CHALMERS
UNIVERSITY OF TECHNOLOGY

Iterative Partial Tracking in Inharmonic Audio Signals

Expanding the YIN Algorithm for Real-Time Monitoring and Treatment of Drum Harmonics

Master's thesis in Sound and Vibration

ALVA FÄHRLIN

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Iterative Partial Tracking in Inharmonic Audio Signals

Expanding the YIN Algorithm for Real-Time Monitoring and Treatment of Drum Harmonics

ALVA FÄHRLIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Architecture and Civil Engineering
Division of Applied Acoustics
Chalmers University of Technology
Gothenburg, Sweden 2026

Iterative Partial Tracking in Inharmonic Audio Signals
Expanding the YIN Algorithm for Real-Time Monitoring and Treatment of Drum Harmonics
ALVA FÄHRLIN

Supervisor: Mikael Sundin, Klevgränd Produkter AB
Examiner: Jens Ahrens, Department of Architecture and Civil Engineering

Master's Thesis 2026
Department of Architecture and Civil Engineering
Division of Applied Acoustics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46-(0)31-772 2200

Typeset in LaTeX
Gothenburg, Sweden 2026

Iterative Partial Tracking in Inharmonic Audio Signals
Expanding the YIN Algorithm for Real-Time Monitoring and Treatment of Drum Harmonics

ALVA FÄHRLIN
Division of Applied Acoustics
Department of Architecture and Civil Engineering
Chalmers University of Technology

Abstract

Time-domain subtraction of partials allows for lossless amplitude scaling of an input signal's tonal components, without affecting the phase. A method for real-time partial tracking in inharmonic transient audio signals, such as drum recordings, was developed by adapting the YIN algorithm [1] and combining it with time-domain subtraction of synthesised sinusoids. The algorithm can be applied block-wise to any tonal audio signal with analysis blocks as short as two times the period of the fundamental frequency of the signal, and step sizes shorter than the typical lookahead of a DAW. This allows for real-time application of the algorithm in a DAW, detecting fundamental frequencies down to at least 60 Hz.

The study features an overview of three methods for partial- and fundamental frequency estimation: lag domain methods, spectral methods, and transient modelling synthesis. Based on the overview, a new method called Iterative Partial Tracking (IPT) was formulated, in order to track partials via fundamental frequency detection alone. A YIN-based IPT implementation was adapted to the requirements set by DAW applicability, and tested on a set of floor tom recordings. The results were found to be promising: up to four partials were estimated in a typical floor tom with a gross pitch error (GPE) of 0%. The fundamental frequency of a floor tom was, in some recordings, estimated with errors that never exceeded one semitone. The accuracy of higher-order partial estimations could be improved by applying shorter, frequency based, analysis frame lengths, and the validation procedure of frequency estimations is subject to further improvements. But in its foundation, the lag-domain approach to tracking partials in inharmonic audio signals seems like a promising method, possible to implement in real-time applications where tonal-atonal signal separation is required.

Keywords: dsp, signal processing, music production, pitch tracking, partial tracking, signal separation, tonal-atonal separation, inharmonic, drums, YIN.

Acknowledgements

This study was conducted together with Klevgränd Produkter AB in Stockholm. I would like to thank the whole company for giving me such a warm welcome and including me in the team with great trust and excitement. Special thanks to Mikael Sundin, Johan Sundhage and Ludwig Ward for supporting me through the process, providing valuable insight and experience, as well as asking curious questions allowing me to explore new paths and methods.

Thank you also to Professor Jens Ahrens at the Division of Applied Acoustics, Chalmers, for suggesting appropriate literature and checking in with me along the way. Your confidence in my capabilities and your genuine interest in music technology has encouraged me to dive deeper into this field, both during this thesis and the years of studies leading up to it.

I would also like to thank Brekke & Strand Akustikk for letting me work in their Gothenburg office, and Shko Mirza for sharing that office with me, as well sharing all the ups and downs of writing and managing our projects. Your dedication and thoroughness is what kept me on track when the path forward wasn't crystal clear.

Professor Malte Kob at HFM Detmold provided me with valuable background information for the study within his course in music acoustics, and encouraged me to conduct a pre-study on drum sound properties within said course. This proved to be a valuable starting point, and improved my understanding of drums considerably.

Lastly, thank you to my opponent Nick Man for your input and suggestions regarding the report, as well as being good friend throughout our studies at Chalmers.

Alva Fährlin, Gothenburg, August 2026

Contents

Glossary

Term	Definition
Atonal	The property of not containing any perceptually noticeable partials, thus lacking a pitch. Atonal sounds or signals may contain transients, noise or both, according to the three-part signal model.
Attack	Musical term referring to the onset time of a sound or audio signal. The attack is the time it takes for the signal to reach its maximum level. Measured in ms.
Birth and death	The appearance and disappearance of partials detected in a signal. A partial is "born" when it is first detected in the signal, and it "dies" when it is no longer detected.
Block, frame	Interchangeable terms for a short segment of a signal, typically up to around 2048 samples long. Quasiperiodic signals are processed in signal blocks/frames, to capture their time-variant behaviour.
DAW	Digital Audio Workstation, refers to any computer program that is used to digitally manipulate and mix audio signals for music production purposes.
Frequency	The rate of repetition in a periodic signal or signal component, measured in cycles per second (Hz).
Fundamental frequency	The frequency of the lowest partial of a tonal sound, or the frequency of the first mode of a sound source. Measured in cycles per second (Hz).
Harmonics, harmonic partials	Also called Overtones. Harmonics are partials whose frequencies are multiples of the fundamental frequency (thus having a harmonic relation to the fundamental). Many music instruments produce harmonic partials, due to the standing wave properties of tubes and strings.
Inharmonic partials	Partials whose frequencies are not a multiple of the fundamental frequency. Inharmonic partials occur when the solution to the wave equation of the sound source is a Bessel function, rather than a simple set of harmonic modes. Inharmonic tonal content can also occur in a signal as a consequence of a strong resonance in a sound source that is unrelated to the pitch and fundamental, such as resonating screws and other attachments.

Modes	The innate oscillation patterns of a physical structure, for example a drumhead. Each mode represents an eigenfunction of the wave equation on the drumhead, and oscillates at its eigenfrequency. The oscillation pattern of a resonating structure can be described by an infinite sum of its modes, and its tonal behaviour is usually well approximated by the strongest lower-order modes.
Noise	A stochastic signal that by definition does not contain repetition, and thus lacks both frequency and, consequently, pitch. Noise is, along with transients, considered atonal sound components.
Overtones	See Harmonics.
Pitch	Musical term referring to the perceived frequency of a sound. Generally, the pitch corresponds to the fundamental frequency of the signal. Measured in cycles per second (Hz).
Plug-in	A piece of add-on software that complements the DAW. Typically designed to perform a specific task such as compression, filtering or pitch correction.
Tonal	The property of having a pitch.
Partial	A singular distinguishable frequency component of a tonal signal. A partial consists of a pure sine wave, and is thus fully described by its frequency, phase offset and amplitude.
Quasiperiodic signal	A signal that is temporarily periodic, at a rate that varies slowly over time. Musical signals are typically quasiperiodic, as their fundamental frequency changes slowly (in comparison to the rate of oscillation) in order to produce different pitches. Not to be confused with quasiperiodic functions, which is a mathematical term.
Residual	Any remaining contents of a sound that cannot be described by a model. In this report, the residual refers to the part of a signal that is not described by the tracked partials.
Tone	A musical term referring to one singular sound with one singular pitch. It has a fundamental frequency and typically also a set of partials that may be harmonic, inharmonic, or a combination of the two. In practice, a tone is the audible result of playing one note on an instrument; two notes played and recorded simultaneously may produce one audio signal, but that signal contains two tones and has two pitches, two fundamental frequencies, and two sets of partials that may or may not coincide in frequency. Tone can also, informally, refer to the qualitative spectral properties of a sound, but this report will use the terms Timbre or Formant when discussing such qualities.
Transient	A short impulse in a signal, such as the initial “hit” of a drum. Transients are, along with noise, considered atonal signal components.

Nomenclature

Indices

j	Sample number
t	Time step
n	Partial number

Variables

f, f_0	Frequency, fundamental frequency	(Hz)
T, T_0	Period, fundamental period	(s)
τ	Lag	(samples)
n	Arbitrary integer, $n \in \mathbb{N}$	
k, m	Arbitrary sample numbers, $k, m \in \mathbb{N}$	
p	Audio signal modelling a partial	
x	Audio input signal	
ϕ	Phase lag	
α	Amplitude	
W	Frame length/Block size	
S	Step size	
r_t, r'_t	Auto-correlation function, Normalised auto-correlation function	
d_t, d'_t	Difference function, Cumulative mean normalised difference function	
r_t, r'_t	Cross-difference function, Normalised cross-difference function	
a_t	Amplitude scaling curve	
z	Polynomial fit	

1 Introduction

This chapter introduces the topic and aims of the report. Section 1.1 presents the background and motivation behind the project: how prior research has focused on speech recordings with smooth dynamics and harmonic partial structures when developing pitch tracking algorithms, thus leaving a gap in knowledge on how to detect pitch in transient and inharmonic signals. Section 1.2 defines the aim of the study: to research and propose a method for partial-tracking and partial-level signal separation in inharmonic signals, and evaluate its performance once implemented. Section 1.3 sets the scope and limitations of the study; in summary limited to multi-partial tracking and tonal-atonal signal separation applied to drum recordings under typical music production conditions.

1.1 Background

Digital effects are a crucial tool for most music producers. In a Digital Audio Workstation (DAW), the music producer applies all sorts of signal processing to recorded or synthesised sounds, to shape them into an aesthetically pleasing piece of music. The signal processing takes place in the DAW, but also in so-called plug-ins, which is a specialised add-on software performing a certain signal processing task within the DAW. Figure 1.1 describes how plug-ins can be applied in unique series for each input channel, before all signals are added together in the mixer.

Mixing live recorded drums is a task as crucial as it is complex. A typical drum signal consists of both tonal components (the perceived pitch of the drum and its

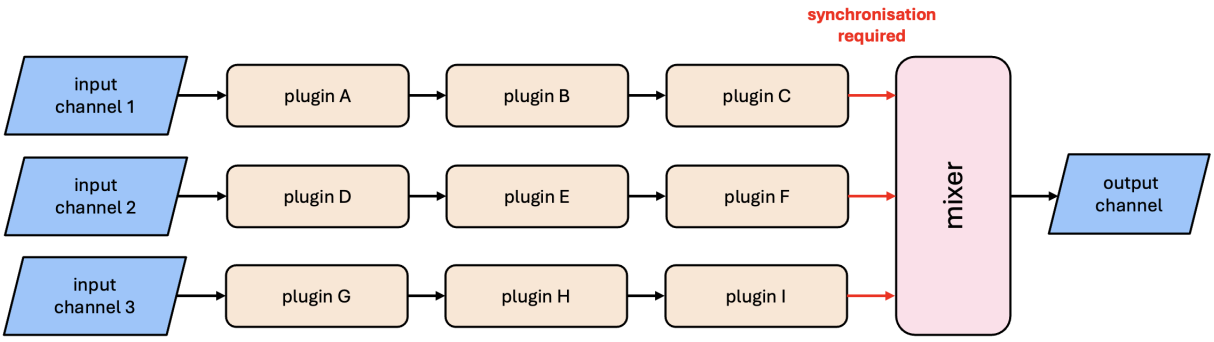


Figure 1.1: A typical DAW workflow with multiple input channels and plug-ins applied in series for each channel.

overtones), and atonal components (the transient "hit", as well as other noise components). Both the tonal and atonal qualities are important to take into account when mixing drum recordings; the transients should convey the proper amount of intensity and be well-balanced in the mix, while the tonal components should fit well with other tonal elements in the song in terms of pitch and duration. If the drums contain strong tonal components at pitches that are dissonant in relation to the key of the song, this can have a negative impact on the perceived quality of the mix. Modifying the pitch can however have undesired effects on the atonal qualities of the signal, such as the attack, duration and timing of the transients. To further complicate things, drums are often recorded with multiple microphones simultaneously. This imposes limitations on which effects can be applied to the recordings without creating phase issues between the channels. If aggressive filtering is applied to one of the recorded drum signals but not the others, the total character of the drum kit may change as phase cancellation occurs between the differently processed channels in the summing stage. How to balance the tonal and atonal qualities on each drum track while simultaneously minimising phase delay issues is a topic commonly discussed by mixing engineers, and considered a particularly difficult part of the mixing process. A tool that separates the tonal and atonal components of drum recordings would therefore be very valuable for a music producer.

Research in tonal extraction has been conducted for over a century [2], and methods for tracking the tonal content of a signal can broadly be divided into two categories: fundamental frequency tracking and partial tracking. Fundamental frequency tracking (f_0 tracking) refers to the detection of the fundamental frequency, f_0, of a tonal signal. Since the f_0 is closely tied to our perception of pitch, these algorithms are also called pitch tracking algorithms. Many f_0 tracking algorithms have been published over the years, and most of them are specifically developed for the human voice. A popular solution is a lag-domain method called the YIN algorithm [1], but frequency-based methods such as short-time transforms can also be used [2], [3]. f_0 tracking is used in research, voice pathology, voice synthesis and for creative applications such as music production [2]. In music production, f_0 tracking is used to alter or "correct" pitch, by quantising the fundamental and its harmonics to musically meaningful values. The operation is called pitch correction or "autotuning", after the pioneering plug-in called AutoTune which was released in 1997 [4].

Partial tracking refers to the detection of all distinguishable partials of a signal, including the fundamental, its overtones, and any other strong tonal components present in the signal. This could be a very valuable approach if applied in the context of tonal-atonal separation of drum signals, but much like pitch tracking, a majority of the publications on partial tracking have optimised their algorithms for harmonic instruments, often using voice recordings as grounds for evaluation. Drums and vocals have very different characteristics, the two main differences being that drums are more transient and that they have inharmonic partial structures that cannot be predicted from the fundamental alone. Partial tracking or correction is not typically used as a music production tool, mainly because the STFT approaches that have been developed by researchers for the purpose of partial tracking, were

not designed to operate within the latency requirements of a DAW.

If a DAW compatible method for partial tracking in inharmonic and transient signals were to be developed, this would allow for partial-specific amplitude adjustments in drum recordings with minimal effects on the signal's atonal characteristics. Such an algorithm could be used to tune individual partials of inharmonic signals in multi-track recordings, as well as enhance, soften, or fully remove drum resonances without the meticulous use of manually set filters.

1.2 Aims

This study aims to research and evaluate methods for partial tracking, with the goal of producing a functioning partial tracker that can be applied to recordings of typical inharmonic drums. Furthermore, the study aims to propose a method for isolating or suppressing individual partials in a way that allows for separate signal processing of the partials and the residual, respectively. Finally, the study should test whether the proposed methods can be combined into one algorithm that tracks and separates partials while fulfilling the requirements of DAW implementation, and evaluate its performance on real drum recordings.

1.2.1 Research questions

The following research questions are posed:

1. Which methods are available for real-time tracking of partials, and which are most suitable for use in a DAW?
2. Which methods are available for real-time suppression, synthesis or separation of partials, and which are most suitable for use in a DAW?
3. With the methods available, it it feasible to construct an algorithm that tracks and separates multiple partials simultaneously in a DAW?
4. How many inharmonic partials can be tracked accurately in one signal?
5. How stable and/or parameter sensitive are the methods?
6. Which properties of the input signal are most influential on the results?

1.3 Scope and limitations

The study focuses on the algorithms and procedures necessary to perform the tasks at hand, rather than producing a fully optimised, ready-to-use application. However, the decisions made should be informed by the restrictions associated with use in a DAW. This section defines the scope and limitations of the study by defining the requirements of the final algorithm, the limitations set on test data, and the limited scope of the evaluation process.

1.3.1 Requirements on the final algorithm

The algorithm should perform the following tasks:

- Track one or multiple partials in an inharmonic source signal.
- Split the incoming signal into one tonal part (consisting of the tracked partials) and one atonal or residual part.
- Output the signals separately – if possible, on a partial-to-partial level.
- Produce results that when mixed together are, if not identical to the input signal, as close as possible to perceptually indistinguishable from the input signal.

The algorithm should be possible to implement as a stand-alone plug-in in a DAW. Apart from being operational at standard sample rates (such as 44.1 kHz) and standard bit depths (16 or 24 bit), this also sets requirements for computation time and buffer sizes. The buffer size is the number of input samples the DAW collects before it produces its first output sample. The concept is analogous to the signal processing term block size, assuming that input and output blocks are of the same size. Different DAW:s deal with buffers a bit differently, but the time gap between pressing 'play', and the sound being played by the computer (also called the *latency* of the DAW) can generally be approximated by dividing the buffer size by the sampling frequency. Usually, the buffer size (and thus, the latency) can be changed in the settings of the DAW. A large buffer size allows more time for the computer to perform its calculations, which is useful if there are many parallel plug-ins running, or if long signal frames are required for output-dependent analysis. A smaller buffer size reduces latency, but demands faster or fewer computations, using shorter signal frames. The term *real-time algorithms* refer to algorithms that operate on a time scale that makes it possible to analyse and process a live stream of audio, with an output latency that is sufficiently short given the circumstances. For live performances and recording, "sufficiently short" is a matter of just a few milliseconds, the shorter the better. The restrictions are looser regarding playback of previously recorded sounds such as in a DAW; here, a latency of up to 23 ms (1024 samples at 44.1 kHz) is typically allowed. The important restriction however, is that for an algorithm to be implemented in a DAW plug-in, it must operate in a way that does not require it to know the input signal beforehand. Both block-wise and sample-by-sample processing are allowed, but every sample that the plug-in needs access to must be added to the buffer, thus increasing its length. This applies even when the signals are pre-recorded. Algorithms that require full knowledge of the entire signal before being able to produce any output are called *offline algorithms*, and are only used in plug-ins in very rare cases.

Since live performances and recording constitute a special case of extremely low latency requirements, this study will not make an attempt at achieving them. Assuming a sample rate of 44.1 kHz, the following requirements define what is considered *real-time functionality* for playback in a DAW:

- The buffer size should not exceed 1024 samples.

- The algorithm should not need access to future samples, beyond the buffer.
- The computation time needed for each signal block or buffer, should not exceed its corresponding duration in the signal.

These requirements assure a latency below 23 ms at 44.1 kHz by limiting the number of samples between input and output and demanding that the algorithm keeps the same speed as the incoming audio. The third requirement is largely a question of optimising the program, and though it should be considered in the choice of methods, it is not to be formally evaluated.

1.3.2 Other limitations

- The signals used in the evaluation are professionally recorded samples of floor toms with a sampling frequency of 44.1 kHz and bit depth of 24 bit.
- The recordings are of single hits on individual drums, with no leakage and minimal background noise present.
- The study focuses on evaluating audible signal features; artifacts outside the typical hearing range of 20 - 20 000 Hz are not considered.

2
Theory

This chapter introduces the theoretical background of the study, and is divided into four main sections. Section 2.1 introduces some terminology to describe audio signals, and provides a background on the characteristics of the drum type toms. Section 2.2 explores and evaluates a set of methods for fundamental frequency estimation and partial tracking, based on their fitness to solve the task at hand. Section 2.3 does the same for signal separation. Finally, the findings are summarised in Section 2.5, where a decision on which methods to implement is made.

2.1 Drum signal properties

This section defines some fundamental properties of audio recordings in general and drums in particular, from which we can understand the typical characteristics of drum recordings.

2.1.1 The three-part signal model

This study uses a model consisting of three main signal components: sinusoids, transients, and noise. The model description is adopted from the definitions used in Transient Modelling Synthesis [5], and it is particularly useful when describing transient audio signals, such as drums. When analysing audio signals with slow attack or soft transients, it is often sufficient to use a two-part model such as the sinusoid plus residual model. The residual of said model includes both noise and, if present, any transient features of the signal [3]. In drum recordings, however, the transient is a crucial part of the signal, and should be considered a separate entity. Equation 2.1 shows how a signal $x(t)$ may be divided into the three signal components, which are further described in the paragraphs below.

$$x(t) = \text{sine}(t) + \text{transient}(t) + \text{noise}(t) \tag{2.1}$$

Sinusoidal component

The sinusoidal component represents the periodic or quasiperiodic parts of the signal, which, if within the audible frequency range, is what humans perceive as tonal content. A sine wave is fully described by its frequency, amplitude and phase offset, and any periodic signal can be described as an infinite sum of sine waves, added in a linear manner. It is thus possible to fully reconstruct a sampled signal by

the means of only sine waves (as is done in Fourier transform) – but stacking an endless number of sine waves in order to replicate atonal features such as noise and transients, is often neither efficient nor perceptually meaningful [5]. In the three-part model, the sinusoidal term should therefore be limited to including the slowly varying, perceptually relevant partials of a signal. These can be harmonic or inharmonic partials, or stem from other slowly modulated periodic features of the signal. The sinusoids thus describe mathematically, what perceptually constitutes the tonal component of a sound.

Transients

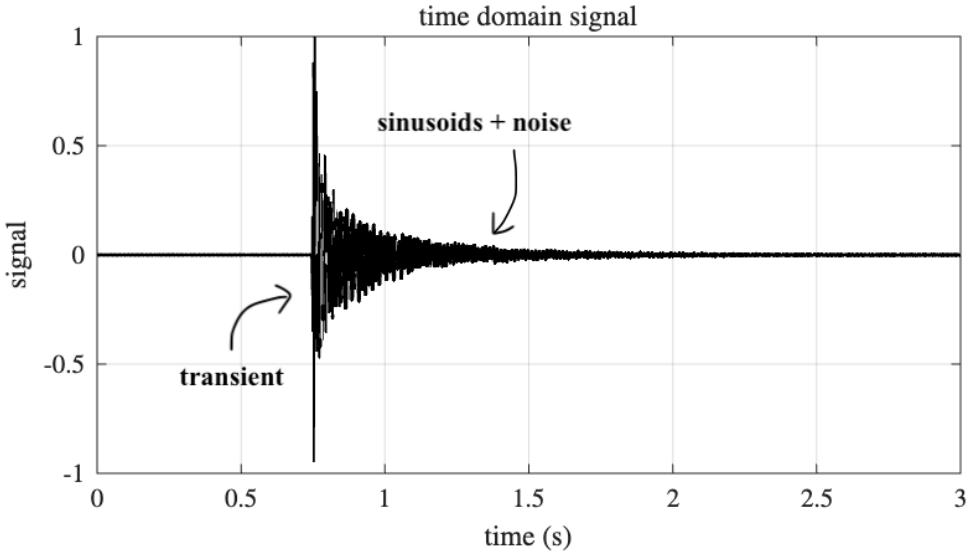
Transients are defined as short bursts of energy in a signal, and are usually described as dirac delta functions or other unit impulses. Another approach, more suitable for music signal analysis, is to model the transient as a one-sided exponential decay envelope, acting on the amplitude of a tone or a noise source [5]. This better corresponds to the typical behaviour associated with transient elements in real, recorded audio signals, and allows for an intuitive understanding of the energy decay associated with an impulse imposed on a physical instrument. The differentiation between transients and other signals components is thus decided by the amplitude envelope rather than the signal content itself; anytime the amplitude changes faster than what is considered "slow" by the sinusoidal or noise models, we are dealing with a transient. Furthermore, if a transient is repeated at a sufficiently high and regular rate, it will eventually be better described by a periodic signal model, and thus move into the sinusoidal domain.

Noise

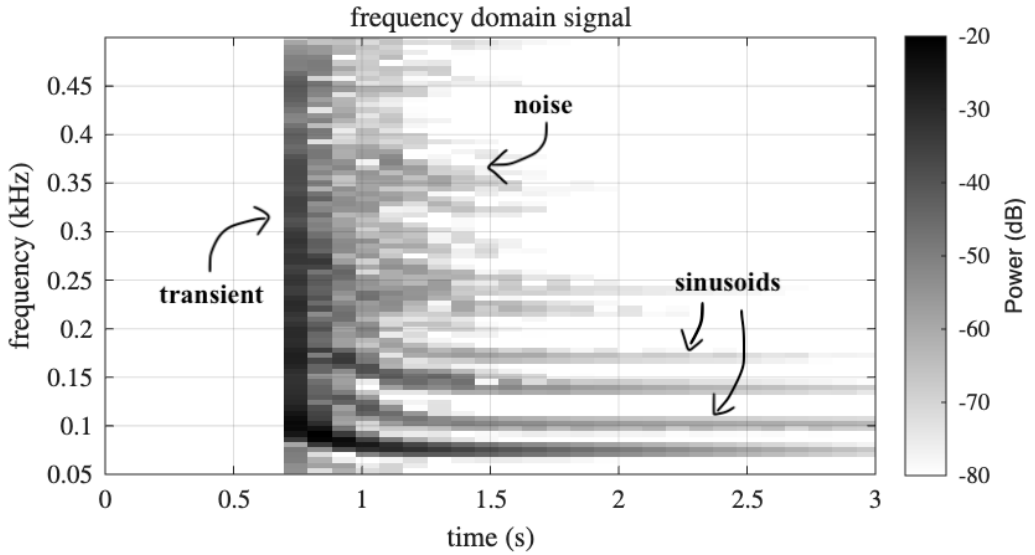
The noise component in this model is the residual of the sinusoidal and transient components, containing stochastic noise that is neither tonal nor transient. A pure noise signal, without tonal components or transients, is perceptually indistinguishable from the result of a slowly time-varying filter, acting on a source of white noise.

2.1.2 Tonal properties of drums

Drums is a large family of instruments, and their degree of tonality varies on a wide spectrum from fully pitched such as xylophone and timpani, to almost no pitch such as bass drums and snares. Toms are somewhere in the middle of that scale, since they produce a distinct pitch, but are not primarily used to produce melodic or harmonic musical features. The pitch that we perceive from a tom is usually consistent with the fundamental frequency, which corresponds to the first oscillation mode of the drumhead. This frequency is however not usually constant throughout one hit, but drops exponentially in the first half second or so, before stabilising. The phenomenon can be seen in Figure 2.1b. The same is true for higher-order modes of the drumhead; they drop in frequency at the same relative rate as the fundamental, producing partials that follow the same curve but at their respective modal frequencies. Depending on various properties of the drum and playing technique (tuning, single- or double-headed, impact velocity and force, to name a few), the drop in



(a) Transient, sinusoidal and noise components in a time domain signal.



(b) Transient, sinusoidal and noise components in a spectrogram.

Figure 2.1: The three-part model of transients, sinusoids and noise, in a floor tom signal.

pitch may span different intervals and drop at different speeds. If the hit is forceful, resulting in a large initial displacement of the membrane, the pitch drop may span 10 % of the original frequency [6].

Drum sounds have inharmonic partial structures. This is an important difference compared to most instruments, since a signal with inharmonic partials is not strictly periodic like a harmonic signal is. Harmonic overtones usually occur in mechanical systems due to the modes of lambda-half or lambda-quarter resonators, such as strings or tubes. A moderately damped harmonic resonator will resonate at its fundamental frequency, as well as a set of *harmonics*: partials with frequencies that

are multiples of the fundamental frequency. The modal behaviour of drumheads is instead governed by Bessel functions; solutions to the wave equation on a membrane. Bessel function roots are not multiples of the first mode like in harmonic resonators, so membrane-based sound sources do not typically present with harmonic overtones unless coupled to strongly harmonic systems [6]. When a drum is hit close to the centre of the drumhead, most of the energy enters through the first mode. The energy is then redistributed to higher-order modes through non-linear coupling. These higher-order modes may end up resonating longer than the initial first-order mode, shifting the initial spectral contour of the sound [6].

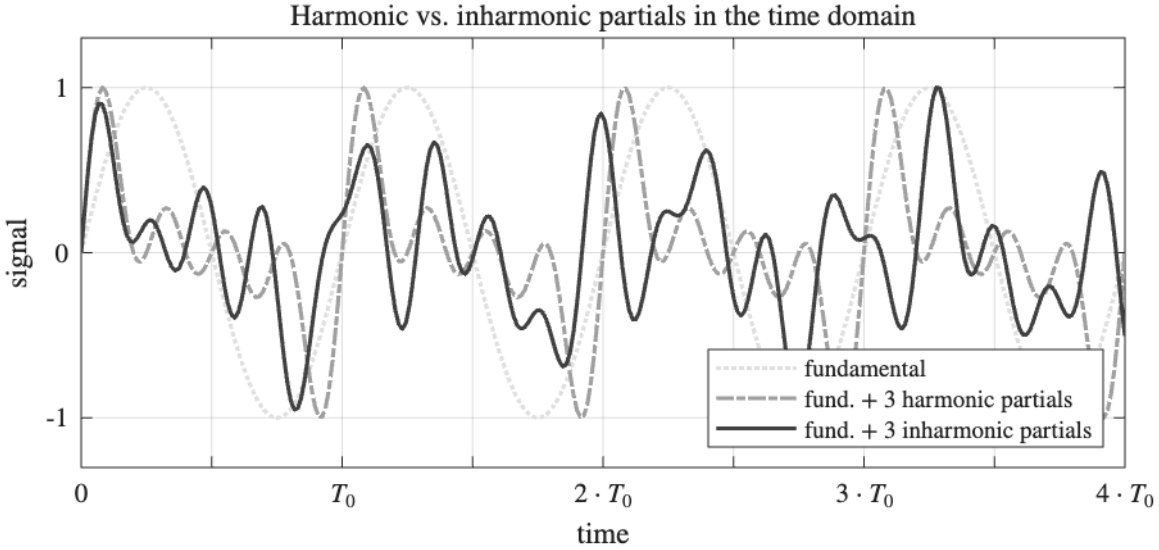


Figure 2.2: A harmonic and an inharmonic signal, with four equally strong partials each. The harmonic partials are of 2, 3, resp. 4 times the fundamental frequency, whereas the inharmonic partials have relative frequencies of 2.16, 3.14 and 4.65.

As mentioned earlier, the inharmonicity of drum sounds plays a role in to which degree we can consider them periodic. A signal with only harmonic partials will, disregarding noise components, be perfectly periodic, since all of the partials fit evenly within one period of the fundamental. An inharmonic partial structure allows for a mismatch so that two consecutive periods of the fundamental frequency are not identical. This is illustrated in Figure 2.2, where the harmonic signal clearly stays periodic at the same period T_0 irrespective of the added partials, but the inharmonic signal show variations between the periods of the fundamental.

2.2 Frequency estimation

Over the years, there have been many attempts at creating algorithms that automatically detect and track the partials or fundamental frequency of a signal. We call algorithms that track fundamental frequency f_0 trackers, and algorithms that track partials are consequently called partial trackers. Frequency estimation takes place in both f_0 trackers and partial trackers, as it refers to the procedure of identifying the frequencies of tonal components in a signal, regardless of their character.

This section will explore three methods for frequency estimation, and their associated methods for phase and amplitude estimation if they exist. Subsection 2.2.1 describes lag domain methods; a class of f_0 trackers including the popular YIN algorithm. Spectral peak tracking is described in Subsection 2.2.2, and refers to frequency domain peak searching which can be used for tracking all partials including the fundamental frequency. Transient Modelling Synthesis, described in Subsection 2.2.3, is a specialised transform that handles transient signals in order to facilitate regular short-time analysis. The three methods are compared and evaluated in Section 2.5.

2.2.1 Lag domain methods

Lag domain methods are a class of fundamental frequency estimation methods based on a simple inherent property of all tonal sounds: time domain periodicity. A perfectly periodic signal that is time-shifted (delayed) by one period is by definition equal to the original, non-shifted, signal. It is therefore possible to identify its period by shifting the signal to different intervals, so-called *lags*, and identify the smallest lag where the signal is equal to the non-shifted original. This lag would by definition correspond to the period of the signal, thereby revealing its fundamental frequency. A musical signal is not perfectly periodic as a whole, but it is what is called *quasiperiodic*: it contains shorter segments, for which the signal can be considered periodic or almost periodic. Even if there are dissimilarities between consecutive periods within such a segment (such as noise, or small deviations in frequency and amplitude), there will still be a non-zero lag that produces the largest similarity when the signal is shifted and compared to the original. Lag domain methods can thus be used to approximate the instantaneous period of a quasiperiodic signal, when applied on a frame-to-frame basis.

There are two common methods for conversion between the time domain and the lag domain: the auto-correlation function and the difference function. The auto-correlation function can be found in the free audio analysis tool Praat [7], first published in 1992. The difference function method is commonly called the YIN method [1], and has been the more popular method since its publication in 2002 [3]. Both methods compare a shifted and an non-shifted signal to produce a so-called *lag curve*, but with different means of comparison, resulting in curves in which either the maxima or the minima will indicate the lags of interest. The two methods are briefly described below, before the general benefits and challenges of lag domain methods is discussed as a whole.

2.2.1.1 The auto-correlation function

Equation (2.2) defines the autocorrelation function (ACF) of a sampled signal $x(j)$ in the range $j \in [t+1, t+W]$, where t indicates an arbitrary initial sample number, W is the frame length and τ is the lag in samples.

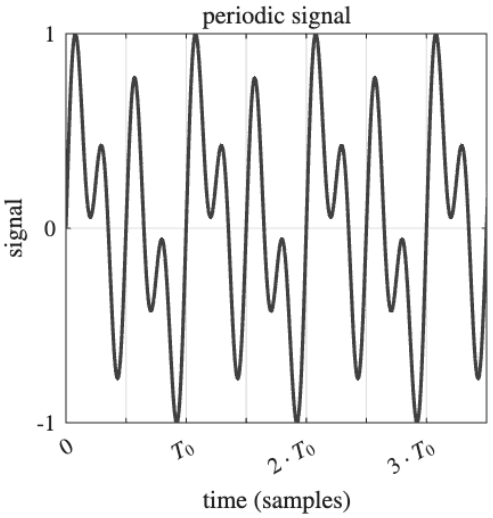
$$r_t(\tau) = \sum_{j=t+1}^{t+W} x(j) \cdot x(j+\tau) \qquad \tau = 0, 1, 2, ... \tag{2.2}$$

The ACF will yield a global maximum for $\tau = 0$. If the signal is periodic, a maximum will also appear at $\tau = T_0$, where T_0 is the period of the signal (in samples). Note that as τ increases, $j + \tau$ will become larger than W. So for a comparison over W samples with a lag range of $[0, \tau_{\max}]$, the actual sampled frame must be $W + \tau_{\max}$ long.

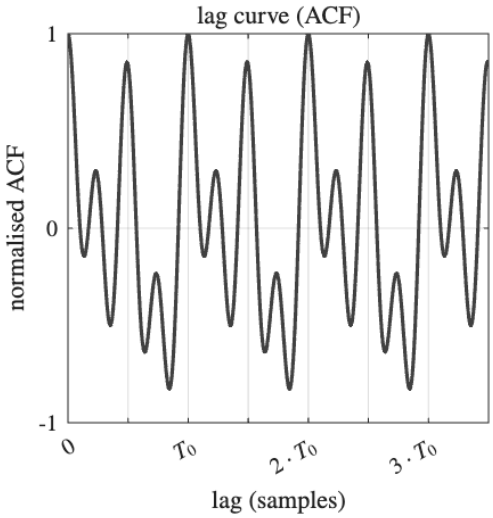
The autocorrelation at $\tau = 0$ corresponds to the power of the signal, and can be used to normalise the signal as shown in Equation (2.3).

$$r'_t(\tau) = \frac{r_t(\tau)}{r_t(0)} = \frac{\sum_{j=t+1}^{t+W} x(j) \cdot x(j+\tau)}{\sum_{j=t+1}^{t+W} [x(j)]^2} \tag{2.3}$$

After the lag curve has been normalised, it might look something like in Figure 2.3b, with global maxima occurring at every multiple of the signal's period. The task of f_0 estimation is thereby reduced to choosing the correct local maximum of the curve, where $\tau = T_0$, in order to compute f_0 as f_s/τ.



(a) A periodic signal consisting of four harmonic partials.



(b) Normalised autocorrelation function (ACF) of the periodic signal shown in 2.3a.

Accidentally choosing τ as a multiple of T_0 will result in what is called a *subharmonic error*, as it results in an f_0 estimation that is too low by a factor of $\frac{1}{n}$, assuming the nth multiple was chosen. A signal with harmonic overtones will also present some degree of periodicity at shorter lags than the fundamental – this can lead to so-called *octave errors*. Octave errors is a misleading name as the principle applies to any harmonic overtones, but since the most common error is the first harmonic,

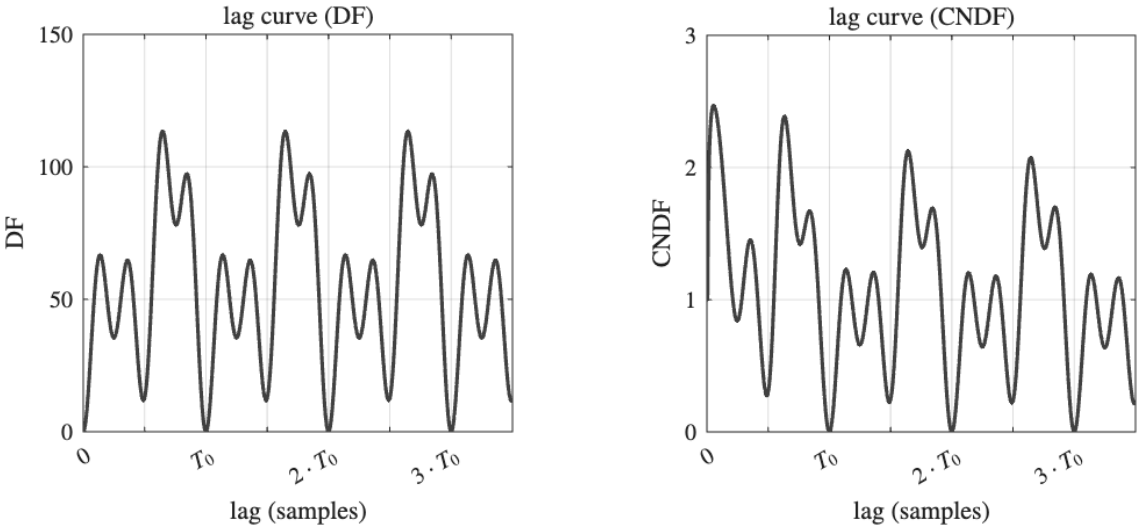
also called the octave, the name has stuck around. If a signal possesses strong low-order harmonics, these will show up in the lag curve as local maxima at shorter lags. Figure 2.3b shows just that: there are three local maxima of lower amplitude between each global maximum. As long as the added partials are harmonic, the global maximum will always be located at the fundamental frequency. If the first or second harmonic has a high amplitude in the time domain, however, its amplitude in the lag domain may come very close to that of the fundamental. This increases the risk of choosing the wrong maximum and thus assuming the wrong period, resulting in the typical octave error.

2.2.1.2 The difference function

An alternative to the ACF is the difference function (DF), proposed by de Cheveigné and Kawahara as a part of the YIN algorithm [1]. YIN is one of the most widely used fundamental frequency tracking tools in real-time applications to this day, sometimes referenced as the standard tool for f_0 estimation [3]. The DF is defined in in Eq. (2.4), as:

$$d_t(\tau) = \sum_{j=t+1}^{t+W} (x(j) - x(j+\tau))^2 \tag{2.4}$$

Similarly to the ACF, the DF compares the signal to itself at different lags, but it does so by subtraction instead of multiplication. Instead of looking for a maximum of the lag curve, we are now looking for a minimum, since a difference of zero would indicate that the two signal segments are identical. Figure 2.4a shows the DF applied to the same signal as in earlier examples.



(a) Difference function (DF).

(b) Cumulative mean normalised difference function (CNDF).

Figure 2.4: DF and CNDF representations of the periodic signal shown in 2.3a. Both curves show global minima at integer multiples of T_0, but the CNDF does so at a normalised scale, and without a minimum at lag zero.

For normalisation, de Cheveigné and Kawahara proposed the *cumulative mean normalised difference function* (CNDF), shown in Eq. (2.5) and Figure 2.4b. In addition to normalising the y-axis and making the curve unitless, it also removes the zero-lag minimum by forcing the lag curve to an initial value of one.

$$d'_t(\tau) = \begin{cases} 1 & \text{if } \tau = 0 \\ d_t(\tau) \Big/ \left[\frac{1}{\tau} \sum_{j=1}^{\tau} d_t(j) \right] & \text{otherwise} \end{cases} \tag{2.5}$$

After normalisation, it is time to identify the correct minimum, where $\tau = T_0$. The YIN algorithm does so by setting a threshold at 0.1, and picking the shortest lag among the local minima of the CNDF that falls below the threshold (or, if no such minimum exists, it picks the global minimum of the CNDF) [1]. This method intends to avoid both sub-harmonic errors and octave errors simultaneously, and the need for an upper frequency limit of the search range is thus in theory eliminated. However, the depth of the minima in the lag curve depend on the actual periodicity of the signal, and the decision on a threshold value therefore depends on the source signal's harmonic properties. The YIN study was performed on speech recordings, for which a threshold of 0.1 was appropriate without the need of fine-tuning [1], but there is no guarantee that this will hold for other types of signals.

2.2.1.3 Tolerance to noise and inharmonicity

The accuracy of lag domain methods highly depends on the degree of actual periodicity of the signal, which in turn is affected by both signal-to-noise ratio and the presence of inharmonic partials. Stochastic noise will lower the degree of periodicity in a general way, as it introduces random differences between periods. This will not affect the location of lag curve minima, but it will impose a noise floor on the lag curve that may affect the prominence or relative ratios between minima (see Fig. 2.5).

The presence of strong inharmonic partials, however, may shift the actual locations of the lag curve minima. As seen in Figure 2.6, the addition of a harmonic partial (in this case, the octave), does not affect the location of the first global maximum. The addition of a slightly shifted octave (2.16 times the fundamental) to the signal, shifts the apparent period of the fundamental downwards. The prominence of the shift depends on the degree of inharmonicity of the partial, but it also highly depends on the relative amplitude. In this example, a 1:1 amplitude relation was used.

2.2.1.4 Frame length

In relation to the maximum expected fundamental period of the signal, $T_{\max}$, frequency estimation by time-shifting requires relatively short signal frames. If the search range is limited to $\tau < \tau_{\max}$, and $\tau_{\max} = T_{\max}$, the frame needs to be $2 \cdot T_{\max}$ to allow for a shift of one full period. If the algorithm is implemented in such a way

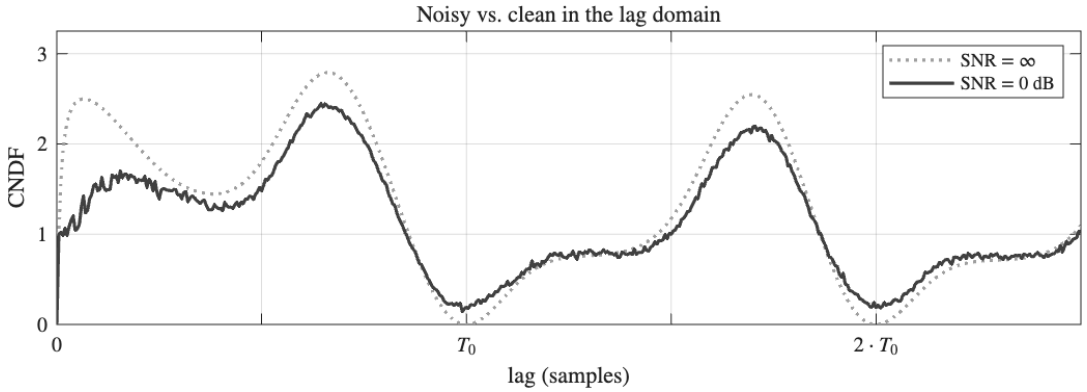


Figure 2.5: Comparison between the CNDF of a noisy signal and a clean signal. The minima of the noisy signal are not as deep as those of the clean signal, and the difference between the fundamental minima and the minima corresponding to harmonics of the signal is also smaller than in the clean signal.

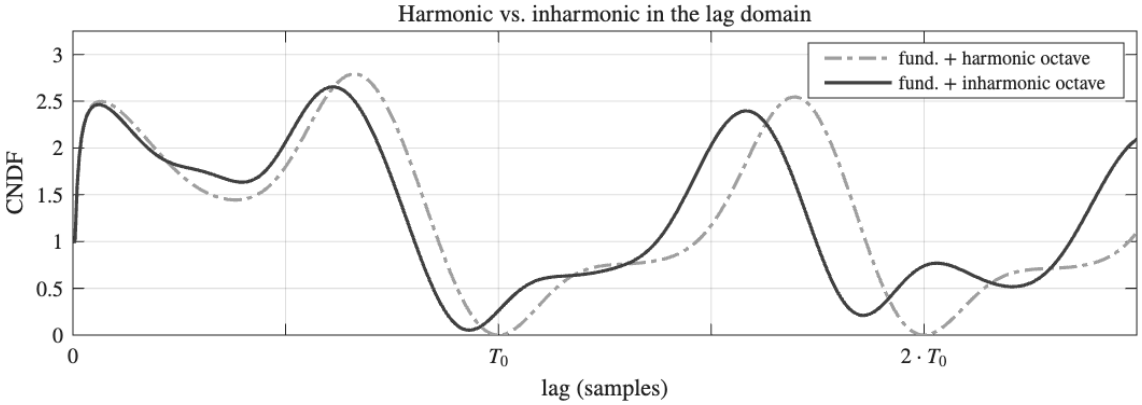


Figure 2.6: Comparison between the CNDF of a signal with a harmonic octave ($f_2 = 2 \cdot f_1$) and an inharmonic octave ($f_2 = 2.16 \cdot f_1$). Adding an inharmonic partial to the signal changes the apparent period of the fundamental.

that it recognises immediately when it has found a proper lag candidate, the actual used frame length may be as short as $T_{\max} + T_0$ [1]. With the minimum-picking algorithm proposed by [1] this is a possibility, but in general the user should aim at setting $T_{\max}$ as close as possible (but above) the expected T_0, in which case the difference in frame length or performance would be small regardless. Increasing the frame length further than $2 \cdot T_{\max}$ reduces noise in the lag domain due to averaging, but it does not resolve the issues caused by inharmonic partials.

2.2.1.5 Phase and amplitude estimation

Lag domain methods estimate fundamental frequency based on periodicity, without respect to phase or amplitude. The phase and amplitude do not need to be estimated if the purpose of the algorithm is simply to track pitch, but if the results are to be synthesised, both amplitude and phase need to be set to appropriate values. This is especially important if the goal is to add or subtract the estimated signals from the input signal, since such operations are only linear if both frequency and

phase are considered (this is explained in more detail in Section 2.3).

One option would be to switch over to spectral analysis, and examine phase and amplitude at the estimated frequency. This requires interpolation and window considerations in the frequency domain, which are discussed in Subsection 2.2.2. If one wants to avoid spectral analysis whatsoever, it is also possible to use the cross-difference function to estimate both phase and amplitude once the frequency has been established. The following paragraphs describe the procedure.

The ACF and DF compares two identical copies of a signal, in order to quantify its correlation to itself. If two *different* signals are compared in a similar manner, this is called cross-correlation or cross-difference. A cross-difference function (CDF) of two pure sine waves would show a minimum at the lag corresponding to the phase difference between the two signals, since that is where they are most similar (if their amplitudes match, they would in fact be identical). This means that we can use a sine wave of arbitrary phase to identify the unknown phase of a partial with a known frequency in another signal. Equation (2.6) shows how a frequency estimation f_{est} can be used to define a zero-phase sine wave, here denoted x_{est}, which approximates the desired partial. In Equation 2.7, the zero-phase sine wave is compared to the original signal x by cross-difference, and the resulting lag curve $c_t(\tau)$ will present minima at the lags where the two functions are most similar (and thus, in phase).

$$x_{\text{est}}(j) = \sin\left[2\pi \cdot f_{\text{est}} \cdot j\right] \tag{2.6}$$

$$c_t(\tau) = \sum_{j=t+1}^{t+W} \left(x_{\text{est}}(j) - x(j+\tau)\right)^2 \tag{2.7}$$

This might seem trivial and like an unnecessary procedure for two sine waves, but if one of the functions is a pure sine wave and the other is a signal with multiple partials, the two signals will be most similar when the sine wave is in phase with its corresponding sine component in the multi-partial signal. This means that the CDF will have exactly one minimum per period, indicating the phase difference between x_{est} and the f_{est} component in x. This is true regardless of any scaling applied to x_{est}, but to reduce the effects of noise and rounding errors it might be a good idea to scale x_{est} so that it approximately matches its expected amplitude (for example by matching it to the maximum detected amplitude of x within the frame). x_{est} can thus be improved by subtracting the lag of the first minimum, τ_ϕ, from the original expression, that then reads as in Eq. (2.8):

$$x_{\text{est}}(j) = \sin\left[2\pi \cdot f_{\text{est}} \cdot j - \tau_\phi\right]. \tag{2.8}$$

In order to estimate amplitude, a similar comparison can be made by applying a range of scaling factors α to the updated x_{est}, subtract them from x, and identify the scaling factor that yields the lowest difference between the two signals. Equation (2.9) describes the process of creating the *scaling curve* $a_t(\alpha)$, whose global minimum indicates the best matching value of α.

$$a_t(\alpha) = \sum_{j=t}^{W} (\alpha \cdot x_{\text{est}}(j) - x(j))^2 \tag{2.9}$$

As a last remark, it is important to note that order of the difference operations applied to estimate all properties of the partial must be frequency-phase-amplitude; as every new step builds upon the previous.

2.2.1.6 Summary on lag domain methods

Lag domain methods are a class of fundamental frequency estimators built on autocorrelation, which are real-time compatible and light in computing. They require signal frames of at least two times the longest expected period, and are not dependent on any particular synthesis/re-synthesis scheme since they are applied directly on the time domain signal. Phase and amplitude of the partial found can be estimated by the use of cross-difference. The core of the procedure is the min/max selection applied to the lag curve, for which the YIN algorithm provides a good alternative for speech signals. The algorithm must be applied iteratively in order to estimate multiple partials. Noisy and/or transient signals may produce lag curves that increase the difficulty of min/max selection, and lag domain methods rely on harmonicity; if strong inharmonic partials are present, the lag curve may indicate the wrong fundamental frequencies entirely.

2.2.2 Spectral peak tracking

Spectral peak tracking is a frequency-domain method in which Short-Time Fourier Transform, STFT, is employed to monitor the spectral development of a signal. STFT is a common analysis tool, used for example to produce spectrograms. The analysis step converts a time-domain signal into a frequency-domain spectrum, and is reversible by inverse Fourier transform (IFFT), allowing for a perfect re-synthesis of the original signal frame. The strongest partials can be identified by finding the most prominent peaks in the frequency spectrum, which allows for tracking of a signal's tonal content, irrespective of its harmonicity. A comprehensive overview of how to perform spectral peak tracking on inharmonic signals can be found in [8], but the general method is briefly summarised below.

STFT is the procedure of applying Fourier transform (FFT) on short, often-times overlapping, signal frames. The length W of a frame dictates the frequency resolution of the resulting spectrum, as the number of frequency points in the spectrum (which spans a range of $[0, f_s/2]$, where f_s is the sampling frequency) is proportional to the number of samples. Zero-padding (adding zeros to the beginning and/or end of the frame) increases the apparent frequency resolution of the resulting spectrum, but it does so by interpolation, thus not revealing any new information about the signal. The same applies to IFFT; the output signal will always have the same length and resolution as the input signal, as they are coupled via the sample rate.

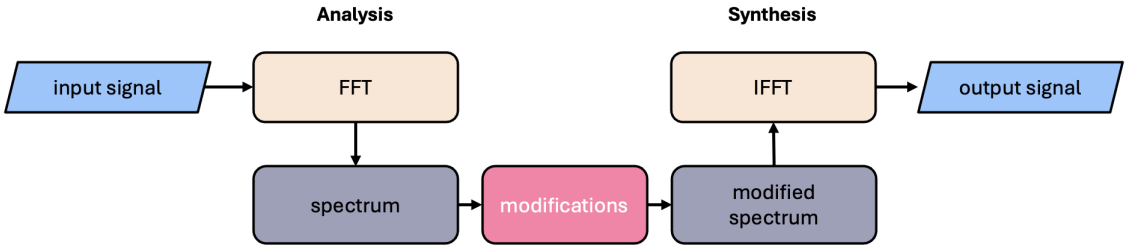


Figure 2.7: Block scheme of an FFT analysis-synthesis process.

An important decision when using STFT, is the choice of window function. Extracting a frame from the full signal to perform STFT is effectively the same as multiplying the signal with a short rectangular function, and this has consequences in the frequency domain. A multiplication in the time domain represents a convolution in the frequency domain, so the resulting spectrum won't be that of the input signal only, but a convolution of a.) the transform of a rectangular function, with b.) the transform of the signal. This means that even a perfectly singular sinusoid in the input signal won't present as a singular spike in the spectrum, but as a symmetric function containing a main lobe and several side lobes, with amplitudes and widths corresponding to the transform of the window that was applied. Properties such as *main lobe width* and *main-to-side lobe ratio* will affect the ability to differentiate between closely located frequency peaks in the spectrum (see Figure 2.8). There is no global answer to which window function is the best; extensive research on this topic has resulted in a multitude of window options to choose from. The choice of window also affects the overlap-add process required for re-synthesis of the signal, as overlapping windows may cause amplitude modulations in the final result if not applied correctly. [3], [8].

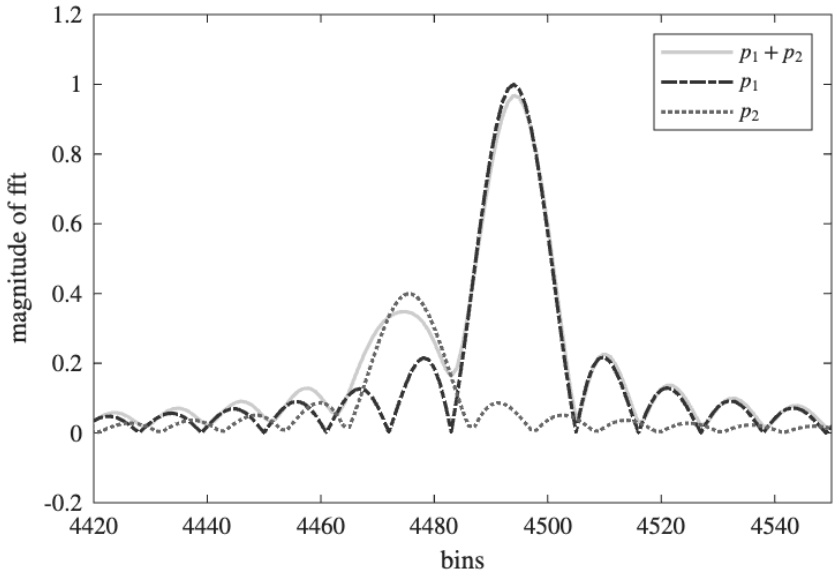


Figure 2.8: Two closely spaced partials in a spectrum, with main and side lobes corresponding to the rectangular function. If the partials are very closely spaced, or of different amplitudes, it can be difficult to immediately determine which peaks are main lobes, and which are side lobes.

Once the signal is transformed into the frequency domain, the actual peak tracking can take place. Using the known window function properties, each local maximum of the spectrum is classified as either a partial or a side lobe of a partial, and the accuracy of peak frequencies can be further improved by parabolic interpolation of neighbouring data points in the frequency domain. When all peaks in a frame have been located and classified, they are sorted and assigned to *tracks*, each representing the temporal evolution of one partial. By analysing both the frequency and phase spectra, all of the parameters necessary for oscillator bank synthesis (frequency, phase and amplitude) can be determined, and the partial can be re-synthesised by an oscillator. The number of partials that can be identified using spectral peak tracking is in theory unlimited, so the user must either limit the number of partials tracked, or set a lowest allowed magnitude, in order to draw the line between partials and noise. Partials can be re-synthesised by an oscillator bank or via IFFT. Noise can be re-synthesised as a time-varying filter applied to white noise (in which case the phase information of any remaining partials is lost), transformed back into the time domain via IFFT, or omitted entirely. Spectral peak tracking can thus serve as a means for data reduction and/or noise reduction, by separating tonal (sinusoidal) and atonal (noise and transients) contents of the signal. [3], [8].

2.2.2.1 Tolerance to noise and inharmonicity

STFT does not rely on harmonicity to estimate frequencies, which means that the spectral peak tracking model should not be negatively affected by inharmonic partials. The presence of broadband noise in a signal will affect the lowest level of magnitude for which partials can be identified, but the effect might vary over the range of the spectrum depending on the spectrum of the noise. High frequency noise should not affect low frequency partial detection, and vice versa. Transients, however, have a documented negative effect on the peak tracking, as they temporarily obscure the spectrum with high level broadband noise [8]. Depending on the relative levels of noise, transients and tonal content, different methods for re-synthesis might be more or less appropriate. STFT is originally a one-part sinusoidal model, but by limiting the number of partials tracked in the frequency domain, it turns into a two-part model of sinusoidal plus residual. Any non-periodic components outside of the tracked partials, including transients, will end up in the residual. Depending on how the residual is re-synthesised, this can put the integrity of the transients at risk, as transients are poorly modelled by filtered white noise [5].

2.2.2.2 Frame length

As anyone who has seen a spectrogram knows, STFT is always a compromise between time and frequency resolution. If the signal is stationary, the demands on time resolution are relaxed, allowing for long frames which increases the frequency resolution. Conversely, if the signal contains transients or rapidly changing partials, the frames must be shorter, which in turn reduces the resolution on the frequency axis by providing fewer frequency bins. The frequency spectrum of an STFT has a resolution of $\Delta f = f_s/W$, where f_s is the sample rate and W is the frame length. The frequency resolution of the spectral peak tracking algorithm can however be much

finer than that of the frequency spectrum itself, since zero-padding and polynomial interpolation opens up for inter-bin localisation of partials. Still, a minimum requirement of frequency resolution is needed to discriminate between closely spaced partials. The exact numbers differ between different window functions, but for a Hamming window with a main lobe width of four bins, the frame length should exceed four times the smallest discernable difference between partials. Or, phrased differently, it is not possible to reliably estimate the frequency of a partial unless the frame fits four periods of its oscillation. [8]

2.2.2.3 Summary on spectral peak tracking

Spectrum analysis is a versatile tool, used widely in digital signal processing. The spectral peak tracking algorithm proposed in [8] provides a full-scale analysis-synthesis system that can track an unlimited number of partials at once. Once the spectral peaks are identified, they can be subtracted in the frequency domain to allow for IFFT re-synthesis, or be used to model a sine wave that is then subtracted from the input signal in the time domain. This method requires in-phase re-synthesis, which in turn requires analysis of both the magnitude and phase spectra, but has the advantage that it decouples the frame length of the output from the frame length of the input. Spectral peak tracking does not rely on the input signal being harmonic, and depending on the spectral distribution of noise, the performance may be more or less affected by noise. The challenge of implementing spectral peak tracking for low-frequency estimation would be choosing the proper window function, to accommodate the requirements on frame length, search range, and discrimination between closely spaced partials.

2.2.3 Transient modelling synthesis

The issues of dealing with transient signals in the frequency domain are targeted in the three-part analysis-synthesis model called Transient Modelling Synthesis (TMS), developed by Verma and Meng [5]. TMS extends the two-part sinusoid-plus-residual model with a third transient part, which is analysed and synthesised separately from the rest.

The method is built on the discrete cosine transform (DCT), which mathematically is defined as follows:

$$C(k) = \beta(k) \sum_{n=0}^{N-1} x_n \cos\left[\frac{(2n+1)k\pi}{2N}\right] \quad \text{for } n, k \in [0, 1, 2, ..., N-1] \tag{2.10}$$

where $\beta(k) = \sqrt{1/N}$ for $k = 1$ and $\beta(k) = \sqrt{2/N}$ otherwise. By transforming the signal into the frequency domain in this manner, a time domain transient becomes a frequency domain oscillation, which in turn can be modelled by a regular STFT as if it were a time domain signal. In other words, the signal is transformed from the time domain to the frequency domain, and then back into a time-like domain. The added transformation step rotates the signal in a way that effectively flips the

time and frequency axis, with the consequence that their respective resolutions also are interchanged. This is elegantly displayed in a series of figures by the authors of the original paper – which an interested reader should look into, since it is fairly short and very well-written [5].

In signals with prominent transients, the authors recommend applying the three analysis steps in the order of Transient, Sinusoidal, Noise [5]. There are no restrictions on which methods to apply in the two latter steps, meaning that this method is supposedly easy to combine with the spectral peak tracking procedure described earlier in this chapter.

2.2.3.1 Tolerance to noise and inharmonicity

The idea of TMS is to model transients separately from sinusoids, allowing for a more precise sinusoid analysis and a more realistic transient re-synthesis. Since transients pose a difficulty for spectral peak tracking models, the performance of such a model should thus be improved if the transients can be removed from the signal before the sinusoidal analysis step takes place. In this sense, adding TMS as a prior step to "clean" the signal before performing spectral peak analysis, should improve the performance of the peak tracking algorithm.

2.2.3.2 Frame length and computational load

TMS requires the transient to only makes out a small portion of the frame, and a frame length of 1 s is thus suggested in the original paper [5]. This unfortunately disqualifies the method from real-time use, since such a latency is approximately 50 times longer than the set requirement.

2.2.3.3 Summary on transient modelling synthesis

If implemented in an offline setting, TMS is a highly flexible approach that should improve the analysis of most audio signals. It does however require a frame length of approximately 1 s, which would create a latency in the same order. A challenge in implementing TMS it is that of asserting that no model "steals" signal components from another; the sinusoidal model should stick to modelling sinusoidal components, the transient model should stick to transients, etc. This issue could be solved by large scale quantification of the signal's tonality and amplitude envelope, which is suggested in [5], but again, won't be possible with high demands on a short lookahead. All in all, TMS is developed for offline analysis, at which it seems to complement the typical sinusoidal plus residual models well.

2.3 Signal separation

Signal separation is the act of deconstructing a mix of signals into its subcomponents. Depending on the richness of the signal, the subcomponents can range from individual sinusoids in one recorded signal, to signals of considerable complexity, such as instrument tracks in a music recording. The approach to signal separation

must be adapted to the complexity of the signals and subcomponents involved, and this section focuses on the first case: methods of low-level signal separation or partial deconstruction. For a comprehensive summary of a broader set of signal separation methods, the reader might look into the recollection found in [9]. Since the aim of this study is full signal separation of individual partials, broadband and narrowband methods such as filtering are not considered. The two methods presented here are instead time domain subtraction and frequency domain subtraction.

2.3.1 Time domain subtraction

If a signal is modelled as a linear combination of a set of components (such as the three-part model described in Section 2.1), simple linear operations such as addition and scaling can be used to combine them freely in the time domain. If a signal $x(t)$ consists of a known component $a(t)$ plus some residual $b(t)$, the unknown residual can be isolated by subtracting $a(t)$ from $x(t)$ as shown in Eq. (2.12).

$$x(t) = a(t) + b(t) \quad \Leftrightarrow \tag{2.11}$$

$$\Leftrightarrow \quad b(t) = x(t) - a(t) \tag{2.12}$$

Once both signal components are known, the original signal can be reconstructed and possibly altered. Each component can be scaled by any constant or modulating function, creating a mix of the original components that can vary over time. Scaling an amplitude to zero effectively removes that component from the output signal, without affecting any properties of any other component. Eq. (2.14) shows how such an output signal can be reconstructed from $x(t)$ and $a(t)$, using any time-variant scaling functions $A(t)$ and $B(t)$.

$$x'(t) = A(t) \cdot a(t) + B(t) \cdot b(t) \tag{2.13}$$

$$= A(t) \cdot a(t) + B(t) \cdot [x(t) - a(t)] \tag{2.14}$$

Signal separation by subtraction is a very powerful tool, but it requires exact knowledge about at least one of the two signal components of every subtraction (of course, more than two signal components can be found and separated in one signal, but the separation process can then be modelled as a series of subtractions including two components: one target and one residual). Essentially, the component to be removed must be entirely known in each step, for the operation to work as intended. In the case of a sine wave, we must thus estimate not only the frequency of the target, but also its phase and amplitude, at intervals that are as close to instantaneous as possible. Only once we know enough about the target to model it, we can perform the subtraction. An error in the estimated frequency of the target will result in slow amplitude modulation of the target in the output (commonly known as "beats"), and any errors in amplitude estimation will lead to imperfect subtraction. Phase estimation errors will also lead to imperfect subtraction, with the addition of a phase shift of the remaining target component. If the frequency, phase and amplitude of

the target are well-estimated, however, subtraction in the time domain offers high flexibility and accuracy, allowing for zero phase effects and zero spectral leakage between partials.

2.3.2 Frequency domain subtraction

It is possible to perform subtraction in the frequency domain as well. Fourier transform is a reversible operation, and it can be treated as a linear operation on individual sinusoids or any signal components, as long as windowing effects are considered (see Section 2.2). Eq. (2.18) displays how the convolution (*) between the transforms of the window function $h(t)$, and the respective linear components $x(t), a(t)$ and $b(t)$ of a signal, allows for linear addition and subtraction in the frequency domain.

$$\begin{aligned} x(t) &= a(t) + b(t) && (2.15)\\ \mathfrak{F}[x(t) \cdot h(t)] &= \mathfrak{F}[(a(t) + b(t)) \cdot h(t)] && (2.16)\\ \mathfrak{F}[x(t)] * \mathfrak{F}[h(t)] &= \mathfrak{F}[(a(t) + b(t))] * \mathfrak{F}[h(t)] && (2.17)\\ &= \mathfrak{F}[(a(t)] * \mathfrak{F}[h(t)] + \mathfrak{F}[b(t))] * \mathfrak{F}[h(t)] && (2.18) \end{aligned}$$

Subtracting a sinusoidal component in the frequency domain is thus not as simple as setting a frequency bin to zero in the FFT; instead the transform of the window function used should be centred at the target frequency, and subtracted from the spectrum at all affected bin locations. The target frequency may or may not be centred in a frequency bin to begin with, but this can be solved by mapping a highly resolved window function transform onto the target frequency, and subtracting its values as they appear at the bin locations. Similarly, partials can of course be added to the spectrum by simply adding window transforms of desired scaling at any location. This allows for the "moving around" of partials on the spectrum, used in for example pitch correction algorithms. To perform a fully linear spectral subtraction, the operation must act on the complex Fourier coefficient; i.e. both in the frequency and phase spectra. In that case, the subtraction of a perfectly estimated sinusoidal component is identical to the corresponding time-domain operation described in the previous section. [8]

2.3.3 Summary on signal separation methods

Time domain subtraction and frequency domain subtraction are two versions of the same operation, separated by only a transform between the two domains. They are equivalent in terms of accuracy and linearity, assuming that their respective input parameters of frequency, phase and amplitude are perfectly estimated. In addition to the three, frequency-domain subtraction also requires knowledge of which window function was used in the original STFT.

2.4 Complementary studies

This section contains some additional theory used in the study, or necessary for understanding it.

2.4.1 Second degree polynomial interpolation

Second degree polynomial interpolation can be used in order to improve the accuracy of extreme points in a sparsely sampled data set. If the sampled array $y(j)$ has a local extreme point at $y(k)$, a second degree polynomial $z(j)$, defined in Equation (2.19), can be fitted to $y(k)$ and its immediate neighbours $y(k-1)$ and $y(k+1)$.

$$z(j) \quad = \quad aj^2 + bj + c \tag{2.19}$$

The symmetry line of $z(j)$, here denoted $k_{\text{interp.}}$, indicates the location of its extreme point, as seen in Equation (2.20) and Figure 2.9.

$$k_{\text{interp.}} \quad = \quad -\frac{b}{2 \cdot a} \tag{2.20}$$

Assuming that the interpolation is representative of the sampled data, $z(k_{\text{interp.}})$ thus provides a more accurate estimation of the actual extreme point in the data set than the sampled $y(k)$, and can be used in its place.

$$y_{\text{peak,interp.}} \quad = \quad z(k_{\text{interp.}}) \tag{2.21}$$

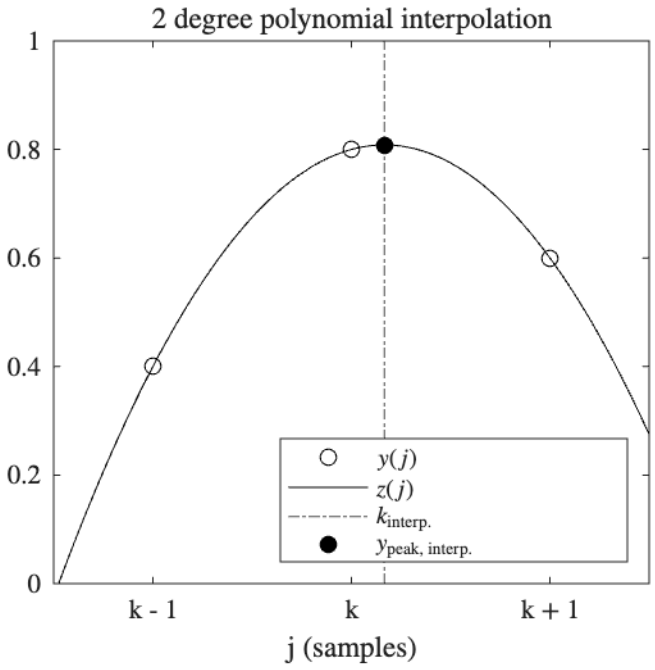


Figure 2.9: Second degree polynomial interpolation of three points in $y(j)$, resulting in an improved estimation $y_{\text{peak, interp.}}$ of the extreme point.

2.4.2 Trajectories, births and deaths

Trajectory matching refers to the act of deciding which partials from the current estimation that should be paired with which partials from the past estimation, in order to create accurate and meaningful partial tracks over time. If the number of detected partials is not constant, a partial tracking algorithm also needs the ability to decide when a partial is "born" (starts existing) and when it "dies" (stops existing). f_0 trackers do not have to deal with the issue of assigning tracks to its estimations as they only track one partial, but are instead often equipped with a voiced/unvoiced detector, deciding whether or not a vowel (and thus, a fundamental) is present in the signal at all.

2.4.3 A few words on step size

The step size decides the speed (or rate of update) of an algorithm. The block size decides the speed of the computations, the resolution of an FFT, and the demands on memory. Regular overlap analysis (for example FFT) assumes that the output block is of the same length as the input, and that it needs to be, because the synthesis occurs through IFFT and overlap adding. However, if the synthesis does not happen through IFFT, there is no law saying that the input and output blocks must be of same length.

2.5 Deciding on a method

This section summarises the findings of the theory chapter, and highlights the benefits and limitations of the proposed methods. Subsection 2.5.1 presents the four criteria on which the methods are evaluated, and the following subsections evaluates them one by one. The decision is made and a final motivation provided in Subsection 2.5.

2.5.1 Criteria and method of evaluation

Considering the properties of toms described in Section 2.1, three key properties are formulated that a partial tracker should have in order to successfully operate on a recording of a tom: ability to detect and reproduce fast pitch changes, ability to estimate inharmonic partials, and ability to function regardless of prominent noise and transient features. In addition to these three properties, the algorithm should fulfil the requirements of real-time compatibility set in Chapter 1: an online approach, a frame length shorter than 1024 samples, and a computing time shorter than the step size in ms. The list below summarises the desired properties, and translates them from qualitative to quantitative where possible.

Criteria

- Real-time compatible → short frame length, fast computing time, online
- Ability to handle fast pitch changes → short frame length
- Ability to handle inharmonic partials

- Ability to handle transients and noise

To begin with, it should be established that the choice of signal separation method can be removed from any criteria or performance discussions: time domain subtraction and frequency domain subtraction are equivalent, so the decision should fall on whichever domain fits best with the chosen method for frequency estimation and synthesis. The decision on synthesis method is also a question of domain compatibility, for which reason the decision on frequency estimation algorithm must be made first. The algorithms are evaluated on basis of the criteria above; a summary of the methods for frequency estimation is provided in Table 2.1.

Table 2.1: Summary on methods for partial and fundamental frequency tracking.

Method	Pros	Cons
Lag domain	• Short frame length: $2 \cdot T_0$. • Fast enough for real-time implementations.	• Assumes harmonic partials. • Phase and amplitude estimation require additional steps. • No explicit method for dealing with transients.
Spectral peak tracking	• Does not assume harmonic partials. • Phase and amplitude included.	• Longer frame length; $4 \cdot T_0$, depending on window choice. • Longer expected computation time due to transforms. • No explicit method for dealing with transients.
Transient modelling synthesis	• Does not assume harmonic partials. • Phase and amplitude included. • Explicitly handles transients. • Allows for individual processing of sinusoids, transients and noise.	• Requires ≈ 1 s frame length. • Increased complexity; more parameters to tune. • Longest expected computation time due to multiple transforms.

2.5.2 Real-time compatibility

Best candidates: Lag domain and Spectral peak tracking.

In Table 2.1, it can be seen that the lag domain methods provide frame lengths of two times the fundamental period, spectral peak tracking requires the double,

and TMS needs approximately 1 s, which cannot be considered real-time by any standard. The choice is thus immediately reduced to the two former methods: lag domain or spectral peak tracking. The typical floor tom fundamental frequencies range down to around 70 Hz, corresponding to a period of around 14 ms. Two periods of the lowest expected fundamental would consequently be 28 ms, meaning that even the lag domain methods, being the fastest of the three, would fail to live up to the expectations of a 23 ms frame length. If the 23 ms requirement is to be fulfilled, this would lead to a lowest detectable frequency of 88 Hz with a lag domain method, and 176 ms with spectral peak tracking. This seems to be yet a plausible reason for the lack of drum-tailored f_0- and partial trackers: the long periods of the fundamental makes it impossible to construct trackers with low enough latency for real-time applications.

But there is a way around the low frequency, high latency issue. If the step size is chosen to be shorter than the block length, an internal memory can be implemented to keep track of prior blocks. That way, the algorithm could use longer analysis blocks, consisting of both current and previous steps, while only producing output of a length corresponding to the step size. In order for this to reduce the response time of the algorithm, in must be accepted that a signal frame with a length corresponding to the difference between the step size and the block size passes through unprocessed, as the initial signal buffer is waiting to fill up. With a past signal buffer analysis scheme such as this, it is possible to reduce the step size below the required 23 ms while keeping the frame length long enough for low frequency estimation. Assuming that the step size is set to one period of the lowest expected fundamental (14 ms), this would result in an initial buffer gap of another 14 ms for the lag domain methods, or 42 ms (three periods) for spectral peak tracking.

2.5.3 Ability to handle fast pitch changes

Best candidates: Lag domain.

The second reason for keeping the frame length short, as stated in the criteria, is that it improves the algorithm's ability to handle fast pitch changes. The argument here is simple; all short-time estimation methods assume that the signal being analysed is divided into frames that are short enough to be considered stationary within each frame, and for a non-stationary signal this becomes less and less true the longer frames that are used. Both lag domain methods and spectral peak tracking assume the estimated parameters to be constant, so if a parameter is changing within one frame, the estimation will be an average of the actual, time-varying value. Assuming the step size is constant, using shorter frames with less overlap minimises the parameter changes within each frame, so that the results of a short-frame analysis contain less backwards-averaging and is more up to date.

2.5.4 Ability to handle inharmonic partials

Best candidates: Spectral peak tracking and TMS.

Lag domain methods are not mathematically designed to deal with inharmonic partials, whereas spectral peak tracking (and, consequently, TMS) is indifferent to the harmonicity of a signal. However, the effect that inharmonic signals have on lag domain accuracy depends on relative amplitude and distance between the partials, and a preliminary informal test run shows that the effect is present but not detrimental to the results.

2.5.5 Ability to handle transients and noise

Best candidates: TMS.

In terms of handling transients, only TMS has an explicit tactic for doing so, although in exchange for a disqualifying frame length requirement. Between lag domain and spectral peak tracking, there is no clear way of comparing the methods in this regard without first building them and performing a test. It should however be noted that a method using longer frames will see the effects of past transients in the analysis for a longer time afterwards, which adds another argument in favour of the lag domain methods.

2.5.6 Decision

The study chose to implement a lag domain method to solve the problem of real-time inharmonic partial tracking. The choice of method thus fell on the YIN algorithm [1], in combination with time domain subtraction. Although not initially intended for multi-partial tracking, the YIN algorithm can be applied iteratively in an analysis-synthesis loop, where each partial is re-synthesised and removed by time domain subtraction, before the signal is analysed again. The runner-up method by a very close margin is spectral peak tracking, which, in combination with past signal buffer analysis, should provide very accurate results for inharmonic signals with a slow rate of pitch change, and low to moderate amplitude of the transients.

3
Methods

This chapter describes the methods of the study in three main sections. Section 3.1 formalises a method for separating tonal and atonal signal components in potentially inharmonic audio signals, called the Iterative Partial Tracking (IPT) method. The IPT method was formulated as a part of this study, and is intended to serve as a framework for designing partial trackers using single-frequency estimation methods. Section 3.2 describes the YIN-based implementation developed in this study, including the methods used for time-stepping, frequency estimation, amplitude and phase estimation and synthesis, as well as the surrounding functions needed for multi-partial tracking. Lastly, the methods of evaluation and testing are described in Section 3.3.

3.1 The IPT method

The IPT method is a block-based method of separating a potentially inharmonic audio signal into one or multiple tonal components and one atonal component, by iteratively estimating and adding/subtracting partials in the form of pure sine waves. The method allows for multi-partial tracking using single-frequency estimation methods such as f_0 trackers, without the need of spectral processing.

Iterative Partial Tracking algorithm

For each block of signal, the following steps are performed:

1. Find a partial by some frequency estimation algorithm (estimate frequency, phase and amplitude)
2. Reconstruct the partial as a sine wave
3. Subtract the partial from the signal in the time domain
4. Let the resulting signal be the new input into step 1, allowing for a new partial to be found
5. Iterate steps 1-4 for the same signal block until the desired amount of partials have been found
6. Match all found partials to the existing partial tracks
7. Step forward and process a new block of signal

Figure 3.1: The seven steps of the IPT method.

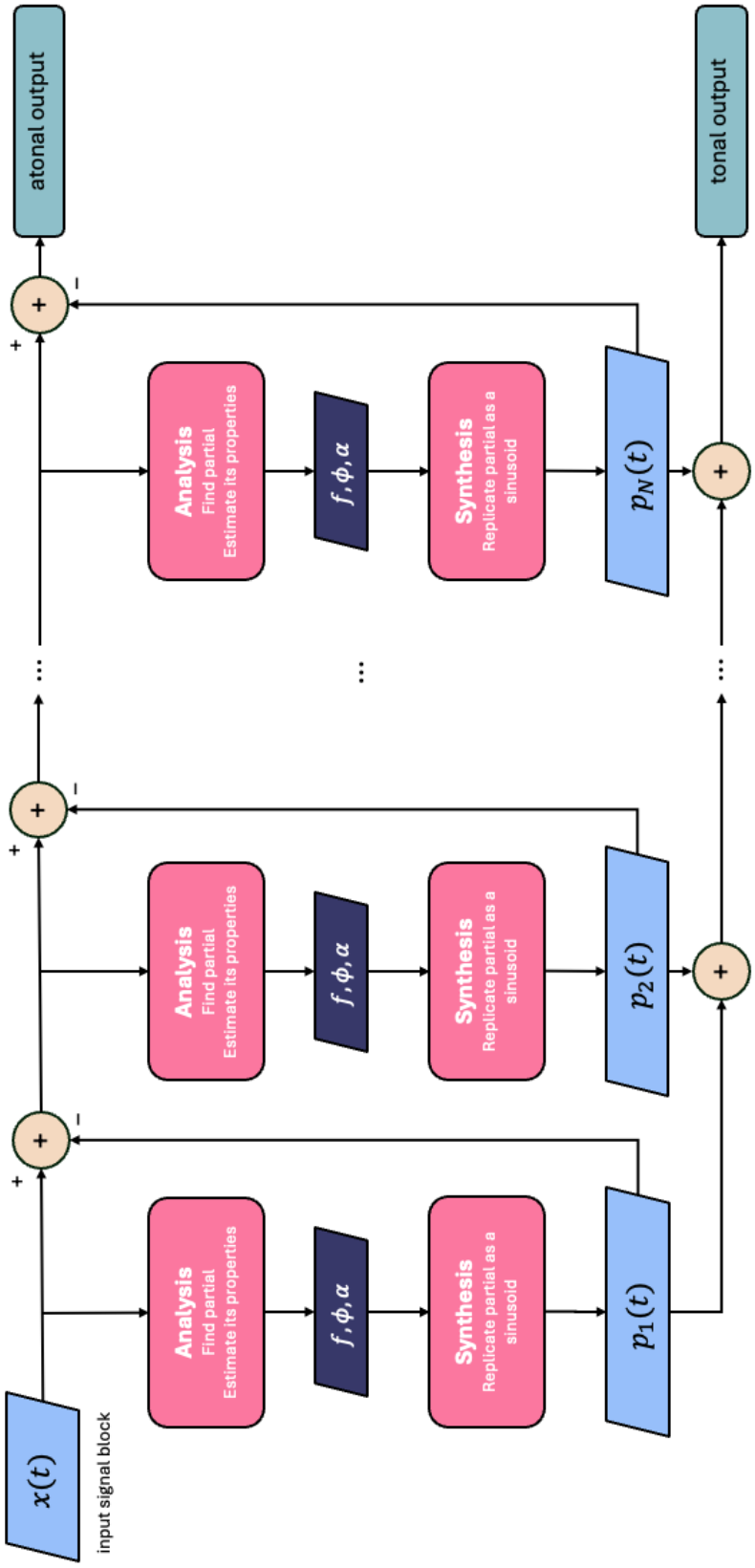


Figure 3.2: A flowchart illustrating the IPT algorithm. Each input signal block has its partials (p_1, p_2, ..., p_N) estimated, synthesised and subtracted one by one. The tonal output thus contains the sum of all estimated partials, whereas the atonal output contains the difference between the input and the tonal output. Each partial could also be given a separate output, to allow for partial-specific output channels.

Figure 3.1 describes the seven steps of the method, and Figure 3.2 displays it as a flowchart with a two-channel output: tonal and atonal. The three first steps of the algorithm (analysis, synthesis and subtraction), are indicated clearly in the flowchart. Steps four, five and seven describe the flow of the algorithm, and step six is not in the flowchart because it occurs after each block is fully processed. Step six in particular could furthermore be viewed as optional, if the implementation is not intended to run in real-time and the frequency estimation method does not require sorted partial tracks to function. The IPT method does not specify in which domain the frequency estimation should take place (it could be lag, frequency or other), but is does specify that the synthesis and subtraction should be performed in the time domain. This is, as was found when the implementation was designed, beneficial because it allows for the estimated partials to be extended into the upcoming signal block, which enables preliminary subtraction of last block's estimations for the first few iterations of the new block. An example of such an implementation is found in Section 3.2.

3.2 Implementation

A collection of functions were designed in order to implement the IPT algorithm in a MATLAB program. The following subsections describe the program, whose outline can be found in Figure 3.3. The details of the partial-specific steps are described under 3.2.3 Frequency estimation, 3.2.4 Phase and amplitude estimation, and 3.2.5 Synthesis of estimated partials. The signal cleaning and post-processing procedures are described under 3.2.2 Signal cleaning and 3.2.6 Time step post-processing. The variables and indexes used in the implementation are defined in Table 3.1.

Table 3.1: Variables and indexing conventions used in the upcoming chapters. Any subscript refers to a variable's value at the indicated index.

Var	Description	Range/Definition
j	index for samples	1, 2, 3, ..., j_{end}
t	index for time steps	1, 2, ..., t_{end}
n	index for partial numbers*	1, 2, ..., N
$x_{\text{full}}(j)$	input signal	$x \in [-1, 1]$ $j = 1, 2, ..., j_{\text{end}}$
$x_{t,n}(j)$	cleaned input signal block at time step t, partial n	$x_{t,n} \in [-1, 1]$ $j = 1, 2, ..., W$
$p_{t,n}(j)$	partial estimation at time step t, partial n	$p_{t,n} \in [-1, 1]$ $j = 1, 2, ..., (W + S)$
W	block size	length of $x_{t,n}$
S	step size	the increment in samples when stepping from $x_t(1)$ to $x_{t+1}(1)$
$f_{t,n}$	frequency estimation	$[f_{\min}, f_{\max}]$
$\phi_{t,n}$	phase estimation	$[0, 1/f_{\min}]$
$\alpha_{t,n}$	amplitude estimation	[0, 1]
$\tilde{p}_{t,n}, \tilde{\alpha}$	preliminary estimations	... of $p_{t,n}$ and α

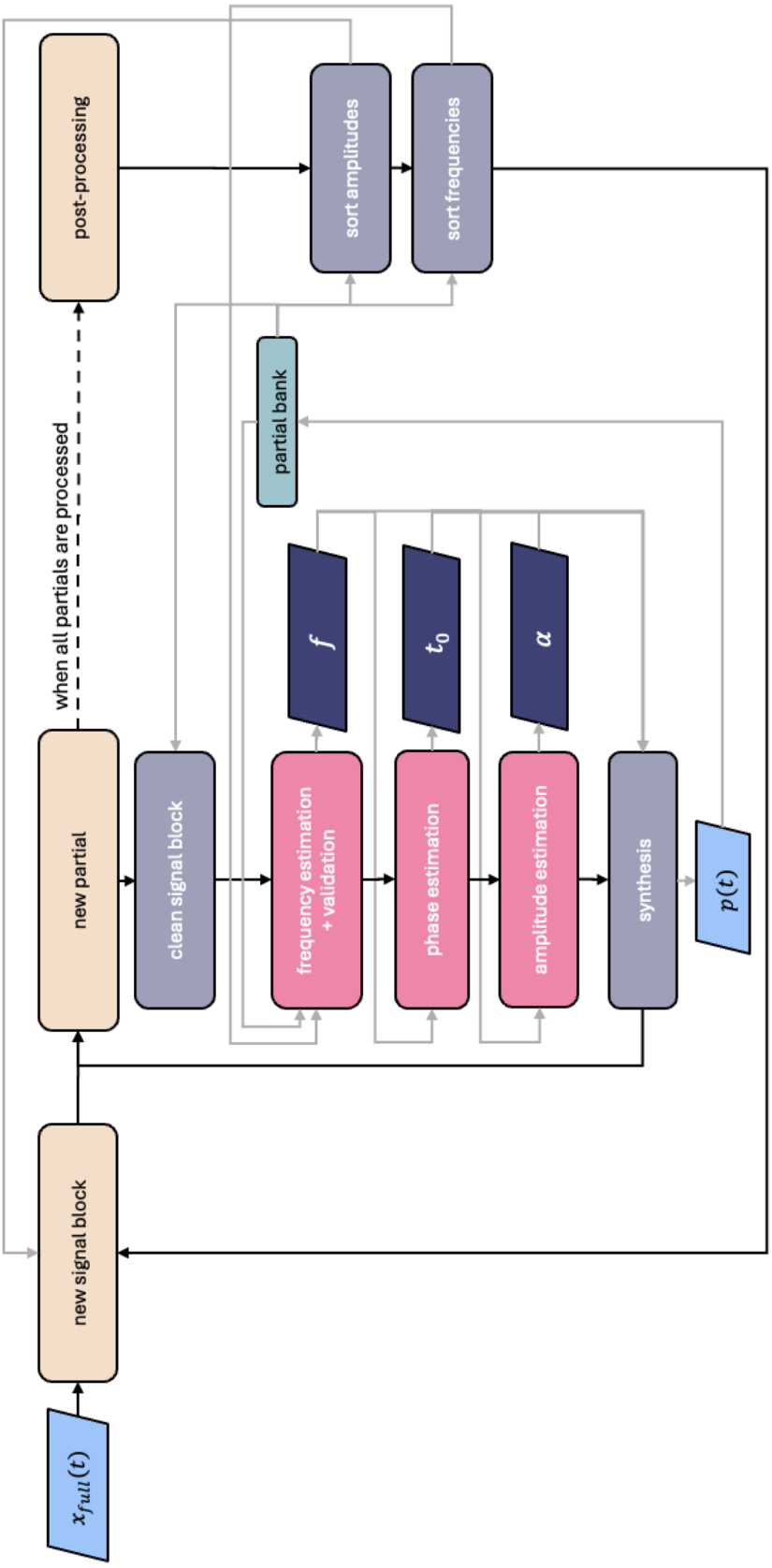


Figure 3.3: The structure of the MATLAB program developed in this study, implementing the IPT algorithm.

*** A comment on partial number indexing**
It makes intuitive sense to number the partials from lowest to highest in frequency, so this is how they will be presented in the results of this study. The reader should however be aware that the partials are not always estimated in this order – on the contrary, it was found preferable to estimate the partials in the order of their most recent amplitude, rather than their frequency. If not stated otherwise, it will be assumed in this chapter that n indicates the order in which the partials are estimated by the algorithm; so that $n = 1$ indicates the first iteration of a given time step, $n = 2$ indicates the second, and so on.

3.2.1 Block extraction and time-stepping

The implementation used a block size of $W = 2 \cdot T_{\max}$, where $T_{\max}$ is the number of samples corresponding to one period of the lowest expected frequency at the current sampling rate ($T_{\max} = f_s / f_{\min}$). The lowest expected frequency was set by the user. The step size was $S = T_{\max}$, resulting in 50% input overlap. The block extracting procedure was designed as follows:

$$x_t(j) = x_{\text{full}}(m) \qquad j \in [1, W], \quad m \in [(t-2) \cdot S + 1, t \cdot S] \tag{3.1}$$

where x_{full} (originally indexed from 1 to the full signal length) is the full input signal, and $x_t(j)$, indexed from 1 to W, is the extracted input block at time step t. Only the "second half" of each output block is used in the actual output, which, by accepting the initial "buffer collection gap" discussed in the theory chapter, allows for a buffer size of $T_{\max}$ samples.

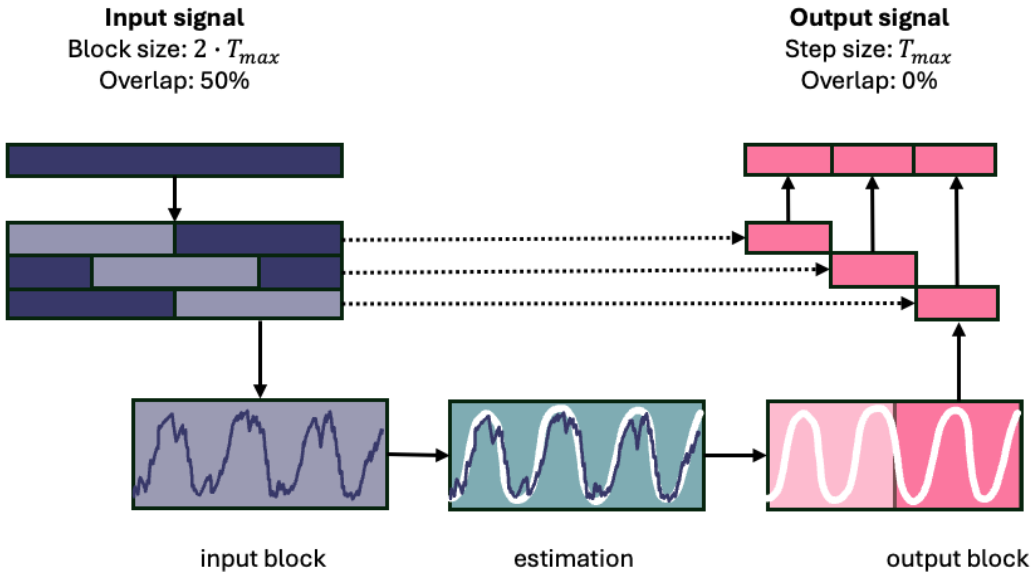


Figure 3.4: A 50% overlap input combined with a zero-overlap output requires that the first half of the output estimation is discarded.

Figure 3.4 visualises the input and output blocks; the light pink segment of the output block is discarded since that part of the signal has already been synthesised by the previous time step.

3.2.2 Signal cleaning

In order to find new partials in every iteration of n, the input signal is continuously "cleaned" from the partials that are already identified. The basis input signal for each time step, x_t, was defined in Section 3.2.1, but each iteration of n requires its own unique input, called $x_{t,n}$.

For an arbitrary partial $p_{t,n}$ at time step t, there will be a set of estimations $p_{t,1}$, $p_{t,2}$, ..., $p_{t,n-1}$ that were already established during the current time step. These are easily subtracted from $x_{t,n}$, as they have the exact same starting point in time. However, there is also a set of partials $p_{t,n+1}$, $p_{t,n+2}$, ..., $p_{t,N}$ that have not yet been established for this time step. Instead of ignoring these, potentially allowing for a mix-up between $p_{t,n}$ and any of the undetermined partials, the program subtracts extended versions of last time step's estimations of the partials, when cleaning the current. This requires the synthesis of partials to span not only the current block, but the upcoming as well.

$$x_{t,n}(j) \quad = \quad x_t(j) \quad - \quad \sum_{k=1}^{n-1} p_{t,n}(j) \quad - \quad \sum_{k=n+1}^{N} p_{t,n}(j+S) \tag{3.2}$$

3.2.3 Frequency estimation

The frequency estimation algorithm was divided into three main steps: correlation, minimum selection, and validation. These are explained one by one in the following subsections.

3.2.3.1 Correlation

The correlation function computes the lag curve according to steps 2 and 3 of the YIN algorithm [1], as described by Equation (3.3),

$$d_{t,n}(\tau) = \sum_{j=1}^{W-\tau_{\max}} \left(x_{t,n}(j) - x_{t,n}(j+\tau)\right)^2 \qquad \tau \in [0, \tau_{\max}] \tag{3.3}$$

where $d_{t,n}(\tau)$ is the difference function at time instance t for partial number n, with respect to the lag τ, and $x_{t,n}$ is the input signal block at time instance t, for partial n, indexed $j = 1, 2, ..., W$, W being the length (or *size*) of the block in samples. Note that Equation (3.3) is identical to Equation (2.4) presented in the theory chapter, with the exceptions of a redefinition of the block size W, and a slight change in time indexing convention, more similar to array indexing in MATLAB. The CNDF, as presented in Eq. (2.5) is applied to the results of Eq. (3.3) to produce a normalised lag curve, containing one or multiple minima at lags corresponding to strong periodic

components of the signal. An example of a typical lag curve can be found in Figure 3.5.

3.2.3.2 Minimum selection

The minimum selection method proposed in [1] proved to be inefficient, when the source material has strong inharmonic partials. The alternative method chosen goes as follows:

1. Apply a short (≤ 10 samples) mean-value smoothing to the lag curve, to reduce the number of local minima in close proximity to one another.
2. "Lowpass" the lag curve by shortening it to the longest allowed lag for this specific iteration and partial.
3. Choose L initial lag candidates by finding all the local minima of the lag curve, and pick the L shortest lags that fall below a certain threshold.
4. "Highpass" the list of candidates by discarding lag candidates that are shorter than the lower lag limit for this specific iteration and partial.
5. Discard lag candidates that are too close to already established partials.
6. Discard lag candidates that are multiples of other minima in the curve, to prevent sub-harmonic errors.
7. Pick the shortest lag that remains.

The lower lag limit of step 4 is not strictly necessary for the algorithm to work, but it was found to improve the results considerably. If no limit is set, the partials tend to diverge upwards in frequency when their amplitudes decrease, as the program is set to favour low lags. It is possible to make both the upper and the lower limit dynamic, so that they adapt to previous measurements and/or other partials' estimations. This was found especially useful when tracking multiple partials.

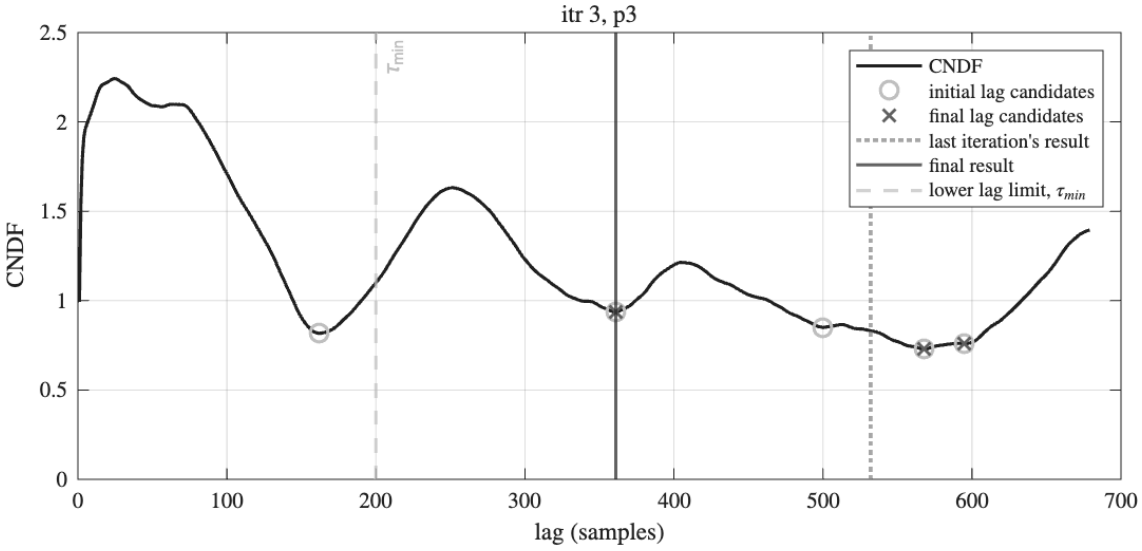


Figure 3.5: The lag curve corresponding to a typical higher-order partial. The curve shows multiple minima of similar prominence, and the minimum selection algorithm applies a set of rules to determine which lag value to select.

Figure 3.5 shows a typical lag curve for one of the higher order partials; it has multiple minima of approximately equal depth. There are 5 minima below the threshold of 1; these are the initial candidates from step 3 in the list above. The first candidate is discarded for being below the lower lag limit, and the third candidate is discarded due to being too close to another established partial (not visible in the figure). Three candidates remain and the shortest one, $\tau_{\min}$, is selected and moves on to the validation stage.

3.2.3.3 Validation

The validation stage ensures that the selected lag is reasonable, and deals with the possible situation that no valid lags were found in the previous stage. There are many ways to do this, but this implementation validated against a weighted backlog of previous lag values, and defined an accepted deviation of new additions based on the standard deviation of the backlog. A soft exponential weighting on the form $w(k) = {C_B}^k$ was applied, emphasising the most recent values of the backlog. k here represents the indexing of the backlog, where $k = 1$ is the oldest entry and $k = W_L - 1$ is the most recent. The backlog was between 10 and 20 iterations long (corresponding to approximately 0.15 to 0.3 seconds of signal, if $f_{\min}$ is set to 65 Hz), and included the current estimation as well.

In order to improve the accuracy of successfully validated lags, second degree polynomial interpolation was implemented. After validation and interpolation, the period was transformed into its corresponding frequency, using Eq. (3.4).

$$f_{t,n} = \frac{f_s}{\tau_{\min}} \tag{3.4}$$

where $f_{t,n}$ is partial n's frequency at the current time index t, and f_s is the sampling frequency of the signal.

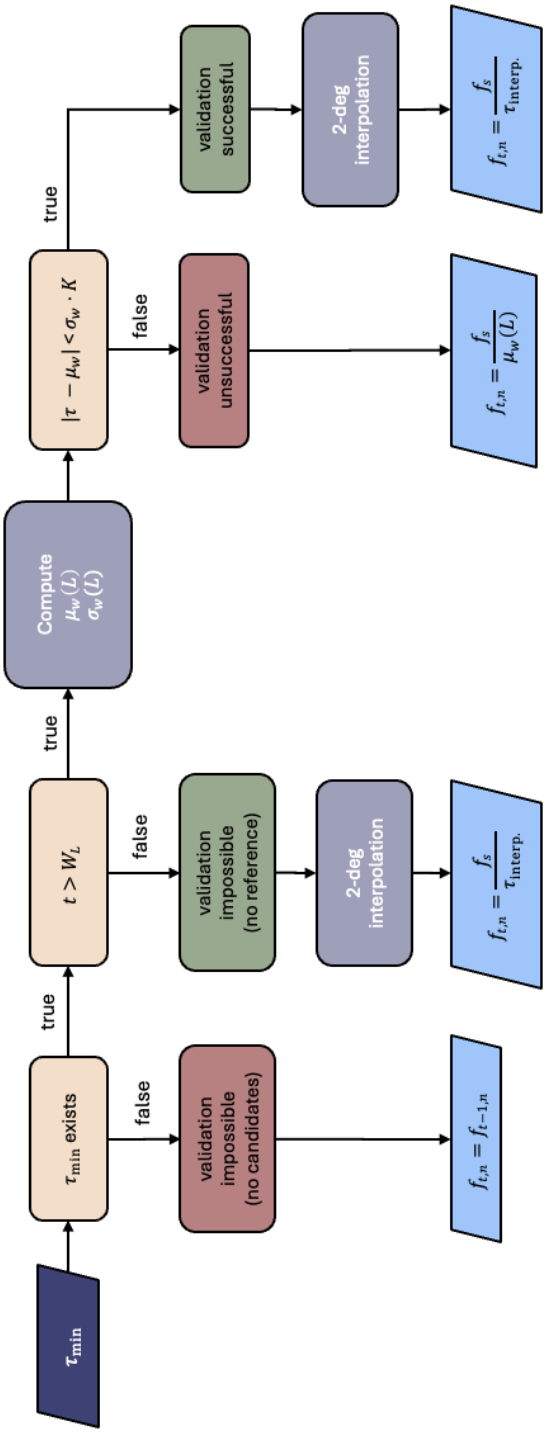


Figure 3.6: A flowchart describing the validation process. First, it is assured that the previous module produced a candidate, $\tau_{\min}$. If not, last iteration's frequency estimation is used and instantly validated. Secondly, it is assured that the backlog is entirely filled up. If not, the current lag candidate is instantly validated, and added to the backlog. The third step is the actual validation, where the current lag candidate is compared to the backlog. If within acceptable range, it is validated, added to the backlog, and passed on to interpolation. If not, the weighted mean of the backlog is used as lag candidate, without interpolation. In this case, the mean of the current candidate and the weighted backlog is added to the backlog.

3.2.4 Phase and amplitude estimation

The cross-difference function was used for estimating the phase of the partials in each block. Equations (3.5) and (3.6) define the process:

$$\tilde{p}_{t,n}(j) = \sin\left[2\pi \cdot f_{t,n} \cdot j\right]) \qquad j \in [1, W - \tau_{\phi,\max}] \tag{3.5}$$

$$c_{t,n}(\tau_\phi) = \sum_{j=1}^{W-\tau_{\phi,\max}} \left(\tilde{\alpha} \cdot \tilde{p}_{t,n}(j) - x_{t,n}(j + \tau_\phi)\right)^2 \qquad \tau_\phi \in [0, \tau_{\phi,\max}] \tag{3.6}$$

where $c_{t,n}(\tau_\phi)$ is the cross-difference function with respect to the phase delay, τ_ϕ. $\tilde{p}_{t,n}$ is the synthesised partial estimation, which is multiplied by a preliminary estimation of the partial's amplitude, $\tilde{\alpha}$. $\tilde{\alpha}$ can be set arbitrarily but should approximate the expected amplitude of the partial; this study used the highest absolute value of the current signal block $x_{t,n}$, as $\tilde{\alpha}$. The cumulative normalisation algorithm of Equation (2.5) was applied in the same manner as when estimating the frequency.

In order to save some computing power for the higher-order partials, $\tau_{\phi,\max}$ can be set to the value of $\tau_{t,n}$ that was found in the frequency estimation. This is equivalent to limiting the phase delay to 2π radians. Such a tight restriction might however introduce issues with respect to the CNDF, in case the optimal phase lag happens to be close to zero. An extra 25 % were therefore added to $\tau_{\phi,\max}$ as a buffer, making sure that near-zero phase lags would be caught in the next period instead.

The minimum selection algorithm of the phase estimator was set to choose the lowest (in lag) minimum of the curve, and if its value was larger than the period of the partial, one period was subtracted from its value. Except for checking that exactly one lag minimum was found, no further validation process was applied to the phase estimation. The chosen phase lag and its neighbours were used as reference points for a second degree polynomial curve fit, just like in the frequency module. The interpolated phase, denoted $\phi_{t,n}$, allows for an update of the partial estimation $\tilde{p}_{t,n}$, as seen in (3.7):

$$\tilde{p}_{t,n}(j) = \sin\left[2\pi \cdot f_{t,n} \cdot (j - \phi_{t,n})\right]. \tag{3.7}$$

With the new partial estimation at hand, another cross-difference was calculated in order to determine the amplitude of the partial. An array of plausible amplitudes, in a range from zero to the largest absolute value of $x_{t,n}$, was generated at a resolution of 101 points. Then, the cross-difference function was applied again, now with respect to the amplitude array.

$$a_{t,n}(\alpha) = \sum_{j=1}^{W} (x_{t,n}(j) - \alpha \cdot \tilde{p}_{t,n}(j))^2 \tag{3.8}$$

No weighting or normalisation was applied to the scaling curve. The minimum selection algorithm was set to identify the amplitude resulting in a global minimum of the scaling curve, and determine that to be $\alpha_{t,n}$. In the case that $\alpha_{t,n} < 80$ dBFS, it is set to zero, thus temporarily "killing" the partial for the duration of the current block. By that, the estimation of the partial at the current time step is complete, and can be described by a sinusoid on the form:

$$p_{t,n}(j) = \alpha_{t,n} \cdot \sin\left[2\pi \cdot f_{t,n} \cdot (j - \phi_{t,n})\right]. \tag{3.9}$$

3.2.5 Synthesis of estimated partials

The synthesis equation of $p_{t,n}(j)$ was defined in Equation (3.9). It turned out to be useful to synthesise $p_{t,n}$ as spanning not only the current signal block but the upcoming block as well, in order to use it in the preliminary signal cleaning process described in Section 3.2.2. Equation (3.9) was thus applied on the range $j \in [1, W + S]$. The addition of the block output to the full-length output technically only requires a slight reformulation of Equation (3.1), as found in Equation (3.10):

$$p_{t,n}(j) = p_{\text{full},n}(m) \qquad j \in [S+1, W], \quad m \in [(t-1) \cdot S + 1, t \cdot S], \tag{3.10}$$

where $p_{\text{full},n}$ refers to the full-length output signal of partial p_n.

However, assuming that any of the parameters differ between the last estimation and the current, it is reasonable to expect a discontinuity of the partial estimation at the border, where $p_{t-1,n}$ ends and $p_{t,n}$ is taking over. Such a discontinuity will be audible in the tonal output, and thus a strategy for smoothing or interpolating between blocks is required.

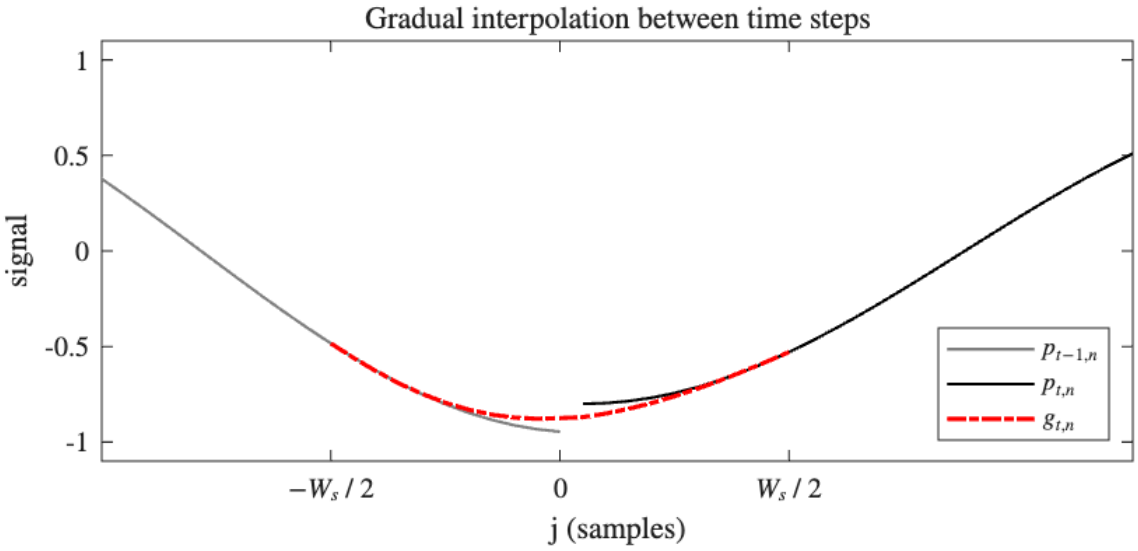


Figure 3.7: To assure that there is no discontinuity at the block borders, a gradual interpolation is performed between blocks.

This study used gradual interpolation as the primary transition strategy between blocks. Figure 3.7 shows an example of the procedure, which contains the following steps: A smoothing range W_S is set by the user, determining the length in samples of the window, centred at the block border, over which a second degree polynomial $g_{t,n}$ is fitted to the estimated signals. W_S should be considerably smaller than half the expected period of any partial, in order to provide a good approximation of a sine wave. Then, a linear weighting is applied, such that the resulting signal fades from $p_{t-1,n}$ to $g_{t,n}$ to $p_{t,n}$, asserting continuity of both the signal and the signal's derivative. The method is not compatible with real-time implementation, as the end of $p_{t-1,n}$ cannot be determined until $p_{t,n}$ has been established. This is the only part of the implementation that does not follow the real-time requirements posed, and other solutions to the inter-block transition problem are suggested in Chapter 5.

3.2.6 Time step post-processing

After all partials have been identified, some post-processing is required before moving on to the next time step. In this implementation, the trajectory matching occurs implicitly by means of the validation process described in Section 3.2.3, and the sorting algorithms described below. Births and deaths are implicitly handled by the amplitude estimation described in Section 3.2.4. The two types of sorting that occurs in the post-processing stage are amplitude sorting and frequency sorting.

3.2.6.1 Amplitude sorting

After all partials have been estimated in a time step, they are sorted from strongest to weakest, based on their respective value of $\alpha_{t,n}$. The amplitude-sorted list of partials decides the order of processing in the next time step, to make sure that the partials of largest amplitude are estimated first.

3.2.6.2 Frequency sorting

If a certain number of time steps has passed since the transient occurred or the signal began (if $t > t_{\text{sort}}$), the partials are also sorted based on their estimated frequency. The purpose of this is to impose frequency restrictions on upcoming iterations, to reduce the risk of partials swapping places over time. After the frequency-sorted list of partials has been established (now, n indicates the frequency ranking so that p_1 is the lowest partial with respect to frequency), each partial's allowed lag range is set according to the following rules:

- $p_{n<N}$ is only allowed lags higher than the average of p_{n+1}'s three most recent lags, multiplied by 1.01.
- $p_{n>1}$ is only allowed lags lower than the average of p_{n-1}'s three most recent lags.

The frequency sorting sets the dynamic lag limits referenced in Section 3.2.3.2, which effectively eliminates all frequency domain overlap of lag candidates before a decision is made.

3.3 Evaluation

This section describes the methods, settings and sample library that were used in the evaluation of the algorithm. The evaluation was divided into three parts: accuracy, settings dependency and influence of signal properties. The accuracy evaluation was performed on test runs with favourable settings and input signals, whereas the effects of changing settings and input signals were explored in the two latter parts, respectively.

The general procedure of the evaluation was the following:

- Run the algorithm on an audio signal from the sample library
- Export the output as $N + 1$ audio files, one for each partial and one atonal signal
- Plot frequency estimations and amplitude estimations partial-wise
- Produce STFT:s, in the form of spectrograms, for the input signal and the atonal output signal, respectively.
- Interpret and analyse the results.

Three measures of accuracy were used in the study: frequency estimation accuracy, amplitude accuracy and separation accuracy.

3.3.1 Frequency estimation accuracy

Frequency estimation accuracy was evaluated by computing the gross pitch error (GPE) and the fine pitch error (FPE) of the partial-wise frequency estimations $f_n(t)$ as a function of the time index. GPE is the standard indicator of pitch estimation accuracy, defined as the relative occurrence of estimation errors larger than 20 %. A frequency error of 20 % corresponds to approximately four semitones, which would be a devastatingly large error in a musical context. A second measurement, called FPE or fine pitch error, was therefore introduced. It is defined in the same way as GPE, but with an error tolerance of 6 %, corresponding roughly to one semitone.

In order to define an error, one needs a set of reference data, *ground truth*, to which the estimate can be compared. Since this study used set of recorded signals with no prior ground truth defined in terms of the partials' frequencies, the spectral median of each partial was used. The following steps were followed to compute the spectral medians:

1. The input signal was plotted as a spectrogram.
2. The N strongest partials in the graph were identified by visual inspection.
3. For each partial, a frequency range was defined.
4. For each partial, a time range where no other partial enters its frequency range, was defined.
5. For each partial, within its given time and frequency ranges, a spectral centre of gravity was generated using MATLAB's function `medfreq`.

The GPE and FPE were computed as percentages of gross and fine pitch errors on

the range where a ground truth could be established for each partial. Since the algorithm was set to discard estimations below -80 dB, the spectrograms were set to a fixed magnitude range of [-80, -20] dB, to better allow for visual assessment of when the partials' magnitudes fell below the -80 dB limit.

3.3.2 Amplitude and separation accuracy

The estimated amplitude curves $\alpha_n(t)$ were examined to get an indication of the quality of individual partial's estimations with regards to all estimated parameters. Here, a smooth and increasingly linear amplitude decay was considered the target result, as that is the typical modal decay behaviour.

To further evaluate whether the estimations of frequency, phase and amplitude were sufficient to allow for precise signal separation, the input signals' spectrograms were visually compared to the atonal output signals' spectrograms. Once again, the spectrograms were set to a fixed magnitude range of [-80, -20] dB. The frame length of the spectrograms was set to $15 \cdot f_s / f_{\min}$, with $14 \cdot f_s / f_{\min}$ overlap. Residue in the output spectrum, at or around the partial's location, was interpreted as an indication that the partial was not perfectly estimated.

3.3.3 Sample library

The sample library consisted of professionally recorded floor tom hits, provided by Klevgrand Produkter AB as featured in their commercially available drum sample packs. The initial library included hundreds of recordings of a range of drums, sampled at different intensities and with some physical modifications, such as damping and choice of mallets. This study chose to focus on two specific drums, Floor Tom 70's and Floor Tom Margaret, which generally performed well in preliminary testing, in order to keep the amount of alterable variables low. Each drum sample was converted to mono by isolating the left channel. No additional compression or gain staging was applied to the signals.

3.3.4 Signal properties

The signal properties evaluated was damping (which strongly correlates to signal duration), and signal velocity. Velocity is a music production term referring to the intensity of a hit; a soft hit has low velocity and a hard hit has high velocity. Velocity is typically measured at a midi scale from 0 to 127, but this study only used the broad classifications soft, medium and hard (corresponding to roughly 1/3 of the midi scale each).

3.3.5 Settings

Unless stated otherwise, all signals were processed using what was found to be the optimal settings, respectively. Generally, a block length slightly longer than two periods of the expected fundamental was used, and between 1 and 4 partials were tracked in the frequency range of [60, 270] Hz. Beyond this, two input parameters

were evaluated separately, in order to describe the behaviour of the algorithm. These were the number of tracked partials, N, and the block size, W.

The block size was altered by changing the block size factor w, otherwise set to 2. This enabled a variable block size $W = w \cdot f_s/f_{\min}$, while S remained fixed at $f_s/f_{\min}$. Increased block size thus entailed increased overlap, but static time resolution. Only two partials were tracked, as the addition of more partials than two resulted in algorithm failure at increased block sizes. The block sizes were changed in increments of whole periods; a block size factor of 5, means that the block is five times the length of one period of the lowest expected partial. A list of the input variables and settings of the program is provided in Table 3.2.

Table 3.2: The user-defined settings of the program, and their typical favourable values.

Symbol	Description	Function	Typical value
N	Number of partials	Global	4
w	Block size factor	Global	2
$[f_{\min}, f_{\max}]$	Frequency search range	Frequency est.	[65, 250]
L	Number of initial lag candidates	Frequency est.	8
$[k_{\text{op},1}, k_{\text{op},2}]$	Relative conflict range for other partials	Frequency est.	[0.95, 1.1]
$[k_{\text{h},1}, k_{\text{h},2}]$	Relative conflict range for sub-harmonic errors	Frequency est.	[0.95, 1.05]
W_L	Backlog size	Frequency val.	20
C_L	Backlog weighting factor	Frequency val.	1.5
K	Backlog tolerance factor	Frequency val.	1
W_S	Smoothing window length	Synthesis	100
t_{sort}	Iteration where frequency sorting begins	Global	10

4
Results

This chapter presents the results of the evaluation of the algorithm. Section 4.1 presents the accuracy evaluation of the output, where the algorithm was given favourable settings and input data. In Section 4.2, the outputs from different algorithm settings are compared on the same signals, in accordance with the second step of the evaluation method. In Section 4.3, the settings are kept constant but the properties of the input signals are varied instead. Section 4.4 summarises all of the findings. The GPE and FPE of all test runs, as well as the frequency search ranges used, are presented in Table 4.1.

Table 4.1: Gross and fine pitch errors per partial, in order from lowest order to highest frequency, for the set of test signals presented in this chapter. A value of 0.0 means that no errors were found in the available range. "undefined" means that there was no partial to match the estimation to. See respective graph and section for further comments.

Signal (velocity + characteristic)	**N**	**GPE (%)**	**FPE (%)**	**Range (Hz)**
Floor Tom 70's (hard)	4	0.0 0.6 0.0 0.0	4.14 11.1 10.6 50.4	[65, 220]
Floor Tom Margaret (hard)	4	0.0 0.3 0.0 6.8	0.0 4.6 15.4 32.5	[65, 255]
Floor Tom Margaret (hard + damped)	2	0.0 45.7	34.8 57.1	[60, 200]
Floor Tom 70's (mid)	4	0.0 undefined 1.3 0.0	21.3 undefined 15.4 98.1	[65, 220]

4.1 Accuracy

This section presents the results, with respect to frequency estimation accuracy, amplitude estimation accuracy, and separation accuracy, for when the algorithm is given favourable settings and input data.

4.1.1 Frequency estimation accuracy

Figures 4.1 and 4.2 show the frequency estimations of the four partials in two different floor tom recordings. As seen in the figures, the four estimations follow the four most prominent tonal components, with the estimation of the fundamental being the smoothest and initially most accurate in both cases. There seems to be multiple reasons for this, but the first and most obvious is the fact that the fundamental has a considerably higher amplitude than the other partials in the beginning of the signal. As discussed in Chapter 2, this prevents lag domain methods from being "distracted" by inharmonic partials when estimating the fundamental.

The relative frequencies of the partials are respectively [1, 1.38, 1.88, 2.29] for the 70's Floor Tom, and [1, 1.49, 2.04, 2.97], for the Margaret Floor Tom. Both these signals are thus inharmonic, but the Margaret is more harmonic than the 70's – in fact, the Margaret's partials can be seen as two sets of harmonic octaves ($p_1 + p_3$ and $p_2 + p_4$), with an inharmonic relation between the two fundamentals, p_1 and p_2. This could be part of the explanation of why p_2 is so well-estimated in Figure 4.2; it is enhanced by p_4, which happens to be its octave.

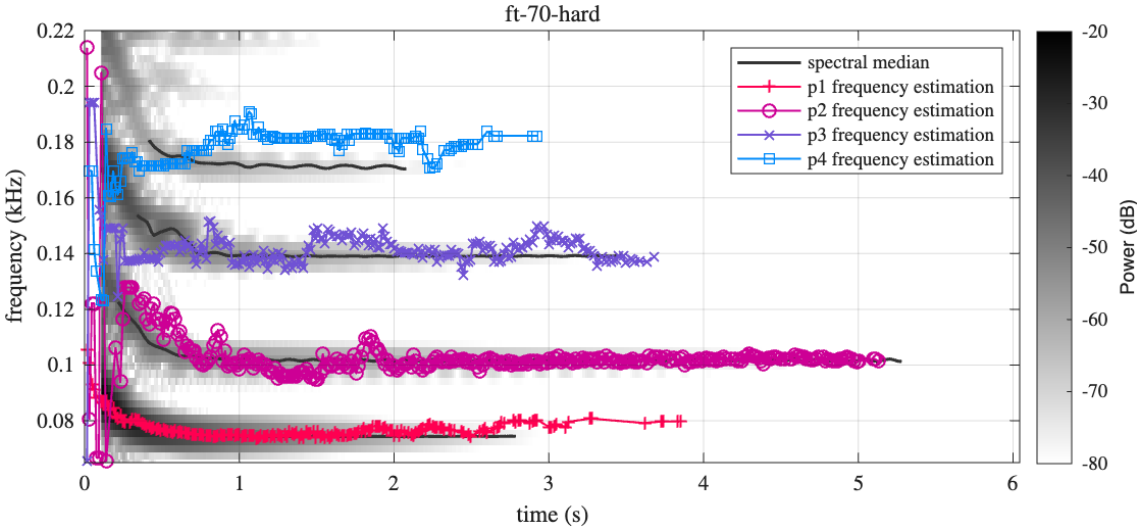


Figure 4.1: Frequency estimations of four partials in the sample Floor Tom 70's (hard), detected in a range of [65, 220] Hz. Block size: 1358 samples, with 50 % overlap.

The algorithm struggles to identify the higher-order partials in the initial segment of both signals; this proved to be true for all of the signals tested. Higher-order partials could generally not be established properly until they had stabilised at their final frequencies; a process that takes between 0.3 and 1 s, depending on the

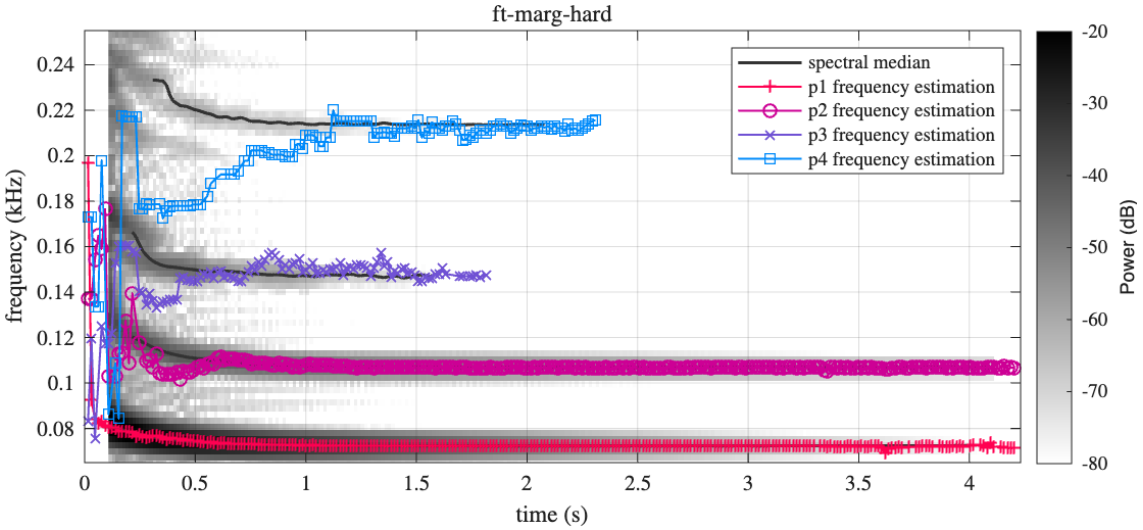


Figure 4.2: Frequency estimations of four partials in the sample Floor Tom Margaret (hard), detected in a range of [65, 255] Hz. Block size: 1358 samples, with 50 % overlap.

signal. The proposed reason for this is that due to being of higher frequency than the fundamental, higher-order partials fit more periods in one processing block, which creates phase offsets towards the end of the block if the frequency is not constant. When the frequency stabilises, this issue disappears and the estimations grow more correct.

4.1.2 Amplitude estimation accuracy

Figures 4.3 and 4.4 show the estimated amplitudes of the partials found in the two signals (same signals as in Figures 4.1 and 4.2). It is apparent from the graphs that the partials whose estimated frequencies align well with the spectrogram, also have smoother decay curves with regards to amplitude. This is especially clear in Figure 4.4, where we see a nearly perfect linear decay for p_1 and p_2 from 0.5 s and onwards, which coincides with the range where the frequency estimation is good. In Figure 4.3 we don't see as clear of a linear decay, but both p_1 and p_2, although a bit noisy, show an overall linear decay behaviour in the window from 1 s to 2 s. A linear decay in dB is a promising indicator of a good overall partial estimation, since that is the expected behaviour from a stable, physical system such as a drum mode.

In Figure 4.3, the amplitude of p_2 oscillates at a regular rate, which might seem surprising at first. But this can be explained by the fact that the second partial in that audio sample, actually consists of two very closely spaced partials, as seen in Figure 4.5. An FFT of the stable segment of the two signals, shows that there are two partials around 100 Hz in the 70's Floor Tom signal; one at 95.5 Hz and one at 102 Hz. Instead of detecting them both, the algorithm estimates them as one partial with a frequency slightly below 102 Hz, and gives it an oscillating amplitude consistent with the resulting beat frequency. The rate of the amplitude modulation in Figure 4.3 is approximately 6 Hz, which corresponds to the difference in frequency between the two peaks making up the partial.

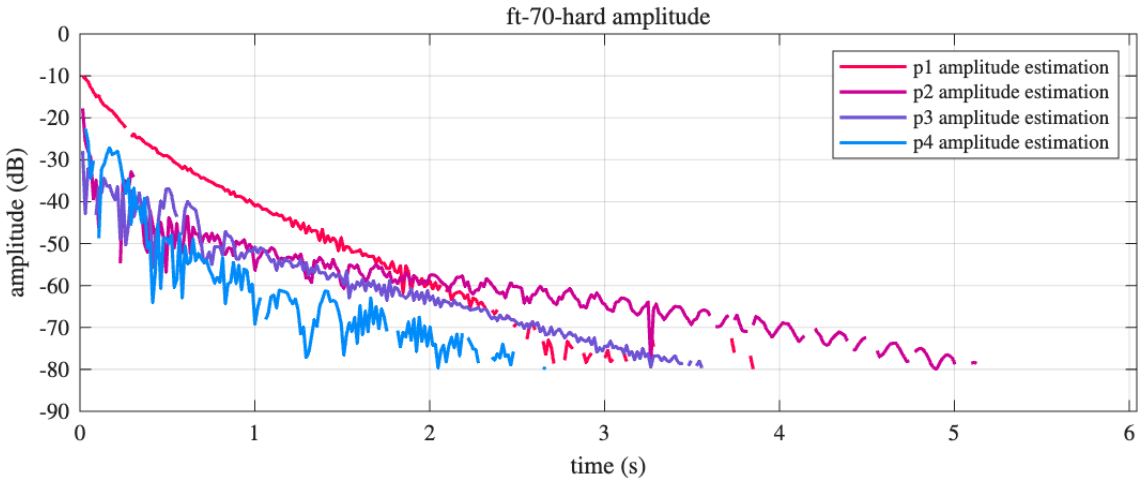


Figure 4.3: Estimated amplitudes of each partial, as a function of time.

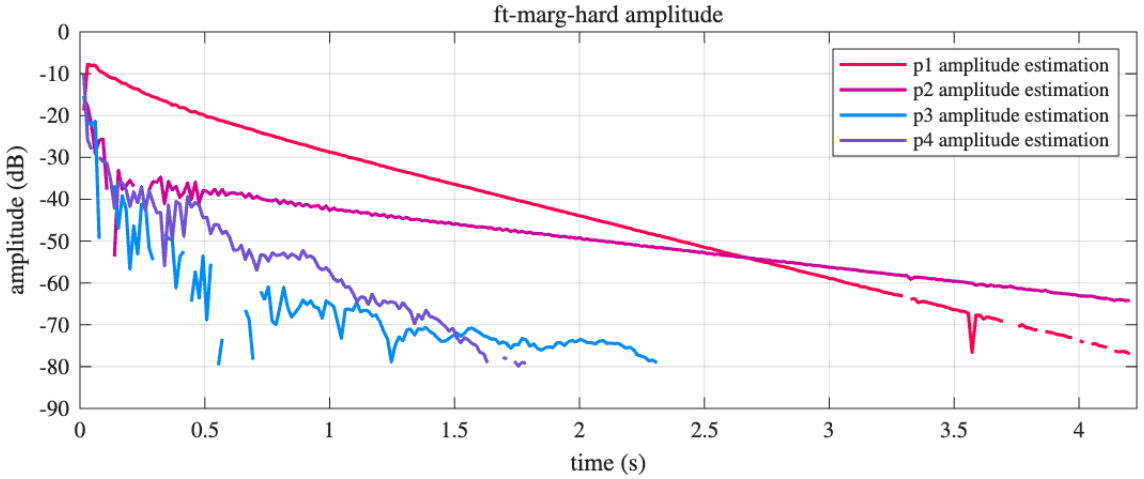


Figure 4.4: Estimated amplitudes of each partial, as a function of time.

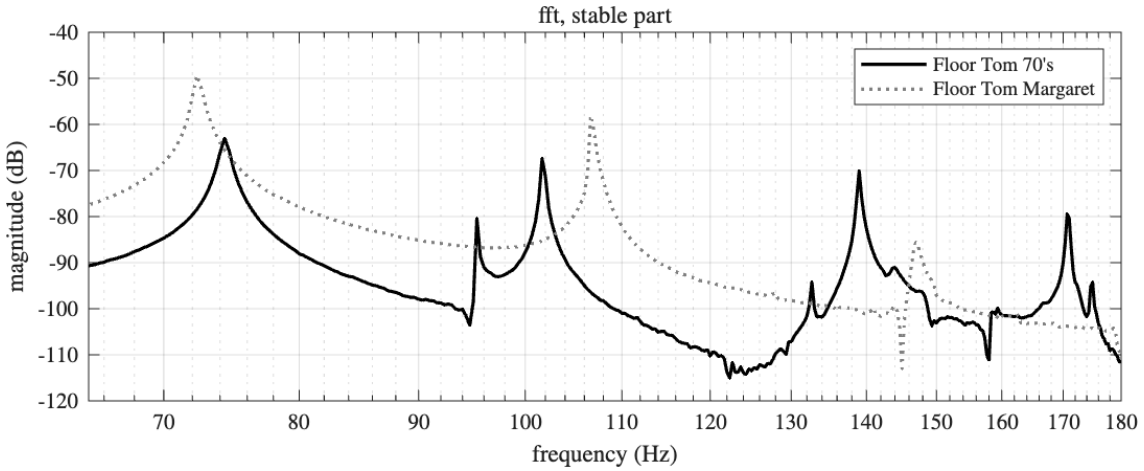
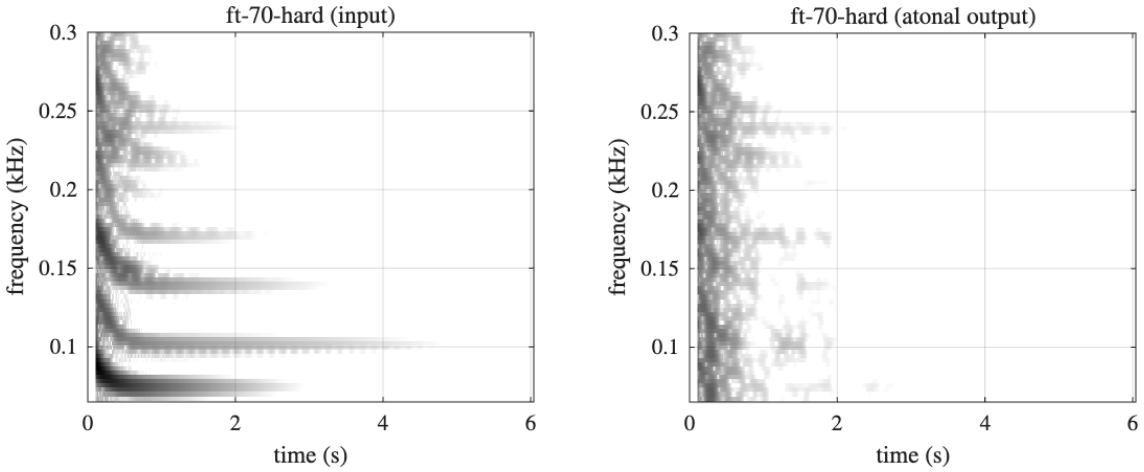


Figure 4.5: FFT of the stable part of the two signals; $1 < t < 4$ s. The 70's Floor Tom has two closely spaced peaks around 100 Hz: at 95.5 and 102 Hz, respectively.

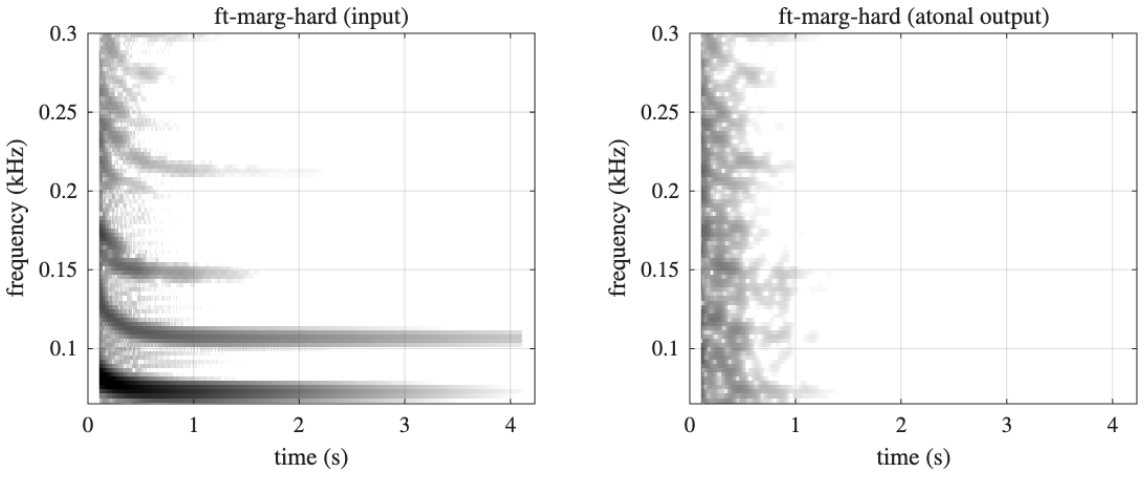
4.1.3 Separation accuracy

Figures 4.6 and 4.7 show the spectrograms of the input and the atonal output of the same two signals as previously. The targeted partials are almost entirely removed, which indicates that even though the frequency estimations are a bit noisy and imprecise, the time-domain subtraction seems to works fairly well. A reason for this might be that since the synthesis blocks are in the order of only one to four periods long (one for the fundamental, between three and four for the fourth partial), a moderate error in the frequency estimation can be compensated for by the phase estimation, allowing for a sufficiently well-matching estimation for such a short frame.



(a) Input signal. (b) Atonal output signal.

Figure 4.6: Spectrograms of the input signal and the atonal output signal, for audio sample Floor Tom 70's (hard).



(a) Input signal. (b) Atonal output signal.

Figure 4.7: Spectrograms of the input signal and the atonal output signal, for audio sample Floor Tom Margaret (hard).

4.2 Settings dependency

This section compares the results from using different settings on the same audio signal. The complete list of settings and variables used in the implementation was given in Chapter 3, and many of them are related to the validation and stability modules of the program. In general, the implementation turned out to be quite sensitive to changing the settings – this particularly applies to the validation parameters, and any settings deciding at which instance to, for example, turn on certain features such as the frequency sorting or the backlog check. When this was done at an unfavourable iteration, the consequences affected the whole estimation. This section instead examines what the effects are of changing more fundamental settings, namely the number of tracked partials, and the block size.

4.2.1 Number of tracked partials

The number of non-tracked partials present in the detection range affects the estimation accuracy of the partials that are tracked. Figure 4.8 shows a one-partial tracking applied on a range containing four partials (of which two are visible in the figure). As seen in the spectrogram, the lower partial (the fundamental, in this case) fades out faster than the higher partial, and this leads the fundamental estimation to "migrate" to the partial with the longer decay time. The migration begins as soon as the non-tracked partial's amplitude surpasses that of the tracked partial. This effect was expected, as the validation algorithm was designed to gradually adapt to changes even when it may have settled on what seems like a stable track. To avoid migration, as many partials as possible should be tracked in each signal. Simply limiting the allowed frequency range does not solve the problem, as higher-order partials are also periodical at multiples of their period, which might fall within the range. Instead, the user should make sure to track enough partials to cover all bases – tracking too many partials has not been found to lead to any negative consequences, other than cluttering the graphs. Figure 4.9 shows a reference tracking of four partials in the same signal (two of them are outside the visible range).

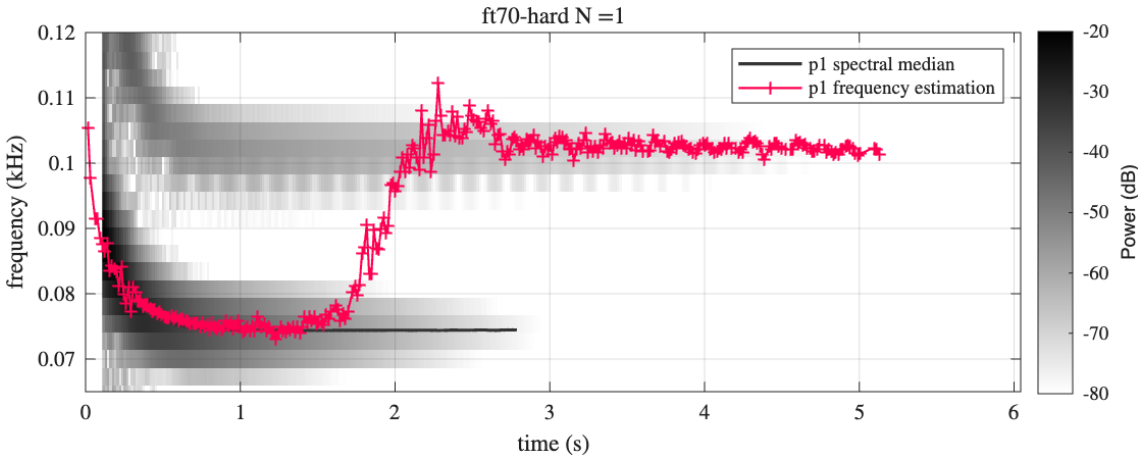


Figure 4.8: Tracking one partial, in Floor Tom 70's (hard). As p_2 surpasses p_1 in amplitude, the estimation of p_1 migrates to p_2, creating an illusion of a pitch bend.

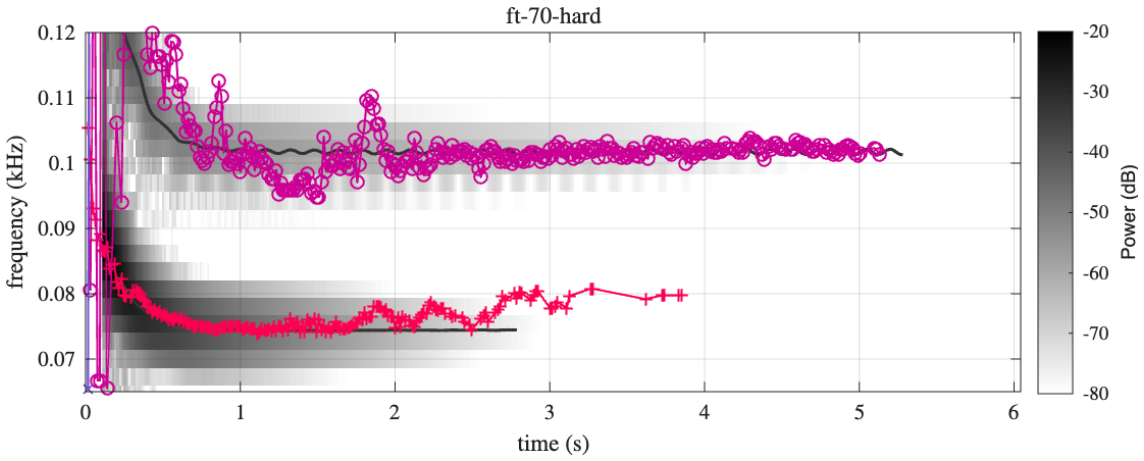


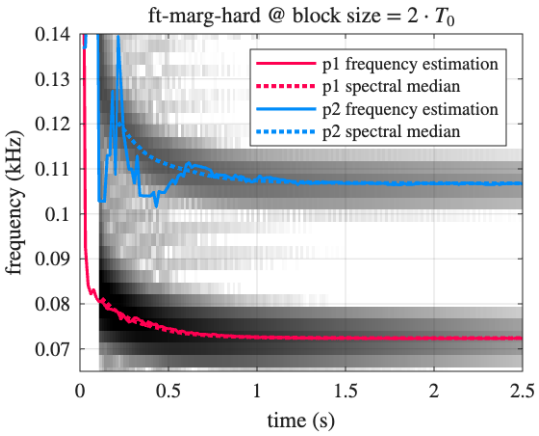
Figure 4.9: Tracking four partials, in Floor Tom 70's (hard). No migration occurs when the sufficient amount of partials are tracked.

4.2.2 Block size

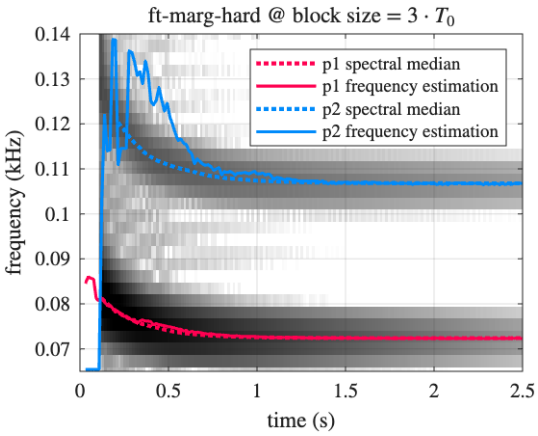
In Figure 4.10, five 2-partial estimations of the same signal are compared, computed using the same general settings with the exception of an increased block size. An immediate observation when increasing the block size, was that it created problems in the frequency estimation of higher-order partials, to the degree of invalidating the entire estimations due to consequential errors. The potential causes of this are discussed in Chapter 5. Meanwhile, the results presented in Figure 4.10 are of two-partial estimations, instead of four, limited to a suitable low-frequency range.

Looking at Figure 4.10, it should be noted that p_1 and p_2 reacts differently to the increased block size. In p_1, we see a delay in the response to the frequency drop as the block size increases, shifting the curve slightly above the spectral median. This was the expected result; since the algorithm bases its estimation on a longer backlog of signal, but imposes the estimation at the latest synthesis block (as opposed to, for example, in the middle of the analysis block), we observe a kind of backwards-averaging that makes the results lag behind in comparison to shorter block-estimations.

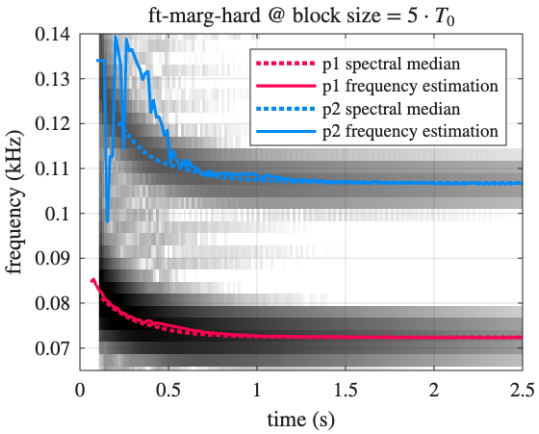
In p_2, however, the increased block size seems to result in random errors to the estimation, with the ultimate consequence that it takes longer to make a decent estimation whatsoever. With the exception of what seems to be a lucky strike in Figure 4.10c, the increased block size correlates to an increased time before p_2 finds its proper frequency, with errors alternating between positive and negative. Using a block size factor of 9 as opposed to 2, increases the time before a correct estimation is made by approximately 0.5 s. The corresponding block sizes in time, are 0.03 s for 2 times the fundamental's period, and 0.14 s for 9 times the fundamental's period. The "stabilisation delay" found in the p_2 estimation may be the same problem that occurred in the higher-order partials, but at a scale small enough to not ruin the results completely. Again, this requires a lengthier explanation, which is attempted in Chapter 5.



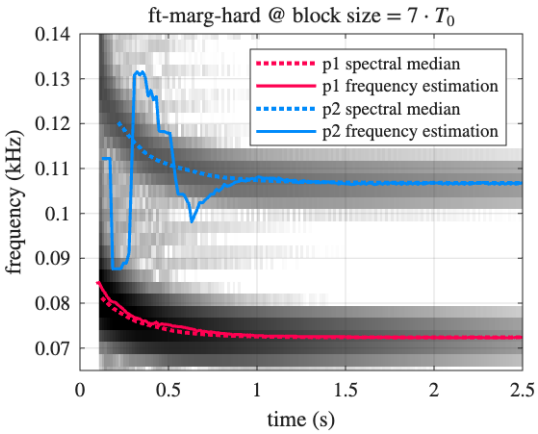
(a) Block size: 2 · 679. Overlap: 50%.



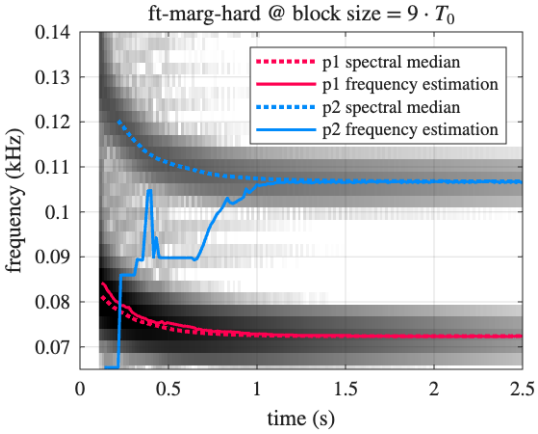
(b) Block size: 3 · 679. Overlap: 67%.



(c) Block size: 5 · 679. Overlap: 80%.



(d) Block size: 7 · 679. Overlap: 86%.



(e) Block size: 9 · 679. Overlap: 89%.

Figure 4.10: Frequency estimation results of the two lowest partials of Margaret Floor Tom, using increasing block sizes between 2 and 9 times the expected fundamental period of the signal, in samples.

4.3 Influence of signal properties

This section compares results on the basis of two input signal properties: damping, and what in musical terms is called "velocity": the forcefulness of the hit.

4.3.1 Damping

Considering that the algorithm so far has not been able to estimate higher-order partials before their frequency has stabilised, it is not surprising that tracking multiple partials in short or damped signals proved to be difficult. There are however promising results indicating that the fundamental can be reliably estimated from the first iteration, even in recordings of damped drums.

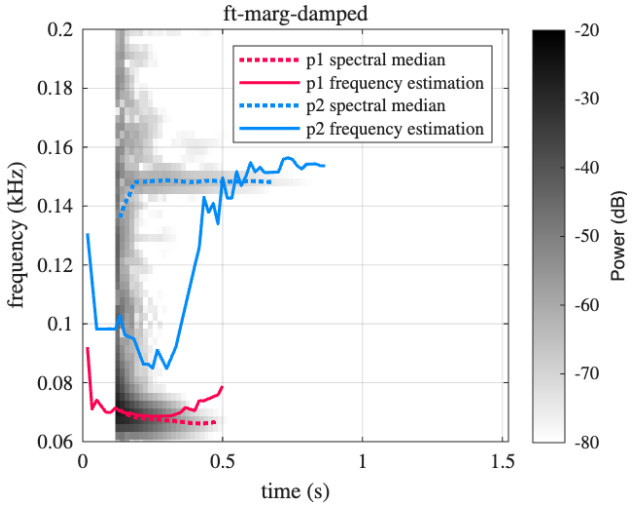
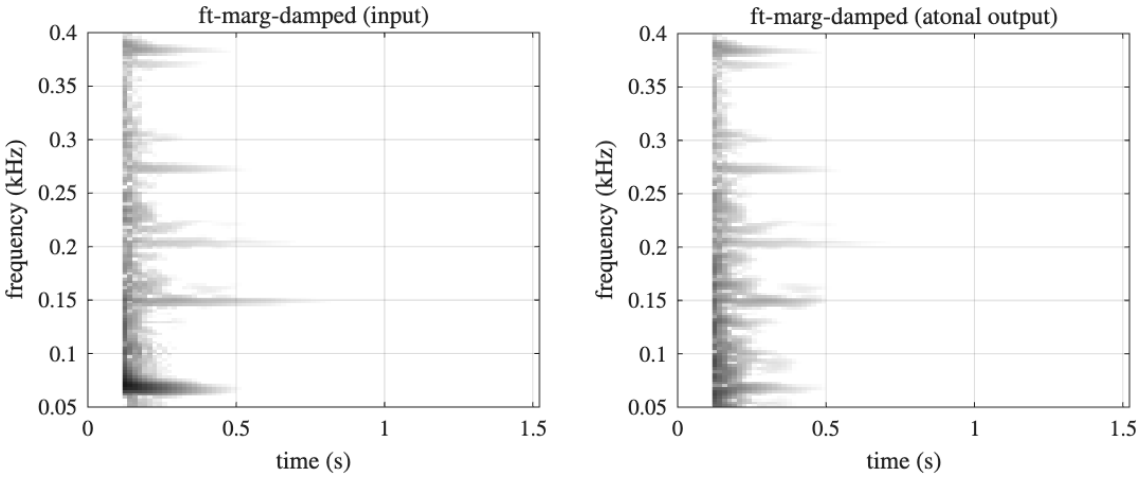


Figure 4.11: Frequency estimations of two partials in the Damped Margaret Floor Tom, detected in a range of [60, 200] Hz. Block size: 1470 samples, with 50 % overlap.



(a) Input signal.

(b) Atonal output signal.

Figure 4.12: Spectrograms of the input signal and the atonal output signal, for audio sample Damped Margaret Floor Tom.

In Figures 4.11 and 4.12, the frequency estimation and subtraction results are shown for a recording of the Margaret Floor Tom, where the drum was heavily damped. This signal is only 1.5 s long, which is less than half the length of its undamped counterpart. The previously dominant partial at 108 Hz is no longer in the signal, and the fundamental drops below -80 dB after around 0.5 s. Even so, the algorithm manages to identify the fundamental immediately, and by using a second partial as a buffer, it can also prevent migration as the fundamental quickly decays. Although not a typical use case (a damped drum is by default less tonal and thus less in need of tonal-atonal separation), this further proves that the algorithm is very successful at quickly identifying the fundamental frequency of short and transient sounds, even when physical damping has been applied.

4.3.2 Velocity

The algorithm's response to dynamics and velocity is examined through comparing different velocity samples of the same drum. Figure 4.13 shows the four-partial estimation of a medium velocity hit on the 70's Floor Tom, with the exact same settings as for the previously shown hard sample. It is clear from the graph that there are now only three prominent partials in the frequency range, which leads to p_2 getting "trapped" between p_1 and p_3, without having anything to estimate. As stated in Chapter 2, the frequency response of a drum hit can be varied greatly, only by hitting the drum in a different way. Looking at Figure 4.14, however, we see that the signal separation seems to be working well regardless. This is due to the fact that p_2, not having anything to estimate, estimates amplitudes so low that they only have a marginal effect on the output. The amplitude of p_2 is shown in Figure 4.15.

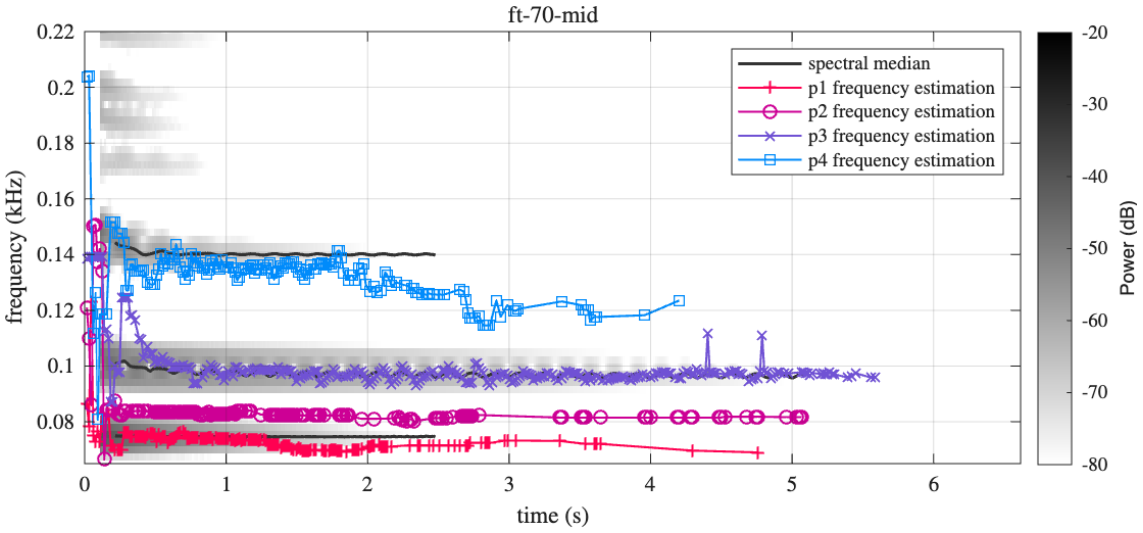
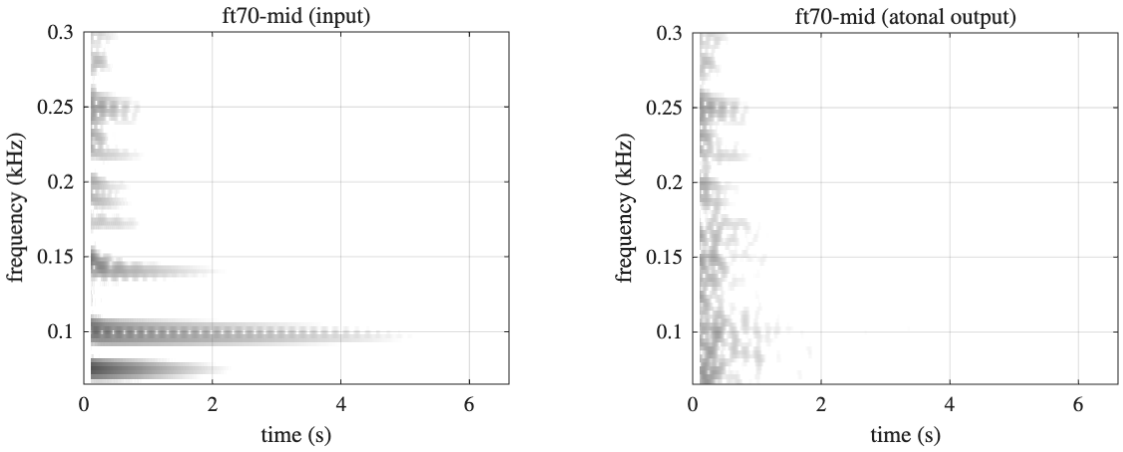


Figure 4.13: Frequency estimations of four partials in the 70's Floor Tom (mid velocity), detected in a range of [65, 220] Hz. Block size: 1358 samples, with 50 % overlap.



(a) Input signal.

(b) Atonal output signal.

Figure 4.14: Spectrograms of the input signal and the atonal output signal, for audio sample 70's Floor Tom (mid velocity).

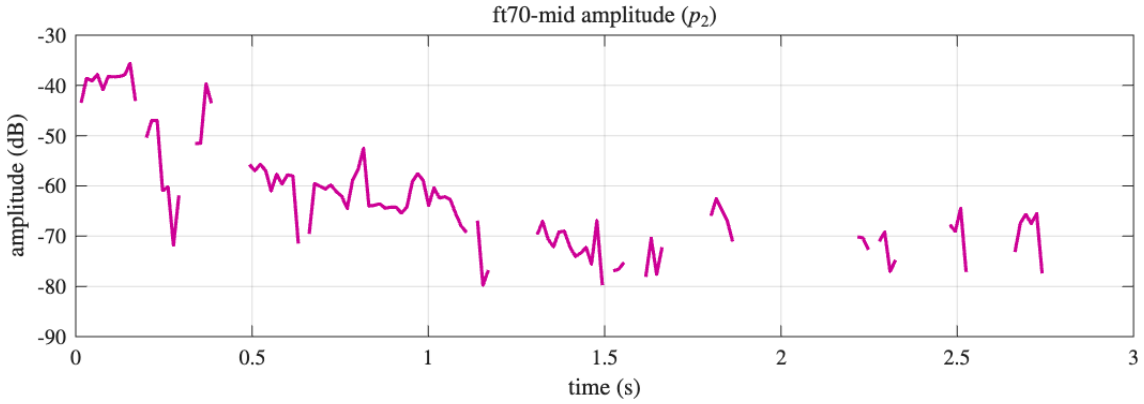


Figure 4.15: Amplitude estimation of p_2, the "trapped" partial.

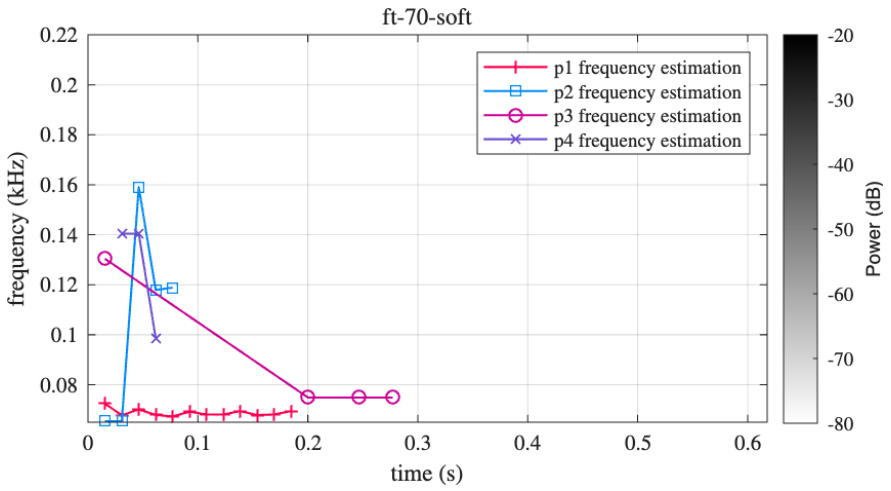


Figure 4.16: Frequency estimations of four partials in the 70's Floor Tom (soft velocity), detected in a range of [65, 220] Hz. Block size: 1358 samples, with 50 % overlap.

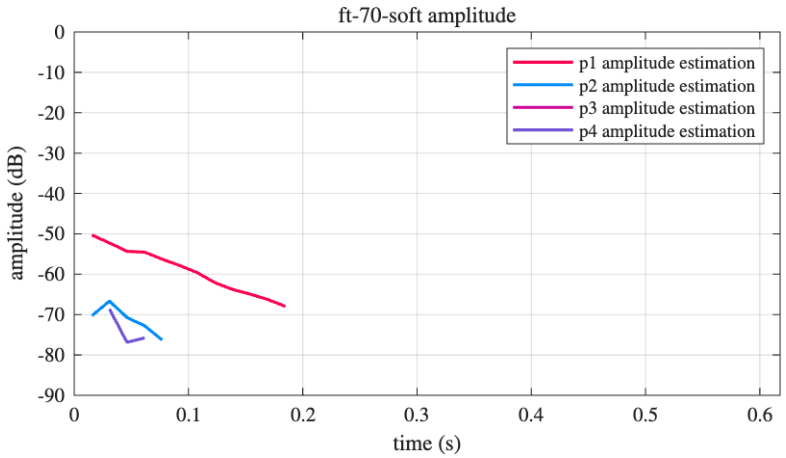


Figure 4.17: Amplitude estimations of four partials in the 70's Floor Tom (mid velocity), detected in a range of [65, 220] Hz. Block size: 1358 samples, with 50 % overlap. Partial p_3 was detected at amplitudes below -80 dBFS, and was thus discarded for the entirety of the signal.

An attempt was made to also analyse a low-velocity sample, but in addition to being very silent, the recordings provided of low-velocity hits were extremely short in comparison to its higher-velocity counterparts. Apart from an indication that the fundamental was still rather well-estimated, there are not many conclusions to draw from Figures 4.16 and 4.17.

4.4 Summary

The findings of the results chapter are summarised in the list below:

- The GPE and FPE are low under favourable operating conditions.
- The fundamental is the best estimated partial in all signals.
- Higher-order partials are not well-estimated until they stabilise in frequency.
- The partials with harmonic overtones present in the signal, are better estimated than those without it.
- Well-estimated partials tend to show modal decay behaviour in amplitude.
- Beating effects occur in the amplitude, when partials are too closely spaced for the algorithm to detect them individually.
- Not tracking enough partials in the signal can lead to "partial migration" and possibly other trajectory matching issues.
- No negative side effects were found of tracking too many partials in the signal.
- Increasing the block size and overlap increases the algorithm's response time to pitch changes, and increases the time it takes for higher-order partials to be correctly estimated.
- Damped signals have fewer partials, but the fundamental is still well-estimated.
- High-velocity signals are the longest and most resonant, leading to a higher degree of accuracy in estimations.
- Medium-velocity signals are also well-estimated, whereas low-velocity signals behave approximately like damped signals.

5
Discussion

This chapter discusses the results and findings of the study through three sections: 5.1 Performance evaluation and possible use cases, 5.2 Further developments and 5.3 Research questions.

5.1 Performance evaluation and possible use cases

The fact that a lag domain method which in theory should not be able to correctly estimate inharmonic partials at all, turned out to produce such accurate results for inharmonic signals, was highly unexpected. Albeit a very limited study, a GPE of 0.0 % in all estimated fundamental partials is quite remarkable, considering that the original publication of the YIN method upon which this model is based, presented an average GPE of 0.5 % in harmonic signals. There are many differences between the studies; most importantly, this study used extremely few recordings in its evaluation and was quite limited in where a ground truth could be established. The results however suggest that a modified version of the YIN algorithm can estimate the fundamental frequency of inharmonic signals quite well, at least under favourable circumstances.

With regards to the audio files that the algorithm produced, the informal judgement of the author is that even the estimations that did not look so good on paper, sounded quite realistic to the human ear when played back partial by partial. Poor frequency estimations were usually self-corrected by low amplitude estimations, creating fluctuations in amplitude rather than frequency. The initial uncertainty of the higher-order partials' frequencies created signals that varied so much between iterations that they sounded atonal rather than tonal during that initial segment, and this was perceived as less disturbing and unauthentic than the occasional amplitude fluctuation.

One of the many beauties of time domain subtraction is that the sum of the output signals is exactly equal to the input signal. In the implementation made here, this entails that should an estimation of one partial contain any errors, they are only heard in the output if additional signal processing is applied to that specific channel, revealing the underlying separation of that partial from the rest of the signal. Assuming that both the tonal and atonal signals are to be used in the output in some way, the demands on an audible error in the output are therefore quite high: to begin with, the error must occur in a partial that is a target for partial-specific

signal processing in the first place, and secondly, the processing applied to that signal must be of such a character that the misestimation is enhanced or revealed. If processing is applied in moderate amounts, it is likely that small errors in the estimations still remain concealed in the final mix.

Using the partial-wise audio output signals, pitch adjustments were made to individual or groups of partials, while leaving the atonal signal untouched. The results sounded plausible, especially in comparison to equivalent operations performed on the signal as a whole. So far, this re-pitching of partials has only been attempted for the best-estimated signals and partials, but an interesting follow-up study would be to evaluate how "bad" such operations sound when applied to estimations that were less successful. Other effects such as distortion, phaser or reverberation were also applied to individual or groups of partials, with interesting results. Partial-specific signal processing opens up a world of possibilities for creating drum sounds that are transparently tailored and tuned, or processed from the inside and out, creating a blend between the acoustic recording and digital effects.

Table 5.1: List of requirements set for the algorithm in Chapter 1, and their fulfilment with the IPT model.

Requirement	**Fulfilment**
Track one or multiple partials in an inharmonic source signal.	Yes, multiple
Split the incoming signal into one tonal part (consisting of the tracked partials) and one atonal or residual part.	Yes
Output the signals on a partial-to-partial level.	Yes
Produce results that when mixed together are identical to the input signal.	Yes
The buffer size should not exceed 1024 samples.	Yes/Depends
The algorithm should not need access to future samples, beyond the buffer	Yes, except one module.
The computation time needed for each signal block or buffer, should not exceed its corresponding duration in the signal.	Likely

5.1.1 Fulfilment of requirements

A formal evaluation of the model implementation is presented in Table 5.1, where the requirements and aims set in the introduction of this report, are all addressed. Most of the requirements were fulfilled, some of them with slight modifications, and one was made likely although not formally tested. The only module that needed access to future samples was the gradual interpolation step of the global synthesis. This is a very minor part of the total implementation, and alternative methods complying with the demands are suggested in Section 5.2.2. The buffer size requirement is fulfilled in the implementation as long as $f_s/f_{\min}$ is less than 1024, which corresponds to a lowest allowed $f_{\min}$ of 43.1 Hz. Since no actual real-time implementation was made, the computation time was not tested in real time. In the offline implementation, the computation time per time step was comfortably below half

the actual duration of the time step, which should indicate that the algorithm is real-time applicable in this regard as well.

5.1.2 The main issue: Block size vs. period

One of the main findings of this study is that it seems unachievable to get a good estimation of the higher-order partials until they have stabilised in frequency. Another finding is that increasing the block size seems to amplify this effect. This section makes an attempt at explaining why that is, based on the mechanisms of the YIN algorithm.

A block-wise YIN implementation assumes that the partial of interest is constant in both frequency and amplitude, within one comparison block. If the frequency of a partial changes during the course of the block, the optimal lag for correlation changes with it. The beginning and the end of the block will thus counteract each other in the lag domain, and produce wider, less distinct, minima in the lag curve. The severity of this effect depends on the rate of change of the frequency, but also on the relative length of the block, compared to the frequency of the signal. Figure 5.1 shows two copies of a frequency sweep from 100 to 70 Hz, one delayed by what should be the optimal lag: the period of the signal at its beginning. If the comparison window only would include the first one or two periods, this lag would indeed yield a very good match, and a low dip in the lag curve. The difference between the signals however increases towards the end of the block, resulting in a situation where the correlation decreases if more signal is analysed.

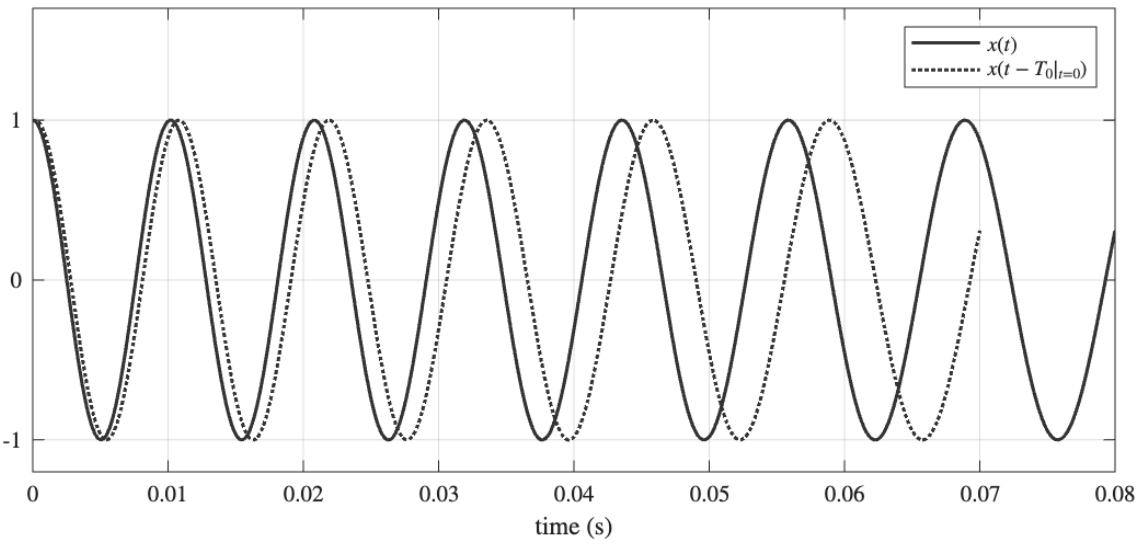


Figure 5.1: (To be replaced) – a comparison of two copies of the same signal, one delayed by its initial period T_0.

One solution to this problem could be to continuously adapt the length of the analysis frame to the expected period of the signal, in order to always use a block size corresponding to just above two periods of the current partial's frequency. The implementation could be simple: just cut the current input signal block to 2.5 times

the last iteration's period of that partial, and adjust $\tau_{\max}$ accordingly. The first issue with this is that it could potentially prohibit fast pitch drops in the partial, as a longer signal would be needed in order to detect them. Another issue is that if a good estimation is not already in place, how would the algorithm know how short to cut the block? If one already knows the approximate pitch curve profile of a signal it is of course possible to automatically adjust the individual block sizes of partials based on that, but this means that the whole signal must be pre-processed before the real analysis can take place.

5.2 Further developments

The implementation of the YIN-based IPT algorithm designed in this study is still a prototype under construction, and should more thorough testing and evaluation be performed, it is almost certain that new types of issues and bugs will arise. This section summarises a few of the most obvious and/or pressing improvements to the algorithm, that are yet to be made.

5.2.1 Validation and stabilisation

One of the main concerns during the development and parameter tuning of the algorithm, was that it was extremely sensitive to changes applied in the frequency validation module. This is a consequence of implementing several pass-or-fail conditions, effectively de-linearising the response of the algorithm. The validation process as a whole could probably be improved quite a bit, for example by implementing statistical tools rather than hard limits, both in choosing the final lag candidate and in validating it against prior estimations. The crux here is to find a good solution that maintains real-time compatibility.

5.2.2 Adjustments to the synthesis stage

The synthesis module broke the restrictions on causality set in the aims of this study, by interpolating the end of each block in order to smooth the edge towards the upcoming one. This must be addressed and changed, if the algorithm is to be implemented in a real-time setting. A possible solution could be to interpolate the estimated parameters (frequency, phase and amplitude), rather than the synthesised signal, between the blocks. Another option would be to simply shift the location of the interpolation area, so that all interpolation occurs in the beginning of the blocks.

5.2.3 Retriggering

The implementation did not include any tactics for dealing with retriggering; i.e. deciding what should happen when the drum is struck again and a new transient appears. A solution that comes to mind is to detect transients via sharp amplitude changes, and reset the algorithm every time a transient occurs.

5.2.4 Preprocessing

An important question to consider when developing something that is intended to become a tool, is of course what it is intended to be used for, and what assumptions that follow from the situation in question. In this study, the aim was to design a partial separator that could operate on drum recordings without any prior knowledge of the input signal. Realistically, if the input is indeed the same drum for the whole duration of the recording, we actually do know some things about that signal from only hearing it once. For example, we know that the physical dimensions of the drum will stay fixed, even though the drummer may vary their playing technique. Further reading would be needed to confirm this, but the recordings used in this study had the common property that the relative distance between partials was constant and individual to each drum, even as the pitch changed in its initial half-second or so. The starting point of the pitch curve varied with velocity, but it did so with the same factor for all partials. This means that if the algorithm was provided with an offline "test signal" in order to characterise the source, it could use the deducted properties of the source in upcoming estimations, allowing for what should be considerably improved results. Such properties could be the number and relative frequencies of partials, and the target pitch of all partials. In practice this means that if a sound engineer records a few hits of the drum during soundcheck, the recording could be analysed using offline methods in order to provide better assumptions for the real-time model. The same method could be employed in a mixing situation; the producer can extract one or two hits from the recording for offline analysis, in order to tune the model for real-time use.

5.3 Research questions

To finalise the discussion on the results, the initial research questions are now addressed individually, based on the findings of this study.

Which methods are available for real-time tracking of partials, and which are most suitable for use in a DAW?

The study found that depending on the buffer size and latency allowed, both spectral and lag domain methods can be used to track partials. There are offline spectral methods designed specifically for inharmonic partial tracking, and aside from requiring longer buffers than lag domain methods, this study found no theoretical limitations in terms of applying them in a DAW compatible manner. However, even if overlap step-synthesis strategies are employed to minimise latency, long analysis blocks result in a slower response time of the algorithm which could affect the accuracy in the fast initial pitch drop of a drum signal. The most promising article on offline inharmonic spectral partial tracking stated that their method had issues establishing trajectories on transient signals [8], and that they improved the results by processing the signal backwards. While a valid tactic for offline processing, such tricks cannot be applied in real-time. Lag methods, such as the YIN algorithm, allow for processing frames as short as two periods of the fundamental. The synthesis

window length does not have to correspond to the analysis window's, which means that it can be even shorter. This makes it easy to perform analysis at high overlaps, without having to overlap the output. The downside of using f_0-based methods for detecting inharmonic partials is that they require the partials to be very prominent in the signal, and the results indicate that only one partial (the fundamental) can be reliably tracked before the pitches have stabilised in frequency. Between lag domain methods and spectral methods, lag domain seems to be the better option in terms of speed and real-time applicability, but spectral methods are likely to be more accurate in an offline setting.

Which methods are available for real-time suppression, synthesis or separation of partials, and which are most suitable for use in a DAW?

The study found that time domain subtraction is an effective tool for separating synthesised partials from a tonal signal. Frequency domain subtraction is mathematically equivalent to time domain subtraction, and can be used in its place in methods where it is more convenient to do so. The accuracy of the subtraction itself is perfect as long as the computation has a rounding error smaller than the bit depth of the signal; what affects the results of the separation is the accuracy of the estimation of the subtracted signal.

With the methods available, it it feasible to construct an algorithm that tracks and separates multiple partials simultaneously in a DAW?

Yes. This study managed to design a real-time partial tracker and separator, using an IPT framework and a modified version of the YIN algorithm. It does not work perfectly for all signals, but with clear input data and fine-tuned settings, it produces very low GPE:s, and with some development its accuracy and stability could be improved even further.

How many inharmonic partials can be tracked reliably in one signal?

In this study's implementation: at least four (but of course, it depends on the signal). The study found that searching for more partials than expected in the signal did not influence the results negatively, but searching for too few led to issues in the frequency detection. All "prominent" partials in the input signal should be tracked, and the frequency range should be set to include them, even if the number and range of partials desired for processing is more limited. Partials that are closely spaced may be identified as one partial with amplitude modulation at the beating frequency. The total number of detected partials depends on their relative amplitudes and spectral distribution in the input signal.

How stable and/or parameter sensitive are the methods?

The YIN-based IPT implementation developed in this study is rather parameter sensitive, in the sense that its stability and validation settings have a great, non-linear, influence on the results. The user-set frequency search range, backlog size, frequency sorting start index, and tolerance factors against harmonic errors and

partial conflicts, all required fine tuning at some point during the study. This is listed as one of the most important areas of future work.

Which properties of the input sound signal are most influential on the results?

Velocity and damping both affect the accuracy and performance of the algorithm, mainly by changing the level, length and tonality of the signal. Long, loud and tonal signals are easier to estimate than short, damped signals with little tonal content. In terms of the imaginable range of input data from only drum recordings, this study is extremely small; a more thorough analysis of a larger data set is required in order to draw further conclusions.

6
Conclusion

This study compared three possible methods for partial estimation in transient inharmonic drum recordings, in order to implement one in a partial estimation-driven tonal-atonal separator. The choice fell on the lag domain method called the YIN algorithm [1], implemented according to an algorithm proposed in this study, called the Iterative Partial Tracking method. The implementation was evaluated for accuracy, settings dependency and input signal dependency, and the results showed that for select ideal samples, it was possible to tune the settings of the algorithm in order to produce frequency estimations with extremely low GPE:s and FPE:s (gross and fine pitch errors, respectively). The algorithm was otherwise quite parameter-sensitive, and further research should focus on improving the validation step of the frequency estimation module, as well as address the need for a retriggering solution, real-time compatible signal smoothing, and the task of identifying higher-order partials faster after each transient.

Bibliography

[1] de Cheveigné A, Kawahara H. YIN, a fundamental frequency estimator for speech and music. Journal of the Acoustical Society of America. 2002 04;111(4). Available from: https://doi.org/10.1121/1.1458024.

[2] Hess W. Pitch Determination of Speech Signals. Springer-Verlag; 1983.

[3] Bonada J, Serra X, Amatriain X, Loscos A. Chapter 10: Spectral Processing. In: Zölzer U, editor. Digital Audio Effects. 2nd ed. John Wiley & Sons, Ltd.; 2011. .

[4] Antares. AutoTune;. Available from: https://www.antarestech.com/.

[5] Verma TS, Meng THY. Extending Spectral Modeling Synthesis with Transient Modeling Synthesis. Computer Music Journal. 2000;24(2). Available from: https://www.jstor.org/stable/3681927.

[6] Fletcher NH, Rossing TD. The Physics of Musical Instruments. Springer-Verlag; 1991.

[7] Boersma P, Weenink D. Praat: doing phonetics by computer;. Available from: https://praat.org/.

[8] PARSHL: A Program for the Analysis/Synthesis of Inharmonic Sounds Based on a Sinusoidal Representation; 1987. Available from: https://ccrma.stanford.edu/files/papers/stanm43.pdf.

[9] Evangelista G, Marchand S, Plumbley MD, Vincent E. Chapter 14: Sound source separation. In: Zölzer U, editor. Digital Audio Effects. 2nd ed. John Wiley & Sons, Ltd.; 2011. .

[10] Arfib D, Keiler F, Zölzer U, Verfaille V, Bonada J. Chapter 7: Time-frequency processing. In: Zölzer U, editor. Digital Audio Effects. 2nd ed. John Wiley & Sons, Ltd.; 2011. .

Appendix

Additional findings: Phase operations

Although not likely to be a candidate for serious signal separation, the study also came across a method for removing residual tonal components from residual signals. In summary, this method can be described as scrambling of the phase, and is called "whisperisation".

It is generally assumed that a periodic signal is continuous at its period borders and has a constant phase offset. Humans do not perceive phase offset in itself, but if the phase offset is not constant from period to period, this may affect the signal to the degree where we don't perceive it as periodic anymore. Random changes in phase offset between periods will in fact render the signal to be non-periodic, as period-to-period similarities are destroyed by the resulting delay. In addition to that, the resulting discontinuities at period borders will contribute to added noise levels in the signal. One way of reducing unwanted periodicity in a signal is thus to randomise its phase, while leaving the frequency spectrum intact. This operation takes place in the spectral domain, on a frame-to-frame basis. For each frame, the following steps are followed:

1. Perform FFT on the signal frame
2. Separate the magnitude and phase spectra by computing the magnitude resp. phase of each FFT coefficient.
3. Replace the value of each bin in the phase spectrum with a random value between 0 and 2π.
4. Reconstruct the complex Fourier coefficients, using the initial magnitudes and the new randomised phase values.
5. Perform IFFT on the results.

Depending on the frame length, this operation will yield different results. When applied to long frames (longer than the period of the sound), the result is an uncertainty of frequency, rather than its full removal. Windows in the same range or shorter than the period of the signal will however "destroy" the periodicity as described above, and replace tonal components with formant-preserved noise, perceptually interpreted as having a hoarse or whispering quality. [10]

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY