Computer speech synthesis

— a systematic method to extract synthesis parameters for formant synthesizers

A Thesis

Submitted To

The Department of Computer Science

The Chinese University of Hong Kong

in partial fullfillment of the requirements

For the Degree Of

Master of Philosophy

By

Yu Wai Leung

June, 1993

Acknowledgment

First of all, I am indebted to Dr. K.H. Wong for his uncountable advices and directions for this project. Thanks also to Mr. Leung Kai-Hon for his invaluable assistance in the development of various speech processing modules. Last but not least, I would like to give my special compliment to those who have attended the listening tests.

Table of Contents

Abstract

Formant speech synthesizers are capable of producing high-quality synthetic speech if the synthesis parameters are carefully tuned **[1]**. However, accurate extraction of formant synthesis parameters from natural speech is not easy. To solve this problem, a feedback analysis system, based on the principle of analysis-by-synthesis, is proposed to extract formant amplitude and bandwidth data from natural speech. A simple iterative spectral matching algorithm is employed in this feedback analysis system. Testing results for several vowels have shown that the feedback analysis system can produce synthetic speech with LPC spectral envelopes well matched with that of natural speech samples.

The most influential factor which determines the synthetic speech quality for a formant-based synthesis-by-rule system is the modeling of formant trajectories —— the locus of the formant parameters in the time domain. Models that based on straight-line interpolation often fail to produce formant trajectories which are accurate enough. In contrast, models based on Bézier curve interpolation **[2]** can give more realistic formant trajectories despite of its relatively high computational complexity. In order to compute Bézier curves efficiently, a polynomial interpolation method is devised. In addition, a formant trajectory generation scheme ,which is based on Bézier curve interpolation, is proposed for the synthesis of Cantonese syllables. Preliminary listening test results have shown that some high quality synthetic Cantonese syllables can be produced.

A parallel formant synthesizer is implemented in software. In order to reduce the unwanted fluctuation of synthetic speech waveform due to rapid formant amplitude transitions, the method of linear interpolation of formant amplitude input is proposed. Moreover, special arrangements are introduced in the synthesizer to facilitate systematic spectral matching.

Introduction

Before we want to create speech from electronic machines, we must know the mechanism of speech production. From that we formulate the electronic equivalents for the human vocal tract. Speech communication is widely accepted as an indispensable part of life. It is one of the most efficient means to express our thoughts and feelings. It appears that spoken output is one of the most essential and vital forms of communication.

In parallel with the development of electronic technology, electronic vocal tract simulators that could generate the human voice come into reality. At first their efforts are not around using digital computers, but rather to construct circuits controlled by humans to produce the imitated speech. As the times matured, computers are utilized in the task of speech synthesis.

Among the various speech synthesis methods, the approach of *formant synthesis* is chosen in this research because (i) high-quality synthetic speech can be obtained, and (ii) new constructs can be easily added into a formant synthesizer **[3]**. However, accurate extraction of synthesis parameters for formant synthesizers is not an easy task.

The work in this research can be roughly divided into three parts: (1) The implementation of a parallel formant synthesizer, LSYNTH; (2) The construction of a feedback analysis system to facilitate the accurate extraction of formant synthesis parameters; and (3) The design of a formant trajectory generation scheme for the production of Cantonese syllables. Here is a brief outline of the contents: The basic theories and background knowledge of speech analysis and synthesis are given in the first three chapters. In Chapter 1, the human speech production mechanism will be examined. Chapter 2 contains a brief review of the basic speech analysis techniques. In Chapter 3, popular speech synthesis methods are reviewed and compared.

The following four chapters contain the major works of this research. In Chapter 4, a detailed description on a parallel speech synthesizer LSYNTH, which is implemented for this research, will be given. In Chapter 5, a feedback speech analysis system used to extract formant parameters is described. This system is based on the principle of analysis-by-synthesis and utilizes a simple iterative spectral matching algorithm. In chapter 6, a formant trajectory generation scheme for the production of Cantonese syllables is presented. This trajectory generation scheme is based on Bézier curve interpolation of formant targets **[2]**. A polynomial interpolation method used to speed up the computation of Bézier curves will also be described. In Chapter 7, listening test results for some synthetic Cantonese syllables are presented. After that a conclusion is given, together with some suggestions on future works.

Chapter 1. Human speech and its production model

1.1. The human vocal system

Acoustic understanding of voice production can be indicated with the help of Fig 1.1. The human *vocal tract* is a non-uniform tube about 17 cm in length. It is terminated at one end by the *vocal cords* and at the other end by the *lips*. The cross sectional area of the tract is determined by the placement of various *articulatory organs* such as *lips, jaw, tongue, velum*, etc.

An *ancillary* cavity, the *nasal tract*, can be coupled to the vocal tract by the trap-door action of *velum*. The nasal tract begins at velum and terminates at the nostrils. During the production of *non-nasal sounds* the velum seals off the nasal cavity and so no sound is radiated from the nostrils.

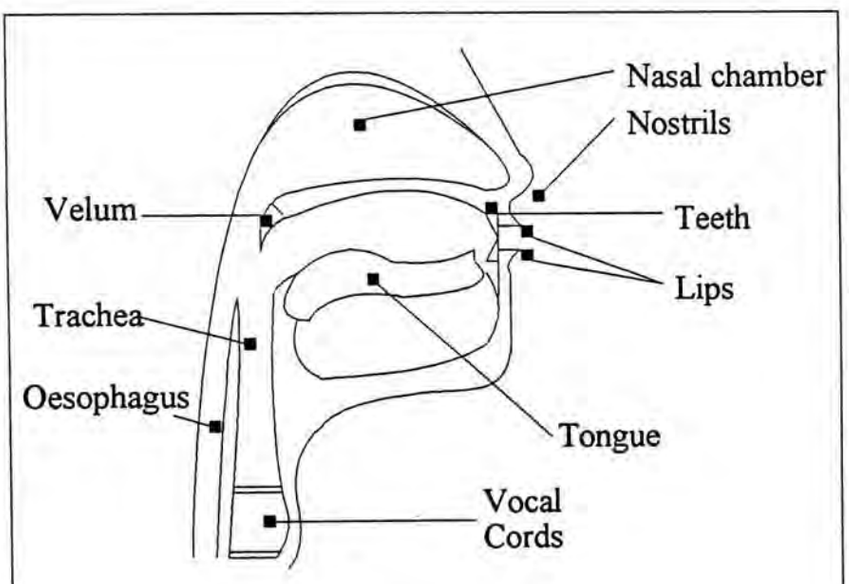


Figure 1.1 Human vocal system

1.2. Speech production mechanism

Sound can be generated in the vocal system in different ways. **Voiced sounds** are produced by elevating the air pressure in the lungs, forcing an air flow through the glottis and causing the vocal cords to vibrate. The interrupted air flow produces quasi-periodic *glottal pulses* which excite the vocal tract. Periodicity of the pulsation determines the *pitch* of the voice tone. The temporal variation of pitch determines the *accent* and *intonation* of uttered words, phrases and sentences. Sometimes, the *nasal tract* is coupled to the vocal tract in the speech production process, generating a **nasal sound**. An **unvoiced sound** is generated by air turbulence rather than by the vibration of the vocal cords. This type of sound can be further classified into the following categories: *Fricative sounds* are generated by forming a narrow constriction by articulators (e.g. teeth, tongue) to achieve a turbulent air flow. *Plosive sounds* are produced by first building up air pressure inside the oral cavity by making a complete closure of the lips, and then release the pressure by opening the lips suddenly.

1.3. Acoustic properties of human speech

Physically, the production process of voiced sound can be modeled by a lossless tube which is excited by a source of quasi-periodic glottal pulses. This tube had its *resonance modes* and are usually called *formants*. In the frequency domain, voiced sounds are characterized by the existence of several formant peaks in their power spectra. The nasal sound is characterized by the presence of *formants* and *anti-formants*. Although a formant is easy to locate in a power spectrum, the location of anti-formants are hard to observe. Usually, sophisticated analysis tools are employed to trace the locations of anti-formants.

Since unvoiced sounds are not generated by vocal cord vibrations, there are no resonance properties observed in the power spectrum. Unvoiced sounds are characterized by a high energy contribution from the high frequency region in the power spectrum (See Fig 1.2).

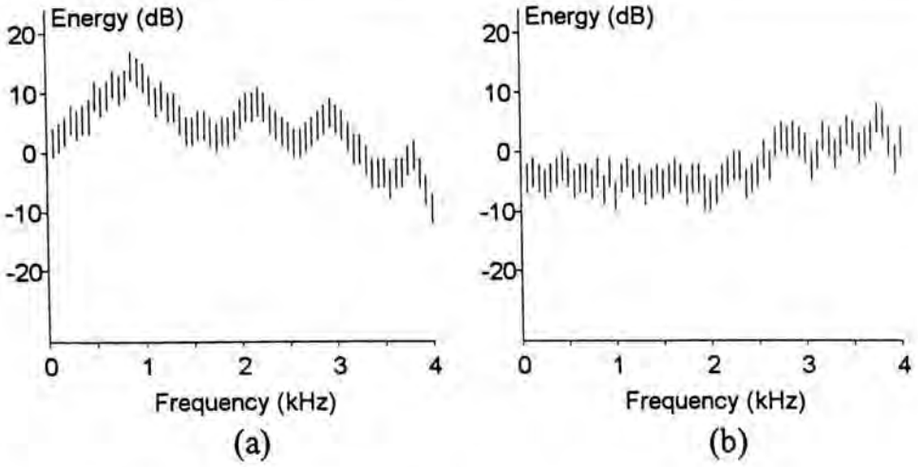


Figure 1.2 Power spectra of speech signals
(a) voiced speech (b) unvoiced speech

1.4. Modeling the speech production process

Modeling of the human speech production process is an important key to the development of speech synthesis technology. The **source-filter model [1]** represents the fundamental concept of modern speech synthesis technology. In this model, the vocal tract is simulated by a set of linear filters which are excited by a sound source. These filters are excited independently to each other to shape the spectrum of the synthetic speech. A *pulse generator* and a *random noise generator* are used to simulate *voiced* and *unvoiced* sound respectively. Finally, a *radiation characteristic* function is used to simulate the radiation of sound waves from the lip/nostrils to the free space. Mathematically this relationship can be written as

$$S(f) = G(f)H(f)R(f)$$

where $S(f)$ is the synthetic speech spectrum,

$G(f)$ is the speech excitation spectrum,

$H(f)$ is the vocal tract transfer function,

$R(f)$ is the lip radiation characteristic.

Even through this model is only an approximation, it is very useful for the analysis of speech phenomenon independently and quantitatively. In fact, this model has contributed significantly to establishing modern acoustical theories and their quantitative descriptions. This model have been adopted by most speech synthesizers, and the synthetic speech quality produced base on this model is quite satisfactory (See Fig 1.3).

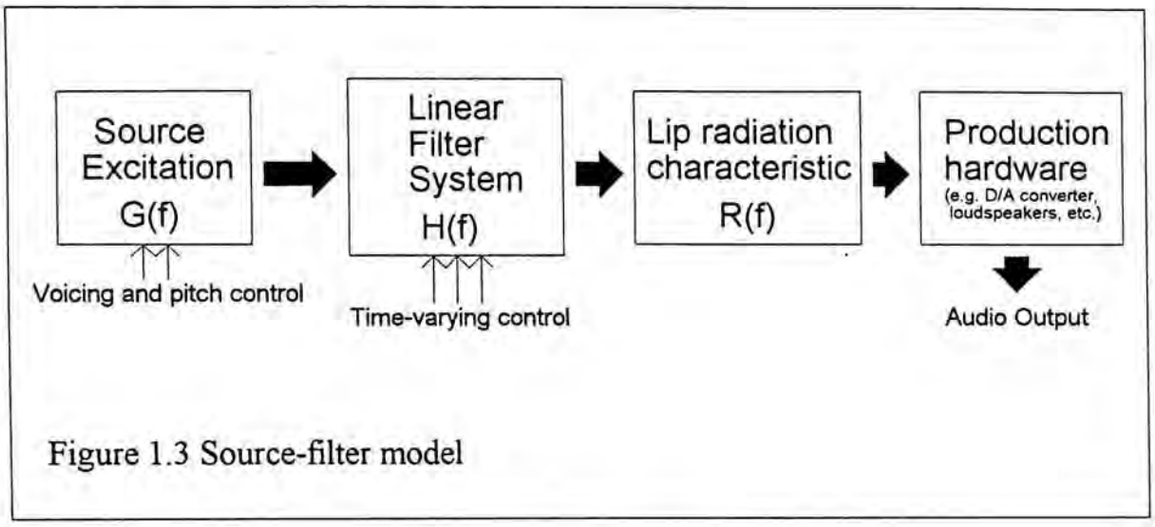


Figure 1.3 Source-filter model

1.5. Speech as the spoken form of a language

Human speech can be viewed as a spoken form of a language. From the viewpoint of phonetics and linguistics, human speech can be represented by an abstract notation consisted of a sequence of phonetic symbols. Phonetically, the basic building blocks of speech are called *phonemes*. Every speech utterances can be written as a sequence of phonemes (plus some additional symbols to indicate other *prosodic* information such as tone, stress, duration, etc.). The International Phonetic Alphabet (IPA) is an universal standard of phonetic notations which is applicable in most languages. Phonemes are usually classified by the points of articulation and their voicing nature, and is roughly divided into two classes: *vowels* and *consonants*.

Apart from phonetical descriptions by phonemes, other *abstract linguistic features* are also important in the notation of human speech. *Intonation* is the way of pitch variation over a phrase or sentence. *Tone* is the pitch variation within a word. It is essential in the identification of words in tonal languages (e.g. Chinese). *Stress* reflects the degree of emphasis. *Stressed sounds* are usually louder and have a higher pitch than other unstressed words.

Chapter 2. Speech analysis techniques

2.1. Short time speech analysis and speech segmentation

A speech signal need to be segmented into small *speech frames* before short-time analysis can be performed. Short-time analysis is an indispensable method to analyze the time-varying characteristic of speech signals. Practically, speech segmentation can be done by sliding a fixed-size time window along the time-axis in a fixed advancement rate. Using this method, all speech frames will have the same size and there is often some degree of overlapping between adjacent speech frames. Another way of doing it is using *pitch synchronous analysis* **[4]**. In a pitch synchronous analysis, the starting point of every glottal pulses are marked. Using these positions, the whole speech signal is segmented into non-overlapping frames with length equal to one pitch period (See Fig 2.1).

2.2. Pre-emphasis

The *power spectrum* of natural speech waveform is a convolution of: (a) vocal tract transfer characteristic, (b) glottal pulse spectrum, and (c) lip radiation characteristic. If we are only interested in analyzing the vocal tract transfer characteristic, it is necessary to reduce the effects of (b) and (c) from the power spectrum. Usually this is accomplished by a technique called *pre-emphasis*. Practically, pre-emphasis is implemented by passing the speech waveform through a filter. A typical pre-emphasis filter can be represented by the transfer function $1-\mu z^{-1}$ where μ is near to unity. Pre-emphasis, when used in *formant tracking*, will generally results in a slight upward shift for the estimated formant frequency locations with respect to no pre-emphasis ($\mu=0$). These shifted values are preferred since the pre-emphasized spectrum is a more accurate representation of the vocal tract transfer characteristic **[5]**.

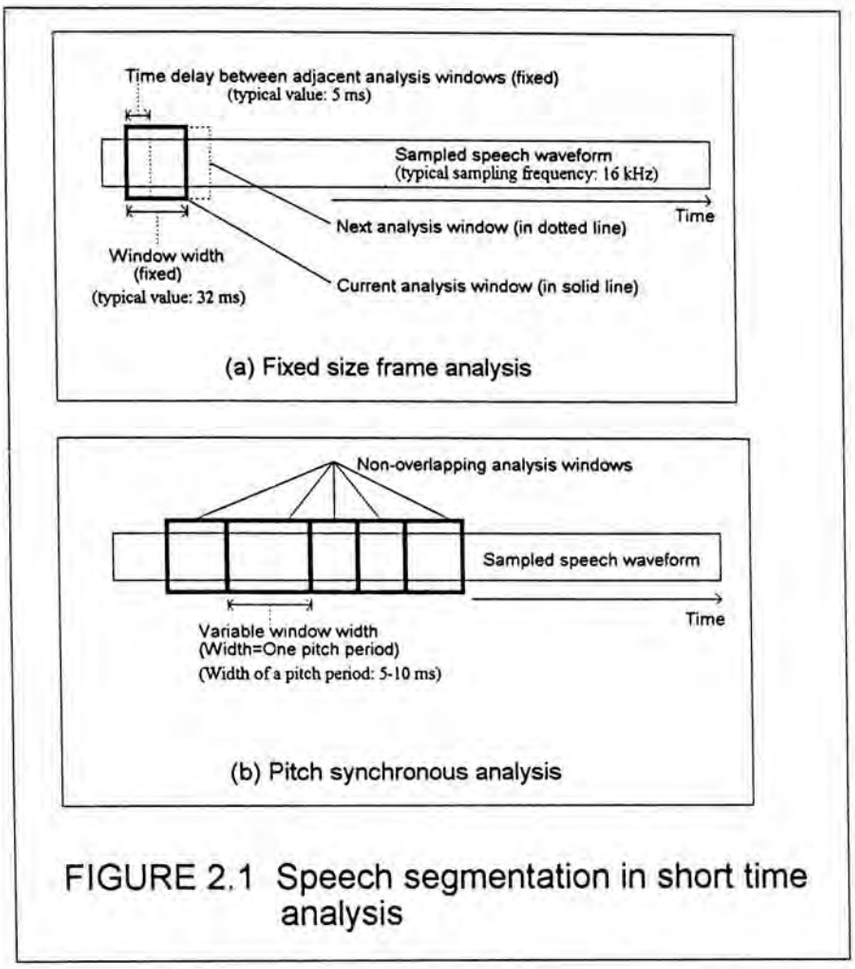


FIGURE 2.1 Speech segmentation in short time analysis

2.3. Linear predictive analysis

Results of past experiments have shown that there is a *high correlation* between adjacent samples of speech waveforms. Then assuming this relationship, a simple linear prediction can be adopted as follows:

$$y_n \approx \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \ldots + \alpha_p y_{n-p} \,. \qquad \text{(Equation 2.1)}$$

This relationship assumes that the sampled value of a speech waveform y_n is predictable by the weighted summation of p samples in the past ($y_{n-1}, y_{n-2}, \ldots, y_{n-p}$), each of these is multiplied by a constant α_i (i=1,2,...,p). These constants are called **linear predictive coefficients** $\{ \alpha_i \}$ (i=1,2,...,p) and p is called the **order of analysis**. The method of seeking the linear predictive coefficients is called **linear predictive analysis**.

2.3.1. Calculation of linear predictive coefficients

To obtain the linear predictive coefficients, one has to minimize the *prediction error* ε_n which is given by:

$$\varepsilon_n = y_n - (\sum_{i=1}^{p} \alpha_i y_{n-i}) \ . \qquad \text{(Equation 2.2)}$$

Mathematically, these linear predictive coefficients can be found by setting each partial derivative of the prediction error with respect to each α_i to zero. After this procedure, we can get a set of p simultaneous equations:

$$\begin{pmatrix} r_0 & r_1 & r_2 & \cdots & r_{p-1} \\ r_1 & r_0 & r_1 & \cdots & r_{p-2} \\ r_2 & r_1 & r_0 & \cdots & r_{p-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{p-1} & r_{p-2} & r_{p-3} & \cdots & r_0 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_p \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_p \end{pmatrix} , \qquad \text{(Equation 2.3)}$$

where r_j is a *correlation coefficient* of waveform $\{y_n\}$. One method of defining r_js is using a finite number of samples from $\{y_n\}$, usually windowed by a window function w_n:

$$r_j = \frac{1}{N} \sum_{n=0}^{N-j-1} y'_n y'_{n+j} = \frac{1}{N} \sum_{n=0}^{N-j-1} y_n y_{n+j} w_n w_{n+j} \ . \qquad \text{(Equation 2.4)}$$

Calculating the linear predictive coefficients using r_j's defined in the above equation is called the **autocorrelation method**.

Since the left hand side matrix in equation 2.3 is both *symmetric* and *toeplitz*, efficient algorithms such as Levinson-Durbin **[6]** method can be used to get the α_is.

2.3.2. Frequency domain implications

In the frequency domain, the input-output relationship of equation 2.1 can be represented by a transfer function H(z) by taking a z-transform on both sides of the equation:

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum_{i=1}^{p} \alpha_i z^{-i}} \quad \text{(Equation 2.5)}$$

Once the linear prediction coefficients are known, the *LPC power spectrum* can be calculated by the following equation:

$$P(\omega) = \frac{\sigma^2}{|A(z)|^2_{z=e^{j\omega}}} = \frac{\sigma^2}{\left|1 - \sum_{i=1}^{p} \alpha_i z^{-i}\right|^2_{z=e^{j\omega}}} \quad \text{, and } \omega = \frac{f}{f_s} \quad \text{(Equation 2.6)}$$

Where σ is the LPC *residue power gain*, $\sigma^2 = r_0 - \sum_{i=1}^{p} \alpha_i r_i$, and ω is the *normalized frequency*, obtained by dividing the actual frequency f by the sampling frequency f_s (both are expressed in Hz).

2.3.3. Discrete LPC spectrum

Practically, the spectral intensities in equation 2.6 are evaluated at N equally spaced points on the unit circle of the z-plane (this is equivalent to evaluate the spectral intensities on equally spaced points on the frequency-axis in the frequency domain), leading to the formula:

$$P_{discrete}(n) = P(n\Delta f) \quad \text{for n} = 0,1,2,\ldots N \; . \quad \text{(Equation 2.7)}$$

Where Δf is the z-plane frequency resolution factor and is equal to 1/N. The corresponding frequency resolution in Hz is given by the quantity $f_s \Delta f$.

2.4. Formant tracking

The ability of measuring the center frequencies of areas of resonance (formants) in the short-time power spectrum of speech is absolutely essential for speech synthesis using formant synthesizers. The process of measuring formant frequencies in successive time frames is known as formant tracking. Among the various techniques in formant tracking, LPC-based algorithms are widely used. This is because LPC-based formant tracking algorithms can give reasonably well formant frequency estimates and are also relatively simple to implement. A typical LPC-based formant tracking algorithm can also be used to give *formant bandwidth* and *formant amplitude* estimates.

2.4.1. Overview

In a typical LPC-based formant extraction system, the sampled speech waveform is first *segmented* into small chunks called *speech frames*. Then some pre-processing (e.g. pre-emphasis, windowing, etc.) will be done to the speech frames. After that **LPC analysis** is carried out for each frame and initial estimates of *formant parameters* (formant frequency, formant bandwidth and formant amplitude) can be obtained by analyzing the LPC transfer function H(z) using techniques like **root-solving** or **peak picking**. These initial estimates are often referred as **raw data** (See Fig 2.2).

A graphical plot of just the raw data over voiced speech intervals is generally sufficient to estimate the *formant frequency locations* in each speech frame by *visual inspection*. However, for automatic formant trajectory estimation, it is necessary to perform decisions to ensure continuity of the trajectories. For example, if there is a reason to believe that a formant peak should exist at some location where one was not extracted, then one needs to be inserted **[5]**. It is important to note that this kind of judgment makes sense only when at least several frames of raw data are available. In addition, a voiced-unvoiced decision is also important in identifying voiced regions from continuous speech.

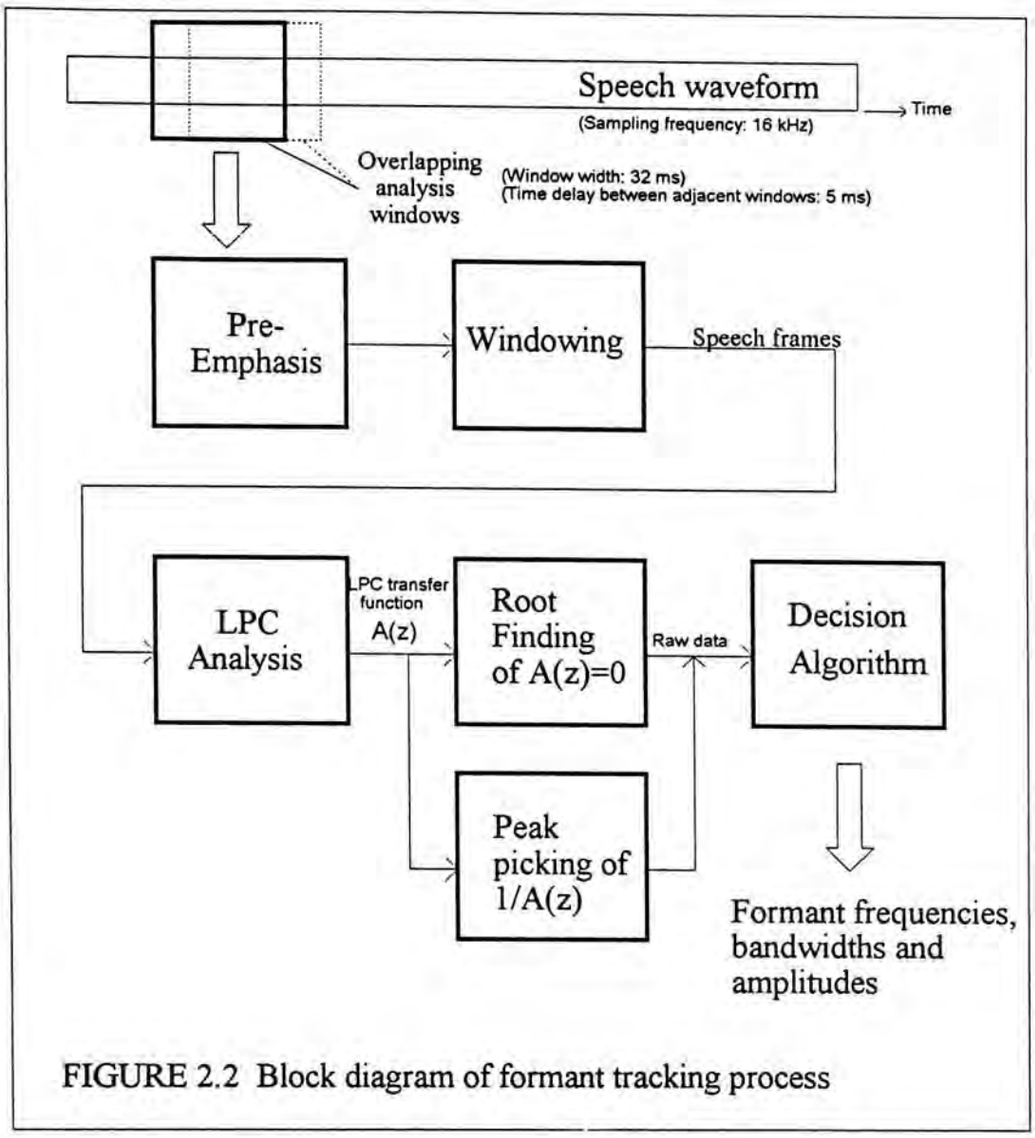


FIGURE 2.2 Block diagram of formant tracking process

2.4.2. Extraction of raw data

An LPC-based formant tracking algorithm, which is designed to give values of formant frequency, bandwidth and amplitude for the first **four** formants in successive time frames, will be described here: Natural speech segment, which is digitized at a sampling rate of 16 kHz and bandlimited to 8 kHz, is first segmented into overlapping frames with a frame width equals to 32 ms (512 samples). The time difference between the starting point of two consecutive frames (advancement) is set to 5 ms (80 samples). *Pre-emphasis* is done in each frame by passing the speech frame to the filter $1-z^{-1}$. A 512 point Hamming window is then applied to each frame before LPC analysis is performed on it.

All speech frames will be labeled as *voiced* or *unvoiced*. This is done by a **pitch extraction algorithm**. The pitch extraction algorithm employed in this tracking algorithm will be discussed in section 2.5.

An 18-th order, autocorrelation LPC analysis is performed on every **voiced** frames, and raw estimates of formant frequencies, bandwidths and amplitudes are obtained for each frame by *polynomial root solving* of the reciprocal of LPC transfer function **A(z)** (See section 2.3.2). From each root $\mathbf{z_i}$ such that $\mathbf{A(z_i)=0}$, the corresponding formant frequency, amplitude and bandwidth are calculated by the following equations:

$$f_i = \frac{1}{2\pi T}\arg(z_i),$$

$$b_i = \frac{i}{\pi T}\left|\log z_i\right|,$$

$$amp_i = \frac{\sigma^2}{\left|A(z_i)\right|^2} \text{ and } A(z) = 1 - \sum_{k=1}^{p}\alpha_k z^{-k}, \text{ where}$$

α_i is the linear predictive (LPC) coefficients (i=1,2,...,p),
p is the LPC analysis order,
σ is the average residual power,
f_i is the formant frequency for z_i,
b_i is the formant bandwidth for z_i,
amp_i is the formant amplitude for z_i,
A(z) is the denominator of the LPC transfer function H(z), H(z)=1/A(z).

Notice that we have use the *spectral intensity* at the formant peaks as the values of *formant amplitudes*. This scheme can only give rough estimates for formant amplitudes because the relation between formant amplitude and spectral intensity is not that simple **[2]**. Moreover, the bandwidth estimates given by LPC analysis is not accurate enough. An iterative method in estimating these type of parameters in greater precision will be described in chapter 5.

Roots that give obviously incorrect estimates will be **eliminated** at once in order to increase the overall efficiency of the tracking algorithm. If any one of the following conditions are met, the corresponding root will be eliminated:

(a) Negative formant frequencies.

(b) Formant bandwidth > 600 Hz.

(c) Formant frequency < 50 Hz.

(d) Formant frequency > 4000 Hz.

The remaining roots are sorted in **ascending** order on their formant frequencies, and the first **6 distinct roots** will be chosen as the **raw data** of the speech frame. Therefore, 18 raw formant parameters are given for each voiced speech frame. Although the formant tracker gives formant parameters for the first four formants only, six sets (roots) of raw formant data are given in each frame by the raw data extractor. It is because there may be some "false" data (e.g. a false formant position) inside the raw data set, so two additional data sets are included to reduce the chance of losing of the "real" data. In the following context, we will denote the **raw data** for frequency, bandwidth and amplitude for the i-th formant in the k-th speech frame by $PF_i(k)$, $PB_i(k)$ and $PA_i(k)$ respectively.

2.4.3. Processing the raw data

1. In the following description, the output of i-th formant frequency estimate for the k-th speech frame by the formant tracking algorithm will be denoted as $F_i(k)$. Only the **formant frequency** trajectory tracking algorithm will be described here. Bandwidth and amplitude trajectories are tackled by a similar algorithm.

2. For each voiced region, selected an **anchor point** which serves as a starting point of formant tracking in the region. The concept of an anchor point can be found in the formant tracking algorithm given by McCandless **[7]**. Currently, this step is done *manually*. Frames

which have reasonably strong energy and having least variations in energy with its adjacent frames are feasible candidates of anchor points.

At the anchor point, we have to assign values to the four formant slots $F_i(k)$ (i=1,2,3,4) from the six possible candidates $PF_j(k)$ (j=1,2,3,4,5,6). That is, we have to find four distinct indexes j_1,j_2,j_3,j_4 ($1<=j_i<=6$, i=1,2,3,4) such that $F_i(k)=PF_{j_i}(k)$ (i=1,2,3,4). Usually, this selection is done by picking up four $PF_j(k)$s which have the greatest *spectral intensities*.

3. Both a **forward branch** and a **backward branch** will be performed from the *anchor point* to the end and the start of the voiced region respectively. When the two branches are done, one voiced region is processed. The tracking task is done if all voiced regions are processed. This is similar to the scheme due to McCandless **[7]** (See Fig 2.3a).

4. ***Minimum distance criterion*** is applied in order to select the appropriate formant estimates **[5]**. The following is the decision algorithm for a **forward branch**. The algorithm for a backward branch is similar.

   $PF_i(k)$ will be assigned to $F_j(k)$ if and only if:

   $| F_j(k-1) - PF_i(k) | < | F_j(k-1) - PF_m(k) |$ for m=1,2,3,4,5,6 $m \neq i$ and

   $| PF_i(k) - F_j(k-1) | < | PF_i(k) - F_n(k-1) |$ for n=1,2,3,4 $n \neq j$.

   If $F_j(k)$ is still unassigned after applying the minimum distance criterion then $F_j(k)=F_j(k-1)$ (See Fig 2.3b).

5. Smoothing the trajectory

   After selecting the appropriate raw data points, the resultant formant trajectory may still have some unaligned parts. If this is the case, formant position in grossly unaligned frames will be adjusted by *interpolation*. Moreover, the resultant trajectory, though having no unaligned parts, may be "bumpy" are needed to be smoothed.

6. Error handling

Occasionally, the user may get an incorrect formant trajectory. The reason may be one of the following: (a) Inappropriate selection of anchor points, (b) Mistake in filling formant slots at the anchor points, (c) Very rapid changes in formant frequencies over a short time. In these cases, manual editing of formant tracks may become necessary.

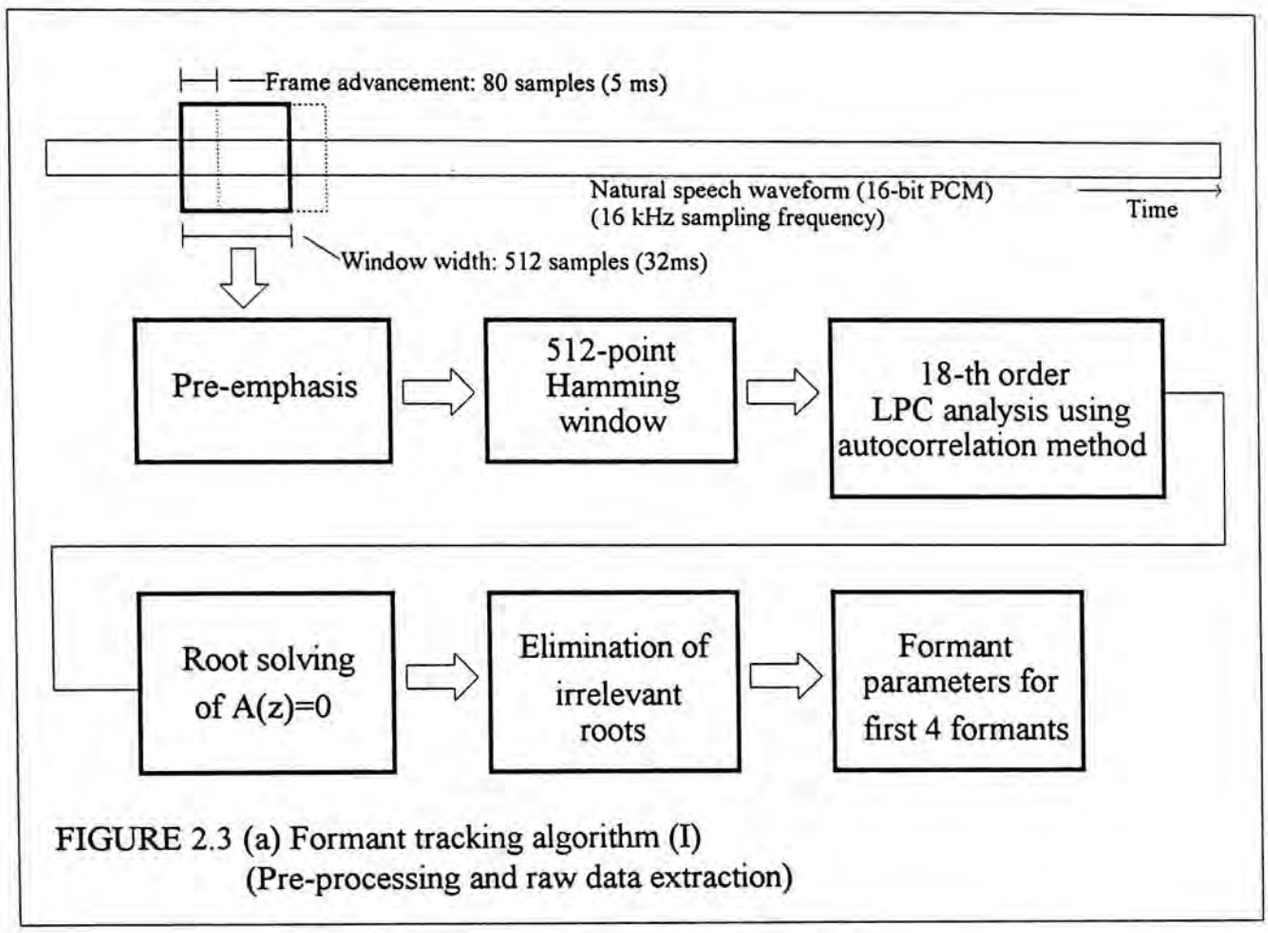


FIGURE 2.3 (a) Formant tracking algorithm (I)
(Pre-processing and raw data extraction)

2.4.4. Implementation and discussions

A program is written to extract raw formant parameters directly from a digitized speech waveform using the scheme mentioned in section 2.4.2. Based on these raw data, a **semi-automatic formant tracking system** based on the scheme proposed in section 2.4.3 is implemented. In the current implementation, the process of (1) selection of anchor points, (2) handling unaligned parts of trajectory and (3) smoothing of trajectory are all done manually

because automatic algorithms for these processes may be error-prone. It is hoped that the above process can be automated in the future implementations.

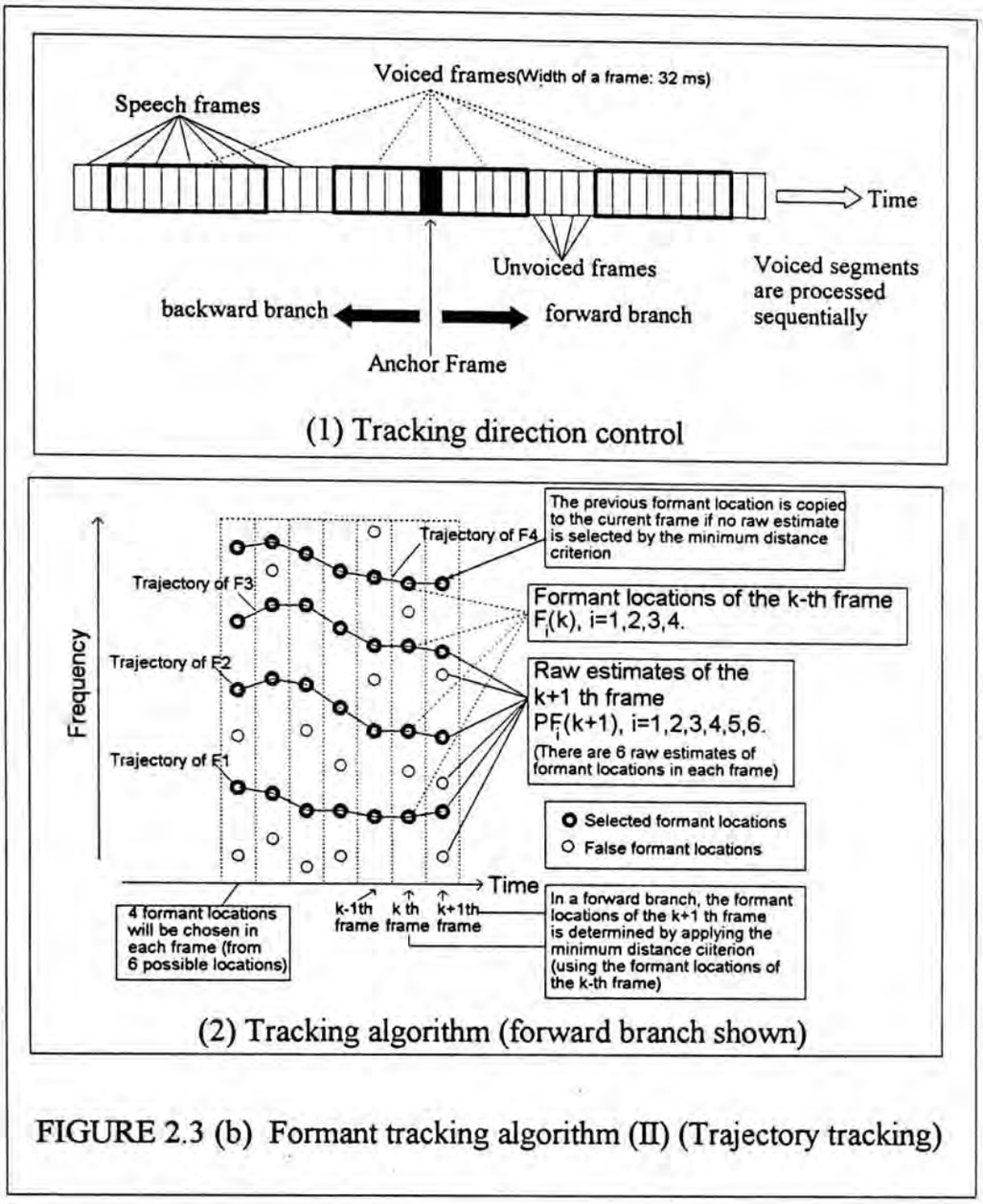


FIGURE 2.3 (b) Formant tracking algorithm (II) (Trajectory tracking)

Reasonably good formant frequency estimates can be obtained by using the proposed formant tracking system. However, there are still room for improvement. For example, the formant amplitude data provided by the formant tracking system are not very accurate. Useful suggestions on the future development of this formant tracking system may include: (1) Using *pitch synchronous analysis* in speech segmentation (2) Devise a more sophisticated rule-based algorithm to link up the formant trajectory in successive time frames (3) Using a DSP

processor in the system to boost up execution speed (4) Other LPC-based algorithms, such as LSP (Line Spectrum Pairs) **[8]**, can be used to assist the process of finding the formant locations, though the computation time will be considerably increased then.

2.5. Pitch determination

The role of the pitch determination algorithm is twofold. It has to provide **voicing decision** as well as giving the **fundamental frequency** estimation if the speech frame is found to be voiced. Voiced/Unvoiced discrimination plays an important role in formant tracking for continuous speech. Accurate pitch extraction is also essential for high quality speech synthesis.

2.5.1. Pitch determination algorithms

Pitch determination algorithms (PDA) can be divided into two broad categories. Time domain PDAs (TDPA) and frequency domain PDAs. (FDPA). There are many popular PDAs such as AMDF(Average Magnitude Difference Function) **[9]**, SIFT(Simplified Inverse Filter Tracking) **[10]**, Cepstral methods **[11]**, etc. Since many PDAs can give resonably well pitch estimates, the implemention of one PDA is sufficient to assist the formant tracking process mentioned in section 2.4. After a careful consideration, the method of **modified autocorrelation [12]** is chosen because it can utilize the linear predictive coefficients calculated in the formant tracking algorithm (described in section 2.3). Thus the pitch determination system can be easily integrated into the formant tracking system described in section 2.4.

2.5.2. Modified autocorrelation method

One of the oldest methods for estimating the fundamental frequency of voiced speech is seeking peaks in the **autocorrelation sequence** of the input speech frame. But the major drawback of this method is the *formant structure* (especially a lowly positioned F1) may

significantly influences the estimation of the pitch period **[5]**. In some extreme cases the pitch period can be completely masked. To alleviate this problem, Saito and Itakura **[12]** had proposed a *modified autocorrelation method* which can reduce the formant influences.

In the **modified autocorrelation** method, speech segments are first pre-processed and then LPC analysis will be done on the pre-processed waveform. After that a **residual signal** is obtained from inverse filtering the original waveform with the LPC inverse filter **A(z)**. That is:

$$R(i) = x(i) - \sum_{j=1}^{p} \alpha_j x(i-j) \quad \text{for i=0,1,...,N-1 ,}$$

where R(i) is the i-th sample of the LPC residual for the speech frame, i=0,1,..,N-1 ,

α_j is the j-th linear predictive (LPC) coefficient, j=1,2,...,p ,

x(i) is the i-th sample of the speech frame, i=0,1,...,N-1 ,

N is the length of the speech frame.

Then an **autocorrelation sequence** is evaluated from the residual signal:

$$S(k) = \sum_{i=0}^{N-k-1} R(i)R(i+k) \qquad \text{for k=0,1,...,N-1 ,}$$

where S(k) is the k-th sample of the autocorrelation sequence,

R(i) is the i-th sample of the LPC residual,

N is the length of the speech frame.

This autocorrelation sequence will be **normalized** and peaks are picked from this normalized sequence. If the peak amplitude exceeds a pre-set threshold, the speech frame is considered to be voiced and the pitch period can be determined from the first peak location, otherwise the speech frame is declared as unvoiced (See Fig 2.4).

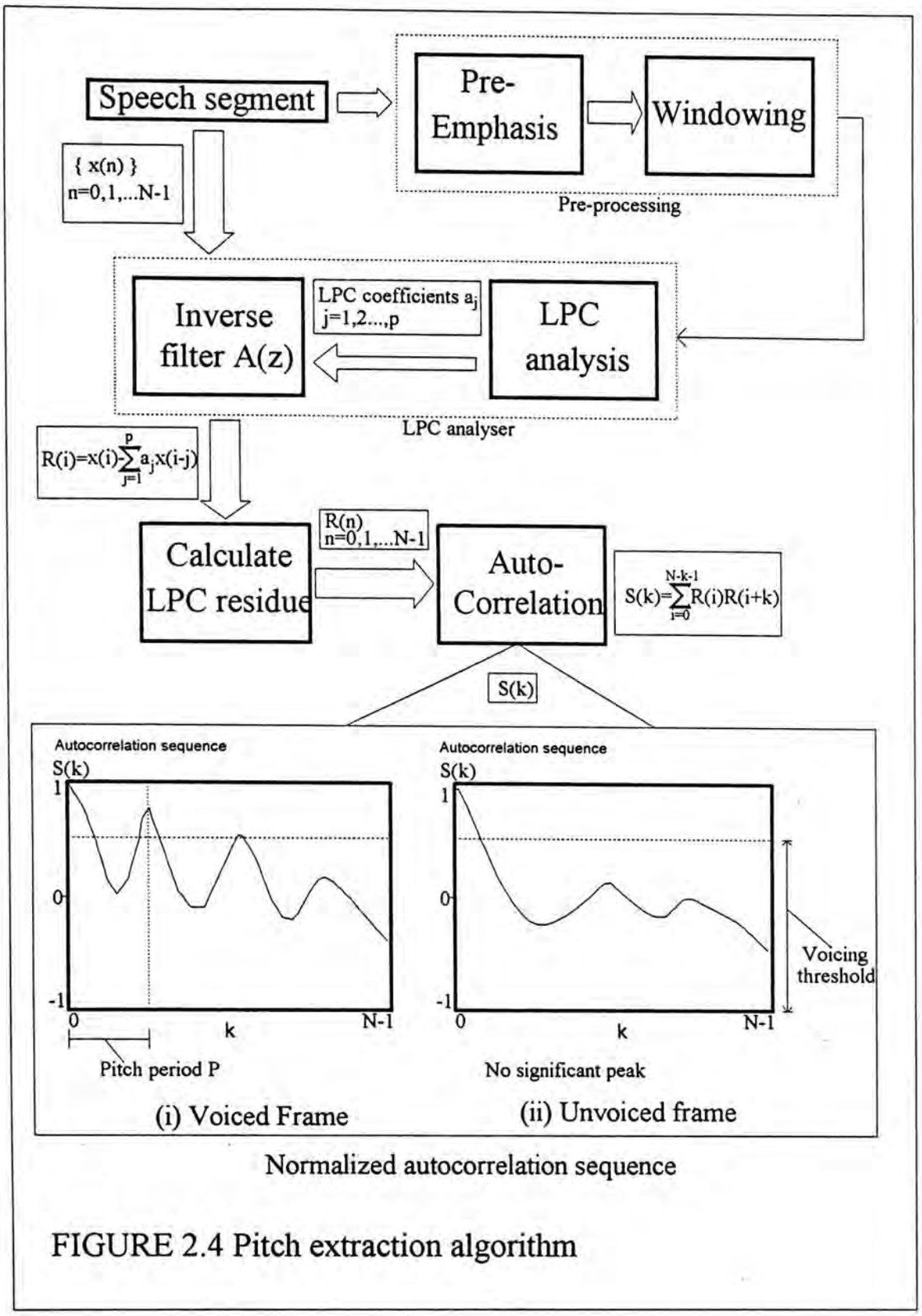


FIGURE 2.4 Pitch extraction algorithm

2.5.3. Implementation and discussions

In the current implementation, modified autocorrelation method is applied with the following conditions: LPC analysis order=18, speech frame length=512 points, pre-emphasis with filter $1-z^{-1}$, the usage of 512-point Hamming window, sampling frequency of input speech equals to 16 kHz. These conditions are just the **same** as those used in formant

tracking. So the linear predictive coefficients obtained by formant tracking can be used to perform the task of both formant tracking and pitch determination. The samples of the autocorrelation sequence is normalized to the range [1,-1]. The value of *voicing threshold* is determined experimentally and is set to **0.42** in implementation.

A number of natural and synthetic speech samples are used to test the performance of the pitch determination algorithm. Testing results for some synthetic speech samples have shown that quite accurate fundamental frequency estimates can be obtained by this algorithm. Voicing decisions are generally correct for unvoiced-voiced transitions. For voiced-unvoiced transitions, this algorithm gives an unvoiced decision a little bit earlier than expected for many test cases. Figure 2.5 shows the results obtained by the pitch determination algorithm using a synthetic waveform as input.

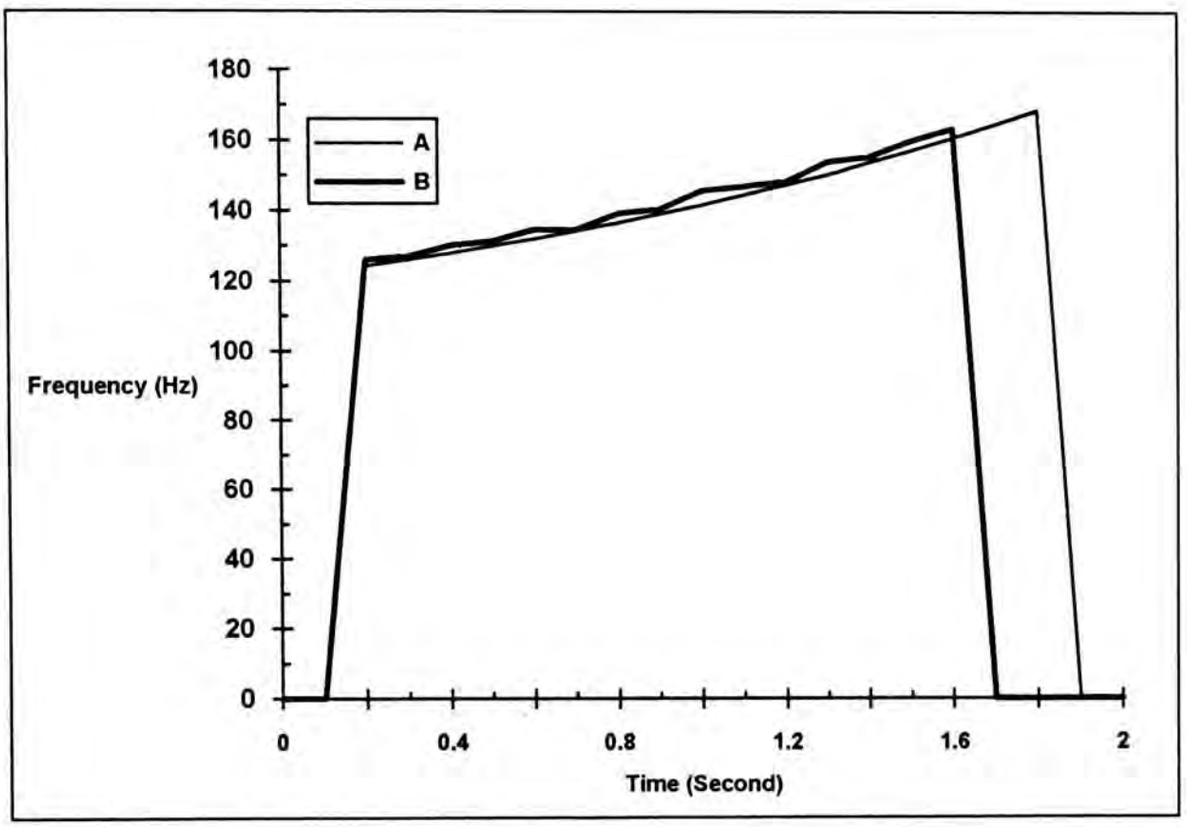


FIGURE 2.5. A synthetic waveform is used to test the performance of the pitch extraction algorithm presented in section 2.4. The synthetic waveform is synthesized by a formant synthesizer LSYNTH (For more details, please refer to chapter 4) with a pitch contour indicated by line A. The pitch contour extracted by the pitch determination algorithm is shown by line B. The unvoiced portions of the pitch contours (both A and B) are set to 0 Hz in the graph for convenience. The synthetic waveform is synthesized with a sampling frequency of 16 kHz.

Chapter 3. Speech synthesis technology

3.1. Overview

Over the years, many speech synthesis techniques were developed. In this chapter, some common speech synthesis techniques will be described, together with some discussions on their advantages and disadvantages.

3.2. Articulatory synthesis

One approach of speech synthesis is to simulate the human speech system by making precise mathematical models of the vocal tract and the articulatory organs directly from their physical characteristics, or establish electrical/mechanical equivalents of the human speech system. Synthetic speech samples are generated based on these mathematical models, usually by solving a set of partial differential equations (e.g. The model proposed by Shirai **[13]**) . Theoretically this approach represents the ultimate solution in the generation of synthetic speech. But the precise acoustic aspects of a complex articulatory model that might account for naturalness (e.g. spectral zero movements, glottal waveform changes, voice-unvoiced transitions, etc.) are not known at this time. Also, the high computational cost of articulatory synthesis precludes the use of these models in practical systems at the present time.

3.3. Concatenation synthesis

The basic principle of **concatenation synthesis** is to reconstruct speech with a set of pre-recorded speech segments, usually stored in a *waveform library*. During the synthesis process, the required speech segments are retrieved and concatenated in a proper way before the playback is done. Compared to other synthesis methods, this method is the simplest and requires the least processing power. Many small-scale commercial speech synthesis systems are based on this method.

A major consideration of using concatenation synthesis is to decide the basic building unit of speech. Clearly, speech synthesis using *sentence concatenation* is not practical since the number of different sentences in a language is extremely large. A more practical way is to concatenate at the word/sub-word level.

3.3.1. Concatenation in the word level

Speech synthesis by word concatenation is widely used in situations that required only a limited vocabulary. e.g. A synthesis system used to tell telephone numbers to the users. The synthetic speech output is constructed by concatenation of pre-recorded words stored in a waveform library. This method is easy to implement and the quality of speech output is quite good. However, there are two apparent limitations: (1) It requires a big storage space and fast data retrieval speed, (2) Coarticulation effects between adjacent words are completely missing.

With the advent of digital technology, speech waveform can be digitized and stored in devices such as magnetic hard disks or CD-ROM. There are also effective speech coding scheme which can greatly reduce the total data storage. Although storage space is not a serious problem, the data retrieval speed may still not be fast enough to produce synthetic speech in real-time for such synthesis systems (e.g. when reading a very long passage). Another disadvantage of this method is that the system cannot pronounce words that is not in the vocabulary of the synthesizer. In addition, a very unnatural feeling is presented when a long passage is "readout" by such a word-concatenation system due to the lack of co-articulation between adjacent words.

3.3.2. Concatenation in the sub-word level

Concatenation in the sub-word level does not require a large data storage space because the number of sub-word units is significantly smaller than the number of words in most languages. Possible sub-word units can be syllables, phonemes, or other specially-defined

unit (e.g. diphones). In a typical language like English, the number of distinct syllables is over ten thousand but the number of distinct phonemes is only about fifty.

Usually, concatenation of phonemes directly fails to produce intelligible synthetic speech because the coarticulation effect at the boundaries between adjacent phonemes is not handled. This problem is proposed to be solved by introducing other sub-word units as the basic building block. One prominent candidate of these sub-word units is *diphones* **[14]**. Diphone is defined as the phonetic unit that is comprised of the rear half of a phoneme plus the front half of another phoneme. Acoustically, diphones are usually defined in a way such that very little co-articulation effects can be found in its starting and ending region. Due to this nice property, we need not handle the co-articulation effects at the boundary of neighboring diphones when doing the concatenation process.

In English, the number of distinct diphones is about 1600. This number is still small when comparing to the total number of words. In practical use, the number of diphones is about 8000 in order to contain other linguistic aspects like prosody, stress, etc. Using this method, synthetic speech with satisfactory quality can be produced. Unlike the word-concatenation method, which have a finite vocabulary, this method can have infinite vocabulary (any word can be broken down into a sequence of diphones). Another advantage of this method is the co-articulation effects of adjacent words can be handled.

However, some smoothing algorithm must be used if there are any formant discontinuity at the boundary of two diphones. In some cases, it is hard to do the smoothing process by simple means. It is also not easy to change certain properties (e.g. pitch, duration,stress, etc.) inside a diphone.

3.4. LPC synthesis

LPC speech synthesis is based on linear-predictive coding (LPC) of speech signals (See Chapter 2). The most important advantage of using LPC is that fairly good synthetic speech quality can be obtained with a rather low bit-rate (at 2 kbit/s). Moreover, pitch and duration of the synthetic speech signals can be varied without disturbing the vocal tract characteristic given by the LPC coefficients. Typically, an LPC synthesizer does the synthesis by the following steps: (1) Retrieved the LPC-coded speech segments from a database, (2) Compute the LPC-decoded speech segments, by LPC decoding and applying an appropriate excitation signal to the LPC synthesis filter, and (3) Concatenate the LPC-decoded speech segments to form the synthetic speech output. Usually, **word** is the basic concatenation unit in LPC synthesis. Although the synthesis process of a LPC synthesizer is in some sense similar to concatenation synthesis, there are several important differences: (a) The pitch and duration data can be varied easily in LPC speech synthesis, while in concatenation synthesis it is more difficult to do so, and (b) The synthesis process of a LPC synthesizer involves the modeling of the speech excitation source.

The coding-decoding process for an LPC synthesizer can be described in Fig 3.1. In the speech coding process, the LPC coefficients (obtained by LPC analysis) of the input speech S, are stored up for each short-time speech frame. However, the LPC residual signal R(t) (contains glottal excitation data) is discarded. For the decoding process, an LPC synthesis filter is employed to reconstruct the synthetic speech signal S' from an artificial residual signal R'(t), using the previously stored LPC coefficients. Usually, R'(t) is modeled by (i) a train of periodic pulses for voiced speech, and (ii) random noises for unvoiced speech. The reason of giving up the original residual signal R(t) is that the storage space required by this signal is as large as that of speech waveform.

Since the original residual R(t) is not employed in the synthesis process, the quality of synthetic speech output may be degraded. In order to compensate for the loss of quality due to

the absence of "true" residue excitation signals, some excitation enhancement methods can be used (e.g. multi-pulse excitation **[15]**).

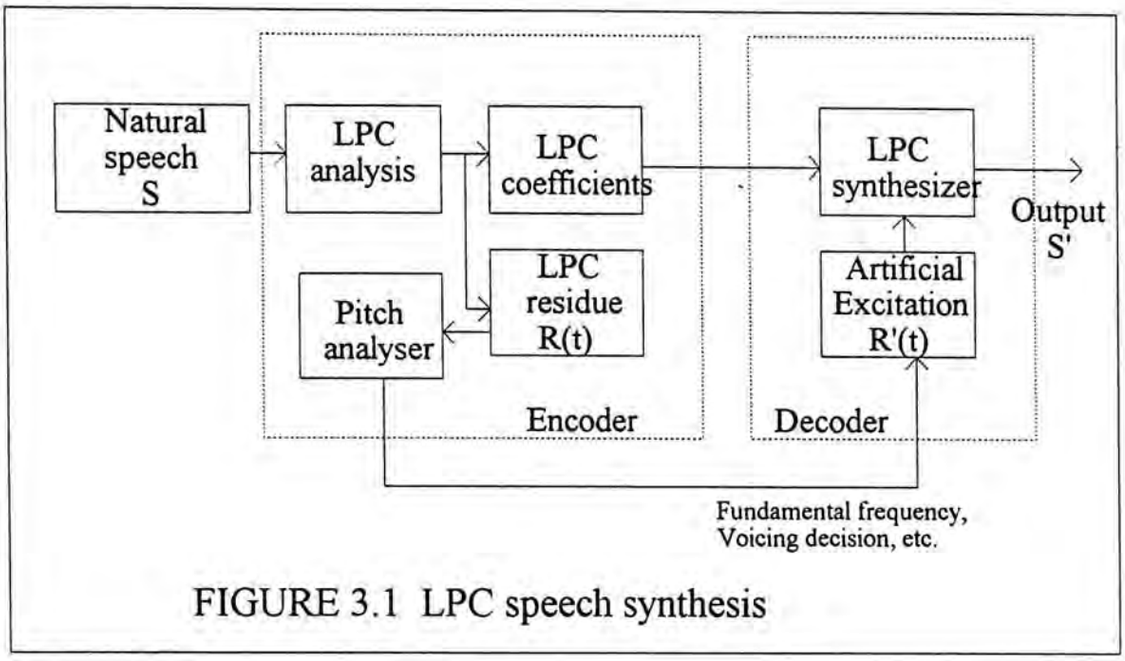


FIGURE 3.1 LPC speech synthesis

One great advantage of using LPC synthesis is the LPC synthesis filter is rather simple to implement. There are also inexpensive LPC synthesis chips available. However, for some types of sound (e.g. nasals), LPC synthesis does not give a good-quality result. Moreover, LPC synthesis cannot effectively handle speech with a high pitch **[16]**, which is common in female speech.

3.5. Formant speech synthesis

The spirit of formant synthesis is to simulate the spectrum of human speech by using a spectral shaping system consisted of resonators and anti-resonators. In a *formant synthesizer*, the spectral shaping system is driven by an *excitation source* which simulates the effect of human glottal pulses. A set of control parameters are used to control the behavior of the synthesizer. Usually, these control parameters are obtained by speech analysis.

Using this method, very high quality synthetic speech can be produced. This is the biggest advantage of using formant synthesis. However, the acquisition process of synthesizer parameters can be painstaking. Moreover, the computational workload for a formant synthesizer is relatively high.

3.6. Synthesis by rule

Given a proper phonetical representation (e.g. a sequence of phonetic symbols) of the desired synthetic speech output, a *synthesis-by-rule* system will generate the required speech synthesis data by a set of rules. These speech synthesis data will then be given to a speech synthesizer to produce the required output. The synthesis rules are often deduced by extensive analysis of human speech samples. Usually, *phoneme* is chosen as the basic building unit of speech in most synthesis-by-rule systems.

One of the outstanding synthesis-by-rule system is attributable to Holmes et al. **[17]**. In this system, abstract linguistic descriptions (written as a sequence of phonemes plus some special symbols) are converted to synthesis parameters such as formant frequencies, voicing source amplitude, fundamental frequency, etc. These synthesis parameters are used to control a formant synthesizer to produce synthetic speech. It is reported that the synthetic speech produced have a high intelligibility. Besides, the synthesis rules are simple enough to implement in many applications.

Another famous speech synthesis-by-rule system is *MITalk* **[3]**. In the *MITalk* text-to-speech system, synthetic speech can be produced from unrestricted English text. The task of *MITalk* can be divided into two parts: (1) analysis-of-text, and (2) the synthesis-of-speech.

Part (1) involves the translation of English text and abstract linguistic feature into pre-defined symbols and parameters. This is done by a rule-based phonological analysis module assisted by dictionary lookup. Part (2) converts the phonetic notations into speech synthesis

parameters and produce synthetic speech output. A set of sophisticated, yet extremely complicated, rules are used to convert the phonetic notations obtained in Part (1) to speech synthesis parameters. These synthesis parameters are used to drive a formant synthesizer to produce the desired output.

The average performance of *MITalk* is quite good. However, since this system is designed specifically for English, the synthesis rules in *MITalk* may not be suitable for other languages. Recently, synthesis-by-rule system based on a model similar to *MITalk* is developed for some languages other than English **[18]**.

Chapter 4. LSYNTH : A parallel formant synthesizer

4.1. Overview

A *speech synthesizer* is responsible to convert a set of time-varying speech parameters into a speech waveform. Fig 4.1 shows the interface between the synthesizer and other modules in the whole speech synthesis system. In the past, speech synthesizers are implemented in analog circuitry. As digital techniques become more sophisticated, most speech synthesizers are implemented in the digital domain nowadays.

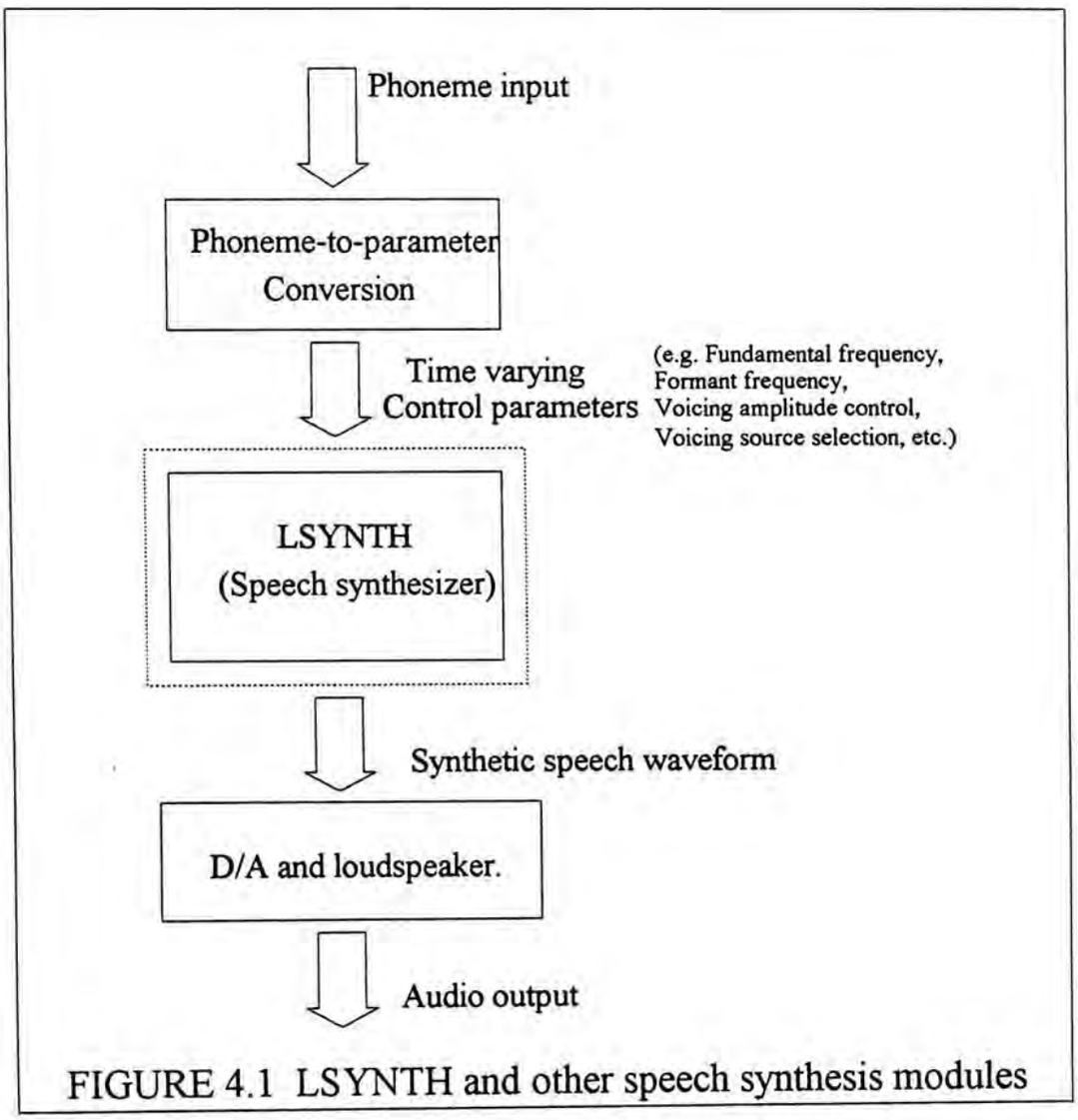


FIGURE 4.1 LSYNTH and other speech synthesis modules

A software formant synthesizer LSYNTH is implemented for this research. The advantage of a software implementation over the construction of dedicated hardware is substantial. Comparing to its hardware counterpart, a software synthesizer does not need repeated calibration. It is also stable and free of electrical noise contamination. Moreover, modification of synthesizer configuration can be easily made.

The task of LSYNTH can be roughly divided into two parts. The first part accepts **formant parameter data** such as formant frequencies, formant bandwidths, fundamental frequency, voicing amplitudes, etc. and derives a set of *difference equation coefficients* for each *digital formant resonators*. In the second part, the difference equation coefficients are used to drive a *digital filter system* (implemented in software) which will produce the synthetic speech waveform samples in memory. The speech samples can then be played back by using an appropriate speech output device.

4.2. Synthesizer configuration: cascade and parallel

A number of hardware and software formant speech synthesizers with different configurations have been proposed by various researchers **[3]**. Of the best synthesizers that have been proposed, two configurations are commonly used. In one type of configuration, the formant resonators that simulate the transfer function of the vocal tract are connected in series. This is called a *cascade* configuration. In the other type of configuration, formant resonators are connected in parallel. Each formant resonator is preceded by an *amplitude control unit* that determines the relative amplitude of a formant peak in the output spectrum of synthetic speech. This is called a *parallel* configuration (See Fig 4.2).

The advantage of a cascade configuration is that there is no need to specify the formant amplitudes for each formant resonator. A second advantage of a cascade configuration is that it is a more accurate model of the vocal tract transfer function during the production of non-nasal vowels and sonorants **[19]**. But this configuration cannot handle fricative and plosive

sounds well **[3]**. A parallel configuration is capable of producing plosives and fricatives much better than a cascade configuration. If the formant amplitudes and bandwidths are carefully tuned, a parallel configuration can generate reasonably good vowels and sonorants **[3]**. Due to its versatility, the parallel configuration is chosen in the implementation.

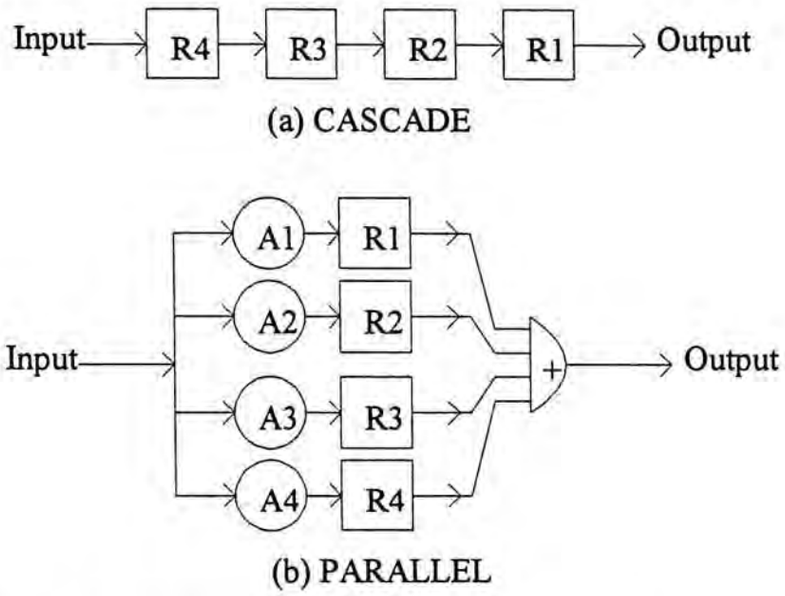


FIGURE 4.2 Formant synthesizer configurations
(a) Cascade (b) Parallel

4.3. Structure of LSYNTH

The general structure of LSYNTH is shown in Fig 4.3. There are two voice sources. A *pulse source* and a *noise source*. The two sources are used to drive 4 parallely connected *formant resonators*. Separate amplitude, bandwidth and frequency controls can be applied to resonators R1,R2,R3 and R4. The outputs of the resonators pass through a *simple differentiator* (for R2,R3,R4) or an *averager* (for R1) before they are mixed in *alternate signs*. The mixed output passes through a lip radiation module and synthetic speech samples are then obtained.

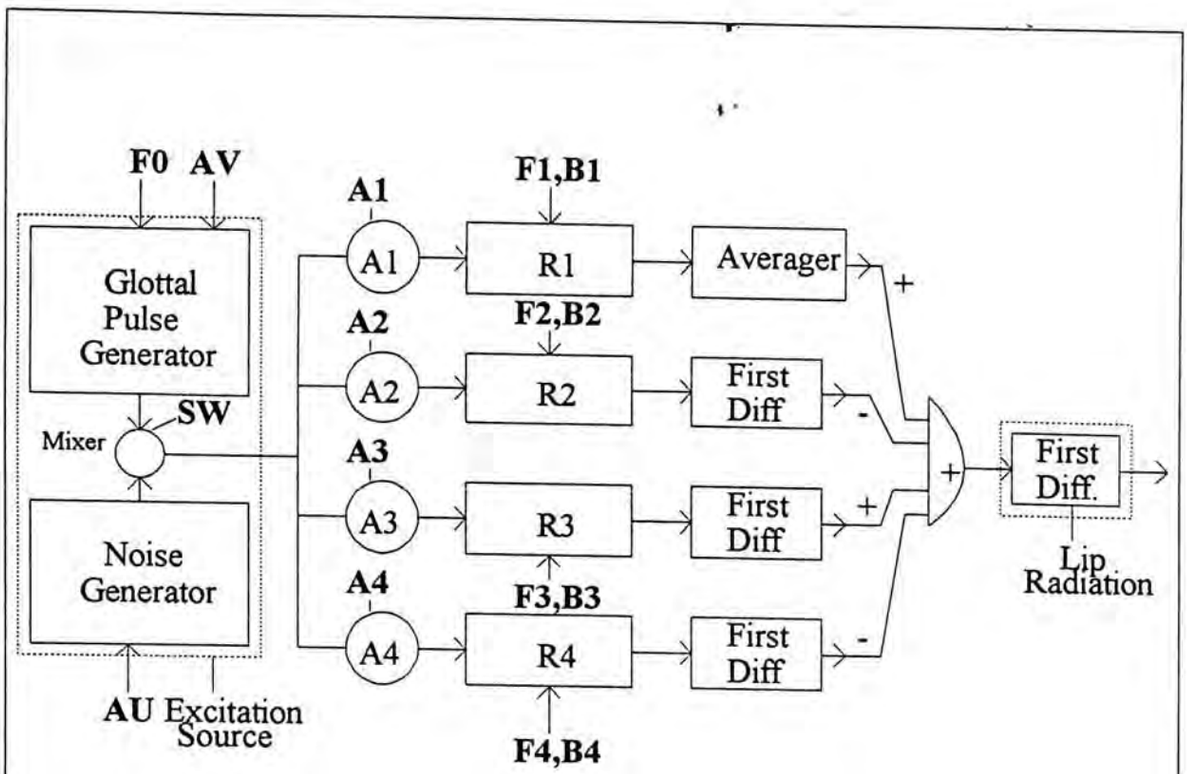


FIGURE 4.3 Block diagram of LSYNTH
(Control parameters are shown in bold letters)

4.3.1. Parameters controlling LSYNTH

There are totally 18 parameters which are used to control the behavior of the synthesizer. Among these 18 parameters, 16 of them can be varied as a function of time. The other two are system parameters (used in system setup). The parameters are listed in Table 4.1 and Table 4.2.

Parameter Name	Description	Unit
F0	Fundamental Frequency (pitch)	Hz
F1,F2,F3,F4	Formant Frequencies	Hz
B1,B2,B3,B4	Formant Bandwidths	Hz
A1,A2,A3,A4	Formant Amplitudes	dB

Table 4.1 Time-varying parameters of LSYNTH (to be continued)

Parameter Name	Description	Unit
SW	Voice source selection	0: Silence >0: Voicing Required
AV	Voicing amplitude (voiced source)	dB
AU	Voicing amplitude (unvoiced source)	dB

Table 4.1 Time-varying parameters of LSYNTH (continued)

Parameter Name	Description	Unit
SF	Sampling Frequency	Hz
FW	Width of one speech frame	No of Samples

Table 4.2 System parameters of LSYNTH

The synthesis strategy of LSYNTH is *synthesis by segments*. This means that the synthesizer produces synthetic speech one frame at a time. The duration of a frame can be specified by the user during system setup. Typically, the duration of a speech frame is around 5-10 ms. Input parameters are considered to be constant inside a frame. This assumption is justified if a speech frame is short (e.g. 5 ms) since the speech articulators are moving rather slowly and the vocal cavity shape cannot be changed too rapidly.

4.3.2. Voice sources

There are two voice sources in LSYNTH (See Fig 4.4). A *pulse source* and a *noise source*. The pulse source is responsible to produce simulated glottal pulses of *voiced sounds*. For speech synthesis, the spectrum of the glottal pulse source is commonly approximated by a spectrum with 12 dB/octave rolloff.

Under the consideration of perceptual results, Rosenberg **[20]** suggests a model to simulate the glottal pulse shape by two sinusoidal segments. This glottal pulse shape is used in the implementation of LSYNTH. The pulse shape is given by equation 4.1

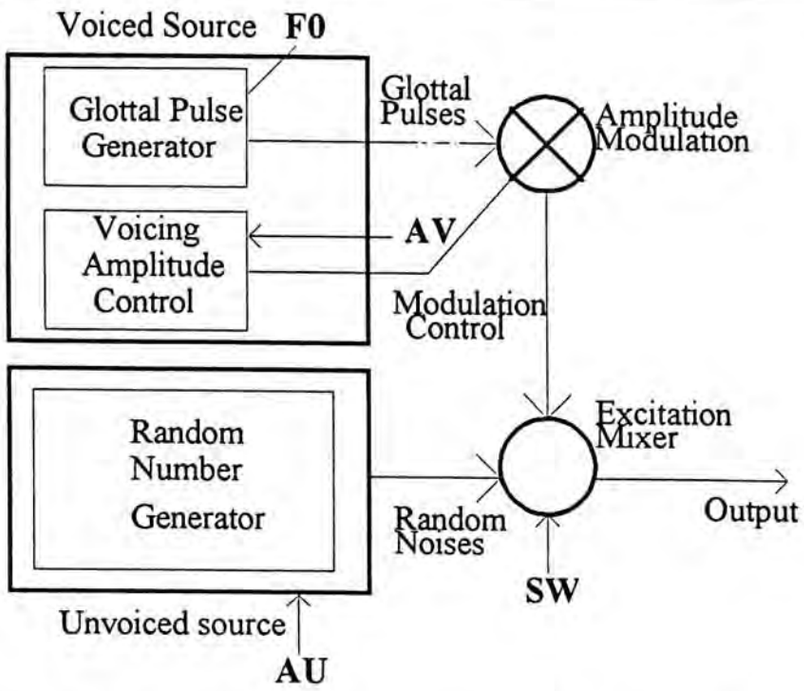


FIGURE 4.4 Excitation sources of LSYNTH

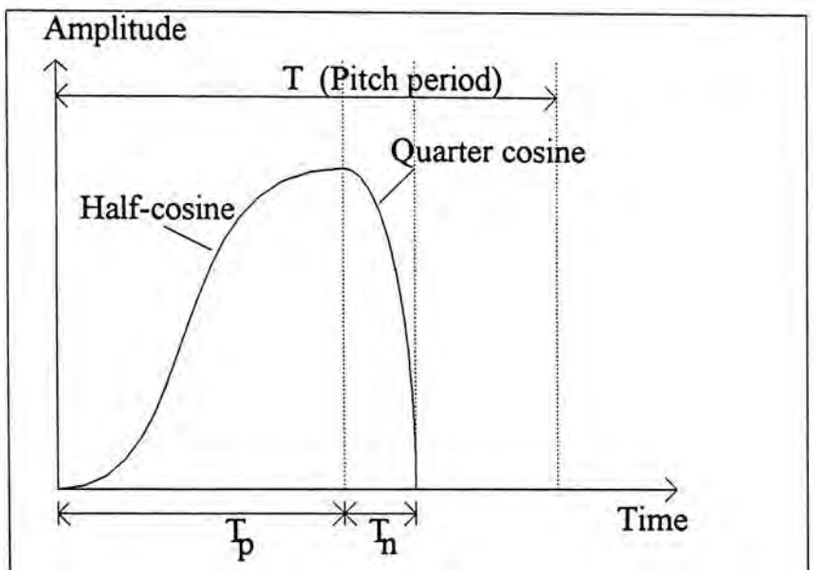


FIGURE 4.5 Glottal pulse shape proposed by Rosenberg

$$g(t) = \begin{cases} \frac{1}{2}(1-\cos(\frac{t}{T_p})) & \cdots\cdots\cdots\cdots\ 0 \le t \le T_p \\ \cos(\frac{t-T_p}{2T_n}\pi) & \cdots\cdots\cdots\cdots\ T_p < t \le T_p + T_n \\ 0 & \cdots\cdots\cdots\cdots\ t > T_p + T_n \end{cases} \qquad \text{(Eqaution 4.1)}$$

where g(t) is the glottal pulse shape. In implementation, T_p=0.4T, T_n=0.16T, T=1/F_0.

The noise source is a simple random number generator which produces random noises. A very broad spectrum can be obtained by analyzing the noise source output. There is an excitation mixer connecting the sound sources to the resonators. This mixer is controlled by the parameter SW. The noise source is coupled to the resonators when necessary (e.g. produce plosives, fricatives) (See Fig 4.4).

4.3.3. Digital formant resonator

The formant resonators of the formant synthesizer are second order IIR (Infinite Impulse Response) recursive filters described in Fig 4.6. Two parameters are used to specified the input-output characteristics of a resonator: the formant frequency F and the resonant bandwidth BW. Samples of output of a digital resonator y[n], are computed from the input sequence x[n] by:

$$y[n]=Ax[n]+By[n-1]+Cy[n-2]$$

where y[n-1] and y[n-2] are the previous two sampled values of the output sequence y[n]. The constants A,B and C are related to the resonant frequency F and the bandwidth BW of a resonator by the equations given by Gold and Rabinder **[21]**:

$$C= -e^{-2\pi BWT} .$$
$$B= 2e^{-\pi BWT}\cos(2\pi fT) .$$
$$A= 1-B-C .$$

Where T is the reciprocal of the sampling frequency of the speech synthesizer,
f is the formant frequency in Hz,
BW is the formant bandwidth in Hz.

The transfer function T(f) of the formant resonator is given by the equation:

$$T(f) = \frac{A}{1 - Bz^{-1} - Cz^{-2}} ,$$

where $z=e^{j2\pi fT}$, j is the square root of -1 and f is the frequency in Hz.

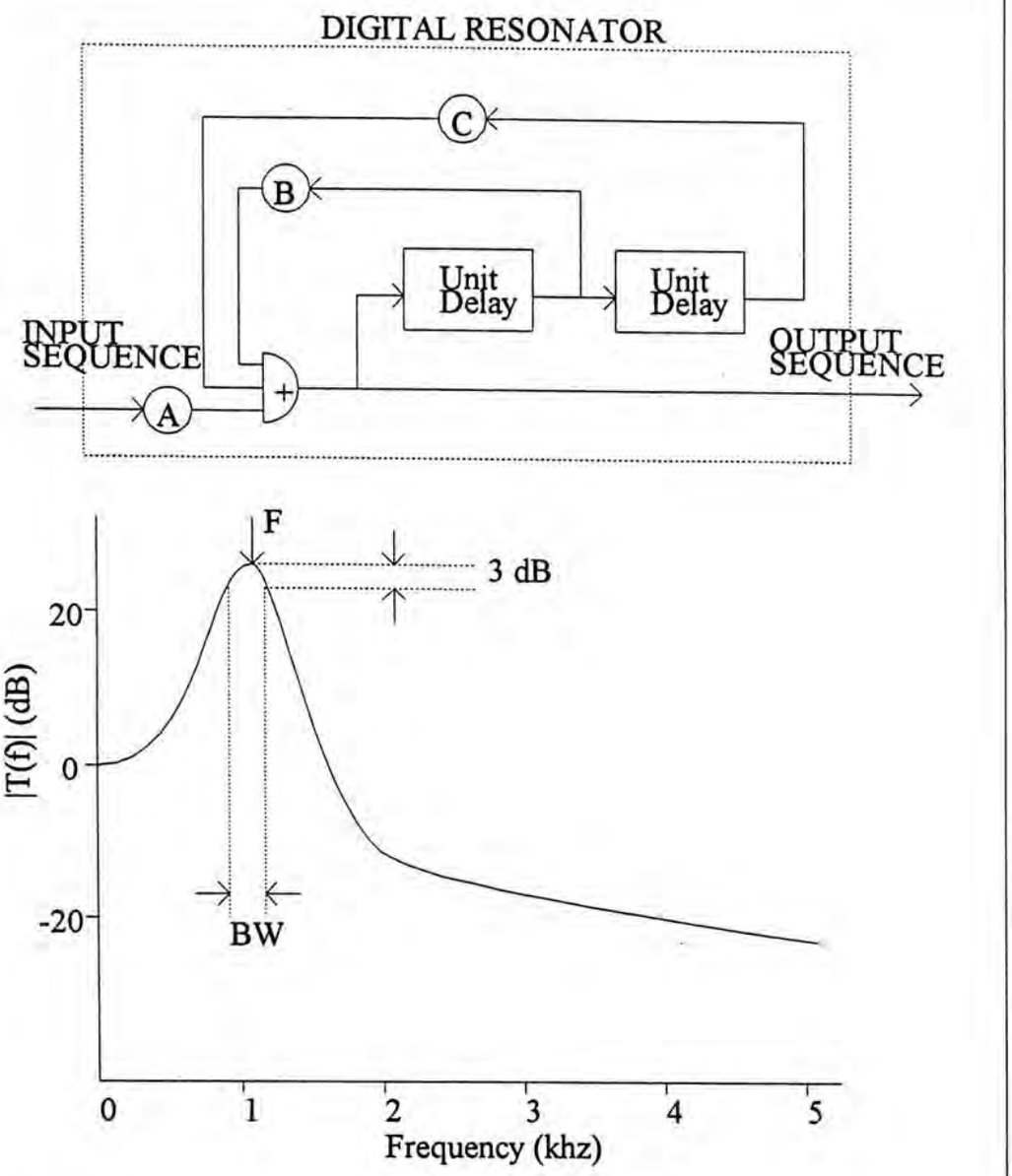


FIGURE 4.6 Block diagram and frequency response of a digital resonator

4.3.4. Lip radiation

The lip radiation module models the effect of directivity patterns of sound radiating from the lips as a function of frequency **[3]**. This transformation is simulated in the synthesizer by taking the first difference of the mixed output of the formant resonators:

$$p[n] = u[n] - u[n-1]$$

where p[n] is the output sequence of the synthesizer,

u[n] is the input sequence of the lip radiation module.

4.3.5. Mixing the outputs of formant resonators

With a parallel formant synthesizer, we are able to control the relative strength of the formant peaks by adjusting the separate formant amplitude parameters for the synthesizer. To synthesize human-like speech, the spectrum of the synthetic speech samples must be sufficiently close to human speech. In order to model the human speech spectrum successfully, Holmes **[22]** has suggested that the following properties should be presented in a parallel formant synthesizer:

(a) Each formant amplitude control should have its main effect in the frequency region near its own formant peak only.
(b) There should be a reasonably speech-like interpolation of spectral shape between the formant peaks.

Many early parallel synthesizers were programmed to add together the outputs of its formant resonators. In other cases, formant outputs were combined in alternating signs **[3]** (See Fig 4.7). For these synthesizers, condition (a) is not achieved because the overall frequency response of the parallel resonator system is, in many places, dependent on the combinations of skirt responses of several formant resonators **[22]**. It is also not uncommon that changing the formant amplitude of one formant peak (e.g. F2) may results in a substantial change in the overall spectral shape in regions far away from the peak (See Fig 4.8).

To alleviate this problem, Holmes **[22]** has suggested two arrangements: (1) *Differentiators* can be used to filter out the energy at low frequencies from resonators other than R1. (2) A *phase correction filter* is used to process the output of F1 before it is mixed with the outputs of other resonators. Holmes has successfully demonstrated that with these arrangements, the condition (a) and (b) can be satisfied in most cases.

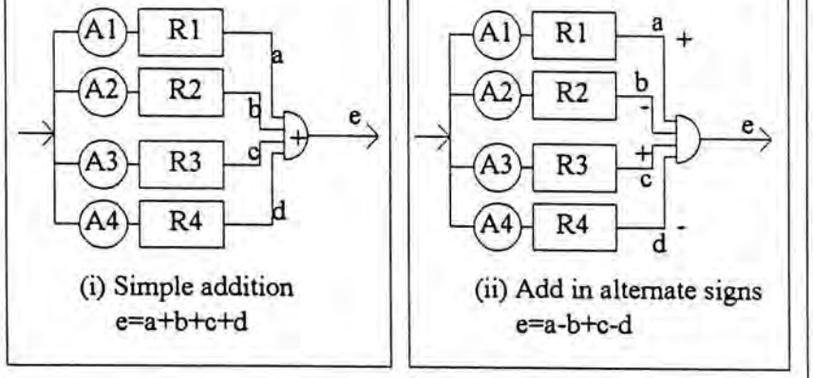


FIGURE 4.7 Methods of adding resonator outputs

In the implementation of LSYNTH, a *simple differentiator* is added to the output of each formant resonator R2, R3 and R4. The input-output characteristic of a simple differentiator can be expressed in the following equation:

$$y[n] = x[n] - x[n-1]$$

where x[n] is the n-th element of input sequence of the differentiator,

y[n] is the n-th element of output sequence of the differentiator.

To reduce the effect of R1 on other formant regions, a simple *averager* is added to the output of R1. The input-output characteristic of an averager can be expressed as:

$$y[n] = \frac{x[n] + x[n-1]}{2}$$

where x[n] is the n-th element of input sequence of the averager,

y[n] is the n-th element of output sequence of the averager.

Results of speech synthesis experiments have shown that with this configuration, condition (a) and (b) stated above can be roughly satisfied in LSYNTH. (See Fig 4.10)

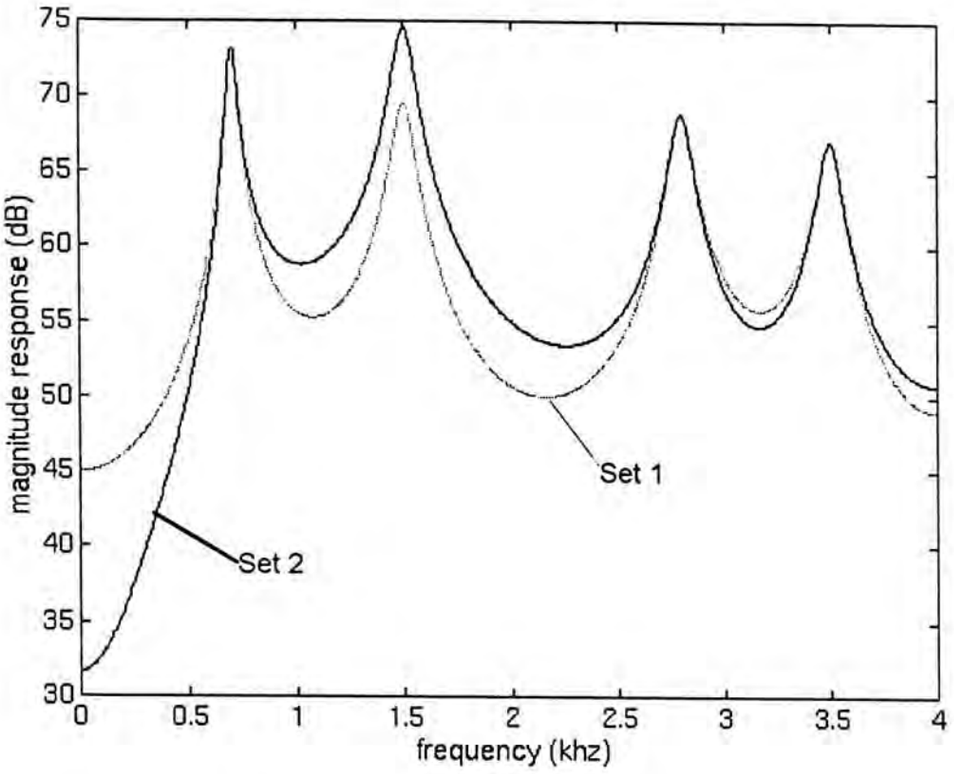


FIGURE 4.8 Effects on synthesizer response by varying formant amplitudes

A synthesizer having the resonator outputs connected in alternating signs is employed (See Fig. 4.7 (ii))

Gray line: Frequency response given input parameter set 1
Black line: Frequency response given input parameter set 2

The only difference between the 2 parameter sets is the 2nd formant amplitude (A2) of the 2nd set is 5 dB greater than A2 in the 1st set.

The 5 dB variation in A2 has caused significant variation on the overall frequency response outside the second formant peak region (especially in the low frequency region on the left of the first formant peak)

Parameter set 1				
F1: 700	F2: 1500	F3: 2800	F4: 3500	Unit: Hz
B1: 50	B2: 100	B3: 100	B4: 100	Unit: Hz
A1: 50	**A2: 45**	A3: 35	A4: 25	Unit: dB
Parameter set 2				
F1: 700	F2: 1500	F3: 2800	F4: 3500	Unit: Hz
B1: 50	B2: 100	B3: 100	B4: 100	Unit: Hz
A1: 50	**A2: 50**	A3: 35	A4: 25	Unit: dB

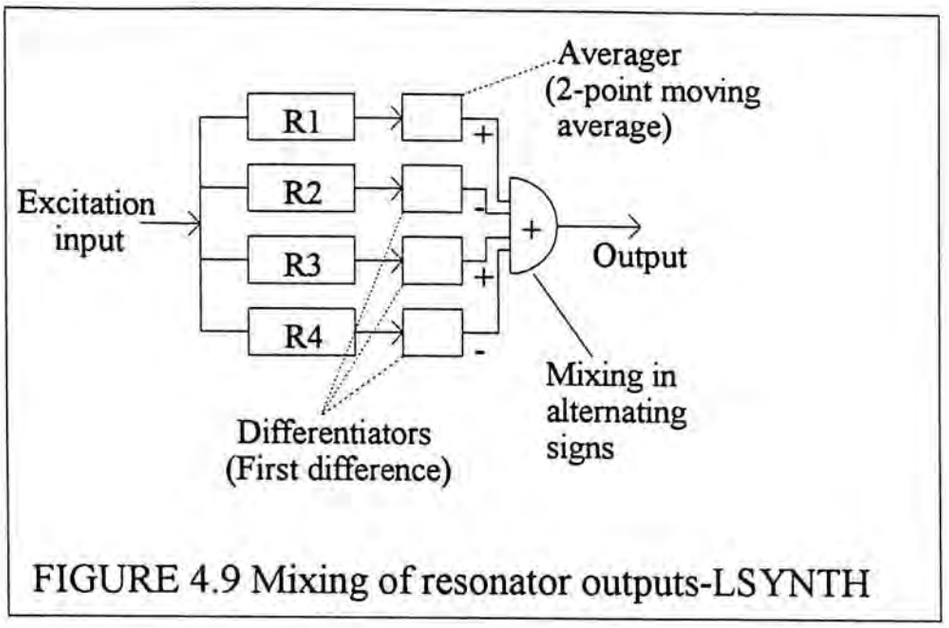


FIGURE 4.9 Mixing of resonator outputs-LSYNTH

4.3.6. Smoothing of excitation input to formant resonators

IIR filters are used extensively in the synthesizer. One of the characteristics of these IIR filters is that they may be unstable. They may cause arithmetic overflow in computation. Significant changes in parameter values (e.g. formant amplitude) in a very short time may introduce "clicks" and "burps" in the synthetic speech output which greatly degrades the quality of the synthetic speech output.

The rapid change in amplitude control of formant resonators during *voiced-unvoiced / unvoiced-voiced* transition may cause serious problems in computing the synthetic speech. After extensive observations in the synthesis process, it is found that "clicks" are most likely to be introduced if the following conditions are met:

(1) A glottal pulse has stretched over two or more consecutive speech frames.

(2) There is substantial variation in formant amplitude values across these speech frames.

In fact, an abrupt change in the excitation signal input would occurred if the two conditions stated above are both satisfied. In LSYNTH this problem is tackled by *linear interpolation of formant amplitude control* within a speech frame. A straight line is used to

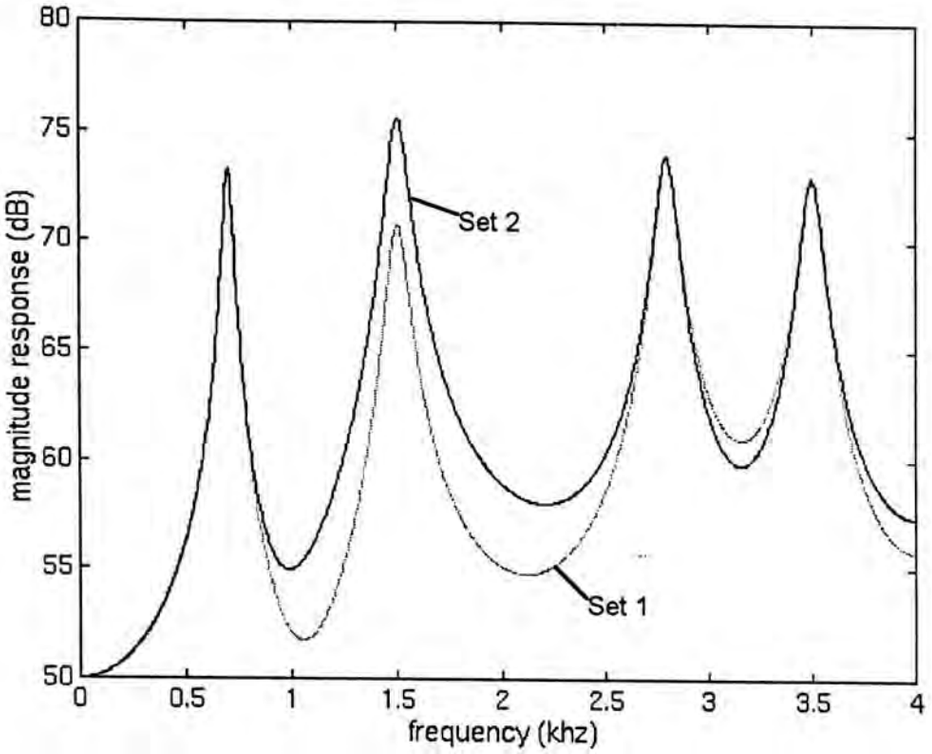


FIGURE 4.10 Effects on synthesizer response by varying formant amplitudes (in LSYNTH)

Gray line: Frequency response given input parameter set 1

Black line: Frequency response given input parameter set 2

The only difference between the 2 parameter sets is the 2nd formant amplitude (A2) in set 2 is 5 dB greater than A2 in set 1.

From the figure, it can be observed that the variation of A2 by 5 dB do not cause significant variation on the overall frequency response outside the region around the second formant peak.

Parameter set 1				
F1: 700	F2: 1500	F3: 2800	F4: 3500	Unit: Hz
B1: 50	B2: 100	B3: 100	B4: 100	Unit: Hz
A1: 50	**A2: 45**	A3: 35	A4: 25	Unit: dB
Parameter set 2				
F1: 700	F2: 1500	F3: 2800	F4: 3500	Unit: Hz
B1: 50	B2: 100	B3: 100	B4: 100	Unit: Hz
A1: 50	**A2: 50**	A3: 35	A4: 25	Unit: dB

join the *target values* of formant amplitude control in the *current frame* and the *next frame*. This line is then used as an *amplitude envelope* which amplitude modulates the excitation waveform (glottal pulse) inside that frame (See Fig 4.11). Mathematically, the input to the i-th digital resonator Ri (for i=1,2,3,4), $G_i(t)$, can be represented by the following equation:

$$G_i(t) = (Amp_i(t_0) + \frac{(Amp_i(t_1) - Amp_i(t_0))(t - t_0)}{t_1 - t_0})E(t) \quad ,$$

where $E(t)$ is the output of the excitation source (for $t_0 \le t \le t_1$),

$Amp_i(t_0)$ is the i-th formant amplitude in the current frame,

$Amp_i(t_1)$ is the i-th formant amplitude in the next frame,

t_0 is the staring time of the current frame,

t_1 is the starting time of the next frame.

Using this scheme, most abrupt changes caused by substantial variation in formant amplitudes can be smoothed out. Fig 4.12 shows the results of applying the linear interpolation scheme on the synthesis of the Cantonese diphthong /ui/.

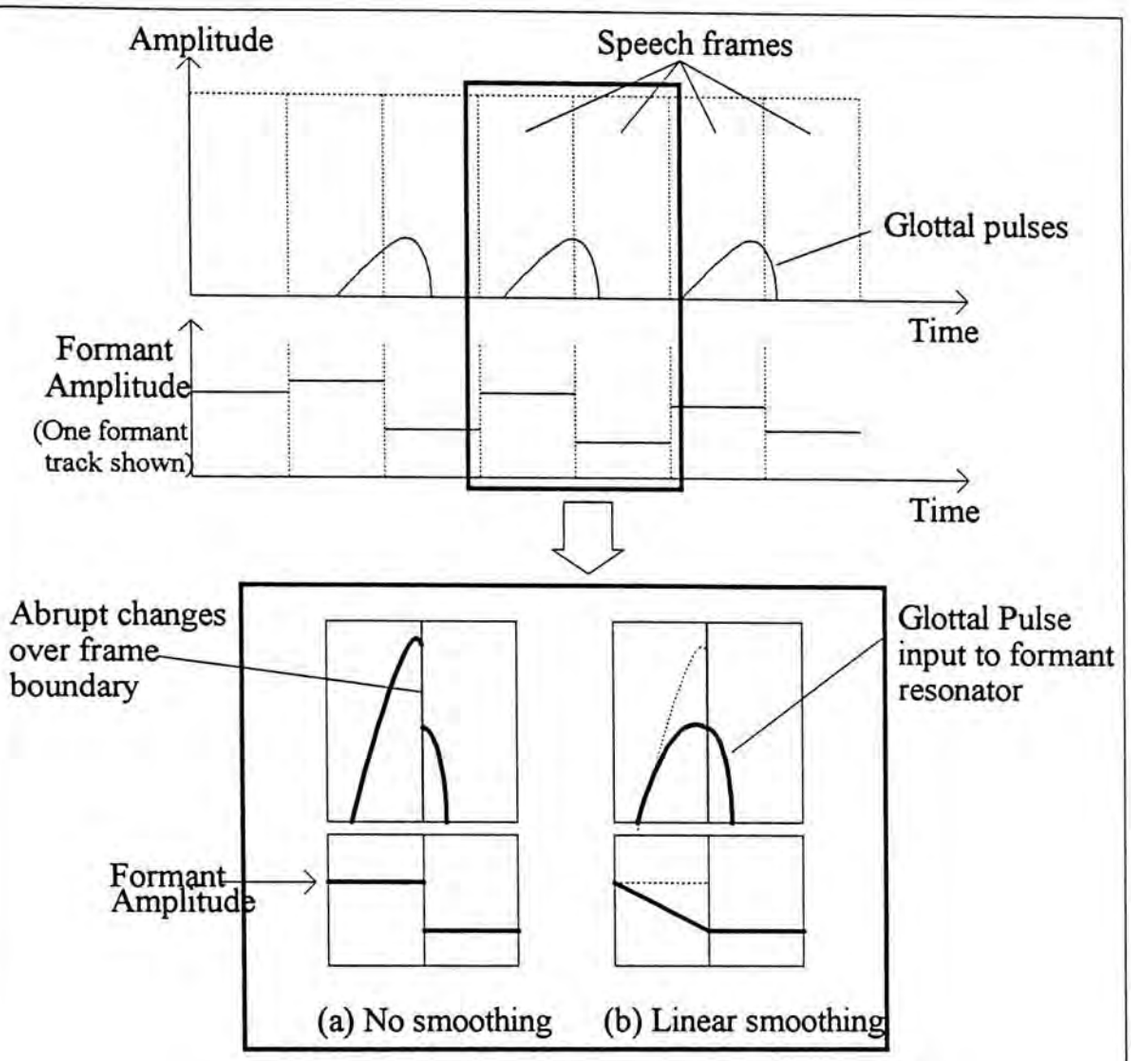


FIGURE 4.11 Linear interpolation of formant amplitude controls across frame boundaries

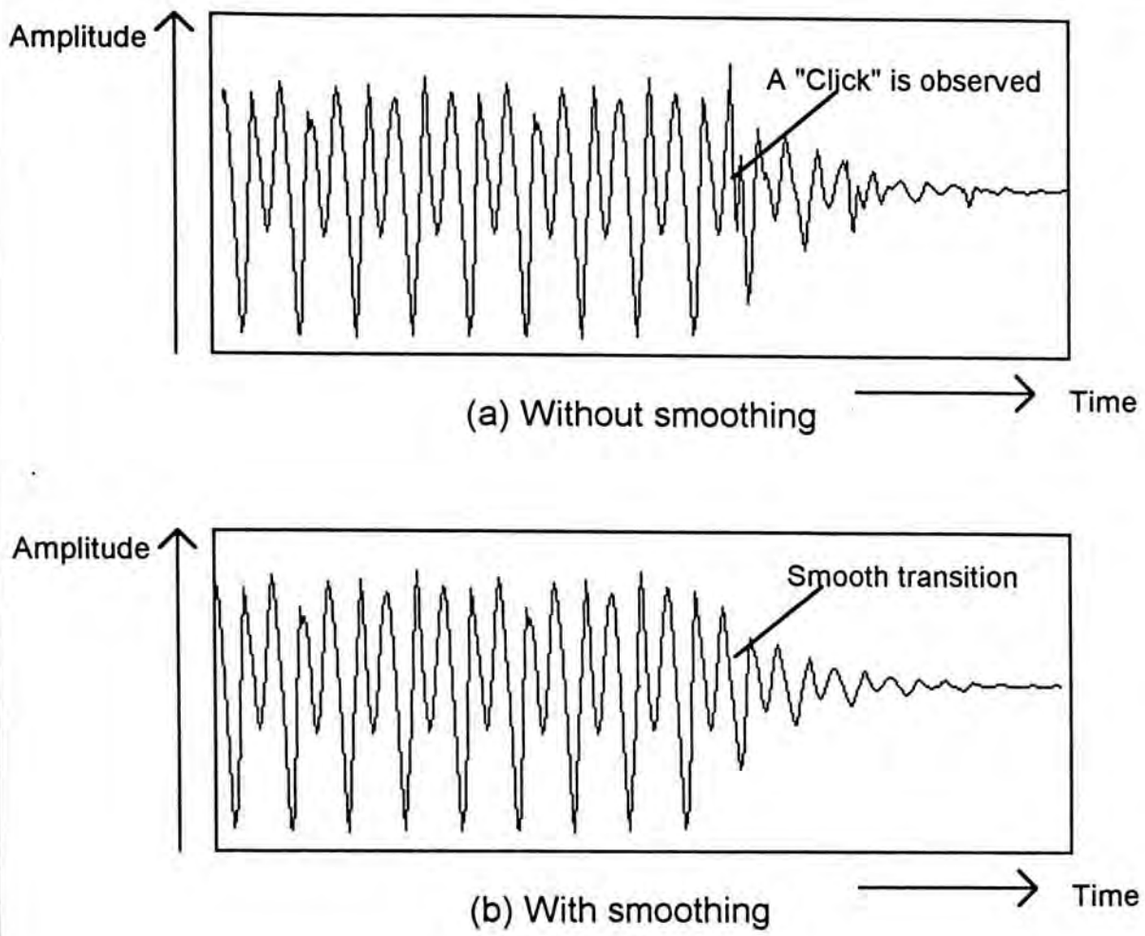


FIGURE 4.12 Effect of linear interpolation of formant amplitude controls

The Cantonese diphthong /ui/ is synthesized by LSYNTH:
(a) Without linear interpolation of formant amplitudes.
(b) With linear interpolation of formant amplitudes.

Unwanted fluctuations in synthetic waveform ("clicks") can be eliminated.

Chapter 5. Automatic formant parameter extraction for parallel formant synthesizers

5.1. Introduction

In order to generate synthetic speech from a formant speech synthesizer, a set of *time-varying speech parameters* is fed into a formant synthesizer which converts the parameters into a speech waveform. Formant synthesizers are capable of generating high-quality synthetic speech. This ability is fully demonstrated by the success of formant-based synthesizers such as *MITalk* **[3]**. However, accurate extraction of speech parameters is absolutely essential if high-quality speech output is desired.

A typical parallel formant synthesizer can accept more than 20 time-varying input parameters. Usually, these parameters are extracted from *natural speech samples*. However, accurate extraction of these parameters from natural speech is not an easy task. A number of techniques have been developed for estimation of parameters such as *formant frequencies*, *pitch* (fundamental frequency) and voicing (e.g. **[7],[24]**). In contrast, the estimation of other parameters, such as formant amplitude and bandwidth, is a more difficult task and has received considerably less attention over the years.

Although a number of algorithms has been developed to estimate formant amplitudes from natural speech, the extraction process may still require an extensive manual training process (e.g. Breen's MLP method **[24]**) or relies on a particular synthesizer configuration (e.g. The method proposed by Lowry et al. **[25]** is developed for the JSRU synthesizer **[26]**). Compared with formant amplitude measurement, the extraction of *formant bandwidths* from natural speech is a significantly harder task, and many existing bandwidth estimation algorithms are error-prone and lacking of accuracy **[27]**. Clearly, the lack of accurate and efficient analysis algorithms to extract these parameters is a major obstacle to the widespread use of formant synthesis in applications requiring high quality speech output.

Analysis-by-synthesis is a classical approach for many speech analysis systems **[11]**. It is based on comparing some *objective measures* (e.g. power spectrum) between the synthetic and natural speech samples and then modify the *input parameters* of the *target speech synthesizer* in order to produce a more "accurate" synthetic waveform. This process is repeated until the "error" between the two waveforms is below a certain level. Practically, analysis-by-synthesis techniques are often used to fine tune speech parameters obtained from traditional speech analysis algorithms (e.g. LPC analysis). This is because traditional speech analysis techniques often can't give parameter outputs that are accurate enough for synthesis purposes.

Based on the principle of analysis-by-synthesis, a simple but effective **feedback analysis system** for the automatic extraction of formant frequencies, amplitudes and bandwidths is proposed. This system, which is consisted of a *parallel formant synthesizer*, *a set of analysis tools*, and a *rule-based feedback control algorithm*, is implemented with partial success. A similar system has been proposed by Leung **[2]**. However, in his system, the spectral matching process is done manually. Another similar research was done by Appendino and Vivalda **[28]**. But their main interest is in deriving *more abstract synthesis parameters* (not physical parameters such as formant frequencies, pitch, etc.) for a synthesis-by-rule system.

5.2. The idea of a feedback analysis system

Human perception is by far still the most important factor in the evaluation of synthetic speech output quality. Researches in finding an *objective speech quality measure* are still in a rather primitive stage **[29]**. *Hand crafting* of parameter tables is still practiced extensively by synthesizer developers in order to adjust the speech output quality. In addition to aural perception, a typical speech synthesizer developer also uses some objective representation of speech, e.g. spectrograms, short time spectral envelope, etc. to help him/her to decide how to fine-tune the speech parameters. This system can be view as a *manual feedback control system*. The development of the **feedback analysis system** can be viewed as an attempt to control the update of speech parameters by a computer algorithm.

In general short time speech frames can be divided into 3 types, *voiced*, *unvoiced* and *mixed* (contains both voiced and unvoiced components). The feedback analysis system is designed to work with voice/mixed frames based on the idea of **spectral matching**. *Unvoiced sounds* are not intended to be processed by this system because spectral matching is not a very useful idea for unvoiced sounds.

5.3. Overview of the feedback analysis system

The *feedback analysis system* proposed is intended to extract *formant frequencies*, *bandwidths* and *amplitudes* (for the first four formants) from natural speech. A block diagram for the feedback analysis system is shown in Fig 5.1. Natural speech waveform is first segmented and pre-processed. Speech segmentation can be done by using fixed-sized frames or conducting a *pitch-synchronous analysis*. Each speech frame will be labeled as *voiced/mixed* or *unvoiced* by a pre-processing module. Then user is required to select a set of *voiced/mixed frames* from the natural speech frames. To analyze formant parameter variations inside a syllable (or a word), appropriate speech frames in different time-locations should be selected. Each selected speech frame will be processed by the following procedure: The frame is analyzed by a *parameter estimator* which gives a first estimate of various speech parameters (e.g. formant frequencies, bandwidths , amplitudes, etc.). A *target speech synthesizer* takes the input from the *parameter estimator* and produces a frame of synthetic speech. Then the *LPC-spectrum* of the synthetic speech and the natural speech are compared by a *spectral corrector*, resulting in adjustments of formant parameters which are then fed into the synthesizer again. The iteration will be stopped if the difference between the synthetic speech spectrum and the natural speech spectrum is below a preset tolerance level. The whole process will be *terminated* if all the selected frames are processed.

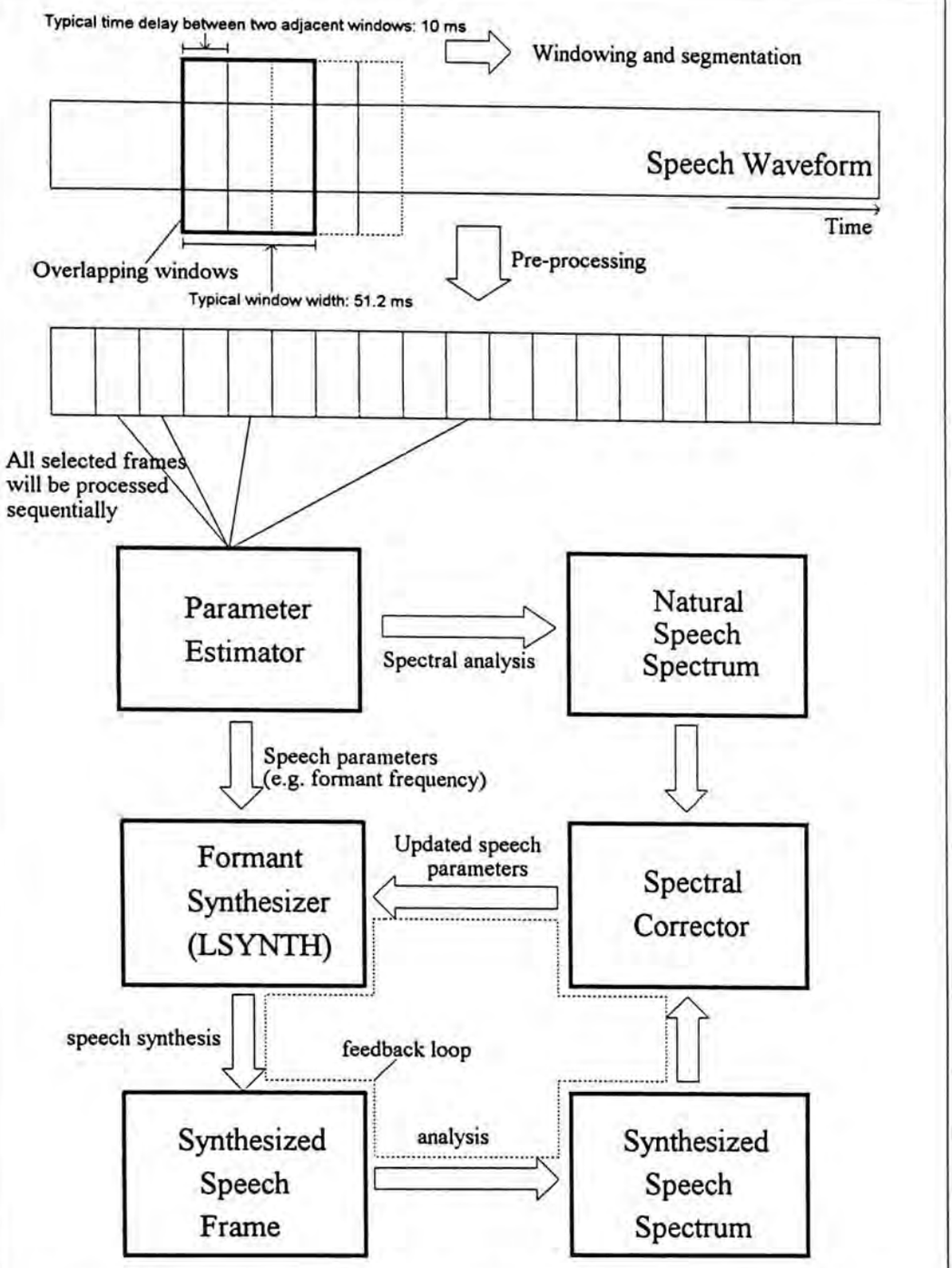


FIG 5.1 Block diagram of feedback synthesis system

5.3.1. Parameter estimator

The following parameters will be extracted by the *parameter estimator*: formant frequencies, bandwidths and amplitudes for the *first four formants*. In addition, the *fundamental frequency* of the speech frame will also be provided. This is done by an LPC-based formant analysis algorithm which is fully described in Chapter 2 (Section 2.4 and 2.5).

5.3.2. Spectral corrector

The main function of the *spectral corrector* is to compute, quantitatively, the differences between the synthetic speech spectrum and natural speech spectrum. The spectral difference obtained is then analyzed by a *spectral matching algorithm.* Decisions on how to modify the speech parameters are then given to the *target speech synthesizer*. Based on these modifications, a new synthetic speech frame, which should have less spectral distortion to the natural speech spectrum, will be produced. Therefore, the *spectral corrector* is the crucial element in the whole *feedback analysis system* (See Fig 5.2).

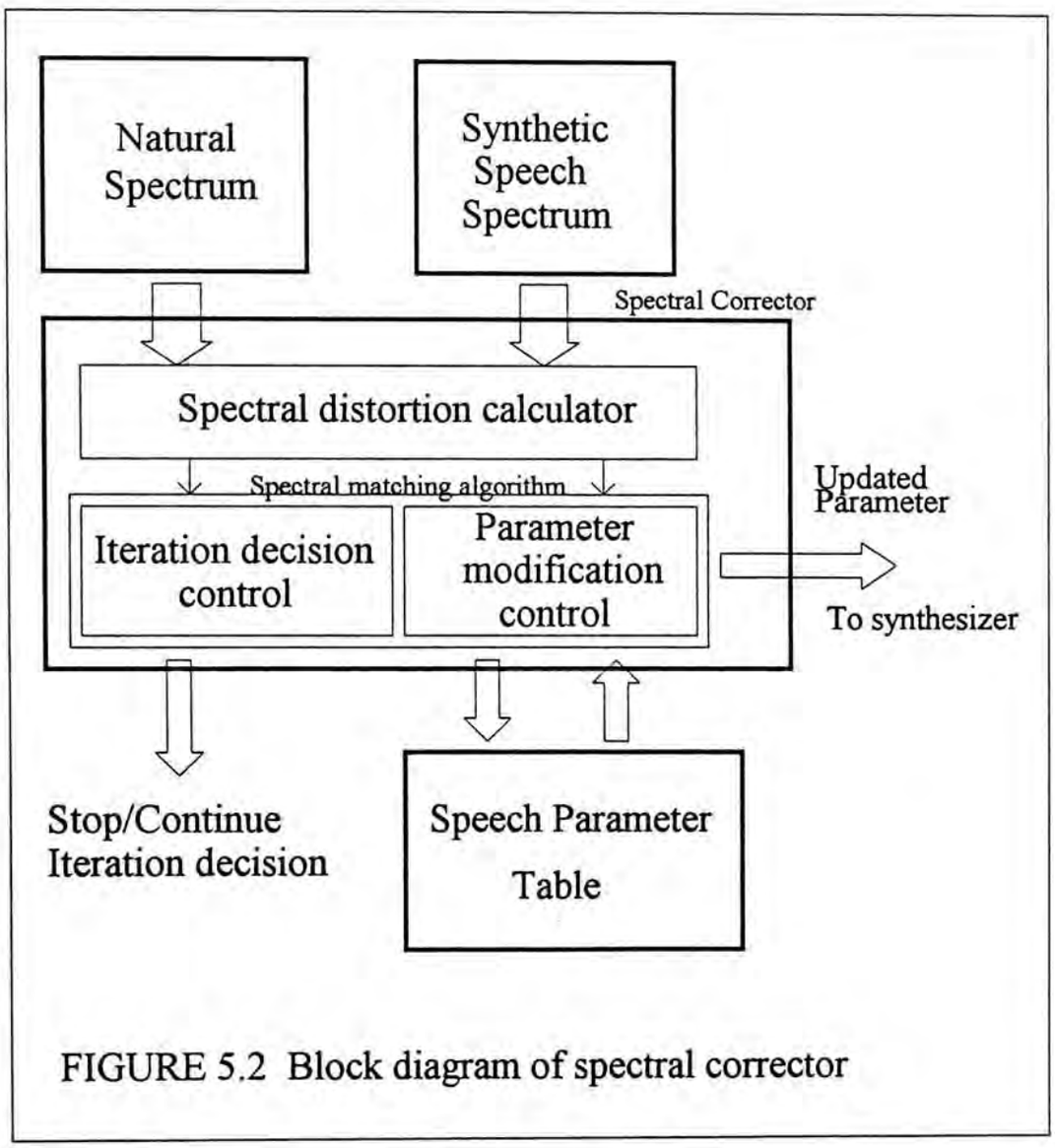


FIGURE 5.2 Block diagram of spectral corrector

5.3.3. Target speech synthesizer

A parallel formant synthesizer LSYNTH is used as the target speech synthesizer in this feedback analysis system. A full description of LSYNTH can be found in Chapter 4.

5.4. Iterative spectral matching algorithm

An iterative spectral matching algorithm is employed in the spectral corrector. This is because it is still extremely difficult to perform the task of spectral matching in one step. In each iteration, the spectra of natural speech and synthetic speech are compared and some *objective spectral difference measures* are evaluated. In the current implementation, the spectra of natural and synthetic speech are approximated by their LPC spectral envelopes. *Spectral matching rules* are then applied, using these spectral difference data, to determine how the speech parameters should be updated.

5.4.1. Pre-processing

The selected natural speech frame (sampled at 10 kHz) is first *pre-emphasized* by a filter $1-\mu z^{-1}$ (μ=1.0). Then a 512-point *Hamming window* is applied on the emphasized speech data before an 18-th order autocorrelation LPC analysis is being carried out. A 512-point discrete LPC spectral envelope, evaluated at equally spaced points on the frequency axis, is computed from the corresponding LPC coefficients (See Chapter 2, section 2.3.3). The same process is applied to the synthetic speech frame. The two spectral envelopes are then compared.

5.4.2. Spectral level adjustment

When comparing the spectrum of natural speech and synthetic speech, one has to make sure that the two spectral envelopes are located at approximately equal energy level. If this is not the case, the computed spectral difference between the two spectra will be great even if the

two spectral envelopes have a similar shape. Therefore, spectral level adjustment is necessary for proper comparison between the two spectra. One way of doing this is to shift one spectral envelope up or down by a constant amount of energy until the strongest formant peaks of the two spectra are equal in strength (See Fig 5.3a and Fig 5.3b).

5.4.3. Partition of formant regions

A typical voiced speech spectrum is characterized by the presence of formant peaks. In a spectral matching process, it is more important to match the spectral shape in *formant peak regions* than in other non-peak regions since most energies are carried in the formant peak regions. In this spectral matching algorithm, **FOUR** formant peak regions will be identified. Inside each formant peak region, the spectral difference of the synthetic and natural speech will be evaluated. Spectral matching rules are then applied to analyze these spectral difference data in order to determine how the input speech parameters should be updated. Each formant peak region is denoted by [$\mathbf{left}_i$,$\mathbf{right}_i$], indicating the region starts from frequency $\mathbf{left}_i$ and ends in frequency $\mathbf{right}_i$, for i=1,2,3,4. The procedure of marking the formant peak regions is described below: (all frequency quantities are measured in Hz)

(a) Label all *formant peaks* from the natural and synthesized speech spectrum

Let α_i be the $\mathbf{i^{th}}$ formant peak in the **natural** speech spectrum. i=1,2,3,4,...

β_j be the $\mathbf{j^{th}}$ formant peak in the **synthesized** speech spectrum. j=1,2,3,4,...

For most of the time, peaks can be located by a simple peak picking algorithm. However, other algorithms such as *zero-finding of LPC polynomials* can be employed if the former algorithm does not give a satisfactory result.

(b) Record the *input formant frequencies* to the target speech synthesizer, θ_i, i=1,2,3,4.

Notice that the observed formant peaks, β_is, are not necessary equal to their corresponding input formant frequencies θ_is, for i=1,2,3,4. The formant synthesizer sometimes cannot faithfully reproduce the peaks exactly at their desired locations.

(c) Peak selection

There may be more than 4 spectral peaks in the natural speech spectrum or synthetic speech spectrum. If this is the case, procedures must be done to select the *relevant formant peaks* and reject other *spurious peaks*.

For each input formant frequency θ_i, (i=1,2,3,4), seek the corresponding formant peaks $\mathbf{n_i}=\alpha_j$, and $\mathbf{s_i}=\beta_k$ such that:

$$|\theta_i-\alpha_{j'}| \geq |\theta_i-\alpha_j| \quad \text{for } \forall\, j'\neq j\ ,$$

$$|\theta_i-\beta_{k'}| \geq |\theta_i-\beta_k| \quad \text{for } \forall\, k'\neq k\ .$$

Where $\mathbf{n_i}$ (i=1,2,3,4) are the four formant frequencies in the natural speech frame,

$\mathbf{s_i}$ (i=1,2,3,4) are the four formant frequencies in the synthetic speech frame.

From the above selection criteria, it is clear that spectral peaks that is *nearest* to the input formant frequencies θ_is will be selected as formant peaks (See Fig 5.3c).

(d) Fixed boundary values calculation

Based on the input formant frequencies to the synthesizer, a rough partition of the spectrum can be made by defining a set of *fixed boundary values*.

The *left fixed boundary values* for formant region i, $\mathbf{fl_i}$ (i=1,2,3,4), is defined as:

$$\mathbf{fl_1}=0 \quad \text{[Hz]}\ ,$$

$$\mathbf{fl_i}=0.5(\theta_i+\theta_{i-1}) \quad \text{[Hz]} \quad \text{for } i=2,3,4.$$

The *right fixed boundary values* for formant region i, $\mathbf{fr_i}$ (i=1,2,3,4), is defined as:

$$\mathbf{fr_i}=0.5(\theta_i+\theta_{i+1}) \quad \text{[Hz]} \quad \text{for } i=1,2,3\ ,$$

$$\mathbf{fr_4}=4500 \quad \text{[Hz]}\ .$$

Under this arrangement, each input formant peak θ_i would be enclosed in the frequency range [$\mathbf{fl_i}$,$\mathbf{fr_i}$] for i=1,2,3,4 (See Fig 5.3d).

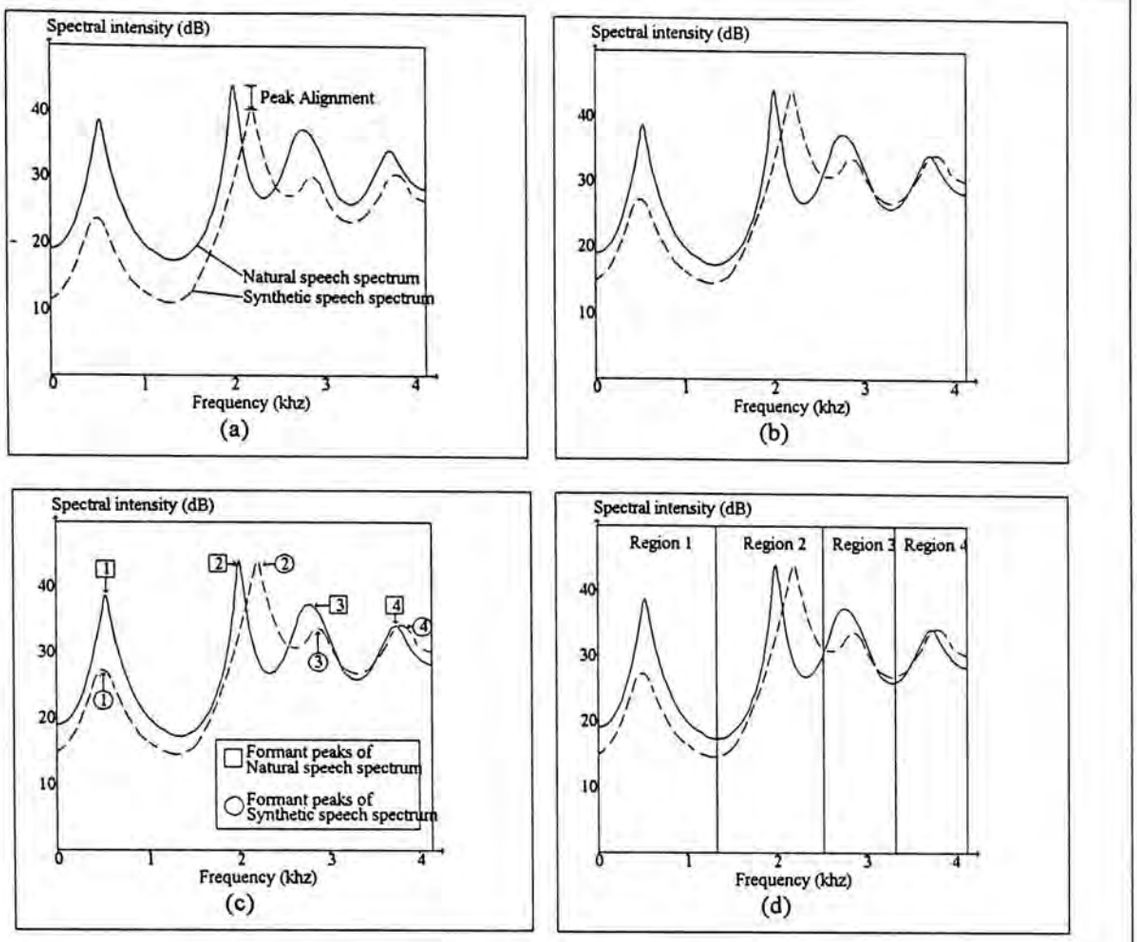


FIGURE 5.3 Partition of formant regions (I)

(a) Spectra of natural speech and synthetic speech before spectral level adjustment
(b) After spectral level adjustment
(c) Identification and selection of 4 formant peaks
(d) Rough partition of spectra by fixed boundary values

(e) Variable boundary values calculation

Smaller spectral peak regions can be obtained by discarding spectral regions with relatively low spectral intensity. This can be done by defining a set of variable boundary values for both the natural speech spectrum ($[\mathbf{nvl}_i , \mathbf{nvr}_i]$, i=1,2,3,4) and synthetic speech spectrum ($[\mathbf{svl}_i, \mathbf{svr}_i]$, i=1,2,3,4).

(I). Natural speech spectrum

The left variable boundary value for formant region i, $\mathbf{nvl_i}$ (i=1,2,3,4), is defined as:

$\mathbf{nvl_i}$=f if $SP_n(\mathbf{n_i}) - SP_n(f) = \delta$ and

$SP_n(\mathbf{n_i}) - SP_n(f') \le \delta$ for all f', $\mathbf{n_i}$>f'>f and $f \in [\, \mathbf{fl_i}, \mathbf{fr_i} \,]$.

Where $SP_n(F)$ is the spectral strength for frequency F in the natural speech spectrum. If there does not exist a f that satisfy the requirement, then $\mathbf{nvl_i}=\mathbf{fl_i}$.

The right variable boundary value for formant region i, $\mathbf{nvr_i}$ (i=1,2,3,4), is defined as:

$\mathbf{nvr_i}$=f if $SP_n(\mathbf{n_i}) - SP_n(f) = \delta$ and

$SP_n(\mathbf{n_i}) - SP_n(f') \le \delta$ for all f', $\mathbf{n_i}$<f'<f and $f \in [\, \mathbf{fl_i}, \mathbf{fr_i} \,]$.

Again, if there does not exist a f that satisfy the requirement, then $\mathbf{nvr_i}=\mathbf{fr_i}$.

(II) Synthetic speech spectrum

The left and right variable boundary values for each formant region i, $\mathbf{svl_i}$ and $\mathbf{svr_i}$, (i=1,2,3,4), are calculated using a similar algorithm as in (I) except the spectral strength values are taken from the synthetic speech spectrum. For implementation purposes, δ is set to **10.0 dB** in (I) and (II). Notice that the width of a formant peak region marked by a pair of variable boundary values is simply the δ-**dB** bandwidth of the corresponding formant peak (See Fig 5.3e and Fig 5.3f).

(f) Marking of formant peak regions

The starting and ending frequencies of each formant peak region i, i=1,2,3,4 are formed by merging the corresponding spectral peak regions of the two spectra.

$\mathbf{left_i} = \min(\mathbf{svl_i}, \mathbf{nvl_i})$,

$\mathbf{right_i} = \max(\mathbf{svr_i}, \mathbf{nvr_i})$.

There are many different strategies of combining the spectral peak regions of natural speech spectrum and synthetic speech spectrum. The above strategy always results in broader formant peak regions. After all, it is important to include both $\mathbf{n_i}$ and $\mathbf{s_i}$ in each formant peak region i (i=1,2,3,4) (See Fig 5.3g).

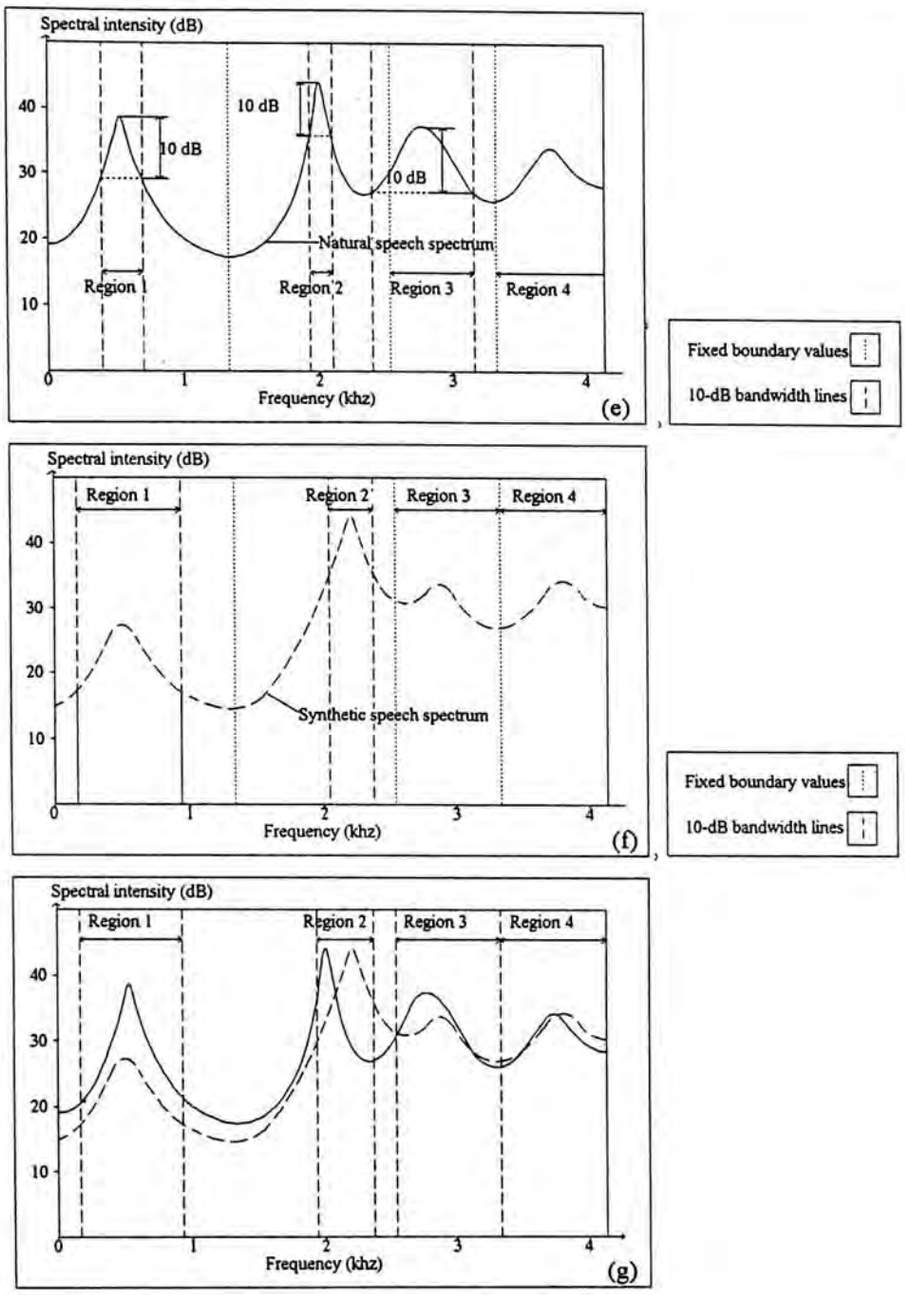


FIGURE 5.3 Partition of formant regions (II)

(e) Marking of variable boundary values (natural speech spectrum)
(f) Marking of variable boundary values (synthetic speech spectrum)
(g) Merging of formant peak regions from the two spectra

5.4.4. Objective spectral difference measures

There are altogether FIVE objective spectral difference measures used in the system, these five types of error measures will be evaluated at every iteration and their descriptions are stated below:

1. Amplitude difference

It is an arithmetic difference between spectral strengths of same-order formant peaks of natural speech spectrum and synthetic speech spectrum:

Adiff(i)=$SP_n(\mathbf{n}_i)$-$SP_s(\mathbf{s}_i)$ for i=1,2,3,4.

Where $SP_n(f)$ is the spectral strength of frequency f in the natural speech spectrum,

$SP_s(f)$ is the spectral strength of frequency f in the synthetic speech spectrum.

2. Frequency difference

It is also an arithmetic difference between the formant peak frequencies

Fdiff(i)=$\mathbf{n}_i$-$\mathbf{s}_i$ for i=1,2,3,4.

3. Bandwidth difference

The 3-dB bandwidth of the i-th formant peak in natural speech is defined as:

$$\mathbf{Bw}_{i,natural} = \mathbf{F}_{i,right} - \mathbf{F}_{i,left}$$

where the calculation of $\mathbf{F}_{i,right}$ and $\mathbf{F}_{i,left}$ is similar to that of variable boundary values $\mathbf{nvl}_i$ and $\mathbf{nvr}_i$ (see section 5.4.3 (e)) except δ is set to 3 dB.

The 3-dB bandwidth of the i-th formant peak in synthetic speech is defined as:

$$\mathbf{Bw}_{i,synthesized} = \mathbf{f}_{i,right} - \mathbf{f}_{i,left}$$

where the calculation of $\mathbf{f}_{i,right}$ and $\mathbf{f}_{i,left}$ is similar to that of variable boundary values $\mathbf{svl}_i$ and $\mathbf{svl}_r$ (see section 5.4.3 (e)) except δ is set to 3 dB.

The bandwidth difference for each region i, (i=1,2,3,4) is taken as:

$$Bdiff(i)=Bw_{i,natural} - Bw_{i,synthesized}.$$

Notice that we sometimes cannot find the 3-dB bandwidth for some badly-shaped formant peaks. If this is the case then a *flag* will be set to signify the presence of such ill-formed peaks. And the bandwidth difference will not be calculated in such cases (See Fig 5.4).

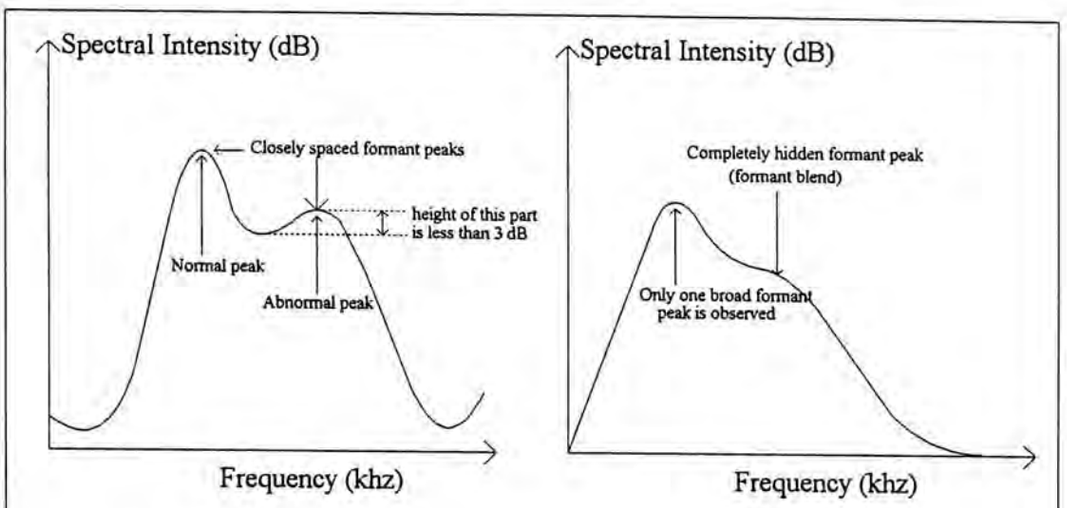


FIGURE 5.4 Some badly-shaped formant peaks

4. Regional distortion measure

The spectral distortion in each formant region is computed by the following equation:

$$Rdiff\ (i) = \frac{1}{k_i} \sum_{f=left_i}^{right_i} \left| SP_n(f)^2 - SP_s(f)^2 \right| ,$$

where k_i is the width of the formant region i (i=1,2,3,4), $k_i=right_i-left_i+1$,

$SP_n(f)$ is the spectral strength of frequency f in natural speech spectrum,

$SP_s(f)$ is the spectral strength of frequency f in synthetic speech spectrum.

5. Global distortion measure

A weighted difference between the natural speech spectrum and synthetic speech spectrum is used as the global distortion between the two spectra:

$$Totaldiff = \sum_f w_f \left| SP_n(f)^2 - SP_s(f)^2 \right|,$$

where w_f is a *weighting factor* which have the following values:

w_f =3.0 if f is in formant region 1,

=2.0 if f is in formant region 2,3 or 4,

=1.0 otherwise.

The weighting factors are determined experimentally after many tests of the feedback analysis system. By summing the differences of the *squares* of the spectral intensities, spectral differences in spectral peak regions will be emphasized in the calculation of spectral distortion.

5.4.5. Spectral matching strategy

There are altogether *twelve parameters* that can be modified by the spectral corrector: 4 formant frequencies, 4 formant bandwidths and 4 formant amplitudes. In order to reduce the complexity of the parameter update algorithm, the following rules are observed:

1. Only **One** parameter will be updated in each iteration.
2. In each iteration, the selected parameter will be increased/decreased by an *integral* number of steps. The physical size of a step for different types of parameters is listed below:

Parameter Type	Step size
Formant frequency (F1-F4)	10 Hz
Formant bandwidth (B1-B4)	5 Hz
Formant Amplitude (A1-A4)	1 dB

Table 5.1 Stepsize of various formant parameters

The complexity of the spectral matching algorithm also depends on the *spectral shaping ability* of the *target speech synthesizer*. A relatively simple spectral matching strategy can be employed if the following conditions are satisfied by the target formant synthesizer.

(a) Formant parameters (e.g. formant amplitude, bandwidth) should only exercise their influence inside their own formant region. They should not cause significant variation of the spectrum shape outside their own formant region.
(b) Normally, an *increase* in formant amplitude should *raise* the spectral height of the corresponding formant peak and vice versa.
(c) Normally, an *increase* in formant bandwidth should *increase* the width of the corresponding formant peak and vice versa.

A formant synthesizer having condition (a) satisfied is favorable since we are sure that the spectral matching process inside a formant peak region will not cause a previously matched formant peak region to become mismatched again. While conditions (b) and (c) can ensure a gradual reduction of difference between the natural speech spectrum and the synthetic speech spectrum in successive iteration steps of the spectral matching algorithm. If conditions (b) and (c) are not satisfied, the parameter update algorithm will be more complex.

The *target speech synthesizer* LSYNTH in this feedback analysis system is specially designed so that the above three conditions are generally satisfied (See section 4.3.5 for details). In fact, other target speech synthesizers can be used as long as they satisfied the three conditions stated above. Based on these nice properties of the target speech synthesizer, a relatively simple spectral matching strategy is devised. Basically, the following steps will be taken sequentially by the spectral matching algorithm.

a) The height of the spectral peaks will be aligned first (matching formant amplitudes).
b) The peak locations are then aligned (matching formant frequencies).
c) Finally, the shape of the formant peaks are adjusted (matching formant bandwidths).

5.4.6. Matching formant amplitudes

The spectral strengths of the four formant peak pairs are matched *in turns*. Firstly, the spectral strengths of the first formant peaks (F1) are aligned, by using an iterative algorithm which is described in the next paragraph. Then the same algorithm is applied to match the second (F2), third (F3) and fourth (F4) formant peaks sequentially. The steps in matching formant peak strengths in each formant region are:

1) Compute the spectral shape of the synthetic speech and obtain the amplitude difference measure for the formant region.
2) If the amplitude difference is smaller than one step (1 dB), the process for this formant region is terminated. The process for each formant region will also be terminated if the number of iterations exceeds a preset limit.
3) Using the amplitude difference measure to get a correction factor by *table lookup* (See Fig 5.5 for an example).
4) A new formant amplitude estimate is produced by adding the correction factor to the old formant amplitude estimate. Jump to step (1) for next iteration.
5) The iteration will also be stopped if the global distortion measure is below a preset tolerance level. In this case the spectral matching algorithm will be terminated.

5.4.7. Matching formant frequencies

Although the LPC-based formant tracking algorithm mentioned in section 2.4 can give quite accurate formant frequency estimates, there may still be minor difference between the corresponding formant peak frequencies in synthetic and natural speech spectra. The algorithm of matching peak frequencies is very much similar to that of matching formant amplitudes: the formant peak regions are processed in turns (in the order of F1-F2-F3-F4), and the formant frequency difference measure is used to produce a new formant frequency estimate through a table lookup method.

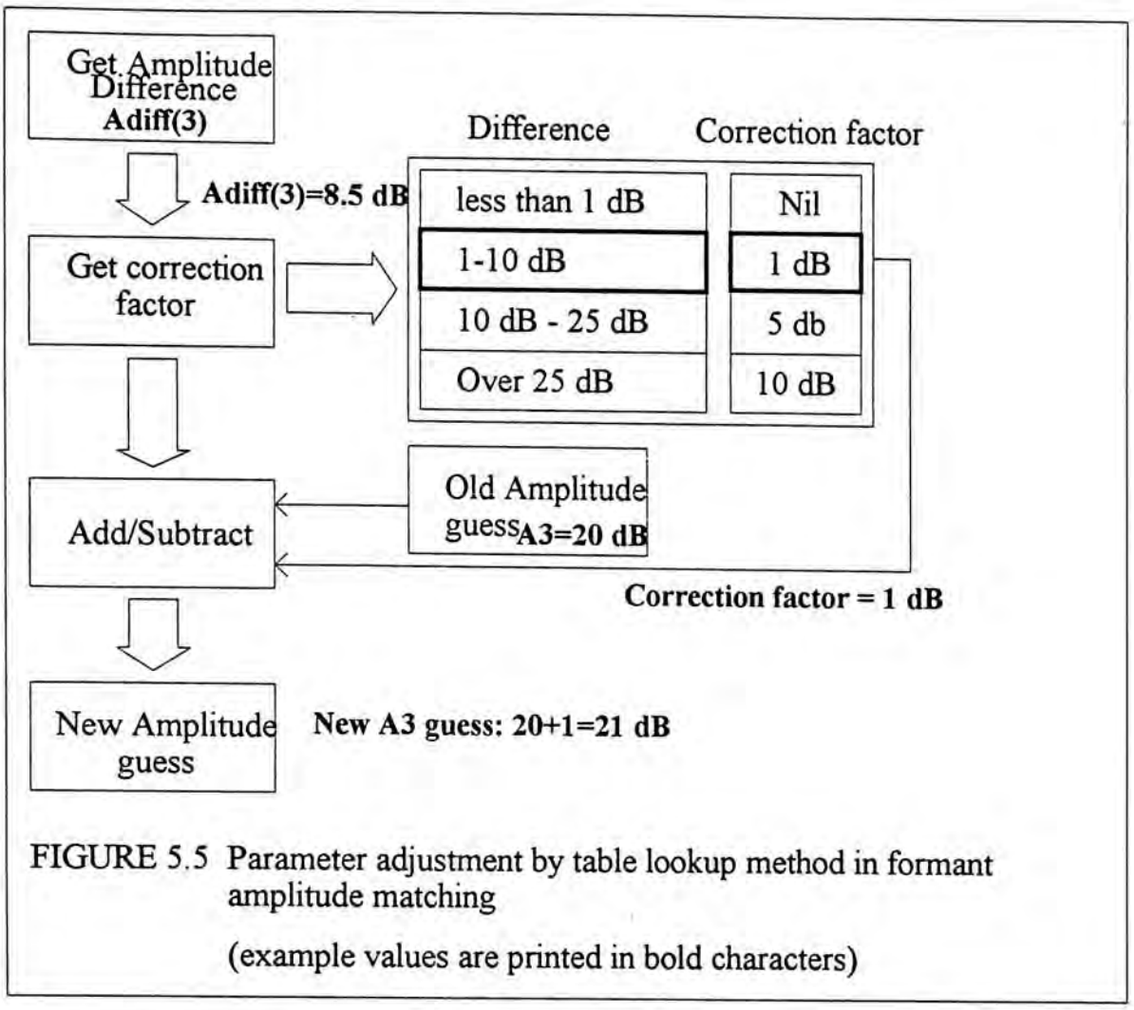


FIGURE 5.5 Parameter adjustment by table lookup method in formant amplitude matching

(example values are printed in bold characters)

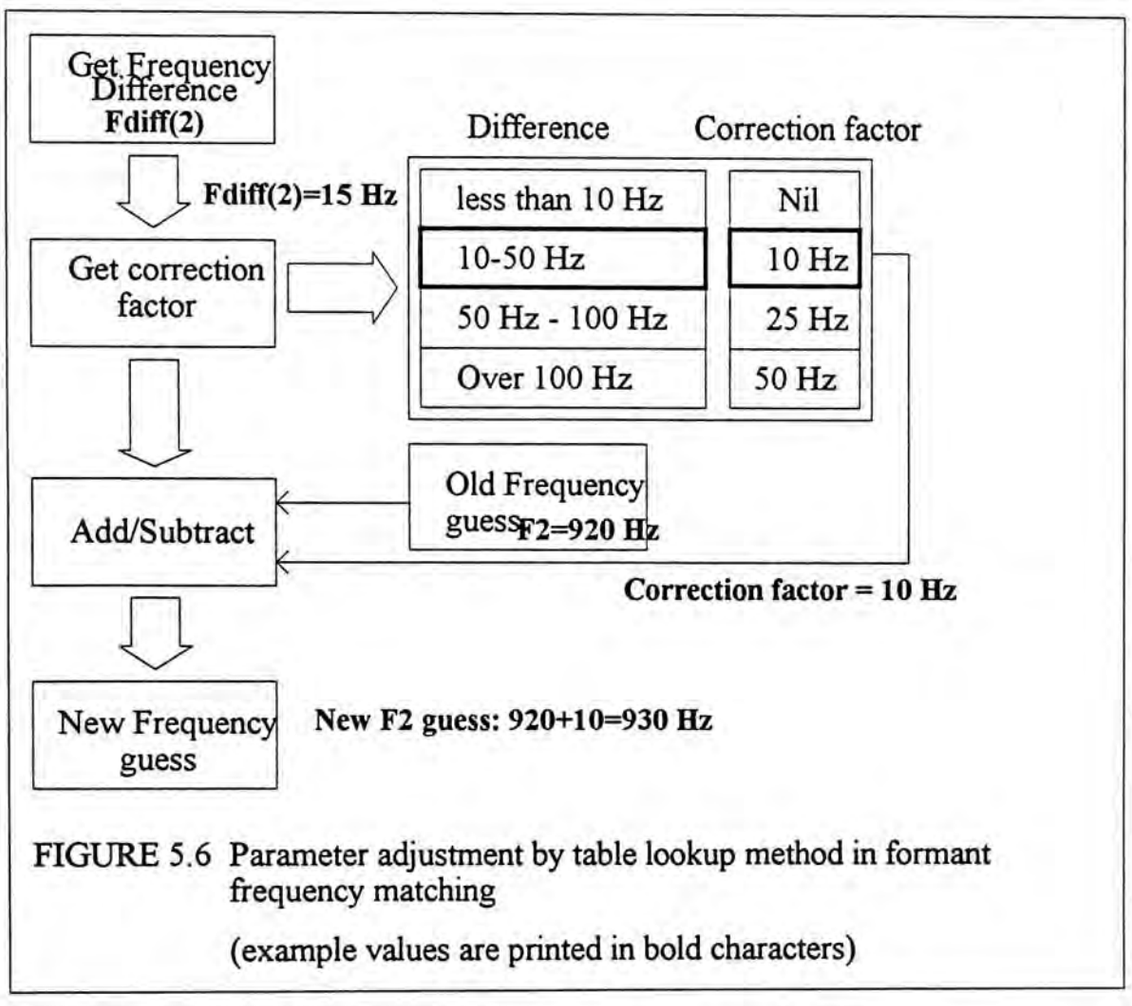


FIGURE 5.6 Parameter adjustment by table lookup method in formant frequency matching

(example values are printed in bold characters)

5.4.8. Matching formant bandwidths

Each formant region will be processed in turns (in the order of F1-F2-F3-F4), similar to that of matching formant amplitudes. For each formant peak region, the formant bandwidth estimate is updated in each iteration by the following procedure:

1) The *regional spectral distortion* of the current formant peak region is recorded.
2) If *bandwidth difference measure* for that formant region is available, we can obtain a new bandwidth estimate by using a table lookup method similar to that of formant amplitude matching algorithm. Otherwise the iteration for this formant region is *terminated.*
3) After adjusting the bandwidth, the spectral strengths of the formant peaks may need to be aligned by using a formant amplitude matching algorithm.
4) After the formant peaks are aligned, the *regional distortion measure* is noted again.
5) If the new regional distortion is smaller than the old regional distortion, then the bandwidth adjustment is regarded as success and the process will jump back to step (1). Otherwise, the adjustment is regarded as a failure. Old bandwidth and amplitude estimates are restored and the iteration for this region is terminated.
6) The iteration will also terminated, if the number of iterations has exceeded a preset limit, or the global distortion measure is below a preset limit.

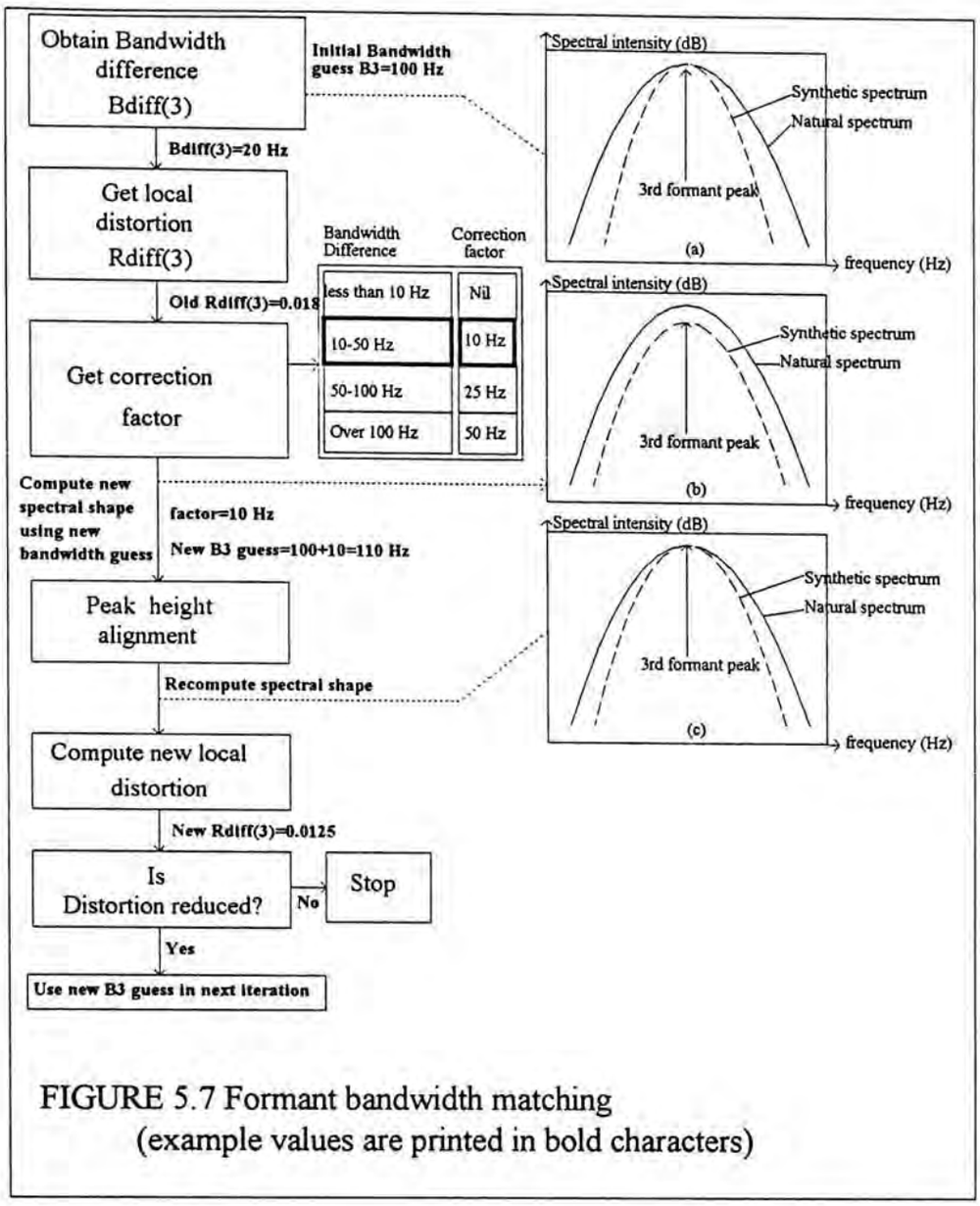


FIGURE 5.7 Formant bandwidth matching
(example values are printed in bold characters)

5.5. Results and discussions

Four pure vowels /a:/, /i:/, /u/ and /æ/ are used to test the performance of the algorithm. First, natural speech samples are recorded for the four vowels. Based on these natural speech samples, *two* synthetic speech spectra are produced for each vowel, by using two different sets of formant parameters obtained by the following methods:

a) An LPC-based formant analysis algorithm (described in chapter 2, section 2.4).
b) By using the feedback spectral matching algorithm, using the formant parameters obtained by method (a) as the initial guess.

For each vowel, synthetic speech spectra obtained by method (a) and (b) are compared with to their corresponding natural speech spectrum. Fig 5.6 shows the synthetic spectra produced. It is observed that the synthetic speech spectrum obtained by method (b) is generally well matched with the natural spectrum. While for method (a), the formant amplitude parameters are often not accurate enough. In some cases, such as /u/ and /a:/, substantial errors are presented in the estimation of the second formant amplitude. The second formant peak of /u/ is even missed in the synthetic speech spectrum. However, by using the feedback spectral matching algorithm, the second formant peak of /u/ can be recovered quite successfully. It is also observed that the spectral matching algorithm can fit the location (formant frequency) and height (formant amplitude) of the formant peaks sufficiently close to the natural speech spectrum.

Using the parameters obtained by methods (a) and (b), two sets of synthetic speech samples for the four vowels are produced. Informal listening tests have indicated that the sound quality of /u/ obtained by method (b) is better than the /u/ obtained by using method (a). However, the sound quality of other vowels are roughly the same. This results demonstrated that although the feedback algorithm is an effective tool in obtaining more accurate formant amplitudes, it does not guarantee an improvement of the synthetic speech sound quality.

The reason why a more spectrally fitted LPC spectral envelope does not always imply a better sound quality may be the "quality" of sound does not depend on its power spectrum only. There are other factors, such as the modeling of glottal excitation **[30]** and pitch contour fluctuations **[31]** that also influence the perceptual quality of synthetic speech. However, how these factors affects the synthetic speech quality is still not fully understood.

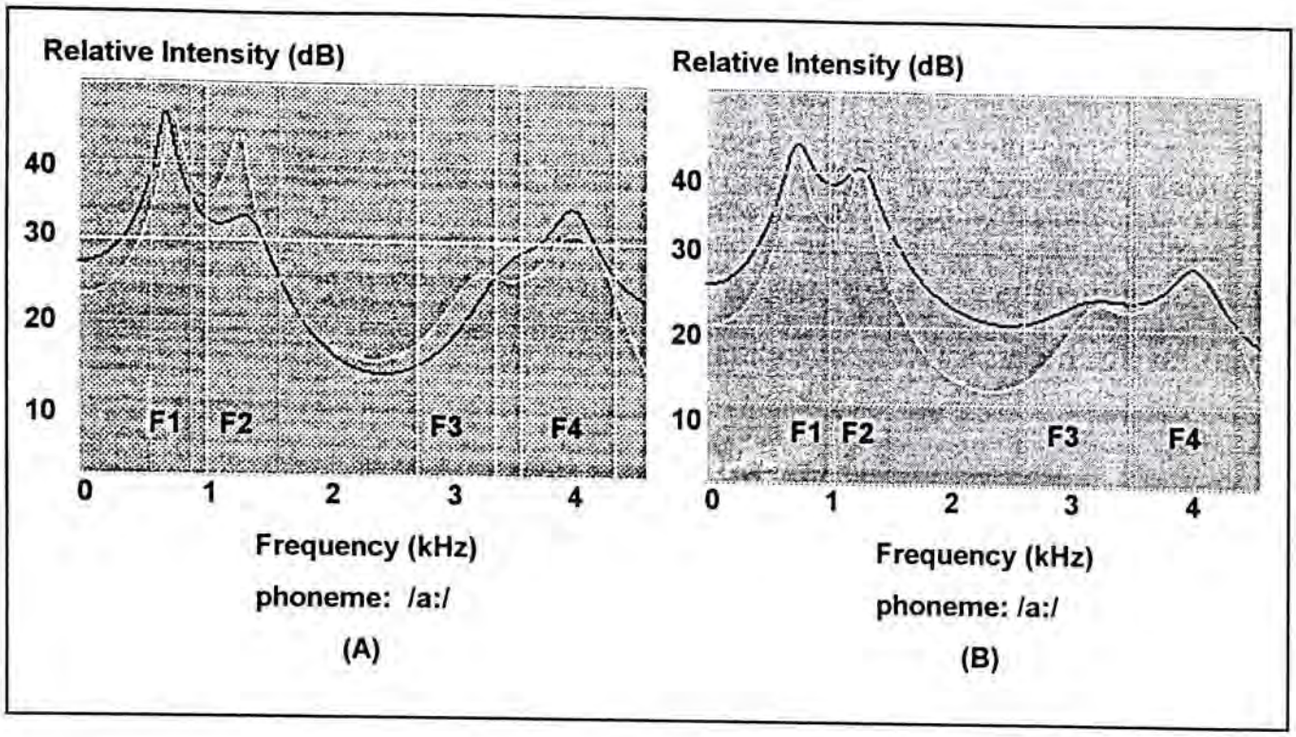
The present spectral matching algorithm is still *time consuming* since a speech synthesizer (LSYNTH), which has a rather high computational complexity, must be called in each iteration. And a typical spectral matching process may trigger 30-50 iterations (See Table 5.2). Moreover, the iterative spectral matching algorithm may run into difficulty in certain

unfavorable conditions like: (a) The natural speech spectrum differs substantially from the synthetic speech spectrum before the first iteration, (b) Abnormal spectral shapes, like a *formant blend*, is presented in the natural or synthetic speech spectrum. In this case, the formant bandwidth matching process cannot be done. Manual adjustment of formant parameters may be necessary when condition (a) or (b) is met.

Vowel	No of Iterations
/a:/	34
/æ/	31
/i:/	46
/u/	37

Table 5.2. No of iterations needed for the four vowels

In order to reduce the number of iterations needed to achieve an acceptable spectral fit, several suggestions may be useful. These suggestions include: (i) Update more than one formant parameter in each iteration, (ii) Use a dedicated hardware to implement the digital filter system inside LSYNTH, (iii) Derive a set of more "intelligent" rules to replace the current spectral matching rules. In addition, more attention should be made to formant bandwidth matching since the current bandwidth matching algorithm can work on well-shaped formant peaks only.



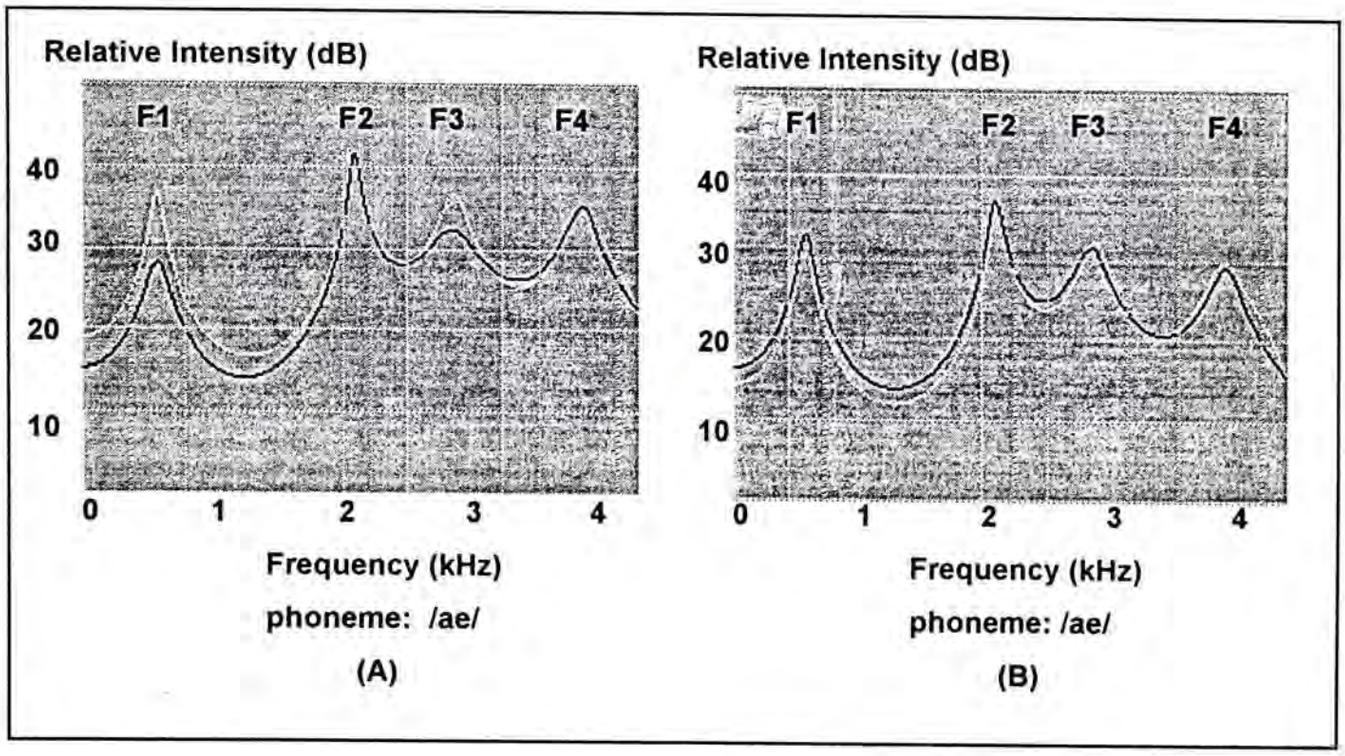
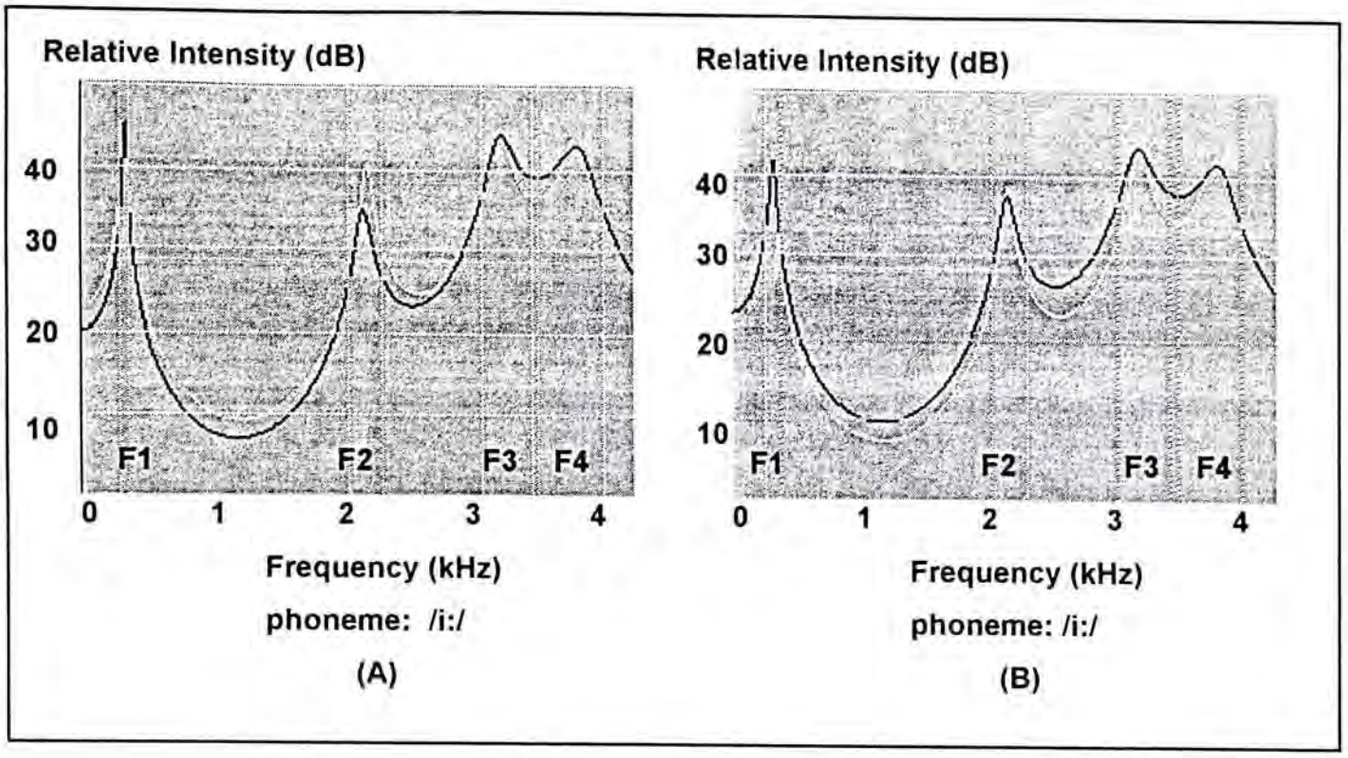


FIGURE 5.8. Comparisions of LPC spectra of natural speech (in white curve) and synthetic speech (in black curve). For graphs marked with (A), the synthesis parameters are obtained by LPC formant analysis. For graphs marked with (B), the synthesis parameters are obtained by the feedback analysis system. The four formant regions of the spectra are marked by white vertical lines.



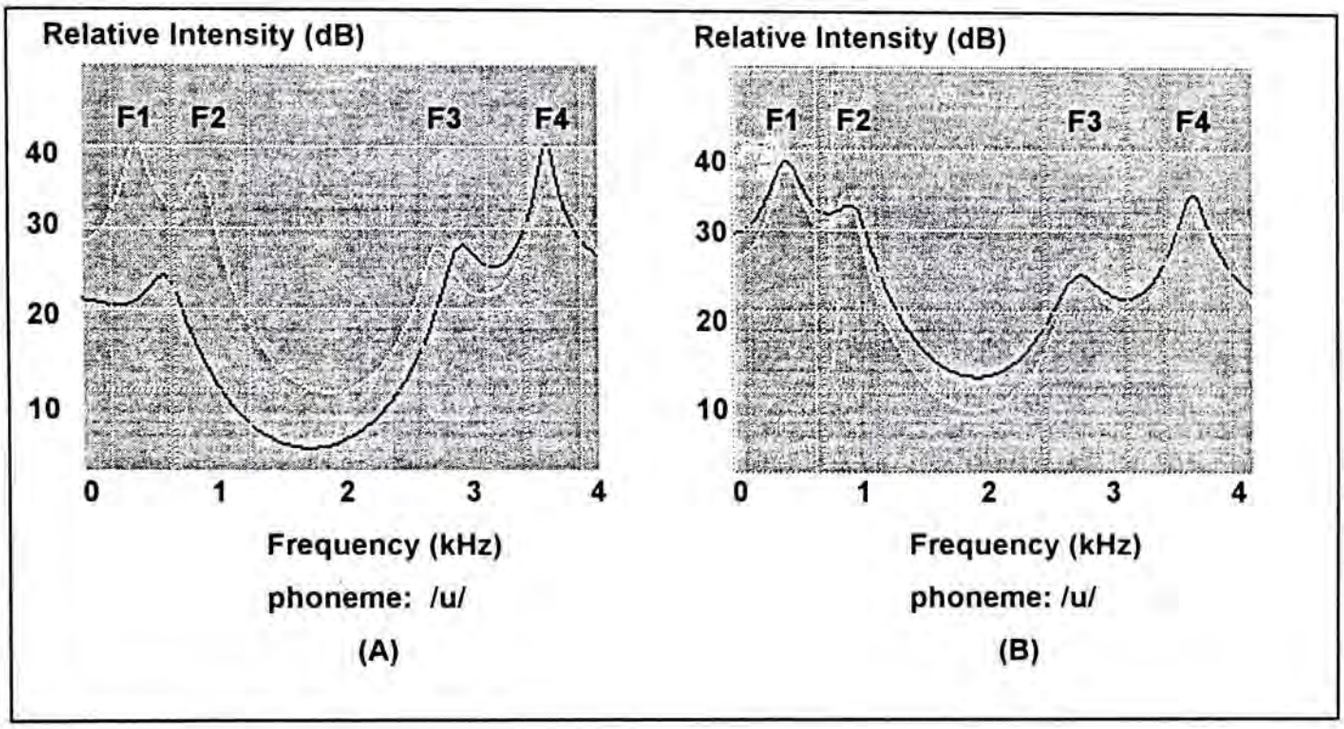


FIGURE 5.8 (Continue)

Chapter 6. Generate formant trajectories for formant synthesizers

6.1. Formant trajectories generation in synthesis-by-rule systems

The principle of synthesis-by-rule is to generate synthesis parameters given a *phonetic transcription* as input. In a typical formant-based text-to-speech system, textual input is transformed into its corresponding *phoneme sequence* by a *text-to-phoneme* module. Then a rule-based *formant trajectory generator* generates the required formant synthesis data from the input phoneme sequence. Usually, formant data of the phonemes are retrieved from a parameter table. Finally, synthesis data are fed to a formant synthesizer to produce the required output (See Fig 6.1).

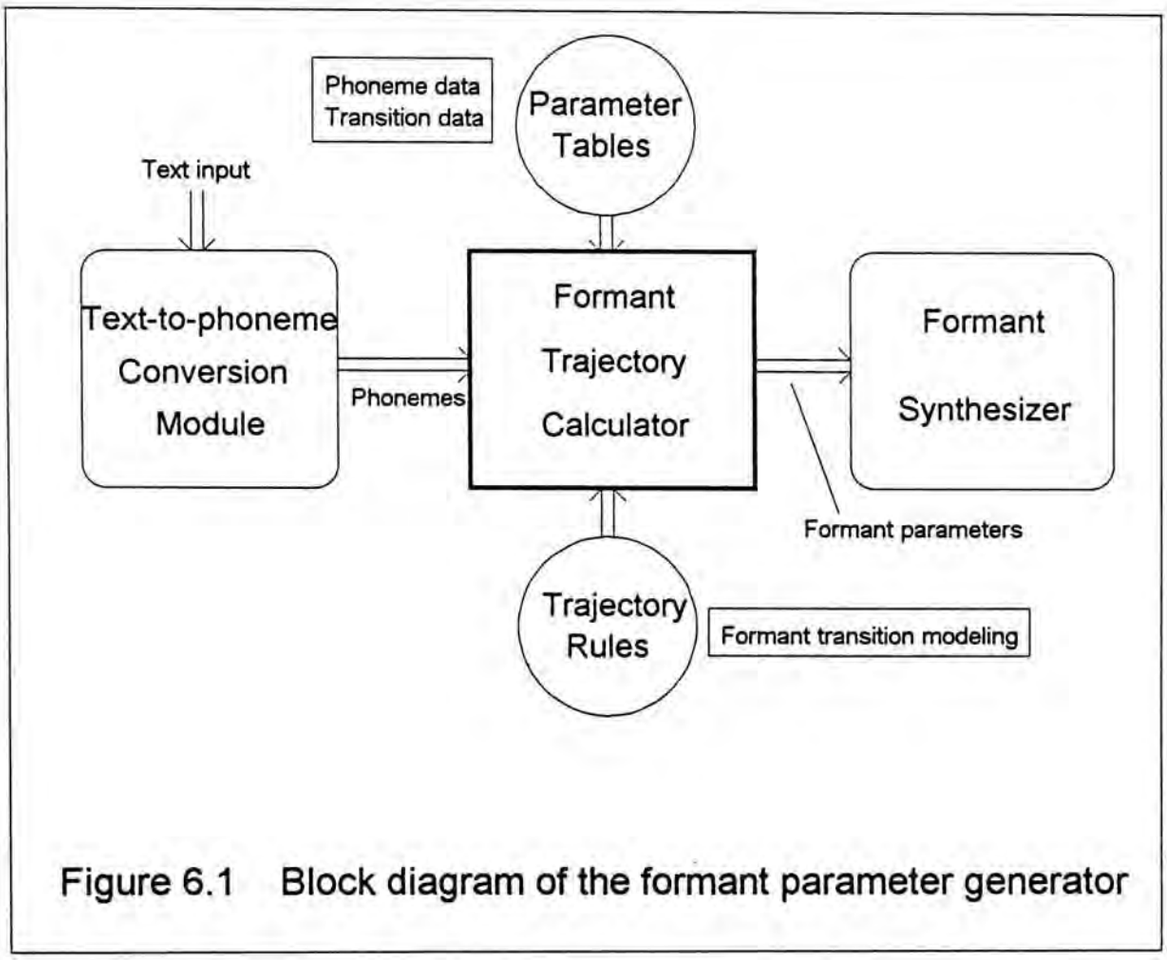


Figure 6.1 Block diagram of the formant parameter generator

Using this method, new words or phrases can be generated easily since all words can be represented in a sequence of phonemes. Moreover, there is a substantial decrease in data storage (for formant parameters) since the number of phonemes is small for most languages (e.g. In English, there are only 50 different phonemes). However, rules must be carefully designed to handle the calculation of *formant transitions* between adjacent phonemes.

6.2. Modeling formant transitions

A set of time-varying *synthesis parameters* (e.g. formant frequencies, formant amplitudes, formant bandwidths, pitch, etc.) must be evaluated during the process of formant speech synthesis. A formant transition (or *trajectory*) is defined as the locus of formant parameter values in the time domain. From observations, variations on formant parameter values (especially for formant frequencies) are often found in the *co-articulation* regions between two phonemes. On the other hand, formant parameters do not vary much outside these co-articulation regions.

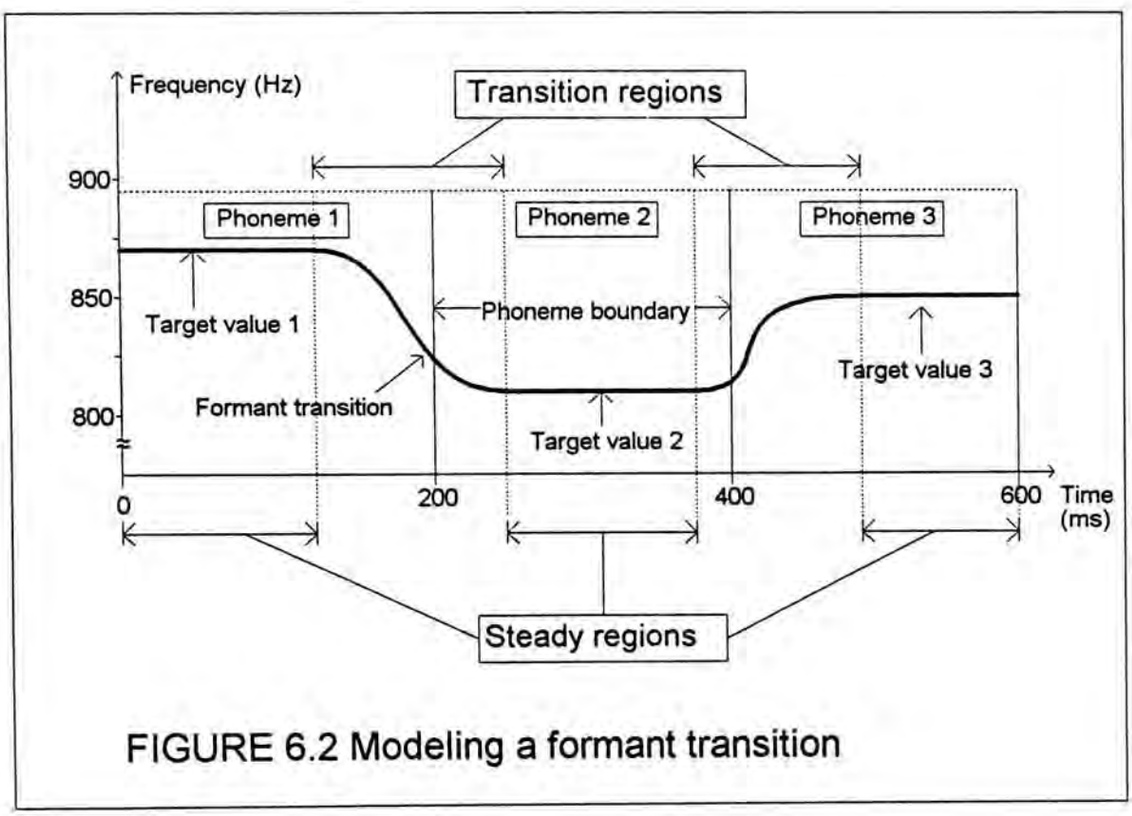


FIGURE 6.2 Modeling a formant transition

A widely accepted rule in modeling formant transitions is to assume there is a fixed *target value* for each formant parameter inside a phoneme. Inside each phoneme, two regions can be identified: (a) *steady region*, (b) *transition region*. In the steady region, formant parameters takes their default "target" value. While in the transition region, *formant trajectories* are drawn to link up the target values of the adjacent phonemes.

A formant trajectory generator reconstructs the formant trajectories in the time domain from the synthesis parameters and rules. *Interpolation methods* are required to estimate the formant transitions between fixed formant targets. Well-known formant synthesizers like *MITalk* **[3]** and *HMS synthesizer* **[17]** make use of *piecewise linear segments* in interpolation. Other interpolation schemes such as decaying/increasing exponential methods **[32]** and bézier curve methods **[2]** are also mentioned in different works.

6.3. Conventional formant transition calculation

For conventional formant transition interpolation, formant trajectories are drawn by linking up target points in the time domain using linear segments. The interpolation scheme of two synthesis-by-rule systems, the *HMS synthesizer* and *MITalk*, will be described below:

The *HMS synthesizer* encodes the format parameters of 25 consonants and 21 vowels (for English) into a *parameter table* that contains 27 entries **[17]**. For each phoneme, target formant frequencies and formant amplitudes of the first 4 formants are stored. In addition, a set of linear trajectory coefficients, together with the duration of phonemes and formant transitions, are also recorded. *Ranking rules*, CV(Consonant-Vowel) trajectory rules and *overlapping* rules are used with the parameter table to evaluate formant parameters for successive time frames.

In the *HMS synthesizer*, a typical CV transition trajectory is represented by two piecewise linear segments and they are shown in Fig 6.3. In the figure, P_a, P_b and P_c are three

turning points of the linear segments. Rules and parameters are used to find out the geometric location of P_a,P_b and P_c. Then *straight lines* are used to join the two targets (P_a and P_c) and the boundary point P_b together to form a formant trajectory.

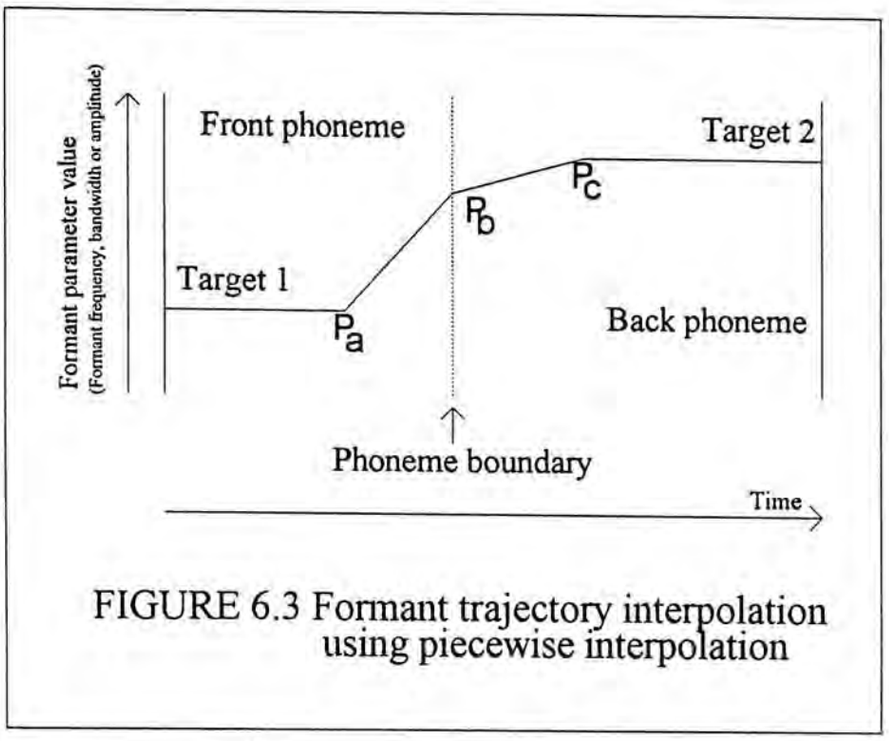


FIGURE 6.3 Formant trajectory interpolation using piecewise interpolation

The original idea of *MITalk* came from the Holmes's model **[3]**, Klatt tried to improve Holmes's model on some over-simplified aspects. In *MITtalk*, formant transitions are classified into four types. Each type is represented by a *template* which is consisted of several piecewise straight line segments **[3]**. Phonemes are divided into 31 *classes* and a complicated rule-based algorithm is used to decide which transition type is suitable according to the phoneme class sequence. Klatt's model is now claimed to be one of the best speech synthesizers with highest intelligibility and fidelity.

6.4. The 4-point Bézier curve model

Formant transition models described in section 6.3 may not often give a close approximation to real speech. By using a suitable formant tracking algorithm, it can be observed that real formant transitions cannot be reasonably approximated by using only a few

straight lines. For this reason, Leung **[2]** has suggested a model based on 4-point Bézier curves. In this model, CV(Consonant-Vowel) and VC(Vowel-Consonant) transitions are modeled by 4-point bézier curves. While more complex transitions are constructed from appropriate combination of two or more CV/VC transitions. This scheme claims to achieve the following goals: (1) Reduce the number of formant transition rules in the interpolation process. (2) Generate more realistic formant trajectories.

A bézier curve with four control points is smooth and continuous. Mathematically, a bézier curve is defined by the following parametric equations:

$$P_x(t) = \sum_{i=0}^{n} B_{x,i} J_{n,i}(t). \qquad \text{(Equation 6.1a)}$$

$$P_y(t) = \sum_{i=0}^{n} B_{y,i} J_{n,i}(t). \qquad \text{(Equation 6.1b)}$$

$$J_{n,i}(t) = \binom{n}{i} t^i (i-t) n^i \quad \text{where} \quad 0 \le t \le 1. \qquad \text{(Equation 6.1c)}$$

Where $B_{x,i}$ and $B_{y,i}$ are the x-coordinate and y-coordinate of the i-th control point of the bézier curve. $P_x(t)$ and $P_y(t)$ are the x and y coordinates of the bézier curve for parameter t ($t \in [0,1]$). The positions of the 4 control points, B0,B1,B2 and B3 are defined in Table 6.1.

Control Point	**Time Value** (x-axis)	**Formant Parameter Value** (y-axis)
B0	start time of transition	target1
B1	variable	target1
B2	variable	target2
B3	ending time of transition	target2

Table 6.1 The positions of the four control points

Where B0 and B3 are two fixed target points of the formant trajectory. The formant parameter value (y-coordinate) of B1 and B2 are fixed to the parameter values of B0 and B3 respectively. The time parameter (x-coordinate) of B1 and B2 are allowed to "slide" along a

horizontal line . The following diagram illustrates a typical bézier curve used to represent formant trajectories (See Fig 6.4).

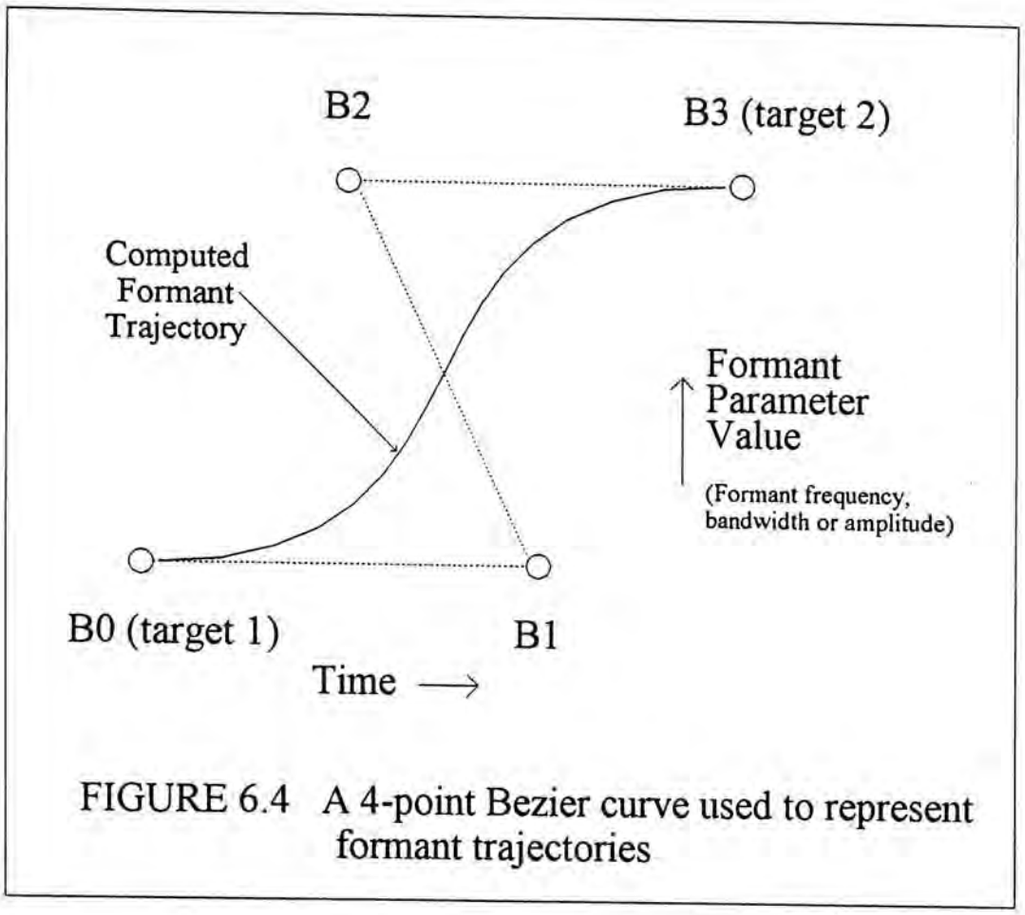


FIGURE 6.4 A 4-point Bezier curve used to represent formant trajectories

6.4.1. Computational aspects of Bézier curve

Bézier curves are defined in parametric form. It is difficult to compute the formant parameter value (y-coordinate) given the time parameter (x-coordinate) as input. One practical way is to compute several *base points* on the curve for different parameters t (See equations 6.1a, 6.1b and 6.1c), and then use linear interpolation to approximate the curve between two base points. (i.e. The curve is approximated by joining straight lines between 2 adjacent base points). The steps of computation using this linear interpolation scheme are listed below:

1. Compute P_i:($P_x(t_i)$,$P_y(t_i)$) (i:0,1,..,k-1) where $t_i = \frac{i}{k-1}$ and k is the number of base points.

   Usually, the value of k is greater than 20 in order to acheive a good approximation.

2. To find the parameter value at time T, locate the time parameter t_n(n:0,1,..,k-1) such that:

   $|P_x(t_n) - T|$ is minimum .

3. The parameter value at time T, $P_y(T)$, is found by linear interpolation between 2 base points

$$P_y(T) = P_y(t_n) + (P_y(t_{n+1}) - P_y(t_n))\frac{T - P_x(t_n)}{P_x(t_{n+1}) - P_x(t_n)}.$$

A *polynomial approximation scheme* is employed in implementation. In this scheme, 16 base points are first computed on the bézier curve. After that a 16-th order polynomial is used to approximate the whole curve. Detailed computation steps are given below:

1. Compute the coordinates of each base point P_i :($P_x(t_i)$,$P_y(t_i)$) for i=0,...,15 where $t_i = \frac{i}{15}$.
2. Devise a 16-th order polynomial function **y=A(x)** such that **$P_y(t_i)$=A($P_x(t_i)$)** for each i (i=0,1,...,15) (i.e. the polynomial **A(x)** passes through all the *base points*). The polynomial **A(x)** can be easily found by using Lagrange's interpolation formula:

$$y = A(x) = \sum_{i=0}^{p} \frac{\prod_{\substack{j=0 \\ j\neq i}}^{p} (x - x_j)}{\prod_{\substack{k=0 \\ k\neq i}}^{p} (x_i - x_k)} y_i . \qquad \text{(Equation 6.2)}$$

   where x_i=$P_x(t_i)$, y_i=$P_y(t_i)$, p is the order of the polynomial.

3. The approximated parameter value at time **T** is simply **A(T)**. For synthesis purposes, the parameter value for the **j-th** speech frame is given by **A(jΔt)** where **Δt** is the width of a speech frame. This is because the synthesis parameters should be held fixed within a single speech frame.

Experimental results shown that this scheme can give a close approximation to a true bézier curve. Comparing to the linear interpolation method mentioned above, this polynomial interpolation scheme has several advantages: (1) Smoother curves can be obtained, (2) Fewer base points are needed to achieve a reasonably good approximation, (3) The estimation error is usually smaller. To speed up the computation, values of $J_{n,i}(t_i)$ (See equation 6.1c) can be pre-computed since the value of t_i's are known in advance. This can reduce the computation effort required in the evaluation of $P_x(t_i)$ and $P_y(t_i)$, since rather complex mathematical functions (e.g. factorial) are involved in the computation of $J_{n,i}(t_i)$.

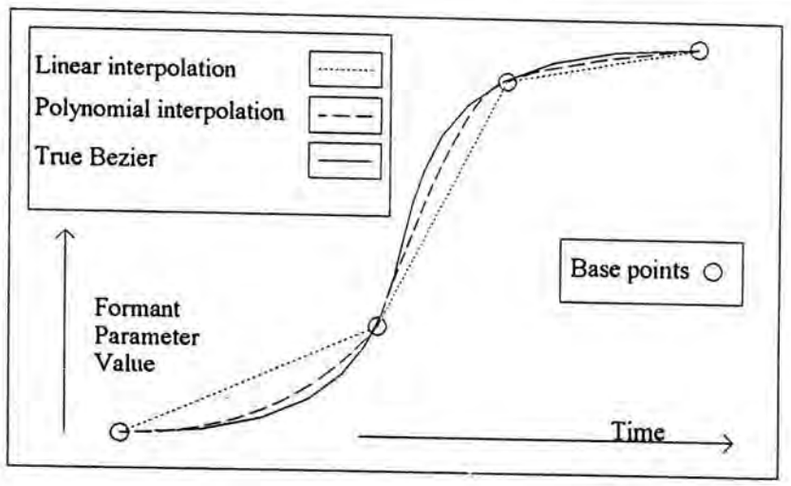


Figure 6.5 Interpolation schemes in the computation of bezier curves (The figure shows the case of using 4 base points)

6.5. Modeling of formant transitions for Cantonese

A model is proposed to generate formant trajectories for *isolated Cantonese words*. For more details of this dialect, please refer to Appendix A. In Cantonese, a word is consisted of at most one **initial** followed by one **final**. There may be no initial in certain words. Phonetically, initials can be regarded as consonants. While the structure of a final can be one of the following: (1) a pure vowel, (2) a diphthong, or (3) a vowel plus one plosive or nasal consonant. Inside a *final*, there is at most one VC(Vowel-Consonant) or VV(Vowel-Vowel)

transition. Usually, a CV transition exists between the end of an *initial* and the start of the succeeding *final*.

Therefore, there are at most *two* CV/VC/VV transitions in each word. Basically, the formant trajectories are assumed to be *stationary* (parameter values do not change) except in the *transition regions* in which the CV/VV/VC transitions took place. Regions that is not in transition regions are called *steady regions* (See Fig 6.2).

6.5.1. Encoding CV/VC/VV transitions

A single CV/VC/VV transition is modeled by using the 4-point Bézier model (described in section 6.4). The parameters that used to encode the curve is described below:

(a) *Total duration* of the transition.

(b) *Positions of the control points*: There are totally 4 control points (B0,B1,B2,B3). The positions of the control points B0 and B3 *need not* be stored since they can be *derived* from the *target points* from the *front* and *back steady regions* adjacent to the transition curve (See Fig 6.6). From the diagram, it is clear that we need only to record the x(*time*)-coordinates of the control points B1 and B2.

(c) The *starting time* of the transition relative to the start of the *front steady region*.

All CV transitions between initials and finals are stored in a *CV transition table*. While VC and VV transitions (occurred inside a final) are stored as *synthesis parameters* for finals in a *speech parameter* table. The structure of this parameter table is described in the next section.

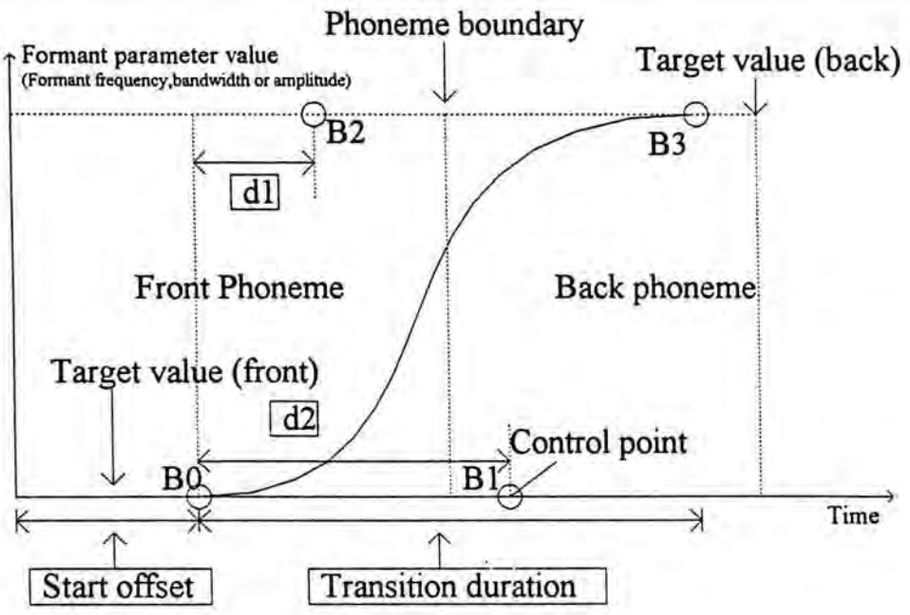


FIGURE 6.6 Defining a formant transition, the boxed quantity is stored in the transition table

6.5.2. Representation of initials and finals in the speech parameter table

(1) Initials

Normally, one set of *fixed target values* is stored for each initial. The target values include: (1) Formant frequencies, bandwidths and amplitudes of the *first four formants* (2) the *total duration* of the initial. However, for some initials (e.g. plosives, fricatives like /b/, /d/, /g/, /p/, etc.) it may be necessary to store more than one set of target values. This is because the "real" target values inside these initials are observed to be different when different types of finals are attached to the initial.

(2) Finals

Since the structure of finals is more complex than initials, a **3-segment structure** is proposed. Under this scheme, each final is partitioned into 3 regions: (a) *Steady region I*, (b) *Transition region*, (c) *Steady region II*. This is designed to cover all possible structures of finals. Inside each *steady region* (I or II), the formant parameters are assumed to be constant.

One set of *fixed target values* is stored for each of these regions. In addition, a *duration parameter* is defined for each region. A *zero* in duration means that the corresponding region does not exist. Under this arrangement it is possible to define formant parameters for some, but not all of the regions described above. For example, it is known that the formant trajectories inside a pure vowel do not vary much. Therefore it is reasonable to define formant data for region (a) only for such finals. In this case the duration parameters of region (b) and (c) can be set to zero.

In the transition region of a final, the corresponding VC/VV transition is encoded as a **4-point bézier curve** using a scheme similar to that described in Section 6.5.1 except that: (a) The duration of the VC/VV transition *need not* be stored since it is equal to the duration of the transition region, (b) The *relative starting time* of the transition is also not stored since the transition must begin at the end of *steady region I*. The total duration of the final is the sum of duration of its three regions (See Fig 6.7).

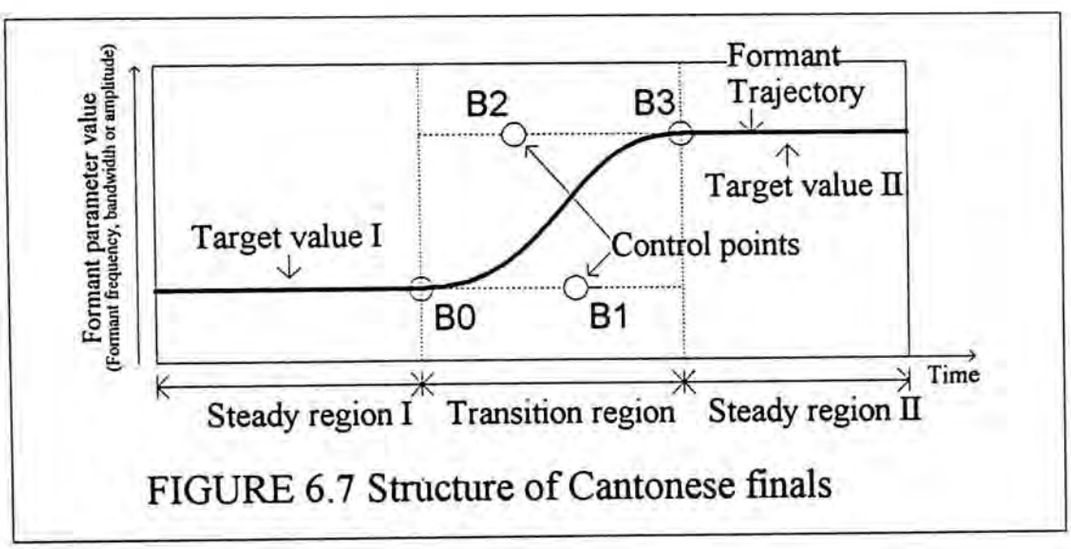


FIGURE 6.7 Structure of Cantonese finals

6.5.3. Trajectory generation

(1) First of all, a *table lookup* is done to retrieve the required synthesis parameters (e.g. target values of formant parameters, duration of phonemes) of the corresponding *initial* and *final*

of the word that is about to be synthesized. All these parameters are stored in the *speech parameter table* (See Fig 6.9).

(2) If the word has an *initial*, do the following: (a) The CV transition curve between the initial and the final is retrieved from the *CV transition table*. (b) The starting and ending time of the CV transition is then determined.

(3) If there exist a steady region (region bound by the starting point of the initial and the beginning of the CV transition) inside the initial, then formant trajectory is drawn in this region using the target value of the initial (See Fig 6.8a).

(4) Check if the *ending time* of the CV transition is in (a) the *steady region I* of the *final* or (b) the *transition region* of the final. Under no circumstances should the ending time of the CV transition be inside the *steady region II* of the *final*.

(5) If the CV transition ends up in *steady region I* of the final, do the following:

(a) Draw a **four-point bézier curve** for the CV transition, using target value of the *initial* as the *front target* and target value of the *final* (in steady region I) as the *back target* (See Fig 6.8b).

(b) Fill the region from the end of the CV curve to the end of *steady region I* of the *final* (if this region exists) by the target value in this region (See Fig 6.8c).

(c) Draw the VC/VV transition curve in the *transitional region* of the final (if this region exists) (See Fig 6.8d).

(d) Draw the formant trajectory in the *steady region II* of the *final* using the target value in that region (if this region exists) (See Fig 6.8d).

(6) If the CV transition ends up in the *transition region* of the *final*, then an **overlapping region** exists. If we still follows the procedures outlined in step (5), we will end up with two curves lying inside the overlapping region. (See Fig 6.10) Clearly, rules must be

applied to solve the this kind of conflict inside the overlapping region. Here a simple but effective method proposed by Holmes **[17]** is adopted. A linear weighted interpolation is applied inside the overlapping region. The resultant trajectory inside the overlapping region is calculated by:

$$Traj_{result}(T) = w_1(T)Traj_1(T) + w_2(T)Traj_2(T). \qquad \text{(Equation 6.3)}$$

where w_1, w_2 are linear weighting functions,

$Traj_1$ is the CV trajectory between the initial and the final,

$Traj_2$ is the VV/VC trajectory inside the final,

$Traj_{result}$ is the resultant formant trajectory.

The definition of two linear weighting functions are:

$$w_1(T) = 1 - \frac{T - T_0}{T_1 - T_0}, \qquad \text{(Equation 6.4)}$$
$$w_2(T) = 1 - w_1(T)$$

where T_0 is the starting time of the VC/VV transition,

T_1 is the ending time of the CV transition.

Figure 6.10 shows the result of applying the overlapping rule. Using this scheme, the conflict inside the overlapping region can be successfully resolved. Formant track continuity is also preserved.

(7) The steps (1) to (6) are applied to calculate the trajectory of the four formant frequencies, (F1-F4), amplitudes (A1-A4) and bandwidths (B1-B4). The formant trajectories generated during the synthesis of several Cantonese finals are shown in Fig 6.11.

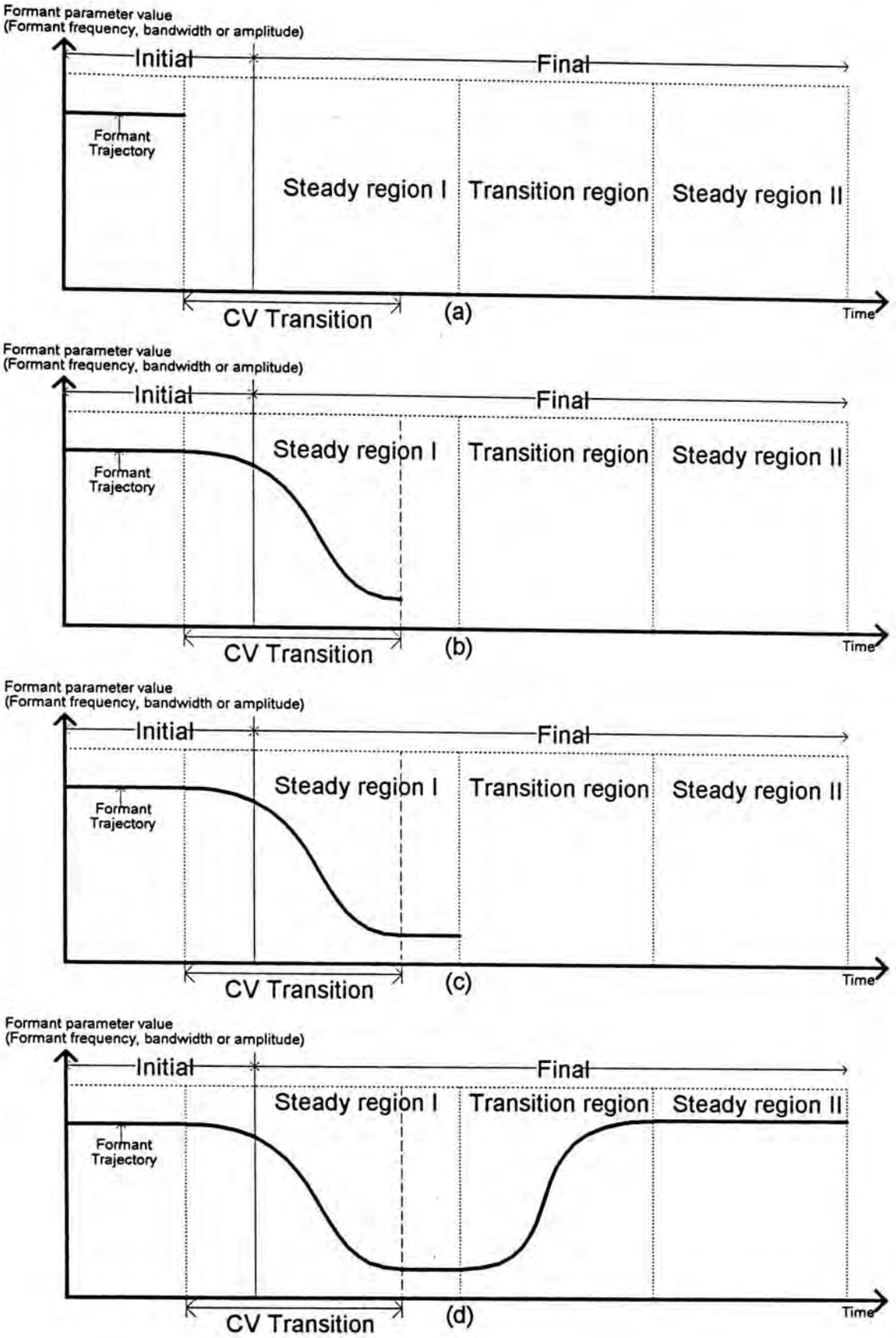


FIGURE 6.8 Steps in formant trajectory generation

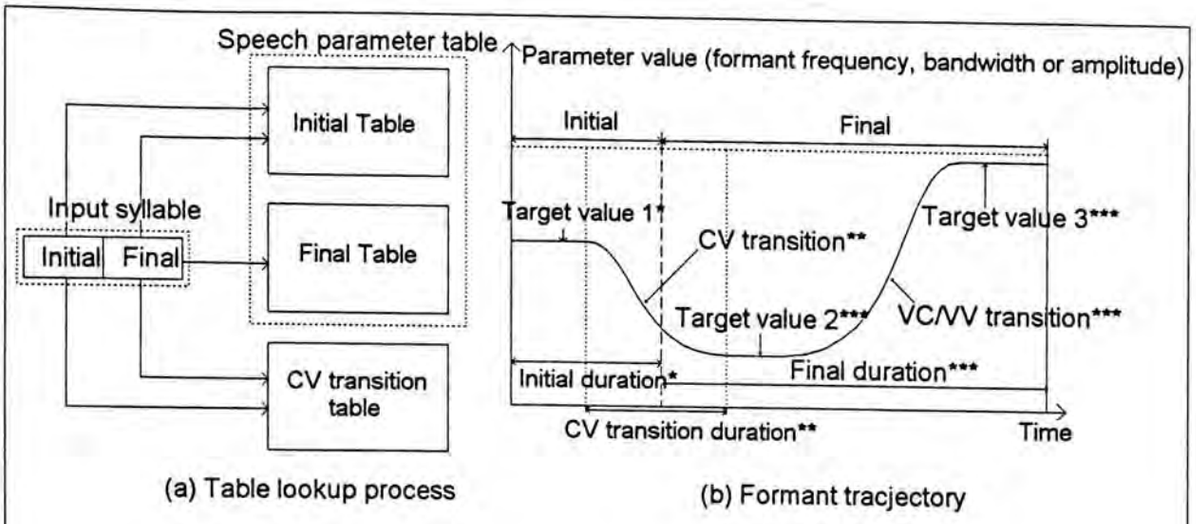


FIGURE 6.9 Retreival of synthesis parameters using table lookup. Parameters obtained from the initial table, CV transition table and final table are marked with (*), (**) and (***) respectively.

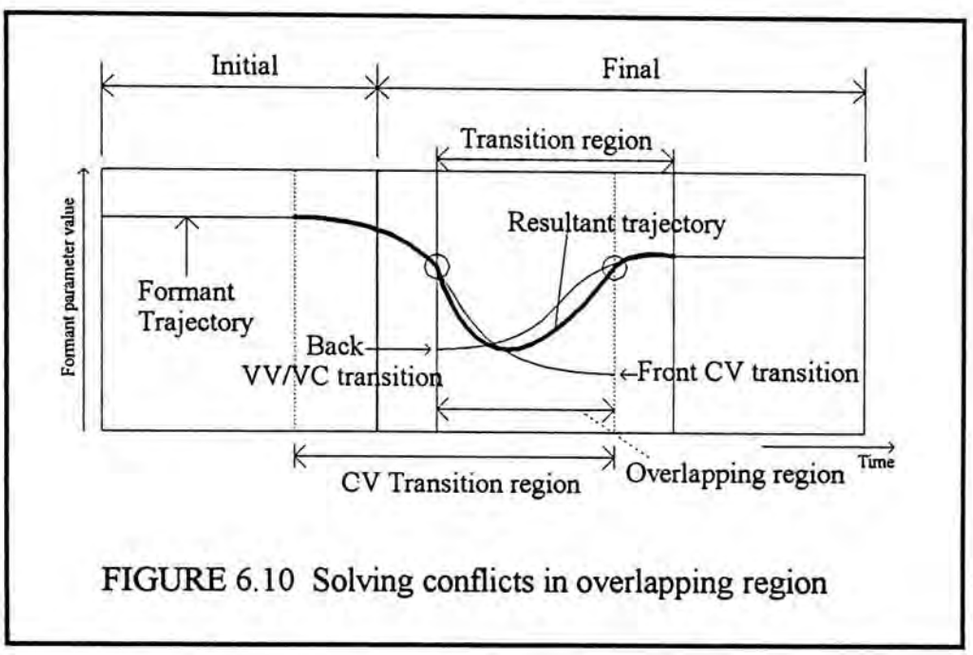
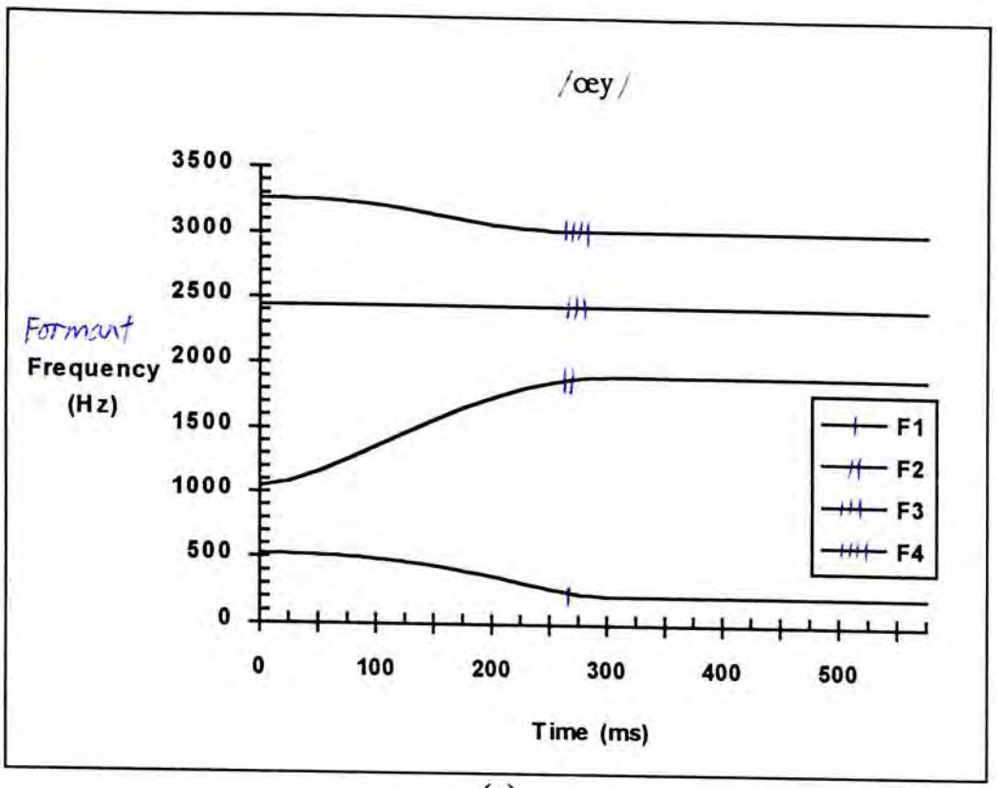
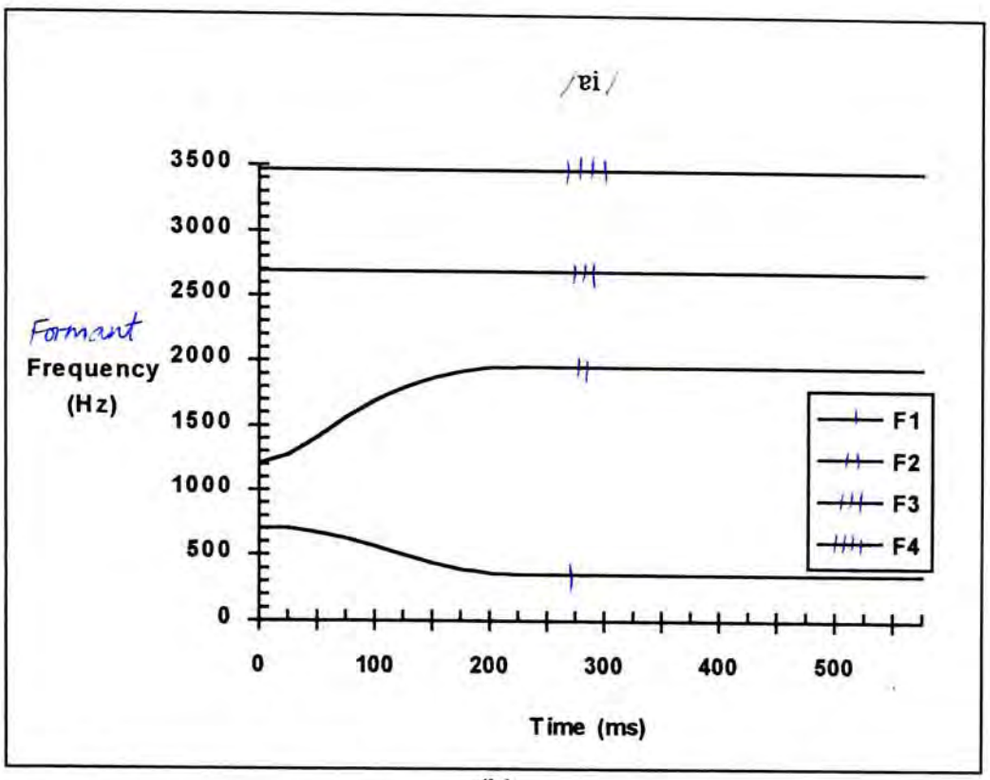


FIGURE 6.10 Solving conflicts in overlapping region



(a)



(b)

FIGURE 6.11 Examples of formant trajectories generated during the synthesis of several Cantonese finals: (a) /œy/ (b) /ɐi/

6.5.4. Handling tones in Cantonese

Cantonese, like Mandarin, is a *tonal language*. That means the fundamental frequency (pitch) is a distinctive part of the word and carries lexical meanings. On the other hand, languages such as English which use the intonation patterns in the sentence to carry certain aspects of meaning, are often classified as *intonation languages*. There are altogether six distinct tones in Cantonese (See appendix A).

In order to synthesize the six different tones, a pitch contour generator **TPIT** is employed **[2]**. Under **TPIT**, each Cantonese tone is represented by a fixed pitch-pattern template. The fundamental frequency trajectory is produced by using these pitch-pattern templates. These templates are carefully designed by analyzing pitch variations in natural speech (See Appendix B). Currently, the minor variations of formant trajectories due to fundamental frequency variations is not handled.

Using formant trajectory and tone generation rules described in this chapter, a number of Cantonese syllables are synthesized using the formant synthesizer LSYNTH. The computation time of the synthesis system is approximately 8 times real time. (i.e. The system needs 8 seconds to synthesize 1 second of synthetic speech). In order to evaluate the intelligibility of these speech samples, a number of preliminary listening tests were done. The results of the listening tests will be described in Chapter 7.

Chapter 7. Some listening test results

7.1. Introduction

Two small-scale listening tests are conducted to evaluate the intelligibility of isolated synthetic Cantonese syllables produced by the following method: (1) Formant trajectories are generated by the formant trajectory computation scheme described in Chapter 6, (2) A tone generator TPIT [2], is employed to simulate the tones in Cantonese, (3) A parallel formant speech synthesizer LSYNTH (described in chapter 4) is utilized to transform the input speech synthesis parameters into a speech waveform.

7.2. Tone recognition test

The purpose of the *tone recognition test* is to evaluate the intelligibility of *tones* in synthetic Cantonese syllables. A tone generation program TPIT is used in the speech synthesis process. A detailed description on this scheme can be founded in Appendix B.

7.2.1. Testing material

Isolated synthetic Cantonese syllables are used as testing material in the *tone recognition test*. It is not feasible to include too many syllables in a single test. A reasonable choice is to place 20-30 syllables in each test. The method of selecting syllables is described as follows: First of all, four finals, /ai/, /ɐi/, /œy/, /y/, are chosen from the Cantonese finals set. (For the pronunciation of these finals, please refer to appendix A) Based on these 4 finals, 24 syllables can be enumerated by assigning the six different tones to each final. (i.e. all final-tone combinations of the four selected finals are included). These 24 syllables are then used as the testing material in the tone recognition test.

7.2.2. Testing procedure

These twenty-four syllables are then arranged in *random order*. Listeners are required to listen to these twenty-four syllables sequentially in that order (so there are 24 questions in the tone recognition test). They have to identify the tone in each syllable by putting down the tone number (valid tone number is from 1 to 6). *Forced choice* is required. i.e. Listeners must give answer to each question.

7.2.3. Results and interpretation

Four listeners are invited to the tone recognition test. The *overall recognition* rate of this test is defined by the following equation:

$$\text{Overall recognition rate} = \frac{\text{No of correctly answered questions}}{\text{Total no of questions answered}}.$$

Out of the total 96 questions (each listener has to answer 24 questions, giving the total to 96), 90 of them are correctly answered, giving an *overall recognition rate* of 93.75%. This is a quite satisfactory result. With reference to wrongly-answered questions, three types of confusion can be observed. (a) Confusion between tone 2 and tone 5 (2 questions wrongly answered). (b) Confusion between tone 3 and tone 6 (3 questions wrongly answered). (c) Confusion between tone 1 and tone 3 (1 question wrongly answered).

The above result has shown that highly intelligible Cantonese tones can be generated by TPIT. However, a large-scale listening test is still required to evaluate the general performance of this tone generation scheme.

The naturalness of the tones produced is generally described as *fairly good*, although some listeners have reflected that the perceived tone is not natural enough.

7.3. Cantonese final recognition test

The purpose of the *Cantonese final recognition test* is to evaluate the intelligibility of synthetic Cantonese syllables consisted of *finals* only. Syllables having *initials* are not included in the testing material because synthesis data for Cantonese initials are not available at the time of conducting the listening test.

7.3.1. Testing material

Of all 53 finals of Cantonese, 24 of them are selected as testing material. All of the 24 selected finals are produced with tone 1. The selected finals are listed in table 7.1. (For the pronunciation of these finals, please refer to appendix A).

ɛk	œ	a	uŋ	ai	ɐi	ɐu	au	u	œy	ik	am
an	œŋ	ou	ɛŋ	ei	ɛ	ɔ	iu	ui	ɐk	ɐt	ɔi

Table 7.1 Selected finals in the Cantonese final recognition test

7.3.2. Testing procedure

These 24 selected finals are then arranged in some *random order*. The listener is required to identify the perceived final by either (i) Writes down the corresponding phonetic symbol (for phonetically trained persons) or (ii) Speak to the referee what he/she has heard. (for phonetically untrained persons). Forced choice is not required.

7.3.3. Results and interpretation

Five persons are invited to the Cantonese final recognition test. Like the tone recognition test, the overall recognition rate for this test is defined as:

$$\text{Overall recognition rate} = \frac{\text{No of correctly answered questions}}{\text{Total no of questions answered}}.$$

The *overall recognition rate* of this test is 74.5%. A brief summary of listening test results for various groups of finals are given below:

(1) Principal vowels (/a/, /ɔ/, /ɛ/, /œ/, /u/)

Five vowels are included in the testing set. The recognition rate in this group is 67%. Among the five vowels, /a/ and /ɔ/ are correctly recognized by all listeners. However, some listeners have difficulties in recognizing the vowels /œ/ and /u/. This may be due to the presence of *buzziness* in these vowels.

(2) Diphthongs (/ai/, /au/, /ɐi/, /ɐu/, /ɔi/, /ei/, /iu/, /œy/, /ou/, /ui/)

Ten diphthongs are included in the testing set. The recognition rate in this group is 76%. Many diphthongs, such as /ai/, /au/, /ɐu/, /œy/ are correctly recognized by all listeners. The relatively high intelligibility of these diphthongs may be due to the accurate reproduction of formant transitions inside these finals.

(3) Finals with nasal ending (/am/, /an/, /uŋ/, /œŋ/, /ɛŋ/)

Five nasal-ending finals are included in the testing set. The recognition rate in this group is 73%. No systematic tests are conducted to evaluate the degree of confusion between /m/, /n/, and /ŋ/. However, there are some confusion observed between the final pairs /am/ and /an/.

(4) Finals with plosive ending (/ɛk/, /ik/, /ɐt/, /ɐk/)

Only a few finals with plosive endings are tested. The recognition rate in this group is 82.5%. The majority of listeners have little problem in recognizing finals ended in a /k/. And the discrimination between /t/ and /k/ is quite satisfactory for the final-pair /ɐk/ and /ɐt/. (Four out of five listeners can correctly recognize the final-pair).

From the above testing results, it can be observed that some highly intelligible Cantonese finals can be successfully synthesized. However, the overall recognition rate is not very high. This may be due to the following reasons: (1) Formant parameters are not accurate enough. This is because the synthesis-by-rule system mentioned in chapter 6 computes the formant trajectory by approximation methods, so there is inevitably some loss in accuracy. (2) Isolated syllables are harder to recognize than continuous speech. This is because many syllables are very short in duration (less than 0.5 second). Moreover, the lack of synthesis data for initials have precluded the possibility of using meaningful sentences (or words) as testing material. (3) The presence of buzziness in synthetic syllables, especially when synthesized with tone 1. (4) The test was conducted in a relatively noisy laboratory.

Although some highly natural syllables are successfully synthesized, many listeners have reflected that the naturalness of many synthetic syllables is not high enough. Clearly, more works on synthesis data acquisition for Cantonese initials and finals are needed before a large-scale listening test can be conducted to evaluate the "true" performance of the Cantonese synthesis system.

7.4. Problems and discussions

Since the listening tests are small-scale ones, it is not possible to draw conclusive remarks from the test results. Although the test results cannot fully reflect the true performance of the synthesis system, it is still worthwhile to include them in this work. No formal *confusion tests* are conducted due to time limitations.

There are also some difficulties in finding suitable persons to attend the listening tests, this is because there are only a few phonetically-trained persons available. Phonetically untrained persons are not very suitable to the listening tests because many syllables in the testing materials are hard to recognize without proper knowledge in the Cantonese phonetic system.

Conclusion

In this research, we have dealt with some of the basic problems in formant-based speech synthesis, they include: (i) inaccurate extraction of formant synthesis parameters, (ii) oversimplifed formant trajectory modeling, (iii) unstability of formant speech synthesizers in the occasion of rapid transition of synthesis parameters. Generally, these three problems affect both the intelligibility and naturalness of synthetic speech output.

A feedback analysis system is implemented in order to facilitate a more accurate formant parameters extraction. Experimental results have shown that this system can give more accurate formant parameters, especially for formant amplitudes, when comparing to the results given by traditional LPC analysis. However, the computation complexity of the feedback analysis system is rather high. Further work should be done in order to reduce its computational complexity. In addition, other aspects of speech modeling such as glottal pulse excitation and pitch fluctuation, etc. should be investigated in order to further improve the naturalness of synthetic speech.

A formant trajectory generation scheme is proposed for the production of Cantonese syllables. The trajectory generation scheme is based on Bézier curve interpolation. The computation complexity of computing Bézier curves is reduced by the introduction of a polynomial interpolation method.

Preliminary listening test results have shown that some high quality synthetic Cantonese syllables can be produced. However, due to limitation of time, synthesis parameters are missing for some Cantonese syllables. It is hoped that these missing parameters can be acquired in the future.

A parallel formant synthesizer is implemented in software. The method of linear interpolation of formant amplitude input is implemented on LSYNTH so that unwanted

waveform fluctuations arise from a rapid formant amplitude transition can be reduced. Currently, LSYNTH cannot synthesize speech in real time because of its high computational complexity. This problem can be solved by using: (i) dedicated hardware to simulate the digital filter system, (ii) Using a DSP processor to carry out computations that involves digital filtering.

Although a set of speech analysis tools have been developed for this research, further improvements on them are favorable. For example, some process in formant tracking, like the selection of anchor points, trajectory smoothing, etc. can be done automatically.

References

[1] Klatt,D.H. (1987). "Review of text-to-speech conversion for English", *J. Acoust. Soc. Am.*, vol. 82, pp. 737-1099.

[2] Leung K.H. (1992). *Computer speech synthesis - To improve the naturalness of a formant-based speech synthesizer*, Mphil. Thesis, Computer Science Dept., The Chinese University of Hong Kong.

[3] Allen.J, Hunnicutt,S. and Klatt, D.H. (1987). *From text to speech: The MITalk system*, Cambridge U.P., Cambridge, U.K.

[4] Medan,Y. and Yair,E. (1989) "Pitch synchoronous spectral analysis scheme for voiced speech", *IEEE trans. on acoustics, speech and signal processing*, vol.37, No.9, pp.1321-1331.

[5] Markel,J.D. and Gray,A.H.,Jr. (1976) *Linear prediction of speech*, Springer-Verlag, Berlin.

[6] Durbin,J. (1959) "Efficient Estimation of Parameters in Moving-Average Models" *Biometrika*, vol.46, pp.306-316.

[7] McCandless,S.S. (1974) "An algorithm for automatic formant extraction using linear predictive spectra" *IEEE trans.* ASSP-22, pp.135-141.

[8] Saito, S. and Nakata, K. (1988) *Fundamentals of speech signal processing*, Academic press, Japan.

[9] Ross,M.J.,et al. (1974) "Average magnitude difference function pitch extractor" *IEEE trans.* ASSP-22, pp.353-362.

[10] Markel,J.D. (1972) "The SIFT Algorithm for Fundamental Frequency Estimation", *IEEE trans.* AU-20, pp.129-137.

[11] Parsons,T.W. (1987) *Voice and speech processing*, McGraw Hill Press, New York.

[12] Itakura,F. and Saito,S. (1968) "Analysis Synthesis Telephony Based Upon the maximum likelihood method" Reports of 6th Int. Cong. Acoust., ed. by Y.Kohasi, Tokyo, C-5-5,C17-20.

[13] Shirai, K. (1992) "Articulatory model," in *Speech Science and Technology*, Saito, S. Ed., Ohmsha Ltd., Japan, pp. 51-61.

[14] O'Shaughnessy,D., et al. (1988) "Diphone speech synthesis", *Speech Communication*, vol.7, pp.55-65.

[15] Atal, B.S. and Caspers, B. (1987) "Role Of Multi-Pulse Excitation In Synthesis Of Natural-Sounding Voiced Speech", *ICASSP-87*, pp. 2388-2391.

[16] Atal,B.S. (1985) "Linear Predictive coding of speech," in Computer speech processing, Fallside,F. and Woods, W.A. Ed., Prentice-Hall, London, pp.81-123.

[17] Holmes,J.N., Mattingly,I.G., and J.N.Shearme (1964), "Speech synthesis by rule", *Language and Speech*, Vol. 7, No. 3, pp.127-143.

[18] Jarvin,H.,et al. (1989) "A Multilingual Text-to-speech System", *ICASSP-89*, pp.2833.

[19] Flanagan, J.L. (1957) "Note on the Design of Terminal Analog Speech Synthesizers", *J.Acoust.Soc.Am.* Vol. 29, pp. 306-310.

[20] Rosenberg,A.E. (1971) "Effects of Glottal pulse shape on the quality of natural vowels", *J. Acoust. Soc. Am.*, vol.49, pp.583-590.

[21] Gold,B. and Rabinder,L.R. (1968) "Analysis of Digital and Analog formant synthesizers" *IEEE Trans. on Audio and Electroacoustics* AU-16, pp.81-94.

[22] Holmes,J.N. (1985) "A parallel-formant synthesizer for machine voice output," in *Computer speech processing*, Fallside,F. and Woods, W.A. Ed., Prentice-Hall, London, pp.163-187.

[23] Markel,J.D. (1972) "Automatic Formant and Fundamental Frequency from a Digital inverse filter formulation" Conference Reprints, 1972 Int. Conf. Speech Commun. Process., Boston, Massachusetts, paper 89, pp.81-84.

[24] Breen,A.P. (1988) "Improving the naturalness of synthetic speech through amplitude transformation", *Proc. Speech '88 (FASE)*, vol.1, pp.67-71.

[25] Lowry,A. et al. (1989) "Analysis and encoding of speech for a parallel formant synthesizer" *ICASSP-89*, pp.2833.

[26] Holmes,W.J, Holmes,J.N. and Judd,M.W. (1990) "Extension of the Bandwidth of the JSRU parallel-formant synthesizer for high-quality synthesis of male and female speech", *ICASSP-90*, pp.313-316.

[27] Dunn, H.K. (1961) "Methods of Measuring Vowel Formant Bandwidths", *J.Acoust.Soc.Am.*,vol. 33,pp. 1731-1746.

[28] Appendino, A. and Vivalda,E. (1988) "Automatic Extraction of Synthesis Parameters For Text-to-speech Synthesis By Rule", Signal Processing IV: Theory and Applications - Proceedings of EUSIPCO-88, Vol II, September, 1988, pp.1055-1058.

[29] Quackenbush,S.R, Barnwell III,T.P., and Clements,M.A. (1988) *Objective measures of speech quality*, Prentice-Hall, London.

[30] Holmes,J.N. (1973) "The influence of Glottal Waveform on the Naturalness of Speech from a parallel formant synthesizer", *IEEE Trans on Audio and Electroacoustics* AU-21, pp.298-305.

[31] Kobayashi,T. and Sekine,H. (1990) "Statistical properties of fluctuation of pitch intervals and its modelling for Natural Synthetic speech", *ICASSP-90*, pp. 321-324.

[32] Wright, R.D. and Elliott, S.J., "Parametric Interpolation In Speech Synthesis", *J.Acoust.Soc.Am.* Vol .87, pp.383-391.

[33] Wong,S.L. (1940) *A Chinese syllabary pronounced according to the dialect of Canton*, Chung Hwa press, Hong Kong.

[34] Chow,M.K. and Yiu,B.C.(1988) *A Standard Cantonese Pronounciation Syllabary*, The Commercial Press, Hong Kong.

Appendix A. The Cantonese phonetic system

A.1. Classification of phonetic units in Cantonese

Up to now, there does not exist an universally accepted standard in the classification of phonetic units in Cantonese. Separate classification schemes can be found in famous works like: (1) "A syllabary pronounced according to the dialect of Canton" by S.L.Wong **[33]** and (2) "Standard Cantonese pronunciation syllabary" by Chow and Yiu **[34]**. The major difference between these two schemes lies on the classification of principal vowels. For the sake of uniformity, the classification scheme proposed by Wong is adopted.

A.2. Analysis of phonetical units of Cantonese

In Cantonese, a syllable is usually divided into two parts: the **initial** (聲 母) and the **final** (韻 母). By *initial* we mean the *consonant* which begins the syllable. By *final* we mean the rime which ends the syllable **[33]**. Due to the mono-syllabic nature of the Chinese language, each Chinese character is pronounced with one syllable.

A Cantonese syllable is represented not only by an *initial* and a *final*, but also by a kind of musical pitch which we called the *tone* (聲 調). In the following context, a Cantonese syllable is denoted by writing down its three components (initial, final and tone) in sequential order from left to right. (e.g. dzuŋ^{1} 中 , where dz is the *initial*, uŋ is the *final* and 1 is the *tone*). Descriptions on Cantonese initials, finals and tones will be given in the following sections.

A.2.1. Finals

In Cantonese, finals can be classified into 5 types:

(1) A principal vowel.

(2) A diphthong formed by a principal vowel followed by /i/ /u/ /y/.

(3) A principal vowel followed by a nasal consonant /m/ /n/ /ŋ/.

(4) A principal vowel followed by a plosive /p/ /t/ /k/.

(5) Nasals /m/ /ŋ/ when fully voiced give 2 independent finals.

There are 53 finals in the Cantonese dialect, which are tabulated below:

	Principal Vowels											
type	a	ɐ	e	ɛ	i	o	ɔ	œ	u	y	m	ŋ
1	a			ɛ	i		ɔ	œ	u	y		
2	ai	ɐi	ei				ɔi		ui			
	au	ɐu			iu	ou						
								œy				
3	am	ɐm			im							
	an	ɐn			in		ɔn	œn	un	yn		
	aŋ	ɐŋ		ɛŋ	iŋ		ɔŋ	œŋ	uŋ			
4	ap	ɐp			ip							
	at	ɐt			it		ɔt	œt	ut	yt		
	ak	ɐk		ɛk	ik		ɔk	œk	uk			
5											m	ŋ

Table A-1. Table of Cantonese finals

A.2.2. Analysis of Cantonese finals

Among the principal vowels, three of them are not found in English, they are /y/, /œ/ and /ɐ/ respectively. /y/ is the *lip-rounded* version of /i/. It is very similar to the German sound /ü/. /œ/ can be treated as the *lip-rounded* version of /ɛ/. While /ɐ/ is similar to the English vowel /ʌ/ but not identical. In fact /ɐ/ is a neutral vowel, somewhere between /ʌ/ and /a/ (close to /ʌ/ more than /a/).

The pronunciation of these finals can be realized by table A-2. In this table, the pronunciations of these 53 finals are indicated by a set of selected Chinese words.

Final	Sample word
a	丫 a¹
ai	拉 lai¹
au	交 gau¹
am	淡 dam⁶
an	曼 man⁶
aŋ	孟 maŋ⁶
ap	甲 gap⁸
at	八 bat⁸
ak	白 bak⁹
ɐi	西 sɐi¹
ɐu	修 sɐu¹
ɐm	心 sɐm¹
ɐn	身 sɐn¹
ɐŋ	生 sɐŋ¹
ɐp	急 gɐp⁷
ɐt	不 bɐt⁷
ɐk	北 bɐk⁷
ei	卑 bei¹

Final	Sample word
ɛ	些 sɛ¹
ɛŋ	釘 dɛŋ¹
ɛk	吃 hɛk⁸
i	衣 yi¹
iu	夭 yiu¹
im	淹 yim¹
in	煙 yin¹
iŋ	英 yiŋ¹
ip	接 dzip⁸
it	折 dzit⁸
ik	直 dzik⁹
ou	高 gou¹
ɔ	柯 ɔ¹
ɔi	哀 ɔi¹
ɔn	安 ɔn¹
ɔŋ	方 fɔŋ¹
ɔt	喝 hɔt⁸
ɔk	作 dzɔk⁸

Final	Sample word
œ	靴 hœ¹
œy	居 gœy¹
œn	殉 sœn¹
œŋ	香 hœŋ¹
œt	卒 dzœt⁷
œk	腳 gœk⁸
u	污 wu¹
ui	偎 wui¹
un	官 gun¹
uŋ	中 dzuŋ¹
ut	沒 mut⁹
uk	屋 uk⁷
y	於 jy¹
yn	冤 jyn¹
yt	月 jyt⁹
m	唔 m⁴
ŋ	吳 ŋ⁴

Table A-2. Pronunciation of Cantonese finals

A.2.3. Initials

There are 19 initials (consonants) which can be classified according to their manner and place of articulation: (See Fig A-3)

	Labial	Labio-dental	Alveolar	Alveo-palatal	Palatal	Velar	Glottal
Plosives	p		t			k	
	b		d			g	
Fricatives		f	s				h
Affricatives				ts			
				dz			
Nasals	m		n			ŋ	
Glides	w				j		
Lateral			l				

Compound	kw	gw

Table A-3 Table of Cantonese Initials

A.2.4. Analysis of Cantonese initials

In Cantonese, there are six plosives, all of them are *voiceless* and the difference are identified by the presence or absence of *aspiration* only. The three aspirated plosives are /p/, /t/, /k/ and the unaspirated ones are /b/, /d/ and /g/.

There are two *affricatives* in the dialect, namely /ts/ and /dz/. They are also unvoiced and can be identified by the presence or absence of *aspiration*.

For other consonants, such as the *fricatives* (/f/, /s/, /h/), the *nasals* (/m/, /n/, /ŋ/) and the *approximants* (/w/, /j/, /l/), they are very similar to their English counterparts. However, the nasal consonant /ŋ/ can be used initially or finally in Cantonese while it can only be used finally in English.

There are also two compound consonants in Cantonese, namely /kw/ and /gw/. The pronunciation of /gw/ is similar to the English sound of "gu" in the word "language" with the voice of g eliminated. /kw/ is simply the *aspirated* variety of /gw/, similar to the sound of "qu" in the word "question". The pronunciation of these initials can be realized by table A-4.

Initial	Sample word
b	巴 ba^1
d	打 da^2
g	家 ga^1
p	爬 pa^4
t	他 ta^1
k	卡 ka^1
ts	差 tsa^1
dz	渣 dza^1
f	花 fa^1
s	沙 sa^1

Initial	Sample word
h	哈 ha^1
m	媽 ma^1
n	拿 na^4
ŋ	牙 $ŋa^4$
w	娃 wa^1
j	也 ja^3
l	啦 la^1
gw	瓜 gwa^1
kw	夸 kwa^1

Table A-4 Pronunciation of Cantonese initials

A.3. Tone

Tone (聲 調) is a kind of *pitch contour* that is vital to the formation of Cantonese syllables. Classically, there are four tones in the Chinese language, they are named as **level**(平), **rising**(上), **going**(去) and **entering**(入) respectively. However, these classical terms no longer indicate the musical values of the tones in actual existence. The tones which actually exist nowadays are nine and can be represented by the following table:

Tone No	Tone type		Sample word
1	Upper level tone	高平	詩 (si^1)
2	Upper rising tone	高上	史 (si^2)
3	Upper going tone	高去	試 (si^3)
4	Lower level tone	低平	時 (si^4)
5	Lower rising tone	低上	市 (si^5)
6	Lower going tone	低去	事 (si^6)
7	Upper entering tone	高入	色 (sik^7)
8	Mid entering tone	中入	錫 $(sɛk^8)$
9	Lower entering tone	低入	食 (sik^9)

Table A-5 The nine tones of Cantonese

However, the so-called *entering tones* are not really special tones at all, their musical values being identical with those tones known as *upper level* (Tone 1), *upper going* (Tone 3) and *lower going* (Tone 6), respectively. But there is a real difference, namely, that syllables classified under the entering tones always end in /p/,/t/ or /k/, while syllables of the other six tones end in sounds other than /p/,/t/,/k/. So there are actually *six different tones* in the dialect **[33]**.

Appendix B. TPIT, a tone trajectory generator for Cantonese

In TPIT **[2]**, the *tone trajectories* (the locus of fundamental frequency over time) of different tones in Cantonese are modeled by six *tone templates* composed of piecewise straight lines, *exponential decay and rising curves.* Since each of the three *entering tones* (Tone 7,8,9) bear close resemblance to one of the six *non-entering tones* (Tone 1,3,6, respectively), the tone trajectories of these *entering tones* are not separately modeled. In the actual synthesis process, the tone templates of tone1,3,6 can be used in the synthesis of tone 7,8,9 respectively. Descriptions on the tone templates of the six non-entering tones are given below: (These descriptions can also be found in Leung **[2]**)

I. Tone 1 (Upper level)

It is simulated by a straight line segment and an exponential rising curve (See Fig B-1).

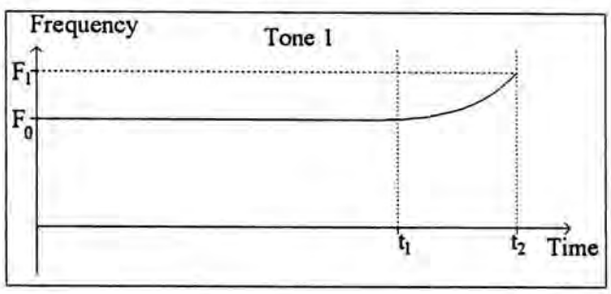


Figure B-1: Tone 1

Where F_0=145 Hz, F_1=160 Hz, t_1=0.74t_2.

From t=0 to t=t_1,

fundamental frequency = 145 Hz, and

from t=t_1 to t=t_2,

fundamental frequency increase exponentially to 160 Hz.

II. Tone 2 (Upper rising)

It is simulated by an exponential decay curve and an exponential rising curve (See Fig B-2).

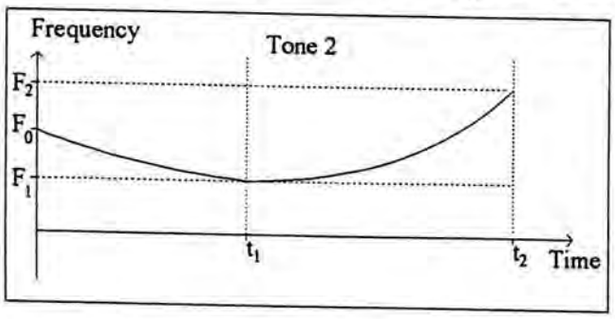


Fig B-2: Tone 2

Where F_0=107 Hz, F_1=103 Hz, F_2=173 Hz, $t_1=0.4t_2$.

From t=0 to $t=t_1$,

fundamental frequency decays exponentially from 107 Hz to 103 Hz, and

from $t=t_1$ to $t=t_2$,

fundamental frequency increase exponentially to 160 Hz.

III. Tone 3 (Upper going)

It is simulated by an exponential decay curve and an exponential rising curve (See Fig B-3).

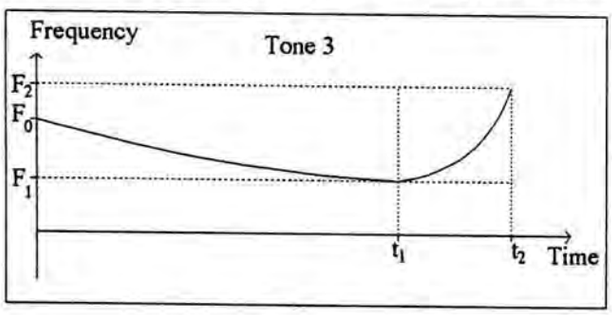


Fig B-3: Tone 3

Where F_0=139 Hz, F_1=123 Hz, F_2=177 Hz ,$t_1=0.78t_2$.

From t=0 to $t=t_1$,

fundamental frequency decays exponentially from 139 Hz to 123 Hz, and

from $t=t_1$ to $t=t_2$,

fundamental frequency increases exponentially to 177 Hz.

IV. Tone 4 (Lower level)

It is simulated by two straight lines (See Fig B-4).

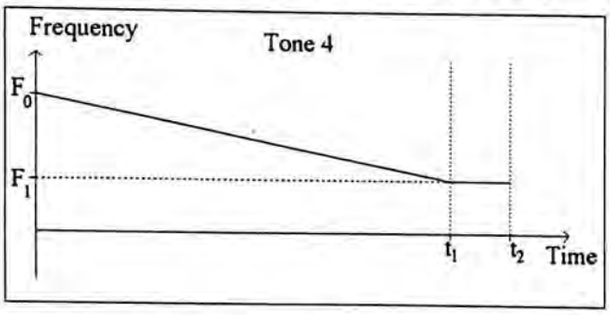


Fig B-4: Tone 4

Where F_0=110 Hz, F_1=70 Hz, t_1=0.93 t_2.

From t=0 to t=t_1,

fundamental frequency decays linearly from 110 Hz to 70 Hz, and

from t=t_1 to t=t_2,

fundamental frequency = 70 Hz.

V. Tone 5 (Lower rising)

It is simulated by an exponential decay curve and an straight line (See Fig B-5).

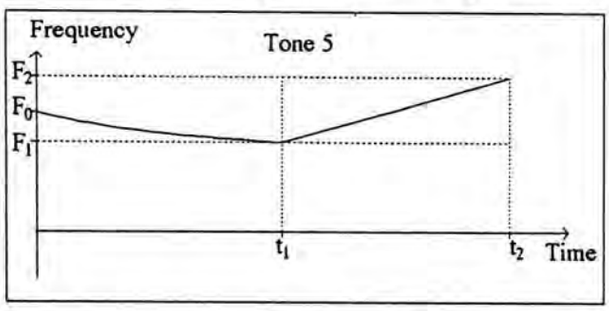


Fig B-5: Tone 5

Where F_0=129 Hz, F_1=106 Hz, F_2=137 Hz, t_1=0.6t_2.

From t=0 to t=t_1,

fundamental frequency decays exponentially from 129 Hz to 106 Hz, and

from t=t_1 to t=t_2,

fundamental frequency increases linearly to 137 Hz.

VI. Tone 6 (Lower going)

It is simulated by two exponential decay curves joined by a straight line (See Fig B-6).

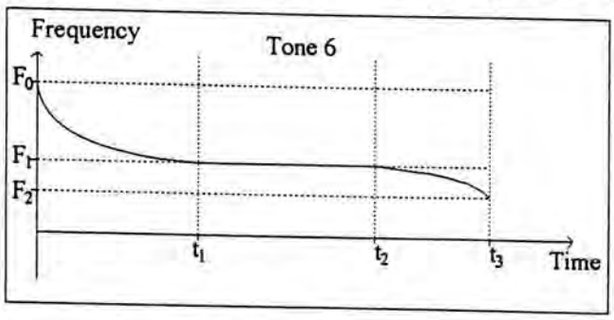


Fig B-6: Tone 6

Where F_0=135 Hz, F_1=106 Hz, F_2=90 Hz, $t_1=0.38t_3$, $t_2=0.76t_3$.

From t=0 to $t=t_1$,

fundamental frequency decays exponentially from 135 Hz to 106 Hz, and

from $t=t_1$ to $t=t_2$,

fundamental frequency = 106 Hz, and

from $t=t_2$ to $t=t_3$,

fundamental frequency decays exponentially to 90 Hz.