	03:47
Come Jah	Coco Tea	02:19
Could You Be Loved	Bob Marley	03:55
Daddy	Queen Ifrica	04:11
Dem Gone	Gentleman	04:02
Different Places	Gentleman	03:35
Diversity	Alborosie	04:10
Divide And Rule	Sizzla and Jah Cure	03:29
Do You	Turbulence	03:53
Don't Haffi Dread	Morgan Heritage	03:55
Dreams Riddim	Pressure	03:16
Equal Share	Lutan Fyah	03:39
Far From Reality	Natural Black	03:02
Fire Pon Dem	Turbulence	03:35
Freedom	Richie Spice and Chuck Fender	04:13
Ganja Farmer	Marlon Asher	03:55
Ganja In My Brain	Ras Matthew	04:41
Gash Dem	Chuck Fender	03:15
Ghetto Living	Don Carlos	03:36
Hail The King	Fantan Mojah	04:00
Herbalist	Alborosie	03:09
High Grade	Jah mason	03:43
Hmm Hmm	Beenie Man	03:37
I Will Survive	Turbulance	03:29
Inna Di Road	Chezidek	03:08
It A Ring	Tonto Irie	04:02
Jah Give Me Strength	Junior Kelly	03:53
Jah Jah Crown	Alborosie	03:37
Jah Name	Daweh Congo and Ras Shiloh	04:10
Jah Works	Sizzla	04:21
Lay Low Riddim	Bounty Killa and Sizzla	02:41
Leave Us Alone	Gentleman	03:24
Like Mountain	Sizzla	03:41
Longing For	Jah Cure	03:47
Loser	Junior Kelly	04:16
Love And Affection	Pressure	03:55
Love Chant	Gentelman	04:33
Love Me Brethren	Luciano	04:03
Mama I Love You	Sean Paul	03:41
Mind Control	Stephen Marley	04:21
Mount Zion	Gentleman	03:41
Mr. Gunman	Mr. Vegas	03:17
My Meditation	Bushman	03:39
Neva Knew	Turbulence	02:19
Never Give In	C'daynger	02:28
Never Quit	Natural Black	02:59
New Day	Gentleman	03:35
Nuh Build Great Man	Fantan Mojah and Jah Cure	03:52
Oh Jah, Can't You See	Barrington Levy	03:15
One Loaf Of Bread	Damian Marley	03:18
Only Jah Love	Fantan Mojah	03:31
Phantom War	Lutan Fyah	03:45
Police Polizia	Alborosie	03:35
Rasta Should Be Deeper	Junior Kelly	03:29
Rastafari Anthem	Alborosie	03:24
Rastafari Mi Salute	LMS	04:03
Rastaman	Cocoa Tea	04:06
Really And Truly	Sizzla	03:51
Receive	Junior Kelly	04:22
Redemption Song	Bob Marley	03:50
Rumours	Gentleman	03:21
Satan Throne	Junior Kelly	03:19
See Dem Coming	Gentleman	03:51

Title	Artist	Duration
Send A Prayer	Gentleman	03:51
Sensi	Gyptian	02:44
Share The Love	Gentleman and Jah Cure	04:01
Smuggling Weed	Teacher Dee	03:29
Squeeze Breast	Movado	02:58
Still Searching	Damian Marley	05:03
Strangest Thing	Mr Easy	03:49
Street Swing Riddim	Sizzla, Fantan Mojah and Gyptian	13:34
Stubborn Woman	Natural Black	03:59
Superior	Gentleman	03:50
Tell Me How Come	Morgan Heritage	03:32
The Mission	Damian Marley and Stephen Marley	04:06
The Traffic Jam	Stephen Marley	03:41
These Are The Fucking Days	Busy Signal	03:55
This Fire	Lutan Fyah	03:34
Tough Life	Junior Kelly	04:06
Tribal War	Alborosie and Luciano	03:48
Unknown Number Private Call	Busy Signal	03:35
What Will it Take	Jah Cure	03:46
Who Am I	Capleton	04:02
Who Dem	Capelton	03:25
Woman I Need You	Sizzla	03:42
Young Girl 2 Rude	Collie Buddz	03:35

A.8 Rock Test Dataset

Also known as rock and roll, this genre is typically composed with electric and acoustic guitars, bass and drums. Rock has a wide dynamic range, but in general still smaller than metal and electronic music ranges. Subgenres include classic, hard, punk, and alternative rock. The complete rock test dataset used for the empirical analysis is given in table A.8.

Table A.8: The rock test dataset.

Title	Artist	Duration
2 Minutes to Midnight	Iron Maiden	06:05
48 Crash	Suzi Quatro	03:59
Angie	Rolling Stones	04:35
Another Brick in the Wall	Pink Floyd	04:00
Battery	Metallica	05:13
Battle Hymn	Manowar	06:58
Big Gun	AC-DC	04:22
Black Dog	Led Zeppelin	04:57
Bohemian Rhapsody	Queen	05:54
Carrie	Europe	04:31
Catch The Rainbow	Rainbow	06:40
Child in Time	Deep Purple	10:21
Crazy	Aerosmith	05:18
Cryin'	Aerosmith	05:09
Desert Rose	Sting	04:49
Do You Like It	Kingdom Come	03:39
Dreamer	Ozzy Osbourne	04:45
Eleanor Rigby	The Beatles	02:09
Englishman In New York	Sting	04:27

Title	Artist	Duration
Ever Dream	Nightwish	04:47
Fireball	Deep Purple	03:25
Forgiven	Within Temptation	04:53
Fragile	Sting	03:57
Girl	The Beatles	02:34
Goldeneye	Tina Turner	04:45
Hair Of The Dog	Nazareth	06:07
Hell Patrol	Judas Priest	03:37
Hey Stoopid	Alice Cooper	04:35
High Voltage	AC-DC	04:16
Highway Star	Deep Purple	06:56
Holy Diver	Dio	05:55
Hotel Califonia	Eagles	06:31
I Can't Dance	Genesis	06:56
I Don't Know	Ozzy Osbourne	05:14
I Don't Want To Miss A Thing	Aerosmith	05:00
I Surrende	Rainbow	04:02
I Wanna Be Your Man	Suzi Quatro	03:23
Immigrant Song	Led Zeppelin	02:26
Iron Maiden	Iron Maiden	04:22
Iron Man	Black Sabbath and Dio	07:20
It's A Heartache	Bonnie Tyler	03:30
July Morning	Uriah Heep	10:34
Kill The King	Rainbow	04:32
Killing Machine	Judas Priest	03:02
King of Rock and Roll	Dio	03:43
Kings And Queens	Aerosmith	03:48
Kings of Metal	Manowar	03:44
Lady In Black	Uriah Heep	04:43
Look At Yourself	Uriah Heep	05:12
Lost In France	Bonnie Tyler	03:54
Lost In Hollywood	Rainbow	04:56
Love Bites	Def Leppard	05:47
Magic Mirror	Yngwie Malmsteen	03:54
Mama	Genesis	06:51
Mama, I'm Coming Home	Ozzy Osbourne	04:13
Man On The Silver Mountain	Rainbow	04:40
Metal Gods	Judas Priest	04:00
Metal Heart	Accept	05:24
Metal Racer	Doro and Warloc	03:44
Michelle	The Beatles	02:48
Money	Pink Floyd	06:23
Nemo	Nightwish	04:39
New Kid In Town	Eagles	05:04
No More Tears	Ozzy Osbourne	07:25
Painkiller.	Judas Priest	06:07
Paranoid	Black Sabbath and Dio	03:47
Poison	Alice Coope	04:30
Private Dancer	Tina Turner	04:05
Razamanaz	Nazareth	04:24
Rock and Roll	Led Zeppelin	03:41
Rock'n'Roll Children	Dio	04:33
Ruby Tuesday	Rolling Stones	03:18
Satisfaction	Rolling Stones	03:46
Seventh Son Of A Seventh Son	Iron Maiden	09:55
Smoke on the Water	Deep Purple	07:37
Somebody To Love	Jefferson Airplane	04:16
Stairway to Heaven	Led Zeppelin	08:03
Stand Up And Shout	Dio	03:18
Still Loving You	Scorpions	06:12
T.N.T.	AC-DC	03:37
Telegram	Nazareth	06:34
That Was Yesterday	Foreigner	03:50

Title	Artist	Duration
The Best	Tina Turner	05:31
The Demon's Whip	Manowar	07:45
The Final Countdown	Europe	05:11
The Howling	Within Temptation	06:31
The Phantom Of The Opera	Nightwish	04:10
The Show Must Go On	Queen	04:32
The Sun Goes Down	Jorn Lande	06:28
Time	Pink Floyd	06:50
TV War	Accept	03:29
Venus	Shocking Blue	03:08
We Are The Champions	Queen	03:03
We Will Rock You	Queen	02:02
Whole Lotta Love	Led Zeppelin	05:35
Wind of Change	Scorpions	05:15
Wish You Were Here	Pink Floyd	05:41
Yesterday	The Beatles	02:10
You Don't Remember, I'll Never Forget	Yngwie Malmsteen	04:31
You Make Me Feel	Bonfire	04:43

A.9 Gramophone Test Set

Table A.9 lists the tracks and albums of the second test set obtained from gramophone records. Due to the lack of availability of sealed gramophone records in mint condition, their excessive cost and the requirement of manually flagging noisy samples, only eight albums were used. The test dataset totals 83 songs .

Table A.9: The gramophone test dataset.

Title	Artist	Album	Duration
Come Together	The Beatles	Abbey Road	04:20
Something	The Beatles	Abbey Road	03:03
Maxwell's Silver Hammer	The Beatles	Abbey Road	03:27
Oh! Darling	The Beatles	Abbey Road	03:26
Octopus's Garden	The Beatles	Abbey Road	02:51
I Want You	The Beatles	Abbey Road	07:47
Here ComesTthe Sun	The Beatles	Abbey Road	03:05
Because	The Beatles	Abbey Road	02:45
You Never Give Me Your Money	The Beatles	Abbey Road	04:02
Sun King	The Beatles	Abbey Road	02:26
Mean Mr Mustard	The Beatles	Abbey Road	01:06
Polythene Pam	The Beatles	Abbey Road	01:12
She Came In Through the Bathroom Window	The Beatles	Abbey Road	01:57
Golden Slumbers	The Beatles	Abbey Road	01:31
Carry The Weight	The Beatles	Abbey Road	01:36
The End	The Beatles	Abbey Road	02:05
First Movement: Allegro Con Brio	Ludwig van Beethoven	Number 3 Op. 55 Eroica	17:05
Second Movement: Marcia Funebre	Ludwig van Beethoven	Number 3 Op. 55 Eroica	16:11
Third Movement: Scherzo And Trio	Ludwig van Beethoven	Number 3 Op. 55 Eroica	05:45
Fourth Movement: Finale	Ludwig van Beethoven	Number 3 Op. 55 Eroica	12:25
Wanted Man	Johnny Cash	At San Quentin	03:24
Wreck Of The Old 97	Johnny Cash	At San Quentin	01:46
I Walk The Line	Johnny Cash	At San Quentin	03:02
Darling Companion	Johnny Cash	At San Quentin	03:25

Title	Artist	Album	Duration
Starkville City Jail	Johnny Cash	At San Quentin	03:26
San Quentin	Johnny Cash	At San Quentin	02:48
A Boy Named Sue	Johnny Cash	At San Quentin	03:45
Peace In The Valley	Johnny Cash	At San Quentin	02:50
Folsom Prison Blues	Johnny Cash	At San Quentin	02:42
Is This Love	Bob Marley	Legend	03:50
No Woman No Cry	Bob Marley	Legend	07:08
Could You Be Loved	Bob Marley	Legend	03:57
Three Little Birds	Bob Marley	Legend	03:00
Buffalo Soldier	Bob Marley	Legend	04:18
Get Up Stand Up	Bob Marley	Legend	03:17
Stir It Up	Bob Marley	Legend	05:30
One Love	Bob Marley	Legend	02:52
I Shot The Sheriff	Bob Marley	Legend	04:40
Waiting In Vain	Bob Marley	Legend	04:16
Redemption Song	Bob Marley	Legend	03:48
Satisfy My Soul	Bob Marley	Legend	04:31
Exodus	Bob Marley	Legend	07:40
Jamming	Bob Marley	Legend	03:31
Miss You	The Rolling Stones	Some Girls	04:48
When The Whip Comes Down	The Rolling Stones	Some Girls	04:20
Just My Imagination	The Rolling Stones	Some Girls	04:38
Some Girls	The Rolling Stones	Some Girls	04:36
Lies	The Rolling Stones	Some Girls	03:11
Far Away Eyes	The Rolling Stones	Some Girls	04:24
Respectable	The Rolling Stones	Some Girls	03:06
Before They Make Me Run	The Rolling Stones	Some Girls	03:25
Beast Of Burden	The Rolling Stones	Some Girls	04:25
Shattered	The Rolling Stones	Some Girls	03:48
Rainy Day Women	Bob Dylan	Greatest Hits	04:40
Blowin' In The Wind	Bob Dylan	Greatest Hits	02:51
The Times They Are A-Changin	Bob Dylan	Greatest Hits	03:16
It Ain't Me Babe	Bob Dylan	Greatest Hits	03:38
Like A Rolling Stone	Bob Dylan	Greatest Hits	06:12
Mr Tambourine Man	Bob Dylan	Greatest Hits	05:31
Subterranean Homesick Blues	Bob Dylan	Greatest Hits	02:22
I Want You	Bob Dylan	Greatest Hits	03:09
Positively 4th Street	Bob Dylan	Greatest Hits	04:12
Just Like A Woman	Bob Dylan	Greatest Hits	04:53
The Changeling	The Doors	LA Woman	04:20
Love Her Madly	The Doors	LA Woman	03:18
Been Down So Long	The Doors	LA Woman	04:40
Cars Hiss By My Window	The Doors	LA Woman	04:10
LA Woman	The Doors	LA Woman	07:49
L'America	The Doors	LA Woman	04:35
Hyacinth House	The Doors	LA Woman	03:10
Crawling King Snake	The Doors	LA Woman	04:57
The WASP	The Doors	LA Woman	04:12
Riders On The Storm	The Doors	LA Woman	07:14
Mojo Pin	Jeff Buckley	Grace	05:42
Grace	Jeff Buckley	Grace	05:22
Last Goodbye	Jeff Buckley	Grace	04:35
Lilac Wine	Jeff Buckley	Grace	04:32
So Real	Jeff Buckley	Grace	04:43
Hallelujah	Jeff Buckley	Grace	06:53
Lover, You Should've Come Over	Jeff Buckley	Grace	06:43
Corpus Christi Carol	Jeff Buckley	Grace	02:25
Eternal Life	Jeff Buckley	Grace	04:52
Dream Brother	Jeff Buckley	Grace	05:26

A.10 Summary

This appendix provided the details of the test data used during the empirical analysis. All the benchmarked songs in the eight genres were listed in order to allow future researches to replicate the results provided in this thesis.

Appendix B

Parameter Configurations

This appendix provides the optimal parameter configurations that were used for the empirical analysis. Fractional factorial design was employed to obtain the optimal parameters. Ten songs in each of the eight genres were used for parameter tuning and the entire set of 800 songs was used to acquire the final empirical results. The parameters that performed best over all genres were chosen as the optimal configurations. The appendix is divided into the optimal parameters for noise detection and interpolation purposes.

B.1 Noise Detection Parameters

Table B.1 contains the optimal parameters for the outlier detection algorithms discussed in chapter 4.

Table B.1: The optimal parameter configurations for the noise detection algorithms.

Algorithm	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5
SS	Threshold: 3.28935	Window Size: 1868			
MAD	Threshold: 3.82218	Window Size: 1600			
MHD	Threshold: 3.27452	Window Size: 2248			
NND	Threshold: 0.68025	Window Size: 324			
MASD	Threshold: 8.09458	Window Size: 64	Start: 1102 Hz	End: 5512 Hz	
APD-STP	Threshold: 0.17881	Window Size: 48	Degree: 1		
APD-OSP	Threshold: 0.17603	Window Size: 48	Degree: 2	Derivatives: 1	
APD-FOP	Threshold: 0.19603	Window Size: 64	Degree: 1		
APD-OFP	Threshold: 0.19728	Window Size: 64	Degree: 2	Derivatives: 1	
APD-NEP	Threshold: 0.15946	Window Size: 2			
APD-LAP	Threshold: 0.17414	Window Size: 2			
APD-HEP	Threshold: 0.54612	Window Size: 2			
APD-AR	Threshold: 0.18060	Window Size: 64	Degree: 2		
APD-MA	Threshold: 0.19028	Window Size: 204	Degree: 1		

Algorithm	Parameter 1	Parameter 2	Parameter 3	Parameter 4	Parameter 5
APD-ARMA	Threshold: 0.18055	Window Size: 64	AR Degree: 2	MA Degree: 2	
APD-ARIMA	Threshold: 0.18160	Window Size: 48	AR Degree: 2	I Degree: 1	MA Degree: 1
APD-ARCH	Threshold: 0.20162	Window Size: 8	ARCH Degree: 1		
APD-GARCH	Threshold: 0.20162	Window Size: 8	ARCH Degree: 1	GARCH Degree: 1	
APD-TDANN	Threshold: 0.07188	Input Neurons: 12	Learning Rate: 0.1		
APD-SRTDANN	Threshold: 0.13375	Input Neurons: 17	Learning Rate: 0.1		

B.2 Interpolation Parameters

Table B.2 shows the optimal parameters for the interpolation algorithms presented in chapter 3 and chapter 5.

Table B.2: The optimal parameter configurations for the interpolation algorithms.

Algorithm	Parameter 1	Parameter 2	Parameter 3	Parameter 4
NNI	Window Size: 2			
SI	Window Size: 284	Similar Samples: 210		
LI	Window Size: 28			
STP	Window Size: 2	Degree: 1		
OSP	Window Size: 6	Degree: 2	Derivatives: 1	
SPS	Window Size: 8	Degree: 1		
FOP	Window Size: 250	Degree: 1		
OFP	Window Size: 270	Degree: 10	Derivatives: 9	
FPS	Window Size: 2	Degree: 1		
NEP	Window Size: 2			
LAP	Window Size: 2			
HEP	Window Size: 2			
AR	Window Size: 1088	Degree: 9		
MA	Window Size: 4	Degree: 1		
ARMA	Window Size: 1440	AR Degree: 9	MA Degree: 2	
ARIMA	Window Size: 1312	AR Degree: 9	I Degree: 1	MA Degree: 4
ARCH	Window Size: 8	ARCH Degree: 1		
GARCH	Window Size: 8	ARCH Degree: 1	GARCH Degree: 1	
FI-TDANN	Input Neurons: 832	Learning Rate: 0.1		
BI-TDANN	Input Neurons: 832	Learning Rate: 0.1		
FI-SRTDANN	Input Neurons: 961	Learning Rate: 0.1		
BI-SRTDANN	Input Neurons: 961	Learning Rate: 0.1		
FSB-TDANN	Input Neurons: 224	Training Epochs 50	Max. Patterns: 256	Pattern Delay: 6
BSB-TDANN	Input Neurons: 224	Training Epochs 50	Max. Patterns: 256	Pattern Delay: 6
FCB-TDANN	Input Neurons: 220	Training Epochs 50	Max. Patterns: 256	Pattern Delay: 8
BCB-TDANN	Input Neurons: 220	Training Epochs 50	Max. Patterns: 256	Pattern Delay: 8
IB-TDANN	Input Neurons: 256	Training Epochs 50	Max. Patterns: 768	Pattern Delay: 8

B.3 Summary

This appendix provided the optimal parameter configurations of the noise detection and interpolation algorithms examined in this thesis. These parameters can be used to reproduce the empirical results provided in chapter 7, appendix C and appendix D.

Appendix C

Noise Detection Results

This appendix provides a detailed outline of the noise detection results for the algorithms discussed in chapter 4. All algorithms were benchmarked with the optimal parameters given in table B.1. Firstly, the algorithms are compared according to their sensitivity for an increasing noise duration. The outlier detectors are then evaluated according to their detection accuracy, sensitivity, specificity and standard deviation between various genres.

C.1 Noise Detection for Different Durations

Table C.1 shows the sensitivity of the proximity and spectral outlier detectors for an increasing noise duration. Table C.2 and table C.3 list the corresponding sensitivity for the predictive outlier detection using polynomials and time series models respectively. The best performance for each noise duration is given in bold. The ARCH and GARCH predictive algorithms have the highest sensitivity for noise durations of one through to seven, ten and eleven samples. The APD-LAP performs best for multivariate noise of eight and nine samples. The NND works best for noise durations of twelve samples and longer.

Table C.1: The noise detection sensitivity of the proximity and spectral approaches for an increasing noise duration.

Duration	SS	MAD	MHD	NND	MASD
1	0.420298	0.374375	0.402291	0.496964	0.003006
2	0.759685	0.679219	0.762981	0.909732	0.012788

Duration	SS	MAD	MHD	NND	MASD
3	0.761032	0.682373	0.764518	0.910197	0.015396
4	0.704125	0.631490	0.707150	0.843917	0.023852
5	0.704366	0.631120	0.706459	0.844095	0.029533
6	0.739033	0.663367	0.741987	0.886612	0.122132
7	0.734672	0.660232	0.738446	0.886330	0.140845
8	0.708090	0.637445	0.712928	0.803063	0.124675
9	0.706984	0.636164	0.711384	0.803243	0.207968
10	0.718087	0.646179	0.723116	0.815939	0.233068
11	0.717247	0.646490	0.722066	0.816049	0.249671
12	0.706973	0.639992	0.712153	**0.813846**	0.285674
13	0.705809	0.638659	0.710984	**0.814038**	0.306277
14	0.707775	0.639886	0.712701	**0.817657**	0.335267
15	0.707012	0.640681	0.712181	**0.817981**	0.334435
16	0.701591	0.640096	0.707473	**0.826685**	0.325452
17	0.699296	0.637864	0.703933	**0.826435**	0.308720
18	0.695957	0.633855	0.701719	**0.817837**	0.314535
19	0.696610	0.635517	0.702897	**0.817766**	0.318458
20	0.696756	0.639219	0.702004	**0.817896**	0.329134
21	0.695353	0.638816	0.700576	**0.818294**	0.349178
22	0.691972	0.636388	0.696908	**0.811926**	0.354919
23	0.689118	0.634340	0.694385	**0.812049**	0.381653
24	0.691645	0.636306	0.696356	**0.817882**	0.378783
25	0.692169	0.637005	0.696142	**0.817626**	0.398737
26	0.686907	0.634270	0.692038	**0.811837**	0.400832
27	0.684614	0.634138	0.689939	**0.812254**	0.400179
28	0.687029	0.637789	0.691844	**0.821299**	0.402307
29	0.685069	0.636330	0.689560	**0.821299**	0.405193
30	0.677707	0.632046	0.683891	**0.816308**	0.408625
31	0.676828	0.631745	0.683116	**0.816232**	0.398887
32	0.680401	0.636424	0.686457	**0.821795**	0.401360
33	0.678730	0.636840	0.684821	**0.821888**	0.403911
34	0.674419	0.633044	0.679280	**0.817361**	0.406462
35	0.675432	0.634153	0.679970	**0.817743**	0.406016
36	0.678801	0.636695	0.682895	**0.826523**	0.407881
37	0.676734	0.635729	0.680471	**0.826409**	0.408549
38	0.671220	0.633236	0.674842	**0.821671**	0.410246
39	0.669961	0.631758	0.673418	**0.821873**	0.414316
40	0.670874	0.633750	0.674633	**0.824214**	0.416025
41	0.669112	0.633794	0.671758	**0.824218**	0.419248
42	0.664075	0.631198	0.667428	**0.820630**	0.420911
43	0.661903	0.631142	0.666512	**0.820597**	0.422901
44	0.662969	0.633119	0.669067	**0.822345**	0.424427
45	0.663179	0.632663	0.668095	**0.822420**	0.422228
46	0.658464	0.628823	0.662251	**0.819039**	0.423557
47	0.655343	0.627340	0.659575	**0.819207**	0.419018
48	0.660740	0.633337	0.664596	**0.826148**	0.425546
49	0.666130	0.634764	0.664848	**0.826259**	0.461071
50	0.666130	0.634764	0.664848	**0.826259**	0.461071
Average	0.681131	0.636569	0.685187	**0.824754**	0.384057

Table C.2: The noise detection sensitivity of the polynomial predictive approach for an increasing noise duration.

Duration	APD-STP	APD-OSP	APD-FOP	APD-OFP	APD-NEP	APD-LAP	APD-HEP
1	0.440863	0.461006	0.498810	0.498950	0.444583	0.453839	0.066696
2	0.806456	0.843320	0.911730	0.913092	0.795918	0.817468	0.476721
3	0.815072	0.847514	0.912553	0.913639	0.816104	0.825098	0.532744
4	0.756463	0.785017	0.845752	0.846109	0.761052	0.770212	0.533303
5	0.761081	0.785936	0.845956	0.845985	0.767471	0.781483	0.593304
6	0.802729	0.827253	0.888333	0.889060	0.809768	0.824468	0.598641

Duration	APD-STP	APD-OSP	APD-FOP	APD-OFP	APD-NEP	APD-LAP	APD-HEP
7	0.805263	0.826655	0.887598	0.888483	0.813070	0.831571	0.572766
8	0.748889	0.765337	0.814422	0.815075	0.721846	**0.828770**	0.559443
9	0.752255	0.765436	0.814599	0.815097	0.723765	**0.831264**	0.562922
10	0.764936	0.777349	0.827522	0.828240	0.714905	0.815327	0.561198
11	0.768443	0.779403	0.827582	0.828554	0.715657	0.817102	0.584525
12	0.755683	0.767038	0.796733	0.794788	0.704723	0.734714	0.579104
13	0.759902	0.768217	0.796533	0.794687	0.706928	0.736522	0.564475
14	0.759765	0.768125	0.793172	0.790415	0.707861	0.736341	0.578828
15	0.762580	0.769375	0.793310	0.790574	0.708645	0.737344	0.579461
16	0.764512	0.768071	0.797791	0.795607	0.701290	0.720667	0.540424
17	0.766617	0.768285	0.797475	0.794948	0.700992	0.721476	0.555212
18	0.762950	0.764470	0.793646	0.791500	0.701390	0.722481	0.554708
19	0.765226	0.765928	0.794004	0.791940	0.702327	0.723016	0.544193
20	0.774384	0.773273	0.799371	0.795029	0.705278	0.725074	0.527555
21	0.775536	0.774216	0.799291	0.794876	0.705263	0.725475	0.527512
22	0.770249	0.769549	0.796114	0.792180	0.706184	0.726854	0.524134
23	0.771498	0.769931	0.795774	0.791954	0.706477	0.727333	0.534853
24	0.776982	0.774596	0.796506	0.796955	0.706157	0.727666	0.527899
25	0.777940	0.775008	0.796272	0.797023	0.706717	0.727969	0.519724
26	0.772696	0.770463	0.792765	0.792705	0.707131	0.727248	0.525855
27	0.773641	0.771790	0.792609	0.792500	0.707926	0.727905	0.525803
28	0.780173	0.777233	0.795758	0.795859	0.706150	0.726552	0.523379
29	0.780454	0.778177	0.795906	0.795924	0.705780	0.726589	0.531764
30	0.776382	0.774459	0.792801	0.792679	0.704610	0.725181	0.529785
31	0.776873	0.774734	0.792775	0.792761	0.704853	0.725144	0.523114
32	0.783899	0.780867	0.798682	0.796488	0.705050	0.726586	0.528095
33	0.784934	0.781685	0.798874	0.796441	0.705103	0.726942	0.527974
34	0.779863	0.776962	0.795878	0.793457	0.703590	0.725353	0.525836
35	0.780852	0.778607	0.796020	0.793487	0.704434	0.725652	0.532616
36	0.788577	0.785320	0.797967	0.798685	0.705967	0.726564	0.531380
37	0.788189	0.784868	0.797437	0.797828	0.705674	0.726948	0.526027
38	0.783743	0.781033	0.794394	0.794552	0.704971	0.726401	0.529648
39	0.784251	0.781754	0.794510	0.794854	0.704526	0.726482	0.529416
40	0.790590	0.786564	0.796401	0.796043	0.703945	0.724941	0.527505
41	0.791208	0.787138	0.796513	0.796200	0.704157	0.724786	0.533074
42	0.787216	0.783725	0.793990	0.793545	0.703011	0.724008	0.531580
43	0.787115	0.783293	0.793528	0.793132	0.703022	0.724350	0.526938
44	0.791644	0.784291	0.795088	0.794164	0.704105	0.724471	0.526171
45	0.792426	0.785423	0.795250	0.794284	0.704963	0.724074	0.525956
46	0.789169	0.782604	0.793036	0.792448	0.705114	0.724192	0.524626
47	0.789028	0.782613	0.792272	0.791923	0.703871	0.724300	0.529554
48	0.794765	0.789929	0.797484	0.796752	0.710222	0.731298	0.532696
49	0.794997	0.790765	0.797679	0.796528	0.710296	0.731685	0.529064
50	0.794997	0.790765	0.797679	0.796528	0.710296	0.731685	0.529064
Average	0.784893	0.783452	0.802095	0.800950	0.712309	0.736020	0.535626

Table C.3: The noise detection sensitivity of the time series models and ANNs predictive approach for an increasing noise duration.

Duration	APD-AR	APD-MA	APD-ARMA	APD-ARIMA	APD-ARCH	APD-GARCH	APD-TDANN	APD-SRTDANN
1	0.454464	0.461428	0.454909	0.440774	**0.499881**	**0.499881**	0.111071	0.008393
2	0.844119	0.842435	0.845093	0.797919	**0.915240**	**0.915240**	0.639722	0.583641
3	0.867832	0.847097	0.867055	0.828868	**0.915560**	**0.915560**	0.734566	0.705442
4	0.806982	0.784123	0.806668	0.778650	**0.848801**	**0.848801**	0.705382	0.670854
5	0.815690	0.783072	0.815346	0.792500	**0.848905**	**0.848905**	0.749936	0.711214
6	0.862928	0.822642	0.862567	0.841531	**0.891541**	**0.891541**	0.793334	0.767556
7	0.866247	0.820184	0.866323	0.848261	**0.891124**	**0.891124**	0.792644	0.788262
8	0.800005	0.753212	0.800501	0.786502	0.817693	0.817693	0.722189	0.754531
9	0.802731	0.753182	0.803333	0.791270	0.817836	0.817836	0.735025	0.766187
10	0.816436	0.764937	0.816452	0.805315	**0.830864**	**0.830864**	0.758747	0.785960

Duration	APD-AR	APD-MA	APD-ARMA	APD-ARIMA	APD-ARCH	APD-GARCH	APD-TDANN	APD-SRTDANN
11	0.818361	0.765731	0.818312	0.808070	**0.830994**	**0.830994**	0.761885	0.793206
12	0.807956	0.742753	0.808179	0.801568	0.796872	0.796872	0.727041	0.723900
13	0.809272	0.742598	0.809359	0.803557	0.796637	0.796637	0.732275	0.728911
14	0.808494	0.740640	0.808631	0.802347	0.792087	0.792087	0.736262	0.737853
15	0.809440	0.741177	0.809601	0.803782	0.792006	0.792006	0.738087	0.742245
16	0.808867	0.743097	0.808741	0.797679	0.795982	0.795982	0.728864	0.725893
17	0.809260	0.742930	0.809124	0.798229	0.795798	0.795798	0.736486	0.730126
18	0.805480	0.739302	0.805309	0.794339	0.792360	0.792360	0.738277	0.736514
19	0.806186	0.740460	0.806174	0.794786	0.792560	0.792560	0.738054	0.739297
20	0.807459	0.746784	0.807260	0.797940	0.794182	0.794182	0.735991	0.732654
21	0.807905	0.746239	0.807404	0.798877	0.794090	0.794090	0.741125	0.735423
22	0.805147	0.743352	0.804932	0.796288	0.791315	0.791315	0.743072	0.739303
23	0.805498	0.742897	0.805425	0.796665	0.791197	0.791197	0.742370	0.740696
24	0.808266	0.746061	0.808177	0.802363	0.795330	0.795330	0.743581	0.735580
25	0.808746	0.745917	0.808694	0.802789	0.795455	0.795455	0.744726	0.736371
26	0.806036	0.741918	0.805893	0.799887	0.791909	0.791909	0.744748	0.739860
27	0.806166	0.742032	0.805839	0.799969	0.791717	0.791717	0.745411	0.740988
28	0.809597	0.746608	0.809186	0.801048	0.795406	0.795406	0.744988	0.735384
29	0.809888	0.747787	0.809616	0.801439	0.795523	0.795523	0.747901	0.737258
30	0.807550	0.744372	0.807425	0.798992	0.792591	0.792591	0.747819	0.740110
31	0.807861	0.744186	0.807827	0.799021	0.792616	0.792616	0.748259	0.741630
32	0.810265	0.748593	0.810221	0.804642	0.796498	0.796498	0.750317	0.737656
33	0.810348	0.748792	0.810187	0.804750	0.796544	0.796544	0.752059	0.738885
34	0.808313	0.745378	0.808147	0.802329	0.793802	0.793802	0.751711	0.740658
35	0.808494	0.746502	0.808260	0.802397	0.793843	0.793843	0.753024	0.740973
36	0.809497	0.751400	0.809226	0.804397	0.797835	0.797835	0.755837	0.739168
37	0.809586	0.750549	0.809407	0.804112	0.797288	0.797288	0.756523	0.739738
38	0.807894	0.747340	0.807431	0.801952	0.794457	0.794457	0.755318	0.740899
39	0.807980	0.747922	0.807675	0.802096	0.794563	0.794563	0.755129	0.742039
40	0.807597	0.751070	0.807427	0.801660	0.793909	0.793909	0.755339	0.739275
41	0.807658	0.750748	0.807453	0.801558	0.794022	0.794022	0.757131	0.740205
42	0.805977	0.747977	0.805895	0.799965	0.791965	0.791965	0.756509	0.741367
43	0.805960	0.747883	0.805701	0.799080	0.791930	0.791930	0.755891	0.742584
44	0.802528	0.748829	0.801864	0.798023	0.793420	0.793420	0.756125	0.740033
45	0.802569	0.749354	0.801969	0.798534	0.793508	0.793508	0.757974	0.740762
46	0.800126	0.747267	0.800067	0.796351	0.791310	0.791310	0.756912	0.742199
47	0.800020	0.746773	0.799937	0.795385	0.790927	0.790927	0.756671	0.741860
48	0.805702	0.752618	0.805891	0.794122	0.798509	0.798509	0.760296	0.744676
49	0.805496	0.753590	0.805638	0.793757	0.798537	0.798537	0.760980	0.744628
50	0.805496	0.753590	0.805638	0.793757	0.798537	0.798537	0.760980	0.744628
Average	0.811191	0.752519	0.822540	0.803464	0.801142	0.801142	0.753882	0.744223

C.2 Noise Detection for Different Genres

Table C.4 shows the noise detection accuracy (MCC) of the evaluated outlier detection algorithms for different genres. The corresponding sensitivity and specificity for the various genres is given in table C.5 and table C.6 respectively. The MAD performed best for classical music. Noise in country and electronic music was most accurately detected using the APD-ARIMA and APD-SRTDANN respectively. Outliers in jazz music were best detected using the MHD, whereas the APD-LAP had the leading performance in the metal, pop and reggae genres. The APD-OFP had the highest detection accuracy in

rock. The best performing algorithm for each genre is given in bold.

Table C.4: The noise detection accuracy (MCC) for different genres.

Algorithm	Classical	Country	Electro	Jazz	Metal	Pop	Reggae	Rock
SS	0.951926	0.873877	0.607356	0.886052	0.707332	0.724012	0.713421	0.875098
MAD	**0.962271**	0.824588	0.537702	0.822593	0.658377	0.653664	0.627439	0.839129
MHD	0.955160	0.878473	0.616193	**0.889799**	0.714632	0.734291	0.723507	0.878881
NND	0.909140	0.829856	0.602421	0.845456	0.709829	0.614396	0.668494	0.851308
MASD	0.608033	0.598760	0.521788	0.595016	0.541185	0.492498	0.512974	0.595583
APD-STP	0.890176	0.822822	0.746661	0.852363	0.787955	0.750420	0.697524	0.866243
APD-OSP	0.873373	0.837015	0.737313	0.861959	0.791422	0.734935	0.724041	0.869901
APD-FOP	0.883528	0.853576	0.767727	0.865233	0.811626	0.763926	0.747050	0.854538
APD-OFP	0.895254	0.860299	0.763816	0.859296	0.806712	0.765350	0.727298	**0.883528**
APD-NEP	0.882419	0.856964	0.761798	0.851209	0.825618	0.801457	0.791315	0.853573
APD-LAP	0.884826	0.863891	0.790151	0.855927	**0.843132**	**0.819096**	**0.804737**	0.860507
APD-HEP	0.917257	0.660361	0.516832	0.652523	0.631021	0.528816	0.600382	0.665352
APD-AR	0.954877	0.879215	0.776779	0.878765	0.808909	0.776253	0.756497	0.852870
APD-MA	0.874753	0.842858	0.592617	0.848724	0.707013	0.621438	0.660757	0.826653
APD-ARMA	0.954944	0.877964	0.776087	0.879280	0.806389	0.779852	0.753114	0.848835
APD-ARIMA	0.956025	**0.889603**	0.759254	0.879509	0.813750	0.775763	0.755577	0.863971
APD-ARCH	0.899874	0.864925	0.779141	0.863870	0.828926	0.781369	0.756995	0.864109
APD-GARCH	0.899874	0.864925	0.779141	0.863870	0.828926	0.781369	0.756995	0.864109
APD-TDANN	0.887571	0.832427	0.683407	0.830236	0.759126	0.702497	0.716537	0.831674
APD-SRTDANN	0.912702	0.837436	**0.790940**	0.832761	0.815509	0.800253	0.793073	0.835042
Average	0.897699	0.832492	0.695356	0.835722	0.759869	0.720083	0.714386	0.834045

Table C.5: The noise detection sensitivity for different genres.

Algorithm	Classical	Country	Electro	Jazz	Metal	Pop	Reggae	Rock
SS	0.911838	0.802164	0.427616	0.810507	0.575321	0.569827	0.557476	0.794302
MAD	0.936132	0.762565	0.369910	0.754064	0.518042	0.512647	0.484098	0.755094
MHD	0.917242	0.804163	0.433373	0.812101	0.579726	0.576562	0.561225	0.797105
NND	0.830590	0.830613	**0.809440**	0.830266	**0.825881**	**0.821288**	**0.820223**	0.829736
MASD	0.377569	0.380596	0.391508	0.380595	0.387199	0.387321	0.386530	0.381136
APD-STP	0.903775	0.801417	0.723307	0.788626	0.783271	0.716827	0.743508	0.818417
APD-OSP	0.907183	0.813099	0.704234	0.803653	0.767666	0.706913	0.745800	0.819069
APD-FOP	0.900288	0.811598	0.761364	0.810573	0.783216	0.753541	0.784370	0.811812
APD-OFP	0.890830	0.807908	0.768479	0.815457	0.782810	0.748966	0.791721	0.801432
APD-NEP	0.883489	0.743956	0.658701	0.743824	0.703708	0.686700	0.696121	0.741970
APD-LAP	0.894877	0.756584	0.708890	0.754653	0.733489	0.718875	0.727181	0.753613
APD-HEP	**0.962933**	0.513660	0.406255	0.509786	0.491484	0.409068	0.491687	0.500131
APD-AR	0.958606	0.831916	0.763110	0.823202	0.773417	0.754258	0.773737	0.811286
APD-MA	0.903140	0.806733	0.608221	0.799413	0.719700	0.680518	0.700524	0.801900
APD-ARMA	0.958070	**0.840038**	0.751977	**0.844596**	0.780815	0.769668	0.801897	**0.833256**
APD-ARIMA	0.962897	0.833799	0.747747	0.832991	0.755027	0.735295	0.764148	0.795803
APD-ARCH	0.887921	0.802530	0.779016	0.813659	0.780375	0.758427	0.785453	0.801757
APD-GARCH	0.887921	0.802530	0.779016	0.813659	0.780375	0.758427	0.785453	0.801757
APD-TDANN	0.860354	0.779407	0.676259	0.782282	0.758385	0.671351	0.714878	0.788137
APD-SRTDANN	0.886542	0.740274	0.703511	0.737808	0.728497	0.701593	0.722917	0.732645
Average	0.881110	0.763277	0.648597	0.763086	0.700420	0.671904	0.691947	0.758518

Table C.6: The noise detection specificity for different genres.

Algorithm	Classical	Country	Electro	Jazz	Metal	Pop	Reggae	Rock
SS	0.999952	0.999689	0.999681	0.999763	0.999791	0.999725	0.999681	0.999810
MAD	0.999907	0.999172	0.999133	0.999094	0.999769	0.999210	0.999063	0.999614
MHD	0.999959	0.999758	**0.999723**	**0.999819**	**0.999829**	**0.999807**	**0.999794**	0.999851
NND	**0.999973**	0.997191	0.984196	0.998319	0.989958	0.986715	0.989629	0.998220
MASD	0.999944	0.999779	0.996648	0.999702	0.997603	0.996491	0.996839	0.999705
APD-STP	0.997848	0.997877	0.996953	0.999285	0.996471	0.997661	0.994900	0.998953
APD-OSP	0.997242	0.998039	0.996979	0.999303	0.997198	0.997323	0.995713	0.999142
APD-FOP	0.997663	0.998624	0.995894	0.999127	0.997775	0.997022	0.995132	0.998755
APD-OFP	0.998265	0.998885	0.995527	0.998808	0.997594	0.997146	0.993000	**1.000000**
APD-NEP	0.997950	**0.999923**	0.998021	0.999770	0.999794	0.999497	0.999123	0.999878
APD-LAP	0.997742	0.999913	0.997771	0.999722	0.999775	0.999446	0.998946	0.999878
APD-HEP	0.998092	0.999031	0.995270	0.998783	0.998576	0.997679	0.997786	0.999308
APD-AR	0.999418	0.999238	0.996317	0.999371	0.997910	0.997421	0.995827	0.998652
APD-MA	0.997261	0.998370	0.992789	0.998795	0.995028	0.992914	0.993689	0.997999
APD-ARMA	0.999407	0.998895	0.997636	0.998805	0.997743	0.996991	0.993068	0.997685
APD-ARIMA	0.999423	0.999508	0.995984	0.999249	0.998528	0.997864	0.996189	0.999345
APD-ARCH	0.998748	0.999207	0.995864	0.999023	0.998484	0.997479	0.995499	0.999267
APD-GARCH	0.998748	0.999207	0.995864	0.999023	0.998484	0.997479	0.995499	0.999267
APD-TDANN	0.999187	0.998988	0.995050	0.998850	0.996965	0.997349	0.996768	0.998799
APD-SRTDANN	0.999419	0.999604	0.998589	0.999510	0.999278	0.999320	0.998811	0.999645
Average	0.998807	0.999045	0.996194	0.999206	0.997828	0.997227	0.996248	0.999189

Table C.7 lists the sample standard deviations of the detection MCC for each algorithm. The lower deviation indicates that outliers were detected consistently over all songs in the dataset. A higher deviation shows that the difference in the noise detection accuracy between different tracks was greater.

Table C.7: The sample standard deviation of the noise detection accuracy (MCC) for different genres.

Algorithm	Overall	Classical	Country	Electro	Jazz	Metal	Pop	Reggae	Rock
SS	0.172186	0.010540	0.089910	0.171585	0.045580	0.232925	0.134128	0.131467	0.094543
MAD	0.194504	0.022404	0.107859	0.164119	0.080978	0.249580	0.141627	0.136741	0.127736
MHD	0.168352	0.010849	0.090463	0.165907	0.045625	0.227273	0.130521	0.128360	0.094616
NND	0.174360	0.004774	0.119865	0.175926	0.079717	0.187380	0.154034	0.157564	0.098061
MASD	0.083418	**0.001300**	0.015285	0.106033	0.027732	0.095755	0.101322	0.093690	0.026282
APD-STP	0.124437	0.109828	0.108410	0.117317	0.041370	0.144875	0.095515	0.128900	0.083222
APD-OSP	0.121825	0.120870	0.107796	0.118644	0.043204	0.130831	0.102895	0.122705	0.072063
APD-FOP	0.116617	0.113252	0.086946	0.139766	0.066791	0.105108	0.107369	0.127656	0.082290
APD-OFP	0.120362	0.110504	0.085447	0.140407	0.069019	0.105679	0.107098	0.130065	0.074525
APD-NEP	0.074389	0.100529	0.015880	0.114862	0.033011	0.040960	0.044999	0.065104	0.023070
APD-LAP	0.070196	0.106235	**0.011468**	0.110718	0.036378	**0.028729**	0.041525	0.066291	0.017905
APD-HEP	0.137766	0.091676	0.042119	0.113791	0.057128	0.070190	0.079390	0.078376	0.037563
APD-AR	0.111351	0.046820	0.055935	0.130685	0.048220	0.101511	0.101549	0.120324	0.087649
APD-MA	0.170093	0.119742	0.097394	0.162843	0.085998	0.166121	0.151513	0.144697	0.111871
APD-ARMA	0.108479	0.046096	0.050632	0.133931	0.051179	0.092192	0.096670	0.118502	0.073920
APD-ARIMA	0.107817	0.039847	0.040251	0.134196	0.053511	0.086478	0.093849	0.114792	0.058951
APD-ARCH	0.107119	0.082598	0.065604	0.140275	0.070998	0.088092	0.103289	0.126195	0.059377
APD-GARCH	0.107119	0.082598	0.065604	0.140275	0.070998	0.088092	0.103289	0.126195	0.059377
APD-TDANN	0.113581	0.030141	0.039986	0.152619	0.046891	0.111846	0.098700	0.106656	0.052752
APD-SRTDANN	**0.055086**	0.026180	0.012363	**0.081343**	**0.027426**	0.036886	**0.032513**	**0.054226**	**0.011208**
Overall	0.145164	0.105584	0.102411	0.168196	0.092814	0.155583	0.138505	0.135786	0.102119

C.3 Summary

This appendix provided the detailed results of the noise detection algorithms that were examined in this thesis. The outlier detectors were evaluated according to their sensitivity for an increasing multivariate noise duration, followed by a detailed report on the detection accuracy, sensitivity, specificity and standard deviation between different genres.

Appendix D

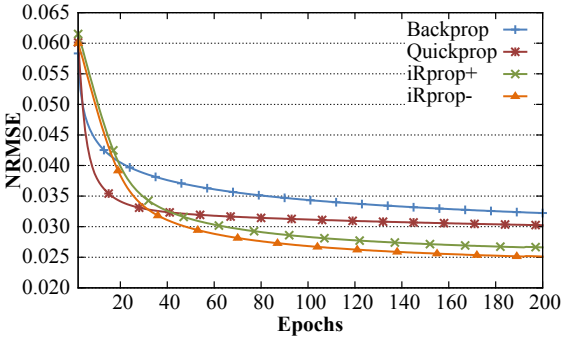
Interpolation Results

This appendix provides comprehensive results for the polynomials, models and algorithms used to interpolate music as discussed in chapter 3 and chapter 5. All algorithms were benchmarked with the optimal parameters given in table B.2. Firstly, the ANNs are analysed according to their performance using different training algorithms, activation functions, learning rates and learning momentums. The interpolation accuracy is then evaluated according to the reconstructed gap size. The different reconstruction algorithms are compared by assessing their interpolation accuracy and the standard deviations of the NRMSEs over different music genres.

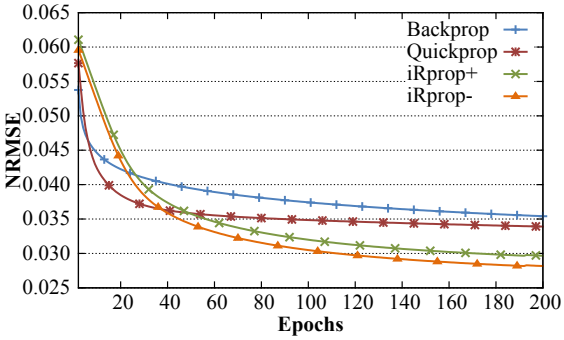
D.1 Artificial Neural Network Configurations

Figure D.1 shows the training NRMSE for the FSB-TDANN, FCB-TDANN and IB-TDANN. Standard batch backpropagation and quickprop had a lower error compared to the iRprop algorithms when trained for a short period. However, iRprop$^-$ performed best for all three ANNs when trained for 38 epochs or longer. iRprop$^-$ had a slightly lower error than iRprop$^+$ for all ANNs. The deviation in the output error between the two variates of iRprop increased as the training continued over more epochs.

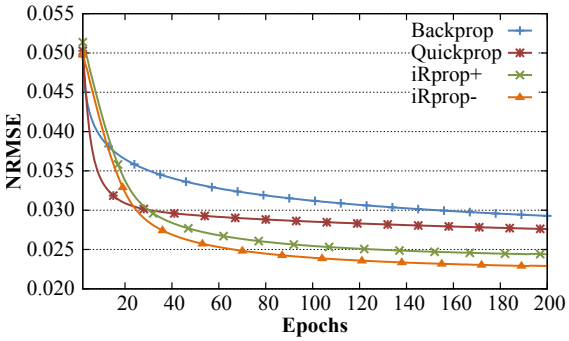
Figure D.2 illustrates the affect of the learning rate and momentum on the interpolation performance of incrementally trained ANNs. Lower rates and momentums performed better with both ANNs achieving the best learning with a rate of 0.1 and without a



(a) The training NRMSE of the FSB-TDANN for different training algorithms.

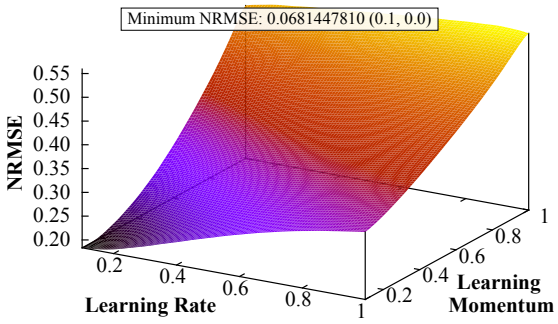


(b) The training NRMSE of the FCB-TDANN for different training algorithms.

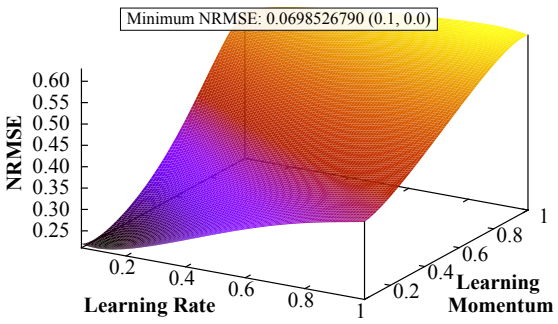


(c) The training NRMSE of the IB-TDANN for different training algorithms.

Figure D.1: The training NRMSE of the batch trained ANNs for different training algorithms.



(d) The output NRMSE of the FI-TDANN for different learning rates and momentums.



(e) The output NRMSE of the FI-SRTDANN for different learning rates and momentums.

Figure D.2: The interpolation NRMSE of the incrementally trained ANNs for different learning rates and momentums.

momentum.

Table D.1 shows the interpolation accuracy of the examined ANNs using different activation functions. The corresponding execution speed is given in table D.2. All ANNs performed best with the symmetric Elliot activation function, except the FI-SRTDANN which benefited more from the hyperbolic tangent activation function. The Elliot function was slightly faster to compute than the hyperbolic tangent function.

Table D.1: The interpolation NRMSE of the ANNs for different activation functions.

Activation Function	FI-TDANN	FI-SRTDANN	FSB-TDANN	FCB-TDANN	IB-TDANN
Symmetric Elliot	**0.071308**	0.079066	**0.080021**	**0.080626**	**0.058749**
Hyperbolic Tangent	0.075551	**0.072680**	0.080583	0.080856	0.058816
Bounded Linear	0.079348	0.074298	0.081192	0.081005	0.058820
Symmetric Gaussian	0.122094	0.106832	0.422160	0.195167	0.341095
Symmetric Sine	0.077475	0.073495	0.080711	0.080937	0.059552
Symmetric Cosine	0.105983	0.119519	0.348429	0.122637	0.257964

Table D.2: The execution speed of the ANNs for different activation functions.

Activation Function	FI-TDANN	FI-SRTDANN	FSB-TDANN	FCB-TDANN	IB-TDANN
Symmetric Elliot	64.46232	**2.502488**	52.02565	**89.62463**	100.0583
Hyperbolic Tangent	64.85891	2.546073	53.08238	94.03838	101.5186
Bounded Linear	64.11473	2.540985	**51.74124**	92.23812	**98.67043**
Symmetric Gaussian	64.66294	2.577210	53.25786	94.30679	101.2180
Symmetric Sine	**63.17457**	2.515352	53.45751	94.54052	101.9032
Symmetric Cosine	64.73592	2.532297	54.59695	95.07163	102.9752

D.2 Interpolation Accuracy for Different Gap Sizes

Table D.3 contains the interpolation results of the duplication and trigonometric interpolation algorithms. Table D.4 and table D.5 lists the performance of the various polynomials and time series models. Table D.6 and table D.7 show the reconstruction results of the nine ANNs that were examined. The best performing algorithm for each gap duration is given in bold. The averages at the bottom of the tables do not represent

the mean NRMSEs over all gap sizes, but were calculated on a per-sample basis over the entire dataset, where each sample in the signals carried the same weight. The ARIMA model was able to most accurately reconstruct gap sizes between one and four samples. The ARMA model performed best for five to six samples and the BI-SRTDANN interpolated gaps of 44, 46, and 47 samples most accurately. All other gap sizes were best reconstructed using an IB-TDANN.

Table D.3: The interpolation accuracy (NRMSE) of the duplication and trigonometric algorithms for different gap sizes.

Gap Size	AWI	MWI	NNI	SI	LI	CI
1	0.024182	0.023755	0.050233	0.024182	0.080049	0.024183
2	0.051816	0.051138	0.047004	0.040778	0.081182	0.034901
3	0.064632	0.064600	0.057035	0.049482	0.081125	0.041618
4	0.077341	0.076040	0.061598	0.056770	0.083943	0.047820
5	0.087097	0.084285	0.067258	0.062474	0.083819	0.052110
6	0.097235	0.092484	0.071411	0.068302	0.086010	0.057344
7	0.105553	0.099535	0.076535	0.073689	0.088860	0.062220
8	0.111533	0.104464	0.079820	0.078255	0.090861	0.066186
9	0.115020	0.107531	0.083350	0.081803	0.091822	0.069310
10	0.118085	0.111665	0.086383	0.085143	0.093479	0.072736
11	0.119813	0.114209	0.088269	0.087063	0.094321	0.074593
12	0.122366	0.117296	0.091073	0.089882	0.096845	0.077757
13	0.125291	0.120533	0.093741	0.091943	0.098625	0.080264
14	0.128146	0.122261	0.095665	0.094225	0.100864	0.082364
15	0.129597	0.124409	0.098478	0.095747	0.102882	0.084398
16	0.134299	0.128319	0.101192	0.098723	0.105848	0.087262
17	0.134947	0.128495	0.102619	0.099742	0.106628	0.088982
18	0.136323	0.129888	0.103508	0.100966	0.108142	0.089841
19	0.138543	0.131752	0.105399	0.102121	0.109871	0.091384
20	0.140625	0.132513	0.106617	0.103902	0.111034	0.092707
21	0.142352	0.133039	0.108848	0.105423	0.112986	0.094605
22	0.143404	0.135574	0.109953	0.106407	0.114675	0.096243
23	0.144053	0.136446	0.111039	0.107307	0.115377	0.097221
24	0.145446	0.136816	0.111570	0.108312	0.117003	0.098010
25	0.146818	0.137378	0.113138	0.109447	0.118328	0.099222
26	0.148229	0.138362	0.114765	0.111119	0.121022	0.100965
27	0.149092	0.139758	0.115365	0.112022	0.121698	0.101879
28	0.149662	0.140381	0.115803	0.112214	0.122512	0.102353
29	0.150980	0.141814	0.117992	0.114095	0.124505	0.104580
30	0.152186	0.142292	0.119020	0.115143	0.124614	0.105707
31	0.153215	0.144045	0.120501	0.116572	0.126163	0.107122
32	0.152997	0.144016	0.119657	0.115908	0.126277	0.106396
33	0.153669	0.144004	0.121361	0.117265	0.126724	0.107929
34	0.152647	0.143373	0.121020	0.116619	0.126807	0.107796
35	0.153086	0.143477	0.122576	0.118636	0.127486	0.109557
36	0.153380	0.143809	0.122934	0.119152	0.127838	0.110079
37	0.153882	0.144354	0.124074	0.119945	0.128484	0.111169
38	0.154925	0.146060	0.125415	0.120622	0.129342	0.112163
39	0.154649	0.145013	0.125764	0.121226	0.128971	0.112728
40	0.154669	0.145006	0.125635	0.121251	0.128844	0.112692
41	0.155374	0.145256	0.126961	0.121838	0.129747	0.113928
42	0.156788	0.146087	0.128186	0.123358	0.130407	0.115073
43	0.156390	0.146721	0.128209	0.123267	0.129937	0.115188
44	0.155382	0.145681	0.129469	0.124313	0.130838	0.116749
45	0.157333	0.147092	0.130278	0.124940	0.131562	0.117232
46	0.154515	0.145449	0.130409	0.124752	0.130278	0.117593

Gap Size	AWI	MWI	NNI	SI	LI	CI
47	0.156665	0.146228	0.131478	0.126028	0.131123	0.118450
48	0.155316	0.146252	0.130982	0.125759	0.130860	0.118112
49	0.155616	0.146102	0.131818	0.126474	0.131411	0.119064
50	0.154249	0.145989	0.131595	0.125987	0.130053	0.118853
Average	0.093215	0.081218	0.111371	0.104806	0.090748	0.087269

Table D.4: The interpolation accuracy (NRMSE) of the polynomials for different gap sizes.

Gap Size	STP	OSP	SPS	FOP	OFP	FPS	LAP	NEP	HEP
1	0.024185	0.050500	0.024185	0.214348	0.194620	0.076628	0.024182	0.024185	0.024182
2	0.034486	0.057235	0.034486	0.207632	0.188423	0.077827	0.034484	0.034486	0.034729
3	0.041023	0.061946	0.041023	0.201194	0.183046	0.078081	0.041021	0.041023	0.041447
4	0.047030	0.067480	0.047030	0.197992	0.181043	0.080265	0.047029	0.047030	0.047638
5	0.051387	0.070553	0.051387	0.192817	0.176477	0.081013	0.051385	0.051387	0.051936
6	0.056583	0.074853	0.056583	0.191177	0.175568	0.082617	0.056583	0.056583	0.057171
7	0.061532	0.079553	0.061532	0.188020	0.173631	0.084810	0.061532	0.061532	0.062058
8	0.065523	0.082715	0.065523	0.185570	0.171893	0.086330	0.065523	0.065523	0.066029
9	0.068714	0.084692	0.068714	0.185031	0.171646	0.087653	0.068714	0.068714	0.069161
10	0.072032	0.087350	0.072032	0.183779	0.170517	0.088919	0.072032	0.072032	0.072579
11	0.073926	0.088311	0.073926	0.181129	0.168554	0.089807	0.073925	0.073926	0.074443
12	0.076986	0.091274	0.076986	0.178969	0.166921	0.091141	0.076986	0.076986	0.077600
13	0.079408	0.092442	0.079408	0.177840	0.166642	0.092968	0.079408	0.079408	0.080102
14	0.081506	0.094422	0.081506	0.176909	0.165903	0.093410	0.081506	0.081506	0.082201
15	0.083300	0.095783	0.083300	0.176152	0.165095	0.095002	0.083300	0.083300	0.084215
16	0.086174	0.098542	0.086174	0.174222	0.163667	0.096448	0.086174	0.086174	0.087079
17	0.087826	0.099398	0.087827	0.173700	0.163559	0.097449	0.087826	0.087826	0.088797
18	0.088731	0.099982	0.088731	0.173051	0.163321	0.097896	0.088731	0.088731	0.089659
19	0.090146	0.101336	0.090146	0.171011	0.162035	0.098856	0.090146	0.090146	0.091191
20	0.091521	0.102777	0.091521	0.169743	0.160803	0.099259	0.091521	0.091521	0.092518
21	0.093283	0.103796	0.093283	0.170271	0.161911	0.100406	0.093283	0.093283	0.094404
22	0.094903	0.105395	0.094903	0.169333	0.160984	0.101525	0.094903	0.094903	0.096045
23	0.095843	0.105534	0.095843	0.167074	0.159047	0.101797	0.095843	0.095843	0.097020
24	0.096704	0.106837	0.096704	0.167587	0.159631	0.102306	0.096704	0.096704	0.097815
25	0.097820	0.107662	0.097820	0.167267	0.159357	0.102844	0.097820	0.097820	0.099019
26	0.099542	0.109055	0.099542	0.167246	0.159938	0.104350	0.099542	0.099542	0.100760
27	0.100549	0.109849	0.100549	0.165871	0.158860	0.104462	0.100549	0.100549	0.101685
28	0.100946	0.109946	0.100946	0.165109	0.158265	0.104847	0.100946	0.100946	0.102153
29	0.103112	0.111908	0.103112	0.165842	0.159017	0.106238	0.103112	0.103112	0.104377
30	0.104242	0.113070	0.104242	0.165060	0.158692	0.107090	0.104242	0.104242	0.105504
31	0.105629	0.114202	0.105629	0.164620	0.158290	0.107992	0.105629	0.105629	0.106918
32	0.104938	0.114036	0.104938	0.164139	0.158179	0.107525	0.104938	0.104938	0.106195
33	0.106399	0.114909	0.106399	0.162975	0.157185	0.108274	0.106399	0.106399	0.107722
34	0.106206	0.114261	0.106206	0.162576	0.156772	0.108118	0.106206	0.106206	0.107586
35	0.108077	0.115918	0.108077	0.163092	0.157781	0.109045	0.108078	0.108077	0.109358
36	0.108620	0.116283	0.108620	0.162619	0.157214	0.109630	0.108620	0.108620	0.109882
37	0.109662	0.117243	0.109662	0.161131	0.155822	0.110424	0.109662	0.109662	0.110969
38	0.110514	0.117716	0.110514	0.162034	0.157205	0.111304	0.110514	0.110514	0.111950
39	0.111133	0.119156	0.111133	0.160848	0.156293	0.111391	0.111134	0.111133	0.112522
40	0.111125	0.118631	0.111125	0.160379	0.155989	0.111285	0.111125	0.111125	0.112488
41	0.112224	0.119927	0.112224	0.160864	0.156576	0.112309	0.112224	0.112224	0.113713
42	0.113421	0.120852	0.113421	0.161597	0.157224	0.113024	0.113422	0.113421	0.114863
43	0.113512	0.120847	0.113512	0.160427	0.156479	0.113038	0.113512	0.113512	0.114976
44	0.115043	0.122434	0.115043	0.159884	0.155838	0.114204	0.115043	0.115043	0.116537
45	0.115499	0.122878	0.115499	0.159891	0.156381	0.114733	0.115499	0.115499	0.117017
46	0.115811	0.122250	0.115811	0.159405	0.156069	0.114580	0.115811	0.115811	0.117375
47	0.116694	0.123970	0.116694	0.160245	0.156732	0.115318	0.116694	0.116694	0.118231
48	0.116393	0.123512	0.116393	0.158635	0.155433	0.114801	0.116394	0.116393	0.117898
49	0.117338	0.123978	0.117338	0.159289	0.155984	0.115763	0.117338	0.117338	0.118850

Gap Size	STP	OSP	SPS	FOP	OFP	FPS	LAP	NEP	HEP
50	0.117086	0.123505	0.117086	0.157785	0.154695	0.115204	0.117086	0.117086	0.118636
Average	0.080057	0.086014	0.080057	0.122412	0.117923	0.081493	0.080058	0.080057	0.081066

Table D.5: The interpolation accuracy (NRMSE) of the time series models for different gap sizes.

Gap Size	AR	MA	ARMA	ARIMA	ARCH	GARCH
1	0.012090	0.038977	0.012064	**0.011472**	0.055553	0.055553
2	0.023240	0.049116	0.023224	**0.022835**	0.061829	0.061829
3	0.031401	0.055699	0.031378	**0.031183**	0.066702	0.066702
4	0.039034	0.062640	0.039000	**0.038859**	0.072819	0.072819
5	0.044383	0.067433	**0.044353**	0.044671	0.076337	0.076337
6	0.049486	0.072941	**0.049439**	0.050077	0.081198	0.081198
7	0.053896	0.078193	0.053858	0.054822	0.085909	0.085909
8	0.057950	0.082463	0.057926	0.059033	0.089290	0.089290
9	0.061695	0.085409	0.061655	0.063194	0.091275	0.091275
10	0.064818	0.088390	0.064770	0.066671	0.093568	0.093568
11	0.066804	0.089986	0.066739	0.068935	0.094672	0.094672
12	0.069088	0.092676	0.069045	0.071632	0.097093	0.097093
13	0.071763	0.094239	0.071701	0.074592	0.098116	0.098116
14	0.073821	0.096329	0.073769	0.077189	0.100111	0.100111
15	0.075696	0.097828	0.075648	0.079232	0.101377	0.101377
16	0.078213	0.100716	0.078174	0.082469	0.104248	0.104248
17	0.079126	0.101620	0.079077	0.083755	0.104862	0.104862
18	0.080961	0.102745	0.080901	0.085608	0.105747	0.105747
19	0.081593	0.103949	0.081519	0.086967	0.106924	0.106924
20	0.083082	0.105584	0.083005	0.089006	0.108355	0.108355
21	0.084626	0.106876	0.084566	0.091017	0.109341	0.109341
22	0.086224	0.107964	0.086137	0.092219	0.110434	0.110434
23	0.086151	0.108615	0.086088	0.093114	0.110782	0.110782
24	0.088078	0.109675	0.088019	0.094992	0.112087	0.112087
25	0.088494	0.110643	0.088411	0.096548	0.112849	0.112849
26	0.090165	0.112349	0.090097	0.098371	0.114525	0.114525
27	0.090628	0.113247	0.090541	0.099782	0.115256	0.115256
28	0.091265	0.113417	0.091174	0.100149	0.115479	0.115479
29	0.092474	0.115142	0.092400	0.101991	0.117028	0.117028
30	0.093698	0.116324	0.093666	0.103746	0.118083	0.118083
31	0.094632	0.117564	0.094539	0.105007	0.119198	0.119198
32	0.094713	0.116975	0.094641	0.105186	0.118821	0.118821
33	0.095714	0.118260	0.095633	0.106925	0.119712	0.119712
34	0.095660	0.117566	0.095569	0.106576	0.119004	0.119004
35	0.096353	0.119379	0.096242	0.108356	0.120578	0.120578
36	0.097311	0.119935	0.097235	0.109134	0.121140	0.121140
37	0.098398	0.120664	0.098311	0.110472	0.121756	0.121756
38	0.098979	0.121235	0.098889	0.111221	0.122387	0.122387
39	0.099129	0.122015	0.099060	0.112434	0.123282	0.123282
40	0.099813	0.121889	0.099697	0.113090	0.122975	0.122975
41	0.100543	0.122601	0.100446	0.113744	0.123731	0.123731
42	0.101371	0.124084	0.101272	0.115470	0.125192	0.125192
43	0.101177	0.123848	0.101088	0.115462	0.124839	0.124839
44	0.101810	0.124968	0.101721	0.116357	0.126017	0.126017
45	0.103121	0.125642	0.103028	0.117488	0.126815	0.126815
46	0.102979	0.125387	0.102888	0.117474	0.126400	0.126400
47	0.103650	0.126709	0.103542	0.119033	0.127756	0.127756
48	0.103371	0.126311	0.103287	0.119253	0.127218	0.127218
49	0.104235	0.127051	0.104162	0.119870	0.127981	0.127981
50	0.103905	0.126447	0.103812	0.119929	0.127119	0.127119
Average	0.071764	0.087952	0.071709	0.080201	0.089057	0.089057

Table D.6: The interpolation accuracy (NRMSE) of the incrementally trained ANNs for different gap sizes.

Gap Size	FI-TDANN	BI-TDANN	FI-SRTDANN	BI-SRTDANN
1	0.064341	0.048505	0.046167	0.029160
2	0.068988	0.056130	0.054733	0.039738
3	0.073577	0.060262	0.060756	0.046170
4	0.079014	0.064817	0.065844	0.051428
5	0.075746	0.063787	0.069009	0.054092
6	0.081003	0.068413	0.071620	0.056627
7	0.084191	0.071369	0.074539	0.059583
8	0.087173	0.074068	0.076410	0.061178
9	0.087765	0.073542	0.078175	0.062242
10	0.085225	0.072189	0.079559	0.063900
11	0.087632	0.074628	0.079980	0.063923
12	0.085693	0.072685	0.080682	0.064858
13	0.087118	0.074933	0.081087	0.065520
14	0.085492	0.071986	0.082458	0.065934
15	0.087985	0.075084	0.082872	0.066583
16	0.087178	0.074008	0.085006	0.067817
17	0.088889	0.075352	0.085146	0.068030
18	0.088731	0.075652	0.084874	0.067915
19	0.090285	0.076603	0.085237	0.067981
20	0.085545	0.072112	0.086180	0.068497
21	0.088263	0.074814	0.086260	0.068529
22	0.089625	0.075554	0.086236	0.068979
23	0.085283	0.073467	0.086134	0.068696
24	0.088811	0.076296	0.087509	0.069659
25	0.090021	0.075305	0.087773	0.069711
26	0.090130	0.075688	0.088223	0.069972
27	0.087396	0.074131	0.088238	0.070132
28	0.089771	0.075537	0.088485	0.070345
29	0.089306	0.075386	0.090092	0.071186
30	0.088711	0.074826	0.089695	0.071097
31	0.088081	0.075357	0.090586	0.071609
32	0.087434	0.074608	0.090716	0.071663
33	0.089166	0.075135	0.091750	0.072357
34	0.088673	0.075169	0.090577	0.071283
35	0.093202	0.077748	0.091893	0.072199
36	0.089253	0.075874	0.091600	0.072355
37	0.087558	0.075270	0.091290	0.072072
38	0.091033	0.076525	0.091742	0.072555
39	0.085883	0.072818	0.092287	0.072637
40	0.087458	0.074644	0.092187	0.072728
41	0.089726	0.075728	0.092963	0.073206
42	0.089169	0.075555	0.093626	0.073699
43	0.090313	0.076152	0.094099	0.073919
44	0.090979	0.075084	0.094264	**0.073545**
45	0.089909	0.075338	0.094599	0.074268
46	0.090529	0.075918	0.094153	**0.073662**
47	0.090139	0.075330	0.094951	**0.074327**
48	0.088643	0.074660	0.095126	0.074174
49	0.090683	0.075653	0.095342	0.074637
50	0.090197	0.076468	0.094757	0.074279
Average	0.068145	0.058917	0.069853	0.054953

Table D.7: The interpolation accuracy (NRMSE) of the batch trained ANNs for different gap sizes.

Gap Size	FSB-TDANN	BSB-TDANN	FCB-TDANN	BCB-TDANN	IB-TDANN
1	0.047136	0.036573	0.043489	0.035030	0.032756
2	0.058490	0.046845	0.053782	0.043340	0.036360
3	0.063257	0.050598	0.059745	0.046904	0.040966
4	0.068352	0.055034	0.066382	0.052619	0.045005
5	0.071084	0.056673	0.069896	0.054966	0.047311
6	0.074983	0.061926	0.073910	0.058921	0.049732
7	0.078467	0.063185	0.076502	0.061831	**0.052767**
8	0.079936	0.065601	0.078516	0.062752	**0.054172**
9	0.081079	0.066100	0.080136	0.065266	**0.055024**
10	0.082828	0.065365	0.081831	0.064295	**0.056675**
11	0.083932	0.068610	0.083344	0.068032	**0.058227**
12	0.083839	0.068341	0.084113	0.067688	**0.059241**
13	0.085721	0.070191	0.085582	0.068802	**0.060640**
14	0.085221	0.068420	0.085429	0.067789	**0.061084**
15	0.086061	0.069405	0.086098	0.068930	**0.061479**
16	0.088751	0.071049	0.088130	0.069989	**0.063120**
17	0.087625	0.071686	0.088387	0.071052	**0.063473**
18	0.087470	0.070771	0.087767	0.070520	**0.063625**
19	0.088075	0.072356	0.088767	0.072029	**0.064893**
20	0.088585	0.070627	0.089554	0.069538	**0.064720**
21	0.090001	0.072827	0.090135	0.072815	**0.065461**
22	0.089914	0.072707	0.090687	0.072407	**0.066153**
23	0.090468	0.073716	0.091153	0.073969	**0.066835**
24	0.090341	0.073431	0.091340	0.073607	**0.066708**
25	0.091400	0.074080	0.092161	0.074091	**0.068051**
26	0.091442	0.074412	0.091922	0.074126	**0.067322**
27	0.091023	0.073149	0.092025	0.073430	**0.067650**
28	0.091698	0.075105	0.092795	0.074671	**0.068235**
29	0.092740	0.075089	0.093912	0.074661	**0.068970**
30	0.092434	0.075130	0.093359	0.075565	**0.068849**
31	0.092664	0.075139	0.093821	0.075034	**0.069449**
32	0.093482	0.074548	0.094187	0.074494	**0.070173**
33	0.094470	0.075724	0.095431	0.076111	**0.070473**
34	0.093596	0.075719	0.094637	0.076330	**0.069810**
35	0.094278	0.077315	0.095666	0.077709	**0.071298**
36	0.094822	0.077439	0.095712	0.077622	**0.071768**
37	0.094193	0.076154	0.095773	0.076857	**0.071430**
38	0.095381	0.077380	0.095782	0.077793	**0.072156**
39	0.094370	0.076025	0.095520	0.075980	**0.072012**
40	0.093749	0.075942	0.095068	0.076415	**0.071900**
41	0.096048	0.076602	0.096819	0.076697	**0.072759**
42	0.095617	0.077073	0.097103	0.077575	**0.073146**
43	0.096827	0.077457	0.098155	0.077724	**0.073698**
44	0.096662	0.077900	0.097645	0.078616	0.073754
45	0.095486	0.079022	0.097521	0.079204	**0.074136**
46	0.097245	0.079783	0.098228	0.079394	0.074074
47	0.097731	0.077903	0.098972	0.078110	0.074391
48	0.095561	0.076980	0.096798	0.077210	**0.073205**
49	0.096952	0.078424	0.098065	0.079276	**0.074027**
50	0.096029	0.078156	0.096948	0.078570	**0.073468**
Average	0.071659	0.058147	0.072788	0.058637	**0.054247**

D.3 Interpolation Accuracy for Different Genres

Table D.8 shows the interpolation accuracy of different algorithms for the eight genres examined in this thesis. Amongst all algorithms, the IB-TDANN interpolated all genres most accurately, except for metal and rock music, which was best reconstructed using a BI-SRTDANN. The lowest interpolation NRMSE for each genre is given in bold.

Table D.8: The interpolation accuracy (NRMSE) for different genres.

Algorithm	Classical	Country	Electro	Jazz	Metal	Pop	Reggae	Rock
AWI	0.094831	0.112448	0.119919	0.117352	0.112000	0.111454	0.122570	0.100393
MWI	0.082085	0.100996	0.114205	0.105853	0.107922	0.105229	0.110541	0.094273
NNI	0.065920	0.085024	0.105180	0.087931	0.103765	0.094996	0.098086	0.085083
SI	0.067175	0.083507	0.099116	0.086747	0.095450	0.090004	0.095424	0.080727
LI	0.064791	0.085723	0.111587	0.089531	0.105977	0.099367	0.101492	0.087248
CI	0.058249	0.075599	0.094691	0.077718	0.093754	0.085322	0.088126	0.076281
STP	0.057997	0.074762	0.093071	0.076897	0.091734	0.083901	0.087085	0.075014
OSP	0.065010	0.082054	0.097175	0.084630	0.096641	0.088736	0.093376	0.080489
SPS	0.057997	0.074762	0.093071	0.076897	0.091734	0.083901	0.087085	0.075014
FOP	0.085755	0.118242	0.141207	0.123283	0.137256	0.132658	0.122470	0.118423
OFP	0.084521	0.113369	0.136857	0.118546	0.130239	0.126993	0.120449	0.112408
FPS	0.056885	0.074597	0.097833	0.077936	0.093035	0.086956	0.088596	0.076109
LAP	0.057997	0.074762	0.093071	0.076897	0.091734	0.083901	0.087085	0.075014
NEP	0.057997	0.074762	0.093071	0.076897	0.091734	0.083901	0.087085	0.075014
HEP	0.058174	0.075469	0.094503	0.077584	0.093540	0.085153	0.087980	0.076127
MA	0.068295	0.084632	0.099053	0.087845	0.095938	0.090286	0.096153	0.081411
AR	0.044684	0.063066	0.089517	0.063907	0.083930	0.079821	0.079634	0.069552
ARMA	0.044558	0.063004	0.089505	0.063807	0.083903	0.079800	0.079607	0.069484
ARIMA	0.057269	0.074376	0.096327	0.075326	0.087477	0.085975	0.090019	0.074839
ARCH	0.070394	0.086568	0.099060	0.089795	0.096041	0.090729	0.097582	0.082284
GARCH	0.070394	0.086568	0.099060	0.089795	0.096041	0.090729	0.097582	0.082284
FI-TDANN	0.031090	0.051052	0.090200	0.049207	0.093168	0.091573	0.073144	0.065724
BI-TDANN	0.026660	0.042709	0.079468	0.041094	0.080332	0.081278	0.063294	0.056501
FI-SRTDANN	0.031740	0.052813	0.095891	0.051151	0.092425	0.089477	0.081454	0.063870
BI-SRTDANN	0.024608	0.041309	0.075553	0.039394	**0.074140**	0.069712	0.063043	**0.051867**
FSB-TDANN	0.032444	0.055585	0.096158	0.053605	0.096645	0.089199	0.075290	0.074350
BSB-TDANN	0.026225	0.044085	0.080297	0.041578	0.080555	0.073196	0.060170	0.059070
FCB-TDANN	0.031616	0.055558	0.099710	0.052884	0.098522	0.092059	0.076620	0.075334
BCB-TDANN	0.025598	0.043566	0.083219	0.040344	0.081443	0.075228	0.060567	0.059130
IB-TDANN	**0.021309**	**0.040269**	**0.074772**	**0.038974**	0.074652	**0.069089**	**0.058380**	0.056536
Average	0.054076	0.073041	0.097745	0.074447	0.095058	0.089687	0.087666	0.076995

Table D.9 lists the sample standard deviation of the interpolation NRMSEs. A lower deviation indicates that the algorithm interpolated more consistently over all tracks in the dataset. A higher deviation shows that some songs were substantially easier to reconstruct than others. The lowest standard deviation for each genre is given in bold.

Table D.9: The sample standard deviation of the interpolation accuracy (NRMSE) for different genres.

Algorithm	Overall	Classical	Country	Electro	Jazz	Metal	Pop	Reggae	Rock
AWI	0.027830	0.021977	0.025992	0.025797	0.027925	0.034479	0.023339	0.027803	0.022458
MWI	0.025612	0.019451	0.022385	0.023925	0.024159	0.032351	0.021383	0.024223	0.020598
NNI	0.027156	0.018514	0.020438	0.024590	0.025018	0.035671	0.022596	0.022999	0.022331
SI	0.024434	0.017924	0.019646	0.022185	0.022904	0.032031	0.020756	0.022229	0.019842
LI	0.026271	0.016855	0.019174	0.021913	0.021426	0.032989	0.021531	0.021475	0.020751
CI	0.025435	0.017649	0.019010	0.022818	0.023978	0.033301	0.021079	0.021148	0.021051
STP	0.024845	0.017472	0.018748	0.022334	0.023524	0.032539	0.020660	0.020905	0.020517
OSP	0.025207	0.018437	0.020034	0.021958	0.024279	0.033520	0.021538	0.022165	0.020974
SPS	0.024845	0.017472	0.018748	0.022334	0.023524	0.032539	0.020660	0.020905	0.020517
FOP	0.031732	0.020708	0.028263	0.029136	0.026987	0.034315	0.025310	0.026830	0.026586
OFP	0.029610	0.019942	0.026099	0.027087	0.024894	0.032453	0.023675	0.025453	0.024462
FPS	0.023598	0.015326	0.016937	0.020754	0.019538	0.029945	0.018893	0.019331	0.018587
NEP	0.024845	0.017472	0.018748	0.022334	0.023524	0.032539	0.020660	0.020905	0.020517
LAP	0.024845	0.017472	0.018748	0.022334	0.023524	0.032539	0.020660	0.020905	0.020517
HEP	0.025381	0.017630	0.018981	0.022773	0.023938	0.033230	0.021038	0.021117	0.021004
AR	**0.023160**	0.013137	0.014368	0.020051	0.018080	0.027782	0.018036	0.018227	0.017639
MA	0.024475	0.018163	0.019910	0.021891	0.023203	0.032223	0.020992	0.022503	0.019955
ARMA	0.023178	0.013130	0.014350	0.020048	0.018057	0.027775	0.018041	0.018232	0.017648
ARIMA	0.023740	0.014976	0.017033	0.021513	0.020492	0.029226	0.020192	0.021518	0.018992
ARCH	0.024445	0.018577	0.020361	0.021629	0.023599	0.032207	0.021148	0.022908	0.019935
GARCH	0.024445	0.018577	0.020361	0.021629	0.023599	0.032207	0.021148	0.022908	0.019935
FI-TDANN	0.029736	0.009670	0.013119	0.015460	0.016634	0.033871	0.021230	0.023098	0.021299
BI-TDANN	0.025292	0.008247	0.010688	0.013068	0.013821	0.028632	0.017259	0.018722	0.017926
FI-SRTDANN	0.033467	0.013758	0.022707	0.025079	0.023273	0.038085	0.024917	0.025082	0.023359
BI-SRTDANN	0.025194	0.008713	0.012190	0.020789	**0.012168**	0.031315	0.018042	0.016631	0.016623
FSB-TDANN	0.029915	0.012148	0.013994	0.022367	0.016567	0.035695	0.018239	0.019075	0.021185
BSB-TDANN	0.023480	**0.003864**	0.007748	0.011416	0.014179	**0.017373**	0.015536	0.022611	**0.015321**
FCB-TDANN	0.032014	0.012998	0.014479	0.028073	0.017276	0.036482	0.019136	0.019281	0.022339
BCB-TDANN	0.024168	0.003866	**0.007552**	**0.011044**	0.014780	0.017768	0.015851	0.023013	0.015598
IB-TDANN	0.025009	0.009071	0.011340	0.018668	0.013205	0.030338	**0.015063**	**0.015693**	0.017860
Overall	0.030439	0.024684	0.026887	0.027121	0.030219	0.035047	0.025333	0.026630	0.024851

D.4 Summary

This appendix provided the detailed results of the interpolation algorithms examined in this thesis. The ANNs performance was analysed with different parameters. The interpolation algorithms were evaluated according to their ability to reconstruct gaps of different durations, followed by a detailed report on the reconstruction accuracy and the standard deviation of the NRMSE for each genre.

Appendix E

Acronyms

The important acronyms used throughout the thesis are listed below in alphabetical order. The acronyms are given in bold on the left-hand side with the corresponding description on the right-hand side.

ACF	Autocorrelation Function
ADC	Analogue-To-Digital
AIC	Akaike Information Criterion
AICC	Akaike Information Criterion Corrected
AN	Artificial Neuron
ANN	Artificial Neural Network
APD	Absolute Predictive Deviation
AR	Autoregressive
ARCH	Autoregressive Conditional Heteroskedasticity
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average

AWI	Adjacent Window Interpolation
BCB-TDANN	Bidirectional Complete Batch TDANN
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BHHH	Berndt-Hall-Hall-Hausman
BI-SRTDANN	Bidirectional Incremental SRTDANN
BI-TDANN	Bidirectional Incremental TDANN
BIC	Bayesian Information Criterion
BSB-TDANN	Bidirectional Separate Batch TDANN
CANN	Cascade Artificial Neural Network
CI	Cosine Interpolation
CML	Conditional Maximum Likelihood
COF	Connectivity Based Outlier Factor
DAC	Digital-To-Analogue
DFP	Davidon-Fletcher-Powell
DFT	Discrete Fourier Transform
DIC	Deviance Information Criterion
DSP	Digital Signal Processing
DTFT	Discrete Time Fourier Transform
EML	Exact Maximum Likelihood
FBOD	Feature Bagging for Outlier Detection
FCB-TDANN	Forward Complete Batch TDANN

FFANN	Feed Forward Artificial Neural Network
FFT	Fast Fourier Transform
FI-SRTDANN	Forward Incremental SRTDANN
FI-TDANN	Forward Incremental TDANN
FIC	Focused Information Criterion
FN	False Negatives
FOP	Fourier Polynomial
FP	False Positives
FPS	Fourier Polynomial Splines
FSB-TDANN	Forward Separate Batch TDANN
FT	Fourier Transform
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
HEP	Hermite Polynomial
HQIC	Hannan-Quinn Information Criterion
IB-TDANN	Interpolation Batch TDANN
IFT	Inverse Fourier Transform
INFLO	Influenced Outlierness
IUOS	Interpreting and Unifying Outlier Scores
kNN	k Nearest Neighbour
L-BFGS	Limited Memory Broyden-Fletcher-Goldfarb-Shanno
L-BFGS-B	Limited Memory Broyden-Fletcher-Goldfarb-Shanno Bounded

LAP	Lagrange Polynomial
LI	Lanczos Interpolation
LLS	Linear Least Squares
LOCI	Local Correlation Integral
LOF	Local Outlier Factor
LoOP	Local Outlier Probability
LPCM	Linear Pulse Code Modulation
MA	Moving Average
MAD	Median Absolute Deviation
MASD	Median Absolute Spectral Deviation
MCC	Matthews Correlation Coefficient
MHD	Mahalanobis Distance
MLE	Maximum Likelihood Estimation
MSE	Mean Squared Error
MWI	Mirroring Window Interpolation
NEP	Newton Polynomial
NLS	Nonlinear Least Squares
NN	Nearest Neighbour Deviation
NN	Nearest Neighbour
NNI	Nearest Neighbour Interpolation
NRMSE	Normalized Root Mean Squared Error

O-LBFGS	Online Limited Memory Broyden-Fletcher-Goldfarb-Shanno
OFP	Osculating Fourier Polynomial
OPG	Outer Product of Gradients
OSP	Osculating Standard Polynomial
PACF	Partial Autocorrelation Function
PAPR	Peak-To-Average-Power Ratio
PCC	Pearson Correlation Coefficient
PCM	Pulse Code Modulation
PDF	Probability Density Function
PPM	Pulse Position Modulation
PSO	Particle Swarm Optimization
PWM	Pulse Width Modulation
RMS	Root Mean Square
RMSE	Root Mean Squared Error
RSS	Residual Sum of Squares
SAT	Speed-Accuracy Tradeoff
SEN	Sensitivity
SI	Similarity Interpolation
SNR	Signal-To-Noise-Ratio
SOM	Self Organizing Maps
SPE	Specificity

SPS	Standard Polynomial Splines
SQNR	Signal-To-Quantization-Noise Ratio
SRANN	Simple Recurrent Artificial Neural Network
SRTDANN	Simple Recurrent Time Delay Artificial Neural Network
SS	Standard Score
STP	Standard Polynomial
SVD	Singular Value Decomposition
SVR	Support Vector Regression
TDANN	Time Delay Artificial Neural Network
TN	True Negatives
TP	True Positives
TSS	Total Sum of Squares

Appendix F

Symbols

This appendix provides a list of important mathematical symbols and notations used throughout the thesis. The symbols are categorized according to the chapters they first appear in and are ordered according to their appearance in the chapters. The symbols are given on the left-hand side with the corresponding description on the right-hand side.

F.1 Chapter 2: Gramophones and Audio Processing

Hz	Hertz, the unit measurement of frequencies
kHz	Kilohertz, equal to 1000 hertz
dB	Decibel, the logarithmic unit to express the ratio between two values
x_i	The i^{th} time delay of a times series
y_i	The i^{th} amplitude of a times series
ƒ	The frequencies as a result of a Fourier transform
$\mathbf{X}^{\mathrm{T}}$	The transpose of a matrix $\mathbf{X}$
$\mathbf{X}^{-1}$	The inverse of a matrix $\mathbf{X}$
$\mathbf{X}^{+}$	The Moore-Penrose pseudoinverse of a matrix $\mathbf{X}$
$\mathbf{X}^{*}$	The conjugate transpose of a matrix $\mathbf{X}$
$\mathrm{E}[y]$	The expected value of a random variable y
$\acute{\sigma}^2$	The biased population variance

$\acute{\sigma}_y^2$	The biased sample variance
σ^2	The unbiased population variance
σ_y^2	The unbiased sample variance
μ_y	The population mean of a series y
$\bar{y}$	The sample mean of a series y
$\text{var}(y)$	The unbiased sample variance of series y, equivalent to σ^2
$\text{cov}(y, z)$	The unbiased covariance between series y and z
$\text{cov}(y)$	The unbiased covariance of series y with itself, equivalent to $\text{var}(y)$
$\text{acov}(y_t, y_s)$	The unbiased autocovariance of series y at time delay t and s
$\text{cor}(y, z)$	The crosscorrelation between series y and z
$\text{acor}(y_t, y_s)$	The autocorrelation of series y at time delay t and s
$\acute{\sigma}$	The biased sample standard deviation
σ	The unbiased sample standard deviation
$(f * g)$	The convolution between function f and g
$(f \star g)$	The crosscorrelation between function f and g
y^*	The complex conjugate of series y

F.2 Chapter 3: Models

α_i	The i^{th} coefficient of the first set of model coefficients
β_i	The i^{th} coefficient of the second set of model coefficients
ε_i	The i^{th} error term
$f^{`}(x)$	The first derivative of function $f(x)$
$f^i(x)$	The i^{th} derivative of function $f(x)$
$f(x^-)$	The left limit of function $f(x)$
$f(x^+)$	The right limit of function $f(x)$
$\mathcal{N}(\mu, \sigma^2)$	A normal distribution
$\mathcal{P}(y\|\beta)$	The probability density function of series y and coefficients β
$\mathcal{L}(\beta\|y)$	The likelihood function, the inverse problem of $\mathcal{P}(y\|\beta)$

$\ell(\beta\|y)$	The log-likelihood function, equivalent to $\ln \mathcal{L}(\beta\|y)$
$\mathbf{H}$	A Hessian matrix
$\mathbf{H}_i$	A Hessian matrix at the i^th iteration during the BFGS optimization
R^2	The coefficient of determination
R^2_a	The adjusted coefficient of determination
L	The lag operator for ARIMA models
$\mathcal{Q}$	The Q-test for ARCH models
net	The net input for an ANs activation function

F.3 Chapter 4: Noise Detection

η	A noise map
$\breve{\eta}_i$	The i^{th} value of a noise mask generated from a noise map η

F.4 Chapter 5: Noise Reconstruction

$sinc(x)$	The normalized sine function

Appendix G

Derived Publications

This appendix provides a list of publications derived from the research of this thesis.

- Christoph F. Stallmann and Andries P. Engelbrecht. A Comparison of Interpolation Algorithms for Gramophone Record Sound Reconstruction. In *International Conference on Signal Processing and Integrated Networks*, pages 14–19, New Delhi, India, February 2014.

- Christoph F. Stallmann and Andries P. Engelbrecht. Gramophone Noise Reconstruction: A Comparative Study of Interpolation Algorithms for Noise Reduction. In *12*$^{\text{th}}$ *International Conference on Signal Processing and Multimedia Applications*, Colmar, France, July 2015. *Accepted.*

- Christoph F. Stallmann and Andries P. Engelbrecht. Digital Noise Detection in Gramophone Recordings. In *23*$^{\text{rd}}$ *ACM International Conference on Multimedia*, Brisbane, Australia, October 2015. *In review at time of publication.*

- Christoph F. Stallmann and Andries P. Engelbrecht. Automated Gramophone Noise Detection and Reconstruction using Neural Networks. *IEEE Transactions on Systems, Man and Cybernetics*, 2015. *In review at time of publication.*