Low-Complexity CodeBook Searching Algorithms for FS1016

Prepared by Chien-Kuang Lin

Advisory by Prof. Po-Ning Chen

In Partial Fulfillment of the Requirements

For the Degree of
Master of Science

Department of Communications Engineering

National Chiao Tung University

Hsinchu, Taiwan 300, R.O.C.

E-mail: u8913520@cc.nctu.edu.tw

June, 2002

Abstract

With the increasing demand for packet-voice transmission, low bit-rate speech coders gradually become a research trend. In February 1991, General Services Administration published Federal Standard 1016 (FS1016) [6] which specifies the requirements for the conversion of analog voice to digital data by a method of 4.8 Kbps CELP. However, the major obstacle for applying it to real-time applications is the mass computational complexity in codebook search.

In this thesis, we improve the FS1016 complexity by presenting a new codebook searching algorithm. The improvement of our new algorithm over the original one is not only illustrated by the reduction of the number of computations required in principle, but also demonstrated through a so-implemented FS1016 ACM driver under Windows Operating System. To conclude the research on the area of packet-voice transmission in our lab, a combined ACM driver with our fast searching algorithm, and a packet-loss recovery scheme developed previously has also been implemented.

Acknowledgements

Thank Prof. Po-Ning Chen for everything he taught and assisted me. I am the most lucky one who has such a nice advisor.

Thank all my family for their support and encouragement during my study, especially my mother, my most elder sister and Shin-Ya.

Thank all the dear mates in the network laboratory, and also my friends in National Tsing-Hua University. They make my graduate life become a joyful journey.

Thank Prof. Yunghsiang S. Han for his brilliant guidance and his recommendation letter for my job application. Thank Mr. Chia-Long Wu for his assistance, particularly during the period when I was only an undergraduate. Also appreciated is Mr. S. C. Shieh for his helpful discussions.

Thank National Chiao-Tung University for providing such a wonderful environment and so many resources for me to learn. Thank Dr. Kai-Zhi Ye for providing scholarship that encourages me a lot.

Contents

List of Figures

Chapter 1

Introduction

The Internet Protocol(IP)-based telephony (also called *VoIP*) is one of the emerging technologies today. As stated in [7], the VoIP traffic is only 310 million minutes in 1998, and fast grows up to 2.7 billion minutes in 1999. It is also predicted to be 135 billion minutes in 2004. As such, the growth rate is approximately 43500 percent from 1998 to 2004. One can realize the benefit of VoIP through the viewpoint respectively from residential and corporate customers. To residential users, a VoIP call provides toll-by-pass services, and thereby is more economic. To corporate consumers that typically equipped with PSTN and LAN/Internet connections in their corporations, an even larger saving in phone bills can be expected by incorporating VoIP, as well as FoIP, services. As a consequence, the deployment of the VoIP telephony network gradually becomes a trend.

A significant part of the VoIP implementation is the *low-bit-rate speech codec*, whose compression rate and computation complexity may sometimes decide the smoothness of an IP call, especially when the bandwidth of the packet networks is limited. Some efforts from standard bodies are therefore devoted to its standardization. A milestone example is perhaps the finaliza-

tion of G.723.1, G.729 [9] and G.729A [16] by ITU.

G.723.1 6.3/5.3 kbits/s codec was originally designed for low-bit-rate videophones. G.729 8kbits/s codec was designed for wireless applications. G.729A was designed for simultaneous voice and data applications [5]. These three low-bit-rate codecs are now adopted as options by the Voice-on-Net standard, H.323. Further improvement on the performance of these codecs can be found in [15, 12]. A recent example of improving their speed by incorporating DSP technique is demonstrated in [8].

Although the codecs mentioned above guarantee low-bit-rate and good-quality speech for various applications, there are some possible causes that may refrain people from using these codecs, of which the major one is perhaps that these codecs are all patented, and hence, when being incorporated, some extra cost is required for paying the patent owners.

In 1991, the US General Services Administration published the Federal Standard 1016 (FS1016) [6], which specifies the conversion of analog voice to digital data by a method of 4.8 Kbps CELP. With the advantage of patent-free, FS1016 was selected by the TNCE, Inc. as an experimental codec for a sponsored research to our laboratory on a loss-packet recovery scheme specifically for use in the public Internet [17]. However successful in the loss-packet recovery capability of the modified FS1016 on the public Internet, its mass computational complexity limits its feasibility for real-time IP-phone applications. This leads us to the quest of improving its computation complexity without deterioration of its voice quality.

The rest of the thesis is organized as follows. Chapter 2 introduces the FS1016 codec, and surveys the techniques that have been proposed in the

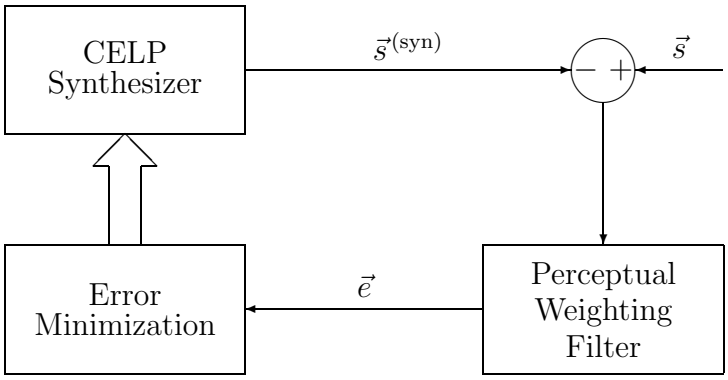
literature to improve its computational complexity. Chapter 3 and Chapter 4 present our newly proposed stochastic codebook search algorithm and adaptive codebook search algorithm for FS1016, respectively. Chapter 5 summarizes the complexity improvement in principle. Chapter 6 concludes the thesis.

Chapter 2

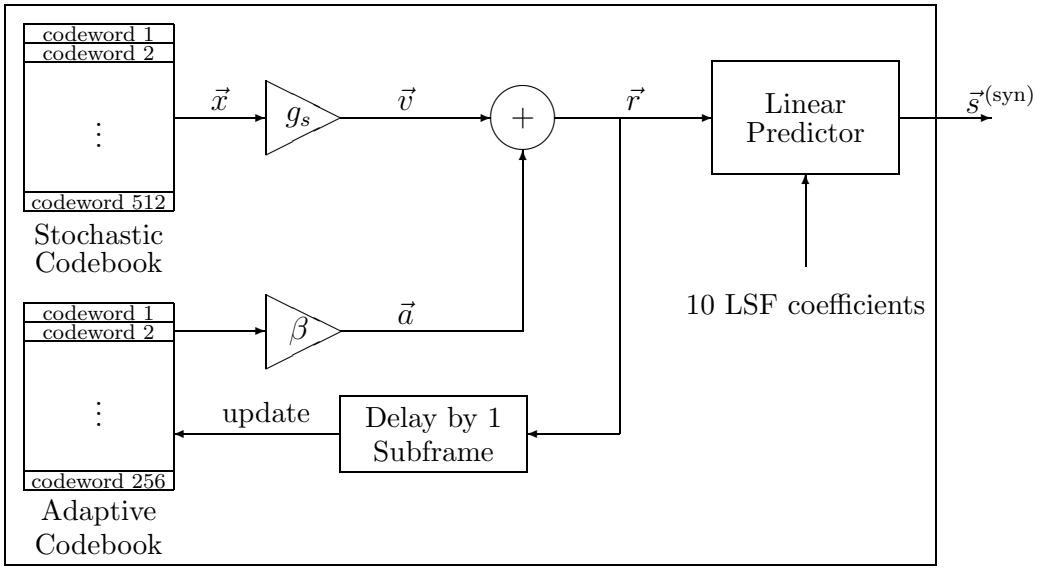
Backgrounds

2.1 Introduction to FS1016

The FS1016 standard is specified based on the Code Excited Linear Prediction (CELP) technique, which compresses sampled speech by analysis-by-synthesis approach [2, 13] (cf. Fig. 2.1), and is made up of three major parts: *short-term linear prediction*, *long-term adaptive codebook search* and *stochastic codebook search*.



(a) CELP analyzer for FS1016.



(b) The detailed functional block of the CELP synthesizer in (a).

Figure 2.1: Functional diagram of FS1016.

The input voice stream to FS1016 is segmented into frames of duration 30ms. Each frame is in turn divided into four subframes of length 7.5ms. Accordingly, with the sampling rate of 8 kHz, there are 240 voice samples per frame, and 60 samples per subframe. When speech is being analyzed, short-term linear prediction is first performed over the entire speech frame to extract the 10 linear prediction coefficients. Afterwards, long-term adaptive codebook search and stochastic codebook search are in sequence applied to each of the four subframes.

The procedure for speech synthesizing is simply a reverse of the speech analyzing.

2.1.1 Short-Term Linear Prediction

In the short-term linear prediction, the speech signal is reasonably assumed stationary within a small observation window [3]. An infinite impulse response (IIR) filter of order ten, named *Linear Predictive Coding* (LPC), is used to fit the spectrum envelope of the original speech signal (cf. Fig. 2.2), which emulates the filtering effect of the vocal tract. The formula of the LPC filter is

$$F_{LPC}(z) = \frac{1}{A(z)}, \tag{2.1}$$

where

$$A(z) = 1 - \sum_{j=1}^{10} \alpha_j z^{-j}. \tag{2.2}$$

In time domain, the synthesized output speech signal $s^{(\mathrm{syn})}(n)$ (i.e., synthesizing the true speech from the lips) due to the excitation input $r(n)$ (i.e.,

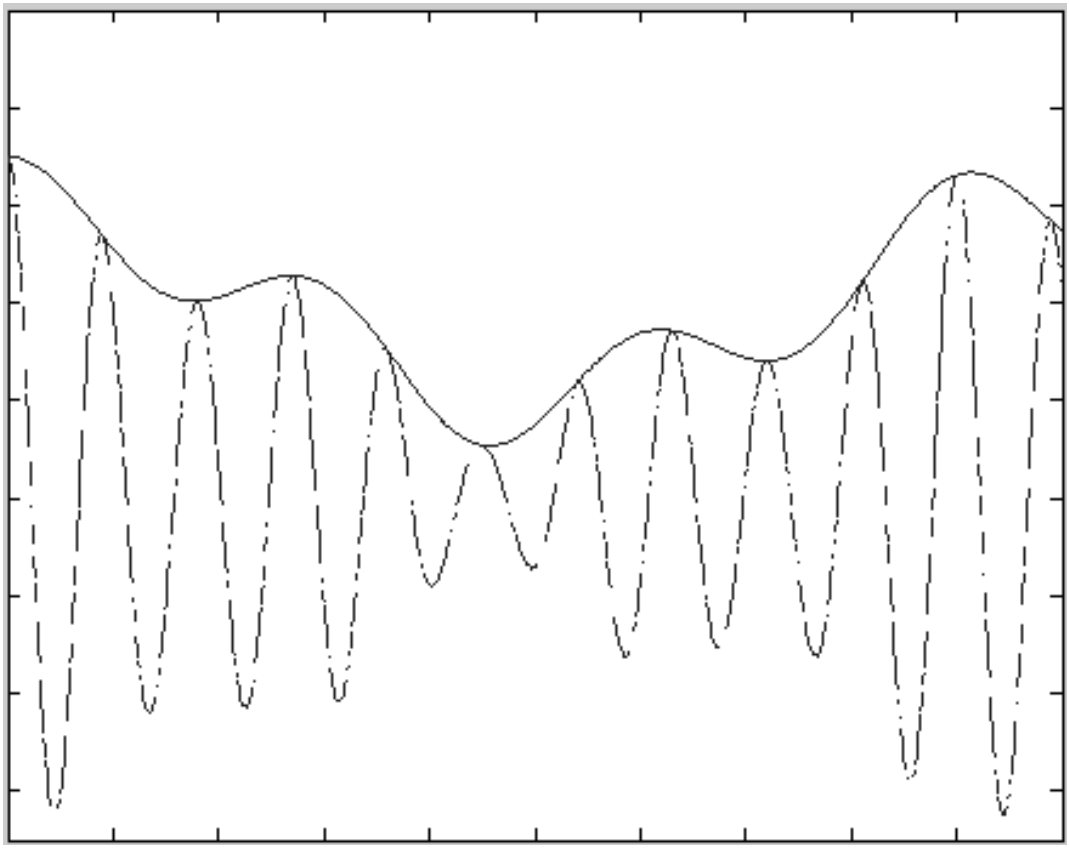


Figure 2.2: Spectrum envelop of the original speech signal.

signals from the glottal excitation) via the filter can be expressed by

$$s^{(\text{syn})}(n) = r(n) + \sum_{j=1}^{10} \alpha_j \cdot s^{(\text{syn})}(n-j). \tag{2.3}$$

Notably, at this stage, the functional blocks for adaptive codebook search and stochastic codebook search in Fig. 2.1(b) are both disabled; hence, $r(n)$ simply emulates an impulse excitation input to the Linear Predictor filter.

The objective of the linear prediction is to make the best estimate of the 10 linear coefficients α_1, α_2, ..., α_{10} such that the true speech $s(n)$ can be well-approximated by the synthesized speech $s^{(\text{syn})}(n)$ under an impulse glottal excitation input $r(n)$.

To cope with the human-ear perpetual effect on speech, an "adjustment" on the synthetic speech, as well as the true speech, is performed before the optimization of the 10 linear coefficients, which is named the *Perpetual Weighted Filter*. Hence, the residual signal $e(n)$ is equal to $w(n) * [s(n) - s^{(\text{syn})}(n)]$, where $w(n)$ is the impulse response of the perpetual weighted filter, and "$*$" denotes the convolution operation. As a result of the adjustment of the perpetual weighted filter, the best estimation of $\alpha_1, \ldots, \alpha_{10}$ is defined based on the minimization of

$$\sum_n e^2(n) = \sum_n \left[w(n) * \left(s(n) - s^{(\text{syn})}(n) \right) \right]^2 \tag{2.4}$$

$$= \sum_n \left[w(n) * \left(s(n) - \sum_{j=1}^{10} \alpha_j s^{(\text{syn})}(n-j) \right) \right]^2 . \tag{2.5}$$

Since the 10 LPC coefficients are frame-wisely computed by an open-loop estimation, its computational complexity, when being compared to the adaptive and stochastic codebook searches, is quite low. The challenge for this step is the quantization loss. As only limited number of bits is reserved for each coefficient, additional quantization errors are unavoidably introduced to the residual signal. In addition, the quantized LPC coefficients by no means guarantee the stability of the resultant IIR filter.

In order to amend the above problems, FS1016 chooses to quantize the *Line Spectral Frequency* (LSF) coefficients, which are conceptually one-to-one mappings of the LPC coefficients [10]. The LSFs are roots of a system function, and locate on the unit circle in the z-domain. Thus, they have the same amplitude, and are only different in their phases. The quantization applied to the LSFs therefore only affects the resultant phases. As a result, the stability of the system (filter) is less vulnerable through quantization.

Although for most of the time, the speech is short-term or frame-wisely stationary in its nature, it is still possible that the coefficients obtained from two consecutive frames are quite different. In order to generate a smooth speech at the decoding phase, FS1016 specifies a weighted interpolation to make a gentle migration in LSF coefficients between two consecutive frames. As illustrated in Fig. 2.3, the computed LPC coefficients for the proceeding frame (frame i) and the next frame (frame $i+1$) are transformed to their equivalent LSF coefficients, $f_i^1, f_i^2, ..., f_i^{10}$ and $f_{i+1}^1, f_{i+1}^2, ..., f_{i+1}^{10}$, for transmission. Then at the decoding phase, the respective LSF coefficients of the subframes of the current frame (cf. Fig. 2.3) are derived from:

$$\text{The LSFs of subframe 1} = (7/8)f_i^j + (1/8)f_{i+1}^j \quad \text{for } j = 1, \ldots, 10. \tag{2.6}$$

$$\text{The LSFs of subframe 2} = (5/8)f_i^j + (3/8)f_{i+1}^j \quad \text{for } j = 1, \ldots, 10. \tag{2.7}$$

$$\text{The LSFs of subframe 3} = (3/8)f_i^j + (5/8)f_{i+1}^j \quad \text{for } j = 1, \ldots, 10. \tag{2.8}$$

$$\text{The LSFs of subframe 4} = (1/8)f_i^j + (7/8)f_{i+1}^j \quad \text{for } j = 1, \ldots, 10. \tag{2.9}$$

The speech of each subframe is thereafter synthesized based on the corresponding interpolated LSF coefficients.

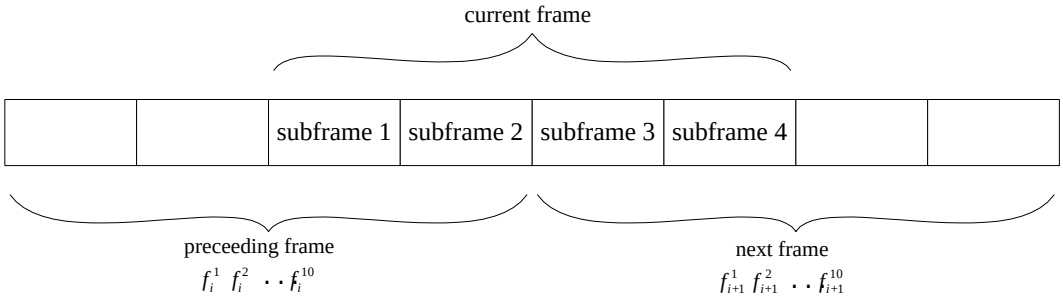


Figure 2.3: Interpolation of LSF coefficients.

Among the 10 LSF coefficients, 4 LSF coefficients are more perceptually sensitive to human ears. Accordingly, FS1016 reserves 4 bits for each of these 4 sensitive LSF coefficients, and puts 3 bits for each of the remaining 6 LSF coefficients. This sums to 34 bits for the 10 LSF coefficients, which consumes a bandwidth of 34 bits/30 msec = 1.133 kbps.

2.1.2 Long-Term Adaptive Codebook Search

In the previous subsection, only the Linear Predictor in the CELP synthesizer (cf. Fig. 2.1(b)) is activated for residual signal minimization, and both the adaptive codebook search and the stochastic codebook search are disabled. After the finding of the best LSP coefficients, the adaptive codebook search will then be activated for further minimization of the residual signal.

In principle, if the vocal tract filter can be accurately modelled by the linear predictor filter, then the residual signal presents exactly the glottal excitation signal. The glottal excitation signal is periodic in nature. Its period is named the *pitch period* or *pitch delay*, and the estimator of the pitch period is called the *Long-Term Predictor* (LTP) or simply the *Pitch Predictor*.

In FS1016, the synthetic glottal excitation signal that corresponds to optimal pitch period for each subframe is selected from an *adaptive codebook* through a closed-loop estimation scheme. The procedure is *adaptive* to the previous $r(n)$, that is, the combined output of the stochastic codebook search and the adaptive code book search due to the previous subframe (cf. Fig. 2.1(b)). The relation between the previous $r(n)$ and the current

pitch predictor output[1] $r(n)$ can be in concept characterized by the LTP filter:

$$F_{LTP}(z) = \frac{1}{1 - \beta \cdot z^{-T}}, \tag{2.10}$$

where T is the pitch period and β is the *pitch gain*. The time-domain expression due to input $r_{\text{initial}}(n)$ is therefore:

$$r(n) = r_{\text{initial}}(n) + \beta r(n - T).$$

FS1016 then searches the best estimates of pitch period T, among those prespecified 256 candidates, and its corresponding pitch gain β such that the minimum mean square of

$$\begin{aligned} \sum_n e^2(n) &= \sum_n \left[w(n) * \left(s(n) - s^{(\text{syn})}(n) \right) \right]^2 \\ &= \sum_n \left[w(n) * \left(s(n) - \beta \cdot r(n - T) - \sum_{j=1}^{10} \alpha_j s^{(\text{syn})}(n - j) \right) \right] \end{aligned} \tag{2.11}$$

is achieved.

All 256 adaptive codewords for selecting to minimize (2.11) are pre-made according to a 147-dimensional vector (ℓ_{-147}, ℓ_{-146}, ℓ_{-145}, ..., ℓ_{-1}) that is subframe-wisely updated according to the previous $r(n)$. Specifically, the update procedure is to remove the oldest 60 components of the 147-dimensional vector, followed by shifting in the 60 components of the previous $r(n)$, i.e.,

$$\ell_{-61} = \ell_{-1}, \ell_{-62} = \ell_{-2}, \ldots, \ell_{-147} = \ell_{-87}$$

and

$$\ell_{-1} = r_{59}, \ell_{-2} = r_{58}, \ldots, \ell_{-60} = r_0.$$

[1]Notably, the stochastic codebook search is disabled at this stage; hence, the current $v(n)$ is set 0 at this moment.

The initial value of this 147-dimensional vector is simply zero.

The first 88 integer-valued-delay 60-dimensional codewords are made up of

$$\begin{array}{cccc} \ell_{-147}, & \ell_{-146}, & \cdots, & \ell_{-88} \\ \ell_{-146}, & \ell_{-145}, & \cdots, & \ell_{-87} \\ \vdots & \vdots & \vdots & \vdots \\ \ell_{-60}, & \ell_{-59}, & \cdots & \ell_{-1}. \end{array}$$

The remaining 40 integer-valued-delay 60-dimensional codewords are defined by

$$\begin{array}{cccccccc} \ell_{-59}, & \ell_{-58}, & \cdots, & \ell_{-3}, & \ell_{-2}, & \ell_{-1}, & \ell_{-59} \\ \ell_{-58}, & \ell_{-57}, & \cdots, & \ell_{-2}, & \ell_{-1}, & \ell_{-58}, & \ell_{-57} \\ \ell_{-57}, & \ell_{-56}, & \cdots, & \ell_{-1}, & \ell_{-57}, & \ell_{-56}, & \ell_{-55} \\ \vdots, & \vdots, & \cdots, & \vdots, & \vdots, & \vdots, & \vdots \\ \ell_{-20}, & \ell_{-19}, & \cdots, \ell_{-1}, \ell_{-20}, \ell_{-19}, \cdots, \ell_{-1}, & \ell_{-20}, & \ell_{-19}, & \ldots, & \ell_{-1}, \end{array}$$

Additional 128 non-integer-valued-delay codewords are obtained by interpolating the two nearest integer-valued-delay codewords. The pitch gain ranges from -1 to 2, is quantized discordantly with equal number of bits assigned for each subframe.

In FS1016, the approaches to search the optimal pitch delays for odd-numbered subframes and even-numbered subframes are different. Based on the maximum match score criterion, the optimal pitch delay (and the corresponding optimal gain) for each odd-numbered subframe is first selected from the 128 integer-valued delays. Denote the resultant optimal integer-valued pitch delay and the corresponding match score by T^* and m^*, respectively. Then, FS1016 tests whether the largest match score corresponding to the sub-multiple pitch delays $(1/2)T^*$, $(1/3)T^*$ and $(1/4)T^*$ is within 1 dB of m^*. If so, update T^* and m^* by the respective sub-multiple pitch delay and match scores. In the end, FS1016 examines the match scores of those

non-integer-valued-delay codewords whose pitch delays are within $I^* - 3$ and $I^* + 3$, where I^* is the corresponding codeword index of T^*, and update the optimal pitch delay once a larger match score than the previous optimal match score is located. The above procedure is specified in FS1016 as *non-full search mode*. As anticipated, if *full-search mode* is adopted, all the 256 candidate codewords are truly examined.

As for the even-numbered subframes, efforts are redirected to locate the optimal pitch delay offset relative to the optimal pitch delay of the previous subframe. Specifically, if the optimal codeword for the previous odd-numbered subframe is indexed by i, then FS1016 searches only the codewords whose indices range from $j = \min[\max(i-31, 1), 193]$, $j+1$, ..., $j+63$ for the current even-numbered subframe. Again, in the *non-full search mode*, only the integer-valued pitch delays (belonging to the 64 candidates) are tested, which yields the optimal integer-valued pitch delay T^* with maximum match score m^*. Afterward, the match scores corresponding to those non-integer-valued delays within $I^* - 3$ and $I^* + 3$ are examined, where I^* is the corresponding codeword index of T^*, and the codeword with the largest match score, among those examined ones, is outputted. Notably, no sub-multiple pitch delays are examined for even-numbered subframes.

Since taking the *non-full search mode* reduces the computational complexity with negligible loss of speech quality, it is taken as default mode except otherwise stated throughout the thesis.

In total, FS1016 distributes 48 bits for the pitch delays and pitch gains of the four subframes in a frame, in which 8 bits and 6 bits are reserved for pitch delays in odd frame and in even frame, respectively. There resultant

transmission rate is thus 48 bits/30 msec = 1.6 kbps.

2.1.3 Stochastic Codebook Search

After the LPC analysis and the pitch prediction, the residual signal become aperiodic, which is often referred to as the *innovation signal*. The noise-like innovation signal, although lack of speech information, can not be neglected. In its absence, the speech will sound awkward and artificial.

In FS1016, a stochastic codebook is employed to approximate the innovation signal. Each codeword in the stochastic codebook has its own index. There are totally 512 codewords in the codebook. To speed up the search process, reduced-size codebooks of 256, 128 and 64 codewords are also specified in FS1016. Nevertheless, better speech quality is achieved by using the codebook with more codewords.

Analogous to the adaptive codebook search, the stochastic codebook search is performed per subframe by a closed-loop operation. Also adopted is the minimum squared error criterion. To facilitate the codebook search, the 512 codewords in the stochastic codebook are drawn from a one-dimensional array of $(+1, 0, -1)$ value with approximately 77% zeros. The consecutive codewords overlap except at the first and the last two components. Such a codebook design has several advantages:

- Only two bits are required to represent the ternary values, $+1$, -1 and 0.
- Multiplying with $+1$ and -1 can be replaced by sign changes, which greatly reduces the computational complexity.

- Adding the product of a term and zero is equivalent to remain unchanged in the original quantity, and the chance of meeting a zero is as high as 77%.

- When convolution operations are performed for two consecutive codewords, the convolution for the second codeword can retain those convolved results obtained from its overlapped part with the previous codeword, and reduce the computational complexity.

In total, FS1016 distributes 56 bits for the stochastic codebook search. The index for the best-fit codeword in the stochastic codebook requires 9 bits for each subframe. The stochastic gain lies between -1330 and 1330 discordantly, and each of the four gains consumes 5 bits. These summarize to the transmission rate of 56 bits/30 msec = 1.866 kbps.

2.2 Issues Regarding Computational Complexity

As aforementioned, the mass computational complexity of FS1016 limits its potential for real-time applications. In [4], Campbell et al evaluated the individual computational complexity for each component of FS1016. Their results showed that the adaptive codebook search consumes about 2.13 MIPs per frame. They also obtained that the stochastic codebook search requires about 0.0270 MIPs for the first codeword, and 0.00399 MIPs for each of the remaining codewords. Hence, for a stochastic codebook of 512 codewords, the average computational load per frame is up to 8.3 MIPs. These two search algorithms indeed cost 10.43 MIPs in total, which largely dominates

the computation requirement of the remaining parts of FS1016.

A true implementation of FS1016 over a 40MHz TMS320c44 floating-point Digital Signal Processing (DSP) board [18], again, confirms the observations in [4]. The time consumed for each part of FS1016 is listed in Tab. 2.1.

Module	Upper bound of time consumed
LPC Analysis, LSF conversion and other operations	0.75 ms
Adaptive codebook search algorithm	108 ms
Stochastic codebook search algorithm	250 ms

Table 2.1: Time consumed in each part of FS1016.

From these two reports, it is clear that the codebook searches greatly dominate the entire execution time. Hence, the speed-up of FS1016 has to first accelerate the two codebook search algorithms.

One possible mean to obtain a quick codebook search algorithm is to re-design the codebook so that a quick search algorithm can be employed. For example, Kao [11] systematically re-structure the codebook so that for each codeword, the 60 ternary values contains exactly 48 zeros, and the remaining 12 non-zero values distribute uniformly over the entire codeword. In other words, if the codeword is treated as a 60-dimensional vector $(c_1, c_2, c_3, \ldots, c_{60}) \in \{-1, 0, +1\}^{60}$, the 12 non-zero terms only reside at c_1, c_6, c_{11}, c_{16}, ..., c_{56} (cf. Fig. 2.4). Since human being is perceptually insensitive to phase variation of speech signal, Kao's artificial well-structured and non-noise-like stochastic codebook has very little impact on the resultant speech quality. Yet, through the new arrangement, the non-zero terms are known to locate in the fixed

positions, and therefore, dramatically reduce the computational complexity of the stochastic codebook search. Other example of new codebook design in terms of random-walk statistics can be found in [1].

One drawback of adopting a new codebook is that the voice quality has to be re-assured by human subjective testing. Another drawback is that the resultant new FS1016 may not be backwardly compatible to the original FS1016. Notably, backward-compatibility to earlier products is usually a basic requirement for commercialized products. This is the main reason why, unlike the work by Kao, Ahmed and Suwaiyel, we focus on the modification of the search algorithm itself, and retain the original FS1016 codebook. In such an approach, the voice quality need not be re-assured, and backward compatibility is secured. Details will be given in the subsequent chapters.

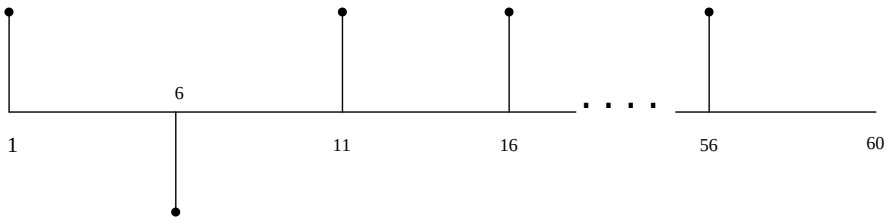


Figure 2.4: Stochastic codeword designed in [11].

Chapter 3

Stochastic Codebook Search Algorithm

3.1 Stochastic Codebook Search Criterion of FS1016

As illustrated in Fig. 2.1(b), the stochastic codebook search algorithm in FS1016 locates the 60-dimensional codeword $\vec{x}$ from a stochastic codebook of size 512 with the respective optimal multiplicative scalar g_s, which optimizes speech quality. Let i be the index of the codeword. Then multiplying codeword $\vec{x}_i$, where $i \leq 1 \leq 512$, with its respective optimal gain g_i yields the corresponding *stochastic excitation vector* $\vec{v}_i = g_i\vec{x}_i$. The synthetic speech $\vec{s}_i^{(\mathrm{syn})}$ is therefore produced by passing the combination of the optimal adaptive excitation vector $\vec{a}$ and the ith stochastic excitation vector $\vec{v}_i$ through the linear predictor, that is,

$$\vec{s}_i^{(\mathrm{syn})} = \mathbf{L}(\vec{a} + \vec{v}_i) + \vec{s}_0, \tag{3.1}$$

where $\mathbf{L}$ represents the equivalent matrix representation of the linear prediction filter that is of the form:

$$\mathbf{L} = \begin{pmatrix} l_1 & 0 & 0 & 0 & 0 \\ l_2 & l_1 & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & 0 & 0 \\ l_{59} & l_{58} & \dots & l_1 & 0 \\ l_{60} & l_{59} & \dots & l_2 & l_1 \end{pmatrix},$$

and $\vec{s}_0$ is the zero input response of the linear prediction filter $\mathbf{L}$. Let $\mathbf{W}$ be the equivalent matrix representation of the perceptual weighting filter. It turns out that $\mathbf{W}$ is also a lower triangular matrix. Then, the weighted error signal due to the difference between the actual speech and the synthetic speech is

$$\begin{aligned} \vec{e}_i &= \mathbf{W}(\vec{s} - \vec{s}_i^{(\mathrm{syn})}) \\ &= \mathbf{W}\vec{s} - \mathbf{WL}\vec{a} - \mathbf{WL}\vec{v}_i - \mathbf{W}\vec{s}_0 \\ &= \vec{t} - g_i\mathbf{M}\vec{x}_i, \end{aligned} \tag{3.2}$$

where, for simplicity, let $\mathbf{M} = \mathbf{WL}$, and $\vec{t} = \mathbf{W}\vec{s} - \mathbf{WL}\vec{a} - \mathbf{W}\vec{s}_0$.

Because of the 450 components that constitute the lowest triangle of matrix $\mathbf{M}$ are nearly zeros, they can be replaced by zeros with negligible voice quality degradation, and the computational complexity of the stochastic codebook search can be fairly reduced. The resultant $\mathbf{M}$ is simplified to:

$$\mathbf{M} = \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & \cdots & m_{1,60} \\ m_{2,1} & m_{2,2} & m_{2,3} & \cdots & m_{2,60} \\ m_{3,1} & m_{3,2} & m_{3,3} & \cdots & m_{3,60} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{60,1} & m_{60,2} & m_{60,3} & \cdots & m_{60,60} \end{pmatrix}$$

$$= \begin{pmatrix} m_1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ m_2 & m_1 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & 0 & 0 & \cdots & 0 \\ m_{30} & m_{29} & \cdots & m_1 & 0 & \cdots & 0 \\ 0 & m_{30} & \cdots & m_2 & m_1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \cdots & 0 \\ 0 & \cdots & 0 & m_{30} & m_{29} & \cdots & m_1 \end{pmatrix}.$$

The square error for selecting $\vec{v}_i$ is given by:

$$\|\vec{e}_i\|^2 = \vec{e}_i^T \vec{e}_i = \vec{t}^T \vec{t} - 2g_i \vec{y}_i^T \vec{t} + g_i^2 \vec{y}_i^T \vec{y}_i, \tag{3.3}$$

where $\vec{y}_i = \mathbf{M}\vec{x}_i$, and the superscript "$T$" represents the matrix transpose operation.

By taking the partial derivative of (3.3) with respective to g_i, we obtain:

$$\frac{\partial \|\vec{e}_i\|^2}{\partial g_i} = -2\vec{y}_i^T \vec{t} + 2g_i \vec{y}_i^T \vec{y}_i = 0, \tag{3.4}$$

which gives the optimal g_i as:

$$g_i = \frac{\vec{y}_i^T \vec{t}}{\vec{y}_i^T \vec{y}_i}. \tag{3.5}$$

Since the first term $\vec{t}^T \vec{t}$ in (3.3) is a constant independent of i, the match score in FS1016, which is aimed to maximize, is therefore defined as

$$matchScore = 2g_i \vec{y}_i^T \vec{t} - g_i^2 \vec{y}_i^T \vec{y}_i, \tag{3.6}$$

Taking the optimal g_i into (3.6) yields that

$$matchScore = \frac{(\vec{y}_i^T \vec{t})^2}{\vec{y}_i^T \vec{y}_i}. \tag{3.7}$$

The quantities of $\vec{y}_i^T \vec{t}$ and $\vec{y}_i^T \vec{y}_i$ are referred to as *correlation* and *energy* of filtered codeword $\vec{y}_i = \mathbf{M}\vec{x}_i$, respectively. As a result, to search the best stochastic codeword with minimum weighted mean square is equivalent to finding a codeword that maximizes the *matchScore* in (3.7).

3.2 Modification of Stochastic Codebook Search Algorithm

In order to reduce the complexity of the stochastic codebook search algorithm, the principle computations required in the algorithm are first addressed.

The stochastic codebook search process is performed per subframe, and 512 candidate codewords must be tested for match score optimization for each subframe. The first step of the match score computation is to calculate the filter output $\vec{y}_i$, which requires a convolution of the filter input $\vec{x}_i$ and the filter matrix $\mathbf{M}$. Afterwards, inner product between the filter output $\vec{y}_i$ and a pre-calculated target vector $\vec{t}$ is performed, which yields the *correlation* of the filter output to the targeted vector $\vec{t}$. Then, the energy of the filter output $\vec{y}_i$ is calculated, which requires another inner product operation of two 60-dimensional vectors. Finally, the match score is equal to the square of the correlation dividing by the energy. The above procedure is repeated 512 times, and thus is quite time-consuming.

FS1016 developed some techniques to speed up the process, such as neglecting the lowest triangle of matrix $\mathbf{M}$. The key accelerating technique, however, is from the ingenious codebook design. As mentioned in the previous chapter, the 512 codewords in the stochastic codebook are drawn from a one-dimensional array of $(+1, 0, -1)$ value with approximately 77% zeros. The consecutive codewords overlap except at the first and the last two components. Therefore, the calculation of the next filter output can be simplified to *sequential additions*, *no operations* or *sequential subtractions* on the previous

filter output, depending on whether the value of the next visited codeword component is $+1$, 0 or -1 (cf. Fig. 3.3). Additionally, the calculation of the energy of the filter output can be reduced to *two subtractions* from the previously obtained energy, if $y_1 = 1 \&\& y_2 = 0$ (cf. Fig. 3.2).

Along the research line of simplifying the computation complexity of the stochastic codebook search, we first observe that the rank of the match scores is mainly determined by the numerator $(\vec{y}_i^T \vec{t})^2$. To uphold this observation, four experiments over Chinese female, Chinese male, American female and American male voice streams are performed. The database for each experiment consists of 100 subframes. As depicted in Fig. 3.1, 90% of the top correlation square (i.e., $(\vec{y}_i^T \vec{t})^2$) has a match score rank under seven (among 512 candidates). This experimental results justify our taking the correlation square (or equivalently, the absolute value of the correlation) as the maximization criterion, rather than the match score.

In light of the new observation, we modify the stochastic codebook search algorithm by using the absolute correlation as a new maximizing criterion. Thus, the new match score becomes

$$\left|\vec{y}_i^T \vec{t}\right| \;=\; \vec{x}_i^T \mathbf{M}^T \vec{t} = \vec{x}_i^T \vec{\lambda},$$

where $\vec{\lambda} = \mathbf{M}^T \vec{t}$. Because $\vec{x}_i$ is only of three values, i.e., $+1$, 0 and -1, and $\vec{\lambda}$ can be pre-calculated, the newly proposed matching score can yield a very simple and quick stochastic codebook search algorithm. By the experimental results in Fig. 3.1, the new algorithm should be backwardly compatible to the original algorithm. The resultant new algorithm is shown in Figs. 3.4 and 3.5. For completeness, the original stochastic codebook search algorithm is

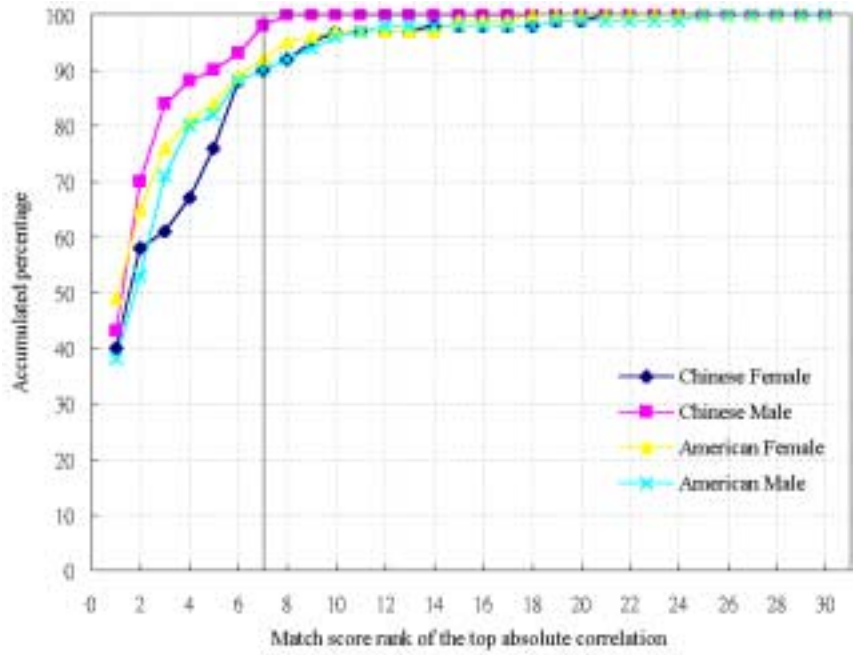


Figure 3.1: Distribution of the match score rank of the top absolute correlation for ChineseFemale.wav, ChineseMale.wav, AmericanFemale.wav and AmericanMale.wav.

also displayed in Figs. 3.2 and 3.3. Speed comparisons between the original algorithm and the newly proposed algorithm will be addressed in Chapter 5.

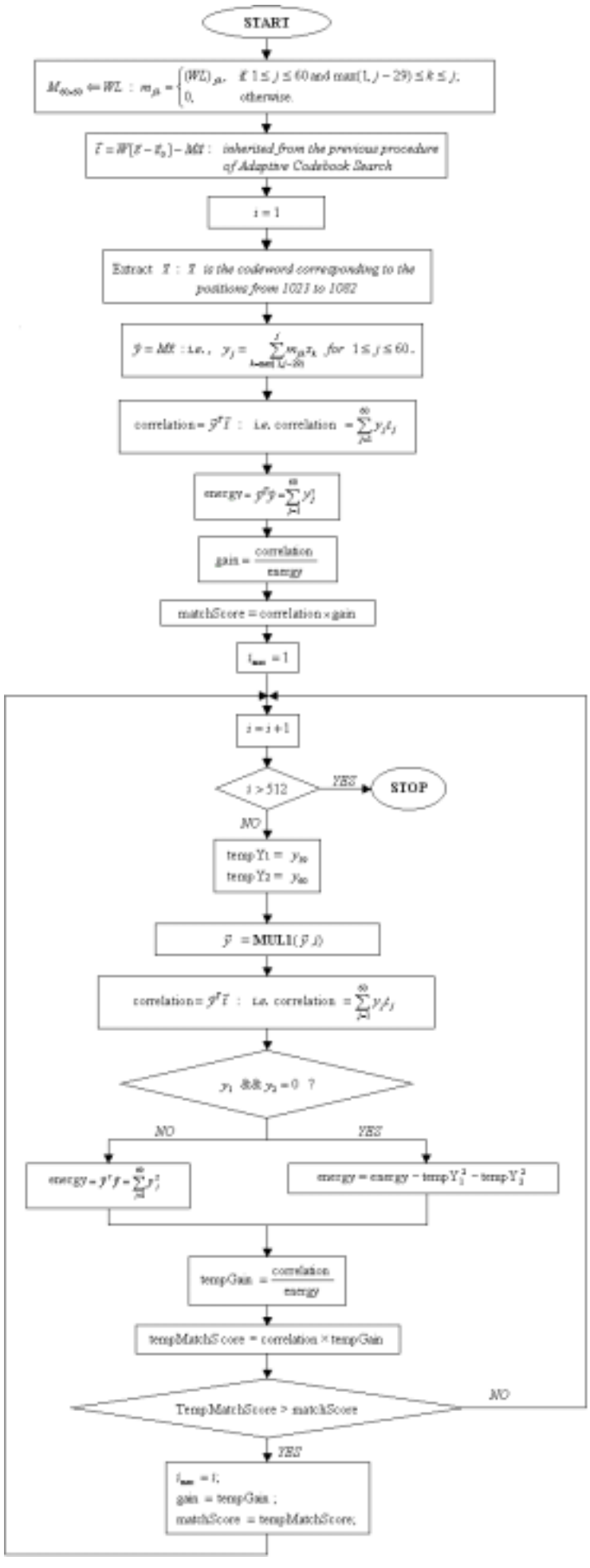


Figure 3.2: The original stochastic codebook search algorithm of FS1016.

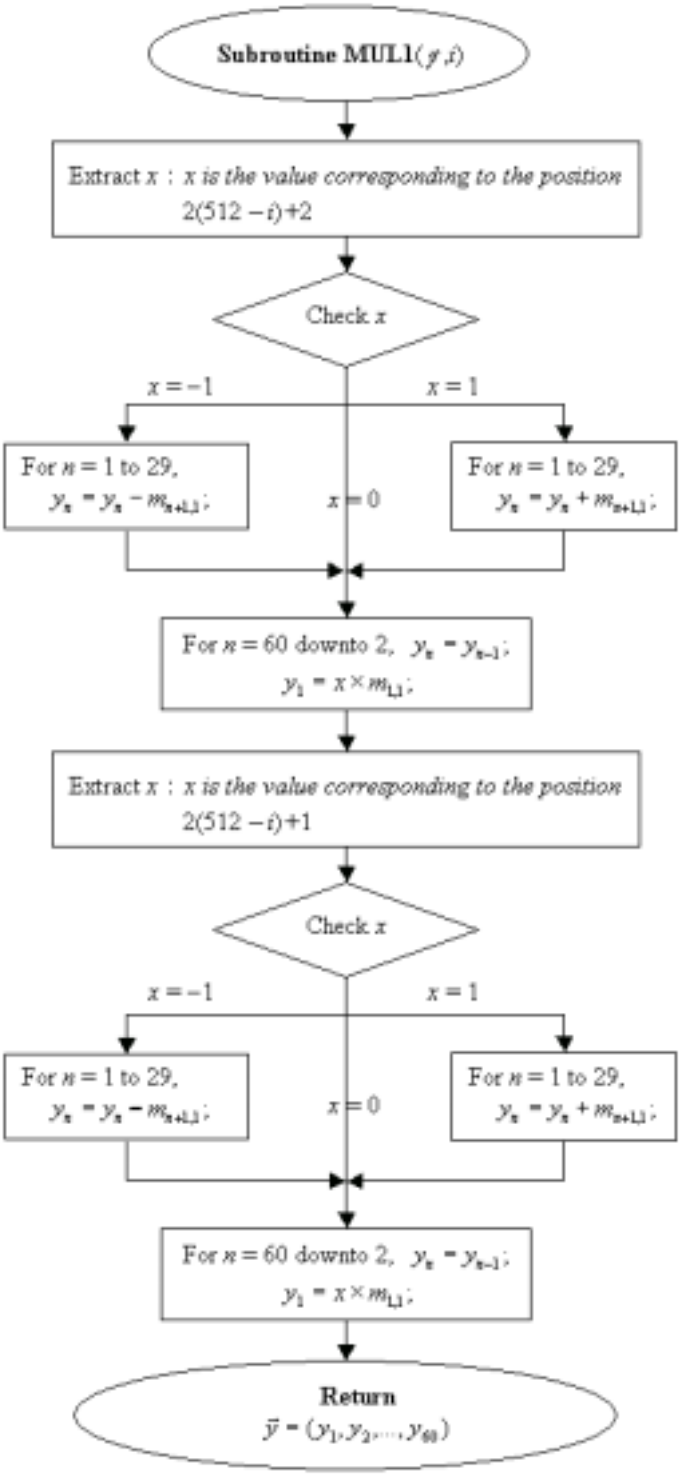


Figure 3.3: The subroutine used in the algorithm displayed in Fig. 3.2.

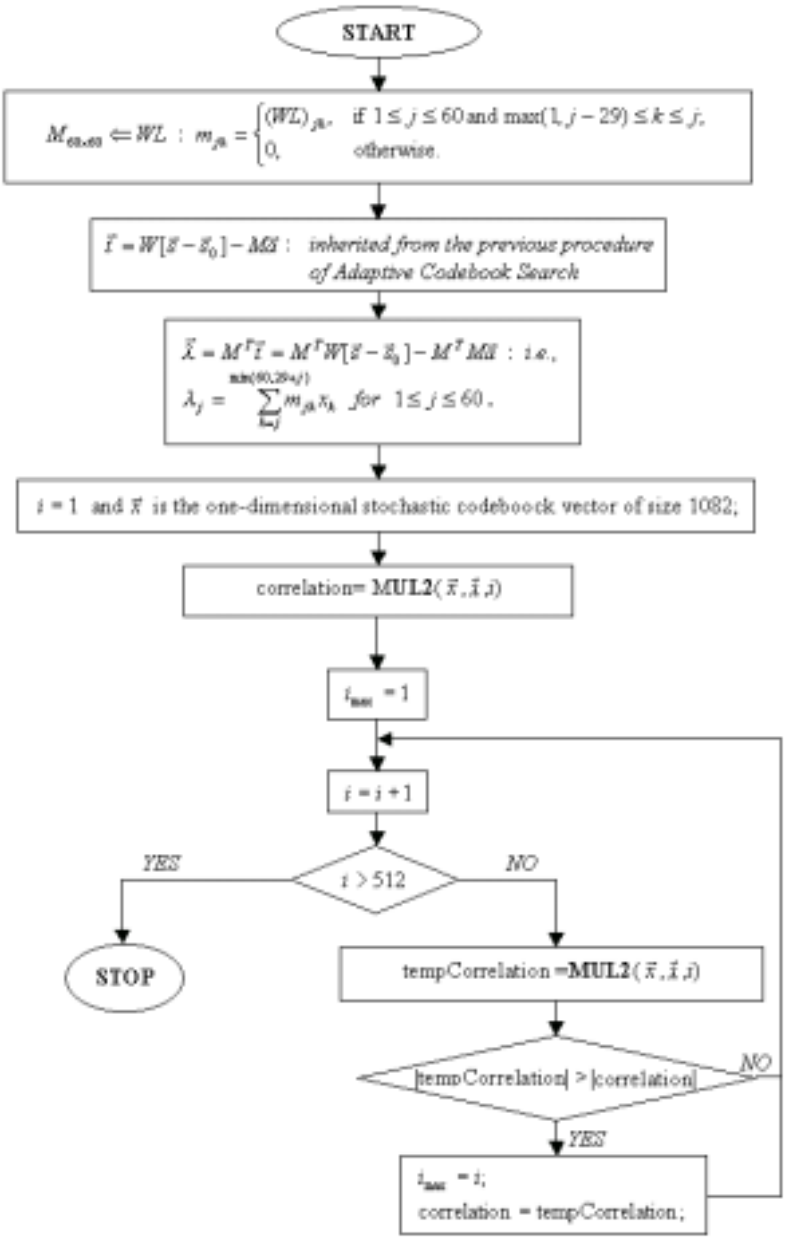


Figure 3.4: Our proposed algorithm for the stochastic codebook search.

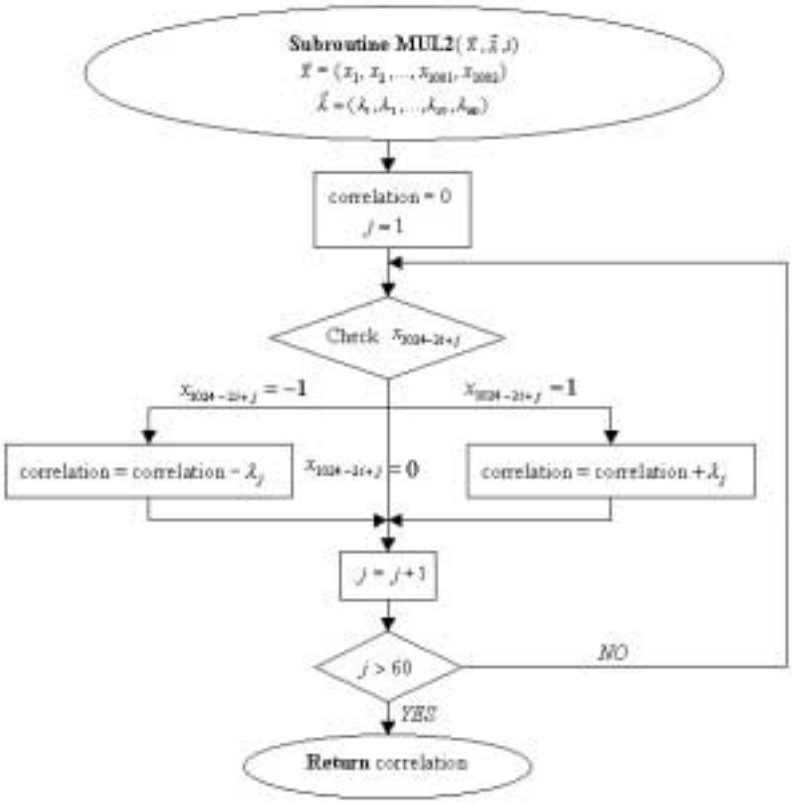


Figure 3.5: The subroutine used in the algorithm displayed in Fig. 3.4.

Chapter 4

Adaptive Codebook Search Algorithm

As presented at the end of Chapter 2, most of the computational time for FS1016 is contributed by the stochastic codebook search. After accelerating the stochastic codebook search, the bottleneck computational effort of FS1016 becomes the adaptive codebook search. We therefore continue our quest of reducing the FS1016 computational time to the new bottleneck.

As stated in Chapter 2, by disabling the stochastic codebook search output $\vec{v}$, the LPT filter gives that

$$r_i(n) = r_{\text{initial}}(n) + \beta_i r_i(n - T_i), \tag{4.1}$$

where i represents the index of the adaptive codewords. Equation (4.1) (or equivalently, the adaptive codebook search operation) can be transformed to its equivalent vector expression as:

$$\vec{r}_i \quad = \quad \beta_i \vec{r}_{-T_i} = \vec{a}_i. \tag{4.2}$$

Notably, the pitch delay T_i can be a non-integer number. The weighted error

signal is therefore equal to:

$$\begin{aligned} \vec{e}_i &= \mathbf{W}(\vec{s} - \vec{s}_i^{(\text{syn})}) \\ &= \mathbf{W}\vec{s} - \mathbf{WL}\vec{r}_i - \mathbf{W}\vec{s}_0 \\ &= \vec{t}_a - \mathbf{M}\vec{r}_i, \end{aligned} \tag{4.3}$$

where

$$\vec{t}_a = \mathbf{W}(\vec{s} - \vec{s}_0) \tag{4.4}$$

is inherited from the previous short-term linear prediction procedure. Hence, replacing $\vec{r}_i$ by $\beta_i \vec{r}_{-T_i}$ yields that

$$\begin{aligned} \|\vec{e}_i\|^2 &= < \vec{t}_a - \beta_i \mathbf{M}\vec{r}_{-T_i}, \vec{t}_a - \beta_i \mathbf{M}\vec{r}_{-T_i} > \\ &= \vec{t}_a^T \vec{t}_a - 2\beta_i \vec{y}_{a,i}^T \vec{t}_a + \beta_i^2 \vec{y}_{a,i}^T \vec{y}_{a,i}, \end{aligned} \tag{4.5}$$

where $\vec{y}_{a,i} = \mathbf{M}\vec{r}_{-T_i}$. Here, we abuse the notation of T to represent both the pitch delay and the transpose operation indicator, since no ambiguity is aroused.

Now similar to the derivation of the stochastic codebook search, the partial derivative with respective β_i gives that

$$\beta_i = \frac{\vec{y}_{a,i}^T \vec{t}_a}{\vec{y}_{a,i}^T \vec{y}_{a,i}}. \tag{4.6}$$

By substituting (4.6) into (4.5), we obtain

$$\|\vec{e}_i\|^2 = \vec{t}_a^T \vec{t}_a - \frac{(\vec{y}_{a,i}^T \vec{t}_a)^2}{\vec{y}_{a,i}^T \vec{y}_{a,i}}, \tag{4.7}$$

and the corresponding match score to be maximized is

$$matchScore = \frac{(\vec{y}_{a,i}^T \vec{t}_a)^2}{\vec{y}_{a,i}^T \vec{y}_{a,i}}. \tag{4.8}$$

Again, similar experiments are performed in this stage to examine the dependence of *matchScore* and *correlation*. The results are summarized in Figs. 4.1–4.4. In these figures, "subframe range 1" corresponds to the subframes that range from 1 to 100, and "subframe range 2", from 501 to 600.

It can be seen that from the experiments on subframe range 1, the codeword with maximum absolute *correlation* almost get the largest *matchScore*; however, for those subframes in subframe range 2, the degree of dependence between *correlation* and *matchScore* decreases. Therefore, by replacing the maximizing match score criterion with the maximizing absolute correlation criterion, some degradation to voice quality can be sensed.

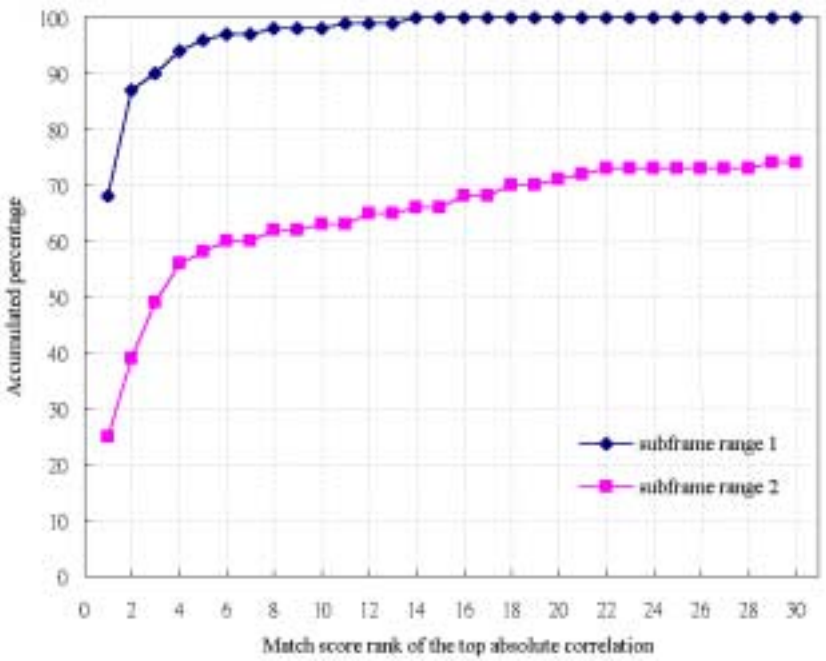


Figure 4.1: Distribution of the match score rank of the top absolute correlation for ChineseFemale.wav.

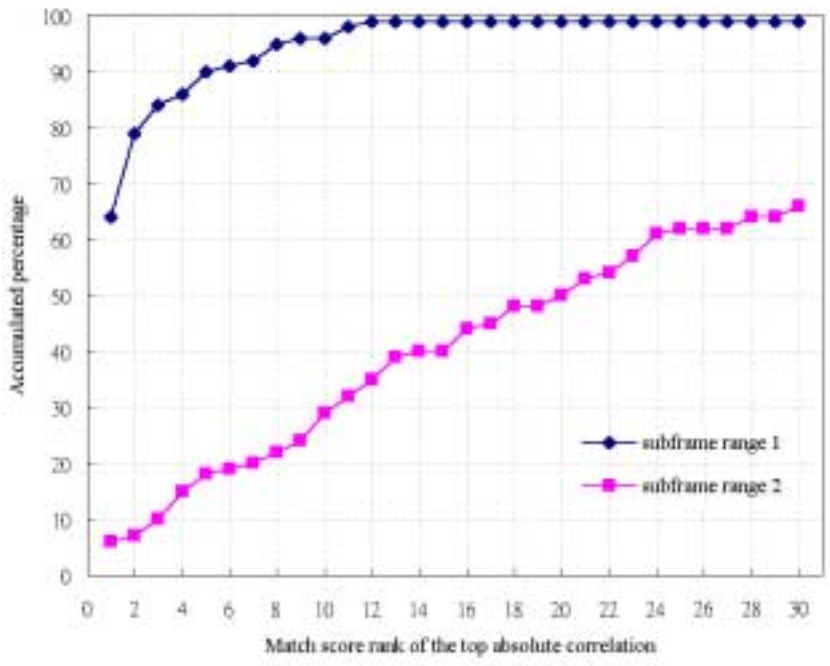


Figure 4.2: Distribution of the match score rank of the top absolute correlation for ChineseMale.wav.

The acceleration of adaptive codebook search by mean of maximizing the absolute correlation is not so significant as that of stochastic codebook search. The reasons are as follows. First, the components of adaptive codewords are floating-point numbers, while the components of the stochastic codewords are simple integers (that is, +1, 0 and −1). As a consequence, multiplications can not be replaced by additions as illustrated in Figs. 3.3 and 3.5. Secondly, unlike the stochastic search algorithm that originally searches 512 codewords for each subframe, the adaptive codebook search algorithm does not search all the 256 codewords for each subframe. Seme efforts are already devoted to simplify the search overhead of each even-numbered subframe (as mentioned in Chapter 2). In addition, there are some residential overhead that cannot

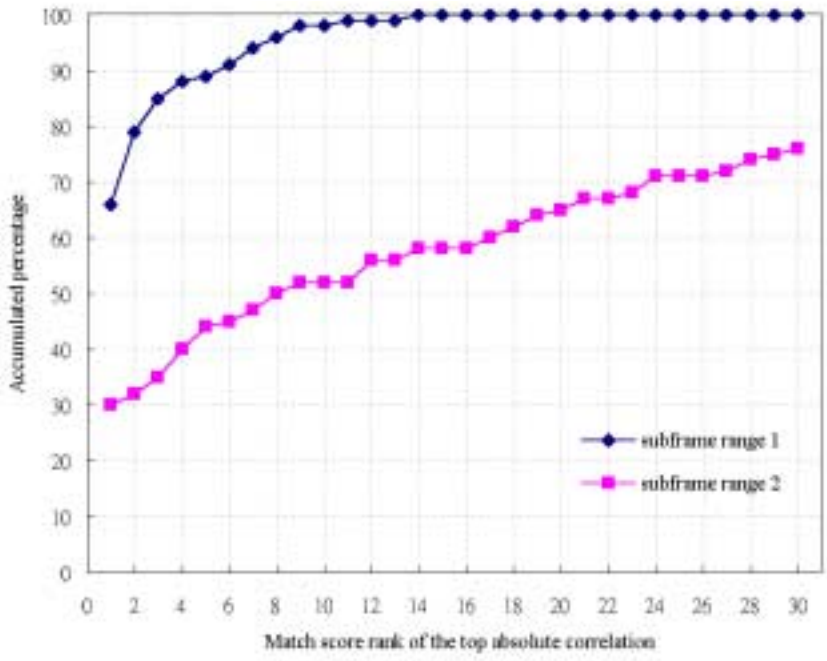


Figure 4.3: Distribution of the match score rank of the top absolute correlation for AmericanFemale.wav.

be removed, such as updating the adaptive codewords. For these reasons, the resultant new search algorithm can only be made a little faster than the original algorithm. Detailed comparison of computational complexity between the original algorithm and the new algorithm will be provided in subsequent chapter.

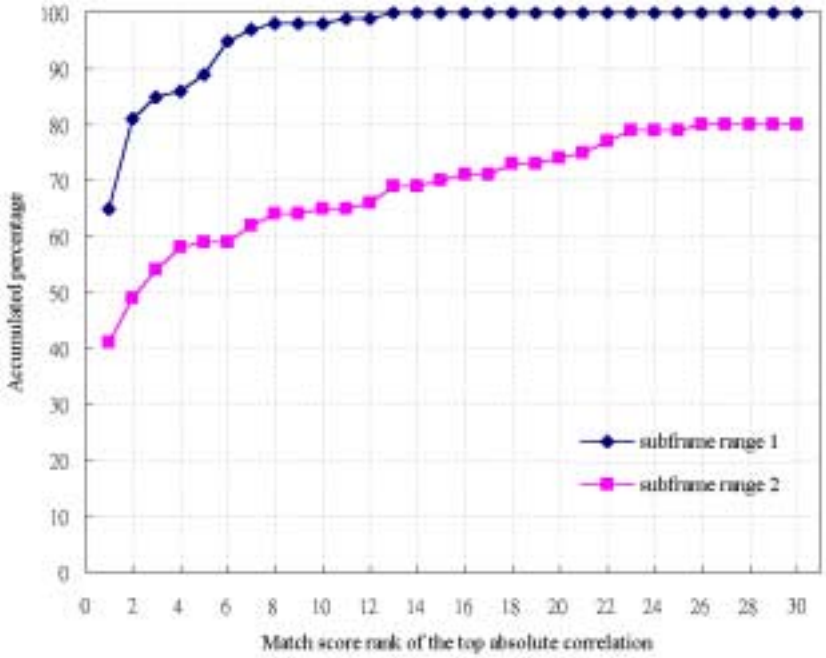


Figure 4.4: Distribution of the match score rank of the top absolute correlation for AmericanMale.wav.

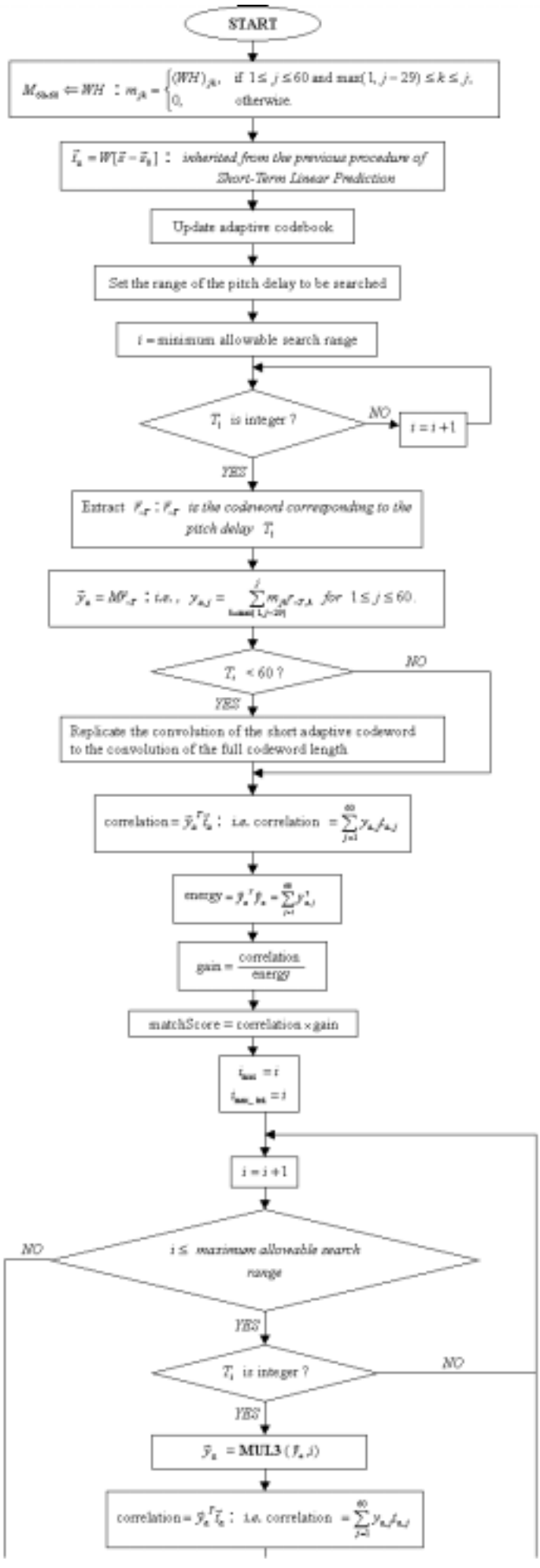


Figure 4.5: The original adaptive codebook search algorithm of FS1016.

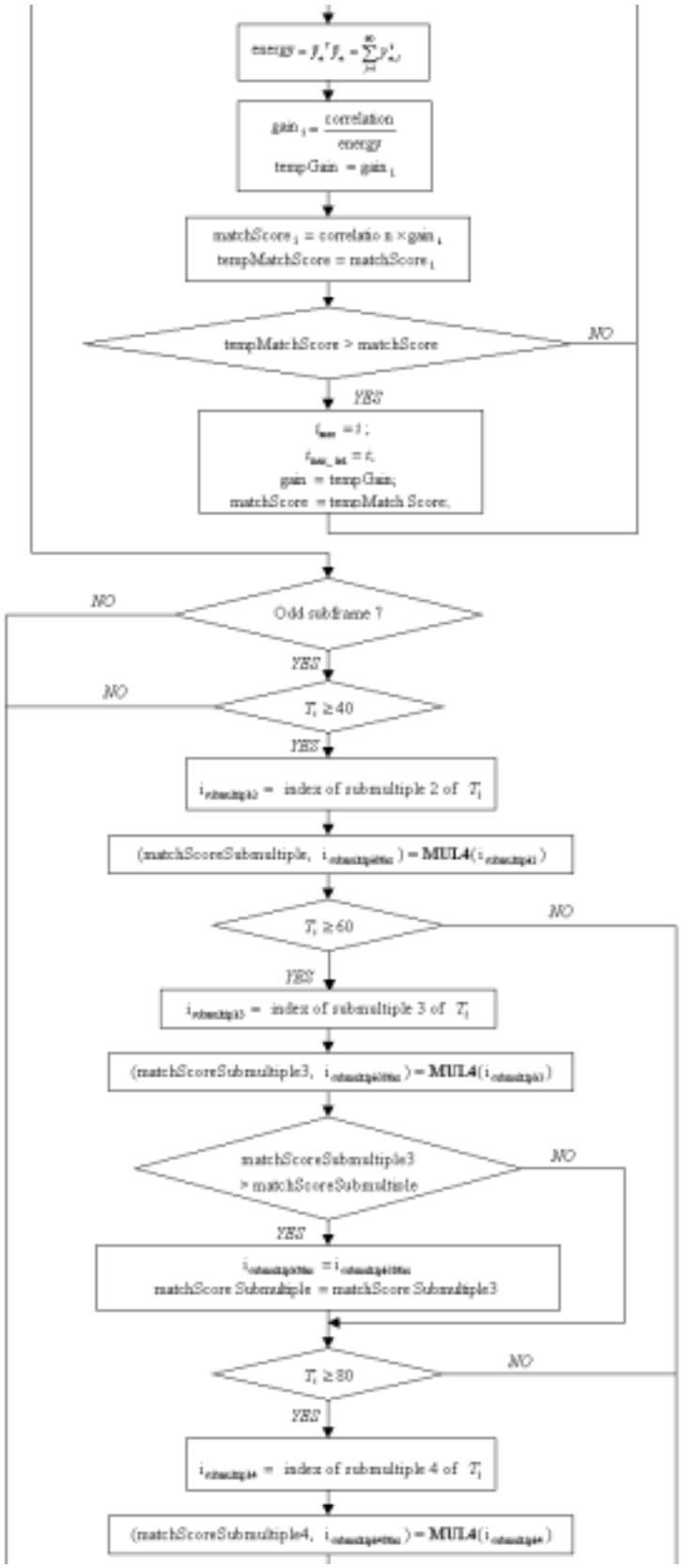


Figure 4.6: Continue from Fig. 4.5.

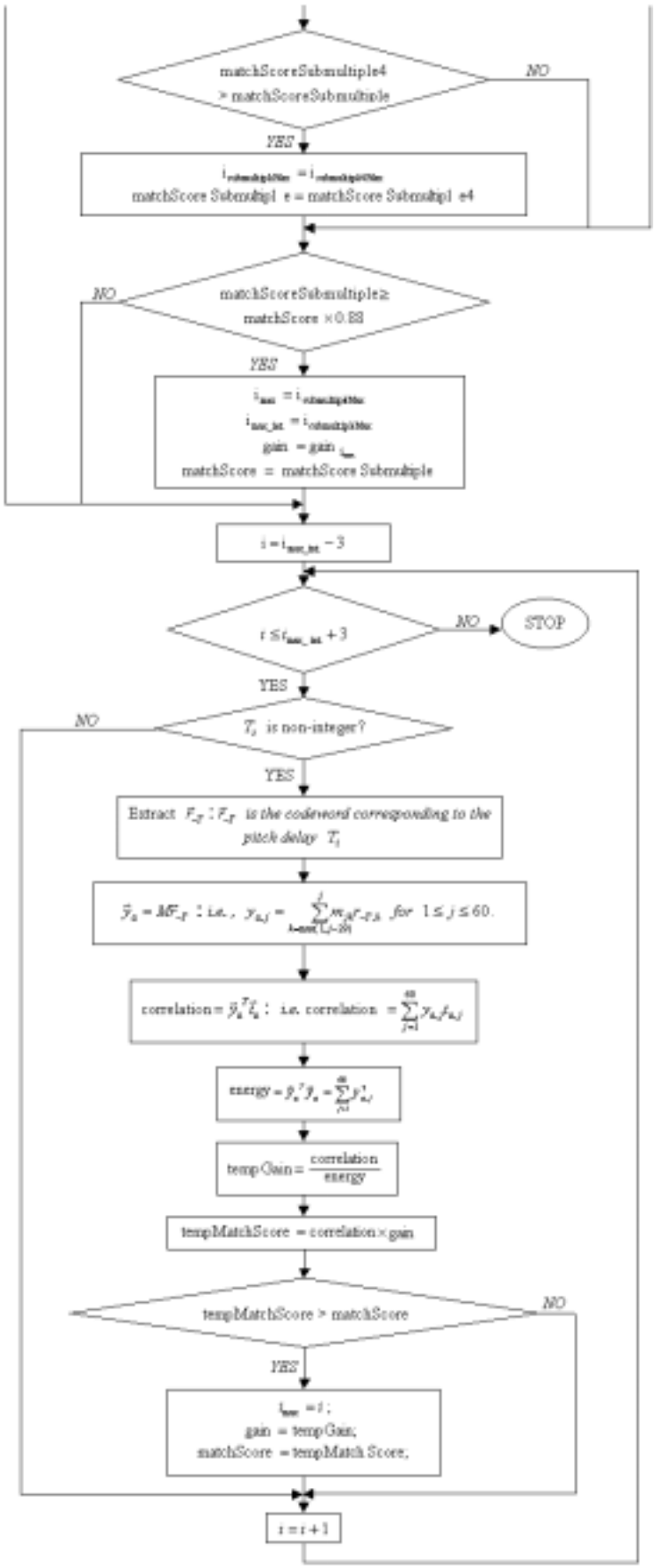


Figure 4.7: Continue from Fig. 4.6.

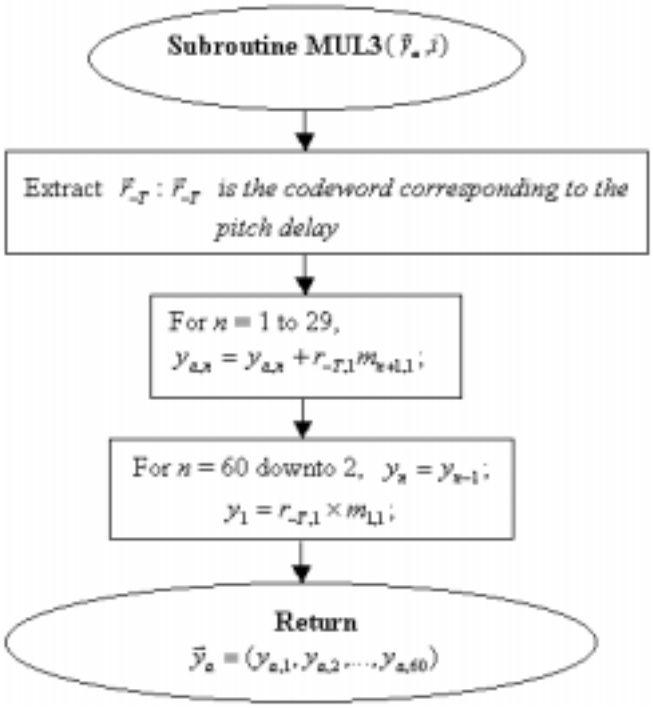


Figure 4.8: The subroutine used in the algorithm displayed in Fig. 4.5.

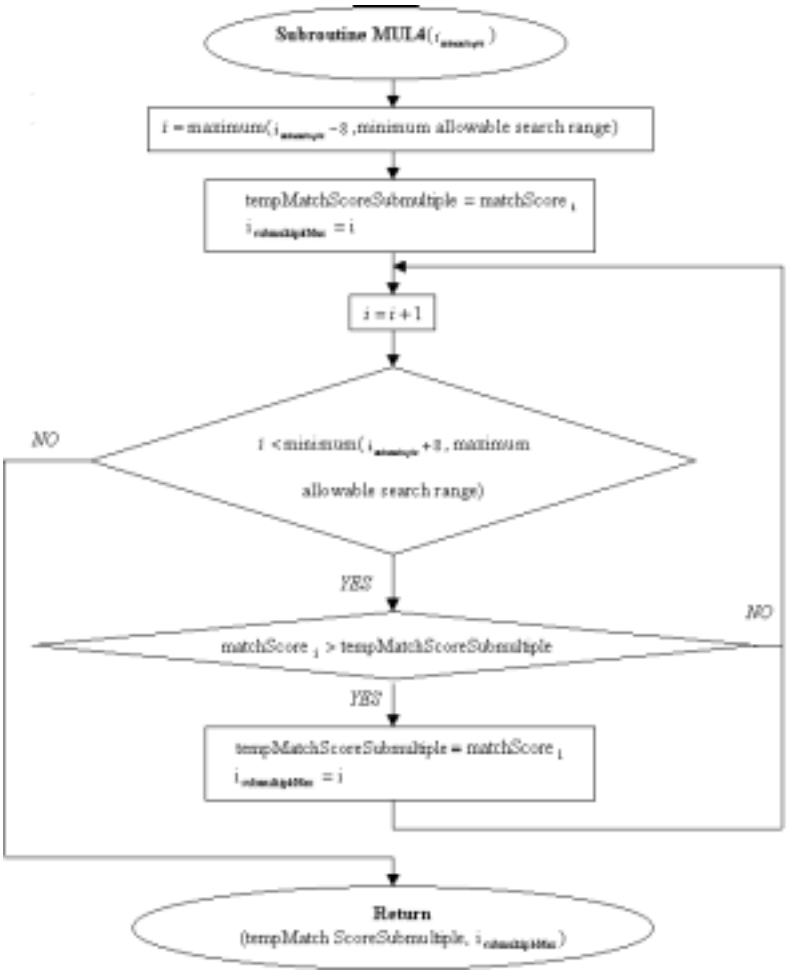


Figure 4.9: The subroutine used in the algorithm displayed in Fig. 4.6.

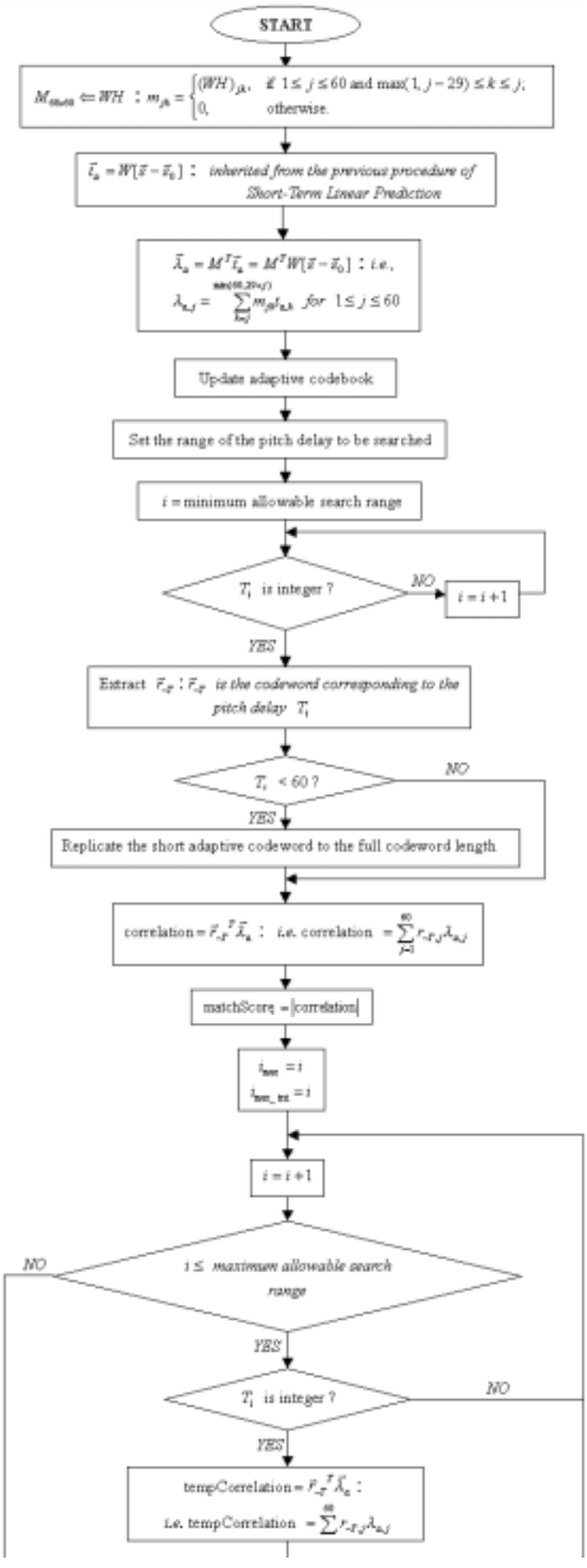


Figure 4.10: Our proposed algorithm for the adaptive codebook search.

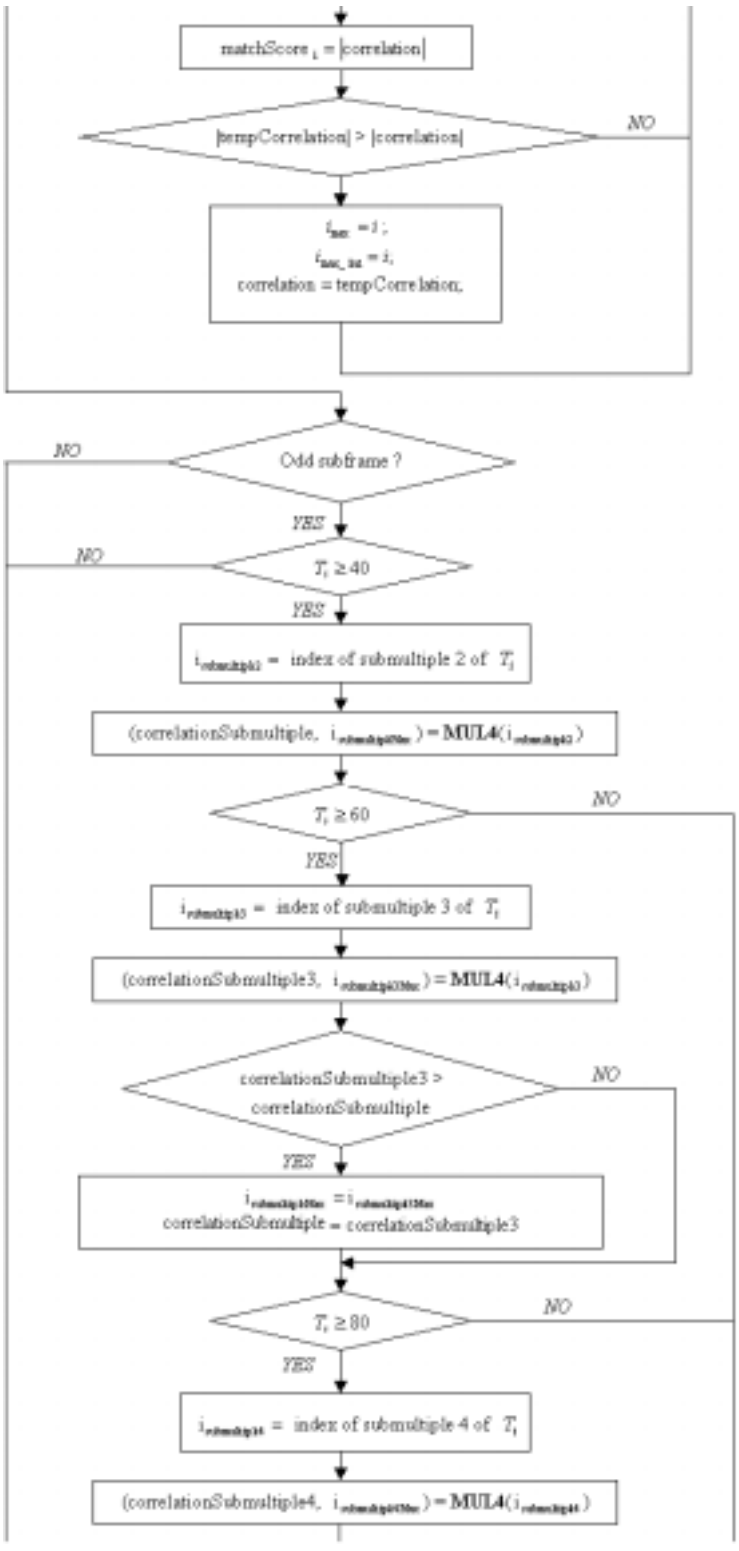


Figure 4.11: Continue from Fig. 4.10.

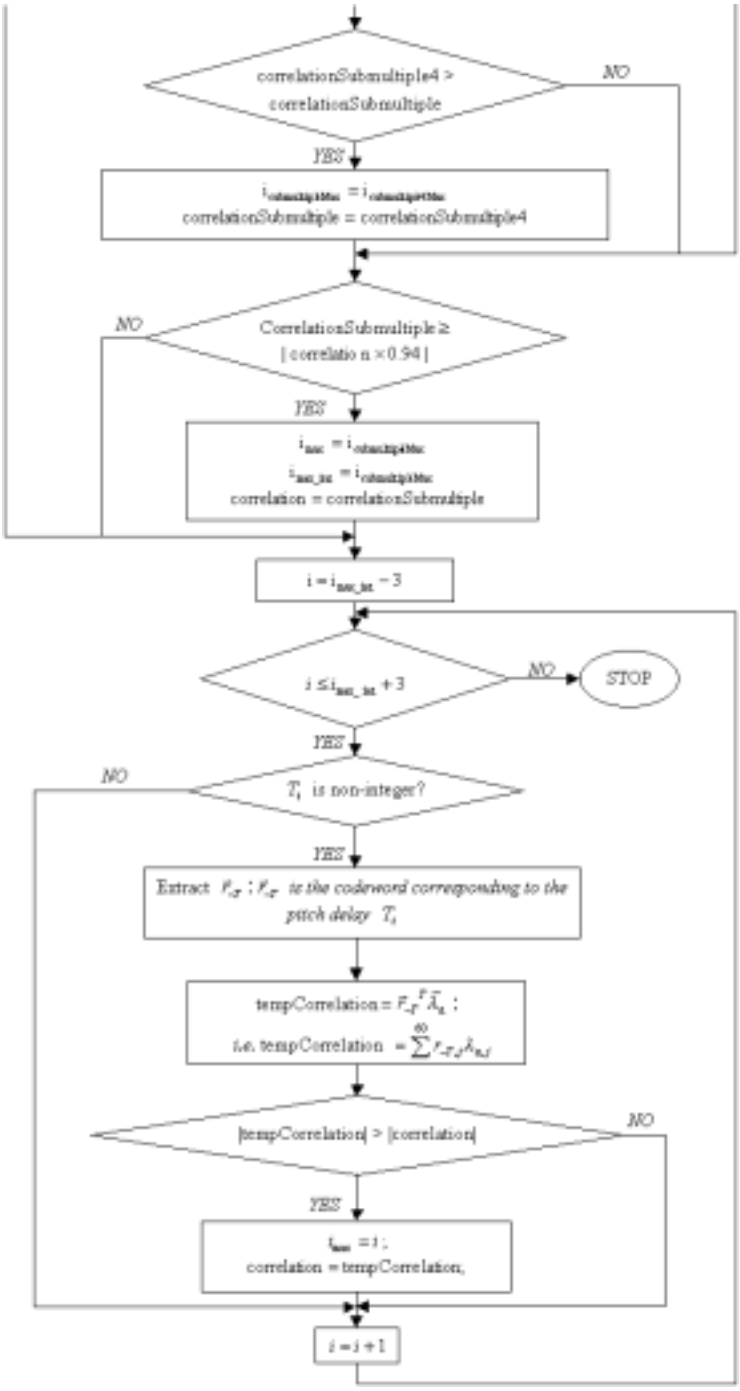


Figure 4.12: Continue from Fig. 4.11.

Chapter 5

Simulation Results

This chapter presents the theoretical complexities of the original codebook search algorithms and the proposed codebook search algorithms. Their execution time based on true measurements of implemented programs are also addressed.

5.1 Computational Complexity of Codebook Search Algorithms

The improvements in computational complexity for the stochastic and the adaptive codebook searches are respectively discussed in two separate subsections.

5.1.1 Computational Complexity of Stochastic Codebook Search Algorithms

Before introducing the computational complexities of the original and newly proposed stochastic codebook search algorithms, some notes on the algorithmic realization of the convolutional operation in Fig. 3.3 must be made.

From the mathematical expression of the energy, its calculation requires

	Mul/Div	Add/Sub
Convolution	1365	1305
Correlation	60	59
Energy	60	59
Gain	1	0
Match Score	1	0

Table 5.1: The computational complexity of the first codeword in the original stochastic codebook search algorithm.

	Mul/Div	Add/Sub
Convolution	1365	1305
Correlation (subroutine MUL2)	0	15

Table 5.2: The computational complexity of the first codeword in the proposed stochastic codebook search algorithm.

60 multiplications and 59 additions. However, since the consecutive codewords are overlapped except at the first and last two positions, the calculation of energy can be simplified to two subtractions to the previous energy, if the next two code bits are both zeros. Since 77% of the code bits are zero, the above procedure greatly reduce the number of computations required. For clarity, the procedure of examining the next two code bits is extracted from the original algorithm as a subroutine in Fig. 3.3.

The same implementation technique can certainly be applied to our new algorithm. As shown in Fig. 3.5, the calculation of correlation for codeword

	Mul/Div	Add/Sub
Convolution (subroutine MUL1)	2×511=1022	29×248=7192
Correlation	60×511=30660	59×511=30149
Energy	60×220+ 2×291=13782	59×220+ 2×291=13562
Gain	1×511	0
Match score	1×511	0

Table 5.3: The computational complexity of the remaining codewords (other than the first one) in the original stochastic codebook search algorithm.

	Mul/Div	Add/Sub
Correlation (subroutine MUL2)	0	7393

Table 5.4: The computational complexity of the remaining codewords (other than the first one) in the proposed stochastic codebook search algorithm.

i, i.e.,

$$\sum_{j=0}^{59} x_{1025-2i+j}\lambda_j,$$

can be replaced by sequential examinations on the next code bits, for which the examined result decides the necessity of adding or subtracting the respective λ_j.

By observing that for the encoding of a subframe, the subroutine in Fig. 3.3 experiences 291 zero pairs, the number of multiplications and additions required by the original algorithm can be accurately established. The results are summarized in Tabs. 5.1 and 5.3, in which the computational

	Mul/Div	Add/Sub	Relative processing time	ratio
Original algorithm	47973	52837	$47973 \times 20 + 52837 \times 1 = 1012297$	28.11
Proposed algorithm	1365	8713	$1365 \times 20 + 8713 \times 1 = 36013$	

Table 5.5: Comparisons of computational complexity between the two stochastic codebook search algorithms. The relative processing time, measured by the unit of one addition operation time, is calculated under the premise that the multiplication operation is 20 times slower than the addition operation.

complexity of the first codeword and the remaining codewords are separately listed, since the first codeword requires more computational operations than the remaining codewords.

Similarly, the computational complexity of the proposed algorithm can be derived by observing that in the subroutine of Fig. 3.5, 15 and 7393 nonzero samples are encountered respectively for the first codeword and the remaining 511 codewords. The results are summarized in Tabs. 5.2 and 5.4.

The total numbers of operations required by both algorithms are summarized in Tab. 5.5. According to [14], the clock cycles consumed by a multiplication operation are approximately 20 times larger than that required by an addition operation. This concludes that the proposed algorithm, when being implemented as an ACM driver over Windows operating system, theoretically improves the old one by a factor of 28.11 in processing time (cf. the

last column of Tab. 5.5).

5.1.2 Computational Complexity of Adaptive Codebook Search Algorithms

	Mul/Div	Add/Sub
Convolution	1365	1305
Correlation	60	59
Energy	60	59
Gain	1	0
Match Score	1	0

Table 5.6: The computational complexity of the first codeword in the original adaptive codebook search algorithm.

	Mul/Div	Add/Sub
Convolution	1365	1305
Correlation	60	59

Table 5.7: The computational complexity of the first codeword in the proposed adaptive codebook search algorithm.

As described in Section 2.1.2, not all the 256 adaptive codewords are searched when non-full search mode is selected. According to a true measurement, the number of the adaptive codewords searched is 83 in average, including 81 codewords for integer-valued pitch delays and 2 codewords for non-integer-valued pitch delay. We therefore use this number as a basis to calculate the computational complexities of the original and proposed adaptive codebook search algorithms.

	Mul/Div	Add/Sub
Convolution	$30\times80+60\times2$ $=2520$	$29\times80+59\times2$ $=2438$
Correlation	$60\times82=4920$	$59\times82=4838$
Energy	$60\times82=4920$	$59\times82=4838$
Gain	$1\times82=82$	0
Match score	$1\times82=82$	0

Table 5.8: The computational complexity of the remaining codewords (other than the first one) in the adaptive codebook search algorithm.

	Mul/Div	Add/Sub
Correlation	$60\times82=4920$	$59\times82=4838$

Table 5.9: The computational complexity of the remaining codewords (other than the first one) in the proposed adaptive codebook search algorithm.

As the consecutive integer-valued-delay codewords are overlapped except at the first and last positions, the convolution operation for codewords other than the first one only requires 30 multiplications and 29 additions. We therefore separately list the calculated complexities for the first codeword and the remaining codewords. The results are summarized in Tabs. 5.6–5.9. Unlike the ternary valued stochastic codewords, the codewords in the adaptive codebook are real-valued; hence, the implementation technique in which a multiplication can be replaced by a conditional addition can not be applied. Accordingly, the speed-up ratio of the proposed adaptive codebook search algorithm against the original one is not as large as that for the stochastic codebook search. From Tab. 5.10. the proposed algorithm only improves the

	Mul/Div	Add/Sub	Relative processing time	ratio
Original algorithm	14011	13537	$14011 \times 20 + 13537 \times 1 = 293757$	2.21
Proposed algorithm	6345	6222	$6345 \times 20 + 6222 \times 1 = 133122$	

Table 5.10: Comparisons of computational complexity between the two adaptive codebook search algorithms. The relative processing time, measured by the unit of one addition operation time, is calculated under the premise that the multiplication operation is 20 times slower than the addition operation.

old one by a ratio of 2.21.

5.2 Processing Time Based on True Measurements

We now measure the actual processing time of the implemented algorithms. The FS1016 CELP sample C code, provided by U.S. Department of Defense, versioned 3.3c, and dated 1 June 1994, is used. It is a well-implemented CELP voice coder with various enhancements.

The sample C program is compiled by Microsoft Visual C++ 6.0 with the option of optimization in speed, and is executed as a form of an ACM driver under Microsoft Windows 98SE Operating System . This ACM form is the actual form in which a codec can be practically called and executed by any applications, such as Netmeeting, in Microsoft operating system. As

shown later, the speed-up ratio of our proposed algorithm does not reach the theoretical value listed in the previous subsection, because extra system overheads are introduced in a practical execution.

Table 5.11 summarizes the average time consumed in encoding a subframe when the stochastic codebook search algorithm is executed for 100,000 times. Four voice streams are experimented. It is surprised that after the modification of the original code, the time consumed by our proposed algorithm is larger than that by the original algorithm (cf. the first and second rows of Tab. 5.11). After further examination, we noticed that in the original FS1016 sample C code, the data type of the stochastic codewords whose values are only elements of $\{-1, 0, +1\}$ is declared as *floating-point numbers*. Hence, the slow-down of our algorithm, even if it requires less computations, is caused by the extra conversion load from the floating-point type to integer type. To be specific, the subroutine in Fig. 3.5 requires 7408 (=15 + 7393) examinations of the equality of *floating-point* x to -1, 0 and $+1$, while the subroutine in Fig. 3.3 only needs 1022 (=511 $\times$ 2) such examinations. We then modified the declaration of the stochastic codewords from *float* to *integer*. Surprisingly, we observed that the speed of the original algorithm slightly decreases, rather increases, by directly changing the data type of stochastic codewords (cf. the first and third rows of Tab. 5.11). We found that this slow-down is simply because while the original algorithm benefits from a faster examination on the quantity x, it suffers from additional conversion load from integers to floating-point numbers when a multiplication of x to a floating-point number is executed. The additional amount of these conversions is 1082, including 60 times to execute $\vec{y} = M\vec{x}$ for the routine in

Fig. 3.2 and 1022 (=511 × 2) times to execute $y_1 = x \times m_{1,1}$ for the routine in Fig. 3.3. The last row of Tab. 5.11 is the time spent by our proposed algorithm based on *fixed-point*-formatted stochastic codewords. Since our algorithm requires much less multiplications, and relies more on conditional additions (conditioned on the value of the codewords), the fixed-point version markedly reduces the true executing time.

Table 5.12 summarizes the average time spent by the original and proposed adaptive codebook search algorithm for 100,000 times. It can be found that the proposed algorithm a little improves the original one.

Finally, Tab. 5.13 lists the time consumed to compress an entire segment of speech using the original FS1016 CELP and the simplified FS1016 CELP with (1) proposed stochastic codebook search algorithm, (2) proposed adaptive codebook search algorithm, and (3) both the proposed stochastic and adaptive codebook search algorithms. From these implementation, our new codec outperforms the old one by a speedup ratio of around 1.7.

	Speech of Chinese Female	Speech of Chinese Male	Speech of American Female	Speech of American Male
Original Algorithm With *floating-point* Stochastic Codebook	78.2s	77.7s	77.9s	78.5s
Proposed Algorithm With *floating-point* Stochastic Codebook	139.3s	139.6s	140.5s	140.0s
Original Algorithm With *fixed-point* Stochastic Codebook	80.1s	81.0s	80.5s	80.8s
Proposed Algorithm With *fixed-point* Stochastic Codebook	32.6s	32.5s	32.7s	32.6s

Table 5.11: Average time consumed in encoding a subframe by executing the original and proposed stochastic codebook search algorithms (with fixed- and floating-point formatted stochastic codewords) for 100,000 times. Four voice streams are experimented.

	Speech of Chinese Female	Speech of Chinese Male	Speech of American Female	Speech of American Male
Original Algorithm	40.1s	40.2s	40.1s	39.9s
Proposed Algorithm	32.5s	32.2s	32.3s	32.6s

Table 5.12: Average time consumed in encoding a subframe by executing the original and proposed adaptive codebook search algorithms for 100,000 times. Four voice streams are experimented.

	Speech of Chinese Female	Speech of Chinese Male	Speech of American Female	Speech of American Male
Total length of the speech	18.38s	47.86s	57.23s	60.00s
Original Algorithm	3.62s	8.97s	10.63s	10.92s
Proposed SCB search Algorithm	2.41s	5.96s	7.01s	7.19s
Proposed ACB search Algorithm	3.38s	8.16s	9.75s	10.13s
Both the Proposed SCB Algorithm and the Proposed ACB Algorithm	2.18s	5.18s	6.19s	6.40s

Table 5.13: Time consumed to convert a segment of analog speech to digital data by using the original FS1016 and simplified FS1016 with (1) proposed stochastic codebook (SCB) search algorithm, (2) proposed adaptive codebook (ACB) search algorithm, and (3) both the proposed stochastic and adaptive codebook search algorithms.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, the computational complexity of FS1016 is simplified by presenting new codebook search algorithms without modifying the structures of the codebooks. The resultant low-bit-rate speech codec is therefore not only compatible to the original FS1016 CELP, but suitable for real-time applications. The research is concluded by an implementation of the new codec in the form of an ACM driver, which is now in true usage in certain commercialized products.

6.2 Future Work

Since the new stochastic search algorithm only consists of addition operations except at the computation of $\vec{\lambda}$, a further speed-up of the new algorithm can be obtained by quantizing the components of $\vec{\lambda}$, and transforming the *floating-point* additions to *fixed-point* addition operations. Such a modification apparently facilitates its future implementation over a fixed-point DSP chip.

Bibliography

[1] M. E. Ahmed and M. I. Al-Suwaiyel, "Fast Method for Code Search in CELP," *IEEE Transacations on Speech and Audio Processing*, vol. 1, no. 3, pp. 315–325, July 1993.

[2] B. S. Atal and M. R. Schroeder, "Code-Excited Linear Prediction (CELP) High-Quality Speech at Very Low Bit Rates," *Int. Conf. Acou., Speech and Signal Processing*, vol. ICASSP–25, pp. 937–940, March 1985.

[3] T. P. Barnwell, K. Nayebi, and C. H. Richardson, *Speech Coding: A Computer Laboratory Textbook*, John Wiley & Sons, 1996.

[4] J. Campbell, V. Welch and T. Tremain, "The New 4800 bps Voice Coding Standard," *Proceedings of Military and Government Speech Tech*, pp. 64–70, 1989.

[5] R. V. Cox and P. Kroon, "Low-Bit-Rate Speech Coders for Multimedia Communication," *IEEE Communications Magazine*, pp. 34-41, December 1996.

[6] General Services Administration Office of Information Resources Management, "Telecommunications: Analog to Digital Conversion of Radio

Voice by 4,800 bit/second Code Excited Linear Prediction (CELP)," February 1991.

[7] G. Thomsen and Y. Jani, "Internet telephony: Goning Like Crazy," *IEEE SPECTRUM*, pp. 52–58, May 2000.

[8] Y. F. Huang, Y. Juan, S. F. Zhang and J. G. Zhang, "Implementation of ITU-T G.723.1 Dual Rate Speech Codec Based on TMS320C6201 DSP," *Proceedings of ICSP2000*, 2000.

[9] ITU-T Study Group, "Coding of Speech at 8 kbits/s Using Conjugate-structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP)," *International Telecommunication Union Telecommunication Standardization Sector*, Draft Recommendation, Version 6.5, Origin E, December 1995.

[10] P. Kabal and R. P. Ramachandran, "The Compuattion of Line Spectral 18. Frequencies Using Chebyshev Polynomials," *IEEE Transaction on ASSP*, pp. 1419–1426, vol. ASSP–34, No.6, December 1986.

[11] Y. H. Kao, *Low Complexity CELP speech Coding at 4.8kbps*, Master Thesis, University of Maryland, College Park, 1990.

[12] C. Montminy and T. Aboulnasr, "Improving the Performance of ITU-T G.729A for VoIP," *ICME'2000*, pp. 433–436, vol. 1, 2000.

[13] Office of the Manager National Communications System, "Details to Assist in Implementation of Federal Standard 1016 CELP," January 1992.

[14] R. Rector and G. Alexy, *The 8086 BOOK includes the 8088*, Berkly, California: McGraw-Hill, 1983.

[15] R. Salami, C. Laflamme, J. P. Adoul, A. Kataoka, S. Hayashi, T. Moriya, C. Lamblin, D. Massaloux, S. Proust, P. Kroon and Y. Shoham, "Design and Description of CS-ACELP: A Toll Quality 8 kb/s Speech Coder," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 2, March 1998.

[16] R. Salami, C. Laflamme, B. Bessette, and J. P. Adoul, "Description of ITU-T Recommendation G.729 Annex A: Reduced Complexity 8kbit/s CS-ACELP Codec," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (Munich-Germany), vol. 2, pp. 775–778, April 1997.

[17] S. C. Shieh, *The Research of the Influence of CELP due to Packet Loss*, Master Thesis, National Chi Nan University, June 2001.

[18] T. T. Teo, E. C. Tan and A. B. Premkumar, "Exploiting High-Performance DSP Hardware for Real-time CELP Implementation," *IEEE TENCON - Speech and Image Technologies for Computing and Telecommunication*, pp. 421–424, 1997.