THE EXTENDED ADAPTIVE QUASI-HARMONIC MODEL IN PYTHON

COMPUTER SCIENCE DEPARTMENT - UNIVERSITY OF CRETE

PANAGIOTIS ANTIVASIS

ADVISOR: YANNIS STYLIANOU

SUPERVISOR: GEORGE P. KAFENTZIS

ACKNOWLEDGEMENTS

I would like to thank the members of my thesis committee: My supervisor, Dr. George P. Kafentzis, my advisor professor Yannis Stylianou and professor Panagiotis Tsakalidis for their assistance and patience, as well as for providing me with the opportunity to broaden my knowledge in the fields of speech processing and sinusoidal modelling. I would also like to thank my family for being supportive with me all those years of me being a student and for always being understanding. Special thanks also to my dear friend Kledi Salla for offering excellent writing advice. Finally, I would like to thank everyone participating in the Subjective Evaluation part of this thesis and provided me with useful information for this work's content.

ABSTRACT

Speech Processing (SP) has developed dramatically in recent years, being employed in a wide range of applications and adopting a variety of schemes. The evolution of SP lead to many attempts to achieve higher quality representation of voice files with the help of the **Sinusoidal Models (SMs)**. This thesis is an implementation of the **extended adaptive Quasi-Harmonic Model (eaQHM)**, a previously researched SM that incorporates speech reconstruction utilizing time-varying exponential functions and exploiting amplitude adaptation, followed by frequency refining. Studies have demonstrated that the eaQHM gives superior flexibility and efficiency in resynthesizing speech than the other SMs. This model had already been implemented in **MATLAB**, however the need for a more accessible and comprehensible approach was critical. The aim of this thesis is to implement the eaQHM model in **Python** and then assess if the code gives satisfactory results.

***Index Terms*—** SP, Speech Reconstruction, SMs, eaQHM, Speech Analysis, Speech Synthesis, Pitch estimation

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

SP Speech Processing

ADC Analog-to-Digital Converter

NTT Nippon Telegraph and Telephone

LPC Linear Predictive Coding

VoIP Voice-over-IP

IVA Intelligent Virtual Assistant

DARPA Defense Advanced Research Projects Agency

CALO Cognitive Assistant that Learns and Organizes

ERP Event Related Brain Potential

EEG Electroencephalography

ASR Automatic Speaker Recognition

MFCC Mel-Frequency Cepstral Coefficient

GMM Gaussian Mixture Model

SVM Support Vector Machine

NN Neural Network

CNN Convolutional Neural Network

RNN Recurrent Neural Network

RF Radio Frequency

DTW Dynamic Time Warping

HMM Hidden Markov Model

LMT Logistic Model Tree

DT Decision Tree

KNN K-Nearest Neighbour

SM Sinusoidal Model

aSM adaptive Sinusoidal Model

aQHM adaptive Quasi-Harmonic Model

eaQHM extended adaptive Quasi-Harmonic Model

SRER Signal-to-Reconstruction-Error Ratio

AI Artificial Intelligence

FFT Fast Fourier Transform

STD Standard Deviation

1 INTRODUCTION

1.1 SPEECH PROCESSING

Indubitably, speech is a critical element of our daily life and it is one of the fundamental human impulses and subsystem voices [1]. It is critical to recognize that just in an average presentation, 100 to 150 words per minute are spoken while in a casual conversation, this increases to 120 to 150 words per minute [2]. Everything from a walk to the grocery store to a TED-ex talk requires us to utter words and phrases that will help us understand each other and be heard. And hence, with the development and prosperity of signal processing, the need to digitally store, process, and transmit speech began to advance. For that purpose, digital processing on voice signals, or in other words **Speech Processing (SP)**, came into play [3].

Having a long history, SP has achieved widespread popularity in recent years, especially amongst computer scientists, who were compelled to devise techniques to process voice signals. SP algorithms have been used in a variety of appliances and fields, involving procedures such as filtering, amplifying [4], decimation, interpolation [5] and reconstruction. This thesis is primarily concerned with **Sinusoidal Models (SMs)**, a SP technique that applies reconstruction to provide a greater approximation of a speech signal [3, 6].

1.2 A BRIEF HISTORICAL BACKGROUND

Initial efforts involved speech recognition and processing, only concerned with recognizing a few fundamental phonetic elements including vowels. In 1952, a technique was created by Stephen Balashek, R. Biddulph, and K. H. Davis, three Bell Labs researchers, which was used to identify digits uttered by a single individual [7]. Fourteen years later, Fumitada Itakura of Nagoya University and Shuzo Saito of Nippon Telegraph and Telephone (NTT) were the first to propose Linear Predictive Coding (LPC), a SP method which was later advanced by Bishnu S. Atal and Manfred R. Schroeder of Bell Labs during the 1970s [8]. LPC has layed the foundations for Voice-over-IP (VoIP) technology and speech synthesizer chips, like the Texas Instruments LPC Speech Chips used in the 1978 Speak & Spell toys [9]. Then, in the early 1990s, speech recognition systems emerged, with Dragon Dictate being the most commercially accessible and the AT&T employed technology created by Lawrence Rabiner and colleagues at

Bell Labs in their Voice Recognition Call Processing service to redirect calls without the use of a human operator [10].

1.3 APPLICATIONS OF SPEECH PROCESSING

SP finds itself useful in a lot of areas. This section will describe how important the appliance of SP is in the fields of speech recognition by **Intelligent Virtual Assistants (IVAs)** [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29], medicine [1, 30, 31, 32, 33, 34] and telecommunications [35, 36, 37, 38].

First and foremost, **IVAs** execute activities or provide services based on voice orders or inquiries which can range from information about weather forecast to playing music or videos, supplement and/or replacement of customer support by human beings, to-do list creations, podcast streaming and information about the news [11, 12, 13, 14]. One very popular IVA is Apple's **Siri**. Created by Dag Kittlaus, Tom Gruber, and UCLA alumnus Adam Cheyer, Siri was initially used for military purposes by Defense Advanced Research Projects Agency (DARPA) for the project named "Cognitive Assistant that Learns and Organizes (CALO)", and was aftewards bought by Apple Inc. with the initial idea taken from a concept video called the Knowledge Navigator, which debuted in 1987 [23, 24, 25, 26]. Amazon **Alexa** is an equally famous IVA which evolved from a precursor with Polish origin known as Ivona, to be later purchased by Amazon and be combined with Amazon Echo one year after [15, 16, 17]. And finally, **Google Assistant**, which is powered by Artificial Intelligence (AI), was officially published in July 2018, and to this day, it is enjoyed by 1 billion devices in total and over 500 million people a month [27, 28, 29].

Medicine also utilizes SP. Specifically in the field of Electroencephalography (EEG), the **Event Related Brain Potentials (ERPs)** were developed in light of the necessity of adaptation to speaker identity and speech error identification in SP. As a result of research, two ERP components N400 and P600 were invented with the purpose of speech semantic and syntactic information processing [30]. It is also astonishing how many distinct applications and techniques exist in the field of vocal pathology detection. First, **Machine Learning** systems employ numerous different algorithms to identify any vocal anomalies in a voice sample and distinguish the healthy from the unhealthy sounds [31, 32]. Another technique is **Automatic Speaker Recognition (ASR)**, which finds speech disorders by integrating patterns from patients who have the same diagnosis. It accomplishes this by first extracting the Mel-Frequency Cepstral Coefficient (MFCC) parameters, i.e. the healthy sections of a speech recording, and then determining whether or not the individual is healthy using the statistical pattern recognition classifiers Gaussian Mixture Model (GMM) and

Support Vector Machine (SVM) [33]. Another approach suggests the usage of **Neural Networks (NNs)** after obtaining the features of the voice, and more specifically **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)** [1, 39]. And lastly, [34] proposes a non-invasive and objective method by examining three unique classifiers within the contexts of supervised learning and disorder detection.

One must not exclude SP in telecommunications. Nobody can argue that speech is nothing more than a regular analog signal that with the usage of an Analog-to-Digital Converter (ADC), someone can obtain a digital form of it by sampling and quantization techniques [35, 36]. Then right after that, the speech signal is ready to be stored and as a result transmitted to the receiver. Telecommunications is a field which has emerged during 1897 and since then, people all around the world have joyfully embraced new wireless communications technologies and services. Mainly utilizing digital signals, telecommunications is continually changing and incorporating more and more applications. In the mobile radio communications aspect, improvements in digital and Radio Frequency (RF) circuit manufacturing, new large-scale circuit integration and various miniaturization technologies that reduce the size, cost, and reliability of portable radio equipment, have emerged [37]. Other utilizations of digital signal processing in telecommunications consist of digital transmission and switching, transmission terminals with pulse-code modulation, transmission terminals with frequency-division multiplex, signaling tones detection, echo control, design considerations for hardware elements and programmable digital signal processor operations [38].

1.4 TECHNIQUES

Without a doubt, SP is an incredibly useful component in many domains. But how does it work? How can someone process a speech audio file? To this day, professionals have never shied away from utilizing SP schemes and, as a result developing new ones, beginning with **Hidden Markov Models (HMMs)** [18, 19, 20, 21, 40, 41] and mainly focusing on **NNs** [1, 42, 43] as well as **deep learning** [39] with the start of the new millennia. This section analyzes some of those schemes and the areas they are involved.

Firstly, **Dynamic Time Warping (DTW)** is a well known approach for determining the best alignment between two supplied (time dependent) sequences under specified constraints and are warped in a nonlinear way to match each other. DTW first employed in automatic speech recognition to find differences between various speech patterns in different speeds [44, 45]. Furthermore, a very popular technique is **HMMs**. Beginning in the mid-1970s, one of the initial uses of HMMs was voice recognition prior to beginning to analyze biological sequences, namely DNA, and later entering the area of bioinformatics [18, 19, 20, 21, 40, 41]. A more modern scheme are **NNs** in different variations. Like Section 1.3 suggests, CNNs, which are based on levels of hierarchy that consist of routing and grouping layers and RNNs, with the intent of simulating temporal sequences and their long-term connections, play a major role in fields like speech pathology detection [1, 39]. NNs are also utilized in the fields of speech synthesis in a variety of styles and languages [42] and in an automated word recognition system that helps orally adept but illiterate persons become literate in the shortest amount of time feasible [43]. Additionally, as mentioned in Section 1.3, the following different **Machine Learning** classifiers are used for various speech applications, such as speech disorders, speaker identification, emotion recognition from speech, and others: **(1) SVMs**: Mostly used to divide data from distinct groups in a linear way. SVMs work by finding the ideal boundary called hyperplane, which needs to be as divergent to both classes as feasible and then divides them while increasing the gap to neighboring neatly separated instances [31, 32]. **(2) K-Nearest Neighbour (KNN)**: To classify data, KNN algorithm compares new data in relation to their distance. It then sorts the data and selects the first k of them to make a decision [31]. **(3) Decision Tree (DT)**: A tree data structure that distributes all data in accordance with specific rules or questions. The latter are represented as internal nodes, with data held as child nodes based on particular answers to those questions [31, 32]. **(4) Logistic Model Tree (LMT)**: Similar to DTs but linear regression model is present in the leaves, producing a piecewise linear regression model [31, 46]. **(5) Naive Bayes**: A classification which gathers all features that utilize the current procedure and focuses its conditional probability on the features in one class [32]. **(6) Ensemble Methods**: Multiple classifiers are used to improve accuracy by combining their predictions. They may also aid in the enhancement of the precision of other classifiers [32]. And lastly, **GMMs**, a useful and necessary technique for estimating probability distribution functions serve as the foundation for many applications of SP, some of them being vocal mimicking mechanisms and voice recognition systems [47, 48, 49, 50].

1.5 SINUSOIDAL MODELLING

Section 1.3 provides a more broad overview of the applications involving SP. However, the primary focus of this thesis is on Signal Modelling, used in many applications involving speech in the past twenty years, some of them being analysis, synthesis, enhancement and modifications [3, 6, 51, 52, 53].

Many experts in signal processing have tried to provide high quality and flexible representations of a signal and speech resynthesizing, utilizing various models. Those models are called **SMs** and they all vary in implementation and quality, but all share the same principle of splitting the signal in frames and representing it as a sum of sinusoids. The most fundamental approach, being the **adaptive Sinusoidal Model (aSM)** exploits the model's local adaptivity on the examined signal. **SMs** have inspired many people in the field to improve models that can capture speech more accurately, while keeping flexibility and naturalness intact, only to be dominated by this main concept: Decomposition of the signal into a Deterministic part, where harmonically related sinusoids are used to describe the quasi-periodic phenomena of speech, and a Stochastic part, which is actually the subtraction of the Deterministic part from the original time-domain speech signal and employs modulated Gaussian noise to represent its non-periodic features, such as friction noise [6, 54]. This concept paved the way for a more complex approach, called **adaptive Quasi-Harmonic Model (aQHM)**, which models the signal with time-varying exponential basis functions and in the end a frequency correction mechanism takes place. This of course provides a higher quality signal which is also quasi-harmonic. Then finally, the **extended adaptive Quasi-Harmonic Model (eaQHM)** expands the previous model by adding amplitude adaptation, thus achieving improved reconstruction of the signal [3, 6].

2 IMPLEMENTATION OF THE EAQHM

2.1 THE EAQHM MODEL

eaQHM has many advantages compared to other **aSMs**. First of all, it has been developed as a full-band model and thus there is no need for a maximum voiced frequency in the analyzed speech. Secondly, as it was previously stated, amplitude adaptation provides improved and more accurate reconstruction of the signal. eaQHM works by initially assuming an harmonic model and then iteratively reconstructing it by applying an f_0 estimation and frequency correction, until the reconstructed speech signal converges in **quasi-harmonicity**.

It all starts by describing a full-band signal as an **AM-FM decomposition**,

$$s(t) = \sum_{k=-K}^{K} A_k(t) e^{j p_k(t)} \tag{1}$$

where $A_k(t)$ being the instantaneous amplitude and $p_k(t)$ the instantaneous phase of the k_{th} component, given by

$$p_k(t) = p_k(t_i) + \int_{t_i}^{t} \frac{2\pi}{f_s} \Big(f_k(u) + c(u) \Big) \, du \tag{2}$$

with $c(u)$ being the phase coherence term [3, 6].

2.1.1 0_{th} *Adaptation*

With the help of an estimated f_0 for each frame, a full-band harmonicity is assumed so that the instantaneous amplitudes and slopes of the frame are extracted using Least Squares. For this reason, a Blackman analysis window $w(t)$ centered in the current time instant (t_i) is multiplied with the analysis frame

$$s(t) = w(t) \sum_{k=-L}^{L} a_k e^{j 2\pi k f_0 t} \tag{3}$$

where a_k being the complex amplitude of the k_{th} harmonic and L is the number of harmonics spanning the whole spectrum up to the Nyquist frequency. Because an initially harmonic model is assumed, no f_0 refinement is required. Finally, after a simple amplitude estimation for each component is applied, the interpolated values of $|a_k|$ and kf_0 are used to reconstruct the signal $s_{rec}(t)$ as

$$s_{rec}(t) = \sum_{k=-L}^{L} |a_k(t)| e^{j p_{k_{int}}(t)} \tag{4}$$

where

$$p_{kint}(t) = \angle a_k(t_i) + \int_{t_i}^{t} \frac{2\pi}{f_s} k f_0(u)\, du \tag{5}$$

[6].

2.1.2 1_{st} *Adaptation and After*

After the 0_{th} adaptation ends, the signal is adapted until it converges to **quasi-harmonicity** and is modeled as

$$s(t) = w(t) \sum_{k=-L}^{L} \left(a_k + t b_k\right) \left| \frac{a_k(t + t_i)}{a_k(t_i)} \right| e^{j p_{kint}(t)} \tag{6}$$

where $p_{kint}(t)$ as in Equation 5, a_k, b_k the complex amplitude and the complex slope of the $k_t h$ component. Then for each time instant, the frequency correction mechanism

$$df_k = \frac{f_s}{2\pi} \frac{\Re\{a_k\}\Im\{b_k\} - \Im\{a_k\}\Re\{b_k\}}{|a_k|^2} \tag{7}$$

is used, only to the components where

$$df_k \leqslant \frac{f_0}{a+1} \tag{8}$$

with a being the adaptation number [6].

2.1.3 *Interpolation*

In this part, for each k_{th} component, the instantaneous values are interpolated according to the following process:

- The instantaneous amplitudes of $|a_k(t)|$ are linearly interpolated
- The instantaneous frequencies $f_k(t)$ are interpolated in 3_{rd} order (spline) interpolation and
- The instantaneous phases $p_k(t)$ are interpolated by integration of instantaneous frequency

After that the signal can be reconstructed as in Equation 1 with the new instantaneous amplitudes and phases [6].

2.1.4 *Signal-to-Reconstruction-Error Ratio (SRER)*

At the end, the **Signal-to-Reconstruction-Error Ratio (SRER)** is computed as

$$\text{SRER} = 20 \log_{10} \frac{\text{std}(s(t))}{\text{std}(s(t) - s_{rec}(t))} \tag{9}$$

where `std` is the Standard Deviation (STD) metric. The above procedure is repeated until the SRER stops increasing, at which point it terminates [6].

2.2 IMPLEMENTATION

For the aforementioned model, there was already a program implemented in **MATLAB**. The given code mainly consisted of functions that perform analysis, synthesis, interpolation, producing structures and some mathematical operations such as filtering and graph plotting. The function of utmost significance was **eaQHMAnalysis**, which applies **extended adaptive Quasi-Harmonic Analysis** to a speech signal, decomposing it into AM-FM components according to the **eaQHM** and iteratively refining it until the reconstructed signal converges to **quasi-harmonicity**. The main objective of this thesis is to convert this algorithm into **Python** language, while keeping the best output in the shortest feasible time. The code was successfully produced using various functions from a lot of **Python** modules. Those are numpy, scipy, pylab, time, statistics, copy, tqdm, warnings and matplotlib. Moreover, many functions were implemented that had to apply some simple operations **MATLAB** can do, like array transposition, array indexing and returning the final element of an array-like structure. Initially the parameters of each signal were extracted from the respective *.mat* files which included most of the signal's information, as well as the settings applied. After using a **Python** version of *SWIPEP Pitch Estimator* [55] created by Disha Garg (available here) for the function, this idea was rejected and all parameters are now created inside the function, while options are passed as input arguments. It is worth mentioning that not only a decomposition is returned but also a resynthesis takes place in **eaQHMAnalysis**. Thus, the function was renamed as **eaQHMAnalysisAndSynthesis** and the reconstructed signal was added to the output variables. The final code is published here.

2.2.1 *Why **Python**?*

From a general perspective, the resulting code is readable and simple to understand, assisting anybody to grasp the structure of the code, even those who are not completely familiar with it. That is because **Python** is well-known for being simple to read by being a high-level programming language [56]. **MATLAB** is also an understandable language but has the disadvantage that it is not free for everyone since it is a commercial software and can only be used by those who own its license. Moreover, additional toolboxes should be bought separately to extend the functionality of the code. On the other hand, **Python** is a free and open-source software, with its modules also being free [57].

As a result, this code can be executed by everyone, and no license is required.

From an expert's point of view, the community of SP and Machine Learning is fairly intimate with **Python** and its basic modules. People specializing in the above fields are very pleased with the way this language works due to its enhanced quality and profitability through the usage of low-level libraries and high-level APIs that are well-maintained. Especially within the last ten years, **Python** has witnessed a remarkable rise in interest regarding the scientific computer community [56]. Moreover, this language provides users with simple mathematical operations other languages find difficult to do or require users to create functions from scratch to perform them. Thus, for a complex and time consuming process such as the algorithm described in Section 2.1, implementing functions for such operations should be the least of the programmer's concerns.

In conclusion, **Python** is an excellent choice for the purpose of SP, as it is an easy language to both read and program and provides programmers with a vast amount of modules. That is why it was selected to implement the eaQHM model, allowing everyone to cope with it and eventually provide new suggestions to enhance it.

2.2.2 *eaQHMAnalysisAndSynthesis*

This function applies **eaQHM Analysis and Synthesis** to a speech signal. As an output, the reconstructed signal along with its components, the SRER per adaptation and the total time elapsed are produced. Due to the usage of many functions and the time complexity of the algorithm, this model is more time consuming than the other models, but achieves a higher decomposition and resynthesis quality of the speech signal.

Initially, the signal is preprocessed according to the options given. At first, a high pass filter may be applied and thenceforth, a *SWIPEP pitch estimator* makes the pitch estimations for the signal. The maximum frequency **Fmax** and partials **Kmax** are measured as

```python
Fmax = int(fs/2-200)

if partials > 0:
    Kmax = partials
else:
    Kmax = int(round(Fmax/min(f0s[:,1])) + 10)
```

Listing 1: Maximum frequency and partials

where **f0s** contain the pitch estimation per frame and **partials** is a variable given as an input which may define **Kmax**. After calculating the voiced and unvoiced frames the preprocess ends.

The signal is then split in time instants. Afterwards, it is iterated up to a fixed number of adaptations ($maxAdpt$) for each time instant within the analysis window and for each two consecutive voiced frames, the following algorithm is initiated:

- In the first adaptation, a full-band harmonicity is assumed and the frame is multiplied with a Blackman Window, thus obtaining the complex amplitudes and slopes via the method of Least Squares.
- After the first adaptation, the FM and AM components of the frame are created, by initially extracting all non-zero frequency values of the frame and the corresponding amplitudes, generating the components previously mentioned containing those values in the appropriate positions, while the rest being zero. Those components are not ready for use yet, as they contain many zero frequency trajectories and amplitudes, so the process must now solve these issues as follows:
  1. A new frequency is *"born"* if there is at least one zero value in the first position of the component and if so, it is replaced with the first non-zero one, for the k_{th} frequency.
  2. A frequency is *"killed"* if there is at least one zero value in the last position of the component and if that is the case then it is replaced with the last non-zero one, as done previously.

  This can be observed more accurately in Listing 2

```
fm_zeros = argwhere(fm[:, k] == 0)
fm_nonzeros = argwhere(fm[:, k])

if len(fm_zeros) != 0:
    fm_zeros_index = fm_zeros[0][0]
    fm_nonzeros_index = fm_nonzeros[0][0]
    if fm_zeros_index == 0:
        fm[fm_zeros_index][k]= fm[fm_nonzeros_index][k]
        am[fm_zeros_index][k] = am[fm_nonzeros_index][k]

        fm_nonzeros = insert(fm_nonzeros, 0,
            fm_zeros_index)

    fm_zeros_index = end(fm_zeros)
    fm_nonzeros_index = end(fm_nonzeros)
    if fm_zeros_index == fm_len-1:
        fm[fm_zeros_index][k] = fm[fm_nonzeros_index][k]
        am[fm_zeros_index][k] = am[fm_nonzeros_index][k]

        fm_nonzeros = append(fm_nonzeros, fm_zeros_index
            )
```

Listing 2: The process of "killing" and/or "giving birth to" frequencies where **fm** is an array-like stucture containing all non-zero instantaneous frequency trajectories. After that process, where all zero frequency trajectories are either "killed" or "born", all those components are linearly interpolated to be extended. This whole procedure is repeated as many times as the number of non-zero frequencies and following its termination, the frame is once again multiplied with a Hamming Window with the application of eaQHM, using the FM and AM components created, and thus once again obtaining the complex amplitudes and slopes. At the end, the correction mechanism is introduced using the obtained items [6].

In both the first adaptation and the following ones, the values of instantaneous amplitudes and phases for each frequency index are estimated via the complex amplitudes and their phases respectively, while the instantaneous frequencies are estimated according to Listing 3.

```
if a == 0:
    fm_recon[tith-1][k] = (k+1)*f0
elif f0 > f0min:
    fm_recon[tith-1][k] = fm_current[tith-1][k] + fmismatch[k]
else:
    fm_recon[tith-1][k] = fm_current[tith-1][k]
```

Listing 3: Instantaneous frequency estimation

Here **a** is the adaptation number, **tith** is the current time instant, **k** is the k_{th} frequency, **fo** is the current pitch estimation, **fomin** is the f_0 threshold and **fmismatch** is the correction mechanism. It can be clearly observed that in the first adaptation, no correction mechanism is used and each frequency is solely evaluated by the pitch estimation of the current time instant. For the rest of the adaptations, the current frequency is selectively fixed by the correction mechanism, depending on whether the estimated frequency exceeds the f_0 threshold or not.

Next, all instantaneous parameters created previously are interpolated over the time instants for each sinusoid, only to the non-zero values of AM components whose indices have time difference. These interpolations are applied as explained in Subsection 2.1.3. As a final step, frequency tracks are generated by unwrapped phases.

At this particular point, the reconstructed signal can be obtained by taking the interpolated amplitude of the mean value of each window's center and using the instantaneous amplitudes and phases to generate a sum of sinusoids, which are then added together. Listing 4 shows this procedure

```
s_recon_tmp = a0_recon + 2*multiply(am_recon, cos(ph_recon)).sum(
    axis=1)
```

Listing 4: Signal Reconstruction

where **ao_recon** is an array-like containing the mean interpolated values and **ao_recon**, **ph_recon** are the instantaneous amplitudes and phases respectively. Finally, the SRER of the current adaptation is calculated. If a SRER is less than some threshold depending on the SRER of the previous adaptation, this means the reconstruction of the signal has adapted and no further adaptation is required. Thus, the iteration terminates and the function returns the necessary output data.

Figures 1, 2 show the console produced with a female speaker speech file as input and the graphs plotted respectively. Similarly for Figures 3, 4 referring to a male speaker file.

2.2.3 *Implementations' Comparison*

In contrast to the original code, the new implementation defaults using the eaQHM model in a full-band analysis while the aQHM model is not supported. For the pitch estimations, only *SWIPEP* can be used and neither the YIN nor the AIR pitch estimators are implemented. Initially, *SWIPEP's* pitch limits were exactly the same as in [6]: $[70, 220]$Hz for male and $[120, 350]$Hz for female speakers. However, thorough examination revealed that those limits did not provide the optimal results and therefore had to be narrowed down into $[70, 180]$Hz for males and $[160, 300]$Hz for females. Two additional domains of $[70, 500]$Hz for other genders and $[300, 600]$Hz for children were added as an option. Pitch limit can also be customized, although it is not suggested. Finally, in this approach, no median smoothing takes place after the pitch estimations, as it was found to severely impair the reconstruction's quality.

2.2.4 *Known Issues*

The resulting code is not without flaws. During the debugging process, it was discovered that the *SWIPEP pitch estimator* did not generate estimates that were identical to those produced by **MATLAB**. Further investigation indicated that the source of the problem was matplotlib.pyplot.specgram, which provided a Fast Fourier Transform (FFT) of the signal that was not as near as it should have been. This causes a chain of divergences, resulting in inaccurate pitch estimations. Additionally, the code's execution time is something that needs to be optimized, however this is less of a problem given **Python's** reputation for being incredibly slow. The next chapter will go into further detail on whether the aforesaid concerns will be problematic.

```
File Selected: D:/eaQHM-analysis-and-synthesis-in-Python/0072_spa_AGC0001_snd_norm_F.wav

You may include a gender (male, female, child or other): female

---- Adaptation No. 0 ----


SRER: 33.18795770944161 dB in Adaptation No: 0
Adaptation Time: 00:00:20

---- Adaptation No. 1 ----


SRER: 40.437593629316446 dB in Adaptation No: 1
Adaptation Time: 00:01:09

---- Adaptation No. 2 ----


SRER: 39.12121867141955 dB in Adaptation No: 2
Adaptation Time: 00:01:13

Signal adapted to 40.437594 dB SRER
Total Time: 00:02:44
```

Figure 1: Output console of the code running a *'.wav'* file of a female speaker. SRER and time elapsed per adaptation are printed.

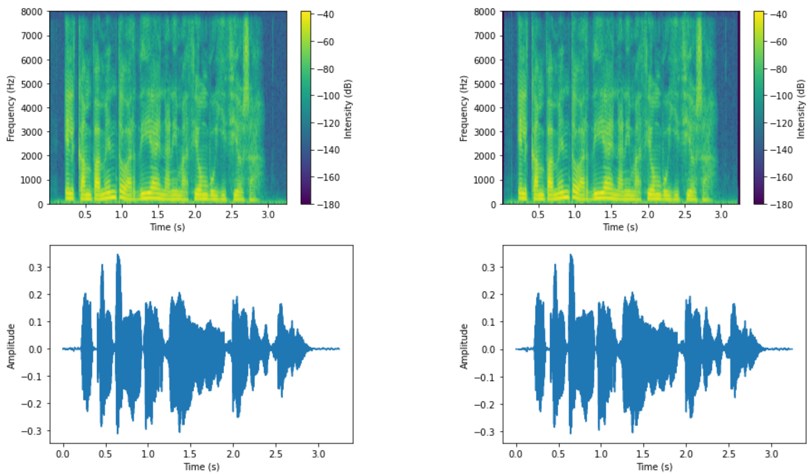


Figure 2: Frequency and Time domains of the female speaker *'.wav'* file (right) and its reconstruction (left).

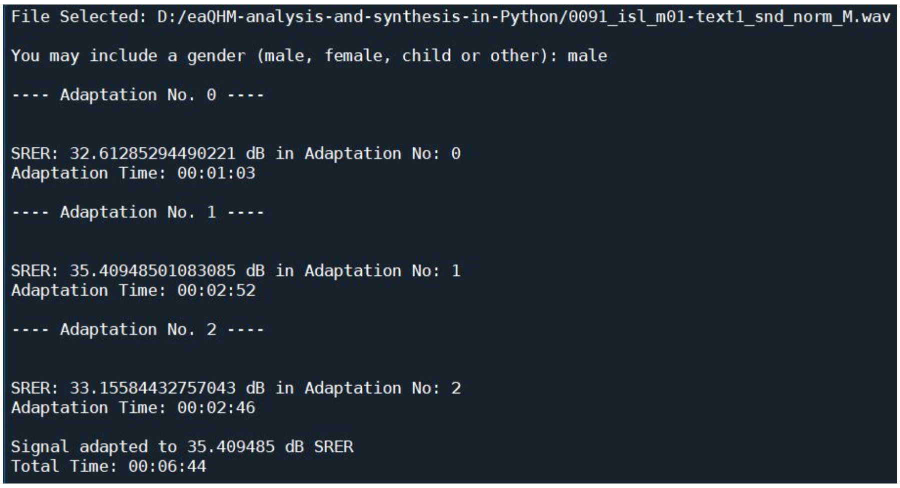


Figure 3: Output console of the code running a *'.wav'* file of a male speaker. SRER and time elapsed per adaptation are printed.

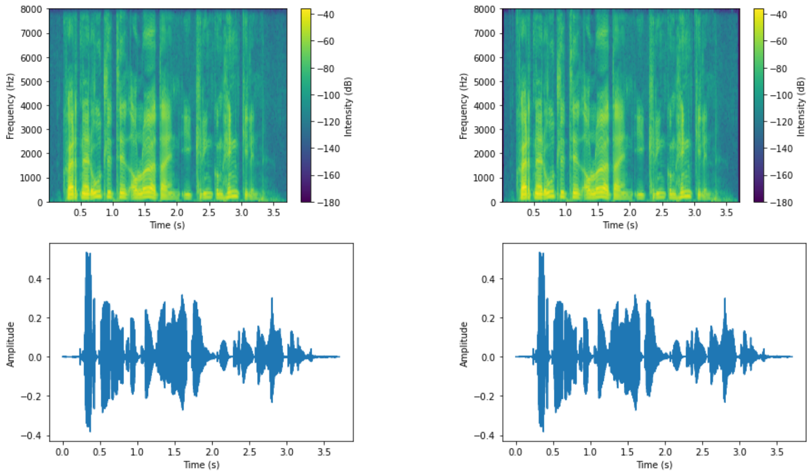


Figure 4: Frequency and Time domains of the male speaker *'.wav'* file (right) and its reconstruction (left).

3 EVALUATION

The translated code was put to a test in order for its accuracy and time consumption to be determined and contrasted with the original. For this purpose two kinds of examinations occurred: *Objective Evaluation* and *Subjective Evaluation*. In *Objective Evaluation*, a database of speech samples was used and several comparisons/metrics were computed for both languages to examine the **eaQHM** model. In *Subjective Evaluation*, all reconstructed database files generated from each code were used for a listening test available online to analyze how the quality of the refining can be captured by the average listener.

The database consists of 32 voice waveforms all sampled in $f_s = 16000\text{Hz}$, with 16 male and 16 female speakers in various languages, them being: *Arabic, English, Finnish, French, German, Greek, Hindu, Icelandic, Italian, Japanese, Korean, Mandarin, Russian, Spanish, Basque* and *Turkish*.

In all files a full waveform analysis was performed with a 15-sample step size and a window size with 3 pitch periods. The maximum number of adaptations allowed was 10. No high pass filter was used and the number of partials was computed from the pitch estimations. The settings of *SWIPEP pitch estimator* were the following: The pitch estimation window was $1ms$ and the pitch limits supported were as mentioned in Subsection 2.2.3.

For the **Python** code, tests were executed in Spyder 4.1.4 IDE with **Python** 3.8.3, whereas for the **MATLAB** code, MATLAB R2015a (8.5.0.197613) was used. Both environments run in Windows 10 64-bit OS on an ASUS FX504GE-DM231T Laptop with 16.0 GB installed RAM and a Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.20 GHz processor.

3.1 OBJECTIVE EVALUATION

In objective analysis, each file from the speech database was examined through time/SRER measurements in both **MATLAB** and **Python** codes. Initially, comparisons of the results each file produced are taken. After that, average and divergence calculations of the results were evaluated with *Mean* and *Standard Deviation (STD)* [58] metrics respectively.

3.1.1 *SRER Evaluation*

In this subsection, the outputs of both the new and the initial implementations are examined to identify the quality of the files' reconstructions the **Python** code produces. A script was created in each language to execute **eaQHMAnalysisAndSynthesis** in **Python** and **eaQHMAnalysis** in **MATLAB** to all speech waveforms of the database and store the results. All those results had their differences measured as

$$D_f(k) = SRER_M(k) - SRER_P(k) \tag{10}$$

with $D_f(k)$, $SRER_M(k)$ and $SRER_P(k)$ being the difference, the SRER calculation from **MATLAB** and from **Python** codes of the k_{th} speech file respectively. Then a threshold was selected to set a minimum amount of dB to be accepted as a successful reconstruction or not. For this evaluation, 5dB was chosen and all differences had to be determined according to that limit. Hence, for the k_{th} difference a state label was included in the following way:

1. **"Improvement"** for $D_f(k) < 0$
2. **"Success"** for $0 \leqslant D_f(k) \leqslant 5$ and
3. **"Failure"** for $D_f(k) \geqslant 5$

By the end of this process the accuracy A of the algorithm was calculated using

$$A = \frac{K - F}{K} 100\% \tag{11}$$

where K and F are the total number of files and "Failure" labels respectively. After applying this evaluation, the results were a 75.8% accuracy with 10 "Improvements", 15 "Successes" and 8 "Failures".

All those measurements can be seen in Table 1. The Table also includes the gender, language and state of each file and for every comparison, the $\max\{SRER_M, SRER_P\}$ is highlighted.

Speech File	Gender	Language	MATLAB SRER (dB)	Python SRER (dB)	Difference	State
0071_spa_KJC0017_snd_norm_M	Male	Spanish	29.7960	33.1006	−3.30	Improvement
0072_spa_AGC0001_snd_norm_F	Female	Spanish	40.0348	40.4376	−0.40	Improvement
0081_eus_IBE0031_snd_norm_M	Male	Basque	37.7325	37.794	−0.06	Improvement
0082_eus_AGE0020_snd_norm_F	Female	Basque	38.7133	33.2453	5.47	Failure
0091_isl_m01-text1_snd_norm_M	Male	Icelandic	33.7482	35.4095	−1.66	Improvement
0092_isl_f01-text1_snd_norm_F	Female	Icelandic	36.4854	33.7814	2.70	Success
0101_ind_ut-ml-m4_snd_norm_M	Male	Hindu	33.1418	30.1426	3.00	Success
0102_ind_f06-063a_snd_norm_F	Female	Hindu	28.6593	29.3309	−0.67	Improvement
0111_tur_evenekm72_snd_norm_M	Male	Turkish	34.3413	29.1157	5.23	Failure
0112_tur_yasemin51_snd_norm_F	Female	Turkish	37.7703	35.8092	1.96	Success
0121_fin_mv_0606_snd_norm_M	Male	Finnish	37.6553	34.4583	3.20	Success
0122_fin_01l_rich_0247_snd_norm_F	Female	Finnish	37.0732	37.2261	−0.15	Improvement
0131_ara_ut-ml-m2_snd_norm_M	Male	Arabic	41.4782	29.2443	12.23	Failure
0132_ara_ut-ml-f1_snd_norm_F	Female	Arabic	34.4193	21.2076	13.21	Failure
0141_chi_ut-ml-m1_snd_norm_M	Male	Mandarin	25.8781	24.8115	1.07	Success
0142_chi_ut-ml-f3_snd_norm_F	Female	Mandarin	33.0566	23.4867	9.57	Failure
0151_kor_ut-ml-m1_snd_norm_M	Male	Korean	29.8495	29.9017	−0.05	Improvement
0152_kor_ut-ml-f3_snd_norm_F	Female	Korean	33.0125	24.306	8.71	Failure
0161_rus_ut-ml-m2_snd_norm_M	Male	Russian	34.5272	31.9083	2.62	Success
0162_rus_ut-ml-f2_snd_norm_F	Female	Russian	35.2661	28.4492	6.82	Failure
nitech_jp_atr503_m001_j31_snd_norm_M	Male	Japanese	39.0687	36.624	2.44	Success
af049orgh_snd_norm_F	Female	Japanese	37.5919	38.784	−1.19	Improvement
arctic_bdl1_snd_norm_M	Male	English	35.7918	31.6805	4.11	Success
arctic_slt1_snd_norm_F	Female	English	41.2167	40.1741	1.04	Success
XavierReference1_2_snd_norm_M	Male	French	38.3113	35.2703	3.04	Success
Christine_01_neutre_snd_norm_F	Female	French	39.1342	37.4503	1.68	Success
emodb_m_39_snd_norm_M	Male	German	33.4637	33.4073	0.06	Success
emodb_f_107_snd_norm_F	Female	German	34.2833	29.7604	4.52	Success
Kostas268_snd_norm_M	Male	Greek	36.4642	30.4209	6.04	Failure
Maria263_snd_norm_F	Female	Greek	31.9856	33.1773	−1.19	Improvement
Luciano_K_It_m_s_snd_norm_M	Male	Italian	32.9363	34.4697	−1.53	Improvement
Tiziana_C_It_f_s_snd_norm_F	Female	Italian	39.0700	36.5872	2.48	Success

Table 1: SRER values and comparisons (dB). The gender, language and state of each file are included. The maximum SRER of each measurement is marked in a yellow color.

Additionally, a SRER per adaptation comparison took place. To evaluate the best, average and worst case scenarios of the algorithm, four specific files were chosen:

1. The "Improvement" with the highest absolute difference (Best-case scenario. Figure 5)

2. The "Success" with the lowest difference (Average best-case scenario. Figure 6)

3. The "Success" with the highest difference (Average worst-case scenario. Figure 7) and

4. The "Failure" with the highest difference (Worst-case scenario. Figure 8)

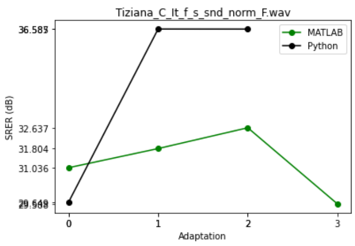


Figure 5: SRER per Adaptation of the best "Improvement"

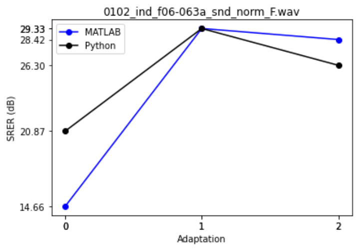


Figure 6: SRER per Adaptation of the best "Success"

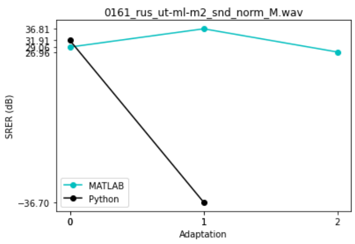


Figure 7: SRER per Adaptation of the worst "Success"

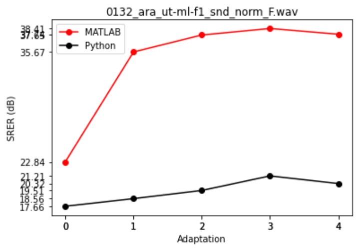


Figure 8: SRER per Adaptation of the worst "Failure"

Figures 5-8 illustrate that the implemented algorithm may terminate in a different adaptation from the original. It can also be assumed that the starting SRER state does not affect the results at the algorithm's termination result.

Table 2 shows the *Mean* and *STD* metrics for each code per gender. The differences of each metric is also given.

Metric (dB) / Language / Gender / Difference (dB)	*Mean*		*STD*	
	MATLAB	**Python**	**MATLAB**	**Python**
Males	34.6	32.4	3.95	3.35
Difference	2.2		0.6	
Females	36.1	32.7	3.37	6.04
Difference	3.4		−2.67	

Table 2: *Mean* and *STD* for the SRERs of each code per gender. The comparisons are included.

It can clearly be observed that the *Mean* for both genders is lower than the initial code, yet the differences are negligible regarding the dB scale. So, merely by looking at the mean numbers, one can conclude that each code may achieve equivalent signal reconstructions. The *STD*, on the other hand, is just smaller for males, meaning that the values are clustered tightly around the mean value and thus the *STD* is lowered [58]. For females though, the value is way greater and exceeds the allowed threshold (5dB). This can be expected since 5 out of 8 "Failures" come from female speakers and 3 of those have big D_f. In this case, the values of females are widely scattered around the average value.

3.1.2 *Time Evaluation*

Additionally with the output measurements, the time duration each file consumed were taken. For this purpose, time measurement modules (time for **Python** and tic toc for **MATLAB**) were utilized. A script was created on each language to run the functions and calculate the total time from the initialization to the termination. However, just one measurement does not provide a full picture of how long each file takes. Therefore, 10 time computations were taken and the mean of them was extracted. Afterwards, similarly with Subsection 3.1.1, the difference of each mean calculation was extracted as

$$D_t(k) = time_M(k) - time_P(k) \tag{12}$$

with $D_t(k)$, $time_M(k)$ and $time_P(k)$ being the difference, the mean execution time from **MATLAB** and from **Python** codes of the k_{th}

speech file respectively. Unlike 3.1.1, if $D_t(k) \ll 0$, that means there is no possibility of improving time for the k_{th} file. Furthermore, because time optimization is not a major concern, neither state labels nor percentage computation are needed.

All those measurements can be seen in Table 3, which also includes the gender and language of each file. For every comparison, the min$\{time_M, time_P\}$ is highlighted.

Speech File	Gender	Language	MATLAB Time (MM:SS)	Python Time (MM:SS)	Difference
0071_spa_KJC0017_snd_norm_M	Male	Spanish	05:03	05:00	00:03
0072_spa_AGC0001_snd_norm_F	Female	Spanish	02:15	04:48	-02:33
0081_eus_IBE0031_snd_norm_M	Male	Basque	06:06	04:59	01:07
0082_eus_AGE0020_snd_norm_F	Female	Basque	04:12	04:29	-00:17
0091_isl_mo1-text1_snd_norm_M	Male	Icelandic	06:25	05:09	01:16
0092_isl_fo1-text1_snd_norm_F	Female	Icelandic	01:42	03:55	-02:13
0101_ind_ut-ml-m4_snd_norm_M	Male	Hindu	03:02	04:16	-01:14
0102_ind_f06-063a_snd_norm_F	Female	Hindu	03:10	03:47	-00:37
0111_tur_evenekm72_snd_norm_M	Male	Turkish	05:39	04:16	01:23
0112_tur_yasemin51_snd_norm_F	Female	Turkish	02:41	03:47	-01:06
0121_fin_mv_0606_snd_norm_M	Male	Finnish	07:39	04:16	03:23
0122_fin_01l_rich_0247_snd_norm_F	Female	Finnish	01:51	03:46	-01:55
0131_ara_ut-ml-m2_snd_norm_M	Male	Arabic	05:50	04:14	01:36
0132_ara_ut-ml-f1_snd_norm_F	Female	Arabic	04:03	03:47	00:16
0141_chi_ut-ml-m1_snd_norm_M	Male	Mandarin	04:28	04:16	00:12
0142_chi_ut-ml-f3_snd_norm_F	Female	Mandarin	02:44	03:48	-01:04
0151_kor_ut-ml-m1_snd_norm_M	Male	Korean	06:54	04:19	02:35
0152_kor_ut-ml-f3_snd_norm_F	Female	Korean	04:55	03:46	01:09
0161_rus_ut-ml-m2_snd_norm_M	Male	Russian	06:21	04:29	01:52
0162_rus_ut-ml-f2_snd_norm_F	Female	Russian	02:53	03:57	-01:04
nitech_jp_atr503_m001_j31_snd_norm_M	Male	Japanese	03:28	04:38	-01:10
af0490rgh_snd_norm_F	Female	Japanese	02:17	04:02	-01:45
arctic_bdl1_snd_norm_M	Male	English	05:04	04:34	00:30
arctic_slt1_snd_norm_F	Female	English	02:32	04:12	-01:40
XavierReference1_2_snd_norm_M	Male	French	04:50	04:23	00:27
Christine_01_neutre_snd_norm_F	Female	French	01:39	04:07	-02:28
emodb_m_39_snd_norm_M	Male	German	02:32	04:22	-01:50
emodb_f_107_snd_norm_F	Female	German	01:32	04:09	-02:37
Kostas268_snd_norm_M	Male	Greek	11:57	04:18	07:39
Maria263_snd_norm_F	Female	Greek	03:44	04:05	-00:21
Luciano_K_It_m_s_snd_norm_M	Male	Italian	04:15	04:19	-00:04
Tiziana_C_It_f_s_snd_norm_F	Female	Italian	02:13	04:07	-01:54

Table 3: Time values and comparisons. The gender and language of each file are included. The minimum time of each measurement is marked in a yellow color.

With only a quick glance, one can say that most files do not achieve a time improvement. However, the *Mean* and *STD* metrics shown in Table 4 suggest slightly different results.

Metric (MM:SS) / Language / Gender	*Mean*		*STD*	
	MATLAB	**Python**	**MATLAB**	**Python**
Males	05:35	04:29	02:11	00:18
Females	02:46	04:02	01:00	00:17

Table 4: *Mean* and *STD* for the execution time measurements of each code per gender.

What the above Table indicates is that in **MATLAB**, the *Mean* for males and females are far more deviant than in **Python**. It may also be expected that **MATLAB** conducts the operation faster for female speakers. This is because female speakers have a higher pitch range, resulting in fewer harmonics and therefore faster processing. **Python** does so too, but performs almost as well for both genders, hence it is safe to assume that, irrespective of the speaker's gender, the implemented algorithm will conduct the procedure with more comparable run-times opposing to the original which will fare incredibly quicker on female speakers. What matters most though is the *STD* which is significantly lower in **Python** than in **MATLAB** findings. This implies that the implemented algorithm's run-time does not depart significantly from the average, but the execution time of the given code might vary greatly. As an outcome, **Python** code is more stable in terms of execution time, while **MATLAB** code will either be extraordinarily fast or extremely sluggish.

3.1.3 *Conclusions*

To summarize, the above examinations show that the implementation may indeed provide successful output. However the SRER evaluation results suggest that there is no guarantee for a better or even a marginal reconstruction, and for female speakers there is a slightly lower possibility for that. As far as the execution duration, it may be satisfying at times but disappointing at others. What cannot be debated though is that no matter the gender of the speaker, the execution time shall be kept as consistent as practicable. This however may vary based on the device's specifications and CPU usage during execution.

3.2 SUBJECTIVE EVALUATION

In subjective analysis, each database file was reconstructed in both **Python** and **MATLAB** implementations before being exported as '.wav' files. All 32 pairs were then uploaded in an online listening test which was then shared to some candidates. Screenshots of the listening test can be seen in Figure 9.

The goal of the test was for applicants to detect any differences in each pair merely based on audio quality (noise, clicks, distortions, amplitude etc.). It was suggested that they use a high-quality audio device (headphones preferably, or else earbuds, monitor speakers or worst-case laptop speakers) and conduct the assessment in a quiet place. Candidates were instructed to listen to both files of each pair and then select a label as: **(1): "Same"** if the files sounded identical or **(2): "Different"** if any difference was detected.

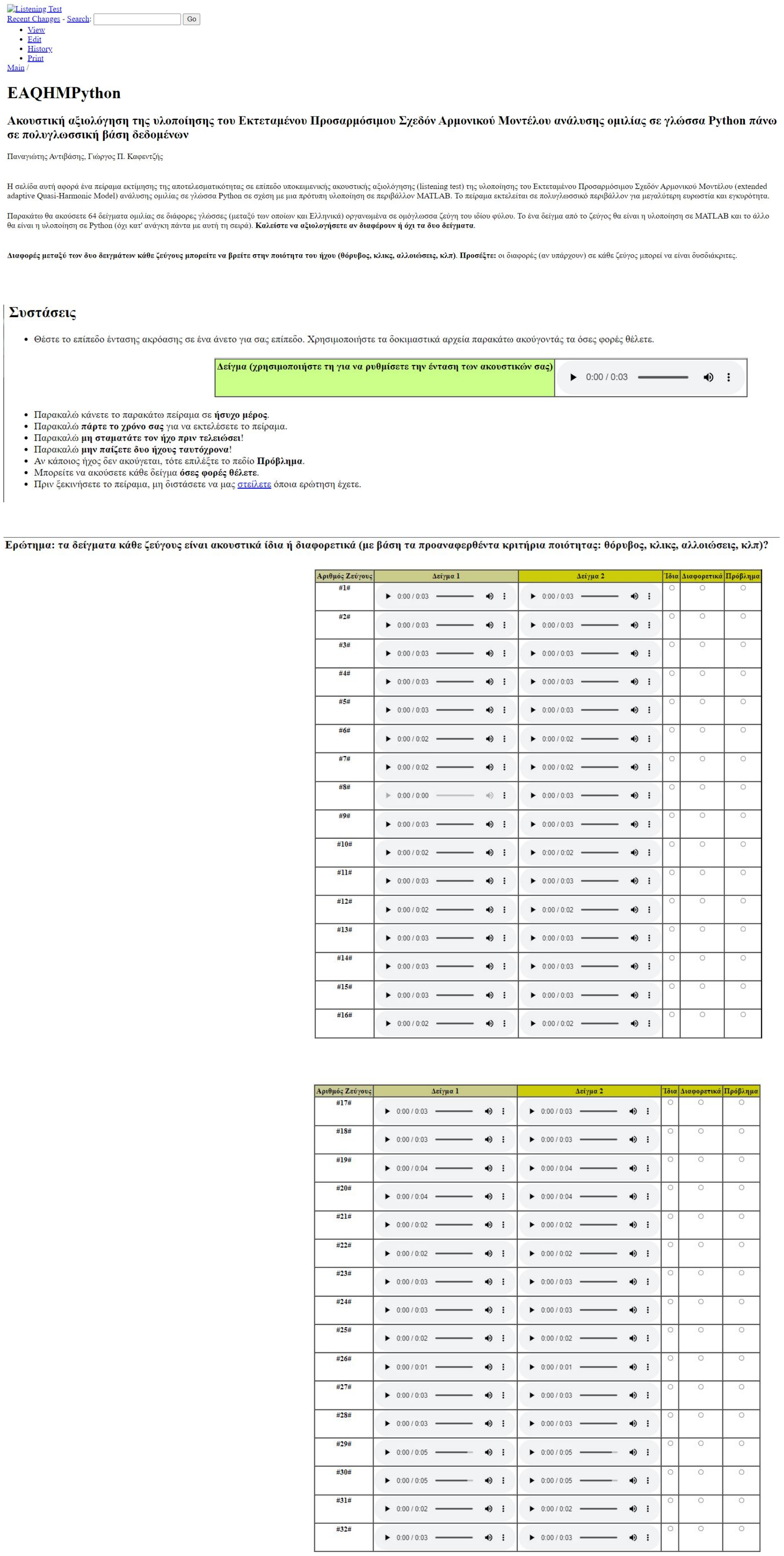
The test was taken by 15 listeners in total, all of them speaking Greek as their first language. At the end of the evaluation, the results were passed in a document file and the overall percentage of **"Same"** labels for each pair was calculated. The criterion of $> 60\%$ was used to determine if two files were identical or not. Given the fairly small number of contenders and the fact that none of them performed the assessment utilising speakers, the audio device used will not be taken into account for this evaluation. It is also important to mention that for the pair corresponding to file "0102_ind_f06-063a_snd_norm_F.wav", 27% of listeners faced technical issues.

The results showed that 76% of the pairs were judged as identical, with only 4 files being identified as different. All those measurements can be seen in Table 5.

To complete the subjective examination, Tables 5 and 1 were contrasted, concentrating mostly on the 4 files labeled as **"Different"**. It was observed that half of those files were **"Improvements"**, whereas the other half were **"Successes"**. Therefore, all **"Failures"** were labeled as **"Same"**, which means that differences in reconstruction quality (if any) are difficult to detect and can thus be disregarded.

3.2.1 *Conclusions*

The listening test showed that despite how SRERs diverge, the outcome files from both **Python** and **MATLAB** codes are similar overall, and if any differences in amplitude or noise do exist, they are hardly captured by the average listener. This lends credence to the thesis' goal, implying that the algorithm generates results that are indeed desirable, regardless the code's success.



Listening Test
Recent Changes - Search: Go
- View
- Edit
- History
- Print

Main /

EAQHMPython

Ακουστική αξιολόγηση της υλοποίησης του Εκτεταμένου Προσαρμόσιμου Σχεδόν Αρμονικού Μοντέλου ανάλυσης ομιλίας σε γλώσσα Python πάνω σε πολυγλωσσική βάση δεδομένων

Παναγιώτης Αντιβάσης, Γιώργος Π. Καφεντζής

Η σελίδα αυτή αφορά ένα πείραμα εκτίμησης της αποτελεσματικότητας σε επίπεδο υποκειμενικής ακουστικής αξιολόγησης (listening test) της υλοποίησης του Εκτεταμένου Προσαρμόσιμου Σχεδόν Αρμονικού Μοντέλου (extended adaptive Quasi-Harmonic Model) ανάλυσης ομιλίας σε γλώσσα Python σε σχέση με μια πρότυπη υλοποίηση σε περιβάλλον MATLAB. Το πείραμα εκτελείται σε πολυγλωσσικό περιβάλλον για μεγαλύτερη ευρωστία και εγκυρότητα.

Παρακάτω θα ακούσετε 64 δείγματα ομιλίας σε διάφορες γλώσσες (μεταξύ των οποίων και Ελληνικά) οργανωμένα σε ομόγλωσσα ζεύγη του ιδίου φύλου. Το ένα δείγμα από το ζεύγος θα είναι η υλοποίηση σε MATLAB και το άλλο θα είναι η υλοποίηση σε Python (όχι κατ' ανάγκη πάντα με αυτή τη σειρά). **Καλείστε να αξιολογήσετε αν διαφέρουν ή όχι τα δυο δείγματα**.

Διαφορές μεταξύ των δυο δειγμάτων κάθε ζεύγους μπορείτε να βρείτε στην ποιότητα του ήχου (θόρυβος, κλικς, αλλοιώσεις, κλπ). Προσέξτε: οι διαφορές (αν υπάρχουν) σε κάθε ζεύγος μπορεί να είναι δυσδιάκριτες.

Συστάσεις

- Θέστε το επίπεδο έντασης ακρόασης σε ένα άνετο για σας επίπεδο. Χρησιμοποιήστε τα δοκιμαστικά αρχεία παρακάτω ακούγοντάς τα όσες φορές θέλετε.

Δείγμα (χρησιμοποιήστε τη για να ρυθμίσετε την ένταση των ακουστικών σας)	0:00 / 0:03

- Παρακαλώ κάνετε το παρακάτω πείραμα σε **ήσυχο μέρος**.
- Παρακαλώ **πάρτε το χρόνο σας** για να εκτελέσετε το πείραμα.
- Παρακαλώ **μη σταματάτε τον ήχο πριν τελειώσει**!
- Παρακαλώ **μην παίζετε δυο ήχους ταυτόχρονα**!
- Αν κάποιος ήχος δεν ακούγεται, τότε επιλέξτε το πεδίο **Πρόβλημα**.
- Μπορείτε να ακούσετε κάθε δείγμα **όσες φορές θέλετε**.
- Πριν ξεκινήσετε το πείραμα, μη διστάσετε να μας στείλετε όποια ερώτηση έχετε.

Ερώτημα: τα δείγματα κάθε ζεύγους είναι ακουστικά ίδια ή διαφορετικά (με βάση τα προαναφερθέντα κριτήρια ποιότητας: θόρυβος, κλικς, αλλοιώσεις, κλπ)?

Αριθμός Ζεύγους	Δείγμα 1	Δείγμα 2	Ίδια	Διαφορετικά	Πρόβλημα
#1#	0:00 / 0:03	0:00 / 0:03	○	○	○
#2#	0:00 / 0:03	0:00 / 0:03	○	○	○
#3#	0:00 / 0:03	0:00 / 0:03	○	○	○
#4#	0:00 / 0:03	0:00 / 0:03	○	○	○
#5#	0:00 / 0:03	0:00 / 0:03	○	○	○
#6#	0:00 / 0:02	0:00 / 0:02	○	○	○
#7#	0:00 / 0:02	0:00 / 0:02	○	○	○
#8#	0:00 / 0:00	0:00 / 0:03	○	○	○
#9#	0:00 / 0:03	0:00 / 0:03	○	○	○
#10#	0:00 / 0:02	0:00 / 0:02	○	○	○
#11#	0:00 / 0:03	0:00 / 0:03	○	○	○
#12#	0:00 / 0:02	0:00 / 0:02	○	○	○
#13#	0:00 / 0:03	0:00 / 0:03	○	○	○
#14#	0:00 / 0:03	0:00 / 0:03	○	○	○
#15#	0:00 / 0:03	0:00 / 0:03	○	○	○
#16#	0:00 / 0:02	0:00 / 0:02	○	○	○

Αριθμός Ζεύγους	Δείγμα 1	Δείγμα 2	Ίδια	Διαφορετικά	Πρόβλημα
#17#	0:00 / 0:03	0:00 / 0:03	○	○	○
#18#	0:00 / 0:03	0:00 / 0:03	○	○	○
#19#	0:00 / 0:04	0:00 / 0:04	○	○	○
#20#	0:00 / 0:04	0:00 / 0:04	○	○	○
#21#	0:00 / 0:02	0:00 / 0:02	○	○	○
#22#	0:00 / 0:02	0:00 / 0:02	○	○	○
#23#	0:00 / 0:03	0:00 / 0:03	○	○	○
#24#	0:00 / 0:03	0:00 / 0:03	○	○	○
#25#	0:00 / 0:02	0:00 / 0:02	○	○	○
#26#	0:00 / 0:01	0:00 / 0:01	○	○	○
#27#	0:00 / 0:03	0:00 / 0:03	○	○	○
#28#	0:00 / 0:03	0:00 / 0:03	○	○	○
#29#	0:00 / 0:05	0:00 / 0:05	○	○	○
#30#	0:00 / 0:05	0:00 / 0:05	○	○	○
#31#	0:00 / 0:02	0:00 / 0:02	○	○	○
#32#	0:00 / 0:03	0:00 / 0:03	○	○	○

Figure 9: The listening test website. The instructions of the test are shown. The file pairs follow next.

Speech File	Gender	Language	"Same" Percentage	Final Result
0071_spa_KJC0017_snd_norm_M	Male	Spanish	87%	Same
0072_spa_AGC0001_snd_norm_F	Female	Spanish	100%	Same
0081_eus_IBE0031_snd_norm_M	Male	Basque	80%	Same
0082_eus_AGE0020_snd_norm_F	Female	Basque	80%	Same
0091_isl_m01-text1_snd_norm_M	Male	Icelandic	73%	Same
0092_isl_f01-text1_snd_norm_F	Female	Icelandic	67%	Same
0101_ind_ut-ml-m4_snd_norm_M	Male	Hindu	80%	Same
0102_ind_f06-063a_snd_norm_F	Female	Hindu	45%	Different
0111_tur_evenekm72_snd_norm_M	Male	Turkish	87%	Same
0112_tur_yasemin51_snd_norm_F	Female	Turkish	73%	Same
0121_fin_mv_0606_snd_norm_M	Male	Finnish	60%	Different
0122_fin_01l_rich_0247_snd_norm_F	Female	Finnish	93%	Same
0131_ara_ut-ml-m2_snd_norm_M	Male	Arabic	73%	Same
0132_ara_ut-ml-f1_snd_norm_F	Female	Arabic	80%	Same
0141_chi_ut-ml-m1_snd_norm_M	Male	Mandarin	73%	Same
0142_chi_ut-ml-f3_snd_norm_F	Female	Mandarin	80%	Same
0151_kor_ut-ml-m1_snd_norm_M	Male	Korean	80%	Same
0152_kor_ut-ml-f3_snd_norm_F	Female	Korean	73%	Same
0161_rus_ut-ml-m2_snd_norm_M	Male	Russian	67%	Same
0162_rus_ut-ml-f2_snd_norm_F	Female	Russian	67%	Same
nitech_jp_atr503_m001_j31_snd_norm_M	Male	Japanese	87%	Same
af049orgh_snd_norm_F	Female	Japanese	73%	Same
arctic_bdl1_snd_norm_M	Male	English	67%	Same
arctic_slt1_snd_norm_F	Female	English	60%	Different
XavierReference1_2_snd_norm_M	Male	French	73%	Same
Christine_01_neutre_snd_norm_F	Female	French	87%	Same
emodb_m_39_snd_norm_M	Male	German	73%	Same
emodb_f_107_snd_norm_F	Female	German	93%	Same
Kostas268_snd_norm_M	Male	Greek	73%	Same
Maria263_snd_norm_F	Female	Greek	60%	Different
Luciano_K_It_m_s_snd_norm_M	Male	Italian	80%	Same
Tiziana_C_It_f_s_snd_norm_F	Female	Italian	93%	Same

Table 5: The results of the listening test. The gender and language of each file are included.

4 CONCLUSION AND FUTURE WORK

4.1 CONCLUSION

This thesis emphasized on the significance of SP in many facets of our life before proceeding into the SMs and their various implementations. Then after focusing on the primary subject, namely the eaQHM, an implementation of it is analyzed and contrasted with a comparable previously completed one. The initial code is developed in **MATLAB**, while the produced one in **Python**. Evaluation showed that there are several results that are not as intended and this needs to be refined. Even such differences, however, are not perceptible to the ordinary listener and cannot be noticed with a single hearing. Overall, the generated code yields marginal or better reconstructions in a perhaps slow but most consistent execution time compared to the original.

4.2 FUTURE WORK

Certainly, there are still improvements that can be made. Even though there are techniques to make **Python** programs run quicker [59], time optimization is not of grave interest, since **Python** is already a very sluggish programming language by nature. What is needed though is the reconstruction optimization and the increment of the code's accuracy. The first idea is to correct the problem with specgram mentioned in Section 2.2.4, most likely by substituting equivalent functions, despite the fact that an attempt has previously been made. Another suggestion is to introduce the pitch estimator YIN. As [6] states, SRERs generated with YIN differs little from SRERs produced with SWIPEP. Thus, its implementation may resolve the pitch estimation problem. And finally, replacing SWIPEP with another version of SWIPE such as pysptk.sptk.swipe or any other stable pitch estimator might be another option for this issue.

Although all the above ideas sound easy on paper due to the elimination of failed reconstructions, applying any of them would nullify the improvements too. Nonetheless, if more testing reveals that the accuracy decreases, those options will be considered.

BIBLIOGRAPHY

[1] S. Syed et al. "Comparative Analysis of CNN and RNN for Voice Pathology Detection." In: *BioMed Research International* 2021 (Apr. 2021), pp. 1–8.

[2] D. Barnard. *Average Speaking Rate and Words per Minute*. Jan. 2018. URL: https://virtualspeech.com/blog/average-speaking-rate-words-per-minute (visited on 12/08/2021).

[3] G. Kafentzis. "Adaptive Sinusoidal Models for Speech with Applications in Speech Modifications and Audio Analysis." In: (June 2014).

[4] I. Salomie. "Amplifier and signal filter." June 2013.

[5] M. Matsuoka and S. Namiki. "Audio signal interpolation device and audio signal interpolation method." Feb. 2014.

[6] G. Kafentzis, O. Rosec, and Y. Stylianou. "Robust Full-band Adaptive Sinusoidal Analysis of Speech." In: May 2014.

[7] B.-H. Juang and L. Rabiner. "Speech Recognition, Automatic: History." In: *Encyclopedia of Language & Linguistics* (Dec. 2006).

[8] R. Gray. "A History of Realtime Digital Speech on Packet Networks: Part II of Linear Predictive Coding and the Internet Protocol." In: *Foundations and Trends in Signal Processing* 3 (Jan. 2010), pp. 203–303.

[9] B. Edwards. *VC&G - VC&G Interview: 30 Years Later, Richard Wiggins Talks Speak & Spell Development*. Dec. 2008. URL: http://www.vintagecomputing.com/index.php/archives/528.

[10] X. Huang, J. Baker, and R. Reddy. "A Historical Perspective of Speech Recognition." In: *Communications of the ACM* 57 (Jan. 2014), pp. 94–103.

[11] M. Hoy. "Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants." In: *Medical Reference Services Quarterly* 37 (Jan. 2018), pp. 81–88.

[12] T. Martin and D. Priest. *The complete list of Alexa commands so far*. Retrieved 10 December 2017. Sept. 2017. URL: https://www.cnet.com/home/smart-home/every-alexa-command-you-can-give-your-amazon-echo-smart-speaker-or-display/.

[13] A. Kongthon et al. "Implementing an online help desk system based on conversational agent." In: Jan. 2009, pp. 450–451.

[14] *Alexa Voice Service Overview (v20160207) | Alexa Voice Service*. URL: https://developer.amazon.com/en-US/docs/alexa/alexa-voice-service/api-overview.html (visited on 12/06/2021).

[15] N. Westerby. *The Gdańsk man who brought US giant Amazon to Poland*. May 2020. URL: https://www.thefirstnews.com/article/the-gdansk-man-who-brought-us-giant-amazon-to-poland-12713 (visited on 12/06/2021).

[16] I. Lunden. *Amazon Gets Into Voice Recognition, Buys Ivona Software To Compete Against Apple's Siri*. Jan. 2013. URL: https://techcrunch.com/2013/01/24/amazon-gets-into-voice-recognition-buys-ivona-software-to-compete-against-apples-siri/ (visited on 12/06/2021).

[17] D. Etherington. *Amazon Echo Is A $199 Connected Speaker Packing An Always-On Siri-Style Assistant*. Retrieved September 2, 2016. Nov. 2014. URL: https://techcrunch.com/2014/11/06/amazon-echo/ (visited on 12/06/2021).

[18] J. Baker. "The DRAGON system–An overview." In: *Acoustics, Speech and Signal Processing, IEEE Transactions on* 23 (Mar. 1975), pp. 24–29.

[19] F. Jelinek, L. Bahl, and R. Mercer. "Design of a linguistic statistical decoder for the recognition of continuous speech." In: *IEEE Transactions on Information Theory* 21.3 (1975), pp. 250–256.

[20] X. Huang, M. Jackm, and Y. Ariki. "Hidden Markov Models for Speech Recognition." In: *Edinburgh University Press* (1990).

[21] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing*. Jan. 2001.

[22] J. Walrand. "Speech Recognition: A." In: June 2021, pp. 205–215. ISBN: 978-3-030-49994-5.

[23] B. Bosker. *SIRI RISING: The Inside Story Of Siri's Origins – And Why She Could Overshadow The iPhone*. Updated December 6, 2017, Retrieved June 10, 2017. Jan. 2013. URL: https://www.huffpost.com/entry/siri-do-engine-apple-iphone_n_2499165 (visited on 12/06/2021).

[24] P. Olson. *Steve Jobs Leaves A Legacy In A.I. With Siri*. This article is more than 10 years old, Retrieved October 5, 2019. URL: https://www.forbes.com/sites/parmyolson/2011/10/06/steve-jobs-leaves-a-legacy-in-a-i-with-siri/?sh=215f1be35dd3 (visited on 12/06/2021).

[25] K. Hodgkins. *Apple's Knowledge Navigator, Siri and the iPhone 4S*. Retrieved June 10, 2017. Oct. 2011. URL: https://www.huffpost.com/entry/siri-do-engine-apple-iphone_n_2499165 (visited on 12/06/2021).

[26] A. Rosen. *Apple Knowledge Navigator Video from 1987 Predicts Siri, iPad and More*. URetrieved June 10, 2017. Oct. 2011. URL: https://www.cultofmac.com/120716/apple-knowledge-navigator-video-from-1987-predicts-siri-ipad-and-more/ (visited on 12/06/2021).

[27] L. Wallace. *The future is AI, and Google just showed Apple how it's done*. Retrieved July 5, 2018. Oct. 2016. URL: https://www.cultofmac.com/447898/google-home-google-assistant-siri-ai/ (visited on 12/08/2021).

[28] D. Bohn. *Google is introducing a new Smart Display platform*. Retrieved January 13, 2018. Jan. 2018. URL: https://www.theverge.com/2018/1/8/16860142/new-google-assistant-speakers-screens-lenovo-jbl-lg-ces-2018 (visited on 12/08/2021).

[29] M. Bronstein. *A more helpful Google Assistant for your every day*. Retrieved January 13, 2018. Jan. 2020. URL: https://blog.google/products/assistant/ces-2020-google-assistant/ (visited on 12/08/2021).

[30] J. Xu. "Adaptations in Speech Processing." PhD thesis. July 2021.

[31] H. M. A. Mohammed et al. "Voice Pathology Classification Using Machine Learning." In: Aug. 2021.

[32] S. Syed et al. "Inter classifier comparison to detect voice pathologies." In: *Mathematical Biosciences and Engineering* 18 (Apr. 2021), pp. 2258–2273.

[33] F. Amara and F. Mohamed. "Voice Pathologies Classification Using GMM And SVM Classifiers." In: *International Journal of Mathematics and Computers in Simulation* 15 (Nov. 2021), pp. 110–114.

[34] P. Harár et al. *Towards Robust Voice Pathology Detection*. July 2019.

[35] R. W. Ralston. "Signal processing - Opportunities for superconductive circuits." In: *IEEE Transactions on Magnetics* 21 (Mar. 1985), pp. 181–185.

[36] H. Noguchi. "Analog-to-digital converter and analog-to-digital conversion method." May 2015.

[37] J. Han. *Wireless Communications by Theodore S. Rappaport (z-lib.org)*. Mar. 2020.

[38] A. V. Oppenheim. *Applications of digital signal processing*. 1978.

[39] T. Ogunfunmi et al. "A Primer on Deep Learning Architectures and Applications in Speech Processing." In: *Circuits, Systems, and Signal Processing* 38 (Aug. 2019).

[40] M.J. Bishop and E.A. Thompson. "Maximum likelihood alignment of DNA sequences* 1." In: *Journal of molecular biology* 190 (Aug. 1986), pp. 159–65.

[41] S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Vol. 3. Apr. 1998. ISBN: 0-521-62971-3.

[42] T. Delić et al. "Cross-Lingual Neural Network Speech Synthesis Based on Multiple Embeddings." In: *International Journal of Interactive Multimedia and Artificial Intelligence* 7 (Dec. 2021), pp. 110–120.

[43] J. Fulcher et al. "A neural network, speech-based approach to literacy." In: Jan. 2003, pp. 40–45.

[44] M. Müller. *Information Retrieval for Music and Motion*. Jan. 2007. ISBN: 978-3-540-74047-6.

[45] M. Müller. "Dynamic time warping." In: *Information Retrieval for Music and Motion* 2 (Jan. 2007), pp. 69–84.

[46] N. Landwehr, M. Hall, and E. Frank. "Logistic Model Trees." In: *Machine Learning* 59 (Feb. 2005), pp. 161–205.

[47] V. Garcia, F. Nielsen, and R. Nock. "Hierarchical Gaussian Mixture Model." In: Jan. 2010, pp. 4070–4073.

[48] D.A. Reynolds and R.C. Rose. "Robust text-independent speaker identification using Gaussian mixture speaker models." In: *IEEE Transactions on Speech and Audio Processing* 3.1 (1995), pp. 72–83.

[49] Y. Stylianou, O. Cappe, and E. Moulines. "Continuous probabilistic transform for voice conversion." In: *IEEE Transactions on Speech and Audio Processing* 6.2 (1998), pp. 131–142.

[50] V. R. Apsingekar and P. L. De Leon. "Speaker Model Clustering for Efficient Speaker Identification in Large Population Applications." In: *IEEE Transactions on Audio, Speech, and Language Processing* 17.4 (2009), pp. 848–853.

[51] R. McAulay and T. Quatieri. "Speech analysis/Synthesis based on a sinusoidal representation." In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34.4 (1986), pp. 744–754.

[52] Y. Agiomyrgiannakis and O. Rosec. "ARX-LF-based source-filter methods for voice modification and transformation." In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2009, pp. 3589–3592.

[53] J. Laroche, Y. Stylianou, and E. Moulines. "HNS: Speech modification based on a harmonic+noise model." In: *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 2. 1993, 550–553 vol.2.

[54] Y. Stylianou. "Decomposition of speech signals into a deterministic and a stochastic part." In: Nov. 1996, 1213–1216 vol.2.

[55] A. Camacho and J. Harris. "A sawtooth waveform inspired pitch estimator for speech and music." In: *The Journal of the Acoustical Society of America* 124 (Oct. 2008), pp. 1638–52.

[56] S. Raschka, J. Patterson, and C. Nolet. "Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence." In: *Information* 11 (Apr. 2020), p. 193.

[57] B. Weber. *MATLAB vs Python: Why and How to Make the Switch*. URL: https://realpython.com/matlab-vs-python/.

[58] J. Lee D.and In and S. Lee. "Standard deviation and standard error of the mean." In: *Korean journal of anesthesiology* 68 (June 2015), pp. 220–3.

[59] N. Koldunov. *How to make your python code run faster*. Feb. 2015. URL: http://earthpy.org/speed.html (visited on 12/16/2021).

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of January 22, 2022 (`classicthesis` version 1.0).