UNIVERSITE LIBRE DE BRUXELLES

Faculté des Sciences Appliquées Année académique 2010 - 2011

A spectral estimator of vocal jitter

Promoteur : J. Schoentgen
Co-promoteur : F. Grenez

Mémoire de fin d'études soumis par Pol Mas Soro en vue de l'obtention du titre d'Ingénieur civil Biomédical

Acknowledgements

This thesis would not have been possible without the involvement of many people who have helped me either directly or indirectly.

Foremost, I would like to thank my supervisors Jean Schoentgen and Francis Grenez for their guidance and encouragement during this work in addition to their many suggestions and their constant support during my research. I am very grateful for all their advice and for all the time they have spent helping me. Thanks for all the discussions that contributed in many insights in this thesis. I would also like to thank Ali Alpan for his collaboration as Matlab technical support and generally speaking the ULB.

Thanks to my family, especially to my father, my brother and Alessia for accompanying me all the time and helping me so much.

Last, but certainly not least, my friends for their undivided aid and support. I would like to mention my friends in Sant Cugat: Xavi, David, Albert, Joan, Enric, Marc, Sofia, Nanda... and in Brussels: Edu, Lara, Colette, Carlota, Oscar, Laura, Laia, Montse, Celia…

Abstract

The purpose of this thesis is to study and implement a spectral method for short-time jitter estimation. Jitter consists in rapid perturbations of the vocal cycle lengths, which can be observed from one cycle to the next when they are sampled, at least, at the rate of the fundamental frequency. Jitter is analyzed for voice quality assessment given that it provides a high correlation with voice disorders.

The method is based on a mathematical model that describes the association of two periodical spike trains. Jitter is modeled as the perturbation of one of those impulse trains with respect to the other. The proposed method computes this perturbation, indirectly, by taking into account spectral properties. By counting the number of crossings between the harmonic and the inter-harmonic contours the perturbation in samples can be obtained. A Matlab application is implemented to ascertain the validity and reliability.

The Praat software is used as a reference for the assessment of the jitter values. Given the references provided by Praat, comparison is made with spectral jitter measurements in different situations. Experiments with ideally perturbed spike trains show that the suggested method produces accurate local estimations of jitter. Additional evaluation relies on testing and analyzing synthetic phonation and connected speech.

A performance appraisal allows us to enhance the method, i.e., to try for a better implementation in order to have more accurate estimates. The results are presented the reliability of the spectral jitter estimator is analyzed.

Acronyms

SJE	short-time Spectral Jitter Estimator
UP	Unit Pulses
SP	Synthetic Phonation
CS	Connected Speech

Contents

List of figures

List of tables

1. Introduction

The voice is a feature that makes each person unique and which plays important roles in daily living. It provides the means to communicate with other people. Whatever benefits the voice provides for an individual, it can be disheartening when a disorder of any kind affects the voice and consequently, one's quality of life.

People may use different types of phonation in different daily situations, either consciously or unconsciously. For instance, whisper when telling a secret, breathy voice when excited, creaky voice during a hangover. Phonation refers to the generation of sound at the larynx via an airstream, including voice; and voice is the sound generated by means of the vibration of vocal folds. By opposing each other with different degrees of tension, the vocal folds create an aperture of varying size and contour [1]. This aperture creates resistance to the stream of air generating laryngeal sound waves with characteristic pitch and intensity.

Voice disorders cause a noticeable alteration of that sound owing to a medical condition. Speech pathologists usually make a distinction between

- Articulation disorders
- Voice disorders (problems with phonation)

Voice quality assessment has received much attention. The medical community sometimes uses subjective techniques for the detection and the diagnostic of voice pathologies; for instance specialists evaluate the voice quality auditorily. The diagnosis of a voice disorder requires a patient history to obtain details about the vocal abnormality. People may frequently use the term hoarseness to describe a vocal impairment [2]. Nevertheless, it is a general term that encompasses a variety of more specific voice disorders such as strained voice, tremor or change in pitch. The purpose of this work is to study one of the best known phenomena in the context of voice disorders, namely jitter.

Jitter is an important characteristic with regard to voice quality assessment. It occurs during voice production and it is defined as small and rapid variations in glottal cycle lengths [3]. Specifically, jitter is a measure of the cycle-to-cycle fluctuations of vocal cycle lengths, which has been mainly used for the description of pathological voices. In terms of signal processing though, jitter is a form of modulation noise typically <3%. Since it characterizes some aspects concerning particular voices, it is a priori expected to find differences in the values of jitter among speakers. Jitter has been reported to become larger in the presence of laryngeal pathologies hence, a higher degree of jitter is observed in roughness or hoarseness.

A deviation from cyclicity is observed either temporally or spectrally. The exact causes of vocal jitter are unknown but, e.g., neurological reasons or turbulent airflow through the glottis may produce it. Even in modal voices, where the perturbation is lower than 1%, jitter is auditorily perceived as well as spectrally observed. Given that jitter is frequency modulation noise, the spectral properties can give us a different insight from the time-domain. In this work, those spectral characteristics are therefore studied seeking to estimate jitter spectrally.

1.1. Motivation

Jitter is difficult to measure as in voice production there is a random shift in every period. This unpredictability is partially due to the pseudo-periodic character of speech, even for sustained vowels. A strictly periodic speech signal, however, would have strong frequency components at the integer multiples of the fundamental frequency referred to as harmonics. Inter-harmonics appear when the speech signal is not strictly periodic in between the harmonics. Spectral descriptions of cyclic perturbations have been based on the separation of the harmonics or the inter-harmonics from the rest of the spectrum. Specifically, differently perturbed signals differ in their combination of the harmonic and the inter-harmonic components as well as their focus on each spectral interval. Since the effects produced by jitter are directly correlated to the size of the harmonics and inter-harmonics, jitter may be estimated spectrally. If we consider only the lower frequencies of a voiced speech segment, this may be closer to being

predictable compared to its structure in higher frequencies where noise effects degrade the spectral response. Although it seems difficult, it is worth analyzing the spectral characteristics of synthetic and actual speech thoroughly with the purpose of estimating jitter spectrally.

One known difficulty is the existence of a large variety of published experimental procedures. The diversity is indeed impressive and as a consequence, one may experience difficulties comparing results obtained in different frameworks. Based on the definition of jitter, many analysis methods have been proposed for the computation of the aperiodicity in the voice signal. The most common methods are time-domain and are based on the estimation of a sequence of pitch period values over a length of time that comprises several periods. This sequence is then used to produce an average value of jitter over the duration of a given number of periods. If N is the total number of periods and u(n) is the jittered sequence, the definitions of widely accepted jitter measurements are given below [4].

- Local jitter is the cycle-to-cycle variability of pitch in N periods (%):

$$100 \times \frac{(1/(N-1)) \sum_{n=1}^{N-1} |u(n+1) - u(n)|}{1/N \sum_{n=1}^{N} u(n)}$$

- Absolute jitter is the cycle-to-cycle variability of pitch in N periods given in time units:

$$\frac{1}{(N-1)} \sum_{n=1}^{N-1} |u(n+1) - u(n)|$$

Since time-domain methods are based on pitch period estimation, they are vulnerable to error in this estimation. Therefore, it is not unusual for the same speech fragment to obtain different jitter estimations when different estimators of the pitch are used. Measurements are consequently quite vulnerable to the variability of pitch detectors. Although the spectral jitter estimator (SJE) uses pitch period information, the spectral behavior of the harmonic and the inter-harmonic contours doesn't depend on the

estimation of the fundamental frequency. Hence, a spectral method for the computation of jitter may be more reliable.

The previous pitch period based methods for measuring jitter can be considered to model the jitter effect using solely its temporal properties. In this work, a frequency-domain method for the estimation of jitter is studied, which is based on a mathematical description of the time-domain properties. Specifically, it is assumed that jitter can be described as the combination of two periodic impulse trains simulating an ideal pulsatile glottal airflow. This modeling of jitter as a cyclic process identifies the perturbation quantitatively as the shift of one of the two periodic spike trains with respect to the other. By inspecting this model in the frequency-domain, it is shown that jitter leads to an identifiable spectral pattern. The spectral characteristics can be used then to, indirectly, obtain jitter estimates by counting the number of intersections between the harmonic and the inter-harmonic contours. The main problem, however, is that speech signals are not spike trains and consequently, the spectral behavior is not as predictable as that of a pulse sequence.

In this work, the estimation of jitter is carried out in the short term instead of calculating an average value for jitter. By producing a sequence of local jitter values on small intervals, more precise results can be obtained without assuming long-term periodicity as in the purely time-domain approaches. Having such a sequence of local jitter measurements may provide better insight in the evolving behavior of jitter. One Matlab application has indeed been implemented for short-time jitter estimation.

The abovementioned mathematical model as well as the implementation for the computation of jitter was developed in a master thesis carried out by Miltiadis Vasilakis and supervised by Yannis Stylianou [5]. Our own version Matlab application as well as a combination of both are developed to ascertain the validity of the method as well as to try for more accurate jitter estimates [6]. The analysis is provided for three different signals:

- **Unit pulse sequence.** The aforesaid unit impulse train based on the mathematical model.
- **Synthetic phonation.** Recordings of synthetic sounds /a/, /i/, /u/, /ia/, /ai/ with average fundamental frequency of 100, 120 and 140 Hz along with a highly perturbed corpus.
- **Actual speech.** Running speech of teacher's voice in seven different days pre and post-lesson.

Praat is used as reference system assuming that it provides reliable jitter estimates. The performance of these methods is appraised by comparing with Praat's estimates. Finally, general conclusion of the developed spectral estimator of vocal jitter is presented taking into account the performance.

1.2. Outline

The contents of this work are organized as follows:

- **Chapter 2 - Spectral jitter estimator.** In this chapter the mathematical model for spectral jitter estimation is shown as well as its properties in time and frequency domains are presented. In addition, the SJE algorithm is given in pseudocode.
- **Chapter 3 - Ideal model**. The way how the initial version of the SJE is developed and some important factors involved are described in this chapter. It is implemented for the ideal pulse sequence. All the analysis of each important feature is also studied.
- **Chapter 4 - SJE implementation**. In this chapter, three different SJE implementations are presented. Two of them are developed and the other one is the SJE implemented by Vasilakis. Some improvements are studied to try for better performance. For instance, the LPC-analysis is used to approach better the ideal situation. Finally those implementations are compared.

- **Chapter 5 - Test and results**. In this chapter the SJE is tested on the ideal impulse sequence, synthetic sounds and connected speech. The outcome provides an overview of the method through numerical data as well as illustrative figures.
- **Chapter 6 - Conclusions**. The SJE is assessed for the three aforementioned signals. The validity of the SJE estimator is analyzed taking into account the aforementioned results. Finally, our point of view and a general conclusion are presented.

2. Spectral jitter estimator

In this chapter, a mathematical model simulating an ideal pulsatile glottal airflow as the association of two periodic spike trains is presented. The frequency-domain analysis is presented taking into account the spectral properties of a perturbed impulse train. The proposed model relies on an identifiable spectral pattern which enables obtaining jitter measurements by counting the number of crossings between the harmonic and the inter-harmonic contours. Local perturbation can be obtained just by counting the number of crossings over frame of a few periods. This is then used for short-time estimation of frequency jitter using a Matlab application that is developed for this purpose. In this chapter, only its pseudocode is presented to give an overview. Additional functional factors are studied later.

2.1. Mathematical model

Jitter may be simulated as a perturbation of a periodic spike train, although the glottal airflow is actually more complex. In the presence of jitter, glottal pulses are ideally modeled as an additive combination of an unperturbed spike train and a shifted spike train [7].

$$g[\mathrm{n}] = \sum_{n=-\infty}^{+\infty} \delta[\mathrm{n} - (2\mathrm{k})\mathrm{P}] + \sum_{n=-\infty}^{+\infty} \delta[\mathrm{n} + \mathcal{E} - (2\mathrm{k}+1)\mathrm{P}],$$

where P is the period and Ɛ is the perturbation, both in samples. As shown in figure 1, the model describes two periodic pulse trains with Ɛ the shift imitating jitter. Notice that Ɛ can vary from 0, when jitter is absent, to P, when the frequency is halved due to the overlap of the two periodic spike trains.

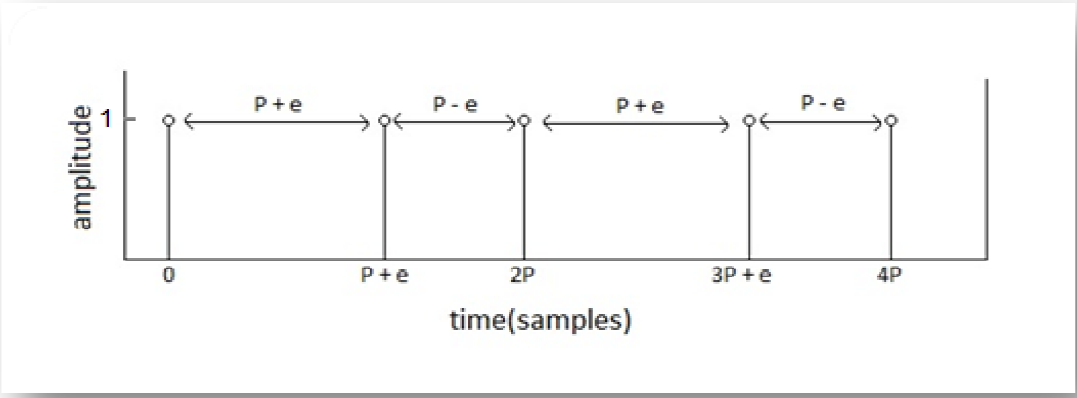

Figure 1. Glottal impulse train of the proposed model for jitter.

The Fourier transform of the cyclically jittered impulse train is given below:

$$G(\omega) = (1 + e^{-j\omega(P-\mathcal{E})}) \sum_{k=-\infty}^{+\infty} \frac{\omega_o}{2} \delta(\omega - k\frac{\omega_o}{2})$$

where ω_o=2π/P is the angular velocity in rads. The square of the magnitude spectrum is:

$$|G(\omega)|^2 = \frac{\omega_o^2}{2} \sum_{k=-\infty}^{+\infty} (1 + \cos[(P - \mathcal{E})k\frac{\omega_o}{2}])\ \delta(\omega - k\frac{\omega_o}{2})$$

The cosine term in the sum can be transformed as follows:

$$1 + \cos\left[(P - \mathcal{E})k\frac{\omega_o}{2}\right] = 1 + \cos(k\pi)\cos(k\frac{\mathcal{E}}{P}\pi)$$

The main period is 2π/P (rad) while the deviation period is 2π/Ɛ (rad). Because both cosine signals have no phase deviation, crossings of the harmonic and the inter-harmonic contours take place at frequencies:

$$(\mathbf{1})\ \omega_k = (k + \frac{1}{2})\frac{\pi}{\mathcal{E}}$$

with $\omega_k \leq \pi$. The log magnitude spectrum can be shown to be:

$$20log_{10}|G(\omega)| = 10log_{10}\left(\frac{\omega_o^2}{2}(1 + \cos\left[(P - \mathcal{E})k\frac{\omega_o}{2}\right])\right)$$

$$\left[\sum_{l=-\infty}^{+\infty}\delta(\omega - l\omega_o) + \sum_{l=-\infty}^{+\infty}\delta(\omega - (l + \frac{1}{2})\omega_o)\right].$$

This last part can be written as:

$$20log_{10}|G(\omega)| = H(\mathcal{E}, \omega) + S(\mathcal{E}, \omega),$$

where

$$H(\mathcal{E}, \omega) = 10log_{10}\left(\frac{\omega_o^2}{2}(1 + \cos[(P - \mathcal{E})\omega]\right)\sum_{l=-\infty}^{+\infty}\delta(\omega - l\omega_o)$$

$$H(\mathcal{E}, \omega) = 10log_{10}\left(\frac{\omega_o^2}{2}(1 + \cos[(P - \mathcal{E})l\omega_o]\right), l \in N$$

is the harmonic part of the log magnitude spectrum, while

$$S(\mathcal{E}, \omega) = 10log_{10}\left(\frac{\omega_o^2}{2}(1 + \cos\left[(P - \mathcal{E})k\frac{\omega_o}{2}\right]\right)\sum_{l=-\infty}^{+\infty}\delta\left(\omega - (l + \frac{1}{2})\omega\right)$$

$$S(\mathcal{E}, \omega) = 10log_{10}\left(\frac{\omega_o^2}{2}(1 + \cos\left[(P - \mathcal{E})(l + \frac{1}{2})\omega_o\right]\right), l \in N$$

is the inter-harmonic part of the log magnitude spectrum that appears due to jitter. It is feasible to obtain the harmonics and the inter-harmonics knowing the frequency-domain position provided by the Fourier Transform of the aforesaid jittered signal. Namely, $l\omega_o$

for the harmonics, and $(l+\frac{1}{2})\omega_o$ for the inter-harmonics with $\omega_o = 2\pi/P$. Notice that when there is no perturbation, i.e. Ɛ=0, the inter-harmonics disappear.

$$S(0,\omega) = 10log_{10}(0) = \nexists$$

Examples of harmonic and inter-harmonic contours for different values of Ɛ are depicted in figure 2.

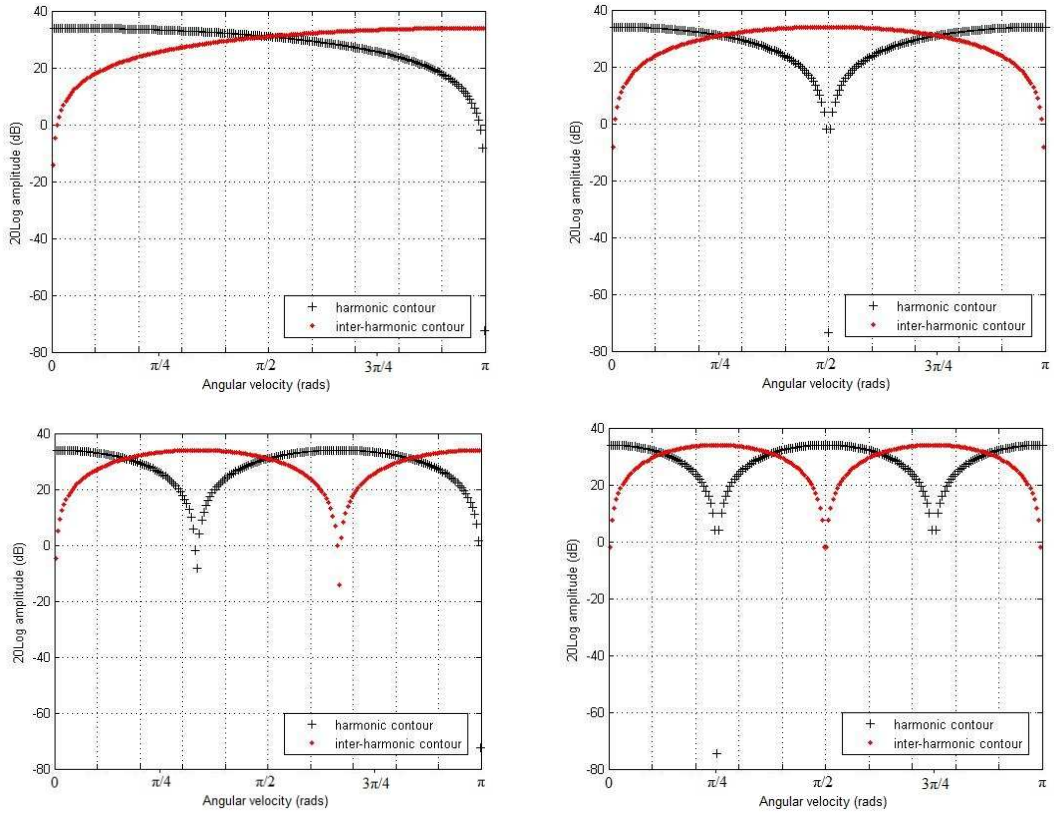

Figure 2. Log magnitude spectra of the harmonic and inter-harmonic contours for Ɛ=1, 2, 3 and 4 respectively.

The harmonic and the inter-harmonic contour follow a pattern owing to the association of two spike trains, with Ɛ equal to the number of crossovers between both contours. The mathematical model predicts where the crossings are located in the ideal situation; nevertheless the prediction is actually approximate as a consequence of, for instance, quantization noise. In real speech, other possible degradation effects like additive noise may also shift the exact position of the intersections and thus, they have to be considered if we want to obtain the harmonics and the inter-harmonics correctly.

For example, for $\varepsilon = 2$ the theoretical positions of the crossings are at $\pi/4$ and $3\pi/4$ whereas for $\varepsilon = 4$ the crossings are at $\pi/8$, $3\pi/8$, $5/8$, and $7\pi/8$. Notice that the locations of the intersections in the spectrum only depend on ε and not on the period of the signal. The spectral behavior of the harmonic and the inter-harmonic contours doesn't rely on the estimation of the fundamental frequency. Accordingly, pitch estimation error may be avoided, at least, in this ideal situation.

Generally speaking, just by counting the number of intersections between the harmonic and inter-harmonic contours, an estimation of ε and thus of jitter can be obtained. The mathematical model given is only an ideal representation of the glottal airflow rate, however sustained vowels or connected speech are far more complex signals. Therefore, we will have to analyze other factors involved in voice production if we seek to obtain accurate jitter estimates.

Thus far, we have assumed that the two periodic spike trains have infinite duration. In real speech signals, however, jitter varies from one cycle to the other; hence local estimations are requested when actual speech is analyzed. In the next section, a short-time jitter estimator is developed based on the mathematical model.

2.2. Short-time jitter estimator

Local jitter estimation involves a short signal fragment, namely a few cycles, to obtain the local perturbation. The procedure relies on a sliding frame used to examine the analyzed signal cycle by cycle. The size and the step of the sliding frame are given by the variables L and S, which represent multiples of the period. The step choice is one period because we want to estimate the local jitter variability in every single period. Regarding the frame size, to measure jitter two periods are needed, at least, to obtain the cycle-to-cycle variability. The frame length is, however, typically between 3 and 4 periods given that we seek to obtain short-time estimates of jitter. The most suitable window size is analyzed later taking into account other factors such as the input signal or the window shape.

A threshold criterion is used to determine whether a contour crossing has occurred. When an intersection between the harmonic and the inter-harmonic contours is observed, the threshold criterion is applied. This threshold is kept constant for all the experiments with a value of 3dB. The initial threshold criterion is explained in section 3.6. In synthetic phonation and actual speech, spectral characteristics are used to develop a more complex intersections detector. However, that method is explained later in section 4.1, where the Vasiliakis implementation is studied.

The obtained short-time jitter sequence consists of integer values corresponding to the cycle perturbation in number of samples. Within the framework of the previous model, the number of intersections give a value equal to $2\times\hat{e}_q$, where $\hat{e}_q$ is the quantized jitter estimate, given in samples in half the spectrum [7]. Consequently, the estimation in time units is

$$(\mathbf{2})\ \hat{e}(F_s) = \hat{e}_q \frac{2 \times 10^6}{F_s} (\mu s)$$

where Fs is the sampling frequency in Hz. We have to take into account that there is an error due to quantization when the number of crossings is converted to jitter in µs. For instance, given a sampling frequency of 16 kHz the quantum is 125 µs whereas for 44 kHz it is 45 µs. The estimation in each frame is that of k*quantum and consequently, by increasing the sampling frequency we reduce the quantization step. A higher sampling frequency may therefore improve the accuracy of the estimation. Regarding our method, a high resolution is needed to reduce quantization error, i.e. to have the same order of magnitude as the lowest jitter value we expect to measure.

The algorithm for short-time jitter estimation is presented in pseudocode in table 1 whereas the Matlab code is given in the appendix.

```
jitter = []
time = []
index = ()
start = ()
pitch = []
pitch = pitch period sequence estimation(s[n])
while start < N
        P = average(pitch) or P=P[index]
        end = start + L*P - 1
        frame = s[start : end]
        frame = window(frame)
        F = 20log10( |FFT(frame)|)
        H = F[harmonic frequencies]
        S = F[inter-harmonic frequencies]
        candidateJitter =intersections which satisfy the threshold criterion
        jitter[index] = valid intersections among candidateJitter
        time[index] = (start + end) / 2
        start = start + S*P
        index = index +1
endwhile
return jitter, time
```

Table 1. SJE given in pseudocode.

The average value of absolute jitter Δ_j is later used for comparison purposes between different implementations of the spectral jitter estimator. If j(n) is the absolute jitter sequence with length N, then the average jitter for the whole sequence is computed in the following way:

$$(\mathbf{3})\ \Delta_j = \frac{\sum_{n=1}^{N} j(n)}{N}\ (\mu s).$$

2.3. Windowing

At this stage, we analyze the spectral consequences of waveform windowing. Regarding our spectral estimation of jitter, the window shape is very important to obtain an accurate estimation of the harmonic and inter-harmonic samples. Windowing with a suitable shape will aid discerning between overlapping harmonic peaks. Therefore, the characteristics of some of the popular windows are worth studying. A brief explanation

is presented in this section, seeking to understand better the windowing consequences on the spectrum.

The most common window shapes are the following:

- **Rectangular**. The rectangular window is the simplest one, taking a piece of the signal without any other modification. Notice that it inserts signal discontinuities at the endpoints as depicted in figure 3.

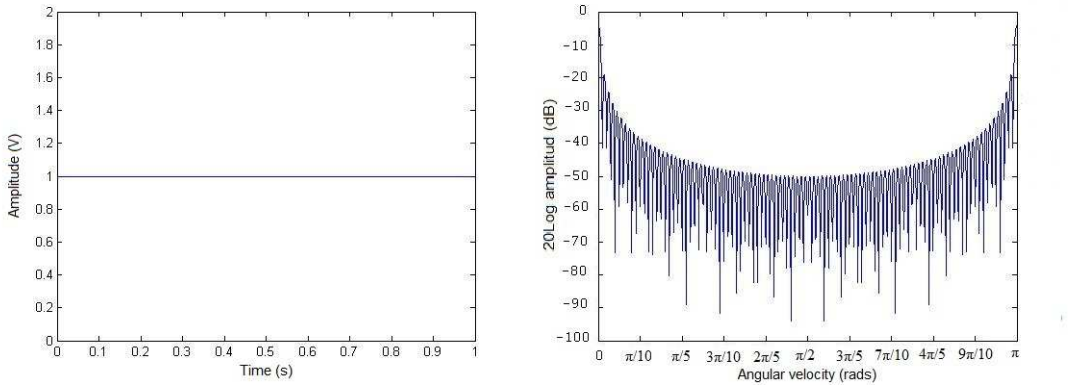

Figure 3. Time-domain and frequency-domain representations of a rectangular window.

- **Triangular**. It has triangular shape corresponding to a Bartlett window with zero-valued end-points as shown in figure 4.

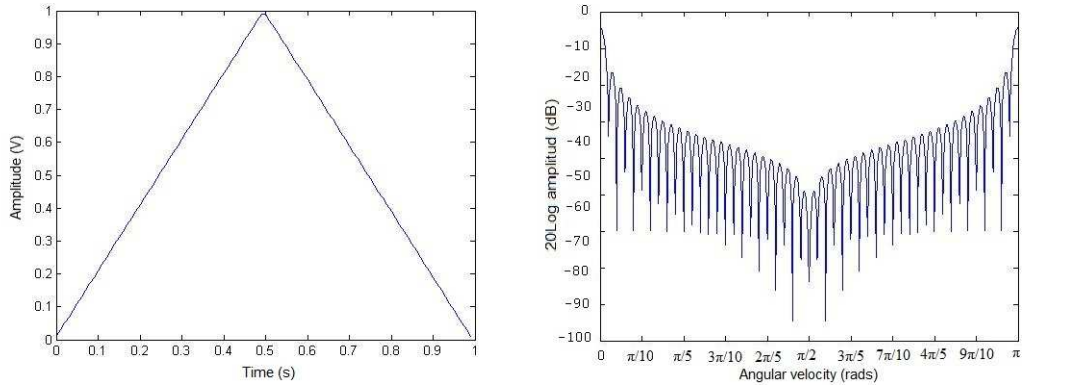

Figure 4. Time-domain and frequency-domain representations of a triangular window.

- **Hanning**. It has the shape of one cycle of a cosine wave with 1 added to it so that it is always positive. See figure 5.

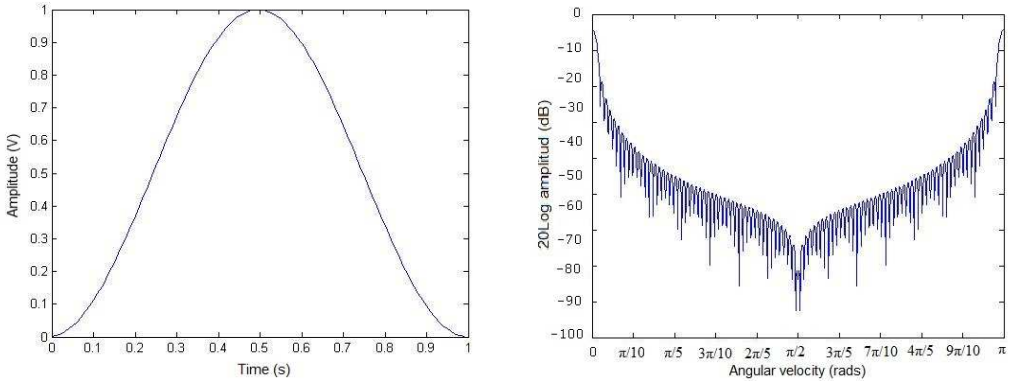

Figure 5. Time-domain and frequency-domain representations of a Hanning window.

- **Hamming**. It has a raised cosine shape as shown in figure 6.

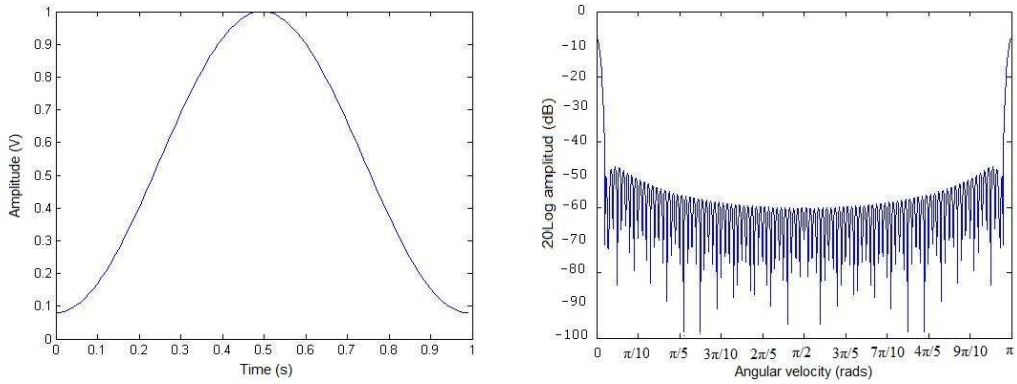

Figure 6. Time-domain and frequency-domain representations of a Hamming window.

The side-lobe roll-off is different depending on the window shape. It is easily perceived in figure 7, where the spectrum of a windowed spike train is depicted. Notice how the side-lobes are reduced when a Hamming window is used instead of a rectangular one.

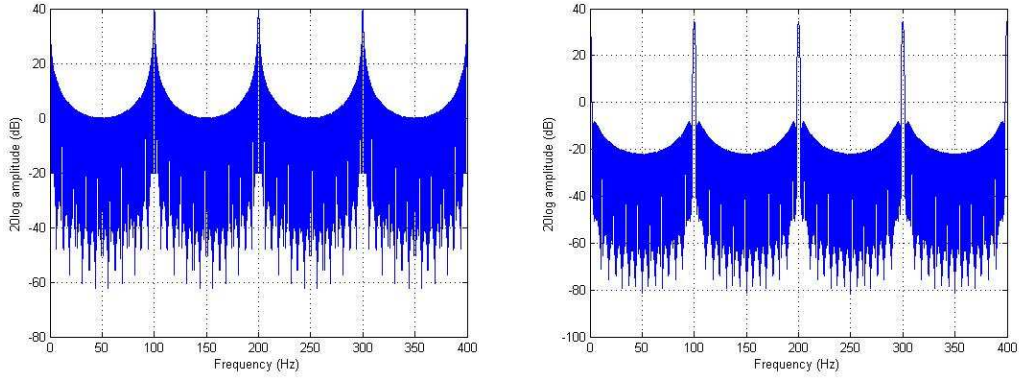

Figure 7. Log magnitude spectra of a windowed spike train using rectangular and Hamming shapes.

An overview of our spectral jitter estimator has already been presented. The next step is therefore to develop a spectral estimator of vocal jitter.

3. SJE Version I

In this chapter, the spectral jitter estimator is developed in Matlab as we seek to obtain a tractable application, which can be easily modified. The mathematical model is tested by implementing the assumed ideal glottal airflow, i.e. two periodic impulse trains that follow the temporal pattern previously established. As a consequence of the technical feasibility of the analysis of pulse sequences we study the spectral characteristics, which are used to estimate jitter.

The chapter starts analyzing the easiest conceivable signal, i.e. an unperturbed spike train, which simulates the absence of jitter. Progressively, other factors are involved till all the functions used to spectrally estimate jitter have been studied. The reliability in the ideal situation is assessed taking into account the spectral behavior of pulse sequences in different situations. Generally speaking, it is essential, before we start to face other difficulties concerning more complex signals, to perfectly understand this simplified model.

3.1. Unperturbed spike train

The purpose of studying an unperturbed spike train is to analyze a signal simulating an unperturbed sustained sound. The spectral characteristics provide the necessary information to determine the harmonic locations, which can be easily observed in the theoretical Fourier transform of an unperturbed impulse train.

$$\sum_{k=-\infty}^{\infty} \delta(t - kT_o) = F_o \sum_{k=-\infty}^{\infty} \delta(f - kF_o)$$

where $F_o = 2\pi/T_o$. Given the position of the temporal peaks, we can identify the frequency-domain harmonic location, which are provided by the formula shown above.

Notice that the Fourier transform of an unperturbed spike train is another impulse train whenever an infinite duration is given. This ideal situation is not reliable when a subset of cycles is analyzed since windowing effects degrade the spectral representation. Because the proposed method is based on short-time estimation, only a few periods of the signal are windowed. Therefore, effects such as the appearance of the side-lobes due to spectral leakage have to be considered.

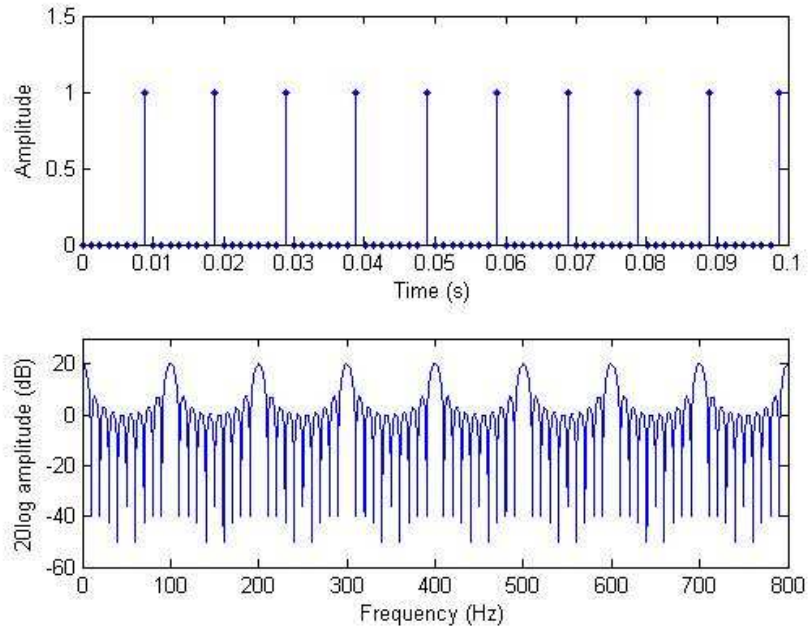

Figure 8. Time-domain and frequency-domain representations of a windowed spike train.

By decreasing the window length, the side-lobes increase and thus it is worth considering an appropriate window size as well as the most suitable shape. An adequate window shape facilitates discerning between similar frequencies and it is therefore easier to obtain the harmonics and the inter-harmonics. Notice the appearance of the side-lobes after windowing with a rectangular window shape in figure 8, where a spike train becomes a *sinc* train due to spectral leakage. The Matlab code corresponding to an unperturbed spike train is presented in appendix 1.

3.2. Perturbed spike train

The next step is to perform a cyclic perturbation with pitch deviation of a constant value. Two periodic unit spike trains are created following the aforementioned model depicted in figure 1. Knowing the sampling frequency and period, an array of ones and zeros following the mathematical model can be easily created. The code corresponding to the assumed ideal glottal airflow is presented in appendix 2. A randomly perturbed spike train is, however, more realistic knowing that we want to utilize our application with synthetic and real speech sounds. Jitter varies from one cycle to the next; therefore it is recommended to create a random shift in every single cycle as given in actual speech. The new Matlab code corresponding to the ideal glottal airflow perturbed cycle by cycle is presented in appendix 3.

Given that our spectral estimator obtains short-time jitter estimates, the result is restricted to the closest periods that have been windowed. Instead of an average over the whole signal, performed by other jitter detectors such as Praat, local jitter values can be obtained. The mathematical model proposed the association of two periodic spike trains that we have changed to a single perturbed spike train. This impulse train has a deviation on the periodicity, due to jitter, that we want to measure. Hence, we need to figure out how to calculate each local jitter value. A reasonable window size for the estimation of jitter in the short term is between three and four periods. Anyway, just by taking into account consecutive peaks in the middle of a windowed frame, the local perturbation can be obtained as shown in figure 9.

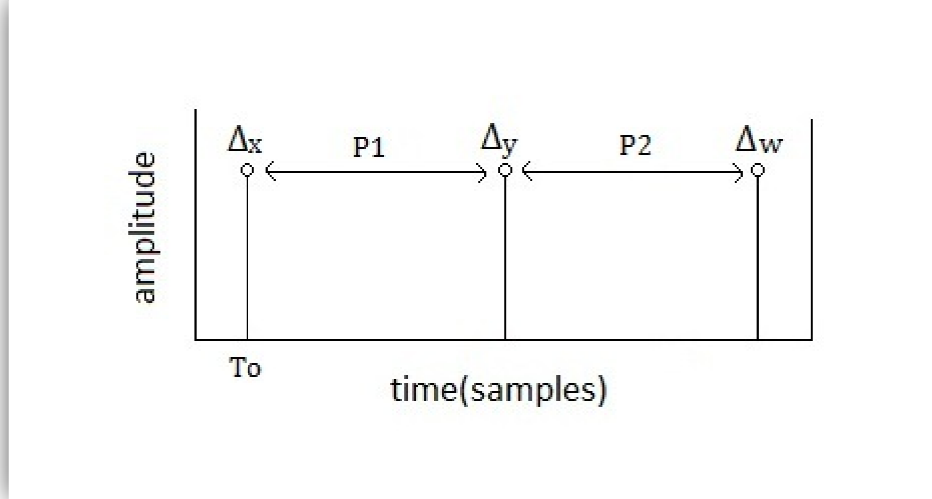

Figure 9. Neighboring pulses used to obtain local jitter.

$$P_1 = (T_o + \Delta_y) - (T_o + \Delta_x)$$

$$P_2 = (T_o + \Delta_w) - (T_o + \Delta_y)$$

$$(4)\ \Delta = P_2 - P_1 = \Delta_w + \Delta_x - 2\Delta_y$$

where Δ is the local jitter in samples and P the period also in samples. Provided that the perturbation is known, estimates for each windowed frame can be compared to the real value. Therefore, it is feasible to assess the reliability of local estimations just comparing to a reference value.

3.3. FFT

For the purpose of analyzing the spectrum of the aforementioned ideal model, a DFT (Discrete Fourier Transform) is performed. The spectral pattern is studied to ascertain whether the crossings between the harmonic and the inter-harmonic contours take place at the positions predicted by the model [8]. The energy of the signal when the FFT is applied has to be conserved. In the Matlab FFT function though, this is not the case and thus we have to apply a correction factor. The FFT performed by Matlab causes an

offset that is corrected by dividing the output by a constant value, which depends on the frame length. In figure 10, the spectrum of the ideal glottal pulse sequence is depicted for Ɛ=2.

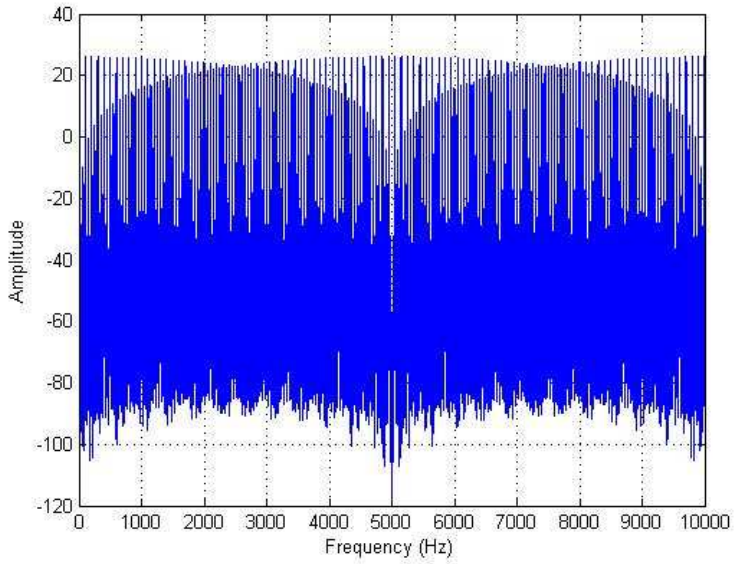

Figure 10. Frequency-domain ideal pulse sequence for Ɛ =2.

Two crossings appear to exist, but in figure 10 they are not easy to distinguish. Hence, we look at a zoom of one crossing in figure 11.

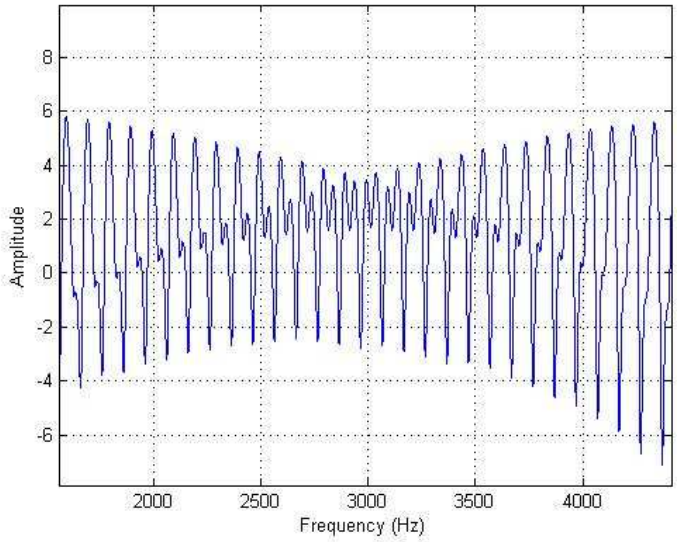

Figure 11. Zoom on the frequency-domain of the ideal pulse sequence.

Notice that when the harmonics increase the inter-harmonics decrease corroborating the model prediction. Realistic signals are more complex than spike trains and accordingly, we will have to analyze their spectrum more carefully. The likeness between the spectra of ideal glottal pulse sequence with random jitter and sustained sounds is analyzed in section 5.2, where synthetic phonation is studied.

3.4. Harmonics and Inter-harmonics detector

To count the number of intersections between spectral contours, we need to obtain the harmonics and the inter-harmonics. We want to measure the amplitude of the harmonic and the inter-harmonic peaks; accordingly the next step is to implement a peak picker. Theoretical crossing positions are provided by the mathematical model. Knowing the actual pitch period would therefore enable tracking the mentioned spectral peaks. However, spectral resolution as well as quantization noise and pitch estimation errors may spoil the estimation of crossing positions. Namely, spurious crossings may be detected if the exact amplitude values corresponding to the harmonics and inter-harmonics are not reported. It is therefore essential to implement a reliable harmonics detector by peak picking if we want to obtain accurate estimates the number of contour crossings.

First of all, we need to look for the highest peaks in close proximity to its theoretical location in the spectrum. The search is performed in a 10% interval around the known theoretical peak position. If a peak is found we accept it as valid, otherwise we keep the theoretical position. Occasionally, the harmonics or inter-harmonics are reduced so much that they disappear due to an overlap with a neighboring peak. That is the reason to keep the theoretical position when no peak is detected. The search starting position is updated from peak to peak, i.e. the preceding harmonic location is used as a reference to find the next one. The spectral peak amplitudes are cyclically shifted due to jitter and thus, it is worth updating. The implemented harmonic and inter-harmonic detector is presented in appendix 4. Additional evaluations are carried out in sustained vowels and connected speech, which are more complex and consequently other analyses are required.

The window is another important factor concerning peak detection. At the moment we do not change the window shape till we have studied all the tasks involved, i.e. a rectangular window is used here. In real speech, however, a rectangular window would never be used because of the influence of the side lobes, which degrade the peak resolution. Magnitude spectra for Ɛ =2 and 4 are depicted in figures 12 and 13 whereas harmonic and inter-harmonic spectral contours for various perturbations (Ɛ) are depicted in figures 14 and 15.

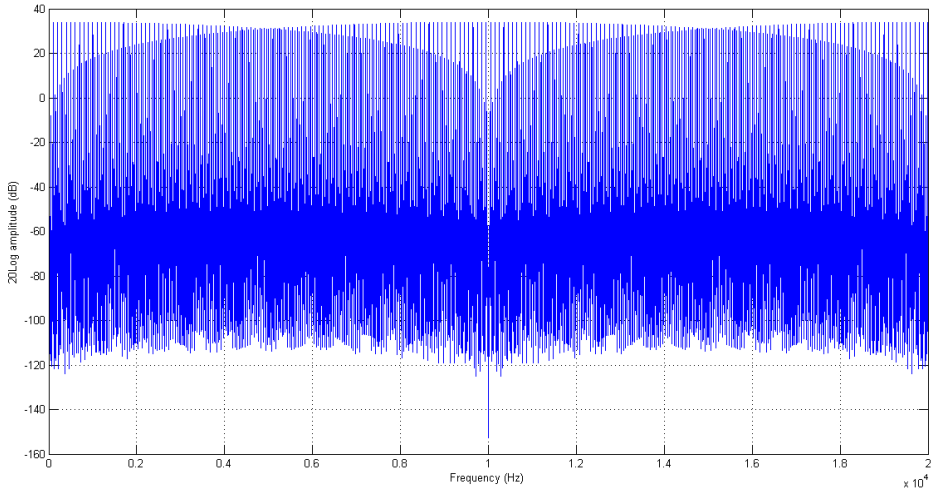

Figure 12. Log magnitude spectrum of an ideal glottal pulse sequence for Ɛ=2 .

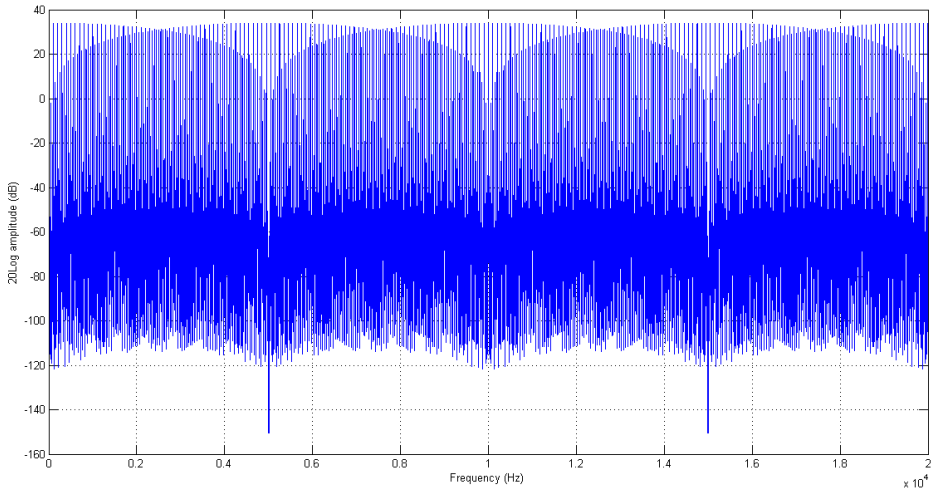

Figure 13. Log magnitude spectrum of an ideal glottal pulse sequence spectrum for Ɛ=4.

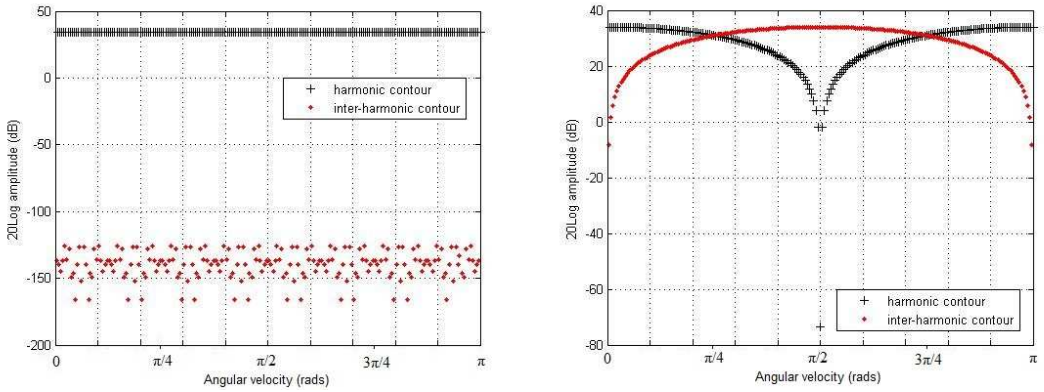

Figure 14. Log magnitude spectra of the harmonic and inter-harmonic contours for $\mathcal{E} = 0$ and $\mathcal{E} = 2$.

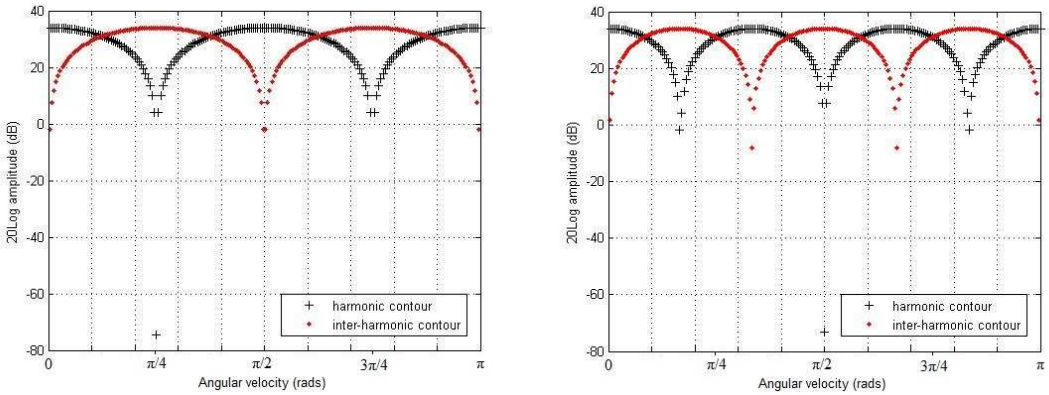

Figure 15. Log magnitude spectra of the harmonic and inter-harmonic contours for $\mathcal{E} = 4$ and $\mathcal{E} = 6$.

In the presence of jitter, the intersections between the contours can be easily observed in the frequency-domain. We have implemented a Matlab system that obtains the harmonic and inter-harmonic contours. Local jitter in samples is observed in figures 14 and 15 as the number of contour crossings between both contours. The intersections detector is, however, presented in section 3.7. Before that, in the subsequent section we take into account additive noise.

3.5. Additive white Gaussian noise

Additive white Gaussian noise is a channel model in which the only impairment to communication is a linear addition of wideband or white noise. Furthermore it has a constant spectral density as well as a Gaussian distribution of amplitude. It produces models which are useful for having a general insight concerning the behavior of a noisy

system. Therefore, adding Gaussian white noise to the ideal glottal pulse sequence allows ascertaining the reliability of the jitter estimates, when noise degrades the input signal.

In figures 16 and 17 the noise power increases whereas the signal power is constant. While the SNR (Signal to Noise Rejection) is getting smaller the peak picker detector response is getting inexact. The spectral peak amplitudes are cyclically shifted due to noise and accordingly, the harmonic and inter-harmonic contours measured are more inaccurate as depicted in the mentioned figures.

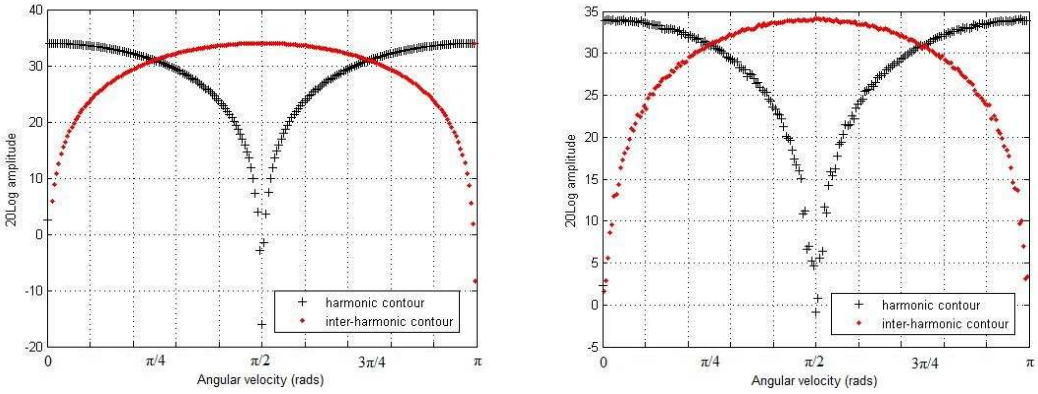

Figure 16. Log magnitude spectra of the harmonic and inter-harmonic contours for SNR=40 dB and SNR=20 dB, respectively.

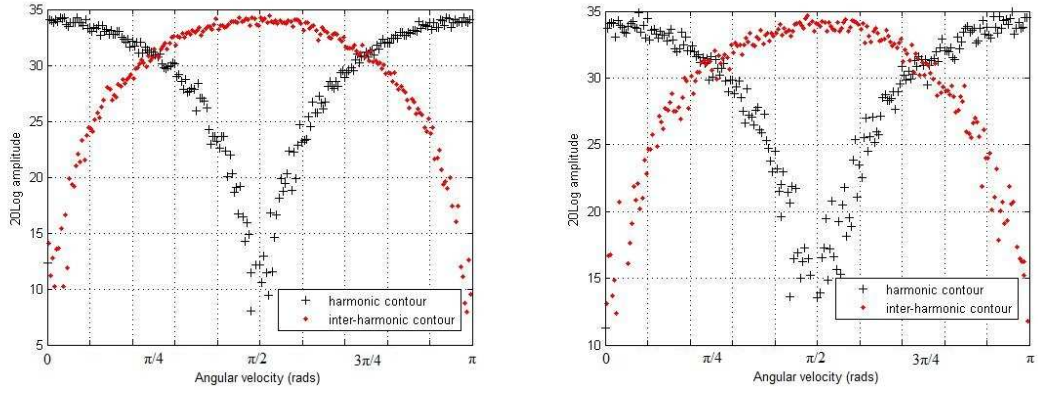

Figure 17. Log magnitude spectra of the harmonic and inter-harmonic contours for SNR=10 dB and SNR=5 dB, respectively.

Noise degrades the input signal and consequently, that of the frequency-domain representation. Since noise has a large influence on the peak picker it is worth finding a

way to reduce the noise effects if we want to obtain the correct number of crossings. One useful way is to keep the noise plus signal peaks and set the other samples to 0.

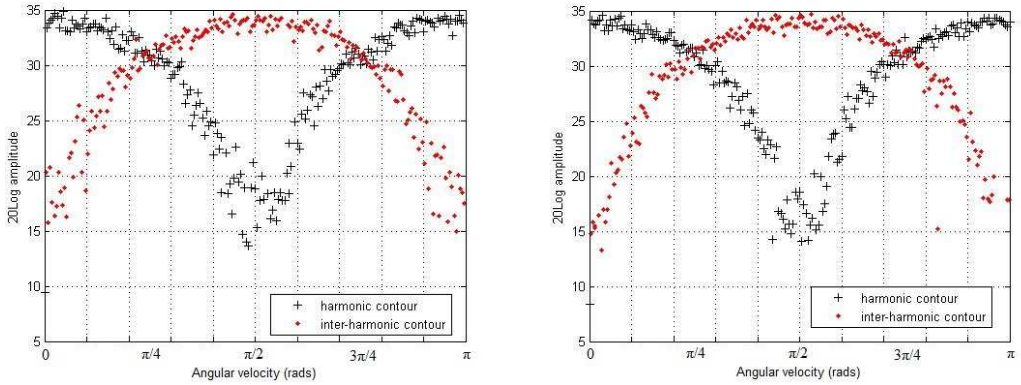

Figure 18. 20Log magnitude spectra of the harmonic and inter-harmonic contours for SNR=5 dB without and with the noise rejection method, respectively.

There is a slight reduction in the noise effects when the rejection method is used as shown in figure 18. The aim of this method is to reduce the noise energy so that the source signal is less degraded. Since this tool is helpful to reduce the noise influence on our detector, it is used whenever additive noise is not negligible. The Matlab code implementing the noisy system along with the noise rejection method is provided in appendix 5.

Although the peak picker is a good method obtaining the harmonics and inter-harmonics, it occasionally fails when noise effects are severe. The noise produces spurious peaks and masks the positions of harmonics and inter-harmonics. The following zoom of the spectrum shows a spurious peak, which is larger than the harmonic and which is therefore reported as a harmonic.

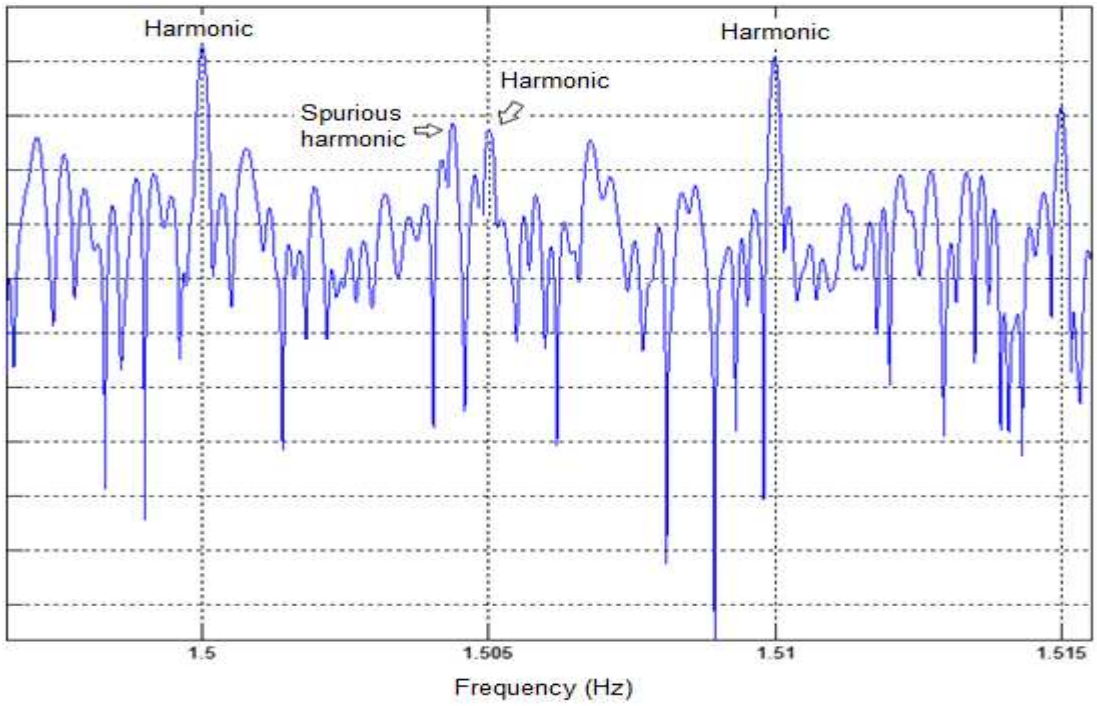

Figure 19. Spurious peak wrongly reported as harmonic.

Since a spurious harmonic is reported, the search starting position is incorrect. As a consequence of updating, the following harmonic may be wrongly reported as well. This may happen several times until the peak picker ends up detecting the inter-harmonic instead of the harmonic. An overlap between the harmonic and the inter-harmonic contours may be reported by the crossings detector as shown in figure 20.

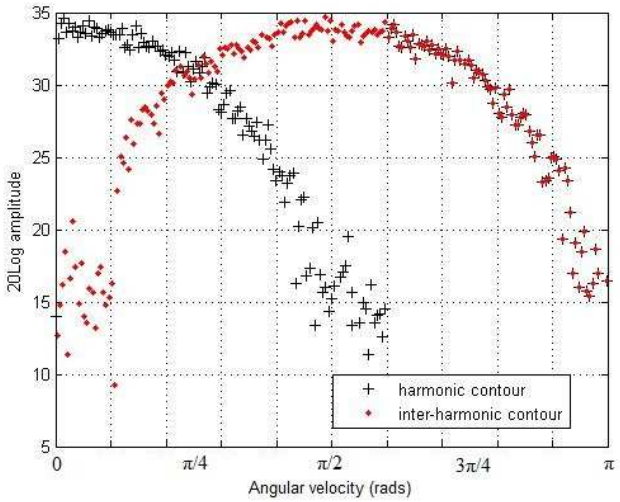

Figure 20. Overlap between the harmonic and the inter-harmonic contours wrongly reported.

Instead of a peak picker the Vasilakis implementation obtains the amplitudes corresponding to the theoretical positions of the harmonics and the inter-harmonics. Since this method is susceptible to quantization noise, they use a heuristic algorithm to obtain the number of crossings. This heuristic, which was developed in Vasilakis master thesis [5], is studied in section 4.1.2. Our intersection detector, which is simpler than that of Vasilakis, is presented in the following chapter.

3.6. Crossings detector

The last stage in the implementation is the development of a Matlab function, which counts the number of crossings corresponding to jitter in samples. If there is an intersection between the harmonic and the inter-harmonic contours, the position is reported as a candidate crossing position. A 3 dB threshold criterion performs further examination to the candidate crossings. Namely, on many occasions we have to reject some of the crossings. Namely, if the largest distance between the contours exceed 3 dB before the next candidate, the initial crossing is accepted otherwise is rejected. Some candidate crossings are rejected and others are finally accepted if they comply with the threshold criterion, as depicted in figure 21.

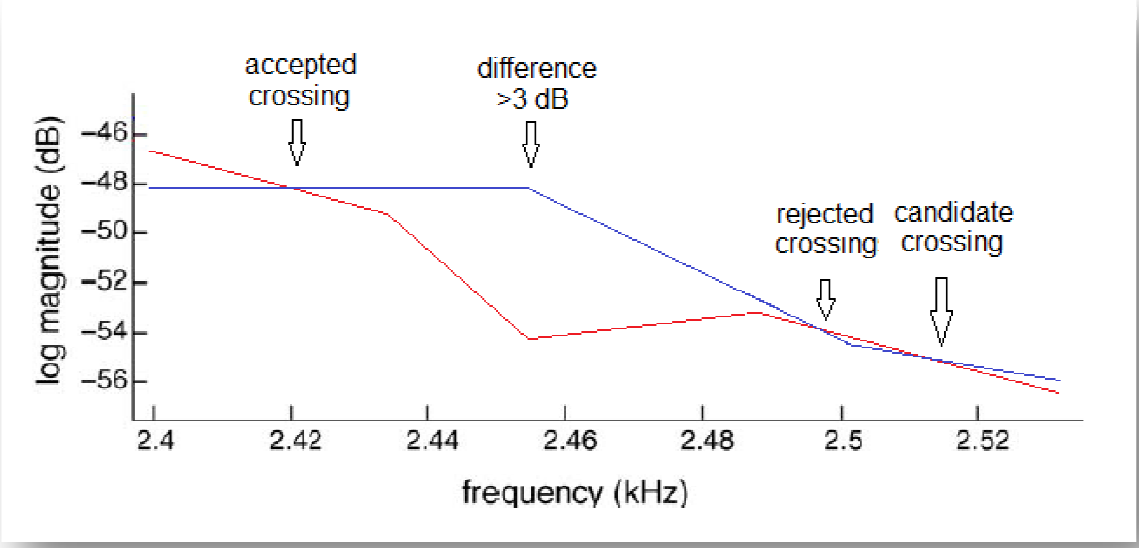

Figure 21. The 3 dB threshold criterion enables rejecting many spurious crossings.

The accepted crossings are further examined to ascertain whether they are at the expected locations or, at least, near the theoretical position provided by the mathematical model. If the candidate location is not in a 10% interval around the theoretical position the crossing is also rejected.

The intersection detector provides an estimation of the local perturbation in each windowed frame by counting the number of crossovers between the harmonic and the inter-harmonic contours. The Matlab algorithm corresponding to the explained method is presented in appendix 6.

3.7. Windowing analysis

Thus far, all the necessary functions to develop a short-time spectral jitter estimator have been implemented; however we still have to decide on the most suitable window. Regarding the ideal situation, the most appropriate window shape is the rectangular. It is the best way to overcome the alias issues between harmonics and inter-harmonics, i.e. it is the best window discerning between overlapped peaks when pulse sequences are analyzed. Furthermore, only two cycles are needed to obtain the local jitter given that the only pulses that have an influence on the estimation are the ones contiguous to the one that is in the center of the frame. When non-rectangular windows are used, like Barlett (triangular) or Hamming, the peak amplitudes at the window edges are reduced so much that they have no influence on the jitter estimate. Therefore, more than three cycles are required to locally estimate jitter when the window is not rectangular.

In real speech, to avoid spurious components to appear due to discontinuities at the frame boundaries, it is necessary to use a non-rectangular window. As far as the ideal pulse sequence is concerned, a triangular window shape is an appropriate choice. It has an adequate tradeoff between resolving comparable strength signals with similar frequencies and side-lobe roll-off. However, when non-rectangular windows are used the data at the frame boundaries is distorted. To minimize the effect of this distortion we noticed that a window length of between three and four periods provides a high enough resolution in the computed spectrum for the estimation to be successful. The applied

triangular window concentrates on the two middle cycles, providing thus the desired short-time precision to obtain local jitter estimates.

Henceforth, we designate by Version I our first SJE implementation, which is presented in appendix 7. Taking into account the technical simplicity of the computation, it is interesting to test our method with synthetic speech sounds. We have to ascertain the similarity between the spectrum of ideal pulse sequence and that of sustained sounds. All the analysis corresponding to the synthetic phonation is presented in the next chapter.

4. SJE Implementations

In the previous chapter, the ideal situation was presented as well as the development of the Version I of the spectral estimator of vocal jitter. Although it is based on the mathematical model, our version was created without taking into account the Vasilakis Matlab implementation. As a consequence, there are differences between both methods which are analyzed in this chapter.

Firstly, the most important characteristics concerning the Vasilakis estimator are studied. A comparison between both methods is also presented. Changes that are introduced are a combination between our implementation and that of Vasilakis creating thus a new version called Version II. LPC-analysis is also applied to all the jitter estimators. It is utilized to convert the input signal into another one that is more similar to the ideal pulse sequence. The performance assessment, however, takes place in chapter 5, where all the results are thoroughly analyzed.

4.1. Vasilakis SJE

The Vasilakis spectral jitter estimator introduces some variations from our initial implementation that have to be considered. Two operations are presented that are the linear interpolation, which is performed for the purpose of obtaining the harmonic and

the inter-harmonic samples, and the heuristic crossing detection. The goal of this heuristic detector is to reduce the overestimation of jitter by rejecting spurious crossings. Both operations are presented in the following sections.

4.1.1.Linear interpolation

The spectral estimation of vocal jitter relies on the number of crossings between the harmonic and the inter-harmonic contours. For that purpose, the Vasilakis implementation applies a linear interpolation to obtain inter-harmonic samples. The harmonic positions are obtained on the base of the known value of the fundamental frequency f_o.

It is known that the spectrum is sampled at the harmonic frequencies $k*f_o$ and the inter-harmonic frequencies $(k+0.5)*f_o$. In order to facilitate the search for intersections between the two contours, the Vasilakis method performs a linear interpolation between frequency pairs of $[k*f_o, (k+1)*f_o]$ to obtain the harmonic contour at frequencies $(k+0.5)*f_o$. When the two known points are given by the coordinates (x_0, y_0) and (x_1, y_1), the linear interpolant is the straight line between these points. Given a value x in the interval (x_o, x_1), the value y along the straight line is obtained from the following equation

$$\frac{y - y_o}{x - x_o} = \frac{y_1 - y_o}{x_1 - x_o}$$

which can be derived geometrically from figure 22.

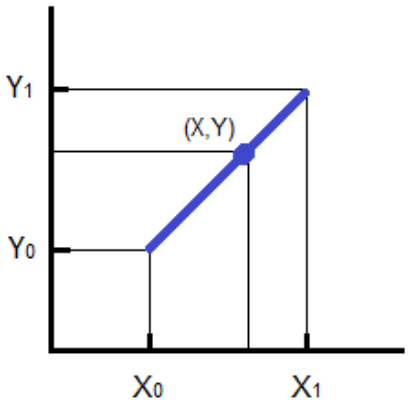

Figure 22. Linear interpolation between the two points (x_0,y_0) and (x_1,y_1).

Solving this equation for y, which is the unknown value at *x*, gives

$$y = y_o + (x - x_o)\frac{y_1 - y_o}{x_1 - x_o} = \frac{(x - x_o)y_1 + (x_1 - x)y_o}{x_1 - x_o}$$

which is the formula for linear interpolation in the interval. In the Vasilakis algorithm, the linear interpolation is used to obtain the inter-harmonic samples considering that the harmonic positions are known.

Their way to obtain the harmonics and the inter-harmonics is based on the supposition that the a priori locations are reliable. The reported positions are, however, not always that of the harmonics or inter-harmonics. An inexact estimation is obtained, which leads to an erroneous estimation of jitter as well. Hence, quantization noise as well as other imperfections in actual speech have to be considered as they produce a degradation of the spectrum. Those problems spoil the estimation of the number of contour crossings, even if a peak picker is used and mostly when noise is larger. A heuristic is therefore used to reject spurious crossings and group others. This heuristic is explained in the section that follows.

4.1.2.Heuristic crossings detector

In section 3.6 we ascertain that our crossings detector based on peak picking is appropriate in the ideal situation. Nevertheless, when one has to deal with spurious crossings a more complex method may be required. Spurious intersections between the two spectral contours may appear, especially in higher frequencies and mostly in signals with large jitter. A heuristic contour crossing detector was therefore developed by Vasilakis [5], which is explained hereafter.

- All observed intersections are considered initially as candidate crossings. Once all the candidate crossings are obtained, starting with the one of the highest possible order, the spectrum is divided in that many equal intervals. At least one crossing is expected to exist in the area around the center of each interval. If true, the candidate jitter value is accepted. Otherwise, the candidate jitter is decreased by one and the process is repeated until a valid estimation or zero is reached.
- A 3 dB threshold criterion is also used to determine whether an intersection between the harmonic and the inter-harmonic contours has occurred. The threshold is used as follows. When a candidate crossing has been detected, i.e. an intersection between both contours has occurred, the distance in dB between the contours is monitored till the next candidate crossing to the right is detected. When the largest distance between the contours exceed 3 dB the initial crossing is accepted otherwise is rejected.
- When more than one candidate crossing is observed in the area around the expected position, they are grouped. Namely, the average position of the nearby candidates is obtained, creating thus a new crossing in the hypothetical location.

In figure 23 the heuristic crossings detector is depicted. Notice that some candidate crossings are rejected, others are grouped and others are finally accepted.

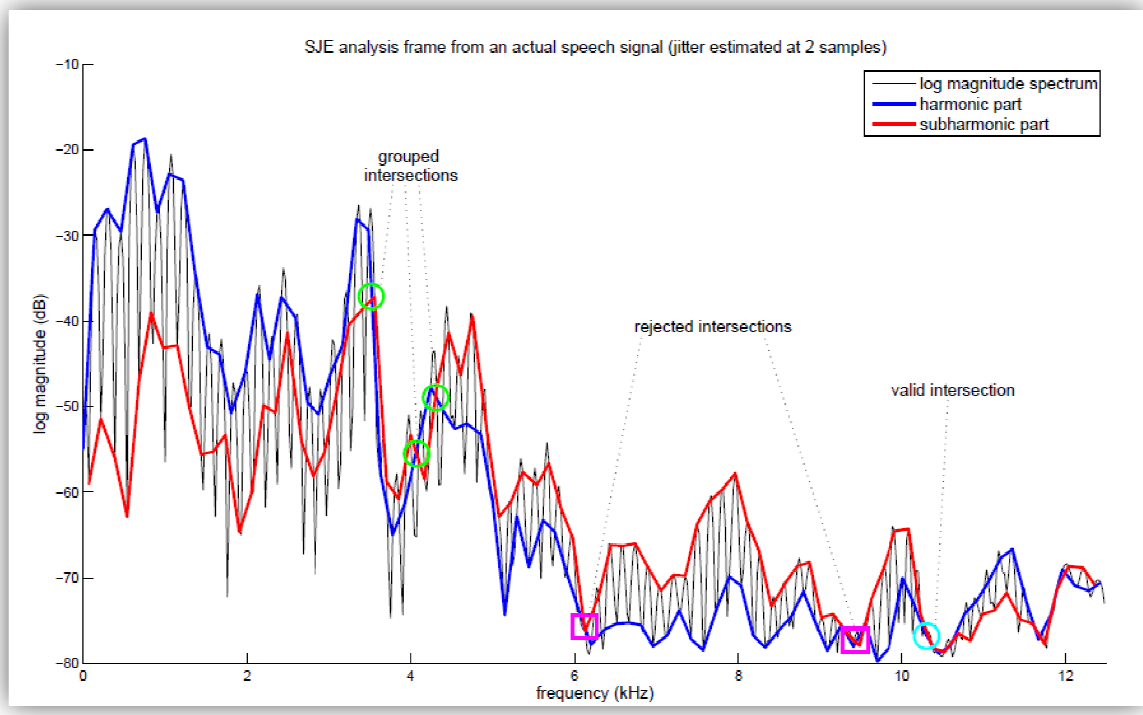

Figure 23. Heuristic method provided in Miltiadis Vasilakis master thesis [5].

The Vasilakis SJE implementation, where the two aforementioned operations are found, is presented in appendix 8 .

4.2. Version II: combined SJE

The spectra of sustained vowels or connected speech are different from those of the previously analyzed ideal excitation pulses. Some modifications are proposed in this section combining the two aforementioned SJE estimators to deal with realistic signals.

Since real speech is more complex than the aforementioned ideal, the Vasilakis estimator is implemented with a heuristic crossing detector. This heuristic may improve our method by decreasing the overestimation owing to the appearance of spurious contour crossings. This heuristic is combined with our peak picker to try for more accurate jitter estimates. Version II, which is presented in appendix 9, is developed especially for speech signals.

4.3. LPC-based estimator

The LPC-based analysis is presented in this section as it a way to approach the ideal pulse sequence. Namely, by means of a LPC-filter it is possible to convert the input signal into a spiky representation that is more similar to a perturbed spike train than the original waveform. By LPC-filtering it is possible to split approximately a speech signal into a pulse train and N-pole filter. The LPC-residue, whose Matlab code is given in appendix 10, is used to ascertain the accuracy of the jitter estimates in this framework.

In order to predict the amplitude of a waveform sample, a linear predictive formula is applied to the amplitude values of the preceding samples. The error, i.e. the LPC-residue, corresponds to the difference between the real amplitude value and the predicted one. Generally speaking, the aim of LPC analysis is not only that of predicting values, but also that of getting the most suitable coefficients such that the LPC-residue is as small as possible.

Linear prediction is a mathematical operation where future values of a discrete-time signal are estimated as a linear function of previous samples. The predictive formula finds the coefficients a=[a_1 a_2 ... a_N]

$$\hat{u}(n) = \sum_{i=1}^{N} a_i u(n-i)$$

where the integer N is called the prediction order, $\hat{u}(n)$ is the predicted signal value, u(n-i) the previous observed values and a_i the predictor coefficients. The error e(n) generated by this estimate is minimized,

$$(\mathbf{5})\ e(n) = u(n) - \hat{u}(n)$$

where u(n) is the true signal value. The LPC-analysis provided by Matlab uses the Levinson-Durbin recursion to solve the normal equations that arise from the least-squares formulation. This computation of the linear prediction coefficients is usually referred to as the autocorrelation method.

Notice in figure 24 how the input signal, namely a sustained /a/ sound, becomes spiky by LPC-filtering.

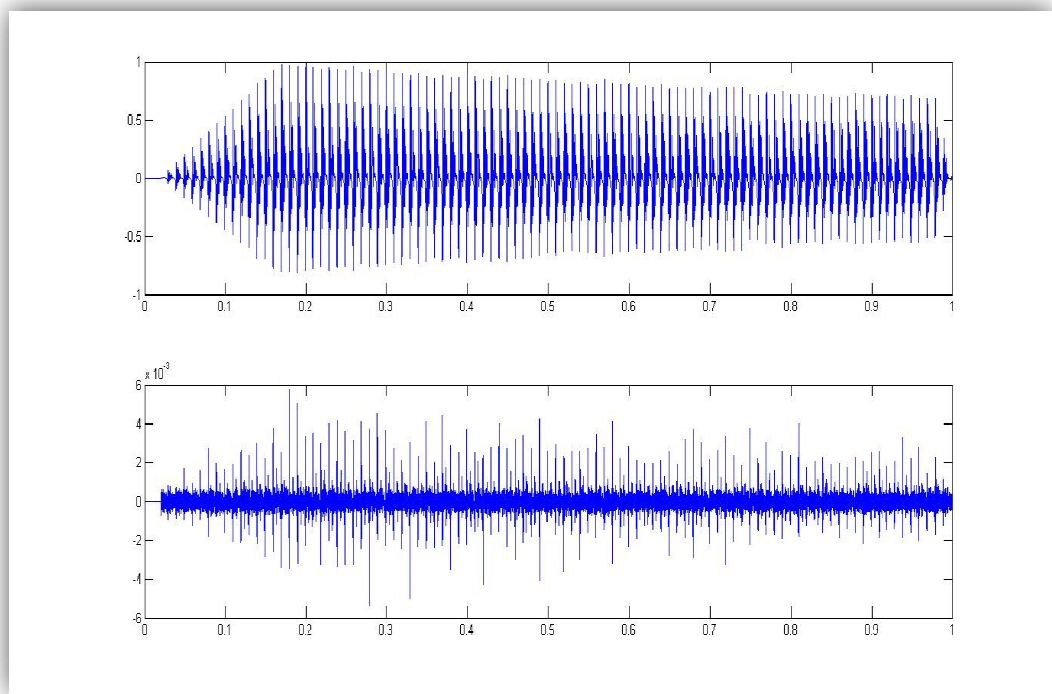

Figure 24. Sound /a/ pre and post LPC-filter respectively.

Although the amplitude of the spikes is not constant as in the ideal situation, the LPC-residue is more similar to a perturbed spike train than the original sound as depicted in figure 24. The variability of the peaks and the background noise may however degrade the estimation. Hence, we have to ascertain whether there is an increase in the accuracy of the jitter estimates when the LPC-filter is used.

The LPC-analysis is applied to the input signal to get the linear filter coefficients. We noticed experimentally that 40 is an appropriate filter order for sounds sampled at 22.05 kHz, which is the frequency corresponding to the synthetic speech sounds that are studied in section 5.2. The LPC-coefficients are then used to obtain the LPC-residue, which is the difference between the real amplitude and the predicted value as shown in formula **(5)**.

Given a sustained sound, the LPC-coefficients can be obtained for the whole signal since there are no transitions. It is therefore not necessary to apply the predictive formula locally. Nevertheless, in sequences of sustained sounds or connected speech the LPC filter coefficients have to be updated over time. In this work, in addition to connected speech we study sustained sounds and pairs of sounds. Regarding pairs of sounds, there is a single transition between first sound and the second. We simplify the analysis by assuming that the predicted LPC-coefficients can be estimated for the first and the second sound separately. In connected speech, however, it would be necessary to carry out local LPC-analysis.

4.4. Differences and similarities between analysis methods

In this work, different implementations for the spectral estimation of vocal jitter are presented. All the methods are based on the same mathematical model but they have implementation differences, which are presented in table 2.

SJE	**Version I**	**Vasilakis**	**Version II**
Detection of harmonics and inter-harmonics	Peak picking	Linear interpolation between theoretical positions	Peak picking
Detection of contour crossings	Threshold+proximity to the expected location (section 3.6)	Heuristic positions detection via thresholding, counting and recombination (section 4.1.2)	Heuristic positions detection via thresholding, counting and recombination (section 4.1.2)

Table 2. Differences between SJE implementations.

Regardless of those differences, they have a common structure. The Matlab applications allow changing parameters that are common to the three implementations. These parameters are presented hereafter.

- **Window type**. It is possible to choose the window type between rectangular, triangular, Hamming and Hanning. By experimenting we determined that these are the most suitable window shapes. The jitter estimates are very similar using any of those on isolated spike trains. However, in actual speech the rectangular is disregarded due to the aliases produced by the temporal discontinuities at the edges.
- **Frame size**. It is possible to choose the frame size however, between three and four periods is an adequate frame length to estimate jitter locally. Four periods is in fact more suitable because increasing the window length improves the resolution and therefore the distinction between harmonics and the inter-harmonics.
- **Hop size**. Although it is possible to modify the hop size, if one wants to obtain all local jitter values the hop has to be fixed to one period.
- **f_o analysis**. Two possible analyses can be chosen among the Matlab functions. Namely, the average fundamental frequency for the whole signal or an array of local f_o values. When ideal signals or sustained vowels are analyzed, the average f_o provides a satisfying estimation. In connected speech, however, f_o changes significantly and accordingly local pitch estimates are required.
- **FFT**. Each method performs a FFT (Fast Fourier Transform) to obtain the frequency-domain representation of the input signal.
- **Threshold**. A threshold criterion is utilized in the three implementations. The threshold is kept constant for all the experiments with a value of 3dB given that it has been observed empirically that it is an adequate value.
- **LPC-filter.** The LPC-analysis can be also applied in the aforementioned methods. Nevertheless, it is only implemented for analyzing sustained sounds or pairs of sounds.

5. Test and results

The validity and performance of the aforementioned models of SJE are analyzed in this chapter. The assessed signals are the assumed ideal pulsatile glottal airflow, with jitter and noise, various synthetic sustained speech sounds and finally, recordings of connected speech. The results of the tests are provided in this chapter by comparing with Praat's jitter estimates.

To assess reliability, Praat was used as a reference system. The signal processing that is involved in Praat's jitter detection is based on the assumptions of local stationarity and periodicity, which enable detecting and isolating vocal cycles or spectral harmonics. These assumptions apply to signals that are stationary fragments of sustained vowels. Even then, these heuristics may fail when the voice is severely hoarse. In the case of signals that are very irregular, insertion or omission errors of speech cycles or spectral harmonics are indeed frequent. These values bias the accuracy of the results and consequently, Praat can only be considered to be reliable when it is used with voice sounds that are feebly or moderately hoarse. Even so, we assume that the average jitter provided by Praat is accurate, despite the mentioned inaccuracy in highly perturbed signals.

Given that the SJE is able to perform local jitter estimates, short-time statistics are also examined through cross-correlation. Cross-correlation is a measure of similarity of two waveforms as a function of a time-lag applied to one of them. Two identical waveforms have a correlation value of 1 whereas two independent have 0 correlation statistically speaking. Local jitter estimates are compared in each windowed frame namely, local jitter obtained by Praat and that of the SJE. The performed tests showed a significant correlation for the ideal pulse sequence but not for synthetic phonation or connected speech. The results are therefore only presented in the ideal situation. Regarding synthetic phonation and actual speech, although the cross-correlation with Praat is not reliable the average jitter enabled discriminating moderately perturbed signals. In real signals we therefore focus on global statistics.

5.1. Unit pulses

For the verification of the validity of the proposed methods 48 synthetic sustained signals and pairs of sounds are used. The corpus has four different jitter levels as well as four additive noise levels. Both values are control parameters of the synthesizer where the synthetic speech sounds were created. The synthetic sounds timbres are /a/, /i/, /u/, /ia/, /ai/ with the following parameters:

- **Fundamental frequency**. 100, 120, 140 kHz.
- **Jitter**. 0.05, 0.15, 0.3, 0.4.
- **Noise**. 0, 1.1, 3.1, 6.1.

Each synthetic signal has a sampling frequency 22.05 kHz, which is not enough for an accurate estimation. We must upsample given that enlarging the resolution reduces the quantization step. After testing various synthetic signals with different sampling frequencies and taking into account the relative error, we concluded that at least 160 kHz are required to have a suitable accuracy. For 160 kHz, the quantization step is q_s=12.375 µs, which is an adequate value because it has the same order as the lowest jitter value we expect to measure. The tests are therefore performed upsampling 8 times the initial sampling frequency.

$$F_s = 8 \times 22.05 = 176.4 \text{ kHz}$$

The new quantization step is then q_s=11.224 µs. The SJE is initially tested in the ideal situation using realistic jitter and noise parameters. Knowing the pulse positions provided by Praat, unit pulses with realistic positions are created by generating a set of pulses in the abovementioned locations. The mathematical model is simulated by creating differently perturbed unit pulse sequences. Our initial implementation, i.e. Version I, and that of Vasilakis are tested and assessed given the ideal pulse sequence.

The spectral characteristics are used to obtain jitter estimates by counting the number of intersections between the harmonic and the inter-harmonic contours. Since the number of intersections is actually just for half of the spectrum, the number of contour

crossings, which correspond to jitter in samples, has also to be doubled. It is therefore converted from samples to µs so that it is analogous to the absolute jitter units provided by Praat. Although Praat's jitter estimates are an average over the whole signal, it also provides a pulse listing of pulse positions in time units. Hence, short-time jitter can be obtained by calculating the cycle-to-cycle fluctuation of the fundamental frequency as shown in formula **(3)**. The SJE provides a sequence of short-time estimates, which are correlated with the estimates provided by Praat. For comparison purposes, we also compute the average value of the local jitter sequence to analyze global statistics as shown in formula **(4)**.

Besides Praat's estimates, the average jitter is also computed in a different way. Namely, the pulse positions provided by Praat are used to compute local jitter as the difference between neighboring cycles. Since the mentioned local jitter is in samples we have to convert to µs so that it is analogous to Praat. Averaging the local jitter sequence we obtain global jitter as given in Praat. Although we have two different reference values of global jitter that of Praat is considered as a reference whereas the other is simply illustrative.

Firstly, local statistics are presented in order to obtain more detailed information about gradual behavior of jitter. Namely, the cross-correlation between Praat's local estimates and that of Version I and Vasilakis methods are performed. Given that Praat may get into trouble with severely perturbed sounds or noisy signals some of the pulses might not be detected. We therefore get rid of those ones and only utilize the subset of pulses detected by Praat to compute local jitter. The results are very similar using a frame size of 3 or 4 times the period; however a window size of 4 periods is used as it provides a slight improvement on the accuracy of jitter estimates.

The cross-correlation is initially depicted for Version I. In figures 25, 26 and 27 three-dimensional plots corresponding to cross-correlation of /a/, /i/, /u/, /ai/ and /ia/ sounds are displayed. Notice that the x-axis is jitter, y-axis is noise and z-axis is cross-correlation. We have 48 colored points corresponding to the previously mentioned synthetic speech signals. Henceforth, the control parameters and the pitch are differenced as given in the mentioned figures.

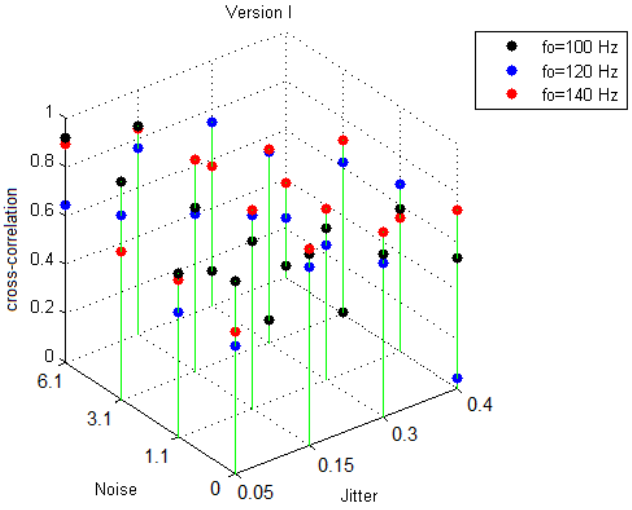

Figure 25. Cross-correlation performed by Version I with sound /a/.

Further information is presented in the appendix. Namely, numerical data corresponding to average jitter in µs is given in appendix 11 whereas cross-correlation is given in appendix 12, where J is jitter and b is additive noise.

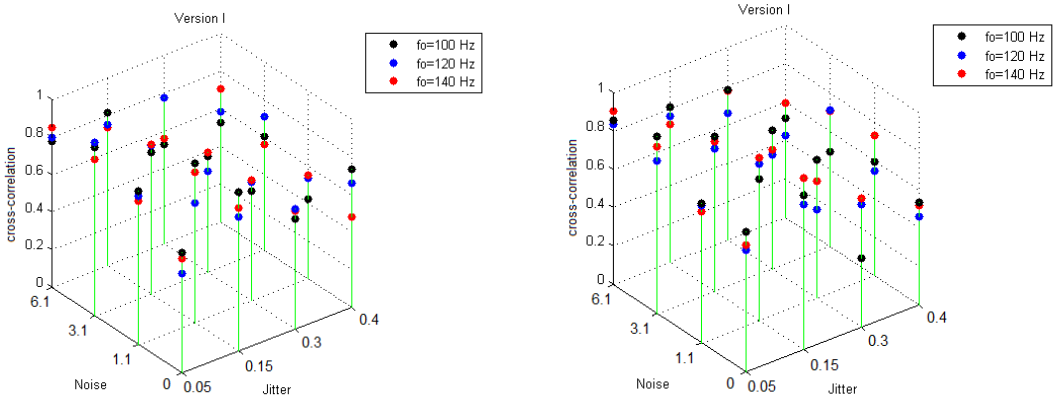

Figure 26. Cross-correlation performed by Version I with sounds /i/ and /u/ respectively.

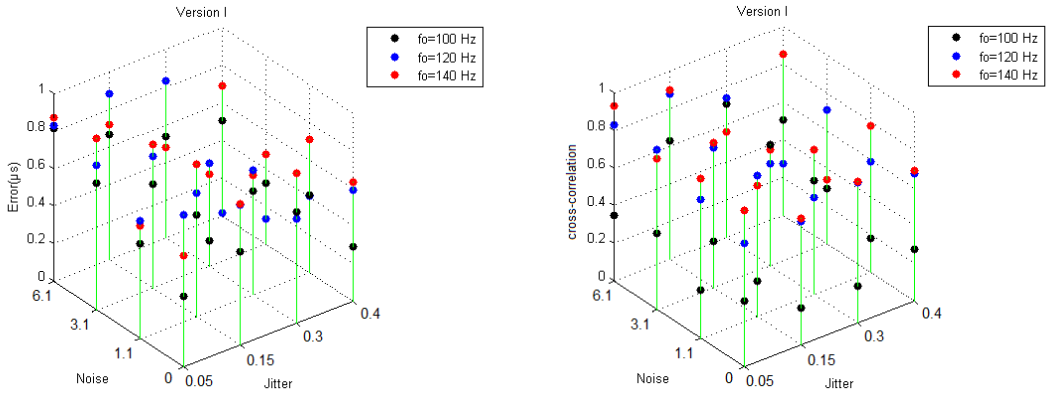

Figure 27. Cross-correlation performed by Version I with sounds /ai/ and /ia/ respectively.

Notice that the estimates are resistant to noise, larger values do not degrade the cross-correlation. Given that the pulse sequence has been created without external additive noise, the noise parameter has the same influence as jitter, i.e. it causes shifts on the pulse sequence. Therefore, enlarging jitter has similar effect as enlarging the noise parameter. Severely perturbed signals cause inaccuracy. When jitter is higher than 0.3 the cross-correlation presents significant dissimilarities among local estimates. Quantization noise may cause degradation in the cross-correlation along with other factors.

The cross-correlation is now depicted for the Vasilakis implementation. In figures 28, 29 and 30 three-dimensional plots corresponding to cross-correlation of /a/, /i/, /u/, /ai/ and /ia/ sounds are displayed. Notice that there is a noticeable decrease in the average cross-correlation from Version I.

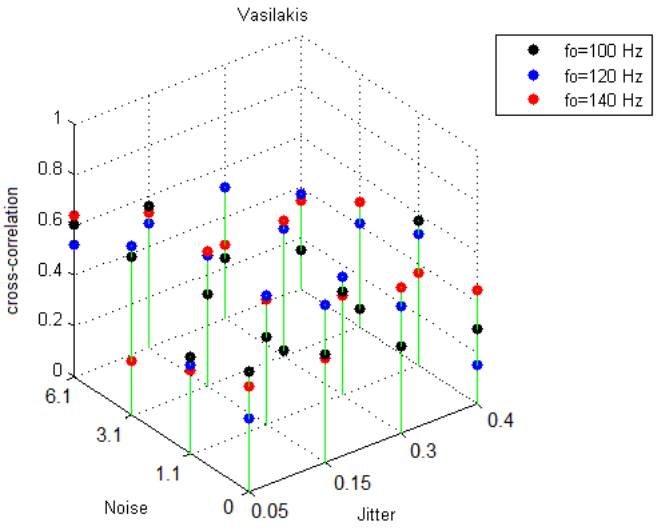

Figure 28. Cross-correlation performed by Vasilakis with sound /a/.

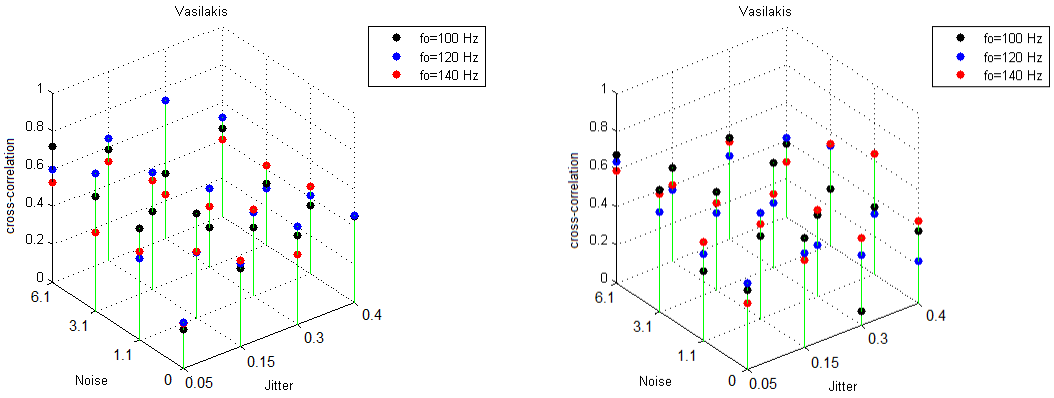

Figure 29. Cross-correlation performed by Vasilakis with sounds /i/ and /u/ respectively.

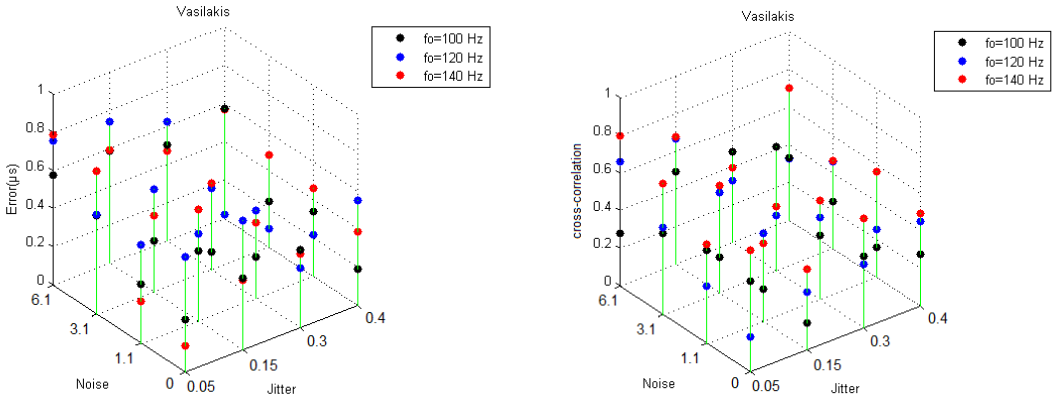

Figure 30. Cross-correlation performed by Vasilakis with sounds /ai/ and /ia/ respectively.

The Vasilakis estimator has a lower average cross-correlation as given in table 3. Namely, our estimator provides an average cross-correlation of 0.6400 whereas that of Vasilakis 0.4385. There is therefore a noticeable improvement in the value, which means that our local jitter estimates are more similar to Praat's than that of Vasilakis.

Average cross-correlation	**/a/**	**/i/**	**/u/**	**/ai/**	**/ia/**	**Average**
Version I	0.6222	0.6865	0.6942	0.5983	0.5987	**0.6400**
Vasilakis	0.4246	0.4500	0.4497	0.4336	0.4347	**0.4385**

Table 3. Average cross-correlation corresponding to Version I and Vasilakis.

The cross-correlation provides information about the time-depending behavior of jitter but we still have to ascertain the accuracy of average jitter. After analyzing local statistics, global statistics are presented. The accuracy of average jitter is assessed by studying the absolute error *e*, i.e. the difference between Praat's average jitter and that of both methods.

$$e = |J_p - J_e|$$

where J_p is the average jitter provided by Praat and J_e that of the SJE. As in the previous analysis, the results are provided for the aforementioned impulse sequence based on a real corpus. The error is initially depicted in µs for Version I. In figures 31, 32 and 33 three-dimensional plots corresponding to error of /a/, /i/, /u/, /ai/ and /ia/ sounds are depicted.

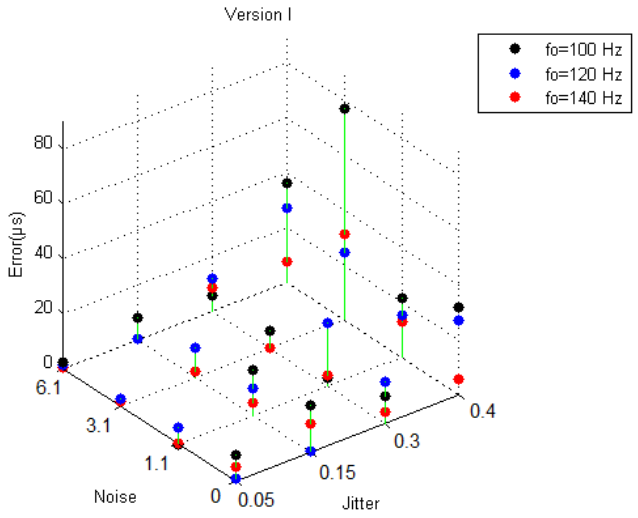

Figure 31. Average jitter error in µs performed by Version I with sound /a/.

Further information is presented in appendix 13. Namely, numerical data corresponding to the average errors in µs, which are used in figures 31-36.

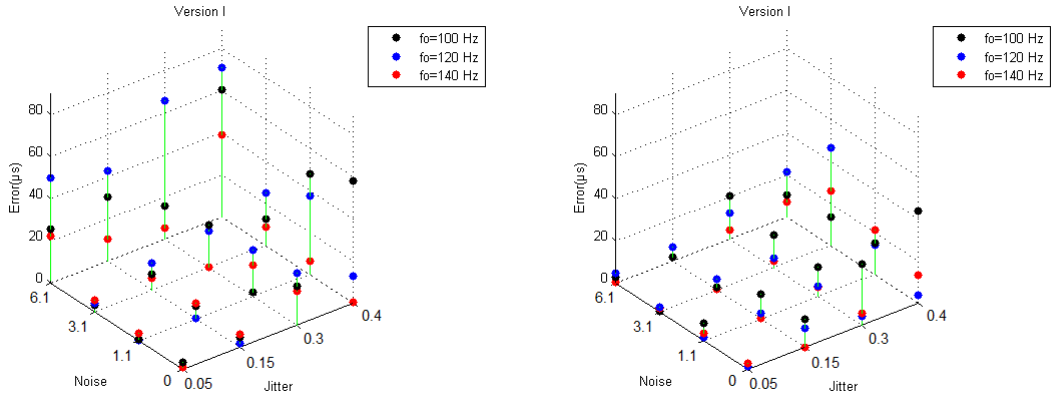

Figure 32. Average jitter error in µs performed by Version I with sounds /i/ and /u/ respectively.

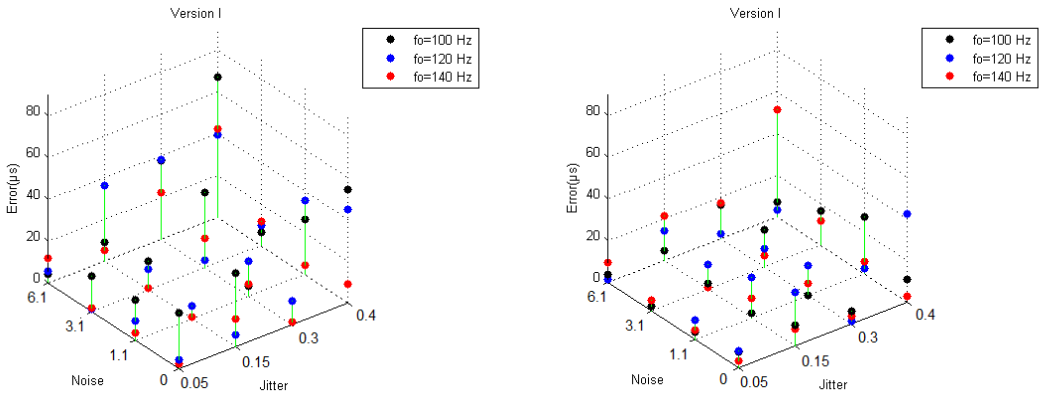

Figure 33. Average jitter error in µs performed by Version I with sounds /ai/ and ia/ respectively.

The difference between Praat's jitter estimates and that of Version I is in the order of some µs for almost all values of jitter and noise. In accordance with local statistics, there is a noticeable inaccuracy for highly perturbed signals. There is a correlation between jitter and error, i.e. large jitter causes a higher estimation error. In figures 31, 32 and 33 we notice that jitter is more influent than noise concerning accuracy on the measurement, even so noise also produces imprecision.

The average error is now depicted for the Vasilakis implementation. Since the error is higher than ours, we have changed the scale. Namely, the z-axis upper limit is now 180 µs whereas 90 µs was displayed before. Although the scaling is different it is interesting to analyze the gradual behavior of error as depicted in figures 34, 35 and 36.

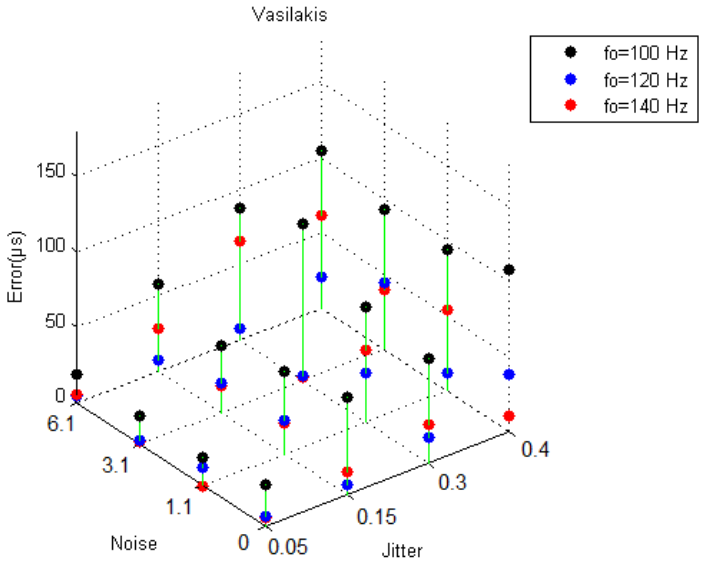

Figure 34. Average jitter error in µs performed by Vasilakis with sound /a/.

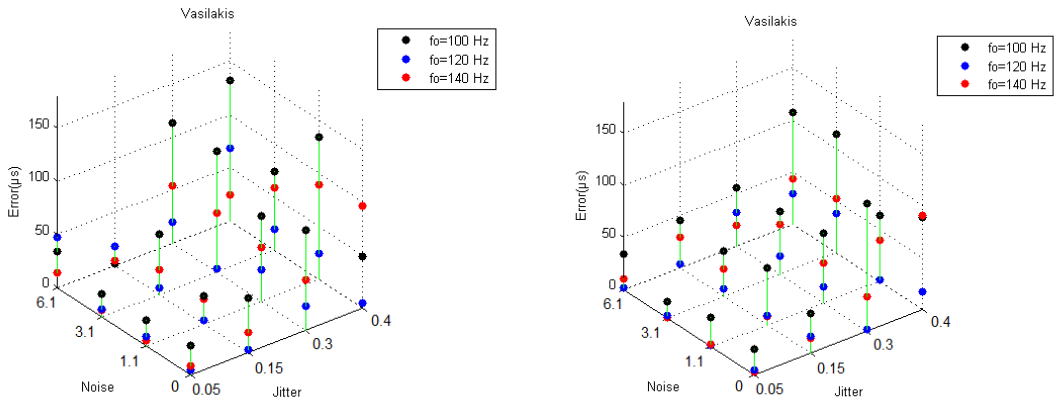

Figure 35. Average jitter error in µs performed by Vasilakis with sounds /i/ and /u/ respectively.

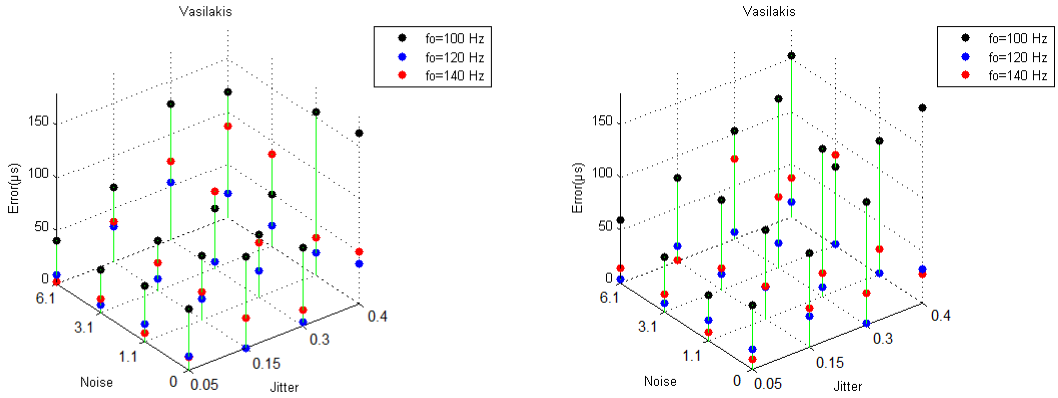

Figure 36. Average jitter error in µs performed by Vasilakis with sounds /ai/ and /ia/ respectively.

Large jitter causes critical effects on the estimation whereas noise has no significant influence. Hence, the Vasilakis implementation can only be considered to be reliable for feebly perturbed signals, at least in the ideal situation. We still have to ascertain the accuracy concerning synthetic and real speech signals given that Vasilakis was developed for that purpose.

Version I provides more precise jitter estimates than the Vasilakis implementation. Namely, the average is 13.2113 µs for Version I whereas 40.7462 µs for Vasilakis as depicted in table 4.

Average error (μs)	/a/	/i/	/u/	/ai/	/ia/	Average
Version I	11.0833	17.8942	9.3422	17.4801	10.2567	**13.2113**
Vasilakis	35.2917	39.7304	33.8848	45.6136	49.2107	**40.7462**

Table 4. Average jitter error in μs performed by Version I and Vasilakis.

More interesting than the absolute errors are the relative errors e_r in (%) as they provide better insight of the legitimacy of the results. At this stage then we ascertain whether the relative errors are reasonable in terms of jitter estimation.

$$e_r = 100 \times |\frac{J_p - J_e}{J_p}|$$

where J_p is the average jitter provided by Praat and J_e that of the SJE. The relative errors are depicted in figures 37- 41 for Version I and that of Vasilakis.

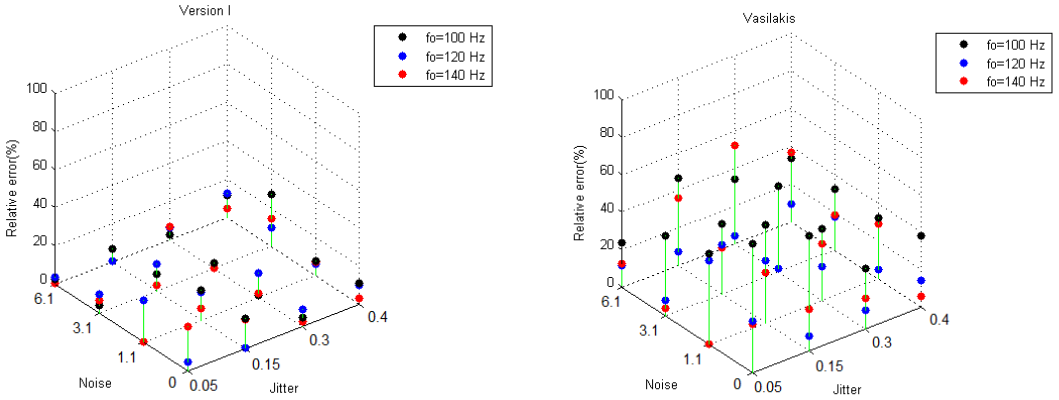

Figure 37. Relative jitter error in (%) performed by Version I and Vasilakis respectively, with sound /a/.

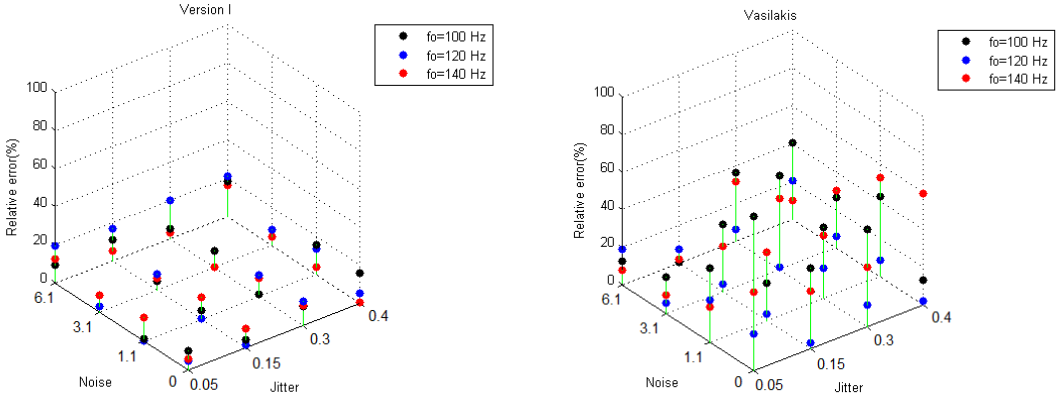

Figure 38. Relative jitter error in (%) performed by Version I and Vasilakis, respectively with sound /i/.

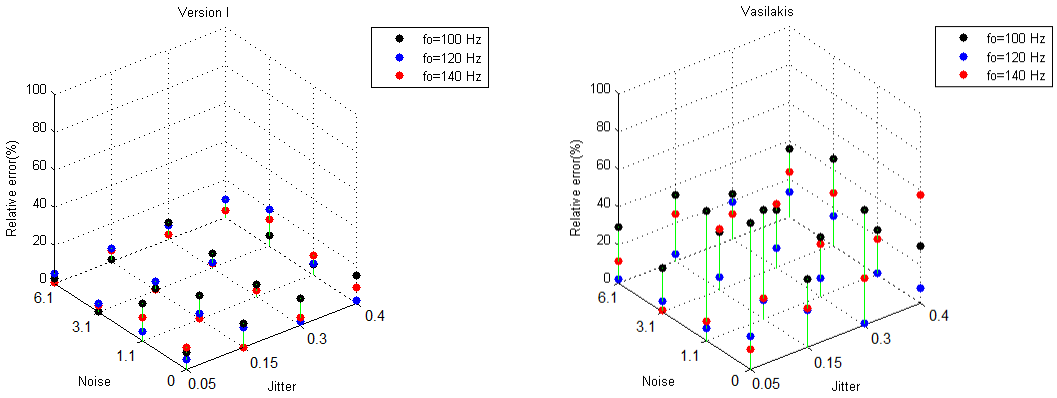

Figure 39. Relative jitter error in (%) performed by Version I and Vasilakis, respectively, with sound /u/.

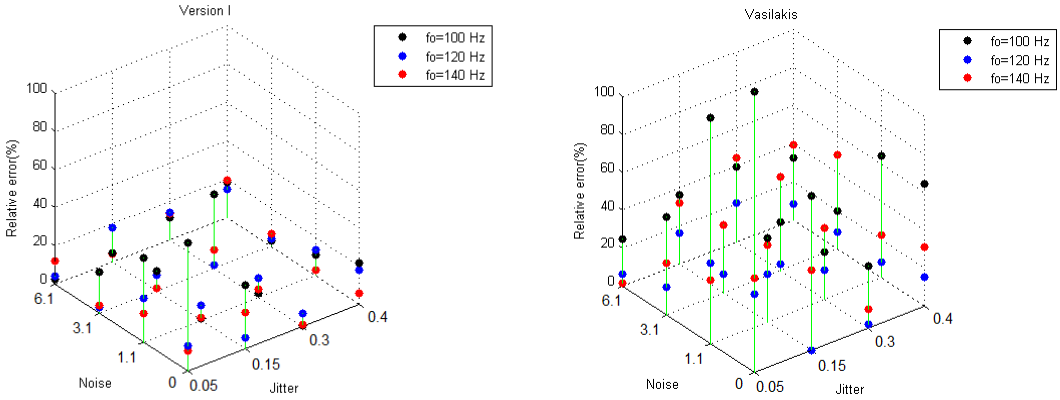

Figure 40. Relative jitter error in (%) performed by Version I and Vasilakis, respectively, with sound /ai/.

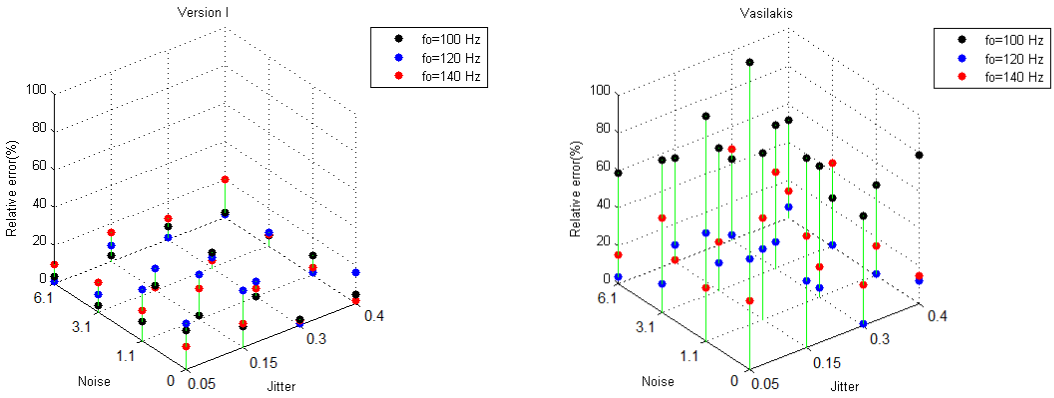

Figure 41. Relative jitter error in (%) performed by Version I and Vasilakis, respectively, with sound /ia/.

Notice that unlike absolute errors, Version I provides relative errors which are more regular in terms of variation than that of the Vasilakis. Version I has therefore reasonably similar relative errors regardless jitter and noise. The average relative error

performed by Version I is 8.8056 % whereas 29.8325 % by Vasilakis as shown in table 5.

Relative error (%)	Sound /a/	Sound /i/	Sound /u/	Sound /ai/	Sound /ia/	Average
Version I	7.9816	8.4543	6.3588	12.2846	8.9489	**8.8056**
Vasilakis	26.2355	24.3535	23.6851	34.0223	40.8662	**29.8325**

Table 5. Relative errors in (%) performed by Version I and Vasilakis.

Generally speaking, the results evidence a noticeable improvement of Version I from the previous SJE implementation carried out by Vasilakis. The average errors, the relative errors and the cross-correlations show a better performance, at least in the ideal situation. The errors are reasonable in terms of jitter; therefore it is worth looking forward to test the SJE with synthetic speech sounds, as provided in the next section.

5.2. Synthetic phonation

Sustained sounds are analyzed especially when the goal is to estimate jitter, additive noise or other cycle irregularities. The reason for using sustained sounds is facility, due to the assumptions of cyclicity and stationarity. In this section, therefore, the validity of the analysis methods is assessed with regard to synthetic phonation. Performance is analyzed by testing the aforementioned corpus, i.e. synthetic recordings corresponding to /a/, /i/, /u/, /ia/, /ai/ sounds with average fundamental frequency of 100, 120 and 140 Hz. Additionally, the method is further assessed by testing a highly perturbed /a/ sounds. This corpus provides different insight into the reliability of the estimates when jitter becomes larger than in standard voices. The synthetic signals are upsampled 8 times to have a suitable sampling frequency. Initially, the corpus has a sampling frequency of 22.05 kHz but the tests are performed with 176.4 kHz.

Before we start to evaluate the results we analyze the spectrum of a sustained sound in the short term. As far as windowing is concerned, a triangular window shape is an

appropriate choice. It has an adequate tradeoff between discerning the harmonics and the inter-harmonics and avoiding alias problems as explained in section 3.7. The mathematical model describes the association of the harmonic and the inter-harmonic contours in a specific way. Namely, jitter leading to an identifiable spectral pattern as the coupling between both contours. However, this assumption remains to be ascertained in sustained sounds. Lower frequencies of a voiced speech segment have almost all the energy hence; it may be easier to predict the structure there than in higher frequencies where noise effects degrade spectral regularity. Upsampling improves resolution but increases unpredictability owing to noise. This limitation is explained in section 6.4. See figure 42, where a windowed frame of sound /a/ is depicted temporally and spectrally.

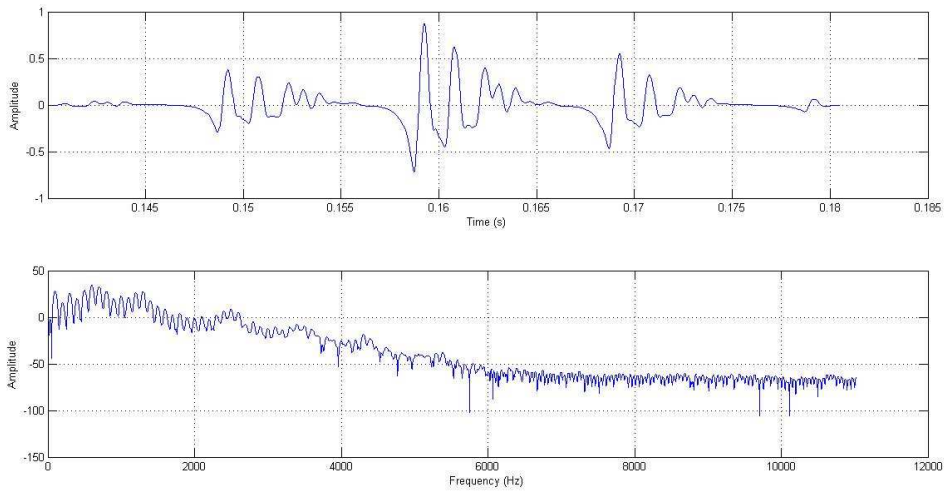

Figure 42. Time-domain and frequency-domain representation of a windowed frame of a sustained /a/ sound. The window used has a triangular shape.

Higher frequencies are noisier than lower and it is therefore more difficult to perform an accurate estimation of the number of contour crossings there. An estimation of jitter might not be reliable if we do not deal with that issue. For that purpose the Vasilakis heuristic crossing detector was developed as presented in section 4.1.2. It tries to reduce the overestimation caused by spurious crossings that may occur mostly in higher frequencies.

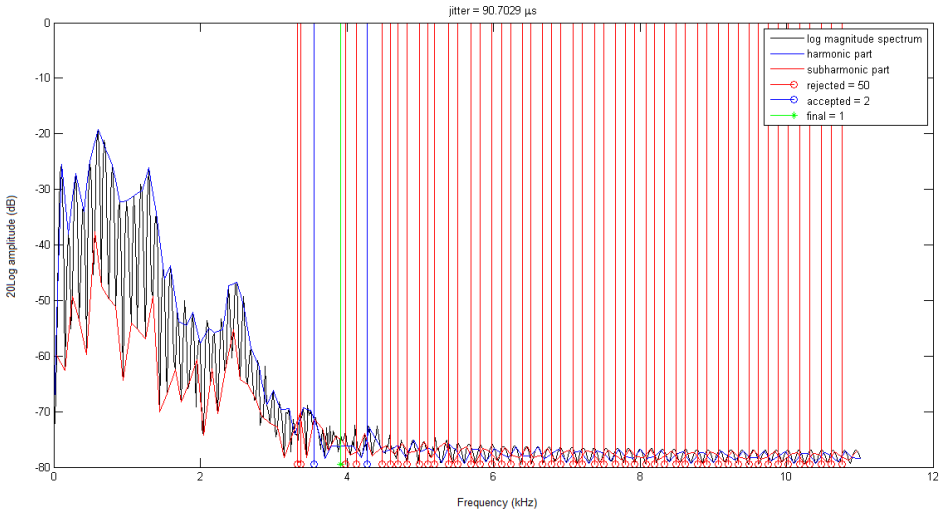

Figure 43. Log magnitude spectrum displayed by the Vasilakis implementation [5].

Figure 43 shows a frame of a synthetic /a/ sound analyzed by the heuristic crossings detector. Note that some of the candidate crossings are rejected and others are accepted but in the end only one satisfies the heuristic criterion. The final estimation is one sample which is converted to time in µs as given in formula **(2)**. When the perturbation is one sample Ɛ=1, the theoretical crossing location obtained from formula **(1)** is π/2. Nevertheless, the heuristic detects the crossing far from the expected position. The reason is that the search is performed in a 25% interval around the theoretical position. Since the obtained crossing is not even close to the position predicted by the theoretical model, it is difficult to ascertain whether it is indeed a crossing. We noticed that reducing the search interval to 10% there was a slight improvement of the accuracy of the estimates and thus, we applied the aforementioned interval to the Vasilakis implementation.

The main problem when speech sounds are analyzed is that the spectrum is different from that of spike trains; hence the theoretical crossing position is not totally reliable. It is difficult to predict the performance of the implemented methods as they have different Matlab algorithms to measure jitter. The results are therefore provided using Version I, Version I LPC-based, Version II, Version II LPC-based and Vasilakis to ascertain their validity.

To represent accuracy of jitter estimates *boxplots* are used. Given the reference estimates provided by Praat, the average error over the whole corpus is depicted by one *boxplot*. In the *boxplots*, the central mark is the median, the edges are the 25th and 75th percentiles, and the outliers are plotted individually. The whiskers extend to the most extreme data points not considered outliers, i.e. values that are out of the bounds of reliability for the Matlab function. The *boxplots* are displayed for sounds /a/, /i/, /u/, /ia/, /ai/ as shown in figures 44, 45 and 46. Note that there is one *boxplot* for each mentioned implementation. Further information, namely numerical data corresponding to the average jitter estimates in µs, is provided in appendix 14.

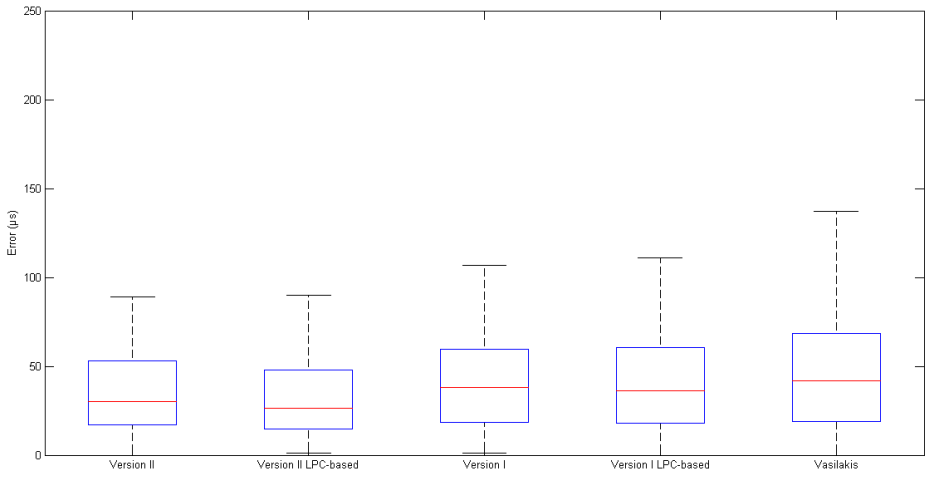

Figure 44. Error *boxplot* in µs corresponding to sound /a/.

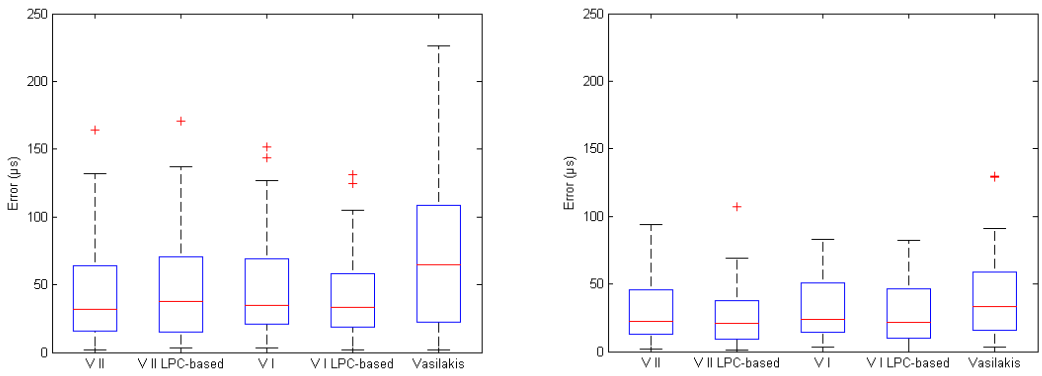

Figure 45. Error *boxplots* in µs corresponding to sounds /i/ and /u/, respectively.

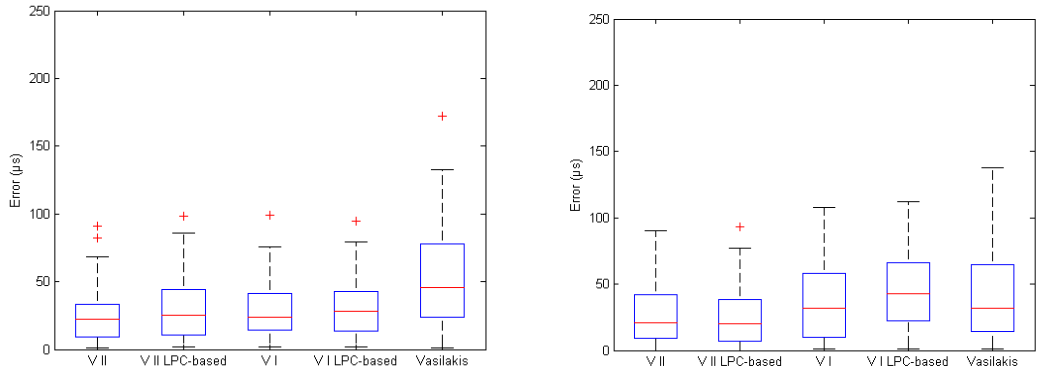

Figure 46. Error *boxplots* in µs corresponding to sounds /ai/ and /ia/, respectively.

Generally speaking, all the implementations have an average error between 25 and 50 µs, as shown in table 6. These jitter values are relatively precise; however the results are not as accurate as in the spike train situation.

Average error (µs)	**/a/**	**/i/**	**/u/**	**/ai/**	**/ia/**	**Average**
Version II	34.8542	44.0000	30.8750	25.7083	27.7083	**32.6292**
Version II LPC-based	33.7917	49.2083	25.4792	29.3958	24.7917	**32.5333**
Version I	40.1875	47.1875	32.5000	29.0625	37.1458	**37.2167**
Version I LPC based	40.8542	41.0833	29.4583	30.2083	46.1875	**37.5583**
Vasilakis	48.6458	73.6250	40.8958	53.8333	41.9792	**51.7958**

Table 6. Average error in µs corresponding to Version I, Version I LPC, Version II, Version II LPC and Vasilakis.

The highest errors are obtained by Vasilakis whereas the most accurate is Version II. We also notice from table 6 that there is no significant difference when the LPC-analysis is performed.

Further information is displayed in figures 47, 48 and 49, which correspond to three-dimensional plots of the errors in µs. In this section, only sound /a/ is displayed; figures corresponding to /i/, /u/, /ia/, /ai/ corpus are presented in appendix 15. Since the colored points do not sit on top of each other, pitch does not provide a noticeable influence on the error. Hence, there is no evident trend concerning pitch influence.

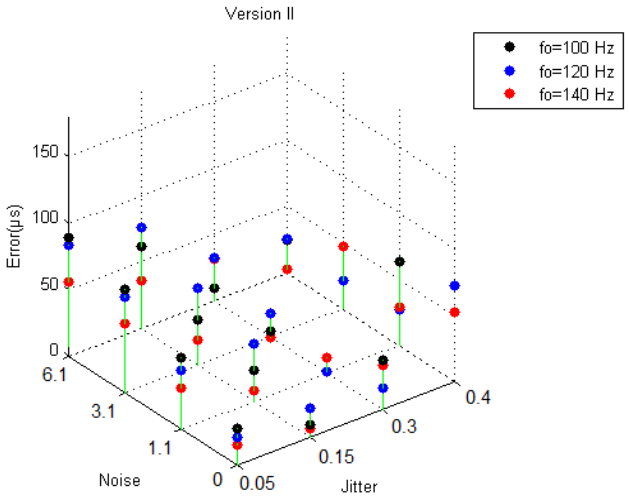

Figure 47. Average error in µs performed by Version II with sound /a/.

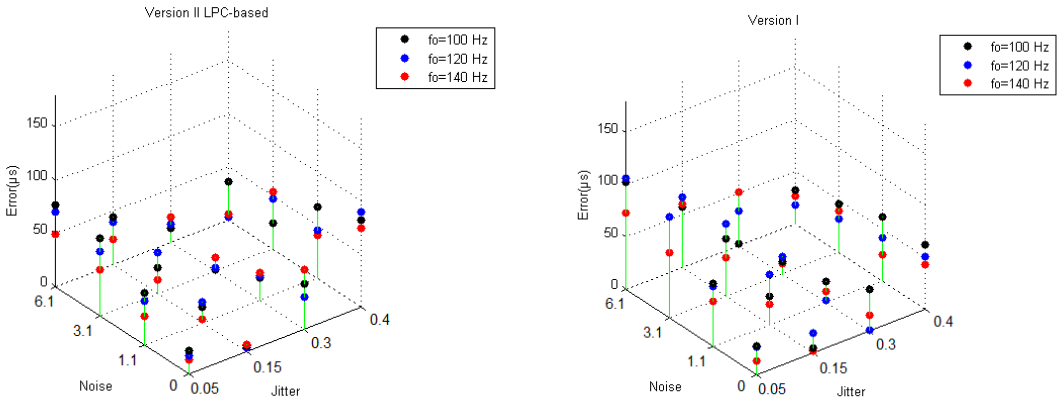

Figure 48. Average error in µs performed by Version II_ LPC and Version I, respectively, with sound /a/.

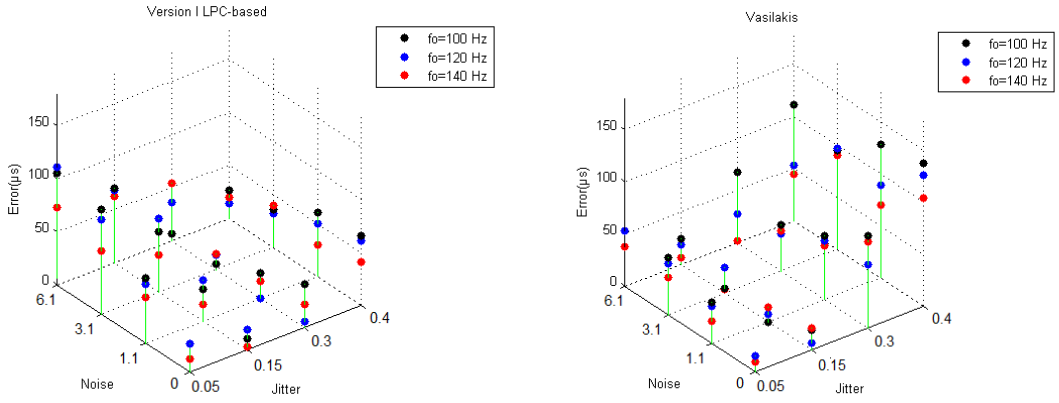

Figure 49. Average error in µs performed by Version I_ LPC and Vasilakis, respectively, with sound /a/.

Regarding the Vasilakis implementation, it provides accurate jitter estimates for weakly perturbed sounds; however it is imprecise for moderately and severely perturbed signals. Noise also degrades the estimation but it is not as influent as jitter. The heuristic crossing detector reduces the noise influence but it is not as useful as the peak picker concerning jitter variation. In conclusion, although it provides quite accurate jitter estimates for weakly perturbed signals, it is unreliable when jitter is significant.

As far as the other models are concerned, they are more resistant to moderately perturbed signals. Although the error increases with large jitter, they provide quite accurate estimates with moderately perturbed signals. Noise causes major degradation given that the peak picker is not as resistant to noise as the heuristic crossing detector.

The LPC-analysis does not provide a significant improvement and thus it is not as useful as we expected. The average errors have the same order of magnitude with or without LPC-filter. Given that there is no noticeable improvement it does not make sense to add computational work to the Matlab function by applying LPC-analysis. Even so, there is a noticeable correlation between methods; jitter estimates are similar whether the LPC-analysis is applied or not. This is reasonable given that the LPC-residue is a spiky representation of a signal, which is obtained applying a predictive formula. The spectrum has therefore to preserve many characteristics respecting the initial representation.

Finally, other perturbed sounds are studied. The aforementioned highly perturbed corpus has 20 different jitter control parameters, namely from 0.05 to 1 in 0.05 steps. No additive noise is involved to sound /a/, which has 44.1 kHz as fundamental frequency. The tests are performed upsampling 4 times the initial sampling frequency, i.e. F_s=176.4 kHz, to have a suitable quantization step. Jitter estimates in µs are provided in table 7. Notice that, in accordance with the previous analyses, jitter estimates can only be considered to be reliable with weakly perturbed speech sounds. Large jitter causes fatal errors and it is therefore hopeless to rely on the estimates provided by any of the studied SJE with highly perturbed signals.

Jitter (parameter)	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Pulses	**42**	**66**	**101**	**130**	**175**	**223**	**234**	**350**	**370**	**347**
Praat (Praat)	**41**	**66**	**101**	**132**	**176**	**222**	**250**	**347**	**377**	**389**
Version II	75	90	105	135	60	135	165	150	195	165
Version II LPC	73	98	123	153	58	149	191	182	198	208
Version I	77	103	108	126	65	135	137	140	143	149
Versin I LPC	56	83	94	115	49	120	133	128	145	145
Vasilakis	68	114	118	147	80	163	138	144	183	171

Table 7. Average jitter in µs corresponding to the highly perturbed sound /a/. Part 1.

Jitter (parameter)	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1
Pulses (Praat)	**332**	**469**	**436**	**416**	**451**	**426**	**480**	**483**	**489**	**517**
Praat	**398**	**533**	**545**	**485**	**552**	**573**	**591**	**639**	**700**	**726**
Version II	75	105	105	120	150	165	180	195	195	195
Version II LPC	79	112	133	148	166	195	212	197	232	218
Version I	91	108	130	135	133	154	145	147	152	152
Versin I LPC	71	91	108	115	121	144	138	148	150	161
Vasilakis	98	97	133	146	158	185	163	191	189	167

Table 7. Average jitter in µs corresponding to the highly perturbed sound /a/. Part 2.

Generally speaking, estimations are occasionally accurate for weakly perturbed signals and mostly Version I and Version II. The average errors are quite reasonable in terms of jitter, at least when the perturbations are small. Furthermore, the results show a slight improvement from the Vasilakis implementation. The LPC-analysis, however, does not improve the accuracy as one may expect. In the next section, we test the SJE with connected speech.

5.3. Connected speech

Connected speech is more informative than sustained sounds. Lack of stationarity as well as greater variability of the conditions under which phonation may take place is considered to be a greater challenge to a speaker's larynx [3]. Furthermore, speakers are less likely to compensate their possible voice problems while producing connected speech than while producing sustained sounds.

Jitter analysis is preferably performed on sustained vowels, because during phonation the speech signal is expected to be pseudo-periodic and thus, in the presence of jitter the aperiodicity is more easily perceived. However, sustained phonation recordings are limited to a small duration, namely a few seconds. After that, pathological speakers may feel discomfort. Even healthy speakers may not be able to maintain a steady voice. To consider the behavior of jitter for a larger interval of time, recordings of connected speech are more useful than sustained sounds. Speakers with a normal pace are able to breathe occasionally, while in sustained phonation a single intake of breath is involved. Connected speech provides therefore longer recordings for further examination. Since the SJE provides a short-time jitter sequence, it is feasible to carry out running speech analysis. All the mentioned factors should be considered to enhance the implementation for real speech signals; however we focus on the results, i.e. the jitter estimates obtained by the previously developed SJE.

Connected speech gives a different insight into the evolving behavior of jitter while speaking. Besides lack of stationarity, pitch varies from cycle to cycle. As a consequence, jitter analysis requires local values of fundamental frequencies to estimate jitter. Since we haven't implemented a f_o extractor, the tests are performed using the local pitch estimates provided by Praat. We use these values to obtain the overall deviation, i.e. the average jitter over the whole signal. As explained in section 3.7, a triangular window is a suitable window to analyze the signal in the short term. In section 5.3 we study sustained sounds and pairs of sounds. There was not a noticeable influence of the transition from one sound to the other; however connected speech is

more complex than pairs of sounds. In this section, the purpose is to ascertain whether enables estimating jitter reliably by means of the SJE.

Along with the unvoiced frames, any voiced frames that do not have at least one voiced neighboring frames in each direction are disregarded. The reason is that it does not make sense to obtain jitter when there is only background noise. The remaining voiced frames are considered as valid frames and thus, the SJE is used solely there to measure local jitter. According to the previous windowing analysis, a frame size of four times the local cycle length is used for the tests.

Since the SJE can only be considered to be reliable for small perturbations, it only makes sense to test it with a modal voice. Experiments are therefore performed for running speech of a teacher. Specifically, recordings corresponding to his voice counting from one to twenty pre and post-lesson. The voice was recorded on seven different days spread out over 2.5 months when giving lectures of between 2 and 3 hours of duration. Although it was recorded by a microphone attached to the neck, the external noise can be considered to be negligible. The influence of the vocal overuse is analyzed comparing jitter estimates pre and post-lesson.

Note the evolving behavior of pitch depicted in figure 50. There is a noticeable trend, namely the average vocal frequency tends to increase over time. Vocal dysperiodicities are also analyzed by means of the signal-to-dysperiodicity ratio (SDR) as depicted in figure 51. In accordance with the pitch, the SDR also has an increasing trend. One may speculate that this trend is a consequence of vocal overuse but it is difficult to ascertain the validity of that assumption.

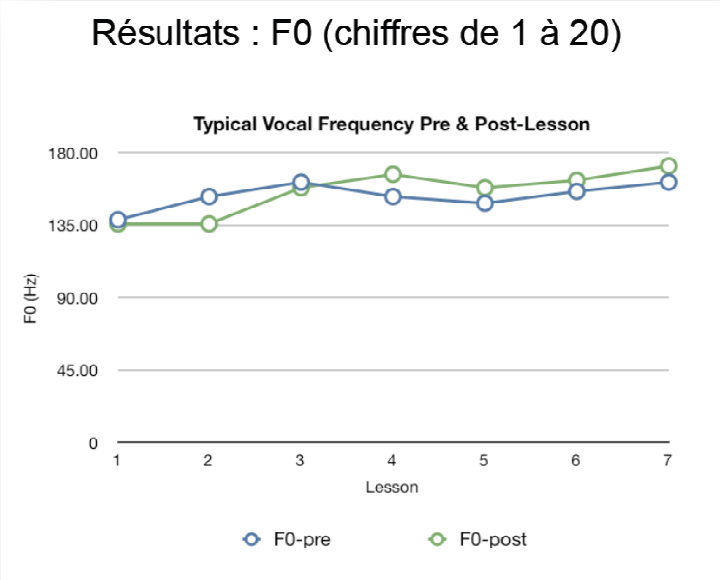

Figure 50. Evolving behavior of pitch over time [10].

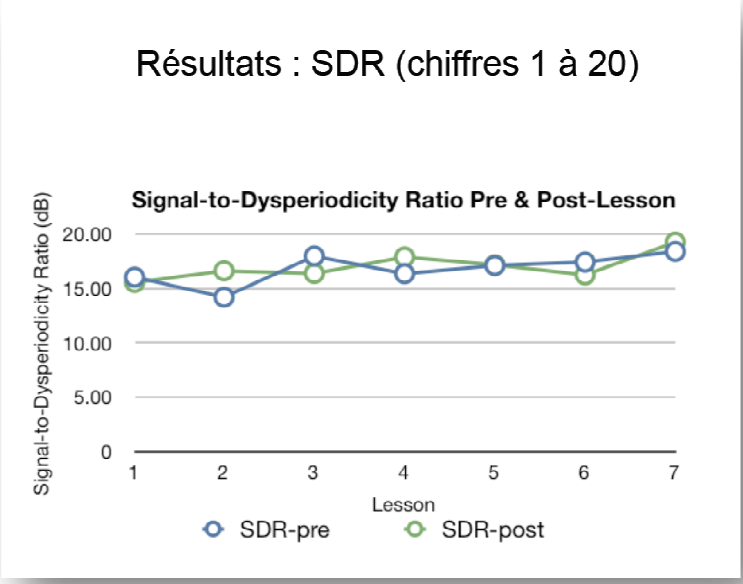

Figure 51. Evolving behavior of the signal-to-dysperiodicity-rati over time [10].

The evolving behavior of jitter over time is also depicted in figure 52. The values displayed, however, are the jitter estimates obtained by Praat. Note the decreasing trend of jitter, which is reasonable given that it has a contrary relation to the pitch.

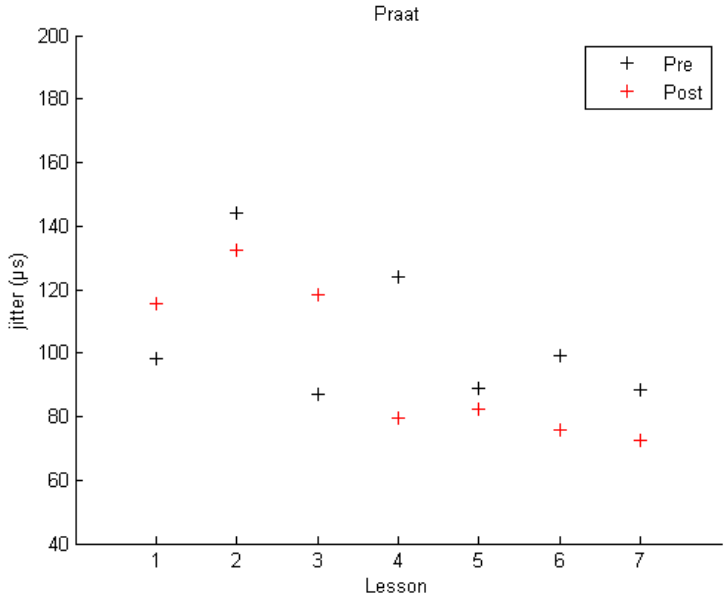

Figure 52. Evolving behavior of jitter over time obtained by means of Praat.

Version II and that of Vasilakis are tested with the aforementioned speech recordings. Results of Version II are now studied by means of graphic analysis. In figures 53 and 54 on the left, the x-axis is Praat's jitter stimates in µs whereas y-axis is that of the corresponding implementation. In figures 53 and 54 on the right, there are displayed jitter estimates in µs during the aforementioned seven days pre and post-lesson.

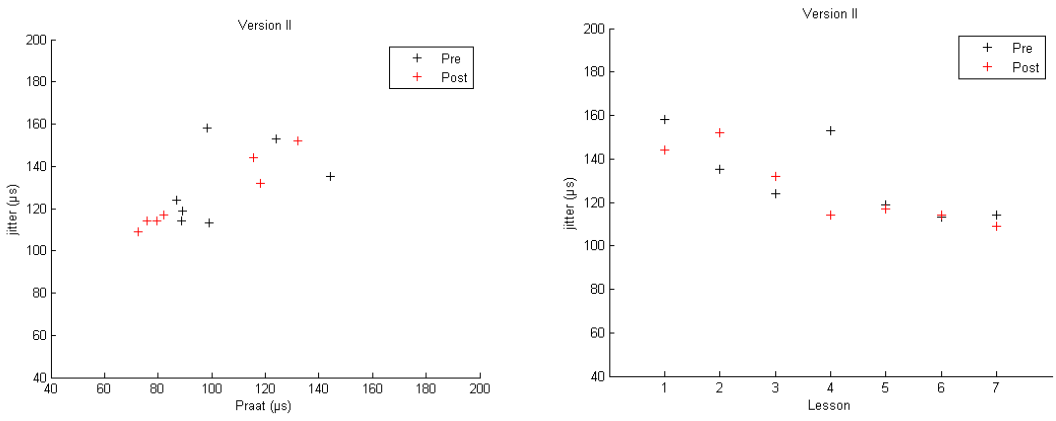

Figure 53. On the left, jitter correlation in µs between Praat and Version II. On the right, evolving behavior of jitter over time obtained by means of Version II.

Although there is an offset of about 40 µs, the results are not as inaccurate as one may fear. One therefore observes a significant correlation between the jitter estimates

obtained by Version II and that of Praat. Notice in figure 53 on the right that there is a decreasing trend contrary to the pitch. It is in agreement with Praat as one may expect. Version II enables discriminating jitter but it provides higher jitter error in connected speech than in synthetic sounds due to the complexity of the signal.

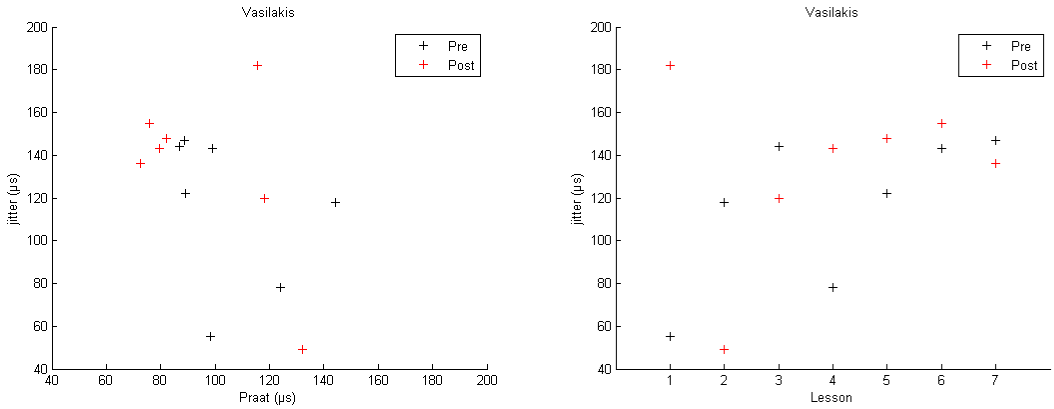

Figure 54. On the left, jitter correlation in µs between Praat and Vasilakis. On the right, evolving behavior of jitter over time obtained by means of Vasilakis.

There is no positive correlation between Vasilakis and Praat as depicted in figure 54 on the left. Furthermore, notice in figure 54 on the right that there is an increasing trend contrary to what we expect. The behavior of the Vasilakis implementation concerning real speech jitter estimation is therefore unexpected and in disagreement with Praat and Version II. Generally speaking, the estimates performed are usually unreliable, mostly for the Vasilakis implementation. Further information, namely numerical data corresponding to average jitter measured in µs, is given in appendix 16.

6. Conclusion

In this work, a method for short-time jitter estimation in the frequency-domain is studied. Jitter provides a large correlation with the pathological nature of voices and thus, an application which obtains jitter is interesting with regard to detection of disordered voices. A mathematical description modeling jitter is presented. This model transfers the jitter estimation problem from the time-domain to the frequency-domain and is used in the development of the short-time Spectral Jitter Estimator (SJE). The mathematical model describes the coupling of two impulse trains to simulate glottal

airflow. The spectrum provides an identifiable pattern which is used to obtain jitter, indirectly, by counting the number of crossings between the harmonic and the inter-harmonic contours. The method assessment suggest that it is possible to quantify jitter via this property.

Two versions of the SJE are developed taking into account the spectral characteristics provided by the mathematical model. Version I is the initial implementation and Version II is an improvement which combines our implementation and that of Vasilakis. The aforesaid implementations are used to ascertain the validity of the method and compare performances. Tests, carried out in different situations, verify that the SJE produces reasonably accurate local estimates of jitter, mostly in feebly perturbed signals. Comparison of the method is made with a widely adopted measurement of jitter, namely Praat. It is used as a reference system to analyze local statistics as well as global. Ideal glottal pulse sequences, synthetic speech sounds and recordings of connected speech are used to assess the reliability of each implementation. A particular conclusion for each signal is presented in the following sections.

6.1. Ideal pulse sequence

Applying the SJE in the ideal situation, the results agree with the prediction of the theoretical model. The exact measurement of jitter in samples, in the short-term, is possible. Specifically, the number of contour crossings corresponds to local jitter as explained in section 3.4. We have therefore ascertained that, when the ideal situation is given, the spectrum follows the spectral model previously studied. In addition, we have developed a system to compute the number of crossings between the harmonic and the inter-harmonic contours. Noise effects are also analyzed to reduce their influence on the SJE. For that purpose, we implement a noise rejection method. Other important factors, such as windowing analysis, are utilized to develop a suitable implementation for the ideal pulse sequence. Version I is therefore created based on the aforementioned analysis of the ideal situation.

In addition to the global data we obtained short-time measurements; local statistics prove that there is a high short-term correlation in the ideal situation. The implementations assessed, i.e. Version I and that of Vasilakis, enable discriminating feebly perturbed signals. The results show a noticeable improvement of Version I from that of Vasilakis in the ideal situation. Both methods, however, obtain reasonable errors in terms of jitter.

As far as the ideal situation is concerned, we have improved the anterior SJE developed by Vasilakis. Given that there are not spurious crossings, the heuristic crossing detector is useless. It is therefore more appropriate to use peak picking instead of assuming a theoretical position. The Vasilakis method is more susceptible to quantization noise, providing thus inaccurate harmonic detection. Peak picking is more suitable as it obtains peak amplitudes corresponding to the harmonics and inter-harmonics. This method, however, may also get into trouble as shown in section 3.5, but it is still more reliable in terms of accuracy. Generally speaking, Version I provides a higher correlation as well as greater accuracy than Vasilakis in the ideal situation.

6.2. Synthetic phonation

Synthetic speech sounds are studied to ascertain the validity of the mathematical model. Tests are performed with /a/, /i/, /u/, /ia/, /ai/ sounds with different fundamental frequencies, perturbations and noise. Additionally, the validity of the method is assessed by analyzing highly perturbed /a/ sounds. Sustained sounds are different from spike trains and thus, the spectral behavior is different from the model proposed in section 2.1. Even so, SJE enables discriminating feebly perturbed signals, mostly by Version I and Version II.

LPC-analysis is studied to convert the input signal into a spiky representation, which is more similar to a perturbed spike train than the original waveform. However, it does not provide a considerable improvement and it is therefore not as useful as we expected. Anyway, we proved that there is a noticeable correlation between methods whether the

LPC-analysis is applied or not, i.e. the obtained average jitter is almost the same in both situations.

Version I, Version II and Vasilakis provided reasonably accurate average jitter for weakly perturbed sounds. Version I and Version II are however more reliable with moderately perturbed sounds than Vasilakis. The analysis of a highly perturbed corpus shows that all the aforementioned implementations are unreliable for large jitter. It is therefore hopeless to use the SJE when jitter is strong. Furthermore, the results also evidence a slight improvement of Version I and Version II compared to the Vasilakis implementation.

6.3. Connected speech

Further evaluation of the SJE is performed using actual running speech recordings of a teacher's voice. Connected speech enables studying the evolving behavior of jitter while speaking. Since the pitch varies from cycle to cycle, local jitter estimation is required as implemented in the SJE. In addition to average jitter, the influence of the vocal overuse is al also analyzed comparing jitter estimates pre and post-lesson. Vocal dysperiodicities are also studied by means of a signal-to-dysperiodicity ratio (SDR). The pitch and the SDR show an increasing trend over time.

Version II enabled describing jitter in modal voices although there is a noticeable offset of about 40 µs. At least, there is a linear correlation between Praat and Version II as expected. Furthermore, jitter shows a decreasing trend contrary to the pitch, which is in accordance with their inverse relation. Regarding Vasilakis, the results are in disagreement with Praat and Version II. The Vasilakis implementation is therefore unreliable concerning real speech jitter estimation.

One may conjecture that the increasing pitch trend is a consequence of vocal overuse but that hypothesis is unproven. The cause of vocal disorders are often vocal overuse of the voice. Anyone who uses his voice excessively may develop a disorder related to vocal abuse mostly in the long term.

6.4. Limitations

Spectral methods for estimate jitter spectrally are being studied currently by scientists given that jitter is difficult to measure. Although jitter is spectrally observed there are limitations with regard to spectral methods. If we consider only the lower frequencies of a voiced speech segment, this may be closer to being predictable compared to its structure in higher frequencies. At high frequencies noise degrades the spectral response and the signal has no large energy. In this work, we increased the sampling frequency to more than 160 kHz to reduce the quantization step. Although we reduce the quantization error, a higher sampling frequency might not improve the accuracy of the estimation. This is a consequence of the unpredictable spectral behavior of vocal jitter at higher frequencies. The sampling frequency would therefore require a tradeoff between resolution and predictability to develop a suitable jitter estimator.

6.5. Future topics

Although our improvement of Vasilakis is slight, we consider it an encouraging result. The spectral methods are still researched and thus, further work is necessary in diverse directions. The spectral behavior still has to be thoroughly studied, mostly in higher frequencies, to predict the exact influence of jitter with regard to the harmonics and inter-harmonics. If the spectrum of a jittered speech sound is understood, it would be easier to improve the method for estimate jitter spectrally. In the short term, however, the goal is to develop a better implementation.

The fact that Version II enables to characterizethe behavior of jitter over time is a feature which is interesting to be examined in the future. It enables to extract other parameters that may be useful for detection of pathological voices apart from the possibility for improving voice quality assessment. Additional features can be therefore included along with jitter detection, which facilitate the detection of voice disorders.

Vocal overuse is another interesting topic which is currently being studied by phoneticians. It is motivating to analyze the pitch behavior over time for many medical

purposes. The exact evolving behavior, however, is so far quite unknown it is therefore worth studying vocal overuse.

7. References

In this work, the following bibliographic references were consulted:

[1] Arnold E. Aronson, Diane Bless. Clinical vocal disorders. 2009.

[2] Bruch, Jean M. and Kamani. Hoarseness in adults. 2009.

[3] J. Schoentgen. Vocal Cues of Disordered Voices. Journal of the Acoustical Society of America, 92:667–680, 2006.

[4] J. Schoentgen. Stochastic models of jitter. Journal of the Acoustical Society of America, 109(4):1631–1650, 2000.

[5] M. Vasilakis and Y. Stylianou. Spectral Based Short-Time Features for Voice Quality Assessment. Master thesis. 2009.

[6] James McClellan. Computer-Based Exercises for Signal Processing Using MATLAB 5. 2002.

[7] M. Vasilakis and Y. Stylianou. Voice Pathology Detection on Running Speech based on Short-Time Jitter Estimations. 2009.

[8] M. Vasilakis and Y. Stylianou. Spectral jitter modeling and estimation. Biomedical Signal Processing & Control 4. 2009.

[9] Ramirez, Robert W. The FFT. Fundamentals and concepts. Englewood Cliffs: Prentice-Hall, 1985.

[10] J. Schoentgen. Quelques observations concernant l'analyse du timbre de locuteurs professionnels. Réunion "Eclipse", 7-février-2011.

8. Appendix

8.1. Unperturbed spike train

```
function SpkT=SpikeTrain(N,Fs,fo)

t=0:1/Fs:N/Fs-1/Fs;
Nfft=2^14;
p=round(Fs/fo);
f = 0:Fs/Nfft:Fs-Fs/Nfft;
SpkT=zeros(1,N);
for i=1:N
      if mod(i,p)==0
            SpkT(i)=1;
      end
end

return SpkT
```

8.2. Ideal pulse sequence I

```
function SpkT=SpikeTrainPert(N,Fs,fo,e)

par=0;
t=0:1/Fs:N/Fs-1/Fs;
Nfft=2^14;
p=round(Fs/fo);
f = 0:Fs/Nfft:Fs-Fs/Nfft;
SpkT=zeros(1,N);
for i=1:N
    if mod(i-1,p)==0&&mod(par,2)==0
        SpkT(i)=1;
        par=1;
    elseif mod(i-1,p)==0&&mod(par,2)==1
        SpkT(i-e)=1;
        SpkT(i)=0;
        par=0;
    end
end

return SpkT
```

8.3. Ideal pulse sequence II

```
function SpkT=SpikeTrainRand(N,Fs,fo,r)

t=0:1/Fs:N/Fs-1/Fs;
Nfft=2^14;
p=round(Fs/fo);
signo=round(rand(1)*1);
f = 0:Fs/Nfft:Fs-Fs/Nfft;
SpkT=zeros(1,N);
for i=1:N
    if mod(i-1,p)==0
        if signo==0
            r=round(rand(1)*r);
            if i==1
                SpkT(1)=1;
            else
                SpkT(i-r)=1;
            end
        else
            r=round(rand(1)*r);
            SpkT(i+r)=1;
        end
    end
end

return SpkT
```

8.4. Harmonic detector

```
function H=HarDet(posact,Nfft,P,Xf,f,per)

H=0;
pic=0;
posH=1;
Xf=20*log10(abs(fft(frame,Nfft)));
Xf = Xf/sqrt(frame_len);
while posact<length(f)/2
    fmax=0;
    jmax=0;
    b=0;
    posP=1;
    for j=-round(Nfft/(per*P)):round(Nfft/(per*P))
        if (posact+j-1)>0&&Xf(posact+j)>Xf(posact+j-1)
        &&Xf(posact+j)>Xf(posact+j+1)
            pic(1,posP)=Xf(posact+j);
            pic(2,posP)=f(posact+j);
            pic(3,posP)=j;
            posP=posP+1;
            b=1;
        end
    end
    if b==1
        maxP=pic(1,1);
        fmax=pic(2,1);
        jmax=pic(3,1);
        for j=1:length(pic(1,:))
            if pic(1,j)>=maxP
                maxP=pic(1,j);
                fmax=pic(2,j);
                jmax=pic(3,j);
            end
        end
        H(1,posH)=maxP;
        H(2,posH)=fmax;
    else
        H(1,posH)=Xf(posact);
        H(2,posH)=f(posact);
    end
    posH=posH+1;
    posact=posact+jmax+round(Nfft/P);
end
```

8.5. Noise rejection method

```
function SpkT=NoisePic(SpkT)

SpkT=awgn(SpkT,SNR,'measured');
for i=2:length(SpkT)-1
    if SpkT(i)>SpkT(i+1)&&SpkT(i)>SpkT(i-1)&&SpkT(i)<0.2
        SpkT(i)=0;
    end
end
```

8.6. Crossings detector

```
function nCross=CrossCont(H,S)

a=0;
crossings=0;
j=1;
if length(S(1,:))<=length(H(1,:))
    fin=length(S(1,:));
else
    fin=length(H(1,:));
end
for i=1:fin
    if abs(H(1,i)-S(1,i))<=3
        crossings(j)=i;
        j=j+1;
        a=1;
    end
end
if a==1
    nCross=length(crossings);
else
    nCross=0;
end
while nCross>0
    p=round(length(H)/nCross);
    cont=zeros(1,nCross);
    sum=0;
    for i=0:nCross-1
        for k=1:length(crossings)
            if crossings(k)>=i*p&&crossings(k)<(i+1)*p
                cont(i+1)=1;
            end
        end
    end
    for j=1:length(cont)
        sum=sum+cont(j);
    end
    if sum==nCross
        break
    else
        nCross=nCross-1;
    end
end
```

8.7. Version I

```
function [jitter,time]=Definitiu(N,Fs,fo,SNR)

perm=4;
r=3;
index=1;
L=4;
Salto=2;
Nfft=2^14;
f = 0:Fs/Nfft:Fs-Fs/Nfft;
[SpkT,E]=SpktGen(N,Fs,fo,r,Nfft);
SpkT=awgn(SpkT,SNR,'measured');
SpkT=NoisePic(SpkT);
P=PitchDet(SpkT,Fs);
posact=PicCentr(SpkT,P);
start=posact-round(L*P/2);
while start<=N
    posact=1;
    final=start+round(L*P);
    if final>=N
        break
    end
    frame=SpkT(start:final);
    tf=start/Fs:1/Fs:final/Fs;
    h=triang(length(frame));
    frame=frame.*h';
    Xf=20*log10(abs(fft(frame,Nfft)));
    H=HarDet(posact,Nfft,P,Xf,f,perm);
    posact=posact+round(Nfft/(P*2));
    S=SubDet(posact,Nfft,P,Xf,f,perm);
    nCross=CrossCont(H,S);
    jitter(index)=nCross;
    time(index)=(start+final)/(Fs*2);
    start=start+Salto*P;
    index=index+1;
    Plot(E,tf,frame,f,Xf,H,S)
end

perc=Perc(jitter,E,Salto);
```

```
function Plot(E,tf,frame,f,Xf,H,S)

figure
E(3,:)=0.7*ones(1,length(E(1,:)));
subplot(2,1,1);
stem(tf,frame);grid on;
for i=1:length(E(1,:))
    text(E(2,i),E(3,i),num2str(E(1,i)));
end
xlabel('Time (s)');
ylabel('Amplitude');
subplot(2,1,2);
plot(f,Xf);grid on
xlabel('Frequency (Hz)');
ylabel('Amplitude');
figure
plot(H(2,:),H(1,:),'k+')
hold on
plot(S(2,:),S(1,:),'r.')
xlabel('Frequency (Hz)');
ylabel('20Log amplitude');
grid on
pause;
close all
```

```
function perc=Perc(jitter,E,Salto)

i=2;
pos=1;
while i<length(E(1,:))-1
    pert(pos)=abs(E(1,i-1)+E(1,i+1)-2*E(1,i));
    i=i+Salto;
    pos=pos+1;
end
pert=pert(2:length(pert));
jitter=jitter(1:length(pert));
cont=1;
tot=1;
for i=1:length(pert)
    if pert(i)==jitter(i)
        cont=cont+1;
    end
    tot=tot+1;
end
perc=cont/tot*100;
```

8.8. Vasilakis

```
function [ time jitter intersections_details ] = SJE(x, fs,
pitch_method, pitch_arg, frame_size, hop_size, win_type, plot_flag,
frame_starts, frame_ends)

time = [];
jitter = [];
intersections_details = {};
len = length(x);
if pitch_method == 1
    pitch_arg = fs/pitch_arg;
    frame_starts = round(1:hop_size*pitch_arg:len);
    frame_ends = round(frame_starts + frame_size*pitch_arg - 1);
    frame_ends(end) = len;
    P = pitch_arg*ones(size(frame_starts));
elseif pitch_method == 2
    pitch_arg(pitch_arg <= 0) = mean(pitch_arg(pitch_arg > 0));
    pitch_arg = fs./pitch_arg;
    frame_starts = round(1:hop_size*pitch_arg:len);
    frame_ends = round(frame_starts + frame_size*pitch_arg - 1);
    frame_ends(end) = len;
    P = pitch_arg;
elseif pitch_method == 3
    frame_starts(find(frame_starts < 1)) = 1;
    frame_ends(find(frame_ends > len)) = len;
    P = pitch_arg;
end
time = (frame_starts + frame_ends - 1)/(2*fs);
for i = 1:length(frame_starts)
    frame_start = frame_starts(i);
    frame_end = frame_ends(i);
    frame_len = frame_end - frame_start + 1;
    win = [];
    if win_type == 1
        win = hanning(frame_len);
    elseif  win_type == 2
        win = hamming(frame_len);
    elseif  win_type == 3
        win = ones(frame_len);
    end
    if nargout == 3
        intersections_details(i).win = win;
    end
    win = win ./ sum(win);
    wl = frame_len;
    nfft = 2^(ceil(log2(wl)));
    w = 0:pi/(nfft/2):pi-pi/(nfft/2);
    f = 0:fs/(nfft/2):fs-1/(nfft/2);
    F = abs(fft(x(frame_start:frame_end).*win, nfft));
    if nargout == 3
        intersections_details(i).winF = x(frame_start:frame_end).*win;
        intersections_details(i).F = F;
    end
```

```
    F = 20*log10(F + 0.0001);
    F = F(1:nfft/2);
    x_frame=x(frame_start:frame_end).*win;
    tf=frame_start/fs:1/fs:frame_end/fs;
    t = round((1:(nfft/2)/P(i):nfft/2));
    t_h = t(1:2:end);
    t_j = t(2:2:end);
    harmonic_peaks = F(t_h);
    jitter_peaks = F(t_j);
    harmonic_w = w(t_h);
    jitter_w = w(t_j);
    harmonic_interpolated_w = jitter_w;
    harmonic_interpolated_peaks = [];
    if length(harmonic_peaks) == length(jitter_peaks)
        harmonic_interpolated_peaks = [
        (harmonic_peaks(2:end) - harmonic_peaks(1:end-
      1)).*(harmonic_interpolated_w(1:end-1)'-harmonic_w(1:end-
      1)')./(harmonic_w(2:end)' - harmonic_w(1:end-1)') +
      harmonic_peaks(1:end-1);
        (harmonic_peaks(end) - harmonic_peaks(end-
      1))*(harmonic_interpolated_w(end)-harmonic_w(end-
      1))./(harmonic_w(end) - harmonic_w(end-1)) + harmonic_peaks(end-
      1)
        ];
    else
        harmonic_interpolated_peaks = [
        (harmonic_peaks(2:end) - harmonic_peaks(1:end-
      1)).*(harmonic_interpolated_w(1:end)'-harmonic_w(1:end-
      1)')./(harmonic_w(2:end)' - harmonic_w(1:end-1)') +
      harmonic_peaks(1:end-1)
        ];
    end
    jitter_interpolated_w = jitter_w;
    jitter_interpolated_peaks = jitter_peaks;

    db_threshold = 3;
    accepted_intersections = [];
    rejected_intersections = [];
    intersections_number = 0;
    hs = harmonic_interpolated_peaks(1);
    js = jitter_interpolated_peaks(1);
    current = 1;
    accepted_flag = 1;
    for k = 2:length(harmonic_interpolated_peaks)
        hp = harmonic_interpolated_peaks(k-1);
        jp = jitter_interpolated_peaks(k-1);
        hn = harmonic_interpolated_peaks(k);
        jn = jitter_interpolated_peaks(k);
        if (hp>=jp && hn<jn) || (hp<jp && hn>=jn)
            if accepted_flag == 1
                accepted_flag = 0;
            elseif accepted_flag == 0
                rejected_intersections = [rejected_intersections;wz];
            end
            current = k - 1;
            wz = ((harmonic_interpolated_peaks(current+1) -
```

```
      jitter_interpolated_peaks(current+1))*harmonic_interpolated_w(cu
      rrent) - (harmonic_interpolated_peaks(current) -
      jitter_interpolated_peaks(current))*harmonic_interpolated_w(curr
      ent+1))/((harmonic_interpolated_peaks(current+1) -
      jitter_interpolated_peaks(current+1)) -
      (harmonic_interpolated_peaks(current) -
      jitter_interpolated_peaks(current)));
        end
        if hs >= js && hn < jn && abs(hn - jn) > db_threshold
            intersections_number = intersections_number + 1;
            hs = hn;
            js = jn;
            accepted_intersections = [accepted_intersections;wz];
            accepted_flag = 1;
        elseif hs < js && hn >= jn && abs(hn - jn) > db_threshold
            intersections_number = intersections_number + 1;
            hs = hn;
            js = jn;
            accepted_intersections = [accepted_intersections;wz];
            accepted_flag = 1;
        end
    end
    final_intersections = [];
    if (intersections_number > 0)
        max_intersections = intersections_number;
        max_intersections_flag = 1;
        intersections_number = 0;
        while max_intersections_flag == 1
            current_intersections_flag = 1;
            current_intersections = 1;
            current_final_intersections = zeros(1, max_intersections);
            while (current_intersections <= max_intersections) &&
      (current_intersections_flag == 1)
                current_center = (current_intersections -
      0.5)*pi/max_intersections;
                current_left = current_center -
      0.25*pi/max_intersections;
                current_right = current_center +
      0.25*pi/max_intersections;
                if ~isempty(find(accepted_intersections >=
current_left & accepted_intersections <= current_right, 1))
                    current_final_intersections(current_intersections)
= mean(accepted_intersections(accepted_intersections >= current_left &
accepted_intersections <= current_right));
                    current_intersections = current_intersections + 1;
                else
                    current_intersections_flag = 0;
                end
            end
            if current_intersections_flag == 1
                final_intersections = current_final_intersections;
                intersections_number = max_intersections;
                max_intersections_flag = 0;
            else
                max_intersections = max_intersections - 1;
            end
```

```
        end
    end
    jitter(i) = intersections_number*2*(10^6)/fs;
    if nargout == 3
        intersections_details(i).FdB = F;
        intersections_details(i).interpolated_w =
      harmonic_interpolated_w;
        intersections_details(i).harmonic_interpolated_peaks =
      harmonic_interpolated_peaks;
        intersections_details(i).jitter_interpolated_peaks =
      jitter_interpolated_peaks;
        intersections_details(i).accepted_intersections =
      accepted_intersections;
        intersections_details(i).rejected_intersections =
      rejected_intersections;
        intersections_details(i).final_intersections =
      final_intersections;
    end
    if plot_flag
        hold off;
        plot(0:frame_end - frame_start, x(frame_start:frame_end), 'k-
      ');
        legend(sprintf('P = %d samples\nt = %f s', P(i), time(i)));
        hold off;
        figure
        plot(w*fs/(2000*pi), F, 'k-');
        hold on;
        hh_w=[];
        hh_peaks=[];
        for jj=1:length(harmonic_w),
            hh_w(jj*2-1)=harmonic_w(jj);
            hh_peaks(jj*2-1)=harmonic_peaks(jj);
        end
        for jj=1:length(harmonic_interpolated_w)
            hh_w(jj*2)=harmonic_interpolated_w(jj);
            hh_peaks(jj*2)=harmonic_interpolated_peaks(jj);
        end
        plot(hh_w*fs/(2000*pi), hh_peaks, 'b-');
        plot(jitter_w*fs/(2000*pi), jitter_peaks, 'r-');

        stem(rejected_intersections*fs/(2000*pi),
      min(F)*ones(size(rejected_intersections)), 'r');
        stem(accepted_intersections*fs/(2000*pi),
      min(F)*ones(size(accepted_intersections)), 'b');
        stem(final_intersections*fs/(2000*pi),
      min(F)*ones(size(final_intersections)), 'g*');
        legend('log magnitude spectrum', 'harmonic part', 'subharmonic
      part', sprintf('rejected = %d', length(rejected_intersections)),
      sprintf('accepted = %d', length(accepted_intersections)),
      sprintf('final = %d', intersections_number));
        title(sprintf('jitter = %g \\mus', jitter(i)));
        hold off;
        pause;
    end
end
return;
```

8.9. Version II

```
function [jitter,time]=DefinitiuProva(N,Fs,fo,SNR,SpkT)

perm=10;
index=1;
L=4;
Salto=1;
Nfft=2^14;
f = 0:Fs/Nfft:Fs-Fs/Nfft;
P=round(N/fo);
posact=PicCentr(SpkT,P);
start=1;
while start<=N
    posact=1;
    final=start+round(L*P);
    if final>=N
        break
    end
    frame=SpkT(start:final);
    tf=start/Fs:1/Fs:final/Fs;
    h=hanning(length(frame));
    frame=frame.*h';
    Xf=20*log10(abs(fft(frame,Nfft)));
    H=HarDet(posact,Nfft,P,Xf,f,perm);
    posact=posact+round(Nfft/(P*2));
    S=SubDet(posact,Nfft,P,Xf,f,perm);
[nCross,rejected_intersections,accepted_intersections,final_intersecti
ons]=CrossCont2(H,S,Fs);
    jitter(index)=nCross*2*1e6/Fs;

    time(index)=(start+final)/(Fs*2);
    start=start+Salto*P;
    index=index+1;
end
```

```
function
[intersections_number,rejected_intersections,accepted_intersections,fi
nal_intersections]=CrossCont2(H,S,Fs)

harmonic_interpolated_w=H(2,:)*pi/Fs;
jitter_interpolated_w=S(2,:)*pi/Fs;
harmonic_interpolated_peaks=H(1,:);
jitter_interpolated_peaks=S(1,:);
db_threshold = 0;
accepted_intersections = [];
rejected_intersections = [];
intersections_number = 0;
hs = harmonic_interpolated_peaks(1);
js = jitter_interpolated_peaks(1);
current = 1;
accepted_flag = 1;
if
length(harmonic_interpolated_peaks)<length(jitter_interpolated_peaks)
        minim=length(harmonic_interpolated_peaks);
else
        minim=length(jitter_interpolated_peaks);
end
for k = 2:minim
      hp = harmonic_interpolated_peaks(k-1);
      jp = jitter_interpolated_peaks(k-1);
      hn = harmonic_interpolated_peaks(k);
      jn = jitter_interpolated_peaks(k);
      if (hp>=jp && hn<jn) || (hp<jp && hn>=jn)
            if accepted_flag == 1
                accepted_flag = 0;
            elseif accepted_flag == 0
                rejected_intersections = [rejected_intersections;wz];
            end
            current = k - 1;
            wz = ((harmonic_interpolated_peaks(current+1) -
jitter_interpolated_peaks(current+1))*harmonic_interpolated_w(current)
- (harmonic_interpolated_peaks(current) -
jitter_interpolated_peaks(current))*harmonic_interpolated_w(current+1)
)/((harmonic_interpolated_peaks(current+1) -
jitter_interpolated_peaks(current+1)) -
(harmonic_interpolated_peaks(current) -
jitter_interpolated_peaks(current)));
        end
        if hs >= js && hn < jn && abs(hn - jn) > db_threshold
            intersections_number = intersections_number + 1;
            hs = hn;
            js = jn;
            accepted_intersections = [accepted_intersections;wz];
            accepted_flag = 1;
        elseif hs < js && hn >= jn && abs(hn - jn) > db_threshold
            intersections_number = intersections_number + 1;
            hs = hn;
            js = jn;
            accepted_intersections = [accepted_intersections;wz];
            accepted_flag = 1;
```

```
        end
    end
    final_intersections = [];
    if (intersections_number > 0)
        max_intersections = intersections_number;
        max_intersections_flag = 1;
        intersections_number = 0;
        while max_intersections_flag == 1
            current_intersections_flag = 1;
            current_intersections = 1;
            current_final_intersections = zeros(1, max_intersections);
            while (current_intersections <= max_intersections) &&
(current_intersections_flag == 1)
                current_center = (current_intersections -
0.5)*pi/max_intersections;
                current_left = current_center -
0.25*pi/max_intersections;
                current_right = current_center +
0.25*pi/max_intersections;
                if ~isempty(find(accepted_intersections >=
current_left & accepted_intersections <= current_right, 1))
                    current_final_intersections(current_intersections)
= mean(accepted_intersections(accepted_intersections >= current_left &
accepted_intersections <= current_right));
                    current_intersections = current_intersections + 1;
                else
                    current_intersections_flag = 0;
                end
            end
            if current_intersections_flag == 1
                final_intersections = current_final_intersections;
                intersections_number = max_intersections;
                max_intersections_flag = 0;
            else
                max_intersections = max_intersections - 1;
            end
        end
    end
```

8.10. LPC-analysis

```
function [Spkt_LPC]=LPC(SpkT,order)

a=lpc(SpkT,order);
Spkt_LPC=filter(a,1,SpkT);
```

8.11. Unit pulses. Average jitter.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Praat**	41	24	23	39	27	24	44	29	29	80	42	42
	Version I	51	23	29	39	33	24	46	31	31	78	40	42
	Vasilakis	70	31	29	58	39	24	63	31	28	99	47	37
0.15	**Praat**	106	79	69	106	67	75	118	77	73	124	98	78
	Version I	123	79	79	122	77	80	107	88	75	133	97	78
	Vasilakis	171	85	84	160	89	95	162	97	91	182	105	106
0.3	**Praat**	214	165	157	200	177	158	230	164	135	258	184	125
	Version I	224	150	152	203	154	162	236	164	135	263	172	133
	Vasilakis	284	149	182	276	144	207	334	160	137	345	191	190
0.4	**Praat**	285	265	185	286	246	184	286	252	212	311	220	166
	Version I	316	238	179	264	231	171	363	228	181	347	247	159
	Vasilakis	392	228	195	379	234	238	379	208	252	415	200	227

Table 8. UP average jitter in µs for sound /a/.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Praat**	34	26	22	62	42	29	112	132	63	272	254	177
	Version I	38	25	24	61	43	33	115	128	68	246	204	155
	Vasilakis	62	31	32	87	52	35	134	125	69	306	206	191
0.15	**Praat**	119	98	62	131	98	65	159	153	96	273	251	188
	Version I	124	100	69	137	98	72	166	140	101	242	208	178
	Vasilakis	170	101	82	157	95	88	216	159	119	274	233	192
0.3	**Praat**	181	189	150	213	189	151	226	201	147	308	329	170
	Version I	200	164	134	211	168	136	246	183	146	293	264	175
	Vasilakis	276	166	198	293	159	202	340	197	202	421	309	224
0.4	**Praat**	364	235	163	315	283	169	277	314	191	328	334	240
	Version I	306	222	164	267	245	163	265	289	182	268	264	202
	Vasilakis	413	240	260	448	259	258	351	295	249	461	266	264

Table 9. UP average jitter in µs for sound /i/.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Praat**	33	27	26	41	29	28	68	46	53	114	91	94
	Version I	36	29	29	49	31	31	69	49	55	112	86	94
	Vasilakis	58	32	29	69	31	25	84	49	53	149	89	104
0.15	**Praat**	107	84	70	95	91	77	144	109	82	130	101	118
	Version I	120	93	70	107	94	77	143	104	82	132	95	125
	Vasilakis	145	100	84	150	100	86	188	102	108	176	105	148
0.3	**Praat**	202	213	136	213	159	140	199	165	145	236	166	152
	Version I	231	209	141	227	154	144	214	170	141	215	178	148
	Vasilakis	325	216	170	280	175	179	260	183	193	292	198	172
0.4	**Praat**	295	226	159	266	229	211	250	240	185	299	229	187
	Version I	251	230	145	251	215	190	263	193	159	289	208	181
	Vasilakis	383	244	249	328	228	251	364	202	237	406	200	231

Table 10. UP average jitter in µs for sound /u/.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Praat**	40	33	25	44	40	25	78	52	48	164	127	98
	Version I	67	37	28	64	50	28	95	54	50	160	122	86
	Vasilakis	99	47	38	97	58	33	118	60	62	205	136	96
0.15	**Praat**	106	103	69	137	78	63	132	119	73	194	199	118
	Version I	142	97	82	138	84	62	145	109	74	185	163	113
	Vasilakis	193	103	98	198	98	89	178	107	99	265	166	156
0.3	**Praat**	225	171	146	237	164	136	218	192	148	324	266	168
	Version I	227	183	144	242	147	141	304	196	133	286	228	145
	Vasilakis	300	175	161	296	139	187	275	186	220	452	212	242
0.4	**Praat**	250	247	158	241	266	160	247	228	177	365	274	220
	Version I	304	202	167	267	231	155	254	237	165	298	314	178
	Vasilakis	411	209	207	396	244	195	296	208	264	484	251	306

Table 11. UP average jitter in µs for sound /ai/.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Praat**	38	34	27	36	35	28	65	56	34	102	112	93
	Version I	46	42	31	33	45	32	62	61	40	98	111	84
	Vasilakis	100	54	37	80	55	36	117	64	51	161	108	79
0.15	**Praat**	89	84	63	96	81	57	113	105	81	144	168	141
	Version I	100	110	72	98	100	66	116	117	82	140	153	120
	Vasilakis	179	114	101	180	111	88	199	120	101	223	153	142
0.3	**Praat**	200	159	143	203	174	140	211	163	133	240	230	159
	Version I	207	161	147	203	160	146	193	172	138	256	227	142
	Vasilakis	317	157	174	344	166	162	371	186	200	342	236	235
0.4	**Praat**	235	259	191	265	222	154	292	235	196	297	248	262
	Version I	246	216	188	292	225	148	309	219	185	290	245	210
	Vasilakis	421	227	219	391	224	178	367	237	282	450	234	299

Table 12. UP average jitter in µs for sound /ia/.

8.12. Unit pulses. Cross-correlation.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.7912	0.5242	0.5822	0.6710	0.5110	0.6433	0.8928	0.7572	0.6095	0.9176	0.6460	0.8955
0.15	0.7855	0.7345	0.8043	0.6863	0.7907	0.8136	0.6722	0.6441	0.8652	0.8520	0.7596	0.8376
0.3	0.6671	0.6289	0.7594	0.6224	0.5507	0.7002	0.0930	0.7796	0.7893	0.1418	0.7473	0.5706
0.4	0.5367	0.0414	0.7307	0.5819	0.6839	0.5448	0.0064	0.6225	0.7119	0.0465	0.2400	0.3832

Table 13. UP crossed correlation between Version I and Praat for sound /a/. Average= 0.6222.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.6414	0.5300	0.6105	0.8176	0.7893	0.7649	0.8961	0.9264	0.8371	0.7763	0.7969	0.8538
0.15	0.8472	0.7158	0.7662	0.8505	0.6357	0.8017	0.7541	0.7930	0.7989	0.8114	0.7512	0.7336
0.3	0.5896	0.6432	0.6302	0.5858	0.6299	0.6420	0.6183	0.5370	0.6396	0.5253	0.7742	0.5575
0.4	0.7356	0.6602	0.4819	0.4248	0.5375	0.5493	0.6053	0.7092	0.5624	0.5246	0.5839	0.7045

Table 14. UP crossed correlation between Version I and Praat for sound /i/. Average= 0.6865.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.7334	0.6385	0.6639	0.7302	0.7195	0.6877	0.9258	0.7991	0.8707	0.8575	0.8376	0.9062
0.15	0.8059	0.7591	0.8983	0.7379	0.8170	0.8495	0.8071	0.7464	0.7810	0.8106	0.7606	0.7202
0.3	0.3583	0.6433	0.6706	0.7238	0.4635	0.6099	0.7196	0.5943	0.6218	0.7793	0.6583	0.7756
0.4	0.5341	0.4593	0.5207	0.5914	0.5461	0.7333	0.4965	0.7119	0.7064	0.5172	0.4284	0.5935

Table 15. UP crossed correlation between Version I and Praat for sound /u/. Average= 0.6942.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.3768	0.8109	0.5948	0.5051	0.6289	0.6026	0.6770	0.7699	0.9143	0.8154	0.8305	0.8705
0.15	0.4980	0.7489	0.7532	0.5449	0.6578	0.8103	0.5556	0.7000	0.7674	0.6629	0.8836	0.7206
0.3	0.5930	0.5555	0.7979	0.5522	0.6613	0.6387	0.1349	0.5460	0.4886	0.5367	0.8362	0.4808
0.4	0.2910	0.5912	0.6329	0.4163	0.4100	0.7107	0.3235	0.1339	0.4763	0.5078	0.0148	0.6902

Table 16. UP crossed correlation between Version I and Praat for sound /ai/. Average= 0.5983.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.3493	0.6577	0.8323	0.2573	0.7402	0.8482	0.4060	0.8525	0.8043	0.3511	0.8321	0.9285
0.15	0.1999	0.6592	0.6742	0.1867	0.7477	0.6945	0.2464	0.7418	0.7706	0.6297	0.8773	0.8973
0.3	0.1952	0.7470	0.7519	0.6072	0.5127	0.7689	0.6422	0.5438	0.6146	0.7045	0.7380	0.5571
0.4	0.2753	0.6794	0.6935	0.1798	0.5861	0.7790	0.2940	0.7135	0.3401	0.5060	0.2701	0.8539

Table 17. UP crossed correlation between Version I and Praat for sound /ai/. Average= 0.5987.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.4785	0.2953	0.4188	0.3859	0.3549	0.3306	0.6300	0.6715	0.2172	0.6011	0.5254	0.6412
0.15	0.4309	0.6236	0.8043	0.3465	0.5125	0.8136	0.3647	0.5153	0.8652	0. 609	0.4909	0.8376
0.3	0.3429	0.5012	0.5795	0.4114	0.4686	0.3937	0.0260	0.5075	0.5358	0.2378	0.5186	0.2885
0.4	0.2961	0.1528	0.4481	0.5741	0.5176	0.3644	0.0707	0.4093	0.4934	0.1493	0.3743	0.3470

Table 18. UP crossed correlation between Vasilakis and Praat for sound /a/. Average= 0.4347.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.2084	0.2446	0.2382	0.5898	0.4319	0.4670	0.6066	0.7316	0.4158	0.7197	0.5950	0.5312
0.15	0.4128	0.4411	0.4559	0.5518	0.3449	0.3524	0.4124	0.6187	0.5750	0.5866	0.6447	0.5213
0.3	0.4716	0.5216	0.3698	0.3638	0.4400	0.4574	0.2097	0.4154	0.3206	0.3396	0.7305	0.2304
0.4	0.4576	0.4621	0.4565	0.3587	0.4134	0.4595	0.3246	0.2991	0.4181	0.4617	0.5189	0.4039

Table 19. UP crossed correlation between Vasilakis and Praat's for sound /i/. Average= 0.4500.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.4189	0.4554	0.3534	0.3703	0.4593	0.5244	0.6454	0.5258	0.6247	0.6784	0.6375	0.5917
0.15	0.5777	0.4993	0.4618	0.4357	0.5571	0.5026	0.5145	0.4068	0.4595	0.4888	0.3770	0.3986
0.3	0.0764	0.3703	0.4616	0.4301	0.2734	0.4562	0.5515	0.3422	0.3907	0.5327	0.4361	0.5112
0.4	0.3797	0.2214	0.4346	0.3548	0.3172	0.6357	0.2960	0.5242	0.5367	0.3846	0.4165	0.2874

Table 20. UP crossed correlation between Vasilakis and Praat for sound /u/. Average= 0.4497.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.2778	0.6053	0.1405	0.3103	0.5186	0.2218	0.5194	0.5219	0.7518	0.5748	0.7538	0.7858
0.15	0.3762	0.6778	0.3643	0.3654	0.4555	0.5842	0.2689	0.5354	0.3980	0.5867	0.7412	0.5910
0.3	0.4100	0.3148	0.3888	0.2176	0.4648	0.3997	0.0934	0.4233	0.4542	0.4982	0.6229	0.4685
0.4	0.1930	0.5531	0.3854	0.3387	0.2204	0.4602	0.2415	0.0995	0.4822	0.5729	0.0195	0.5662

Table 21. UP crossed correlation between Vasilakis and Praat for sound /u/. Average= 0.4336.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	0.4853	0.1859	0.6482	0.4948	0.3037	0.5254	0.4321	0.4657	0.6953	0.2804	0.6588	0.8006
0.15	0.1440	0.3090	0.4319	0.1721	0.4703	0.4165	0.1888	0.5342	0.5698	0.4885	0.6637	0.6780
0.3	0.3804	0.3418	0.5814	0.3397	0.4353	0.5263	0.6601	0.2919	0.3396	0.4819	0.3267	0.3935
0.4	0.2771	0.4479	0.4921	0.1602	0.2562	0.5599	0.2503	0.4596	0.4652	0.3283	0.3262	0.6994

Table 22. UP crossed correlation between Vasilakis and Praat for sound /ia/. Average 0.4347.

8.13. Unit pulses. Average error.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	10	1	6	0	6	0	2	2	2	2	2	0
0.15	17	0	10	16	10	5	11	11	2	9	1	0
0.3	10	15	5	3	23	4	6	0	0	5	12	8
0.4	31	27	6	22	15	13	77	24	31	36	27	7

Table 23. UP jitter error in µs between Version I and Praat for sound /a/. Average= 11.0833 µs.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	3	1	1	1	0	4	3	4	6	25	50	22
0.15	5	2	6	6	0	7	7	13	6	30	43	10
0.3	19	25	16	2	22	15	20	17	0	16	65	5
0.4	58	13	1	48	37	6	12	25	9	60	71	39

Table 24. UP jitter error in µs between Version I and Praat for sound /i/. Average= 17.8942 µs.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	3	1	3	8	2	3	0	2	2	3	5	0
0.15	13	9	0	12	3	0	1	5	0	2	7	7
0.3	29	4	6	14	5	4	15	4	3	20	12	4
0.4	44	4	13	15	14	21	13	47	26	10	21	7

Table 25. UP jitter error in µs between Version I and Praat for sound /u/. Average= 9.3422µs.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	27	4	3	19	9	4	17	2	2	4	6	12
0.15	35	6	13	1	6	1	13	10	1	9	36	5
0.3	1	12	2	5	17	6	85	4	14	38	38	22
0.4	54	45	9	26	35	4	7	9	12	67	39	42

Table 26. UP jitter error in µs between Version I and Praat for sound /ai/. Average= 17.4801µs.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	27	4	3	19	9	4	17	2	2	4	6	12
0.15	35	6	13	1	6	1	13	10	1	9	36	5
0.3	1	12	2	5	17	6	85	4	14	38	38	22
0.4	54	45	9	26	35	4	7	9	12	67	39	42

Table 27. UP jitter error in µs between Version I and Praat for sound /ia/. Average= 10.2567 µs.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	29	7	6	19	12	0	19	2	1	19	5	5
0.15	65	6	15	54	22	20	44	20	18	58	7	28
0.3	70	16	25	76	33	49	104	4	2	87	7	65
0.4	107	37	10	93	12	54	93	44	40	104	20	61

Table 28. UP jitter error in µs between Vasilakis and Praat for sound /a/. Average= 35.2917 µs.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	28	5	9	25	10	5	22	7	6	34	48	14
0.15	51	3	19	26	3	23	57	6	23	1	18	4
0.3	94	23	48	80	30	51	114	3	56	113	20	54
0.4	49	5	97	133	24	89	74	19	58	132	68	24

Table 29. UP jitter error in µs between Vasilakis and Praat for sound /i/. Average= 39.7304 µs.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	25	5	3	28	2	3	16	3	1	34	2	11
0.15	38	17	14	55	9	9	44	7	26	45	4	29
0.3	123	2	34	67	16	39	61	17	48	56	33	20
0.4	89	18	91	63	1	40	114	37	52	107	29	44

Table 30. UP jitter error in µs between Vasilakis and Praat for sound /u/. Average 33.8848 µs.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	59	4	3	53	9	4	40	2	2	41	6	12
0.15	87	6	13	61	6	1	47	10	1	70	36	5
0.3	74	12	2	59	26	6	56	6	14	128	54	22
0.4	162	45	9	155	35	4	49	9	12	119	39	42

Table 31. UP jitter error in µs between Vasilakis and Praat for sound /ai/. Average 45.6136 µs.

J\b	0			1.1			3.1			6.1		
	100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	62	20	10	43	20	8	52	8	17	60	4	14
0.15	90	30	38	85	30	31	86	15	21	79	14	1
0.3	117	2	31	141	9	23	161	23	68	102	6	76
0.4	186	32	28	127	1	24	75	2	86	153	14	37

Table 32. UP jitter error in µs between Vasilakis and Praat for sound /ia/. Average 49.2107 µs.

8.14. Synthetic phonation. Average jitter.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Pulses**	**41**	**24**	**23**	**39**	**27**	**24**	**44**	**29**	**29**	**80**	**42**	**43**
	Praat	**43**	**24**	**23**	**39**	**27**	**24**	**46**	**29**	**29**	**83**	**42**	**44**
	Greek	69	45	39	93	71	55	122	101	81	169	126	99
	GreekLPC	63	42	37	88	69	51	117	90	73	157	112	92
	Mine	69	51	37	100	85	67	141	126	92	183	149	116
	MineLPC	68	51	36	101	83	67	143	118	89	186	153	116
	Mix	57	40	33	79	63	46	99	78	65	133	95	81
0.15	**Pulses**	**106**	**79**	**68**	**106**	**67**	**75**	**118**	**77**	**72**	**124**	**98**	**78**
	Praat	**107**	**78**	**67**	**108**	**67**	**73**	**120**	**75**	**72**	**123**	**96**	**80**
	Greek	116	101	75	129	110	83	152	135	91	186	174	114
	GreekLPC	112	83	61	121	87	72	142	116	85	169	138	102
	Mine	111	98	66	133	115	95	173	146	109	182	165	139
	MineLPC	117	98	65	136	106	91	175	146	107	194	166	141
	Mix	87	72	47	106	74	61	122	101	75	149	117	84
0.3	**Pulses**	**214**	**165**	**158**	**200**	**177**	**158**	**230**	**164**	**135**	**258**	**184**	**124**
	Praat	**214**	**165**	**158**	**200**	**175**	**158**	**228**	**161**	**135**	**256**	**181**	**126**
	Greek	177	149	124	201	176	146	226	182	135	249	216	155
	GreekLPC	171	134	102	179	154	132	229	167	122	245	201	147
	Mine	174	164	143	180	175	147	218	181	145	256	217	175
	MineLPC	173	159	136	176	177	141	225	178	150	264	220	178
	Mix	125	104	75	139	120	106	186	128	96	191	157	122
0.4	**Pulses**	**285**	**265**	**185**	**286**	**246**	**183**	**286**	**252**	**213**	**311**	**220**	**167**
	Praat	**280**	**261**	**184**	**286**	**251**	**184**	**284**	**269**	**211**	**310**	**221**	**166**
	Greek	213	193	133	223	219	154	265	231	166	287	245	165
	GreekLPC	203	175	111	219	201	143	262	205	159	275	222	162
	Mine	223	214	142	225	205	158	240	220	174	279	238	193
	MineLPC	219	204	144	226	196	153	250	220	174	285	234	187
	Mix	148	139	81	158	157	113	191	155	123	200	167	123

Table 33. SP average jitter in µs for sound /a/.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Pulses**	**34**	**26**	**22**	**62**	**42**	**29**	**112**	**132**	**62**	**272**	**254**	**177**
	Praat	**35**	**27**	**23**	**64**	**42**	**30**	**111**	**131**	**61**	**271**	**259**	**175**
	Vasilakis	65	49	41	96	74	55	135	97	82	181	125	99
	VasilakisLPC	61	45	28	93	72	48	122	90	77	158	121	94
	Mine	70	53	40	109	80	64	145	111	96	203	145	113
	MineLPC	70	52	44	108	81	67	146	118	90	207	155	115
	Mix	64	44	29	83	62	47	102	87	66	145	107	79
0.15	**Pulses**	**119**	**98**	**63**	**131**	**98**	**65**	**159**	**153**	**95**	**273**	**251**	**190**
	Praat	**118**	**98**	**62**	**133**	**98**	**66**	**157**	**153**	**95**	**273**	**252**	**190**
	Vasilakis	112	90	72	139	100	79	163	123	97	211	154	123
	VasilakisLPC	108	83	59	125	95	70	156	112	87	203	136	101
	Mine	122	101	75	140	104	84	172	122	107	204	159	133
	MineLPC	114	89	66	138	100	86	175	132	110	212	169	135
	Mix	100	72	45	108	83	65	148	97	75	140	117	96
0.3	**Pulses**	**181**	**189**	**150**	**213**	**189**	**151**	**226**	**201**	**146**	**308**	**329**	**173**
	Praat	**183**	**187**	**150**	**214**	**190**	**150**	**223**	**203**	**149**	**305**	**327**	**174**
	Vasilakis	162	139	112	203	152	124	222	170	134	262	197	157
	VasilakisLPC	148	129	98	198	145	110	219	156	125	259	192	140
	Mine	177	146	117	190	152	128	199	160	126	249	177	160
	MineLPC	177	145	116	194	154	128	207	170	135	260	204	170
	Mix	142	154	78	160	120	98	171	107	113	193	138	117
0.4	**Pulses**	**364**	**235**	**163**	**315**	**283**	**170**	**277**	**314**	**190**	**328**	**334**	**240**
	Praat	**367**	**235**	**164**	**310**	**282**	**170**	**289**	**319**	**186**	**346**	**333**	**236**
	Vasilakis	200	164	141	249	188	155	241	214	157	266	217	182
	VasilakisLPC	193	140	115	244	192	140	250	213	155	265	208	172
	Mine	220	159	131	221	188	138	245	201	153	251	207	166
	MineLPC	233	168	130	232	198	137	244	216	168	286	229	202
	Mix	165	167	133	179	153	126	190	111	121	208	155	85

Table 34. SP average jitter in µs for sound /i/.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Pulses**	**33**	**27**	**26**	**41**	**29**	**28**	**68**	**46**	**53**	**114**	**91**	**94**
	Praat	**32**	**27**	**26**	**42**	**29**	**28**	**69**	**46**	**53**	**122**	**93**	**95**
	Vasilakis	77	52	44	99	72	63	134	104	83	199	131	117
	VasilakisLPC	70	48	34	92	70	54	115	97	77	175	123	93
	Mine	74	50	47	117	80	72	146	113	90	188	144	123
	MineLPC	76	50	41	104	78	72	146	115	96	185	148	128
	Mix	64	44	29	83	62	47	102	87	66	145	107	79
0.15	**Pulses**	**107**	**84**	**69**	**95**	**91**	**77**	**144**	**109**	**82**	**130**	**101**	**118**
	Praat	**105**	**84**	**69**	**96**	**91**	**77**	**145**	**107**	**81**	**131**	**100**	**120**
	Vasilakis	161	90	73	158	106	93	206	129	98	224	167	139
	VasilakisLPC	118	82	58	133	94	75	168	114	92	176	146	112
	Mine	126	93	73	141	112	86	197	122	109	206	152	139
	MineLPC	125	84	66	144	114	85	192	129	112	212	164	146
	Mix	120	72	45	108	83	65	140	97	75	100	117	96
0.3	**Pulses**	**202**	**213**	**136**	**213**	**159**	**140**	**199**	**165**	**146**	**236**	**166**	**152**
	Praat	**202**	**212**	**134**	**219**	**162**	**138**	**198**	**162**	**150**	**235**	**164**	**151**
	Vasilakis	223	217	122	224	162	136	247	152	160	277	183	166
	VasilakisLPC	188	189	98	203	150	121	218	144	136	241	169	142
	Mine	208	173	112	193	141	121	228	143	140	243	145	155
	MineLPC	202	198	118	202	146	135	230	148	156	246	163	172
	Mix	142	154	78	160	120	98	171	107	113	193	138	117
0.4	**Pulses**	**295**	**226**	**157**	**266**	**229**	**213**	**250**	**240**	**185**	**299**	**229**	**189**
	Praat	**309**	**223**	**159**	**265**	**225**	**215**	**246**	**244**	**191**	**295**	**228**	**191**
	Vasilakis	228	238	134	263	206	170	281	159	176	288	218	187
	VasilakisLPC	226	223	117	238	192	157	257	133	151	286	200	174
	Mine	241	218	133	257	175	158	261	157	155	284	183	176
	MineLPC	244	230	137	261	185	168	255	158	170	300	203	191
	Mix	165	167	85	179	153	126	190	111	121	208	155	133

Table 35. SP average jitter in µs for sound /u/.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Pulses**	**40**	**33**	**25**	**44**	**40**	**25**	**78**	**52**	**49**	**164**	**127**	**98**
	Praat	**35**	**27**	**23**	**64**	**42**	**30**	**111**	**131**	**61**	**271**	**259**	**175**
	Vasilakis	71	49	41	91	73	61	128	98	82	179	135	104
	VasilakisLPC	66	46	34	89	72	52	122	91	75	161	112	96
	Mine	74	59	46	120	86	72	138	114	96	192	165	124
	MineLPC	79	62	45	123	83	73	148	119	96	200	171	132
	Mix	61	42	31	81	66	49	103	81	65	132	93	82
0.15	**Pulses**	**106**	**103**	**69**	**137**	**78**	**64**	**132**	**119**	**73**	**194**	**199**	**118**
	Praat	**118**	**98**	**62**	**133**	**98**	**66**	**157**	**153**	**95**	**273**	**252**	**190**
	Vasilakis	127	96	76	142	98	84	163	124	99	205	173	128
	VasilakisLPC	115	86	58	135	90	71	152	113	87	181	147	108
	Mine	137	99	77	155	101	87	172	126	105	196	181	134
	MineLPC	135	112	84	167	112	91	175	138	108	206	179	135
	Mix	89	71	48	114	77	60	126	93	71	145	123	86
0.3	**Pulses**	**225**	**171**	**145**	**237**	**164**	**136**	**218**	**192**	**147**	**324**	**266**	**168**
	Praat	**183**	**187**	**150**	**214**	**190**	**150**	**223**	**203**	**149**	**305**	**327**	**174**
	Vasilakis	199	143	117	213	148	131	238	195	145	256	210	163
	VasilakisLPC	185	135	100	204	134	118	228	186	129	252	188	143
	Mine	187	157	130	200	152	127	239	194	154	257	213	163
	MineLPC	201	158	130	205	155	122	250	196	151	259	211	166
	Mix	130	97	80	158	107	99	165	139	104	191	138	117
0.4	**Pulses**	**250**	**247**	**160**	**241**	**266**	**160**	**247**	**228**	**178**	**365**	**274**	**220**
	Praat	**367**	**235**	**164**	**310**	**282**	**170**	**289**	**319**	**186**	**346**	**333**	**236**
	Vasilakis	203	165	137	252	207	149	240	205	177	274	242	180
	VasilakisLPC	197	161	113	235	198	135	224	194	168	267	230	168
	Mine	222	205	139	232	216	143	223	206	175	266	227	200
	MineLPC	231	187	147	234	225	152	243	212	174	270	222	204
	Mix	139	115	86	183	144	101	164	143	128	193	162	126

Table 36. SP average jitter in µs for sound /ai/.

J\b		0			1.1			3.1			6.1		
		100	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**	**100**	**120**	**140**
0.05	**Pulses**	**38**	**34**	**27**	**36**	**35**	**28**	**65**	**56**	**34**	**102**	**112**	**93**
	Praat	**38**	**33**	**27**	**36**	**35**	**28**	**65**	**55**	**34**	**65**	**111**	**92**
	Vasilakis	60	40	27	86	69	43	123	91	73	170	122	94
	VasilakisLPC	54	28	16	81	60	39	110	85	63	163	107	78
	Mine	85	48	36	113	97	69	173	122	88	203	166	120
	MineLPC	94	56	40	120	99	73	177	123	90	209	176	115
	Mix	44	26	10	72	49	36	92	69	48	116	87	62
0.15	**Pulses**	**89**	**84**	**64**	**96**	**81**	**57**	**113**	**105**	**80**	**144**	**168**	**140**
	Praat	**89**	**84**	**63**	**96**	**80**	**56**	**114**	**103**	**80**	**144**	**166**	**137**
	Vasilakis	129	92	83	162	127	91	180	131	101	211	168	119
	VasilakisLPC	112	87	64	148	113	71	161	111	85	196	141	98
	Mine	163	128	83	189	143	108	204	157	125	210	187	134
	MineLPC	171	154	105	198	170	123	203	173	121	234	204	138
	Mix	87	62	47	114	88	58	121	93	71	142	105	80
0.3	**Pulses**	**200**	**159**	**143**	**203**	**174**	**140**	**211**	**163**	**132**	**240**	**230**	**160**
	Praat	**200**	**157**	**144**	**204**	**179**	**137**	**228**	**163**	**130**	**236**	**227**	**160**
	Vasilakis	202	143	123	266	207	151	250	214	156	283	230	177
	VasilakisLPC	193	124	99	250	183	139	232	188	139	254	208	162
	Mine	222	143	116	248	199	148	244	195	163	260	225	165
	MineLPC	249	174	135	269	214	171	270	213	174	294	229	189
	Mix	139	96	76	175	133	105	171	142	112	188	150	127
0.4	**Pulses**	**235**	**259**	**191**	**265**	**222**	**155**	**292**	**235**	**195**	**297**	**248**	**260**
	Praat	**235**	**259**	**198**	**265**	**222**	**153**	**292**	**231**	**196**	**295**	**257**	**257**
	Vasilakis	226	169	150	272	235	167	272	226	191	299	243	214
	VasilakisLPC	221	166	121	280	221	145	270	216	170	298	241	183
	Mine	249	188	152	270	223	154	289	233	185	293	238	188
	MineLPC	270	204	173	292	240	184	305	245	198	329	247	217
	Mix	152	121	89	199	159	107	189	157	126	210	171	134

Table 37. SP average jitter in µs for sound /ia/.

8.15. Synthetic phonation. Average error.

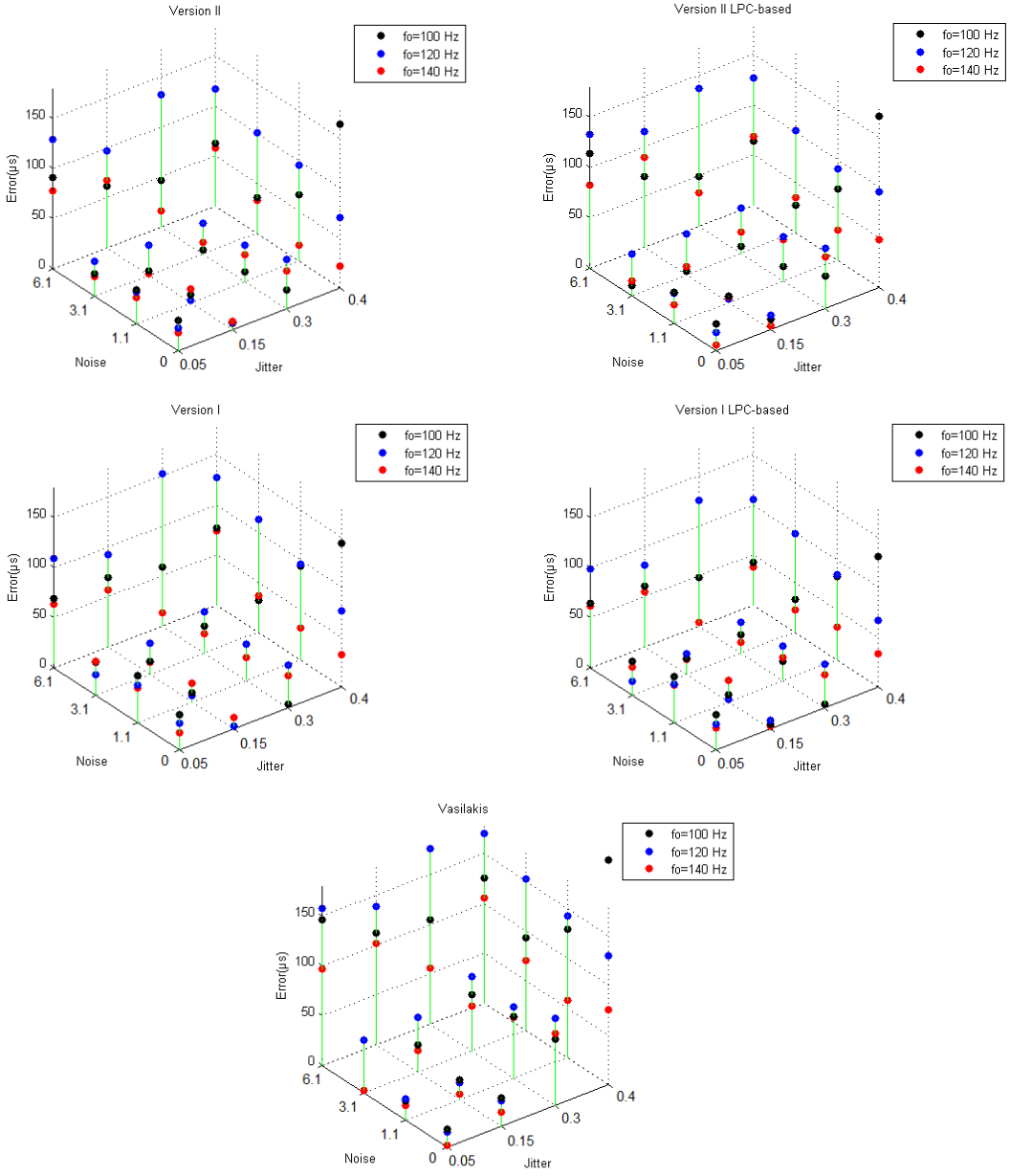

Figure 55. SP jitter error in µs obtained by Version I, Version I _LPC, Version II, Version II _LPC and Vasilakis, respectively, for sound /i/.

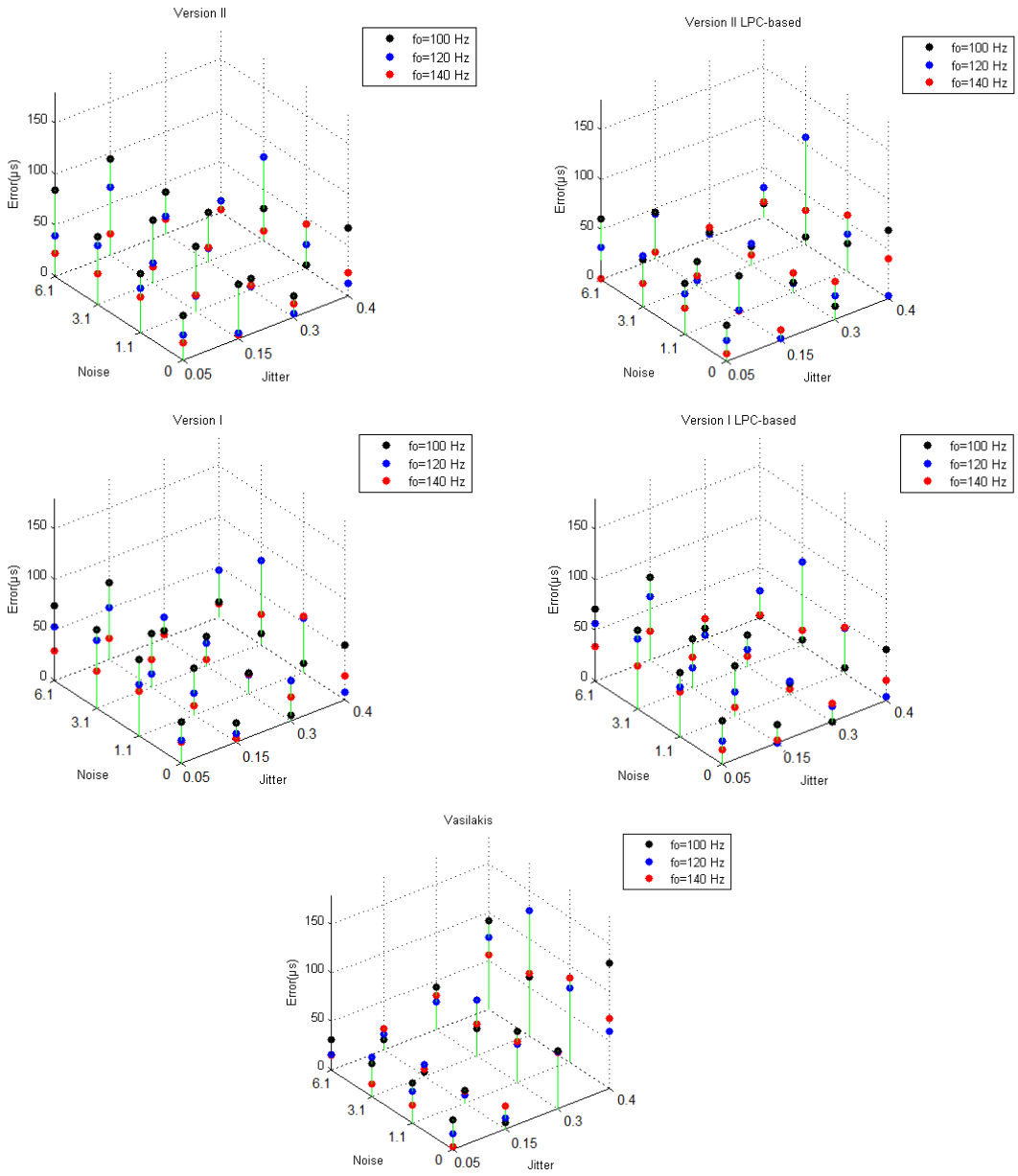

Figure 56. SP jitter error in µs obtained by Version I, Version I _LPC, Version II, Version II _LPC and Vasilakis, respectively, for sound /u/.

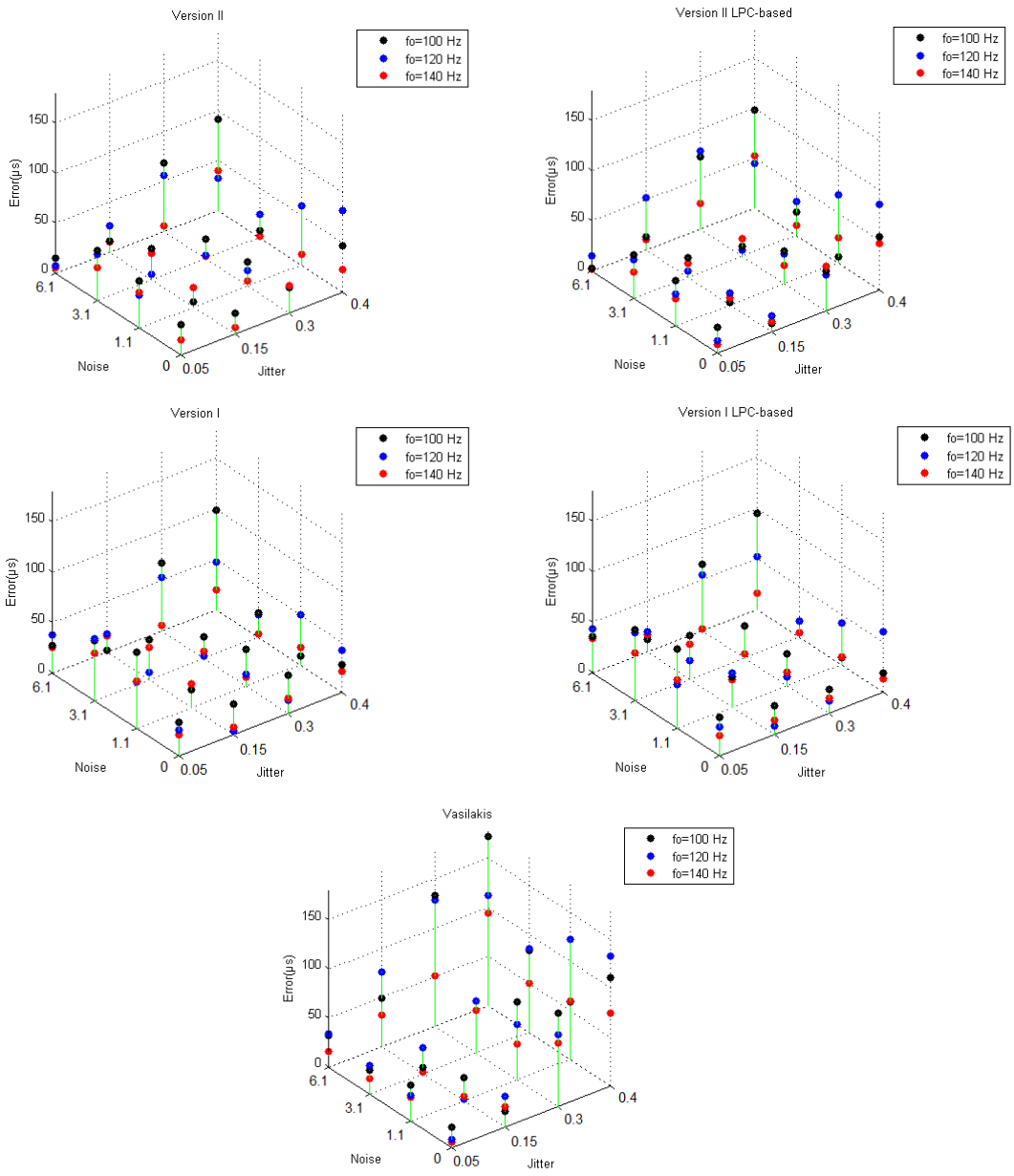

Figure 57. SP jitter error in µs obtained by Version I, Version I _LPC, Version II, Version II _LPC and Vasilakis, respectively, for sound /ai/.

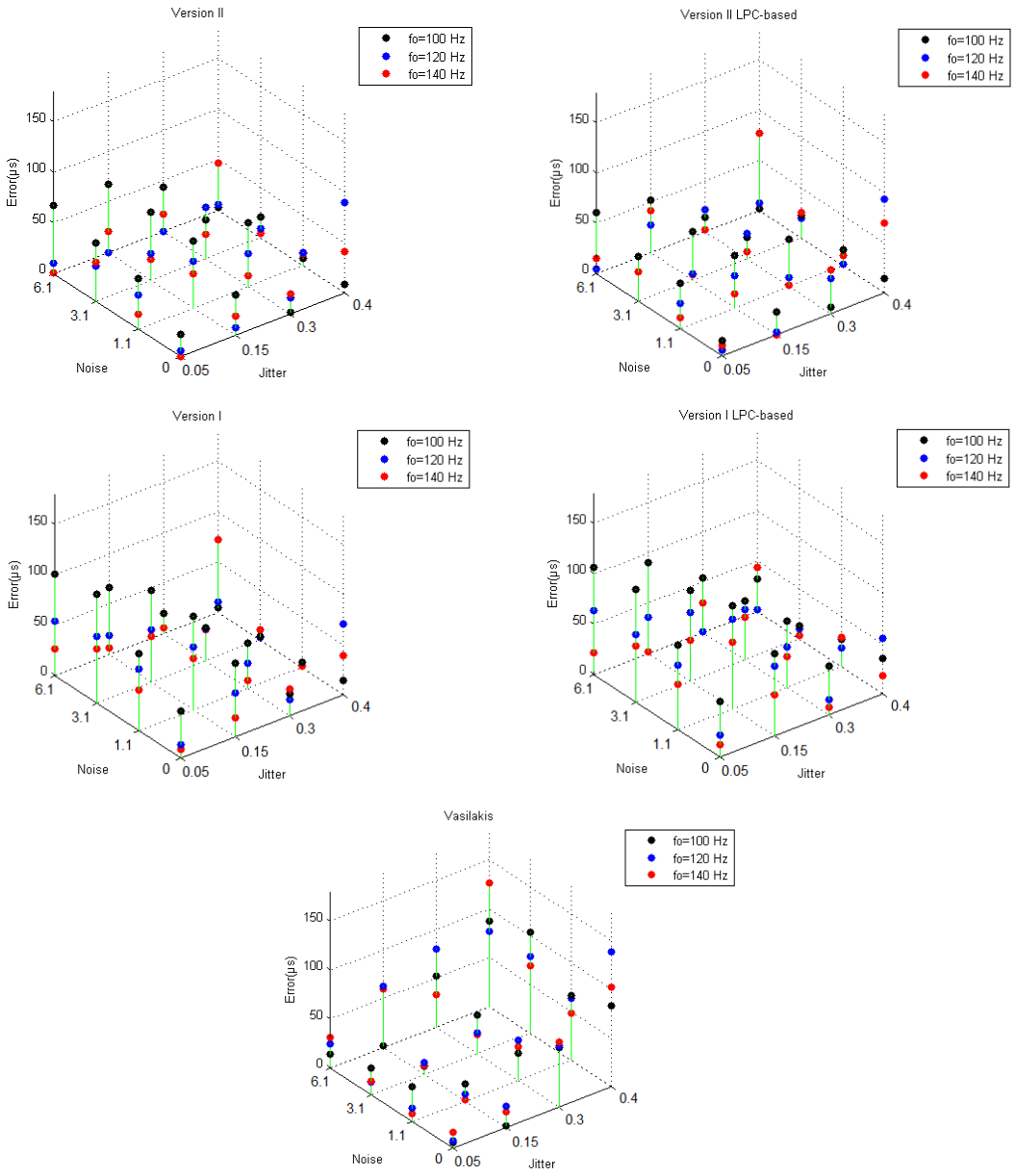

Figure 58. SP jitter error in µs obtained by Version I, Version I _LPC, Version II, Version II _LPC and Vasilakis, respectively, for sound /ia/.

8.16. Connected speech. Average jitter.

Lesson	1-pre	1-post	2-pre	2-post	3-pre	3-post	4-pre	4-post
Praat	98	116	144	132	87	118	124	80
Vasilakis	55	182	118	49	144	120	78	143
Version II	158	144	135	152	124	132	153	114

Table 38 part 1. CS average jitter in µs pre and post-lesson.

Lesson	5-pre	5-post	6-pre	6-post	7-pre	7-post
Praat	89	82	99	76	89	73
Vasilakis	122	148	143	155	147	136
Version II	119	117	113	114	114	109

Table 38 part 2. CS average jitter in µs pre and post-lesson.