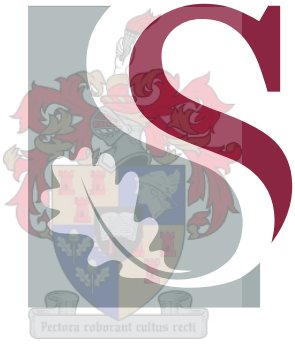
A Parametric Monophone Speech Synthesis System

Gideon Klompje



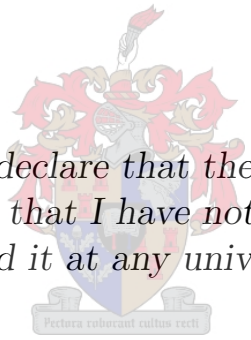
Thesis presented in partial fulfilment of the requirements for the degree
Master of Science in Electronic Engineering
at the University of Stellenbosch

Supervisor: Dr T.R. Niesler

December 2006

Pectora roborant cultus recti

Declaration



I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

SIGNATURE DATE

Abstract

Keywords: excitation signal modelling, LPC synthesis, monophone synthesis, multi-lingual speech synthesis, rule-based speech synthesis, speech signal modelling, speech synthesis, text-to-speech (TTS)

Speech is the primary and most natural means of communication between human beings. With the rapid spread of technology across the globe and the increased number of personal and public applications for digital equipment in recent years, the need for human/machine interaction has increased dramatically. Synthetic speech is audible speech produced by a machine automatically. A text-to-speech (TTS) system is one that converts bodies of text into digital speech signals which can be heard and understood by a person.

Current TTS systems generally require large annotated speech corpora in the languages for which they are developed. For many languages these resources are not available. In their absence, a TTS system generates synthetic speech by means of mathematical algorithms constrained by certain rules.

This thesis describes the design and implementation of a rule-based speech generation algorithm for use in a TTS system. The system allows the type, emphasis, pitch and other parameters associated with a sound and its particular mode of articulation to be specified. However, no attempt is made to model prosodic and other higher-level information. Instead, this is assumed known. The algorithm uses linear predictive (LP) models of monophone speech units, which greatly reduces the amount of data required for development in a new language. A novel approach to the interpolation of monophone speech units is presented to allow realistic transitions between monophone units. Additionally, novel algorithms for estimation and modelling of the harmonic and stochastic content of an excitation signal are presented. This is used to determine the amount of voiced and unvoiced energy present in individual speech sounds.

Promising results were obtained when evaluating the developed system's South African English speech output using two widely used speech intelligibility tests, namely the modified rhyme test (MRT) and semantically unpredictable sentences (SUS).

Opsomming

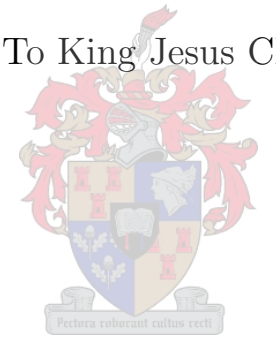
Spraak is die primêre en mees natuurlike vorm van kommunikasie tussen mense. Saam met die versnelde verspreiding van tegnologie regoor die wêreld en die enorme hoeveelheid persoonlike en publieke toepassings van digitale toerusting het die behoefte aan 'n meer persoonlike koppelvlak tussen mens en masjien aansienlik gegroei. Sintetiese spraak is spraak wat outomaties deur 'n masjien opgewek word deur middel van 'n sogenaamde teks-na-spraak ("text-to-speech", TTS) stelsel, sodanig dat dit deur 'n mens gehoor en verstaan kan word.

Huidige TTS-stelsels benodig oor die algemeen groot hoeveelhede spraakdata vir elke taal waarvoor hul gebruik word. Sulke spraakdatabasisse is nie altyd beskikbaar in 'n taal waarvoor 'n TTS-stelsel ontwikkel word nie. Indien hierdie data ontbreek, moet 'n TTS-stelsel the sintetiese spraak deur middel van wiskundige algoritmes opwek wat deur 'n stel reëls bepaal word.

Hierdie tesis beskryf die ontwerp en implementasie van 'n reël-gedrewe spraakopwekkingsalgoritme vir gebruik in 'n TTS-stelsel. Die stelsel laat die gebruiker toe om die tipe, klem, toonhoogte en ander parameters wat met spraakklanke en hul artikulasie verband hou, te spesifiseer. Geen poging word egter aangewend om sulke prosodiese en ander hoë-vlak elemente, wat as bekend geag word, te modelleer nie. Die algoritme maak gebruik van lineêre voorspelling om monofoon spraakeenhede to modelleer. Gebruik van hierdie eenhede perk die hoeveelheid spraakdata wat deur die stelsel benodig word ten einde 'n nuwe taal te kan naboots, drasties in. 'n Nuwe algoritme vir die interpolasie van monofoon spraakeenhede is ontwikkel vir hierdie doel. Verdere bydraes word gelewer deur die definisie van nuwe algoritmes waarvolgens die harmoniese en stogastiese inhoud van 'n spraakeenheid afgeskat en gemodelleer kan word. Hierdie inligting word gebruik om die stem- en stemlose komponente van individuele spraakklanke te bepaal.

Die stelsel se sintetiese spraak in Suid-Afrikaanse Engels is met behulp van twee standaard verstaanbaarheidstoetse gemeet, naamlik die "modified rhyme test" (MRT) en semanties onvoorspelbare sinne ("semantically unpredictable sentences", SUS), wat belowende resultate opgelewer het.

To King Jesus Christ



Acknowledgements

Thank you to:

- My loving wife Nerina, for always being patient and supportive, especially when I wasn't.
- My parents, for being great friends, and spending their fortunes on my studies for the past seven years.
- Dr. Thomas Niesler, for doing so much more than was expected of him. I am unaware of another study leader that covers so many extra miles for his students.
- Ludwig Schwardt, for presenting me with a wealth of DSP knowledge.
- Prof. Johan du Preez, for teaching me some of the black arts of pattern recognition.
- Dr. Gert-Jan van Rooyen, for this thesis template.
- Valérie Hazan of the UCL[1] Department of Phonetics and Linguistics, for providing a pre-generated list of 250 semantically unpredictable sentences.
- Alison Wileman of SU-CLaST[2], for producing the phonetic dictionary used to determine the phonetic transcriptions of the SUS test words.
- Everyone that participated in the listening tests (for your creative semantically unpredictable sentences!).
- The NRF[3], for substantial financial assistance in 2006.
- And finally, the DSP lab people: Hansie, Rinus, Jaco, Richard, George, Eugene, Willie, Michael and all the rest, for fuelling my passion for engineering with DSP coffee.

[1] University College London

[2] Stellenbosch University Centre for Language and Speech Technology

[3] National Research Foundation

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ACF	autocorrelation function
ADC	analogue-to-digital converter
AM	amplitude modulation
ANN	artificial neural network
AST	African Speech Technology project
DFT	discrete Fourier transform
DP	dynamic programming
DRT	diagnostic rhyme test
FFT	fast Fourier transform
FM	frequency modulation
GSL	GNU Scientific Library
GUI	graphical user interface
HMM	hidden Markov model
HNM	harmonics plus noise model
I/O	input/output
LP	linear prediction
LPC	linear predictor coefficient
LSF	line spectral frequency
LSP	line spectral pair
MFCC	Mel frequency cepstral coefficient
MRT	modified rhyme test
OQ	(vocal folds) open quotient
OSS	Open Sound System^{TM}
PC	personal computer
PCM	pulse code modulation
PDF	probability density function
RC	reflection coefficient
RK	Rosenberg-Klatt (excitation signal model)
SUS	semantically unpredictable sentences
TTS	text-to-speech
V/U	voiced/unvoiced

Variables

symbol	description
$\varepsilon(\cdot)$	LP error/residual signal
a_k	k^{th} LPC
A_n	unvoiced magnitude factor
A_v	voiced magnitude factor
$\frac{1}{A(z)}$	LP filter
dB	decibel, logarithmic power scale ($= 20\log_{10}(\cdot)$)
F_0	pitch $= \frac{1}{T_0}$ $[Hz]$
F_{max}	maximum voicing frequency $[Hz]$
F_s	sampling frequency $= \frac{1}{T}$ $[Hz]$
$H(\cdot)$	entropy
$H_D(\cdot)$	differential entropy (negentropy)
Hz	Hertz, unit of frequency
k_i	i^{th} RC
Kb	kilobyte, unit of computer memory (= 1024 bytes)
kurt$(\cdot)$	kurtosis
m_{f_ε}	expectation of residual Gaussianity
Mb	megabyte, unit of computer memory (= 1024 Kb)
p	LP order
s	seconds, unit of time
T	sampling period $= \frac{1}{F_s}$ $[s]$
T_0	pitch period $= \frac{1}{F_0}$ $[s]$
$x(\cdot)$	some discrete time signal

Operations

$\mathcal{E}\{\cdot\}$	expected value
$\mathcal{F}\{\cdot\}$	Fourier transform
$\mathcal{F}^{-1}\{\cdot\}$	inverse Fourier transform
$r_{xx}(\cdot)$	autocorrelation sequence of x
$\mathcal{Z}\{\cdot\}$	$\mathcal{Z}$-transform

Chapter 1

Introduction

Synthetic speech is speech that is produced by an entity other than a human being. The topic has been a popular research theme for over two centuries probably because humans are interactive beings, which leads us to desire interaction with all objects in our environment. It becomes an even more exciting prospect when a person is able to interact with his own invention. With the use of electronics so widespread in today's society, we shall restrict our attention to speech produced by a machine electronically. Speech is the most natural form of interaction between people, and adding to a machine the ability to produce such speech gives it human qualities. In fact, one can associate the development of synthetic speech with that of artificial intelligence, since both attempt to bridge the gap between man and machine by adding human qualities to machines.

The usefulness of synthetic speech stretches much further than simple amusement, however. There are many practical situations today which can greatly benefit from it. In fact, any situation where interaction takes place between a machine and a human being can potentially benefit if the machine is able to communicate with the person in his or her language. With the growing prevalence of automated systems, this becomes an increasingly common situation.

1.1 Applications of Speech Synthesis

The usefulness of speech synthesis is not to be underestimated. It would seem that industry is eager to adopt commercial quality synthesisers, but current systems are often limited in terms of their functionality. This should not, however, prevent us from exploring the possibilities and stressing how much can be gained using speech synthesis to bring humans and machines into closer fellowship. Listed below are a few categories of current (and possibly future) applications of speech synthesis:

Spoken dialogue systems. These are interactive systems designed for a specific purpose, such as flight or hotel reservation systems or information retrieval systems. They are often designed for the user to be able to interact with the system via telephone. These systems in particular have begun to apply speech synthesis in recent years, because

their speech output is task-specific and therefore requires only a limited vocabulary to function adequately. The design and implementation of the speech generation section of one such system is described in [48].

Educational systems. The use of speech synthesis may greatly reduce the teaching requirements for language learning by teaching a person how to pronounce words in a specific language. A learner may listen to a computer pronouncing words they see on screen and hence begin to read, write and speak that language by interactive learning. This would also benefit the learners in terms of personal attention received, as "teachers" would be as many as there are computers available.

Aid for the disabled. Speech synthesis would make computers much more accessible to the blind and people with lexical difficulties. If information on a computer system (e.g. web page content) were to be retrieved and read to individuals who can't read themselves, they would have access to information which is otherwise difficult for them to obtain. Portable devices capable of speech synthesis would greatly ease communication with a person with a speech impediment by allowing him or her to convert text into speech for anyone to understand.

Translation systems. It would be convenient to know that one can travel to any location on the planet and be able to communicate with the locals in their native tongue with relative ease. Software applications for handheld devices that converts an utterance from a source language, received as text or even spoken utterances, to a target language and even produces the output utterance synthetically are already beginning to appear. Conversing with someone in a foreign language over the internet or even via telephone could be much easier if speech synthesis were involved. Some such systems have appeared in the last five years, albeit with a fairly limited variety of languages and applications.

This is but a small list of possibilities of the application of speech synthesis to life in general. It by no means encompasses the entire range of applications, but hopefully stresses the importance of continued research in this field. It provides more than sufficient motive for developing a flexible and portable speech synthesis system which is capable of producing intelligible speech for a variety of languages.

1.2 Motivation for this Study

This study concerns itself with the development of a flexible speech generation system which is not restricted to any specific language. Language-independence is particularly important in a country such as South Africa, which has eleven official languages. Project restrictions did not allow testing in multiple languages, but the system is designed to synthesise speech in any target language if given a suitable phonetic description.

Africa may benefit greatly from especially the educational applications of this type of speech synthesis system, since illiteracy is widespread and qualified teachers are few in number in many locations. Disabled persons may also be aided in ways such as those listed earlier if they were to gain access to devices that employ such a system.

Before continuing, let us first take a brief tour of the history associated with artificial speech and some of the current approaches to the problem of synthesising speech.

1.3 History of Artificial Speech

This section lists but a few of the early milestones in the development of speech synthesis. For a more complete history, see [22] and [25] as well as their references.

The earliest forms of artificial speech date as far back as the late 1700's:

- 1779 — Professor Christian Kratzenstein of St. Petersburg produced a set of acoustic tubes which were able to produce certain vowel sounds when excited by a vibrating reed.
- 1791 — Wolfgang von Kempelen (Vienna) produced his "Acoustic-Mechanical Speech Machine", which consisted of a pressure chamber (lungs), a vibrating reed (vocal chords) and a leather tube (vocal tract) which could be manipulated to produce various sounds and sound combinations. The machine could also produce some consonant sounds using constricted airflow chambers.
- Mid-1800's — Charles Whetstone produced a more complicated version of Von Kempelen's speech machine. This was the first device that could produce whole words.
- 1922 — The first fully electrical synthetic speech generation system was introduced by Stewart. It consisted of a buzzer (voicing) and two resonator circuits which were used to define vowel sounds.
- 1939 — Homer Dudley presented his "Voice Operating Demonstrator" (VODER), which is considered the first speech synthesiser. This complicated device consisted of a keyboard and pedals (much like an organ) which could be "played" by trained individuals to produce synthetic speech.

Dudley's VODER raised several eyebrows at the New York World's Fair in 1939, which seemed to spark an increase in the research and development of speech synthesis systems. Since then, many systems have been developed and TTS research has become a popular research topic, even to this day.

1.4 Synthesis Methods

This section provides some background detail concerning the methods commonly applied in the speech synthesis field. For a more comprehensive overview, the reader is referred to [22]

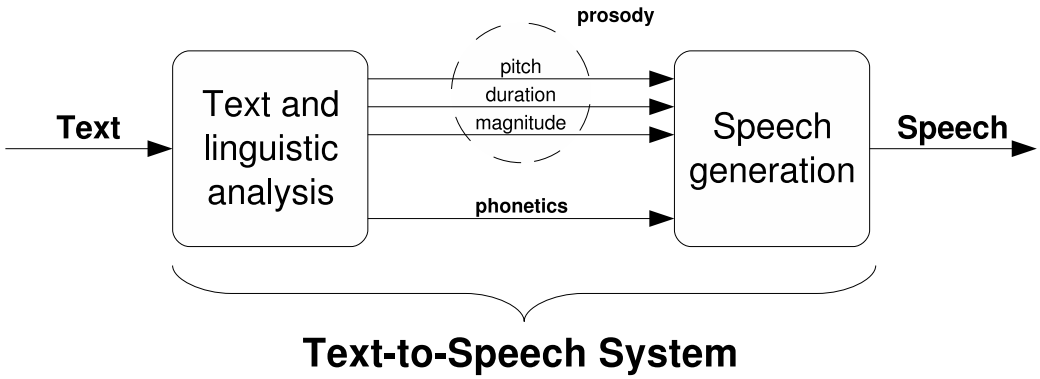


Figure 1.1: *The components of a TTS system.*

and [25].

A complete TTS system can be separated into two main phases, which are illustrated in figure 1.1:

1. Text and linguistic analysis, referred to as high-level synthesis, and
2. Speech generation, referred to as low-level synthesis.

The aim of the first phase is to analyse the input text grammatically and semantically and expand it into a more detailed representation. This expansion includes the generation of phonetic as well as prosodic information. The output of the first phase can be seen as a detailed description of the desired speech output, for input to the second phase. The development of the text and linguistic analysis component of a TTS system is mostly a linguistic problem, and this thesis will focus only on the second phase.

Low-level synthesis concerns itself with creating speech sounds in the form of a sequence of digital samples from the description given by phase one. In creating this part of a TTS system, the designer is faced with a choice between two major approaches to the problem:

1. Concatenative synthesis, and
2. Parametric (rule- or model-based) synthesis.

The first approach generally performs synthesis by concatenating prerecorded speech segments to form the desired utterances. The length of these segments is for the designer to decide, and needs to be chosen carefully with the system's goals in mind. Because this form of speech synthesis uses true speech samples to form its output, its great advantage is its high naturalness.

The second approach uses a mathematical model to generate synthetic speech. Such a model should then be able to generate the desired sounds directly and independently of prerecorded examples. There are various models with various levels of complexity to choose

from. Here, again, the designer faces the task of choosing among the models for the most appropriate, given the task at hand. The main advantages of model-based systems are often their low memory and computational requirements, as well as their flexibility.

Many hybrid systems which attempt to draw from the advantages of both have also been proposed. However, since these mostly employ concatenative synthesis methods and only apply parametric models to smooth out discontinuities at concatenation boundaries, we will view them as concatenative synthesis systems themselves.

1.4.1 Concatenative synthesis

Concatenative speech synthesis systems consist of a unit database, a selection algorithm and, if required, one or more smoothing algorithms. Speech unit concatenation has become the dominant approach in speech synthesis over the last fifteen years or so. This is mainly because the rule-based systems of previous decades failed to achieve acceptable levels of naturalness. These older systems could often synthesise individual speech sounds with some success, but failed when attempting to model the transitions between phones adequately. Certain types of phones are also harder to model than others, especially nasalised sounds, such as /m/ ("*m*onkey") or /n/ ("*n*ote"), and phones containing a mixed voiced and unvoiced (fricative) sounds, such as /z/ ("*z*oo") or /v/ ("en*v*y"). The concatenative synthesis approach can avoid these and other problems by implicit modelling, i.e. copying relevant sounds from the database without an understanding of the underlying generative mechanisms of such speech segments.

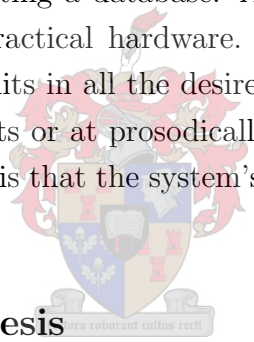
Because a concatenative synthesis system is heavily dependent upon its database of recordings, the choice of concatenation units determines the synthesis quality to a very large degree. There are many speech units for which such systems can be designed, such as words, syllables, demisyllables, phones, diphones or triphones [25]. Generally speaking, the longer the unit, the more units are required in the database and therefore the greater the memory requirements of the system. Longer units, however, are often better candidates for implicit modelling of effects such as co-articulation. On the other hand, the use of shorter units implies that more concatenation points, and therefore a greater number of potentially audible discontinuities, exist within a synthesised utterance [43]. Many current systems use variable length units to find a balance between the advantages and disadvantages of longer and shorter units, but this complicates the database design and unit selection algorithm.

The design and construction of the speech database is a task of major importance in concatenative speech synthesis systems. Careful task-oriented design is crucial to providing the selection algorithm with suitable candidates for concatenation [4], [8], [48]. Many aspects of natural-sounding synthetic speech depend upon the particular units chosen, and therefore the database must provide a selection which allows flexibility in terms of prosodic and phonetic context for the selection algorithm. In most systems today, the database contains a variety of phonetically identical recordings, but which differ in terms of context and prosodic content. Although this expands the database considerably, it is desirable in most cases be-

cause prosodic modification of a recorded speech segment is often problematic, especially when drastic modifications are necessary.

Once the database has been assembled, an algorithm is required which optimally selects units suitable for concatenation. The algorithm must take into account factors such as prosodic and phonetic context and choose those segments which most closely resemble the desired output utterance while introducing as little distortion as possible at the concatenation points. Usually a smoothing strategy is applied to remove the discontinuity effects at concatenation boundaries. Most systems use overlap-add methods such as TD-PSOLA, originally developed by *France Telecom*, for this purpose. LP-PSOLA, presented in [10], is one alternative, but there are many variations. If the database is limited, however, the need for spectral smoothing and prosodic modification algorithms increases, and there are various strategies that can be employed in this situation [5], [13], [33], [37] and [42].

The main advantage of concatenative speech synthesis is its high naturalness due to implicit modelling of the human speech production system. This is also the reason why it is the preferred speech synthesis method today. However, these systems inherently suffer from certain limiting disadvantages [9]. One disadvantage is the tremendous amount of effort involved in recording and annotating a database. Also, the size of the database is limited by the memory constraints of practical hardware. Hence, a database usually cannot be sufficiently large to contain all units in all the desired contexts. This often leads to audible distortions at concatenation points or at prosodically modified segments in the synthesised utterance. Another disadvantage is that the system's speaking style is limited to that which is available in the database.



1.4.2 Parametric synthesis

This form of speech synthesis is performed using some approximate model of speech production. Some models are based on the human speech production system, but this is not a requirement. Other models are more concerned with modelling the speech signal itself without any direct relation to its source. As long as intelligible speech can be produced, both types of models are acceptable.

Before a model for speech synthesis is chosen, it must be decided how that model is to be used. This generally entails choosing which speech units to model, such as monophones, diphones or triphones. If the units being modelled are not stationary, as is the case with diphones, triphones and certain monophone types, a scheme must be chosen to account for the changes in the signal over time. The most common way of doing this is by subdividing the signal into even smaller parts in a process called windowing. Typical window lengths range between $20ms$ and $30ms$ (often chosen to overlap to increase time resolution), as speech signals are generally quite stationary for such short periods of time.

The set of models that attempt to model the human vocal tract directly by means of parameters such as tongue positions, lip positions, etc. are called articulatory synthesisers. Accurate measurements of such articulator movements can only be made by specialised

equipment, and these measurements are often three dimensional. These complex model parameters then need to be converted to a sequence of digital samples to produce the synthetic speech. Because this is a very difficult and computationally expensive task [25] and the relationship between articulator positions/movements and produced sounds is not generally unique, these models have not yet found widespread acceptance in major TTS systems.

Choosing an analysis method often requires careful consideration of its advantages and disadvantages. Simplicity and stability are key requirements. Probably the largest collection of parametric speech production models are based on what is called the source-filter model of speech production. This model separates the speech signal into two distinct parts:

- A source, often called the excitation, which can be voiced (due to the vibration of the vocal chords at the glottis), unvoiced (due frication caused by the tongue, throat, lips, etc.) or a combination of the two.
- A filter, which represents the effects that the vocal tract has on the source signal.

Although it is not entirely accurate, these two components are often assumed independent and therefore separable for simplicity. Various analysis methods to separate source and filter in a speech signal are available [19]. Some of the earlier parametric TTS systems were formant synthesisers, for which the speech sounds are defined by their formant frequencies and bandwidths (see section 2.2.1) and relate to the filter part of the speech signal [1]. The source signal is often not modelled explicitly, but rather the original speech waveform is inverse filtered using the estimated filter to obtain a residual signal, after which unit selection and concatenation techniques are applied. Many systems make use of a codebook of excitation signals, an approach often used in speech coding. For example, a codebook of polynomial excitation signals is used in the work described in [6]. Chapter 3 deals with excitation models in more detail.

To conclude, parametric speech synthesis systems have the advantage over concatenative systems of being potentially much more flexible in terms of the modification of prosody and other speech parameters. This is because synthetic speech is artificially generated and can be specified completely. Concatenative systems must modify existing concatenation units (recordings) to accurately follow desired phonetic combinations and prosodic contours, modifications which may introduce audible distortions. Other advantages of using parametric models include comparatively low memory and computational requirements, which makes them suitable candidates for low bit rate speech coding and other real time applications. The main drawback of parametric models is that, at best, they merely approximate the fundamental characteristics of natural speech, and simplicity often comes at the cost of quality. Therefore, although many parametric speech systems produce intelligible synthetic speech, their output is usually not particularly natural.

1.5 Project Scope

This thesis describes the design and implementation of a flexible speech generation system requiring few language-dependent resources. It is hoped that this will aid in the rapid development of multi-lingual TTS applications, such as pilot systems, in Southern Africa. Although the proposed model also aims at synthesising speech with a high degree of naturalness, this is considered to be of less importance than intelligibility. The argument is that a machine's voice should be allowed to sound different to that of a human just as one speaker's voice and speaking style can differ from that of another, as long as the two are able to communicate freely. A parametric approach was chosen to ensure model flexibility for the purpose of multilingual speech synthesis because the system is aimed at the synthesis of a variety of African languages.

System design is aided by the use of high level languages such as Matlab and *Python* and implementation is in the *C* language. *C* was chosen to maximise portability and the system libraries' modular design aids research into TTS by allowing the use of different speech modelling and parameter interpolation schemes.

Monophones were chosen as the basic synthesis units, and linear prediction (LP) for speech modelling. Monophones were chosen because they can completely define a language phonetically and each phone can be represented using a single linear predictor coefficient (LPC) vector. This is in contrast to the use of diphones/triphones, of which there is a much larger number per language and modelling requires multiple parameter vectors for each unit. The use of monophones and parametric modelling also eases the adaptation of the system to a new language greatly, since the number of monophones is always much smaller than the number of diphones or triphones, and many languages do not have comprehensive annotated databases from which the latter synthesis units can be extracted. South African English, for example, only has about fifty monophones. Because of the difficulty of modelling the parameter transitions between phones, almost all TTS systems that were encountered in literature model diphones or larger units for synthesis. No detailed reference to a system of similar architecture using monophone units could therefore be found for comparison. The most similar system (in terms of monophone modelling and interpolation) that was found is the formant synthesiser described in [1].

Different interpolation schemes were examined to model co-articulation effects between phones. Chapter 4 deals with these and presents an interpolation scheme suitable for monophone synthesis. LP analysis was chosen because of its simplicity and stability and because it can adequately estimate a filter model for the vocal tract, allowing for independent manipulation of source and filter. Among the literature reviewed, no current TTS system was found that performs LPC synthesis by rule-based interpolation. This is because the majority of TTS systems over the last fifteen years have been developed using data-driven approaches for automatic parameter generation models such as HMM's or ANN's.

Various attempts were made at defining a suitable parametric excitation model for flexible and emotional speech generation because good control over prosodic content in the synthetic

speech utterance is crucial. Accurate modelling of the voiced and unvoiced components of excitation signals was found to influence synthesis quality strongly and hence given particular attention. Chapter 3 assesses these requirements and proposes the parametric models used in the development of this project for flexible and emotional speech synthesis. The modular design of the system allows the excitation model to be interchanged simply.

Multiple prosodic contours are overlaid within an utterance to ensure maximum flexibility [44]. Although not all of the advantages of this approach are exploited in the current system, the modular design of the system ensures that future incorporation of a prosodic contour generation module should require little or no modification to the existing system.

1.6 Thesis Overview

This information in this thesis is presented as follows:

- Chapter 2 begins with a short description of the human speech production system in section 2.1. Thereafter, some of the main approaches to the analysis of speech signals are presented. Listed in section 2.5 are some of the modelling techniques and parameters chosen for the development of the speech generation system described in this thesis.

- Chapter 3 deals with excitation signal modelling. Some commonly used models are presented in section 3.2, and a number of methods for estimating their associated parameters are presented in section 3.3, including a novel approach to the estimation of the voiced/unvoiced content of residual signals. Section 3.4 presents a discussion on the modelling of plosive sounds, which required special attention. Section 3.5 contains a discussion on speech prosody and how it can be incorporated into the presented excitation signal models.

- Chapter 4. The algorithms which were used to generate fluent inter-phone transitions by interpolating the monophone parameter vectors for both the filter (section 4.1) and source (section 4.2) parameters are developed in this chapter.

- Chapter 5 presents the speech generation system's functionality using short discussions of its core module functions. Also shown in this chapter are data flow diagrams which detail the modular design of the system as well as the flow of information during the two major phases, namely analysis and synthesis.

- Chapter 6 evaluates the intelligibility of the system developed in chapters 2 and 4 using two widely used test sets (MRT and SUS). This chapter presents and discusses the results obtained from these experiments.

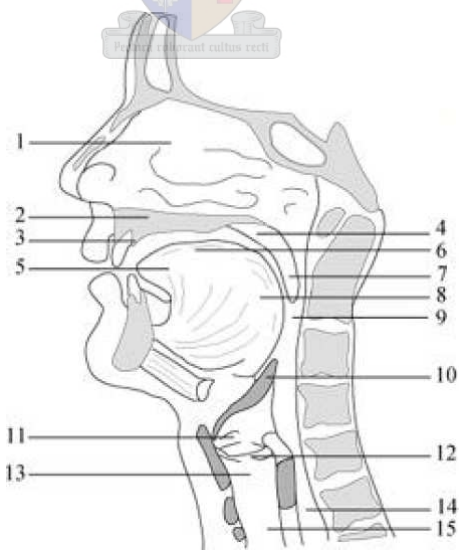
- Chapter 7 concludes the thesis and presents some recommendations for possible future work.

Chapter 2

Speech Signal Analysis

Before we describe in detail the speech production model used in this project, it is necessary to cover some basics concerning speech and the speech signal. This chapter provides a review of the relevant established methods used in the analysis of digital speech signals, as well as the terminology and some symbolic notations that will be adopted in the following chapters. Before commencing the analysis of speech signals, however, let us first discuss the human speech production system in order to gain an understanding of how speech signals are produced. Knowledge of this process may aid us in developing methods for the analysis and modelling of speech signals.

2.1 Human Speech Production



(1) Nasal cavity, (2) Hard palate, (3) Alveoral ridge, (4) Soft palate (Velum), (5) Tip of the tongue (Apex), (6) Dorsum, (7) Uvula, (8) Radix, (9) Pharynx, (10) Epiglottis, (11) False vocal cords, (12) Vocal cords, (13) Larynx, (14) Oesophagus, and (15) Trachea.

Figure 2.1: *The human speech production system (reproduced from [25]).*

This section contains a short summary of the human speech production system[1] in order to lay the foundation for the basic modelling of speech signals.

Sound is produced by pressure from the lungs and diaphragm, which creates an airflow through the vocal organs depicted in figure 2.1. The speaker determines the identity of the sound by the shape of his/her vocal cavity. The voicing quality of a sound is determined by the opening between the vocal chords, called the glottis. By rapidly opening and closing, the glottis produces near-periodic acoustic energy pulses which are perceived by a listener as the speaker's voice. Unvoiced sounds are produced by keeping the glottis open while constricting the flow of air elsewhere, or by a constriction at the glottis itself. There are also various sounds which are formed by a combination of these, such that the glottis produces voicing, but airflow is also constricted somewhere else. Sounds with a strong nasal quality are produced by opening the velum to allow an increased flow of air through the nasal cavity relative to the oral cavity. A complete closure in the oral cavity when doing so results in a completely nasal sound.

Generally, the shape of the oral cavity is responsible for the identity of the sound, whereas the glottis determines the type of excitation which is produced. From a speech signal analysis point of view, we should attempt to accurately model the effects of both these aspects of the speech signal if we are to produce natural-sounding speech artificially.

2.2 The Discrete Time Speech Signal

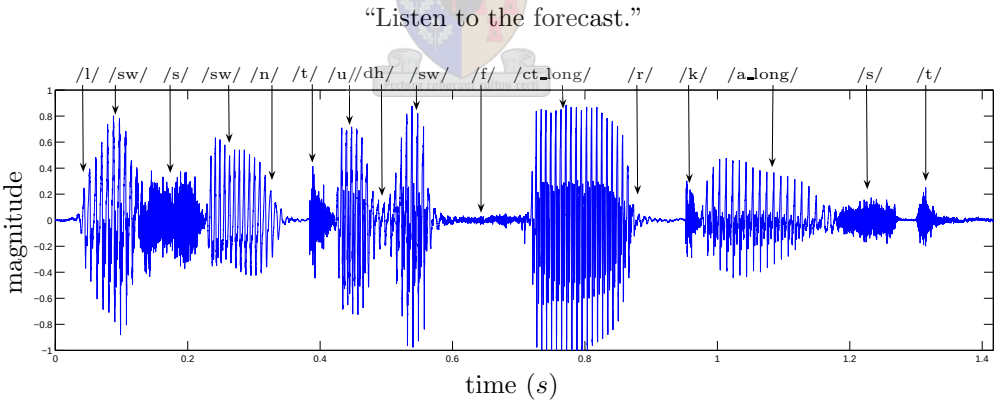


Figure 2.2: *Example of a discrete time speech signal.*

If we plot the samples of a discrete time speech signal against time, we see a waveform such as the one shown in figure 2.2. The waveform was recorded at a sampling rate[2], F_s, of $24kHz$. Some important observations can be made by studying such waveforms. For

[1] The information contained in section 2.1 was obtained from [25], including figure 2.1.

[2] Information regarding digital signals, sampling, etc. can be found in [35].

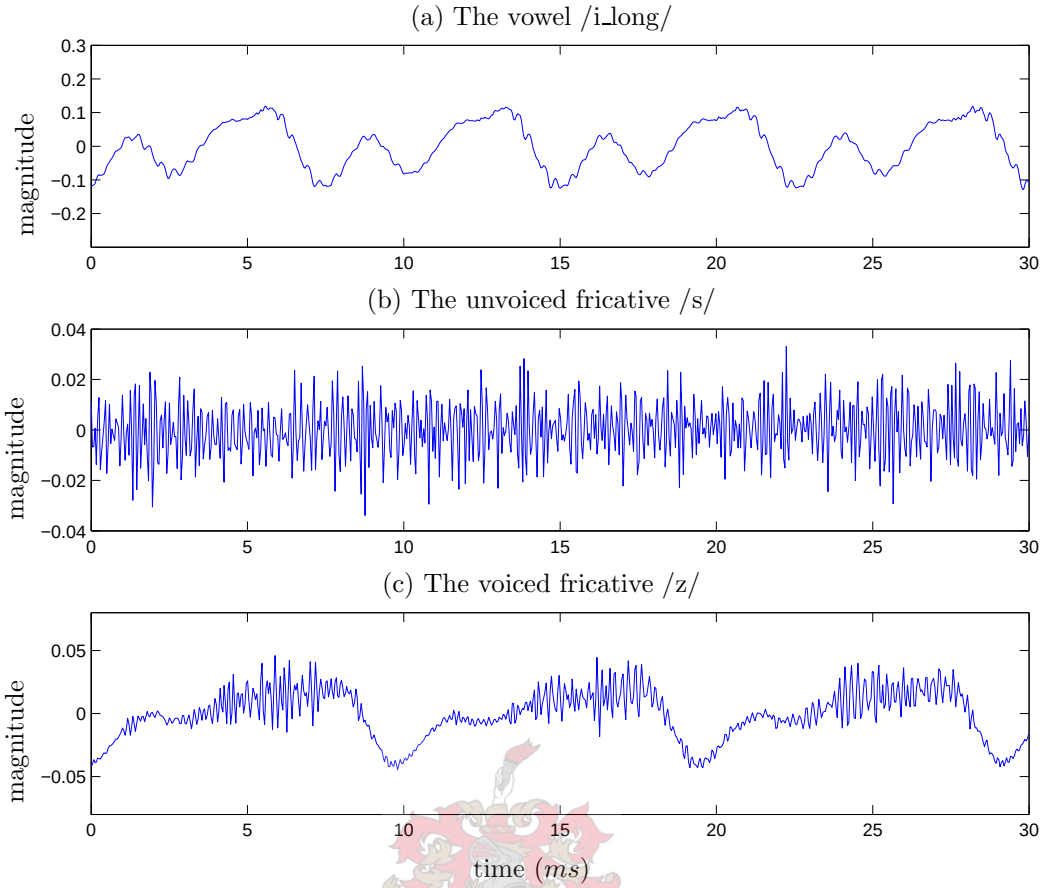


Figure 2.3: *Examples of various discrete time speech sounds.*

example, note the difference between the voiced sounds (such /sw/ and /ct_long/)[3] and the unvoiced sounds (such as /s/ and /f/). The voiced sounds contain some periodic structure, noticeable by the almost evenly spaced peaks in the waveform, whereas the unvoiced sounds have no discernible time structure. Figure 2.3 illustrates this more clearly by showing some closer views of the vowel /i_long/ ("k*ee*p"), the unvoiced fricative /s/ ("*s*ome") and the voiced fricative /z/ ("*z*ero"). As we would expect, the voiced fricative sound has a repeating waveform, but also contains an element of noise.

The periodic nature of voiced sounds is what causes what we perceive as the tone of the sound. Referring again to figure 2.3(a), we can see that the waveform repeats roughly every $7.5ms$. This interval is called the pitch period of the sound, designated by T_0, and its reciprocal the pitch, designated by F_0, where in this case $F_0 \approx 133$ Hertz (Hz), which is a typical value for an adult male speaker. In terms of the human speech production system, F_0 is the rate at which the glottis opens and closes. Different vowel sounds have different

[3]The phones and examples of words in which they occur can be found in Appendix A.

waveform shapes, but they are all roughly periodic by nature due to the sequence of energy pulses generated at the glottis. The shape of the vocal tract at the time the sound is made is what causes these differences in waveform shape and therefore sound, and in turn allows human ears to distinguish between the various sounds.

The duration of the signals shown in figure 2.3 is $30ms$, which is a typical window length for speech signal analysis because of the near stationary nature of the speech signal over such short intervals. Analysis windows must also be long enough to contain all the necessary information for a parameter estimation algorithm to perform well. Generally, the longer the window the better the model, but this only holds for stationary signals. The time-varying nature of speech limits the length of speech signal analysis windows. However, if we are able to guarantee that a sound remains constant for an extended duration, we may improve the accuracy of our estimated model to some extent. The use of monophones gives us the ability to record such sustained speech sounds, since most monophones do not require pronounced vocal tract movements. Exceptions in South African English are stop sounds, also called plosive sounds, such as /k/ ("*k*ick"), /p/ ("*p*it") and /d/ ("*d*eath"), and diphthongs/tripthongs. More attention will be given to the modelling of plosive sounds in section 3.4. Diphthongs and tripthongs can be produced by a smoothed transition between the individual phones they consist of. For example, the diphthong /vt_Inkic/ ("f*i*ne") can be synthesised using a smoothed transition between the vowels /vt/ ("p*u*blic") and /ic/ ("h*i*m"). The modelling of such smooth transitions is treated in chapter 4.

2.2.1 Speech Spectra

Frequency analysis is one of the most powerful tools in signal processing. It aims to determine which frequency components are present in a signal and how prominent each component is. The spectrum of a continuous time signal $x(t)$ is obtained using the Fourier transform, which is defined by the following integral:

$$X(f) = \mathcal{F}\{x(t)\} = \int_{-\infty}^{\infty} e^{-j2\pi f\tau} x(\tau)\, d\tau \tag{2.1}$$

where $X(f)$ is the Fourier transform (spectrum) of $x(t)$ and f is the frequency in Hz. For discrete time signals we use the discrete Fourier transform (DFT), which is defined as:

$$X(f_\omega) = \sum_{n=-\infty}^{\infty} e^{-j2\pi f_\omega n} x(nT) \tag{2.2}$$

where $f_\omega = fT$ is the frequency in cycles/sample, n is the sample index, and $T = \frac{1}{F_s}$ the sampling period. In practice, $X(f_\omega)$ is found by applying the FFT, which is a computationally efficient algorithm for calculating the DFT. Note that $X(f_\omega)$ is calculated by summing from $-\infty$ to ∞, but in practice we observe only a finite number of samples N such that $n = 0 \ldots N-1$. This means that we can merely estimate the spectrum of a discrete time signal because we have to make assumptions concerning the unobserved samples. Because

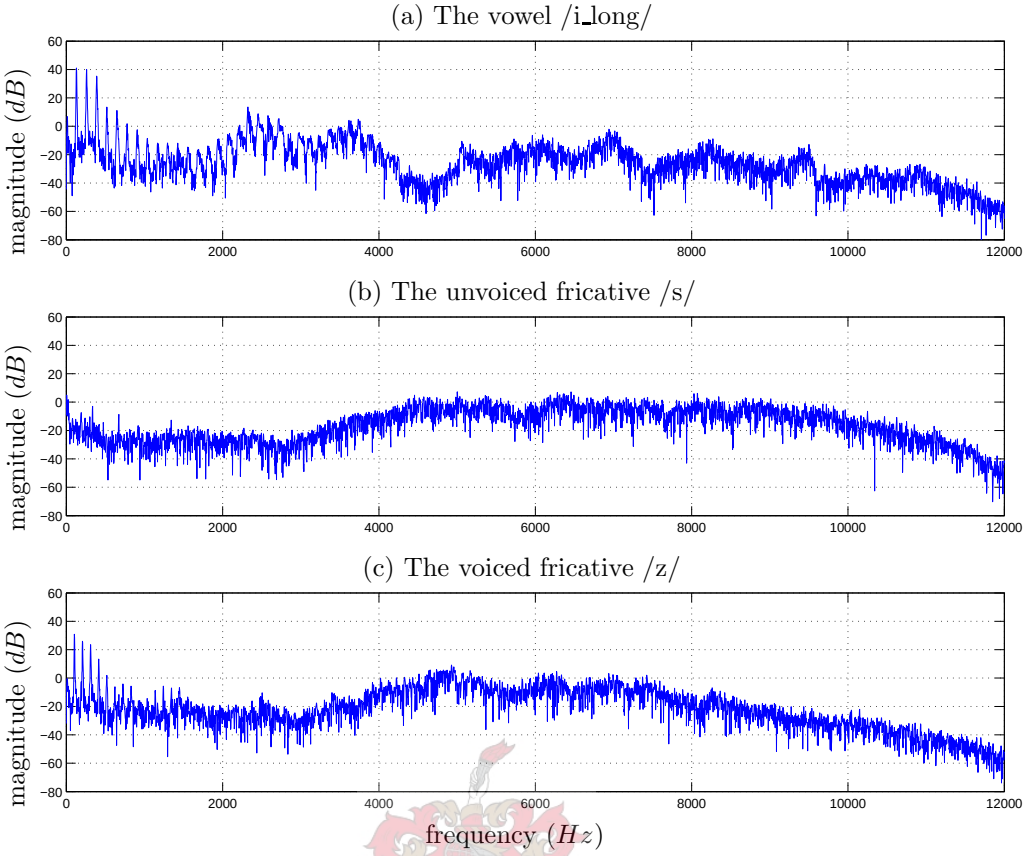


Figure 2.4: *Examples of speech signal spectra for various sounds.*

of this, numerous algorithms exist which estimate the spectrum of a signal, each with its own advantages and disadvantages [20]. The DFT forms the basis of most spectral estimation in speech processing, and will be used to calculate all spectra shown in this thesis, unless otherwise noted. Equations 2.1 and 2.2 both give complex-valued answers. The most common way of representing these complex values is to separate them into magnitude and phase components. The power spectrum is the squared magnitude of the spectrum, and the phase spectrum represents the phases. In this section we will limit our discussion to power spectra and we will represent their values on a logarithmic scale to better display the low amplitudes. Phase is often ignored in speech signal processing due to the fact that the human ear is insensitive to phase.

Figure 2.4 shows the spectra of some speech sounds. Note the differences between the different spectra, specifically the evenly spaced peaks in the lower frequency range of 2.4(a) and (c). These are due to the voicing of these sounds and are called harmonics. Their spacing is equal to the frequency of the lowest frequency harmonic, called the fundamental harmonic F_0, which is also the pitch of the sound. Counting the number of peaks up to $2kHz$ in 2.4(a) gives us 15, and $\frac{2000}{15} \approx 133Hz$, which is the same as the pitch we estimated for

the discrete time signal of the same sound in section 2.2. Note that 2.4(b) has no harmonics because /s/ is an unvoiced sound, and is hence not periodic. Also note that the energy of the noisy frequency bands ($4000 < f < 10000$) of the spectrum of 2.4(b) and (c) are larger than that in (a) because of the strong unvoiced component in both /s/ and /z/.

Another interesting fact about speech signal spectra is that the energy seems to be concentrated in certain frequency bands, called formants. Generally, the first three formants (F_1, F_2 and F_3) can be identified without much difficulty, but the higher formants are not always as easily discerned. For example, figure 2.4(a) shows the energy to be concentrated roughly around $300Hz$, $2400Hz$ and $3700Hz$. Formants are caused by resonances in the vocal tract, and the frequencies and bandwidths of these resonances depend on the shape of the vocal tract, i.e. the sound being made. It is these resonances, in conjunction with the voiced and unvoiced excitation signals, which define the particular sound being produced and allow a listener to distinguish between the various possible speech sounds.

As is evident in figure 2.4, the majority of the voiced component of a speech sound is found below $4kHz$. This, together with the fact that telephone speech has a bandwidth of $4kHz$, is the reason why many systems use $F_s = 8kHz$. However, noting the spectra in figure 2.4, we can see that there is still a lot of information in the signal above $4kHz$, but its energy is comparatively low. Although speech recorded at $8kHz$ is intelligible when played back, its quality is fairly poor when compared to speech recorded using higher sampling rates. For the purpose of high quality model estimation for synthesis, we need to choose F_s large enough to encompass all spectral information that is conveyed in a speech signal. Experiments indicate that this information is concentrated below $12kHz$, and therefore the sampling rate of the signals used to calculate the spectra in figure 2.4 is $F_s = 24kHz$.

The spectrum of a speech signal shows us some of the components contained in the signal more clearly than a time series. It also seems to support the notion of the source-filter model of speech production if we note that each spectrum has a particular envelope, and that this envelope is unique for each sound. Remembering that time domain filtering (convolution) is equivalent to frequency multiplication by an envelope (the filter's spectrum), it makes sense to assume that the speech spectrum envelope is representative of the filter part of the model, and everything else can be considered to be the source. As noted in section 1.4.2, there are various techniques that can be applied to separate these two components. Sections 2.3 and 2.4 deal with the modelling of the filter and chapter 3 concerns itself with the modelling of the source signal.

2.2.2 Spectrograms

As we have seen, the spectrum of a speech signal provides a very useful visualisation of its frequency components, but it is limited to a stationary segment of the signal. To fully represent a time-varying spectrum would require a three dimensional system of axes, which is impractical on paper. To overcome this problem, we make use of the spectrogram, which is a two dimensional (frequency versus time) representation using shades of colour to represent

"Listen to the forecast."

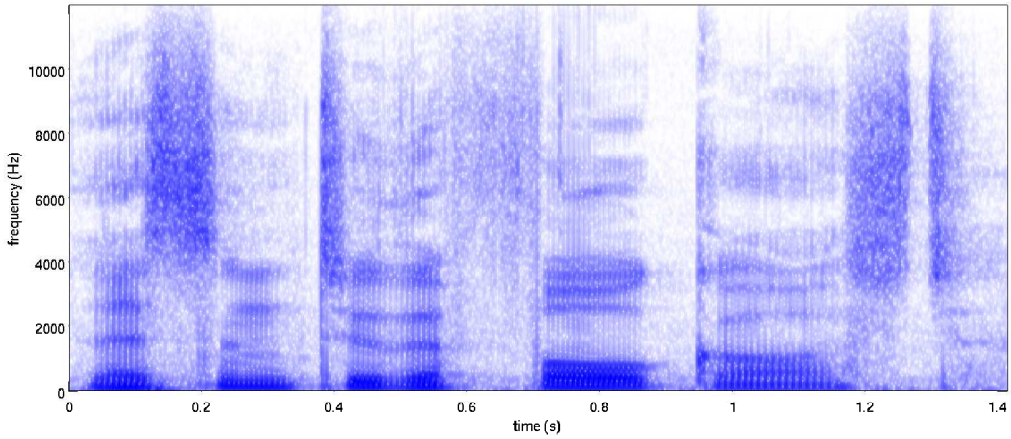


Figure 2.5: *Wideband spectrogram of a speech signal.*

"Listen to the forecast."

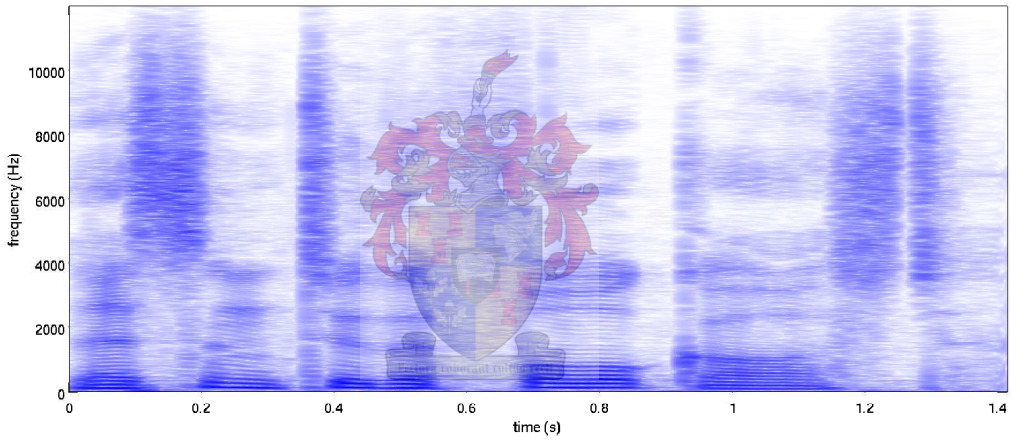


Figure 2.6: *Narrowband spectrogram of a speech signal.*

the third dimension (magnitude). The spectrogram is obtained by segmenting the speech signal into frames, which are often chosen to overlap to increase resolution on the time axis. The DFT is then used to calculate the spectrum of each frame, and its magnitude values are then represented as coloured points on the frequency axis. The frame length has a visible effect on the spectrogram, and we can therefore distinguish between wideband (shorter frame) and narrowband (longer frame) spectrograms.

Figures 2.5 and 2.6 show a wideband and a narrowband spectrogram, respectively. The frame length is $10ms$ in the wideband case and $50ms$ in the narrowband case and a $2.5ms$ step size (distance between successive spectra) was used in both cases. For reference, the speech signal used in both cases is the same as that of figure 2.2. The difference between the two spectrograms is a consequence of the trade-off between time resolution and freqency

resolution. When the frames are short, less information is available to accurately estimate the frequency components, resulting in lower frequency resolution. Using longer frames improves this, but has a "smearing" effect on the time axis due to the time averageing introduced by the extended frames. Upon closer inspection of the two spectrograms, we note that the harmonic tracks are clearly visible at the voiced sounds in the narrowband spectrogram, but are hardly discernible in the wideband case due to the lack of frequency resolution. However, there are vertical striations present in the wideband case which are not visible in the narrowband spectrogram due to the time averageing. These are indicative of the roughly periodic energy pulses caused by the vibrating vocal chords. The unvoiced sections in both spectrograms are also clearly noticeable by their stochastic nature.

In both spectrograms we see broad bands across time which have different frequency positions. These indicate the changing formant locations (refer to section 2.2.1), and are known as formant tracks. The ability to visualise this makes the spectrogram a very valuable tool with which to analyse the speech signal because of the wealth of information contained in the formant locations and movements. This information is so complete that it is possible for trained individuals to "read" formant tracks and derive the original utterance from a spectrogram accurately. From a modelling perspective, this is encouraging because it suggests that there is enough information contained in the spectral envelope to distinctly represent the various speech sounds. If we are able to capture this information in a filter model, it should be possible to re-synthesise the original sound. Extraction of the filter information from the speech signal is exactly what LP and cepstral analysis techniques are aimed at. These techniques are described in sections 2.3 and 2.4, respectively.

Although figures 2.5 and 2.6 do not show formant tracks for most of the unvoiced consonants in the utterance, this does not imply that unvoiced sounds never exhibit formant structure. They are missing in this case is because the majority of the unvoiced sounds in the utterance, such as /s/, /t/ and /f/ have their frication generated near the front of the mouth, which means that there is a shorter section of the vocal tract containing resonances which can modify the spectral characteristics of the signal. Upon closer inspection of the /k/ sound at about $0.95s$, we can see formant behaviour, albeit limited due to the short duration of the plosive sound.

2.3 Linear Prediction

The term "linear prediction" refers to the idea that one can predict the n^{th} value of a discrete time signal by a linear combination (weighted sum) of previous values. Of course, this is an approximation and there is usually some prediction error due to the difference between the predicted value and the actual value. More formally:

$$x(n) = -\sum_{k=1}^{p} a_k x(n-k) + \varepsilon(n) \tag{2.3}$$

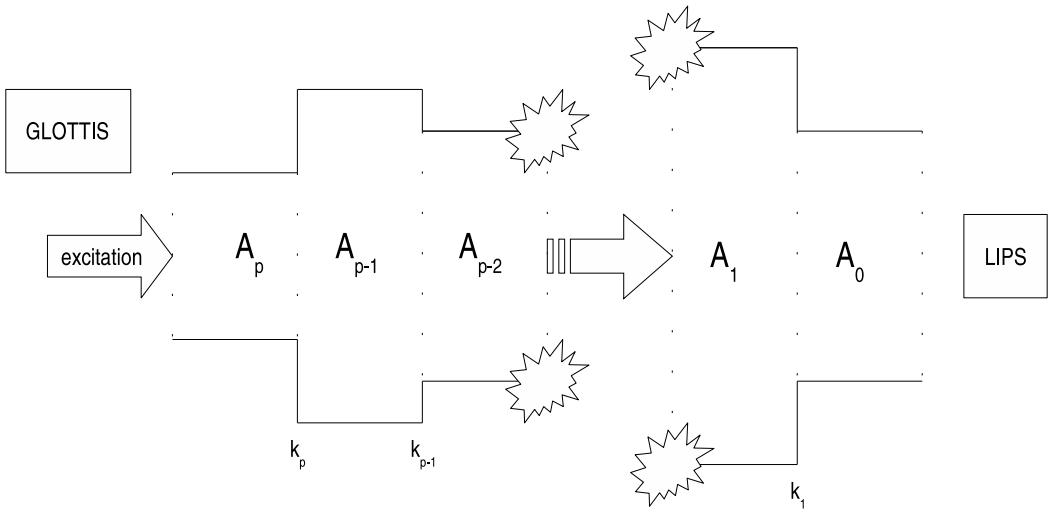


Figure 2.7: *The tube model of speech production.*

where $x(n)$ is the actual n^{th} (current) signal value, p is the number of past values included in the model (called the LP order), $-\sum_{k=1}^{p} a_k x(n-k) = \hat{x}(n)$ is our estimate of $x(n)$, $\varepsilon(n)$ is the unpredictable part of the signal, also termed the LP residual, and $a_1 \ldots a_p$ are the linear predictor coefficients (LPC's). Rearranging, we find:

$$x(n) - \hat{x}(n) = \varepsilon(n) \tag{2.4}$$

which shows us that the residual $\varepsilon(n)$ is equal to the prediction error. If we rewrite equation 2.3 by defining $a_0 = 1$, we find:

$$\sum_{k=0}^{p} a_k x(n-k) = \varepsilon(n)$$

Now, applying the $\mathcal{Z}$-transform both sides to obtain its frequency equivalent, we find:

$$\begin{aligned} \mathcal{Z}\{\sum_{k=0}^{p} a_k x(n-k)\} &= \mathcal{Z}\{\varepsilon(n)\} \\ A(z)X(z) &= E(z) \end{aligned}$$

or

$$X(z) = \frac{1}{A(z)} E(z) \tag{2.5}$$

where $X(z)$ is the $\mathcal{Z}$-transform of $x(n)$, $E(z)$ is the $\mathcal{Z}$-transform of $\varepsilon(n)$ and $A(z) = a_0 + a_1 z^{-1} + \ldots + a_p z^{-p}$.

Equation 2.5 represents the linear predictor as a filtering operation $\frac{1}{A(z)}$ on the prediction error $E(z)$ to obtain the signal $X(z)$, which fits in very well with the source-filter model of

speech production. In fact, equation 2.3 is one form of the equation which defines a lossless acoustic tube consisting of p cylindrical segments, as shown in figure 2.7, each with area A_i so that $A_i = \left(\frac{1-k_i}{1+k_i}\right) A_{i-1}$, where the coefficients k_i are termed reflection coefficients (RC's) because they define the amount of acoustic energy reflected at each cylinder boundary. RC's can be derived from the LPC's and are discussed further in section 2.3.6. Because the human vocal tract may be viewed approximately as an acoustic tube (see section 2.1), there is a physical link between the LP filter and the actual speech production process.

2.3.1 LP parameter estimation

For the LP model to be used, we must find some way of calculating the linear predictor coefficients a_k for $k = 1 \ldots p$ in equation 2.3. Calculating the LPC's for some known speech sound will yield a parametric model for that particular speech sound, one which we may use to synthesise speech at a later stage. If we have one such model for every phoneme in a language, we have a set of models which defines that language phonetically.

If one assumes in equation 2.3 that $\varepsilon(n)$ is uncorrelated with all $x(n-k)$ for $k > 0$, i.e. all past values of x, one can derive the *Yule-Walker* equations given by:

$$\mathbf{Ra} = -\mathbf{r} \tag{2.6}$$

where $\mathbf{r}$ is the autocorrelation sequence vector $[r_{xx}(1) \ldots r_{xx}(p)]^T$ of $x(n)$, $\mathbf{R} = \mathcal{E}\{\mathbf{x}(n)\mathbf{x}^H(n)\}$ is the $p \times p$ correlation matrix of $x(n)$ and $\mathbf{a}$ is the LPC vector $[a_1 \ldots a_p]^T$. Knowing this, we can find $\mathbf{a}$ directly from 2.6 via:

$$\mathbf{a} = -\mathbf{R}^{-1}\mathbf{r} \tag{2.7}$$

There are several methods for estimating the ACF of a discrete time signal x (and therefore $\mathbf{a}$) with a limited number of samples, of which the most popular approach is known simply as the autocorrelation method. This method is preferred above other methods such as the covariance and modified covariance methods because it guarantees that the parameters will represent a stable filter $A(z)$. An unstable filter is unsuitable for synthesis because its output values increase exponentially over time, causing severe distortions to occur in the synthetic utterance. In practice, we use the *Levinson-Durbin* recursions to calculate the LPC's when using the autocorrelation method because it is much more efficient than direct evaluation of equation 2.7, which involves matrix inversion.

A critical factor in the use of the LP model is the choice of the LP order p. Given the earlier reference to the tube model of the vocal tract, one would be inclined to think that the higher the LP order, the better the model. Although this may be true theoretically, we find in practice that the LP filter begins to model not only the vocal tract filter, but also the periodic structure of the voiced excitation signal. This is because equation 2.3 assumes the prediction error is uncorrelated to past values of the output, which is mostly true within a pitch period. However, once p becomes large enough for the predictor to encompass more than one pitch period, the correlations between successive pitch periods

begin to have effects on the estimated filter model, which is undesirable because we wish to model only the vocal tract filter using LP. Another limiting factor on p is the fact that a model with more parameters requires more resources. It must be mentioned, however, that the required LP order is proportional to the sampling frequency of the speech signal being modelled if model accuracy is to be maintained. Typically, LP orders of around 8–12 are used for $8kHz$ signals, whereas orders of 16–22 are necessary for $16kHz$ signals. Since the signals analysed in this chapter were all recorded at $F_s = 24kHz$, we expect that p should be around 24–32 to accurately model the vocal tract filter.

2.3.2 LP speech spectra

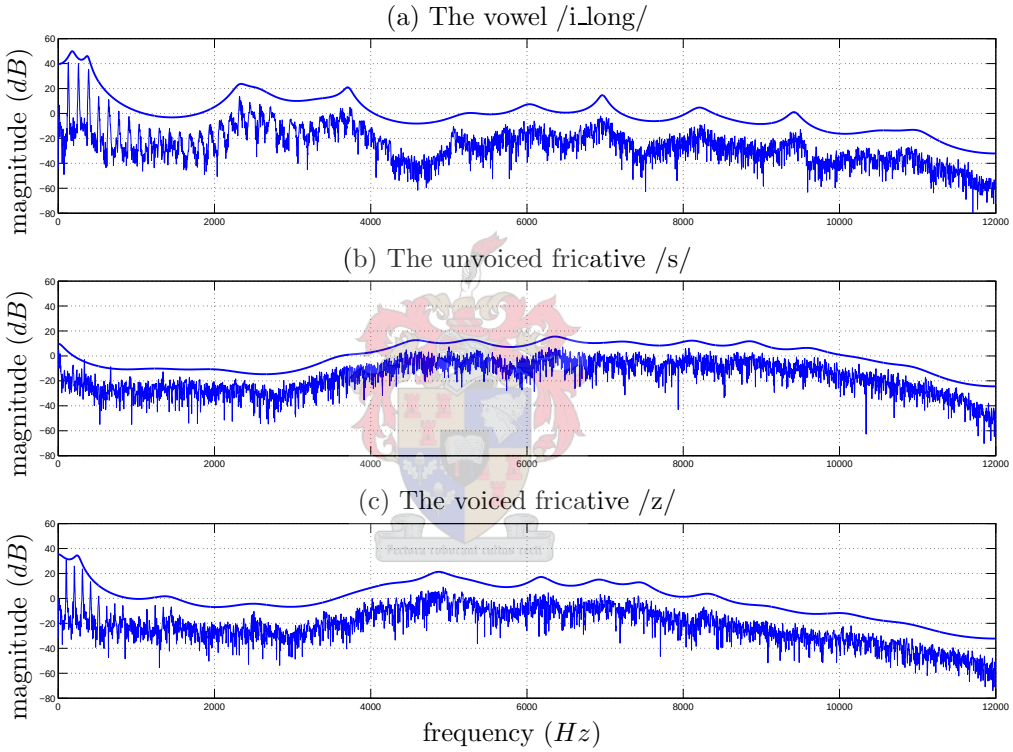


Figure 2.8: *LP spectra of different speech sounds.*

Once the LPC's have been calculated, they form the desired all-pole vocal tract filter model given by $\frac{1}{A(z)}$, called the LP filter. This filter model estimates the resonances in the vocal tract by modelling the spectral envelope of the given discrete time signal. To illustrate this, figure 2.8 shows the LP filter (the bold, "smooth lines") together with original spectra of figure 2.4 for reference. Note how closely the LP filter models the general frequency characteristics of the speech signal. Of special interest is the fact that the LP filter contains peaks around the formant frequencies, which are very useful for formant analysis. Note that the LP filters shown in figure 2.4 were obtained using $p = 30$. Using $p < 30$, we find that

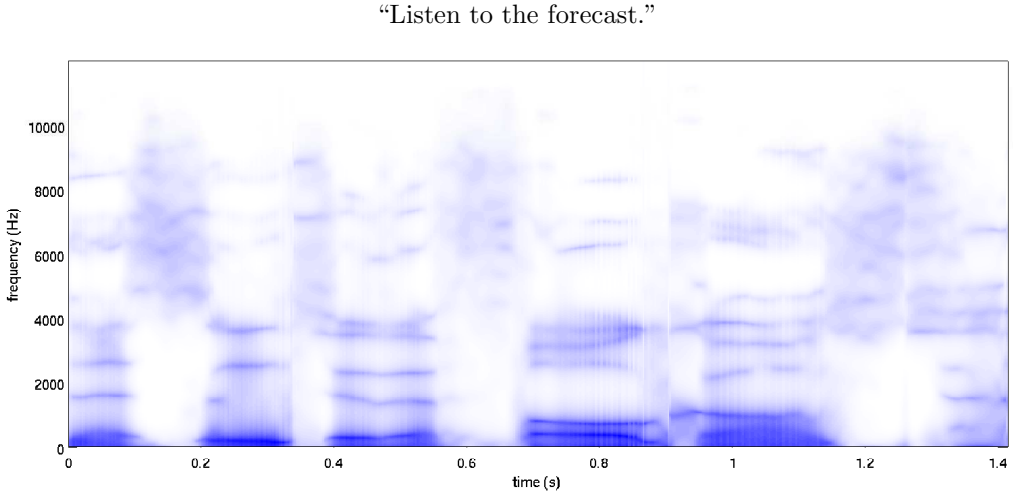


Figure 2.9: *LP spectrogram of a speech signal.*

the two formant peaks at about $200Hz$ and $400Hz$ are indistinguishable. This agrees with what we anticipated earlier (section 2.3.1) concerning the required p for $24kHz$ signals.

Because the LP filter differs for each individual speech sound, we may wish to view how the LP model changes over time for an entire utterance. Therefore, in a manner similar to that described in section 2.2.2, we calculate the LP filter of individual speech frames to obtain the LP spectrogram. Figure 2.9 shows the LP spectrogram of the same utterance of figures 2.2, 2.5 and 2.6. Note that we use a longer frame length of $50ms$ ($2.5ms$ spacing), the same as that of the narrowband spectrogram, to maximise frequency resolution, which is of primary importance for estimating the filter. As we would expect, we can see the formant tracks much more clearly than before, and the harmonic peaks are no longer discernible. This is encouraging if we consider that we can now model the individual speech sounds as well as their transitions using a parametric model. Chapter 4 deals with the modelling of the transitions between phones.

2.3.3 LP residuals

Now that we have a model for the filter component of the source-filter model of speech production, what remains is the source component. This excitation signal is already available as it is the by-product of LP analysis. Remember that equation 2.4 points to $\varepsilon(n)$ as the prediction error, but equation 2.5 indicates that $E(z)$ (and therefore $\varepsilon(n)$) is, in fact, the excitation signal that is being filtered by $\frac{1}{A(z)}$ to obtain the speech signal. If we then write equation 2.5 in its original form as:

$$E(z) = A(z)X(z)$$

we see that we can obtain $\varepsilon(n)$ by filtering $x(n)$ by the all-zero filter $A(z)$. This operation is called inverse LP filtering and gives us the LP residual $\varepsilon(n)$.

Many systems use the LP residual directly for synthesis, often in the form of a codebook of entries from which the appropriate $\varepsilon(n)$ is selected according to some criterion. Such strategies may offer more natural synthetic speech, but can suffer from the same data dependencies and concatenation discontinuities as ordinary concatenative schemes. For a flexible and data independent speech synthesis system, a parametric model of the excitation signal is required. It must be emphasised that it is the LP residual that is to be modelled, since LP filtering will be applied to the model $\varepsilon(n)$ during synthesis. Chapter 3 is dedicated to LP residual modelling, as it is not a simple matter and many different approaches are used in practice.

2.3.4 Pre-emphasis

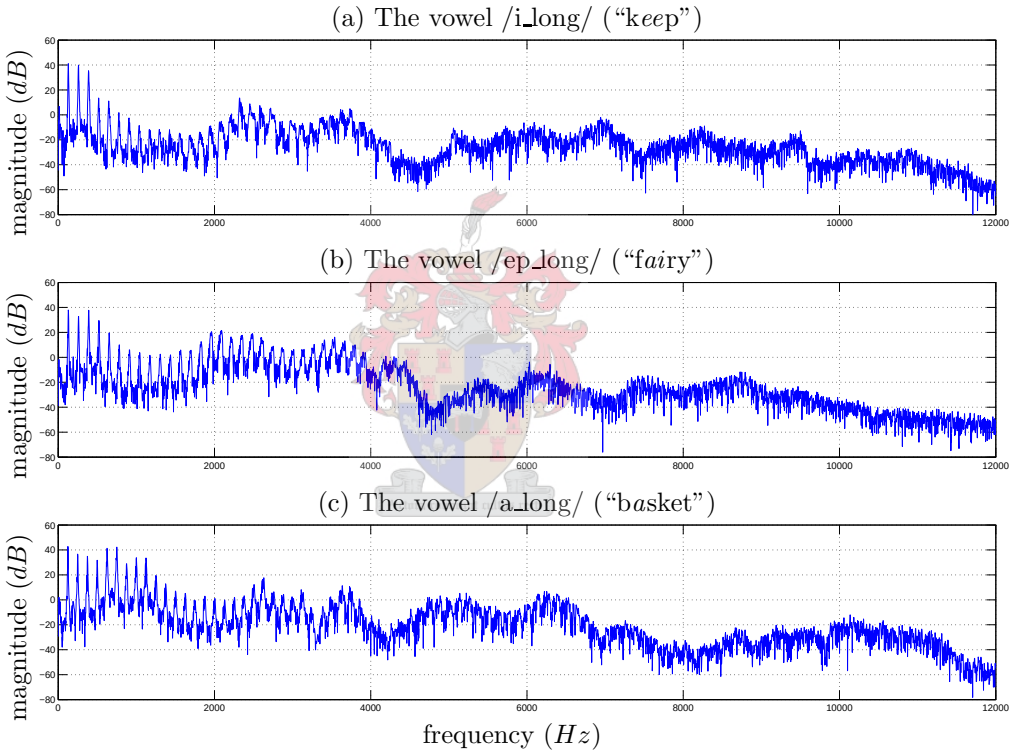


Figure 2.10: *Examples of vowel spectra.*

Figure 2.10 shows the spectra of three different vowels. As can be seen, vowel speech spectra have a fairly consistent downward slope (about $-20dB$ per decade) as frequency increases. For LP and other energy-based formant or envelope estimations, this leads to preferential modelling of the lower formants. To avoid this, some systems apply a simple single- or 3-zero highpass filter before estimating the parameters. This process is called pre-emphasis and is aimed at increasing the model's accuracy at the higher formants. Once the model has been estimated, the pre-emphasis needs to be reversed in order to retain the

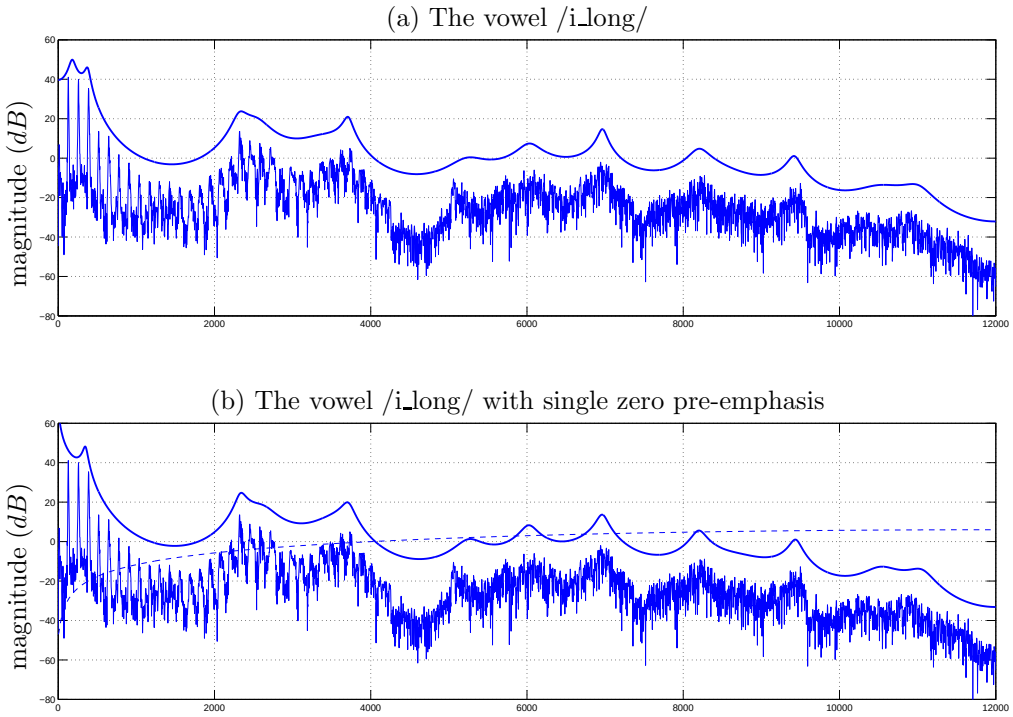


Figure 2.11: *The effects of pre-emphasis.*

natural speech slope. This is called de-emphasis and uses the inverse of the pre-emphasis filter.

Pre-emphasis is a very useful technique when F_s is low ($8kHz$) and/or when the LP order is chosen to be small. One can then clearly hear the difference in synthesis quality, since lack of high frequency resolution tends to make the synthetic speech sound more unnatural. However, when the sampling rate and the LP order are high, as in our case, the advantages of pre-emphasis are limited and its effect is not audible in the synthetic speech. In fact, applying a pre-emphasis filter in such a case merely degrades the LP model's performance when estimating the lower frequency components, especially when peaks are situated close together in frequency. Figure 2.11 illustrates this effect, where $p = 30$ in both cases and a single zero inverse LP filter was used as the pre-emphasis filter in 2.11(b), indicated by the dashed line. Again, the LP filters are indicated by the thick lines above the spectra. Note that the LP filter in 2.11(b) does not discern two peaks below $500Hz$ as in (a), but has slightly more defined peaks in the regions above $2kHz$ than the LP filter in (a).

2.3.5 Warped LP

It has often been noted that certain frequency bands are perceptually more important than others [36]. This psychoacoustic phenomenon has been applied to the modelling of speech

signals by warping of the frequency scale such that the perceptually more important frequency bands are exaggerated. Frequency warping as it is applied to linear prediction is defined in [41]. As mentioned in [19], the main advantage of frequency warping is the reduction of the model order p without a significant loss in quality. Warped linear prediction is achieved by modifying the computation of the ACF.

Warpred LP was not used for this project because of the potential problems it introduces during synthesis (see [19]) as well as the fact that model order reduction was not considered a necessary step, at least not for an initial implementation.

2.3.6 LPC representations

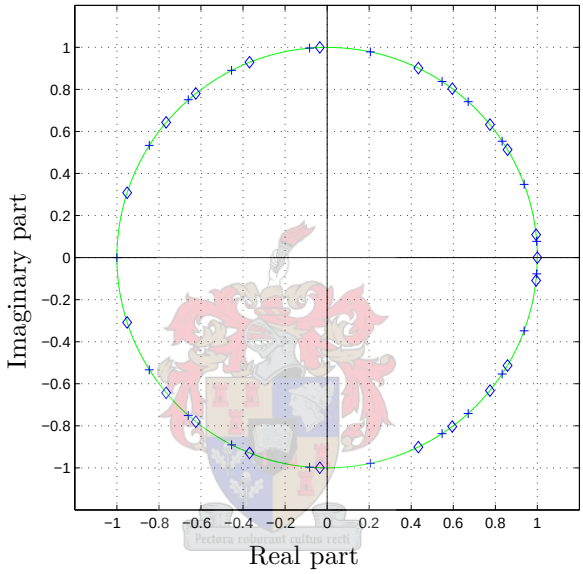


Figure 2.12: *LSF locations.*

Because $A(z)$ is a polynomial, there are many different ways to represent its coefficients. For example, one can use the LPC's directly or the real and imaginary parts of its poles, or their magnitudes and phases. Another representation, which is a by-product of the *Levinson-Durbin* recursions, is the set of RC's $k_1 \ldots k_p$ referred to in section 2.3. RC's have the useful property that their values are always in the range $-1 \leq k_i \leq 1$. This property makes them more suitable candidates for quantisation in applications such as low bit-rate speech coding than the LPC's themselves.

Another representation of the LPC's is by the line spectral frequencies (LSF's) as defined in [18]. These are defined by first forming the symmetric and anti-symmetric polynomials, $P(z)$ and $Q(z)$ respectively, of $A(z)$, defined as follows:

$$P(z) = A(z) + z^{-(p+1)}A(z^{-1}) \tag{2.8}$$

$$Q(z) = A(z) - z^{-(p+1)}A(z^{-1}) \tag{2.9}$$

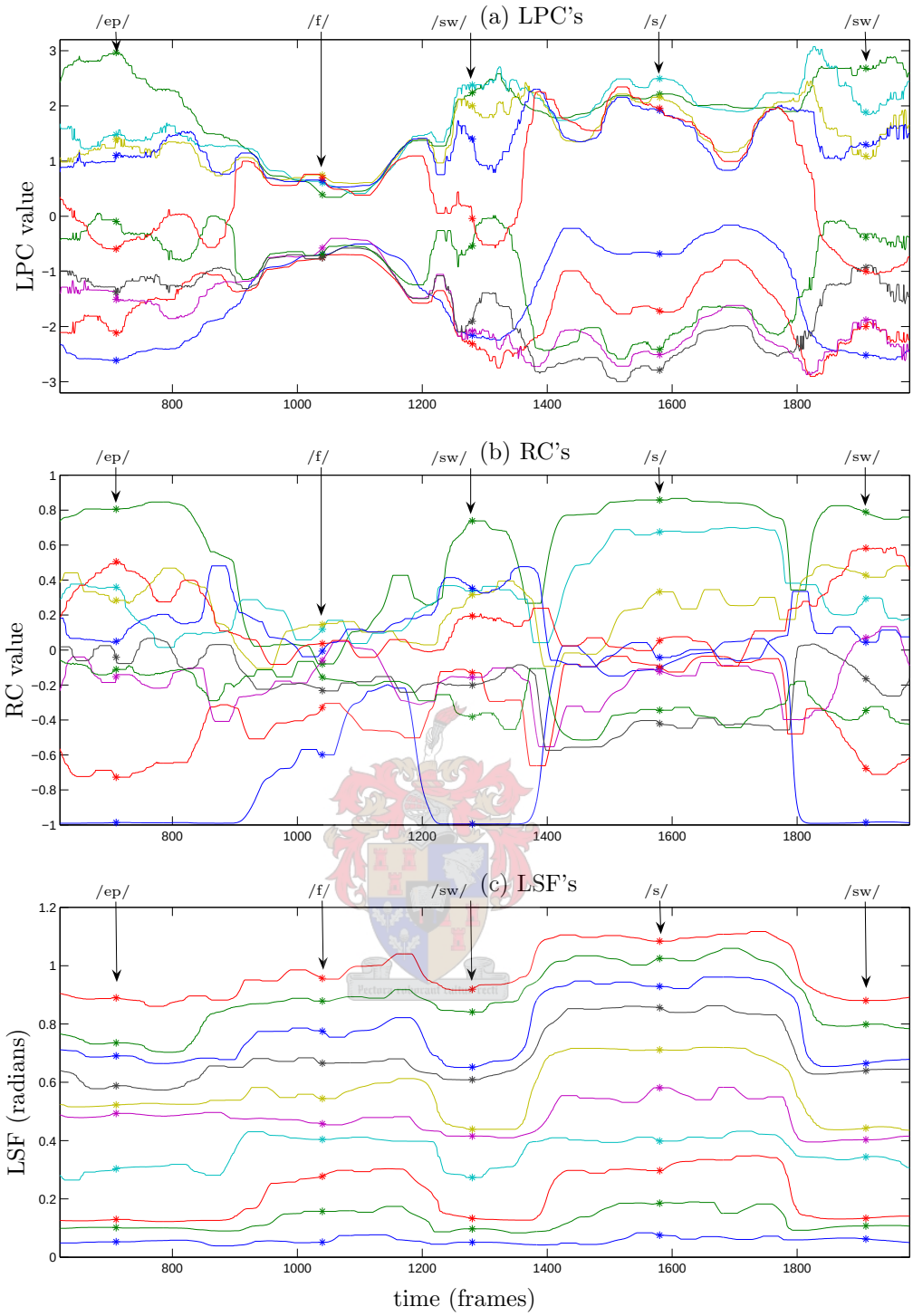


Figure 2.13: *Different LPC representation trajectories within the word "d**efici**t".*

The LSF's are defined using roots of $P(z)$ and $Q(z)$. Figure 2.12 shows the locations of these for a 20-pole LP filter. The pluses indicate the roots of $P(z)$ and the diamonds the roots of $Q(z)$. Note that all these roots are located on the unit circle and form complex conjugate pairs, except for the two real roots at 0 and π, which are always present. Also note that the roots of $P(z)$ and $Q(z)$ are interleaved on the unit circle. For these reasons, it is sufficient to

define the LSF's as the angles of those roots which are located above the real axis (positive imaginary part). LSF's have found widespread use in coding schemes because, just as RC's, they are limited to a finite range of values. Also, LSF's are very stable parameters when used in interpolation schemes, since interpolating between two sets of LSF's guarantees a stable filter [23].

Figure 2.13 shows the trajectories of the three different representations of the LP parameters within a word, with the centre locations of the various sounds indicated by asterisks. The parameters were obtained by estimating a 30^{th} order LP filter for each $30ms$ frame in increments of about $0.4ms$, after which median filtering was applied to remove estimation errors. Only the first ten parameters are shown in each case for clarity. Note that the LSF's in (c) are, as stated earlier, always sequentially ordered and that the different LSF graphs therefore never cross each other. Also note that there is a strong correlation between the trajectories of the successive LSF's and, in fact, similarities between the transitions between different speech sound pairs. Contrast this to the LPC and RC trajectories, which do not appear to behave consistently in the transition regions between sounds, although there are correlations between the coefficients. In addition to the advantages of the LSF representation of LPC's mentioned earlier, these new insights seem to indicate that LSF's are particularly suited to modelling transitions between sounds using interpolation methods. This suggests that it may be possible to synthesise speech using only a sequence of parametrically defined phones. Such an LSF interpolation strategy is developed section 4.1.

2.3.7 Synthesis using LP

LPC's model the vocal tract transfer function, but not the excitation signal. Therefore, although we have a parametric model for the filter part of the speech signal in the form of an LPC vector, we require excitation signals in order to produce synthetic speech using linear prediction. Some systems contain a database of LP residual signals from which the synthesiser selects an appropriate entry for each generated frame. Although this is probably the approach that yields the most natural speech output, it requires significant storage capacity. More importantly: prosody is modelled implicitly; hence the method is data-dependent and not easily portable. Other systems, including the one developed in this thesis, attempt to model the LP residuals parametrically, thereby keeping the number of parameters required for the excitation signal to a minimum and allowing precise control of the prosodic information carried by the excitation signal. Chapter 3 presents some of the models typically used for this purpose.

2.4 The Cepstrum

Within the framework provided by the source-filter model of speech production, the cepstrum is an alternative method which aims to separate source from filter. It achieves this by applying a technique known as homomorphic filtering. Although the cepstrum was not used

to characterise sounds in this work, it does represent an alternative that could be employed in future, and will therefore be described briefly in the following.

2.4.1 Homomorphic filtering

As we have already seen, the vocal tract transfer function is a smoothed version of the speech spectrum and the excitation consists of a quickly varying ripple in frequency, be it voiced or unvoiced. Normally, if we wish to separate the slowly varying components of a signal from the rapidly varying components, we simply apply linear lowpass and highpass filters. This is not possible in this case, however, as linear filters can only separate signals of different frequencies that have been added together and we wish to separate two signals which have been multiplied:

$$X(f) = V(f)E(f)$$

where $X(f)$ is the speech signal spectrum, $V(f)$ is the vocal tract transfer function spectrum and $E(f)$ the excitation spectrum. The solution is found when we consider that the logarithm of a multiplication yields a summation:

$$\begin{aligned} \log X(f) &= \log\left[V(f)E(f)\right] \\ &= \log V(f) + \log E(f) \end{aligned}$$

The definition of the cepstrum of a speech signal $x(t)$ then follows directly as:

$$c(t_c) = \mathcal{F}^{-1}\{\log \mathcal{F}\{x(t)\}\} \tag{2.10}$$

where $c(t_c)$ is the cepstrum and t_c is its unit of (relative) time.

Knowing this, we can find an estimate of the vocal tract transfer function $V(f)$ by lowpass filtering and the excitation signal $E(f)$ by highpass filtering of the cepstrum, and applying the following formulae:

$$\begin{aligned} V(f) &= e^{\mathcal{F}\{\check{c}_{t_c}\}} \\ E(f) &= e^{\mathcal{F}\{\hat{c}_{t_c}\}} \end{aligned}$$

where $\check{c}_{t_c}$ denotes the lowpass filtered cepstrum and $\hat{c}_{t_c}$ the highpass filtered cepstrum. This type of filtering is called homomorphic filtering. When applied to the cepstrum, some use the term "liftering" and refer to the t_c as units of "quefrency". For a discrete time signal $x(t)$, the cepstrum is the sequence of coefficients $c_0, c_1, c_2, \ldots c_{N-1}$, termed the cepstral coefficients, and each coefficient c_n corresponds to the frequency $\frac{F_s}{n}$. Choosing the filter passband is then equivalent to choosing the set of cepstral coefficients which represent the desired signal component, $c_0 \ldots c_h$ for the vocal tract transfer function and $c_g \ldots c_{N-1}$ for the excitation. Typical assumptions at this point are that $F_1 > 500Hz$ for $V(f)$ and $F_0 < 250Hz$ for $E(f)$. From the correspondence between cepstral coefficients and frequency, we find that $h = 48$ and $g = 96$ for $F_s = 24kHz$ under these assumptions. Note that the number of parameters necessary to encode the vocal tract transfer function, 48, is approximately 50% larger than the required LP order at the same sampling rate.

2.4.2 Cepstral vocal tract filter estimates

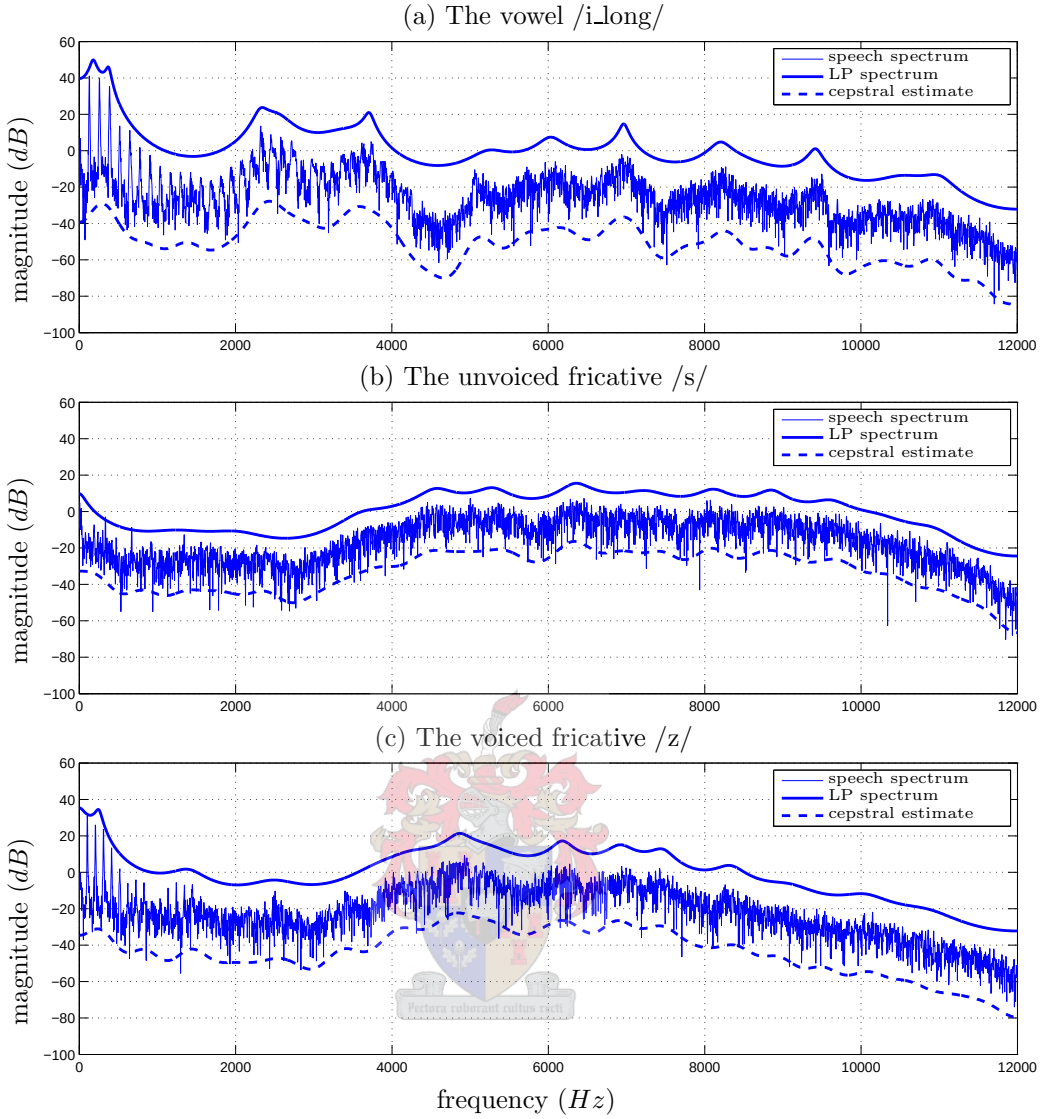


Figure 2.14: *Cepstral vocal tract filter estimate examples.*

Figure 2.14 shows the cepstral vocal tract spectrum estimates of the same speech sounds considered in previous sections. For reference, the original speech spectra as well as the LP filter estimates are shown as indicated. These figures illustrate some of the main differences between LP filters and cepstral smoothed spectra. Firstly, note that the cepstral estimates have rounded peaks, whereas the LP spectra peak more sharply. This is mainly due to the all-pole nature of the LP filter. Also note that this apparently causes the LP filter to be marginally less accurate in modelling the speech spectrum envelope at the spectral peaks, and even more so in the spectral valleys, while the cepstrum seems to follow the original spectrum more closely regardless of the peaks and valleys.

Another important difference that is evident in figure 2.14(a) and (c) is that difference in magnitude between the cepstral estimate and the LP spectrum is larger in the lower

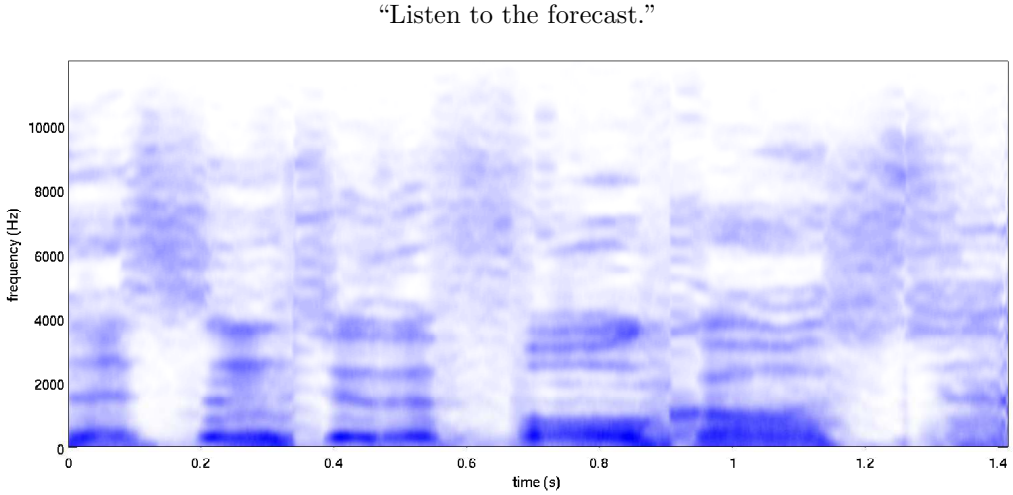


Figure 2.15: *Cepstral smoothed spectrogram of a speech signal.*

frequency range where the voiced energy is concentrated. This is an important difference because it illustrates that LP is an energy-based estimator as noted in section 2.3.4, whereas the cepstrum is not. It is also an important fact to consider when choosing a synthesis model, since the excitation signal model will have to take into account that an excitation signal which is to be filtered using a cepstral vocal tract filter estimate should be sloped more sharply in frequency than one to be LP filtered.

Figure 2.15 shows the smoothed spectrogram of the same utterance used in previous spectrograms, but using the cepstrum to estimate the vocal tract transfer function. The frames are $50ms$ in duration with $10ms$ spacing. Compared to figure 2.9, we find that the formant tracks are broader in the cepstrum case. This is because, as noted earlier, the LP spectrum has sharper peaks. The only other significant difference between the two spectrograms is the fact that the cepstrum appears to contain more energy at higher frequencies than the LP model (note the slightly darker upper frequency regions). This is because the colour scale used in the spectrograms is scaled according to the range of values in the given function and, as noted earlier, the LP filter models the energy envelope of the voiced speech in the lower frequency range, causing a larger range of values in the LP spectrogram. Other than these differences, the cepstral estimates appear to model the spectral envelope just as well as the LP estimates.

2.4.3 Mel frequency cepstral coefficients

As in the case of warped LP, frequency warping has been applied to cepstral coefficients. The frequency scale that is most often applied in the case of cepstral coefficients is the Mel scale (see [36]). In a typical application, the cepstral coefficients are calculated using a Mel (rather than linear) scale bank of filters, resulting in what is commonly referred to

as Mel frequency cepstral coefficients (MFCC's). In fact, MFCC's (together with their first and second derivatives) are the most widely used representation of the speech signal for recognition purposes.

2.4.4 Synthesis using the cepstrum

Because of the success of MFCC's in speech recognition as well as speaker recognition and verification, several researchers have begun to use MFCC's as a speech signal representation for speech synthesis in recent years. The advantages of doing so mostly stem from the well established statistical methods used to model speech signals in recognition systems. During synthesis, the parameters are generated by using ANN or HMM topologies similar to those usually used by recognition systems. The design an implementation of one such system is described in detail in [8].

As with LPC's, MFCC's represent the filter part of a speech signal, which means that an excitation signal is required for synthesis. Some of the same residual signal models presented in chapter 3 have been used for HMM synthesis. In order to impose the spectral envelope of MFCC's on the excitation signal, a Mel log spectral approximation (MLSA) filter is used (see [28] and its references, as well as [46]).

2.5 Chapter Summary

This chapter presented a short discussion of the human speech production system, after which we reviewed some established digital speech signal analysis methods. Among these, time waveforms and spectrograms were shown to be useful tools for visualising speech signals. Two approaches to source-filter modelling of speech signals were shown: linear prediction and the cepstrum. The advantages of the LSF representation of the linear predictor coefficients for use in a rule-based interpolation scheme were also highlighted.

For this work, $F_s = 24kHz$ was chosen for analysis and synthesis. Analysis frames of $250ms$ were used for parameter estimation, except in the case of plosive sounds, where a frame length of $33ms$ was chosen due to the short duration of these sounds. Linear prediction (usually $p = 30$) was chosen as the speech signal model because of its simplicity and flexibility, and because LSF's (which are derived from LPC's) are suitable candidates for monophone interpolation (see figure 2.13). A Hanning window was applied prior to LPC estimation. Although pre-emphasis was not applied to the final speech model which was used for our experiments, it was implemented for the sake of future modelling experiments. No frequency warping was applied.

Chapter 3

Excitation Signal Modelling

In this chapter, we assume the use of a source-filter model of speech production and turn our attention to the understanding and modelling of the source component of speech sounds. A flexible parametric model of the source (excitation) signal is required in order to avoid data dependency and to maximise the naturalness and intelligibility of the synthetic speech. In section 2.3.3, we saw that an estimate of the source component of the source-filter model of speech production can be obtained by performing inverse LP filtering. Although there are source estimation methods associated with other speech models, such as the cepstrum (section 2.4.1), examples in this chapter are restricted to LP residuals since the synthesis system developed in this thesis makes use of LP techniques to separate source from filter. Note, however, that the residual signal models and their associated parameter estimation techniques are not restricted to LP residual signals.

We firstly examine some LP residual signals belonging to different sound classes in order to gain a better understanding of the signal characteristics we wish to model. After this, we turn our attention to the modelling of these residual signals. At this stage, we present the most common way of modelling the unvoiced component of a residual signal, namely by means of Gaussian noise, as well as three different models that have been proposed for the voiced component. The first two of these, namely the impulse train and Rosenberg-Klatt model, are commonly used in current speech synthesis systems. The third, a sinusoidal voicing model, is less frequently used due to its higher complexity.

Next, several methods are presented for estimating the residual signal model parameters. With these methods, the speech synthesis system can automatically determine the parameters it requires to reproduce a monophone (in conjunction with its LSF vector). Among these are two established methods for estimating the Gaussianity[1] of a signal, which are used to determine the degree of voicing of a speech signal, as well as a third novel algorithm which accomplishes the same, but at a reduced computational load. Also presented is a modelling scheme for plosive sounds (which cannot be considered stationary monophones), as well as

[1]The term "Gaussianity" will be used to refer to the degree to which a signal's distribution approaches that of Gaussian data.

a discussion of speech prosody and how each of its components can be modelled using the excitation signal parameters.

3.1 Examples of LP Residuals

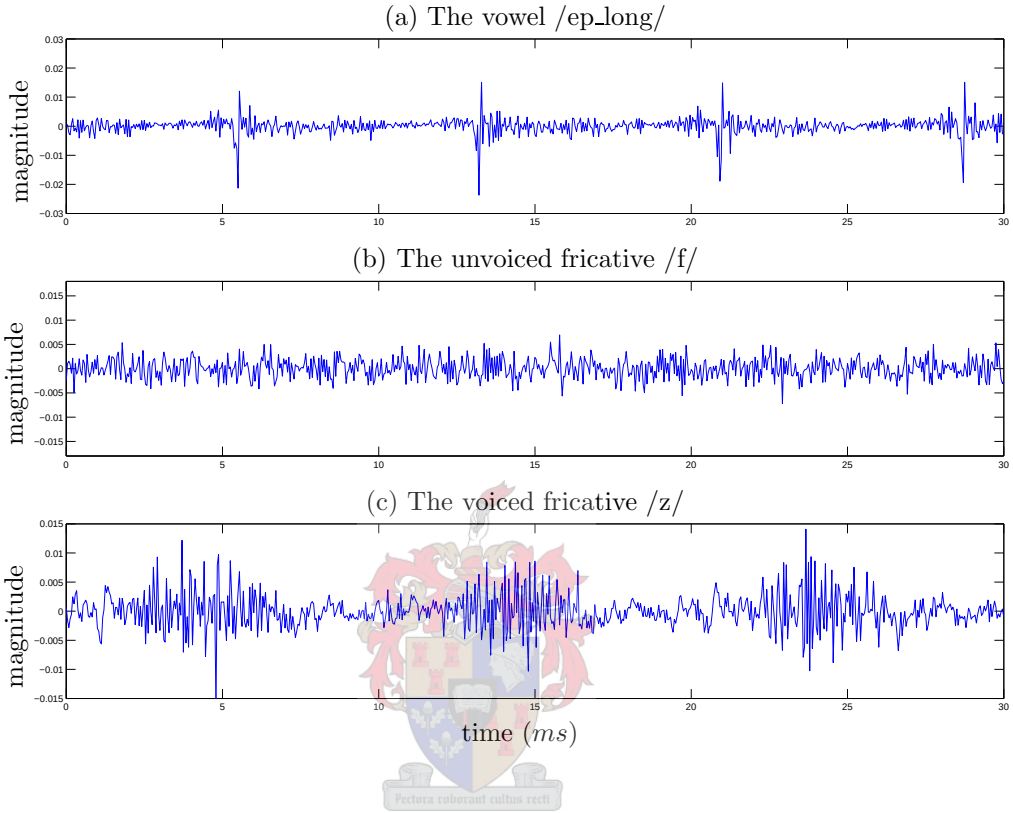


Figure 3.1: *LP residuals of different speech sounds.*

Figure 3.1 shows the LP residual signals for of the phones /ep_long/ (as in "f*ai*ry"), /f/ (as in "*f*our") and /z/ (as in "*z*ero"). Figure 3.2 shows the FFT magnitude spectra of these LP residuals. Note that the magnitude scale is no longer dB as was used in chapter 2. However, as was the case in chapter 2, the signals were sampled at $24kHz$ and the LP order used for inverse filtering is $p = 30$. No pre-emphasis was applied prior to LP estimation.

The first aspect of the excitation signal that is apparent from figure 3.1 is that the phones /ep_long/ and /z/ contain some periodic component, whereas /f/ seems aperiodic. As discussed in section 2.2, this is due to the voicing associated with the first two sounds. Voicing manifests itself in the spectra of these residuals as a band of harmonics in the lower frequency region (noted in section 2.2.1). The magnitude of these harmonics decrease with increasing frequency.

Furthermore, all the time signals of figure 3.1 also contain a stochastic (noise) component, which is visible in the spectra of figure 3.2 as spectrally flat (white) noise. For the voiced

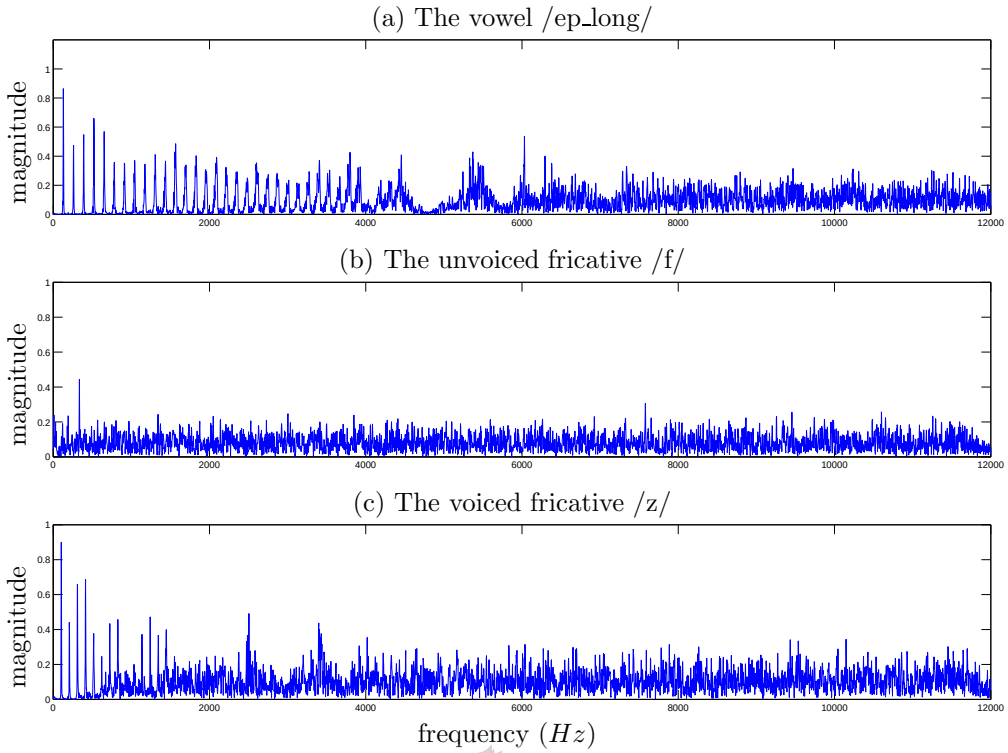


Figure 3.2: *LP residual spectra of different speech sounds.*

residual spectra (figures 3.2(a) and (c)), this occurs only at higher frequencies, beyond the band of harmonics[2]. The primary cause of this noise is the frication caused by the passing of air through the vocal tract. For simplicity, we have assumed that all noise introduced by the recording process is negligible. The noise component for the voiced sounds /ep_long/ and /z/ is suppressed among the band of harmonics, as noted in section 2.4.2, due to the "flattening" effect on the spectrum of inverse LP filtering. Spectral flattening tends to remove the energy envelope, which is why this type of filtering is often also called "whitening". Because of this, we find that the spectral energy (including that of the noise component) is suppressed more in regions of the speech spectrum where energy is concentrated. Figure 3.2(b) seems to support this argument, since the noise component in the purely unvoiced phone /f/ appears flat over all frequencies.

[2] "Band of harmonics" refers here to the lower frequencies where harmonic peaks are visible in the spectrum, up to the frequency where they are no longer discernible from the noise element.

3.2 Modelling LP Residuals

It is widely accepted in speech research that the two components discussed in section 3.1, namely voiced (harmonic) and unvoiced (stochastic), are sufficient to model the excitation function. Most approaches also assume that these two components are uncorrelated and can therefore be generated separately and combined linearly. In the following sections, we present some techniques often used in practice to model these two speech excitation signal components.

3.2.1 Unvoiced speech residuals as Gaussian noise

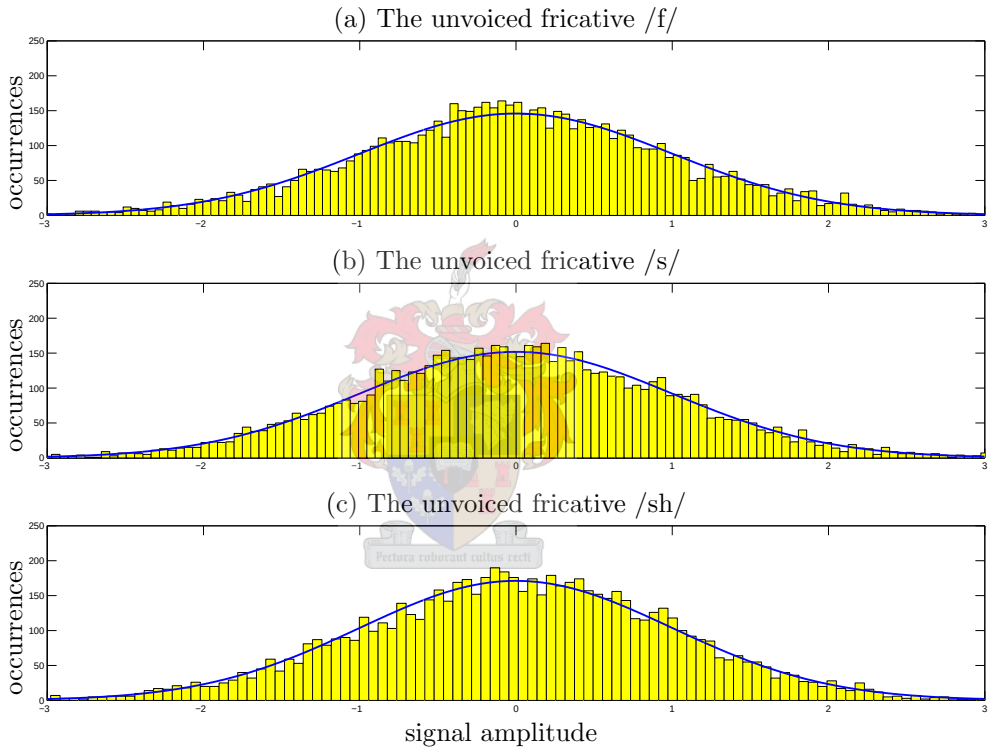


Figure 3.3: *Histograms of different unvoiced phone LP residuals.*

Figures 3.1 and 3.2 and the resulting discussions in section 3.1 have suggested that the stochastic component of a source signal may be assumed to have a flat spectral envelope. Although white noise is a very simple model for the stochastic component, it is well-suited to the problem as it agrees with observations of actual speech signals. It may, for reasons such as stated in section 3.1, be desirable to apply some type of filtering in order to modify the noise spectral envelope where it overlaps with the voicing spectral components, resulting in coloured rather than white noise. In the speech synthesis system developed in this thesis, however, informal listening tests seemed to indicate that such filtering is not crucial from an

audible perspective.

It is not sufficient to specify only the spectral envelope of the stochastic component. In order to generate the noise signal, the probability distribution governing the stochastic signal must be known. The shape of an appropriate distribution may be determined empirically by using histograms to estimate the distribution of signal values. Figure 3.3 shows the histograms of the LP residuals (after subtracting their respective means and dividing by their respective standard deviations) of the unvoiced fricative sounds /f/ (as in "*f*our"), /s/ (as in "*s*ome") and /sh/ (as in "*sh*ine"). Note that all three sounds have very similar sample distributions. Superimposed on the histograms are Gaussian density functions, scaled according to the number of samples considered. Note that the Gaussian density is a very good approximation of the actual sample distribution found in all three signals. Hence, the Gaussian probability density function (PDF) is often used to generate stochastic signal components in synthetic speech. For a one dimensional signal, it is defined in [32] as:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma_X^2}} e^{-(x-\mu_X)^2/2\sigma_X^2} \tag{3.1}$$

where x is the Gaussian random variable (signal), $\mu_X = \mathcal{E}\{x\}$ is its mean value and $\sigma_X^2 = \mathcal{E}\{(x-\mu_X)^2\}$ its variance. If $\mu_X = 0$, σ_X^2 is equal to the mean energy of the signal. The associated standard deviation σ_X will be denoted by A_n from this point onward. Although ideal white noise cannot be generated using conventional computational methods, there are numerous algorithms which are able to approximate the characteristics of Gaussian white noise adequately.

3.2.2 Voiced speech residuals as an impulse train

Closer inspection of figure 3.1(a) reveals that each pitch period contains a short section of values which are relatively large in magnitude (at approximately $5.5ms$, $13.5ms$, $21ms$ and $29ms$). These are indicative of the periodic energy bursts produced at the glottis for voiced sounds (section 2.1). A very simple way of approximating this is by means of an impulse train, which is defined as:

$$g(x) = \begin{cases} 1 & , x = nT_0; \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

where n is any integer. Figure 3.4 shows $g(x)$ as an approximation to the LP residual of the vowel /ep_long/, with the impulses located at the instances of the energy bursts in the LP residual. Remember that the LP residual also contains a noise component, whereas the impulse train shown does not, which accounts largely for the differences between the two signals between pulse locations. Because the FFT of an impulse train with period T_0 is simply a magnitude-scaled impulse train with period $\frac{1}{T_0}Hz$, we find that the spectrum of an impulse train approximates the pulse-like harmonics of the spectrum of a voiced LP residual (see figure 3.2(a)).

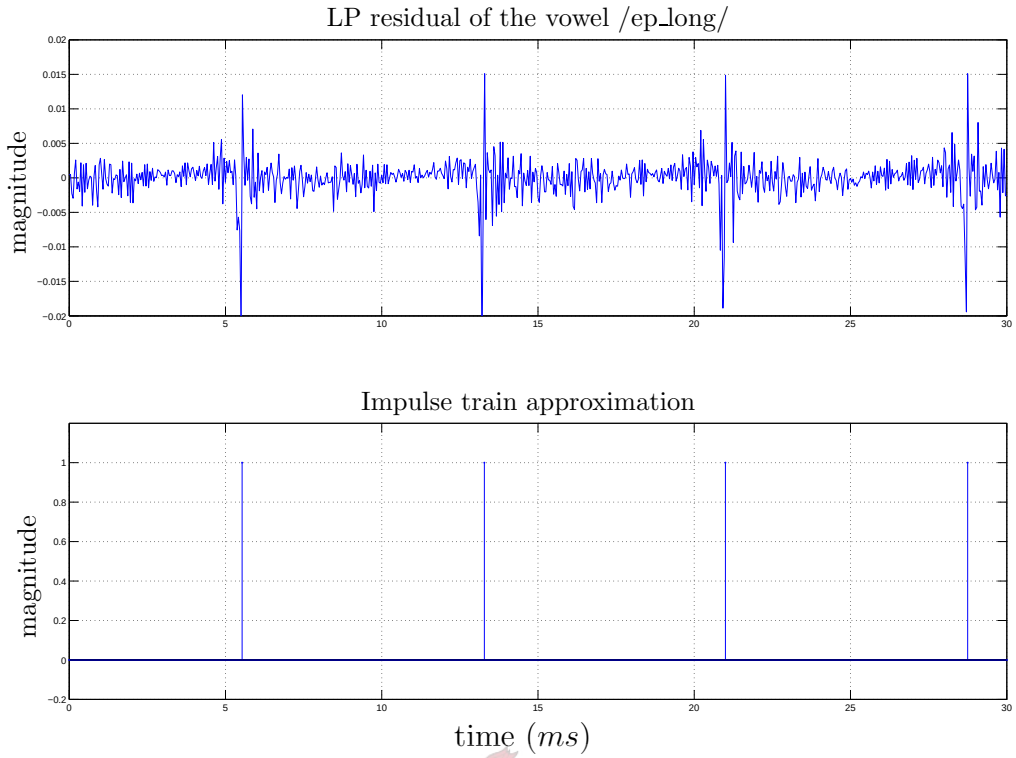


Figure 3.4: *Impulse train approximation of a voiced LP residual.*

The impulse train is one of the most widely used approximations for voiced speech source signals because of its simplicity. In fact, the model requires only two parameters, F_0 and the voiced magnitude (henceforth denoted by A_v). It also requires almost no computation, since the majority of the values are zero and the nonzero values are specified by A_v. The main disadvantage of the impulse train is evident when considering its spectrum. Note that the impulse-like harmonics in figure 3.2(a) seem to decay to almost zero over 0–5kHz. The harmonics of an impulse train do however not decay, and hence this model does not account for the frequency envelope that is typical of voiced speech.

3.2.3 Voiced speech residuals represented by polynomials

Many researchers have proposed the use of polynomials as parametric excitation functions for speech synthesis. This is usually done by finding an optimal polynomial approximation for a single pitch period [6], [14]. Polynomial modelling of the excitation is widely used because of its flexibility and the small number of parameters required to specify such a function. Finding the optimal polynomial estimate for a given signal is also a well-defined problem. Most models are applied to the integral of the LP residual, which is preferred because polynomials are more suitable for modelling slowly varying waveforms than rapidly varying waveforms, and integration is a form of lowpass filtering (smoothing). As an illustration of

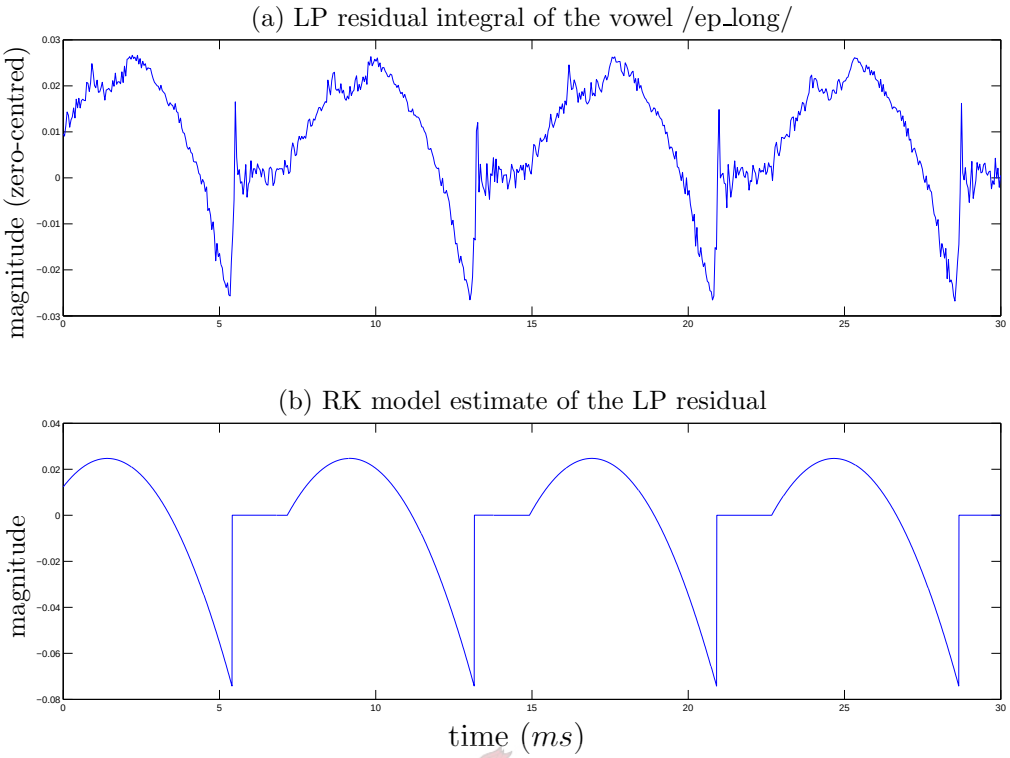


Figure 3.5: *The Rosenberg-Klatt residual integral model.*

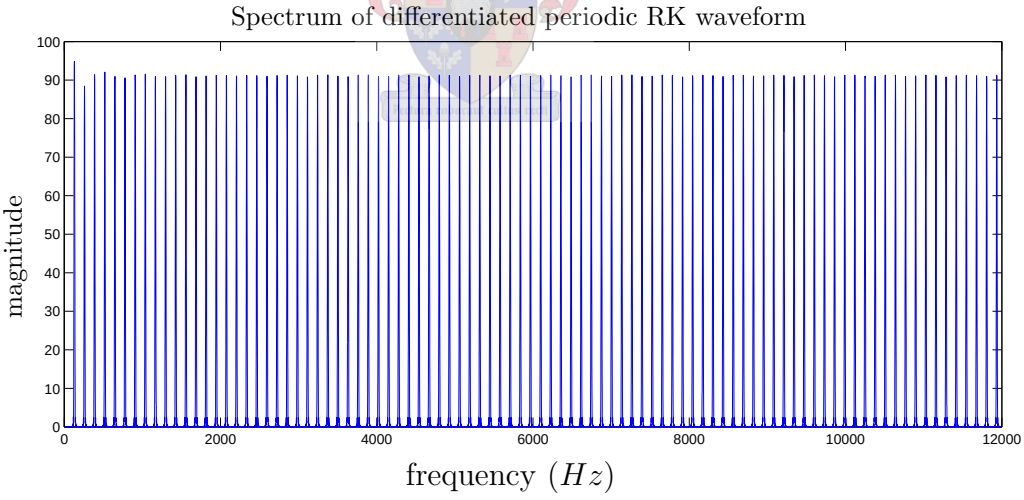


Figure 3.6: *Spectrum of the differentiated Rosenberg-Klatt residual integral model.*

this, figure 3.5(a) shows the integral of the LP residual of the vowel /ep_long/. Compare this to its time waveform as seen in figure 3.1(a), which shows each pitch period varying more rapidly over time.

A particular polynomial model for a pitch period of the residual integral that has been

widely adopted in speech synthesis applications is the *Rosenberg-Klatt* (RK) model [21], which is defined as follows:

$$g(x) = \begin{cases} 2ax - 3bx^2 & , 0 \leq x < OQT_0; \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

where

$$a = \frac{27A_v}{4OQ^2T_0}, \text{ and } b = \frac{27A_v}{4OQ^3T_0^2}$$

and OQ is the fraction of T_0 referred to as the open quotient, which is a measure of the duration for which the vocal folds are apart during each pitch period. Figure 3.5(b) shows an RK approximation of the LP residual integral of the vowel /ep_long/ as shown in figure 3.5(a). Note the similarity between the waveform shapes. Note also that there are differences between the magnitude ranges of the two signals. This may be because the RK model definition is not exactly suited to the glottal excitation of the particular speaker of this sound. However, it is possible to adjust the polynomial parameters in order to match the specific speaker. Figure 3.6 shows the spectrum of a differentiated sequence of consecutive identical RK waveforms. Note that the harmonics of the RK spectrum exhibit a fairly constant and non-decaying/diminishing amplitude over the entire frequency range, as was also the case for the impulse train.

3.2.4 Voiced speech residuals as a sum of sinusoids

One particular speech production model which does not fall into the category of source-filter models is called the harmonics plus noise model (HNM) [42]. The driving concept behind this model is the use of a set of sinusoids with variable amplitudes, frequencies and phases to model the harmonic behaviour of voiced speech up to a certain maximum voicing frequency F_{max} and additive Gaussian noise to model the unvoiced part [2]. Doing so allows the spectral envelope of the voiced speech to be modelled implicitly in the amplitudes of the sinusoids. Within sinusoidal models, amplitude modulation (AM) and frequency modulation (FM) techniques are sometimes applied for coding and/or synthesis [34]. Estimation of the amplitudes, frequencies and phases from recorded speech is, however, not simple and invariably involves some form of peak-picking.

Instead of using an HNM as a complete speech production model, it can be applied to the modelling of only the excitation signal. Knowing that the spectral representation of a sinusoid in the positive frequency domain is an impulse, we can use this model to improve on the shortcomings of the impulse train considered in section 3.2.2, since it allows us to alter the harmonics' frequency envelope through changing the amplitudes of the sinusoids. It also allows us the freedom of specifying F_{max} by restricting the number of harmonics.

Let us consider again the example of the vowel /ep_long/. Note (figure 3.2(a), repeated in figure 3.8) that harmonic peaks are present in the vowel spectrum up to a frequency of

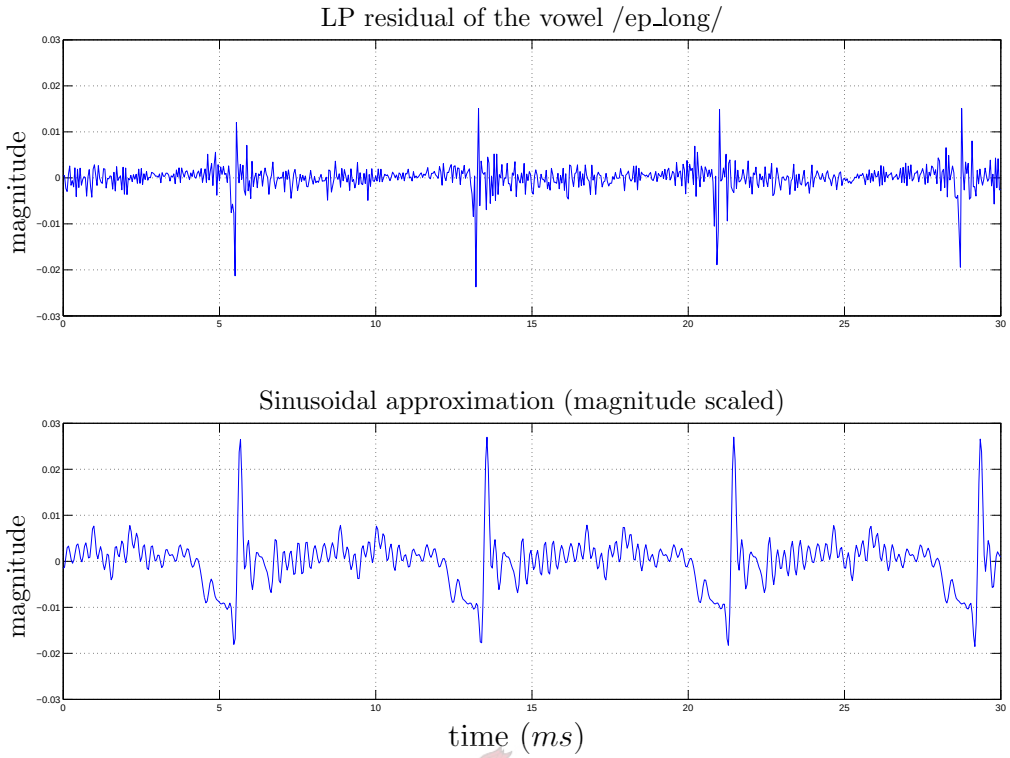


Figure 3.7: *Sinusoidal approximation of a voiced LP residual.*

roughly $4.7kHz$. By using this value as F_{max} and by manually locating the harmonic frequencies from the spectrum, we can determine the harmonic amplitudes and phases directly from the complex-valued $X(f_\omega)$ using equation 2.2. If we sum together a set of sinusoids with these amplitudes, phases and frequencies over the appropriate time range, we find the approximation shown in figure 3.7, with its FFT spectrum shown in figure 3.8. Note the almost identical frequency envelope of the harmonics. Also note that this model's time-domain waveform appears to be a better approximation to the measured excitation signal than an impulse train (figure 3.4). The differences between the actual and estimated pitch pulses in figure 3.7 are due to a slight error in the estimation of F_0.

It is impractical to use the sinusoid amplitudes, phases and frequencies as parameters, firstly because of the large number of these parameters and secondly because F_0 and F_{max} are not constant over all sounds of an utterance, and therefore the number of sinusoids and their locations must change over time. A more suitable way of modelling these parameters would be by using parametric functions which are dependent on F_0 and F_{max}. These functions can then be used to determine each sinusoid's parameters at any particular point in time.

The speech synthesis system developed in this thesis adopts the sinusoidal voicing source with additive Gaussian white noise as its primary excitation model for monophones because of this model's ability to include information pertaining to the harmonic frequency envelope.

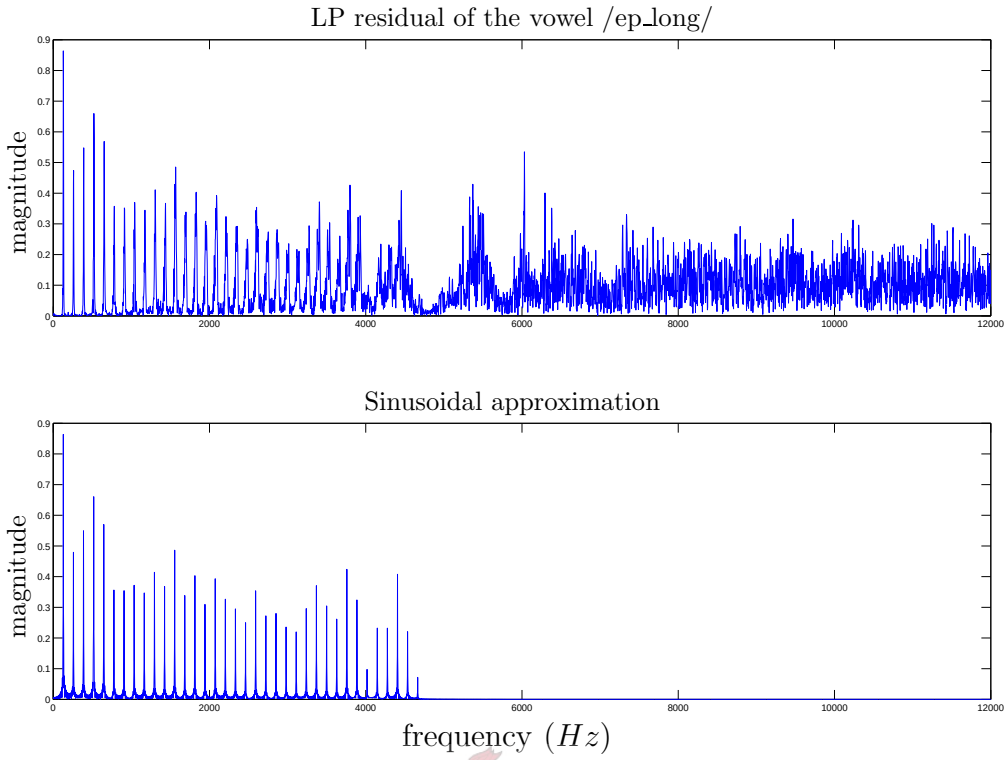


Figure 3.8: *Spectrum of sinusoidal approximation of a voiced LP residual.*

The following section presents three algorithms for calculating the magnitude parameters A_n and A_v by estimating a voiced/unvoiced (V/U) ratio, as well as an algorithm for estimating F_{max} from any particular monophone LP residual signal. Note that the algorithms pertaining to the estimation of A_n and A_v can be applied to all the source signal models presented in this chapter.

3.3 Parameter Estimation

Because we have assumed that the harmonic and stochastic components of an LP residual signal are uncorrelated, we require knowledge of each particular sound's voiced and unvoiced content for synthesis. Many systems attempt to classify a sound as being either voiced or unvoiced, and use this classification to generate the excitation signal as either harmonic or as noise, scaled by a global magnitude factor to account for signal energy. Although this classification is effective for purely voiced sounds (such as vowels) and purely unvoiced sounds (such as unvoiced fricatives), it fails to account for sound classes such as voiced fricatives, which contain both harmonic and stochastic components. It also does not allow the model to incorporate a measure of "breathiness" for voiced sounds, a factor which varies between different speakers.

In order to achieve maximal flexibility in terms of the harmonic and stochastic content, a model requires that A_n and A_v be specified separately. This allows the model to synthesise each sound by generating the harmonic and stochastic parts separately before combining them. However, separating the voiced and unvoiced components from within a single observation of a signal is not easy. Although there are algorithms (such as MUSIC [20]) which can separate two uncorrelated signals, these often perform computationally expensive subspace analyses. As we are not as concerned about the individual signals as in their respective contributions to the combined signal's energy, a simpler solution to the problem of finding A_n and A_v is to somehow estimate a V/U ratio for each sound, which can then be used to combine the signals by using the mean signal energy to calculate A_n and A_v. Many researchers make use of residual signal autocorrelation sequences in order to measure its periodicity for finding the V/U ratio. Other methods have also been proposed, such as spectral tilt measures [30] and correlation-based estimates for different frequency bands [49].

The following sections present several measures which are aimed at estimating the V/U ratio of a residual signal by measuring its Gaussianity. This approach can be taken because the noise component in a speech LP residual signal has a Gaussian distribution (section 3.2.1). The first of these is a novel expectation-based estimate. The latter two, namely kurtosis and entropy, are well-defined in literature [17], [38]. Among the literature considered, this particular approach to V/U ratio estimation for residual signals was not found.

3.3.1 Gaussianity by expectation

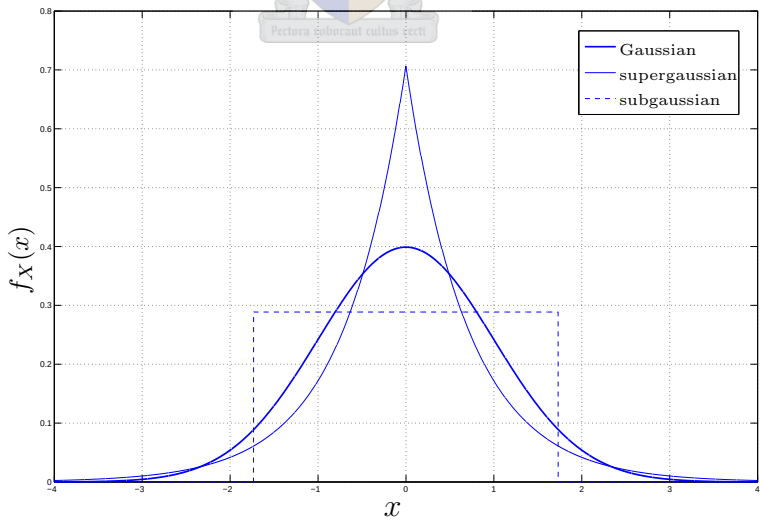


Figure 3.9: *Examples of Gaussian, supergaussian and subgaussian PDF's.*

Consider a Gaussian random variable x with mean $\mu_X = 0$ and variance $\sigma_X^2 = 1$. From

Sound class	Phoneme	m_{f_ε}	$\sqrt[4]{\text{kurt}(\tilde{\varepsilon})}$	$H_D(\hat{\varepsilon})$
Vowel	/i_long/	0.307	1.342	0.106
	/ep_long/	0.322	2.008	0.240
	/a_long/	0.316	1.757	0.182
Nasal	/m/	0.292	1.083	0.029
	/n/	0.298	1.252	0.056
	/nj/	0.301	1.341	0.067
Approximant	/rt/	0.296	1.200	0.045
	/l/	0.294	1.155	0.036
	/w/	0.287	0.892	0.016
Voiced Fricative	/v/	0.292	1.073	0.028
	/z/	0.296	1.140	0.042
	/zh/	0.302	1.255	0.074
Unvoiced Fricative	/f/	0.284	0.580	0.010
	/s/	0.283	0.573	0.008
	/sh/	0.283	0.615	0.009
Voiced Plosive	/g/	0.314	1.606	0.177
	/b/	0.317	1.775	0.215
	/d/	0.313	1.715	0.177
Unvoiced Plosive	/k/	0.314	1.524	0.239
	/p/	0.335	1.840	0.544
	/t/	0.317	1.591	0.274

Table 3.1: *Gaussianity measures for various sound classes.*

equation 3.1, the PDF of x then has the form:

$$f_X(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} \tag{3.4}$$

We know (from [32]) that the expected value of a function g of a random variable x with PDF $f_X(x)$ is given by:

$$\mathcal{E}[g(x)] = \int_{-\infty}^{\infty} g(x) f_X(x) dx \tag{3.5}$$

By substituting $f_X(x)$ from 3.4 as $g(x)$ in 3.5, we now calculate the expected value of the

PDF of a zero mean, unity variance Gaussian random variable as:

$$\begin{aligned}
\mathcal{E}[f_X(x)] &= \int_{-\infty}^{\infty} f_X(x) f_X(x) dx \\
&= \int_{-\infty}^{\infty} (\frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2})^2 dx \\
&= \frac{1}{4\sqrt{\pi}} (\frac{2}{\sqrt{\pi}} \int_{-\infty}^{0} e^{-x^2} dx + \frac{2}{\sqrt{\pi}} \int_{0}^{\infty} e^{-x^2} dx) \\
&= \frac{1}{4\sqrt{\pi}} (- \lim_{a \to -\infty} \mathrm{erf}(a) + \lim_{a \to \infty} \mathrm{erf}(a)) \\
&= \frac{1}{2\sqrt{\pi}}
\end{aligned} \tag{3.6}$$

where $\mathrm{erf}(a) = \frac{2}{\sqrt{\pi}} \int_0^a e^{-t^2} dt$ is known as the error function [51], with $\lim_{a \to \infty} \mathrm{erf}(a) = \lim_{a \to -\infty} \mathrm{erf}(a) = 1$. We can now quantify the Gaussianity of a residual signal $\varepsilon(n)$ by estimating the expectation:

$$m_{f_\varepsilon} = \frac{1}{N} \sum_{n=0}^{N-1} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\hat{\varepsilon}(n)^2} \tag{3.7}$$

where $\hat{\varepsilon}(n) = \frac{\varepsilon(n) - \mu_\varepsilon}{\sigma_\varepsilon}$ is the normalised residual for comparison with the result of equation 3.6. Because unvoiced residual signals are approximately Gaussian, we expect m_{f_ε} to be close to $\frac{1}{2\sqrt{\pi}}$ for these sounds, whereas residual signals that contain some voicing should yield higher values because their true distribution is supergaussian (more peaked than a Gaussian, see figure 3.9).

Table 3.1 lists some values of m_{f_ε} for some examples of the different sound classes. As expected, vowels have the highest values of m_{f_ε}. Values for voiced sounds containing more unvoiced content (nasals, approximants and voiced fricatives) are lower, with unvoiced fricatives producing the lowest values of m_{f_ε}, which are very close to the theoretical value of $\frac{1}{2\sqrt{\pi}}$ (≈ 0.2821). Table 3.1 confirms that m_{f_ε} is a candidate for V/U ratio estimation for these sounds. Plosives sounds, however, do not appear to be distinguishable as voiced or unvoiced using m_{f_ε}. This is due to the non-stationary behaviour of these sounds, as they are produced by a short burst of energy. This energy envelope apparently causes a plosive sound residual to be more supergaussian, even when the sound itself is completely unvoiced. Values obtained using entropy- and kurtosis-based Gaussianity measures are not shown in Table 3.1, but it should be noted that these yielded very similar results over all sound classes considered.

The measure m_{f_ε}, as calculated by the above procedure, was not found in other literature. Rather, it was derived from the concept of measuring a signal's Gaussianity by direct comparison of its distribution to that of theoretical Gaussian data. It was preferred to kurtosis and entropy because of its greater computational efficiency.

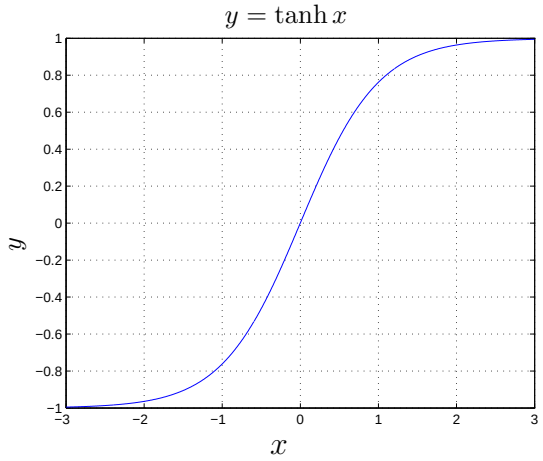


Figure 3.10: *Hyperbolic tangent function for suppressing outliers.*

3.3.2 Gaussianity by kurtosis

Kurtosis is defined as a normalised form of the fourth central moment of a random variable [17]:

$$\begin{aligned} \text{kurt}(x) &= \mathcal{E}\{x^4\} - 3(\mathcal{E}\{x^2\})^2 \\ &= \mathcal{E}\{x^4\} - 3 \qquad (\text{when } \mu_X = 0 \text{ and } \sigma_X^2 = 1) \end{aligned} \tag{3.8}$$

where $\text{kurt}(x)$ denotes the kurtosis of x. It is used as a measure of Gaussianity because it is exactly zero when x has a Gaussian distribution, positive for a supergaussian distribution and negative for a subgaussian distribution (see figure 3.9). One property of kurtosis which restricts its accuracy, however, is the fact that equation 3.8 is very sensitive to outliers in the data. Because of this, some implementations transform the variable x by a nonlinear function which is approximately linear close to the mean value while suppressing or limiting extreme values, such as the hyperbolic tangent function [17] (see figure 3.10). We therefore define $\tilde{\varepsilon}(n) = \tanh \hat{\varepsilon}(n) = \tanh \frac{\varepsilon(n) - \mu_\varepsilon}{\sigma_\varepsilon}$. In order to use kurtosis as a measure of Gaussianity for the purpose of finding the linear scaling constants A_n and A_v, it is necessary to use $\sqrt[4]{\text{kurt}(\tilde{\varepsilon})}$ rather than kurtosis itself, as kurtosis is proportional to the fourth central moment of the samples and not their magnitudes.

Table 3.1 lists some values of $\sqrt[4]{\text{kurt}(\tilde{\varepsilon})}$ for some examples of the different sound classes. As expected, we find that the largest values are found for the vowel residuals, which are the most supergaussian of the residual signals. We also find that unvoiced fricatives have the lowest values, with all other sound classes ranging between vowels and unvoiced fricatives, except the plosive sounds. These are, as in the case of m_{f_ε}, problematic because their values are similar to those associated with vowels, even in the case of unvoiced plosives. Note that the values of $\sqrt[4]{\text{kurt}(\tilde{\varepsilon})}$ for the unvoiced fricatives are not very close to zero as we would expect, but are still noticeably smaller than for the other classes. This may be due to the fact that the kurtosis estimate is small, yet not exactly zero for these practical signals, and

therefore the root operation enlarges the values considerably. Overall, however, the values in table 3.1 indicate that $\sqrt[4]{\text{kurt}(\tilde{\varepsilon})}$ is another suitable measure of Gaussianity for V/U ratio estimation, but suffers from similar inaccuracies for plosive sound residuals as m_{f_ε}.

3.3.3 Gaussianity by entropy

Entropy is measure of the uncertainty of the outcome of a set of events and is formally defined in [38]. It is often used in telecommunications systems as a measure of the amount or rate of information for a given channel. Entropy is defined as:

$$H = -\sum_{n=1}^{N} p_n \log p_n \tag{3.9}$$

where $p_1 \dots p_N$ is the set of event probabilities. Over a finite range, H is a maximum when all the p_n are equal ($= \frac{1}{N}$), which is intuitively the most uncertain case, and $H = 0$ only if all the p_n are zero save one (which is 1), which is the most predictable case. In the continuous case, entropy is defined as:

$$H(x) = -\int_{-\infty}^{\infty} p(x) \log p(x) \mathrm{d}x \tag{3.10}$$

where H denotes the entropy and $p(x)$ is the PDF of x. Over the range $(-\infty, \infty)$, $H(x)$ is a maximum when x has a Gaussian distribution. For this reason entropy is often used as a measure of Gaussianity [17]. In the case of $\mu_X = 0$ (taken from [38]):

$$\begin{aligned}
p(x) &= \frac{1}{\sqrt{2\pi\sigma_X^2}} e^{-\frac{x^2}{2\sigma_X^2}} \\
-\log p(x) &= \log\sqrt{2\pi\sigma_X^2} + \frac{x^2}{2\sigma_X^2} \\
H(x) &= -\int p(x) \log p(x) \mathrm{d}x \\
&= \int p(x) \log\sqrt{2\pi\sigma_X^2} \mathrm{d}x + \int p(x) \frac{x^2}{2\sigma_X^2} (d)x \\
&= \log\sqrt{2\pi\sigma_X^2} + \frac{\sigma_X^2}{2\sigma_X^2} \\
&= \log\sqrt{2\pi}\sigma_X + \log\sqrt{e} \\
&= \log\sqrt{2\pi e}\sigma_X
\end{aligned} \tag{3.11}$$

Since a signal's Gaussianity can be measured by comparing its entropy to the result of 3.11, some systems use differential entropy (also called negentropy) instead of entropy as a measure, which is defined as:

$$H_D(x) = H(x_G) - H(x) \geq 0 \tag{3.12}$$

where x_G is a Gaussian random variable with the same variance as x. In the case where $\sigma_X^2 = 1$, $H(x_G) = \log\sqrt{2\pi e}$, which is a constant. Because $p(x)$ is difficult to determine for

sampled data with an unknown distribution, entropy is often approximated in practice. One approximation of negentropy is given by:

$$H_D(x) \approx \frac{1}{12}\mathcal{E}\{x^3\}^2 + \frac{1}{48}\text{kurt}(x)^2 \tag{3.13}$$

where $\text{kurt}(x)$ is the kurtosis of x (see equation 3.8). However, as with kurtosis, equation 3.13 suffers from sensitivity to outliers, besides being computationally expensive. A more direct approach to estimating entropy is to note that the integral of equation 3.10 is in the form of an expectation (according to equation 3.5):

$$H(x) = -\int_{-\infty}^{\infty} p(x)\log p(x)\mathrm{d}x = -\mathcal{E}\{\log p(x)\} \tag{3.14}$$

so that entropy can therefore be estimated by finding the (negative) mean value of $\log p(x)$:

$$H(x) \approx -\frac{1}{N}\sum_{n=0}^{N-1}\log p(x(n)) \tag{3.15}$$

where all that remains is finding estimates for the $p(x(n))$. Although the form of equation 3.15 is very similar to that of m_{f_ε} (equation 3.6), we cannot at this point assume that x is Gaussian, since equation 3.15 would then simplify to $\log\sqrt{2\pi e}\sigma_X$. Another possible method for estimating the $p(x(n))$ is to use a histogram-based approach, whereby each $p(x(n))$ is determined by the number of samples in the proximity of $x(n)$ as a fraction of the total number of samples. We therefore define:

$$H_D(\hat{\varepsilon}) = \log\sqrt{2\pi e} + \frac{1}{N}\sum_{n=0}^{N-1}\log p(\hat{\varepsilon}(n)) \tag{3.16}$$

as an estimate for negentropy, where the $p(\hat{\varepsilon}(n))$ are estimated using a histogram of $\hat{\varepsilon}$.

Table 3.1 lists the values of $H_D(\hat{\varepsilon})$ as calculated for some examples of the various sound classes. It confirms that unvoiced fricatives have the most Gaussian residuals, whereas vowels have the least Gaussian residuals (largest $H_D(\hat{\varepsilon})$). All other sounds have entropy values between these two classes, as expected. As with the other Gaussianity measures considered in this chapter, the values of $H_D(\hat{\varepsilon})$ for plosive sounds do not reflect the V/U ratio of their respective residual signals. This is, as mentioned earlier, due to the non-stationary nature of these sounds. Furthermore, it is possible that the histogram probability estimates of the relatively small plosive residual windows are less accurate than in the case of the longer windows associated with the other sound classes. For all the sound classes besides plosives, table 3.1 confirms that entropy, in the form of the negentropy estimate $H_D(\hat{\varepsilon})$ given by equation 3.16, is a useful measure of Gaussianity for the purpose of V/U ratio estimation.

3.3.4 Maximum voicing frequency estimation

Table 3.1 shows that the Gaussianity measures considered in the previous sections are all capable of quantifying the V/U ratio of a residual signal. However, closer inspection of

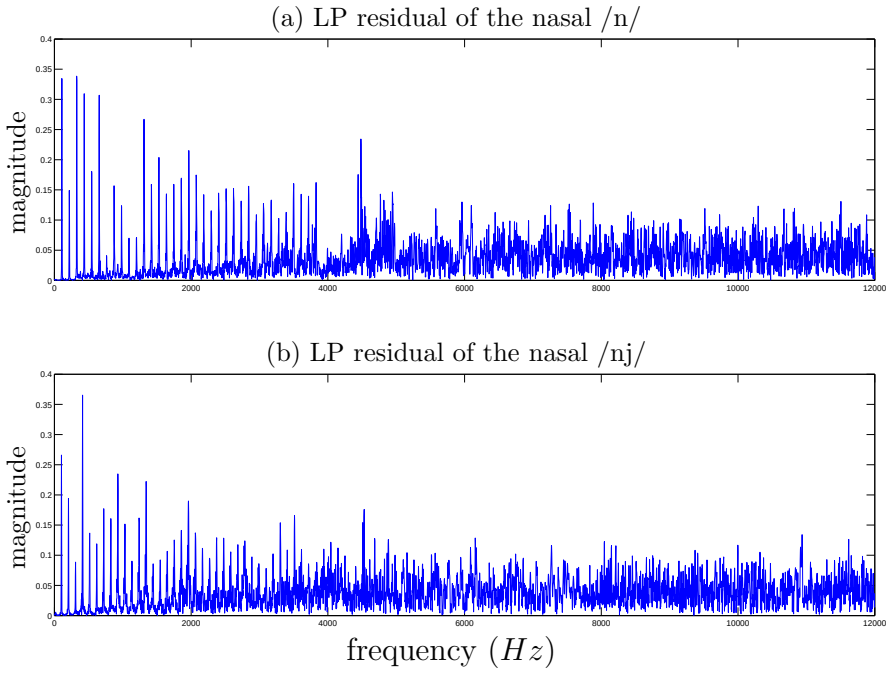


Figure 3.11: *LP residual spectra of two nasal sounds.*

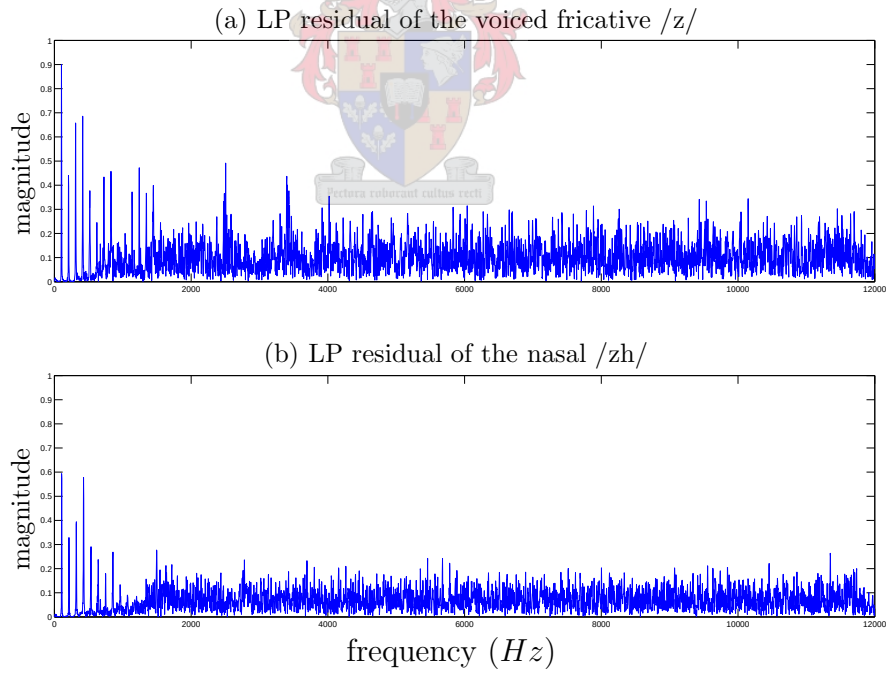


Figure 3.12: *LP residual spectra of two voiced fricative sounds.*

the values reveals that these measures alone may not be sufficient to model the different sounds effectively. For example, note that the Gaussianity measures yield values within similar ranges for nasals, approximants and voiced fricatives. However, frication is not deliberately introduced when producing a nasal sound, whereas approximants may contain some measure of frication due to the slight constriction of air flow, and voiced fricatives do contain deliberate frication. These characteristics of the three sound classes seem to indicate that voiced fricatives should appear the most Gaussian, with approximants second and nasals the most non-Gaussian. However, early experiments yielded very similar synthetic residual signals for these sound classes due to the similarities in their V/U ratios, which is undesirable if we consider that the LP residuals of these sounds have different spectral distributions, as will be shown shortly.

Figures 3.11 and 3.12 provide the answer as to why the overall Gaussianity measures do not reflect the differences in the LP residuals between nasals and voiced fricatives. Note that the harmonics in the case of the nasal residual signals are discernible up to between $4kHz$ and $5kHz$. In contrast, for the voiced fricative residuals the harmonics appear to decay more rapidly with frequency and are discernible only up to between $1.5kHz$ and $2.5kHz$. Also, noting the scales of the vertical axes, we find that the voiced fricative residual harmonics are larger in magnitude than those of the nasal sounds. These two factors combine to cause the global measures of Gaussianity to be similar for these sounds, since they do not take into account the harmonic frequency envelope. The problem becomes apparent when using a global Gaussianity measure to deduce the V/U ratios with which to generate mixed excitation signals during synthesis. During informal listening tests, this was found to result in unnatural nasals and approximants due to a large stochastic component. For this reason, some source signal models make use of multiple V/U ratio estimates for different frequency bands to encode the harmonic frequency envelope, which seems to yield more natural sounding synthetic speech [49]. Others have proposed the use of correlation-based measures between successive pitch periods in order to estimate the aperiodicity of the signal, which is a major contributing factor to the harmonic frequency envelope and the maximum observable voiced frequency (F_{max}) [31].

In order to avoid the need for a large number of parameters to encode the harmonic frequency envelope of a residual signal, some systems make use of a parameter called the maximum voicing frequency (F_{max}), which determines the frequency cutoff beyond which very little or no harmonic content is generated. Of the models considered in section 3.2, only the sinusoidal model allows direct manipulation of the harmonic frequency envelope and F_{max} by allowing us to specify the number of harmonics and their respective magnitudes. In order to estimate F_{max} for a given LP residual, we need some measure of how the residual's Gaussianity changes over frequency, so that we can define some threshold beyond which we can assume the harmonic content to be zero, thereby finding F_{max}. Within this section, we present a strategy for estimating F_{max} and illustrate that this parameter can be used to refine the sinusoidal model such that a more accurate representation of the voiced

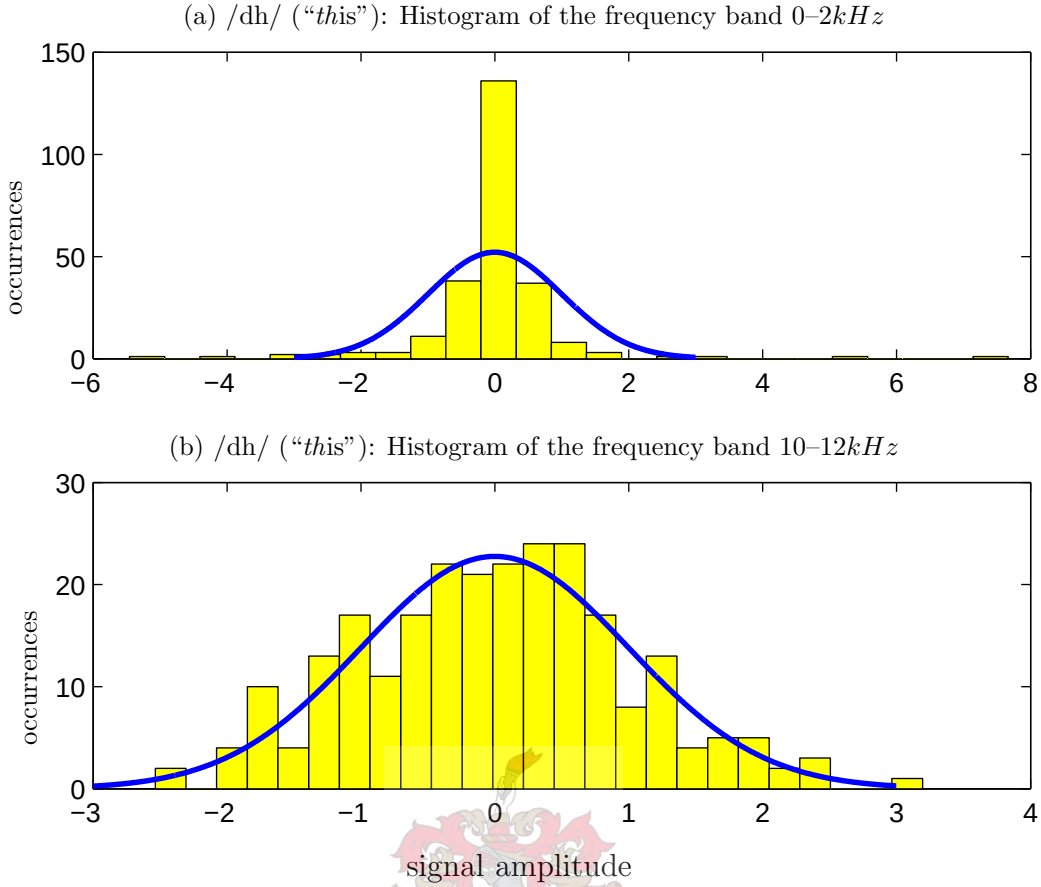


Figure 3.13: *Histograms of residual spectrum voiced and unvoiced frequency bands.*

and unvoiced components of an LP residual is obtained based on their respective spectral distributions. Note that using F_{max} replaces the need for V/U ratio estimation of an LP residual as described in the previous sections, yet we still require a measure of Gaussianity for the estimation of F_{max}. The Gaussianity measures of sections 3.3.1–3.3.3, which have thus far been applied to time signals, are applied to FFT spectra in this section. It has been observed that, even though the FFT operation may alter the distribution of the signals, we still find that the FFT samples in the lower (voiced) frequencies appear more supergaussian than the higher (unvoiced) frequencies. The histograms shown in figure 3.13 support this claim by showing empirical distributions for the voiced fricative /dh/. We therefore proceed by assuming that the Gaussianity measures will yield values closer to that of ideal Gaussian noise at high frequencies than at low frequencies if voicing is present.

Although all three Gaussianity measures were implemented and found to be suitable candidates for the estimation of F_{max} (see Appendix B), m_{f_ε} was preferred for this project because it is computationally more efficient than kurt($\tilde{\varepsilon}$) and $H_D(\hat{\varepsilon})$. Figure 3.14 shows the variation of m_{f_ε} as a function of frequency for the LP residuals of the nasal sound /nj/ (as in thi*ng*) and the voiced fricative /zh/ (as in *g*enre). It is useful to compare these with their

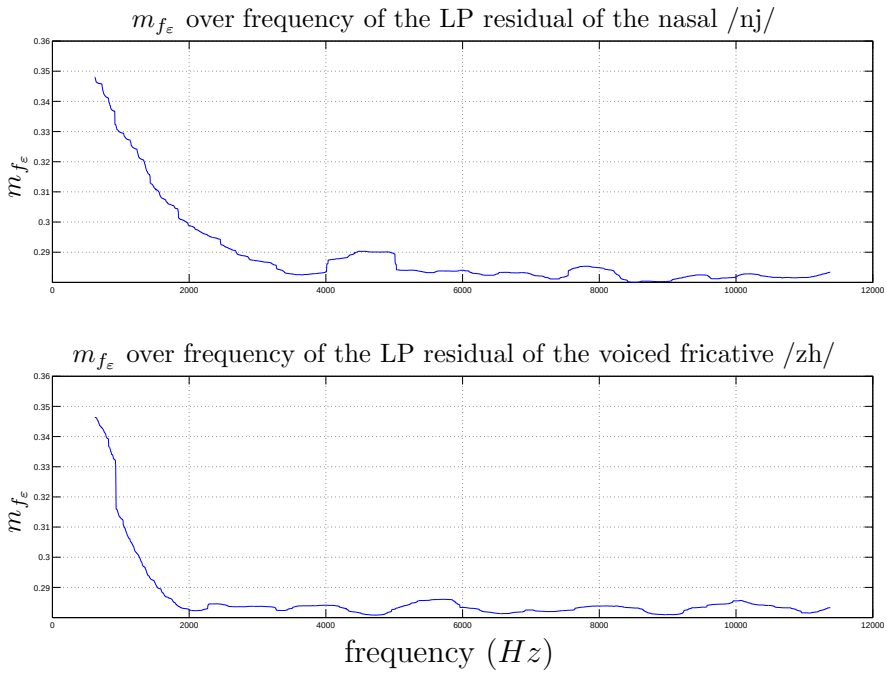


Figure 3.14: *(non-)Gaussianity over frequency for two different sounds.*

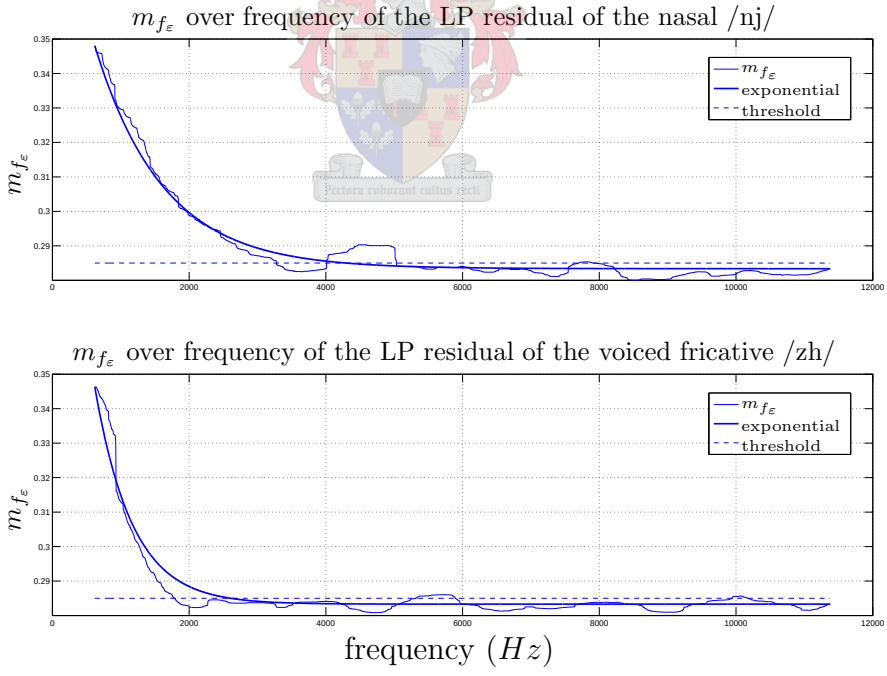


Figure 3.15: *Exponential curve fitting to residual Gaussianity for finding F_{max}.*

spectra shown in figures 3.11(b) and 3.12(b). Note that m_{f_ε} cannot be calculated directly from the magnitude spectrum. Instead, we separate the real and imaginary parts of the FFT and find the graphs in figure 3.14 by averageing the values of m_{f_ε} calculated for these two signal components. Each m_{f_ε} value was computed over a $1kHz$ window of the FFT centred at the corresponding frequency. A median filter was subsequently applied to the sequence of m_{f_ε} values. The median filter removes outliers in the m_{f_ε} curve that occur due to the stochastic nature of the spectral estimates. The initial and final segments of the m_{f_ε} graphs are not shown, since these fall outside of the $1kHz$ window.

The estimation of F_{max} from the graphs shown can be approached in a variety of ways. One possibility is to observe (for example, in figure 3.14) that the measure m_{f_ε} for an LP residual signal of a sound that contains voicing exhibits an approximately exponential decay over frequency. Note that the graphs tend to the theoretical value of $\frac{1}{2\sqrt{\pi}}$ (≈ 0.2821) for Gaussian data associated with the parameter m_{f_ε}. Knowing this, we can fit a monotonically decreasing exponential curve to the graph to ensure that each point on the graph corresponds to a single frequency. The fitting procedure, which was developed specifically for this project, is not a trivial one and is presented in Appendix B. Experimentally, it was found that a value of 0.285 is a suitable threshold for finding F_{max} when using m_{f_ε} as a measure of Gaussianity. The exponential approximations are shown in figure 3.15. Setting the threshold as indicated results in $F_{max} \approx 4.3KHz$ for /nj/ and $F_{max} \approx 2.6KHz$ for /zh/, which corresponds approximately to the frequencies at which the harmonics are no longer discernible in figures 3.11(b) and 3.12(b). When doing so for a variety of sounds using all three Gaussianity measures noted in this chapter, we find that this is a suitable approach to estimating F_{max} (see Appendix B).

Figure 3.16 shows some synthetic excitation signal spectra obtained using the sinusoidal/noise model for various sounds. The parameter F_{max} was estimated using the procedure outlined in Appendix B for each (voiced) sound. Comparing these to the LP residual signal spectra in figure 3.2, we find that the harmonic frequency envelopes of the synthetic phones /ep_long/ and /z/ are quite similar to those of their LP residual spectra. However, the stochastic components of the synthetic excitations of these two sounds differ substantially from those present in their LP residuals, which are larger in magnitude at frequencies above the harmonic band. This is due to the earlier assumption that the unvoiced energy is spectrally flat, and therefore the total unvoiced energy is spread evenly across all frequencies in the synthetic residuals. The synthetic excitation signal of the unvoiced fricative /f/ is very similar to its LP residual signal.

When using voicing models which have a flat spectral envelope such as the impulse or RK models, one can model the frequency envelope either by lowpass filtering or by adding a stochastic component to the length of each pitch period [31]. This stochastic component is referred to as aperiodicity or "jitter", and is often used in speech synthesis systems because the voiced component of natural speech is not exactly periodic [22], [29].

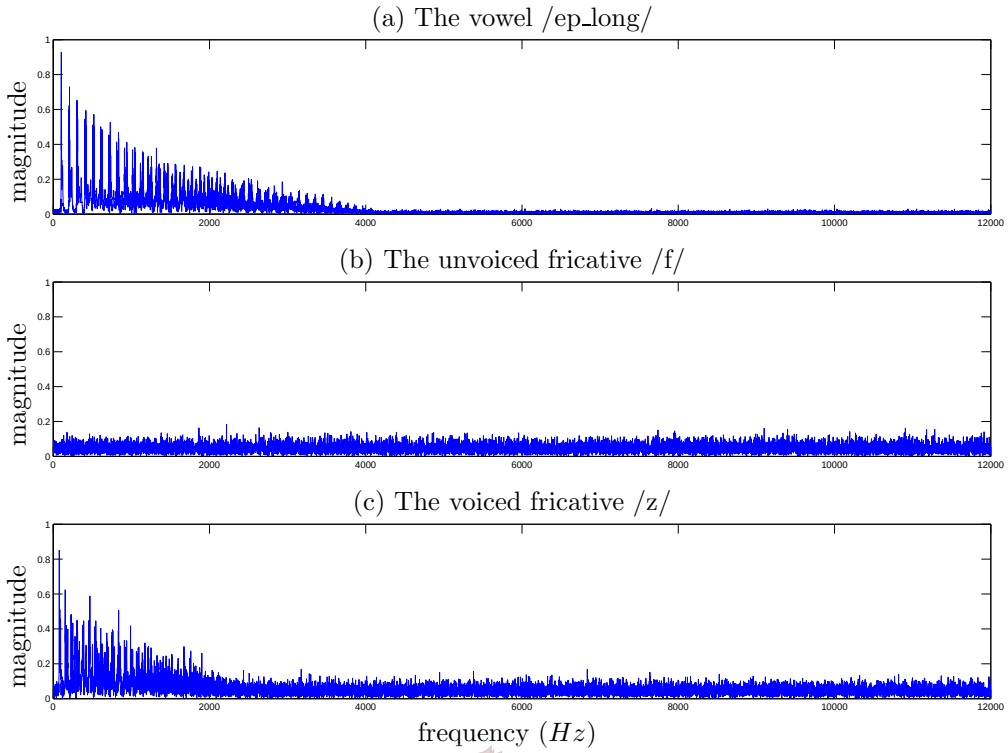


Figure 3.16: *Synthetic excitation spectra of different speech sounds (sinusoids/noise model).*

3.4 Modelling Plosives

In section 2.2, we noted that special attention needs to be given to plosive (stop) sounds, since these are not as stationary in nature as other sounds. Typically, these sounds are produced by a short burst of energy that occurs when pressure that has built up behind a constriction formed at the lips, tongue, palate or glottis is suddenly released. Figure 3.17 illustrates the time-domain signals of unvoiced plosives sounds, and figure 3.18 of voiced sounds. Note that the duration of these signals is about 800 samples ($33ms$ at $24kHz$) for the unvoiced and between 3600 and 4800 samples ($150ms$ and $200ms$) for the voiced plosives.

A Rayleigh PDF was chosen to model the pulse-like magnitude envelope of unvoiced plosive sounds. This function is defined in [32] as:

$$f_X(x) = \begin{cases} \frac{2}{b}(x-a)e^{-(x-a)^2/b} & , x \geq a; \\ 0 & , x < a. \end{cases} \tag{3.17}$$

where a and b are suitable constants. Figure 3.19 shows the magnitude of the excitation signal of the unvoiced plosive sound /t/ after smoothing by lowpass filtering ($200Hz$ pass band). Also shown is a scaled Rayleigh PDF, which illustrates that this function does model the general trend of the signal magnitude envelope. A magnitude-scaled version of equation 3.17 is therefore applied to the magnitude parameters A_n and A_v during the

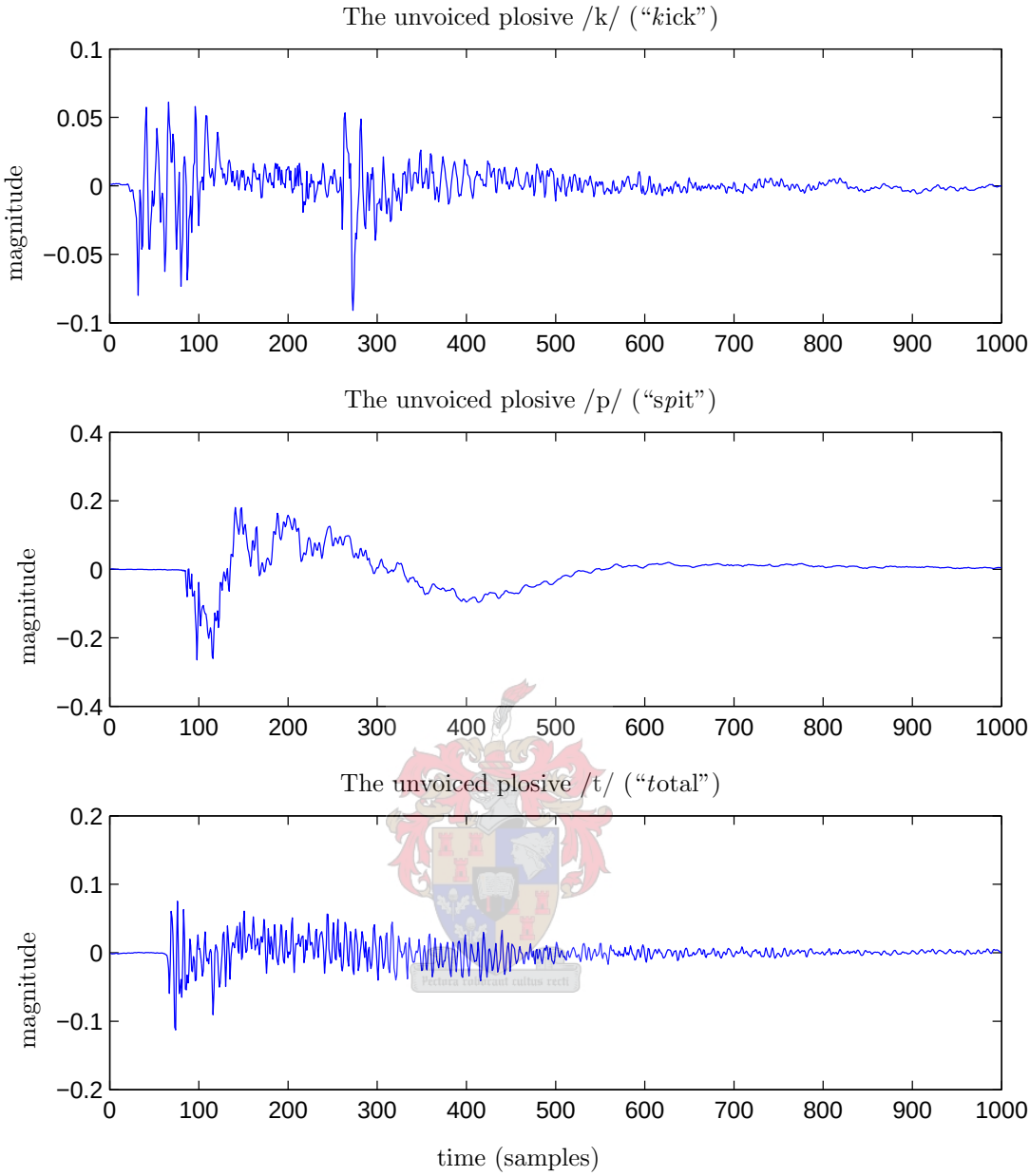


Figure 3.17: *Examples of unvoiced plosive sounds.*

constant (filter) section of plosive sounds to model their time-variant behaviour, with the surrounding parameter transitions adjusted to ensure smooth trajectories. Although this was found to be a suitable approach for unvoiced plosive sounds, voiced plosives remain problematic for the following reasons (see figures 3.17 and 3.18):

- The magnitude envelope of voiced plosives appears to have a different shape to that of unvoiced plosives, and

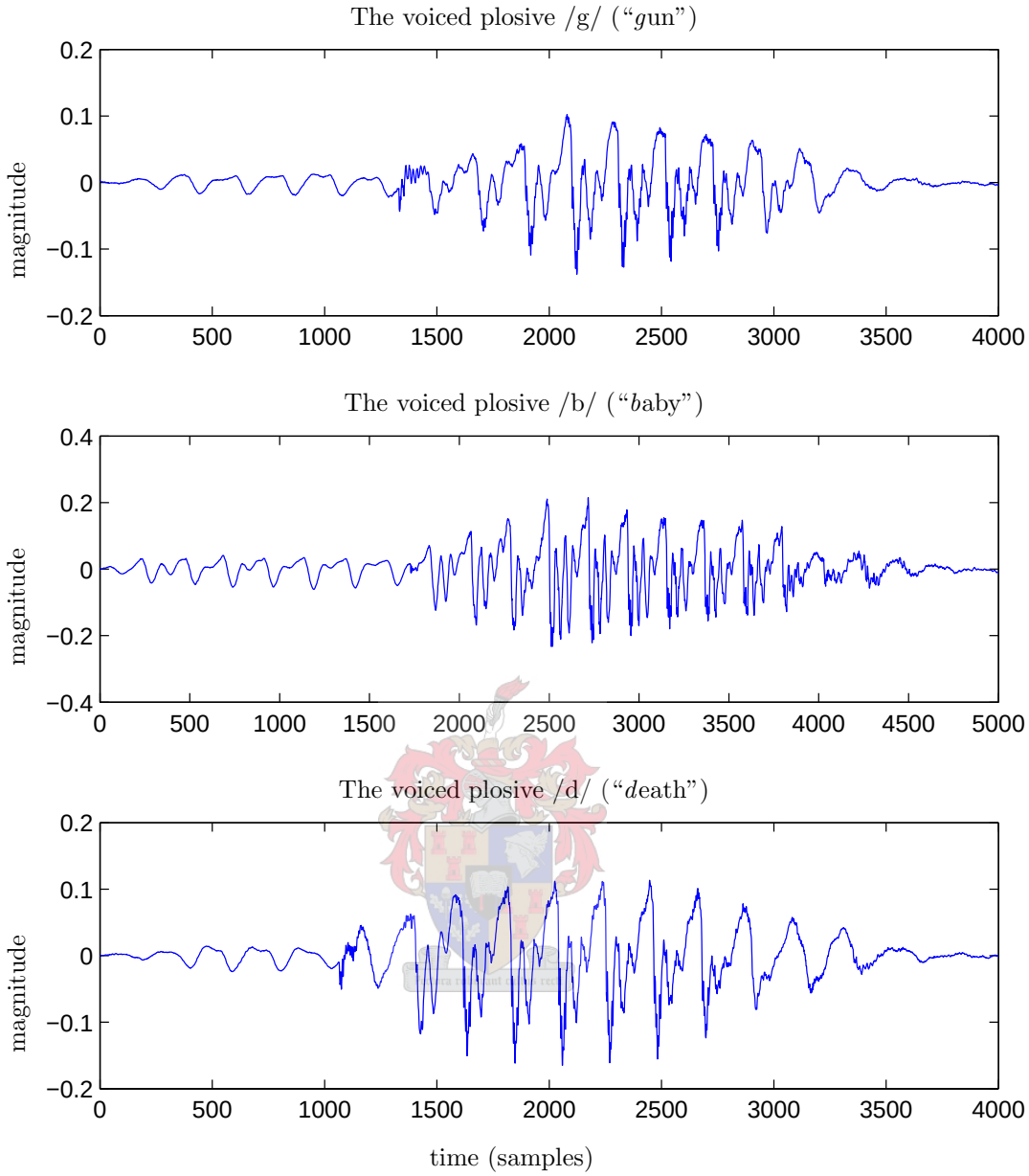


Figure 3.18: *Examples of voiced plosive sounds.*

- a degree of voicing is present before the onset of the energy burst.

Because of these differences, the Rayleigh magnitude envelope was only applied to the unvoiced source parameter A_n. As the final evaluations in chapter 6 will show, the modelling of voiced plosives will require further work in order to improve synthesis quality.

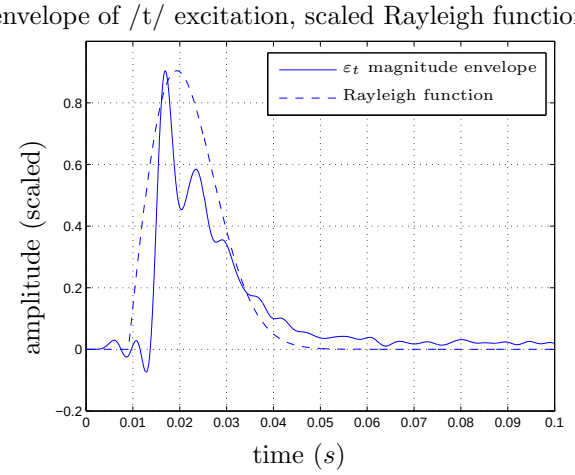


Figure 3.19: *The Rayleigh PDF as a magnitude envelope for plosive sounds.*

3.5 Prosodic Contours

Prosody is the term used to refer to the combined effect of three distinct properties of speech signals which convey a large amount of speaker-dependent information. These properties are the pitch, magnitude and duration of speech sounds. Any speech synthesis system which aims to produce natural-sounding speech requires some method of incorporating prosodic contours into its utterances. As stated in chapter 1, integrating prosodic information into synthetic utterances within a concatenative synthesis system is often problematic, as it requires either a very large corpus or modifications to the prerecorded speech segments, a process which often introduces audible artifacts. Within the proposed parametric model, however, the prosodic parameters can be included in the source signal generation process, so that no modifications to an existing waveform are necessary. This is an extremely important advantage and is key to the overall goal of developing a flexible and portable speech generation system.

Although prosodic parameters are often modelled as linear (individual) contours, the idea of superpositional prosodic contours has become popular within the last decade [11], [44]. A superpositional model assumes that prosodic parameters are influenced at various levels in a spoken utterance, for example sub-word level, word level, sentence level and even conversational level. It must be emphasised at this point that the work described in this thesis does not cover prosodic contour generation, but rather aims to present a flexible model suitable for the integration of prosodic contours, should they be available. The following sections present a short discussion and suggested representation of each prosodic parameter.

The speech generation system for this project uses a two-level architecture for prosodic information (its implementation is described in chapter 5). The lower (local) level pertains to individual phonetic segments added to the utterance, which can also be entire words or parts of a word, depending on how it is added to the utterance. The upper (global) level pertains to an utterance consisting of multiple phonetic segments, where the utterance can

be a phrase or other part of a sentence, an entire sentence or even a larger section. If desired, either of the levels can be ignored (defaulted to a constant) for each prosodic parameter, making the system suitable for linear as well as superpositional prosodic contours. Since both levels are always present, the local level can be ignored for desired phonetic segments only, if needed.

3.5.1 Pitch

The parameter F_0 can be used to model pitch within a source signal model. Since the models considered in this chapter assume that the voiced and unvoiced components of a residual excitation model can be generated independently, this parameter affects only the voiced part, which contains the harmonic content that causes us to hear the pitch/tone of a voiced sound. For example, in the case of the impulse train voicing model (section 3.2.2), F_0 would (inversely) determine the spacing between successive impulses. In the case of the sinusoidal model, F_0 is the frequency of the fundamental harmonic (sinusoid), and the frequency of the k^th harmonic is given by kF_0. Note that a parametric model allows us to specify the pitch arbitrarily, so that in principle female pitch contours can be used even if the phone recordings for which the models are estimated were made by a male. This, among other aspects, makes the parametric synthesis model suitable for polyglot synthesis and possibly even voice conversion.

Parametric source signal models generally have the property that pitch information can be included in the generation process for each sample where it is available. In the case where the pitch contour is specified for each synthesis sample, the model can take it into account as each sample is generated. Depending on the nature of the pitch contour itself, this has the advantage that pitch variations are quite fluent, whereas time-domain pitch modification algorithms usually perform scaling on a per pitch period basis, which may introduce audible artifacts if the scaling factor is large or the variations are rapid. For the system described in chapter 5, pitch contours can be included at both the local and the global level.

3.5.2 Magnitude

Magnitude is an indication of the stress that can be placed on particular sounds, which varies with speaking style, emotion, etc. From a signal processing point of view, it is concerned with the energy of a sound. Most systems use a single magnitude parameter to vary the stress patterns of a synthetic utterance, and although this approach is sufficient, it may not be flexible enough to model certain effects such as a temporary variations in the amount of frication (breathiness) within an utterance or independent stress of the voiced or unvoiced component of a sound. The system described in chapter 5 therefore contains magnitude contours for both voiced and unvoiced content, both implemented at the local and global level, as discussed earlier. These contours are treated separately from the V/U ratio-dependent parameters A_v and A_n discussed in the previous sections of this chapter, which are concerned

with the modelling of the phone residual signals (their smoothed transitions are discussed in section 4.2).

Another advantage of the independent control of the voiced and unvoiced magnitude parameters is that the speaking style can be altered so that, for example, the phone recordings of one speaker my be used to synthesise speech with different breathiness quality so as to sound more like another speaker. Again, this is advantageous should the model be used for voice conversion or polyglot synthesis.

3.5.3 Duration

Duration control is an important aspect of a flexible speech synthesis system because speaking rate differs substantially between speakers, and because speakers often vary their speaking rate according to their emotional state or for emphasis. Although it is possible to use a smooth curve to represent the speaking rate of an utterance, its values would have to be translated to constant scale factors for the number of samples which are associated with each discrete segment of the utterance, such as the phones, silences, and the transition regions between phones. It is therefore more appropriate to simply specify the segment durations or to scale the default duration values in order to vary the speaking rate of a synthetic utterance. For this reason, the system described in chapter 5 allows the user to specify duration for each element of a phonetic segment (local level) without the use of smooth contours. Duration control is also not implemented at the global level for the same reason.

3.6 Chapter Summary

After presenting some initial examples and discussions concerning LP residual signals in section 3.1, we discussed various models that can be used for the excitation signal within a source-filter framework (section 3.2). These include Gaussian noise for the unvoiced component as well as impulses, RK-waveforms and a sinusoidal model for the voiced component. A mixed excitation model incorporating a sinusoidal voiced component and a Gaussian noise unvoiced component was developed and preferred because of its flexibility.

Next, we presented three measures for determining the voiced/unvoiced ratio of a residual signal by using the Gaussianity measures entropy, kurtosis and a novel expectation-based estimate (section 3.3). A novel method for determining the maximum voicing frequency of voiced sounds was also developed in section 3.3.4, paying particular attention to the difficulty experienced when using the Gaussianity measures globally for voiced fricatives, nasals and approximants.

Finally, we defined a magnitude envelope model for the stochastic energy burst present in unvoiced plosive sounds in section 3.4, and presented a discussion on speech prosody and how it can be modelled using the excitation signal model parameters A_n, A_v and F_{max}

(section 3.5).

Chapter 4

Parameter Interpolation

Chapter 2 introduced a number of parametric representations of the vocal tract (filter) component of the speech signal, whereas chapter 3 explored various excitation signal models. With both these components of a complete source-filter model of speech production specified, what remains is to generate the time-varying values of the source and excitation model parameters in order to mimic the fluent nature of human speech. As motivated in section 1.5, monophones were chosen as the basic synthesis unit for the purpose of minimising the system's dependency on the database. Using monophones also allows the system to be employed in cases where very little recorded information is available for a particular language, possibly as little as a single recording for each phone. The database therefore contains only representations of individual phones in isolation and does not reflect the dynamic behaviour of the speech signal that occurs at each transition between adjacent phones in fluent speech. This chapter is concerned with the development of a method by which we can generate appropriate transitions between phones in the synthetic speech signal by calculating a set of transition parameter vectors which can be inserted between adjacent phones.

Section 1.4.2 briefly discussed two strategies that can be employed for the purpose of generating these transitional parameters. The first, referred to as parameter *generation*, aims to train dynamic models such as ANN's or HMM's on speech data that is representative of the transitions. Note that this database can be separate from the original database which is used to define the monophone parameter vectors, so that the dynamic models can be applied to different phone sets, and possibly even to different languages. The second approach, parameter *interpolation*, is aimed at using a particular algorithm for calculating the transient vectors directly from the phone vectors. Parameter interpolation may be more flexible than parameter generation because its underlying algorithm can be defined parametrically to allow certain aspects of the transitions to be customised, whereas parameter generation schemes depend mainly on the information available in the data used for training. For the cases where transient behaviour is speaker- or language-dependent, this flexibility becomes even more important. A hybrid strategy is presented in [15], which mixes data-derived and interpolated parameters in an attempt to draw from the strengths of both approaches.

Parameter *interpolation* was chosen as a strategy because it requires no additional database

for training and because of its flexibility. Both of these factors are very important for a language independent speech synthesis system, especially when language resources are scarce (see sections 1.2 and 1.5). Remember, from previous chapters, that each phone parameter vector consists of a set of 30 LSF's at $24kHz$, as well as the source signal parameters A_n, A_v and possibly F_{max}. Section 4.1 introduces a suitable parameter interpolation algorithm for calculating the transient filter coefficients by interpolating the LSF's, and section 4.2 presents a method for interpolating the source parameters. The reasons for interpolating source and filter parameters separately will be explained in section 4.2.

4.1 LSF Interpolation

As discussed in chapter 2, the filter parameters represent the effect of the vocal tract shape on the speech sound. This implies that the changes in vocal tract shape are encoded in the way the filter parameters vary over time. Viewed differently, this means that any particular filter parameter interpolation scheme relates to a specific vocal tract movement, which may or may not be similar to the natural movements of a human vocal tract. Any good interpolation scheme should aim to emulate natural vocal tract movements as closely as possible. Literature describing a current mainstream TTS system which interpolates between monophones could not be found. In most cases where interpolation methods are used, however, they are employed as strategies to overcome the discontinuities which are introduced by the concatenation of recorded units, which are usually diphones or larger units (see section 1.4.1) [5], [13], [33], [37] and [42].

Section 2.3.6 explored a number of representations of the LP parameters, and it was concluded that the LSF representation was the most suitable candidate for interpolation (refer to figure 2.13). In the simplest case, the LSF's can be kept constant for the duration of each phone, which is equivalent to phone concatenation and intuitively does not yield very good results due to the discontinuities at the phone boundaries. A simple alternative which is often used to smooth out spectral discontinuities in concatenative systems is linear interpolation. Although linear interpolation has been shown to reduce spectral distortion significantly relative to direct concatenation, figure 2.13(c) seems to suggest that a more fluent (nonlinear) parameter transition is required in order to accurately mimic the vocal tract behaviour in the inter-phone transition regions. The following sections present two possible approximation methods.

4.1.1 Bézier segments

The use of Bézier segments for modelling LSF trajectories within diphones (which corresponds to the inter-phone transitions we wish to model) is proposed in [45]. A Bézier

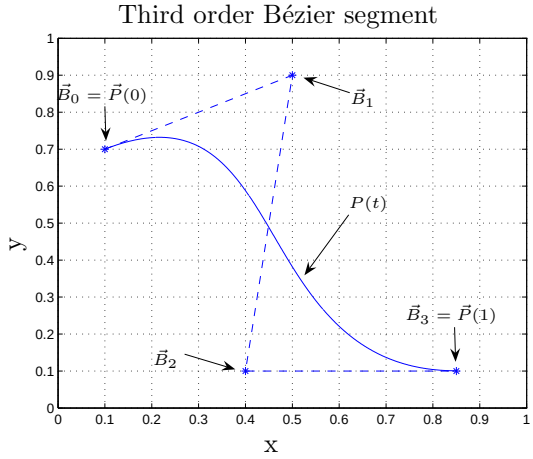


Figure 4.1: *An example of a Bézier segment.*

segment $\vec{P}(t)$ of order n is defined as:

$$\vec{P}(t) = \sum_{i=0}^{n} \vec{B}_i J_{n,i}(t) \qquad 0 \leq t \leq 1, \tag{4.1}$$

where

$$\vec{B}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad \text{and} \qquad J_{n,i}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

where t is an independent variable defining a curve $\vec{P}(t)$ in (x, y). Each $\vec{B}_i$ denotes the i^{th} vertex (x_i, y_i) of a defining polygon. Figure 4.1 shows an example of $\vec{P}(t)$ with $\vec{B}_0 \ldots \vec{B}_3$ indicated by the asterisks. Note that $\vec{P}(t)$ passes through $\vec{B}_0$ and $\vec{B}_3$, whereas $\vec{B}_1$ and $\vec{B}_2$ specify the initial and final tangents of the curve, respectively. In our case, choosing $n = 3$ implies that $\vec{B}_0$ corresponds to a particular LSF belonging to the preceding phone and $\vec{B}_3$ to the same LSF of the successor phone, while the LSF transition is determined by $\vec{B}_1$ and $\vec{B}_2$. When generating a sequence of phones, we require that a corresponding series of target values be specified. Considering the k^{th} phone in a sequence, this requires that $\vec{B}_3(k-1) = \vec{B}_0(k)$ and the parameters $\vec{B}_2(k-1)$, $\vec{B}_3(k-1)$ and $\vec{B}_1(k)$ form a straight line to ensure a smooth transition.

As is mentioned in [45], one of the great advantages of using Bézier curves is that a set of optimal[1] $\vec{B}_i$ can be calculated for each LSF parameter trajectory, which means that the curves can be trained over a given data set. However, because the system being developed here is aimed at minimising data dependency, these advantages cannot fully be utilised. As an alternative, we would require some formula according to which a $\vec{B}_1$ and $\vec{B}_2$ can be

[1] "Optimal" in this sense means that the difference between the estimated Bézier segment and the original parameter trajectory can be minimised with respect to some distortion measure.

calculated for each inter-phone transition, which may lead to a complex set of rules that, for he k^{th} phone in a sequence, depend on the surrounding phones $k-1$ and $k+1$. To avoid the necessity of defining such rules manually, section 4.1.2 explores an interpolation scheme which mimics the inter-phone transitions in a more implicit manner.

4.1.2 B-spline curves

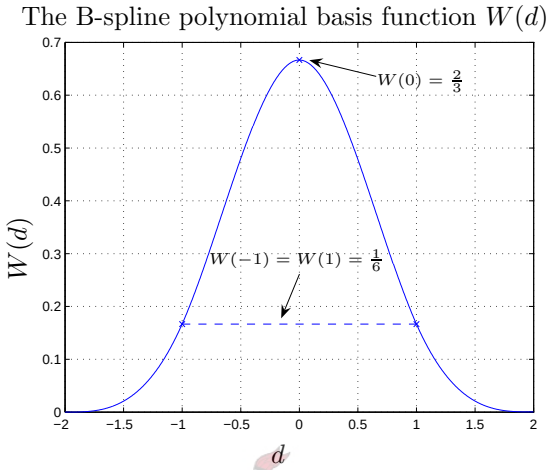


Figure 4.2: *The "Gaussian" B-spline basis function.*

A B-spline curve is an interpolation method which is calculated for K target points M_k such that $k = 0 \ldots K-1$. The driving concept behind the interpolation is that each interpolated point $p(t)$ for $0 \leq t \leq K-1$ is calculated as a weighted contribution of its surrounding target points. The nearest target point has the greatest effect on $p(t)$ to ensure that it follows the general trend of the sequence $M_0 \ldots M_{K-1}$. For the purpose of LSF interpolation between monophones, each LSF's trajectory can be modelled by a single B-spline curve such that the k^{th} value of that LSF which corresponds to the k^{th} phone in a particular sequence is equal to M_k. This implies that the smooth curve $p(t)$ represents that LSF's transitions at each point where $t \neq k$. To ensure that $p(t)$ forms a smooth curve, the target point weights must be specified as a continuous function $W(d)$ (called the basis function) of the distance $d_k(t) = k - t$ so that W is a maximum when $d = 0$ and $W(d)$ decreases as $|d|$ increases. In practice, a polynomial approximation of the Gaussian function is used:

$$W(d) = \begin{cases} \frac{1}{6}(3|d|^3 - 6d^2 + 4) & , 0 \leq |d| < 1; \\ \frac{1}{6}(2 - |d|)^3 & , 1 \leq |d| < 2; \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

Note that the graph spans the distances $-2 < d < 2$, which limits the number of contributing target points to 2 or 3. Figure 4.2 illustrates $W(d)$ as defined by equation 4.2.

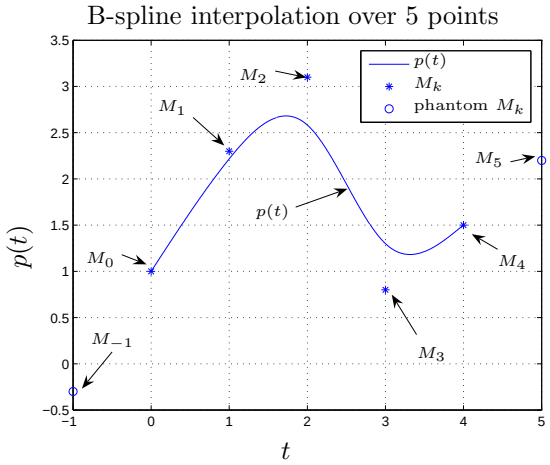


Figure 4.3: *B-spline interpolation with phantom points.*

One problem that arises when we simply use the above method to interpolate between the M_k occurs when t is close to 0 or $K-1$. Because there are no target points at -1 (before the first phone) and K (beyond the last phone), $p(t)$ will tend towards zero when approaching these points. To avoid this, we need to define two (phantom) target points M_{-1} and M_K such that the contributions at the point $t=0$ sum to the value of M_1 and the contributions at $t=K-1$ sum to M_{K-1}:

$$\begin{aligned} W(-1)M_{-1} + W(0)M_0 + W(1)M_1 &= M_0 \\ \frac{1}{6}M_{-1} + \frac{2}{3}M_0 + \frac{1}{6}M_1 &= M_0 \\ \Rightarrow M_{-1} &= 2M_0 - M_1 \qquad (4.3) \\ \text{and (similarly) } M_K &= 2M_{K-1} - M_{K-2} \qquad (4.4) \end{aligned}$$

The above results indicate that the phantom target point M_{-1} is opposite M_1 with respect to M_0 such that it cancels the effect of M_1 at $t=0$, with M_K doing the same with regards to M_{K-2} at $t=K-1$.

Figure 4.3 illustrates $p(t)$ for a set of $K=5$ target points, with phantom points inserted at $t=-1$ and $t=5$ according to equations 4.3 and 4.4, respectively. Remember that $t=0$ corresponds to the first and $t=K-1$ to the last LSF value in the sequence and all $p(t)$ for $t<0$ will be discarded so that the final $p(t)$ still spans $0 \leq t \leq K-1$.

In figure 4.3, note that $p(t)$ does not pass through each target point because of the weighted effect of its surrounding points. Although this may be desirable as a means to model co-articulation when applied to LSF's, we may wish to require that each phone be articulated precisely, which means that we need a mechanism to ensure that $p(t)$ passes through the M_k. This can be done by transforming the target values such that they cause the value of $p(t)$ to be equal to the original target values at their specific locations $t=0 \ldots K-1$.

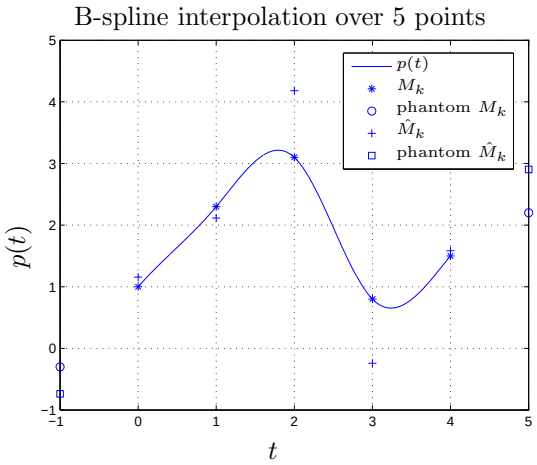


Figure 4.4: *B-spline interpolation with transformed target points.*

To do this, we need to solve the set of linear equations given by:

$$M_k = W(-1)\hat{M}_{k-1} + W(0)\hat{M}_k + W(1)\hat{M}_{k+1} \tag{4.5}$$

where $\hat{M}_{-1} \dots \hat{M}_K$ denote the transformed target points. Note that there are $M+2$ transformed points and only M original target points. In order to solve for the $\hat{M}_k$, we require two more equations. One possibility is to use the phantom points M_{-1} and M_K as given by equations 4.3 and 4.4. By then writing the equations of 4.5 in matrix form, we can calculate the $\hat{M}_k$ directly:

$$\mathbf{Q} = \begin{bmatrix} W(0) & W(1) & 0 & \cdots & 0 & 0 & 0 \\ W(-1) & W(0) & W(1) & \cdots & 0 & 0 & 0 \\ 0 & W(-1) & W(0) & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & W(0) & W(1) & 0 \\ 0 & 0 & 0 & \cdots & W(-1) & W(0) & W(1) \\ 0 & 0 & 0 & \cdots & 0 & W(-1) & W(0) \end{bmatrix},$$

$$\hat{\mathbf{M}} = \begin{bmatrix} \hat{M}_{-1} \\ \vdots \\ \hat{M}_K \end{bmatrix} \quad \text{and} \quad \mathbf{M} = \begin{bmatrix} 2M_0 - M_1 \\ M_0 \\ \vdots \\ M_{K-1} \\ 2M_{K-1} - M_{K-2} \end{bmatrix}$$

so that

$$\begin{aligned} \mathbf{Q}\hat{\mathbf{M}} &= \mathbf{M} \\ \Rightarrow \hat{\mathbf{M}} &= \mathbf{Q}^{-1}\mathbf{M} \end{aligned} \tag{4.6}$$

where $\mathbf{Q}^{-1}$ denotes the inverse of the matrix $\mathbf{Q}$.

Applying equation 4.6 yields $p(t)$ as shown in figure 4.4, to be compared to $p(t)$ for the same target point set shown in figure 4.3.

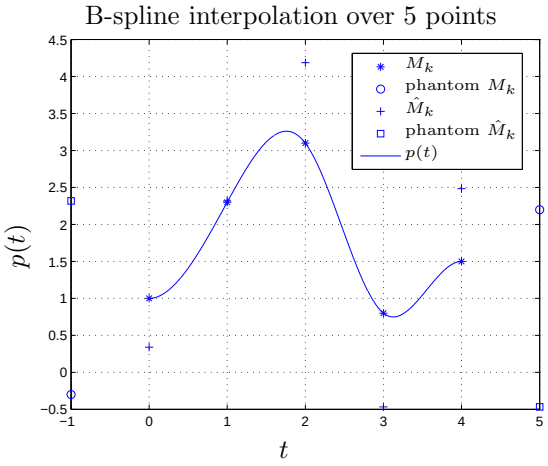


Figure 4.5: *B-spline interpolation with zero initial and final gradients.*

Note that $p(t)$ in figures 4.3 and 4.4 has non-zero slopes at $t = 0$ and $t = K - 1$. This may be problematic if we desire an utterance to start or end with a steady sound, which is often the case. We may change this by specifying the gradients g_0 and g_{K-1} of $p(t)$ at $t = 0$ and $t = K - 1$, respectively, by noting that these gradients (with respect to t) are given by:

$$g_0 = \frac{M_1 - M_{-1}}{2} \tag{4.7}$$

$$g_{K-1} = \frac{M_K - M_{K-2}}{2} \tag{4.8}$$

Now, instead of using the phantom target points M_{-1} and M_K to define the first and last entries of the vector $\mathbf{M}$ in equation 4.6, we set them equal to the desired gradients (0 in this case) so that $\mathbf{M}$ and the first and last rows of the matrix $\mathbf{Q}$ become:

$$\mathbf{M}^T = \begin{bmatrix} 0 & M_0 & \dots & M_{K-1} & 0 \end{bmatrix},$$

$$\mathbf{Q} \text{ (row 1)} = \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} & 0 & \dots & 0 \end{bmatrix}$$

and

$$\mathbf{Q} \text{ (row } K+2) = \begin{bmatrix} 0 & \dots & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$$

where $\mathbf{M}^T$ denotes the transpose of $\mathbf{M}$.

Solving equation 4.6 with these constraints results in $\hat{M}_0 = \hat{M}_2$ and $\hat{M}_K = \hat{M}_{K-2}$, which in turn sets $p(0) = M_0$ and $p(K-1) = M_{K-1}$ (from equations 4.5 and 4.6), which means that the resulting $p(t)$ still passes through all the target points M_k. The result is shown in figure 4.5, where the same target points of figures 4.3 and 4.4 are used for reference.

4.1.3 Duration control using B-splines

The B-splines described in section 4.1.2 possess a number of shortcomings which must be addressed before they are suitable for monophone interpolation. The first of these is that the target points M_k have to be equally spaced at intervals of 1 unit in t, meaning that we are unable to specify sounds with different lengths within an interpolated utterance. Furthermore, we cannot control for how long $p(t)$ remains constant at each target point.

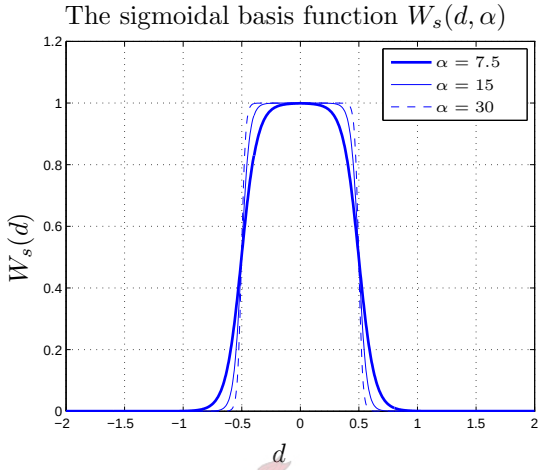


Figure 4.6: *A sigmoidal basis function for B-splines.*

In order to take unequal spacing of the target points into account, the t-axis can be scaled to fit the target points by specifying the discrete increments of t as needed. However, such scaling causes changes in the gradient of $p(t)$, which may result in audible artifacts wherever the increment changes significantly in a region where the gradient of $p(t)$ is not zero. To avoid such discontinuities in the gradient of $p(t)$, one strategy is to ensure that the gradient of $p(t)$ is close to zero at each target point. Figure 4.5, however, shows that this is not necessarily the case. This is mostly due to the fact that the basis function $W(d)$ of equation 4.2 does not remain close to its maximum value in the region around $d = 0$. A new basis function which would give higher preference to the closest target value at a point t was therefore developed:

$$W_s(d, \alpha) = \begin{cases} \frac{c}{1+e^{\alpha(2|d|-1)}} & , 0 \leq |d| < 2; \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

where

$$c = \left(\int_{-1}^{3} \frac{1}{1+e^{\alpha x}} \mathrm{d}x \right)^{-1} = \left(x - \frac{1}{\alpha} \log(1+e^{\alpha x}) \Big|_{-1}^{3} \right)^{-1}$$

to ensure that $\int_{-2}^{2} W_s(d, \alpha) \mathrm{d}d = 1$.

Note that the constant scale factor c is not required when transforming the target points according to equation 4.6. The function $W_s(d, \alpha)$ is called sigmoidal due to the fact that it

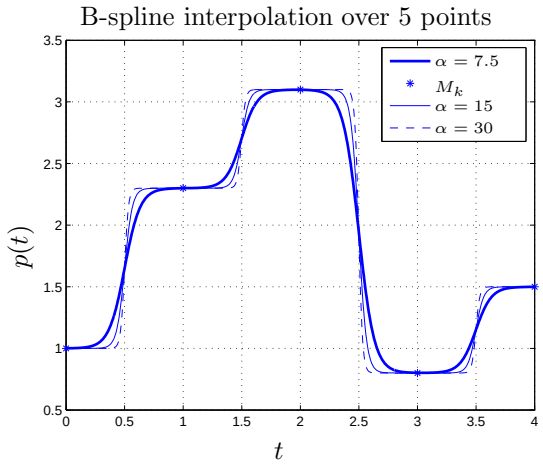


Figure 4.7: *B-spline interpolation using a sigmoidal basis function.*

is defined by two functions of the form $\frac{1}{1+e^{\alpha x}}$ (called the sigmoid function) over the range $-1 \leq x < 3$ placed symmetrically about $d = 0$ and compressing the axis to fit the range $-2 < d < 2$. The function is shown in figure 4.6 for different values of α, which illustrates that we can use this parameter to specify the "sharpness" of the transitions. When using $W_s(d, \alpha)$ as a basis function for B-spline interpolation of the same target points shown earlier, we find the resulting $p(t)$ shown in figure 4.7, which confirms that increasing α causes the transitions to take place more rapidly. Figure 4.7 also shows that the gradient of $p(t)$ is close to zero at each target point, which makes this type of interpolation suitable for the axis scaling necessary to vary the transition durations. It was found that $\alpha \geq 7.5$ ensures that $W_s(d, \alpha)$ is not sharply peaked at $d = 0$, since smaller values of α tend to introduce unwanted sharp peaks in the resulting $p(t)$.

One problem remains, which is the fact that more flexibility is required in terms of specifying the duration of each monophone in an interpolated set separately from the transition durations between successive monophones. Because the basis function $W_s(d, \alpha)$ ensures that the gradient of $p(t)$ is close to zero at the target points (which are the phone centres), we can solve this simply by inserting constant LSF segments[2] at each target point after $p(t)$ was calculated over only the transition regions. Another possible solution is to duplicate the target points so that the region between an original and a duplicated target point represents a constant segment, and its duration is specified by the spacing of the two points. Note that neither approach will work well when using the original $W(d)$ of equation 4.2, as it does not guarantee a gradient which is close to zero at the target points. In the implementation of the system described in this thesis, the former solution was chosen as it computationally much less expensive than the latter.

Figure 4.8 shows the time-varying ($50ms$ frames, 10 sample increments) measurements of

[2]A constant LSF segment is a set of duplicate LSF vectors for a specified duration.

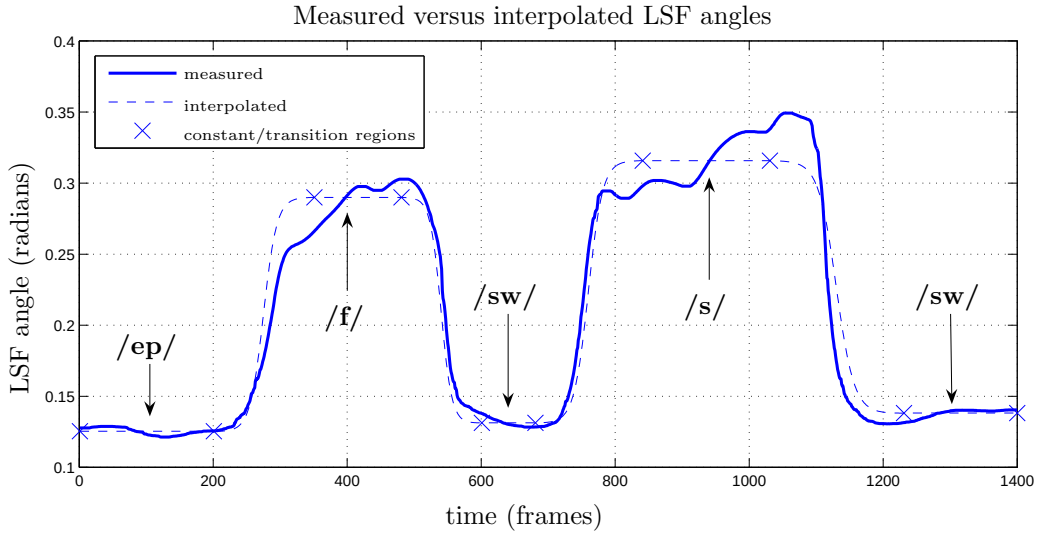


Figure 4.8: *Interpolation of the third LSF within the word "d**efici**t".*

the 3rd LSF of a 30th order LSF vector within the phonetic sequence /ep/→/f/→/sw/→/s/→/sw/ of the word "d*efici*t" (recorded at $24kHz$) together with the interpolated curve through its target values, where constant and transition regions are separated by crosses. Note that it is difficult to measure the LSF trajectories accurately on a short time-frame basis, and therefore the variations in the measured LSF shown in figure 4.8 may be due to inaccuracies in the estimation process rather than actual vocal tract movements, as they are not consistently visible when varying the frame length. Overall, however, the synthetic transitions are fairly similar to the measured transitions, which indicates that the presented B-spline interpolation algorithm is suitable for modelling LSF transitions between phones.

4.2 Excitation Parameter Interpolation

Initial experiments applied the same interpolation scheme described in the previous sections to the excitation signal parameters, since these are included in each phone's parameter vector. However, it soon became clear that this was not ideal. Co-articulation appears more relevant to the vocal tract movements than to the excitation signal, because the oral cavity is physically limited to a certain movement rate, whereas the source signal can change much more abruptly at the glottis. It is for this reason that it was decided to allow the source signal parameters to vary more abruptly than the vocal tract parameters. According to [1], $40ms$ is a suitable duration for the source signal parameter transitions of most phonetic combinations[3]. In the event of a filter parameter transition which is shorter than $40ms$, the

[3]Since a pre-processor for duration modelling is not present, a default transition duration ($40ms$) had to be chosen. For more natural speech output, however, a rule set derived from speech data is required for

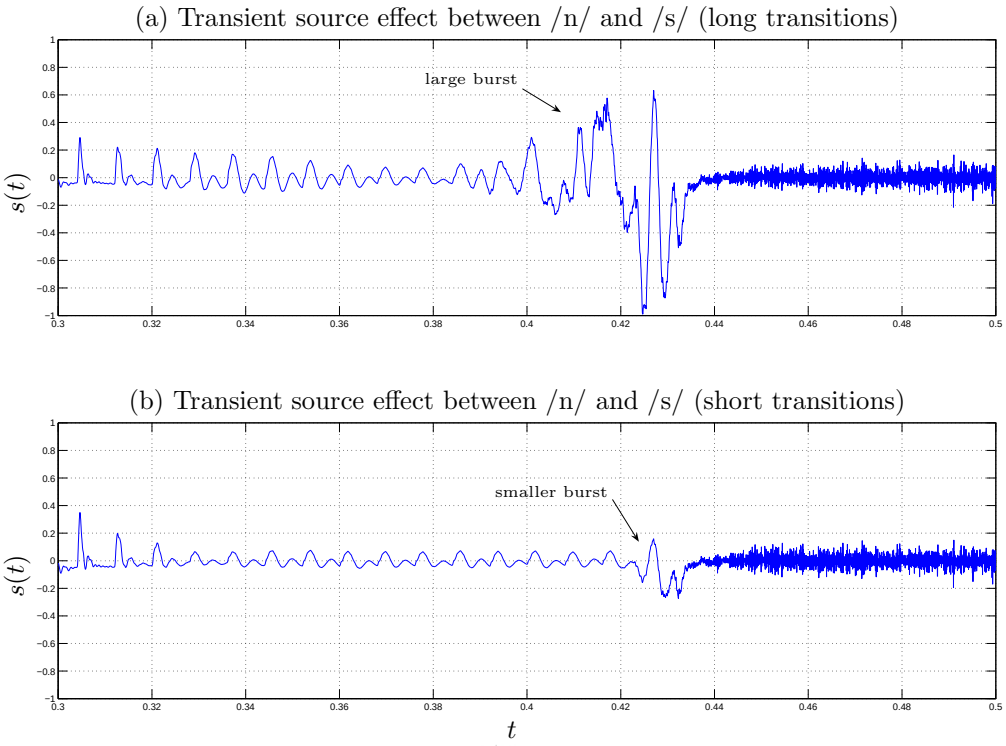


Figure 4.9: *The stochastic energy burst transient effect.*

source signal parameter transition duration is limited so as to not exceed that of the filter parameters.

A secondary motivation for a separate interpolation scheme for the excitation parameters was the difficulty experienced when modelling transitions between sounds with different degrees of voicing, an effect which is more pronounced whenever the V/U ratios of the two sounds differs substantially. As the V/U ratio changes together with the filter parameters during the transition, there are regions where the spectral shape of the filter parameters of one sound is to a certain extent applied to the source signal of the other sound. This can result in considerable and unwanted amplification of an unvoiced sound. In the few individual cases that were considered, it was observed that this is due to the generally much larger gain of the voiced sound's LP filter at lower frequencies than that of the unvoiced sound (see section 2.4.2). This causes a burst of amplification of the unvoiced excitation component before the LP filter coefficients have settled to their target unvoiced values. By shortening the source signal transitions relative to the filter transitions, we can ensure that the latter is already advanced to a point where the source signal of the unvoiced sound is less affected by the filter of the voiced sound. To ensure that this solution applies both when

effective duration modelling [1].

moving from a voiced to an unvoiced sound and vice versa, the source signal transitions are placed at the centres of the filter transitions. Figure 4.9(a) illustrates the problem and (b) the intended solution for a synthetic transition between the nasal /n/ (as in *n*ot) and the unvoiced fricative /s/ (as in *s*ome). Note that the stochastic energy burst is still present when using the shorter duration in (b), but that it is much less pronounced than in (a).

Although the problem is limited by the aforementioned approach, preliminary testing showed that certain phone combinations still produce audible artifacts, especially when the filter transition duration is short. Therefore, in order to suppress the effect more effectively, the source parameter transitions were shifted in time such that a parameter increase occurs only during the final section of a filter transition, and a decrease during the initial section. By doing so, we avoid the situation where a large amount of energy is introduced into a frequency band where it may not be sufficiently suppressed. Experimentally, it was found that the "safe" region of a filter transition for a change in A_n is in its initial or final quarter, but that the transitions of A_v and F_{max} can span the entire duration of the filter transition without producing transient energy bursts.

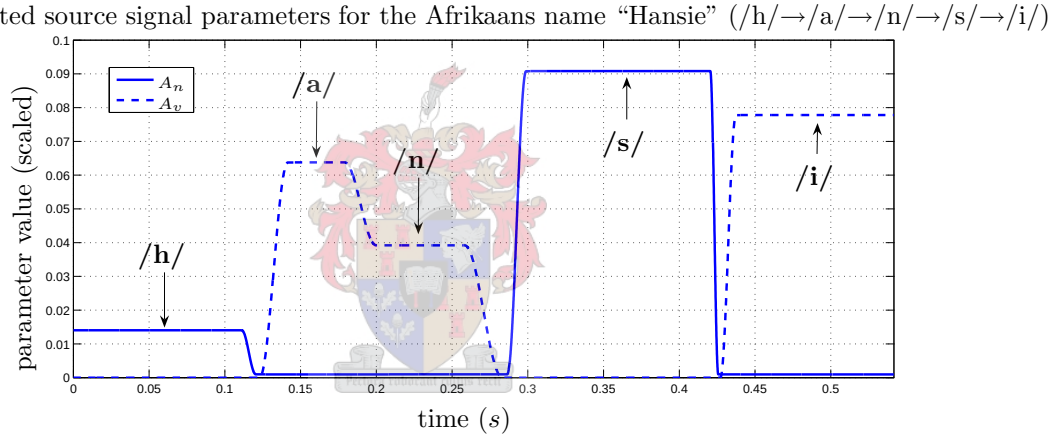


Figure 4.10: *Example of the sinusoidal interpolation of source signal parameters.*

The particular interpolation algorithm used to generate the source signal parameter transitions was found to be of little perceptual importance during informal listening tests. It is suspected that the $40ms$ window is short enough to make it difficult to discern slight variations in the source signal parameters. It was decided to use a scaled and offset half period of a cosine function to ensure the smooth excitation parameter transition. It must be mentioned that $40ms$ should not be considered an absolute, but should rather be scalable in the event where synthetic speech with a variable speaking rate is required, for example when synthesising speech with a different voice quality. Scalability was not implemented at this stage, however, since no duration modelling front-end to the system currently exists. Figure 4.10 shows the interpolated curves of the parameters A_n and A_v for the Afrikaans name "Hansie" as calculated by the described procedure. For clarity, F_{max} (which is interpolated in the same way) is not shown.

4.3 Chapter Summary and Conclusion

Two interpolation algorithms for calculating the LSF vectors within the transition regions between monophones in a synthetic utterance were presented in section 4.1. The first uses Bézier segments for the individual LSF transitions and the second uses B-spline curves. The second algorithm was preferred due to its combined simplicity and flexibility. Two modifications to B-spline interpolation were developed in section 4.1.3 which were not found in the reviewed literature. These are:

- A new sigmoidal basis function that allows for scalable transition rates and ensures an interpolated curve gradient that is close to zero at its target points.
- Flexibility in terms of the duration of constant (phone) and transitional (inter-phone) segments.

The developed LSF interpolation algorithm depends solely on a sequence of specified phone LSF vectors, and is flexible in terms of the durations of the constant and transition regions of such a sequence.

Next, an interpolation algorithm for calculating the source signal parameter transitions between phones was presented in section 4.2. The algorithm is designed for use in conjunction with the LSF interpolation scheme presented in section 4.1.2.

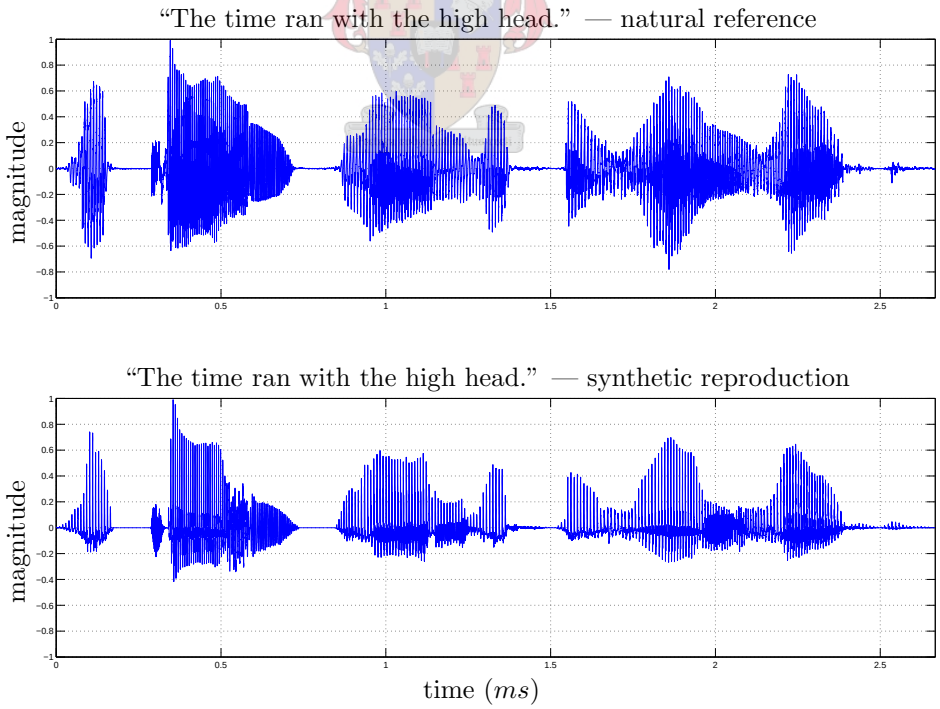


Figure 4.11: *Time waveform of a synthetic sentence (natural reference included).*

"The time ran with the high head."
natural reference (above) and synthetic reproduction (below)

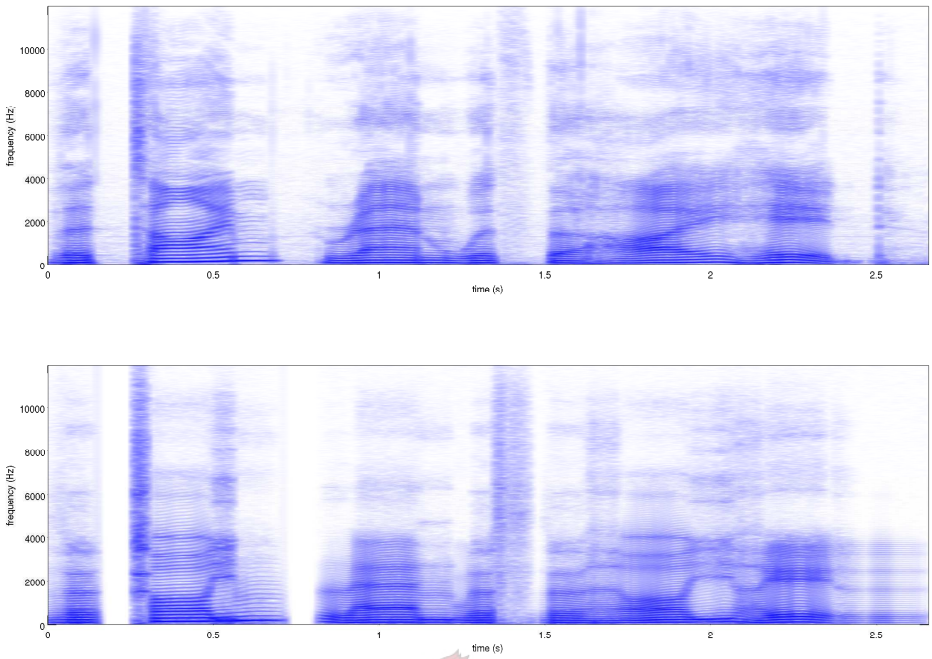


Figure 4.12: *Wideband spectrogram of a synthetic sentence (natural reference included).*

Finally, figures 4.11 and 4.12 show the time waveforms and spectrograms, respectively, of a recorded sentence and its synthetic reproduction as was used during the system's evaluation. The prosodic contours used in the synthetic sentence were estimated from the natural reference (recording) according to the methods described in section 6.2.4, but the phone models were obtained from the phone database (see table A.1) which was recorded independently from the natural sentence shown. The similarities in pitch, magnitude and duration are therefore not surprising. Of note is that the voiced/unvoiced content, including the harmonic frequency band defined by F_{max}, of the synthetic sentence is very similar to that of the natural reference. The only clear exception to this is the final speech sound in the sentence, /d/, which is a voiced plosive and is known to be problematic (see section 3.4). Also, note the similarities between the two spectrograms with regard to formant positions. The formant trajectories are similar in shape, but it appears that the estimated duration model for this example did not allow for sufficiently large transition durations, as the filter transitions are shorter in the case of the synthetic utterance than for the natural reference.

Chapter 5

System Description

The previous chapters have described the relevant methods and algorithms that have been researched and/or developed as part of a parametric monophone speech synthesis system. This chapter describes the system's implementation by short discussions of its libraries and their main functions. The entire system was developed in the *C* language, and only the (optionally compiled) front end and the graphical user interface (GUI) were written in *Python*. Certain mathematical functions for random number generation, FFT calculation, polynomial root finding and matrix algebra were imported from the *GNU Scientific Library* (GSL, see `http://www.gnu.org/software/gsl/`). Audio streaming was implemented using the *Open Sound System*^{TM} (OSS).

Figure 5.1 shows a complete system block diagram. This is later divided into an analysis data flow diagram (figure 5.2) and a synthesis data flow diagram (figure 5.3), both of which are discussed later in this chapter. The system design allows for simple modification by the interchanging of compatible modules. For example, the source signal module is interchangeable between, say, the impulse plus noise model of section 3.2.2 and the harmonics plus noise model of section 3.2.4, because the excitation signal module's header file remains the same. The parametric modelling and synthesis module, which pertains to the particular parametric speech model used, is also interchangeable. In our implementation, this module performs LP analysis (including conversions between LPC's and LSF's) and synthesis, and is used in conjunction with the excitation signal module. However, the excitation signal module is linked only to the parametric modelling and synthesis module, which implies that the core module is unaware of the particular implementation of excitation signal. This configuration was chosen to ensure that future speech models need not assume separability of source and filter. Similarly, the parameter interpolation module is interchangeable, but not necessary. This allows the system to change, for example, the current B-spline interpolation (section 4.1.2) algorithm to an HMM-based LSF generation scheme, or even to remove it altogether (if the monophone synthesis units were to be replaced with, say, diphones).

The description of the system is divided into three sections. The first of these describes the system's initialisation phase, during which the variables pertaining to the speech model are defined. The second section contains a description of the functions associated with the

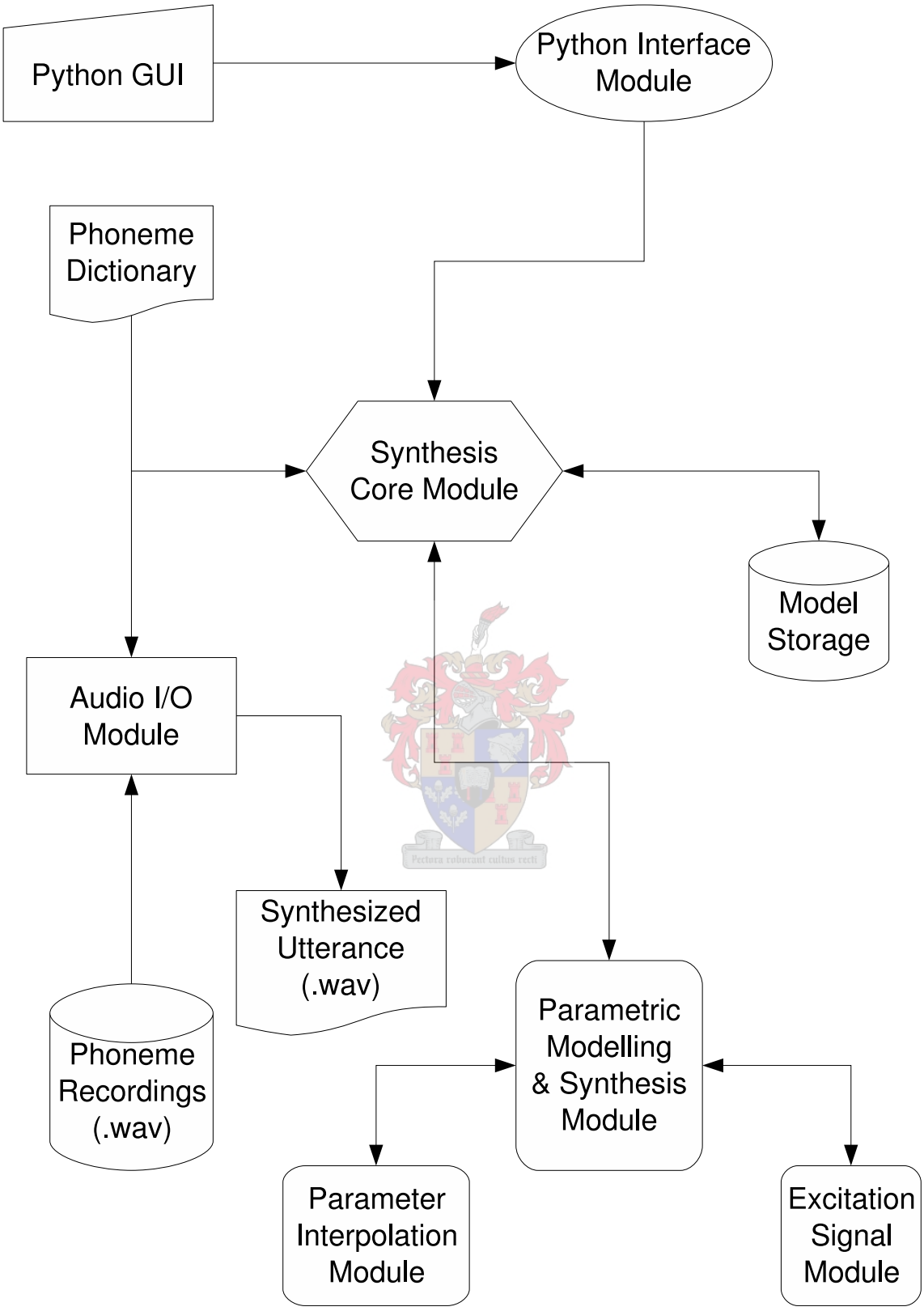


Figure 5.1: *System Block Diagram.*

speech *analysis* phase, in which the parametric monophone models are estimated and stored for later use. The final section is concerned with the *synthesis* phase, during which the relevant parametric phone models are loaded into memory and interpolated as necessary to perform synthesis. The *Python* interface and GUI will not be discussed in this chapter, as these merely serve as an interface to the core module.

From this point, the system will be described in terms of the main functions provided by the synthesis core module. In each case, all the relevant functions which are called during the execution of the particular core module function will be discussed briefly, including those functions which belong to the other modules. The core module is governed by a set of state variables which ensure that the system remains stable and does not, for example, attempt to estimate the phone models before the location of the phone recordings is known. Before continuing, note that all core module functions and types are preceded by "`sco_`". Each particular module has its own identifier prefix, which will be indicated as they are encountered. Also, each module contains a set of error codes and an internal error handler. These will not be discussed as they do not form part of the speech synthesis system's functionality, but rather serve to ensure that the system itself remains stable.

5.1 Initialisation

The initialisation phase is the collection of functions which need to be executed before either the analysis phase or the synthesis phase can commence. During this step, the parameters which pertain to the particular speech model to be used are set and the modules relevant to the analysis and synthesis phases are initialised. All this is done in a single call to the core module function:

```
sco_errortype sco_models_init(char *dict, ...)
```

This function receives as input the file name of the phone dictionary as well as a variable argument list (indicated by "...") containing the model-specific parameters used during the initialisation of the parametric models. The function returns an error code (of the integer type "`sco_errortype`") which describes any problem which may have occurred during its execution. A phone dictionary is a file which serves as an index to the recordings of the different phones in case the models need to be estimated and stored for later use. A short example of such a dictionary is as follows:

```
*fu Fricatives/Unvoiced/
/f    f.wav          [10000;15999]
/T    th.wav         [20000;25999]
*fv Fricatives/Voiced/
/D    dh.wav         [26000;31999]
*vv Vowels/
```

```
/I    i_long.wav    [16000;21999]
/e    ep.wav        [3000;4499]
```

This particular dictionary defines five sounds, which includes two unvoiced fricatives, one voiced fricative and two vowels. The dictionary is divided into sections, each defining a particular type of phone. Each section begins with an asterisk ("*") character, which is followed by a phone type identifier ("f" for fricative, "v" for vowel, etc.) and a character which specifies whether the phone contains any voicing ("v" for voiced, "u" for unvoiced). These two identifier characters are not necessary (for cases where the information is unavailable), but are helpful during the parameter estimation process if specified. For example, a V/U ratio measure (see section 3.3) may incorrectly detect that a certain unvoiced plosive sound contains a degree of voicing. During estimation, this result is adjusted if it is known that the phone is unvoiced. Similarly, the system may suppress the amount of unvoiced energy in vowels to avoid a breathy vowel sound. Also, each section heading contains the subpath where the sections' recordings are located. Within each section of the dictionary, each particular phone is defined by a single line, which starts with a slash ("/"), followed by a mapping symbol (used to represent the sound when a synthetic utterance is defined), the name of the file containing the recorded samples and the particular sample range to use for estimation in square brackets.

The function `sco_models_init()` uses a variable argument list because the parametric modelling and synthesis module and the excitation signal module can change, and therefore do not always receive the same arguments or number of arguments as input. The variable argument list is itself passed to the parametric modelling and synthesis module, at which point the particular parameters contained therein are known and parsed appropriately. For this purpose, the function `sco_models_init()` invokes the following function:

```
model_errortype model_init(va_list *ap, int *params)
```

The prefix "`model_`" is used to identify all functions and types belonging to the parametric modelling and synthesis module. The above function receives a pointer `ap` to the variable argument list and a pointer `params` to a variable which will contain the phone parameter vector length, known only after this function has completed its execution. In our case, the parametric modelling and synthesis module uses LSF's for modelling the filter part of the speech signal, which means that `ap` should contain, among others, the desired sampling rate, the LP order and information pertaining to if and how pre-emphasis should be applied. In our case, the relevant speech model is a source-filter model of speech production, which means that the function `model_init()` will also need to initialise its excitation signal module. It does so by invoking the following function:

```
src_errortype src_init(va_list *ap, int *params)
```

The prefix "`src_`" is used to identify all functions and types belonging to the excitation signal module. The variable argument list `ap` in this case is the same variable argument list

passed to `model_init()`, which has at this point been parsed up to the point where the source signal parameters are located, which may be parameters such as the type of V/U ratio estimation to use, and will be parsed further during the execution of `src_init()`. This function also needs to specify (to `model_init()`) the number of parameters which need to be reserved for the source signal model in the phone parameter vectors, parameters such as A_n, A_v and F_{max}. The variable `params` is used for this purpose.

Finally, `model_init()` must allow the parameter interpolation module to be initialised, which it does by means of the following function:

```
itp_errortype itp_init(va_list *ap)
```

The prefix "`itp_`" is used to identify all functions and types belonging to the parameter interpolation module. Again, the variable argument list is the same as before, which at this point has been parsed up to where the interpolation scheme's parameters are located. Once the interpolation module parameters have been initialised, the parameter setup step is complete and core module's state variables are updated to reflect this so that the next step can commence.

5.2 Analysis Phase

The analysis phase is the collection of functions which are concerned with calculating and storing the speech model parameters so that these can be stored and later retrieved for use during the synthesis phase. It is important to note that the functions provided by the analysis phase are implemented as an independent part of the system, such that once the analysis phase is complete and the necessary parameters have been stored, these functions are no longer necessary for synthesis. This design is motivated by the ideal that the system should eventually be portable to handheld or other devices with limited memory and/or computational power. This implies that, although the analysis phase may be computationally expensive and the phone recordings can be large, synthesis can be performed using only the stored parameters while discarding the original recordings from which they were estimated (as well as the analysis code). Doing so results in a greatly reduced memory footprint, from about 3.7 Mb for the PCM (pulse code modulation, uncompressed) .wav recordings to less than 13 Kb (uncompressed) for 48 phones in South African English, a 99.67% reduction.

With all the relevant parameters having been set during the initialisation phase, the parametric modelling and synthesis module is ready to estimate a parameter vector for each phone indexed by the phone dictionary. Note that if the analysis phase has already been executed for a certain model configuration, the initialisation phase explained in the previous section is sufficient for the system to continue to the synthesis phase. This is because the core module's state variables are updated after the initialisation phase and will reflect whether a set of monophone parameter vectors for that particular model configuration already exists. Figure 5.2 shows a data flow diagram of the steps performed during the analysis phase, with

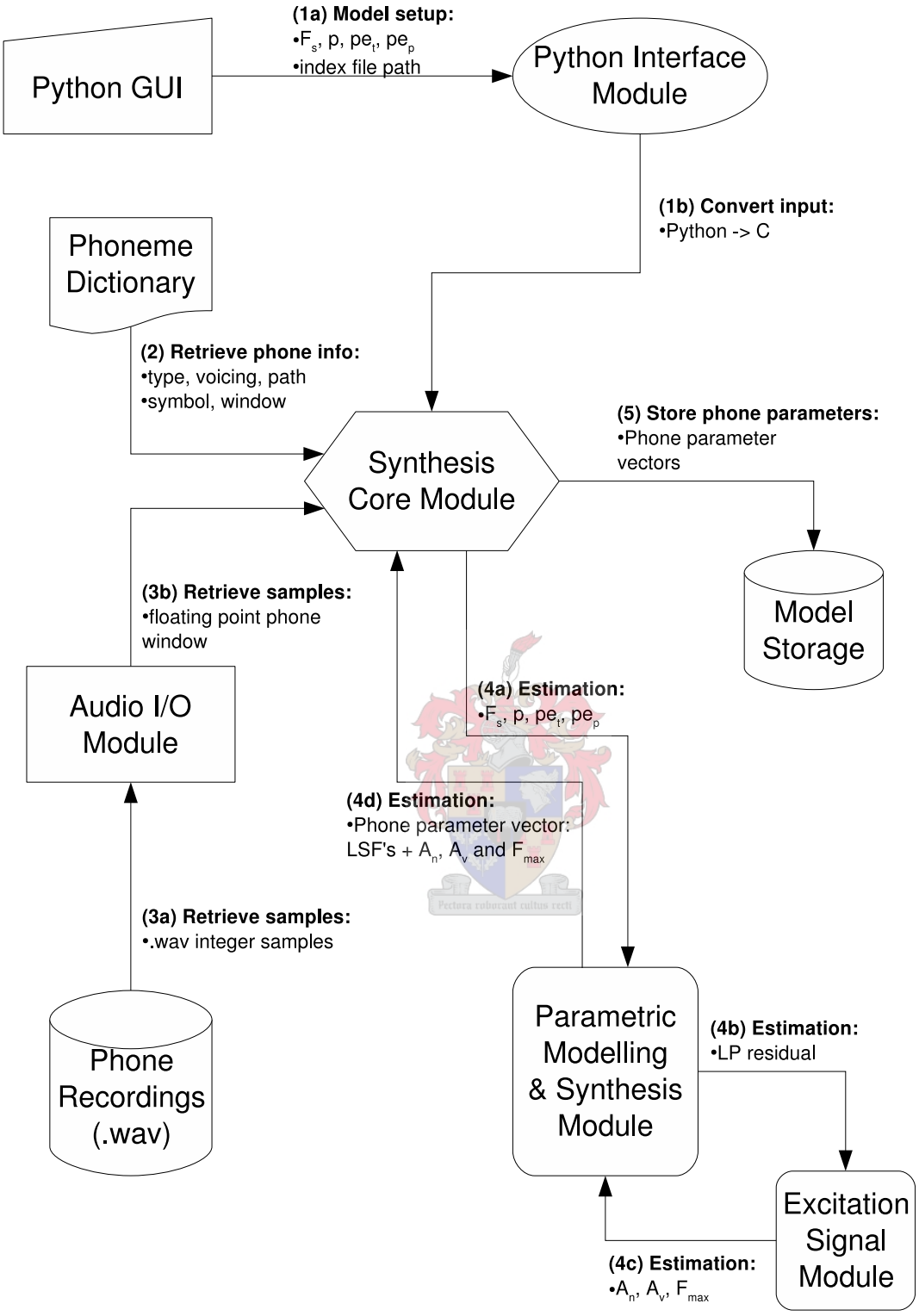


Figure 5.2: *System behaviour during the analysis phase.*

F_s = sampling rate, p = LP order, pe_t = pre-emphasis type, pe_p = pre-emphasis order, A_n = noise magnitude, A_v = voiced magnitude, F_{max} = maximum voicing frequency

the initialisation phase included as a first step. The estimation process is started by calling the core module function:

```
sco_errortype sco_models_compile(void)
```

This function has no arguments, as all the relevant parameters have been set during the initialisation step. This function traverses the phone dictionary and performs the analysis procedure for each phone encountered. The first step in this procedure is to invoke the audio I/O (input/output) module to retrieve the specified samples of a particular phone from disk, which is done by calling:

```
wav_errortype wav_getArray(char wavName[], long *N, int *Fs)
```

The prefix "`wav_`" is used to identify all functions and types belonging to the .wav file section of the audio I/O module. This function reads all the samples from the .wav recording specified by `wavName` into a shared buffer and returns the number of samples read in `N` as well as the sampling rate in `Fs`. Once this is done, `sco_models_compile()` copies only the samples specified by the phone's dictionary entry from the shared buffer for further analysis. The next step is to invoke the parametric modelling and synthesis module to perform the analysis on the floating point samples, which is done by the function:

```
model_errortype model_getparams(double *x, long N, char ph_t, char ph_v,
                                double *a)
```

This function receives as input a pointer `x` to the recorded samples, the number of samples `N`, the phone type `ph_t` and voicing `ph_v` specified by the dictionary and a pointer `a` to the parameter vector created by the core module, which is where the estimated parameters will be located. The implemented parametric modelling and synthesis module uses LP to model the speech signal. The first step is therefore to pre-emphasise (section 2.3.4) the samples in `x`, after which the LP parameters are estimated using the *Levinson-Durbin* algorithm and converted to LSF's, which represent the filter parameters of the current phone and are stored in `a`.

Before the excitation signal module can be invoked to estimate the source parameters, `model_getparams()` applies inverse LP filtering to the samples in `x` to obtain its LP residual. The source signal module is then invoked to estimate the source signal parameters from this residual by a call to the function:

```
src_errortype src_getparams(double *e, long N, char ph_t, char ph_v,
                            double *a)
```

This function receives as input a pointer `e` to the residual signal, its length `N`, the phone type `ph_t` and voicing `ph_v` and a pointer `a` to the location in the current phone's parameter vector where the source signal parameters are to be stored. During its execution,

`src_getparams()` estimates the source signal parameters (such as A_n, A_v and F_{max}) by the methods described in section 3.3. The particular V/U ratio estimator (kurtosis, entropy or m_{f_ε}) used is specified during the initialisation step.

After the calculation of the source signal parameters, the current phone's parameter vector is complete and it is saved to disk. Once this procedure has been followed for all the phones encountered in the dictionary, the analysis phase is complete and the core module's state variables are updated to reflect this. From this point, we refer to such a stored collection of monophone parameter vectors as a *parameter dictionary*, not to be confused with the *phone dictionary*, which refers to the index file used during analysis.

5.3 Synthesis Phase

The synthesis phase describes that part of the system's functionality which is concerned with generating a discrete time synthetic speech signal from a phonetic description of an utterance using the parametric models found in a parameter dictionary. The steps performed during this phase are summarised in the data flow diagram of figure 5.3. At this point, we define a phone *segment* as a string of phone symbols (see section 5.1) and their associated prosodic contours. A segment usually defines a single word, but can also be used to represent either smaller or larger units, depending on the desired pronunciation, which is affected by the segmentation because parameter interpolation is performed over each segment in a sequence individually. We define such a sequence of phone segments, together with global prosodic contours, as an *utterance*. An utterance usually defines a sentence, but may also represent smaller units, such as phrases, or larger units spanning multiple sentences. There are practical considerations, though, since longer utterances require a significant amount of memory to synthesise. This is due to the fact that, after interpolation, each sample of the synthetic utterance represents an entire parameter vector. For example, a 10-second utterance synthesised at $24kHz$ with an LP order of 30 requires about 67 Mb of memory after interpolation, but only about 1.83 Mb once the parameter vectors have been translated to floating point samples. When saved to a 16-bit (mono) PCM .wav file, it requires only about 470 Kb.

The synthesis phase is divided into two steps. The first, interpolation, is performed whenever a phone segment is added to an utterance. During the second step the utterance is synthesised, i.e. the interpolated parameter vectors are converted to a discrete time signal and can be streamed to an audio device or saved to disk in .wav format.

5.3.1 Interpolation

Each time a phone segment is added to the utterance, the system needs to set up the necessary parameter vectors and interpolate them over the duration of the segment. A segment can be added using the core module function:

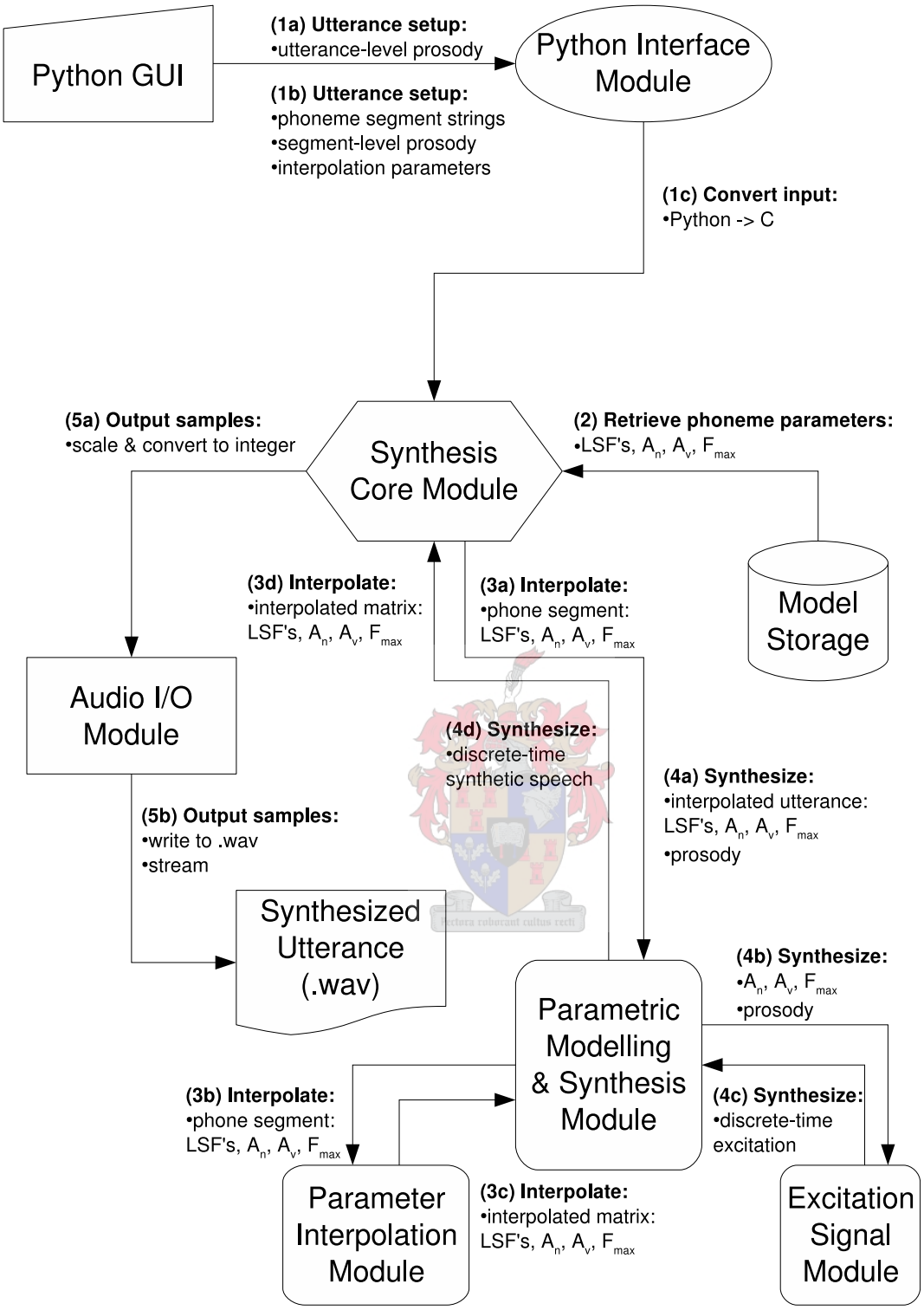


Figure 5.3: *System behaviour during the synthesis phase.*

A_n = noise magnitude, A_v = voiced magnitude, F_{max} = maximum voicing frequency

```
sco_errortype sco_segment_add(char *s, itp_curve *f0, itp_curve *a_n,
                              itp_curve *a_v, long *dur, int index, ...)
```

This function receives as input the mapped phone symbol string `s`, pointers `f0` (pitch), `a_n` (unvoiced magnitude) and `a_v` (voiced magnitude) to the segment-level prosodic curve definitions, a pointer `dur` to the segment durations, an index `index` in the current utterance and a variable argument list. The type `itp_curve` is a parametric definition of a smooth curve, such as a sinusoid or the scaled Rayleigh PDF referred to in section 3.4. It is a *C* "`struct`" that contains the type of curve, a duration, an array of curve parameters and, optionally, a name of a curve file located on a storage device. By making use of `itp_curve`'s, the system ensures flexibility in terms of the generation of prosodic contours. Note that the curve file that is referenced by an `itp_curve` can be generated by any program, not just the system described in this thesis, and its length is automatically adjusted to fit the length of the `itp_curve` when it is loaded. Note that the voiced and unvoiced magnitude parameters are specified separately. This configuration allows us the freedom to synthesise voices with different breathy qualities, or even vary the amount of frication within a segment, if desired. The segment duration referenced by `dur` must be an array of length $2k + 1$ for a mapped symbol string `s` of length k. This means that the first entry in the array contains the duration of the first phone, and every second (odd-numbered) entry thereafter the duration of the consecutive phone. The alternate (even-numbered) entries contain the transition durations. The parameter `index` refers to the position[1] in the current utterance where the segment should be inserted, and the variable argument list contains information that pertains to the interpolation scheme, such as the parameter α of the B-spline sigmoidal basis function (equation 4.9).

Before continuing to the parameter interpolation, `sco_segment_add()` creates a new segment in the utterance at the location indicated by `index`. It then reads each new phone's parameter vector from the parameter dictionary, copying the vectors of those phones which already exist in memory. The core module also provides a function to remove a specific segment from the defined utterance as well as a function which clears the entire utterance. These functions are, respectively:

```
sco_errortype sco_segment_remove(int index)
```

and

```
sco_errortype sco_utterance_clear(void)
```

The removal function `sco_segment_remove()` receives as input only the index `index` of the segment that needs to be removed. It proceeds to de-allocate all memory that was

[1] For an utterance consisting of N segments, the segments are ordered numerically $(0 \ldots N-1)$, such that *index* refers to the numerical index in the utterance where the current segment should be inserted.

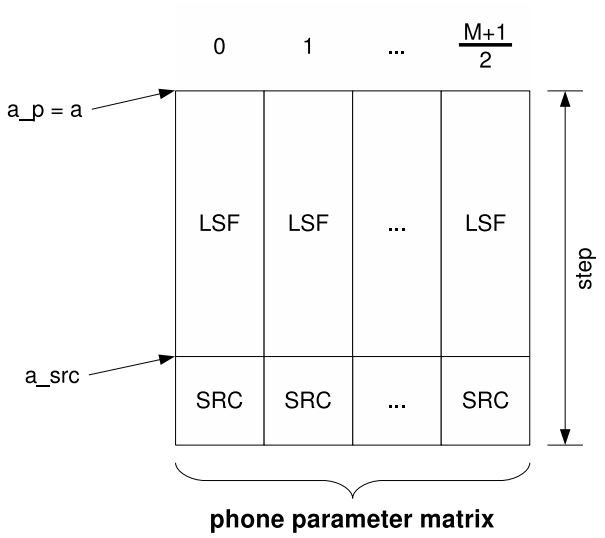


Figure 5.4: *An example of a matrix of phone parameter vectors.*

previously allocated to the segment and removes it from the indexed list of phone segments. The function `sco_utterance_clear()` receives no arguments, as it iteratively removes all the segments in memory similarly to `sco_segment_remove()`.

The source signal parameter interpolation scheme may vary according to the types of the different phones within a segment. Therefore, to obtain this information, `sco_segment_add()` invokes the following function:

```
sco_errortype get_phone_types(char *s, char *types)
```

This function reads the types of all the phones referred to by the symbols in `s` from the *phone* dictionary, and stores this information in `types`. Note that the phone types need not be known, but may improve the final synthesis quality because certain sound types such as plosives may require special treatment (see section 3.4). Once the parameter vectors and the phone types are known, `sco_segment_add()` invokes the following function to begin the interpolation procedure:

```
model_errortype model_interp(va_list *ap, double *a_p, char *ph_types,
                             int M, long *D, double **L)
```

This function receives as input a pointer `ap` to the variable argument list of `sco_segment_add()`, a pointer `a_p` to the phone parameter matrix consisting of the consecutive parameter vectors as illustrated in figure 5.4, the phone types `ph_types`, the number of durations `M`, the duration array `D` and a pointer `L` which will refer to the interpolated parameter matrix, for which memory is allocated during the execution of `model_interp()`. If $\texttt{M} = 1$, which is to say there is only one phone in the segment, `model_interp()` simply copies the single parameter vector for the required duration. Otherwise, the following function is invoked to perform the interpolation:

```
itp_errortype itp_interp(va_list *ap, double *a, int step, int K, int M,
                         long *N, double *L)
```

This is the function which interpolates the filter parameters according the modified B-spline interpolation algorithm developed in section 4.1.2. Its arguments are the variable argument list `ap` passed along from `sco_segment_add()`, the parameter matrix `a` (indicated in figure 5.4), the number of parameters[2] `step` between successive LSF vectors in the phone parameter matrix (equal to a single parameter vector length, including source signal parameters, since these need to be ignored), the number of parameters (LSF's) `K` to interpolate, the number of durations `M` (implying that the number of phones in `a` is `(M+1)/2` as shown in figure 5.4), the duration array `N` and the interpolated parameter matrix `L`, where the result will be stored. After obtaining the relevant parameters from `ap`, the function proceeds to perform the interpolation for each of the `K` parameters, saving the interpolated curves in `L`. The number of vectors written to `L` is therefore equal to the sum of all the durations in `N`. Note that only the transition (even-numbered) entries in `N` are used for B-spline interpolation, and the LSF's pertaining to the constant section (odd-numbered) entries in `N` are merely copied.

Upon completion, `itp_interp()` returns control to `model_interp()`, which proceeds to convert all the interpolated LSF vectors to LPC's for synthesis. Having done this, the excitation signal parameters need to be interpolated, for which purpose `model_interp()` performs a call to the following function:

```
src_errortype src_interp(double *a_src, int step, char *ph_types, int Md,
                         long *Nd, double *L_src)
```

Note that `src_interp()` is an excitation signal module function. This is because of the fact that the excitation signal module itself is interchangeable between different source signal models, which may mean that the source signal parameters need to be interpolated using a different algorithm for a different model. This function receives as input a pointer `a_src` to the first source signal parameter in the first phone parameter vector (which is equal to the parameters `a_p` of `model_interp` and `a` of `itp_interp()` plus the number of LSF's p, see figure 5.4), the number of parameters `step` between successive sets of source signal parameters in the phone parameter matrix (equal to a single parameter vector length), the phone types `ph_types`, the length `Md` of the durations array (the same as `M` in `itp_interp()`), the durations array `Nd` (the same as `N` in `itp_interp()`) and the location `L_src` of a temporary array (managed by `model_interp`) in which to store the interpolated parameters. Using the information contained in `ph_types` (if available), `src_interp()` generates a parameter curve for each source signal parameter. As mentioned in section 4.2, the implemented algorithm uses sinusoidal transitions of $40ms$, positioned in the centre of each specified transition of

[2]The value of the parameter *step* is therefore the parameter vector length of each phone in *a*.

Nd. These are specified using itp_curve's, for which the curve samples are generated using the following function:

```
itp_errortype itp_calc_curve(itp_curve *c, double *x)
```

This function has as its input arguments a pointer c to an itp_curve and a pointer to the location where the calculated curve should be stored. As mentioned earlier, the *C* "struct" itp_curve contains all the information needed to calculate the curve, or a path string where it can be loaded from disk.

Once the sinusoidal transition curves have been generated between the values found in a_src, src_interp() returns control to model_interp(), which proceeds to copy the interpolated source signal parameter curves into the interpolated parameter matrix. This step concludes the interpolation procedure for a single phone segment, which means that the interpolated parameter matrix sufficiently describes the phonetic content of that segment for synthesis. However, before synthesis can begin, sco_segment_add() concludes the interpolation step by generating/loading the specified prosodic curves f0, a_n and a_v using itp_calc_curve().

5.3.2 Synthesis

Just prior to the LP synthesis procedure, the global (utterance-level) prosodic contours must be specified. This is done by the following core module function:

```
sco_errortype sco_utterance_init(itp_curve *f0, itp_curve *a_n,
                                 itp_curve *a_v)
```

It receives as input the three prosodic contours as itp_curve's f0, a_n and a_v, from which the curve samples are calculated/retrieved using itp_calc_curve(). Having done this, the entire utterance is parametrically defined in memory and synthesis can be performed using the following core module function:

```
sco_errortype sco_utterance_synthesise(void)
```

Note that this function requires no parameters, as all the information regarding the sample generation is contained in the core module's global interpolated segment parameter matrices and prosodic contours, and all the information pertaining to the parametric models themselves have been set up in the parametric modelling and synthesis as well as the excitation signal modules. For this purpose, sco_utterance_synthesise() invokes the following function:

```
model_errortype model_synthesise(Param_desc Param, Phon_desc Phon,
                                 double *y)
```

This function belongs to the parametric modelling and synthesis module because the core module is unaware of the particulars of the parametric speech model that is being used. As inputs, `model_synthesise()` receives a `Param_desc` descriptor (pointer to a "`struct`") `Param`, which contains the utterance-level parameters (F_s, p, F_0, A_n, A_v and the total utterance length), a `Phon_desc` descriptor `Phon`, which is a phone segment descriptor and the array `y` in which the synthetic speech samples are to be stored. A `Phon_desc` contains the segment length, the interpolated parameter matrix and the prosodic curves of a single segment, as well as a pointer to another `Phon_desc`. This pointer allows the phone segments to be tied together as a list, and therefore `Phon` should refer to the first segment in the list so that each consecutive segment can be found by traversing the list. While doing so, `model_synthesise()` calculates each output sample using the past p output samples of the LP filter corresponding to the current sample, and the appropriate excitation signal sample as input. Each excitation signal sample must be calculated from the source signal parameters in the interpolated parameter matrix as well as the corresponding values of the prosodic curves at that point, all of which is done by the following function:

```
src_errortype src_gen(double *a, double f0, double a_n, double a_v,
                      double *e)
```

This function has the following arguments: `a`, the source signal parameters in the interpolated parameter matrix; `f0`, `a_n` and `a_v`, the instantaneous pitch, unvoiced magnitude and voiced magnitude values, respectively, and a pointer `e` to the excitation signal sample. Note that the instantaneous values of `f0`, `a_n` and `a_v` are found by multiplying their utterance-level (absolute) values by their segment-level (relative) counterparts. In our case, `src_gen()` uses a GSL random number generator to generate the Gaussian noise of the unvoiced part and the sinusoidal model of section 3.2.4 for the voiced part.

Once the LP filtering operations have been applied at each desired point to obtain the output samples in `y`, the synthetic utterance is complete and can be streamed to an audio device and/or written to a .wav file. The streaming is done by calling the following function:

```
aud_errortype aud_play(int Fs, long N, int channels, size_t nbits)
```

The prefix "`aud_`" is used to identify all functions and types belonging to the OSS streaming section of the audio I/O module. This function receives as input the sampling rate `Fs`, the number of samples `N`, the number of audio channels `channels` and the number of bits per sample `nbits`. Using these parameters, this function streams the output samples to an audio device using an OSS interface. Note that the output samples are not specified in the arguments of `aud_play()`. This is because the samples need to be converted to integer values and copied to a buffer located in the audio I/O module prior to the call to `aud_play()`. To write the output samples to a .wav file on disk, the audio I/O module provides the following function:

```
wav_errortype wav_writeArray(char wavName[], long N, long Fs)
```

This function receives as input arguments the name of the output file `wavName`, the number of samples `N` and the sampling rate `Fs`. The current implementation supports only the 16-bit mono PCM .wav format. Again, the output samples need to be converted to integer values and copied to a buffer located in the audio I/O module prior to the call to `wav_writeArray()`.

5.4 Chapter Summary and Conclusion

This chapter has described the software implementation of a parametric monophone speech synthesis system. The system's modular design allows the interchangeability of several of its modules, including all those which pertain to the speech signal modelling, synthesis and interpolation methods and algorithms. This eases future modifications and improvements to these aspects of the system. The design also simplifies the testing and comparison of different modelling techniques, as the external system interface remains the same when these modules are changed. The system has a very small memory footprint of approximately 120 Kb, which includes the parametric phone dictionary (database) of 47 phones in South African English shown in Appendix A. In principle, this makes the system suitable for use in portable devices. Because the system was designed for minimal data dependency, adapting it to different languages should be as simple as creating a new phone dictionary which indexes the recordings of the new language's phones. Such a dictionary file requires minimal linguistic information. However, some languages contain special sound classes (such as the "click" sounds common in Southern African languages) which may require special modelling techniques.

Chapter 6

Evaluation and Results

The evaluation of synthetic speech is not a simple process. This is mainly due to the fact that speech synthesisers are used for human interaction, which means that the ultimate test of quality is the reaction of the listener. This implies that subjective tests are the most appropriate way of measuring the quality of synthetic speech. Although various objective metrics have been developed for the evaluation of synthetic speech signals [24], these are most useful for determining the amount of information lost when speech is encoded, as is the case in speech coding algorithms. However, these metrics require an original speech signal as a reference to compare to the (distorted) coded speech signal. For an utterance generated by a TTS system, such a reference utterance generally does not exist. For more complete discussions on TTS system evaluation, see [23], [25] and [40].

For the reasons stated above, it was decided to perform only subjective tests for the evaluation of the synthesiser developed in this thesis. Unless otherwise stated, all test material was generated using the LSF interpolation algorithm specified in sections 4.1.2 together with the excitation signal parameter interpolation method described in section 4.2. Plosive sound energy bursts were modelled according to the method described in section 3.4. The excitation signal used was the sinusoidal model of section 3.2.4 with additive Gaussian white noise (see section 3.2.1). A set of high quality phone recordings ($F_s = 24kHz$) were made in a quiet room using a high quality microphone connected to a mixer and a high quality USB audio capture device. The list of sounds used is shown in table A.1 of Appendix A. A total of 30 LPC's were estimated for each sound without pre-emphasis using the *Levinson-Durbin* algorithm. The parameters A_n, A_v and F_{max} were estimated[1] according to the methods described in section 3.3. The parameter F_{max} was estimated using the measure m_{f_ε} defined in section 3.3.1 with exponential curve fitting applied according to the procedure detailed in Appendix B.

[1]For voiced plosives, F_{max} was limited to the range $2kHz \leq F_{max} \leq 8kHz$ due to the difficulty associated with estimating this parameter for plosive sounds (see section 3.3).

6.1 Subjective Tests

The general procedure of a subjective test is to assemble a group of listeners and require each to listen to and subsequently evaluate a number of synthetic utterances. There are several variables within this procedure, each of which has a significant impact on the acquired results. The first of these variables is the choice of audience. Listeners should generally be chosen to be representative of the system's target audience, and the size of the group should be large enough such that the results from the group are statistically representative of the target audience. Most published evaluations consider between 10 and 20 listeners to be sufficient for this purpose. Next, the utterances which will be evaluated must be chosen. Choices vary in terms of the number, type (words, phrases, full sentences, etc.), order of presentation and how many times a listener is allowed to hear each utterance. Finally, the specific questions asked to the listeners must be chosen carefully, since these impact directly on the listeners' response. The questions must be formulated in such a way that the responses are aimed specifically at a those aspects of the synthetic utterance that are under evaluation.

Two aspects of synthetic speech that are most often evaluated are intelligibility and naturalness. The former is concerned with measuring the understandability of the synthetic speech, whereas the latter is aimed at quantifying how the quality of the synthetic utterance approaches that of a natural utterance. Interestingly, the most intelligible TTS system is not always also the most natural [7]. The synthesiser developed in this thesis has been aimed at producing intelligible speech (see section 1.5), and very little attention has been given to the modelling of prosody, which is one of the main factors influencing naturalness [27]. Hence, only intelligibility has been tested for. The incorporation of prosodic modelling for improved naturalness would include the development of a text- and linguistic analysis front-end, and remains the subject of future work (see section 7.2.1).

Sections 6.1.1 and 6.1.2 describe two types of tests which are commonly used in the evaluation of the intelligibility of TTS systems, namely rhyme tests and semantically unpredictable sentences. Both of these tests were used to evaluate the system described in chapter 5, and the test procedures and results are shown in section 6.2.

6.1.1 Rhyme Tests

The diagnostic rhyme test (DRT), originally introduced in [12], is aimed at measuring the confuseability of certain speech sounds. The test consists of 96 word pairs which differ only with respect to their initial consonants, chosen to represent certain acoustic qualities such as voicing, nasality and compactness. For each word pair, only one word is played to the listener, and he/she is required to identify which of the two was heard. One can use the resulting average error rate of a group of listeners as an indication of the degree of intelligibility of a TTS system. The overall confuseability between certain sounds can also be calculated.

A related test, the modified rhyme test (MRT), is described in [16]. This consists of 50

sets, called ensembles, of 6 words each. There are 25 ensembles where the initial consonant differs for each of the 6 words, and 25 ensembles where the final consonant varies. The complete list of MRT words is shown in Tables C.1 and C.2 of Appendix C. As opposed to the DRT, the MRT tests not only for initial but also for final consonant confuseability. For this reason the MRT was chosen as an evaluation method. The test procedure is to present the listener with all 50 ensembles in random order. One of the 6 ensemble words is randomly selected and played to the listener (only once) in each case. The listener must then choose the correct word from the particular ensemble. As for the DRT, the results can be used to quantify the intelligibility of a TTS system based on the confuseability of its consonants.

6.1.2 Semantically Unpredictable Sentences

Semantically unpredictable sentences (SUS) are sentences which are syntactically correct, but meaningless. They are constructed by randomly choosing words within defined syntactic classes. The use of SUS for the evaluation of TTS intelligibility was introduced in [3], and it has since been integrated into several TTS evaluation procedures, for example [47]. The motivation behind the use of SUS is to prevent a listener from being aided by the context of a word in order to identify it. The test forces the listener to focus on the acoustic quality of each word and avoids the deduction of a word's identity from its context. Because SUS sentences are difficult to understand, results are often very poor when evaluated at sentence-level. Many researchers therefore measure the accuracy at word-level or even phone-level after optimal alignment of the true SUS sentence with the listener's answer (insertions and deletions contribute as errors).

A total of 5 different syntactic structures were presented in [3], but these do vary across different languages. A typical SUS test procedure allows a listener to become accustomed to the acoustic and linguistic aspects of the test sentences using 10 sentences (2 from each syntactic structure), and uses 50 sentences (10 from each structure) for the test itself, although larger sets are not uncommon (for example, [7]). To prevent learning effects, listeners are only allowed to hear each sentence once.

6.2 Testing and Results

As mentioned in section 6.1, the focus of the listening tests was to evaluate the intelligibility of the synthetic speech. For this purpose, we carried out an MRT and a SUS test. The following sections detail the test conditions and procedure and present the results.

6.2.1 Test Conditions and Procedure

The tests were administered in a laboratory where, on average, 10–15 people were present at any given time. Background noise was therefore not eliminated, but it was not sufficient to disrupt the tests at any stage. The tests were performed using high quality headphones

connected to a personal computer (PC) via an external audio device. All audio files (PCM, 16 bits, mono) were played at a sampling rate of $24kHz$.

A simple GUI was developed in order to ease the test procedure. Appendix D shows all the relevant steps as they were presented to the listeners. After some general information regarding the tests (see figure D.2), each listener was presented first with the MRT, followed by the SUS test. More detailed instructions were shown immediately before each test.

6.2.2 Listeners

1. **Gender?**

Male	Female
23	2

2. **Age group?**

20–24	25–29	30–39
15	8	2

3. **How much synthetic speech have you heard in the past?**

"None"	"A few words or short sentences."	"More than ten full sentences."	"I hear synthetic speech on a regular basis."
3	9	10	3

4. **Is South African English your first language?**

Yes	No
6	19

Table 6.1: *Listening tests: group information*

A total of 25 listeners participated in the tests. Each listener was asked to provide some background information (see figure D.3), and their replies are summarised in table 6.1. Of note is that for the majority of the listeners, South African English is not considered to be their first language. This may have had a slightly negative impact on the results, since all tests were performed on South African English utterances. Also, the group does not appear to be biased as far as synthetic speech exposure is concerned[2].

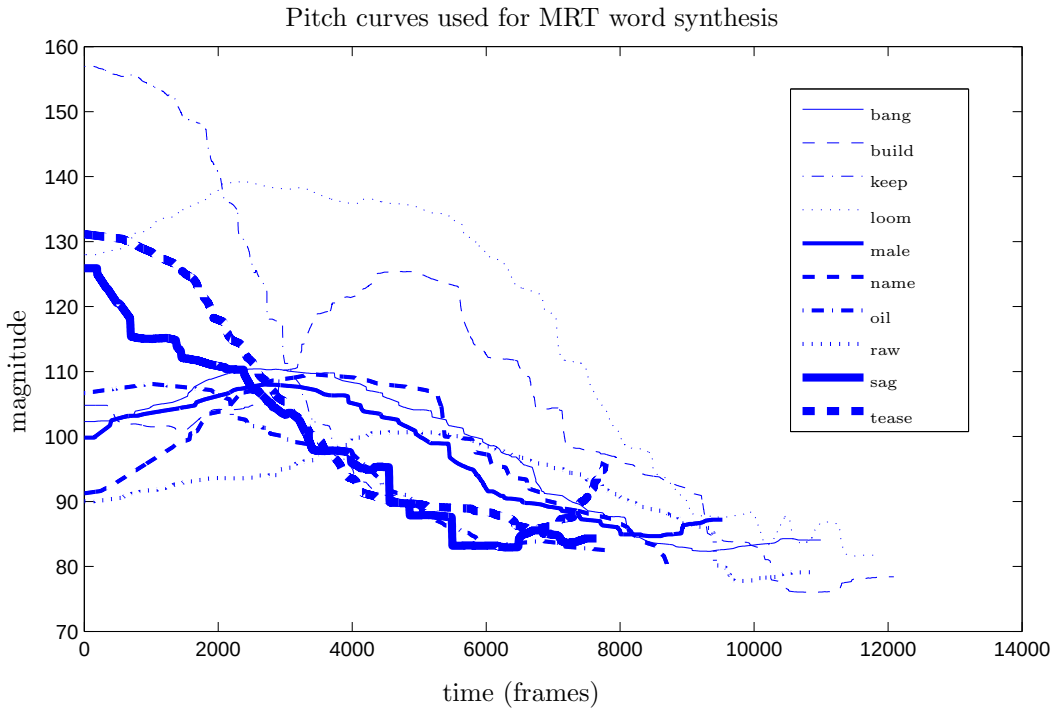


Figure 6.1: *The 10 pitch curves extracted for synthesis of the 300 MRT words.*

Test set	Number of words	Correct
All words included	1200	67.67%
Words containing voiced plosives omitted	817	73.81%
Words containing approximants omitted	910	68.57%
Words containing either of the above omitted	596	75.00%

Table 6.2: *MRT scores for various word subsets.*

6.2.3 MRT - procedure and results

In order to generate the complete set of MRT words without a text- and linguistic analysis front-end, each word in the MRT list was manually transcribed to a phonetic representation using the phones listed in table A.1. A duration model was manually defined for each of the 50 ensembles for the common phone segments within the ensemble. For the initial or final consonants, durations were set according to a simple set of rules:

- Affricates — combination of a plosive ($8.3ms$) and a fricative ($31.25ms$).
- Approximants — $125ms$, unless it was the final consonant in the word ($167ms$).

[2] A strong presence of synthetic speech experts in the group may cause the test results to be overly positive, since exposure to other forms of synthetic speech may aid such a person in identifying certain sounds.

Speech output	Average Correct
Natural speech	99.47%
Formant synthesisers	88.55%
LPC synthesiser	64.44%
Concatenative synthesiser	62.78%

Table 6.3: *MRT scores for natural speech and several TTS systems (from [26]).*

- Fricatives — 83ms, unless it was the final consonant in the word (167ms).
- Nasals — 83ms, unless it was the final consonant in the word (167ms).
- Plosives — 33ms, unless it was the final consonant in the word (62.5ms).
- Transitions — the lesser of 42ms and the duration of the preceding phone, unless the preceding phone was an approximant (125ms).

It must be stated here that this procedure was necessary but not optimised and therefore not ideal. Automatic duration modelling belongs to the realm of prosodic modelling, and falls outside of the scope of this thesis. With regards to the modelling of intonation (pitch and magnitude), all words were synthesised at the same constant magnitude level and a pitch curve was randomly selected for each instance from the set of ten curves shown in figure 6.1. These pitch curves were estimated (and corrected manually) from the recordings of the words shown in the legend. The curve trajectories were deliberately chosen to be very similar in an effort to prevent listeners from being biased toward any particular pitch curve.

After presenting the listener with a set of instructions regarding the MRT (see figure D.4), 2 ensembles (1 initial consonant, 1 final consonant) were chosen at random for each listener to train on. The listener had the freedom to listen to all twelve words at his or her leisure (see figure D.5). Upon choosing to proceed, the listener was presented with the remaining 48 ensembles, one at a time, according to the procedure described in section 6.1.1 (see figure D.6). Scoring is simple, since each answer was chosen from an ensemble, and results are shown in table 6.2.

A comparative MRT evaluation of natural speech, 7 formant synthesisers, 1 LPC synthesiser and 1 segment concatenation synthesiser is presented in [26]. The results are summarised in table 6.3. As one could expect, natural speech yielded the highest word accuracies. Although the formant synthesisers averaged 88.55%, their individual accuracies ranged from 62.56% to 96.75%. Comparing the overall accuracy of 67.67% obtained by the system described earlier, we find that it compares best with the LPC synthesizer at 64.44%, which makes sense because our system is itself an LPC synthesizer.

Also shown in table 6.2 is the percentage of correct words when all words containing voiced plosives are omitted from the scoring set. These sounds include /b/ (as in "*b*aby"),

Test set	Accuracy
Sentences	0.53%
Words	41.25%
Words (article "the" omitted)	29.14%
Phonetic transcriptions	47.97%

Table 6.4: *Overall SUS test scores on sentence-, word- and phone-level.*

/d/ (as in "*d*eath") and /g/ (as in "*g*un"). It is shown because of the known difficulty the current synthesiser has in modelling voiced plosives (see section 3.4). Further analysis showed that 174 errors (44.85% of all errors) occurred in words containing voiced plosives, 131 (33.76% of all errors) of which are the result of a voiced plosive being classified incorrectly.

Some listeners commented that the approximants /rt/ (as in "*r*ed") and /l/ (as in "*l*egs") were very unclear. The data seems to support this statement when these sounds are omitted from the scoring set, as there is a slight increase in the word accuracy when doing so, although it is less pronounced than in the case of voiced plosives. Analysis showed that 102 errors (26.29% of all errors) occurred in words containing /rt/ or /l/, of which 38 (9.79% of all errors) were caused by an incorrect classification of one of these sounds.

Other sound classes were selectively omitted from the scoring, but resulted in only small changes in the word accuracy, and are therefore not shown in table 6.2. The final row in the table shows that the accuracy rises to 75% when all words that contain any of the problematic sounds (/b/, /d/, /g/, /rt/ and /l/) are omitted.

6.2.4 SUS - procedure and results

For the SUS test, only 15 sentences (3 from each syntactic structure) were synthesised due to the difficulty associated with the manual definition of the phonetic and prosodic elements. For a more complete study (such as [7]), as many as 300 sentences were used. The 15 sentences were first recorded, after which each was manually transcribed phonetically using the sounds of table A.1. Once this was done, phone and transition durations were manually extracted from each recording for use in the generation of the corresponding synthetic utterance. A pitch curve was also estimated (and corrected manually) from each recording, and used for synthesis of the appropriate sentence. Because all phone models are not synthesised at the same energy level, some words in the utterance were very loud compared to others. To correct this, a smoothed magnitude envelope was calculated for both the recording and the synthetic sentence. These envelopes were then used to normalise the energy envelope of a synthetic sentence and impose that of the recorded utterance on each synthetic utterance. Although the relative loudness of the speech sounds was improved greatly using this procedure, certain artifacts were introduced in sections of the synthetic utterances

Sentence	Word-level	Word-level (w/o "the")	Phonetic
1–1 The state went at the hot point.	51.19%	38.33%	58.93%
1–2 The school stayed for the new tube.	42.29%	31.20%	46.67%
1–3 The time ran with the high head.	65.14%	56.80%	71.33%
2–1 The poor sense hit the tax.	38.67%	19.00%	42.82%
2–2 The short field found the step.	59.33%	42.00%	64.00%
2–3 The thin job got the voice.	32.67%	12.00%	40.00%
3–1 Start the trial and the fund.	36.00%	18.00%	43.58%
3–2 Call the game and the front.	45.33%	42.00%	57.78%
3–3 Live the plant or the test.	68.33%	61.00%	72.94%
4–1 When does the dog lead the hard set?	17.00%	10.67%	26.18%
4–2 Why does the sign learn the green bed?	26.00%	20.00%	27.09%
4–3 How does the chance plan the cold roof?	54.00%	46.67%	58.17%
5–1 The song marked the branch that burned.	27.43%	7.20%	38.96%
5–2 The truth helped the leg that failed.	32.57%	15.20%	35.83%
5–3 The top drew the pool that died.	32.00%	23.20%	45.89%

Table 6.5: *Individual SUS scores at word- and phone-level.*

where the phone durations (especially in transition regions) of the synthetic sentences did not correspond exactly to the recordings.

Because the SUS test was performed directly after the MRT, and because only 15 sentences were generated, it was decided that no listener training is necessary prior to the SUS tests. It was assumed that a listener was used to the acoustic quality of the synthetic speech after the MRT. In order to prepare the listener for the linguistic abnormalities in SUS, the test instructions showed 5 example sentences (1 from each syntactic structure) which were chosen to be minimally correlated to the test sentences at word-level (see figure D.7). The sentences were presented according to the procedure detailed in section 6.1.2 (see figures D.8 and D.9). The syntactic structures were taken from [3] and are shown, together with the 15 test sentences, in table C.3 of Appendix C.

The results of the SUS test are shown in table 6.4. At first glance, a sentence accuracy of 0.53% (only sentences 3–3 and 4–3 were transcribed correctly, once each) appears to be very poor. However, other studies show that sentence-level SUS test scores are typically in the range 10–20% for natural as well as synthetic speech, which reflects the difficulty of the test [3]. This is attributed to the cognitive difficulty associated with transcribing semanticaly unpredictable sentences. Some listeners commented that the speaking rate of some sentences was too high for them to clearly discern the different words in the sentences. Together with the phonetic and prosodic constraints of the synthesis system described earlier, these facts

provide some insights into the cause of the low sentence accuracy obtained.

Using the *HResults* tool from the *HTK* toolset developed at the Cambridge University Engineering Department (http://htk.eng.cam.ac.uk), a dynamic programming (DP) optimal alignment between the original and transcribed sentences at word-level was performed. Table 6.4 shows that the overall word-level accuracy is 41.25%. Also shown is the word-level accuracy for the sentences when the article "the" was not included in the scoring procedure. The decreased accuracy when doing so shows that this word is more easily identifiable than the other words occurring in the sentences. By similar measurement of the (DP-aligned) phonetic transcriptions of the sentences as well as the listeners' transcriptions, we find that the overall phonetic accuracy[3] is higher than the word-level accuracy. This indicates that many of the incorrect word transcriptions were phonetically similar to the actual words.

A comparative French SUS test evaluation of natural speech, 3 diphone-based synthesisers and 3 unit selection synthesisers is presented in [7]. The results indicate word accuracies[4] of between 60% and 75% and phone accuracies of between 70% and 85% for the synthesizers, whereas natural speech yielded the best results (both word and phone accuracy of approximately 90%).

Table 6.5 shows the word-level and phonetic accuracies for each of the sentences individually. It is shown here because some listeners commented that the longer sentences (syntactic structures 4 and 5) proved more difficult to remember during transcription. Apart from sentence 4–3, the results seem to support this. When omitting these two structures, the word-level accuracy increases to 48.77% (35.59% without articles) and the phonetic accuracy to 55.34%. Table 6.5 also shows that the results vary substantially between the different sentences. The low accuracies obtained for some of the sentences are possibly the result of the artifacts introduced by imposing the magnitude envelope of the original recordings being more pronounced in these sentences than in others.

[3] "Phonetic accuracy" refers to the number of phonetic units (such as phones or diphones) classified correctly.

[4] In [7], word-level accuracies were obtained by measuring the total number of correct *phonetic transcriptions* of words.

Chapter 7

Summary and Conclusion

7.1 Project Summary

A speech generation system for use within a TTS system was developed in this thesis. The system uses parametric models of monophone units for synthesis in the form of LPC's and excitation signal parameters. A number of advantages can be associated with the parametric and modular implementation of the system, including the simple extension to new languages as well as the system's potential portability to different devices. The developed system is also well suited as a vehicle for further research.

A novel approach to the estimation of the excitation signal parameters (section 3.3.4) was developed. This method was shown to be more computationally efficient than established methods for estimating the Gaussianity of LP residual signals. Furthermore, the harmonic cutoff frequency estimation algorithm for LP residuals was shown to be suitable and robust for most sound classes using any of these Gaussianity measures (Appendix B). A new approach to the interpolation of monophone parameter vectors was presented in chapter 4. Using this method, smoothed parameter transitions not unlike those associated with natural speech can be generated to avoid the necessity of a speech corpus containing diphones or larger units. Time waveforms and spectrograms of the synthetic speech indicate the similarities between the generated speech and recorded speech.

The speech synthesis system was implemented (chapter 5) and the intelligibility of its speech output evaluated (chapter 6). The speech generation system software was implemented in a modular fashion with the aim of future expansion by others explicitly in mind. The following sections describe some directions such further work might take.

7.2 Recommendations for Future Work

7.2.1 Text preprocessing

The developed system represents only the speech generation section of a full TTS system. A full TTS system requires an additional text preprocessor which translates bodies of text to

phonetic and prosodic representations, which can be used as input to the speech generation system. A markup standard such as SABLE [39], could be used for such a representation. Without such a preprocessor, the synthetic speech output is limited in terms of its intelligibility and naturalness. In addition, without the use of a text preprocessor, phonetic and prosodic elements must be defined manually, a very slow and laborious process which limits the amount of speech data that can be generated for evaluation (see chapter 6).

7.2.2 Multi-linguality

Although the system was designed to be suitable for use in different languages, only South African English was implemented and evaluated. Some Afrikaans utterances were informally generated, but no proper evaluation was performed. It therefore remains to be seen whether or not the speech generation algorithm is suitable for multi-lingual speech synthesis. As described in chapter 5, little or no modification to the system is necessary for adding a new language. The only necessary step involves the recording of a monophone speech database for the language.

7.2.3 Portability

The data-independent nature of the speech generation system makes it potentially suitable for portability to systems with limited memory and/or computational power, such as personal organisers or cellular telephones. A hardware implementation may also be possible, since many microprocessors support development in the *C* language.

The interpolation process currently has the highest memory and computational requirements of all tasks performed during synthesis. This is because an LSF vector is generated for each output sample. It may be possible to spread interpolation points apart such that each speech parameter vector remains constant for longer than a sampling period without any significant impact on the system's output quality. Also, the current implementation completes the interpolation of an entire utterance before performing LPC synthesis. This results in substantial memory requirements which can be avoided by performing interpolation and synthesis simultaneously or at shorter intervals.

7.2.4 Polyglot synthesis

The term "polyglot" synthesis refers to speech synthesis that can be performed using a variety of synthetic voices. The parametric nature of the speech models used in this project makes it a suitable candidate for polyglot speech synthesis, since speech characteristics such as speaking rate, pitch, loudness, breathiness and others can be easily manipulated using the phone model parameters. Further research may, however, be required in order to determine the relationship between the model parameters and general speech characteristics.

7.2.5 Vocal tract models

The monophone vocal tract models are simple LPC vectors. Several techniques for potentially improving the quality of the vocal tract models, such as frequency warping and lip radiation characteristics [1], exist and should be implemented and their effects on the quality of the speech output evaluated. As stated in chapter 5, the system's speech modelling module is interchangeable, which means that improvements at this level do not require changes to other system modules. Other possibilities for estimating vocal tract filter effects include cepstral and ARX ([31], [50]) coefficients. Using these, however, will most probably necessitate changes to the current (LSF) interpolation algorithm.

7.2.6 Excitation signal models

Although several excitation signal models were defined in chapter 3, only two of these (impulse/noise and sinusoids/noise) were implemented and only the sinusoidal model was evaluated. Once a text preprocessor has been added to the system, the impact of the particular excitation signal model may be evaluated more readily. Future work should include the development, implementation and evaluation of new excitation signal models to determine which produces the most natural speech output. A new excitation signal model should include stochastic aspects more like natural speech such as jitter and shimmer [22], [29]. Also, the effect of highpass filtering to model the noise frequency envelope should be explored (see section 3.2.1).

One could use a parametric function, such as the exponential function used to approximate the spectral Gaussianity curves in Appendix B to model the harmonic frequency envelope. This is in contrast to the current method which assumes a linear decay of the harmonic amplitudes with frequency up to the estimated parameter F_{max}.

7.2.7 Interpolation

Although the LSF interpolation scheme presented in section 4.1.2 produces inter-phone transitions similar to those associated with natural speech, it suppresses co-articulation effects. The impact that a particular phone's context has on its pronunciation should be explored further, the modelling of which could result in necessary modifications to this aspect of the system.

Excitation parameter interpolation is performed separately to LSF interpolation for the reasons stated in section 4.2. The current interpolation algorithm used for the excitation signal parameter transitions has not yet been evaluated. Further research may be required in order to find a more suitable interpolation algorithm, which should include a comparative study between different transition modelling schemes.

7.2.8 Modelling of particular sound classes

As was mentioned in section 3.4, the modelling of voiced plosive sounds was particularly problematic. The MRT results presented in section 6.2.3 (see table 6.2) illustrate the negative impact these sounds had on the intelligibility of the system's output speech. These issues need to be addressed and more effective modelling techniques for the different sound classes developed before the output speech quality for these sounds can be considered intelligible and natural. Additional modelling may become necessary for specific sound classes when the system is adapted to incorporate a particular language containing new sound classes that cannot be adequately modelled using the same techniques associated with the currently supported sound classes.

7.3 Conclusion

The design and implementation of the speech generation system developed in this thesis was aimed at the rapid deployment of TTS to new and under-resourced languages. Although time did not allow for formal testing in multiple languages, informal tests suggest that the system may be well suited for this purpose. Compared to concatenative synthesisers, which are the current commercial TTS standard, the amount of transcribed speech data required for a new target language is minimal. An added associated advantage is that the system has a very small memory footprint and fairly low computational requirements.

Even without a text preprocessing front-end, results of the evaluation of the system in South African English (chapter 6) show that the synthetic speech generated by this system is moderately intelligible. Since the system's implementation encompasses only the speech generation section of a full TTS synthesis system, these results cannot be used for direct comparison with other synthesisers before the necessary modules of a full TTS system are added.

The system's modular implementation makes it a useful platform for the development of different speech modelling techniques. The effect on speech intelligibility of any future changes to the system may be evaluated using the GUI shown in Appendix D. The evaluation is therefore repeatable and the current results can be used as a benchmark when evaluating future improvements to the system, such as those listed in sections 7.2.1–7.2.8.

Bibliography

[1] ALLEN, J., HUNNICUTT, M. S., and KLATT, D., *From Text to Speech: The MITalk System*. Cambridge University Press, 1987.

[2] BAILLY, G., "Accurate Estimation of Sinusoidal Parameters in an Harmonic+Noise Model for Speech Synthesis." in *Proceedings of EuroSpeech 1999*.

[3] BENOÎT, C., GRICE, M., and HAZAN, V., "The SUS test: A Method for the Assessment of Text-to-Speech Synthesis Intelligibility Using Semantically Unpredictable Sentences." *Speech Communication*, 1996, Vol. 18, No. 4, No. 4, p. 381.

[4] BLACK, A. W. and LENZO, K. A., "Multilingual Text-to-Speech Synthesis." in *Proceedings of International Conference on Acoustics, Speech and Signal Processing 2004*.

[5] CHAPPELL, D. and HANSEN, J., "Spectral Smoothing for Concatenative Speech Synthesis." in *Proceedings of International Conference on Spoken Language Processing 1998*.

[6] CHILDERS, D. G. and HU, T., "Speech Synthesis by Glottal Excited Linear Prediction." *Journal of the Acoustical Society of America*, October 1994, Vol. 96, No. 4, pp. 2026–2036.

[7] DE MAREÜIL, P. B., D'ALESSANDRO, C., RAAKE, A., BAILLY, G., GARCIA, M.-N., and MOREL, M., "A Joint Intelligibility Evaluation of French Text-to-Speech Synthesis Systems: the EvaSy SUS/ACR Campaign." in *Proceedings of Language Resources and Evaluation Conference*, 2006.

[8] DONOVAN, R. E., *Trainable Speech Synthesis*. PhD thesis, Cambridge University, 1996.

[9] EDGINGTON, M., "Investigating the Limitations of Concatenative Synthesis." in *Proceedings of EuroSpeech 1997*.

[10] EDGINGTON, M. and LOWRY, A., "Residual-Based Speech Modification Algorithms for Text-to-Speech Synthesis." in *Proceedings of International Conference on Spoken Language Processing 1996*.

[11] ESPOSITO, R. and YANG, L.-C., "Levels of Prosodic Representation in Spoken Discourse: An Empirical Approach." in *Proceedings of Eurospeech 1999*.

[12] FAIRBANKS, G., "Test of Phonemic Differerentiation: The Rhyme Test." *Journal of the Acoustical Society of America*, 1958, Vol. 30, pp. 596–600.

[13] FERENCZ, A., NAGY, I., KOVÁCS, T.-C., RAŢIU, T., and FERENCZ, M., "On a Hybrid Time Domain-LPC Technique for Prosody Superimposing Used for Speech Synthesis." in *Proceedings of EuroSpeech 1999*.

[14] GUTTIÉRREZ ARRIOLA, J., GIMÉNEZ DE LOS GALANES, F., SAVOJI, M., and PARDO, J., "Speech Synthesis and Prosody Modification Using Segmentation and Modeling of the Excitation Signal." in *Proceedings of Eurospeech 1997*.

[15] HÖGBERG, J., "Data Driven Formant Synthesis." in *Proceedings of EuroSpeech 1997*.

[16] HOUSE, A. S., WILLIAMS, C., HECKER, M. H. L., and KRYTER, K. D., "Psychoacoustic Speech Tests: A Modified Rhyme Test." *Journal of the Acoustical Society of America*, 1963, Vol. 35, p. 1899.

[17] HYVÄRINEN, A., KARHUNEN, J., and OJA, E., *Independent Component Analysis*. New York: John Wiley & sons, Inc., 2001.

[18] ITAKURA, F., "Line Spectrum Representation of Linear Predictive Coefficients of Speech Signals." *Journal of the Acoustical Society of America*, 1975, Vol. 57, p. S35.

[19] KARJALAINEN, M. and PAATERO, T., "Generalized Source-Filter Structures for Speech Synthesis." in *Proceedings of EuroSpeech 2001*.

[20] KAY, S. M., *Modern Spectral Estimation: Theory & Application*. Englewood Cliffs, New Jersey 07632: Prentice Hall, 1988.

[21] KLATT, D. H. and KLATT, L. C., "Analysis, Synthesis and Perception for Voice Quality Variations Among Female and Male Talkers." *Journal of the Acoustical Society of America*, 1990, Vol. 87, pp. 820–857.

[22] KLATT, D. H., "Review of Text-to-Speech Conversion for English." *Journal of the Acoustical Society of America*, 1987, Vol. 82, pp. 737–793.

[23] KLEIJN, W. B. and PALIWAL, K. K. (Eds), *Speech Coding and Synthesis*. Amsterdam: Elsevier Science, 1998.

[24] KRITZINGER, C., "Low Bit Rate Speech Coding." Master's thesis, University of Stellenbosch, 2006.

[25] LEMMETTY, S., "Review of Speech Synthesis Technology." Master's thesis, Helsinki University of Technology, 1999.

[26] LOGAN, J. S., GREENE, B. G., and PISONI, D. B., "Segmental Intelligibility of Synthetic Speech Produced by Rule." *Journal of the Acoustical Society of America*, August 1989, Vol. 86, No. 2, pp. 566–581.

[27] LOPÉZ-GONZALO, E., RODRÍGUEZ-GARCÍA, J. M., HERNÁNDEZ-GÓMEZ, L., and VILLAR, J. M., "Automatic Prosodic Modelling for Speaker and Task Adaptation in Text-to-Speech." in *Proceedings of International Conference on Acoustics, Speech and Signal Processing 1997*.

[28] MASUKO, T., TOKUDA, K., KOBAYASHI, T., and IMAI, S., "Speech Synthesis using HMM's with Dynamic Features." in *Proceedings of International Conference on Spoken Language Processing 1996*.

[29] MILENKOVIC, P. H., "Voice Source Model for Continuous Control of Pitch Period." *Journal of the Acoustical Society of America*, February 1993, Vol. 93, No. 2, pp. 1087–1096.

[30] MURPHY, P. J., "Spectral Tilt as a Perturbation-free Measurement of Noise Levels in Voice Signals." in *Proceedings of Eurospeech 2001*.

[31] OHTSUKA, T. and KASUYA, H., "Aperiodicity Control in ARX-Based Speech Analysis-Synthesis Method." in *Proceedings of Eurospeech 2001*.

[32] PEEBLES, P. Z., *Probability, Random Variables and Random Signal Principles*. Fourth edition. 1221 Avenue of the Americas, New York, NY, 10020: Irwin/McGraw-Hill, 2001.

[33] PFITZINGER, H. R., "DFW-based Spectral Smoothing for Concatenative Speech Synthesis." in *Proceedings of International Conference on Spoken Language Processing 2004*.

[34] POTAMIANOS, A. and MARAGOS, P., "Speech Analysis and Synthesis using an AM-FM Modulation Model." in *Proceedings of Eurospeech 1997*.

[35] PROAKIS, J. G. and MANOLAKIS, D. G., *Digital Signal Processing: Principles, Algorithms, and Applications*, pp. 23–33. Upper Saddle River, New Jersey 07458: Prentice Hall, Third edition, 1996.

[36] RABINER, L. and BIING-HWANG, J., *Fundamentals of Speech Recognition*. Englewood Cliffs, New Jersey 07632: PTR Prentice Hall, 1993.

[37] RANK, E., "Exploiting Improved Parameter Smoothing within a Hybrid Concatenative/LPC Speech Synthesizer." in *Proceedings of Eurospeech 1999*.

[38] SHANNON, C. E., "A Mathematical Theory of Communication." *The Bell System Technical Journal*, 1948, Vol. 27, pp. 379–423, 623–656.

[39] SPROAT, R., HUNT, A., OSTENDORF, M., TAYLOR, P., BLACK, A., LENZO, K., and EDGINGTON, M., "SABLE: A Standard for TTS Markup." in *Proceedings of International Conference on Spoken Language Processing 1998*.

[40] STEVENS, C., LEES, N., VONWILLER, J., and BURNHAM, D., "On-line Experimental Methods to Evaluate Text-to-Speech (TTS) Synthesis: Effects of Voice Gender and Signal Quality on Intelligibility, Naturalness and Preference." *Computer Speech and Language*, 2005, Vol. 19, p. 129.

[41] STRUBE, H. W., "Linear Prediction on a Warped Frequency Scale." *Journal of the Acoustical Society of America*, 1980, Vol. 68, No. 4, No. 4, p. 1071.

[42] STYLIANOU, Y., DUTOIT, T., and SCHROETER, J., "Diphone Concatenation using a Harmonic plus Noise Model of Speech." in *Proceedings of EuroSpeech 1997*.

[43] SYRDAL, A. K., "Phonetic Effects on Listener Detection of Vowel Concatenation." in *Proceedings of EuroSpeech 2001*.

[44] TATHAM, M., LEWIS, E., and MORTON, K., "An Advanced Intonation Model for Synthesis." in *Proceedings of EuroSpeech 1999*.

[45] TEIXEIRA DE JESUS, L. M. and CAWLEY, G. C., "Speech Coding and Synthesis using Parametric Curves." in *Proceedings of EuroSpeech 1997*.

[46] TOKUDA, K., MASUKO, T., YAMADA, T., KOBAYASHI, T., and IMAI, S., "An Algorithm for Speech Parameter Generation from Continuous Mixture HMM's with Dynamic Features." in *Proceedings of EuroSpeech 1995*.

[47] VAN SANTEN, J. P. H., POLS, L. C. W., ABE, M., KAHN, D., KELLER, E., and VONWILLER, J., "Report on the Third ESCA TTS Workshop Evaluation Procedure." in 3^{rd} *Workshop on Speech Synthesis*, 1998.

[48] VISAGIE, A. S., "Speech Generation in a Spoken Dialogue System." Master's thesis, University of Stellenbosch, 2004.

[49] YOSHIMURA, T., TOKUDA, K., MASUKO, T., KOBAYASHI, T., and KITAMURA, T., "Mixed Excitation for HMM-based Speech Synthesis." in *Proceedings of EuroSpeech 2001*.

[50] ZHU, W. and KASUYA, H., "A New Speech Synthesis System Based on the ARX Speech Production Model." in *Proceedings of International Conference on Spoken Language Processing 1996*.

[51] ZILL, D. G. and CULLEN, M. R., *Advanced Engineering Mathematics*. Second edition. 40 Tall Pine Drive, Sudbury, MA 01776: Jones and Bartlett Publishers, 2000.

Appendix A

African Speech Technology Phones Used

DESCRIPTION	ASTBET	English
Stops		
Voiceless Bilabial Plosive	p	pit
Voiced Bilabial Plosive	b	baby
Voiceless Alveolar Plosive	t	total
Voiced Alveolar Plosive	d	death
Voiceless Velar Plosive	k	kick
Voiced Velar Plosive	g	gun
Fricatives		
Voiceless Labiodental Fricative	f	four
Voiced Labiodental Fricative	v	vat
Voiceless Dental Fricative	th	thing
Voiced Dental Fricative	dh	this
Voiceless Alveolar Fricative	s	some
Voiced Alveolar Fricative	z	zero
Voiceless Post-Alveolar Fricative	sh	shine
Voiced Post-Alveolar Fricative	zh	genre
Voiceless Velar Fricative	x	*Gauteng*
Voiceless Glottal Fricative	h	hand
Voiced Glottal Fricative	hht	*Johannes*
Approximants		
Alveolar Approximant	rt	red
Alveolar Lateral Approximant	l	legs
Palatal Approximant	j	yes
Voiced labio-velar Approximant	w	west

Nasals		
Bilabial Nasal	`m`	man
Alveolar Nasal	`n`	not
Velar Nasal	`nj`	thing
Vowels		
High Front Vowel	`i`	*Piet*
High Front Vowel with duration	`i_long`	keep
Lax Front Vowel	`ic`	him
High Back Vowel	`u`	*Kapkaroord*
High Back Vowel with duration	`u_long`	blue
Lax Back Vowel	`hs`	push
Mid-high Front Vowel with duration	`e_long`	*Vrede*
Rounded Mid-high Back Vowel	`o_long`	*Sibongile*
Mid-low Front Vowel	`ep`	nest
Mid-low Front Vowel with duration	`ep_long`	fairy
Rounded Mid-low Front Vowel	`oe`	nurse
Rounded Mid-low Front Vowel with duration	`oe_long`	burst
Central Vowel with duration	`epr_long`	turn
Rounded Mid-low Back Vowel	`ct`	*Hartenbos*
Rounded Mid-low Back Vowel with duration	`ct_long`	bore
Low Back Vowel	`ab`	hot
Lax Mid-low Vowel	`vt`	hut
Low Central Vowel	`a`	*Garsfontein*
Low Central Vowel with duration	`a_long`	*Klerksdorp*
Low Back Vowel with duration	`as_long`	harp
Central Vowel (Schwa)	`sw`	the
Mid-low Front Vowel	`ae`	average
Mid-low Front Vowel with duration	`ae_long`	dad

Table A.1: *The AST phones used for the synthesis phone dictionary.*

Appendix B

Estimating the Maximum Voicing Frequency

In section 3.3.4, we introduced a method for estimating the maximum voicing frequency, or harmonic cutoff frequency, F_{max} from the frame-by-frame Gaussianity measures m_{f_ε}, kurt$(\tilde{\varepsilon})$ and $H(\hat{\varepsilon})$ calculated from the FFT spectrum. This chapter details the algorithm which was used to fit an exponential function to these curves in order to estimate F_{max}. Because m_{f_ε} was chosen for the development of the project, examples in this appendix make use of this measure only. However, experiments involving kurt$(\tilde{\varepsilon})$ and $H(\hat{\varepsilon})$ have indicated that the procedure described here is equally suitable for these measures.
Figures B.1, B.2, B.3 and B.4 show the curves obtained by calculating the Gaussianity measure m_{f_ε} on a frame-by-frame basis from the LP residual spectra for one example of each of the different (voiced) sound classes considered[1]. Note that the algorithm provides adequate estimates of F_{max}, which is the frequency at which the magnitudes of the harmonics have decayed and are no longer discernible. Also note that $G(x)$ is not a smooth curve in any of the figures and that there is often some degree of variability in $G(x)$ around F_{max}. A simple approach would be to find F_{max} by setting a threshold for the the Gaussianity beyond which we can assume that the energy associated by the harmonics is negligible. However, since m_{f_ε} is not guaranteed to decrease monotonically over frequency, finding a single value for F_{max} in this way may not always be possible. Instead we approximate the actual curve by a theoretical curve which decreases monotonically over frequency and apply a threshold to this. We cannot use mean squared (or other similar distance measure) error minimisation algorithm for this purpose because the variations in the curve values around F_{max} may introduce a strong bias in the mean value of the error. Let us therefore attempt to find a more appropriate solution from a more theoretical approach.

From this point, we will refer to the data (the residual spectrum Gaussianity curve) as $G(x)$, where x represents the frequency (not necessarily in Hz) and G can be m_{f_ε}, $\sqrt[4]{\text{kurt}(\tilde{\varepsilon})}$

[1]This does not include plosive sounds, as Gaussianity could not be reliably estimated for these sounds (see section 3.3).

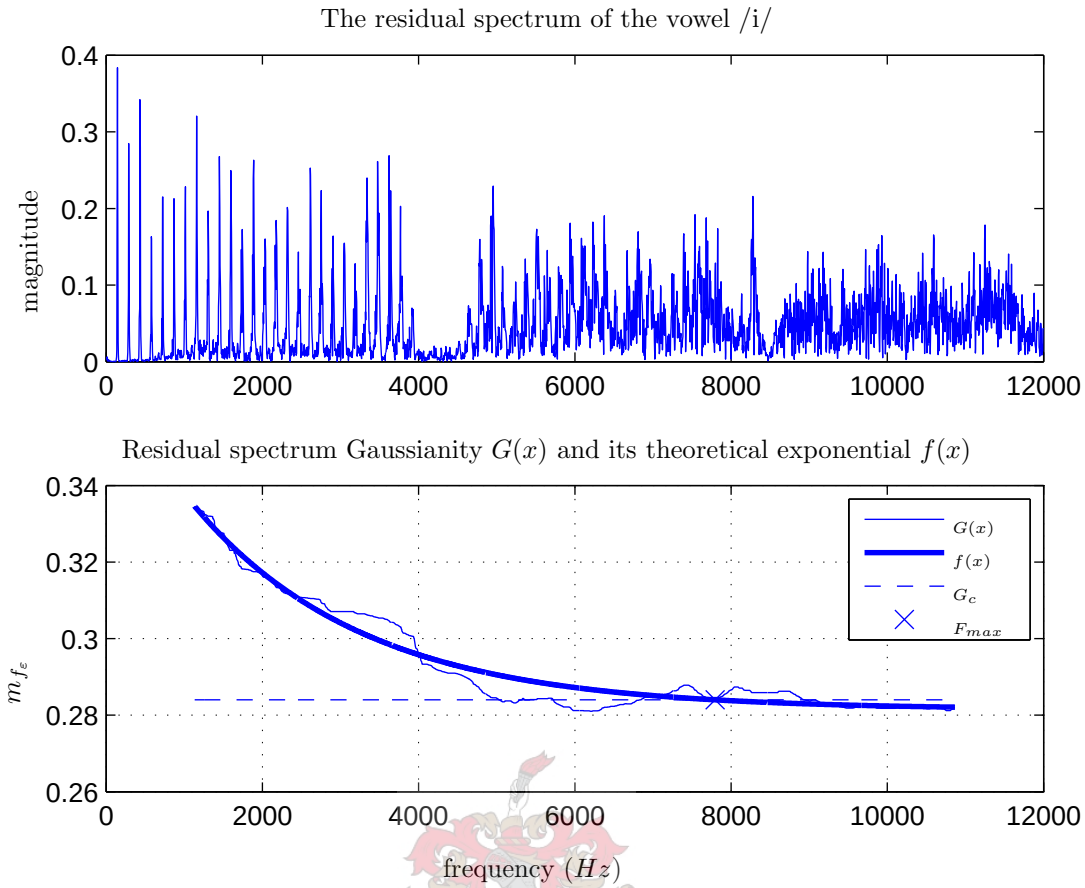


Figure B.1: *Finding F_{max} for the vowel /i/.*

or $H_D(\hat{\varepsilon})$. From the observed general shape of the curves, we consider the following exponential function as a possible approximation to $G(x)$:

$$y = f(x) = be^{ax} + c \tag{B.1}$$

where x is the independent variable (frequency, in our case), $y = f(x)$ is the dependent variable (the theoretical approximate of $G(x)$) and a, b and c determine the shape of y. The variables a, b and c represent three degrees of freedom, which means that we need to find at least three independent equations in a, b and c before $f(x)$ is uniquely determined. We begin solving for a in equation B.1, which yields the general solution:

$$a = \frac{1}{x} \log \frac{y - c}{b} \tag{B.2}$$

Now let us require that $f(x)$ and $G(x)$ should intersect at $(x_L, G(x_L) = f(x_L))$. This point should be chosen such that the intersection is at a suitable location for fitting $f(x)$ to $G(x)$. Experimentally, choosing x_L to represent a frequency point in the range $1kHz \leq 2kHz$ was found to yield good results on average over all sound classes. This is because $F_{max} > 2kHz$ in most cases, which means that the variability in $G(x)$ does not play a significant role at

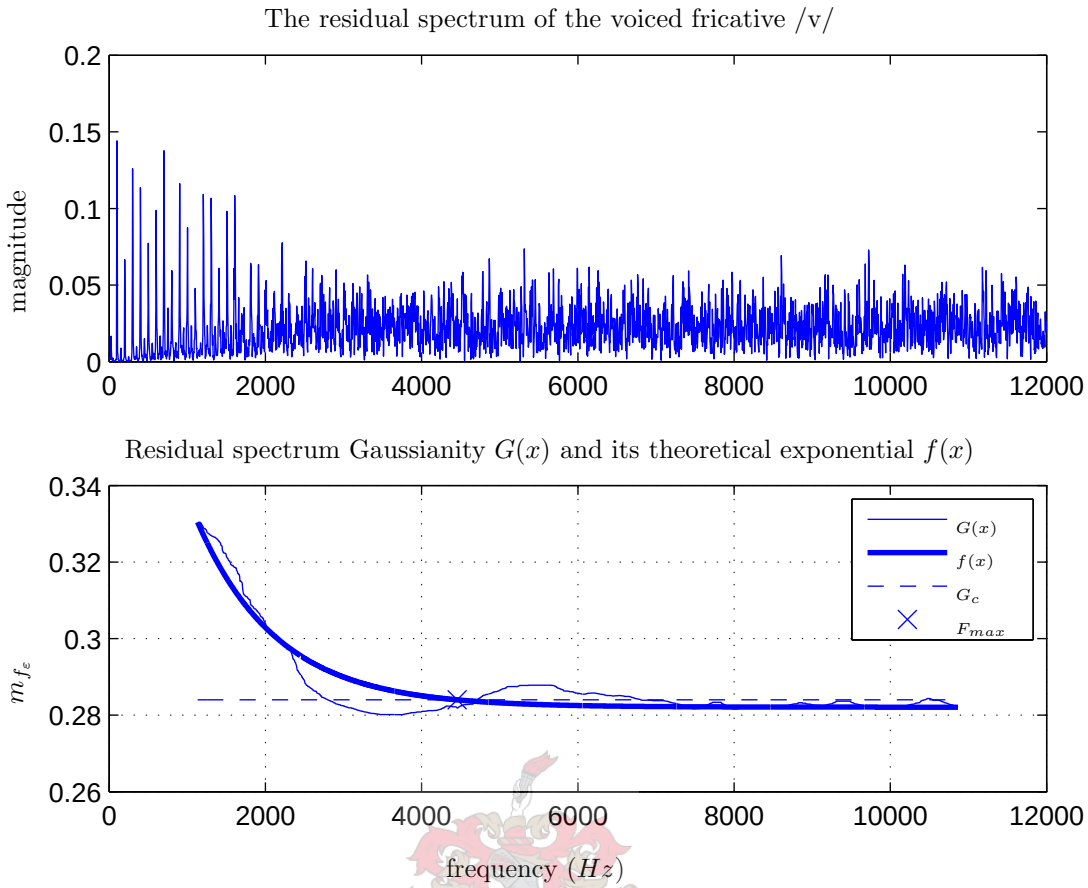


Figure B.2: *Finding F_{max} for the voiced fricative /v/.*

x_L. Equation B.2 then becomes:

$$a = \frac{1}{x_L} \log \frac{G(x_L) - c}{b} \tag{B.3}$$

To find b, we can solve equation B.1 using the value of $G(x)$ at any two further frequencies. Care should be taken, however, as the particular frequency points chosen will determine the value of the parameter b, which is the magnitude scaling constant for $f(x)$. We have chosen these two frequency points as x_0 and x_N, such that f should intersect G at its initial and final values (within the bounds we are considering, which is the range $0Hz \leq x \leq \frac{F_s}{2}$). Figure B.5 illustrates the locations of these points. We then substitute $f(x_0) = y_0 = G(x_0)$ and $f(x_N) = y_N = G(x_N)$ into equation B.1 and solve for x_0 and x_N, respectively, which yields:

$$x_0 = \frac{1}{a} \log \frac{G(x_0) - c}{b} \tag{B.4}$$

$$x_N = \frac{1}{a} \log \frac{G(x_N) - c}{b} \tag{B.5}$$

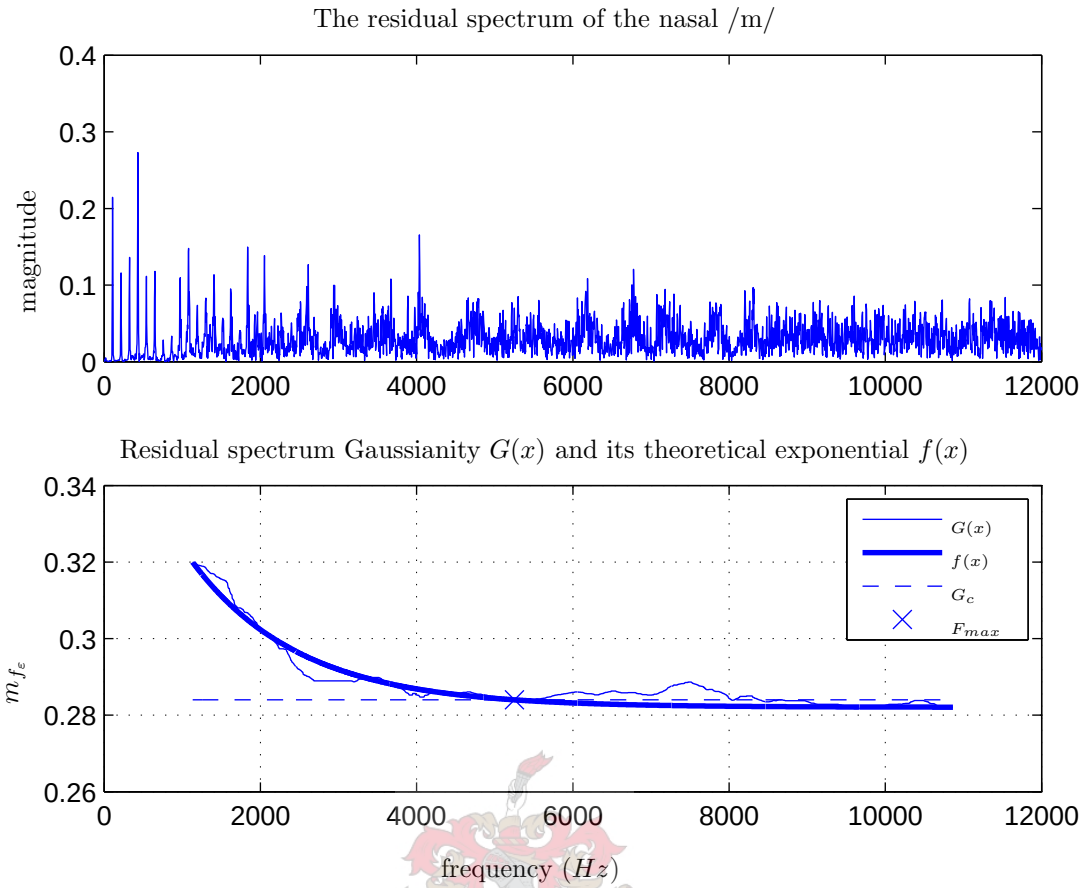


Figure B.3: *Finding F_{max} for the nasal /m/.*

Dividing equation B.4 by equation B.5, we find:

$$\begin{aligned}
\frac{x_0}{x_N} &= \frac{\log \frac{G(x_0)-c}{b}}{\log \frac{G(x_N)-c}{b}} \\
\frac{x_0}{x_N} \log \frac{G(x_N)-c}{b} &= \log \frac{G(x_0)-c}{b} \\
\log \left(\frac{G(x_N)-c}{b} \right)^{\frac{x_0}{x_N}} &= \log \frac{G(x_0)-c}{b} \\
\left(\frac{G(x_N)-c}{b} \right)^{\frac{x_0}{x_N}} &= \frac{G(x_0)-c}{b} \\
b &= \left(\frac{(G(x_0)-c)^{x_N}}{(G(x_N)-c)^{x_0}} \right)^{\frac{1}{x_N-x_0}} \qquad \text{(B.6)}
\end{aligned}$$

We can calculate c by first substituting the points $(x_0, G(x_0))$ and $(x_N, G(x_N))$ into equation B.1 and solving for $G(x_0)$ and $G(x_N)$, respectively, which yields:

$$G(x_0) = be^{ax_0} + c \qquad \text{(B.7)}$$

$$G(x_N) = be^{ax_N} + c \qquad \text{(B.8)}$$

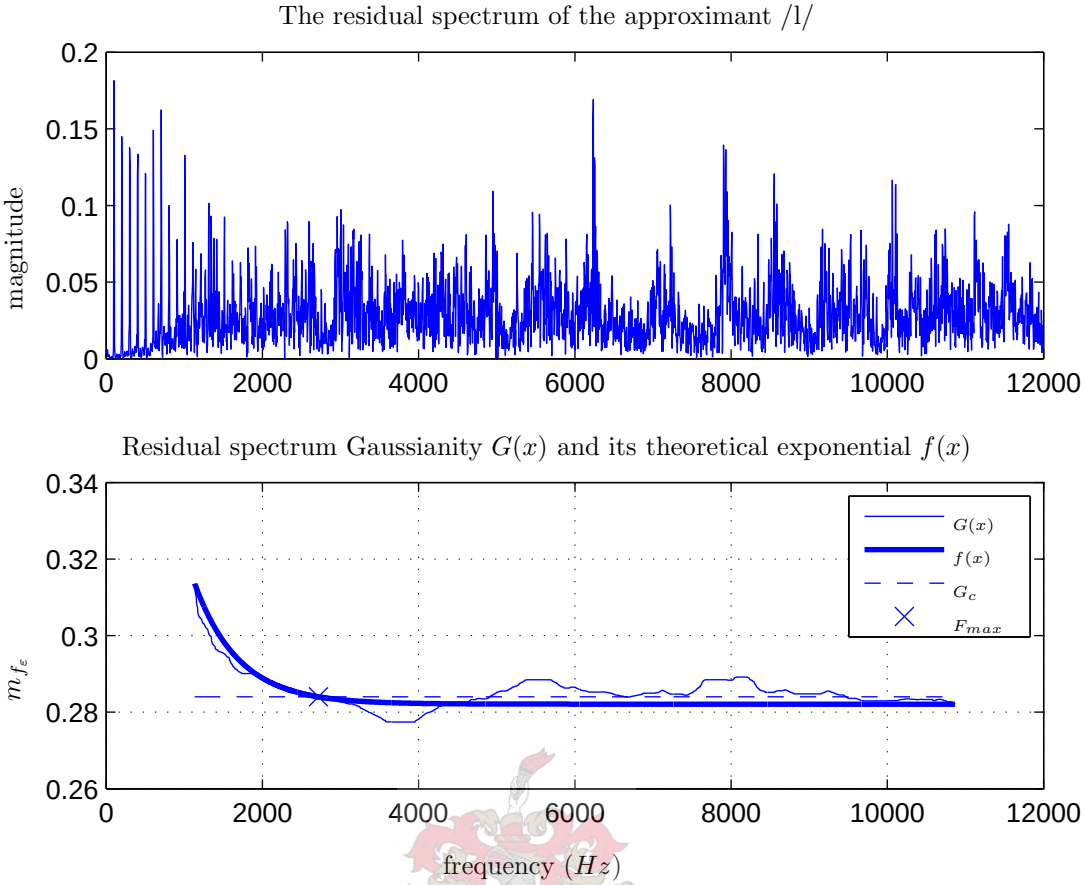


Figure B.4: *Finding F_{max} for the approximant /l/.*

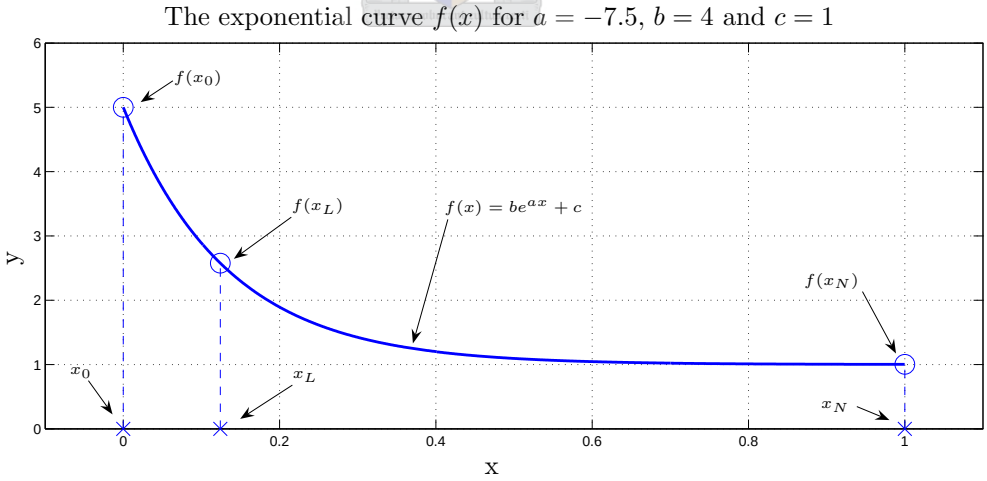


Figure B.5: *The exponential function for approximating spectral Gaussianity curves.*

and then adding the results of equations B.7 and B.8 and solving for c, which results in:

$$c = \frac{1}{2}\left(G(x_0) + G(x_N) + b(e^{ax_0} + e^{ax_N})\right) \tag{B.9}$$

Although we now have three equations (in B.3, B.6 and B.9) to solve for a, b and c, the non-linear nature of these equations prevents us from doing so by simple arithmetic. Instead, we choose initial values for the following parameters:

- $x_0 = 0$, $x_N = 1$ and $\frac{1}{12} \leq x_L \leq \frac{1}{6}$. This is possible since the (linear) frequency variable x is arbitrary, as long as it represents the entire domain of G. These choices also simplify equation B.6 to $b = G(x_0) - c$.
- $c = G(x_N)$. This is suitable because the slope of $G(x)$ tends to zero at high frequencies and $f(x) \to c$ for large values of x[2].

and iterate over the following steps until the parameters converge to some set of values:

- calculate b from equation B.6,
- calculate a from equation B.3, and
- find a new estimate of c using equation B.9.

Experimentally, it was found that the parameters typically converge sufficiently after 10–20 iterations. The above algorithm was tested by supplying it with a theoretical $G(x)$ in the form of equation B.1, and it successfully converged to the true values of a, b and c in each case where $a < 0$. The number of iterations in each case is determined by the accuracy of the initial estimate $c = G(x_N)$.

Now that we have an algorithm for calculating the variables a, b and c, we can uniquely specify an $f(x)$ for any $G(x)$ we encounter. In $f(x)$, we have an estimate of $G(x)$ which increases/decreases monotonically with frequency. This implies that we are now able to set a magnitude threshold for $G(x)$ beyond which we assume that the harmonic content is zero. This threshold can be chosen to be close to the theoretical value of the particular Gaussianity estimator when applied to Gaussian data (see, for instance, equations 3.6 and 3.11), since the residual spectrum becomes more Gaussian as frequency increases. The value of F_{max} then follows directly from equation B.1 as:

$$F_{max} = \frac{1}{a} \log \frac{G_c - c}{b} \tag{B.10}$$

where G_c is the chosen cutoff value for $G(x)$. Figures B.1–B.4 show that the above method is well suited to the task of finding the harmonic cutoff frequency from LP residual spectra.

[2]This is because all the residual spectra considered strive toward Gaussian noise at high frequencies. This results in curves of $G(x)$ which "flatten" as frequency increases, resulting in $a < 0$ for any approximate $f(x)$.

Appendix C

Test Material

Word-initial consonant					
went	sent	bent	dent	tent	rent
hold	cold	told	fold	sold	gold
kit	bit	fit	hit	wit	sit
must	bust	gust	rust	dust	just
bed	led	fed	red	wed	shed
pin	sin	tin	fin	din	win
not	tot	got	pot	hot	lot
vest	test	rest	best	west	nest
way	may	say	pay	day	gay
pig	big	dig	wig	rig	fig
shop	mop	cop	top	hop	pop
coil	oil	soil	toil	boil	foil
same	name	game	tame	came	fame
peel	reel	feel	eel	keel	heel
hark	dark	mark	bark	park	lark
thaw	law	raw	paw	jaw	saw
pen	hen	men	then	den	ten
heat	neat	feat	seat	meat	beat
dip	sip	hip	tip	lip	rip
hang	sang	bang	rang	fang	gang
took	cook	look	hook	shook	book
fill	kill	will	hill	till	bill
bale	gale	sale	tale	pale	male
wick	sick	kick	lick	pick	tick
fun	sun	bun	gun	run	nun

Table C.1: *The 25 MRT ensembles with variable word-initial consonants.*

Word-final consonant					
pat	pad	pan	path	pack	pass
lane	lay	late	lake	lace	lame
teak	team	teal	teach	tear	tease
din	dill	dim	dig	dip	did
dug	dung	duck	dud	dub	dun
sum	sun	sung	sup	sub	sud
seep	seen	seethe	seek	seem	seed
pig	pill	pin	pip	pit	pick
back	bath	bad	bass	bat	ban
pale	pace	page	pane	pay	pave
cane	case	cape	cake	came	cave
tan	tang	tap	tack	tam	tab
fit	fib	fizz	fill	fig	fin
heave	hear	heat	heal	heap	heath
cup	cut	cud	cuff	cuss	cub
puff	puck	pub	pus	pup	pun
bean	beach	beat	beak	bead	beam
kill	kin	kit	kick	king	kid
mass	math	map	mat	man	mad
ray	raze	rate	rave	rake	race
save	same	sale	sane	sake	safe
sill	sick	sip	sing	sit	sin
peace	peas	peak	peach	peat	peal
bun	bus	but	bug	buck	buff
sag	sat	sass	sack	sad	sap

Table C.2: *The 25 MRT ensembles with variable word-final consonants.*

Syntactic Structure	Code	Sentence
(1)	1–1	The state went at the hot point.
S – V – Adv	1–2	The school stayed for the new tube.
D + N + Vi + Pre + D + J + N	1–3	The time ran with the high head.
(2)	2–1	The poor sense hit the tax.
S – V – DO	2–2	The short field found the step.
D + J + N + Vt + D + N	2–3	The thin job got the voice.
(3)	3–1	Start the trial and the fund.
V – DO	3–2	Call the game and the front.
Vt + D + N + C + D + N	3–3	Live the plant or the test.
(4)	4–1	When does the dog lead the hard set?
QW – V – S – DO	4–2	Why does the sign learn the green bed?
Qa + Va + D + N + Vt + D + J + N	4–3	How does the chance plan the cold roof?
(5)	5–1	The song marked the branch that burned.
S – V – CDO	5–2	The truth helped the leg that failed.
D + N + Vt + D + N + Pror + Vi	5–3	The top drew the pool that died.

(Adv = adverbial, C = conjunction, CDO = complex direct object, D = determiner, DO = direct object, J = adjective, N = noun, Pre = preposition, Pror = relative pronoun, Qa = question adverbial, QW = question word, S = subject, V = verb, Va = auxiliary verb, Vi = intransitive verb, Vt = transitive verb)

Table C.3: *The SUS 5 semantic structures and 15 sentences used for testing.*

Appendix D

Graphical User Interface for Intelligibility Tests

A graphical user interface (GUI) was developed in the *Python* language using the *PyGTK* module (see `www.pygtk.org`). The GUI automates the two speech intelligibility tests (MRT and SUS) which are outlined in sections 6.1.1 and 6.1.2 in a user-friendly manner. It also simplifies the extraction of results from the listener responses compared to the use of answer sheets. The figures that follow show screenshots of the GUI ordered as they would occur during a typical test.

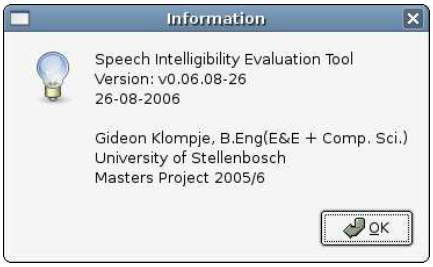


Figure D.1: *The test GUI information window.*

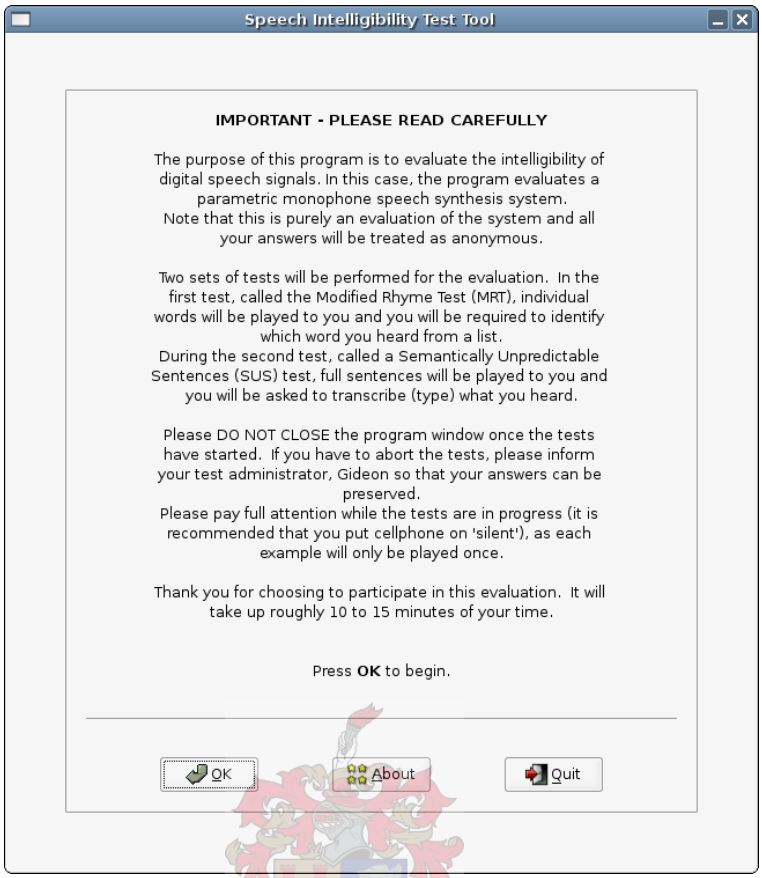


Figure D.2: *The test GUI main window with general instructions.*

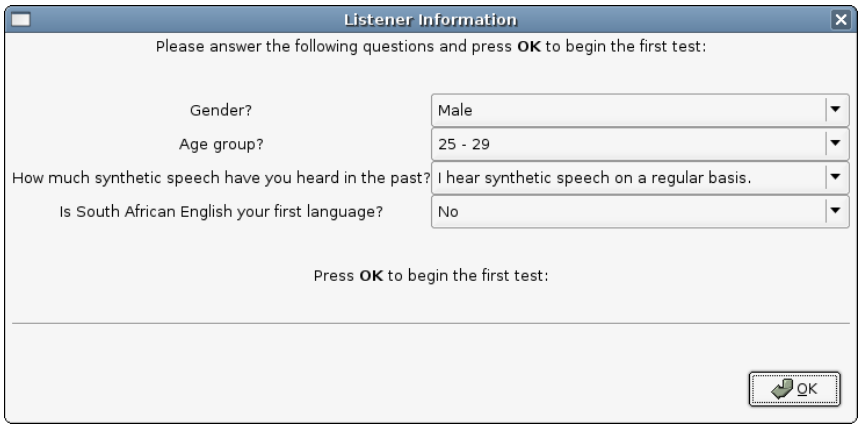


Figure D.3: *The test GUI listener information dialogue.*

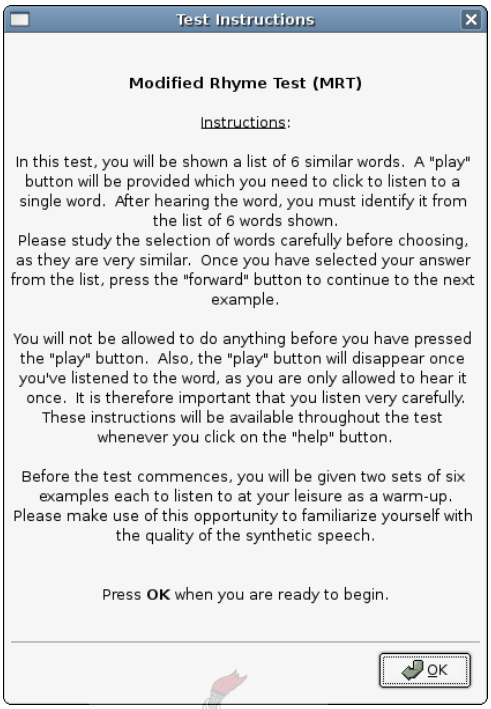


Figure D.4: *The test GUI MRT instructions.*

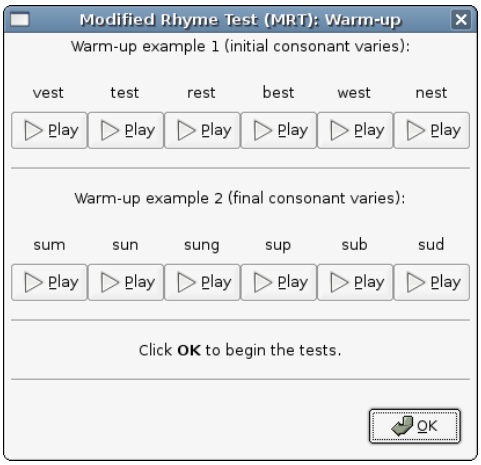


Figure D.5: *The test GUI MRT warm-up dialogue.*

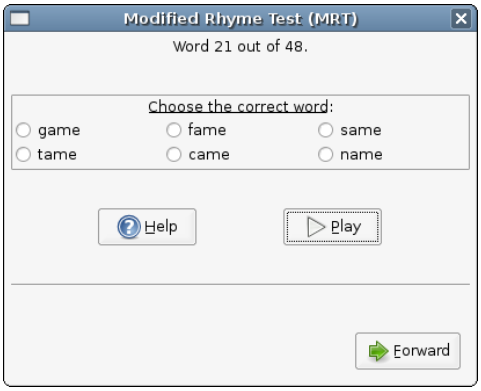


Figure D.6: *The test GUI MRT dialogue.*

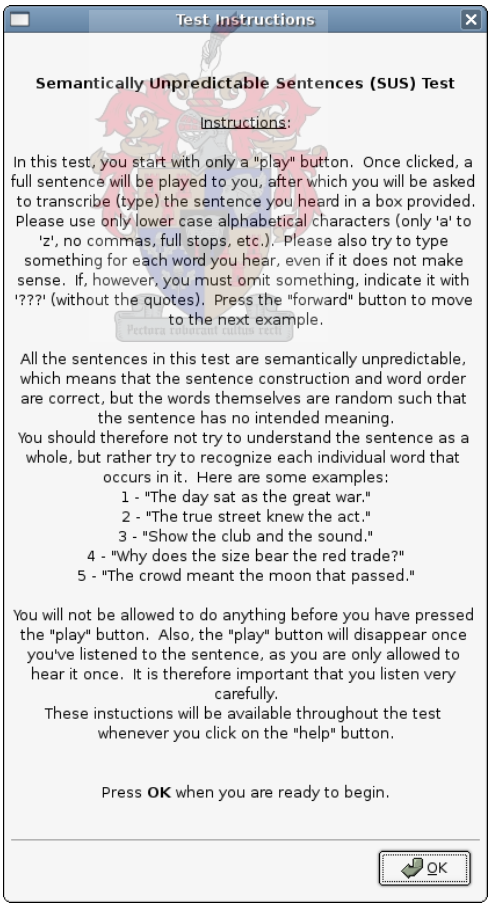


Figure D.7: *The test GUI SUS test instructions.*

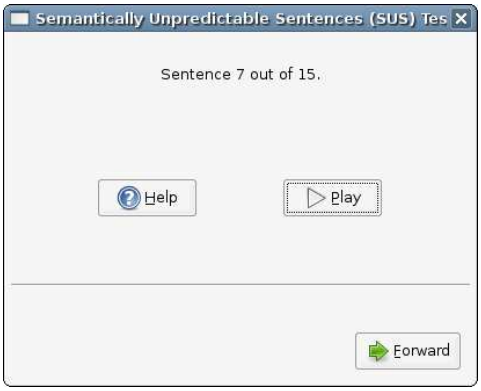


Figure D.8: *The test GUI SUS test dialogue: listening step.*

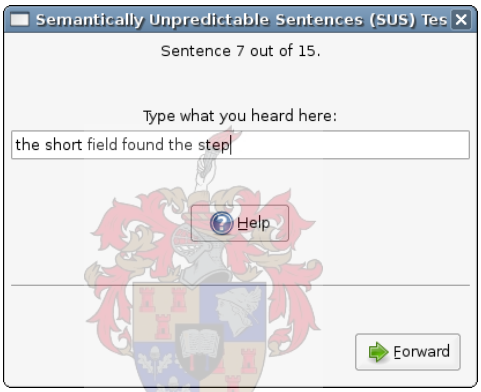


Figure D.9: *The test GUI SUS test dialogue: transcription step.*

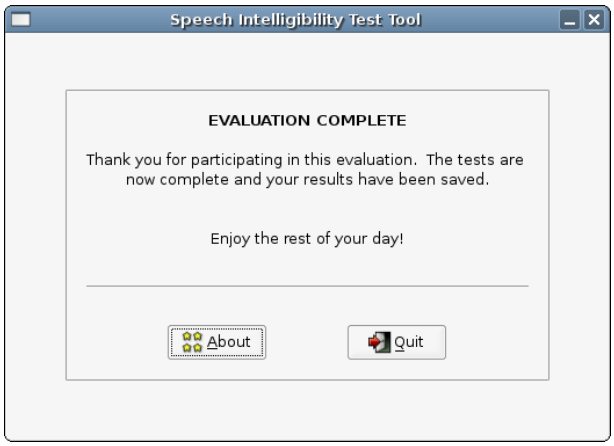


Figure D.10: *The test GUI final message window.*